

Eurotherm[®]

by Schneider Electric

Eurotherm PAC
LIN Blocks Reference
Manual

Issue 33

August 2019
HA082375U003

Contents

CHAPTER 1	INTRODUCTION	9
	What is LIN?	9
	What is a LIN Database?	9
	What is a LIN function block?	10
	What is LINtools?	10
	Instrument support of function blocks	10
	Status symbols used in block parameter tables	11
	What are Current and Legacy Instruments?	11
CHAPTER 2	BATCH FUNCTION BLOCKS	19
	SFC_CON: SFC Control Block	20
	SFC_MON: SFC Monitor Block	23
	SFC_DISP: SFC Display Block	24
	SFC_DISP_EX: SFC Display Block	26
	RECORD: Record Block	27
	DISCREP: Discrepancy Block	30
	RCP_SET: Recipe Set Block	33
	SFC_CON: Recipe Line Block	34
	BAT_CTRL: Batch Control Block	36
	BATCHCONTROL: Batch Control Block	40
CHAPTER 3	COMMUNICATION FUNCTION BLOCKS	45
	GW_CON: GateWay Configuration Block	46
	GW_TBL: GateWay Table Block	50
	GWPROFM_CON: GateWay Profibus Master Config. Block	53
	GWPROFS_CON: GateWay Profibus Slave Config. Block	58
	RAW_COM: Raw Communications Block	62
CHAPTER 4	CONDITION FUNCTION BLOCKS	67
	INVERT: Analogue Inversion Block	68
	CHAR: Characterisation Block	69
	UCHAR: Analogue Input Characterisation Block	70
	FILTER: Filter Block	71
	LEADLAG: Lead/Lag Block	73
	AN_ALARM: Analogue Alarm Block	75
	DIGALARM: Digital Alarm Block	77
	LEAD_LAG: Lead/Lag (Filter) Block	78
	RANGE: Range Block	80
	FLOWCOMP: Compensated Flow Block	81
	GASCONC: Natural Gas Concentration Data Block	84
	AGA8DATA: AGA8 Calculation Block	88
	CARB_DIFF: Carbon Diffusion Calculation Block	92
	STEEL_SPEC: Steel Composition Block	98
	ZIRCONIA: Zirconia Block	99

	TC_LIFE: Thermocouple Life Expectancy block	107
	TC_LIFE_EX: Thermocouple Life Extension Block	111
	TC_SEL: Thermocouple Selector block	113
CHAPTER 5	CONFIGURATION FUNCTION BLOCKS	115
	PROGRAM: PROGRAM Configuration Block	116
	PROGT600: T600 PROGRAM Configuration Block	116
	T600: T600 Configuration Block	117
	T940: T940(X) Configuration Block	123
	T225: T225(X) Configuration Block	127
	TACTICIAN: Tactician Configuration Block	130
	Eycon-10: Eycon 10 Configuration Block	136
	Eycon-20: Eycon 20 Configuration Block	136
CHAPTER 6	CONTROL FUNCTION BLOCKS	140
	PID: PID Control Block	141
	ANMS: Analogue Manual Station Block	148
	DGMS: Digital Manual Station Block	150
	SIM: Simulation Block	152
	AN_CONN: Analogue Connections Block	154
	DG_CONN: Digital Connections Block	156
	TP_CONN: Tepid Data Connection Block	158
	SETPOINT: Setpoint Block	159
	3_TERM: Incremental PID Block	163
	MAN_STAT: Manual Station Block	168
	MODE: Mode Block	172
	PID_CONN: PID Connection Block	176
	PID_LINK: PID Linking Block	178
	AN_DATA: Analogue Data Block	180
	LOOP_PID: Loop Proportional, Integral, Derivative Block	182
	TUNE_SET: PID Tuning Set Block	198
CHAPTER 7	CONVERT FUNCTION BLOCKS	202
	ENUMENUM: Enumerated To Enumerated Converter Block	203
	UINTENUM: Integer To Enumerated Converter Block	204
	ENUMUINT: Enumerated To Integer Converter Block	205
	REALTIME: Real and Time Converter Block	206
CHAPTER 8	DCM FUNCTION BLOCKS	208
	DCM Function Block Sub-categories	208
CHAPTER 9	DIAGNOSTIC FUNCTION BLOCKS	209
	AGA8DIAG: AGA8 Diagnostic Block	210
	ALH_DIAG: Alarm History Diagnostic Block	212
	ALINDIAG: ALIN MAC/LLC Diagnostic Block	213
	AMC_DIAG: Application Master Comms Diagnostic Block	214
	BCS_DIAG: Routing Broadcast Diagnostic Block	215
	CON_DIAG: Connection Diagnostic Block	216
	CON_ENT: Connection Entry Block	217
	CON_TBL: Connection Table Block	218
	DB_DIAG: Database Diagnostic Block	219
	DDR_DIAG: Data Recording Facility Diagnostic Block	220
	EDB_DIAG: External Database Diagnostic Block	221

EDB_TBL: External Database Table Block	223
EIO_DIAG: Eurotherm I/O Diagnostic Block	224
ELINDIAG: Ethernet LIN Diagnostic Block	227
EMAPDIAG: ELIN Mapping Diagnostic Block	229
ETH_RT_LIM: Ethernet Rate Limit Diagnostic Block	230
FDDADIAG: FTP Distributed Data Archiving Diagnostic Block	232
FSM_DIAG: File System Management Diagnostic Block	234
FTQ_DIAG: PRMT Queues Diagnostic Block	236
FWD_DIAG: Forwarding Statistics Block	237
FWD_LOG: Forwarding Log Block	238
ICM_DIAG: Interprocessor Comms Mechanism Statistics Block	239
IDENTITY: Instrument Identification/Status Diagnostic Block	240
ISB_DEXT: Internal Serial Bus Diagnostic Extension Block	245
ISB_DIAG: Internal Serial Bus Performance Block	246
LIN_DEXT: LIN High-level Diagnostic Extension Block	247
LINMAPD: LIN Mapping Diagnostic Block	248
LLC_DIAG: Logical Link Control (LLC) Diagnostic Block	251
NATCDIAG: Network Audit Trail Consumer Diagnostic Block	253
NATPDIAG: Network Audit Trail Provider Diagnostic Block	254
NET_DIAG: Network Diagnostic Block	255
NETHOST: netHOST Diagnostic Block	256
NODE_MAP: LIN Node Protocol Block	258
OPT_DIAG: Options Diagnostic Block	259
PBUSDIAG: Profibus Diagnostic Block	261
PMC_DIAG: Profibus Line Diagnostic Block	265
PNL_DIAG: Front-Panel Diagnostic Block	267
PRPDIAG: Port Resolution Protocol Diagnostic Block	268
PS_TASK: T940X Task Diagnostic Block	269
PS_TUNE: T940 Performance Block	271
RARCDIAG: Data Record Archive Diagnostic Block	273
RED_CTRL: Redundancy Control Block	276
RMEMDIAG: Data Record Memory Diagnostic Block	279
ROUTETBL: Routing Table Block	282
RSRCDIAG: Resource Diagnostic Block	283
RTB_DIAG: Routing Table Diagnostic Block	285
SD_DIAG: SD Card Diagnostic Block	287
SFC_DIAG: Sequence Diagnostic Block	289
SUM_DIAG: Summary Diagnostic Block	290
T600TUNE: T600 Performance Block	293
TACTTUNE: Tactician Task Summary Block	295
TOD_DIAG: Time-Of-Day Diagnostic Block	297
USERTASK: User Task Diagnostic Block	299
XEC_DIAG: Task Diagnostic Block	301
CHAPTER 10 HISTORIAN FUNCTION BLOCKS	302
CHAPTER 11 I/O FUNCTION BLOCKS	303
AN_IP: Analogue Input Block	304
AN_OUT: Analogue Output Block	310
DG_IN: 8-channel Digital Input Block	313
DG_OUT: 8-channel Digital Output Block	316
AI_CALIB: Analogue Input Calibration Block	325
AO_CALIB: Analogue Output Calibration Block	329
MOD_UIO: Module Input/Output Block	332
AI_UIO: Analogue Input Block	334

	AO_UIO: Analogue Output Block	344
	CALIB_UIO: Analogue Input/Output Calibration Block	348
	DI_UIO: Digital Input Block	352
	DO_UIO: Digital Output Block	355
	TPO_UIO: Time Proportioning Output Block	358
	FI_UIO: Frequency Input Block	361
	MOD_DI_UIO: Multi-Channel Digital Input Module Block	369
	MOD_DO_UIO: Multi-Channel Digital Output Module Block	373
	VP_UIO: Valve Positioner Block	376
CHAPTER 12	LOGIC FUNCTION BLOCKS	379
	PULSE: Pulse Block	380
	LOGIC FUNCTION BLOCKS	382
	AND4: AND Block	382
	OR4: OR block	382
	XOR4: Exclusive-OR Block	382
	NOT: NOT Block	383
	LATCH: Latch Block	384
	COUNT: Count Block	386
	COMPARE: Compare Block	389
CHAPTER 13	MATHS FUNCTION BLOCKS	392
	ADD2: Add Block	393
	SUB2: Subtract Block	393
	MUL2: Multiply Block	393
	DIV2: Divide Block	393
	EXPR: Expression Block	394
	ACTION: Action Block	397
	DIGACT: Digital Action Block	399
	ACT_2A2W3T: Action Block with Gated Downtimers	401
CHAPTER 14	OPERATOR FUNCTION BLOCKS	403
	PNL_CMD: Panel Command Block	404
	PNL_DICT: Panel Dictionary Block	408
	PNL_MSG: Panel Message Block	409
	PNL_DLG: Panel Dialogue Block	413
	PNL_ACC: Panel Access Block	416
	READER: Reader Block	418
	EVENT: Event Block	420
CHAPTER 15	ORGANISE FUNCTION BLOCKS	422
	AREA: Area Block	423
	GROUP: Group Block	425
	LOGDEV: Logging Device Block	427
	LGROUP: Log Group Block	429
	LOGGRPEX: Log Group Extension Block	431
	LPTDEV: Printer Device Block	432
	PGROUP: Printer Group Block	434
CHAPTER 16	PROGRAMMER FUNCTION BLOCKS	435
	SPP_CTRL: Setpoint Programmer Control Block	436

	SPP_DIG: Setpoint Programmer Digital Block	438
	SPP_RAMP: Setpoint Programmer Local Ramp Block	440
	SPP_EXT: Setpoint Programmer Extension Block	443
	PROGCTRL: Programmer Control Block	445
	PROGCHAN: Programmer Channel Block	453
	SEGMENT: Programmer Segment Block	462
CHAPTER 17	RECORDER FUNCTION BLOCKS	467
	DR_ANCHP: Data Recording Analogue Channel Point Block	468
	DR_DGCHP: Data Recording Digital Channel Point Block	471
	DR_ALARM: Alarm Point Block	473
	DR_REPRT: Report Point Block	474
	RGROUP: Data Recording Group Block	475
	HISTDATA: HISTORICAL DATA COLLECTION BLOCK	478
CHAPTER 18	S6000 FUNCTION BLOCKS	479
CHAPTER 19	SELECTOR FUNCTION BLOCKS	480
	SELECT: Selector Block	481
	SWITCH: Switch Block	482
	ALC: Alarm Collection Block	483
	2OF3VOTE: Best-average Block	484
	TAG: TAG Block	486
CHAPTER 20	TAN FUNCTION BLOCKS	487
CHAPTER 21	TIMING FUNCTION BLOCKS	488
	SEQ: Sequence Block	489
	SEQE: Sequence Extension Block	494
	TOTAL: Totalisation Block	495
	DTIME: Dearthime Block	497
	TIMER: Timer Block	499
	TIMEDATE: Time/Date Event Block	501
	TPO: Time-proportioning Output Block	504
	DELAY: Delay Block	506
	RATE_ALM: Rate Alarm Block	508
	RATE_LMT: Rate Limit Block	511
	TOT_CON: Totalisation Connections Block	514
	TOTAL2: Totalisation Block	515
APPENDIX A	CONTROL LOOP OPERATING MODES	518
	The Active Control Mode	519
	PID Block 3-term Control Algorithm	520
	Integral Balance & Integral Desaturation	522
APPENDIX B	THE LIN APPLICATION LAYER	528
	1 Configuring A Cached Block	529
	2 Block-caching At Runtime	530
	3 Caching A Second Block	531

4 Caching Another Block In The Opposite Direction532
5 Caching Blocks Between Three Nodes533

APPENDIX C OBSOLETE PRODUCT SUPPORT534

Instrument support of function blocks534

APPENDIX D COMMON FIELD PARAMETERS543

Block Specification Menu544
Alarms.545
Filepath parameter547

Index548

CHAPTER 1 INTRODUCTION

This manual describes the current core LIN function blocks that are available for developing applications and control strategies in instruments that can be connected to the Local Instrument Network (LIN). The purpose and workings of each LIN function block are explained, specification menu parameters are defined and inputs and outputs indicated, giving detailed information that is required to configure the LIN function blocks.

There are legacy instruments that support LIN function blocks that are no longer used on the current products and consequently this issue of the LIN Blocks Reference Manual does not cover those blocks. Appendix C provides further details on legacy instrument support but full details are provided in the LIN Blocks Reference Manual HA028375U003 issue 15 (Vintage).

The LIN function blocks are grouped into categories, each category occupying a complete chapter of the manual. Refer to the *Contents* list or the *Index* to locate an individual LIN function block. *Table 2* lists the LIN function block categories and primary function of each LIN function block.

WHAT IS LIN?

Local Instrument Network or LIN provides a communications network which allows peer-to-peer communications and file transfer between instruments. For current instruments, it is supported via Ethernet connectivity, but for legacy instruments, it is also supported via coax, Arcnet and serial communications connections.

Communications between LIN and third party instruments is supported by communication function blocks (e.g. Modbus, Profibus, etc) and a general purpose Raw Communications block is provided for more obscure protocols, e.g. Weigh Scales, Bar Code readers, etc.

It is also worth noting that the term 'LIN' is used in a general manner to describe or refer to the EurothermSuite DCS system or elements and/or instruments thereof.

WHAT IS A LIN DATABASE?

A LIN Database groups data into blocks of related data to form a control strategy for a particular application. A function block can represent an input, an output, a controller, and so on. The LIN configuration tool (LINtools Engineering Studio) and display packages (i.e. User Screen Editor) recognise different types of function block, and handle them appropriately.

A LIN Database (.dbf), also known as a 'strategy', is a database that runs in a LIN instrument providing control, data and rules. It is loaded by the LIN Instrument at runtime (typically on power up) and provides the instrument software the ability to control and monitor signals from sensors in the plant/system, (e.g. an industrial plant), and then output the signals back to actuators.

The cycle of signal input, signal processing, and signal output to the entity is repeated continuously while the database is run in the instrument using task priorities 1-4.

More than one LIN instrument can be involved in controlling a single entity, but only one LIN Database can run in a single LIN instrument at any one time.

A LIN Database can be configured as a standard single layer LIN Database or blended multi-layer LIN Database, operating in conjunction with one or more LIN Sequences (SFCs) running in the LIN instrument. It can also make use of LIN Actions (Structured Text ST/Ladder Logic) stored in action files in the LIN instrument.

In LINtools, a LIN Database is represented and configured graphically as an arrangement of inter connected LIN function blocks, where source to destination links, are represented as wires between blocks. LIN functions blocks are picked from a library palette and placed as appropriate in the workspace area of the LINtools Engineering Studio.

Data is passed over the network, from node to node as required by 'caching' function blocks as appropriate. This is covered in detail in *Appendix B*.

WHAT IS A LIN FUNCTION BLOCK?

LINtools uses a block-structured approach (similar to objects) to configure a control strategy, where a library of ready-made function blocks are available to perform the processing as required. Any strategy can use a combination of LIN blocks, Control Module blocks and Application blocks to create a strategy suitable for the instrument application.

A function block is an instance of a reusable module of program code, called a template, dedicated to a particular type of processing operation - e.g. the ADD2 template adds two numbers. In general, function blocks take in analogue and/or digital signals via their inputs, process them in a variety of ways, and then pass the results on via their outputs. The function blocks are then 'soft-wired' together so that the signals can flow between them to execute the required control strategy.

WHAT IS LINTOOLS?

LINtools is a powerful multi-purpose Windows-based software package for use both off- and on-line with key features as listed below:

- **Offline.** LINtools creates and modifies a Local Instrument Network (LIN) based process strategy, sequences, and actions for a range of target instruments. It is also used to configure instrument properties, e.g. time-zone, time sync, protocol name, Alarm Suppression, etc.
- **Online.** For monitoring and interacting with control and sequence strategies running in remote instruments across the ELIN or ALIN providing a versatile commissioning and strategy debugging toolkit.
- **Online Reconfiguration.** Reconfigure an operational instrument via an Ethernet network.
- **I/O Configuration.** Creation and modification of the block-structured I/O subsystem.
- **Comms Configuration.** Configuration of the Data Exchange requirements to third party devices, e.g. Modbus, Profibus etc.
- **Data Recording Configuration.** For recording values from defined groups of parameters, and configure the appearance of each defined parameter when shown in the 'Review' data reporting tool. Additional configuration of the Instrument Properties provides an archiving strategy for the recorded values.
- **Setpoint Programming.** For creation of the blocks required as an interface to the Programmer Editor. The blocks are created with a default configuration using a Wizard, but are used to display the configuration of the Programmer Editor when online.
- **Intellectual Property Protection.** For encryption of specific application file types to prevent the loss of customer Intellectual Property.

LINtools uses and outputs fully compatible files that can be saved locally to hard disk and transferred to and from target instruments at high speed over the LIN via ELIN (Ethernet), or for legacy instruments.

INSTRUMENT SUPPORT OF FUNCTION BLOCKS

Note that not all the function blocks described in this manual can be run in all LIN instruments. *Table 2* also indicates the degree of support currently offered for each LIN function block by a range of instruments.

Please refer to the appropriate document accompanying a particular instrument for details of any individual LIN function block parameters that are not supported, or only partially supported, or that have been special functions when used in that instrument.

STATUS SYMBOLS USED IN BLOCK PARAMETER TABLES

Block parameter status symbols, below, explains the symbols that appear in the block field parameter tables throughout this manual, to indicate parameter 'status'. The symbols are also located at the end of this manual, together with a binary/hexadecimal/decimal conversion chart.

Symbol	Meaning
	Connectable as an output from the block to the control strategy
	Connectable as an input to the block from the control strategy.
	Parameter becomes effectively read-only if connected.
	Input and output connectable
	Read-only. Exceptions or special conditions may be noted
	Write-only (i.e. self-resets to FALSE after being made TRUE)
	Available for 'partial' access by default* (via T640 series instruments 'INS' button)
	Alarm field(s)

Table 1 Block parameter status symbols

**Partial access can be customised via LINtools.*

WHAT ARE CURRENT AND LEGACY INSTRUMENTS?

In many instances this manual refers to current and legacy instruments. Therefore in the simplest terms, this issue of the LIN Blocks Reference Manual is consistent and maintained for the following current instruments:

Eurotherm PAC T2750

Eurotherm PAC T2550

Eycon™ 10/20 Visual Supervisor

T225(X) ELIN/ALIN Bridge Unit

All other products are considered to be legacy instruments. However the following two instruments fall into a legacy category in terms of maturity but are still currently used:

T640 Multi-loop Controller

T820 HMI

T940X Process Supervisor

The information relating to these instruments has been carried over into this manual and is consistent with the contents of the legacy instruments issue of this manual, i.e.

LIN Blocks Reference Manual HA028375U003 issue 15 (Vintage)

Category	Block	Function	T640	T940(X)	T225(X)	Eycon-10/20	T2550/T820	T2750
BATCH	BAT_CTRL	Loading & control of a batch		✓		✓		
	DISCREP	Transmitted/received digital signal-matching to diagnose plant faults	2/1✓	✓		✓	✓	✓
	RCP_LINE	Control downloading of recipes from a .UYR file in associated RCP_SET block				✓		
	RCP_SET	Control the recipe set (i.e. file to be used) for a number of recipe lines				✓		
	RECORD	Storage/retrieval of analogue/digital values for runtime use	2/1✓	✓		✓	✓	✓
	SFC_CON	Sequence (SFC) control, selection, and running	2/1S	✓		✓	✓	✓
	SFC_DISP	Display/monitoring/control of remotely running sequence (SFC)	2/1S	✓		✓	✓	✓
	SFC_MON	Sequence (SFC) runtime monitoring	2/1S	✓		✓	✓	✓
COMM	GW_CON	Control the Communications subsystem				✓	✓	✓
	GW_TBL	Show the Modbus configuration diagnostic table information				✓	✓	✓
	GWProfM_CON	Control the Profibus Master Communications subsystem				2/0✓		2/0✓
	GWProfS_CON	Control the Profibus Slave Communications subsystem					3/0✓	
	RAW_COM	Raw Communications				5/0✓	7/0✓	✓
CONDITN	AGA8DATA	American Gas Association Report #8 calculations	4/1A	✓		✓	✓	✓
	AN_ALARM	Alarm, with absolute/Deviation/Rate alarms	✓	✓		✓	✓	✓
	CHAR	16-point analogue characteriser	✓	✓		✓	✓	✓
	DIGALARM	Digital alarm	✓	✓		✓	✓	✓
	FILTER	First-order filter	2/1✓	✓		✓	✓	✓
	FLOWCOMP	Computes flow-rate, corrected for pressure, temperature, and density	✓	✓		✓	✓	✓
	GASCONC	Natural gas concentration data storage and validation	4/1A			✓	✓	✓
	INVERT	Inverts signal about HR, LR limits	✓	✓		✓	✓	✓
	LEADLAG	Lead-lag		✓		✓	✓	✓
	LEAD_LAG	Lead-lag	✓	✓		✓	✓	✓
	RANGE	Re-ranges an analogue input	✓	✓		✓	✓	✓
	UCHAR	16-point characteriser for analogue input blocks	✓	✓		✓	✓	✓
	CARB_DIFF	Computes data for Carbon Diffusion curve				2/0✓		
	STEEL_SPEC	Material specification for Carbon Diffusion process				2/0✓		
	ZIRCONIA	Receives and displays values from a Probe for Carbon Diffusion process					3/0✓	✓
	TC_LIFE	Life expectancy diagnosis of load thermocouples for furnace applications				3/0✓	5/0✓	✓
TC_LIFE_EX	Thermocouple life extension block for custom life data						3/0✓	
TC_SEL	Multiple load thermocouples management for furnace applications				3/0✓	5/0✓	✓	
CONFIG	PROGT600	System block	✓					
	T600	System block	✓					
	T940	System block		✓				
	T225	System block			✓			
	Eycon-10/20	System block				✓		
	Tactician	System block					✓	✓
CONTROL...	3_TERM	Incremental form of the PID block	✓	✓		✓	✓	✓

Category	Block	Function	T640	T940(X)	T225(X)	Eycon-10/20	T2550/T820	T2750
...CONTROL	ANMS	Analogue manual station	✓	✓		✓	✓	✓
	AN_CONN	Analogue connections	✓	✓		✓	✓	✓
	AN_DATA	Collection of analogue values for Carbon Diffusion process				2/0✓	*	✓
	DGMS	Digital manual stations	✓	✓		✓	✓	✓
	DG_CONN	Digital connections	✓	✓		✓	✓	✓
	MAN_STAT	Manual station, with connections to front panel displays	✓	✓		✓	✓	✓
	MODE	Control mode selection, with pushbutton masking	✓	✓		✓	✓	✓
	PID	PID control function	2/1✓	✓		✓	✓	✓
	PID_CONN	'Faceplate' for SETPOINT/3_TERM/MAN_STAT/MODE combination	✓					
	PID_LINK	'Faceplate' for SETPOINT/3_TERM/MAN_STAT/MODE combination		✓		✓	✓	✓
	SETPOINT	Generates a setpoint, with bias, limits, and alarms	✓	✓		✓	✓	✓
	SIM	Simulates two first order lags or capacity, with noise	2/1✓	✓		✓	✓	✓
	TP_CONN	Specifies up to 9 fields as EEPROM 'tepid data' at power-down	2/1✓					
	LOOP_PID	Controls a process to an operating point, Setpoint						3/0✓
TUNE_SET	Set of PID values to maximise performance from the PID control						3/0✓	✓
CONVERT	ENUMENUM	Convert an enumerated value from one enumeration to another				✓	*	*
	UINTENUM	Convert an integer into an enumeration				✓	*	*
	ENUMUINT	Convert an enumeration into an integer				✓	*	*
	REALTIME	Convert Real and Time values				✓	3/0✓	✓
DCM	Refer to the LIN Block Reference Manual HA082375U003, iss 15 (Vintage)							
DIAG...	AGA8DIAG	AGA8 calculation diagnostics	4/1A	✓		✓	✓	✓
	ALH_DIAG	T800/T2900/T940 alarm history diagnostics		2/2✓	✓	✓	✓	✓
	ALINDIAG	Arcnet local instrument network (ALIN) diagnostics	2/1✓	✓	✓	✓		
	AMC_DIAG	Application master comms diagnostics		✓				
	BCS_DIAG	Routing broadcast diagnostics			✓			
	CON_DIAG	Connection diagnostics			✓			
	CON_ENT	Connection entry			✓			
	CON_TBL	Connection table			✓			
	DB_DIAG	Database resource information	✓	✓	✓	✓	✓	✓
	DDR_DIAG	T800/T2900 data recording facility diagnostics				✓		
	EDB_DIAG	External remote database information	2/1✓	✓	✓	✓	✓	✓
	EDB_TBL	External database diagnostics	2/1✓	✓	✓	✓	✓	✓
	ELINDIAG	Ethernet LIN diagnostic block		✓	✓	✓	✓	✓
	EMAPDIAG	Ethernet mapping diagnostics block				✓	✓	✓
	EIO_DIAG	Eurotherm I/O diagnostic block				✓	✓	✓
	ETH_RT_LIM	Ethernet rate limit protection				✓	✓	✓
	FDDADIAG	FTP Distributed Data Archiving diagnostic block				✓		
	FSM_DIAG	File System Management diagnostics					✓	✓

Category	Block	Function	T640	T940(X)	T225(X)	Eycon-10/20	T2550/T820	T2750
...DIAG	FTQ_DIAG	PRMT queues statistics and status		✓			✓	✓
	FWD_DIAG	Forwarding statistics			✓			
	FWD_LOG	Forwarding log			✓			
	ICM_DIAG	ICM diagnostics		✓			✓	✓
	IDENTITY	Instrument Identification		✓	✓	✓	✓	✓
	ISB_DEXT	ISB (internal serial bus) high-level performance statistics	2/1✓					
	ISB_DIAG	ISB (internal serial bus) diagnostics	2/1✓					
	ISE_DIAG	Industrial Strategy Engine diagnostics block				✓		
	ISE_TUNE	Industrial Strategy Engine tune block				✓		
	LIN_DEXT	LIN high level performance statistics	2/1✓	✓		✓	✓	✓
	LINMAPD	Local instrument network (LIN) mapping diagnostics		4/1✓	✓		✓	✓
	LLC_DIAG	Local Link Control diagnostics		✓	✓	✓	✓	✓
	MDBDIAG	Modbus diagnostic block				✓		
	NATCDIAG	Network Audit Trail Consumer diagnostic block				✓		
	NATPDIAG	Network Audit Trail Provider diagnostic block				✓		
	NET_DIAG	Network diagnostics		✓	✓			
	NETHOST	NetHOST diagnostics						2/0✓
	NODE_MAP	LIN node protocol			✓			
	OPT_DIAG	Options/Licences Control diagnostics		✓			✓	✓
	PBUSDIAG	Profibus diagnostic					✓	
	PMC_DIAG	Profibus line diagnostics		1/4✓				
	PNL_DIAG	T800/T2900 front panel diagnostics				✓	✓	
	PRPDIAG	Port resolution protocol diagnostics			✓	✓	✓	✓
	PS_TASK	T940 task diagnostics		✓		✓		
	PS_TUNE	T940 performance		1/3✓				
	RARCDIAG	Data Record archive diagnostics						4.0✓
	RED_CTRL	Redundancy control			✓			✓
	RMEMDIAG	Data Record memory diagnostics						4.0✓
	ROUTETBL	Routing table	2/1✓	✓	✓	✓	✓	✓
	RSRCDIAG	Resource & memory diagnostics	4/1✓		✓	✓	✓	✓
	RTB_DIAG	Routing table diagnostics	2/1✓	✓	✓	✓	✓	✓
	SCKDIAG	Socket diagnostic block				✓		
	SD_DIAG	SD Card diagnostic block						4/0✓
	SFC_DIAG	Sequence diagnostics and statistics (resources in use and available)	4/1✓	✓		✓	✓	✓
SUM_DIAG	Summary diagnostics		✓	✓	✓	✓	✓	
T600TUNE	Performance statistics	✓						
TACTTUNE	Tactician Tack Summary					✓	✓	
TOD_DIAG	Time-of-day diagnostics & control	✓	✓	✓	✓	✓	✓	

Category	Block	Function	T640	T940(X)	T225(X)	Eycon-10/20	T2550/T820	T2750
	USERTASK	User Task diagnostics		4/1✓			✓	✓
	XEC_DIAG	Task diagnostics	2/1✓			✓		4/0✓

Category	Block	Function	T640	T940(X)	T225(X)	Eycon-10/20	T2550/T820	T2750
HIST	Refer to the LIN Block Reference Manual HA082375U003, iss 15 (Vintage)							
I/O	AI_CALIB	Guides operator through analogue input calibration	✓					
	AN_IP	Analogue input channels	✓					
	AN_OUT	Analogue output channels	✓					
	AO_CALIB	Guides operator through analogue output calibration	✓					
	DGPULS_4	Pulse outputs (1-shot, pulse train, dual pulse, time-proportioned output)	✓					
	DG_IN	Digital input channels	✓					
	DG_OUT	Digital output channels	✓					
	MOD_UIO	Tactician module input/output block					✓	✓
	AI_UIO	Tactician analogue input block					✓	✓
	AO_UIO	Tactician analogue output block					✓	✓
	CALIB_UIO	Tactician calibration block					✓	✓
	DI_UIO	Tactician digital input block					✓	✓
	DO_UIO	Tactician digital output block					✓	✓
	TPO_UIO	Tactician time proportioning output block					✓	✓
	FI_UIO	Tactician frequency input block					✓	✓
	MOD_DO_UIO	Tactician multiple channel digital output block					✓	✓
	MOD_DI_UIO	Tactician multiple channel digital input block					✓	✓
	VP_UIO	Tactician valve position output block					3/0✓	✓
LOGIC	AND4	4-input AND boolean function	✓	✓		✓	✓	✓
	COMPARE	Indicates greater/less than/equal of 2 inputs	✓	✓		✓	✓	✓
	COUNT	UP/DOWN pulse counter with START/END count target	✓	✓		✓	✓	✓
	LATCH	D-type flip-Flop function	✓	✓		✓	✓	✓
	NOT	NOT boolean function	✓	✓		✓	✓	✓
	OR4	4-input OR boolean function	✓	✓		✓	✓	✓
	PULSE	Pulse output, (monostable) function	✓	✓		✓	✓	✓
	XOR4	4-input exclusive-OR boolean function	✓	✓		✓	✓	✓
MATHS	ACTION	Action control, with use of stored variables and elapsed time	2/1✓	✓		✓	✓	✓
	ADD2	Adds 2 inputs	✓	✓		✓	✓	✓
	DIGACT	Action control, with use of stored digital variables and elapsed time	✓	✓		✓	✓	✓
	DIV2	Divides 2 inputs	✓	✓		✓	✓	✓
	EXPR	Free-format maths expression with up to 4 inputs	✓	✓		✓	✓	✓
	MUL2	Multiplies 2 inputs	✓	✓		✓	✓	✓
	SUB2	Subtracts 2 inputs	✓	✓		✓	✓	✓
	ACT_2A2W3T	Action control, with use of stored variables and 3 downtimers				3.0✓	✓	✓

Category	Block	Function	T640	T940(X)	T225(X)	Eycon-10/20	T2550/T820	T2750	
OPERATOR	PNL_ACC	Panel access block - enable & monitor user logons (Eycon™ 10/20 Visual Supervisor)				✓			
	PNL_CMD	Panel command block - take command of the panel (Eycon™ 10/20 Visual Supervisor)				✓			
	PNL_DLG	Panel dialogue block - create dialogue box (Eycon™ 10/20 Visual Supervisor)				✓			
	PNL_MSG	Panel message block - create message (Eycon™ 10/20 Visual Supervisor)				✓			
	READER	Reader block - enable use of barcode readers, etc. (Eycon™ 10/20 Visual Supervisor)				3.0✓			
	EVENT	Event block - enable use of an action if an event occurs (Eycon™ 10/20 Visual Supervisor)				✓			
ORGANISE	AREA	Associate GROUP blocks with an area				✓	✓	✓	
	GROUP	Associate Eycon™ 10/20 Visual Supervisor display and recorder channels into a group				✓	✓	✓	
	LGROUP	Collect data from point blocks for archiving				✓			
	LOGDEV	Specify and control access to an archive medium				✓			
	LOGRPEX	Extend number of points logged by LGROUP block				✓			
	LPTDEV	Specify and control access to a printing device				✓			
	PGROUP	Collect data from point blocks for printing				✓			
PROGRAMMER	SPP_CTRL	Monitor, schedule, & control the setpoint programmer (Eycon™ 10/20 Visual Supervisor)				✓			
	SPP_DIG	Wire out digital setpoints from the setpoint programmer (Eycon™ 10/20 Visual Supervisor)				✓			
	SPP_EXT	Provide additional control over running setpoint program (Eycon™ 10/20 Visual Supervisor)				✓			
	SPP_RAMP	Allow local ramping of setpoints in the setpoint programmer (Eycon™ 10/20 Visual Supervisor)				✓	✓	✓	
	PROGCTRL	Provides control for the Programmer (Eycon™ 10/20 Visual Supervisor/Tactician T2550)				3/0✓	5/0✓	✓	
	PROGCHAN	Configuration of a channel (Eycon™ 10/20 Visual Supervisor/Tactician T2550)				3/0✓	5/0✓	✓	
RECORDER	SEGMENT	Display of up to 4 Segments of a channel (Eycon™ 10/20 Visual Supervisor/Tactician T2550)				3/0✓	5/0✓	✓	
	DR_ALARM	Alarm filtering when recording to log files and/or printers				✓			
	DR_ANCHP	Data recording analogue channel point block				✓			
	DR_DGCHP	Data recording digital channel point block				✓			
	DR_REPRT	Generate user-defined text reports and send them to printers				✓			
	HISTDATA	Allows the inclusion of trend and alarm/event data into a printed report.				✓			
	RGROUP	Data recording point group block					4.0✓	✓	
56000	Refer to the LIN Block Reference Manual HA082375U003, iss 15 (Vintage).								
SELECTOR	2OF3VOTE	Selects 'best' input from three, by averaging only the within-tolerance inputs	✓	✓		✓	✓	✓	
	ALC	Alarm collection producing a common logic O/P	✓	✓		✓	✓	✓	
	SELECT	Outputs highest, middle, lowest, or median of 2, 3, or 4 inputs	✓	✓		✓	✓	✓	
	SWITCH	Single-pole double-throw switch for analogue signals	✓	✓		✓	✓	✓	
	TAG	Specifies a user task (loop) tagname, selected from list of 8 tags	✓						
TAN	Refer to the LIN Block Reference Manual HA082375U003, iss 15 (Vintage).								

Category	Block	Function	T640	T940(X)	T225(X)	Eycon-10/20	T2550/T820	T2750
TIMING	DELAY	Delay for deadtime applications	✓	✓		✓	✓	✓
	DTIME	Delay, for deadtime applications		✓			✓	✓
	RATE_ALM	Up- and down-rate alarm applied to PV, with OP held at last non-alarm value	✓	✓		✓	✓	✓
	RATE_LMT	Rate-limiter and ramp generator	✓	✓		✓	✓	✓
	SEQ	Multi-segment slope/level/time, 15 O/P digitals	✓	✓		✓	✓	✓
	SEQE	SEQ extender	✓	✓		✓	✓	✓
	TIMEDATE	Clock and calendar event	2/1✓	✓		✓	✓	✓
	TIMER	Timer	✓	✓		✓	✓	✓
	TOTAL	Totaliser (integrator) for analogue variable	✓	✓		✓	✓	✓
	TOTAL2	Totaliser (integrator) for analogue variable, with calibration mode	4/1✓	✓		✓	✓	✓
	TOT_CON	Placeholder for 32-bit parameter values (e.g. totalisations)	4/1✓	✓		✓	✓	✓
	TPO	Time-Proportioning output	4/1✓	✓		✓	✓	✓

✓ = fully supported; S = fully supported in Sequence version only;

A = supported in 'Advanced' option only.

Revision number (e.g. 2/1) denotes software version where block first appeared.

*Partial access can be customised via LINtools

Table 2 Function block categories & instrument support

CHAPTER 2 BATCH FUNCTION BLOCKS

The BATCH category of Function Block Templates provides the control strategy with functions for batch and sequence programming.

SFC_CON: SFC CONTROL BLOCK

Block function

The SFC (Sequential Function Chart) Control block, allows LIN Sequences to be selected and run during the main Strategy runtime.

Digital signals wired to the block from the control strategy can load a Sequence file (specified in *FileName*) to RAM (*Load*), run the loaded LIN Sequence (*Run*), interrupt the LIN Sequence whilst allowing current actions to continue (*Hold*), or reset it (*Init*). The *Filepath* parameter allows a LIN node and drive to be specified when the source of the LIN Sequence (.sbf) file is remote.

The SFC_CON block provides alarm outputs warning of file-loading errors (*LoadFile*), requests for actions to be run that are already running (*Actions*), potentially disruptive RAM memory overloads due to sequence processing (*Overload*), and also a 'handshake' bit (*Clear*) that goes low only when a sequence file is successfully loaded.

Input priorities

(See the *Block specification menu* section below for more details on the input parameters.)

- *Load* has priority 1 (highest). If it goes FALSE the LIN Sequence stops and is cleared from RAM whatever the states of the other inputs.
- *Run* has priority 2, and its state determines whether *Hold* and *Init* have any effect. *Init* has no effect when *Run* is TRUE; *Hold* has no effect when *Run* is FALSE.
- *Hold* has priority 3. Its state controls the execution of transitions, which can occur only while the sequence is running. When the LIN Sequence is stopped, *Hold* has no effect.
- *Init* also has priority 3. Its state controls the resetting of the LIN Sequence, but only when *Run* is FALSE.

Reset control: the Init parameter

Holding *Init* TRUE means that each time the sequence is stopped, it resets. That is, all steps are de-activated, 'F' qualified actions are executed, and timers zeroed. If the sequence is re-started it commences at the initial step(s).

Holding *Init* FALSE means that active steps remain active and timers are not reset when the LIN Sequence is stopped, but are 'frozen'. 'F' qualified actions are not executed as the LIN Sequence stops. If the LIN Sequence is re-started, it continues from where it left off with the same steps active and the timers simply continuing. Restarted steps do not re-execute 'P' qualified actions.

Changing *Init* from FALSE to TRUE, with the sequence stopped, resets the sequence, but 'F' qualified actions in de-activated steps are *not* executed this time. Changing *Init* from TRUE back to FALSE again after a reset does not put the sequence back to where it was. It always restarts from its initial step(s) if *Init* has been TRUE while *Run* was FALSE.

Block parameters

Symbols used in *Table 3* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Run	Start/continue running sequence	T/F	
Hold	Hold sequence at current step(s)	T/F	
Init	Return to initial step(s) (if Run FALSE)	T/F	
Load	Load specified sequence to RAM	T/F	
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
Clear	File load 'handshake' bit (low = O.K. load)	T/F	
LoadFile	File-loading error (hardware/software)	T/F	
Actions	Request to run an already-running action	T/F	
Overload	RAM overloaded by sequence processing	T/F	
Combined	OR-ing of all Alarms bits	T/F	
FileName	Filename of sequence to be loaded	Alphanumeric	
Filepath	LIN location of file specified in FileName	Alphanumeric	
DispBlk	Name of linked SFC_DISP block	Alphanumeric	

Table 3 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these 'header' fields.

Run. With *Run* TRUE the sequence loaded in RAM runs, i.e. has active step(s), or resumes running. (The *Init* parameter determines whether the sequence resets or not.) With *Run* FALSE the sequence stops running. Note that the *Load* input can override *Run*.

Hold. While *Hold* is TRUE a running sequence continues to run but transitions from one step to the next are blocked, even if the transition condition is TRUE. (*Hold* has no effect while *Run* is FALSE.)

Init. Making *Init* TRUE returns the sequence to its initial step(s) if *Run* is or becomes FALSE. With *Init* TRUE and *Run* TRUE, initialisation is enabled but not executed (unless *Run* becomes FALSE).

Load. A rising edge, FALSE-to-TRUE transition, input to *Load* copies the sequence specified by the *FileName* and *Filepath* parameters to RAM, where it can be controlled via the other block inputs. *Load* must remain TRUE to keep the loaded sequence in RAM; making it FALSE unloads the sequence and clears the memory. The *Load* input overrides all the others.

Alarms. See *Appendix D page 545* for a general description of the *Alarms* field.

- **Software.** TRUE if memory corrupted, or cached block goes off-line.
- **Clear.** Sets high when there is no sequence loaded in RAM, but resets when a sequence load is successful. Can be used as a 'handshake' bit for the load operation.
- **LoadFile.** Indicates file-loading errors caused by either hardware (e.g. faulty drive) or software (e.g. non-existent filename or path) faults.
- **Actions.** Indicates an invalid block.field is referenced in an action, or an illegal request to initiate an action that is already running, e.g. an S-qualifier action repeated in a step. Note that it is not illegal to request an already-stopped action to cease.
- **Overload.** Indicates the RAM is overloaded by sequence processing, i.e. the Sequence is too 'busy'. In these conditions sequence behaviour could be affected, e.g. actions or transitions not performed.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

FileName. User filename of up to eight characters specifying the sequence file to be loaded to RAM at runtime. *FileName* can be changed at runtime, provided the sequence is not loaded (*Load* = FALSE).

Filepath. Specifies the location of the LIN Sequence file named in the *FileName* parameter. See “Filepath parameter” on page 547. for details.

Note: It is strongly recommended that remote filepaths are NOT used and in particular, not redundant processor configured instruments.

DispBlk. Tagname of the SFC_DISP block (if any) linked to the SFC_CON block. The SFC_DISP block allows a sequence running in a remote instrument to be displayed/monitored/controlled via LINTools’ ‘Connect’ facility.

The *DispBlk* field is automatically loaded with the linked SFC_DISP block’s tagname when the link is made from the SFC_DISP block end, and automatically clears when the link is broken. Also, entering a tagname in *DispBlk* automatically enters the corresponding tagname in the SFC_DISP block’s *SFCBlock* field, to complete the link, see the SFC_DISP block, for more details.

SFC_MON: SFC MONITOR BLOCK

Block function

SFC (Sequential Function Chart) Monitor blocks allow the running of sequences to be monitored during the main control strategy runtime. Each SFC_MON block can monitor up to eight steps in the sequence being controlled by the corresponding SFC_CON block (named in the *SfcBlock* parameter). There is no restriction other than available RAM on the number of SFC_CON and SFC_MON blocks in a strategy. For sequences of greater than eight steps, extra SFC_MON blocks can be created as needed.

The SFC_MON block has eight trios of parameters indicating the alphanumeric name of the step (*Step_A* to *Step_H*), whether or not the step is currently active (digitals *A_X* to *H_X*), and the time (secs) the step has been active (analogues *A_T* to *H_T*). The eight digitals and analogues are available as read-only outputs to the control strategy. The block has no input connections.

The block provides a special *SfcBlock* alarm output warning if the SFC_CON block tagname specified in the SFC_MON block's *SfcBlock* field is invalid for any reason (e.g. non-existent or referring to the wrong block type).

Block parameters

Symbols used in *Table 4* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
SfcBlock	Tagname of corresponding SFC_CON block	Alphanumeric	
Step_A to Step_H	Names of monitored steps	Alphanumeric	
A_X to H_X	Activity states of steps A to H (TRUE = Active)	T/F	
A_T to H_T	Running time (secs) of steps A to H	Eng	
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
SfcBlock	Invalid SfcBlock field specified	T/F	
Combined	OR-ing of all Alarms bits	T/F	

Table 4 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these 'header' fields.

SfcBlock. 'Tagname' (*Block* field) of the SFC_CON block that loads and controls the monitored sequence.

Step_A to Step_H. Names of the (up to) eight steps in the monitored sequence, to which the corresponding activity (*A_X* to *H_X*) and timing (*A_T* to *H_T*) parameters refer.

A_X to H_X. TRUE if the corresponding step (*Step_A* to *Step_H*) is currently active. Note that for an active step the parameter remains TRUE while a sequence is STOPped (*Init* FALSE, *Run* FALSE), but goes FALSE when the sequence is RESET (*Init* TRUE, *Run* FALSE).

A_T to H_T. Indicate the time in seconds that the corresponding step (*Step_A* to *Step_H*) has been active. Note that for an active step the parameter 'freezes' while a sequence is STOPped (*Init* FALSE, *Run* FALSE), but is zeroed when the sequence is RESET (*Init* TRUE, *Run* FALSE).

NOTE. As an alternative to these parameters, 'T' can be used in a structured text statement or transition to indicate step activity time, without reference to an SFC_MON block. E.g. **REACT.T** is the activity time in seconds of the step called 'REACT'.

Alarms. See *Appendix D page 545* for a general description of the *Alarms* field.

- **Software.** TRUE if memory corrupted, or cached block goes off-line.
- **SfcBlock.** Indicates that the *SfcBlock* field is invalid, e.g. specifies a non-existent SFC_CON block.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

SFC_DISP: SFC DISPLAY BLOCK

Block function

The SFC Display block allows a sequence running in a remote instrument to be graphically displayed, monitored, and controlled via LINTools' 'Connect' facility.

In use, the SFC_DISP block together with an SFC_CON block are run in the remote instrument, linked to each other via their respective *DispBlk* and *SfcBlock* parameters.

Block parameters

Symbols used in *Table 5* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
SfcBlock	Tagname of linked SFC_CON block	Alphanumeric	
ActStepA to E	Names of active steps (5 max. displayed)	Alphanumeric	
StepName*	Specifies existing step name	Alphanumeric	
SAtt Out*	Specifies attributes to be changed (Bit-mask)	Numeric	
SAtt In*	Specifies required attribute values (Bit-mask)	Numeric	
SDone*	TRUE flags completion of write operation	T/F	
Token	Name of step holding TRAK token	Alphanumeric	
Extend	Name of block that contains additional SFC steps/actions	Alphanumeric	
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
Bad_Seq	No linked SFC_CON block, or corrupt SFC in RAM	T/F	
More_Sts	More steps currently active than can be displayed	T/F	
More_Act	More actions currently active than can be displayed	T/F	
Clear	No Sequence Loaded	T/F	
Combined	OR-ing of all Alarms bits	T/F	
AActionA to D	Names of active actions (4 max. displayed)		
TrActIx*	Specify transition to be addressed		
TrTrIx*			
TAtt Out*	Specifies attributes to be changed (Bit-mask)	Numeric	
TAtt In*	Specifies required attribute values (Bit-mask)	Numeric	
TDone*	TRUE flags completion of write operation	T/F	
ClrHlds	TRUE clears all HOLDS, ENABLES, DISABLES	T/F	
Running	TRUE runs SFC_DISP block update routine	T/F	

**For manufacturers use only*

Table 5 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these 'header' fields.

SfcBlock. Tagname (*Block* field) of the SFC_CON block linked to the SFC_DISP block. At any time, only one SFC_CON block can be linked to a given SFC_DISP block, and vice-versa. The software automatically ensures this at configuration time as follows. When a tagname is entered into the *SfcBlock* field, any existing conflicting links are cleared (with null fields), and the nominated SFC_CON block's *DispBlk* field is written to, establishing a two-way link. Corresponding automatic actions also occur if the link is made from the SFC_CON block's *DispBlk* field.

ActStepA to ActStepE. Names of (up to) five currently active steps in the monitored sequence, displayed in no significant order. For a remotely running sequence using LINTools, a maximum of five steps can be displayed at a time unless the *Extend* parameter is used (see below), in which case there is no theoretical limit.

Token. Name of the step (if any) currently holding the TRAK token, i.e. located under the cursor in the runtime sequence screen, with the TRAK facility activated.

NOTE. The TRAK facility makes the cursor follow the currently active step as it progresses round the displayed sequential function chart.

Extend. Optional. Used to hold the name of a block type SFC_DISP_EX which defines additional SFC steps and actions. The SFC_DISP block permits the remote monitoring of five SFC steps and four SFC actions. By using an SFC_DISP_EX block, an additional 16 steps and actions can be monitored. Multiple SFC_DISP_EX blocks can be chained together to increase this number further. The *Extend* parameter contains the name of the first SFC_DISP_EX block.

Alarms. See *Appendix D page 545* for a general description of the *Alarms* field.

- **Software.** TRUE if memory corrupted, or cached block goes off-line.
- **Bad_Seq.** TRUE if no linked SFC_CON block found, or corrupt sequence in RAM.
- **More_Sts.** TRUE if more than five steps are active, and therefore not all can be displayed in the *ActStepA* to *E* fields. This figure increases by multiples of sixteen for each SFC_DISP_EX block linked to this block (using the *Extend* parameter).
- **More_Act.** TRUE if more than four actions are active, and therefore not all can be displayed in the *AActionA* to *D* fields. This figure increases by multiples of sixteen for each SFC_DISP_EX block linked to this block (using the *Extend* parameter).
- **Clear.** TRUE if there is no sequence loaded (via the SFC_CON block).
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

AActionA to AActionD. Names of (up to) four currently active actions (including ROOT) in the monitored sequence, listed in no significant order. For a remotely running sequence using LINTools or a locally running sequence, the number of actions displayed is not limited.

ClrHlds. Setting *ClrHlds* TRUE clears all HOLD STEP, ENABLE or DISABLE TRANSITION commands applied to steps or transitions via the runtime sequence screen. *ClrHlds* self-resets after use.

Running. Must be TRUE for the SFC_DISP block to be updated. *Running* sets TRUE automatically whenever a remote sequence is selected for runtime viewing, and resets to FALSE when the runtime sequence screen is quit.

SFC_DISP_EX: SFC DISPLAY BLOCK

Block function

The SFC_DISP_EX block is an extension to the SFC Display (SFC_DISP) block. It allows for an additional 16 steps and actions to be graphically displayed, monitored, and controlled via the LINTools' 'Connect' facility. Using only the SFC_DISP block (without the SFC_DISP_EX block), only five steps and four actions can be displayed.

To extend the number of steps and actions further, additional SFC_DISP_EX blocks can be linked in a daisy-chain fashion from this block. It is not recommended, however, that multiple SFC_DISP_EX blocks are used. For information on the SFC_DISP block, refer to page 24.

Block parameters

Symbols used in *Table 6* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
DispBlk	Name of the block this is extending	Alphanumeric	
ActStepA to P	Names of active steps	Alphanumeric	
AActionA to P	Names of active actions	Alphanumeric	
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
Combined	OR-ing of all Alarms bits	T/F	
Extend	Name of block that contains additional SFC steps/actions	Alphanumeric	

Table 6 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these 'header' fields.

DispBlk. The name of the root SFC_DISP block being extended.

ActStepA to ActStepP. Names of (up to) 16 currently active steps in the monitored sequence, displayed in no significant order. For a remotely running sequence using LINTools, a maximum of twenty-one steps can be displayed at a time (five from the SFC_DISP block and an additional sixteen from this block) unless the *Extend* parameter is used (to add multiples of an additional sixteen).

AActionA to AActionP. Names of (up to) 16 currently active actions in the monitored sequence, listed in no significant order.

Alarms. See *Appendix D page 545* for a general description of the *Alarms* field.

- **Software.** TRUE if memory corrupted, or cached block goes off-line.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

Extend. Optional. Used to hold the name of an additional SFC_DISP_EX block to add a further sixteen SFC steps and actions if required.

RECORD: RECORD BLOCK

Block function

This block stores and retrieves sets of analogue and digital values for use in control strategies and sequences, as recipes for example. It holds up to eight analogue and sixteen digital values (in the *A0 - A7* and *Bit0 - Bit15* parameters). It can be triggered (via the *Get* input), at runtime or during configuration, to load new sets of values into its parameters from special record files, and to save complete sets of values to these files (via the *Put* input).

Record files

Record files are created automatically by this block as required (named via its *Filename* parameter plus extension *.RCD*) and can each hold up to 50 complete sets of values (records). The *RecordNo* parameter, which can be a control strategy input, specifies the number of the record to be loaded/saved. Records are numbered from 0 to 49 in each record file, and each record includes an (up to) eight-character text caption.

A record file prepared with a series of records is used to implement a batch control mechanism. Part of each record's data may be interpreted as a process number, i.e. to define which sequence is run, the rest of the data being parameters for that particular process. Stepping through the records in series (via the *GetNext* parameter) will then run a predefined series of batches.

Runtime data collection

The block can input data from and output it to a control strategy, allowing it to collect, store, and use values generated as the control strategy runs, e.g. new recipes, results and statistics from the process.

The block provides a special alarm output, *File*, indicating bad file name, bad record number, and load/save hardware/software errors.

Block parameters

Symbols used in *Table 7* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these 'header' fields.

Caption. 8-character (max.) text caption to identify a data record, stored with each record.

A0 to A7. Set of 8 analogue values held in the RECORD block, for use in the control strategy and/or sequences as recipe values etc. These values can be input from or output to the control strategy, and can be loaded from or saved to specified data records at runtime or during configuration, triggered by the *Get* and *Put* block inputs.

Parameter	Function	Units	Status
Caption	User-defined (stored) record identifier	Alphanumeric	
A0 to A7	Analogue values	Eng	
D	Digital values	ABCD hex	
Bit0	Digital value 1	T/F	D
Bit1	Digital value 2	T/F	
Bit2	Digital value 3	T/F	
Bit3	Digital value 4	T/F	
Bit4	Digital value 5	T/F	C
Bit5	Digital value 6	T/F	
Bit6	Digital value 7	T/F	
Bit7	Digital value 8	T/F	
Bit8	Digital value 9	T/F	B
Bit9	Digital value 10	T/F	
Bit10	Digital value 11	T/F	
Bit11	Digital value 12	T/F	
Bit12	Digital value 13	T/F	A
Bit13	Digital value 14	T/F	
Bit14	Digital value 15	T/F	
Bit15	Digital value 16	T/F	
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
File	Bad filename/record no. or faulty/missing drive	T/F	
Combined	OR-ing of all Alarms bits	T/F	
FileName	Filename of record file for load/save operations	Alphanumeric	
FilePath	Record file's LIN node address / drive code	Alphanumeric	
RecordNo	Filename record number (0 - 49)	Eng*	
Get	Load record block values from specified record	T/F	
GetNext	Increment RecordNo & get record	T/F	
Put	Copy record block values to specified record	T/F	
GetOK	'Get successful' flag (else File alarm set)	T/F	
PutOK	'Put successful' flag (else File alarm set)	T/F	

*Fractional numbers input are rounded up to integer values.

Table 7 Block parameters

D. Bitfield of 16 digital values (*Bit0* to *Bit15*) held in the RECORD block, used as for the *A0* to *A7* parameters.

Alarms. See Appendix D page 545 for a general description of the *Alarms* field.

- **Software.** TRUE if memory corrupted, or cached block goes off-line.
- **File.** TRUE if *Put* or *Get* attempted involving a bad (e.g. non-existent) filename or a bad (e.g. blank or out-of-range) record number.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

FileName. 8-character (max.) filename for the storage of up to 50 records. The specified filename becomes the root filename of a record file, with the extension .RCD automatically added. *FileName* can be changed at runtime, via the block specification menu, to access other records.

Filepath. Specifies the location of the LIN Sequence file named in the *FileName* parameter. See "Filepath parameter" on page 547. for details.

RecordNo. Individual record number (0 to 49) to be accessed in a record file (specified by the *FileName* parameter) when the *Get*, *GetNext*, or *Put* operations are triggered. A record file can store up to 50 records, each record holding a complete set of 8 analogue and 16 digital values, together with a text *Caption*.

Get. Inputting a rising edge (FALSE-to-TRUE transition) to *Get* copies the analogue and digital values and text caption stored in the record specified by *FileName*, *FilePath*, and *RecordNo* to the corresponding parameters of this block (*A0* to *A7*, *D*, and *Caption*). Any existing values are overwritten. *Get* self-resets at runtime if not connected.

NOTE 1. To *Get* data from (or *Put* data to) a record file stored in a remote instrument, a RECORD block should be local to the instrument where the record file resides, and cached in the instrument that is to *Get* or

Put the data (see Figure 1). In this way the data transfer is by block update, which is faster and less restricted than server/client file transfer.

Warning: When data is loaded from a file by asserting the Get or GetNext, do not make a Transition in an SFC depend on file data values (Caption, A0..A7, D) in a duplex system. The devices can desynchronise.

NOTE 2. Never wire to the Get or Put inputs of a cached RECORD block. This is because the corresponding real block cannot detect the wiring and so self-resets these inputs, in conflict with the cached block's non-resetting (wired) inputs. Instead, operate Get or Put manually, or via an SFC statement, or by transmitting digitals via a DG_CONN block cached in the remote instrument and wired there to the local RECORD block, as exemplified in Figure 1.

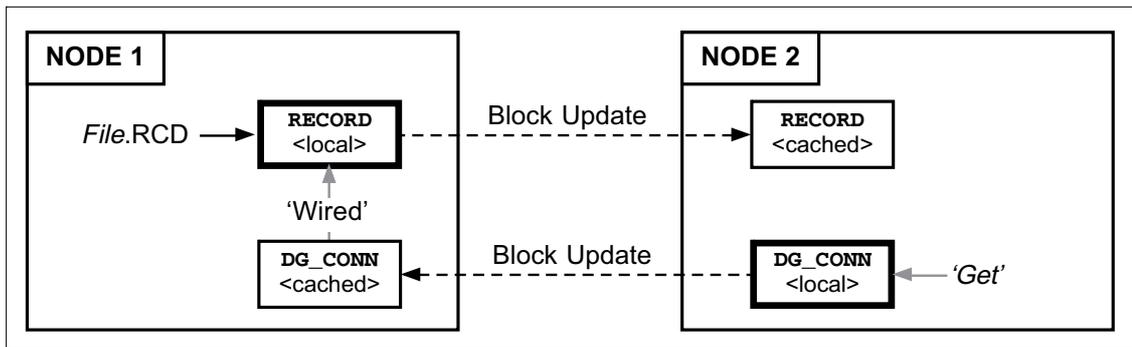


Figure 1 'Get' used across the network via cached blocks

GetNext. Making GetNext TRUE loads the next record from the file, i.e. increments RecordNo and then does a 'Get'. After record number 49 the number 'rolls over' to 0. RecordNo may be preset to 49 (or -1) to allow GetNext to read in record number 0. GetNext self-resets after use.

Warning: When data is loaded from a file by asserting the Get or GetNext, do not make a Transition in an SFC depend on file data values (Caption, A0..A7, D) in a duplex system. The devices can desynchronise.

Put. Inputting a rising edge (FALSE-to-TRUE transition) to Put copies the analogue and digital values and text caption currently stored in the RECORD block (in A0 to A7, D, and Caption) to the record specified by FileName, FilePath, and RecordNo. Any existing values in the record are overwritten. Put automatically reverts to FALSE.

NOTE 1. Please refer to Notes 1 and 2 in the Get section above.

NOTE 2. Put, when applied to any instrument's E: drive at runtime can be used only to overwrite an existing record, not to create a new one.

NOTE 3. The filing system used in older T640 instruments (before v4/1) does not allow files on the E: device to grow. So, with these instruments, the very first Put must specify a RecordNo equal to the highest record number that will be required by the strategy. This reserves enough E: drive filing space to accommodate all record numbers from the specified maximum down to record number zero.

GetOK. Sets TRUE after a Get or GetNext operation is successful. Otherwise GetOK sets FALSE and the File alarm sets.

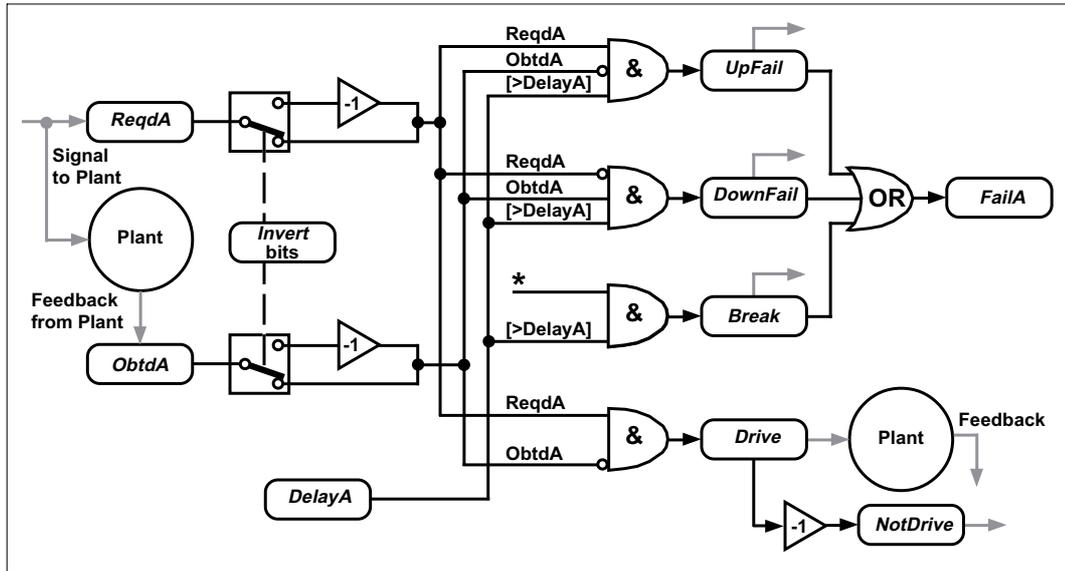
IMPORTANT: In T940(X) prior to version 5, and T2550 prior to version 1.2, synchronised duplex devices may de-synchronise if this field is tested in a Transition.

PutOK. Sets TRUE after a Put operation is successful. Otherwise PutOK sets FALSE and the File alarm sets.

IMPORTANT: In T940(X) prior to version 5, and T2550 prior to version 1.2, synchronised duplex devices may de-synchronise if this field is tested in a Transition.

DISCREP: DISCREPANCY BLOCK

Block function



*Refer to the Block specification menu section for details on the Break parameter.

Figure 2 Block schematic - Channel A

Please refer to the schematic in *Figure 2*. The DISCREP block provides a means of checking that digitally controlled plant devices are responding correctly. It does this essentially by comparing signals transmitted to the plant (the ‘required’ states) with corresponding signals fed back from the plant (the ‘obtained’ states). If these signals do not match in specified ways (allowing for specified time delays), appropriate block output bits and alarms are set which can be used as warnings, diagnostics, and as links to a control strategy.

Inputs

The block contains four identical and independent channels, A to D, which can be individually enabled/disabled if required (via the *DisableA* to *DisableD* inputs). For clarity, only one of these (Channel A) is schematised in *Figure 2*. Each channel also has an input monitoring the digital state required (*ReqdA* to *ReqdD*), and an input for the feedback device state actually obtained (*ObtdA* to *ObtdD*). All ‘Reqd’ and ‘Obtd’ inputs can be individually inverted via the block’s *Invert* bitfield.

Outputs

Each channel has five digital outputs (grouped in the *OutA* to *OutD* bitfields) containing outputs *Drive* and *NotDrive*, *UpFail*, *DownFail*, and *Break*.

These outputs set when certain discrepancies occur between the ‘required’ and ‘obtained’ inputs, schematised in *Figure 2* and detailed in the *Block specification menu* section below. The *Drive* output can be used for the ‘On/Off’ control of an actuator. A time delay parameter for each channel (*DelayA* to *DelayD*) allows a tolerable delay (e.g. ‘travel time’) to be defined between a ‘Reqd’ signal changing state and its ‘Obtd’ signal switching accordingly, before an error is flagged. *UpFail* sets if the device fails to turn on, and *DownFail* sets if the device fails to turn off, after the delay. *Break* is asserted when ‘Obtd’, having equalled ‘Reqd’, subsequently changes state (after the delay time, to allow for ‘contact bounce’).

Alarm output bits (*FailA* to *FailD*) indicate for each channel if its *UpFail*, *DownFail*, or *Break* parameters have been set.

Block parameters

Symbols used in *Table 8* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Invert	Input inversion bits	Bitfield	
ReqdA	Invert ReqdA input	T/F	
ObtdA	Invert ObtdA input	T/F	
ReqdB	Invert ReqdB input	T/F	
ObtdB	Invert ObtdB input	T/F	
ReqdC	Invert ReqdC input	T/F	
ObtdC	Invert ObtdC input	T/F	
ReqdD	Invert ReqdD input	T/F	
ObtdD	Invert ObtdD input	T/F	
ReqdA to ReqdD	Required element state, channels A to D	T/F	
ObtdA to ObtdD	Obtained element state, channels A to D	T/F	
DelayA to DelayD	Time delay, channels A to D	Eng	
DisableA to Disabled	Channel disable, channels A to D	T/F	
OutA to OutD	Outputs, channels A to D	Bitfield	
Drive	'ON/OFF' control output	T/F	
NotDrive	Inverse of drive	T/F	
UpFail	Flags failure of obtd. to set (in <delay)	T/F	
DownFail	Flags failure of obtd. to reset (in <delay)	T/F	
Break	Flags break in obtd. state (after delay)	T/F	
Alarms			
Software	Block RAM data sumcheck error/network failure	T/F	
FailA to FailD	Fault on channel A to D (UpFail, DownFail, Break)	T/F	
Combined	OR-ing of all Alarms bits	T/F	

Table 8 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D* page 544 for details of these 'header' fields.

Invert. 8-bit bitfield set to invert the corresponding block input. Discrepancy checking operations are carried out on the inputs *after* any inversions, as shown in *Figure 2*.

ReqdA to ReqdD. Each of these four digital inputs (together with its *Invert* bit) indicates the required state of a plant element, for comparison with the actual obtained state of its feedback signal (*ObtdA* to *ObtdD*). The 'Reqd' and 'Obtd' parameter states should equalise when the plant element has reached the required state.

ObtdA to ObtdD. Each of these four digital inputs (together with its *Invert* bit) is used to feed back a signal indicating the actual obtained state of a plant element.

DelayA to DelayD. Analogue parameters specifying for each channel the time delay allowed between 'Reqd' changing state and 'Obtd' switching accordingly, before an error (*UpFail* or *DownFail*) is flagged. The *Break* output also makes use of the *Delay* parameter. The delay timers zero and restart whenever the corresponding 'Reqd' parameter changes state.

NOTE. In *Figure 3*, [*>DelayA*] signifies 'time specified by *DelayA* exceeded' which can be TRUE or FALSE.

DisableA to Disabled. Making these inputs TRUE disables the corresponding block channels, i.e. no outputs are generated whatever the state of the inputs. These parameters default to FALSE (i.e. enabled) when not connected.

OutA to OutD. Bitfields of five outputs indicating for each channel the results of the discrepancy checking involving its 'Reqd', 'Obtd', and 'Delay' parameters. (Refer to the Block schematic in *Figure 2*, above.)

- **Drive.** TRUE when 'Reqd' is TRUE and 'Obtd' is FALSE. *Drive* can be used in 'ON/OFF control mode' to drive an actuator until the feedback 'Obtd' indicates that it has reached the required position 'Reqd'.
- **NotDrive.** Inverse of the *Drive* parameter, for use with negative logic.

- **UpFail.** TRUE when 'Reqd' is TRUE but 'Obtd' remains FALSE for longer than the time allowed by 'Delay'. See *Figure 3*.
- **DownFail.** TRUE when 'Reqd' is FALSE but 'Obtd' remains TRUE for longer than the time allowed by 'Delay'.

Figure 3 shows how *UpFail* and *DownFail* work. In the Figure, *ReqdA* goes high at time A, starting the *DelayA* timer. *ObtdA* attempts to follow but does not succeed until time C, after some bouncing and beyond the allowed delay. *UpFail* sets at B, the end of the delay time, but resets at time C, because *ObtdA* has now switched on. At D *ReqdA* resets, but *ObtdA* is still high at the end of the newly triggered delay time so *DownFail* sets at E. *ObtdA* resets to the desired state at F and then bounces, causing *DownFail* to copy the failure pattern.

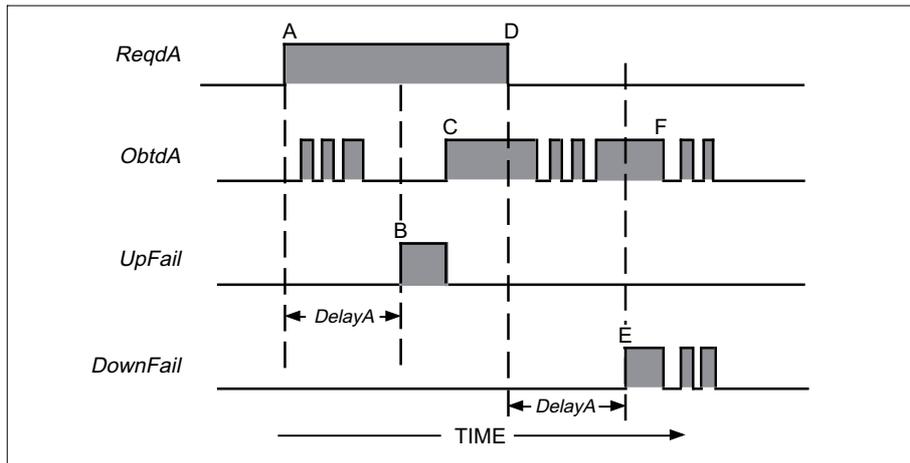


Figure 3 Action of the UpFail & DownFail outputs- Channel A

- **Break.** Break sets TRUE if 'Obtd', having equalled 'Reqd' (since 'Reqd' last switched), subsequently changes state. To prevent bouncing contacts triggering an unwanted *Break*, this output is not asserted until the time specified by 'Delay' has elapsed after 'Reqd' last changed state. *Figure 3* shows how *Break* works.

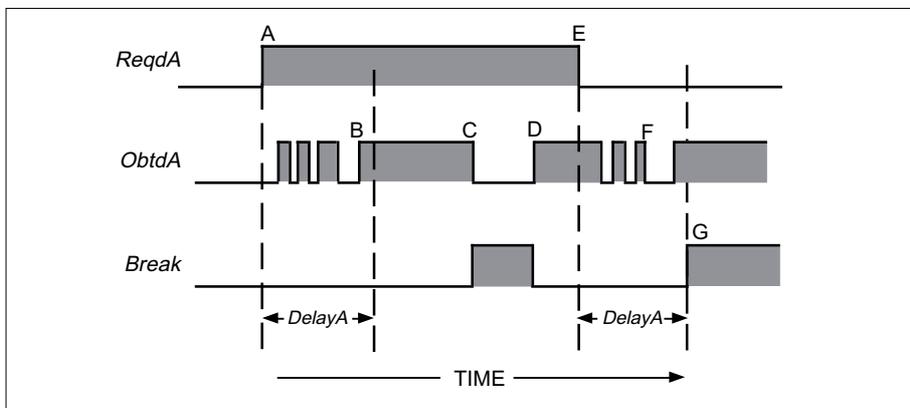


Figure 4 Action of the Break output - Channel A

In *Figure 3*, *ReqdA* goes high at time A, starting the *DelayA* timer. *ObtdA* attempts to follow *ReqdA* and succeeds by time B, after some bouncing. *Break* is not set because B is within the delay time specified. However, *ObtdA* later fails between times C and D, setting *Break* for that period. Note that *Break* is non-latching. At E *ReqdA* resets, but *ObtdA*, despite temporarily resetting by time F, fails again and is left high. So *Break* sets at time G, the end of the newly triggered delay period.

Alarms. See *Appendix D page 545* for a general description of the *Alarms* field.

- **Software.** TRUE if memory corrupted, or cached block goes off-line.
- **FailA to FailD.** Fault on Channel A to D, i.e. *UpFail*, *DownFail*, or *Break* TRUE.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

RCP_SET: RECIPE SET BLOCK

Block function

The RCP_SET function block controls the recipe set, i.e. file to be used, for a number of recipe lines.

Block parameters

Symbols used in *Table 9* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Load	Rising edge loads *.UYR file	T/F	
Filepath	File path for Filename	Alphanumeric	
Filename	Base name of *.UYR file	Alphanumeric	
InUse	TRUE when line being downloaded, i.e. file in use	T/F	
Options	<i>(Options reserved for future use)</i>	(ABCD)	
Active	TRUE while report is being generated	T/F	
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
Config	Too many RCP_SET blocks in database	T/F	
File	Error in Recipe file (*.UYR) on load	T/F	
Combined	OR-ing of all Alarms bits	T/F	
FileErr	File error code	Menu	
FileErrLn	Line number on which FileErr occurred	Integer	
Line1 - Line16	Names of associated RCP_LINE blocks	Alphanumeric	

Table 9 Block parameters

Block specification menu

The following is given in addition to *Table 9*.

InUse. TRUE when a line is being downloaded, to indicate the file is in use. While *InUse* is TRUE, a new file may not be loaded. FALSE indicates that no line is being downloaded.

FileErr. (OK /FORMAT /TOO_BIG /BAD_REF /ACTIVE /NO_FILE /OTHER) File error code, i.e. reason for the file error. The options mean:

- **OK.** No error.
- **FORMAT.** Bad file format.
- **TOO_BIG.** Recipe set too big, i.e. too many lines, too many recipes, too many variables.
- **BAD_REF.** Illegal or non-existent LIN Database reference.
- **ACTIVE.** Recipe file set in use at the moment.
- **NO_FILE.** No file found.
- **OTHER.** Undefined error.

Line1 - Line16. Names of associated RCP_LINE blocks. (*Only Line1 to Line8 currently writeable.*)

SFC_CON: RECIPE LINE BLOCK

Block function

The RCP_LINE function block controls the downloading of recipes from a .UYR file in the associated RCP_SET block. Up to sixteen RCP_LINE blocks may be attached to a single RCP_SET block, via the RCP_SET block's *Line1* to *Line16* fields.

Block parameters

Symbols used in *Table 10* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
RecipeNo	Recipe number	Integer	
Download	Rising edge downloads recipe specified by RecipeNo	T/F	▶□
Abort	Rising edge aborts recipe specified by RecipeNo	T/F	▶□
Capture	Rising edge captures current values & saves file	T/F	▶□
Options	<i>(Options reserved for future use)</i>	(ABCD)	
RprtName	File name for recipe report	Alphanumeric	
TrigRprt	Rising edge generates text report of recipe	T/F	▶□
Alarms			▶□ 📖 🔊
Software	Block RAM data sumcheck error / network failure	T/F	
Config	Block not attached to a RCP_SET block	T/F	
Failed	If the <i>Failed</i> parameter is TRUE, this alarm bit is also set TRUE.	T/F	
Combined	OR-ing of all Alarms bits	T/F	
Status	Current status of recipe	Menu	▶□ 📖
Active	TRUE while download is in progress	T/F	▶□ 📖
Failed	TRUE if last download failed for some reason	T/F	▶□ 📖

Table 10 Block parameters

The following is given in addition to *Table 10*.

TrigRprt. A rising edge input to *TrigRprt* generates a text report of the recipe. The report generated by the RCP_LINE block is a comma-separated-variable file with extension .TXT. It has the following format:

- **Line 1.** Audit trail details of .UYR file
- **Line 2.** Date and time of report
- **Line 3.** Name of RCP_SET block, RCP_LINE block, .UYR file name, line name, recipe name
- **Line 4+.** Name of variable, recipe value, live value (monitor value if specified).

EXAMPLE. The following TREND.UYR on RCP_SET block 'SET1' running recipe 'Run Prog' on RCP_LINE block 'LINE1' could generate the subsequent report text:

```
UYR,1
2,24/08/00,10:55:34, Mike Fox
,30
,Setpoint:1,Monitor,Run Prog
File Name,SppCtrl.RqNxtPrg,SppCtrl.CurrProg,TREND
Ready,SppCtrl.NxtRdy,SppCtrl.State,TRUE
Duration,,SppCtrl.ProgDur,
```

The report generated would be:

```
2,24/08/00,10:55:34,A N Other
,20/09/00,16:34:25,
SET1 ,LINE1 ,TREND.UYR,1,Run Prog
File Name,TREND,TREND,TREND
Ready,TRUE,TRUE,RUN
Duration,,00:37:10
```

Alarms.

■ **Failed.** This is an alarm representation of the *Failed* field (described below).

Status. (RESET /DOWNLOAD /COMPLETE /FAILED) Current status of recipe. The options mean:

- **RESET.** No loaded recipe.
- **DOWNLOAD.** Download in progress.
- **COMPLETE.** Download completed successfully.
- **FAILED.** Last download failed.

ACTIVE. TRUE whilst a download is in progress.

FAILED. TRUE if a download attempt fails. This could be for a number of reasons including, some of the LIN database references in the recipe are invalid, or failed to verify.

BAT_CTRL: BATCH CONTROL BLOCK

Block function

The BAT_CTRL function block governs the loading (from a .UYB file) and control of a batch on instruments that have an inbuilt HMI (for example, the Eycon). It provides a LIN Database environment with a command interface into the relevant batch engine, along with relevant status and support information.

Block parameters

Symbols used in *Table 11* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Mode	Batch operating mode (Only Slave implemented)	Menu	
Filepath	File path for Filename	Alphanumeric	
Filename	Base name of the .UYB file	Alphanumeric	
Options	(Options reserved for future use)	(ABCD)	
Command	Commands into the batch controller	Menu	
CmndFlg	Booleans version of Command	(AB)CD hex	
*[6] LOAD	Load batch & transition to Idle state	T/F	D
[7] RESET	Transition to Reset (or optionally Idle) state	T/F	
[5] START	Initialise batch & transition to Starting state	T/F	
[2] HOLD	Transition to [TRUE] Holding, [FALSE] Restarting, state	T/F	
[4] ABORT	Transition to Aborting state	T/F	C
†[3] STOP	Transition to Stopping state	T/F	
†[1] PAUSE	Transition to [TRUE] Pausing, [FALSE] Resuming, state	T/F	
		T/F	
CmndHshk	Command handshake flags	(AB)CD hex	
STARTED	Starting > Started transition	T/F	D
HELD	Holding > Held transition	T/F	
RESTART	Restarting > Restarted transition	T/F	
ABORTED	Aborting > Aborted transition	T/F	
†STOPPED	Stopping > Stopped transition	T/F	C
†PAUSED	Pausing > Paused transition	T/F	
†RESUMED	Resuming > Resumed transition	T/F	
		T/F	
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
Config	Configuration error	T/F	
File	File processing error	T/F	
Failed	Batch runtime failure (Refer to CondFlg)	T/F	
Combined	OR-ing of all Alarms bits	T/F	
FileErr	Reason for file error	Menu	
FileErLn	Zero if no error, else line no. of first parsing error met	Integer	
StrtDate	Date when batch started	Date	
StrtTime	Time when batch started	Time	
EndDate	Date when batch ended	Date	
EndTime	Time when batch ended	Time	
BatchNo	Unique batch number	Integer	
BatchId	Batch ID string	Alphanumeric	
State	Current state of batch engine's state machine	Menu	

Continued...

Parameter	Function	Units	Status
<i>Continued...</i>			
StateFlg	Booleans version of State	ABCD hex	<input type="checkbox"/>
RESET	No batch is currently loaded	T/F	D
IDLE	Batch loaded & ready to start	T/F	
STARTING	Transitioning from Idle to Running	T/F	
RUNNING	Batch executing normally	T/F	
COMPLETE	Batch automatically terminated normally	T/F	C
HOLDING	Transitioning from Running to Held	T/F	
HELD	Batch manually suspended indefinitely	T/F	
RESTART	Transitioning from Held to Running	T/F	
†PAUSING	Transitioning from Running to Paused	T/F	B
†PAUSED	Batch manually suspended, typically only briefly	T/F	
†RESUMING	Transitioning from Paused to Running	T/F	
†STOPPING	Transitioning from Running to Stopped	T/F	A
†STOPPED	Batch manually terminated normally	T/F	
ABORTING	Transitioning to Aborted	T/F	
ABORTED	Batch manually terminated, with minimal delay	T/F	
FAILED	Batch execution terminated due to unrecoverable failure	T/F	
CondFlg	Batch condition flags	(AB)CD hex	<input type="checkbox"/>
LOADED	Batch is currently loaded	T/F	D
ACTIVE	Start command accepted, batch not yet reached final state	T/F	
		T/F	
		T/F	C
TIMEOUT	Phase logic failed to respond before timeout expired	T/F	
RCP_FAIL	Failure originated from within recipe system	T/F	
LOG_FAIL	Failure originated from within logging system	T/F	
CurPhase	Batch phase currently being executed	Integer	<input type="checkbox"/>

†States not currently implemented

Table 11 Block parameters

Block specification menu

The following is given in addition to *Table 11*.

Filepath. The full path to the specified file. See “Filepath parameter” on page 547 for details.

Filename. Eight-character (max.) filename for the UYB file.

Command. (LOAD/RESET/START/HOLD/RESTART/PAUSE/RESUME/STOP/ABORT/UNLOAD)

Commands into the batch controller. Refer to the *CmndFlg* parameter in *Table 11* for more information on these commands. *Figure 5* shows how some of the commands work in the batch engine state diagram.

CmndFlg. Booleans version of the *Command* parameter. Each *CmndFlg* field is level-sensitive, but is ignored unless the batch engine’s state is relevant to the command (see *Figure 5*).

Acceptance of a command guarantees avoidance of repeated acceptance of the same command. Where multiple command flags are active, they are attempted in descending priority order. (Priorities are shown in *Table 11*.)

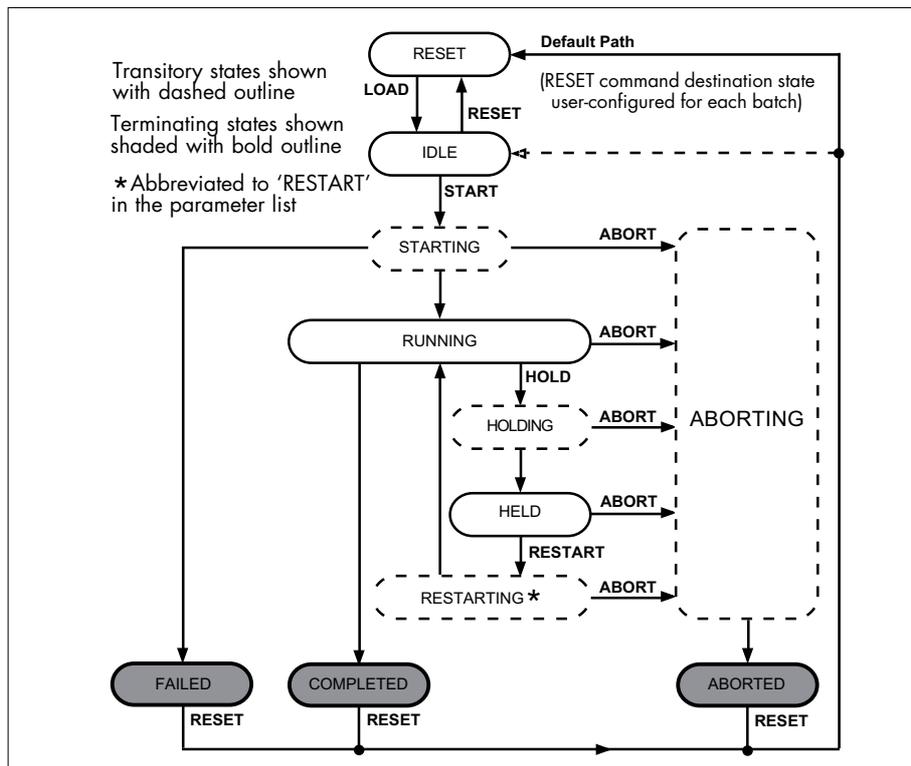


Figure 5 Batch engine state diagram

FileErr. (OK /FORMAT /TOO_BIG /BAD_REF /BAD_ID/BAD_PHAS/BAD_STAT/NO_FILE /FILE_WRI/ SHAR_BLK/BAD_DICT/OTHER) File error code, i.e. reason for the file error. The options mean:

- **OK.** No error.
- **FORMAT.** Bad file format.
- **TOO_BIG.** Batch definition too big, e.g. too many phases.
- **BAD_REF.** Illegal or non-existent LIN database reference.
- **BAD_ID.** Problem with batch controller ID.
- **BAD_PHAS.** (Not implemented.)
- **BAD_STAT.** Invalid batch state.
- **NO_FILE.** No file found.
- **FILE_WRI.** (Not implemented.)
- **SHAR_BLK.** LIN database reference illegally shared by multiple batch controllers.
- **BAD_DICT.** Illegal or non-existent dictionary reference.
- **OTHER.** Undefined error.

State. (RESET/IDLE/STARTING/RUNNING/COMPLETE/HOLDING/HELD/RESTART/PAUSING/PAUSED/ RESUMING/STOPPING/STOPPED/ABORTING/ABORTED/FAILED) Current state within batch engine's state machine. Refer to the *StateFlg* parameter in *Table 11* for more information on these states, and also to *Figure 5*.

Parameter	Function	Units	Status
<i>Continued...</i>			
StateFlg	Booleans version of State	ABCD hex	 
RESET	No batch is currently loaded	T/F	1
IDLE	Batch loaded & ready to start	T/F	2
STARTING	Transitioning from Idle to Running	T/F	4
RUNNING	Batch executing normally	T/F	8
COMPLETE	Batch automatically terminated normally	T/F	1
HOLDING	Transitioning from Running to Held	T/F	2
HELD	Batch manually suspended indefinitely	T/F	4
RESTART	Transitioning from Held to Running	T/F	8
			1
			2
			4
			8
			D
			1
			2
			4
			8
			C
			1
			2
			4
			8
			B
STOPPED	Batch manually terminated normally	T/F	1
ABORTING	Transitioning to Aborted	T/F	2
ABORTED	Batch manually terminated, with minimal delay	T/F	4
FAILED	Batch execution terminated due to unrecoverable failure	T/F	8
			A
CondFlg	Batch condition flags	ABCD hex	 
LOADED	Batch is currently loaded	T/F	1
ACTIVE	Start command accepted, batch not yet reached final state	T/F	2
			4
			8
			D
NOTREADY	TRUE if the batch is not ready to start	T/F	1
TIMEOUT	Phase logic failed to respond before timeout expired	T/F	2
RCP_FAIL	Failure originated from within recipe system	T/F	4
LOG_FAIL	Failure originated from within logging system	T/F	8
SPP_FAIL	Failure originated from within Setpoint Programmer system	T/F	1
CFG_ERR	TRUE if no batch resource available	T/F	2
NOT_IDLE	TRUE if the batch is not currently in the IDLE state	T/F	4
SPP_LOAD	TRUE if the associated programmer block has not loaded	T/F	8
BATID_RQ	TRUE if the BatchId must be modified before starting batch	T/F	1
BF_WIDTH	TRUE if the written string for a field is too long or too short	T/F	2
BF_WR_RQ	TRUE if a compulsory write for a field has not yet occurred	T/F	4
BF_MD_RQ	TRUE if a compulsory update of a field has not yet occurred	T/F	8
RCP_ERR	TRUE if there is a problem with the associated recipe	T/F	A
CurPhase	Batch phase currently being executed	Integer	
PhaseNam	Name of the current phase	Alphanumeric	
RCP_LINE	The name of the associated RCP_LINE block	Alphanumeric	
RecipeNo	Recipe number (from <i>RecipeNo</i> in the RCP_LINE block)	Integer	
BatchFields Page			
Count	The required number of batch user fields (0-7)	Integer	
Desc1-7	The description of a batch user field (presented at the HMI)	Alphanumeric	
Data1-7	The contents of a batch user field (read/write at the HMI)	Alphanumeric	
Extend	Reserved for possible future functionality	Alphanumeric	
BF_ActRQ	Indicates the number of the field requiring action - 0 if none	Alphanumeric	

Table 12 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

The following is given in addition to *Table 12*.

Main Page

This page is used to control a batch and provide status information on the current batch (including start and end time stamps, batch state, and current active phase.)

Filepath. The full path to the specified file. See “Filepath parameter” on page 547. for details.

Filename. Eight-character (max.) filename for the UYB file.

Command. (LOAD/RESET/START/HOLD/RESTART/PAUSE/RESUME/ABORT/UNLOAD) Commands into the batch controller. Refer to the *CmndFlg* parameter in *Table 12* for more information on these commands. *Figure 6* shows how some of the commands work in the batch engine state diagram.

CmndFlg. Booleans version of the *Command* parameter. Each *CmndFlg* field is level-sensitive, but is ignored unless the batch engine’s state is relevant to the command (see *Figure 6*).

Acceptance of a command guarantees avoidance of repeated acceptance of the same command. Where multiple command flags are active, they are attempted in descending priority order. (Priorities are shown in *Table 12*.)

Note: If a hot start occurs whilst the block is in the IDLE state, the block will revert to the RESET state.

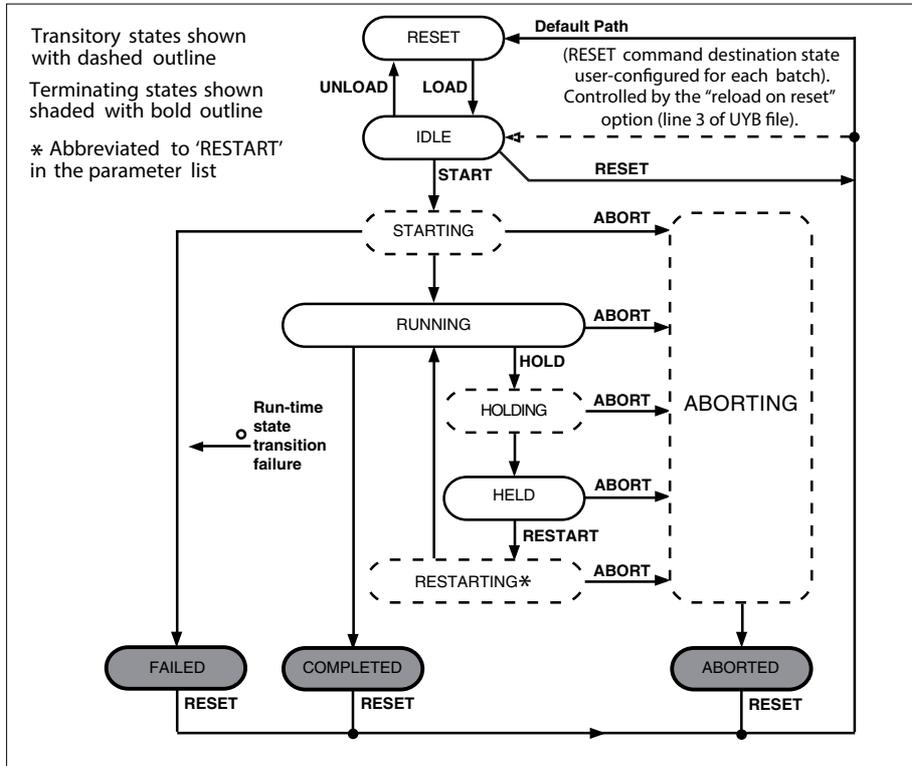


Figure 6 Batch engine state diagram

Alarms. Provides alarm information as appropriate.

- **Config.** Occurs if the block cannot find a spare Batch engine resource. Typically this occurs when more BatchControl blocks are configured than the instrument can support. For example, if the instrument has a limit of 8 batches, the ninth block will show a Config alarm.
- **File.** Occurs when an error is encountered trying to load a UYB file. Refer to *FileErr* and *FilErLn* for reason or location.

FileErr. (OK /FORMAT /TOO_BIG /BAD_REF /BAD_ID/BAD_PHAS/BAD_STAT/NO_FILE /FILE_WRI/SHAR_BLK/BAD_DICT/OTHER) File error code, i.e. reason for the file error. The options mean:

- **OK.** No error.
- **FORMAT.** Bad file format.
- **TOO_BIG.** Batch definition too big, e.g. too many phases.
- **BAD_REF.** Illegal or non-existent LIN database reference.
- **BAD_ID.** Problem with batch controller ID.
- **BAD_PHAS.** (Not implemented.)
- **BAD_STAT.** Invalid batch state.
- **NO_FILE.** No file found.
- **FILE_WRI.** (Not implemented.)

- **SHAR_BLK.** LIN database reference illegally shared by multiple batch controllers.
- **BAD_DICT.** Illegal or non-existent dictionary reference.
- **OTHER.** Undefined error.

StrtDate. The date when the batch was started.

StrtTime. The time when the batch was started.

EndDate. The date when the batch was last terminated.

EndTime. The time when the batch was last terminated.

BatchNo. An instance number which is unique across the lifetime of the product. For each new batch instance, the *BatchNo* parameter increments (when entering the IDLE state). When operating on a synchronised duplex pair, the Secondary assumes the *BatchNo* sequence of the Primary.

BatchId. A string which is updated to provide a text representation of the *BatchNo* parameter, and has an optional prefix if one is specified in the UYB file. The user can modify this text before starting the batch, if required. Maximum 20 (UTF-16) characters.

State. (RESET/IDLE/STARTING/RUNNING/COMPLETE/HOLDING/HELD/RESTART/STOPPED/ABORTING/ABORTED/FAILED) Current state within batch engine's state machine. Refer to the *StateFlg* parameter in *Table 12* for more information on these states, and also to *Figure 6*.

CondFlg. Batch condition flags. Boolean flags indicating the current condition of the batch. Most of these described in *Table 12* are self-explanatory, the following require further explanation:

- **CFG_ERR.** TRUE if no batch resource available (typically too many BATCHCONTROL blocks have been configured).
- **SPP_LOAD.** TRUE if the associated programmer block has not loaded (programmer is identified by where the *PhaseNo* field is wired from).
- **BATID_RQ.** TRUE if the user is required to modify the *BatchId* before starting the batch.
- **BF_WIDTH.** TRUE for one of the fields *Data1-7* if the width of the string is greater or less than the width specifiers in the UYB file.
- **BF_WR_RQ.** TRUE for one of the fields *Data1-7* if the field is required to be written, but the write has not yet occurred.
- **BF_MD_RQ.** TRUE for one of the fields *Data1-7* if the field is required to be modified, but the write has not yet occurred.
- **RCP_ERR.** TRUE if there is a problem with the associated recipe (typically *RecipeNo* is invalid).

For those items "BF_?????" the field *BF_ActRQ* indicates which of the *Data* fields (1 - 7) requires action.

CurPhase. The phase number of the batch currently being executed. This field is typically written from a programmer or SFC. When the batch reaches the end, write 0 to this field to transition to the COMPLETED state.

PhaseNam. The name associated with the current phase number (in the *CurPhase* parameter) which is defined in the batch's UYP file.

RCP_LINE. The name of the associated *RCP_LINE* function block as specified in line 6 of the batch's UYB file.

RecipeNo. The value of this parameter matches that of the *RecipeNo* parameter in the function block referenced by the *RCP_LINE* parameter above. If the *RCP_LINE* parameter is empty, this parameter is set to zero. This parameter is bi-directional, in that you can write to *RecipeNo* in either this block or the referenced block, and the other will update to match.

BatchFields Page

This page is used to define the batch user fields.

Count. The required number of batch user fields (the number of *Desc* and *Data* parameters) which will be used and written to the UHH file (valid values range from 0 to 7). The value is automatically updated when the UYB file is loaded

and set to match the number of batch user fields defined in the file. Note that the Batch ID, File Name, and Recipe Name are always written to the UHH file in addition to any batch user fields selected here.

Desc1 to Desc7. The description of a batch user field. This field is presented at the batch HMI as a label indicating what the user should enter in the *DataN* field (see below). This field is initialised with a default value from the batch's UYB file. Maximum 20 (UTF-16) characters.

Data1 to Data7. The data contents of a batch user field. This field is intended to be presented at the HMI for the user to enter batch-related metadata. The user is presented with an initialised default value from the batch's UYB file. Maximum 60 (UTF-16) characters.

Extend. Not currently implemented.

Note: When the UYB file is unloaded, the *Count* field is set to 0, and the *Desc* and *Data* fields are cleared.

CHAPTER 3 COMMUNICATION FUNCTION BLOCKS

The COMMUNICATION Function Block category comprises communications blocks that are intended to control and/or interface to the communications subsystem; however they are subject to change without notice. If this happens, block sizes will be maintained to allow old databases to be loaded, but the meanings of some fields may alter. The COMM blocks need not be present for correct LIN operation.

GW_CON: GATEWAY CONFIGURATION BLOCK

Block function

This block displays information that is normally available in a Modbus configuration's Port Properties table. In the block, some fields map to the appropriate field in the Modbus Port Properties table, see *MODBUS Tools Help*, Part no. HA028988, for detailed information.

A GW_CON block is used on instruments that support multiple index GW (i.e. more than GWF configuration running simultaneously). A single GW_CON block must be created for each GWF configuration that is to be run. This means that each GateWay index requires a GW_CON block.

Block parameters

Symbols used in this table are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
FileName	8 character GWF filename	Alphanumeric	
Reload	TRUE halts, reloads, and restarts the file if successful	T/F	
GWindex	Shows index number of the active GW index	Integer	
MaxIndex	Max. permitted value of GWindex		
Mode	Master/Slave mode		
InstrNo	Zero in Master mode or Instr no. in Slave mode	Integer	
Type	Communications interface, Serial or TCP	Alphanumeric	
Port	Serial port Name, or TCP port Number	Alphanumeric	
Timeout	Timeout value	Secs	
TalkThru	Name of TCP port master-GW index of slave	Alphanumeric	
Baud	Shows value if Serial port configured Port field	Integer	
Parity	Shows value if Serial port configured Port field	Enum	
DataBits	Shows value if Serial port configured Port field	Enum	
StopBits	Shows value if Serial port configured Port field	Enum	
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
BadFile	TRUE if .gwf is corrupt, missing, or failed to load	T/F	
BadDBF	TRUE if .gwf is not associated with loaded .dbf file	T/F	
BadPort	TRUE if Port does not support requested modbus-GW mode	T/F	
DuplPort	TRUE if Port is already in use	T/F	
TooMany	TRUE if MaxIndex field limit is attained	T/F	
TableOfI	TRUE if one or more tables are offline	T/F	
Config32	TRUE if one or more of the bits in Cfg32Err is TRUE	T/F	
TalkThru	TRUE if GW-master port cannot be resolved	T/F	
PendSave	TRUE if instrument and configuration differ	T/F	
Combined	OR-ing of all Alarms bits	T/F	
Diag0	Mapped to internal diagnostic register	Integer	
Cfg32Err	Inconsistent or 32-bit configuration error	Integer	
diagReg	Diagnostic register, bits currently allocated	Integer	
queryDat	Query data as transmitted	Integer	
ipDelim	Input delimiter as transmitted	Integer	
BdMsgCt	Master Only. Quantity of bad messages received	Integer	
CrcErrCt	Quantity of CRC errors received	Integer	
ErrRspCt	Quantity of error messages sent by slave	Integer	
GoodRxCt	Quantity of acceptable messages received	Integer	
NonRspCt	Quantity of responses not received	Integer	
BadChrCt	Quantity of bad character received	Integer	
ScanPer ^[1]	Ticks taken to scan all tables via comms (master mode only)	tick	
Period ^[1]	Ticks taken to check all tables compared to database	tick	
used ^[1]	Number of ticks taken to check Tables	tick	
delay ^[1]	Number of ticks before starting next ScanPer	tick	
TableCnt	Quantity of Modbus tables used	Integer	

*Read-only unless otherwise stated.

^[1]Instruments that support, single Block Servers run a 4ms xec Tick, The T640 and instruments that support the USERTASK block use Multiple Block Servers and run a 5ms xec tick.

Table 13 Block parameters

Block specification menu

The following is given in addition to *Table 13*.

Dbase, Block, Type. See *Appendix D page 544* for details of these 'header' fields.

FileName. The field containing the 8-character string base file name of the Modbus GateWay File (.gwf file) to be loaded.

NOTE The .gwf file specified here MUST have been created using the *MODBUS Tools software*, see *MODBUS Tools Help*, Part no. HA028988. It MUST also contain a reference to the currently loaded .dbf file. If this referenced file does not meet these criteria the *Alarms.BadDBF* bit will be set.

Reload. TRUE/FALSE. Setting this TRUE forces the active Modbus GW configuration to be temporarily suspended while it is reloaded from the specified *FileName*. The field will auto-reset to FALSE once the reload has been initiated. This can be used to effect a crude form of online reconfiguration by replacing an entire Modbus GW configuration.

NOTE This operation will cause asynchronous update of the block data which will have an impact when operating in duplex synchronised mode.

GWindex. Shows the index number defined by the active Modbus GW configuration. The number ranges from 1 to the maximum number of Modbus GW configurations supported by the configured product, e.g. 3 for the T2550.

NOTE This is the same value as displayed on the 'Select GW index' page of the Terminal Configurator.

MaxIndex. Shows the maximum number of Modbus GW configurations supported by the configured product, e.g. 3 for the T2550.

Type. Serial/TCP. Shows the defined communications interface type.

Mode. Master/Slave. Shows the configured operating mode.

InstrNo. Shows the instrument number when *Mode* is configured to Slave, or Zero when *Mode* is configured to master.

Port. Shows the name of the physical Serial port (COM1, COM2, etc.), or the number of the TCP port (default 502). This field will remain blank when *Mode* is configured as master.

Timeout. Shows the Timeout value of the Modbus GW configurations in seconds.

TalkThru. If Talk-Through is configured this shows the Name of the physical Serial port or Logical Ethernet port running Modbus GW-master to which this Modbus GW-slave is performing Talk-Thru.

Baud, Parity, DataBits, StopBits. Only applicable if a Serial port is specified in the Port field. Shows the operating parameters of the Serial port specified in the Port field.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **BadFile.** TRUE/FALSE. TRUE if the specified .gwf file is corrupt, missing or failed to load. The .gwf file will fail to load if the GW memory has been exhausted, and is considered corrupt if it was not created using the Windows compatible *MODBUS Tools software*, see *MODBUS Tools Help*, Part no. HA028988.

- **BadDBF.** TRUE/FALSE. TRUE if the specified .gwf file is not associated with the currently loaded .dbf file.

- **BadPort.** TRUE/FALSE. TRUE if the configured communications port fails to initialise correctly or the specified .gwf file is attempting to link to a physical serial port or logical TCP index that does not support the specified Modbus GW configuration. Also set to TRUE if the Modbus-Master is not licensed.

- **DuplPort.** TRUE/FALSE. TRUE if the specified .gwf file is attempting to instantiate more than 1, TCP-master and/or TCP-slave, or link to a physical Serial port, COM1, COM2, etc., that already has a GW index associated with it.

NOTE Only one TCP-master or TCP-slave is permitted per instrument.

- **TooMany.** TRUE/FALSE. TRUE if all available Modbus GW configurations are already used, e.g. this alarm would be set TRUE in the fourth GW_CON block of a T2550, as the T2550 only supports 3 Modbus GW configurations.

- **TableOfI.** TRUE/FALSE. TRUE if one or more of the tables in this Modbus GW configuration is 'offline'. This alarm indicates a Modbus communications error. In slaves, this bit is set TRUE if the table has not been written to or read from in the period specified by the *Timeout* parameter, see *MODBUS Tools Help*, Part no. HA028988. In masters, the bit sets TRUE if the slave does not respond to the scanning routines within *Timeout* seconds, i.e. it is effectively 'offline'.

This field can be wired into a TOT_CON block's *UserAlrm* field (or any other suitable input) to indicate Modbus communications failures.

- **TalkThru.** TRUE/FALSE. TRUE if the requested Talk-Through Modbus GW-master port cannot be resolved. This means there is no running Modbus GW-master configuration currently associated with the TalkThru port defined in the *TalkThru* field.
- **Config32.** TRUE/FALSE. TRUE if one or more *Cfg32Err* bits are set TRUE.
- **PendSave.** TRUE/FALSE. TRUE if the image in the instrument memory of this Modbus GW Facility has been modified, e.g. due to removal of invalid block references, since it was loaded from the .gwf file.

NOTE A save of the .gwf file must be performed to make the file match the contents of memory.

Diag0. This field maps directly to the diagnostic register of the Modbus tables. The function of the fields in the diagnostic register allow access to specific Modbus tables.

Cfg32Err. Sets TRUE if a 32-bit configuration error or an inconsistent configuration is detected. It is possible to configure a 32-bit register, see *MODBUS Tools Online Help*, Part no. HA028988, and then inadvertently make the configuration illegal by changing table size, scan count, etc. (a 32-bit configuration error prevents the Modbus communications from running).

An inconsistent configuration is detected when the function codes specified for a table do not allow for all the read/write requirements, M ← S, M → S, etc., of each of the registers/digitals within that table, however the Modbus communications will continue to run.

NOTE If a GW_CON block is not configured, a 32-bit configuration error (*Cfg32Err*) sets the *Disable continuous scan* field in the Modbus GW-master mode diagnostic register, and the Modbus table will not communicate.

diagReg. Diagnostic register, bits currently allocated: Bit 5, Slave in listen-only mode.

queryDat. Query data as transmitted by function code 8 sub code 0.

ipDelim. Input delimiter as transmitted by function code 8 sub code 3.

BdMsgCt. Master only. Quantity of bad messages received. This increments if the reply received from a Modbus GW slave contains an instrument number or function code that does not correspond to the request.

CrcErrCt. Quantity of Cyclic Redundancy Check (CRC) errors received.

ErrRspCt. Quantity of error messages sent by slave.

GoodRxCt. Quantity of acceptable messages received.

NonRspCt. Quantity of responses not received. The *NonRspCt* only increments when a transaction is attempted, but a response is not returned. This will cause the *Alarms.TableOfI* field to be set TRUE.

BadChrCt. Quantity of bad character received.

ScanPer. Master communications scan time. Shows the number of XEC time units taken to complete a scan (read) of each of the configured tables.

NOTE A fault, e.g. cable is disconnected or Slave device loses power, will cause a Master device to lose communications with a Slave device. Once the fault is fixed, Serial link communications can take up to a further 30 secs to be re-established. However, it can take up to 60 secs to re-establish Modbus TCP communications.

Period. Table maintenance task. Shows the number of XEC time units taken to update all data in the Modbus registers of all tables. This value depends on the number of parameters mapped to the Modbus address space, and the number of writes made from the Modbus tables in the Master device to function blocks cached within the Slave device. If data is written from the Modbus address space to the LIN Database, it is not updated from the LIN Database to the Modbus address space until the following scan.

NOTE Data can only be updated in one direction each scan, and only from the Modbus address space to the LIN Database if the value was changed by the Master device. This parameter is not affected if writing to local function blocks.

used. Table maintenance task. This is the total number of ticks taken to check Tables in the Modbus GW configuration.

delay. Table maintenance task. The number of ticks between the completion of one scan of all the Tables and starting the next scan of all the Tables, in XEC time units.

TableCnt. The total number of Tables in the Modbus GW configuration. This value corresponds to the number of entries in the Tables page of the *MODBUS Tools software*, see *MODBUS Tools Help*, Part no. HA028988.

GW_TBL: GATEWAY TABLE BLOCK

Block function

This block displays information that is normally available in a Modbus configuration's diagnostic table. See the *MODBUS Tools Help*, Part no. HA028988, for detailed information on Modbus diagnostic tables.

Using the GW_TBL block is a more efficient way to access Modbus diagnostic data than via a table, and also has the advantage of releasing one table for the configuration.

NOTE Configuring the GW_TBL block disables the corresponding diagnostic tables, thereby avoiding bit-value conflicts.

In the block, the *TableNo* field maps directly to the Modbus table requested as shown below.

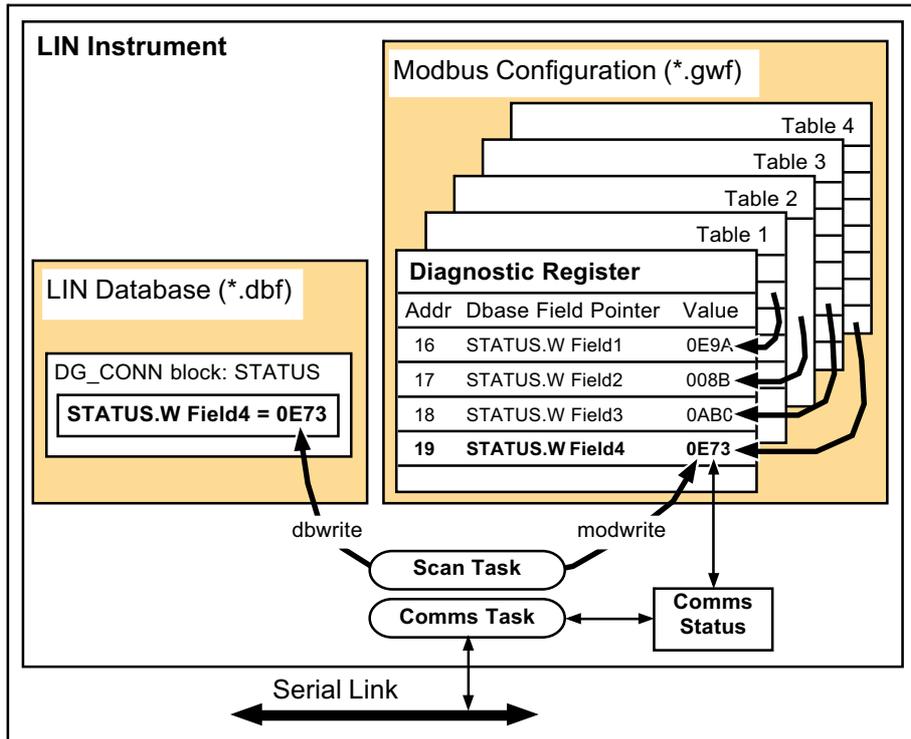


Figure 7 Modbus instrument database - diagnostic table schematic

Block parameters

Symbols used in this table are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Page	Page reference to Tables 1 - 16, 17 - 32, or 33 - 49	Integer	
Table1 to Table16	Table monitor and control bits	ABCD hex	
Online	TRUE if table accessed/answered within timeout	T/F	D
DisScan	TRUE stops master polling slave	T/F	
Single	TRUE lets sequence initialise single scan	T/F	
Complete	TRUE when master has completed scan of slave	T/F	
DisWr	TRUE stops master writing to slave	T/F	C
ForceWr	TRUE forces a write	T/F	
TickSlip	TRUE if scan is incomplete within defined tick rate	T/F	
CFG32Err	TRUE if inconsistent or 32-bit configuration error	T/F	
ScanErr1	Read error code	T/F	B
ScanErr2		T/F	
ScanErr3		T/F	
ScanErr4		T/F	
WrErr1	Write error code	T/F	A
WrErr2		T/F	
WrErr3		T/F	
WrErr4		T/F	
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
BadCon	TRUE if specified GW_CON block is not valid	T/F	
BadTask	TRUE if GwBlock field refers to block on different task	T/F	
BadPage	TRUE if Page is already used by a GW_TBL block	T/F	
TableOfl	TRUE if TableOfl subfield non-zero (i.e. any table offline)	T/F	
Unused		T/F	
Unused		T/F	
Combined	OR-ing of all Alarms bits	T/F	
GwBlock	GW_CON block controlling the GW configuration	Enum	
TableOfl	Otable offline/online status	ABCD hex	
Table1	TRUE if Table 1 offline	T/F	D
Table2	TRUE if Table 2 offline	T/F	
Table3	TRUE if Table 3 offline	T/F	
Table4	TRUE if Table 4 offline	T/F	
Table5	TRUE if Table 5 offline	T/F	C
Table6	TRUE if Table 6 offline	T/F	
Table7	TRUE if Table 7 offline	T/F	
Table8	TRUE if Table 8 offline	T/F	
Table9	TRUE if Table 9 offline	T/F	B
Table10	TRUE if Table 10 offline	T/F	
Table11	TRUE if Table 11 offline	T/F	
Table12	TRUE if Table 12 offline	T/F	
Table13	TRUE if Table 13 offline	T/F	A
Table14	TRUE if Table 14 offline	T/F	
Table15	TRUE if Table 15 offline	T/F	
Table16	TRUE if Table 16 offline	T/F	

Table 14 Block parameters

Block specification menu

The following is given in addition to *Table 14*.

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

Page. This field defines the group of 16 tables the block references. The *Alarm.BadPage* bit is set TRUE if the block attempts to reference a set of tables already used by another GW_TBL block.

NOTE Only one GW_TBL block can map to a single set of tables, but multiple GW_TBL blocks can be used to simultaneously monitor multiple tables. .

Page number	Diagnostic Table
0	1 to 16
1	17 to 32
2	33 to 48
3	49 to 64

Table 15 Diagnostic Tables References

Table1 to Table16. Set of 16 bitfields each containing monitor and control bits for the corresponding Modbus table. The use of these bits depends on whether the table is associated with a Master or Slave GW configuration, see *LINtools help*, Part no. RM263001U055, or the *MODBUS Tools Help*, Part no. HA028988, for detailed information on Modbus diagnostic tables.

GwBlock. This shows the GW_CON block that controls the GW configuration defined in the GW_CON block *FileName* field.

TableOfI. Bitfield with 16 bits, *Table1* to *Table16*, referring to the corresponding tables in the Modbus configuration. In slaves, this bit is set TRUE if the table has not been written to or read from in the period specified by the *Timeout* parameter in the Gateway configuration *Setup* menu or the GW_CON block *Timeout* field. In Masters, the bit sets TRUE if the slave does not respond to the scanning routines within *Timeout* seconds, i.e. it is effectively ‘offline’.

NOTE In Slaves the *TableOfI* bit is the inverse of the *Tablen* field’s *Online* bit. In masters the *TableOfI* bit can be set only if the Slave number is non-zero and the *Tablen* field’s *DisScan* bit is FALSE.

Each bit of this field can be wired into a TOT_CON block’s *UserAlrm* field (or any other suitable input) to indicate Modbus communications failures.

GWPROFM_CON: GATEWAY PROFIBUS MASTER CONFIG. BLOCK

Block function

This block displays information about a Profibus Master GateWay. Some fields in the block map directly to the appropriate field in the *Profibus Tools software*, see *Profibus Tools Help*, for detailed information.

It is designed for use by instruments operating as Master devices communicating via a Profibus Network that support multiple Profibus index GW, i.e. more than one GWF configuration running simultaneously. A single GWProfM_CON block must be created for each GWF configuration that is to be run, i.e. a single GWProfM_CON block per GateWay index. The block provides GateWay diagnostics and standard Profibus communication diagnostic parameters for a device at a defined Slave Address.

Block parameters

Symbols used in this table are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status*
FileName	8 character GWF filename	Alphanumeric	
Reload	TRUE halts, reloads, and restarts the .gwf file	T/F	
GWIndex	Shows index number of the active GW index	Integer	
MaxIndex	Max. permitted value of GWindex	Integer	
TableCnt	Quantity of Profibus tables used	Integer	
Port	Communications port in use	Alphanumeric	
Address	Address of Master node	Integer	
BaudRate	Comms frequency rate value	Integer	
MaxDev	Max. Slave devices supported	Integer	
ConfDev	Quantity of slave devices configured	Integer	
ActvDev	Quantity of communicating slave devices	Integer	
IPMemUse	Cyclic Input space used in DPRAM	%	
OPMemUse	Cyclic Output space used in DPRAM	%	
ScanRate	Time taken to update a single cycle	µS	
TblRate	Time taken to update all cyclic tables	µS	
DiagRate	Time taken to update all extended diagnostic tables	µS	
AcycRate	Time taken to update all acyclic tables	µS	
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
BadFile	TRUE if .gwf is corrupt, missing, or failed to load	T/F	
BadDBF	TRUE if .gwf is not associated with loaded .dbf file	T/F	
BadCfg	TRUE if Profibus Configuration error is detected	T/F	
ImgSize	TRUE if process data is too large	T/F	
HWError	TRUE if Profibus hardware error is detected	T/F	
ComsErr	TRUE if comms error in any slave is detected	T/F	
TooMany	TRUE if MaxIndex field limit is attained	T/F	
TableFlt	TRUE if internal fault in one or more tables	T/F	
BadPort	TRUE if port configuration and hardware conflict	T/F	
PendSave	TRUE if instrument and configuration differ	T/F	
MemFail	TRUE if I/O data has exceeded RAM limits	T/F	
Combined	OR-ing of all Alarms bits	T/F	
HWState	Current hardware condition	Enum	
ResetCnt	Failure counters, self-reset to zero	T/F	
ClpFail	Quantity of failed cyclic input data attempts	Integer	
COPFail	Quantity of failed cyclic output data attempts	Integer	
AlpFail	Quantity of failed acyclic read requests	Integer	
AOpFail	Quantity of failed acyclic write requests	Integer	
SlaveAdd	Address of inspected Slave device	Integer	
stdDiag1	Byte 1 of standard Profibus diagnostics	AB Hex	
NonExist	TRUE if slave failed to reply	T/F	
NotReady	TRUE if slave not ready for data transfer	T/F	
CfgFault	TRUE if slave detects a configuration fault	T/F	
ExtDiag	TRUE if valid extended diagnostics are available	T/F	
NotSupp	TRUE if slave does not support requested feature	T/F	
InSlvRes	TRUE if slave response is not compatible	T/F	
ParamFlt	TRUE if slave detects a parameter fault	T/F	

Continued...

Parameter	Function	Units	Status*
<i>Continued...</i>			
MstLock	TRUE if slave is communicating with another master	T/F	
stdDiag2	Byte 2 of standard Profibus diagnostics	AB Hex	<input type="checkbox"/>
ParamReq	TRUE if slave requires configuration	T/F	
StatDiag	TRUE if slave application not ready for data transfer	T/F	
DPSlave	TRUE if device is Profibus DP slave	T/F	
WdogOn	TRUE if slave has watchdog enabled	T/F	
FrzeMode	TRUE if slave freeze mode enabled	T/F	
SyncMode	TRUE if slave sync mode enabled	T/F	
Reserve6	Spare	T/F	
Deactive	TRUE if slave is deactive	T/F	
stdDiag3	Byte 3 of standard Profibus diagnostics	AB Hex	<input type="checkbox"/>
Reserve0	} Reserved	T/F	
to			
Reserve6			
ExtDiagOv	TRUE if Extended diagnostic data overflow	T/F	
MastAddr	Address of device that parameterised/configured inspected slave	Integer	
IdentNum	Profibus device type identifier of inspected device	Integer	
CmsErr1 to 8	Slave communications error bits, one bit per slave.	ABCD Hex	<input type="checkbox"/>

*Read-only unless otherwise stated.

Table 16 Block parameters

Block specification menu

The following is given in addition to *Table 16*.

Dbase, Block, Type. See *Appendix D page 544* for details of these 'header' fields.

FileName. The field containing the 8-character string base file name of the Profibus GateWay File, .gwf file, to be loaded.

NOTE The .gwf file specified here MUST have been created using the *Profibus Tools software*, see *Profibus Tools Help File*. It MUST also contain a reference to the currently loaded .dbf file. An invalid .dbf entry will set *Alarms.BadDBF* TRUE.

Reload. TRUE/FALSE. Setting this TRUE forces the active Profibus GW configuration to be temporarily suspended while the configuration specified in *FileName* is reloaded. All last know data values are retained, but are updated to reflect the new values when the reload is complete. It will auto-reset to FALSE once the reload has been initiated and can be used to perform a basic form of online reconfiguration, simply by replacing an entire configuration.

GWindex. Shows the index number defined by the active Profibus GW configuration. The number ranges from 1 to the maximum number of Profibus GW configurations supported by the configured product, see specific Instrument manual.

MaxIndex. Shows the maximum number of Profibus GW configurations supported by the configured product, see specific Instrument manual.

TableCnt. Shows the total number of Tables in the Profibus GW configuration. This corresponds to the number of entries in the *Profibus Tools software*, see *Profibus Tools Help file*.

Port. Shows the configured port name currently used by the instance shown in *GWindex*.

Address. Shows the configured Profibus Master node address.

BaudRate. Shows the frequency used by this device to transmit and receive Data bits.

NOTE Both transmitter and receiver must use the same baud rate value.

MaxDev. Shows the maximum number of devices supported by the Profibus Master, i.e. the total number of devices it can communicate with.

ConfDev. Shows the total number of slave devices currently configured.

ActvDev. Shows the total number of slave devices currently communicating.

IpMemUse. Shows the total amount of Cyclic Input space used for all devices.

OpMemUse. Shows the total amount of Cyclic Output space used for all devices.

ScanRate. Shows the total number of μS taken to complete a single update cycle.

TblRate. Shows the total number of μS taken to completely update all tables with the Cyclic process data from all active slave devices.

NOTE The number of currently active slave devices is shown in *ActvDev*.

DiagRate. Shows the total number of μS taken to completely update all extended diagnostic tables with the process data from all active slave devices.

AcycRate. Shows the total number of μS taken to completely update all acyclic tables to and from all active slave devices.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Asserted, if a sumcheck error in block's RAM data occurs.
- **BadFile.** Asserted, if the specified .gwf file is corrupt, missing or failed to load. The .gwf file will fail to load if the GW memory has been exhausted, and is considered corrupt if it was not created using the Windows compatible *Profibus Tools software*, see *Profibus Tools Help* file.
- **BadDBF.** Asserted, if the specified .gwf file is not associated with the currently loaded .dbf file.
- **BadCfg.** Asserted, if the specified .upb file of the Profibus card is corrupt or missing.
- **ImgSize.** Asserted, if the size of the file containing the process data is too large for one or more slaves.
- **HWError.** Asserted, if the process data is not transferred between the slave devices because of an error condition, as shown in *HWState*. This will be set TRUE if the Profibus Master is in a fatal error state, and is no longer exchanging data between any slave devices.

NOTE This error should never occur, but can be resolved by reloading the Profibus GW configuration, *Reload* set TRUE, or power cycling the instrument.

- **ComsErr.** Asserted, if process data cannot be exchanged because one or more slaves are currently in an error condition, derived from *ComsErr1* to *ComsErr8*.
- **TooMany.** Asserted, if all available Profibus GW configurations are already used.
- **TableFlt.** Asserted, if an internal fault is detected in one or more of the tables in this Profibus GW configuration, i.e. a table is configured incorrectly or contains invalid data.
- **BadPort.** Asserted, if the configured communications port fails to initialise correctly or the specified .gwf file is attempting to link to a physical port index that does not support the specified Profibus GW configuration. This is caused when the Port configuration in the Instrument does not correspond to the Port configuration in the Instrument Properties dialog.
- **PendSave.** Asserted, if the image in the instrument memory of this Profibus GW Facility has been modified, e.g. due to removal of invalid block references during Online Reconfiguration of Database file, since it was loaded from the .gwf file.

NOTE A save of the .gwf file must be performed to make the file match the contents of memory.

- **Memfail.** Asserted, if the Profibus Master of the instrument has exceeded its Dual Port RAM limits for input data, output data or both.
- **Combined.** Asserted, if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

HWState. (Inactive/Loading/Validating/Searching/Starting/Active/Stopping/Stopped/Error) Identifies the current status of the Profibus hardware. **Inactive** indicates the master is in the stop state, generally shown during power up. This parameter shows **Loading** while the configuration file is being saved to RAM memory, and changes to **Validating** when starting to ensure the loaded configuration file is operational. When the configuration file has loaded successfully, the instrument searches the network, *HWState* shows **Searching**, for the devices specified in the configuration file. This field can remain in this state for a number of seconds while each device is initiated. When the device communications start, *HWState* shows **Starting**. This parameter will show **Active** when configured devices, are actively communicating via the network. When communications with the devices are being closed down *HWState* shows **Stopping**. This parameter will remain in this state until all communications, including network maintenance have finished. If the Profibus Master fails to communicate with any slaves because a fatal error has occurred *HWState* shows **Error**.

ResetCnt. (TRUE/FALSE). Resets the *ClpFail*, *COpFail*, *AlpFail*, and *AOpFail* failure counters to 0 (zero).

ClpFail, COpFail. Shows the total number of failed cyclic data transfer attempts to read and write the process input and output data, respectively. Each parameter value increments each time the access is denied to the process input or output data areas, respectively of any devices, slaves.

NOTE The cyclic input and output tables support all data types and contains all the cyclic input and output data associated with a single device, slave.

AlpFail, AOpFail. Shows the total number of failed acyclic data transfer read and write attempts, respectively. Each parameter value increments each time the Profibus Master, or Slave, rejects an associated acyclic read and write attempt.

NOTE 1 The acyclic input and output table supports all data types and contains all the acyclic input and output data associated with all devices configured.

NOTE 2 Any reads of diagnostics information count as acyclic reads. This means that *AlpFail* will increment continually if *SlaveAdd* is set to an unconfigured slave address.

SlaveAdd. Identifies the device, slave, by address number, for standard Profibus diagnostics inspection. Applicable standard diagnostic relating to this device are shown in *stdDiag1*, *stdDiag2*, *stdDiag3*, *MastAddr*, and *IdentNum*. On later versions of software, this field defaults on startup to 255. This deliberately-high invalid address disables attempts to read the standard diagnostics, and thus avoids incrementing the *AlpFail* count. If *SlaveAdd* is set to the master address, this will also disable attempts to read the standard diagnostics.

stdDiag1. Provides 8 subfields used to show the current status of the device defined by *SlaveAdd*. This corresponds to Standard Status 1 of the standard Profibus diagnostics inspection, as shown in the table below.

Standard Diagnostic Bytes	Description
0	Standard Status 1
1	Standard Status 2
2	Standard Status 3
3	Profibus Master Address that configured Slave
4	Manufacturer Identification number - high byte
5	Manufacturer Identification number - low byte

Note Extended Diagnostic bytes are shown in the Profibus Master Configurator.

- **NonExist.** TRUE, if the device did not reply to the last telegram.
- **NotReady.** TRUE, if the device not prepared for data transfer.
- **CfgFault.** TRUE, if the device is signalling a configuration fault.
- **ExtDiag.** TRUE, if the device has valid extended diagnostics data available.
- **NotSupp.** TRUE, if the device does not support a feature requested by the master.
- **InSlvRes.** TRUE, if the device response is not DP, a Profibus protocol, compatible.
- **ParamFlt.** TRUE, if the device reports a parameterisation error.
- **MstLock.** TRUE, if the device is currently exchanging data with another master.

stdDiag2. Provides 8 subfields used to show the current status of the device defined by *SlaveAdd*. This corresponds to Standard Status 2 of the standard Profibus diagnostics inspection, as shown in the table below.

- **ParamReq.** TRUE, if the device requires configuration and Parameterisation.
- **StatDiag.** TRUE, if the device is signalling static diagnostics, or the application not ready for data transfer.
- **DPSlave.** TRUE, if the device is a Profibus DP slave.
- **WdogOn.** TRUE, if the watchdog feature is enabled, and will activate the slave specific watchdog action if the slave determines master-slave communications have failed.
- **Frzemode.** TRUE, if the device is currently operating in freeze mode.

■ **SyncMode.** TRUE, if the device is currently operating in synchronisation mode.

■ **Deactive.** TRUE, if the device is currently deactivated.

stdDiag3. Provides 8 subfields used to show the current status of the device defined by *SlaveAdd*. This corresponds to Standard Status 3 of the standard Profibus diagnostics inspection, as shown in the table below.

■ **ExDiagOv.** TRUE, if the device is currently experiencing too much extended diagnostic data.

MastAddr. Identifies the Profibus Master that configured the slave defined in *SlaveAdd*, by address number, corresponding to Byte 3 for standard Profibus diagnostics inspection, as shown in the table below.

IdentNum. Shows the Profibus device type identification number of the instrument at the location defined by *SlaveAdd*, corresponding to Byte 4 and Byte 5 for standard Profibus diagnostics inspection, as shown in the table below.

ComsErr1 to ComsErr8. Shows the status of the instrument at the address defined by each subfield, e.g. *ComsErr1* indicates communication errors for slaves at address 0 to address 15, *ComsErr2* indicates communication errors for slaves at address 16 to address 30, etc..

NOTE The subfield corresponding to the address of the master will always remain FALSE.

■ **Slave0 to Slave125.** TRUE, if the device indicated by the subfield is not online, or is in error, e.g. not communicating. FALSE indicates slave is online and communicating.

GWPROFS_CON: GATEWAY PROFIBUS SLAVE CONFIG. BLOCK

Block function

This block displays information about a Profibus Slave GateWay.

Specific Modbus Address registers **MUST** be configured correctly to ensure devices communicate effectively, see specific Instrument Handbook for details.

It is designed for use by instruments communicating via a Profibus protocol that have a Profibus card installed and are operating as Slave devices. A single GWProfs_CON block must be used for each supported instrument type fitted with a Profibus card.

Block parameters

Symbols used in this table are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status*
FileName	8 character GWF filename	Alphanumeric	
Reload	TRUE halts, reloads, and restarts the .gwf file	T/F	
GWIndex	Shows index number of the active GW index	Integer	
MaxIndex	Max. permitted value of GWIndex	Integer	
Port	Communications port in use	Alphanumeric	
SlveAddr	Address of this device	Integer	
InstChck	Instrument validity	Integer	
PriChck	Expected Profibus register config	Integer	
PriType	Profibus card used in Primary device	Integer	
PriVers	S/W version of Profibus card in Primary device	Hex	
PriStat	Slave status of Primary device	ABCD	
Ready	TRUE if waiting for response	T/F	<div style="display: flex; align-items: center;"> <div style="border-left: 1px solid black; padding-left: 5px; margin-right: 5px;"> 1 2 4 8 </div> D </div>
Online	TRUE if device is online	T/F	
SPC3Err	TRUE if SPC3 error is detected	T/F	
ROMCSErr	TRUE if ROMCS failure is detected	T/F	
MicroErr	TRUE if CPU failure is detected	T/F	<div style="display: flex; align-items: center;"> <div style="border-left: 1px solid black; padding-left: 5px; margin-right: 5px;"> 1 2 4 8 </div> C </div>
RedData	TRUE if redundancy data is required	T/F	
Timeout	Timeout value	Secs	
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
BadFile	TRUE if .gwf is corrupt, missing, or failed to load	T/F	
BadDBF	TRUE if .gwf is not associated with loaded .dbf file	T/F	
BadPort	TRUE if port configuration and hardware conflict	T/F	
TooMany	TRUE if MaxIndex field limit is attained	T/F	
TableOfl	TRUE if one or more tables are offline	T/F	
PendSave	TRUE if instrument and configuration differ	T/F	
MisMatch	TRUE if InstChck and PriChck conflict	T/F	
Config	TRUE if Profibus Configuration error is detected	T/F	
Combined	OR-ing of all Alarms bits	T/F	
ErrRspCt	Quantity of error messages sent by slave	Integer	
GoodRxCt	Quantity of acceptable messages received	Integer	
ScanPer	Repeat rate to check all Tables	µS	
Period	Repeat rate to check all Tables	µS	
used	Number of ticks taken to check all Tables	µS	
delay	Number of ticks before starting next ScanPer	µS	
TableCnt	Quantity of Profibus tables used		

Continued...

Parameter	Function	Units	Status*
<i>Continued...</i>			
InstStat	Instrument status	(AB)CD	
OnLine	TRUE if device is online	T/F	D
Primary	TRUE if this is Primary device	T/F	
RightPro	TRUE if this is CPU on the right of a redundant pair	T/F	
RedPair	TRUE if this is a Duplex device	T/F	
Synched	TRUE if CPU are synchronised	T/F	C
ConfMis	TRUE if invalid configuration is detected	T/F	
SecChck	Expected Profibus register config	Integer	
SecType	Profibus card in Secondary device	Enum	
SecVers	S/W version of Profibus card in Secondary device	Hex	
SecStat	Slave status of Secondary device	(AB)CD	
Ready	TRUE if waiting for response	T/F	D
Online	TRUE if device is online	T/F	
SPC3Err	TRUE if power supply error is detected	T/F	
ROMCSerr	TRUE if µROM sumcheck failure is detected	T/F	
MicroErr	TRUE if CPU failure is detected	T/F	C
RedData	TRUE if redundancy data is required	T/F	

*Read-only unless otherwise stated.

Table 17 Block parameters

Block specification menu

The following is given in addition to *Table 17*.

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

FileName. The field containing the 8-character string base file name of the Profibus GateWay File, .gwf file, to be loaded.

NOTE The .gwf file specified here MUST have been created using the *Modbus Tools software*, see *Modbus Tools Help File*. It MUST also contain a reference to the currently loaded .dbf file. An invalid entry will set *Alarms.BadDBF* TRUE.

Reload. TRUE/FALSE. Setting this TRUE forces the active Profibus GW configuration to be temporarily suspended while the configuration specified in *FileName* is reloaded. All last know data values are retained, but are updated to reflect the new values when the reload is complete. It will auto-reset to FALSE once the reload has been initiated and can be used to perform a basic form of online reconfiguration, simply by replacing an entire configuration.

GWindex. Shows the index number defined by the active Profibus GW configuration. The number ranges from 1 to the maximum number of Profibus GW configurations supported by the configured product, see specific Instrument manual.

MaxIndex. Shows the maximum number of Profibus GW configurations supported by the configured product, see specific Instrument manual.

Port. Shows the configured port name currently used by the instance shown in *GWindex*.

SlveAddr. Shows the configured Profibus Slave node address.

InstChck, PriChck, SecChck. Shows a numeric value calculated from the Cyclic Input data, and Cyclic Output data and the Demand Data, if used, to represent the configuration of the Primary slave and Secondary slave, respectively, of a redundant pair. When the configuration in the Primary and Secondary processors of a redundant pair is correct, the value shown in *PriChck* and *SecChck* should be identical.

NOTE InstChck, for future use.

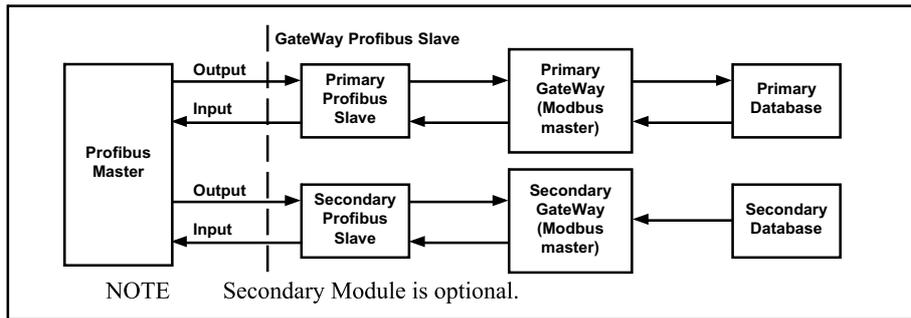


Table 18 Redudancy support schematic

PriType, SecType. Shows a numeric value representing the type of Profibus card fitted in the Primary Slave and Secondary Slave, respectively, of a redundant pair.

PriVers, SecVers. Shows the software version number for the type of Profibus card fitted in the Primary Slave and Secondary Slave, respectively, of a redundant pair.

PriStat, SecStat. Shows the current state of the Primary Slave and Secondary Slave, respectively of a redundant pair. If the device remains inactive for 2 seconds, this field automatically sets to 0 (zero).

- **Ready.** TRUE indicates the defined Slave is currently waiting for parameterisation from the Profibus Master device.
- **Online.** TRUE indicates the device is currently active in cyclic communications.
- **SPC3Err.** TRUE indicates the failure of the Profibus card because the Profibus ASIC test has failed.
- **ROMCSErr.** TRUE indicates the failure of the Profibus card because a microprocessor ROM checksum failure has occurred.
- **MicroErr.** TRUE indicates the failure of the Profibus card because the microprocessor fuse settings have been incorrectly configured.
- **RedData.** In a T2550/T2750 slave, if this bit is TRUE, the T2550/T2750 reports its redundancy status in Profibus register 1000, overwriting any real data that is configured to be in the register. A feature in the Eycon Profibus master exists to use this redundancy data, and this option is enabled through the Redundancy Data parameter in the User Parameter Data from the Profibus Master Configurator in LINTools.

Caution: *When RedData is enabled, Profibus register 1000 must not be used for process data as the value will be overwritten.*

Timeout. Shows the Timeout value of the Profibus GW configurations in seconds.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Asserted, if a sumcheck error in block’s RAM data occurs.
- **BadFile.** Asserted, if the specified .gwf file is corrupt, missing or failed to load. The .gwf file will fail to load if the GW memory has been exhausted, and is considered corrupt if it was not created using the Windows compatible *Modbus Tools software*, see *Modbus Tools Help* file.
- **BadDBF.** Asserted, if the specified .gwf file is not associated with the curenly loaded .dbf file.
- **BadPort.** Asserted, if the configured communications port fails to initialise correctly or the specified .gwf file is attempting to link to a physical port index that does not support the specified Profibus GW configuration. This is caused when the Port configuration in the Instrument does not correspond to the Port configuration in the Instrument Properties dialog.
- **TooMany.** Asserted, if all available Profibus GW configurations are already used, e.g. set TRUE if a second GWProfS_CON block is selected, but the instrument only supports 1 Profibus GW configuration, see specific Instrument handbook.

- **TableOffl.** TRUE/FALSE. TRUE if one or more of the tables in this Profibus GW configuration is 'offline'. This alarm indicates a Profibus communications error. This is set TRUE if the table has not been written to or read from in the period specified by the *Timeout* parameter, see *Modbus Tools Help*, possibly caused by a failure in the Profibus Card. Any such failure is indicated by *PriStat.SPC3Err*, *PriStat.ROMCSErr*, *PriStat.MicroErr* or *SecStat.SPC3Err*, *SecStat.ROMCSErr*, *SecStat.MicroErr*.
- **PendSave.** Asserted, if the image in the instrument memory of this Profibus GW Facility has been modified, e.g. due to removal of invalid block references, since it was loaded from the .gwf file.

NOTE A save of the .gwf file must be performed to make the file match the contents of memory.
- **MisMatch.** Not Supported. Asserted, if the values shown in *InstChck* and *PriChck* do not correspond.
- **Config.** Asserted, if a configuration error has been detected, i.e. *SlvAddr* is more than 125.
- **Combined.** Asserted, if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

ErrRspCt. The total number of error messages sent by this device, indicated by the address shown in *SlveAddr*.

GoodRxCt. Quantity of acceptable messages received by device, indicated by the address shown in *SlveAddr*.

ScanPer. Master communications scan time. Shows the number of time units taken to complete a scan (read) of each of the configured tables.

NOTE A fault, e.g. cable is disconnected or Slave device loses power, will cause a Master device to lose communications with a Slave device. Once the fault is fixed, Serial link communications can take up to a further 30 secs to be re-established. However, it can take up to 60 secs to re-establish communications.

Period. Table maintenance task. Shows the number of time units taken to update all data in the Profibus registers of all tables. This value depends on the number of parameters mapped to the Modbus address space, and the number of writes made from the Modbus tables in the Master device to function blocks cached within the Slave device. If data is written from the Profibus address space to the LIN Database, it is not updated from the LIN Database to the Profibus address space until the following scan.

NOTE Data can only be updated in one direction each scan, and only from the Profibus address space to the LIN Database if the value was changed by the Master device. This parameter is not affected if writing to local function blocks.

used. Table maintenance task. This is the total number of time units taken to check all Tables in the Profibus GW configuration.

delay. Table maintenance task. The number of time units between the completion of one scan of all the Tables and starting the next scan of all the Tables.

TableCnt. Shows the total number of Tables in the Profibus GW configuration. This value corresponds to the number of entries in the *Modbus Tools software*, see *Modbus Tools Help* file.

InstStat. Shows the current state of the Primary Slave and Secondary Slave, respectively of a redundant pair defined in *SlveAddr*.

- **Online.** TRUE indicates the device is currently active in cyclic communications.
- **Primary.** TRUE indicates the device is currently operating as the primary Slave of a redundant pair.
- **RightPro.** TRUE indicates the device is currently running the strategy via the processor on the right hand side of a redundant pair.
- **RedPair.** TRUE indicates the device is currently one of a redundant pair, operating in Duplex mode.
- **Synched.** TRUE indicates the strategy in both Primary Slave and Secondary Slave are synchronised, i.e. the strategies are the same.
- **ConfMis.** TRUE indicates a conflict between the .gsd file and the .gwf file configuration.

RAW_COM: RAW COMMUNICATIONS BLOCK

Block function

This block provides low level control of a serial communications port and also has the additional ability to execute structured text (ST) Actions. The block performs basic functions first and then executes any ST Actions that have been created. The ST is stored in a file (.STO) and will be dealt with in the same manner as for an 'Action' block but cannot access data outside the Raw Comms block. For protocols that are too complex to be handled using the ST inside the block an SFC should be used to drive the block.

It is designed for use by instruments supporting serial communication ports, e.g. T2550, Eycon 10/20 Visual Supervisor. Also see the Raw Comms User Guide, HA030511 for detailed information on applications and examples.

Block parameters

Symbols used in this table are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status*				
ActName	ST Action(s) defined in 'FileName' as below	Alphanumeric					
FileName	File name containing ST Action(s) defined in 'ActName'	Alphanumeric					
Tx_Value	Transmit Bytes Buffer (1020 max)	ByteSeq					
Tx_State	Transmitter status	Enum					
Tx_Trig	Initiates transmission when TRUE	T/F					
Options	Communication Options when TRUE	ABCD					
Alt_Term	Treat Rx termination bytes as alternatives	T/F	<table border="1"> <tr><td>1</td></tr> <tr><td>2</td></tr> <tr><td>4</td></tr> <tr><td>8</td></tr> </table> D	1	2	4	8
1							
2							
4							
8							
FlshOnTx	Rx Buffer flush, before every transmission	T/F					
DropRefl	Ignore reflected characters	T/F					
SlaveTx	Tri-state serial port when not transmitting	T/F					
Rx_Del	Apply 'Delete' character	T/F	<table border="1"> <tr><td>1</td></tr> <tr><td>2</td></tr> <tr><td>4</td></tr> <tr><td>8</td></tr> </table> C	1	2	4	8
1							
2							
4							
8							
Echo	Transmit all received characters	T/F					
LoopBack	All transmitted characters are also received	T/F					
TxMute	Inhibits transmission	T/F	<table border="1"> <tr><td>1</td></tr> <tr><td>2</td></tr> <tr><td>4</td></tr> <tr><td>8</td></tr> </table> B	1	2	4	8
1							
2							
4							
8							
RXMute	Discard received data	T/F					
Device	Identifies Comms Port associated with the block	Enum					
Baud	Tx/Rx Baud rate	Integer					
Parity	None, Odd or Even	Enum					
DataBits	Tx/Rx bits/character, 5, 6, 7 or 8	Enum					
StopBits	TX/Rx stop bits, 1 or 2	Enum					
Alarms							
Software	Block RAM data sumcheck error / network failure	T/F					
NoAction	.STO file or Action defined in 'ActName' not found	T/F					
BadActn	ST evaluation error at runtime	T/F					
BadDev	Invalid or unavailable device	T/F					
Device	Low level device failure, e.g. Parity error	T/F					
UserAlm1	Controlled via Structured Text	T/F					
UserAlm2	Controlled via Structured Text	T/F					
UserAlm3	Controlled via Structured Text	T/F					
UserAlm4	Controlled via Structured Text	T/F					
Combined	OR-ing of all Alarms bits	T/F					
Status	Communication Error conditions	ABCD					
RxChLost	Receive Buffer overflow	T/F	<table border="1"> <tr><td>1</td></tr> <tr><td>2</td></tr> <tr><td>4</td></tr> <tr><td>8</td></tr> </table> D	1	2	4	8
1							
2							
4							
8							
RxParity	Receive Parity error	T/F					
RxOver	Receive overrun error	T/F					
RxFrame	Receive framing error	T/F					
RxBreak	Receive break condition detected	T/F	<table border="1"> <tr><td>1</td></tr> <tr><td>2</td></tr> <tr><td>4</td></tr> <tr><td>8</td></tr> </table> C	1	2	4	8
1							
2							
4							
8							
RxFrcErr	Rx_State has been set to 'ERROR'	T/F					
TxChLost	Attempted transmission error	T/F					
TxFrcErr	Tx_State has been set to 'ERROR'	T/F					
Rx_Value	Receive Bytes Buffer (1020 max)	ByteSeq					
Rx_State	Receiver status	Enum					

Continued...

Parameter	Function	Units	Status*
<i>Continued...</i>			
Rx_Trig	Initiates reception when TRUE	T/F	🔍📖
Rx_Flush	Flush character buffer when TRUE	T/F	🔍📖
Rx_Max	Max characters received before passing to Rx_Value	Integer	🔍📖
Rx_Term	Character termination sequence definition	Bytes	📖
Rx_TermN	Number of characters read after Rx_Term	Integer	🔍📖
Rx_Del	Optional 'Delete' character	Byte	📖
Rx_DelEc	Transmission string when Rx_Del received	ByteSeq	
<i>Variables Page</i>			
Buffer1 to Buffer4	256 Byte buffer x 4	ByteSeq	📖
Byte1 to Byte4	Bitfield x 4, used as Integers or separate Booleans	AB Hex	🔍📖
Word1 to Word4	Bitfield x 4, used as Integers or separate Booleans	ABCD Hex	🔍📖
I1 to I4	32-bit signed integer variable x 4	Integer	🔍📖
DwnTmr1 to DwnTmr4	Down-timer x 4	Secs	🔍📖
A1 to A12	Floating point variable x 12	Float	🔍📖

*Read-only unless otherwise stated.

Table 19 Block parameters

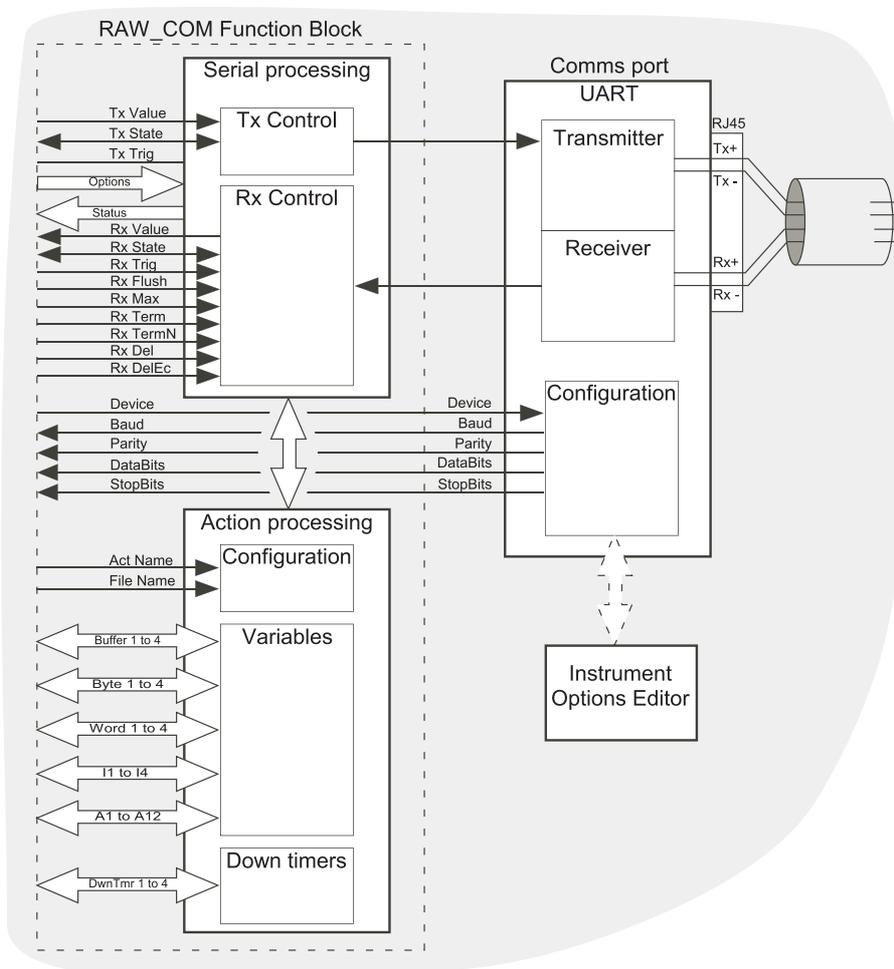


Figure 8 Functional Diagram

The above diagram depicts the functional relationship between the RAW_COM function block, communications port and associated UART. The communications port fields (e.g. Baud, Parity, etc) are configured using the Instrument Options Editor and are therefore 'Read only' from within the RAW_COM function block.

Block specification menu

The following is given in addition to *Table 19*.

Dbase, Block, Type. See *Appendix D page 544* for details of these 'header' fields.

ActName. The name (8-characters max) given to any Structured Text (ST) Action created for the block as defined in *FileName.STO* as specified below. The file holds the compiled structured text.

Filename. Specifies the file name (8-characters max) containing the Structured Text (ST) Action specified by the *ActName* parameter above.

Tx_Value. This is the buffer used to hold the character sequence to be transmitted. The buffer can contain up to 1020 characters.

Tx_State. This parameter indicates the current state of the transmitter. Values are *OK*, *PENDING*, *ERROR* and *WRITE*. Transmission can be initiated by setting this parameter to *WRITE*. The state will automatically change to *PENDING* until transmission is completed. On completion it changes to *OK*, or *ERROR* if an error occurred during transmission.

Tx_Trig. This is provided to allow control of the Raw Comms block by wiring. Setting this parameter to TRUE initiates transmission.

Options. Bitfield, setting communications options. All Options default to FALSE.

- **AltTerm.** When TRUE this allows *Rx_Term* below to be treated as a set of alternative single termination bytes any of which identifies the end of a line of input. When FALSE, *Rx_Term* is treated as a sequence of bytes.
- **FlshOnTx.** When TRUE, the receive buffer is always flushed immediately prior to any transmission.
- **DropRefl.** Inhibit character reflection if during transmission from the LIN device's serial port, characters are reflected back into it, e.g. 3-wire cabling. When TRUE for all characters transmitted, an equal amount of received characters are ignored.

NOTE: No check is performed to ensure that the ignored characters match the transmitted characters.

- **SlaveTx.** This field is for future use. It is intended to facilitate 'tri-stating' of the serial port transmitter for multi-drop slave applications when set TRUE. Current supported hardware always supports 'tri-stating' mode regardless of setting this field to TRUE or FALSE.
- **Rx_Del.** When TRUE, this enables the automatic processing of 'delete' characters in the input stream, i.e. the removal of the delete character itself plus the preceding character (assuming the latter has not already been processed and reached the *Rx_Value* buffer). The byte value to be interpreted as the 'delete' character is defined by the *Rx_Del* field below. If *Options.Echo* is also set, then the 'delete' character is not echoed, instead the optional sequence defined by the *Rx_DelEc* field below is substituted on the condition that a character was actually deleted.
- **Echo.** When TRUE, all received data (excluding loopback) is retransmitted, used for example with a 'dumb' terminal.

NOTE: The echoed data may be conditioned by the *Rx_DelEc* parameter below.

- **LoopBack.** When TRUE, all transmitted data (excluding echo) will appear as input. Used for test purposes only.
- **TxMute.** When TRUE, this inhibits transmission of all data (including echo), but has no other effect, i.e. the internal transmit processing continues to function as if the data has been transmitted.
- **RxMute.** When TRUE, discards incoming data (does NOT discard loopback data).

Device. This identifies the Comms port to which the block refers. For a T2550 it will always be RAW1. For Eycon 10/20 Visual Supervisor it may be RAW1 or RAW2, as there are 2 serial ports available.

Baud. The Tx/Rx Baud rate (independent Baud rates cannot be set for Tx and Rx). The value of this read only field is configured using the *Instrument Options Editor*. Supported Baud rates are 1200, 2400, 4800, 9600, 19200 and 38400.

Parity. The value of this read only field is configured using the *Instrument Options Editor*. Supported values are NONE, ODD and EVEN.

DataBits. The value of this read only field is configured using the *Instrument Options Editor*. It sets the number of bits per character for both receive and transmit. If this is set less than 8 then the most significant (8 - DataBits) bits will be ignored when sending and forced to zero when receiving. The number of bits per character supported in this field are 5, 6, 7 or 8.

NOTE. Current hardware does not support 5 or 6 Data Bits.

StopBits. The value of this read only field is configured using the *Instrument Options Editor*. It sets the number of stop bits expected by the receiver and sent by the transmitter. The number of stop bits supported are 1 or 2.

Alarms.

- **Software.** Asserted, if a sumcheck error in block's RAM data occurs or caching failure.
- **NoAction.** The Structured Text (ST) Action as defined in ActName above or Filename.STO cannot be found.
- **BadActn.** Set if an ST evaluation error occurs at runtime
- **BadDev.** The configured device is invalid, e.g. Requires configuration using the *Instrument Options Editor*. For Visual Supervisor only, can also mean a Raw Comms licensing failure.
- **Device.** A low-level communication device failure, e.g. Parity error. The status field provides details of the failure.
- **UserAlm1.** Controlled from structured text.
- **UserAlm2.** Controlled from structured text.
- **UserAlm3.** Controlled from structured text.
- **UserAlm4.** Controlled from structured text.
- **Combined.** True if any alarm is active. It adopts the same status message and priority number as the block's highest priority active alarm.

Status. Status bits indicate the following error conditions. 0 = no error has occurred

- **RxChLost.** An internal receive buffer has overflowed causing characters to be lost. This may also be caused by very heavily loaded applications and/or large comms packet sizes, especially if above 512 bytes.
- **RxParity.** A parity error was detected on a received character.
- **RxOver.** An overrun error was detected on a received character.
- **RxFrame.** A framing error was detected on a received character.
- **RxBreak.** A break condition has been detected on the receive line (not supported on current hardware).
- **RxFrcErr.** *Rx_State* below has been set to *ERROR* to force an error.
- **TxChLost.** An attempt has been made to send a new message before the last transmission has completed.
- **TxFrcErr.** *Tx_State* above has been set to *ERROR* to force an error.

Rx_Value. This is the character buffer used to hold the character sequence received from the serial port. The buffer can contain up to 1020 characters. It is volatile.

Rx_State. Current Receive status. Values are *OK*, *PENDING*, *ERROR*, *READ* and *FLUSH*. Reception can be initiated by setting this parameter to *READ*. The state will automatically change to *PENDING* until reception is completed. On completion it changes to *OK*, or *ERROR* if an error occurred during reception. Setting it to *FLUSH* clears any characters that have been received and are waiting to be copied into the *Rx_Value* buffer. Characters already in the *Rx_Value* buffer are not affected.

Rx_Trig. This is provided to allow control of the Raw Comms block by wiring. Setting this parameter TRUE initiates reception.

Rx_Flush. Similar to *Rx_Trig* except that it conditions the *Rx_State* to *FLUSH*.

Rx_Max. This specifies the maximum number of characters held in the internal receive buffer before being passed on to the block's *Rx_Value* buffer. The value should be in the range 1-1020. 1020 is the maximum length of the *Rx_Value* message.

NOTE. Characters are passed to the Rx_Value buffer either on the maximum number of characters as defined in Rx_Max being reached or a termination sequence of characters has been received as defined in Rx_Term.

Rx_Term. This specifies a termination sequence of characters that is used to identify the end of a line of input. If Rx_Term is left blank the input will be read from an internal receive buffer into the Rx_Value buffer until Rx_Max characters have been received. If Options.AltTerm is set, then these characters are treated as alternatives rather than a sequence.

Rx_TermN. If Rx_Term is not blank, this specifies an additional number of characters to be read after the termination sequence has been received. This is intended to simplify receiving a message which for example has a terminating sequence followed by a CRC or BCC. The limits for this field are 0 - 1020.

Rx_Del. Specifies an optional 'delete' character, to be applied to the incoming character stream if Options.Rx_Del is set. This would typically be used when communicating with a terminal of some kind.

Rx_DelEc. If Options.Echo and Options.Rx_Del are both set, this specifies the string to be transmitted whenever an actual deletion takes place as a result of the character Rx_Del being received. This is typically "\$08\$20\$08", i.e. back-space, space, back-space.

NOTE. If a character is not available for deletion, then the defined string in Rx_DelEc is not transmitted.

Variables. General Purpose Variables accessed by the associated Raw Comms structured text and/or from external SFCs. They can be used as workspace, to hold results and to accept input values.

Buffer1 to Buffer4. These provide access to four character buffers which can be used as workspace by the associated Raw Comms structured text or from an external SFC. Each buffer can contain up to 256 characters.

Byte1 to Byte4. Four bitfields, each of which can be treated as an 8-bit integer or 8 separate Booleans.

Word1 to Word4. Four bitfields, each of which can be treated as a 16-bit integer or 16 separate Booleans.

I1 to I4. Four 32-bit signed integer variables.

DwnTmr1 to DwnTmr4. Four down-timers which indicate seconds as floating point. When non-zero, they count down automatically (updating at every block update) until they reach zero. They can be written to and read at any time.

A1 to A12. Twelve floating point variables.

CHAPTER 4 CONDITION FUNCTION BLOCKS

The CONDITION category of Function Block Templates provides the control strategy with functions for signal conditioning, linearisation, filtering, etc.

INVERT: ANALOGUE INVERSION BLOCK

Block function

Please refer to the schematic in *Figure 9*. The INVERT block inverts the sense of an analogue input signal, offsets it by the quantity ($HR + LR$).

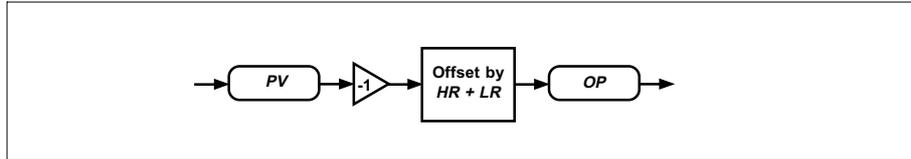


Figure 9 Block schematic

Block parameters

Symbols used in *Table 20* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
PV	Process Variable (Block Input)	Eng	▶◀
OP	Inverted Output value	Eng	◻▶
HR, LR	OP Offset (=HR+LR). High & Low Graphics Range (PV & OP)	Eng	▶◻
Alarms			◻▶ 📖 🔊
Software	Data Corruption/Communications Fault	T/F	
Combined	OR-ing of all Alarms bits	T/F	

Table 20 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

PV. Input value.

OP. Inverted and offset output value.

HR, LR. These parameters offset the inverted output: $OP=HR+LR-PV$. They also range graphic objects (Bars or Trends) linked to *PV* or *OP*; *HR* and *LR* define the 100% and 0% displays, respectively.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

CHAR: CHARACTERISATION BLOCK

Block function

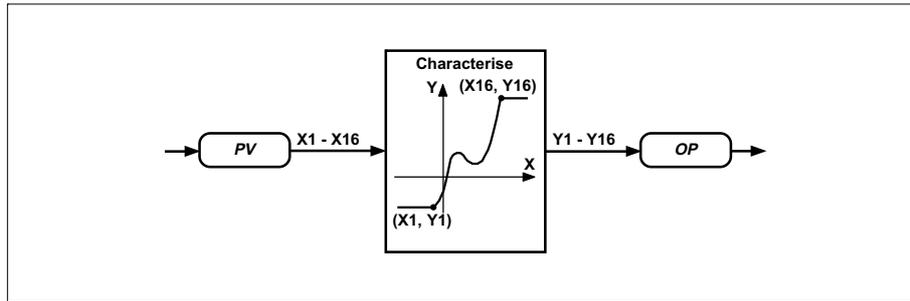


Figure 10 Block schematic

Please refer to the schematic in *Figure 10*. The CHAR (Characteriser) block provides a 4-quadrant analogue characteriser function with 16 breakpoints, entered as X (input) and Y (output) co-ordinate pairs. The X co-ordinates must increase monotonically, but the Y co-ordinates can have ascending, descending, or recurrent values. Any spare X co-ordinates are automatically filled up to the sixteenth point by copying the highest X-value entered in the table.

The characteristic curve is interpolated linearly between breakpoints. Inputs less than $X1$ or greater than $X16$ are output as $Y1$ or $Y16$, respectively (as suggested in the schematic).

Block parameters

Symbols used in *Table 21* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
PV	Process Variable (Block Input)	Eng	
OP	Block Output	Eng2	
X1 to X16	Input Values	Eng	
Alarms			
Software	Data Corruption/Communications Fault	T/F	
Combined	OR-ing of all Alarms bits	T/F	
HR_OP, LR_OP	High & Low Graphics Range (OP)	Eng2	
Y1 to Y16	Output Values	Eng2	

Table 21 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

PV. Input value for characterisation. If $PV < X1$ or $PV > X16$, the output OP is set equal to $Y1$ or $Y16$ respectively.

Linear interpolation is used between breakpoints. Linked graphic objects (Bar or Trend) are ranged by $X1$ and $X16$, which define the 0% and 100% PV displays, respectively.

OP. Output value after characterisation. Linked graphics are ranged by HR_OP (100% display) and LR_OP (0% display).

X1 to X16. X input co-ordinates, which must monotonically increase.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.

- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

HR_OP, LR_OP. High & Low OP ranging parameters for linked graphics.

Y1 to Y16. Y output co-ordinates, which have no value restrictions.

UCHAR: ANALOGUE INPUT CHARACTERISATION BLOCK

Block function

Unlike most other blocks, the UCHAR block is not intended for interconnection with other blocks in the database; instead it provides the data store for a 16 breakpoint characterisation which may be used by a single or multiple number of analogue input blocks. This is achieved by setting the 'Char' field to 'User' and by entering the UCHAR block name in the UserChar field of the analogue input block (e.g. ANIN, TCOUPLE, RTD, FULL_TC8, AN_IP, ANIN_6, RTD_6, AI_UIO).

The block supports 16 X (input) / Y (output) pairs expressed in engineering units where the X & Y values must be monotonic.

For some legacy products an older different version of this block is supported which uses normalised integer values - see Lin Blocks Reference Manual HA082375U003 issue 15 (Vintage).

NOTE. A UCHAR block used to linearise thermocouple measurements involving cold junction compensation should include a point at the cold junction temperature (e.g. 0°C, 0mV).

Block parameters

Symbols used in *Table 22* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
X0 to X15	Input values	Numeric	
Alarms			  
Software	Data corruption/communications fault	T/F	
Combined	OR-ing of all Alarms bits	T/F	
Y0 to Y15	Output values	Numeric	

Table 22 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these 'header' fields.

X0 to X15. X input co-ordinates, in engineering units which must monotonically increase.

NOTE. Values for X0 - X15 and Y0 - Y15 must be entered in the same units as specified in the corresponding Analogue Input block.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block's RAM data.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

Y0 to Y15. Y output co-ordinates corresponding to the X0 - X15 set, as described above.

FILTER: FILTER BLOCK

Block function

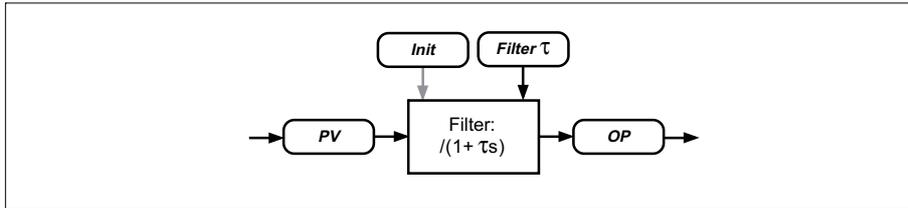


Figure 11 Block schematic

Please refer to the schematic in *Figure 11*. The FILTER block implements a first order filter function which can be expressed as:

$$OP = PV / (1 + \tau s)$$

where τ is the filter time constant and s the Laplace Operator $d(\)/dt$. In practice the algorithm recalculates OP every block sample time as:

$$OP_n = OP_{n-1} + \frac{T_s}{\tau} (PV_n - OP_{n-1})$$

where -

OP_n = OP at current iteration of algorithm

OP_{n-1} = OP at previous iteration of algorithm

T_s = Block sample time (seconds)

τ = Filter time constant (seconds). (See *Figure 12*)

PV_n = PV at current iteration of algorithm

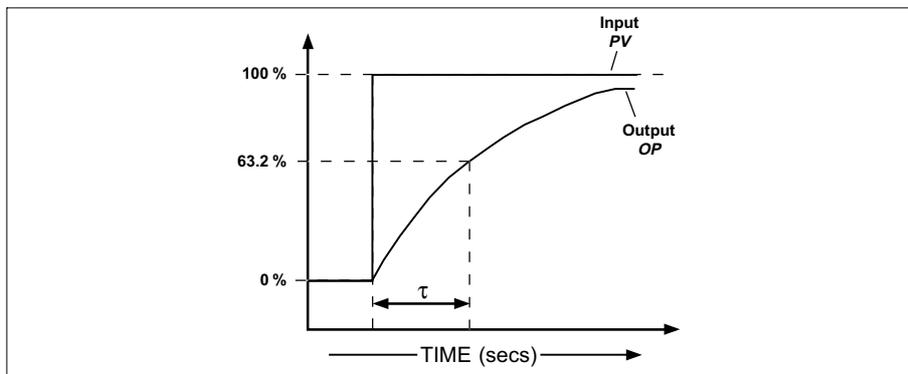


Figure 12 Definition of the FILTER block filter time constant

Block parameters

Symbols used in *Table 23* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
PV	Process Variable (Block Input)	Eng	☐
OP	Filtered Output	Eng	☐ ➔ 📖
HR, LR Alarms	High & Low Graphics Range (PV & OP)	Eng	☐ ➔ 📖 ? 🗣️
Software Combined	Data Corruption/Communications Fault	T/F	
	OR-ing of all Alarms bits	T/F	
Filter	Filter Time Constant	Secs	☐ ?
Init	Initialisation	T/F	☐ 🗒️ ?

Table 23 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these 'header' fields.

PV. Input value.

OP. Filtered output value.

HR, LR. High & Low Range for graphic objects (Bar, or Trend) linked to *PV* or *OP*. *HR* and *LR* define the 100% and 0% displays, respectively.

Filter. Filter time constant ($=\tau$ seconds). See *Figure 12* for definition.

Init. Initialisation. Initialises the filter by setting $OP = OP_{n-1} = PV$ when *Init* is TRUE. *Init* resets to FALSE after acting (write-only).

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block's RAM data.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

LEADLAG: LEAD/LAG BLOCK

NOTE. Though similarly named, the LEADLAG block described here is distinct from the LEAD_LAG block, described in the LEAD_LAG block.

Block function

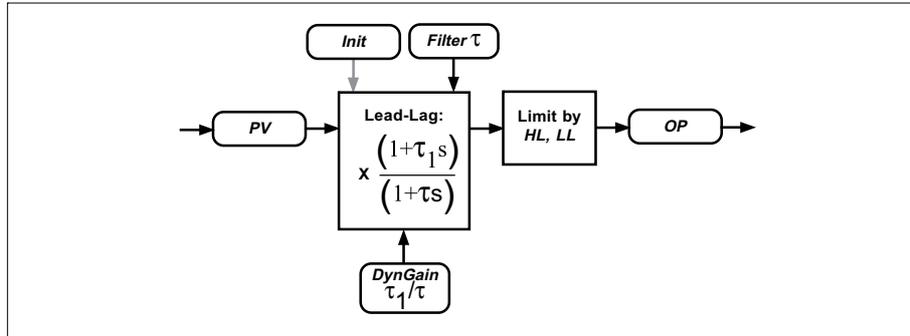


Figure 13 Block schematic

Please refer to the schematic in *Figure 13*. The Lead/Lag block allows the output to lead or lag the measurement in the frequency domain. The block function can be expressed as:

$$OP = PV \frac{(1 + \tau_1 s)}{(1 + \tau s)}$$

where τ_1 is the lead time constant, τ the filter (lag) time constant, and s the Laplace Operator $d(\)/dt$.

Dynamic gain (the *DynGain* parameter) is defined as τ_1/τ . Note that setting the dynamic gain to 0 converts the lead-lag function into a first order filter, and setting it to 1 makes *OP* track *PV*.

Block parameters

Symbols used in *Table 24* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

PV. Input value

OP. Lead-lagged output value.

Parameter	Function	Units	Status
PV	Process Variable (Block Input)	Eng	<input type="checkbox"/>
OP	Output value	Eng	<input type="checkbox"/>
HL, LL	High & Low Output Limit High & Low Graphics Range (PV & OP)	Eng	
Alarms			<input type="checkbox"/>
Software	Data Corruption/Communications Fault	T/F	
Combined	OR-ing of all Alarms bits	T/F	
Filter	Filter Time Constant	Secs	<input type="checkbox"/>
DynGain	Dynamic Gain	Numeric	<input type="checkbox"/>
Init	Initialisation	T/F	<input type="checkbox"/>

Table 24 Block parameters

HL, LL. High & Low output limits. These parameters also act as the High and Low range for graphic objects (Bar, Trend) linked to *PV* or *OP*. *HL* and *LL* define the 100% and 0% displays, respectively.

Filter. Filter (lag) time constant ($=\tau$ seconds).

DynGain. Dynamic Gain, defined as τ_1/τ .

Init. Initialisation. Initialises the block by setting $OP = OP_{n-1} = PV$ when *Init* is TRUE. *Init* resets to FALSE after acting (write-only).

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block's RAM data.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

AN_ALARM: ANALOGUE ALARM BLOCK

Block function

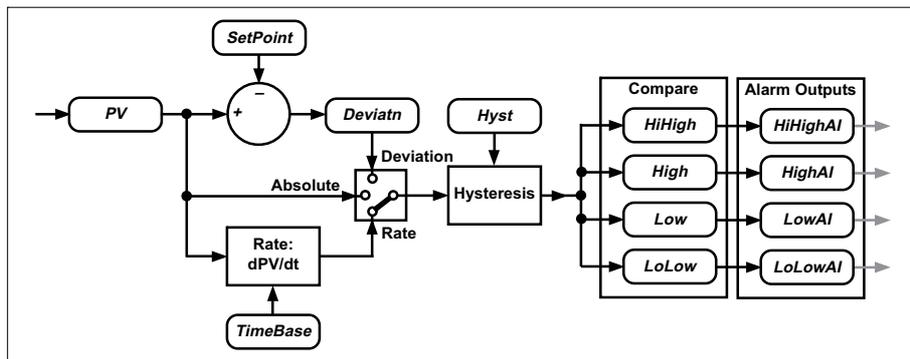


Figure 14 Block schematic

Please refer to the schematic in *Figure 14*. The Analogue Alarm block offers additional alarm capabilities to those built into some of the function blocks. The block can be set to flag either absolute, deviation, or rate alarms, with an adjustable hysteresis band and two independent pairs of alarm levels.

Block parameters

Symbols used in *Table 25* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

Type. (ABSOLUTE/DEVIATION/RATE). Selects type of alarm function.

PV. Input value.

HR, LR. High & Low range for graphic objects (Bar, Trend) linked to *PV*. *HR* and *LR* define the 100% and 0% displays, respectively.

TimeBase. (SECONDS/MINUTES). Rate alarms timebase.

SetPoint. Setpoint for deviation alarm function.

Hyst. Hysteresis bandwidth value, applied inside all alarm levels. *Hyst* is expressed in engineering units/second (or / minute) when applied to rate alarms.

Parameter	Function	Units	Status
Type	Alarm type select	Menu	
PV	Process variable (Block Input)	Eng	☐
HR, LR	PV High & Low Graphics Range	Eng	☐☐ 🔍?
TimeBase	Rate Alarms Timebase Select	Menu	
SetPoint	Setpoint for Deviation Alarms	Eng	☐☐
Hyst	Hysteresis Bandwidth	Eng	☐☐
HiHigh	High High Alarm Level	Eng	☐☐ 🔍?
High	High Alarm Level	Eng	☐☐ 🔍?
Low	Low Alarm Level	Eng	☐☐ 🔍?
LoLow	Low Low alarm Level	Eng	☐☐ 🔍?
Deviatn	Deviation (PV-SP)	Eng	☐☐ 📖
Alarms			☐☐ 📖 🔊
Software	Data Corruption/Communications Fault	T/F	
HiHighAI	High High Alarm	T/F	
HighAI	High Alarm	T/F	
LowAI	Low Alarm	T/F	
LoLowAI	Low Low Alarm	T/F	
Combined	OR-ing of all Alarms bits	T/F	
Disable	Alarms Disable	T/F	☐☐

Table 25 Block parameters

HiHigh, High, & Low, LoLow. High High, High, and Low Low, Low alarm levels. The action of these four multipurpose parameters depends on which type of alarm function is selected (via the *Type* parameter):

- **Absolute Alarms.** Alarms are set if the block input *PV* moves outside defined levels:

HiHighAl = TRUE when $PV > HiHigh$
HighAl = TRUE when $PV > High$
LowAl = TRUE when $PV < Low$
LoLowAl = TRUE when $PV < LoLow$

An alarm is not reset immediately *PV* returns to the alarm level - *PV* must be inside the level by a margin equal to the *Hyst* parameter before the alarm resets. This hysteresis permits clean transitions into and out of the alarm condition.

- **Deviation Alarms.** The high alarms are set when the positive deviation exceeds the defined levels. The low alarms are set when the negative deviation exceeds the levels:

HiHighAl = TRUE when $(PV - SetPoint) > HiHigh$
HighAl = TRUE when $(PV - SetPoint) > High$
LowAl = TRUE when $(SetPoint - PV) > Low$
LoLowAl = TRUE when $(SetPoint - PV) > LoLow$.

Hysteresis is applied to deviation values as it is to *PV* in absolute alarms.

- **Rate Alarms.** The high alarms are set when *PV*'s rate of increase exceeds the defined levels. The low alarms are set when *PV*'s rate exceeds the defined level and remains until the *PV*'s rate clears the defined level:

HiHighAl = TRUE when $dPV/dt > HiHigh$
HighAl = TRUE when $dPV/dt > High$
LowAl = TRUE when $-dPV/dt > Low$
LoLowAl = TRUE when $-dPV/dt > LoLow$.

The resolution of this function is dependent on the scanning time of *PV*.

Hysteresis is applied to changes in dPV/dt in a corresponding way to other alarms, but for rate alarms the *Hyst* parameter is expressed in engineering units *per second or minute* (defined by *TimeBase*).

Deviatn. Deviation (Error), defined as $PV - SetPoint$.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block's RAM data.
- **HiHighAl.** High High alarm.
- **HighAl.** High alarm.
- **LowAl.** Low alarm.
- **LoLowAl.** Low Low alarm.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

Disable. This digital input disables alarm annunciation. (Software alarm not affected.)

DIGALARM: DIGITAL ALARM BLOCK

Block function

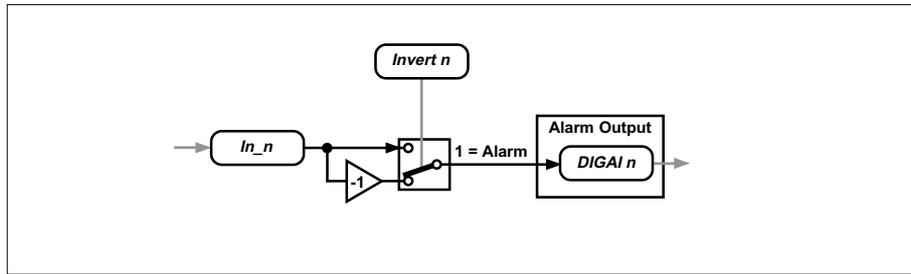


Figure 15 Block schematic

Please refer to the schematic in *Figure 15*, which shows only one (the *n*th) of the eight block channels. The Digital Alarm block allows logic signals within a control strategy to activate alarms. The alarm states of the eight triggering input signals can be individually set to TRUE or FALSE (via the *Invert* parameter), and can be collectively enabled or disabled.

Block parameters

Symbols used in *Table 26* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Invert	Alarm Input Invert States	CD hex	☛☞
Bit0	Channel 1 Field Input Inverted	T/F	D
Bit1	Channel 2 Field Input Inverted	T/F	
Bit2	Channel 3 Field Input Inverted	T/F	
Bit3	Channel 4 Field Input Inverted	T/F	
Bit4	Channel 5 Field Input Inverted	T/F	C
Bit5	Channel 6 Field Input Inverted	T/F	
Bit6	Channel 7 Field Input Inverted	T/F	
Bit7	Channel 8 Field Input Inverted	T/F	
In_1 to In_8	Digital Inputs from Control Strategy	T/F	☛☞
Alarms			☛☞ ☞☞ ☞☞☞☞
Software	Data Corruption/Communications Fault	T/F	
DigAl1 to DigAl8	Digital Alarm Output States	T/F	
Combined	OR-ing of all Alarms bits	T/F	
Disable	Disable DigAl1 to DigAl8	T/F	☛☞

Table 26 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

Invert. Bitfield defining the input states needed to trip alarms. When an *Invert* bit is FALSE (the default state) the corresponding alarm trips on a TRUE (high) input. When the *Invert* bit is TRUE, a FALSE (low) input trips the alarm.

In_1 to In_8. Show the actual states of eight input channels (which may be connected to the control strategy).

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.

- **DigAl1 to DigAl8.** Alarm status of each input. You can assign an 8-character (max.) name to each of these alarms by entering it into the corresponding *In_1* to *In_8* ‘units’ field at configuration time. This name appears at runtime in the message bar if the alarm is flagged, and also in the Alarm Summary page, and in printouts. The default alarm name (blank comments field) is ‘DigAln’.

- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

Disable. Disables annunciation and block outputs for the *DigAl1* to *DigAl8* alarms, and also *DigAl1* to *DigAl8* in the Combined Alarm output. Disable does not affect the Software Alarm or Combined Software alarm output.

LEAD_LAG: LEAD/LAG (FILTER) BLOCK

NOTE. Though similarly named, the LEAD_LAG block described here is distinct from the LEADLAG block.

Block function

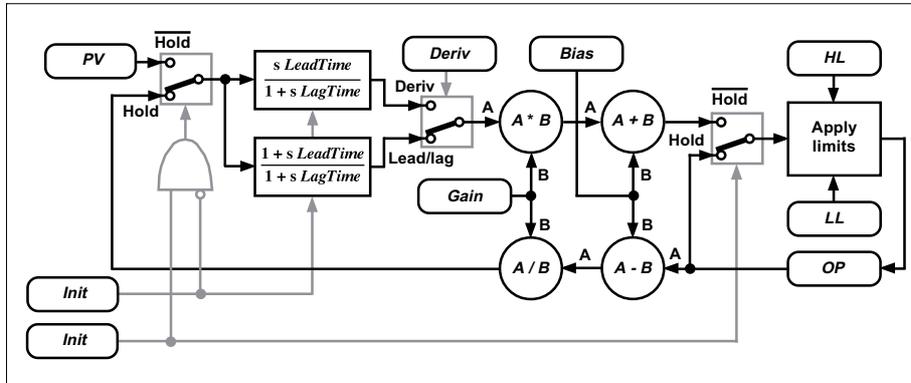


Figure 16 Block schematic

Please refer to the schematic in *Figure 16*. The LEAD_LAG block provides a lead/lag filter for use in feed-forward computations, or for setting up plant models.

Lead/lag operation. In lead/lag operation (*Options.Deriv* FALSE), as a conventional lead/lag filter, the block output *OP* and input *PV* are related by

$$L(OP) = Gain \left(\frac{1 + s \text{LeadTime}}{1 + s \text{LagTime}} \right) L(PV) + L(\text{Bias})$$

where

Gain	= overall filter gain
LeadTime, LagTime	= lead, lag time constants resp. (sec or min)
Bias	= bias added before output.
s	= Laplace Transform Operator d()/dt
L(variable)	= Laplace transform of variable

Filtered Derivative. With *Options.Deriv* TRUE, the DC gain becomes zero and *OP* is now a filtered derivative characteristic:

$$L(OP) = Gain \left(\frac{s \text{LeadTime}}{1 + s \text{LagTime}} \right) L(PV) + L(\text{Bias})$$

Block parameters

Symbols used in *Table 27* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
PV	Filter Input	Eng1	
OP	Filter Output	Eng2	
HL	Output high limit	Eng2	
LL	Output low limit	Eng2	
Gain	Filter Gain	Float	
Bias	Feedforward	Eng2	
TimeBase	Timebase select	Menu	
LeadTime	Lead Time Constant	Sec/Min	
LagTime	Lag Time Constant	Sec/Min	
Options			
Init	Initialises the filter	T/F	
Hold	Hold OP value	T/F	
Deriv	TRUE=Derivative, FALSE=Lead/lag	T/F	
Alarms			
Software	Block RAM data sumcheck error	T/F	
Combined	OR-ing of all Alarms bits	T/F	

Table 27 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

PV. Filter input.

OP. Filter output.

HL, LL. High and Low output limits. These parameters also act as the high and low range for graphic objects (Bar, Trend) linked to *PV* or *OP*. *HL* and *LL* define the 100% and 0% displays, respectively.

Gain. Filter gain, defined in the equations given in the *Block function* section above.

Bias. Feedforward. Specifies the output bias added in the filter equation before the result is passed to the filter output *OP*.

TimeBase. (Seconds/Minutes). Timebase select. Selects units for the two filter time constants, *LeadTime* and *LagTime*.

LeadTime, LagTime. Lead and Lag filter time constants, respectively, as defined in the equations given above.

Options.

- **Init.** Initialises the filter by setting $OP = OP_{n-1} = PV$ when *Init* is TRUE.
- **Hold.** Holds the value of *OP*. Note that *Hold* also back-calculates and initialises the filter to produce the required output, so that the removal of *Hold* is bumpless. If *Init* and *Hold* are both active, *Init* overrides *Hold* for initialising the filter.
- **Deriv.** TRUE selects a filtered derivative action, with zero DC gain; FALSE selects a conventional lead/lag filtering action.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

RANGE: RANGE BLOCK

Block function

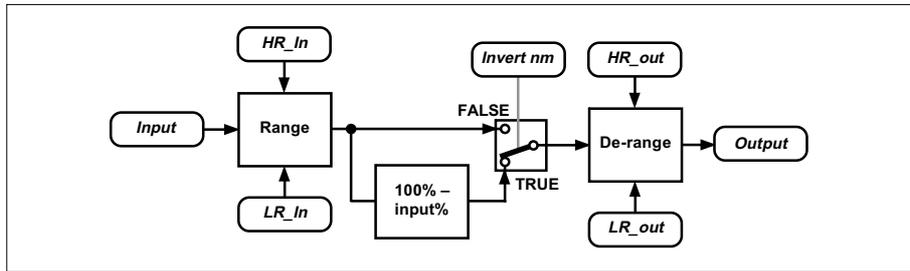


Figure 17 Block schematic

Please refer to the schematic in *Figure 17*. The RANGE block re-ranges an analogue input.

Block parameters

Symbols used in *Table 28* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Input	Input to be re-ranged	EngA	↔ ↔
HR_in	Input high range	EngA	↔ ↔ ?
LR_in	Input low range	EngA	↔ ↔ ?
Output	Re-ranged output	EngB	↔ ↔ ?
HR_out	Output high range	EngB	↔ ↔ ?
LR_out	Output low range	EngB	↔ ↔ ?
Options			↔ ↔
Invert	Inverts the output sense	T/F	↔ ↔
Alarms			∅ ∅ ∅ ∅
Software	Block RAM data sumcheck error	T/F	↔ ↔ ?
Combined	OR-ing of all Alarms bits	T/F	↔ ↔ ?

Table 28 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

Input. Block input.

HR_in, LR_in. Input high and low range, respectively. Define in engineering units the span of the block’s analogue input value. Note that *LR_in* may exceed *HR_in* if required.

Output. Re-ranged block output.

HR_out, LR_out. Output high and low range, respectively. Define in engineering units the span of the block’s analogue output value. Note that *LR_out* may exceed *HR_out* if required.

Options.

- **Invert.** Inverts the sense of the block output if TRUE. E.g. with *LR_out* = 0.000 units and *HR_out* = 100.000 units, an *Output* value of 30.000 units becomes 70.000 units when *Invert* is TRUE.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

FLOWCOMP: COMPENSATED FLOW BLOCK

Block function

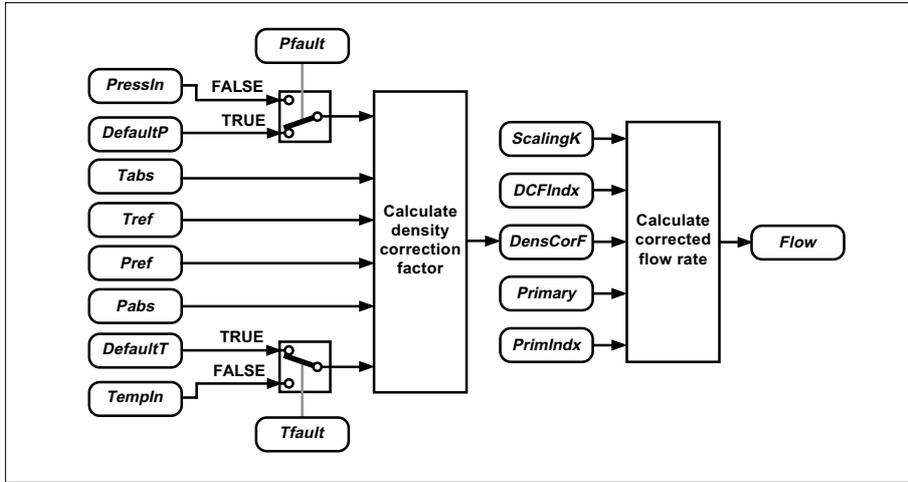


Figure 18 Block schematic

Please refer to the schematic in *Figure 18*. The FLOWCOMP block converts a raw gas flow measurement into a pressure- and temperature-corrected flow output signal.

The block calculates a density correction factor from pressure and temperature measurements, and uses this to compensate the raw flow measurement. User-set default pressure and temperature values are automatically substituted if digital inputs to the block indicate these measurements are faulty.

The density correction factor is calculated as

$$DensCorF = \frac{(Tref + Tabs) \times (\{PressIn\} OR DefaultP) + Pabs}{(TempIn OR DefaultT) + Tabs) \times (\{Pref\} + Pabs)}$$

and from this the fully compensated flow rate is calculated as

$$Flow = ScalingK \times Primary^{PrimIdx} \times DensCorF^{DCFIdx}$$

(Mnemonics are defined in *Table 29* and in the *Block specification menu* section.)

NOTE. *DensCorF* defaults to a value of 1 if the denominator of the *DensCorF* expression evaluates to 0.

Block parameters

Symbols used in *Table 29* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Primary	Raw flow or dp measurement	EngA	◻◻
PressIn	Pressure measurement	EngB	◻◻
TempIn	Temperature measurement	EngC	◻◻
DefaultP	Default pressure	EngB	◻◻ ?
DefaultT	Default temperature	EngC	◻◻ ?
Pref	Reference pressure	EngB	◻◻ ?
Tref	Reference temperature	EngC	◻◻ ?
Pabs	Absolute pressure		◻◻
Tabs	Absolute temperature		◻◻
ScalingK	Scaling factor	EngD	◻◻
DensCorF	Density correction factor		◻◻
Flow	Corrected flow	EngE	◻◻
Alarms			◻◻
Software	Block RAM data sumcheck error	T/F	◻◻
Combined	OR-ing of all Alarms bits	T/F	◻◻
Faults			◻◻
Pfault	PressIn input fault	T/F	
Tfault	TempIn input fault	T/F	
PrimIndx	Primary index	Menu	?
DCFIndx	Density correction factor index	Menu	?

Table 29 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

Primary. Raw flow or differential pressure (e.g. orifice plate) measurement.

PressIn. Pressure measurement of the flowing gas, used in the density correction factor expression if *Pfault* is FALSE. This is an excess pressure and must have the same units as *Pabs*.

TempIn. Temperature measurement of the flowing gas, used in the density correction factor expression if *Tfault* is FALSE. This must have the same units as *Tabs*.

DefaultP. Default pressure value, used in the density correction factor expression instead of *PressIn* if *Pfault* is TRUE. This must have the same units as *Pabs*.

DefaultT. Default temperature value, used in the density correction factor expression instead of *TempIn* if *Tfault* is TRUE. This must have the same units as *Tabs*.

Pref. Reference pressure, having the same units as *Pabs*. *Pref* is an excess pressure and is used in the density correction factor calculation to scale measured pressure values.

Tref. Reference temperature, having the same units as *Tabs*. *Tref* is used in the density correction factor calculation to scale measured temperature values.

Pabs. Absolute atmospheric pressure, relative to a vacuum. *Pabs* is used to convert excess pressure values to the absolute values required in the density correction factor calculation.

Tabs. The temperature on the absolute scale that is equivalent to zero degrees on the corresponding relative scale. E.g. 273.200 (K) equates to 0 (°C) approximately. *Tabs* is used to convert relative temperature values to the absolute values required in the density correction factor calculation.

ScalingK. A scaling factor used in the calculation of corrected flow.

DensCorF. Density correction factor, calculated as described in the *Block function* section above. Used to compensate the measured gas flow rate for variations in temperature and pressure from their reference values.

Flow. The scaled and corrected flow rate, derived from the raw measured flow rate after compensation for pressure and temperature variations.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block's RAM data.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

Faults.

- **Pfault.** A TRUE input to this parameter causes *DefaultP* to be substituted for *PressIn* in the density correction factor calculation.
- **Tfault.** A TRUE input to this parameter causes *DefaultT* to be substituted for *TempIn* in the density correction factor calculation.

PrimIdx. (1/2, 1, 3/2, 5/2) The power to which the raw flow measurement value *Primary* is raised in the corrected flow calculation.

DCFIdx. (0, 1/2, 1, -1/2, -1) The power to which the density correction factor *DensCorF* is raised in the corrected flow calculation.

GASCONC: NATURAL GAS CONCENTRATION DATA BLOCK

Block function

The GASCONC block contains the gas concentration values for use in calculations of compressibility and supercompressibility of natural gas mixtures - particularly the AGA8 (American Gas Association Report #8) calculations. The calculations themselves are carried out in an associated block - e.g. the AGA8DATA block, separately described.

The GASCONC block stores the concentrations of up to 21 gases, expressed in Mole percentage units. Fields are provided that report the total gas Mole% (*Total*), and allow a tolerance on this total to be specified (*TotTol*). An alarm is tripped (*BadTotal*) if the total percentage of the entered data differs from 100% by more than this tolerance. The block also checks the data against AGA8 Normal and Expanded range limits (Table 1 of the Report). Alarms trip (*NotNorm* and *OutRange* resp.) if any entry exceeds these limits.

NOTE. 'Normal' range limits define concentration limits for each gas within which the AGA8 calculations produce their most accurate results. 'Expanded' range limits define concentration limits beyond which the accuracy of the computation is not quoted in the Report, and use of these values is not recommended.

The block holds two levels of data - the visible data entered in the Mole% fields, and a (hidden) validated and normalised copy of this data that is actually being used by the associated AGA8DATA block(s) to perform the calculations. Normally these two sets of data are effectively identical, but they can differ if new values have been entered. A parameter - *GasData* - acts as a monitor/control interface between the two sets of data, alerting the operator if they differ, and letting him either put the new data into use, or recall the current in-use data to overwrite the visible block fields. The new total Mole% and range limits are checked whenever an update of the in-use data is requested via *GasData*. If the total Mole% is out of tolerance, the alarm trips and *GasData* reports 'Invalid'. In this case the new data is not allowed to update the current in-use data. Instead, the calculations continue using the unchanged in-use data until the operator enters valid data or recalls the in-use data to the block fields.

The *GasData* field can be configured to load a default set of data and automatically initiate calculations immediately if a cold start occurs.

NOTE. A single GASCONC block can act as the data source for many AGA8DATA blocks.

Block parameters

Symbols used in *Table 30* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Methane	Methane concentration	Mole%	
Nitrogen	Nitrogen concentration	Mole%	
CrbDiOx	Carbon dioxide concentration	Mole%	
Ethane	Ethane concentration	Mole%	
Propane	Propane concentration	Mole%	
Water	Water concentration	Mole%	
HSulphid	Hydrogen sulphide concentration	Mole%	
Hydrogen	Hydrogen concentration	Mole%	
CrbMonOx	Carbon monoxide concentration	Mole%	
Oxygen	Oxygen concentration	Mole%	
iButane	i-Butane concentration	Mole%	
nButane	n-Butane concentration	Mole%	
iPentane	i-Pentane concentration	Mole%	
nPentane	n-Pentane concentration	Mole%	
Alarms			
Software	Block RAM data sumcheck error	T/F	
BadTotal	Total \neq 100% by more than TotTol%	T/F	
NotNorm	Composition outside AGA8 Normal Range	T/F	
OutRange	Composition outside AGA8 Expanded Range	T/F	
Combined	OR-ing of all Alarms bits	T/F	
nHexane	n-Hexane concentration	Mole%	
nHeptane	n-Heptane concentration	Mole%	
nOctane	n-Octane concentration	Mole%	
nNonane	n-Nonane concentration	Mole%	
nDecane	n-Decane concentration	Mole%	
Helium	Helium concentration	Mole%	
Argon	Argon concentration	Mole%	
Total	Total of mole percentages	Mole%	
TotTol	Allowed tolerance in total %	Mole%	
GasData	Status of data entered/action required (Fig 4-11)	Menu	

Table 30 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

Methane to Argon. These 21 fields specify the concentrations of each gas in the mixture, in terms of Mole%, in the range 0-100%. Note that the values in these fields may not be the same as the (hidden) values actually being used by the associated AGA8DATA block performing the calculations. (See the *Block function* section above.) If they are identical — which is the normal running state for the block - the *GasData* parameter displays the legend ‘NoAction’.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.
- **BadTotal.** Trips if an update is selected (via *GasData*) and the total of all 21 gas Mole% values (*Total*) differs from 100% by more than the specified tolerance (*TotTol*).

The alarm clears only when an update is selected with a set of entered data having a total Mole% value that is within tolerance. While the alarm is asserted, the data set is not accepted and calculations continue using old values.

- **NotNorm.** Trips if an update is selected and at least one of the gas Mole% values is outside the AGA8 Normal Range limits. The alarm clears only when an update is selected with a set of entered values which are all within their Normal Range limits. Note that although *NotNorm* trips, the data is nevertheless accepted for use in the calculations. The operator must check the alarm and decide if the in-use data needs to be modified.
- **OutRange.** Trips if an update is selected and at least one of the gas concentration values is outside the AGA8 Expanded Range limits. The alarm clears only when an update is selected with all composition values inside their Expanded Range limits. Note that although *OutRange* trips, the data is accepted and the operator must decide if the in-use data needs to be modified.

- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

Total. Sum of all 21 entered Mole percentages in the *Methane* to *Argon* fields. The *BadTotal* alarm trips if this sum differs from 100.0% by more than the tolerance specified in the *TotTol* field.

NOTE. Despite the allowed tolerance in the *entered* data, the block automatically normalises the *in-use* data so that the Mole% figures actually used by the calculations always add up to exactly 100.0%.

TotTol. Specifies an allowed tolerance in the value of *Total*. (See NOTE above.)

GasData. (Waiting/NoAction/Changed/Invalid/Update/Recall). This field acts as an operator monitor/control interface between the visible data entered in the *Methane* to *Argon* fields, and the (hidden) validated copy of this data actually in use by the associated AGA8DATA block(s). (See the *Block function* section above). *Figure 19* shows the relationships between the items in the *GasData* menu. The items are as follows.

- **Waiting.** Indicates that there is no data in the hidden in-use data area, and therefore that calculations cannot be performed by any associated AGA8DATA blocks. The block is waiting for the operator to load data from the visible fields to the in-use area, via the ‘Update’ menu item (see below). This situation can arise after a cold start, but is avoided by saving the LIN database with default composition data entered, and the *GasData* parameter set to ‘Update’. Then, immediately after a cold start the default data automatically loads and calculations begin. Note that altering any data in the gas concentration fields switches *GasData*’s state from ‘Waiting’ to ‘Changed’, but calculations cannot start until valid data is copied to the in-use area.

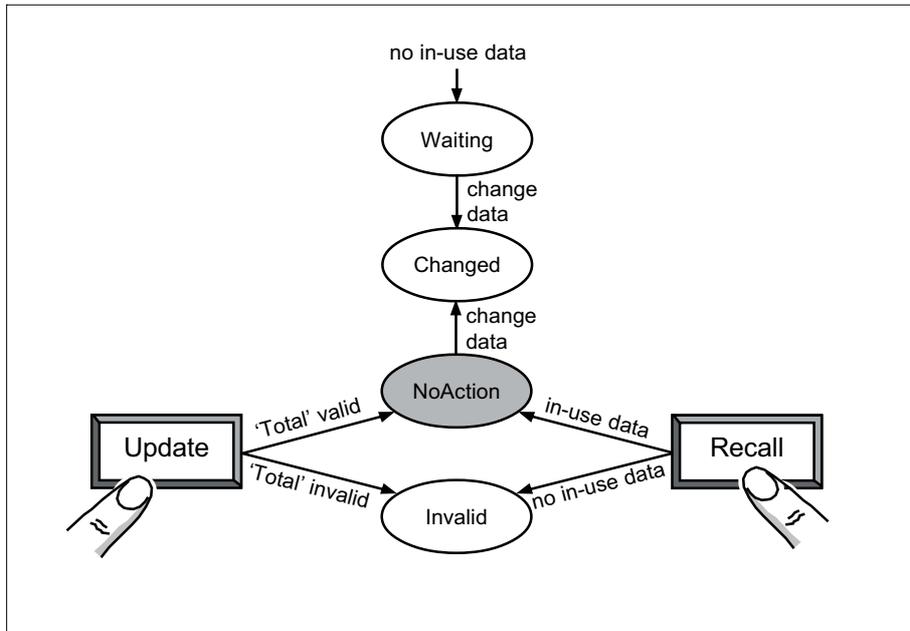


Figure 19 GasData parameter states

- **NoAction.** The normal running state of the block (shaded in *Figure 19*). Indicates that calculations are running and the visible entered composition data and the hidden in-use data are the same. No operator action is required.
- **Changed.** Indicates that the visible data entered in the Mole% fields is different from the data in the hidden in-use fields, on which the calculations are based. ‘Changed’ remains displayed until the operator takes further action.

Copying the in-use data back to the visible fields (via the ‘Recall’ command) restores the ‘Changed’ state to ‘NoAction’. Alternatively, the changed data may be loaded to the in-use area via the ‘Update’ command, see below. The state entered by *GasData* will then depend on whether or not the changed data proves to be valid (see next).

- **Invalid.** Indicates that the data the operator is attempting to load to the in-use area (via 'Update') has a *Total* value that differs from 100.0% by more than the specified tolerance *TotTol*. I.e. its Mole% values do not add up to 100%, and so cannot be used. Instead, the calculations continue using the existing (hidden) data, and the *BadTotal* alarm is raised. The state remains 'Invalid' until the operator enters a set of valid data and requests another 'Update'. Alternatively, 'Recall' may be used to restore the valid in-use data to the visible fields.
- **Update.** 'Update' can be selected at any time, whatever state is displayed in the *GasData* field. This command is used by the operator to overwrite the current (hidden) in-use data with new data in the visible block fields.

Entering 'Update' initiates a routine that tests each value in the block's gas concentration fields against the AGA8 Normal and Expanded Range limits. Alarms are raised if any values are out of range (*NotNorm* and *OutRange*) but this does not prevent the new data from being deemed valid. The routine also checks the new *Total* value against 100% (plus specified tolerance) to determine if the composition total is valid. If this test fails, the *BadTotal* alarm is raised, *GasData* adopts the 'Invalid' state, and the new data is rejected.

Otherwise, the valid (even if out of range) data is loaded to the in-use area and *GasData* reverts to 'NoAction', its normal running state. The operator must decide how any *NotNorm* and/or *OutRange* alarm(s) need responding to.

NOTE. To ensure that calculations start immediately after a cold start of the system, save the LIN database with default composition data entered and *GasData* set to 'Update'. See the 'Waiting' section above.

- **Recall.** 'Recall' can be selected at any time, whatever state is displayed in the *GasData* field. This command copies the current (hidden) in-use data to the visible block concentration fields. *GasData* then displays 'NoAction', the block's normal running state. Note that if there is no in-use data in the hidden area for any reason, e.g. after a cold start - selecting 'Recall' causes *GasData* to adopt the 'Invalid' state.

AGA8DATA: AGA8 CALCULATION BLOCK

Block function

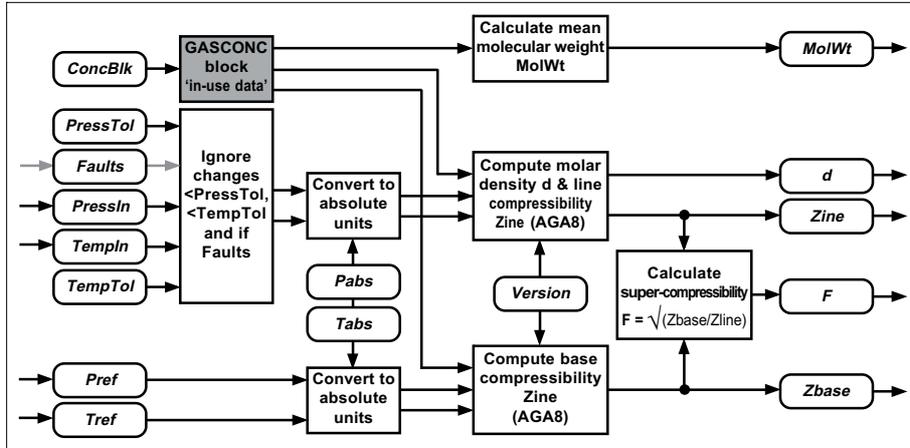


Figure 20 Block schematic

Please refer to the schematic in *Figure 20*. The AGA8DATA block uses the AGA8 (American Gas Association Report #8) set of equations to calculate the compressibility (*Zline*) and supercompressibility (*F*) of a natural gas mixture, from its measured line pressure (*PressIn*) and line temperature (*TempIn*). An associated GASCONC block, specified in the *ConcBlk* field, provides the mixture’s composition data. The block also calculates the mixture’s base compressibility (*Zbase*), molar density (*d*) and mean molecular weight (*MolWt*).

Digital inputs are available (via the *Faults* field) for connecting to I/O fail outputs indicating faulty pressure and/or temperature measurements. When set they cause the affected readings to be ignored and the calculations to use the previous (good) values.

Pressure and temperature tolerances (*PressTol*, *TempTol*) can be specified that reduce the computational overhead on the instrument. Changes smaller than these values are treated as ‘noise’ and ignored by the block, so preventing needless re-evaluation of the pressure/temperature-dependent parts of the AGA8 calculation.

Alarms are provided that trip if the associated GASCONC block cannot be accessed, if calculations have stopped, if the density calculation fails, or if the calculated compressibilities fall outside an acceptable range.

Accuracy. The AGA8DATA/GASCONC blocks have been tested for computational accuracy by inputting the full range of natural gas compositions, pressures, and temperatures given in *Tables A.3-1* and *A.5-2* in *Appendix A* of the *AGA8 Report*. The results produced agreed with those quoted in the Report tables to better than 0.001%, except for the ‘high-CO₂’ case, where the agreement was better than 0.0014%.

Update rates. Owing to the complexity of the algorithms involved, the AGA8DATA block does not update its calculated output values at every database iteration. Specifically, if pressure changes alone are triggering re-evaluations of the relevant parts of the calculation, the outputs are updated once every 20 database iterations, approximately. For temperature changes alone, the update rate is every 30 database iterations, and for gas composition changes (via the GASCONC block) the rate falls to once per 300 database iterations. Combinations of changes cause updating at the slowest of the relevant rates. E.g. a change in both line pressure and temperature causes the outputs to update once every 30 iterations; i.e. every 3 seconds at a typical database scan rate of 100ms.

NOTE The compressibility figures output by the AGA8DATA block can be used to normalise measured natural gas flows - compensating for deviations from the ideal gas laws - and so allow accurate flow measurement and fiscal metering.

Block parameters

Symbols used in *Table 31* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
ConcBck	Tagname of GASCONC block sourcing data	Alphanumeric	
PressIn	Input (line) pressure (relative units, e.g. BarG)	EngA	
TempIn	Input (line) temperature (relative units, e.g. °C)	EngB	
Pref	Reference pressure	EngA	
Tref	Reference temperature	EngB	
Pabs	Atmospheric pressure relative to a vacuum	EngA	
Tabs	Absolute temperature offset (e.g. 273.15°C)	EngB	
PressTol	Line pressure tolerance (measurement 'noise')	EngA	
TempTol	Line temperature tolerance (measurement 'noise')	EngB	
Alarms			
Software	Block RAM data sumcheck error	T/F	
ConcBck	GASCONC block cannot be found/accessed	T/F	
NoCalcs	Calculations halted (bad temp & press, or 'Waiting')	T/F	
Converge	Last molar density (d) calculation iteration failed	T/F	
ZorFOOR	Zline, Zbase, or F outside of range 0.5-1.5	T/F	
Combined	OR-ing of all Alarms bits	T/F	
Zbase	Compressibility of mixture at reference conditions		
Zline	Compressibility of mixture at line conditions		
F	Supercompressibility ($F = \frac{Z_{base}}{Z_{line}}$)		
d	Molar density of the mixture (mol/dm ³)		
MolWt	Mean molecular weight of the mixture		
Faults			
Pfault	PressIn input fault	T/F	
Tfault	TempIn input fault	T/F	
Version	Selects AGA8 version of algorithm (e.g. 1992)	Menu	$\emptyset \emptyset \emptyset \emptyset$

Table 31 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D* page 544 for details of these 'header' fields.

ConcBck. Specifies at configuration time the tagname (*Block* field) of the GASCONC block that is to be the source of the gas composition data for this calculation. *ConcBck* is not changeable at runtime. Note that several AGA8DATA blocks can be associated with a given GASCONC block. If the specified block does not exist, is cached, or cannot be accessed, the *ConcBck* alarm trips.

NOTE. The AGA8DATA and GASCONC blocks must both be local blocks in the same database. This means that the supervisory system must download to all units that require the same data.

PressIn. Line pressure measurement of the flowing gas, used in the AGA8 calculations. This is in relative units (e.g. BarG). A change in the value of *PressIn* (greater than *PressTol*) initiates a re-evaluation of the pressure-dependent parts of the calculation using the new value, provided that the pressure input is healthy (*Pfault* FALSE). If not, the bad reading is ignored and the last good pressure value remains in use.

TempIn. Line temperature measurement of the flowing gas, used in the AGA8 calculations. *TempIn* is in relative units (i.e. °C or °F). A change in the value of *TempIn* (greater than *TempTol*) initiates a re-evaluation of the temperature-dependent parts of the calculation using the new value, provided that the temperature input is healthy (*Tfault* FALSE). If not, the bad reading is ignored and the last good temperature value remains in use.

NOTE. The temperature units used must be consistent with the temperature system specified in the database's header block, via the *IP_type* parameter (i.e. Imperial = °F, SI = °C). But note that the selection of Absolute or Relative units via *IP_type* is immaterial for the AGA8DATA block.

Pref. Reference pressure, in the same units as *PressIn* (e.g. BarG). *Pref* specifies the reference (base) condition for the pressure measurement upon which the base compressibility is calculated.

Tref. Reference temperature, in the same units as *TempIn* (e.g. °C). *Tref* specifies the reference (base) condition for the temperature measurement upon which the base compressibility is calculated.

Pabs. Atmospheric pressure relative to a vacuum, in the same units as *PressIn* (e.g. 1.01325 BarG). *Pabs* is used to convert gauge pressure values to the absolute values required in the AGA8 calculations. The field's default value is as specified for SI units in the AGA8 report.

Tabs. The temperature on the absolute scale that is equivalent to zero degrees on the corresponding relative scale. E.g. 273.15 (K) equates to 0 (°C) approximately. *Tabs* is used to convert relative temperature values to the absolute values required in the AGA8 calculations. The field's default value is as specified for SI units in the AGA8 report.

PressTol. Specifies how much the pressure input to the block (*PressIn*) must change before the pressure-dependent parts of the AGA8 calculations are re-evaluated. This allows small changes in pressure (particularly those outside the accuracy of the transmitter) to be ignored and so reduce the computational overhead on the instrument. The pressure in use by the calculations remains at its last value until *PressIn* differs from the in-use value by more than *PressTol*. At this point - if the *Pfault* input is healthy - the current *PressIn* becomes the new in-use pressure.

NOTE 1. Setting *PressTol* to zero causes any change detected in pressure, however small, to trigger re-evaluation. A negative *PressTol* value forces the calculation to run continuously, slowing down the block update time.

NOTE 2. After a cold start, database download, or update of the data in the associated GASCONC block, the complete calculation is forced to run.

TempTol. Specifies how much the temperature input to the block (*TempIn*) must change before the temperature-dependent parts of the AGA8 calculations are re-evaluated. This allows small temperature changes to be ignored and reduces the computational overhead on the instrument. The in-use temperature holds until *TempIn* differs from it by more than *TempTol*. At this point - if the *Tfault* input is healthy - the current *TempIn* becomes the new in-use temperature. (Setting *TempTol* to zero or a negative value has effects corresponding to those for *PressTol*. See the NOTES in the previous section.)

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block's RAM data.
- **ConcBlck.** Trips if the block specified in the *ConcBlck* field cannot be found, is cached, or cannot be accessed.
- **NoCalcs.** Trips if the AGA8 calculations are not being performed because both the temperature and pressure inputs have failed (*Pfault* and *Tfault* TRUE), or because no valid composition data has been loaded to the block after a cold start (the associated GASCONC block's *GasData* field flags 'Waiting').
- **Converge.** Trips if the iteration process to compute *d* (molar density) fails. This alarm clears itself next time the iteration succeeds.
- **ZorFOOR.** Calculated compressibilities out of range. Trips if *Zline*, *Zbase*, or *F* fall outside the range 0.5 to 1.5. Note that the values are not limited, only alarmed.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

Zbase. Base gas compressibility. The gas compressibility under reference conditions of temperature (*Tref*) and pressure (*Pref*). This field is updated by the instrument whenever the concentration data, or *Tref* or *Pref*, change.

Zline. Line gas compressibility. The gas compressibility under line conditions of temperature (*TempIn*) and pressure (*PressIn*). This field is updated by the instrument as each calculation cycle completes.

F. Gas supercompressibility. $F = \sqrt{[Zbase/Zline]}$. This field is updated as each calculation cycle completes.

d. Molar gas density of the mixture. This field is updated as each calculation cycle completes. The units are mol/dm³. Note that the gas density in kg/m³ (equivalent to g/dm³) units is given by *d* x *MolWt*.

MolWt. Mean molecular weight of the gas mixture - a weighted mean value. This field is updated by the instrument whenever the data in the associated GASCONC block changes.

Faults. Input field connected from digital signals representing the health of the pressure and temperature I/O.

- **Pfault.** A TRUE input to this parameter, indicating a fault, causes the last (good) pressure reading to remain the in-use value for the AGA8 calculations.
- **Tfault.** A TRUE input to this parameter, indicating a fault, causes the last (good) temperature reading to remain the in-use value for the AGA8 calculations.

NOTE. If both *Pfault* and *Tfault* are TRUE, *Zline*, *Zbase*, *F* and *d* hold their last values and the *NoCalcs* alarm trips.

Version. (1992) Selects which version of the AGA8 report to use as the source of the update algorithm for the block. (*The 1992 version of the report, reprinted in July 1994, is the only one currently implemented.*)

NOTE. This block implements the *Detail Characterization Method*, as defined in the Report.

CARB_DIFF: CARBON DIFFUSION CALCULATION BLOCK

Block function

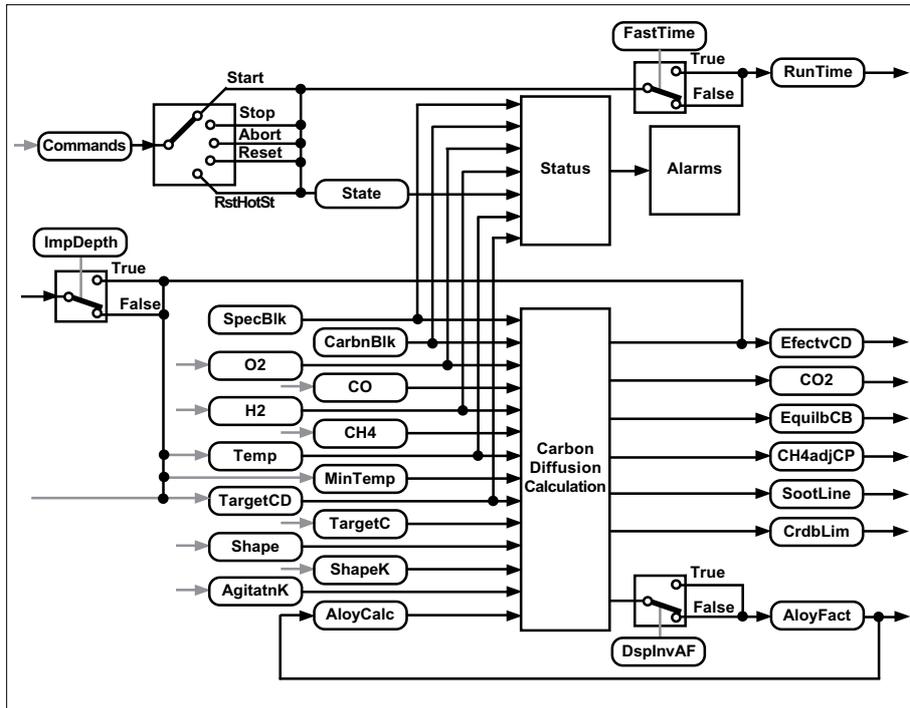


Figure 21 Block schematic

Please refer to *Figure 21*. The CARB_DIFF block is the main interface for ‘Live’ values. It provides the required control for real-time modelling of the Carbon Diffusion process used for metal surface hardening in the heat treatment industry. Values from this block can be used in the Carbon Diffusion calculation.

NOTE The STEEL_SPEC block is used to specify the initial elemental composition of the material.

Carbon Diffusion Model. The Carbon Diffusion Model uses two dynamic process inputs, Temperature and Oxygen. A Temperature input is used as expected, but the Oxygen input is used to provide increased accuracy in the ‘Carbon Profile’ value.

NOTE The Carbon Diffusion calculation may provide inaccurate values if the input parameters exceed the defined limits. When defined limits are exceeded, appropriate alarms are asserted.

Calculation HotStart. If the Carbon Diffusion calculation model is interrupted, i.e. a power failure occurs, the last set of input values are used to recover up to a maximum of 15 minutes of lost time, while the calculations were missed. When the database restarts, the last input values are applied until any lost time is regained allowing the calculations to continue to execute as normal. However, if the Carbon Diffusion calculation model is interrupted for more than 15 minutes, i.e. a fault exists, the database restarts showing *Status.BadHotSt* and *Alarms.BadHotSt* set TRUE, because 15 minutes of lost time has been regained, but further lost time remains. This means the calculations displayed do not correspond to the operation of the furnace. The calculations will continue, but remain behind the operation of the furnace by the amount of time exceeding the 15 minutes maximum, i.e. the Carbon Diffusion calculation model will be 10mins behind if the calculations stopped for 25mins.

Block parameters

Symbols used in this table are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
State	Calculation and results condition	Enum	
O2	Oxygen level in furnace	%	
CO	Carbon Monoxide level in furnace	%	
H2	Hydrogen level in furnace	%	
CH4	Methane level in furnace	%	
Temp	Measured Temp. in furnace	EngA	
MinTemp	Carbon diffusion start temp	EngA	
TargetCD	Target Case Depth	EngB	
TargetC	Target Carbon content	%C	
Shape	For future use - Material shape in furnace	Enum	
ShapeK	For future use - Material shape in furnace qualifier	Enum	
SpecBlk	Block name of material spec block		
CarbnBlk	Block name showing Carbon Profile results		
HrdnsBlk	Block name showing Hardness Profile results		
AgitatnK	Furnace agitation factor		
Alarms			
Software	Block RAM data sumcheck error	T/F	
Config	Licensing failure	T/F	
SpecBlk	Invalid specification block	T/F	
CarbnBlk	Invalid Carbon Profile results block	T/F	
HrdnsBlk	Invalid Hardness Profile results block	T/F	
Overload	Carbon Diffusion model calculations out of time	T/F	
AloyFact	Calculated Alloy factor/Material spec exceeded	T/F	
BadO2	Averaged Oxygen % input out of range	T/F	
BadH2	Hydrogen level range exceeded	T/F	
BadGas	Input gas + Model derived Gas exceeds 1	T/F	
BadTemp	Averaged temperature range exceeded	T/F	
Combined	OR-ing of all Alarms bits	T/F	
CO2	Calculated Carbon Dioxide level in furnace	%	
EquilbCP	Calculated Equilibrium Carbon Potential	%C	
CH4adjCP	EquilbCP adjustment for Methane existence	%C	
EfectvCD	Current diffusion depth at Carbon content level	EngB	
SootLine	Calculated soot deposit level	%C	
CrbdLim	Calculated Carbide limit	%C	
MaxDepth	Maximum depth to model	Enum	
AloyCalc	Alloy Factor calculation method	Enum	
AloyFact	Calculated alloy factor		
Commands	User requested calculation instructions	(AB)CD hex	
Start	Begin calculation	T/F - 1	D
Stop	Halt calculation on completion	T/F - 2	
Abort	Terminate the current calculation	T/F - 4	
Reset	Reset results data	T/F - 8	
RstHotSt	Reset hot start bits	T/F - 1	C
		- 2	
		- 4	
		- 8	
Status	Condition of calculation	ABCD hex	
NotReady	Calculation cannot proceed	T/F - 1	D
Ready	Calculation can proceed	T/F - 2	
WaitTrig	Calculation requested, awaiting MinTemp	T/F - 4	
Running	Calculation progressing	T/F - 8	
Done	Calculation completed successfully	T/F - 1	C
Aborted	Calculation terminated	T/F - 2	
		- 4	
		- 8	

Continued...

Parameter	Function	Units	Status
<i>Continued...</i>			
TargetCD	TargetC reached at TargetCD Value	T/F	B
HotStrt	HotStart successful	T/F	
BadHotSt	HotStart successful, maximum period exceeded	T/F	
BadO2	Averaged Oxygen % input out of range	T/F	
BadH2	Hydrogen level range out of range	T/F	A
BadGas	Input gas + Model derived Gas exceeds 1	T/F	
BadTemp	Averaged temperature range out of range	T/F	
Options		(ABC)D hex	→
ImpDepth	Measure material depth in Imperial units	T/F	D
FastTime	Carbon Diffusion curve demonstrator	T/F	
DsplnVAF	Display inverted Alloy Factor	T/F	
RunTime	Elapsed time in running state	Mins	
Version	Algorithm version	Enum	→

Table 32 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

State. (NotReady/Ready/WaitTrig/Running/Done/Aborted). Shows the current state of the calculations and results creating the Carbon Diffusion profile, see *Status*.

O2. The required percentage of Oxygen contained in the furnace, and applied in the Carbon Diffusion calculations.

CO, H2, CH4. The percentage of Carbon Monoxide (CO), Hydrogen (H2), or Methane (CH4) contained in the furnace, determined by the equipment connected to the furnace. It shows a constant required level or an externally applied default value until a Gas Analyser provides a valid reading.

Temp. The required temperature in the furnace, and applied in the Carbon Diffusion calculations. The units of this field are determined in the *IP_Type* field of the header block.

MinTemp. Specifies the minimum temperature required before the Carbon Diffusion model will start operating. The units of this field are determined in the *IP_Type* field of the header block.

NOTE The minimum temperature for Carbon Diffusion to occur is usually taken to be 820°C.

TargetCD. Specifies the Target Case Depth. This is the depth of the hardened outer coating of the material.

NOTE A Boost-Diffuse cycle can be achieved by changing this, and/or the *TargetC* value trigger points for the program or recipe.

TargetC. Specifies the Target Carbon content of the material.

NOTE A Boost-Diffuse cycle can be achieved by changing this, and/or the *TargetCD* value trigger points for the program or recipe.

Shape. For future use - (Flat/Concave/Convex). Specifies the shape of material in the furnace.

ShapeK. For future use - Specifies the dimensions relating to the value shown in *Shape*. When *Shape* shows **Flat**, this value defines the thickness of the material, but if *Shape* shows **Concave** or **Convex** this value shows the radius of the curve in the material.

SpecBlk. Specifies the Block name of a related specification block, i.e. STEEL_SPEC, used to define the composition of the material. The *Alarms.SpecBlk* is asserted if an invalid block name is entered.

CarbnBlk. Specifies the Block name of the AN_DATA block used to collect the data required to generate the Carbon Diffusion curve. The *Alarms.CarbnBlk* is asserted if an invalid block name is entered.

HrdnsBlk. For future use - Specifies the Block name of the related AN_DATA block used to collect the data required to generate the Hardness curve. The *Alarms.HrdnsBlk* is asserted if an invalid block name is entered.

AgitatnK. Specifies the constant value representing the level of agitation required. Generally, configured during commissioning of the furnace.

Alarms. See *Appendix D* page 545 for a general description of the Alarms field.

- **Software.** Sumcheck error in block's RAM data.
- **Config.** Asserted, if the block violates the license limits.
- **SpecBlk.** Asserted, if an invalid block name is entered in the *SpecBlk* field or the block does not exist.
- **CarbnBlk.** Asserted, if an invalid block name is entered in the *CarbnBlk* field or the block does not exist.
- **HrdnsBlk.** For future use - Asserted, if an invalid block name is entered in the *HrdnsBlk* field or the block does not exist.
- **OverLoad.** Asserted, if the calculations fail to complete within the internal time schedule causing the carbon diffusion model to become inaccurate. This could be due to extreme carbon diffusion parameters combined with an extremely heavily loaded application. However, the AN_DATA block will continue to show the results it receives. This can be reset by setting *Commands.RstHotSt* TRUE, or will be reset automatically when *Status.Done* or *Status.Aborted* show TRUE.
- **BadHotSt.** Asserted, if a HotStart occurred, i.e. after a power failure, and the last set of input values have been used to recover the maximum 15 minutes of lost time, but further lost time remains. This means the calculations displayed do not correspond to the operation of the furnace. The calculations will continue, but remain behind the operation of the furnace by the amount of time exceeding the 15 minutes maximum, i.e. the Carbon Diffusion calculation model will be 10mins behind if the calculations stopped for 25mins. This can be reset by setting *Commands.RstHotSt* TRUE, or will be reset automatically when *Status.Done* or *Status.Aborted* show TRUE. *Status.BadHotSt* also shows TRUE. If all lost time calculations, 15 minutes maximum, can be regained using the last set of input values, the calculations will continue to execute as normal.
- **AlloyFact.** TRUE, if the Alloy Factor exceeds the 0.5 to 1.5 range, or the sum of the elemental composition from the block defined by *SpecBlk*, excluding Carbon, exceeds 10%. This alarm automatically sets *AlloyFact* to 1.0, used for all further calculations.

NOTE This alarm can be suppressed by setting *AloyCalc* to '=1.0'.

- **BadO2.** Asserted, if the averaged O2% exceeds the range 1.0E-26 to 1.0E-16, set when *Status.BadO2* is TRUE while *State.Running* or *State.WaitTrig* is set. When TRUE, a clipped averaged O2% is used in the calculation.
- **BadH2.** Asserted, if the H2 level exceeds the range 1.5*CO to 2.5*CO, set when *Status.BadH2* is TRUE while *State.Running* or *State.WaitTrig* is set. When TRUE, a clipped value is used in the calculation.
- **BadGas.** Asserted, if the partial pressures sum of the input gasses and model-derived gasses exceeds '1' while *State.Running* or *State.WaitTrig*. This is used in conjunction with *Status.BadGas* to show the equilibrium compositions total more than 100%.
- **BadTemp.** Asserted, if the averaged temperature in the furnace exceeds the range 750°C to 1100°C, with an automatic 0.5°C hysteresis value, while *State.Running* is set.

NOTE The range 1382°F to 2012°F, with an automatic 0.9°F hysteresis value, can be used if *IP_Type.Imperial* set in the header block.

- **Combined.** Asserted, if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

CO2. Shows the calculated percentage level of Carbon Dioxide in the furnace.

EquilbCP. Specifies the calculated equilibrium Carbon Potential value.

CH4adjCP. Specifies the calculated Carbon Potential, adjusted to compensate for the Methane available in the furnace.

EfectvCD. Specifies the current effective case depth, i.e. the maximum depth at which the requested carbon content, *TargetC* has been achieved, using linear interpolation between elements in the model. If the target has not been met at any depth or if it has been met at the maximum modelled depth, this shows 0 (zero).

NOTE If the *TargetC* input is changed, this will be updated immediately.

SootLine. Specifies the calculated level of Carbon Potential that would cause a Soot deposit in the material. It is evaluated each time the calculation is triggered based on the current furnace conditions.

CarbdLim. Specifies the calculated level of Carbon Potential that would cause a Carbide deposit in the material. It is evaluated each time the calculation is triggered based on the current furnace conditions.

MaxDepth. (1 mm/2 mm/4 mm/6 mm/8 mm or 0.040"/0.080"/0.160"/0.240"/0.320") Specifies the maximum modelled depth of Carbon Diffusion in the material and describes the data presented in the associated AN_DATA block. This is dependant on *Options.ImpDepth* which defines which set of values are shown.

AloyCalc. (Standard/Sulfur/=1.0) Specifies the method used to calculate the Alloy Factor, shown in *AloyFact*. **Standard** shows the industry standard equation is used. **Sulfur** shows an equation including a term for the Sulfur element is calculated and used. **=1.0** shows a fixed value of 1.0 is used in the Alloy calculation.

AloyFact. Shows the metallurgical factor calculated using the method defined in *AloyCalc*. *Alarms.AloyFact* is asserted, if the value calculated exceeds the specified range. This value can be numerically inverted using *Options.DspInvAF*.

Commands. Subfield bits used to control the calculations, as displayed in *State*.

- **Start.** TRUE, specifies the user has requested the calculations to begin. Sets *Status.WaitTrig* TRUE, and is indicated when *State* shows WaitTrig.
- **Stop.** TRUE, specifies the user has requested the calculations have been completed. Sets *Status.Done* TRUE, and is indicated when *State* shows Done.
- **Abort.** TRUE, specifies the user has requested the currently active calculation to be terminated. Sets *Status.Aborted* TRUE, and is indicated when *State* shows Aborted.
- **Reset.** TRUE, specifies the user has requested the calculated results data must be set to 0 (zero). Sets *Status.Ready* TRUE, and is indicated when *State* shows Ready.

NOTE After the calculations have been stopped, *Commands.Stop*, TRUE, or aborted, *Commands.Abort*, TRUE, the calculated results data must be reset, *Commands.Reset*, TRUE.

- **RstHotSt.** TRUE, specifies the user has requested the HotStart must be reset, *Status.HotStrt* TRUE and/or *Status.BadHotSt* TRUE.

Status. Bitfield indicating general conditions.

- **NotReady.** If TRUE, no action can be taken because the configuration is invalid.
- **Ready.** If TRUE, the configuration is valid and is waiting for the Start request.
- **WaitTrig.** If TRUE, the configuration is valid and calculations will start when the value specified in *MinTemp* is achieved.
- **Running.** If TRUE, calculations will occur every minute, but can be stopped at any time. However, the calculations will never halt automatically.
- **Done.** If TRUE, calculations have been completed and results are frozen until this state is cleared.
- **Aborted.** If TRUE, calculations have been terminated and results are frozen until the state is cleared.
- **TargetCD.** If TRUE, the specified Carbon Potential, *TargetC*, has been achieved at a depth equal to or greater than the specified Target Case Depth, *TargetCD*. However, this field is updated immediately if the *TargetC* and/or *TargetCD* values are changed.

NOTE If the carbon profile is not decreasing, this can be set TRUE even though the carbon percentage at *TargetCD* is less than *TargetC*. However, it would have already achieved the *TargetCD* value.

- **HotStart.** If TRUE, a HotStart occurred while the calculations were running. This will reset automatically when entering the Ready state, *State* shows Ready. Alternatively, it can be reset using *Commands.RstHotSt*.
- **BadHotSt.** If TRUE, a HotStart occurred, i.e. after a power failure, and the last set of input values have been used to recover the maximum 15 minutes of lost time, but further lost time remains. This means the calculations displayed do not correspond to the operation of the furnace, see *Alarms.BadHotSt*.
- **BadO2.** If TRUE, the averaged O2% exceeds the range 1.0E-26 to 1.0E-16. Sets *Alarms.BadO2* TRUE while *State.Running* or *State.WaitTrig* is set. When TRUE, a clipped averaged O2% is used in the calculation.
- **BadH2.** If TRUE, the H2 level exceeds the range 1.5*CO to 2.5*CO. Sets *Alarms.BadH2* TRUE while *State.Running* or *State.WaitTrig* is set. When TRUE, a clipped value is used in the calculation.

- **BadGas.** If TRUE, the partial pressures sum of the input gasses and model-derived gasses exceeds '1' while *State.Running* or *State.WaitTrig* is set.
- **BadTemp.** If TRUE, the averaged temperature in the furnace exceeds the range 750°C to 1100°C, with an automatic 0.5°C hysteresis value, while *State.Running* is set.

NOTE The range 1382°F to 2012°F, with an automatic 0.9°F hysteresis value, can be used if *IP_Type.Imperial* set in the header block.

Options. Bitfield for optional configuration.

- **ImpDepth.** TRUE, causes the measurement values to appear as Imperial units, ignoring the *IP_Type* configuration in the header block. This allows SI units to be shown for temperatures, and Imperial units for depth measurements or vice-versa.

NOTE This parameter also effects the values displayed in the AN_DATA block configured in *CarbnBlk*.

- **FastTime.** TRUE, causes a Carbon Diffusion curve to be illustrated over an accelerated period. The actual time it would have taken to generate this is shown in *RunTime*.

NOTE Generally of use for product demonstration.

- **DsplnVAF.** TRUE, causes the Alloy Factor value to appear numerically inverted. This does not have an effect on the calculations.

RunTime. Shows the time elapsed while in the Running state. This is the real-time if *Options.FastTime* set FALSE, or the accelerated time period if *Options.FastTime* set TRUE.

Version. Shows the version of Carbon Diffusion algorithm used.

STEEL_SPEC: STEEL COMPOSITION BLOCK

Block function

This block is a list of parameters used to define the initial elemental composition of the steel that is being treated using the Carbon Diffusion process. Used in conjunction with the CARB_DIFF block via the *SpecBlk* field.

Block parameters

Symbols used in *Table 33* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
C	Quantity of Carbon in material	%	▶▶
Si	Quantity of Silicon in material	%	▶▶
Mn	Quantity of Manganese in material	%	▶▶
Ni	Quantity of Nickel in material	%	▶▶
Cr	Quantity of Chromium in material	%	▶▶
Mo	Quantity of Molybdenum in material	%	▶▶
Cu	Quantity of Copper in material	%	▶▶
V	Quantity of Vanadium in material	%	▶▶
Al	Quantity of Aluminium in material	%	▶▶
S	Quantity of Sulfur in material	%	▶▶
Alarms			▶▶  
Software	Block RAM data sumcheck error	T/F	
Combined	OR-ing of all Alarms bits	T/F	

Table 33 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

C. This shows the percentage of the carbon initially present in the material up to a maximum of 1%.

Si, Mn, Ni, Cr, Mo, Cu, V, Al, S. Each field shows the percentage of the element initially present in the material within a range 0.01% to 5.00%.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.
- **Combined.** TRUE, if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

ZIRCONIA: ZIRCONIA BLOCK

Block function

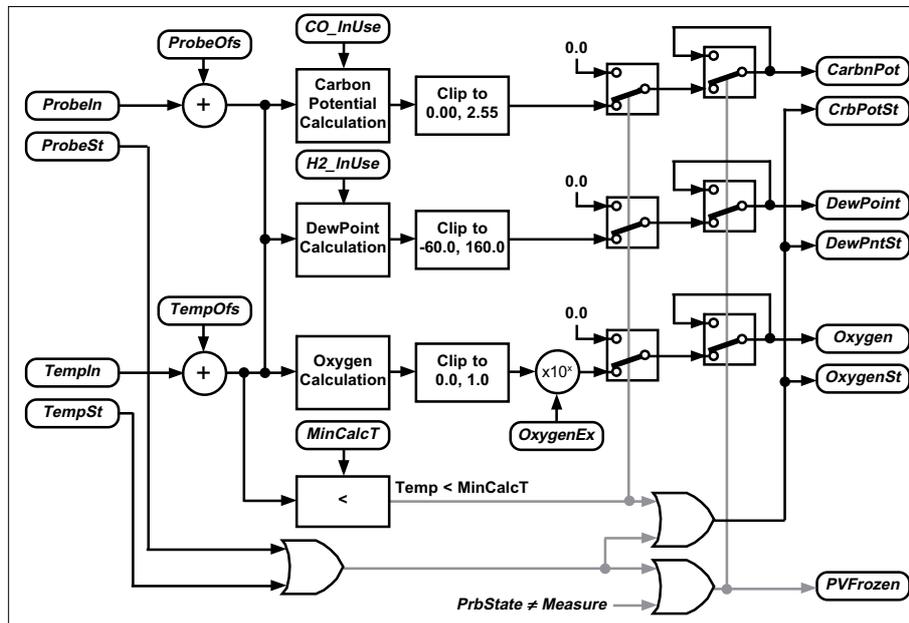


Figure 22 Block schematic

Please refer to the schematic in *Figure 22*. The Zirconia block calculates Oxygen, Carbon and Dew Point values derived from the Probe Temperature, the Probe mV and Remote Gas Reference values.

NOTE This block functions using a number of pages. Fields can be located using `<Page>.<Field>.<Subfield>` convention.

Probe Clean. Sensors require regular cleaning. Cleaning (Burn Off) is performed by forcing compressed air through the Probe and can be initiated either manually, *Probe Clean.ClnStart*, or automatically, *Probe Clean.ClnFreq*, using a timed period. During cleaning the PV output is frozen, *Main.PVFrozen*.

Health Alarm. After cleaning an alarm output, *Alarms.ProbeWrn*, is generated if the PV does not return to 95% of its value within a specified time. This indicates that the probe is deteriorating and should be replaced.

Soot Alarm. In addition to other alarms which may be detected, an alarm can be raised when the atmospheric conditions are such that carbon will be deposited as soot on all surfaces inside the furnace.

Block parameters

Symbols used in this table are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Main Page			
ProbeTyp	Connected Probe type	Enum	
OxygnTyp	Oxygen calculation algorithm used	Enum	
Probeln	Probe mV Input value	mV	
ProbeSt	Probe Input status	Enum	
ProbeOfs	Probe mV offset value	mV	
TempIn	Temperature Probe Input value	EngA	
TempSt	Temperature Probe Status	Enum	
TempOfs	Temperature Probe mV offset value	EngB	
MinCalcT	Minimum calculation temperature	EngA	
ProcFact	Probe compensation value (MMI Only)		
Alarms			
Software	Block data sumcheck error/network failure	T/F	
Probe	Probe Input status failure	T/F	
Temp	Temperature Input status failure	T/F	
CO_Remte	Remote CO gas status failure	T/F	
H2_Remte	Remote H2 gas status failure	T/F	
ClnRcWrn	Input failed to recover after clean sequence	T/F	
ClnAbort	Last probe clean sequence was aborted	T/F	
ClnTemp	Max temperature exceeded during clean sequence	T/F	
CantCln	Clean sequence failed to start on request	T/F	
ImpRcWrn	Input failed to recover after impedance measurement	T/F	
PrlmpHi	Impedance threshold exceeded	T/F	
ImpAbort	Last impedance measurement was aborted	T/F	
SootWrn	Conditions causing Sooting detected	T/F	
Combined	OR-ing of all Alarms bits	T/F	
PVFrozen	Hold outputs control	Menu	
PrbState	Current Probe operating state	Enum	
CarbnPot	Calculated Carbon Potential	%C	
CarbPotSt	Carbon Potential status	Enum	
aC_CO_O2	CO and Oxygen reaction value		
DewPoint	Calculated Dewpoint	EngB	
DewPntSt	Dewpoint status	Enum	
Oxygen	Calculated oxygen value	EngC	
OxygenSt	Oxygen status	Enum	
OxygenEx	Exponent units for 'Log Oxygen' type calculation		
Tolerance	Soot warning tolerance factor		
SootWrn	Sooting conditions detected		
Ballnt	Balance Integral control		
Probe Clean Page			
ClnEnabl	Permit probe clean	Enum	
ClnStart	Control probe clean sequence		
ClnAbort	Abort probe clean sequence		
ClnMsgRt	Reset cleaning related fields		
ClnFreq	Schedule for periodic probe cleaning	hh:mm:ss	
ClnTime	Duration of probe clean sequence	hh:mm:ss	
ClnMaxT	Maximum probe clean sequence temperature	EngA	
MnCnRcvT	Minimum recovery time after probe clean sequence	hh:mm:ss	
MxCnRcvT	Maximum recovery time after probe clean sequence	hh:mm:ss	
Alarms			
See Main Page			
TimToCln	Countdown to next clean	hh:mm:ss	
ClnRcovT	Recovery time after probe clean sequence	hh:mm:ss	
ClnRcvWn	Probe Input failed to recover after clean sequence		
LastClnmV	ProbemV on completion of last probe clean sequence	mV	
ClnValve	Demand for purge gas	Enum	

Continued...

Parameter	Function	Units	Status
<i>Continued...</i>			
Gas Ref Page			
CO_Ideal	Ideal CO gas reference value	%	
CO_Local	CO gas reference value	%	
CO_Remte	Remote CO gas reference value	%	
CO_RemSt	Remote CO gas reference status	Enum	
CO_RemEn	Remote CO gas reference enable	Enum	
CO_InUse	Working CO gas reference value	%	
H2_Local	H2 gas reference value	%	
H2_Remte	Remote H2 gas reference value	%	
H2_RemSt	Remote H2 gas reference status	Enum	
H2_RemEn	Remote H2 gas reference enable	Enum	
H2_InUse	Working H2 gas reference value	%	
Alarms			
See Main Page			
Impedance Page			
ImpStart	Start probe impedance measurement	Enum	
ImpAbort	Abort probe impedance measurement	Enum	
ImpMsgRt	Clear probe impedance measurement related fields	Enum	
MxImRcvT	Maximum probe impedance measuring recovery time	hh:mm:ss	
ImpTstR	Probe impedance test load resistance	kOhms	
MaxImp	Maximum probe impedance value	kOhms	
Alarms			
See Main Page			
Impednce	Measured probe impedance value	kOhms	
PbImphi	Probe impedance high limit exceeded	Menu	
ImpRcovT	Time taken to recover after impedance measurement	hh:mm:ss	
ImpRcWrn	Input failed to recover after impedance measurement	Enum	
ApplyRes	Demand for probe impedance test load	Enum	

Table 34 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

NOTE These fields automatically appear in each of the pages in this block.

Main Page

This page is used to define the operational mode of the loop.

ProbeTyp. (MMI/AACC/Drayton/Accucarb/SSI/MacDhui/Bosch/BarberC/Feronova/ProbeMV). Defines the type of probe/measurement calculations to be used. The Carbon Potential, DewPoint,aC_CO_O2, and Oxygen concentration values used by all types, excluding ProbeMV, are derived from the inputs to the block.

If **Feronova** is selected, the probe/measurement calculations will only be accurate within the working range, 700°C to 1100°C, 950mV to 1200mV, of an oxygen probe.

If **ProbeMV** is selected, *ProbeIn* is offset and transferred to the *Main.CarbnPot*, and DewPoint and Oxygen calculations are inhibited. The cleaning and impedance measurement features of this block continue to operate.

OxygenTyp. (Nernst/NernstBo/NernstCP/Feronova). Defines the oxygen calculation to be used. **Nernst** applies the standard Nernst equation used to calculate the oxygen concentration. **NernstBo** applies a Nernst equation specially used for a Bosch Probe, and the **NernstCP** uses the calculated *Main.CarbnPot* to back calculate the mV used in the Nernst equation, and uses the *Gas Refs.CO_Ideal* value. NernstCP can be used to obtain the required *Main.CarbnPot* value by changing *Gas Refs.CO_Local/Gas Refs.CO_Remote* without disrupting the oxygen calculation. **Feronova** is an alternative oxygen concentration calculation.

ProbeIn. The mV input from the probe subject to the *Main.ProbeOfs* before calculations.

ProbeSt. (Good/Bad). The status of the probe mV input, normally connected from the *Alarms.PVError* of any analogue input block. **Good** shows that no fault exists, **Bad** shows a failed probe input to the block, e.g. I/O module failure.

ProbeOfs. The offset for the probe in mV, applied to *Main.ProbeIn* before calculations.

TempIn. The measured temperature in the furnace subject to the *Main.TempOfs* before calculations. This is the measured temperature of the load under treatment, or the measured temperature of the Zirconia probe. If measured within the load, *Main.TempOfs* is not usually required.

NOTE During probe cleaning, the slow dynamic response of the load thermocouple inhibits the maximum temperature probe protection mechanism, *Probe Clean.ClnMaxT*. If temperature is measured within the Zirconia probe, *Main.TempOfs* can be used to provide a temperature more representative of the load.

During probe cleaning, the fast dynamic response of the probe thermocouple allows the maximum temperature probe protection mechanism, *Probe Clean.ClnMaxT*, to protect the probe if rapid exothermic temperature rises occur during Burn-Off.

NOTE Because the *Probe Clean.ClnMaxT* feature is intended to protect the probe, that *Probe Clean.ClnMaxT* is compared with the raw *Main.TempIn* without the *Main.TempOfs* being applied.

TempSt. (Good/Bad). Indicates the condition of the temperature input, normally connected from the *Alarms.PVError* of any analogue input block to indicate *TempIn* status. **Good** shows that no fault exists, **Bad** shows a failed temperature input, e.g. thermocouple failure.

TempOfs. The offset for Temperature in mV, applied to *Main.TempIn* before calculations.

MinCalcT. The minimum valid temperature in the furnace for calculations to begin or probe cleaning, and probe impedance to commence. If the temperature (*Main.TempIn* + *Main.TempOfs*) is below this value, all calculations, *Main.CarbnPot*, *Main.DewPoint* and *Main.Oxygen* are set to 0 (zero), the probe cleaning sequence and probe impedance measurement sequence are inhibited, and *Main.CrbPotSt*, *Main.DewPntSt* and *Main.OxygenSt* show **Bad**.

ProcFact. This is applicable to the MMI probe only. It is used to compensate for the varying abilities of some alloys to absorb carbon and clipped in the range 1 to 999.

Alarms. See *Appendix D page 545* for a general description of the Alarms field. This field appears on each page of the block.

- **Software.** Sumcheck error in block's RAM data.
- **Probe.** Asserted if a Probe input failure has occurred, *Main.ProbeSt* shows **Bad**.
- **Temp.** Asserted if the furnace Temperature input failure has occurred, *Main.TempSt* shows **Bad**.
- **CO_Remte.** Asserted if the remote CO (Carbon Monoxide) gas status has failed, *Gas Ref.CO_RemSt* shows **Bad**.
- **H2_Remte.** Asserted if the remote H2 (Hydrogen) gas status has failed, *Gas Ref.H2_RemSt* shows **Bad**.
- **ClnRcvWn.** Asserted if the *Main.ProbeIn* mV did not recover to 95% of its value prior to cleaning, within the stated maximum recovery time, *Probe Clean.MxCnRcvT*. This indicates the probe is deteriorating and should be replaced. Returns FALSE after the next successful probe clean sequence, or when *Probe Clean.ClnMsgRt* set **Yes**.
- **ClnAbort.** Asserted if the last probe clean sequence was aborted. Returns FALSE after the next successful probe clean sequence, or when *Probe Clean.ClnMsgRt* set **Yes**.
- **ClnTemp.** Asserted if the last probe clean sequence was aborted because the temperature exceeded the value in *Probe Clean.ClnMaxT*. Returns FALSE after the next successful probe clean sequence, or when *Probe Clean.ClnMsgRt* set **Yes**.
- **CantCln.** Asserted if a probe clean is requested when the *Main.PrbState* is **Not Ready**. This may be caused by either the user requesting a probe clean by setting *Probe Clean.ClnStart* set **Yes**, or by an automatically time scheduled clean request.

If a clean is requested when *Main.PrbState* is **Not Ready**, *Probe Clean.ClnStart* remains set at **Yes**. If *Probe Clean.ClnStart* is set **No** or, *Probe Clean.ClnAbort* is set **Yes**, this alarm is set FALSE. If *Probe Clean.ClnStart* is set **Yes** when the condition causing *Main.PrbState* to show **Not Ready** is cleared, a probe clean will being immediately.

- **ImpRcvWn.** Asserted if *Main.ProbeIn* mV did not recover to 99% of its value prior to the probe impedance measurement, within the stated maximum recovery time, *Impedance.MxCnRcvT*. Returns FALSE after the next successful impedance measurement sequence, or when *Impedance.ImpMsgRt* set **Yes**.
- **PrbImpHi.** Derived from *Impedance.PrbImpHi*. Asserted if the measured probe impedance exceeds the value defined in *Impedance.MaxImp*.
- **ImpAbort.** Asserted if the last probe impedance measurement sequence was aborted. Returns FALSE after the next successful probe impedance measurement sequence, or when *Impedance.ImpMsgRt* set **Yes**.
- **SootWrn.** Asserted if the *Main.SootWrn* shows **Yes** indicating the Probe has detected atmospheric conditions that may cause a deposit of soot on surfaces in the furnace.
- **Combined.** Asserted if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

PVFrozen. (No/Yes). Activated automatically during each probe clean sequence or impedance measurement indicating the *Main.CarbnPot*, *Main.DewPoint*, *Main.aC_CO_O2* and *Main.Oxygen* outputs are currently held. This can be wired from to hold carbon atmosphere control while probe cleaning sequence or probe impedance measurement is in progress.

PrbState. (Measure/Clean/ClnRecov/TestImp/ImpRecov/NotReady). Shows the current state of the calculations, cleaning, and impedance measurement.

NotReady indicates the normal state of the probe cleaning sequence if the furnace is not in use. This state occurs if the temperature (*TempIn* + *TempOfs*) used in the furnace is below the configurable minimum temperature, *Main.MinCalcT* or *Main.ProbeSt* or *Main.TempSt* show **Bad**. If the probe clean sequence or impedance measurement is in progress when the temperature falls below the value defined in *Main.MinCalcT* or an input fails the probe clean sequence and impedance measurement sequence will abort and, after the appropriate recovery time *Main.PrbState* shows **NotReady**.

Measure indicates the normal state of the block when a furnace is in use and carbon potential, oxygen concentration and dewpoint calculations are being made.

Clean indicates the probe Burn-Off phase of the probe cleaning sequence. This state is initiated by, manually setting *Probe Clean.ClnStart* **Yes** or, automatically when the scheduled probe cleaning sequence is due to start if *Main.PrbState* shows **Measure**. During this state *Probe Clean.ClnValve* is set to **Yes** indicating that the purge gas should be applied to the probe. The probe cleaning sequence will last the duration defined in *Probe Clean.ClnTime* unless the *Main.ProbeSt* or *Main.TempSt* show **Bad**, the temperature (*TempIn* + *TempOfs*) used in the furnace is below the configurable minimum temperature, *Main.MinCalcT* or above *Probe Clean.ClnMaxT*, the probe cleaning sequence is aborted by setting *Probe Clean.ClnStart* **No**, or *Probe Clean.ClnAbort* is set **Yes**. After successful completion of the probe cleaning sequence *Probe Clean.LastClnmV* records the probe mV as a quality factor, but will show **-1** if the probe cleaning sequence completed abnormally. During the probe Burn-Off phase, outputs are frozen, *Main.PVFrozen* set **Yes**.

ClnRecov indicates the probe is adjusting to the furnace atmosphere, preparing for normal operating conditions. During this state, *Probe Clean.ClnValve* is set **No** to stop the purge gas and *Main.ProbeIn* is monitored to ensure it recovers to 95% of the original value. When *Main.ProbeIn* reaches 95% of the original value *Main.PrbState* will return to **Measure**, subject to a minimum and maximum recovery time, *Probe Clean.MnxCnRcvT* and *Probe Clean.MxCnRcvT*. *Main.Alarms.ClnRcWrn* is set TRUE if 95% of the original value has not been reached within the *Probe Clean.MxCnRcvT* and *Main.PrbState* will return to **Measure**. During the recovery phase, outputs are frozen, *Main.PVFrozen* set **Yes**.

TestImp indicates that the probe impedance is being measured. This state is initiated by setting *Impedance.ImpStart* **Yes** when *Main.PrbState* shows **Measure**. During this state, *Impedance.ApplyRes* is set **Yes**, indicating a load resistance should be applied across the probe mV input. The impedance measurement phase will last for 30s unless the temperature (*TempIn* + *TempOfs*) used in the furnace is below the configurable minimum temperature, *Main.MinCalcT*, *Main.ProbeSt* or *Main.TempSt* show **Bad**, the measurement is aborted *Impedance.ImpStart* set **No**, or *Impedance.ImpAbort* is set to **Yes**. After successful completion of the impedance measurement, *Impedance.Impednce* shows the measured probe impedance, but will show **-1** if the test was aborted early. During the impedance measurement phase, outputs are frozen, *Main.PVFrozen* set **Yes**.

NOTE *Main.PrbState* should be wired to a DO_UIO block associated with the ZI module, that includes a 10kΩ impedance measurement load built in. If an alternative load is used, *Impedance.ImpTstR* must be changed accordingly.

ImpRecov indicates the probe is adjusting to the removal of the load, and is preparing for normal operating conditions. During this state, *Impedance.ApplyRes* is set **No** and *Main.ProbeIn* is monitored to ensure it recovers to 99% of the original value. When *Main.ProbeIn* reaches 99% of the original value *Main.PrbState* will return to **Measure**, subject to a maximum recovery time, *Impedance.MxImpRcvT*. *Main.Alarms.ImpRcWrn* is set TRUE if 99% of the original value has not been reached within the *Impedance.MxImpRcvT* and *Main.PrbState* will return to **Measure**. During the recovery phase, outputs are frozen, *Main.PVFrozen* set **Yes**.

CarbnPot. Shows the calculated Carbon Potential existing in the furnace in the range 0 to 2.55. This is calculated using temperature and carbon probe mV signals based on the ratio of oxygen concentration outside the furnace to the level of oxygen in the furnace. If the furnace temperature (*Main.TempIn* + *Main.TempOfs*) is below the value shown in *Main.MinCalcT*, *Main.CarbnPot* is forced to 0 (zero). If above the *Main.MinCalcT* value, and *Main.ProbeSt* or *Main.TempSt* show **Bad**, the last good value is retained.

CrbPotSt. (Good/Bad). Indicates the status of the Carbon Potential value. Set **Bad** when *Main.ProbeSt* or *Main.TempSt* show **Bad**, or if the furnace temperature (*Main.TempIn* + *Main.TempOfs*) is below the value shown in *Main.MinCalcT*.

aC_CO_O2. Shows the calculated carbon activity for the surface gas reaction between CO and Oxygen, applicable to the **Ferronova** probe type only. This is derived using,

$$(CO \leftrightarrow C + \frac{1}{2}O_2), \text{ therefore, } aC_CO_O2 = f(pCO, p1/2O2)$$

DewPoint. Shows the calculated Dewpoint within the range -60 to 160. If the furnace temperature (*Main.TempIn* + *Main.TempOfs*) is below the value shown in *Main.MinCalcT*, *Main.DewPoint* is forced to 0 (zero). If above the *Main.MinCalcT* value and *Main.DewPntSt* is set **Bad**, the last good value is retained.

DewPntSt. (Good/Bad). Indicates the status of *Main.DewPoint*. Set **Bad** when *Main.ProbeSt* or *Main.TempSt* show **Bad**, or if the furnace temperature (*Main.TempIn* + *Main.TempOfs*) is below the value shown in *Main.MinCalcT*.

Oxygen. Shows the calculated Oxygen in units determined by *Main.OxygenEx*. If the furnace temperature (*Main.TempIn* + *Main.TempOfs*) is below the value shown in *Main.MinCalcT*, *Main.Oxygen* is forced to 0 (zero). If above the *Main.MinCalcT* value, and *Main.OxygenSt* is **Bad**, the last good value is retained.

NOTE The calculation before applying the *Main.OxygenEx* is clipped in the range 0 to 100% (partial pressure 1.0).

OxygenSt. (Good/Bad). Indicates the status of the Oxygen Potential value. Set **Bad** when *Main.ProbeSt* or *Main.TempSt* show **Bad**, or if the furnace temperature (*Main.TempIn* + *Main.TempOfs*) is below the value shown in *Main.MinCalcT*.

OxygenEx. The Oxygen exponent in the range 0 to 19. This determines the units used in *Main.Oxygen*. The default, 2, is Oxygen in %. A value of 0 is partial pressure of Oxygen, and a value of 6 presents Oxygen in Parts Per Million, PPM.

NOTE To calculate **Log Oxygen**, this should be set to 2 and the *Main.Oxygen* output should be wired to an EXPR block to calculate the log.

Tolernce. A multiplicative scaling factor applied to soot line, before comparison with the Carbon Potential, used to determine *Main.SootWrn*. Adjusting this will raise or lower the soot tolerance by the amount entered and can be used to remove any nuisance alarms derived from *Main.SootWrn*.

NOTE The soot line calculation employs a fixed Alloy factor of 1.0. Use the CARB_DIFF block if a Soot Line figure including alloy factor is required.

SootWrn. (No/Yes). Shows **Yes** if the Probe has detected atmospheric conditions that may cause a deposit of soot on surfaces in the furnace. Sets *Main.Alarms.SootWrn* TRUE.

Ballnt. (No/Yes). Indicates any PID control loop wired to the output of this block should perform an integral balance to prevent steps, bumps, due to a proportional kick. *Main.Ballnt* shows **Yes** for one update when exiting the probe cleaning sequence, or impedance measurement sequence, or the **NotReady** state, or when CO or H2 gas references change between local and remote.

Probe Clean Page

ClnEnabl. (Yes/No). Indicates that automatic probe cleaning is permitted. When set **No**, probe cleaning will not start, but will complete if already in progress.

NOTE The probe clean sequence is inhibited if *Probe Clean.ClnFreq* is set to 00:00:00.

ClnStart. (No/Yes). Used to manually start the probe clean sequence. If the probe clean sequence is started manually, *Probe Clean.ClnStart* set **Yes**, *Probe Clean.ClnFreq* is disabled until the probe clean sequence is completed, and *Probe Clean.TimToCln* is reset.

ClnAbort. (No/Yes). Used to cancel the current probe clean sequence. If the probe clean sequence is aborted, *Probe Clean.ClnAbort* set **Yes**, *Probe Clean.LastClnmV* is set **-1** and *Main.PrbState* shows **ClnRecov**. Measurement will only resume after the probe recovers in the normal way subject to *Probe Clean.MnCnRcvT* and *Probe Clean.MxCnRcvT*.

ClnMsgRt. (No/Yes). When set **Yes**, *Probe Clean.ClnRcWrn*, *Main.Alarms.ClnRcvWn*, *Main.Alarms.ClnAbort*, and *Main.Alarms.ClnTemp* are reset. Automatically returns **No**.

ClnFreq. Sets the interval between probe cleaning sequences in hh:mm:ss format, i.e. 04:00:00 equals 4 hour intervals. The scheduled periodic probe cleaning sequence is disabled if set 00:00:00.

ClnTime. Defines the time taken to complete the Burn-Off phase of the probe cleaning sequence, see *Main.PrbState*.

ClnMaxT. Defines the maximum temperature allowed during the probe cleaning sequence. The probe cleaning sequence is aborted if this value is exceeded. Sets *Main.Alarms.ClnTemp* TRUE.

MnCnRcvT. Defines the minimum recovery time allowed after each probe cleaning sequence before measurement resumes.

MxCnRcvT. Defines the maximum recovery time allowed after each probe cleaning sequence before measurement resumes. Sets *Main.Alarms.ClnRcWrn* TRUE if 95% of the original value is not obtained before this time is exceeded.

TimToCln. Shows the time remaining in hh:mm:ss, until the next scheduled periodic probe cleaning sequence is due to start.

ClnRcovT. Shows the time taken for *Main.ProbeIn* to obtain 95% of the original value prior to starting the probe clean sequence. If this value exceeds *Probe Clean.MxCnRcvT*, *Probe Clean.ClnRcovT* is set 00:00:00, and *Probe Clean.ClnRcvWn* is set **Yes**.

ClnRcvWn. (No/Yes). Indicates that performance of the probe has degraded. If set **Yes**, *Main.ProbeIn* did not recover to 95% of the original value in the permitted time, *Probe Clean.MxCnRcvT*. This will reset **No** after completion of the next successful probe clean sequence, or if *Probe Clean.ClnMsgRt* is set **Yes**.

LstClnmV. This shows the probe mV value at the end of the last clean, before *Main.PrbState* shows **Recovery**. This is a good indicator of probe health. If the clean failed, or was aborted, this will show **-1**, but a value more than 200mV generally indicates a fault in the probe.

ClnValv. (No/Yes). Indicates the probe clean valve state. An instruction to provide the purge gas to the probe. Set **Yes** when *Main.PrbState* shows **Cleaning**, and **No** at all other times.

Gas Ref Page

CO_Ideal. Defines the gas reference value used to calculate the mV carbon potential when *Main.OxygenTyp* shows **NernstCP**, see *Main.OxygenTyp*.

CO_Local/H2_Local. Sets local CO and H2 Endothermic Gas Reference value used when remote gas is not in use.

NOTE The gas reference is included in the process factors for the MMI probes, *Main.ProbeTyp* shows MMI, and uses defaults 20.0% for CO gas and 40.0% for H2 gas.

CO_Remte/H2_Remte. Defines the remote CO and H2 Endothermic Gas Reference from a gas analyser. The remote value is shown when wired from an input/comms block.

CO_RemSt/H2_RemSt. (Good/Bad). Defines the remote CO and H2 Endothermic Gas Reference Status. The remote CO and H2 status is shown when wired from an input/comms block. **Good** indicates good remote gas input, **Bad** indicates a failed remote gas input, i.e. I/O module calculations are derived from *Gas Refs.CO_Local* and *Gas Refs.H2_Local*.

CO_RemEn/H2_RemEn. Remote CO and H2 Gas Reference Enable. This enables the remote gas reference, *Gas Refs.CO_Remte*, and *Gas Refs.H2_Remte*, respectively.

CO_InUse/H2_InUse. Shows the CO and H2 Endothermic Gas Reference value used in the Carbon Potential and DewPoint calculations, respectively. Follows *Gas Refs.CO_Local* and *Gas Refs.H2_Local*, unless *Gas Refs.CO_RemEn* and *Gas Refs.H2_RemEn* shows **Yes** and *Gas Refs.CO_RemSt* and *H2_RemSt* shows **FALSE**, when it follows *Gas Refs.CO_Remte* and *Gas Refs.H2_Remte*.

Impedance Page

ImpStart. (No/Yes). Starts the probe cleaning sequence when set **Yes**.

ImpAbort. (No/Yes). Used to cancel the current impedance measurement. If the impedance measurement is aborted, *Impedance.ImpAbort* set **Yes**, *Impedance.Impednce* is set **-1** and *Main.PrbState* shows **ImpRecov**. Measurement will only resume after the probe recovers.

ImpMsgRt. (No/Yes). When set **Yes**, *Impedance.ImpRcvWn*, *Impedance.PrbImpHi*, *Main.Alarms.ImpRcvWn*, *Main.Alarms.PrbImpHi*, and *Main.Alarms.ImpAbort* are reset. Automatically resets **No**.

MxImRcvT. Sets the maximum recovery time allowed after each impedance measurement. Sets *Main.Alarms.ImpRcvWn* TRUE if 99% of the original value is not obtained before this time is exceeded.

ImpTstR. Set the value of the load resistor used in the impedance measurement. Use the default, 10k Ω , if a ZI module is fitted to a T2550 subsystem.

MaxImp. Defines the threshold value of *Impedance.PrbImpHi*. If the value is exceeded *Main.Alarms.PrbImpHi* is set TRUE.

Impednce. Shows the measured probe impedance value.

PrbImpHi. (No/Yes). Shows the maximum probe impedance threshold value has been exceeded. If the *Impedance.MaxImp* value is exceeded, this shows **Yes**, and *Main.Alarms.PrbImpHi* is set TRUE.

ImpRcovT. Shows the time taken for *Main.ProbeIn* value to obtain more than 99% of the original value prior to starting the probe impedance measurement. If this value exceeds *Impedance.MxImRcvT*, *Impedance.ImpRcovT* is set 00:00:00, and *Impedance.ImpRcvWn* is set **Yes**.

ImpRcvWn. (No/Yes). Shows the *Main.ProbeIn* value has failed to recover 99% of the original value in the permitted time, *Impedance.ImpRcovT*, prior to starting the probe impedance measurement. This will reset **No** after completion of the next successful probe impedance measurement, or if *Impedance.ImpMsgRt* is set **Yes**.

ApplyRes. (No/Yes). Indicates the use of the test resistance. If set **Yes**, a test resistance value defined in *Impedance.ImpTstR* should be applied across the probe input. This can be wired to a DO_UIO block associated with channel 1 of a ZI module, in a T2550 I/O subsystem.

NOTE Each ZI module has a built-in 10K Ω test load.

TC_LIFE: THERMOCOUPLE LIFE EXPECTANCY BLOCK

Block function

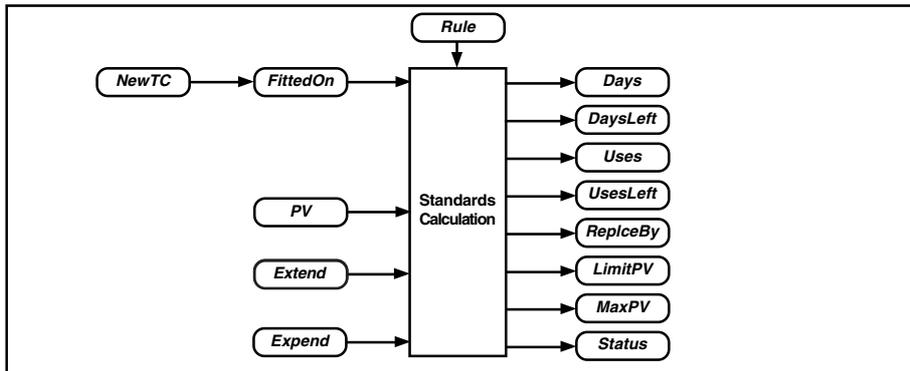


Figure 23 Block schematic

This block provides monitoring and diagnostics information concerning the life expectancy of base metal load thermocouples used in furnaces complying with a defined standard. The standards supported are the AMS2750D and AMS2750E. In addition, a user-defined thermocouple life data range is also supported by specifying the rule as EXTEND, and referring to a TC_LIFE_EX block which contains the life data.

Block parameters

Symbols used in this table are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Rule	Applied standards calculation (AMS2750D/E or EXTEND)	Enum	
Expend	Expendable thermocouple installed	T/F	☐➔
PV	Process variable from thermocouple	Eng	☐➔
FittedOn	Date new thermocouple installed	Date	☐➔
NewTC	New thermocouple installed	T/F	☐➔
Alarms			☐➔ 📖 🔊
Software	Block RAM data sumcheck error	T/F	
Expired	Acceptable life expectancy exceeded	T/F	
Replace	Last valid use of thermocouple detected	T/F	
Uninit	Uninitialised thermocouple detected	T/F	
Combined	OR-ing of all Alarms bits	T/F	
LegacyRI	A legacy thermocouple life rule is being used (AMS2750D)	T/F	
BadExtnd	Rule parameter is EXTEND, and Extend parameter not defined	T/F	
FileStat	Writing to the filing system not successful	T/F	
Uses	Number of uses	Integer	☐➔
UsesLeft	Number of uses remaining	Integer	☐➔
Days	Number of days used	Integer	☐➔
DaysLeft	Number of days use remaining	Integer	☐➔
ReplceBy	Last day of use	Date	
SerialNo	Serial number of thermocouple	Integer	
Extend	Name of block that defines additional thermocouple life data		☐➔
MaxPV	Max recorded temperature	Eng	☐➔
LimitPV	Max permitted temperature	Eng	☐➔
Status	Condition of thermocouple		☐➔ 📖
Expired	Max use of thermocouple exceeded	T/F	
Replace	Last use indicated	T/F	
Uninit	Detected uninitialised thermocouple	T/F	
LegacyRI	A legacy thermocouple life rule is being used (AMS2750D)	T/F	
BadExtnd	Rule parameter is EXTEND, and Extend parameter not defined	T/F	
FileStat	Writing to the filing system not successful	T/F	

Table 35 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

Rule. (AMS2750D/AMS2750E/EXTEND). Defines the standard used to calculate the usage values, with a default value of AMS2750E. Setting *Rule* to AMS2750D allows backward compatibility with the AMS2750D standard, but also asserts the *Status.LegacyRl* and *Alarm.LegacyRl* bits indicating a legacy standard being used. Setting *Rule* to EXTEND obtains thermocouple life data from the TC_LIFE_EX block whose name is defined in the *Extend* field.

Expend. Used to indicate the use of an expendable thermocouple. Set to TRUE if using expendable thermocouple, and to reset the calculations accordingly.

PV. Process variable derived from thermocouple based on the units defined in *IP_Type* of the corresponding header block. That is: °C, °F, K, or R.

FittedOn. Shows when the thermocouple was installed. When written to, or when *NewTC* is set TRUE, the life expectancy calculations, *Uses*, *UsesLeft*, *Days*, *DaysLeft*, and *ReplceBy*, are reset.

NewTC. (TRUE/FALSE). Used to write the date a new thermocouple is installed to *FittedOn* and reset the life expectancy calculations for the thermocouple.

SerialNo. Used to show the Serial number (30 characters maximum) of the thermocouple installed on the date shown in *FittedOn*. This should be input by the user when the thermocouple is installed.

Extend. Optional. Used to hold the name of a block type TC_LIFE_EX which defines additional thermocouple life data. This allows the use of thermocouple life data that differs from the AMS2750D or AMS2750E standards. The contents of this parameter is ignored unless *Rule* is set to EXTEND.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.
- **Expired.** Asserted, when the corresponding *Status.Expired* is TRUE. Indicates the acceptable life expectancy of the thermocouple has been exceeded.
- **Replace.** Asserted, when the corresponding *Status.Replace* is TRUE. Indicates the acceptable usage of the thermocouple has been reached and it should be replaced after this use.
- **Uninit.** Asserted, when the corresponding *Status.Uninit* is TRUE. Indicates the installed thermocouple calculation is not initialised.
- **Combined.** Asserted if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.
- **LegacyRl.** Asserted if the *Rule* parameter is set to AMS2750D, indicating a legacy thermocouple life rule is being used.
- **BadExtnd.** Asserted if the *Rule* parameter is set to EXTEND, and the extension block, identified by the *Extend* parameter, is undefined. When *Alarms.BadExtnd* is asserted, *Alarms.Uninit* is also asserted.
- **FileStat.** Asserted if an attempt to write to the filing system was unsuccessful.

Uses. Shows the total usage to date. This is incremented each time the temperature rises above 50°C. The temperature must persist at (or above) each threshold value for more than 10 seconds before the usage data for that threshold is incremented by 1. For example, once 650°C is exceeded, the temperature will have to drop below 650°C and remain there for at least 10 seconds and then rise above 650°C for at least another 10 seconds before this is incremented by 1

Standard	Thermocouple type	Thermocouple range	Uses (or Days)
AMS2750D	Non-expendable	<650°C	270 (or 90 days)
		>650°C - ≤980°C	180 (or 90 days)
		>980°C - ≤1205°C	90 (or 30 days)
		>1205°C - ≤1260°C	10
	Expendable	>1260°C	1
		<650°C	30
AMS2750E	Non-expendable	>650°C	1
		<650°C	270 (or 90 days)
		>650°C - ≤980°C	180 (or 90 days)
		>980°C - ≤1205°C	90 (or 90 days)
	Expendable	>1205°C - ≤1260°C	10 (or 90 days)
		>1260°C	1 (or 90 days)
EXTEND	Non-expendable	<650°C	30 (or 90 days)
	Expendable	>650°C	1 (or 90 days)
		User-defined. Refer to TC_LIFE_EX	User-defined
		User-defined. Refer to TC_LIFE_EX	User-defined

Table 36 Thermocouple Usage values

UsesLeft. Shows the maximum number of uses remaining in the thermocouple. The acceptable life of the thermocouple may be limited by time rather than usage, and the number of uses may be reduced if operating at a higher temperatures.

Days. Shows the total number of days the thermocouple has been used to date.

DaysLeft. Shows the maximum number of days remaining for the thermocouple. The acceptable life of the thermocouple may be limited by usage rather than time.

ReplceBy. Shows the calculated date indicating the last day of use. By default, this will show the calculated 90 days from when the thermocouple was installed, derived from *FittedOn*. If a thermocouple has already expired, the date indicated is the date that it expired.

MaxPV. Shows the maximum temperature recorded by the thermocouple, since fitted.

LimitPV. Shows the maximum temperature permitted. When using a new thermocouple this shows 9999, but will decrease over time and use until it shows 0, indicating the life of the thermocouple has expired. Sets *Status.Expired* TRUE. The *LimitPV* field is updated as soon as a production run starts, so should be checked before (and not during) the commencement of a production run.

Status. Bitfield indicating thermocouple conditions.

- **Expired.** TRUE, if the maximum permitted life expectancy of the thermocouple has been exceeded.
- **Replace.** TRUE, if the acceptable usage limit of the thermocouple has been reached. This indicates the last use of the thermocouple has commenced.
- **Uninit.** TRUE, if the installed thermocouple is not initialised when *FittedOn* is written to. Sets *Alarms.Uninit* TRUE.
- **LegacyRl.** TRUE, if the *Rule* parameter is set to AMS2750D, indicating a legacy thermocouple life rule is being used.
- **BadExtnd.** TRUE, the *Rule* parameter is set to EXTEND, and the extension block, identified by the *Extend* parameter, is undefined. When *Status.BadExtnd* is asserted, *Status.Uninit* is also asserted.
- **FileStat.** TRUE, if an attempt to write to the filing system was unsuccessful.

TC_LIFE_EX: THERMOCOUPLE LIFE EXTENSION BLOCK

Block function

The TC_LIFE_EX block is an extension to the thermocouple life (TC_LIFE) block. It allows for user-definable thermocouple life data to be used in the TC_LIFE block, extending the functionality beyond just the AMS2750D and AMS2750E standards. One TC_LIFE_EX block may be referenced by many TC_LIFE blocks.

The TC_LIFE_EX block allows up to five temperature ranges to be defined for both non-expendable and expendable thermocouples. The ranges are defined using a list of maximum threshold temperatures (1 to 5) where each value in the list is greater than the previous temperature specified (that is, temperature 1 \leq temperature 2 \leq temperature 3 \leq temperature 4 \leq temperature 5). Note that the temperature range starts at 50°C.

For example, if the threshold temperatures were 650°C, 980°C, 1205°C, and 1260°C, then:

- range 1 would be defined as >50°C and \leq 650°C,
- range 2 would be defined as >650°C and \leq 980°C,
- range 3 would be defined as >980°C and \leq 1205°C,
- range 4 would be defined as >1205°C and \leq 1260°C, and
- range 5 would be defined as >1260°C.

For each temperature range (identified by the upper temperature threshold), the number of thermocouple *uses* and *days* is defined in an ever-decreasing manner (that is, the highest temperature has the fewest permitted uses/days). A *uses* or *days* value of 0 represents no limit is set.

The maximum temperature that can be specified is 9999, and is entered in the current strategy's temperature units (that is, C/F/K/R). As temperature threshold number 5 must be the highest possible temperature, this is always set to 9999 (or "MAX") and cannot be changed.

Block parameters

Symbols used in this table are explained in *Table 1*. Parameter information is given in *Table 37* following.

Parameter	Function	Units	Status
NTemp1Rg	Non-expendable T/C temperature range 1	Eng	
NUses1Rg	Non-expendable T/C maximum uses (for temp range 1)	Integer	
NDays1Rg	Non-expendable T/C maximum days (for temp range 1)	Integer	
NTemp2Rg	Non-expendable T/C temperature range 2	Eng	
NUses2Rg	Non-expendable T/C maximum uses (for temp range 2)	Integer	
NDays2Rg	Non-expendable T/C maximum days (for temp range 2)	Integer	
NTemp3Rg	Non-expendable T/C temperature range 3	Eng	
NUses3Rg	Non-expendable T/C maximum uses (for temp range 3)	Integer	
NDays3Rg	Non-expendable T/C maximum days (for temp range 3)	Integer	
NTemp4Rg	Non-expendable T/C temperature range 4	Eng	
NUses4Rg	Non-expendable T/C maximum uses (for temp range 4)	Integer	
NDays4Rg	Non-expendable T/C maximum days (for temp range 4)	Integer	
NTemp5Rg	Non-expendable T/C temperature range 5	Eng	
NUses5Rg	Non-expendable T/C maximum uses (for temp range 5)	Integer	
NDays5Rg	Non-expendable T/C maximum days (for temp range 5)	Integer	
ETemp1Rg	Expendable T/C temperature range 1	Eng	
EUses1Rg	Expendable T/C maximum uses (for temp range 1)	Integer	
EDays1Rg	Expendable T/C maximum days (for temp range 1)	Integer	
ETemp2Rg	Expendable T/C temperature range 2	Eng	
EUses2Rg	Expendable T/C maximum uses (for temp range 2)	Integer	
EDays2Rg	Expendable T/C maximum days (for temp range 2)	Integer	
ETemp3Rg	Expendable T/C temperature range 3	Eng	
EUses3Rg	Expendable T/C maximum uses (for temp range 3)	Integer	
EDays3Rg	Expendable T/C maximum days (for temp range 3)	Integer	
ETemp4Rg	Expendable T/C temperature range 4	Eng	
EUses4Rg	Expendable T/C maximum uses (for temp range 4)	Integer	
EDays4Rg	Expendable T/C maximum days (for temp range 4)	Integer	

Parameter	Function	Units	Status
ETemp5Rg	Expendable T/C temperature range 5	Eng	
EUses5Rg	Expendable T/C maximum uses (for temp range 5)	Integer	
EDays5Rg	Expendable T/C maximum days (for temp range 5)	Integer	

Table 37 Block parameters

Block specification menu

The following is given in addition to *Table 37*.

NTemp1Rg to NTemp5Rg. These fields specify the temperature ranges (1 to 5) that have a direct relationship to the service life of a non-expendable thermocouple (refer to the *NUses1Rg* and *NDays1Rg* parameters, for example). The temperature ranges are listed as monotonically increasing values, in that $NTemp1Rg \leq NTemp2Rg \leq NTemp3Rg$, and so on. *NTemp5Rg* is not writeable, and always holds the maximum value of 9999 (or “MAX”).

NUses1Rg to NUses5Rg. These fields specify the maximum number of times (uses) the non-expendable thermocouple can reach the corresponding temperature threshold specified in *NTemp1Rg* to *NTemp5Rg* respectively. The number of uses listed for each temperature threshold is always less than or equal to the previous one in the series. Enter a zero to indicate that the corresponding temperature range has no limit to the maximum number of uses.

NDays1Rg to NDays5Rg. These fields specify the maximum number of days the non-expendable thermocouple can reach the corresponding temperature threshold specified in *NTemp1Rg* to *NTemp5Rg* respectively. The number of days listed for each temperature threshold is always less than or equal to the previous one in the series. Enter a zero to indicate that the corresponding temperature range has no limit to the maximum number of days.

ETemp1Rg to ETemp5Rg. These fields specify the temperature ranges in a similar way to *NTemp1Rg* to *NTemp5Rg*, except they relate to the service life of an expendable thermocouple. Refer to the *NTemp1Rg* to *NTemp5Rg* parameters for details. *ETemp5Rg* is not writeable, and always holds the maximum value of 9999 (or “MAX”).

EUses1Rg to EUses5Rg. These fields specify the maximum number of times (uses) the thermocouple can be used in a similar way to *NUses1Rg* to *NUses5Rg*, except they relate to the service life of an expendable thermocouple. Refer to the *NUses1Rg* to *NUses5Rg* parameters for details.

EDays1Rg to EDays5Rg. These fields specify the maximum number of days the thermocouple can reach the corresponding temperature threshold in a similar way to *NDays1Rg* to *NDays5Rg*, except they relate to the service life of an expendable thermocouple. Refer to the *NDays1Rg* to *NDays5Rg* parameters for details.

TC_SEL: THERMOCOUPLE SELECTOR BLOCK

Block function

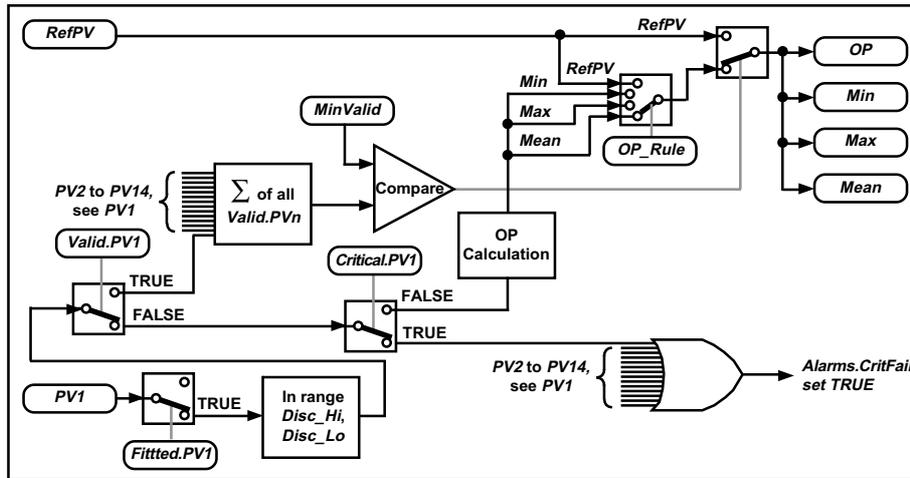


Figure 24 Block schematic

This block is used to manage multiple load thermocouples for furnace applications. It provides for a maximum of 14 thermocouples which is sufficient for a 2 control zone furnace using a standard **2 butted cubes** topology, i.e. apex of cubes plus centre points. It will derive a summary temperature used to control against a built in set of rules that may be dynamically changed by the wiring, e.g. change from **Min** to **Max** on change of segment in the Setpoint Program. It can also be configured to discount thermocouples that are out of range.

Block parameters

Symbols used in this table are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units*	Status*
RefPV	Reference PV	Eng	<input type="checkbox"/>
PV1 to PV14	Process variable from thermocouples	Eng	<input type="checkbox"/>
Alarms			
Software	Block RAM data sumcheck error	T/F	<input type="checkbox"/>
InvalidPV	Fitted and valid thermocouples do not correspond	T/F	<input type="checkbox"/>
CritFail	Discounted critical thermocouples detected	T/F	<input type="checkbox"/>
Combined	OR-ing of all Alarms bits	T/F	<input type="checkbox"/>
Fitted PV1 to PV14	Installed thermocouple	T/F	<input type="checkbox"/>
Critical PV1 to PV14			
Disc_HI	Maximum discounting limit temperature	Eng	<input type="checkbox"/>
Disc_LO	Minimum discounting limit temperature	Eng	<input type="checkbox"/>
MinValid	Defines minimum number of valid thermocouples required	Integer	<input type="checkbox"/>
Valid PV1 to PV14	Shows valid thermocouple operation	T/F	<input type="checkbox"/>
OP_Rule			
Ref	Controls Ref output rule	T/F	<input type="checkbox"/>
Min	Controls Min output rule	T/F	<input type="checkbox"/>
Max	Controls Max output rule	T/F	<input type="checkbox"/>
Mean	Controls Mean output rule	T/F	<input type="checkbox"/>

Continued...

Parameter	Function	Units*	Status*
<i>Continued...</i>			
OP	Calculated output load temperature	Eng	<input type="checkbox"/> <input type="checkbox"/>
Min	Lowest input value from valid thermocouples	Eng	<input type="checkbox"/> <input type="checkbox"/>
Max	Highest input value from valid thermocouples	Eng	<input type="checkbox"/> <input type="checkbox"/>
Mean	Meaned input value from valid thermocouples	Eng	<input type="checkbox"/> <input type="checkbox"/>

Table 38 Block Parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

RefPV. Defines the reference temperature used when a load thermocouple is invalid.

PV1 to PV14. Thermocouple input value.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.
- **InvalidPV.** Asserted, if the number of *Valid* subfields that are FALSE exceeds the value defined in *MinValid*. This forces *OP* to *RefPV*.
- **CritFail.** Asserted, when a thermocouple defined as critical, *Critical.PVn* set TRUE, has been discounted because the PV breaches the limits defined by *Disc_HI* or *Disc_LO*.
- **Combined.** Asserted if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

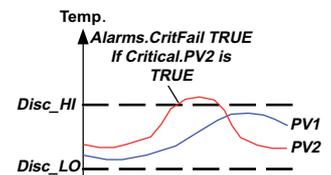
Fitted. Bitfields indicating fitted thermocouples.

- **PV1 to PV14.** TRUE, when thermocouple is fitted.

Critical. Bitfields indicating fitted thermocouples are critical to the operation.

- **PV1 to PV14.** TRUE, when thermocouple is critical. Each subfield is used in conjunction with *Disc_HI* and *Disc_LO* to set *Alarms.CritFail* TRUE.

Disc_HI, Disc_LO. These define the discount limits for the fitted thermocouples. *Disc_HI* defines the high value, and *Disc_LO* defines the low value, that when breached by a thermocouple input value (*PVn*) defined as critical (*Critical.PVn* TRUE) sets *Alarms.CritFail* TRUE.



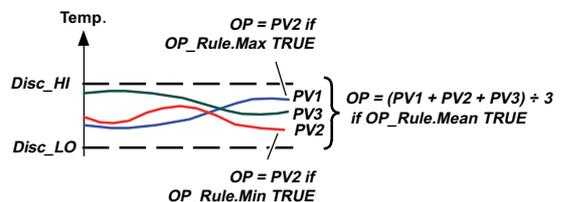
MinValid. Defines the minimum required number of valid thermocouples allowed. This is used in conjunction with *OP_Rule* to provide a calculated value for *OP*.

Valid. Bitfields indicating thermocouples that are fitted and are not discounted.

- **PV1 to PV14.** TRUE, when thermocouple is fitted and the corresponding PV is between the configured *Disc_HI* and *Disc_LO* values. FALSE, when thermocouple is not fitted and/or the corresponding PV has breached the configured *Disc_HI* and *Disc_LO* values.

OP_Rule. Bitfields for controlling the rule used to provide the *OP* value. These bitfields operate in priority order.

- **Ref.** Highest priority bitfield. TRUE, indicates that *OP* is set to the value specified in *RefPV*. This rule is also used if insufficient thermocouples are fitted and valid, see *Fitted* and *Valid*.
- **Min.** TRUE, indicates that *OP* is set to the minimum value derived from only valid thermocouples, see *Valid*.
- **Max.** TRUE, indicates that *OP* is set to the maximum value derived from only valid thermocouples, see *Valid*.
- **Mean.** Lowest priority bitfield. TRUE, indicates that *OP* is set to a calculated arithmetic mean value derived from the valid thermocouples, see *Valid*.



OP. Shows the output load temperature derived using *OP_Rule*. This value will default to *RefPV* if insufficient thermocouples are fitted and valid, see *Fitted* and *Valid*.

Min, Max, Mean. These show the lowest, highest and calculated mean, from current PV of all valid thermocouples respectively. Each value defaults to *RefPV* if insufficient thermocouples are valid, see *Valid*.

CHAPTER 5 CONFIGURATION FUNCTION BLOCKS

This category of Template Function Block provides the Strategy with functions for defining the type of instrument on which a Strategy will run. A single instrument specific Configuration Template Block must exist in each LIN Database file, .dbf.

If creating a Strategy that consists of blended LIN Database layers, each layer must have its own type of 'dummy' header block. The PROGRAM block is used on each LIN Database file layer, .dbf, of the Strategy in any LIN Instrument except the T600 Series LIN Instruments, which uses the specific, PROGT600, block.

PROGRAM: PROGRAM CONFIGURATION BLOCK

PROGT600: T600 PROGRAM CONFIGURATION BLOCK

Block function

These blocks are a type of 'header' block. They are a 'dummy' header block, containing only the TagName, Type, Task, LIN Name, DBase, and Rate fields and are specific to a target instrument, or group of related target instruments. Each block must be present in each of the LIN Database layers that will be blended together to form a complete (regular) LIN Database in that specific LIN Instrument.

NOTE. The PROGT600 header block is unique to the T600 Series Instrument.

The header blocks in each of the LIN Database layers in the blend must have identical Name fields, i.e. the name of the LIN Database to be run in the target instrument. In the blended LIN Database, the 'dummy' header blocks are discarded and replaced by the (regular) header block configured in the base.dbf layer of the Strategy.

A Blended Database (or Strategy) is a Read Only file that is the result of a Build command on a selected LIN Node containing layered LIN Database files (.dbf) and auto generated layers (.ujc).

The Build command creates the Default DBF from any number of *.dbf and *.ujc files in a specified order as configured in the buildlst.tbl. During a Build 'dummy' header and cached blocks are removed but all other blocks are blended into a single LIN Database.

NOTE. The DefaultDBF and Auto generated layers are all Read Only files and do NOT require a graphics file, .grf.

Block parameters

The block has no parameters other than the block header parameters, see *Appendix D page 544* for details of these 'header' fields.

Block specification menu

These fields are used to control the operation of the block.

Tagname, Type, Task. These are block related parameters used to control the operation of the block within the LIN Database.

LIN Name, DBase, Rate. These are LIN Database related parameters used to control the communications between LIN Instruments.

T600: T600 CONFIGURATION BLOCK

Block function

The T600 block is a 'header' block that must appear in a control strategy to be executed in a T600 Series instrument, and acts as a general-purpose block for status information on the whole instrument. In particular, it allows local access via the LIN to the network's Real-Time Clock/calendar data, required for alarm handling. The block also specifies a mains noise-rejection frequency, temperature system for a single instrument, a coldstart (system reset) time, and contains a brownout alarm that latches on when the power is interrupted for a specified time. The module type assigned to each of the eight I/O module addresses on the internal serial communications bus is also indicated by the T600 block.

Block parameters

Symbols used in *Table 39* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Time	Current time-of-day	hh:mm:ss	
Date	Current date	dd/mm/yy	
IP_type	Mains line frequency, temperature system, etc.	ABCD hex	
60Hz	Mains frequency (TRUE = 60Hz, FALSE = 50Hz)	T/F	D
Imperial	Temp. system (TRUE = Imperial, FALSE = SI)	T/F	
		T/F	
		T/F	
		T/F	C
		T/F	
		T/F	
		T/F	
Sequence	Sequence option memory module fitted	T/F	B
FastROM	Fast ROM fitted	T/F	
FourLoop	Four-loop option memory module fitted	T/F	
		T/F	
Big_RAM	Enough RAM fitted on motherboard for sequences	T/F	A
FixedFun	Fixed-function option memory module fitted	T/F	
AdvaFeat	Advanced features option 'A' fitted (AGA8 calcs.)	T/F	
		T/F	
Site1 to Site8	Module type identifier	Alphanumeric	
BrownOut	Power-off time to trip BrownOut alarm	mins	
ColdStrt	Minimum power-off time to trigger cold start	mins	
Node	ALIN network address	(01-FE hex)	
Model	T600 model identifier		
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
BrownOut	Power failure longer than BrownOut	T/F	
ComS/W	Common 'local' block software error	T/F	
UTskEr1 to 4	User task 1-4 error (halted)	T/F	
MainPSU	Main PSU failed	T/F	
StbyPSU	Standby PSU failed	T/F	
RTCinit	Uninitialised Real-Time Clock	T/F	
Combined	OR-ing of all Alarms bits	T/F	
UstrAlm	User alarm relay in alarm (contacts open)	T/F	
AreaNo	Security key area	(0 - 63)	
TimeOut	Database inspect mode timeout	secs.	
Options	Front panel, comms, & database save options	ABCD hex	
FPdis1	Disable front-panel access to loop 1	T/F	D
FPdis2	Disable front-panel access to loop 2	T/F	
FPdis3	Disable front-panel access to loop 3	T/F	
FPdis4	Disable front-panel access to Loop 4	T/F	
NoKeyPrt	Partial inspect without key	T/F	C
NoKeyFul	Full inspect without key	T/F	
LEDtest	Front panel LED test	T/F	
CommsDis	Disable field writes	T/F	

Continued...

Parameter	Function	Units	Status
<i>Continued...</i>			
FullSave	Save full database to file	T/F	1
PartSave	Save partial database to file	T/F	2
BinSpd1	Binary comms speed selection	T/F	4
BinSpd2	Binary comms speed selection	T/F	8
Protectd	Database file is encrypted on save	T/F	1
NoBinSOH	Disable use of SOH (<i>absent in T640 v4/1 onwards</i>)	T/F	2
E2Form1		T/F	4
E2Form2		T/F	8
Status	Comms/hardware status	ABCD hex	
PwrFail	Power fail indicator	T/F	1
TmpPFail	Temporary power fail indicator	T/F	2
CommsAlm	Common block comms alarm	T/F	4
Alarm	Alarm relay tripped	T/F	8
PrtKey	Partial security key active	T/F	1
FulKey	Full security key active	T/F	2
BoardID1		T/F	4
BoardID2		T/F	8
S11 Mbus	DIP sw'bank 1, Sw1 status (ON=MODBUS, OFF=TCS binary)	T/F	1
S12 Xisb	DIP sw'bank 1, Sw2 status (ON=external ISB)	T/F	2
S13 Cold	DIP sw'bank 1, Sw3 status (ON=cold start)	T/F	4
S14 Warm	DIP sw'bank 1, Sw4 status (ON=warm start)	T/F	8
S15 UWdg	DIP sw'bank 1, Sw5 status (W/dog En. on UT)	T/F	1
S16 Def0	DIP sw'bank 1, Sw6 status (Bit 0, pre-conf. dbase)	T/F	2
S17 Def1	DIP sw'bank 1, Sw7 status (Bit 1, pre-conf. dbase)	T/F	4
S18 Def2	DIP sw'bank 1, Sw8 status (Bit 2, pre-conf. dbase)	T/F	8
MinRpt1-4	Minimum repeat, user tasks 1-4	secs.	
TaskHalt	User task start/stop (TRUE halts, FALSE restarts)	Bitfield	
UsrTask1	User task 1 control	T/F	
UsrTask2	User task 2 control	T/F	
UsrTask3	User task 3 control	T/F	
UsrTask4	User task 4 control	T/F	
AnConBlk	'Operator data' AN_CONN blockname	Alphanumeric	
DgConBlk	'Operator data' DG_CONN blockname	Alphanumeric	
Log_File	Number of closed log file in E ² ROM	(1-99)	

Table 39 Block parameters

Block specification menu

This information is given in addition to *Table 39*.

Dbase, Block, Type. See *Appendix D page 544* for details of these 'header' fields.

Time, Date. Provide local access to time and calendar information, which are required for alarm handling. Time and date are maintained between units on the peer-to-peer communications link (in the case of the T640 controller which has no Real-Time Clock hardware). If the Real-Time Clock is not initialised, the *RTCinit* alarm field sets.

IP_type. Bitfield specifying mains frequency and temperature system. It also indicates some hardware options fitted. Specifications apply to an entire T600 Series instrument and affect TCOUPLE, FULL_TC8, RTD, and UCHAR blocks, internal linearisation tables and cold-junction compensation. Units cannot be mixed in a single instrument, but different units on the LIN can be set differently.

- **60Hz.** Set TRUE for 60Hz, or FALSE for 50Hz mains operation. This sets the frequency for the noise-rejection circuitry in the I/O modules.
- **Imperial.** TRUE selects the Imperial temperature system - Fahrenheit (°F)/Rankine (R). FALSE selects the SI system - Celsius (°C)/Kelvin (K).
- **Big_RAM.** With v4/1 T640 software, *Big_RAM* is TRUE if 256K of RAM is fitted on the motherboard - i.e. enough to run sequences in these instruments. Older instruments need only 128K RAM to run sequences, and a TRUE bit indicates this is fitted. FALSE indicates insufficient RAM fitted for sequences. Fitting a sequence option memory module when *Big_RAM* is FALSE does not allow sequences to be run.

Site1 to Site8. These fields identify the internal serial bus module type assigned to each of the ISB I/O module addresses (1 - 8) on the internal serial communications bus.

BrownOut. Specifies the duration (mins) of a power interruption required to trip the *BrownOut* alarm. In practice, this can be defined as the maximum duration of power supply interruption that can be tolerated without an adverse reaction from the plant.

ColdStrt. Specifies the minimum duration (mins) of a power interruption that will cause a coldstart of a T600. After a coldstart, complete re-initialisation of the parameter database and the control strategy occurs. A 'warmstart' can occur after power interruptions less than *ColdStrt*; i.e. the current control strategy restarts with the existing parameter database and operating modes.

Setting *ColdStrt* to zero is a special case. Then, after a power interruption of any duration the instrument either coldstarts if coldstart is enabled, or warmstarts - and trips the *BrownOut* alarm - if not.

NOTE How the instrument starts up after a power interruption is determined by the settings of Switchbank 1, Switches 3 and 4 (Cold and Warm Start Enable, resp.) Both should be 'ON' to provide full warmstart and coldstart capability.

Node. The ALIN network address. This field is read-only, in the range 01 to FE hexadecimal.

Model. Numeric part of the T600 model identifier. E.g. '640' = T640 instrument.

Software version number. The instrument's software version number is automatically loaded into the *Alarms* parameter 'units' field at startup, and can be read by inspecting that field. (E.g. For a T640 the INS button collects the Alarm field, but the value is displayed in the Tag field after pressing the ALM button.) Note that the 'units' field is not transmitted over the LIN, and so cannot be seen using LINtools' VIEW facility.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block's RAM data.
- **BrownOut.** Indicates a power failure in excess of the time duration set in the *BrownOut* field. This alarm appears as an unacknowledged 'event' alarm, but has no acknowledged state.
- **ComS/W.** Indicates a common 'local' block software error.
- **UTskEr1 to UTskEr4.** Indicates that the corresponding user task (1 to 4) has a sumcheck error.
- **MainPSU.** Indicates that the main PSU has failed (*24V dc version only*).
- **StbyPSU.** Indicates that the standby PSU has failed (*24V dc version only*).
- **RTCinit.** Indicates that the Real-Time Clock has not been initialised.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

UsrAlm. This field is connectable to allow the control strategy to drive the instrument's watchdog relay as a user alarm. A TRUE input opens the normally-closed relay contacts. However, when Switchbank 1, Switch 5 is 'ON' (Watchdog Enable on User Task), *UsrAlm* is forced TRUE (i.e. contacts open) by a user task sumcheck error.

AreaNo. Security key area (0 - 31). Specifies the plant area in which this instrument is located. If *AreaNo* is in the range 1 - 31, a security key of the same area number is required to gain access to the front panel database INSPECT modes. If *AreaNo* is 0, any area number security key is accepted. (*At the current software issue only areas 0 to 8 are available.*)

TimeOut. Used in database INSPECT modes. If no front-panel pushbuttons are pressed for *TimeOut* seconds, the front panel reverts to its normal DISPLAY mode. Setting *TimeOut* to 0 disables this facility.

Options. Bitfield for selecting the T600 Series instrument front panel, communications, and database save options.

- **FPdis1 to FPdis4.** Setting these bits TRUE prohibits front-panel operator access to loops 1 to 4, respectively.

- **NoKeyPrt.** Setting this bit TRUE allows access to the front panel database ‘partial’ INSPECT modes without having to use the appropriate security key.

NOTE When using LINTools for configuration, by ‘right clicking’ in any block/field within the database, the field can be promoted to be viewed under ‘Partial Acces’ by selecting ‘Engineer Access’ from the drop down menu.
- **NoKeyFul.** Setting this bit TRUE allows access to the front panel database ‘full’ INSPECT modes without having to use the appropriate security key. Note that *NoKeyFul* overrides *NoKeyPrt* if both are TRUE, and that these two bits cannot be altered via the front-panel INSPECT mode if only partial access is in use.
- **LEDtest.** A TRUE input to this bit causes a front panel LED test to be performed.
- **CommsDis.** If TRUE this bit disables UNCONFIRMED field writes only. UNCONFIRMED field writes result from connections into cached blocks. Values written using LINTools while online and connected to the instrument are CONFIRMED field writes and are therefore unaffected by this bit. It does not disable writes to blocks cached in the *local* T600 Series instrument, and also prevents binary comms selects.
- **FullSave, PartSave.** Allow the current database (in RAM) to be saved to the same filename from which it was loaded. Making *FullSave* TRUE saves the entire RAM image. *PartSave* saves the RAM image excluding the ‘tepid (operator) data’, which retains the values in the existing file. *FullSave* overrides *PartSave* if both are TRUE.

NOTE Tepid data is saved once per user task iteration, and at power-down, and comprises the *SL*, *OP*, and *MODE* values for each PID loop, plus one AN_CONN and one DG_CONN block (specified by *AnConBlk* and *DgConBlk*, resp.). When a warmstart is attempted, if the RAM image proves to be invalid a coldstart is performed and the tepid data is overlaid. Note that a single TP_CONN block, named in *AnConBlk*, can specify tepid data as an alternative to using an AN_CONN and a DG_CONN block.
- **BinSpd1, BinSpd2.** Allow the line speed of the RS422 binary communications link to be set, according to *Table 40*.

BinSpd1	BinSpd2	Selected Baud Rate
FALSE	FALSE	9600
FALSE	TRUE	4800
TRUE	FALSE	1200
TRUE	TRUE	600

Table 40 Binary comms link Baud Rate selection

- **Protectd.** Allows this database to be protected by being saved in an encrypted format, provided that the T640’s memory module contains a special encryption code. *Protectd* cannot be set TRUE unless a valid code is present, and once set TRUE it cannot be reset FALSE (*except in T640-FF instruments*). With *Protectd* TRUE, saving the database using full or part save causes the .DBF file to be stored in an encrypted form, based on the value of the particular code present in the module.

Encrypted .DBF files can be unscrambled only by a T640 instrument having the correct encryption code in its memory module.
(*This feature is not supported by standard T640 instruments.*)
- **NoBinSOH.** A T640 instrument transmits the SOH control character (‘Start Of Header’) to throttle the binary comms, should the CPU loading on this particular T640 become excessive. SOH is interpreted as ‘extend timeout to 250ms’. The T640 also inserts a 40ms quiet period between the SOH transmission and the data characters that follow. For certain older supervisors that do not support the SOH character, set *NoBinSOH* to TRUE to disable SOH transmission. (*Absent in T640 v4/1 onwards.*)

NOTE Disabling SOH does not affect the 40ms quiet period.

- **E2Form1, E2Form2.** These two bits allow the E2ROM filing system to be completely reformatted, which may be required if it has been corrupted, e.g. by loss of power during a filing operation. To avoid accidental operation of this feature, the bits must be set and reset in a particular sequence before the reformat takes place. If any step is wrongly performed, all bits must be reset to FALSE and the sequence started again from step 1. The four steps in the sequence are:

- 1 Set *E2Form1* TRUE
- 2 Set *E2Form1* FALSE
- 3 Set *E2Form2* TRUE
- 4 Set *E2Form1* TRUE

As each step is carried out it is announced on the front panel with a corresponding flashing alarm message: **E2ROM 1** up to **E2ROM 4**. Only when the fourth message appears is the reformat executed. After successful reformatting, the four bits automatically reset FALSE, and the current .DBF and .RUN files are regenerated from RAM.

Status. Read-only bitfield indicating the T600 Series instrument's communications and hardware status.

- **PwrFail.** This bit latches TRUE whenever the instrument is powered up, and so acts as a power failure flag.
- **TmpPFail.** Indicates that a temporary power failure has occurred. The bit sets in the first iteration after power up of the user task containing the T600 block, then auto-resets in the second iteration. In a 'temporary' power failure some data is lost/corrupted and so the database file must be reloaded to RAM, but 'tepid data' is overlaid as well. (See the *Options* section above, under *FullSave, PartSave*, for details of 'tepid data'.)
- **CommsAlm.** This bit is set if *any* cached block within the unit is in software alarm due specifically to a communications failure (not a sumcheck error). The *CommsAlm* signal can therefore be used by a supervisory unit (e.g. Eycon™ 10/20 Visual Supervisor) to monitor the health of all inter-T600 communications, even when the affected blocks themselves are not visible from the Eycon™ 10/20 Visual Supervisor. It is only necessary to cache the communicating T600 blocks in the Eycon™ 10/20 Visual Supervisor, to make their *CommsAlm* bits accessible.
- **Alarm.** Indicates that the alarm relay is in operation, i.e. its contacts are *open*.
- **PrtKey, FulKey.** TRUE indicates that a valid security key is currently being used with the front panel in 'partial' and 'full' inspection modes, respectively.
- **BoardID1 to BoardID2.** These two bits indicate the CPU clock frequency on the T600 instrument's motherboard. At the current issue of hardware the only available version is 12.5MHz, indicated by both bits being FALSE.
- **S11 to S18.** Indicate status of T600 Series instrument's Switchbank 1, Switches 1 to 8, respectively. TRUE = On, FALSE = Off.

MinRpt1 to MinRpt4. These fields allow the minimum desired repeat rate for tasks 1 to 4, respectively, to be specified to within 10 ms. When the server initialises, it does not try to execute the task *more* frequently than the specified interval, but it may execute it *less* frequently if there is insufficient CPU time available. Setting these fields to zero makes the system select the minimum possible values, i.e. the fastest possible repeat rates.

NOTE Refer to the T640 Reference Manual and User Guide (Part no. HA082468), User Task Tuning section.

TaskHalt. This bitfield allows the user tasks to be stopped and started by the control strategy.

- **UsrTask1 to UsrTask4.** When the input is TRUE the corresponding user task (1 to 4 respectively) is halted. The task restarts when the input returns to FALSE. Note that you cannot halt the user task containing the T600 block.

AnConBlk. This field can be used in two ways. It can specify the tagname (*Block* field) of a single AN_CONN block that is to contain 'tepid data', and so have all its configuration data saved at power down. Alternatively it can specify the tagname of a TP_CONN block, which replaces both an AN_CONN and a DG_CONN block for this purpose. In this case entering the TP_CONN block name in *AnConBlk* automatically clears the *DgConBlk* field.

DgConBlk. Specifies the tagname (*Block* field) of the single DG_CONN block that is to contain digital 'tepid data', and so have all its configuration data saved at power down. Note that if a TP_CONN block has been named in *AnConBlk*,

which replaces both an AN_CONN and a DG_CONN block for this purpose, the *DgConBlk* field automatically clears and cannot be written to.

Log_File. Whenever parameters are altered via the front panel database INSPECT modes, all such changes are recorded in a logfile in the T600 Series instrument's E2ROM. The log's filename has the same root as the .DBF file from which the strategy was loaded, but with extension *.Lnn*, where *nn* ranges from 01 to 99. '*nn*' increments as each logfile fills and closes.

The *Log_File* field shows the number (*nn*) of the latest logfile that has been closed, and is ready to be uploaded to a supervisor if required.

NOTE Only two logfiles are retained in E2ROM: the closed file specified in *Log_File*, and the subsequent open file which is currently being written to. While the current logfile is being filled - which takes at least one minute - the previous (closed) logfile is available for uploading. But as soon as the current file fills and closes, the value in *Log_File* increments and the previous file is overwritten. A zero value in *Log_File* means that there are no valid logfiles in E2ROM.

Parameter	Function	Units	Status
<i>Continued...</i>			
Options	Comms/hardware status	(A)BCD hex	
CommsDis	TRUE disables incoming ELIN/ALIN field writes	T/F	D
Protectd	TRUE encrypts/deciphers databases on save/load	T/F	
CONFspd	TRUE=Configurator mode, FALSE=Run mode	T/F	
SaveBkUp	Save full database to file	T/F	
SaveDBF	Save partial database to file	T/F	C
ForceDmp	TRUE generate dump file	T/F	
StallMB1	TRUE causes Modbus line 1 to hold dbase till polled	T/F	B
StallMB2	TRUE causes Modbus line 2 to hold dbase till polled	T/F	
StallMB3	TRUE causes Modbus line 3 to hold dbase till polled	T/F	A
StallMB4	TRUE causes Modbus line 4 to hold dbase till polled	T/F	
StallPB1	TRUE causes Profibus line 1 to hold dbase till polled	T/F	
StallPB2	TRUE causes Profibus line 2 to hold dbase till polled	T/F	
AllSubnt	TRUE permits unrestricted ELIN comms	T/F	
CldStPri	This T940(X) is the cold start primary CPU	T/F	
TaskRpt1 - 4	Minimum repeat task 1-4 (implemented V4/1 & T940X)	Secs	
TaskHalt	User task start/stop(implemented V4/1 & T940X)	(ABC)D hex	
UsrTask1	User task 1 control	T/F	
UsrTask2	User task 2 control	T/F	
UsrTask3	User task 3 control	T/F	
UsrTask4	User task 4 control	T/F	
TaskPri1 - 4	Displays relative priority of user task 1 - 4, resp. 1 - Highest priority, to 4 - Lowest priority (implemented V4/1)	Integer	

Table 41 Block parameters

Block specification menu

Dbase, Block, Type. See Appendix D page 544 for details of these ‘header’ fields.

Time, Date. Independent Real-Time Clock and calendar, respectively. These fields can be used to change the time and date in the instrument while the system is running ‘on-line’.

IP_type. Bitfield specifying mains frequency and temperature system/units. Specifications apply to an entire instrument and affect internal linearisation tables and cold-junction compensation. Units cannot be mixed in a single T940(X), but different T940(X)s on the ELIN/ALIN can be set differently.

- **60Hz.** Set TRUE for 60Hz, or FALSE for 50Hz mains operation.
- **Imperial.** TRUE selects the Imperial temperature system - Fahrenheit (°F)/Rankine (R). FALSE selects the SI system - Celsius (°C)/Kelvin (K).
- **Absolute.** TRUE selects absolute temperature units - Kelvin/Rankine. FALSE selects relative units - Celsius/Fahrenheit.

ColdStrt. Specifies the minimum duration (mins) of a power interruption that will cause a cold startup. A ‘warmstart’ occurs after power interruptions less than *ColdStrt*; i.e. the current control strategy restarts with the existing parameter database and operating modes. After a coldstart, complete re-initialisation of the parameter database occurs and the control strategy specified in the .RUN extension file is loaded and run (if possible).

When a database file, *dbase.DBF*, is downloaded from a remote node or loaded using the in-built terminal configurator, a *dbase.RUN* file is created, and all other files with the .RUN extension are deleted. On coldstart, the filing system is examined for a file with the .RUN extension. If found, the corresponding database file is loaded and run. If that fails, the instrument does not start its runtime.

BrownOut. Specifies the duration (mins) of a power interruption required to trip the *BrownOut* alarm.

Relays. Inputs to the bits in this field control the energisation of the three relays. The *WdogRly* bit overrides the normal setting of the watchdog relay (i.e. energised/closed = ‘healthy’, de-energised/open = ‘database fault’) **and should be used for test purposes only.**

NOTE The watchdog LED on the front panel works totally independently of the *WdogRly* setting.

Alarms. See *Appendix D* page 545 for a general description of the Alarms field.

- **Software.** Sumcheck error in block's RAM data.
- **BrownOut.** Indicates a power failure in excess of the time duration set in the *BrownOut* field. This alarm appears as an unacknowledged 'event' alarm, but has no acknowledged state.
- **ComS/W.** Sets if a software alarm occurs in any 'local' block in the database.
- **PSU.** Power loss of either A or B power supplies.
- **OverTemp.** Sets if the internal sensor detects a temperature high enough to damage or upset the correct functioning of the electronics.
- **ExtBat.** Sets if the external battery is absent or defective.
- **IntBat.** Sets if the internal rechargeable battery is absent or defective.
- **MainFan.** Sets if the stall detector on the enclosure-mounted fan signals failure.
- **CPUFan.** Sets if the stall detector on the CPU-mounted fan signals failure.
- **Chngovr.** Sets if there has been a changeover of primary controller in a redundant system.
- **I/OComms.** Sets if one or more of the I/O comms mechanisms has failed.
- **WdogLoom.** Sets if the watchdog control wiring loom is disconnected from the CPU, e.g. after servicing the unit. *If this alarm trips, return the unit to the factory for inspection.*
- **CPFFail.** Indicates a failure of the coldstart parameter file to execute. This file contains parameter values to be used in the event of a cold start.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

Features. (*This bitfield is currently unused*).

Node. The ELIN/ALIN node address of the instrument, as read from the T940(X) address switches at run-time.

Status. Bitfield indicating the T940(X) communications and hardware status.

- **ColdStrt.** TRUE indicates that the T940(X)'s front-panel rotary 'Restart' switch is set to either the 'cold' or the 'hot/cold' position.
- **HotStrt.** TRUE indicates that the 'Restart' switch is set to either the 'hot' or the 'hot/cold' position.
NOTE From the four possible TRUE/FALSE combinations of these two bits you can deduce the setting of the Restart switch, e.g. both bits TRUE indicates the 'hot/cold' position.
- **ExtBat.** TRUE indicates a missing or defective external battery.
- **IntBat.** TRUE indicates a missing or defective internal rechargeable battery.
- **Red.** TRUE if the T940(X) is a redundant machine.
- **MainFan.** TRUE if the T940(X) main enclosure-mounted fan has failed.
- **CPUFan.** TRUE if the T940(X) CPU-mounted fan has failed.
- **WdogLoom.** TRUE if the watchdog control wiring loom is disconnected from the CPU, see *Alarms* section.
- **Chngovr.** TRUE if there has been a changeover of primary controller in a redundant system.
- **CommsAlm.** This bit is set if *any* cached block within the T940(X) unit is in software alarm due specifically to a communications failure, not a sumcheck error. The *CommsAlm* signal can therefore be used by a supervisory instrument to monitor the health of all inter-T940(X) communications, even when the affected blocks themselves are not visible from the supervisor. It is only necessary to cache the communicating *T940(X)* blocks in the supervisor, to make their *CommsAlm* bits accessible.

This bit can be used in conjunction with the *ComS/W* alarm to determine if either a comms failure OR a sumcheck failure has occurred.

- **PwrFail.** Set if the database was started by a power-up (rather than by a user request, e.g. via the terminal configurator or a ELIN/ALIN remote command). Note that *PwrFail* is user-resettable.

- **TmpPFail.** This bit sets in the same way as *PwrFail*, but it automatically resets at the second database iteration.

Options. Bitfield setting certain communications and hardware status options.

- **CommsDis.** If TRUE this bit disables UNCONFIRMED field writes only. UNCONFIRMED field writes result from connections into cached blocks. Values written using LINTools while online and connected to the instrument are CONFIRMED field writes and are therefore unaffected by this bit.
- **Protectd.** TRUE causes database strategies to be ‘scrambled’ when saved and ‘un-scrambled’ when loaded. (*This feature is not currently supported.*)
- **CONFspd.** Selects configurator speed. TRUE selects ‘Configurator mode’. FALSE selects ‘Run mode’. Configurator mode causes control to operate at reduced efficiency (~85%) but remain unaffected when the resident configurator is run. Run mode lets control operate at maximum efficiency while the configurator is not running, but slows it down to ~53% whenever the configurator is run.
- **SaveDBF, SaveBkUp.** Allow the current database (in RAM) to be saved to the same filename from which it was loaded. Making *SaveBkUp* TRUE saves the entire RAM image as a .ubd file. *SaveDBF* saves the RAM image excluding the ‘tepid (operator) data’, that retains the values in the existing file. *SaveBkUp* overrides *SaveDBF* if both are TRUE.

NOTE Tepid data is saved once per user task iteration, and at power-down. It comprises the *SL*, *OP*, and *MODE* values for each PID loop, plus any user-specified parameters, as listed in the .tpf file (.tpf file is a file associated by filename, with the running database file). .tpf file parameters are specified by one *block.field* type string per line. The tepid start is attempted when a hotstart cannot be performed and consists of a coldstart and a tepid data overlay.

- **ForceDmp.** If TRUE, a dump file of the current configuration of this instrument will generated.
- **StallMB1 to StallMB4.** At database startup, TRUE causes the corresponding Modbus comms line (1-4) to poll all parameters and update them before allowing the database to start running. This ensures, for example, that all I/O values are up-to-date. These bits also operate in redundant systems at changeover, when the new primary takes over. They must be set for redundant changeover to function correctly.
- **StallPB1 to StallPB2.** These bits work as for the *StallMBn* bits (see previous), but with the corresponding Profibus comms lines (1-2).

AllSubnt. If TRUE, it permits unrestricted communications with all ELIN addresses on the network, if set FALSE this instrument will only communicate with ELIN addresses on the same logical subnet.

CldStPri. TRUE indicates that this T940(X) CPU is the left hand processor module, FALSE indicates that it is the right hand processor module.

TaskRpt1 to TaskRpt4. These fields become active when this block is used in a T940X. The fields allow the minimum desired repeat rate for tasks 1 to 4, respectively, to be specified to within 10 ms. When the server initialises, it does not try to execute the task *more* frequently than the specified interval, but it may execute it *less* frequently if there is insufficient CPU time available. Setting these fields to zero makes the system select the minimum possible values, i.e. the fastest possible repeat rates, see *T940X Process Supervisor Handbook (Part no. HA028225), User Task Tuning section*.

TaskHalt. This field become active when this block is used in a T940X. This bitfield allows the user tasks to be stopped and started by the control strategy.

- **Task_1 to Task_4.** When the input is TRUE the corresponding user task (1 to 4 respectively) is halted. The task restarts when the input returns to FALSE.

NOTE The user task containing the header block cannot be halted.

T225: T225(X) CONFIGURATION BLOCK

Block function

The T225 block must appear in the control strategy to be executed in the T225 or T225X. It allows access to the unit's Real-Time Clock and calendar, as well as indicating network addresses, ELIN/ALIN status and relay states.

Block parameters

Symbols used in *Table 42* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Time	Real-Time Clock time of day	hh:mm:ss	
Date	Real-Time Clock calendar	dd/mm/yy	
Options	Operational tests	(ABC)D hex	
LEDTest*	TRUE = test front panel LEDs	T/F	
WDTest*	TRUE = test Watchdog	T/F	
ForceDmp*	TRUE = test generate dump file	T/F	
EnaLog*	TRUE = enable Log file	T/F	
ELINAddr	ELIN address ABCD, AB=segment; CD=MAC	ABCD hex	
ALINAddr	ALIN address ABCD, AB=segment; CD=MAC	ABCD hex	
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
CommsAlm	Cached block comms failure	T/F	
MainPSU	Main PSU failure	T/F	
Battery	Battery failure	T/F	
RCTinit	Real-Time Clock not initialised	T/F	
HighTemp	ProcTemp >80C or CardTemp >60C	T/F	
ELIN	Comms failure on ELIN	T/F	
ALIN	Comms failure on ALIN	T/F	
CPFFail	Coldstart Parameter File execution failure	T/F	
EventLog	EventLog file entry has occurred	T/F	
Combined	OR-ing of all Alarms bits	T/F	
Status	Comms/hardware status	ABCD hex	
PwrFail	Set on power-up (can be cleared by user)	T/F	D
TmpPFail	Set on power-up (self-clears after 1 scan)	T/F	
CommsAlm	At least 1 cached block is not updating	T/F	
		T/F	
		T/F	C
ELINFail	ELIN (Out1) status (TRUE = fail)	T/F	
ALINFail	ALIN (Out2) status (TRUE = fail)	T/F	
		T/F	
		T/F	B
FwdRly	FWD relay state (TRUE = closed, i.e. healthy)	T/F	
WdogRst	TRUE = System reset by Watchdog failure	T/F	
BattLow	TRUE = Battery voltage low	T/F	
		T/F	A
BattHigh	TRUE = Battery voltage high	T/F	
In1	Input 1 (ETH1) status	T/F	
In2	Input 2 (ETH2) status (not supported)	T/F	
		T/F	
CPFile	TRUE = Coldstart Parameter File read	T/F	
EventLog	EventLog file entry has occurred	T/F	
		T/F	
PSU_Stat	Power Supply status	ABCD hex	
VcoreHi	Not used, always zero (0 - FALSE)	T/F	D
VcoreLo	Not used, always zero (0 - FALSE)	T/F	
3V3Hi	TRUE = 3.3v Supply > 3.4	T/F	
3V3Lo	TRUE = 3.3v supply < 3.2	T/F	
		T/F	C
5VHi	TRUE = 5V supply > 5.2	T/F	
5VLo	TRUE = 5V suppl < 4.8	T/F	
12VHi	TRUE - 12V supply > 12.8	T/F	
		T/F	B
12VLo	TRUE = 12V supply < 11.5	T/F	
m5VHi	Not used, always zero (0 - FALSE)	T/F	
m5VLo	Not used, always zero (0 - FALSE)	T/F	
		T/F	B
m12vHi	TRUE = -12V supply < -12.8	T/F	
m12vLo	TRUE = -12V supply > -11.5	T/F	

Continued...

Parameter	Function	Units	Status
<i>Continued...</i>			
Switches	Motherboard switchbank settings (TRUE = 'on')	ABCD hex	
S1 to S16	Always zero (0 - FALSE)	T/F	
ProcTemp			
CardTemp	Printed Circuit Board temperature	Deg C	

* *Manufacturers use only.*

Table 42 Block parameters

Block specification menu

This information is given in addition to *Table 42*.

Dbase, Block, Type. See *Appendix D page 544* for details of these 'header' fields.

Time, Date. Independent Real-Time Clock and calendar, in the instrument. If required, these fields can be set by the user after power-up.

Options. Bitfield setting specific communications and hardware status options.

- **LEDtest.** Manufacturers use only. A TRUE input to this bit causes a front panel LED test to be performed.
- **WDTest.** Manufacturers use only. A TRUE input to this bit causes a Watchdog test to be performed.
- **ForceDmp.** T225X only. A TRUE input to this bit is used to generate a dump file of the current configuration of this instrument.
- **EnaLog.** T225X only. A TRUE input to this bit enables event logging to the EventLog file, .udz.

ELINAddr, ALINAddr. This shows the address of this instrument on the Local Instrument Network using Ethernet, ELIN and using ArcNet, ALIN.

NOTE This *ELINAddr* and the *ALINAddr* fields will display an identical value.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block's RAM data.
- **CommsAlm.** Asserted if *any* cached block within the unit is in software alarm due specifically to a communications failure, not a sumcheck error. The *CommsAlm* signal can therefore be used by a supervisory unit to monitor the health of all communications, even when the affected blocks themselves are not visible from the unit. It is only necessary to cache the communicating blocks, to make their *CommsAlm* bits accessible.
- **MainPSU.** T225 only. Asserted if the main PSU has failed.
- **Battery.** T225 only. Asserted if battery voltage is less than 2.7V. (Default alarm priority = 10.)
- **RTCinit.** Asserted if the Real-Time Clock has not been initialised.
- **HighTemp.** T225 only. Asserted if the internal sensor detects a temperature high enough to damage or upset the correct functioning of the electronics. This is defined as the Processor temperature exceeding 80°C or the Card temperature exceeding 60°C.
- **ELIN.** Asserted if a communications failure is detected on the ELIN set via the *Status.ELINFail*.
- **ALIN.** Asserted if a communications failure is detected on the ALIN set via the *Status.ALINFail*.
- **CPFFail.** Asserted if a failure of the coldstart parameter file to execute has occurred. The .cpf contains parameter values to be used in the event of a coldstart. (Default alarm priority = 4, but can be set to 0 (zero) if the .cpf file is not used.)
- **EventLog.** T225X only. Asserted if a severity 'error' or 'Major error' has been written to the EventLog file, .udz. This indicates further investigation is required because an unexpected problem in the execution of the instrument has occurred. (Default alarm priority = 4.)
- **Combined.** Asserted if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

Status. Bitfield indicating the unit's communications and hardware status.

- **PwrFail.** TRUE if the database was started by a power-up (rather than by a user request, e.g. via the terminal configurator or a ELIN/ALIN remote command).
NOTE *PwrFail* is user-resettable.
- **TmpPFail.** This bit sets in the same way as *PwrFail*, but it automatically resets at the second database iteration.
- **CommsAlm.** TRUE if *any* cached block within the unit is in software alarm due specifically to a communications failure, not a sumcheck error. The *CommsAlm* signal can therefore be used by a supervisory unit to monitor the health of all communications, even when the affected blocks themselves are not visible from the supervisor. It is only necessary to cache the communicating blocks, to make their *CommsAlm* bits accessible.

This bit can be used in conjunction with the *ComS/W* alarm to determine if either a comms failure OR a sumcheck failure has occurred.

- **ELINFail.** TRUE if a communications failure is detected on the ELIN, also sets *Alarms.ELIN* TRUE.
- **ALINFail.** TRUE if a communications failure is detected on the ALIN, also sets *Alarms.ALIN* TRUE.
- **FwdRly.** TRUE by default. FALSE if the Forwarding relay is open. The normal operation of the Forwarding relay is TRUE = energised/closed = 'healthy', FALSE = de-energised/open = 'database fault', **and should be used for test purposes only.**
- **WdogRst.** T225 only. TRUE if the last system reset was generated by watchdog failure.
- **BattLow.** T225 only. TRUE if the on board battery voltage reads less than 2.7V.
- **BattHigh.** T225 only. TRUE if the on board battery voltage reads greater than 3.2V.
- **In1.** Shows the current status of the ETH1 connector.
- **In2.** Shows the current status of the ETH2 connector.
- **CPFile.** TRUE if a Coldstart Parameter File (.cpf) has been found.
- **EventLog.** T225X only. TRUE indicates a severity 'error' or 'Major error' has been written to the EventLog file, .udz. This indicates further investigation is required because an unexpected problem in the execution of the instrument has occurred. Wiring to this field can allow it to be reset.

PSU_Stat. T225 only. Power Supply status

- **VcoreHi.** Not used, always zero (0 - FALSE)
- **VcoreLo.** Not used, always zero (0 - FALSE)
- **3V3Hi.** TRUE if the 3.3V Power Supply greater than 3.4V.
- **3V3Lo.** TRUE if the 3.3V Power Supply less than 3.2V.
- **5VHi.** TRUE if the 5V Power Supply greater than 5.2V.
- **5VLo.** TRUE if the 5V Power Supply less than 4.8V.
- **12VHi.** TRUE if the 12V Power Supply greater than 12.8V.
- **12VLo.** TRUE if the 12V Power Supply less than 11.8V.
- **m5VHi.** Not used, always zero (0 - FALSE)
- **m5VLo.** Not used, always zero (0 - FALSE)
- **m12vHi.** TRUE if the -12V Power Supply greater than -12.8V.
- **m12vLo.** TRUE if the -12V Power Supply less than -11.8V.

Switches. Motherboard switchbank settings (TRUE = 'on')

- **S1 to S16.** Always zero (0 - FALSE)

ProcTemp. T225 only. This shows the current temperature of the processor. If this value exceeds 80°C the *Alarms.HighTemp* is set TRUE.

CardTemp. T225 only. This shows the current temperature of the Printed Circuit Board (PCB). If this value exceeds 60°C the *Alarms.HighTemp* is set TRUE.

TACTICIAN: TACTICIAN CONFIGURATION BLOCK

Block function

This block must appear in a strategy to be executed in a Tactician series instrument. It allows access to the base unit's Real-Time Clock (independently configured for each unit), specifies temperature units for a single unit, specifies a coldstart (system reset) time and contains a brown-out alarm that latches on when the power is interrupted for a specified time. The block also indicates a variety of communications/hardware settings, and allows the selection of various user options

NOTE. This Configuration block is used for the Tactician Series instruments. The specific instrument type is defined using the Model parameter.

Block parameters

Symbols used in *Table 43* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Time	Real-Time Clock time of day	hh:mm:ss	
Date	Real-Time Clock calendar	dd/mm/yy	
IP_type	Mains line frequency, temperature units	(ABC)D hex	
60Hz	Mains frequency (TRUE = 60Hz, FALSE = 50Hz)	T/F	D
Imperial	Temp. system (TRUE = Imperial, FALSE = SI)	T/F	
Absolute	Temp. units (TRUE = Absolute, FALSE = Relative)	T/F	
			8
IO_Mode	Operating mode control	(ABC)D hex	
EnaTest	Enable Test mode	T/F	D
EnaSim	Enable Simulate mode	T/F	
HoldOut	Hold electrical output values if strategy is stopped	T/F	
			8
BrownOut	Power-off time to trip BrownOut alarm	Mins	
ColdStrt	Min. power-off time to trigger cold start on powerup	Mins	
Model	Tactician Series instrument/slot variant base unit	Menu	 *
Alarms			  
Software	Block RAM data sumcheck error / network failure	T/F	
BrownOut	Power failure longer than BrownOut (default priority 10)	T/F	
ComS/W	Common 'local' block software error	T/F	
Sumcheck	Sumcheck error (default priority 10)	T/F	
NodeSw	LIN Node address changed detected	T/F	
CPFail	Coldstart parameter file execution failure	T/F	
ComH/W	Hardware failure (absent or defective)	T/F	
BattLow	Battery supply low	T/F	
BattFail	Battery failure (absent or defective)	T/F	
BattFail	Battery failure (absent or defective)	T/F	
RTCfail	Real-time clock failure	T/F	
PLicence	Licence exceeded	T/F	
**SLicence	Licence exceeded on Secondary controller	T/F	
SlowTask	User task repeat period failure	T/F	
EventLog	EventLog file entry has occurred	T/F	
ResetOfI		T/F	
Combined	OR-ing of all Alarms bits	T/F	
Node	LIN node address, read from source	(01-FE hex)	
Cluster	Name to describe a group of instruments	Menu	
Options	Comms/hardware status	(ABC)D hex	
CommsDis	TRUE disables incoming LIN field writes	T/F	D
AllSubnt	TRUE permits unrestricted ELIN comms	T/F	
ForceDmp	TRUE generate dump file.	T/F	
SaveBkUp	Save full database to file	T/F	8
SaveDBF	Save partial database to file	T/F	C
BattFit	TRUE starts battery alarms. FALSE battery not requested	T/F	
DISPtest	Illuminates all LEDs under software control	T/F	
CONFspd	TRUE=Configurator mode, FALSE=Run mode	T/F	8

Continued...

Parameter	Function	Units	Status
<i>Continued...</i>			
CachdPEs	TRUE enables display of cached block field values	T/F	B
UnSupAll	TRUE Unsuppresses all local Suppressed Alarms	T/F	
UsrAlm	Controls the watchdog relay (T2750 only)	T/F	
GwDcmEmu	TRUE enables gateway to operate in <i>DCM mode</i>	T/F	
Status	Comms/hardware status	AB(C)D hex	
ColdStrt	TRUE = SW2:S2 off, SW2:S3 on (Cold start enable)	T/F	D
HotStrt	TRUE = SW2:S2 on, SW2:S3 off (Hot start enable)	T/F	
DmpInPrg	TRUE = executing ForceDmp	T/F	
CommsAlm	Cached block comms failure	T/F	
EventLog	EventLog file entry has occurred	T/F	C
MultiSFC	TRUE = database is a multi-task sfc database	T/F	
P1PwFail	TRUE = PSU connected to terminal P1 has failed	T/F	
P2PwFail	TRUE = PSU connected to terminal P2 has failed	T/F	
FilSysEr	TRUE = SD card is in quarantine mode	T/F	B
OtherCPU	TRUE = Second processor slot used	T/F	
**SavActive	Suspend 'SaveBkUp' or 'SaveDBF'	T/F	
**CldStPri	This is the cold start primary controller	T/F	
**Red	This is a redundant machine	T/F	A
**Chngovr	A changeover has occurred in a redundant system	T/F	
PwrFail	Database started by power-up (user-resettable)	T/F	
TmpPFail	As PwrFail, but auto-resets on 2nd dbase iteration	T/F	

** Redundant systems only

Table 43 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these 'header' fields.

Time, Date. Independent Real-Time Clock and calendar, respectively. These fields can be used to change the time and date in the instrument while the system is running 'on-line'.

NOTE This cannot be amended if the instrument's real time clock is enslaved.

IP_type. Bitfield specifying mains frequency and temperature system/units. Specifications apply to an entire Tactician and affect internal linearisation tables and cold-junction compensation. Units cannot be mixed in a single Tactician, but each Tactician on the LIN can be set differently.

- **60Hz.** Set TRUE for 60Hz, or FALSE for 50Hz mains operation.
- **Imperial.** TRUE selects the Imperial temperature system - Fahrenheit (°F)/Rankine (R). FALSE selects the SI system - Celsius (°C)/Kelvin (K).
- **Absolute.** TRUE selects absolute temperature units - Kelvin/Rankine. FALSE selects relative units - Celsius/Fahrenheit.

IO_Mode. 16-bit subfield providing control for the Database operating mode.

- **EnaTest.** Set TRUE to allow Test mode on other I/O blocks to be set. When *IO_mode.EnaTest* is TRUE on the TACTICIAN block, then the channels on other I/O modules can be set to TEST. When the *IO_Mode.EnaTest* field on the TACTICIAN block is subsequently cleared, all channel blocks in TEST mode revert to their fallback.
- **EnaSim.** Set TRUE to allow Simulate mode on other I/O blocks to be set. When *IO_Mode.EnaSim* is TRUE on the TACTICIAN block, then the channels on other I/O modules can be set to SIMULATE. When the *IO_Mode.EnaTest* field on the TACTICIAN block is subsequently cleared, all channel blocks in SIMULATE mode revert to their fallback.
- **HoldOut.** Set FALSE (default) to reset the current electrical output values to 0 (zero), or TRUE, to hold the current electrical output values, if the strategy is stopped.

BrownOut. Specifies the duration (mins) of a power interruption required to set *Alarms.BrownOut* TRUE.

NOTE A brown-out alarm is only set when the database is configured to Hotstart.

ColdStrt. Specifies the minimum duration (mins) of a power interruption that will cause a coldstart of the instrument when the *ColdStart* hardware switches (Options - SW2:S1 and SW:S2) are set. When a LIN Database file, *dbase.dbf*, is downloaded from a remote node or loaded using the Tactician's local terminal, a *dbase.run* file is created, and all other files with the .run extension are deleted. On coldstart, the Tactician searches its filing system for a .run extension file. If found, the corresponding *dbase.dbf* is loaded and run. If that fails, the Tactician does not start its runtime.

NOTE If hardware switches (Options - SW2) are set to Off then an auto-generated database is created and run.

Model. (Unknown/T2550-*/T820/T830/T840/T860/T870). Instrument type and the type of base unit, defined by the number of slots if applicable, e.g. T2550-16 indicates a T2550 instrument in a 16 slot base.

NOTE (*) is the total number of module slots in the base unit.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block's RAM data.
- **BrownOut.** Asserted if a power failure in excess of the time duration set in the *BrownOut* field has occurred, and appears as an unacknowledged 'event' alarm, without an acknowledged state.
- **ComS/W.** Indicates a 'local' software alarm occurring in any block in the database.
- **Sumcheck.** Asserted in conjunction with *ComS/W* if a sumcheck alarm occurs in any block in the running database.
- **NodeSw.** Asserted if the Node address switches have altered since the unit was powered up.

NOTE This Alarm field has no effect when used on the T2550 instrument.
- **CPFfail.** Asserted if a failure of the coldstart parameter file to execute has occurred. The .cpf contains parameter values to be used in the event of a cold start.
- **ComH/W.** Asserted if a hardware alarm occurs in this instrument. This alarm is active if any of the following are TRUE:
 - One or more of the PSUs have failed
 - A Status of *Missing*, *BadType*, *BadSite* or *BadTask* is TRUE on a MOD_UIO block
 - A Status of *Missing*, *BadType*, *BadSite*, *BadTask*, *BadSetup*, or *OverTemp* is TRUE, or *HwFlt* is non-zero on a MOD_DI_UIO or MOD_DO_UIO block
 - A Status of *HwFlt*, *HwAuxFlt*, *BadTask* or *OverTemp* is TRUE on an AI_UIO, AO_UIO, DI_UIO, DO_UIO, VP_UIO, TP_UIO or FI_UIO block.
 - The file system error status bit in this block, *Status.FilSysEr*, is TRUE.
- **BattLow.** Asserted if the battery supply is below the required voltage or defective.
- **BattFail.** Asserted if the battery has failed.
- **RTCFail.** Asserted if a problem with the Real-Time Clock is detected.
- **PLicence.** Asserted if the current configuration exceeds the licence levels of this product. This priority 15 alarm cannot be disabled, or acknowledged when active.
- **SLicence.** (Redundant system only) Asserted if the current configuration exceeds the licence levels of this product on the Secondary unit in a 2 processor redundant system. This priority 15 alarm cannot be disabled, or acknowledged when active.
- **SlowTask.** Asserted if User Tasks are running slower than the configuration engineer requested.
- **EventLog.** Asserted if a severity 'error' or 'Major error' has been written to the EventLog file, .udz. This indicates further investigation is required because an unexpected problem in the execution of the instrument has occurred. The *EventLog* alarm follows the state of *Status.EventLog* (TRUE = in alarm, FALSE = out of alarm). Write to the status word or *Status.EventLog* bit to clear the bit to get a new alarm for the next important event log entry.
- **ResetOfI.** Asserted if the maximum of 2560 parameters in the Reset Data Set supported by the Cold Start Parameter file (.cpf) has been exceeded, or if the .cpf file is missing. Of these 2560 parameters, 3 are required for Date, Time and Checksum, and are used to validate the data.

- **Combined.** Asserted if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

Node. The LIN node address of the instrument, as read from the instrument address switches at runtime.

Cluster. The name of the group of instruments of which this instrument is a part of.

NOTE If the instrument is not part of a cluster this field will remain blank.

Options. Bitfield setting Tactician communications and hardware status options.

- **CommsDis.** If TRUE, this bit disables UNCONFIRMED field writes only. UNCONFIRMED field writes result from connections into cached blocks. Values written using LINTools while online and connected to the instrument are CONFIRMED field writes and are therefore unaffected by this bit.
- **AllSubnt.** If TRUE, it permits unrestricted communications with all ELIN addresses on the network, if set FALSE this instrument will only communicate with ELIN addresses on the same logical subnet.
- **ForceDmp.** If TRUE, a dump file of the current configuration of this instrument will generated.
- **SaveBkUp, SaveDBF.** If TRUE, this allows the current database (in RAM) to be saved to the same filename from that it was loaded. Making *SaveBkUp* TRUE saves the entire RAM image as a .ubd file. *SaveDBF* saves the RAM image excluding the 'Reset (operator) data', that retains the values in the existing file. *SaveBkUp* overrides *SaveDBF* if both are TRUE.

NOTE Reset data comprises Setpoint, Output and Mode for each PID loop, plus any user-specified parameters, as listed in the Coldstart Parameter File, .cpf, using the running Database filename and a .cpf extension. The .cpf file consists of a list of coldstart parameters and values and a Reset data overlay and is used if a hotstart fails. .cpf parameters set in the file are specified by one *block.field* type string per line.

- **BattFit.** If TRUE, this bit activates the *Alarms.BattLow* and *Alarms.BattFail* and illuminates the relevant battery status LED's. If set FALSE, it is assumed that the battery is deliberately not fitted and has no effect on the battery alarms and LED's.
- **DISPtest.** If TRUE, this bit illuminates all LEDs on the instrument that are under software control.

NOTE The 'Ethernet (speed)' and 'Ethernet (activity)' LEDs are excluded in the T2550 instrument, and the Secondary processor LEDs will only illuminate if the processors are synchronised

- **CONFspd.** Selects configurator speed. If TRUE, this bit selects 'Configurator mode'; FALSE selects 'Run mode'. In Configurator mode, more processing time for the resident Configurator or FTP is allocated. This mode may cause the strategy to run slower than usual, because a larger amount of CPU is allowed for the Configurator or FTP. Setting the Configurator mode has no effect on the strategy if sufficient CPU already exists. In Run mode, the strategy operates at maximum efficiency.

IMPORTANT *Beware if temporarily changing the Options.CONFspd bit, it will become permanent if the control strategy is saved. If saved, the Options.CONFspd bit can be reset, but any following Cold Start will reload the LIN Database containing the saved value.*

- **CachdPEs** If TRUE, the instrument (T2550/T820 V5.0 or later) will display all information collected from the block in the remote instrument (only if this instrument also supports the *CachdPEs*), including units, text and field name references.
- **UnSupAll.** When set TRUE it will unsuppress all alarms in the instrument's local database regardless of whether they have been set to indefinite or with a time duration. The UnSupAll bit will then auto-reset to FALSE as it only reacts to a rising edge. If held TRUE, Alarm Suppression will continue to function as normal until the next rising edge. Wiring out from the UnSupAll bit serves no purpose.
- **UsrAlm.** Controls the state of the watchdog relay (T2750 only). When the instrument is powered off, or is powered and no strategy is running, the watchdog relay is open (alarm state). When a strategy is running, the relay is driven according to the state of this bit. A value of FALSE implies no alarm (the relay is energised and the contacts closed). A value of TRUE implies an alarm state, and opens the relay.
- **GwDcmEmu.** Profibus Gateway DCM emulation mode. This allows the behaviour of the gateway system to act in a standard mode, or as DCM blocks have acted in the past. If TRUE, the gateway acts in a DCM mode when working with DCM blocks. If false, the gateway acts in the more traditional manner. When working with non-DCM blocks, this option has no impact.

The following table shows the differences in operation that the *GwDcmEmu* parameter controls.

GwDcmEmu set TRUE	GwDcmEmu set FALSE
Booleans are forced to the values of false (0) or true (1).	Booleans are interpreted as false (0) or true (any non-zero value).
ENUM fields transfer the value, only.	ENUM fields are transferred as two bytes. The first is the value, and the second if the enum limit.
REAL fields, when mapped to UNSIGNED 16-bit number, act as if it is unsigned.	REAL fields, when mapped to UNSIGNED 16-bit numbers always assume that 16-bit integers are signed, regardless of their true type (signed/unsigned).
TIME fields are transferred as a count of milliseconds.	TIME fields are first converted to ISO8601.

Table 44 Impact of the GwDcmEmu options parameter

Status. Bitfield indicating the Tactician communications and hardware status.

- **ColdStrt.** TRUE indicates this instrument will attempt a ‘cold start’.
- **HotStrt.** TRUE indicates this instrument will attempt a ‘hot start’.
- **DmpInPrg.** TRUE indicates this instrument is currently executing a ‘ForceDmp’.
- **CommsAlm.** This bit is set if *any* cached block within the Tactician unit is in software alarm due specifically to a communications failure (not a sumcheck error). The *CommsAlm* signal can therefore be used by a supervisory instrument to monitor the health of all inter-Tactician communications, even when the affected blocks themselves are not visible from the supervisor. It is only necessary to cache the communicating Tactician blocks, to make their *CommsAlm* bits accessible.
This bit can be used in conjunction with the *ComS/W* alarm to determine if either a comms failure OR a sumcheck failure has occurred.
- **EventLog.** TRUE indicates a severity ‘error’ or ‘Major error’ has been written to the EventLog file, .udz. This indicates further investigation is required because an unexpected problem in the execution of the instrument has occurred. Wiring to this field can allow it to be reset. Write to the status word or *Status.EventLog* bit to clear the bit to get a new alarm for the next important event log entry.
- **MultiSFC.** If this bit is FALSE when the database is loaded, it indicates that it is a pre-multi task sfc database. If this is the case, then the SFC_CON blocks are all forced on to task 4 where they used to operate. The flag is then set so that this will not be repeated in future. If the flag is set, then SFC_CON blocks may be allocated to any task.
- **P1PwFail.** TRUE indicates the power supply connected to the terminal labelled “P1” has failed. Only on products that can independently check their PSUs (the T2750, for example).
- **P2PwFail.** TRUE indicates the power supply connected to the terminal labelled “P2” has failed. Only on products that can independently check their PSUs (the T2750, for example).
- **FilSysEr.** TRUE indicates the instrument’s SD card is in quarantine mode. Quarantine mode occurs when the health of the instrument’s file system (SD card) is deemed to have an issue. During quarantine, the local file system is no longer accessible, although the control strategy continues to run. Any aspects of the control strategy configuration which requires access to the filing system (RECORD blocks, load SFCs, or Modbus or Profibus communications, for example) fail gracefully with the appropriate alarms. When *FilSysEr* is set TRUE, the *ComH/W* alarm is also asserted.
- **OtherCPU.** The bit is set TRUE if a functional CPU is fitted in the other slot of the same backplane regardless of whether these processors are configured to operate as simplex or duplex.
- **SavActiv.** (Redundant system only) TRUE indicates this instrument has resumed execution of the database, but the current saved database continues to be copied to the secondary controller.

Caution

Removal of power while SavActiv is set TRUE will corrupt the dbase.DBF on the secondary controller.

- **CldStPri.** (Redundant system only) TRUE indicates that this CPU instrument is the cold start primary, i.e. it will attempt to start-up as the primary controller after a power failure, if the relative status of the primary and secondary units at power down could not be resolved.

- **Red.** (Redundant system only) TRUE if configured as a redundant machine.
- **Chngovr.** (Redundant system only) TRUE if there has been a changeover of primary controller in a redundant system. Write to the status word or *Status.Chngovr* bit to clear the bit to get a new alarm for the next changeover event.
- **PwrFail.** Set if the LIN Database was started by a power-up (rather than by a user request, e.g. via the terminal configurator or a LIN remote command).

NOTE PwrFail is user-resettable. Write to the status word or *Status.PwrFail* bit to clear the bit to get a new alarm for the next power fail event.
- **TmpPFail.** This bit sets in the same way as *PwrFail*, but it automatically resets at the second database iteration.

EYCON-10: EYCON 10 CONFIGURATION BLOCK EYCON-20: EYCON 20 CONFIGURATION BLOCK

Block function

These blocks must appear in a strategy to be executed by the instrument type. It allows access to the instrument's Real-Time Clock (independently set up for each instrument), specifies temperature units for a single instrument, a coldstart (system reset) time and contains a brown-out alarm that latches on when the power is interrupted for a configured time. It also indicates a variety of instrument comms/hardware settings, and allows the selection of various user options.

Block parameters

Symbols used in *Table 45* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Time	Real-Time Clock time of day	hh:mm:ss	
Date	Real-Time Clock calendar	dd/mm/yy	
IP_type	Mains line frequency, temperature units	(ABC)D hex	
60Hz	Mains frequency (TRUE = 60Hz, FALSE = 50Hz)	T/F	D
Imperial	Temp. system (TRUE = Imperial, FALSE = SI)	T/F	
Absolute	Temp. units (TRUE = Absolute, FALSE = Relative)	T/F	
		1 2 4 8	
ColdStrt	Min. power-off time to trigger cold start on powerup	Mins	
BrownOut	Power-off time to trip BrownOut alarm	Mins	
Features	Software options	(ABC)D hex	
ModMst	Modbus master support, TCP and Serial	T/F	D
Soft_L1	Recipe and Programmer support	T/F	
Soft_L2	Batch file and Report file support	T/F	
Auditor	Audit operation support	T/F	
Node	ELIN node address, read from source	(01-FE hex)	
Options	Comms/hardware status	ABCD hex	
CommsDis	TRUE disables incoming ELIN field writes	T/F	D
Protectd	TRUE encrypts/deciphers databases on save/load	T/F	
CONFspd	TRUE=Configurator mode, FALSE=Run mode	T/F	
FullSave	TRUE saves current database to .DBF file	T/F	
PartSave	TRUE part-saves current database to .SBF file	T/F	C
BLOCKspd	TRUE speeds up running of function blocks	T/F	
PANELspd	TRUE improves front-panel response	T/F	
ForceDmp	TRUE generate dump file	T/F	
CachdPEs	TRUE enables display of cached block field values	T/F	B
UnSupAll	TRUE Unsuppresses all local Suppressed Alarms	T/F	
		1 2 4 8	
		T/F	
ClrBrOut	TRUE enables auto-clear brownout alarm	T/F	A
ColdStart	TRUE enables coldstart strategy	T/F	
WarmStrt	TRUE enables warmstart strategy (<i>not implemented</i>)	T/F	
HotStart	TRUE enables hotstart strategy	T/F	
Status	Comms/hardware status	ABCD hex	
		T/F	D
		T/F	
		T/F	
		1 2 4 8	
BattBad	Battery failed or absent	T/F	C
		T/F	
		T/F	
		1 2 4 8	
		T/F	B
		T/F	
		T/F	
		1 2 4 8	
BcstStrm	Ethernet rate protection	T/F	A
CommsAlm	Cached block comms failure (<i>not implemented</i>)	T/F	
PwrFail	Database was started by power-up (user-resettable)	T/F	
TmpPFail	As PwrFail, but auto-resets on 2nd dbase iteration	T/F	

Continued...

Parameter	Function	Units	Status
<i>Continued...</i>			
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
BrownOut	Power failure longer than BrownOut	T/F	
S6000	Software (comms) alarm in any S6000 block (<i>not implemented</i>)	T/F	
BadBat	Low battery voltage	T/F	
RTCinit	Real-Time Clock has not been initialised	T/F	
CPFail	Coldstart parameter file execution error	T/F	
BcstStrm	Ethernet rate protection	T/F	
Combined	OR-ing of all Alarms bits	T/F	
TaskRpt1 - 4	Required task repeat interval (<i>only TaskRpt1 implemented</i>)	Secs	
TaskHalt	User task halt settings (<i>not implemented</i>)	(ABC)D hex	
Task_1	TRUE halts user task 1	T/F	D 1 2 4 8
Task_2	TRUE halts user task 2	T/F	
Task_3	TRUE halts user task 3	T/F	
Task_4	TRUE halts user task 4	T/F	
TaskPri1 - 4	Required priority of user task 1 - 4, resp. (<i>not implemented</i>)	Integer	

Table 45 Block parameters

Block specification menu

This information is given in addition to *Table 45*.

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

Time, Date. Independent Real-Time Clock and calendar, respectively. These fields can be used to change the time and date in the instrument while the system is running ‘on-line’.

IP_type. Bitfield specifying mains frequency and temperature system/units. Specifications apply to an entire instrument and affect internal linearisation tables and cold-junction compensation. Units cannot be mixed in a single instrument, but each instrument on the ELIN can be set differently.

- **60Hz.** Set TRUE for 60Hz, or FALSE for 50Hz mains operation.
- **Imperial.** TRUE selects the Imperial temperature system - Fahrenheit (°F)/Rankine (R). FALSE selects the SI system - Celsius (°C)/Kelvin (K).
- **Absolute.** TRUE selects absolute temperature units - Kelvin/Rankine. FALSE selects relative units - Celsius/Fahrenheit.

ColdStrt. Specifies the minimum duration (minutes) of a power interruption that will cause a coldstart of the instrument. A ‘warmstart’ occurs after power interruptions shorter than *ColdStrt*; i.e. the current control strategy restarts with the existing parameter database and operating modes. After a coldstart, however, complete re-initialisation of the parameter database occurs and the control strategy specified in the .RUN extension file is loaded and run (if possible).

When a LIN Database file *filename.DBF* is downloaded from a remote node or loaded using the instrument’s local terminal, a *filename.RUN* file is created, and all other files with the .RUN extension are deleted. If a coldstart occurs, the instrument searches its filing system for a file with the .RUN extension. If found, the corresponding LIN Database file is loaded and run. If that fails, the instrument does not start its runtime.

BrownOut. Specifies the duration (minutes) of a power interruption required to trip the *BrownOut* alarm.

Features. Bitfield specifying instrument supported features, such as Auditor, Modbus (Serial or TCP) communications.

- **ModMst.** TRUE, if configured as Modbus Serial or TCP master.
- **Soft_L1, Soft_L2.** TRUE, if instrument supports Recipes and Setpoint Programmer and, Batch and Report respectively.
- **Auditor.** TRUE, if instrument supports Auditor functionality.

Options. Bitfield setting certain communications and hardware status options.

- **CommsDis.** If TRUE this bit disables UNCONFIRMED field writes only. UNCONFIRMED field writes result from connections into cached blocks. Values written using LINtools while online and connected to the instrument are CONFIRMED field writes and are therefore unaffected by this bit.
- **Protectd.** TRUE causes database strategies to be ‘scrambled’ when saved and ‘unscrambled’ when loaded. *(This feature is not currently supported.)*
- **CONFspd.** Selects configurator speed. TRUE selects ‘Configurator mode’. FALSE selects ‘Run mode’. Configurator mode causes control to operate at reduced efficiency (~85%) but remain unaffected when the resident configurator is run. Run mode lets control operate at maximum efficiency while the configurator is not running, but slows it down to ~53% whenever the configurator is run.
- **FullSave.** TRUE saves the current database to the current .dbf file.
- **PartSave.** TRUE saves the current database to file, *filename*.SBF, temporarily suspending the execution of the database during the save processes. *filename* is the root name of the current database file.
- **BLOCKspd.** TRUE increases the amount of CPU allocated to running function blocks. This improves field input-to-output times for Profibus systems, but at the expense of slowing down the front-panel response.
- **PANELspd.** TRUE decreases the amount of CPU allocated to running function blocks (contrast previous parameter). This improves front-panel responsiveness, and on Modbus-only systems may yield a slight improvement in Modbus throughput.

NOTE Setting *PANELspd* TRUE is **not** recommended for high-performance Profibus systems. Setting both *BLOCKspd* and *PANELspd* has no effect.

- **ForceDmp.** TRUE, generates a dump file of the current configuration of this instrument.
- **ClrBrOut.** TRUE enables auto-clear brownout alarm.
- **ColdStart.** TRUE enables coldstart control strategy.
- **WarmStrt.** TRUE enables warmstart control strategy *(not implemented)*.
- **HotStart.** TRUE enables hotstart control strategy.
- **CachdPEs** TRUE enables the instrument (Eycon V3.0 or later) to display all information collected from the block in the remote instrument (only if this instrument also supports the *CachdPEs*), including units, text and field name references.
- **UnSupAll.** When set TRUE it will unsuppress all alarms in the instrument’s local database regardless of whether they have been set to indefinite or with a time duration. The UnSupAll bit will then auto-reset to FALSE as it only reacts to a rising edge. If held TRUE, Alarm Suppression will continue to function as normal until the next rising edge. Wiring out from the UnSupAll bit serves no purpose.

Status. Bitfield indicating the instrument communications and hardware status.

- **BattBad.** Battery failed or absent.
- **BcstStrm.** TRUE, indicates the processor is not accepting all Ethernet traffic presented, but is allowing LIN communications, sets *Alarms.BcstStrm* TRUE.
- **CommsAlm.** This bit is set if *any* cached block within the instrument is in software alarm due specifically to a communications failure (not a sumcheck error). The *CommsAlm* signal can therefore be used by a supervisory instrument to monitor the health of all inter-instrument communications, even when the affected blocks themselves are not visible from the supervisor. It is only necessary to cache the communicating instrument blocks in the supervisor, to make the *CommsAlm* bits accessible. This bit can be used in conjunction with the *ComS/W* alarm to determine if either a comms failure OR a sumcheck failure has occurred.
- **PwrFail.** Set if the database was started by a power-up (rather than by a user request, e.g. via the terminal configurator or a ELIN remote command).

NOTE The *PwrFail* is user-resettable.

- **TmpPFail.** This bit sets in the same way as *PwrFail*, but it automatically resets at the second database iteration.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block's RAM data.
- **BrownOut.** Asserted if a power failure in excess of the time duration set in the *BrownOut* field has occurred, and appears as an unacknowledged 'event' alarm, without an acknowledged state.
- **S6000.** Software (comms) alarm in any S6000 block connected to this instrument.
- **BadBat.** Asserted if the battery supply is below the required voltage or defective.

NOTE Depending on the application of the instrument, it may be appropriate to set the alarm priority value high. When set to 6 or above, this alarm will require acknowledging, requesting a response from the user.
- **RTCinit.** Indicates that the Real-Time Clock has not been initialised.
- **CPFfail.** Asserted if a failure of the coldstart parameter file, .cpf, to execute has occurred. The .cpf contains parameter values to be used in the event of a cold start.
- **BcstStrm.** Asserted if the processor is not processing all Ethernet traffic presented, but is allowing LIN communications, set by *Status.BcstStrm* TRUE.
- **Combined.** Asserted if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

TaskRpt1 - 4. Specifies a repeat interval for Task 1. Only values in the range 40ms to 200ms are accepted. The values are rounded to the nearest 10ms. Repeat interval for Tasks 2, 3 and 4 are not implemented.

CHAPTER 6 CONTROL FUNCTION BLOCKS

The CONTROL category of Function Block Templates provides the control strategy with functions for controlling the control loop operations, PID, load simulation, etc. of an instrument.

The 'SETPOINT/3_TERM/MAN_STAT/MODE' control blocks are intended to be used as a combination to provide more flexibility than the single compact PID block. However, these blocks were introduced in the now mature T600 multi-loop controller and hence the description of each block is biased around this controller with references to various parameters being connected to the fascia e.g. Output (OP) of the MAN_STAT block is connected to the T600 output bargraph. These blocks are also available in current instruments (see *Table 2*) and therefore when reading the description of these blocks for implementation in current instruments, all references to the T640 should be ignored. Consequently the parameters that are assigned to the T600 fascia will require soft-wiring as appropriate in the target instrument, e.g. Eycon™ 10/20 Visual Supervisor.

PID: PID CONTROL BLOCK

Block function

Please refer to *Figure 25*. The PID Control block performs two main functions. It generates a resultant internal setpoint *SP* (upper part of the schematic), and then uses this setpoint together with a process variable input *PV* to generate a PID control output *OP* (lower part of the schematic).

The main features of the PID block are;

- Non interactive 3-Term control.
- P, P+I, P+D, PID and On-Off hysteresis control variants.
- 7 modes of operation.
- Interlocks for bumpless transfer in cascade control.
- Variable feedforward term (manual reset).
- Gain scheduling facilities.
- Setpoint trim.
- Integral term desaturation.
- Integral term balance.
- Filtered derivative term (reduces susceptibility to noise).
- Absolute and deviation alarms.

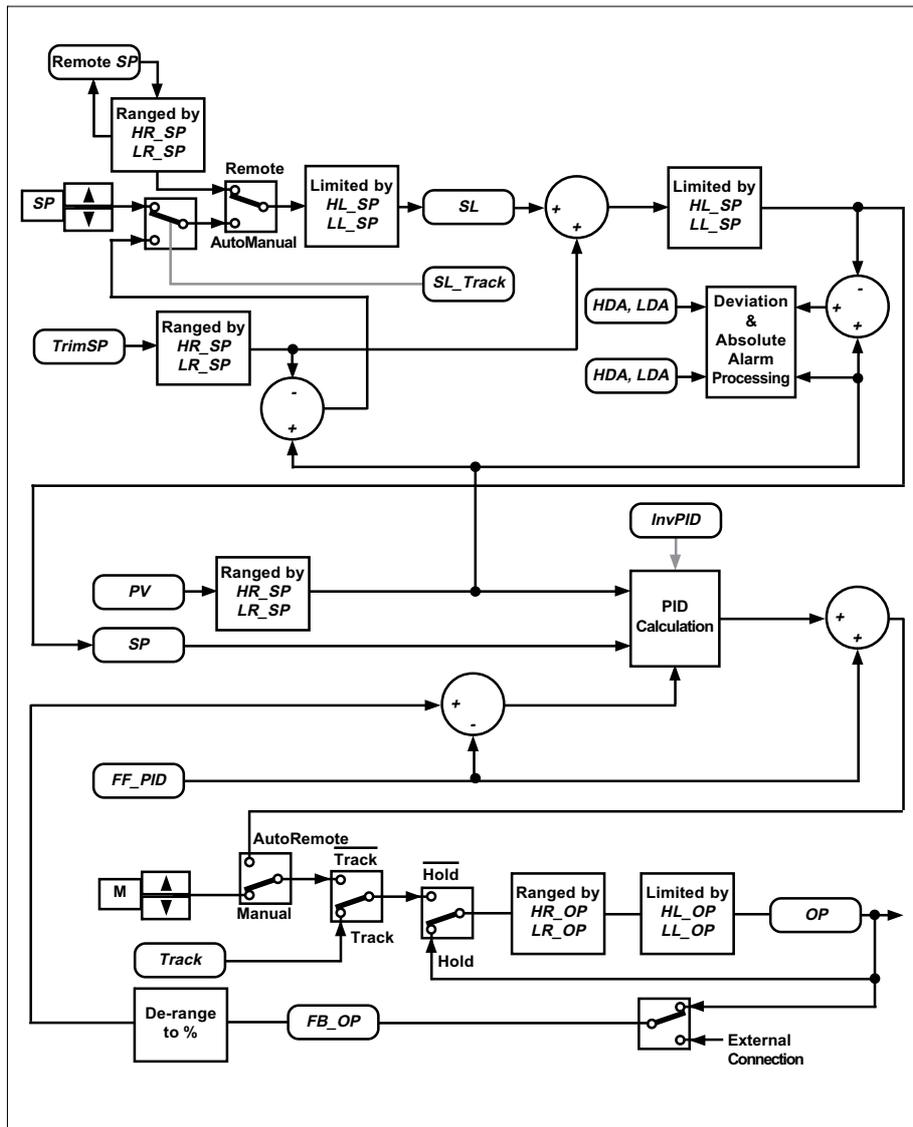


Figure 25 Block schematic

On-off control

Simple on-off control is implemented using the PID block by setting the proportional band XP to zero. This changes the function of TI which now sets the deadband (hysteresis) below the resultant setpoint (see *Figure 26*).

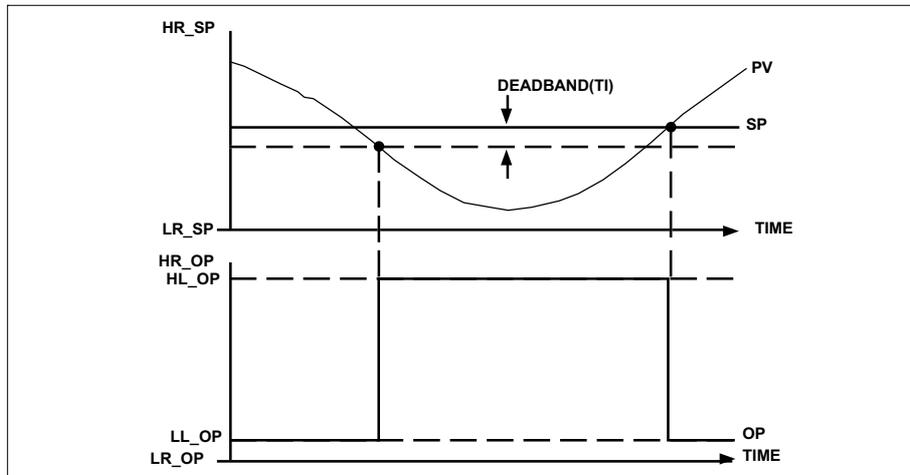


Figure 26 PID block on-off control

Setpoint definition

The resultant setpoint SP is the value ultimately used to control the process variable in a loop. SP derives from a parameter called the local setpoint (SL) which is the setpoint the operator can alter (except in the remote operating mode).

In automatic mode, the local setpoint can always be adjusted unless measures are taken to stop this.

In remote mode, the local setpoint is controlled by $RemoteSP$, and this ensures that transfer from remote into any other mode of operation is bumpless. However, the procedure for transferring into remote mode is not necessarily bumpless, and often requires protection. This implies the use of track modes in the preceding blocks - standard practice in cascaded systems.

In manual, track and hold modes there are two alternatives available for control of the local setpoint, defined by SL_Track in the *Options* field.

If $SL_Track = TRUE$, the local setpoint automatically follows the process variable. This simplifies the operation of most control loops, and avoids the possibility of setpoint disturbances occurring as a result of careless operation. The second alternative is to allow the operator to maintain control of the local setpoint in these modes of operation. This is achieved by setting $SL_Track = FALSE$.

The $TrimSP$ facility allows a bias to be added to the local setpoint. This feature is often used for plant optimisation and production control.

NOTE The setpoint limits defined by parameters HL_SP and LL_SP act independently on both the local setpoint (SL) and the resultant setpoint (SP) in all modes of operation.

Operating mode characteristics

The PID control block can operate in one of several different *control modes*, each having its own way of controlling the loop. The operating characteristics of these modes are now summarised. For information on mode selection, interaction and priority, the PID block 3-Term control algorithm, and integral balance and integral desaturation techniques, see Appendix A, *Control Loop Operating Modes*.

Hold mode

The controller output value OP is locked in hold mode. The local setpoint either follows PV or stays constant depending on the status of SL_Track .

Track mode

The controller output value OP is controlled by *Track*. The local setpoint either follows PV or stays constant depending on the status of SL_Track .

Manual mode

The controller output *OP* has read/write status. The local setpoint *SL* either follows *PV* or stays constant depending on the status of *SL_Track*.

Automatic mode (AUTO)

OP is controlled as a function of *PV* and *SP*. *SL* has read/write status.

Remote mode

OP is controlled as a function of *PV* and *SP*. *SL* is controlled by *RemoteSP*, which has read/write status. Remote mode is a conditional mode of operation. *EnaRem* (remote mode enable) must be TRUE to permit operation in remote mode. If *EnaRem* is FALSE the control loop defaults into forced automatic mode if remote mode is selected.

Forced automatic mode (F_AUTO)

This mode is selected by default when remote mode is selected without being enabled. Forced automatic mode has the same operational characteristics as automatic mode, with the exception that the control loop transfers to remote mode as soon as *EnaRem* is TRUE.

Forced manual mode (F_MAN)

The operating characteristics of this mode are similar to manual mode. The mode is normally used to indicate a fault condition which precludes operation in automatic or remote modes, e.g. loss of *PV* input, data corruption, etc.

Forced manual mode can only be selected from the *ModeSel* field and therefore its application is user-defined. Setpoint limits and output limits are active in all operating modes.

Block parameters

Symbols used in *Table 46* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Mode	Current operating mode	Menu	
FallBack	Suppressed operating mode	Menu	
PV	Process Variable (PV)	Eng	
SP	Resultant setpoint	Eng	
OP	Controller output	%	
SL	Local setpoint	Eng	
TrimSP	Local setpoint trim	Eng	
RemoteSP	Remote setpoint	Eng	
Track	Track input	%	
HR_SP	Setpoint/process variable high range	Eng	
LR_SP	Setpoint/process variable low range	Eng	
HL_SP	Setpoint high limit	Eng	
LL_SP	Setpoint low limit	Eng	
HR_OP	Output high range	%	
LR_OP	Output low range	%	
HL_OP	Output high limit	%	
LL_OP	Output low limit	%	
Alarms			
Software	Block RAM data sumcheck error/network failure	T/F	
HighAbs	High absolute alarm	T/F	
LowAbs	Low absolute alarm	T/F	
HighDev	High deviation alarm	T/F	
LowDev	Low deviation alarm	T/F	
Combined	OR-ing of all Alarms bits	T/F	
HAA	High absolute alarm limit	Eng	
LAA	Low absolute alarm limit	Eng	
HDA	High deviation alarm limit	Eng	
LDA	Low deviation alarm limit	Eng	
TimeBase	TI, TD time units (secs/mins)	Menu	
XP	Proportional band	%	
TI	Integral time constant/deadband %	%	

Continued...

Parameter	Function	Units	Status
<i>Continued...</i>			
TD	Derivative time constant	%	<input type="checkbox"/>
Options	Controller options	AB hex	<input type="checkbox"/>
InvPID	3-Term action	T/F	
SL_Track	SL tracks PV	T/F	
IntBalSL	Integral balance on SL writes	T/F	
IntBalXP	Integral balance on XP changes	T/F	
IntBal	Force integral balance	T/F	
MSELFunc	Only one ModeSel mode bit can be active	T/F	
DevAlmLm	Specifies deviation alarm based on SP/limited SP	T/F	
SelMode	Select mode	AB hex	<input type="checkbox"/>
SelHold	Hold mode select	T/F	
SelTrack	Track mode select	T/F	
SelRem	Remote mode select	T/F	
EnaRem	Remote mode enable	T/F	
SelAuto	Automatic mode select	T/F	
SelMan	Manual mode select (Not used)	T/F	
SelFMan	Forced manual mode select	T/F	
ModeSel	Modes selected	AB hex	<input type="checkbox"/>
EnaRem	Enable remote (to slave)	T/F	
HoldSel	Hold mode selected	T/F	
TrackSel	Track mode selected	T/F	
RemSel	Remote mode selected	T/F	
AutoSel	Automatic mode selected	T/F	
ManSel	Manual mode selected	T/F	
FAutoSel	Forced automatic mode selected	T/F	
FmanSel	Forced manual mode selected	T/F	
ModeAct	Active mode	AB hex	<input type="checkbox"/>
NotRem	Remote mode not active	T/F	
HoldAct	Hold mode active	T/F	
TrackAct	Track mode active	T/F	
RemAct	Remote mode active	T/F	
AutoAct	Automatic mode active	T/F	
ManAct	Manual mode active	T/F	
FAutoAct	Forced automatic mode active	T/F	
FManAct	Forced manual mode active	T/F	
FF_PID	Feedforward	%	<input type="checkbox"/>
FB_OP	Output feedback	%	<input type="checkbox"/>

Table 46 Block parameters

Block specification menu

Dbase, Block, Type. See Appendix D page 544 for details of these ‘header’ fields.

Mode. (HOLD/TRACK/MANUAL/AUTO/REMOTE/F_MAN/F_AUTO). Current operating mode. Can be used to select manual, automatic and remote modes, depending on the status of the *SelMode* parameter. In track or hold modes, *Mode* selects the fallback mode.

FallBack. (MANUAL/AUTO/REMOTE/F_MAN/F_AUTO). Indicates mode of operation adopted if no modes are selected via the *SelMode* parameter, see *Selector Function Blocks* section.

PV. Process variable.

SP. Resultant setpoint.

OP. Controller actuation output. This field has read/write status in MANUAL and F_MAN modes; it is read-only in all other modes.

SL. Local setpoint. This value represents the operator setpoint for the control loop. A setpoint trim is added to *SL* to produce *SP*, used as the desired value for control. *SL* has read/write status in automatic mode, and read-only status in remote mode where *SL* follows *RemoteSP*. The read/write status of *SL* in manual, track and hold modes is conditional, and configured in *SL_Track*. Setpoint limits are used in all modes of operation and act on both *SL* and the *SP* values independently.

TrimSP. Setpoint trim.

RemoteSP. Remote setpoint. Controls setpoint in remote mode only.

Track. Controls *OP* in track mode.

HR_SP, LR_SP. High & low ranges, resp., of setpoint and process variable in engineering units.

HL_SP, LL_SP. High & low setpoint limits, resp. Setpoint limits are active in all modes of operation and act independently on *SL* and *SP*.

HR_OP, LR_OP. High & low ranges, resp., of output in engineering units.

HL_OP, LL_OP. High & low limits, resp., of output in engineering units. Output limits are active in all modes of operation.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block's RAM data.
- **HighAbs, LowAbs.** High & low absolute alarms set by parameters *HAA* and *LAA*, respectively, with a 0.5% hysteresis band on each limit.
- **HighDev, LowDev.** High & low deviation alarms set by parameters *HDA* and *LDA*, respectively, with a 0.5% hysteresis band on each limit. A high deviation alarm occurs when $PV - SP > HDA$, and a low deviation when $SP - PV > LDA$.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

HAA, LAA. High & low absolute alarm limits, resp., in engineering units.

HDA, LDA. High & low deviation alarm limits, resp., in engineering units.

TimeBase. (Secs/Mins). Specifies time units for *TI* and *TD*.

XP. Proportional band (%). The on-off hysteretic control option is selected when $XP = 0$. The function of *TI* is altered and is used to specify the deadband when on-off control is selected.

TI. Dual function: Integral time constant, seconds/minutes ($XP \neq 0$), or Deadband % ($XP = 0$). With $XP = 0$, *TI* specifies the deadband as a percentage of the setpoint range.

NOTE. The deadband function operates below the resultant setpoint.

TD. Derivative time constant (seconds/minutes).

Options.

- **InvPID.** Selects 3-Term control action sense. The error term in the Process Automation controllers is defined as $ER = PV - SP$. When *InvPID* = FALSE, an increase in ER produces a corresponding decrease in the 3-term output to compensate. When *InvPID* = TRUE, increasing ER produces a corresponding increase in the 3-term output to compensate.
- **SL_Track.** Causes the local setpoint to follow *PV* when either hold, track or manual modes are active. The local setpoint is still subject to the action of setpoint limits.
- **IntBalSL.** Affects the operation of the controller in automatic mode. When *IntBalSL* = TRUE, any write to the local setpoint *SL* automatically triggers an integral term balance procedure on the 3-term output, see *Appendix A, Control Loop Operating Modes*, for details of integral term balance). When *IntBalSL* = FALSE, a change to *SL* results in the normal 3-term response.

NOTE. When TRUE, this will cause an integral term balance even for writes that do not change the value of *SL*.

- **IntBalXP.** Allows the integral term balance to be triggered automatically on changes to the proportional band (*XP*). Normally set to TRUE, *IntBalXP* should be disabled in gain-scheduling applications, where it would interfere with proper control response.
- **IntBal.** A rising edge into this bit forces an integral term balance to be performed, regardless of the other *Options* inputs.
- **MSElFunc.** When TRUE, this bit modifies the action of the *ModeSel* parameter so that only one of the mutually exclusive mode bits is active at any one time. Defaults TRUE on block creation.
- **DevAlmLm.** When TRUE, deviation alarms are based on the limited version of *SP*. When FALSE, deviation alarms are based on *SP* prior to the application of any limits.

SelMode. This bitfield is used to select controller modes via digital inputs from the control strategy. The following is given in addition to *Table 46*.

- **SelRem.** Selects remote mode. Remote mode is a conditional mode of operation which requires to be enabled. *EnaRem* must be TRUE to permit operation in remote mode, otherwise forced automatic mode is selected by default.
- **EnaRem.** Enables remote mode. TRUE permits operation in remote mode.
- FALSE causes the control to be transferred into forced automatic mode when remote mode is selected.

ModeSel. This bitfield indicates the modes that have been *selected*. The mode actually *active* is the selected mode with highest priority. The following is given in addition to *Table 46*.

- **EnaRem.** Output connection from master to slave controller for cascade control applications. This connection is FALSE when hold or manual mode is selected (not necessarily active) in the master controller. This output should be connected to the *SelMode.EnaRem* input on a slave controller.

ModeAct. This bitfield shows the control mode currently active. The following is given in addition to *Table 46*.

- **NotRem.** Output connection from the slave controller into the master for cascade control applications. This connection is TRUE when the slave controller is not operating in remote mode. This output should be connected into *SelMode.SelTrack* on the master controller.

FF_PID. Feed forward. Used to add a bias to the 3-Term output. This is sometimes called ‘manual reset’.

FB_OP. Feedback value. Used in integral desaturation, to indicate when the controller output has been restricted by the action of output limits, see Appendix A, *Integral Balance & Integral Desaturation*.

ANMS: ANALOGUE MANUAL STATION BLOCK

Block function

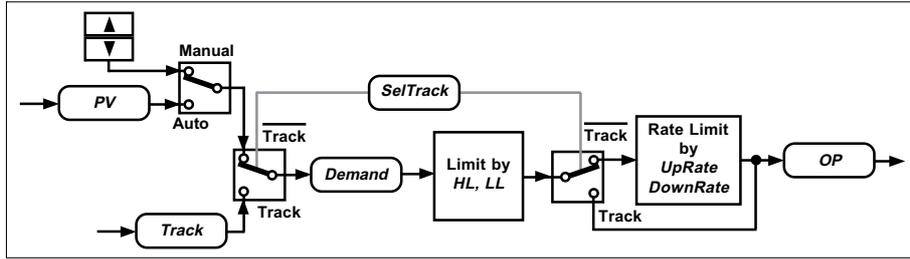


Figure 27 Block schematic

Please refer to the schematic in *Figure 27*. The Analogue Manual Station block allows the operator to interact with a plant variable at runtime. The station provides output limits, asymmetric rate limits, and auto/manual/track control with fallback.

In track mode (highest priority), *OP* is derived from the *Track* parameter. In automatic mode *OP* follows *PV*, and in manual mode *OP* can be adjusted via the *Demand* parameter. Note that in track mode *OP* is subject to output limits but not rate limits. In auto and manual both output limits and rate limits (if activated) apply.

Block parameters

Symbols used in *Table 47* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Mode	Current operating mode	Menu	
Fallback	Suppressed operating mode	Menu	
PV	Process variable	Eng	
Track	Controls OP in track mode	Eng	
Demand	Controls OP in manual mode	Eng	
OP	Analogue output	Eng	
HR, LR	PV, OP high & low graphics range	Eng	
HL, LL	High & low OP limits	Eng	
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
Combined	OR-ing of all Alarms bits	T/F	
UpRate	Limits OP increase rate	Eng/TimeBase	
DownRate	Limits OP decrease rate	Eng/TimeBase	
TimeBase	Time units select (Rate limits)	Menu	
SelRate	Activates rate limits	T/F	
SelTrack	Track mode select	T/F	
NotAuto	Not auto mode	T/F	

Table 47 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

Mode. (AUTO/MANUAL/TRACK). Current operating mode. Can be used to select automatic and manual. (With *SelTrack* TRUE, *Mode* selects the fallback mode.)

Fallback. (AUTO/MANUAL). Indicates next (suppressed) operating mode.

PV. Process Variable (input value).

Track. Controls *OP* in track mode.

Demand. Controls *OP* in manual mode.

OP. Output value from station.

HR, LR. High & low range for graphic objects (Bar, Trend) linked to *PV* and *OP*. *HR* and *LR* define the 100% and 0% displays, respectively.

HL, LL. High & low *OP* limits.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block's RAM data.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

UpRate. Specifies maximum rate of increase of *OP* (i.e. 'rate limit') in engineering units/time. Time units are specified by the *TimeBase* parameter.

DownRate. Specifies maximum rate of decrease of *OP* (i.e. 'rate limit') in engineering units/time.

TimeBase. (Secs/Mins/Hours/Days) Selects time units for rate limit functions, *UpRate* and *DownRate*.

SelRate. Activates rate limit functions, *UpRate* and *DownRate*.

SelTrack. Selects track mode. With *SelTrack* TRUE, the *Mode* parameter defines the fallback (suppressed) mode.

NotAuto. Indicates that the station is not operating in auto mode.

DGMS: DIGITAL MANUAL STATION BLOCK

Block function

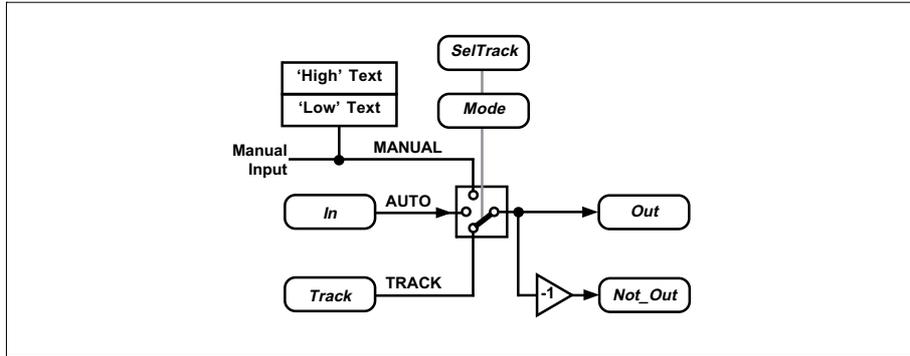


Figure 28 Block schematic*

Please refer to the schematic in *Figure 28*. The Digital Manual Station block allows the operator to switch a logic signal within a control strategy at runtime. In automatic mode the *Out* parameter follows the state of the *In* parameter. In manual mode, *Out* can be set independently (via the specification menu or a Supervisory unit e.g. Eycon™ 10/20 Visual Supervisor). In track mode*, *Out* follows the *Track** parameter.

Block parameters

Symbols used in *Table 48* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Mode	Current operating mode	Menu	
In	Input	T/F	<input type="checkbox"/>
Out	Output	T/F	<input type="checkbox"/>
Not_Out	Complement of out (= Out)	T/F	<input type="checkbox"/>
Track*	Track value	T/F	<input type="checkbox"/>
SelTrack*	TRUE = track mode selected	T/F	<input type="checkbox"/>
NotAuto	TRUE = mode is not automatic	T/F	<input type="checkbox"/>
Status	Mode status information	(ABC)D hex	<input type="checkbox"/>
FallMan	Track: F=Auto, T=Man; ELSE fallback indicated	T/F	<div style="display: flex; align-items: center;"> <div style="border: 1px solid black; padding: 2px; margin-right: 5px;">1</div> <div style="border: 1px solid black; padding: 2px; margin-right: 5px;">2</div> <div style="border: 1px solid black; padding: 2px; margin-right: 5px;">4</div> <div style="border: 1px solid black; padding: 2px;">8</div> <div style="margin-left: 5px; font-weight: bold; font-size: 1.2em;">D</div> </div>
		T/F	
		T/F	
		T/F	
Alarms			<input type="checkbox"/>
Software	Block RAM data sumcheck error / network failure	T/F	<input type="checkbox"/>
Combined	OR-ing of all Alarms bits	T/F	<input type="checkbox"/>

Table 48 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

Mode. (AUTO/MANUAL/TRACK*). Current operating mode. Can be used to select automatic and manual. TRACK* must be selected by *SelTrack**.

In. Input state. Controls the output in automatic mode.

Out. Output state. Controls the output in manual mode.

Not_Out. Complementary form of the output (*Out*).

Track*. Controls *Out* in track mode.

SelTrack*. TRUE selects track mode. With *SelTrack* TRUE, the *Mode* parameter defines the fallback (suppressed) mode.

NotAuto. Indicates that the station is not operating in auto mode.

Status. 16-bit status field, with only bit 0 in use.

- **FallMan.** When the block is not in Track mode, *FallMan* is FALSE when in Auto, TRUE when in Manual. When the block is in Track, *FallMan* indicates what mode the block will fall back to when Track is removed, FALSE means Auto, TRUE means Manual.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block's RAM data.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

*NOTE. The track facility is not available in early versions of the DGMS block.

SIM: SIMULATION BLOCK

Block function

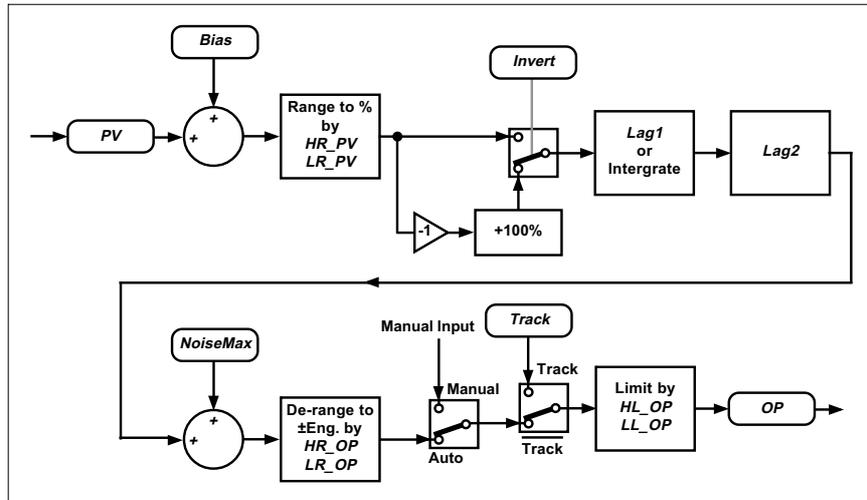


Figure 29 Block schematic

Please refer to the schematic in *Figure 29*. The Simulation block provides a simulation of plant characteristics and can be used for proving the operation of a control strategy off-line.

The SIM block has two first-order lag functions with pseudo-random noise. A further option allows an integrator function to be substituted in place of the first lag function for simulation of capacity. The block also includes *PV* bias, inversion, initialisation and auto/manual/track control with fallback.

Track mode, the highest priority, is selected via the *SelTrack* parameter. In track mode *OP* is controlled by *Track*. In automatic mode *OP* is derived from *PV* according to the block characteristics. In manual mode *OP* can be adjusted independently. Note that output limits are active in *all* operating modes.

Block parameters

Symbols used in *Table 49* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Mode	Current operating mode	Menu	
Fallback	Suppressed operating mode	Menu	
PV	Process variable	Eng1	
Bias	PV bias	Eng1	
Track	Controls OP in track mode	Eng1	
HR_PV	Input high range (& PV graphics range)	Eng1	
LR_PV	Input low range (& PV graphics range)	Eng1	
OP	Block output	Eng2	
HR_OP	Output high range (& OP graphics range)	Eng2	
LR_OP	Output low range (& OP graphics range)	Eng2	
HL_OP	Output high limit	Eng2	
LL_OP	Output low limit	Eng2	
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
Combined	OR-ing of all Alarms bits	T/F	
NoiseMax	Pseudo-random noise max. amplitude	Eng2	
Lag1	1st filter (or integrator) time constant	Time	
Lag2	2nd filter time constant	Time	
TimeBase	Lag1 & lag2 time units select	Menu	
Intgr	Lag1 function select (integrator/filter)	T/F	
Invert	PV sense invert select	T/F	
Init	Block filters initialisation	T/F	
SelTrack	Track mode select	T/F	

Table 49 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these 'header' fields.

Mode. (AUTO/MANUAL/TRACK). Current operating mode.

Fallback. (AUTO/MANUAL). Indicates next (suppressed) operating mode.

PV. Input value.

Bias. Input bias.

Track. Controls *OP* in track mode.

HR_PV, LR_PV. Input high & low range. Also, high & low range for graphic objects linked to *PV* (Bar, Trend). *HR_PV* and *LR_PV* define the 100% and 0% displays, respectively.

OP. Output value after simulation functions. This value has read/write status in manual mode.

HR_OP, LR_OP. Output high & low range. Also, high & Low range for graphic objects linked to *OP* (Bar, Trend).

HL_OP, LL_OP. Output high & low limits.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

■ **Software.** Sumcheck error in block's RAM data.

■ **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

NoiseMax. Sets the maximum amplitude value of the pseudo-random noise function.

Lag1. 1st filter time constant or integrator constant, depending on the value of the *Intgr* parameter. Defines either first-order low-pass filter time constant, or integrator constant (expressed in engineering units per unit time).

Lag2. 2nd filter time constant. Defines first-order low-pass filter time constant.

TimeBase. (Secs/Mins/Hours/Days) Time units for *Lag1* and *Lag2*.

Intgr. Selects function for the *Lag1* parameter: TRUE = integrator, FALSE = first-order low-pass filter.

Invert. If *Intgr* is FALSE, *Invert* = TRUE inverts the sense of *PV* (after biasing and ranging to percentage) by subtracting *PV*% from 100%. If *Intgr* is TRUE, the inversion is done by subtracting *PV*% from 0% instead.

Init. Initialises the block filters by momentarily (for two algorithm iterations) making their outputs equal to their inputs. Operates only in auto mode.

SelTrack. Selects track mode, and makes the existing mode the fallback mode.

AN_CONN: ANALOGUE CONNECTIONS BLOCK

Block function

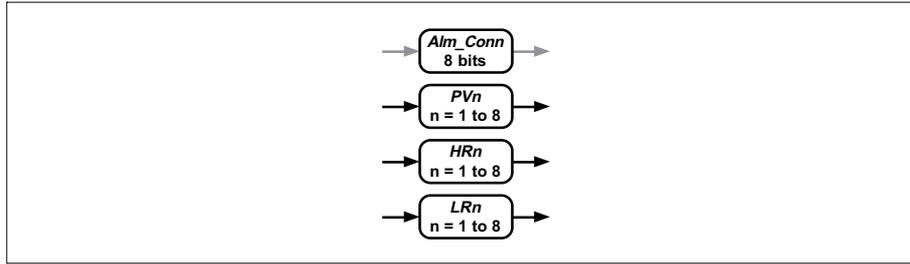


Figure 30 Block schematic

Please refer to *Figure 30*. The Analogue Connection block collects and retransmits up to 24 analogue signals, together with a set of eight digitals. The analogues are grouped into eight sets of *PV*, *HR*, and *LR* signals to correspond to the trio of analogue outputs from a single ANIN block. The main use of the AN_CONN block is to collect into a single block the signals from up to eight ANIN blocks, running in an I/O module, before transmitting them across the LIN (to a cached AN_CONN block). The *Alm Conn* connections allow for one chosen digital (usually an alarm output) to be transmitted from each ANIN block, although the connections can actually be used for any purpose.

The benefits of transmitting analogue signals across the LIN via AN_CONN blocks are: fewer blocks to cache, less network traffic, and faster block scanning because less time is spent on network activity. The block can also be used to buffer and synchronise data transmitted across the LIN.

Block parameters

Symbols used in *Table 50* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Alm Conn	Alarm connections (general purpose)	(AB)CD Hex	
Alarm 1	Alarm connection 1	T/F	D
Alarm 2	Alarm connection 2	T/F	
Alarm 3	Alarm connection 3	T/F	
Alarm 4	Alarm connection 4	T/F	
Alarm 5	Alarm connection 5	T/F	C
Alarm 6	Alarm connection 6	T/F	
Alarm 7	Alarm connection 7	T/F	
Alarm 8	Alarm connection 8	T/F	
PV1 to PV8	Analogue input 1 to input 8	Eng1-Eng8	
HR1 to HR8	High range PV1 to high range PV8	Eng1-Eng8	
LR1 to LR8	Low range PV1 to low range PV8	Eng1-Eng8	
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
Combined	OR-ing of all Alarms bits	T/F	

Table 50 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

Alm Conn. Bitfield containing eight digital parameters, *Alarm 1* to *Alarm 8*, that can be used to transmit any digital signals. Usually each connection would be used to transmit a digital selected from the *Alarms* field of an Analogue Input block.

NOTE The digital parameters in this bitfield do *not* revert to the alarm state in the event of a block communications software alarm.

PV1 to PV8. Analogue inputs retransmitted as outputs, ranged by the corresponding *HR1*, *LR1* to *HR8*, *LR8* parameters. Normally, but not necessarily, *PV1* to *PV8* would originate in corresponding ANIN blocks.

HR1, LR1 to HR8, LR8. Analogue inputs retransmitted as outputs, which range the corresponding *PV1* to *PV8* parameters. Normally, but not necessarily, these range parameters would be connected to the appropriate *HR* and *LR* of the ANIN block providing *PV*; but they can equally be used to re-range the *PV*s to other values if required.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block's RAM data.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

DG_CONN: DIGITAL CONNECTIONS BLOCK

Block function

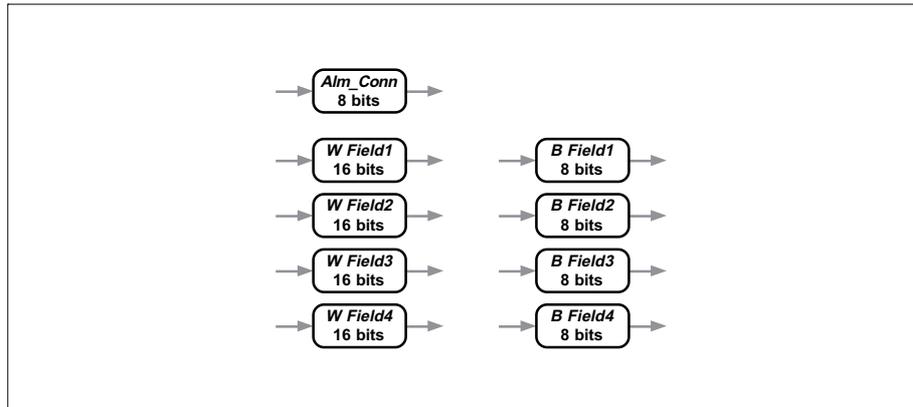


Figure 31 Block schematic

Please refer to *Figure 31*. The Digital Connections block collects and retransmits a total of up to 104 digital signals. The digitals are grouped for convenience into four 16-bit words (*W Field1* to *W Field4*), four 8-bit bytes (*B Field1* to *B Field4*), plus an 8-bit *Alm Conn* field. This capacity would allow a single DG_CONN block to collect the strategy input signals from up to thirteen DGIN_8 blocks, running in an I/O unit, before transmitting them across the LIN (to a corresponding cached DG_CONN block). Alternatively, with fewer pooled DGIN blocks, additional associated signals (e.g. status outputs) could be collected and transmitted. In fact the connections can be used for any digital signals, from any blocks.

NOTE The connections should not be used for alarms, because they do *not* revert to the alarm state (i.e. fail safe) in the event of a block communications software alarm.

The benefits of transmitting digital signals across the LIN via DG_CONN blocks are: fewer blocks to cache, less network traffic, and faster block scanning because less time is spent on network activity. The block can also be used to buffer and synchronise data transmitted across the LIN.

Block parameters

Symbols used in *Table 51* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Alm Conn	Alarm connections (General purpose)	(AB)CD Hex	☛☞
Alarm 1	Alarm connection 1	T/F 1	D
Alarm 2	Alarm connection 2	T/F 2	
Alarm 3	Alarm connection 3	T/F 4	
Alarm 4	Alarm connection 4	T/F 8	
Alarm 5	Alarm connection 5	T/F 1	C
Alarm 6	Alarm connection 6	T/F 2	
Alarm 7	Alarm connection 7	T/F 4	
Alarm 8	Alarm connection 8	T/F 8	
W Fieldn (n=1 to 4)	Digital input/output words	ABCD Hex	☛☞
Bit0	Bit 0 digital input/output	T/F 1	D
Bit1	Bit 1 digital input/output	T/F 2	
Bit2	Bit 2 digital input/output	T/F 4	
Bit3	Bit 3 digital input/output	T/F 8	
Bit4	Bit 4 digital input/output	T/F 1	C
Bit5	Bit 5 digital input/output	T/F 2	
Bit6	Bit 6 digital input/output	T/F 4	
Bit7	Bit 7 digital input/output	T/F 8	
Bit8	Bit 8 digital input/output	T/F 1	B
Bit9	Bit 9 digital input/output	T/F 2	
BitA	Bit A digital input/output	T/F 4	
BitB	Bit B digital input/output	T/F 8	
BitC	Bit C digital input/output	T/F 1	A
BitD	Bit D digital input/output	T/F 2	
BitE	Bit E digital input/output	T/F 4	
BitF	Bit F digital input/output	T/F 8	
Alarms			☛☞ ☞☞ ☞☞☞
Software	Block RAM data sumcheck error / network failure	T/F	
Combined	OR-ing of all Alarms bits	T/F	
B Fieldn (n=1 to 4)	Digital input/output bytes	(AB)CD Hex	☛☞
Bit0	Bit 0 digital input/output	T/F 1	D
Bit1	Bit 1 digital input/output	T/F 2	
Bit2	Bit 2 digital input/output	T/F 4	
Bit3	Bit 3 digital input/output	T/F 8	
Bit4	Bit 4 digital input/output	T/F 1	C
Bit5	Bit 5 digital input/output	T/F 2	
Bit6	Bit 6 digital input/output	T/F 4	
Bit7	Bit 7 digital input/output	T/F 8	

Table 51 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

Alm Conn. Bitfield containing eight digital parameters, labelled *Alarm 1* to *Alarm 8*, that can be used to collect and retransmit alarm signals from digital input blocks, or any other digital signals.

NOTE The digital parameters in this bitfield do *not* revert to the alarm state in the event of a block communications software alarm.

W Field1 to W Field4. Word fields 1 to 4. Four general-purpose 16-bit fields (*Bit0* to *BitF*) that can be used to collect and retransmit digital signals.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

B Field1 to B Field4. Byte fields 1 to 4. Four general-purpose 8-bit fields (*Bit0* to *Bit7*) that can be used to collect and retransmit digital signals.

TP_CONN: TEPID DATA CONNECTION BLOCK

Block function

The TP_CONN block allows up to nine fields to be specified for saving to E2PROM as part of the ‘tepid data’ at power-down. Should there be a subsequent ‘tepid start’, these fields are restored to the values they had at the time of power-down, see relevant T600 Series product literature for tepid start and tepid data information.

The nine fields are specified by block name (in the *Blockn* parameters) and field name (in the *Fieldn* parameters). They can be in any blocks in the database, but cannot be *Alarms* fields, text string fields, or the *EXPR* field of the *EXPR* block. Specifying a forbidden or non-existent field sets the corresponding *BadField* bit TRUE.

Only one TP_CONN block is allowed per LIN Database. Its block name is written to the T600 block’s *AnConBlk* parameter.

NOTE If a TP_CONN block is named in the T600 block, this precludes the use of either an AN_CONN or a DG_CONN block to specify tepid data.

Block parameters

Symbols used in *Table 52* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
BadField	Invalid field flags	(A)BCD hex	
Field1	TRUE = non-existent or forbidden Field1	T/F	D
Field2	TRUE = non-existent or forbidden Field2	T/F	
Field3	TRUE = non-existent or forbidden Field3	T/F	
Field4	TRUE = non-existent or forbidden Field4	T/F	
Field5	TRUE = non-existent or forbidden Field5	T/F	C
Field6	TRUE = non-existent or forbidden Field6	T/F	
Field7	TRUE = non-existent or forbidden Field7	T/F	
Field8	TRUE = non-existent or forbidden Field8	T/F	
Field9	TRUE = non-existent or forbidden Field9	T/F	B
Unused			
Unused			
Unused			
Block1 to Block9	Specify block names containing Field1 to Field9 fields		
Alarms			
Software	Block RAM data sumcheck error/network failure	T/F	
Combined	OR-ing of all Alarms bits	T/F	
Field1 to Field9	Specify fields contained in Block1 to Block9, resp.	Alphanumeric	

Table 52 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

BadField. Read-only bitfield whose nine bits (*Field1* to *Field9*) indicate if the corresponding fields specified by the *Fieldn* parameter are valid fields. A TRUE bit denotes that a forbidden or non-existent field has been specified.

Block1 to Block9. Specify up to nine database block names (*Block* parameters), containing fields, specified in the *Field1* to *Field9* parameters, respectively - to be saved as tepid data at power-down.

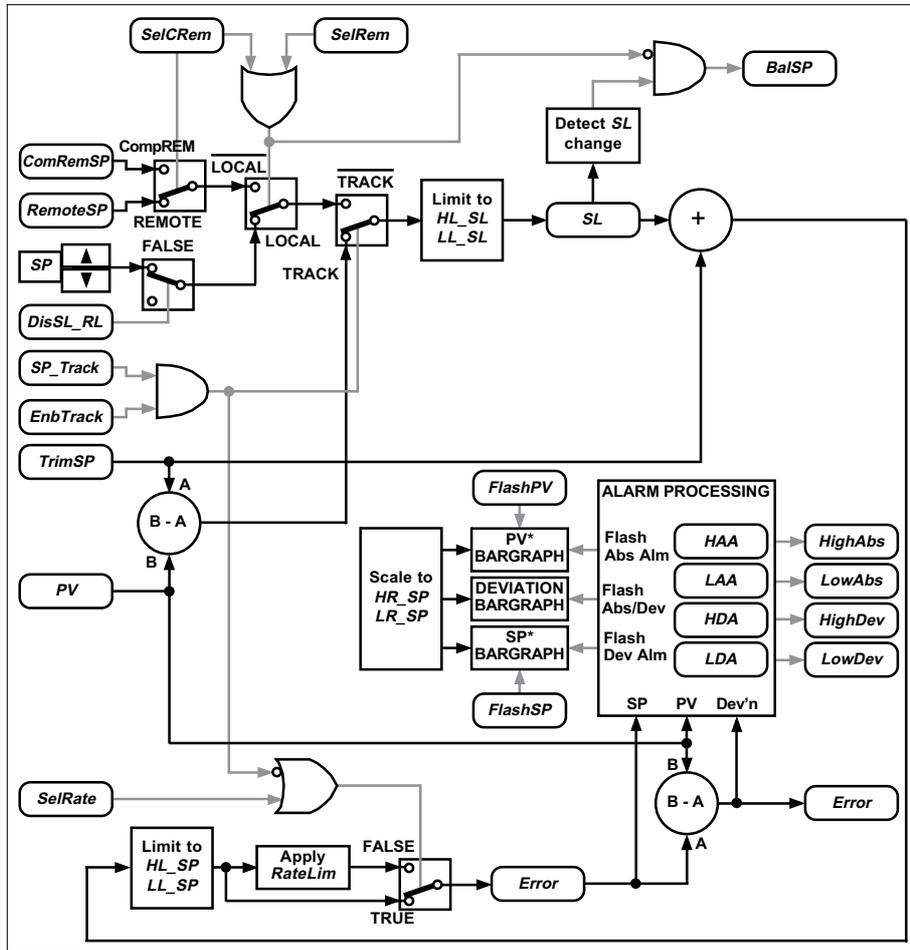
Field1 to Field9. Specify up to nine field names, contained in the blocks specified in the *Block1* to *Block9* parameters, respectively, to be saved as tepid data at power-down.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

SETPOINT: SETPOINT BLOCK

Block function



**PV & SP bargraph outputs are linked to the T600 fascia displays only if the block's assigned loop is the one selected for main display.*

Figure 32 Block schematic

Please refer to *page 140* for usage implications when using this block and the schematic in *Figure 32*. The SETPOINT block is intended to be used as part of the SETPOINT/3_TERM/MAN_STAT/MODE combination of control blocks, which gives more flexibility than the simple PID block. The block generates a read-only resultant setpoint *SP* from a variety of sources, subjecting it to high/low limits, trim, rate limits, and providing absolute and deviation alarms.

Block parameters

Symbols used in Table 53 are explained in Table 1. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Mode	Current operating mode	Menu	
FallBack	Fallback (suppressed) operating mode	Menu	
PV	Process Variable (PV)	Eng	
SP	Resultant setpoint	Eng	
SL	Local setpoint	Eng	
TrimSP	Bias added to SL	Eng	
RemoteSP	Remote setpoint	Eng	
ComRemSP	Computer remote setpoint	Eng	
Error	PV – SP	Eng	
HR_SP	High range (for displays)	Eng	
LR_SP	Low range (for displays)	Eng	
HL_SP	High limit of SP	Eng	
LL_SP	Low limit of SP	Eng	
HL_SL	High limit of SL	Eng	
LL_SL	Low limit of SL	Eng	
Alarms			
Software	Block RAM data sumcheck error/network failure	T/F	
HighAbs	In high absolute alarm	T/F	
LowAbs	In low absolute alarm	T/F	
HighDev	In high deviation alarm	T/F	
LowDev	In low deviation alarm	T/F	
Combined	OR-ing of all Alarms bits	T/F	
HAA	High absolute alarm limit	Eng	
LAA	Low absolute alarm limit	Eng	
HDA	High deviation alarm limit	Eng	
LDA	Low deviation alarm limit	Eng	
Hyst	Hysteresis of alarms	Eng	
Dis_DP	Decimal point position (T600 front panel)	0 - 4	
DevnBar	Customises front-panel deviation bargraph	Menu	
RateLim	Rate limit for SP	Eng	
TimeBase	Rate limit time units	Menu	
Options		(A)BCD hex	
SP_Track	SP track PV select	T/F	
EnbTrack	Enable SP_Track bit	T/F	
SelRem	Remote select	T/F	D
SelCRem	Computer remote select	T/F	
DisSL_RL	Disable SL raise/lower	T/F	
SelRate	Enable rate-limiting	T/F	C
FlashPV	Cause PV bargraph to flash	T/F	
FlashSP	Cause SP bargraph to flash	T/F	
Show_SP	Force display of SP in units display of front panel	T/F	
Unused			B
Unused			
Unused			
Status		(ABC)D Hex	
BalSP	Request balance on SP change (one iteration)	T/F	
Unused			D
Unused			
Unused			

Table 53 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

Mode. (LOCAL/REMOTE/CompREM/TRACK). *Mode* is a read-only parameter showing the block’s current operating mode resulting from the various options bits. In order of priority the possible modes are: TRACK (highest priority), LOCAL, CompREM and REMOTE (lowest priority):

- **TRACK** is selected when both *Options.SP_Track* and *Options.EnbTrack* are TRUE. Note that in the T600 standard (pre-configured) strategies, *EnbTrack* is left unconnected. This allows the user to enable/disable the track feature via the front panel, by setting *EnbTrack* TRUE or FALSE, respectively.
- **LOCAL** mode is selected when both *Options.SelRem* and *Options.SelCRem* are FALSE. In local mode, the value of *SL* can be adjusted using the front panel raise/lower pushbuttons. This can be disabled by setting the *Options.DisSL_RL* bit TRUE.
- **CompREM** (computer remote) mode is selected when *Options.SelCRem* is TRUE.
- **REMOTE** mode is selected when *Options.SelRem* is TRUE, and *Options.SelCRem* is FALSE.

FallBack. (LOCAL/REMOTE/CompREM). Indicates mode of operation adopted if the current mode is deselected.

PV. Process variable.

SP. Resultant setpoint.

SL. Local setpoint. *SL* is subjected to trim, high and low limits, and rate-of-change limits before it becomes *SP*, the resultant setpoint. Setpoint limits are used in all modes of operation and act on both *SL* and the *SP* values independently.

TrimSP. Setpoint trim.

RemoteSP. Remote setpoint. Controls setpoint in remote mode only.

ComRemSP. Computer remote setpoint. Controls setpoint in CompREM mode only.

Error. This field shows the difference between *PV* and *SP*, i.e. $Error = PV - SP$.

HR_SP, LR_SP. High and low ranges, resp., for the T600 front-panel displays of setpoint, deviation, and process variable (in engineering units).

HL_SP, LL_SP. High and low limits, resp., of the resultant setpoint *SP*.

HL_SL, LL_SL. High and low limits, resp., of the local setpoint *SL*. Note that by applying independent limiting to *SL* and *SP*, the range of *SP* is not reduced by increasing the value of *TrimSP*, which would be the case with only a single set of limits.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.
- **HighAbs, LowAbs.** High and low absolute *PV* alarms set by parameters *HAA* and *LAA*, respectively, with a user-set hysteresis band (*Hyst*) on each limit. When in absolute alarm the T600 front-panel *PV* bargraph flashes. Note that it also flashes if *Options.FlashPV* is TRUE.
- **HighDev, LowDev.** High and low deviation (*Error*) alarms set by parameters *HDA* and *LDA*, respectively, with a user-set hysteresis band (*Hyst*) on each limit. A high deviation alarm occurs when $Error > HDA$, and a low deviation when $-Error > LDA$. When in deviation alarm the T600 front-panel *SP* bargraph flashes. Note that it also flashes if *Options.FlashSP* is TRUE.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

HAA, LAA. High and low absolute alarm limits respectively, on *PV*, in engineering units. These alarm settings can be displayed on the T600 front panel’s *PV* bargraph as reverse-lit LEDs, by pressing the raise and lower pushbuttons simultaneously.

HDA, LDA. High and low deviation (*Error*) alarm limits respectively, in engineering units. They allow alarms to be annunciated when *PV* deviates from *SP* by more than the specified amount. *HDA* and *LDA* can be displayed on the T600 front panel’s *SP* bargraph as reverse-lit LEDs, by pressing the raise and lower pushbuttons simultaneously.

Hyst. Specifies a hysteresis value, in engineering units, on the absolute and deviation alarms. Once an alarm has been annunciated, it is not cleared until the value causing the alarm has returned inside the limit by an amount specified by the hysteresis parameter.

Dis_DP. Specifies the position of the decimal point in the T600 front-panel 5-digit display. The permitted range of *Dis_DP* is 0 to 4, which defines the number of digits to appear after the decimal point.

DevnBar. (*Abs_PV*, 1/2/3, 1/5/10, 10/20/30). Specifies the graduations and usage of the T600 front-panel summary deviation bargraph attached to this block. **Abs_PV** configures it as an 'absolute PV' bargraph, with each LED segment representing 16.67% of the PV's range. **1/2/3**, **1/5/10**, and **10/20/30** make it a deviation bar with graduations of 1-2-3, 1-5-10 and 10-20-30 % deviation per segment, respectively.

RateLim. Rate-of-change limit for *SP*, in engineering units per second or per minute (as specified by the *TimeBase* parameter). Rate-limiting is applied only if *Options.SelRate* is TRUE, and if the block is not in track mode.

TimeBase. (Secs/Mins). Specifies time units for the rate-limiting parameter *RateLim*.

Options. Bitfield specifying the operating mode, and various other operational options of the block. The following is given in addition to *Table 53* and *Figure 32*.

- **Show_SP.** The value of *SP* appears on the T600 front panel SP-W bargraph only if the user task (loop) to which the block is assigned is selected as the main loop display. With *Options.Show_SP* TRUE, *SP* is also shown numerically in the 5-digit display.

Status.

- **BalSP.** This flag sets TRUE (for one block iteration only) if the value of *SL* has changed at this iteration, and the block is not in a remote mode. This occurs if *SL* has been written to directly, or if the raise/lower front-panel pushbuttons have been used. *BalSP* is also TRUE if the change in *SL* is due to a change into one of the remote modes or a change between remote modes.

BalSP can be connected to the *SPbal* input of the *3_TERM* block, to prevent 'kicks' in the controller output when this occurs, see *3_TERM: Incremental PID Block* section.

3_TERM: INCREMENTAL PID BLOCK

Block function

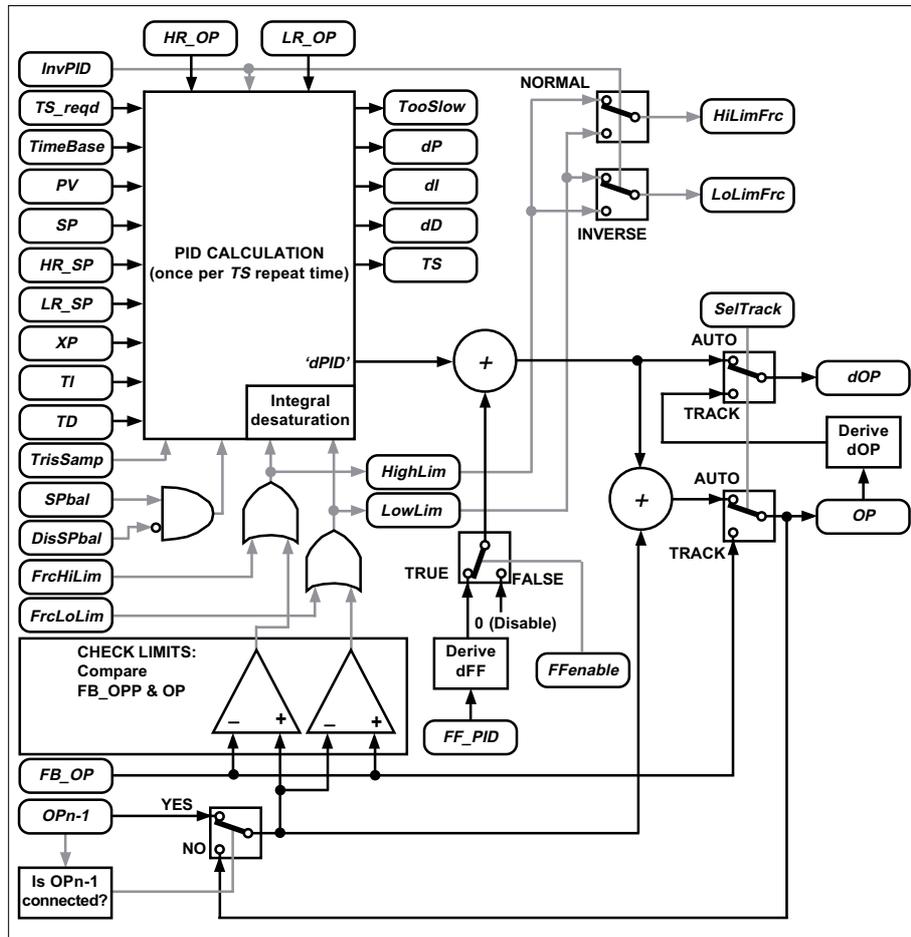


Figure 33 Block schematic

Please refer to the Block schematic in *Figure 33*. The 3_TERM block is an incremental form of Proportional-Integral-Derivative control block, intended to be used as part of the SETPOINT/3_TERM/MAN_STAT/MODE combination of control blocks, which gives more flexibility than the simple PID block. The block generates an incremental output *dOP*, suitable for driving incremental-type actuators, and also a normal output *OP*.

The 3_TERM block derives the incremental output *dOP*, and the normal output *OP*, at the *n*th iteration, as:

$$dOP_n = -(100/XP_n)[\Delta ER_n + (TS_n/TI_n)ER_n + (TD_n/TS_n)\Delta_2PV_n] + \Delta FF_PID_n,$$

and $OP_n = OP_{n-1} + dOP_n$

where *dOP*, *OP*, *PV*, *SP*, *XP*, *TS*, *TI*, *TD*, *FB_OP*, and *FF_PID* are parameters defined in the *Block specification menu* section below, and

$$\Delta FF_PID_n = FF_PID_n - FF_PID_{n-1}$$

$$ER_n = PV_n - SP_n$$

$$\Delta ER_n = ER_n - ER_{n-1}$$

$$\Delta_2PV_n = \Delta PV_n - \Delta PV_{n-1}$$

$$\Delta PV_n = PV_n - PV_{n-1}$$

$$\Delta SP_n = SP_n - SP_{n-1}.$$

This form of the PID algorithm has certain advantageous features, including:

- Tuning constants can be changed dynamically without causing ‘bumps’ in the output.
- Boundless raise/lower valve position control is easy to emulate.

- The sense of the PID control can be changed bumplessly.

The following features correspond to those in the standard PID implementation (e.g. as used in the PID block):

- Proportional kick on *SP*-changes can be removed by setting ΔSP_n , in the expression $\Delta ER_n = \Delta PV_n - \Delta SP_n$, to zero for that iteration.
- Integral desaturation is performed by replacing ER_n in the integral term by:

$$XP_n/100 \times (FB_OP_n - OP_{n-1})$$

Block parameters

Symbols used in *Table 54* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Mode	Operating mode	Menu	
PV	Process Variable (PV)	EngA	↔
SP	Control setpoint	EngA	↔
OP	PID output	EngB	↔
HR_SP	High range for PV and SP	EngA	↔
LR_SP	Low range for PV and SP	EngA	↔
HR_OP	High range for OP, FF_PID, FB_OP	EngB	↔
LR_OP	Low range for OP, FF_PID, FB_OP	EngB	↔
OP _{n-1}	Previous value of OP	EngB	↔
dOP	Change in OP this iteration	EngB	↔
dP	Change due to P term	EngB	↔
dI	Change due to I term	EngB	↔
dD	Change due to D term	EngB	↔
Alarms			↔
Software	Block RAM data sumcheck error / network failure	T/F	
TooSlow	If TS > (TI/10)	T/F	
Combined	OR-ing of all Alarms bits	T/F	
TimeBase	Selects units for TI, TD, TS	Menu	↔
XP	Proportional band	%	↔
TI	Integral time	Secs/Mins	↔
TD	Derivative time	Secs/Mins	↔
TS_reqd	Requested value of TS	Secs/Mins	↔
TS	Algorithm sampling time	Secs/Mins	↔
Options		(A)BCD Hex	↔
TrigSamp	Trigger PID execution	T/F	1
InvPID	Invert sense of PID	T/F	2
SPbal	Set dSP to zero (balance)	T/F	4
DisSPbal	Disable SPbal	T/F	8
FFenable	Enable the feedforward term	T/F	1
SelTrack	Disable PID (select track mode)	T/F	2
SelOnOff	Selects on/off control mode	T/F	4
HiDband	High deadband for on/off mode	T/F	8
LoDband	Low deadband for on/off mode	T/F	1
AbsNol	Use absolute PD algorithm when TI is zero	T/F	2
FrcHiLim	Force PID to act as if in high limit	T/F	4
FrcLoLim	Force PID to act as if in low limit	T/F	8
Status		(AB)CD Hex	↔
LowLim	OP is below low limit	T/F	1
HighLim	OP is above high limit	T/F	2
ThisTime	PID update this iteration	T/F	4
HiLimFrc	Force master controller to act as if in high limit	T/F	8
LoLimFrc	Force master controller to act as if in low limit	T/F	1
Unused			2
Unused			4
Unused			8
FF_PID	Feedforward	EngB	↔
FB_OP	Feedback (after limiting)	EngB	↔
DeadBand	Deadband for on/off control mode	EngA	↔

Table 54 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

Mode. (AUTO/TRACK) Current operating mode.

PV. Process variable.

SP. Control setpoint.

OP. PID output. In automatic mode, *OP* is derived by adding *dOP* (the change in PID output) to the *OP*-value from the previous iteration, *OP_{n-1}*. *OP_{n-1}* can be sourced internally from *OP* itself, or externally from the *OP_{n-1}* input if connected (automatically detected by the block). See *Figure 33*. In track mode, *OP* simply tracks the *FB_OP* input.

HR_SP, LR_SP. High and low ranges, resp., of *SP* and *PV* in engineering units.

HR_OP, LR_OP. High and low ranges, resp., of *OP*, *FF_PID*, and *FB_OP* in engineering units.

OP_{n-1}. This input can be connected to an external ‘measured position’ output as a source of *OP_{n-1}* from which is generated a valid current *OP_n*-value (e.g. for display purposes). Alternatively, if *OP_{n-1}* is unconnected, the internal *OP*-value from the previous iteration of the block (*OP_{n-1}*) is automatically used.

dOP. Change in the value of *OP* at this block iteration, i.e. $OP_n - OP_{n-1}$.

dP. Change in the value of the proportional (P) term at this block iteration.

dI. Change in the value of the integral (I) term at this block iteration.

dD. Change in the value of the derivative (D) term at this block iteration.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.
- **TooSlow.** If the value of the PID algorithm sampling time *TS* exceeds $TI/10$, the algorithm repeat rate is considered to be dangerously low, and the *TooSlow* alarm trips. This can be used to trap situations where dynamic tuning results in the 3_TERM block requiring a faster loop repeat rate than is being requested.

NOTE *TooSlow* is disabled when *TI* is set to zero.

- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

TimeBase. (Secs/Mins). Specifies time units for *TI*, *TD*, and *TS*.

XP. Proportional band (%).

TI. Integral time constant, in seconds or minutes according to the value of *TimeBase*. Setting $TI = 0$ disables the integral term, i.e. configures incremental PD mode.

TD. Derivative time constant, in seconds or minutes according to *TimeBase*.

TS_reqd. Specifies a requested value of *TS*, the algorithm sampling time, in seconds or minutes according to the value of *TimeBase*. This is not necessarily the actual value adopted by *TS* (see next section).

TS. PID algorithm sampling time, in seconds or minutes according to the value of *TimeBase*. *TS* takes the larger of the following two values:

- The larger of *TI* and *TD*, divided by 512.
- The value specified in *TS_reqd*.

If the value calculated for *TS*, which is always the actual interval between PID executions, in this way is not a multiple of the loop repeat rate, it is rounded up to the next nearest multiple of loop repeat rate.

If *TS* exceeds the block repeat time, *PV* and *SP* are sampled at intervals of *TS* for use in the PID algorithm (and *SPbal* latches between PID executions). Also, with *TS* greater than the block repeat time, a rising edge to *TrigSamp* triggers a PID algorithm execution and resets the *TS* timer.

Options. A bitfield allowing inputs to control the operation of the block.

- **TrigSamp.** Triggers an execution of the PID algorithm.

- **InvPID.** Selects 3-Term control action sense. With *InvPID* FALSE an increase in the error term ($PV - SP$) produces a corresponding decrease in the 3-Term output to compensate. With *InvPID* TRUE, increasing the error produces a corresponding increase in the 3-Term output to compensate.
- **SPbal.** Used to avoid ‘proportional kick’ in the control output that can occur with *SP*-changes. It does this by deriving the proportional term from *PV*-changes rather than error-changes. Specifically, a TRUE input to *SPbal* sets ΔSP_n (i.e. $SP_n - SP_{n-1}$) to zero for one iteration of the PID algorithm. Note that the *Options.DisSPbal* bit must be FALSE for *SPbal* to function.
SPbal is intended to be connected from the SETPOINT block’s *Status.BalSP* output, so that manual writes to *SL* - locally or via the supervisor communications - do not cause controller output ‘kicks’.
- **DisSPbal.** TRUE disables the action of the *SPbal* bit.
- **FFenable.** Enables the application of the feedforward term, *FF_PID*, to the *dOP* output. Note that in absolute PD mode (*Options.AbsNoI* TRUE), feedforward cannot be disabled and *FFenable* is ignored.
- **SelTrack.** Selects TRACK mode. This disables PID action and forces *OP* to track *FB_OP* (feedback output, after any limiting), and *dOP* to be derived from this.
With *SelTrack* TRUE, all intermediate accumulators are zeroed and internal stores of previous iteration values of *PV* and *FF_PID* are set to equal their current values (e.g. $PV_{n-1} = PV_n$) to ensure fully bumpless transfer back to PID control.
- **SelOnOff.** Selects on/off control mode, with a deadband (hysteresis) specified by *DeadBand*, operating above or below the setpoint, as specified by the *HiDband* and *LoDband* bits.

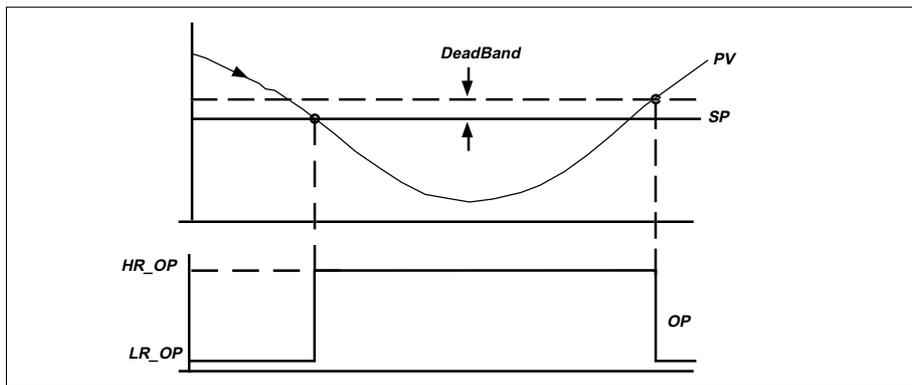


Figure 34 3_TERM block on/off control - HiDband TRUE, LoDband FALSE

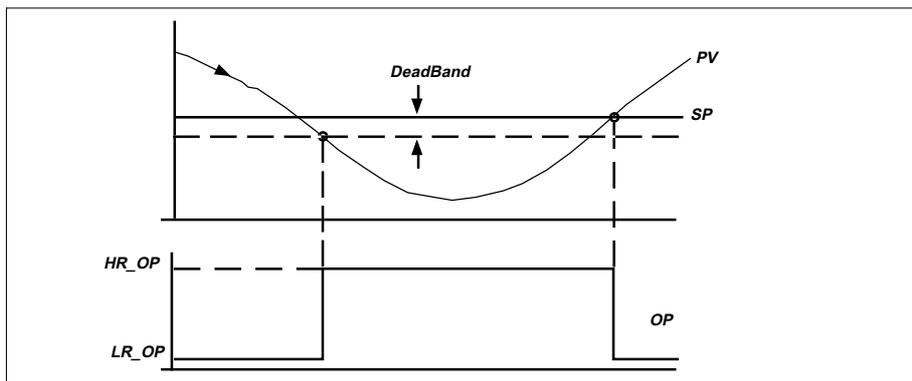


Figure 35 3_TERM block on/off control - HiDband FALSE, LoDband TRUE

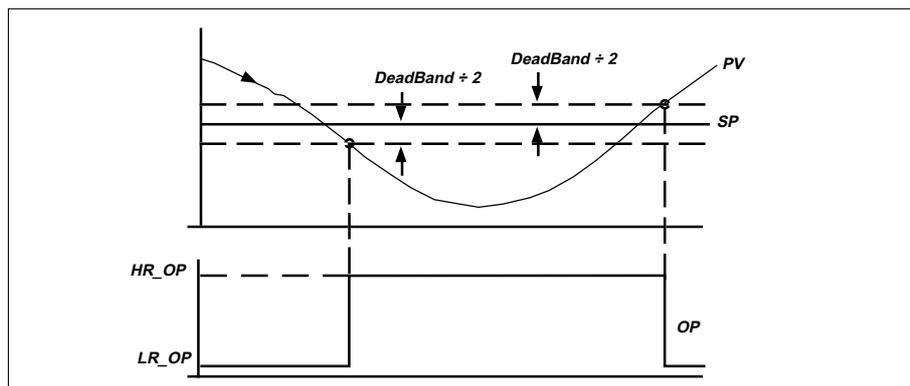


Figure 36 3_TERM block on/off control - HiDband TRUE, LoDband TRUE

- **HiDband, LoDband.** Select a 'high' deadband (Figure 34) or a 'low' deadband (Figure 35), respectively, in on/off control mode, of width specified by *DeadBand*. With both bits TRUE, a pair of deadbands operate about *SP* (Figure 36). Both bits FALSE disables the deadband completely.
- **AbsNoI.** With *TI* = 0 to disable the integral term, set *AbsNoI* TRUE to select an absolute (non-incremental) PD algorithm in which *dP*, *dI*, *dD*, and *dOP* are disabled (zero).
- **FrcHiLim, FrcLoLim.** TRUE forces the PID algorithm to act as if the output is being high-limited, or low-limited, respectively, and integral desaturation is therefore applied.

Status. Bitfield indicating the status of the PID algorithm.

- **LowLim.** TRUE indicates that *OP* is below a low limit, i.e. below the fed-back (limited) output, or is being forced to act as if it were (by *FrcLoLim*).
- **HighLim.** TRUE indicates that *OP* is above a high limit, i.e. above the fed-back (limited) output, or is being forced to act as if it were (by *FrcHiLim*).
- **ThisTime.** TRUE indicates that the PID algorithm has been updated during this iteration of the 3_TERM block.
- **HiLimFrc.** TRUE indicates that the *HighLim* parameter is TRUE (or with inverse-acting output, that *LowLim* is TRUE). This output is intended to be used to force a master controller block to act as if in limit, e.g. in a cascade pair.
- **LoLimFrc.** TRUE indicates that the *LowLim* parameter is TRUE (or with inverse-acting output, that *HighLim* is TRUE). This output is intended to be used to force a master controller block to act as if in limit, e.g. in a cascade pair.

FF_PID. Feed-forward. Used to add a bias to the 3-term output. This is sometimes called 'manual reset'.

NOTE. *FF_PID* is ranged $\pm(HR_OP - LR_OP)$.

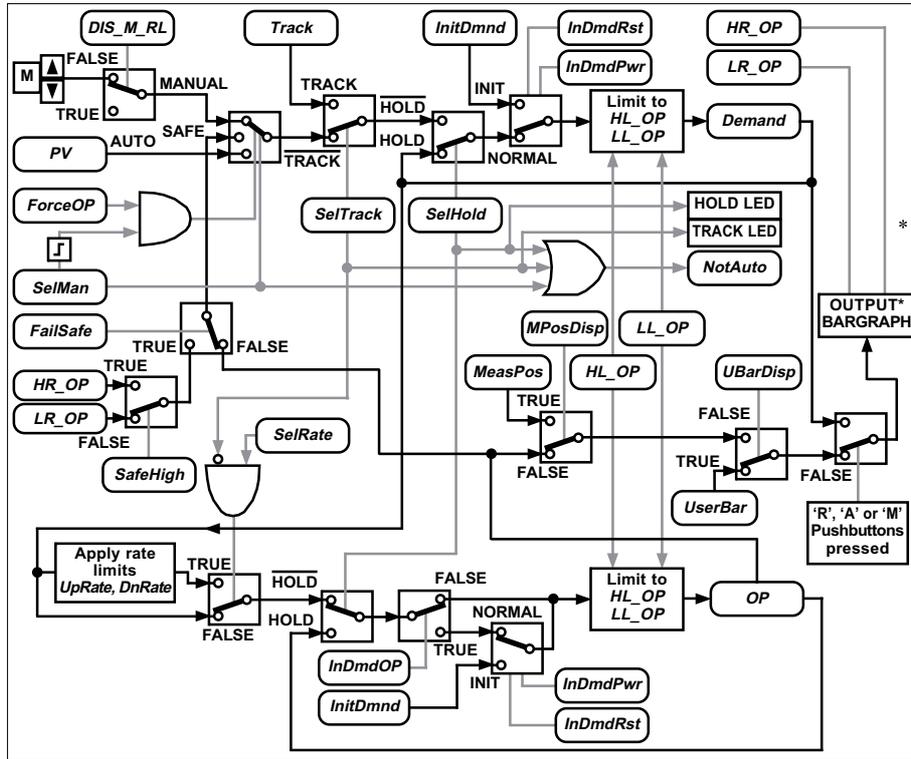
FB_OP. A feedback value of *OP* from the last iteration of the PID block, after it has been subjected to any external limiting. *FB_OP* is used to hold the output value when in track mode, and also to detect output-limiting - by comparing it with *OP* - so that integral desaturation can be applied.

DeadBand. Specifies the breadth of the deadband(s) to be applied when the block is in on/off control mode. *Options.HiDband* and *Options.LoDband* determine how the deadband(s) operate(s) about *SP*, see Figure 34, Figure 35, and Figure 36.

NOTE With both *HiDband* and *LoDband* FALSE, deadband action is disabled; with both parameters TRUE, half the deadband is applied to each side of *SP*.

MAN_STAT: MANUAL STATION BLOCK

Block function



*Linked to the real fascia display only if the associated loop is the one selected for main loop display

Figure 37 Block schematic

Please refer to *page 140* for usage implications when using this block and the schematic in *Figure 37*. The MAN_STAT block is intended to be used as part of the SETPOINT/3_TERM/MAN_STAT/MODE combination of control blocks, which gives more flexibility than the simple PID block.

In automatic mode the output *OP* follows the input *PV*, typically a 3_TERM block output, subject to high/low- and rate-limiting. In manual mode the operator can use the raise/lower T600 front-panel pushbuttons to vary *OP*, within the limits. In track mode (highest priority), *OP* is derived from the *Track* parameter and remains high/low- (but not rate-) limited. A hold mode is provided to ‘freeze’ both *OP* and the *Demand* parameter. The block’s *MeasPos* and *MPosDisp* parameters let you select between displaying a ‘measured position’ or *OP* on the T600 output bargraph, and with *UBarDisp* you can drive each bargraph segment directly from the strategy (via the *UserBar* bitfield). The block also drives the **H** and **T** LEDs below the associated loop’s deviation bargraph.

Note that the output bargraph is driven only if this block’s user task is the one selected for main loop display.

Block parameters

Symbols used in *Table 55* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Mode	Operating mode	Menu	
FallBack	Fallback (suppressed) operating mode	Menu	
PV	Manual station input	Eng	
Track	Output track value	Eng	
Demand	Output (before rate limiting)	Eng	
MeasPos	Measured position (for display)	Eng	
OP	Manual station output	Eng	
HR_OP	High range (for output bargraph & failsafe value)	Eng	
LR_OP	Low range (for output bargraph & failsafe value)	Eng	
HL_OP	Output high limit	Eng	
LL_OP	Output low limit	Eng	
InitDmnd	Value for Demand/OP at start of task	Eng	
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
Combined	OR-ing of all Alarms bits	T/F	
UpRate	Rising rate limit	Eng	
DnRate	Falling rate limit	Eng	
TimeBase	Specifies rate-limiting time units	Menu	
Dis_DP	Decimal point position	0 - 4	
Options		ABCD hex	
SelTrack	Track mode select	T/F	1
SelHold	Hold mode select	T/F	2
SelMan	TRUE selects manual; FALSE selects automatic	T/F	4
SelRate	Enable rate limiting	T/F	8
Dis_M_RL	Disable changes via 'M' pushbutton	T/F	1
ForceOP	Enables failsafe action	T/F	2
FailSafe	Selects failsafe value for OP (TRUE = HR or LR)	T/F	4
SafeHigh	'Safe' condition is high	T/F	8
FlashOP	Flash the OP bargraph	T/F	1
SelOnOff	Select on/off control (demand either HL or LL)	T/F	2
InDmdRst	Demand = InitDmnd at task re-start (not power-up)	T/F	4
InDmdPwr	Demand = InitDmnd at power-up	T/F	8
UBarDisp	Connect UserBar field to fascia output bargraph	T/F	1
MPosDisp	Connect MeasPos field to fascia output bargraph	T/F	2
InDmdOP	OP copies Demand at task re-start & power-up	T/F	4
Unused			8
Status		(ABCD) Hex	
NotAuto	Not in auto mode (i.e. in hold, track, or manual)	T/F	1
Unused			2
Unused			4
Unused			8
UserBar	Direct strategy drive of output bargraph segments	(A)BCD hex	
Bit0	Bargraph segment 0 lit (left-most)	T/F	1
Bit1	Bargraph segment 1 lit	T/F	2
Bit2	Bargraph segment 2 lit	T/F	4
Bit3	Bargraph segment 3 lit	T/F	8
Bit4	Bargraph segment 4 lit	T/F	1
Bit5	Bargraph segment 5 lit	T/F	2
Bit6	Bargraph segment 6 lit	T/F	4
Bit7	Bargraph segment 7 lit	T/F	8
Bit8	Bargraph segment 8 lit	T/F	1
Bit9	Bargraph segment 9 lit (right-most)	T/F	2
			4
			8

Table 55 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

Mode. (MANUAL/AUTO/TRACK/HOLD/INIT). Current (read-only) operating mode. The corresponding *Options* parameters (*SelMan*, *SelTrack*, and *SelHold*) must be used to select the mode. INIT mode is indicated briefly at power-up or task start-up if *InitDmnd* has been configured to supply an initial *Demand* value.

Fallback. (MANUAL/AUTO/TRACK). Indicates next (suppressed) operating mode.

PV. Manual station input (process variable).

Track. Controls *Demand* and *OP* in track mode.

Demand. Manual station output before any rate-limiting, displayed on the T600 front-panel output bargraph when the M, A, or R pushbuttons are pressed.

MeasPos. Externally-generated ‘measured position’ value, displayed on the T600 front-panel output bargraph subject to the status of *MPosDisp* and *UBarDisp*.

OP. Output value from the station, displayed on the T600 front-panel output bargraph subject to the status of *MPosDisp* and *UBarDisp*.

HR_OP, LR_OP. High and low range for the T600 output bargraph display. *HR_OP* and *LR_OP* define the 100% and 0% displays, respectively. These parameters, together with *SafeHigh*, also define the ‘failsafe’ value adopted by *Demand* when the block is switched into manual mode by a rising edge input to *Options.SelMan*.

HL_OP, LL_OP. High and low output limits acting on both *OP* and *Demand*.

InitDmnd. Value adopted by *Demand* (subject to high/low limits) when a power-up or task restart occurs, if the *InDmdPwr* or *InDmdRst* parameters, respectively, are TRUE.

OP also adopts the *InitDmnd* value (subject to limits) at power-up or task-restart, according to *InDmdPwr* and *InDmdRst*, provided the *InDmdOP* parameter is TRUE. These initialisations of the *Demand* and *OP* fields can be used to ensure a bumpless startup.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.

- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

UpRate. Specifies maximum rate of increase of *OP* (i.e. rate limit) in engineering units/time. Time units are specified by the *TimeBase* parameter. Limiting is applied only when *SelRate* is TRUE and *SelTrack* is FALSE (i.e. not in track mode).

DnRate. Specifies maximum rate of decrease of *OP* (i.e. rate limit) in engineering units/time. Time units are specified by the *TimeBase* parameter. Limiting is applied only when *SelRate* is TRUE and *SelTrack* is FALSE (i.e. not in track mode).

TimeBase. (Secs/Mins) Selects time units for rate limit functions, *UpRate* and *DnRate*.

Dis_DP. Specifies the position of the decimal point in the T600 front-panel 5-digit display. The permitted range is 0 to 4, which defines the number of digits to appear after the decimal point.

Options. The following information is given in addition to *Table 55*.

- **ForceOP, FailSafe, SafeHigh.** With *ForceOP* TRUE, a *SelMan* FALSE-to-TRUE transition causes the block to go into a ‘safe’ mode for one iteration only, before normal manual mode takes over. In this transient mode, *Demand* adopts a value that depends on the states of *FailSafe* and *SafeHigh*, as shown in the following table.

	FailSafe = FALSE	FailSafe = TRUE
SafeHigh = FALSE	Demand = OP	Demand = LR_OP*
SafeHigh = TRUE	Demand = OP	Demand = HR_OP*

*Limited by *HL_OP, LL_OP*

This facility can be used to implement ‘manual/low on o/c PV’ or similar strategies.

Status.

- **NotAuto.** Indicates that the manual station is not operating in auto mode.

UserBar. The ten digital inputs *UserBar.Bit0* (left-most segment) to *UserBar.Bit9* (right-most segment) allow the strategy to drive the T600 front-panel output bargraph segments directly and individually. *UserBar* is connected to the real front panel only if the *Options.UBarDisp* bit is TRUE, and the block's associated user task is the one selected for main loop display.

MODE: MODE BLOCK

Block function

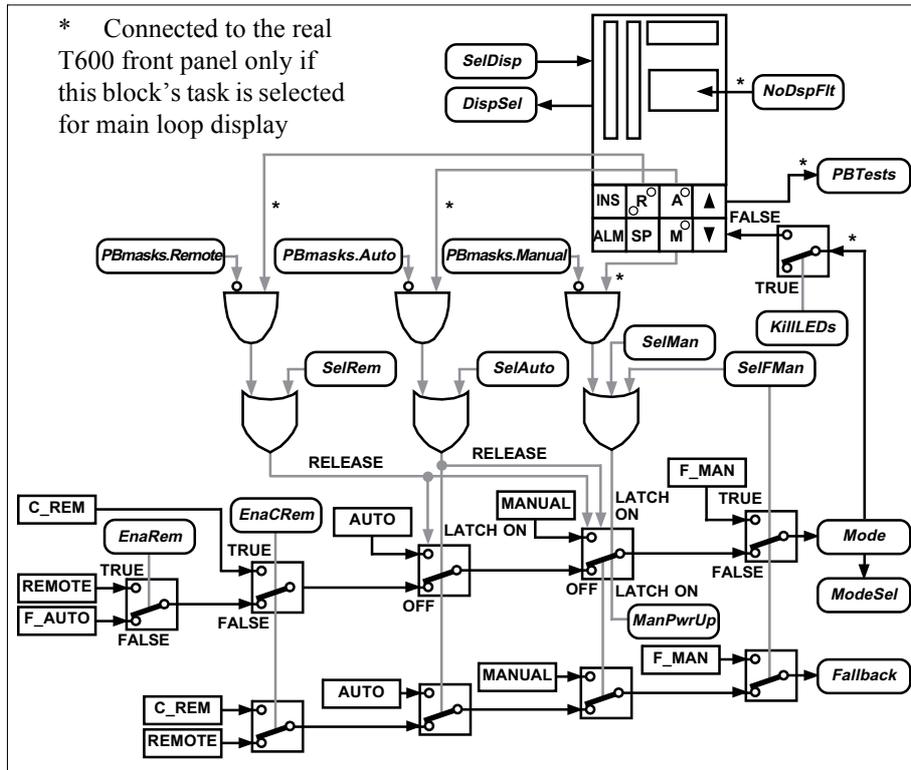


Figure 38 Block schematic

Please refer to *page 140* for usage implications when using this block and the schematic in *Figure 38*. The MODE block is intended to be used as part of the SETPOINT/3_TERM/MAN_STAT/MODE block combination, which gives more flexibility than the simple PID block.

The block operates a ‘prioritised latching mode control’ algorithm, schematised in *Figure 38* by a series of switches, some of them latching. The position of each switch along the route to the *Mode* parameter determines the priority of the operating mode selected by that switch. E.g. F_MAN (forced manual) mode, while selected by *SelfMan* being TRUE, overrides all other modes because its switch is closest to the *Mode* parameter. Conversely, F_AUTO (forced auto) and REMOTE modes have the lowest priority as their switch is furthest from *Mode* and so can be overridden by any of the others.

The MANUAL and AUTO switches are latching, i.e. they remain ‘on’ even if the TRUE condition that turned them on goes FALSE. If this happens, a latched MANUAL switch can be released by selecting either AUTO or ‘remote’, and both MANUAL and AUTO are released by selecting ‘remote’.

The operating mode actually in force is indicated by *Mode*, and is available as an output via the *ModeSel* bits. The fallback mode, held in *FallBack*, is not available as an output and is for monitoring purposes only. The source of *FallBack* is schematised in *Figure 38* as a second series of switches ‘ganged’ to the corresponding mode selection switches.

When the associated user task has been assigned for main loop display, the MODE block drives the R, A, and M T600 front-panel pushbutton LEDs, and the mode characters under the loop’s deviation bargraph. The pushbutton LEDs can be disabled via the *KilledLEDs* input (in the *Options* bitfield). R, A, and M pushbutton masks are available to prevent unwanted operator mode-changes, via the *PBmasks* inputs. All eight front-panel pushbuttons are latched on a per-task-execution basis, so that fleeting button-pushes are not missed by the task. These latched depressions are also available as outputs via the *PBTests* bitfield, for use by other blocks in the task.

Block parameters

Symbols used in *Table 56* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Mode	Operating mode	Menu	
FallBack	Fallback mode	Menu	
SelMode		(AB)CD hex	
EnaRem	Enable remote mode	T/F	
EnaCRem	Enable computer remote mode	T/F	
SelRem	Select remote mode	T/F	
SelAuto	Select auto mode	T/F	
SelMan	Select manual mode	T/F	
SelFMan	Select forced manual mode	T/F	
Unused			
Unused			
Options		(ABC)D hex	
SelDisp	Force this loop to display	T/F	
KillLEDs	Disable M, A, & R LEDs	T/F	
ManPwrUp	Force to manual at power up	T/F	
Unused			
PBmasks			
Remote	Disable pushbutton select of remote	T/F	
Auto	Disable pushbutton select of auto	T/F	
Manual	Disable pushbutton select of manual	T/F	
Unused			
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
Hardware	I/O Module failure / transmitter PSU false	T/F	
Combined	OR-ing of all Alarms bits	T/F	
ModeSel		(AB)CD Hex	
RemSel	Remote mode selected	T/F	
AutoSel	Auto mode selected	T/F	
MorFMsel	Manual or FM mode selected	T/F	
FAutoSel F.	Auto mode (back to R) sel.	T/F	
CRemSel	Computer remote selected	T/F	
Unused			
Unused			
Unused			
Status		(ABC)D Hex	
DispSel	This loop is on display	T/F	
Unused			
Unused			
Unused			
PBtests			
Down	Lower pushbutton depressed	T/F	
Up	Raise pushbutton depressed	T/F	
Remote	R pushbutton depressed	T/F	
Auto	A pushbutton depressed	T/F	
Manual	M pushbutton depressed	T/F	
SetPoint	SP pushbutton depressed	T/F	
Alarm	ALM pushbutton depressed	T/F	
Inspect	INS pushbutton depressed	T/F	

Table 56 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

Mode. (MANUAL/F_MAN/AUTO/C_REM/REMOTE/F_AUTO). The block’s currently-selected operating mode. Manual and auto can be selected directly via the *Mode* field, but the others are read-only. In order of priority the possible modes are: F_MAN (highest priority), MANUAL, AUTO, C_REM, REMOTE and F_AUTO (lowest priority):

- **F_MAN** (forced manual) is selected only while *SelMode.SelFMan* is TRUE, and overrides all other modes.
- **MANUAL** mode can be selected by writing directly to *Mode*. It is also selected either by making *SelMode.SelMan* TRUE, or by pressing the **M** front-panel pushbutton (provided *PBmasks.Manual* is FALSE). Manual mode ‘latches on’ when selected and remains so until released (by selecting AUTO or a remote mode). Note that manual mode also latches on when *SelMode.SelFMan* is made TRUE, but is overridden by F_MAN until *SelFMan* is reset to FALSE, at which point MANUAL takes over.
- **AUTO** mode can be selected by writing directly to *Mode*. It is also selected either by making *SelMode.SelAuto* TRUE, or by pressing the **A** front-panel pushbutton (provided *PBmasks.Auto* is FALSE). Automatic mode ‘latches on’ when selected and remains so until released (by selecting a remote mode).
- **C_REM** (computer remote) mode is active only while *SelMode.EnaCRem* is TRUE, and no higher priority mode is operative. Make *SelMode.SelRem* TRUE, or press the **R** front-panel pushbutton (provided *PBmasks.Remote* is FALSE), to release AUTO or MANUAL modes and allow C_REM to operate.
- **REMOTE** mode is active only while *SelMode.EnaRem* is TRUE, *.EnaCRem* is FALSE, and no higher priority mode is operative.
- **F_AUTO** (forced automatic) mode is active only while all *SelMode* bits are FALSE, and AUTO or MANUAL is not ‘latched on’.

FallBack. (MANUAL/F_MAN/AUTO/C_REM/REMOTE). The block’s fallback mode, for monitoring purposes only. *Figure 38* shows schematically how the fallback mode is derived according to the value of *Mode*.

SelMode. Bitfield for enabling/selecting the block’s *Mode* value. See *Mode* section above for explanations of how these bits act, and also *Figure 38* and *Table 56*. The mode can also be changed (to AUTO or MANUAL) by writing to the *Mode* field directly, provided all *SelMode* mode select bits (*SelRem*, *SelAuto*, *SelMan*, or *SelFMan*) are FALSE.

Owing to the latch/release properties of the MANUAL and AUTO ‘selector switches’, the *SelRem*, *SelAuto*, *SelMan*, and *SelFMan* bits need only be transiently TRUE for mode-changes to occur.

NOTE A T600 front panel mode pushbutton can cause a change of mode only if it is not masked, and if all *SelMode* mode select bits (*SelRem*, *SelAuto*, *SelMan*, or *SelFMan*) are FALSE. Communications writes to these four fields are latching (cleared at block update time), so that mode-change writes by a supervisory system are processed even when these fields are connected to other blocks in the LIN Database.

Options. Bitfield for selecting various T600 front-panel display options (and mode adopted at power-up).

- **SelDisp.** TRUE forces the user task to which the block is assigned to occupy the T600 front panel’s main loop display. Note that if more than one user task has its *SelDisp* TRUE at the same time, the highest priority task is the one displayed. (User task 1 has the highest priority, and user task 4 the lowest.)
- **KilledDs.** TRUE disables the **R**, **A**, and **M** T600 front-panel pushbutton LEDs (only if this user task is occupying the main loop display).
- **ManPwrUp.** TRUE forces the block to adopt MANUAL mode at T600 power-up.
- **NoDspFlt.** TRUE disables the first-order 1-second time constant filter normally operating on the T600 front-panel 5-digit display. This is useful when a setpoint is being displayed, which, unlike a process variable, does not usually require the filtering action.

PBmasks. Bitfield for masking, i.e. inhibiting the action of, the **R**(emote), **A**(uto), and **M**(anual) T600 front-panel mode-change pushbuttons. A TRUE *Remote*, *Auto*, or *Manual* bit masks the corresponding push button, when the user task containing the MODE block is assigned for main loop display. If the loop is not being displayed, the pushbuttons have no effect anyway.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.

- **F_Manual.** TRUE if block is in forced manual (F_MAN) mode.
 - **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.
- ModeSel.** Bitfield indicating the block's active mode, i.e. the value of the *Mode* parameter. See *Table 56*. Only one *ModeSel* bit can be active at one time. Note that manual and forced manual modes are not separately flagged.
- Status.** Bitfield indicating (*DispSel* TRUE) if the user task containing this MODE block is occupying the T600 front panel's main loop display.
- PBTests.** Bitfield indicating (by TRUE bits) which of the eight T600 front-panel pushbuttons are currently being pressed. See *Table 56*. *PBTests* operates only when the MODE block's user task is occupying the T600 main loop display; if not, all bits remain FALSE.

PID_CONN: PID CONNECTION BLOCK

Block function

The PID_CONN block has the same template (block specification menu) as the PID block, but is used as a ‘faceplate block’ for the SETPOINT/3_TERM/MAN_STAT/MODE block combination. The PID_CONN block automatically associates itself with the appropriate blocks in the same loop; it can then be cached where it may be used for faceplate display.

Most of the PID_CONN fields are direct copies of the equivalent fields in the associated SETPOINT, 3_TERM, MAN_STAT, or MODE blocks. Writes, or attempted writes, to these fields are re-directed to the corresponding fields in the actual source block. Some PID_CONN fields are more complicated logical derivatives of source fields. Note that if the PID_CONN block cannot find sources for any of its fields, these fields are not overwritten and become available for local data storage. The PID_CONN field sources are listed in *Table 57*.

NOTE Wiring. When using a PID_CONN block to concentrate SETPOINT/3_TERM/MAN_STAT/MODE blocks, you should wire consistently. Connect strategy wiring into either the PID_CONN block, or the true source blocks, *but not both*. This applies equally to wiring into cached copies of remote source blocks, especially when bitfields are involved. Wiring into individual bits of bitfields at different locations can cause some wires to fail.

Alarm processing

Although the values of HAA, LAA, HDA, and LDA are copies of the equivalent values in the SETPOINT block, the alarm processing is performed locally in the PID_CONN block. This allows different alarm priorities to be selected, making it possible to process alarms in one block only.

PID_CONN field	Source
Mode	IF MAN_STAT.Mode = Hold or Track, THEN Mode = MAN_STAT.Mode ELSE IF MODE.Mode = C_REM, THEN Mode = REMOTE ELSE Mode = MODE.Mode
FallBack	IF MAN_STAT.Mode = NOT(Hold or Track) THEN IF MODE.FallBack = C_REM, THEN FallBack = REMOTE ELSE FallBack = MODE.FallBack ELSE IF MODE.Mode = C_REM, THEN FallBack = REMOTE ELSE FallBack = MODE.Mode
PV	SETPOINT.PV
SP	SETPOINT.SP
OP	MAN_STAT.Demand
SL	SETPOINT.SL
TrimSP	SETPOINT.TrimSP
RemoteSP	SETPOINT.RemoteSP/ComRemSP (depends on Mode)
Track	MAN_STAT.Track
HR_SP	SETPOINT.HR_SP
LR_SP	SETPOINT.LR_SP
HL_SP	SETPOINT.HL_SP
LL_SP	SETPOINT.LL_SP
HR_OP	MAN_STAT.HR_OP
LR_OP	MAN_STAT.LR_OP
HL_OP	MAN_STAT.HL_OP
LL_OP	MAN_STAT.LL_OP
HAA	SETPOINT.HAA
LAA	SETPOINT.LAA
HDA	SETPOINT.HDA
LDA	SETPOINT.LDA
TimeBase	3_TERM.TimeBase
XP	3_TERM.XP
TI	3_TERM.TI
TD	3_TERM.TD

Continued...

PID_CONN field		Source	
<i>Continued...</i>			
Options	.InvPID	3_TERM.Options.InvPID	
	.SL_Track	SETPOINT.Options.EnbTrack	
	.IntBalSL	NOT(3_TERM.Options.DisSPbal)	
	.IntBal	3_TERM.Options.SPbal	
SelMode	.SelHold	MAN_STAT.Options.SelHold	
	.SelTrack	MAN_STAT.Options.SelTrack	
	.SelRem	MODE.SelMode.SelRem	
	.EnaRem	MODE.SelMode. (EnaRem OR EnaCRem)	
	.SelAuto	MODE.SelMode.SelAuto	
	.SelMan	MODE.SelMode.SelMan	
	.SelFMan	MODE.SelMode.SelFMan	
ModeSel	.EnaRem	NOT(PID_CONN.ModeSel.HoldSel OR PID_CONN.ModeSel.TrackSel OR PID_CONN.ModeSel.ManSel OR (PID_CONN.FallBack = Manual))	
	'Sel' bits	.HoldSel	PID_CONN.SelMode.SelHold
		.TrackSel	PID_CONN.SelMode.SelTrack
		.RemSel	MODE.ModeSel.(RemSel OR CRemSel)
		.AutoSel	MODE.ModeSel.AutoSel
		.ManSel	MODE.ModeSel.MorFMsel AND (MODE.Mode = Manual)
		.FAutoSel	MODE.ModeSel.FAutoSel
ModeAct	.FManSel	MODE.ModeSel.MorFMsel AND (MODE.Mode = Forced Manual)	
	.NotRem	NOT(PID_CONN.Mode = REMOTE)	
	.HoldAct		
	.TrackAct	IF ModeSel.HoldSel = TRUE THEN ModeAct.HoldAct = TRUE, and rest of these seven 'Act' bits FALSE,	
	.RemAct		
	.AutoAct	ELSE IF ModeSel.TrackSel = TRUE THEN ModeAct.TrackAct = TRUE, and rest of these seven 'Act' bits FALSE,	
	.ManAct		
.FAutoAct	ELSE ModeAct.('Act' bits) = ModeSel.('Sel' bits)		
.FManAct			
FF_PID		3_TERM.FF_PID	
FB_OP		3_TERM.FB_OP	

Table 57 PID_CONN block field sources

PID_LINK: PID LINKING BLOCK

Block function

The PID_LINK block has almost the same template (block specification menu) as the PID_CONN and PID blocks, and is used as a 'faceplate block' for the SETPOINT/3_TERM/MAN_STAT/MODE block combination. You link the PID_LINK block to these four control blocks by entering their tagnames in four special fields, *SPblock*, *T3block*, *MSblock*, and *MDblock*, respectively. The block can then be cached to a supervisory computer where it may be used for faceplate display. NOTE. The PID_LINK block must be running in the same database as the control blocks that it links to, i.e. local to the SETPOINT/3_TERM/MAN_STAT/MODE blocks, and not cached. The PID_LINK block should also be local to any blocks that reference it, e.g. the SL6366R or SL6360R blocks.

Most of the PID_LINK fields are direct copies of the equivalent fields in the associated SETPOINT, 3_TERM, MAN_STAT, or MODE blocks. Writes, or attempted writes, to these fields are re-directed to the corresponding fields in the actual source block. Some PID_LINK fields are more complicated logical derivatives of source fields. Note that if the PID_LINK block cannot find sources for any of its fields, these fields are not overwritten and become available for local data storage. The PID_LINK field sources (and the four tagname fields) are listed in *Table 58*.

NOTE **Wiring.** When using a PID_LINK block to concentrate SETPOINT/3_TERM/MAN_STAT/MODE blocks, you should wire consistently. Connect strategy wiring into either the PID_LINK block, or the true source blocks, **but not both**. This applies equally to wiring into cached copies of remote source blocks, especially when bitfields are involved. Wiring into individual bits of bitfields at different locations can cause some wires to fail.

Alarm processing

Although the values of *HAA*, *LAA*, *HDA*, and *LDA* are copies of the equivalent values in the SETPOINT block, the alarm processing is performed locally in the PID_LINK block. This allows different alarm priorities to be selected, making it possible to process alarms in one block only.

PID_LINK field	Source
Mode	IF MAN_STAT.Mode = Hold or Track, THEN Mode = MAN_STAT.Mode ELSE IF MODE.Mode = C_REM, THEN Mode = REMOTE ELSE Mode = MODE.Mode
FallBack	IF MAN_STAT.Mode = NOT(Hold or Track) THEN IF MODE.FallBack = C_REM, THEN FallBack = REMOTE ELSE FallBack = MODE.FallBack ELSE IF MODE.Mode = C_REM, THEN FallBack = REMOTE ELSE FallBack = MODE.Mode
PV	SETPOINT.PV
SP	SETPOINT.SP
OP	MAN_STAT.Demand
SL	SETPOINT.SL
TrimSP	SETPOINT.TrimSP
RemoteSP	SETPOINT.RemoteSP/ComRemSP (depends on Mode)
Track	MAN_STAT.Track
HR_SP	SETPOINT.HR_SP
LR_SP	SETPOINT.LR_SP
HL_SP	SETPOINT.HL_SP
LL_SP	SETPOINT.LL_SP
HR_OP	MAN_STAT.HR_OP
LR_OP	MAN_STAT.LR_OP
HL_OP	MAN_STAT.HL_OP
LL_OP	MAN_STAT.LL_OP
HAA	SETPOINT.HAA
LAA	SETPOINT.LAA
HDA	SETPOINT.HDA
LDA	SETPOINT.LDA

Continued...

PID_LINK field		Source
<i>Continued...</i>		
TimeBase		3_TERM.TimeBase
XP		3_TERM.XP
TI		3_TERM.TI
TD		3_TERM.TD
Options	.InvPID	3_TERM.Options.InvPID
	.SL_Track	SETPOINT.Options.EnbTrack
	.IntBalSL	NOT(3_TERM.Options.DisSPbal)
	.IntBal	3_TERM.Options.SPbal
SelMode	.SelHold	MAN_STAT.Options.SelHold
	.SelTrack	MAN_STAT.Options.SelTrack
	.SelRem	MODE.SelMode.SelRem
	.EnaRem	MODE.SelMode. (EnaRem OR EnaCRem)
	.SelAuto	MODE.SelMode.SelAuto
	.SelMan	MODE.SelMode.SelMan
	.SelFMan	MODE.SelMode.SelFMan
ModeSel	.EnaRem	NOT(PID_CONN.ModeSel.HoldSel OR PID_LINK.ModeSel.TrackSel OR PID_LINK.ModeSel.ManSel OR (PID_LINK.FallBack = Manual))
	.HoldSel	PID_LINK.SelMode.SelHold
	.TrackSel	PID_LINK.SelMode.SelTrack
	.RemSel	MODE.ModeSel.(RemSel OR CRemSel)
	.AutoSel	MODE.ModeSel.AutoSel
	.ManSel	MODE.ModeSel.MorFMsel AND (MODE.Mode = Manual)
	.FAutoSel	MODE.ModeSel.FAutoSel
	.FManSel	MODE.ModeSel.MorFMsel AND (MODE.Mode = Forced Manual)
ModeAct	.NotRem	NOT(PID_LINK.Mode = REMOTE)
	.HoldAct	IF ModeSel.HoldSel = TRUE THEN ModeAct.HoldAct = TRUE, and rest of these seven 'Act' bits FALSE, ELSE IF ModeSel.TrackSel = TRUE THEN ModeAct.TrackAct = TRUE, and rest of these seven 'Act' bits FALSE, ELSE ModeAct.('Act' bits) = ModeSel.('Sel' bits)
	.TrackAct	
	.RemAct	
	.AutoAct	
	.ManAct	
	.FAutoAct	
	.FManAct	
FF_PID		3_TERM.FF_PID
FB_OP		3_TERM.FB_OP
SPblock		Tagname of linked SETPOINT block [Alphanumeric]
T3block		Tagname of linked 3_TERM block [Alphanumeric]
MSblock		Tagname of linked MAN_STAT block [Alphanumeric]
MDblock		Tagname of linked MODE block [Alphanumeric]

Table 58 PID_LINK block field sources

AN_DATA: ANALOGUE DATA BLOCK

Block function

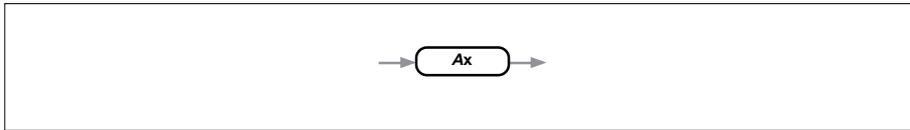


Figure 39 Block schematic

This block collects up to 100 analogue signals, derived from the CARB_DIFF block, defined in *CarbnBlk*. The collected values are the result of the Carbon Diffusion calculation generated via the CARB_DIFF block. Values from the CARB_DIFF block can be used to present a Carbon Diffusion curve on an Eycon Visual Supervisor screen.

NOTE This block functions using a number of pages. Fields can be located using <Page>.<Field>.<Subfield> convention.

Block parameters

Symbols used in *Table 59* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
A0 to A39 Page			
Next	Next Block containing collected values		
A0 to A19	20 collected values		↔
Alarms			☐ 📖 🔊
Software Combined	Block RAM data sumcheck error OR-ing of all Alarms bits	T/F T/F	
A20 to A39	20 collected values		↔
A40 to A79 Page			
Next	Next Block containing collected values		
A40 to A59	20 collected values		↔
Alarms			☐ 📖 🔊
See A0 to A39 Page			
A60 to A79	20 collected values		↔
A80 to A99 Page			
Next	Next Block containing collected values		
A80 to A99	20 collected values		↔
Alarms			☐ 📖 🔊
See A0 to A39 Page			

Table 59 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

A0 to A39, A40 to A79, A80 to A99 Page

These pages are used to show 100 values derived from the CARB_DIFF block.

Next. For future use only. Block name relating to the AN_DATA block showing the next configured 100 values.

A0 to A19, A20 to A39, A40 to A59, A60 to A79, A80 to A99. Analogue values from the CARB_DIFF block. These fields are subject to the configuration of *IP_Type*, and *Options.ImpDepth* parameters in the CARB_DIFF block.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.
- **Combined.** Asserted if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

LOOP_PID: LOOP PROPORTIONAL, INTEGRAL, DERIVATIVE BLOCK

Block function

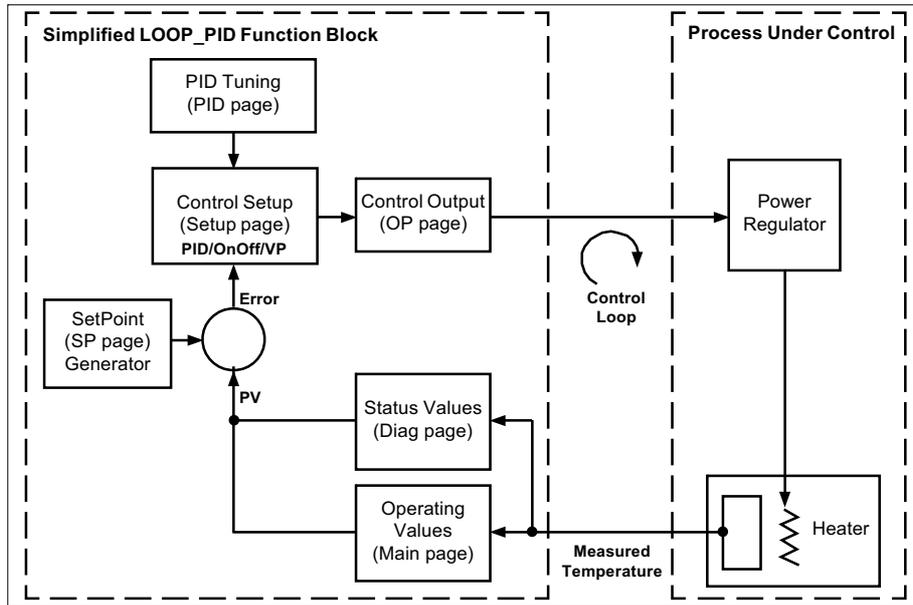


Figure 40 Block schematic

Please refer to the schematic in *Figure 40*. This block supports two outputs, Channel 1 and Channel 2 and has parameters used to configure the control, e.g. *Ch1Ctrl* and *Ch2Ctrl*, and tuning, e.g. *PB1*, *Ti1*, and *Td1*, automatically or manually, of a control loop. Each channel can be configured for PID, On/Off or Valve Position (bounded, VPB or unbounded, VPU) loop control. However, Channel 2 is generally used for the cooling process of the loop control. A single PID set of tuning parameters are included in the block, but up to seven additional PID sets of tuning parameters can be used by referencing individual TUNE_SET blocks, see *PID.Set2* to *PID.Set8*.

NOTE This block functions using a number of pages. For display purposes, fields can be located using *<Page>.<Field>.<Subfield>* convention.

The LOOP_PID block is updated each task execution and is used to automatically control a process to an operating point, Setpoint. The Setpoint of a control loop may be static or moving under the supervision of a program or remote system.

Block parameters

Symbols used in this table are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

NOTE The *Alarms* fields in this block appear on all pages.

Parameter	Function	Units	Status
Main Page			
Mode	Current operating mode	Enum	☐➡
AutoMan	Operating mode control	Enum	☐➡
PV	Process Variable (PV)	Eng	☐➡
PVStat	Condition of PV	Enum	☐➡
Inhibit	Control inhibit	Enum	☐➡
TargetSP	Target Setpoint value	Eng	☐➡
WSP	Working Setpoint value	Eng	☐➡📖
Alarms	These alarms appear on all pages in the block		☐➡📖🔊
Software	Block data sumcheck error/network failure	T/F	
HiHi	High high absolute alarm limit exceeded	T/F	
Hi	High absolute alarm limit exceeded	T/F	
Lo	Low absolute alarm limit exceeded	T/F	
LoLo	Low low absolute alarm limit exceeded	T/F	

Continued...

Parameter	Function	Units	Status
<i>Continued...</i>			
DevHi	High deviation alarm limit exceeded	T/F	
DevLo	Low deviation alarm limit exceeded	T/F	
LpBreak	Loop break detected	T/F	
SensorB	Sensor break detected	T/F	
Combined	OR-ing of all Alarms bits	T/F	
WrkOP	Combined Ch1 and Ch2 values	Integer	
IntHold	Stop Integral action	Enum	
SelMode	Select mode	(A)BCD hex	
		1 2 4 8	D
EnaRem	Enable Remote mode	T/F	
SelAuto	Automatic mode select	T/F	
SelMan	Manual mode select	T/F	
		1 2 4 8	C
ModeSel	Modes selected	(A)BCD hex	
		1 2 4 8	D
TrackSel	Track mode selected	T/F	
RemSel	Remote mode selected	T/F	
AutoSel	Automatic mode selected	T/F	
ManSel	Manual mode selected	T/F	
FManSel	Forced manual mode selected	T/F	
TuneSel	Tuning mode selected	T/F	
		1 2 4 8	C
PCalSel	Pot calibration mode selected	T/F	
InhibSel	Inhibit mode selected	T/F	
		1 2 4 8	B
ModeAct	Active mode	(A)BCD hex	
		1 2 4 8	D
NotRem	Remote mode not active	T/F	
TrackAct	Track mode active	T/F	
RemAct	Remote mode active	T/F	
AutoAct	Automatic mode active	T/F	
ManAct	Manual mode active	T/F	
FManAct	Forced manual mode active	T/F	
TuneAct	Forced automatic mode active	T/F	
		1 2 4 8	C
PCalAct	Pot calibration mode active	T/F	
InhibAct	Forced automatic mode active	T/F	
		1 2 4 8	B
Setup Page			
Ch1Ctr ^[1]	Channel 1 control type	Enum	
Ch2Ctr ^[1]	Channel 2 control type	Enum	
CtrlAct ^[1]	Control action	Enum	
PB_Units ^[1]	Proportional Band units	Enum	
DerivTyp ^[1]	Derivative type	Enum	
Alarms			
See Main Page			
Tune Page			
Enable	Auto Tune function control	Enum	
HiOutput	Maximum Auto Tune output power limit	Integer	
LoOutput	Minimum Auto Tune output power limit	Integer	
R2G	Determines if R2G is derived during auto-tuning	Enum	
Alarms			
See Main Page			
State	Auto Tune condition	Enum	
Stage	Current stage of Auto Tune function	Enum	
StageTim	Active Auto Tune stage timer	Integer	
PID Page			
ShedTyp	Method of PID set selection	Enum	
NumSets	Number of PID sets used in loop control	Integer	

Parameter	Function	Units	Status
<i>Continued...</i>			
RemInpUt	Remote value when SchedTyp = Remote	Integer	
ActivSet	Currently selected PID set		
IntBal	Integral balance control	Enum	
PB1	Proportional Band term of PID set 1	Eng/%	
Ti1	Integral Time term of PID set 1	Secs	
Td1	Derivative Time term of PID set 1	Secs	
R2G1	Relative Cool (Channel 2) Gain of PID set 1	Integer	
CBH1	Value preventing undershoot on cool down	Integer	
CBL1	Value preventing overshoot on heat up	Integer	
MR1	Fixed power level if Tin is set Off	%	
LBT1	Loop Break detection time period	Secs	
OPHi1	Gain scheduled maximum output limit	%	
OPLo1	Gain scheduled minimum output limit	%	
Alarms			
See Main Page			
Set2			
to	Block name of associated PID set (Tune_Set block)	Alpha-numeric	
Set8			
SP Page			
RangeHi	Maximum Setpoint limit within control loop	Eng	
RangeLo	Minimum Setpoint limit within control loop	Eng	
SPselect	Active Setpoint control	Enum	
SP1	Local primary Setpoint value	Eng	
SP2	Local secondary (standby) Setpoint value	Eng	
SPHiLim	Maximum local Setpoint limit	Eng	
SPLoLim	Minimum local Setpoint limit	Eng	
AltSPEn	Externally sourced Setpoint control	Enum	
AltSP	Value of externally sourced Setpoint	Integer	
RateSP	Maximum WSP change rate	Eng/Min	
RateDone	Setpoint is achieved	Enum	
SPRateDS	Disable SP rate limit	Enum	
ServToPV ^[1]	Drive WSP to PV on SP change	Enum	
Alarms			
See Main Page			
SPTrim	Setpoint offset value	Eng	
SPTrimHi	Maximum Setpoint offset value permitted	Eng	
SPTrimLo	Minimum Setpoint offset value permitted	Eng	
ManTrack ^[1]	Manual tracking control	Enum	
SPTrack ^[1]	Setpoint tracking control	Enum	
TrackPV	PV value tracked by program	Eng	
TkPVstat	Status of TrackPV	Enum	
TrackSP	SP value tracked when ManTrack is On	Eng	
SPIntBal ^[1]	Integral balance on SP change control	Enum	
OP Page			
OutputHi	Maximum output (OP) limit	%	
OutputLo	Minimum output (OP) limit	%	
Ch1Outpt	Channel 1 OP value	%	
Ch2Outpt	Channel 2 OP value (cooling)	%	
Ch2DeadB	Channel 2 Deadband value	%	
RateOP	OP rate limit value	%/Min	
RateDis	Disable OP rate limit	Enum	
C1OnOfHs	Channel 1 Hysteresis value	Eng	
C2OnOfHs	Channel 2 Hysteresis value	Eng	
Ch1TravT	Valve travel time from 0% -100% on Channel 1	Secs	
Ch2TravT	Valve travel time from 0% -100% on Channel 2	Secs	
PotCal	Runs PotCal process on selected channel	Enum	
NudgeUp	Channel 1 open On-Time movement control	Enum	
NudgeDn	Channel 1 close On-Time movement control	Enum	

Parameter	Function	Units	Status
			<i>Continued...</i>
<i>Continued...</i>			
C1PotPos	Channel 1 valve position input	%	
C1PotBrk	Channel 1 potentiometer break detected	Enum	
C2PotPos	Channel 2 valve position input	%	
C2Potbrk	Channel 2 potentiometer break detected	Enum	
PbrkMode	Potentiometer Break action control	Enum	
SbrkMode	Sensor Break action control	Enum	
Alarms			
See Main Page			
SbrkOP	Loop OP value if SbrkMode = SbrkOP	Enum	
SafeOP	Loop OP value if Main.InHibit = Yes	Enum	
ManMode	Mode of manual operation	Enum	
ManOP	Loop OP value during manual mode	%	
ForcedOP	Forced Loop OP value during manual mode in Step	%	
ManStart	Startup in manual control	Enum	
PwrffEnb ^[1]	For future use - Power FeedForward control	Enum	
PwrffIn	For future use - Measured voltage for Power FeedForward	Volts	
CoolType ^[1]	Cooling method	Enum	
FFType ^[1]	OP FeedForward control	Enum	
FFGain	OP FeedForward gain value	Integer	
FFOffset	OP FeedForward offset value	%	
FFTrimLm	Bias added to PID OP FeedForward	%	
FFRem	FeedForward value derived from Remote source	%	
FFOP	Calculated FeedForward value	%	
TrackOP	Loop OP value tracked when TrackEN = Yes	Integer	
TrackEn	Loop OP tracking control	Enum	
RemOPL	Loop OP low limit	%	
RemOPH	Loop OP high limit	%	
Diag Page			
Error	Calculated error (PV – SP)	Eng	
TargetOP	Target OP before rate limit	%	
WrkOPHi ^[1]	Working OP high power limit	%	
WrkOPLo ^[1]	Working OP low power limit	%	
LpBreak	Loop Break detected	Enum	
PropOP	Quantity of Control OP = Proportional term	%	
InOP	Quantity of Control OP = Integral term	%	
DerivOP	Quantity of Control OP = Derivative term	%	
SensorB	Sensor Break detected	Enum	
Alarms			
See Main Page			
SchedPB	Scheduled Proportional Band	Eng/%	
SchedTi	Scheduled Integral Time	Secs	
SchedTd	Scheduled Derviative Time	Secs	
SchedR2G	Scheduled Relative Cool Gain (Channel 2)	Integer	
SchedCBH	Scheduled CutBack high	Integer	
SchedCBL	Scheduled CutBack low	Integer	
SchedMR	Scheduled Manual Reset	%	
SchdLPBk	Scheduled Loop break time	Secs	
SchdOPHi	Scheduled OP high limit	%	
SchdOPLo	Scheduled OP low limit	%	
Alarms Page			
HiHi	High high absolute alarm level	Eng	
Hi	High absolute alarm limit	Eng	
Lo	Low absolute alarm limit	Eng	
LoLo	Low low absolute alarm level	Eng	
DevHi	High deviation alarm limit	Eng	
DevLo	Low deviation alarm limit	Eng	
Hyst	Hysteresis of alarm limit	Eng	

^[1]Read only at runtime. Other fields may be restricted according to configuration.

Table 60 Block parameters

Block specification menu

Dbase, Block, Type. See Appendix D page 544 for details of these ‘header’ fields.

NOTE These fields automatically appear on each of the pages in this block.

Main Page

This page is used to define the operational mode of the loop control.

Mode. (Track/Manual/Auto/Remote/F_Man/Tune/Pot_Cal/Inhibit). Current operating mode. Used to select manual, automatic and remote modes. Writing to this field during **Pot_Cal** or **Inhibit** operation automatically changes *Main.AutoMan* and *SP.AltSPEn*, effecting the resultant mode on exit, shown in priority order below.

Mode	OP. PotCal	Main. Inhibit	Main. AutoMan	Main. PVstat	OP. TrackEn	Tune. Enable	SP. AltSPEn	Output Action
Pot_Cal	Ch1/Ch2	Irrelevant	Irrelevant	Irrelevant	Irrelevant	Irrelevant	Irrelevant	Derived via Pot_Cal
Inhibit	Off	Yes	Irrelevant	Irrelevant	Irrelevant	Irrelevant	Irrelevant	OP=SafeOutVal
Manual	Off	No	Man	Irrelevant	Irrelevant	Irrelevant	Irrelevant	OP=ManOP*
F_Man	Off	No	Auto	Bad	Irrelevant	Irrelevant	Irrelevant	If SbrkMode=SbrkOP OP=SbrkOP or if SbrkMode=Hold OP=<unchanged>
Track	Off	No	Auto	Good	On	Irrelevant	Irrelevant	OP=TrackOP
Tune	Off	No	Auto	Good	Off	On	Irrelevant	Derived via Auto Tune
Auto	Off	No	Auto	Good	Off	Off	No	SP=Local
Remote	Off	No	Auto	Good	Off	Off	Yes	SP=Alternate

*If *OP.ManMode=Track*, *OP.ManOP* tracks *OP* when *Main.Mode=Auto*.
If *OP.ManMode=Step*, *OP.ForcedOP* is copied to *OP.ManOP* when *Main.Mode* changes to *Manual*.

Table 61 Mode Output Action

NOTE To provide a strategy that enables sensor break action (*OP.SbrkMode* only supported when *Mode* is *AUTO*) and the ability to write to the output (only supported when *Mode* is *MANUAL*) wire *ModeSel.FManSel* to *SelMode.SelMan*. If a sensor break occurs this will cause the instrument to operate in Forced Manual mode (*ModeSel.FManSel* is *TRUE* and *Mode* is *F_Man*) after the configured sensor break action has been applied. The required output can then be written to *OP.ManOP* while operating in *Mode* is *F_Man*.

AutoMan. (AUTO/MAN). Controls the operating mode of the loop, indicating how the output power is determined. In order to make the selection (Manual or Auto) persis-tent for a cold start, the *AutoMan* field should be set accordingly (only when it is not wired to).

- **Automatic Control.** During Automatic Control, *PV* is continuously monitored and compared to the Setpoint. The output power is calculated and used to minimise any difference.
- **Manual Control.** During Manual Control, the operator controls the output power. The power delivered to the process may be edited directly from the instrument or via the communications network. However, the loop continues to be monitored, allowing a smooth change when Automatic is selected.

PV. The *PV* (Process Variable) input value, typically wired from an analog input.

PVStat. (Good/Bad). The condition of the *PV* (Process Variable). Good indicates *Main.PV* is displaying a trusted value derived from the plant/system via an input block, typically an *AI_UIO* block. Bad indicates *Main.PV* is displaying a value that has been compromised, i.e. a hardware failure or the I/O module is removed.

Inhibit. (Yes/No). Controls the loop controlling. If *YES*, the loop will stop control and the output of the loop will be set to the safe output value, *OP.SafeOutVal*.

TargetSP. The Target Setpoint (*SP*) is the expected operating Setpoint value for the control loop, and is derived from *SP.SP1*, *SP.SP2* or *SP.AltSP*.

WSP. Shows the current value of the Setpoint being used by the control loop, and is derived from *SP.SP1*, *SP.SP2* or *SP.AltSP* and may have a rate limit applied.

Alarms. See Appendix D page 545 for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.

- **Hi, Lo.** Asserted, if the value shown in *PV* is greater than *Alarms.Hi*, High Absolute Alarm Limit, and less than *Alarms.Lo*, Low Absolute Alarm Limit. The alarms do not clear until *PV* has returned within these levels by more than the value specified in *Alarms.Hyst*.
- **HiHi, LoLo.** Asserted, if the value shown in *PV* is greater than *Alarms.HiHi*, High High Absolute Alarm Limit, and less than *Alarms.LoLo*, Low Low Absolute Alarm Limit. The alarms do not clear until *PV* has returned within these levels by more than the value specified in *Alarms.Hyst*.
- **DevHi, DevLo.** Asserted, if the error ($PV - SP$) is greater than *Alarms.DevHi*, High Deviation Alarm Limit, or the error ($SP - PV$) is less than *Alarms.DevLo*, Low Deviation Alarm Limit. The alarms do not clear until *Error* has returned within these levels by more than the value specified in *Alarms.Hyst*.
- **LpBreak.** Asserted, if *Diag.LpBreak* is set **Yes**.
- **SensorB.** Asserted, if *Diag.SensorB* is set to **Yes**.
- **Combined.** Asserted, if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

WrkOP. The total output of the loop prior to dividing Channel 1 and Channel 2 outputs.

IntHold. (Yes/No). If YES, the integral component of the PID calculation will be frozen. It will hold at its current value but will not integrate any disturbances in the plant. This is equivalent to switching to PD control using a preconfigured manual reset value.

SelMode. Bitfields used to select controller modes via digital inputs from the strategy.

- **EnaRem.** TRUE enables Remote mode operation. FALSE disables Remote mode operation. Control is transferred to Automatic mode when changing from TRUE to FALSE during Remote mode.
- **SelAuto, SelMan.** Set TRUE for Automatic operation, unless *SelMode.SelMan* shows TRUE. Manual operation is specified when *SelMode.SelMan* is TRUE.

ModeSel. Bitfields showing the modes that have been requested but not operational. The mode with the highest priority is determined as the *active* mode.

- **TrackSel.** Shows TRUE if *OP.TrackEn* is **On** indicating loop output tracking is requested.
- **RemSel.** Shows TRUE if *SP.AltSPEn* is **Yes** indicating the use of an Alternate (Remote) Setpoint is requested.
- **AutoSel.** Shows TRUE if *Main.AutoMan* is **Auto** indicating Automatic mode is requested.
- **ManSel.** Shows TRUE if *Main.AutoMan* is **Man** indicating Manual mode is requested.
- **FManSel.** Shows TRUE if *Main.PVstat* is **Bad** indicating Forced Manual mode is requested.

NOTE To enable the user to write the required value for *Main.WrkOP* via *OP.ManOP* when in Forced Manual mode (*ModeSel.FManSel* is TRUE) this bitfield must be connected to *SelMode.SelMan*. This will set the operating mode of the instrument to Manual (*SelMode.SelMan* shows TRUE) after the configured sensor break action (*OP.SbrkMode*) has been applied for a single database iteration only.

- **TuneSel.** Shows TRUE if *Tune.Enable* is **On**. This indicates automatic control loop tuning is requested.
- **PCalSel.** Shows TRUE if *OP.PotCal* is **On**. This indicates calibration of the potentiometer is requested.
- **InhibSel.** Shows TRUE if *Main.Inhibit* is **Yes**. This indicates a command to stop the control and output of the loop and set the safe output value (*OP.SafeOP*) is requested.

ModeAct. Bitfield showing the block's active mode, i.e. the value shown in *Main.Mode*. Only one *ModeAct* bit can be TRUE at a time.

- **NotRem.** Output connection from the slave controller into the master for cascade control applications. This connection is TRUE when the slave controller is not operating in Remote mode (*ModeSel.RemSel* is FALSE). This output should be connected into *SelMode.SelTrack* on the master controller.
- **TrackAct, RemAct, AutoAct, ManAct, FManAct, TuneAct, PCalAct, InhibAct.** Only the bitfield corresponding to the value shown in *Main.Mode* is set TRUE indicating the Mode currently active.

Setup Page

This page is used to configure the control loop operation.

Ch1Ctrl, Ch2Ctrl. (Off/On/PID/VPU/VPB). Selects an algorithm for channel 1 and channel 2. In temperature control applications, Channel 1 is usually the heating channel, Channel 2 is the cooling channel.

- **Off.** The channel Control Output is not configured.
- **On.** The channel Control Output is configured as On/Off. This is the simplest means of control but gives rise to fluctuations in the PV. A degree of hysteresis or deadband must be set in On/Off control if the operation of the switching devices is to be reduced and relay chatter to be avoided. This form of control leads to oscillation of the PV that can affect the quality of the final product and is undesirable. The alternative is to use Three (3-)Term Control, known as *PID* control.
- **PID.** The channel Control Output is configured as PID control. PID, or 3-Term Control, is an algorithm that controls the process to a Setpoint. PID provides more stable control by continuously adjusting the output to compensate for the *Diag.Error*, the rate of change or the error plus any continuous offsets.

PID control must be tuned to the process it is controlling,

- A Proportional Band (*PBI*) that is too small causes temperature cycling, or control loop instability. However, if set too large it causes sluggish control, or a weak change of power in response to deviations of temperature.
- A Integral time (*TiI*) that is too small will give an over eager response causing instability. However, if set too large it will slow down both the approach to Setpoint on a start up and the return to Setpoint after a disturbance.
- A Derivative time (*TdI*) that is set too large can causing an over eager power boost or throttle back.
- **VPU, VPB (Motorised Valve Control).** This mode can be configured as an alternative to the standard PID control algorithm. It is performed by delivering open or close pulses in response to the control demand signal. The motorised valve algorithm can operate in,
 - boundless mode (**VPU**), that does not require a position feedback potentiometer for control purposes. If a position feedback potentiometer is connected it can only be used to display the valve's position. It is implemented as a velocity mode algorithm.
 - bounded, (or position, (**VPB**)), control mode, that does requires a position feedback potentiometer. This is closed-loop control determined by the valve's position. It is implemented as a PID algorithm designed specifically for positioning motorised valves, and used to drive a position loop.

CtrlAct. (Reverse/Direct). Configures the direction of the control. i.e. reverse or direct acting. Reverse Acting is the normally used for heating control and is also referred to as Negative Feedback. This increases the output when PV is less than SP. Direct Acting is normally used for cooling control and is also referred to as Positive Feedback. This increases the output when PV is more than SP.

IMPORTANT This is the opposite to the control action in the PID block or 3-Term block.

PB_Units. (EngUnits/%). Configures the presentation style of the instruments Proportional Band. When set as %, it shows a percentage of the loop span, derived from the *SP.RangeHi* to *SP.RangeLo* values.

DerivTyp. (PV/Error). Defines when the derivative calculation occurs. When set as **PV**, the derivative only occurs when PV changes. When set as **Error**, the derivative occurs when either PV or Setpoint changes.

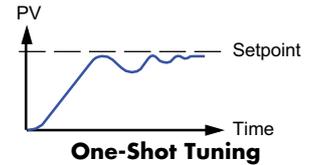
Alarms. See *Main.Alarms* and *Appendix D page 545* for a description of the Alarms field.

Tune Page

This page is used to automatically configure, Auto Tune, the control for optimum performance.

Enable. (Off/On). Start the Auto Tune process. When starting a one minute delay occurs, allowing any loop control to settle. During this time the loop Setpoint can be edited. It is best to start tuning with the process at ambient temperature. This allows the tuner to calculate the Cutback High, *PID.CBHI*, and Cutback Low, *PID.CBLI*, values more accurately. When tuning is complete, the control parameters are adjusted to best suit the characteristics of the load.

NOTE Care must be taken to ensure oscillations of the PV will not damage the process being tuned. It is recommended that the Setpoint for tuning is set below the normal running Setpoint value.



The 'one-shot' tuner works by switching the output on and off to induce an oscillation in the measured value. From the amplitude and period of the oscillation, it calculates the tuning parameter values. Some processes will not tolerate full heating or cooling being applied during tuning, so the level of heating or cooling can be restricted by setting the heating and cooling power limits. However, the measured value must oscillate to some degree for the tuner to be able to calculate values.

A one-shot tune can be performed at any time, but is normally performed only once during the initial commissioning of the process. However, if the process under control becomes unstable (because some characteristics have changed), it can be re-tuned.

HiOutput, LoOutput. Specifies the maximum (*Tune.HiOutput*) and minimum (*Tune.LoOutput*) percentage (%) output power level supplied during the automatic tuning process. The Auto Tune outputs will be clipped to the appropriate value.

R2G. Determines whether R2G derivation is derived during auto tuning. The default value is 'Yes', meaning R2G derivation is derived.

Alarms. See *Main.Alarms* and *Appendix D page 545* for a description of the Alarms field.

State. (Off/Running/Ready/Complete/Timeout/Ti_Limit/R2G_Limit). Reads and shows the condition of the Auto Tune function. **Off**, set when *Main.Inhibit* is set Yes, **Running**, **Ready**, and **Complete** indicate the various states of the Auto Tune function. **Timeout**, **Ti_Limit**, and **R2G_Limit** are error conditions. **Timeout**, occurs if any one stage is not completed within one hour, generally caused by an open loop, or if there is no response to the demand from the control. A very heavily lagged system can also produce a timeout if the cooling rate is very slow. **Ti_Limit**, occurs if Auto Tune calculates a value for the Integral time greater than the maximum allowable Integral time setting, i.e. 99999 seconds, indicating the loop is not responding or tuning is taking too long. **R2G_Limit**, occurs if the calculated R2G value is outside the range 0.1 and 10.0. It can be caused if the gain difference between heating and cooling is too large, or if during heat/cool control, when heating or cooling is turned off or not working correctly.

Stage. (Reset/None/Monitor/CurrentSP/NewSP/ToSP/Max/Min). Shows the current stage of the Auto Tune function. **Reset** indicates *Main.Inhibit* has been set to Yes, *Tune.State* shows Off, cancelling and resetting the tuning function. **ToSP** indicates the heat, or cool output is on. **Max** and **Min** indicate the power output is on or off, respectively.

StageTim. Shows the time taken since *Tune.Stage* changed.

PID Page

This page is used to configure the PID control for optimum performance.

- **Integral Action and Manual Reset.** In 3-term control (or PID control), the integral term 'TiI' (or *Ti* in an associated *Tune_Set* block) automatically removes steady state errors from the Setpoint. If control is operating in 2-term mode (PD mode), the Integral term is set OFF. Under these conditions the measured value may not settle precisely at Setpoint, causing the *manual reset* parameter to represent the value of the power output that will be delivered when the error is zero. This value must be set manually in order to remove the steady state error.
- **Loop Response.** The combination of correctly set P, I and D terms provide a stabilizing effect on a process, but depend totally on the nature of the process to be controlled. Balancing these terms is referred to as *tuning the loop*. Incorrect tuning causes poor loop response described as,
 - *Under Damped*, indicates that oscillation is prevented, but leads to an overshoot of PV followed by decaying oscillation to finally settle at the Setpoint. This type of response achieves Setpoint in the minimum time, but overshoot may cause problems and the loop can be sensitive to sudden changes in PV. This can result in further decaying oscillations before settling once again.
 - *Critically Damped*, indicates that overshoot does not occur and the process responds to changes in a controlled, non oscillatory manner.
 - *Over Damped*, indicates that the loop responds in a controlled but sluggish manner that results in a loop performance that is not ideal and unnecessarily slow.

NOTE In plastics extrusion, a barrel zone has a different response to a casting roll, drive loop, thickness control loop or pressure loop. Best performance is achieved when all loop tuning parameters are set to the optimum values.

SchedTyp. (Off/Set/SP/PV/Error/OP/Rem). Specifies the method of transferring control between both upper and lower boundaries of PID sets. Gain Scheduling is provided to allow different tuned PID sets to function at different temperatures, used in some processes. Gain Scheduling uses a loop parameter, e.g. PV or Setpoint, to select the active PID set, but can also apply specific PID settings at different operating points of the process.

- **Off.** A PID set is not used.
- **Manual.** A PID set can be selected by the operator. The selection of the required set can be controlled if blocks are wired correctly.
- **SP.** The PID set selection is derived from the Working SetPoint value, *Main.WSP*, and includes an internally defined hysteresis of 0.1% of the loop span.
- **PV.** The PID set selection is derived for the measured PV, *Main.PV*, and includes an internally defined hysteresis of 0.1% of the loop span.
- **Error.** The PID set selection is derived from the difference between the PV and the SP, *Diag.Error*, and includes an internally defined hysteresis of 0.1% of the loop span.
- **OP.** The PID set selection is derived from the Working OutPut value, *Main.WrkOP*, and includes an internally defined hysteresis of 0.5% of the output span.
- **Remote.** The PID set selection is derived from the input value from a remote parameter, *SP.RemInput*, and includes an internally defined hysteresis of 0.1% of the loop span.

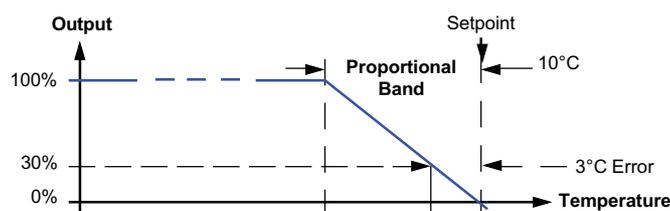
NumSets. Specifies the maximum number of permitted PID sets. Each additional configured PID set is configured in an individual Tune_Set block, and is associated using *PID.Set2* to *PID.Set8*.

RemInput. Specifies that value that causes the selection of a configured PID set, if *PID.SchedTyp* is set to REMOTE.

ActivSet. Shows the PID set currently in use.

IntBal. (On/Off). Controls the Integral Balance calculation. When set On the Integral Balance calculation is applied, preventing abrupt changes, bumps, and allows the output power to change gradually in accordance with the demand from the PID algorithm.

PB1. Specifies the Proportional Band value in display units or %, relating to PID set 1. This delivers an output that is proportional to the size of the error signal, over the range of PVs which linear gain action occurs before the output saturates at maximum or minimum. This value should be set as low as possible without causing oscillation.



Ti1. Specifies the Integral Time value relating to the PID set 1. It is used to remove steady state control offsets by ramping the output up or down in proportion to the amplitude and duration of the error signal. Off indicates the Integral Time is disabled.

NOTE During Proportional only control, an error between SP and PV must exist in order to deliver power. This can lead to the PV oscillating about the Setpoint or the PV settling at a point away from the Setpoint. By enabling Integral action, the control will monitor the error and add further power demand to remove the steady state errors.

Caution

The Integral Time should not be disabled (*PID.Ti1* set to OFF) when Channel 1 control type (*PID.Ch1Ctrl*) is set to Valve Positioner - Unbounded (VPU). This is because only through process error and integral action, can

the valve position be corrected, which could be unknown at power-up. Failure to adhere to this could result in *PID.Ch1Outpt* having the value '*', and the outputs toggling periodically.

Td1. Specifies the Derivative Time value relating to the PID set 1. It is used to achieve better stability, prevent Overshoot and Undershoot and restore the PV rapidly if a change in demand occurs. Off indicates that the Derivative Time is disabled.

A derivative action, *Setup.DerivTyp*, configured on PV allows changes in the Setpoint without causing the output to bump. The derivative monitors the loops rate of change, and reacts to steps in the PV by changing the output to remove the transient. Increasing the Derivative Time will reduce the settling time of the loop after a transient change.

NOTE If the loop control is unstable and causing excessive output changes because the derivative is amplifying noise from the PV, it is recommended the Derivative Time is disabled, *PID.Td1* set Off, and the loop is re-tuned.

R2G1. Shows the Relative Cooling (Channel 2) Gain control output relating to the PID set 1 and is only present if cooling has been configured. Sets the cooling Proportional Band, to a value that equals the heat Proportional Band value divided by the cool gain value. This value compensates for the different quantities of energy needed to heat, as opposed to that needed to cool a process.

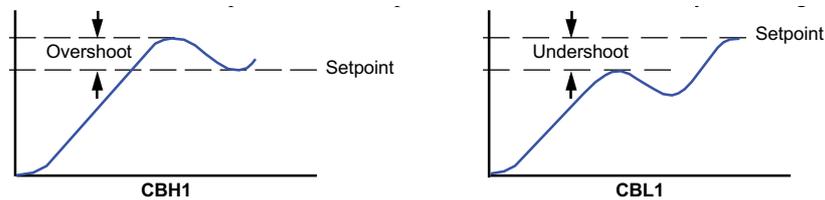


Figure 41 CutBack diagram

Example Water cooling applications may require a relative cool gain of 4. This means that cooling is 4 times faster than the heat process.

CBH1, CBL1. Specifies the number of display units, defined in *Setup.PBUnits*, above or below the Setpoint that will increase or decrease the output power. This is used to prevent undershoot on cool down and overshoot on heat up when large step changes in the process relating to the PID set 1 occur. **Auto** indicates a default value of 3Pb will be used.

MR1. Specifies the power required to eliminate the steady state error from Proportional control only. It is used to remove PV offsets from the Setpoint, and introduce a fixed power level to the output. This replaces the integral component when Integral Time (*Ti1*) is set to Off, but must be set to 0 (zero) when Integral Time is used.

LBT1. Specifies the maximum Loop Break Time, relating to the PID set 1. A Loop Break is detected if the PV does not respond to a change in the output before this time exceeded. If this value is exceeded, *Diag.LPBreak* shows YES, *Alarms.LpBreak* is set TRUE, and the output power will drive to high or low limit. Off indicates that the Loop Break Time is disabled.

During PID control, Loop Break detection sets *Diag.LPBreak* YES if the PV has not moved by $0.5 \times Pb$ in this time period. A typical value of $12 \times Td$, is set during Auto Tune.

During On/Off control, Loop Break detection is based on Loop Break Time as $0.1 \times SPAN$ where $SPAN = SP.RangeHi - SP.RangeLo$. Therefore, if the output is at the limit and *PV* has not moved by $0.1 \times SPAN$ in the Loop Break Time *Diag.LPBreak* is set YES.

NOTE Load and partial load failure are not the same as Loop Break detection.

OPHi1, OPLo1. Specifies the maximum and minimum Gain Scheduled Output limits relating to the PID set 1. These are ignored if the main output limit or the remote limit exceed these limits.

Alarms. See *Main.Alarms* and *Appendix D page 545* for a description of the Alarms field.

Set2 to Set8. Defines the block name of each additional configured PID set tuning parameters derived from individual Tune_Set blocks. Any used field is written to the associated Tune_Set block.

SP Page

This page is used to configure the Setpoint (SP) control. To provide adequate control a number of SP are provided, *SP1*, *SP2*, and *AltSP*, and appropriate tracking modes.

- Tracking.** Tracking modes can prevent step changes in the Setpoint when switching modes of the control, e.g. if *SP.ManTrack* is enabled (Yes), the process can be started in manual mode, controlling the power, the Setpoint will track the changes in PV. When the control is switched to automatic mode, the Setpoint will maintain the process at the current measurement.

RangeHi, RangeLo. Specifies the Range limits that provide a set of absolute maximums and minimums for Setpoints within the control loop. Any derived Setpoints are ultimately clipped to these Range limits. If *PB_Units* is set to % of loop span, the span is derived from these values.

SPSelect. Shows the currently selected Setpoint, *SP1* or *SP2* of the local loop.

SP1. Shows the SetPoint 1 value. This is the primary control Setpoint.

SP2. Shows the SetPoint 2 value. This is an alternative Setpoint, generally used as a standby value.

SPHiLim, SPLoLim. Specifies the maximum and minimum Setpoint limits respectively, and limits the Setpoint value.

AltSPEn. (No/Yes). Controls the use of an Alternate (Remote) Setpoint. When enabled (On), *Main.TargetSP* is derived from *SP.AltSP*. This cannot be enabled when *SelMode.EnaRem* is FALSE (see main page).

AltSP. Shows the value derived from an Alternate (Remote) Setpoint, sourced externally to the loop. It may be wired into the loop from an analogue input.

RateSP. Specifies the maximum Setpoint Rate Limit Value. This indicates the maximum rate that *Main.WSP* can change. It is used to protect the load from thermal shock that can be caused by large step changes in Setpoint.

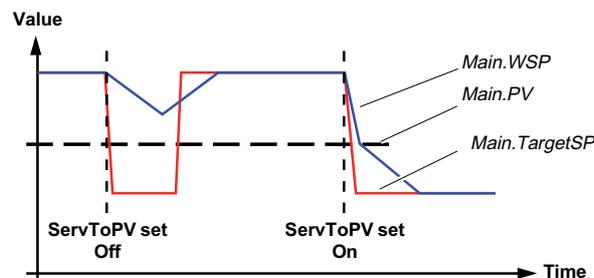


Figure 42 PV starting point

RateDone. (Yes/No). Shows the Setpoint Rate Limit, *SP.RateSP*, has achieved target, i.e. *Main.WSP* value has reached value defined in *Main.TargetSP*.

SPRateDs. (Yes/No). Controls the use of the Setpoint Rate Limit value. When disabled, *SP.SPRateDs* set Yes, the Working Setpoint (*Main.WSP*) can step change, when set No, it is prevented from changing too rapidly by the rate limiter, *SP.RateSP*.

ServToPV. (On/Off). Controls the starting point of *Main.WSP* after power cycling the instrument. When enabled (On), the the measured PV, *Main.PV*, is used as a start point for the *Main.WSP*. This decreases the time required for the *Main.WSP* to arrive at the *Main.TargetSP*. When disabled (Off), *Main.WSP* is reset and restarted.

Alarms. See *Main.Alarms* and *Appendix D page 545* for a description of the Alarms field.

SPTrim. Specifies the offset value applied to the Setpoint. The trim can be fed to the control via an analogue input, or from a wired function. Trims can be used in multi-zone applications to create a shaped profile along the machine.

SPTrimHi, SPTrimLo. Specifies the maximum and minimum amount of trim (*SP.SPTrim*) that can be added to the Setpoint, respectively.

ManTrack. (On/Off). Controls the use of the Manual Tracking. When enabled (On), any steps in Setpoint when switching between Manual and Automatic modes are removed. When the loop is switched from Manual to Auto, the Setpoint is set to the current PV. This is useful if the load is started in Manual Mode, then later switched to Auto to maintain the operating point.

SPTrack. (On/Off). Controls the use of the Setpoint Tracking. When enabled (On) the local Setpoint is selected, and copied to *SP.TrackSP*. Tracking now ensures that *SP.AltSP* tracks this value. When the *SP.AltSP* is selected it initially

takes on the tracked value ensuring a bumpless transfer. This Setpoint is then adopted gradually. A similar action takes place when returning to the local Setpoint.

TrackPV. Shows the tracked PV value when the program is tracking.

TkPVStat. Shows the condition of *TrackPV*, see *PVStat*.

TrackSP. Shows the tracked SP value when manual tracking, *SP.ManTrack*, is set On.

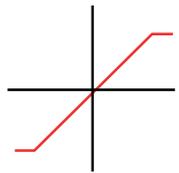
SPIntBal. (On/Off). Controls the Integral Balance calculation on change of *Main.TargetSP*. When set On, the Setpoint Integral Balance calculation is applied when *Main.TargetSP* changes, preventing bumps, and allowing the output power to change gradually in accordance with the demand from the PID algorithm. If set Off, the Setpoint Integral Balance calculation is not applied when *Main.TargetSP* changes.

OP Page

This page is used to configure the output (OP) control, and depends on the loop configuration.

- **On/Off Control.** Hysteresis, Deadband, Safe Output Values, Manual Output Value. The control can be configured as a dual channel control, e.g. each channel can be configured as a different control type, i.e. PID Heat, and On/Off Cool.
- **PID Control.** Power limits, Power Rate Limits, Safe Output Values, Manual Output Value, Power Feedforward settings.
- **Valve Position Control.** Travel Times, Potentiometer feedback values, Potentiometer Break Settings, Safe Output Values, Manual Drive control.

OutputHi, OutputLo. Shows the maximum output and minimum (maximum negative) output power value delivered by Channel 1 and Channel 2. By reducing the high power limit, *OutputHi*, it is possible to reduce the rate of change of the process. However, the ability to react to disturbance is also reduced when reducing the power limit. Typically, *OutputLo* will be set to 0% for Channel 1 (heat) only, and set to -100% for Channel 1/Channel 2 (heat/cool) algorithm.

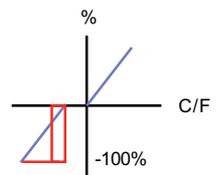


Ch1Output, Ch2Output. Shows the output value of Channel 1 and Channel 2 respectively. The Channel 1 output shows the positive power values (0 to *OP.OutputHi*) used by the heat output, and is typically wired to the control output (time proportioning or DC output). The Channel 2 output is the negative portion of the control output (0 - *OP.OutputLo*) for heat/cool applications, and is inverted to a positive number so that it can be wired into one of the outputs (time proportioning or DC outputs).

Ch2DeadB. Specifies the deadband percentage value. This is a difference between output 1 control going off and output 2 control coming on and/or vice versa.

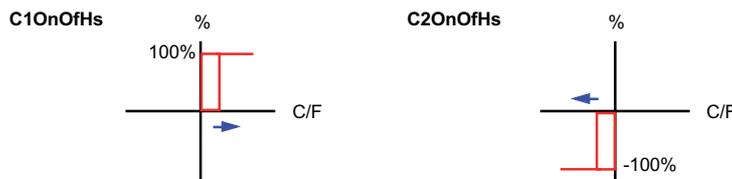
NOTE For on/off control this is taken as a percentage of the hysteresis.

RateOP. Specifies the maximum rate the output from the PID can change in % change per minute, if the Output Rate Limit Disable, *OP.RateDis*, is set Off. It prevents rapid changes in output from damaging the process or the heater elements.



RateDis. (Off/On). Controls the use of the Output Rate Limit. The Output Rate limit may be disabled by setting its value to 0.0, however, in some applications it is useful to wire to this field so that it can be switched on/off during stages of the process.

NOTE It can be used with programmer event outputs to control the output rate of change during a particular segment.

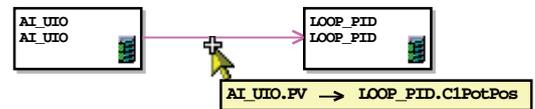


C1OnOfHs, C2OnOfHs. Specifies the Hysteresis value in the units configured for the PV of Channel 1 and Channel 2 respectively. In a Heat/Cool application it defines the point below Setpoint where the output will turn on. The output will turn off when the PV is at Setpoint. This is used to prevent the output from chattering at the control Setpoint. If the

hysteresis is set to 0 any change to the PV when at Setpoint will cause the output to switch. The hysteresis should be set to a value that provides an acceptable life for the output contacts, but does not cause unacceptable oscillations in the PV.

NOTE Use PID control if this performance is unacceptable.

C1TravT, C2TravT. Shows the valve travel time in seconds for the Channel 1 and Channel 2 outputs to travel from 0% (closed) to 100% (open). In a Valve Positioner application, Channel 1 is connected to both a Raise and a Lower output. In a Heat/Cool application Channel 1 is the heating valve and, Channel 2 is the cooling valve.



NOTE This value can be calculated using the automatic *OP.PotCal*.

PotCal. (Off/CalibrateCh1/CalibrateCh2). Controls the automatic calibration of the specified Channel position input, e.g. in a Heat/Cool application, the Channel 2 potentiometer must be calibrated, *PotCal* = CalibrateCh2, if a valve is used to control the cooling of a process. This will also calibrate the corresponding channel travel time, *OP.C1TravT*, or *OP.C2TravT*. The automatic calibration process will fail if a hardware fault occurs or online reconfiguration is attempted.

NOTE For automatic calibration the Potentiometer input modules must be fitted and wired directly to Channel 1 or Channel 2 potentiometer position parameters, *OP.C1PotPos* and *OP.C2PotPos*, in this block.

NudgeUp, NudgeDn. (Yes/No). Controls the valve movement. It is used to open, *OP.NudgeUp*, or close, *OP.NudgeDn*, the valve by a minimum On-Time, allowing digital communications to control the valve.

NOTE On-Time is the period of time power is applied to move the valve in a defined direction.

C1PotPos, C2PotPos. Shows the measured potentiometer position feedback indicating the position of Channel 1 and Channel 2 actuators respectively. This is used when bounded VP control, *Setup.Ch1Ctrl* or *Setup.Ch2Ctrl* is set VPB and allows the control algorithm to use the Potentiometer Position as the PV of the positional loop.

NOTE *PotCal* can be used to automatically calibrate the potentiometer feedback.

C1PotBrk, C2PotBrk. (Good/Bad). Shows Channel 1 and Channel 2 potentiometer position status respectively. **Bad** shows the potentiometer feedback signal for the channel, *OP.C1PotPos* or *OP.C2PotPos* is in a break condition. This parameter requires that the potentiometer position is wired from an input channel.

PbrkMode. (Raise/Lower/Rest/Continue). Defines the action to be taken during bounded VP control, *Setup.Ch1Ctrl* or *Setup.Ch2Ctrl* is set VPB, if the potentiometer feedback signal is bad, i.e. is in a break condition. If *C1PotBrk* or *C2PotBrk* set On, **Raise** indicates the output will raise the actuator, **Lower** indicates the output will lower the actuator. **Rest** indicates the output will hold the actuator at its current position, and **Continue** indicates the output will use a model to predict the actuator position.

SbrkMode. (SbrkOP/Hold). Defines the output action if the *PV* is bad, i.e. the sensor has failed. **SbrkOP** indicates the output is derived from *SbrkOP* if a sensor break occurs, and **Hold** indicates the output will remain at the last value calculated before the sensor break occurred.

Alarms. See *Main.Alarms* and *Appendix D page 545* for a general description of the Alarms field.

SbrkOP. Specifies the output power value, -100.0% to 100.0% if both output channels are configured, or 0.0% to 100.0%, to be used when the input, *PV*, is bad (*Main.PVstat* is Bad) and *SbrkMode* is set to **SbrkOP**. Sets *Main.Alarms.SensorB* TRUE.

SafeOP. Sets the output power when *Main.Inhibit* is set Yes.

ManMode. (Track/Step/LastMOP). Specifies the power applied in Manual Mode. **Track** indicates the output is derived from the last control output. **Step** indicates the output is derived from *OP.ForcedOP*. **LastMOP** indicates the output is derived from the value last configured by the operator, *OP.ManOP*.

ManOP. Specifies the manual mode output power. This is a user defined output value used when *Main.AutoMan* is set Man and *OP.ManMode* is set LastMOP or Track. If *OP.ManMode* is set to Track, *OP.ManOP* follows the control output when *Main.AutoMan* is set to Auto.

Warning In manual mode the control limits the maximum power to the power limits, however, it can be dangerous if the instrument is left unattended at a high power setting. It is important that over range alarms are configured to protect the process, and processes preferably fitted with an independent over range alarm unit.

ForcedOP. Specifies the forced manual mode output power. This is the defined output value used when *Main.AutoMan* is set **Man** and *OP.ManMode* is set **Step**.

ManStart. (Off/On). Controls the mode at startup. If *OP.ManStart* set **On**, *Main.AutoMan* is set to **Man** at startup. Otherwise, *Main.AutoMan* remain unchanged.

PwrffEnb. For future use - (Yes/No). Enables the output power to be corrected for fluctuations in the line voltage when using electrical heating (FeedForward). Yes, indicates the line voltage is monitored and will compensate for fluctuations before they affect the process temperature.

Example If the line voltage falls by 20% while a process is running at 25% power, and the temperature operating at the Setpoint, the heater power would drop by 36% because of the square law dependence of power on voltage. The temperature would eventually fall, causing the thermocouple and control to increase the On-Time in an attempt to bring the temperature back to Setpoint. Meanwhile the material would be running cooler than optimum which may cause some imperfection in the product.

NOTE This is only applicable to electric heaters and **MUST** be set to **No** if using any other heating medium such as gas, steam or heat transfer oil.

PwrffIn. For future use - Input to provide the measured mains voltage to the Power FeedForward calculation when *OP.PwrffEn* is set **Yes**.

CoolType. (Linear/Oil/Water/Fan). Specifies the type of cooling used. **Linear** indicates the cooling demand is derived directly from the PID demand, **Oil**, **Water** and **Fan** are derived non-linearly from the PID demand.

NOTE **Oil** is pulsed in a linear manner, but is a more direct cooling method and needs a lower cool gain than fan cooling. **Water** cooling does not operate well in areas running well above 100°C. When using water cooling this delivers shortened pulses of water for the first few percent of the cooling range, when the water is likely to be flashing off into steam. This compensates for the transition out of the initial strong evaporative cooling. **Fan** cooling is gentler than water cooling and not so immediate or decisive because of the long heat transfer path through the finned aluminum cooler and barrel. A cool gain setting of 3 upwards would be typical and delivery of pulses to the blower would be linear, i.e. the on time would increase proportionally with percentage cool demand determined by the control.

FFType. (None/Remote/SP/PV). Specifies the source of the *OP.FFOP* value, and is scaled and added to the control output. **None** indicates this function is disabled, **Remote** indicates the *OP.FFOP* value is derived from a remote source. **SP** indicates the *OP.FFOP* value is derived from *Main.WSP*, and scaled by *OP.FFGain* and *OP.FFOffset*, restricting the range of effect. **PV** indicates the *OP.FFOP* value is derived from *Main.PV*, and is scaled onto the output range by *OP.FFGain* and *OP.FFOffset*.

FFGain. Defines the value multiplied by *OP.FFOP*.

FFOffset. Defines the offset value applied to *OP.FFOP*.

FFTrimLm. Defines the symmetrical limit to *Main.WrkOP*, trimming *OP.FFOP* if the limit is exceeded.

FFRem. Defines an alternative value from the strategy to be used as the primary control variable in the FeedForward strategy. *OP.FFGain* and *OP.FFOffset* are not applied to this value.

FFOP. Shows the calculated FeedForward value.

TrackOP. Shows the value for the loop output to track when *OP.TrackEn* is enabled.

TrackEn. (Off/On). Controls the use of the loop output tracking. When enabled (On), the output of the loop will follow the track output value. The loop will bumplessly return to control when *OP.TrackEn* is set Off.

RemOPL, RemOPH. Specifies the minimum and maximum limit of the loop output from a remote source or calculation.

Diag Page

This page provides diagnostic parameters that can aid control loop commissioning.

Error. Shows the calculated error between *Main.WSP* and *Main.PV*.

TargetOP. Shows the requested control output. This is the target of the active output if an output rate limit, *OP.RateOP*, is configured.

WrkOPHi, WrkOPLo. Shows the resolved high and low output power limits. It is used to limit the output power of the loop and is derived from *Diag.SchdOPHi* and *Diag.SchdOPLo*, the *OP.RemOPH* and *OP.RemOPL* and the *OP.OutputHi* and *OP.OutputLo*.

LpBreak. (No/Yes). Shows a Loop Break has occurred. This is detected if the PV does not respond to a change in the output before the time specified in *PID.LBT1* or the currently operating PID set, *PID.Set2* to *PID.Set8*. This sets *Alarms.LpBreak* TRUE.

PropOP. Shows the contribution of the Proportional Band to the control output.

InOP. Shows the contribution of the Integral time term to the control output.

DerivOP. Shows the contribution of the Derivative time term to the control output.

SensorB. (No/Yes). Shows a Sensor Break has occurred. This is detected if the PV is bad, and triggers *OP.SbrkMode* and *OP.SbrkOP*, if applicable.

Alarms. See *Main.Alarms* and *Appendix D page 545* for a general description of the Alarms field.

SchedPB. Shows the Scheduled Proportional Band value as set by the active PID list and determined by Gain Scheduling, as shown in *PID.PB1* or currently active PID set.

SchedTi. (Off/On). Controls the use of the Integral time term value as set by the active PID list and determined by Gain Scheduling, as shown in *PID.Ti1* or currently active PID set.

SchedTd. (Off/On). Controls the use of the Derivative time term value as set by the active PID list and determined by Gain Scheduling, as shown in *PID.Td1* or currently active PID set.

SchedR2G. Shows the current Relative Cool Gain value as set by the active PID list and determined by Gain Scheduling, as shown in *PID.R2G1* or currently active PID set.

SchedCBH, SchedCBL. Shows the current CutBack High and CutBack Low value as set by the active PID list and determined by Gain Scheduling, as shown in *PID.CBH1* and *PID.CBL1* or currently active PID set. **Auto** indicates a default value of 3Pb will be used.

SchedMR. Shows the current Scheduled Manual Reset value as set by the active PID list and determined by Gain Scheduling, as shown in *PID.MR1* or currently active PID set.

SchdLPBk. Shows the current Scheduled Loop Break Time value as set by the active PID list and determined by Gain Scheduling, as shown in *PID.LBT1* or currently active PID set.

SchdOPHi, SchdOPLo. Shows the current Output High and Output Low values as set by the active PID list and determined by Gain Scheduling, as shown in *PID.OPHi1* and *PID.OPLo1* or currently active PID set.

Alarms Page

This page provides alarm parameters that can aid control loop commissioning.

HiHi, LoLo. Shows the High High and Low Low Absolute Alarm limit. *HiHi* and *LoLo* parameters define when the corresponding Alarms are set TRUE, e.g. *Main.Alarms.HiHi* shows TRUE, if PV exceeds an *Alarms.HiHi* set at 90. Both parameters operate with a user-configured hysteresis band (*Alarms.Hyst*).

Hi, Lo. Shows the High and Low Absolute Alarm limit. *Hi* and *Lo* parameters define when the corresponding Alarms are set TRUE, e.g. *Main.Alarms.Hi* shows TRUE, if PV exceeds an *Alarms.Hi* set at 90. Both parameters operate with a user-configured hysteresis band (*Alarms.Hyst*).

DevHi, DevLo. Shows the High and Low Deviation (*Error*) Alarm limits respectively. *DevHi*, and *DevLo* define when the corresponding alarms are set TRUE, if *PV* deviates from *SP* by more than the specified amount, and operates with a user-set hysteresis band (*Alarms.Hyst*) on each limit.

Hyst. Hysteresis bandwidth value, applied to all alarm levels in engineering units. It provides a clean transition in and out of Alarm conditions.

Alarms. See *Main.Alarms* and *Appendix D* page 545 for a description of the Alarms field.

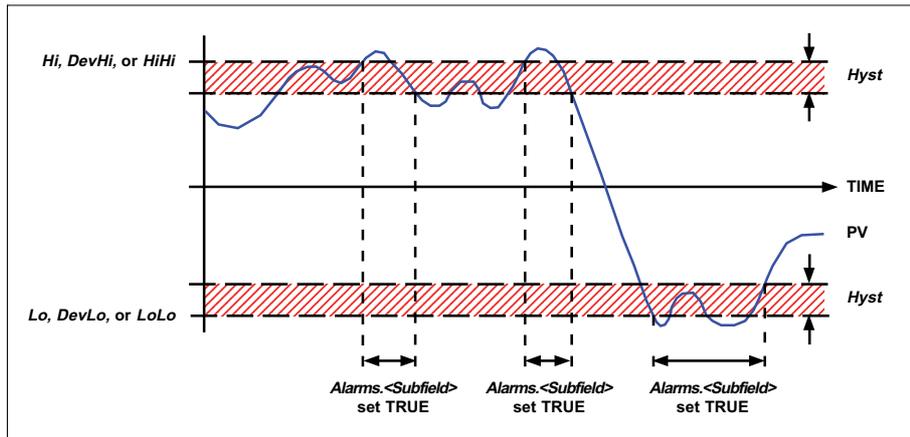


Figure 43 Hysteresis control

TUNE_SET: PID TUNING SET BLOCK

Block function

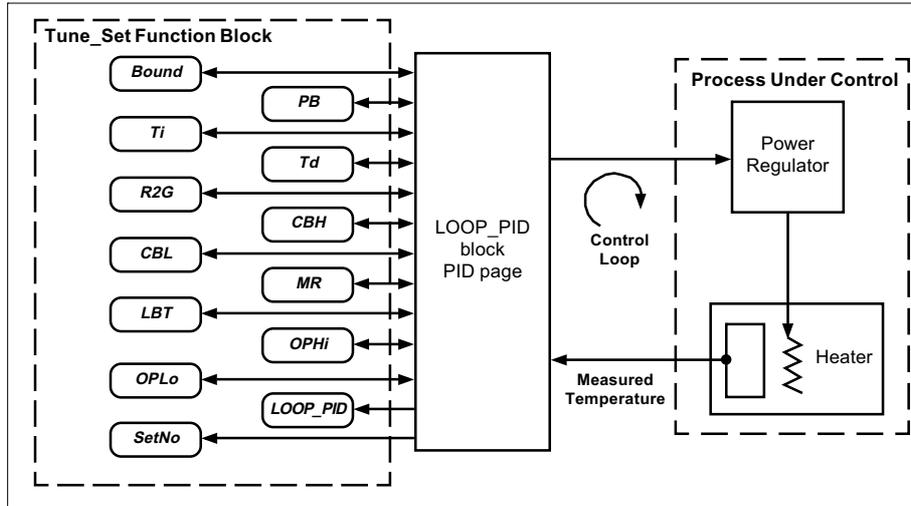


Figure 44 Block schematic

Please refer to the schematic in *Figure 44*. Each Tune_Set block provides another set of PID tuning parameters, that are referenced by the *PID.Set2* to *PID.Set8* fields in the LOOP_PID block.

NOTE An initial default set of tuning parameters is provided within the LOOP_PID block.

This block is used to obtain maximum performance from the PID control by matching, or tuning, the instrument strategy to the process under control. Maximum performance is derived from good control, that means,

- Stable ‘straight-line’ control of the process at Setpoint without fluctuation.
- Acceptable overshoot or undershoot of the process Setpoint.
- Quick response to deviations from the Setpoint caused by external disturbances, thereby restoring the process rapidly to the Setpoint value.

NOTE If using ‘Proportional only’, ‘PD’, or ‘PI’ control, the ‘TI’ or ‘TD’ parameters MUST be set OFF before commencing the tuning cycle. The tuning process ignores these parameters and will not calculate a value for them.

The standard method of tuning a PID loop is to use the advanced tuning algorithms inbuilt in the LOOP_PID block, to automatically test the loop and implement the optimum PID control parameters.

NOTE Care should be taken to ensure that the oscillations of the Process Value will not damage the process being tuned. For tuning purposes, it is recommended that the Setpoint is configured to a value below the normal running Setpoint value.

The ‘one-shot’ tuner works by switching the output on and off to induce an oscillation in the measured value. The optimum PID control tuning parameter values are calculated from the amplitude and period of the oscillation. It can be performed at any time, but normally it is performed only once during the initial commissioning of the process. However, if the process under control subsequently becomes unstable, because its characteristics have changed, it can be re-tuned for the change of characteristics.

Some processes will not tolerate full heating or cooling being applied during tuning, so the level of heating or cooling can be restricted by setting the heating and cooling power limits. However, the measured value must oscillate to some degree for the tuner to be able to calculate values.

NOTE It is best to start tuning with the process at ambient temperature. This allows the tuner to accurately calculate the CutBack Low, CBL, and CutBack High, CBH, values which restrict the amount of overshoot, or undershoot.

Block parameters

Symbols used in the following table are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Bound	PID tune set boundary	Integer	↔ □ □
PB	Proportional Band term of PID set	Eng/%	↔ □
Ti	Integral Time term of PID set	Secs	↔ □
Td	Derivative Time term of PID set	Secs	↔ □
R2G	Relative Cool Gain (Channel 2) of PID set	Integer	↔ □
CBH	Value preventing undershoot on cool down	Integer	↔ □
CBL	Value preventing overshoot on heat up	Integer	↔ □
MR	Fixed power level if Ti is set Off	%	↔ □
LBT	Loop Break detection time period	Secs	↔ □
OPHi	Gain scheduled maximum output limit	%	□
OPLo	Gain scheduled minimum output limit	%	□
Alarms			□ □ □
Software	Block data sumcheck error/network failure	T/F	
Combined	OR-ing of all Alarms bits	T/F	
LOOP_PID	Associated LOOP_PID block	Enum	□
SetNo	PID Tuning set number	Integer	□

Table 62 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

NOTE These fields automatically appear on each of the pages in this block.

Bound. Shows the PID Tune Set boundary. This value is compared against the monitored parameter, e.g. PV, to determine the active set and applies to each of the sets configured in the *PID.Setn* fields related to the LOOP_PID block.

Example If the monitored parameter is below the boundary Set 1 is active. If above the boundary, Set 2 is active.

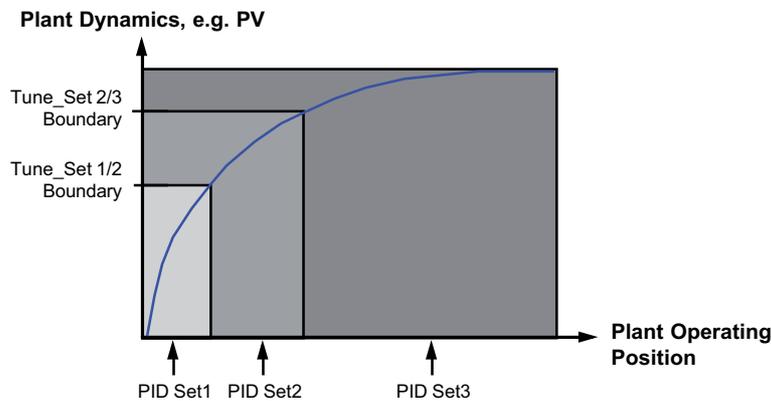


Figure Tune_Setb Tune_Set boundaries

Figure 45 Tune_Set boundaries

PB. Specifies the Proportional Band value relating to this PID Tuning set. This is the range of PVs over which linear gain action occurs before the output saturates at maximum or minimum, and should be set as low as possible without causing oscillation.

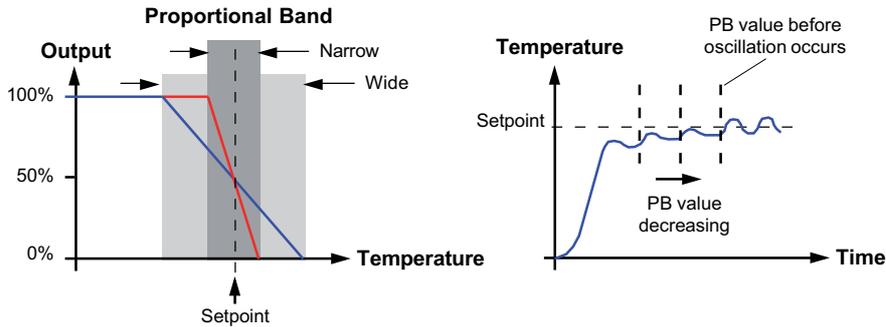
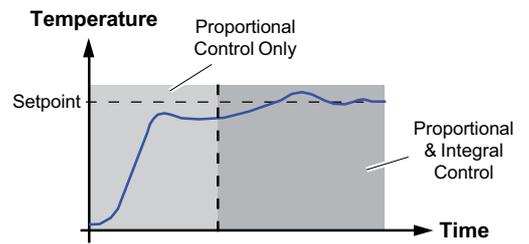


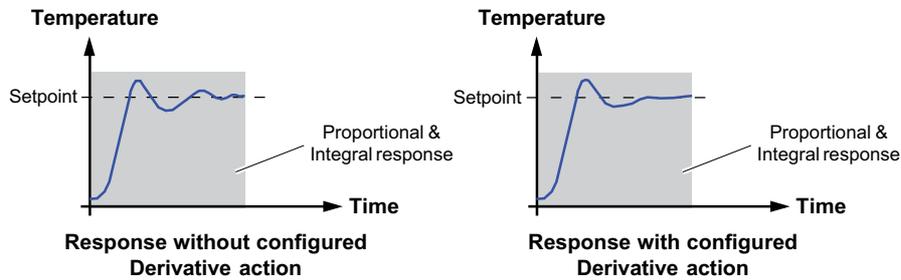
Figure 46 Proportional Band configuration

Ti. Specifies the Integral Time value relating to this PID Tuning set. It is used to remove steady state control offsets by ramping the output up or down in proportion to the amplitude and duration of the error signal. Off indicates that the Integral Time is disabled.

NOTE During Proportional only control, an error between Setpoint and PV must exist in order to deliver power. This can lead to the PV oscillating about the Setpoint or the PV settling at a point away from the Setpoint. By enabling Integral action, the control will monitor the error and add further power demand to remove the steady state errors.



Td. Specifies the Derivative Time value relating to this PID Tuning set. It is used to achieve better stability, prevent Overshoot and Undershoot and restore the PV rapidly if a change in demand occurs. Off indicates that the Derivative Time is disabled.



A derivative action, *Setup.DerivTyp*, configured on PV allows changes in the Setpoint without causing the output to bump. The derivative monitors the loops rate of change, and reacts to steps in the PV by changing the output to remove the transient. Increasing the Derivative Time will reduce the settling time of the loop after a transient change.

NOTE If the loop control is unstable and causing excessive output changes because the derivative is amplifying noise from the PV, it is recommended that the Derivative Time is disabled, *PID.Td1* set Off, and the loop is re-tuned.

R2G. Shows the Relative Channel 2 Gain control output relating to this PID Tuning set. This value is in relation to the channel 1 control output. It compensates for the different quantities of energy needed to heat, as opposed to that needed to cool, a process.

NOTE For example, water cooling applications may require a relative cool gain of 4. This means that cooling is 4 times faster than the heat process.

CBH, CBL. Specifies the number of display units above or below the Setpoint that will increase or decrease the output power. This is used to reduce overshoot and undershoot for large step changes in the process relating to this PID Tuning set. **Auto** indicates a default value of 3Pb will be used.

MR. Specifies the power required to eliminate the steady state error from proportional only control. It is used to remove PV offsets from the Setpoint, and introduces a fixed power level to the output. This replaces the integral component when Integral Time (*Ti*) is set to Off.

LBT. Specifies the maximum Loop Break Time, relating to this PID Tuning set. A Loop Break is detected if the PV does not respond to a change in the output before this time is exceeded. If this value is exceeded the Loop Break alarm, *Diag.LPBreak*, is set **Yes** and the output power will drive to high or low limit. Off indicates that the Loop Break Time is disabled.

The Loop Break detection in PID control, sets *Diag.LPBreak* **Yes** if the PV has not moved by $0.5 \times P_b$ in the Loop Break Time. The Loop Break Time is set by the Auto Tune, a typical value is $12 \times T_d$.

During On/Off control Loop Break is based on Loop Break Time as $0.1 \times \text{SPAN}$ where $\text{SPAN} = SP.\text{RangeHi} - SP.\text{RangeLo}$. Therefore, if the output is at limit and *PV* has not moved by $0.1 \times \text{SPAN}$ in the Loop Break Time a loop break is indicated.

NOTE This must not be confused with load failure and partial load failure.

OPHi, OPLo. Specifies the maximum and minimum Gain Scheduled Output limits relating to this PID Tuning set. These are ignored if the main output limit or the remote limit exceed these limits.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block's RAM data.
- **Combined.** Asserted, if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

LOOP_PID. Shows the block name of the LOOP_PID block related to this set of tuning parameters.

SetNo. Shows the Set number field, *Setn*, of the associated LOOP_PID block this Tune_Set block is configured to.

CHAPTER 7 CONVERT FUNCTION BLOCKS

This category of Function Block Templates provides the control strategy with functions for converting dissimilar database field types, particularly enumerated values.

This means that dissimilar field types can be converted via an appropriate CONVERT block.

ENUMENUM: ENUMERATED TO ENUMERATED CONVERTER BLOCK

Block function

The ENUMENUM block converts an enumerated value from one enumeration to another. The block requires both source and destination to be wired.

An enumerated field consists of a text string associated with a number. Enumerated fields in different LIN block types having the same text strings may well have different sets of associated numbers. The ENUMENUM block works by reading in the text string associated with the numerical value of the source block enumeration (connected to fields *SrcEnum1* to *8*), searching the text strings in the destination block enumeration (connected to fields *DstEnum1* to *8*) for an exact match, then outputting to the destination block the associated numerical value found in that block. Alarms (*Wire1* to *8*) are raised if a connection is un-wired or no match can be found.

NOTE. This block is not intended for fanning out to different enumerations and may not work in the way expected under such circumstances. Each wire-out should be fanned out only to instances of a single block type.

Block parameters

Symbols used in *Table 63* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
SrcEnum1 to 8	Source enumeration (values 0 - 20)	ENUM	
DstEnum1 to 8	Destination enumeration (values 0 - 20)	ENUM	
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
Wire1 to 8	TRUE if no text string match, or missing connection	T/F	
Combined	OR-ing of all Alarms bits	T/F	

Table 63 Block parameters

UINTENUM: INTEGER TO ENUMERATED CONVERTER BLOCK

Block function

The UINTENUM block converts an integer into an enumeration. The block requires the destination enumeration to be wired.

An enumerated field consists of a text string associated with a number. The UINTENUM block works by reading in the source integer (input via fields *SrcUint1* to *8*), and writing this number to the destination block enumeration (via output fields *DstEnum1* to *8*), thus selecting the associated text string (if any) in the destination block. Alarms (*Wire1* to *8*) are raised if a destination connection is not wired, or if the source integer does not match any of the permitted enumeration values in the destination block.

Block parameters

Symbols used in *Table 64* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
SrcUint1 to 8	Source integer (values 0 - 20)	UINT	
DstEnum1 to 8	Destination enumeration (values 0 - 20)	ENUM	
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
Wire1 to 8	TRUE if integer out of range, or un-wired destination	T/F	
Combined	OR-ing of all Alarms bits	T/F	

Table 64 Block parameters

ENUMUINT: ENUMERATED TO INTEGER CONVERTER BLOCK

Block function

The ENUMUINT block converts an enumeration into an integer. The block requires the source enumeration to be wired.

An enumerated field consists of a text string associated with a number. The ENUMUINT block works by reading in the numerical value of the source block enumeration (input via fields *SrcEnum1* to *8*), and writing this number to the destination integer (via output fields *DstUint1* to *8*).

Block parameters

Symbols used in *Table 65* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
SrcEnum1 to 8	Source enumeration (values 0 - 20)	ENUM	
DstUint1 to 8	Destination integer (values 0 - 20)	UINT	
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
Combined	OR-ing of all Alarms bits	T/F	

Table 65 Block parameters

REALTIME: REAL AND TIME CONVERTER BLOCK

Block function

This block has 8 sets of parameters, each representing a single time value. Each time value can be converted between a total number of defined units and a corresponding HH:MM:SS value. This is used to allow interconnection between fields that express time (durations) as a number, and fields that represent time (durations) in the HH:MM:SS form.

Block parameters

Symbols used in *Table 66* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Status	Conversion status	(AB)CD hex	
BadTime1 to BadTime8	Conversion status of parameter set	T/F	
Rtime1 to Rtime8	Time shown as a real number		
TimeBas1 to TimeBas8	Units defining Rtime value	Enum	
Direct1 to Direct8	Type of conversion	Enum	
Ttime1 to Ttime8	Time shown as HH:MM:SS	Time	
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
BadTime1 to BadTime8	Conversion failure of parameter set	T/F	
Combined	OR-ing of all Alarms bits	T/F	

Table 66 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

Status. Bitfield indicating general conversion error conditions. Please refer to *Table 66* for details.

■ **BadTime1 to BadTime8.** Set TRUE, if conversion fails, causing output field either *Rtime(n)* to remain unchanged and *Time(n)* to show *??:?:??*.

Rtime1 to Rtime8. Shows a Time expressed as a real number, in units defined by the associated *TimeBas(n)* field. If a conversion fails, the associated *Status.BadTime(n)* and *Alarms.BadTime(n)* are set TRUE, and this value remains unchanged.

NOTE Values that lead to a conversion above 37:56:16 (*Rtime(n)* maximum value of 134216 if using a “second” time base, or 2236 if using a “minute” time base) are not supported.

TimeBas1 to TimeBas8. (Secs/Mins/Hours). Defines the units used to express the value in associated *Rtime(n)* field.

Direct1 to Direct8. Defines the direction of the conversion. Select **RealtoTime** to convert to a value in *Rtime(n)* to the a value shown in *Time(n)*. Select **TimetoReal** to convert a value in *Time(n)* to the a value shown in *Rtime(n)*.

Ttime1 to Ttime8. Shows a Time expressed as HH:MM:SS. If a conversion fails, the associated *Status.BadTime(n)* and *Alarms.BadTime(n)* are set TRUE, and this value shows *??:?:??*.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

■ **Software.** Sumcheck error in block’s RAM data.

■ **BadTime1 to BadTime8.** Asserted, if corresponding *Status.BadTime(n)* is set TRUE, when a conversion fails.

- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

CHAPTER 8 DCM FUNCTION BLOCKS

These blocks are used only in the legacy instruments T2900/T800 and T940(X). Current instruments do not support these blocks running locally.

The Devolved Control Module (DCM) Function Block Templates provide the control strategy with function for collecting data from I/O Subsystems. DCM blocks running in the T2900/T800/T940(X) provide communications across a network with remote 2500 (Input/Output System), and other Series 2000 target instruments.

Each DCM block presents a view on particular data values in the target instrument, and also allows configuration of the communications parameters. The remote 2500/Series 2000 data appears in the local DCM block as input and/or output fields that can be 'wired' to the control strategy running in the T2900/T800/T940(X), and so interacted with.

DCM FUNCTION BLOCK SUB-CATEGORIES

The function blocks in the DCM category can be divided into eight sub-categories.

Refer to the LIN Block Reference Manual Issue HA082375U003 15 (Vintage) for details.

CHAPTER 9 DIAGNOSTIC FUNCTION BLOCKS

The DIAGNOSTIC category of Function Block Templates provide the control strategy with functions for assisting the diagnosis of potential or existing faults. The blocks are intended principally for use by the system designer; and therefore are subject to change without notice. If this happens, block sizes will be maintained to allow old databases to be loaded, but the meanings of some fields may alter. The DIAG blocks need not be present for correct LIN operation.

AGA8DIAG: AGA8 DIAGNOSTIC BLOCK

Block function

The AGA8DIAG block provides a set of diagnostics on the detailed workings of a named AGA8DATA (AGA8 calculation) block running in the control strategy. The fields in this block are detailed in the set of equations published in the American Gas Association Report #8.

Block parameters

Symbols used in this table are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Block specification menu

The following is given in addition to *Table 67*.

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

AGA8blk. Name of the AGA8DATA block to be diagnosed. Select this from the pulldown menu of block names.

AGA8name. Name of AGA8DATA block selected for diagnosing. This usually reads the same as the *AGA8blk* field, except when the AGA8DIAG block is cached.

Parameter	Function	Units	Status	
AGA8blk	Name of AGA8DATA block to be diagnosed	Menu		
AGA8name	String version of AGA8blk seen for cached blocks	Alphanumeric		
Search	TRUE searches for next AGA8DATA block	T/F		
Pwork	Pressure currently being worked with	Eng		
Twork	Temperature currently being worked with	Eng		
Pbase	Base pressure currently being worked with	Eng		
Tbase	Base temperature currently being worked with	Eng		
Dbase	Base density currently being worked with	Eng		
uu rk3p0 ww q2p0 hh bmix	Intermediate calculation values (see AGA8 Report)			
Alarms				
Software		Block RAM data sumcheck error / network failure	T/F	
AGA8Blck		Invalid AGA8blk field, so block not updating	T/F	
Combined		OR-ing of all Alarms bits	T/F	
n		Array index to be used for Bn (0 - 9)	Integer	
Bn	One entry of the B[n] array	Eng		
m	Array index to be used for Fm (0 - 9)	Integer		
Fm	One entry of the F[m] array	Eng		
State	Part of AGA8 calculation currently being performed	Menu		
Statel	Array index showing this calculation phase pass no.	Integer		
Statej	Array index showing this calculation phase pass no.	Integer		
ConcTime	Response time to a change in gas concentration	Secs		
TempTime	Response time to a change in gas temperature	Secs		
PresTime	Response time to a change in gas pressure	Secs		
ExecTime	Execution time of the block per database cycle	Msec		
MaxTime	Max. execution time of the block per database cycle	Msec		
MaxState	Calculation state at which MaxTime reached	Menu		

Table 67 Block parameters

State. (Init/Poll/Chardl/SiBase/BrktBase/DdtlBase/ZdtlBase/NewTemp/Temp/NewPres/Bracket/Ddetail/Zdetail). Part of AGA8 calculation currently being performed:

- **Init.** Block is initialising.
- **Poll.** Polling for input changes.

- **Chardl.** New composition calculation.
- **SiBase.** Convert Base reference to SI units.
- **BrktBase.** Bracket P for D calculation (Tbase and Pbase calculations).
- **DdtlBase.** Determining D for References.
- **ZdtlBase.** Determining Z for References.
- **NewTemp.** Collecting new temperature.
- **Temp.** Running temperature calculation.
- **NewPres.** Collecting new pressure.
- **Bracket.** Bracket P for D calculation (T and P calculations).
- **Ddetail.** Determining D for pressure.
- **Zdetail.** Determining Z for pressure.

MaxState. (Init/Poll/Chardl/SiBase/BrktBase/DdtlBase/ZdtlBase/NewTemp/Temp/NewPres/Bracket/Ddetail/Zdetail). Calculation state at the point when the execution time of the block reached its maximum value so far (held in *MaxTime*). Menu items have the same meanings as those of the *State* parameter, see *State*.

ALH_DIAG: ALARM HISTORY DIAGNOSTIC BLOCK

Block function

The ALH_DIAG block gives statistics on the alarm history facility of the T2550/Eycon™ 10/20 Visual Supervisor/T800/T2900/T940 instrument in which the block is running.

Block parameters

Symbols used in this table are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
SeqNo	Sequence number of the alarms history	Integer	
NumAct	Number of active alarms	Integer	
NumUnack	Number of unacknowledged alarms	Integer	
ACK_ALL	TRUE = acknowledge all alarms	T/F	
Alarms			  
Software	Block RAM data sumcheck error / network failure	T/F	
Combined	OR-ing of all Alarms bits	T/F	

Table 68 Block parameters

Block specification menu

The following is given in addition to *Table 68*.

SeqNo. The sequence number of the alarms history. Increments each time a change occurs.

ALINDIAG: ALIN MAC/LLC DIAGNOSTIC BLOCK

Block function

The ALINDIAG block gives information on the low-level ALIN Media Access Control (MAC) and Logical Link Control (LLC) performance.

Block parameters

Symbols used in this table are explained in *Table 1*.

Parameter	Function	Units	Status
MACstate	State of the Arcnet chip MAC		
ThisNode	This nodes ALIN MAC address		
NextNode	The MAC address of the next node (may not be known)		
PrevNode	The MAC address of the previous node (may not be known)		
DiagStat	Arcnet chip diagnostic status		
TENT_ID	Not valid during normal operation		
EXC_NAK	Set indicates excessive NAKs		
Token	Set indicates that a token has been received from another node		
RCV_ACT	Set indicates receiver activity		
DUP-ID	Not valid during normal operation		
MyRecon	Set indicates that this node caused last Arcnet reconfiguration		
StatusRg	Status register		
TA	TRUE = Transmitter Available for transmitting	T/F	
TMA	TRUE = Transmitter Message Acknowledged	T/F	
Recon	TRUE = Line Idle timeout (Reconfiguration)	T/F	
Test	Used for Test & diagnostics. (Normally FALSE)	T/F	
POR	TRUE = Power-On Reset carried out	T/F	
Spare1, Spare2		T/F	
RI	TRUE = Receiver Inhibited (not enabled)	T/F	
ClearCnt	TRUE resets the following 5 parameters to zero	T/F	
ChipRst	Quantity of software resets of ALIN chip	Integer	
TxAbsort	Quantity of transmit aborts	Integer	
TxFABort	Quantity of failed transmit aborts	Integer	
MyRecon	Quantity of ALIN reconfigurations caused by this node	Integer	
Recon	Quantity of ALIN reconfigurations	Integer	
Alarms			
Software	Block RAM data sumcheck error/network failure	T/F	
Combined	OR-ing of all Alarms bits	T/F	
LLCstate	Operating mode of LLC state machine	Menu	
SAPsfree	Quantity of spare service access points (SAPs)	Integer	
SAPsbusy	Quantity of SAPs in use	Integer	
Tx_free	Quantity of spare transmit buffers	Integer	
Tx_alloc	Quantity of transmit buffers in use	Integer	
Tx_busy	Quantity of transmit buffers busy	Integer	
Tx_ready	Quantity of transmit buffers waiting	Integer	
Rx_free	Quantity of spare receive buffers	Integer	
Rx_alloc	Quantity of receive buffers in use	Integer	
Rx_busy	Quantity of receive buffers busy	Integer	
Rx_bufav	Quantity of receive buffers available	Integer	
Rx_ready	Quantity of receive buffers ready	Integer	

Table 69 Block parameters

AMC_DIAG: APPLICATION MASTER COMMS DIAGNOSTIC BLOCK

Block function

The AMC_DIAG block gives communications statistics for a given application master comms line, and for a selected node on that line.

Block parameters

Symbols used in this table are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Port	Port number	Menu	
L_Update	Total cyclic updates	Integer	
L_MisUpd	Total cyclic updates missed	Integer	
L_Req	Total requests on line	Integer	
L_Resp	Total good responses	Integer	
L_Reject	Total requests rejected on line	Integer	
L_Lost	Total messages lost due to downtime on line	Integer	
L_LkTimO	Total link timeouts on line	Integer	
L_LkErrs	Total link errors on line	Integer	
L_ApNak	Total slave NAKs on line	Integer	
L_Stat	Status of line	(ABC)D Hex	
NodeSusp	Cyclic comms on one or more nodes suspended	T/F	D
LineSusp	Cyclic comms on all nodes on line suspended	T/F	
Spare		T/F	
Spare		T/F	
L_Reset	TRUE = Reset line statistics	T/F	
MemCfg	Percentage of configuration memory used	Eng	
MemTrs	Percentage of transient memory used	Eng	
MemTrsHI	Percentage of transient memory high water mark	Eng	
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
Combined	OR-ing of all Alarms bits	T/F	
Node	Node number (0 - 255, 0 = OFF)	Integer	
N_Req	Total requests on node	Integer	
N_Resp	Total good responses on node	Integer	
N_Reject	Total requests rejected on node	Integer	
N_Lost	Total messages lost due to downtime on node	Integer	
N_LkTimO	Total link timeouts on node	Integer	
N_LkErrs	Total link errors on node	Integer	
N_ApNak	Total slave NAKs on node	Integer	
N_Stat	Status of node	(ABC)D Hex	
NodeSusp	Cyclic comms on this node suspended	T/F	D
Spare		T/F	
Spare		T/F	
Spare		T/F	
N_Reset	TRUE = Reset node statistics	T/F	
N_Filter	Selection of read/write classes	(AB)CD Hex	
CyclicR	TRUE selects cyclic reads	T/F	D
CyclicW	TRUE selects cyclic writes	T/F	
AcyclicR	TRUE selects acyclic reads	T/F	
AcyclicW	TRUE selects acyclic writes	T/F	
AdHocR	TRUE selects ad hoc reads	T/F	C
AdHocW	TRUE selects ad hoc writes	T/F	
Spare		T/F	
Spare		T/F	

Table 70 Block parameters

Block specification menu

The following is given in addition to AMC_DIAG-1.

Port. (MODBUS_1/MODBUS_2/MODBUS_3/MODBUS_4/PROFIDP_1/PROFIDP_2). Definition of comms port and protocol number. This allows for multiple ports on the same protocol. Default is MODBUS_1.

BCS_DIAG: ROUTING BROADCAST DIAGNOSTIC BLOCK

Block function

The BCS_DIAG block displays statistics on the routing broadcast process within a bridge.

Block parameters

Symbols used in this table are explained in *Table 1*.

Parameter	Function	Units	Status
Alarms			  
Software	Block RAM data sumcheck error / network failure		
Combined	OR-ing of all Alarms bits		
<i>For the LIN port:</i>			
LSMTime	Timebase to measure sister bridge count		
LTimeDue	Time due for next broadcast		
LTime	Timestamp for tx broadcast		
LTimeSlip	Slip on TX broadcast		
LTimePrd	Period for TX broadcast		
LPhTime	Time between my broadcast and next sister broadcast		
LPhErr	The error on the last measured phase		
LBcstCnt	Routing broadcast count		
LSRxCnt	Number of sister routing broadcasts received		
LRxCnt	Total number of routing broadcasts received		
LSMCnt	Number of measurement periods		
LSCnt	Number of sister bridges		
LSRxFnd	Number of sisters found so far		
LSMode	In sister mode (boolean)		
LMyPhase	I was the last sister node to transmit (boolean)		
LSpltSeg	Split segment detected		
<i>For the ALIN port:</i>			
ASMTime	Timebase to measure sister bridge count		
ATimeDue	Time due for next broadcast		
ATime	Timestamp for tx broadcast		
ATimeSlip	Slip on TX broadcast		
ATimePrd	Period for TX broadcast		
APhTime	Time between my broadcast and next sister broadcast		
APhErr	The error on the last measured phase		
ABcstCnt	Routing broadcast count		
ASRxCnt	Number of sister routing broadcasts received		
ARxCnt	Total number of routing broadcasts received		
ASMCnt	Number of measurement periods		
ASCnt	Number of sister bridges		
ASRxFnd	Number of sisters found so far		
ASMode	In sister mode (boolean)		
AMyPhase	I was the last sister node to transmit (boolean)		
ASpltSeg	Split segment detected		

Table 71 Block parameters

CON_DIAG: CONNECTION DIAGNOSTIC BLOCK

Block function

The CON_DIAG block displays statistics on connections through a bridge.

Block parameters

Symbols used in this table are explained in *Table 1*. Parameter information is given in *Table 72* following.

Parameter	Function	Units	Status
Entries	Number of connection table entries in use		
Search	Average search length for lookups		
SearchF	Number of comparisons on failed search		
Lookup	Number of lookups		
LookupF	Number of failed lookups		
Delete	Number of deletions		
DeleteF	Number of failed deletions		
DeleteB	Number of broken deletions		
Add	Number of additions		
AddF	Number of failed additions		
Alarms			  
Software	Block RAM data sumcheck error / network failure		
Combined	OR-ing of all Alarms bits		
Searchx	Index of searches by search length		
SearchFx	Index of failed searches by search length		

Table 72 Block parameters

CON_ENT: CONNECTION ENTRY BLOCK

Block function

The CON_ENT block displays information on a particular connection through a bridge.

Block parameters

Symbols used in this table are explained in *Table 1*. Parameter information is given in *Table 73* following.

Parameter	Function	Units	Status
Index	Selects a particular connection table entry		
Alarms			  
Software	Block RAM data sumcheck error / network failure		
Combined	OR-ing of all Alarms bits		
<i>For the originator of the connection:</i>			
ONode	Address of originating node		
OName	Database name or "***File**" of originating node		
OPort	Port of originating node		
OStatus	Connection status		
SevError	Set on severe error		
NoHops	Hop count has been exhausted		
LostMsg	Set on lost message		
ChngHops	Set if hop count has changed		
UsedSAP	Set if a particular SAP was banned on repeat connection		
HasBuff	This connection has a valid buffer		
HasHand	This connection has a valid LLC handle		
OHops	Number of hops from this bridge to the originating node		
OLMAC	Local MAC address on originating port		
OLSAP	Local SAP on originating port		
ORMAC	Remote MAC address on originating port		
ORSAP	Remote SAP on originating port		
OFwdCnt	Count of messages forwarded towards originating node		
ORtryCnt	Count of retried message transmissions towards originating node		
OFailCnt	Count of failed message transmissions towards originating node		
OTstamp	Timestamp of last transmission towards originating node		
<i>For the target of the connection:</i>			
TNode	Address of originating node		
TName	Database name or "***File**" of originating node		
TPort	Port of originating node		
TStatus	Connection status		
SevError	Set on severe error		
NoHops	Hop count has been exhausted		
LostMsg	Set on lost message		
ChngHops	Set if hop count has changed		
UsedSAP	Set if a particular SAP was banned on a repeat connection		
HasBuff	This connection has a valid buffer		
HasHand	This connection has a valid LLC handle		
THops	Number of hops from this bridge to the originating node		
TLMAC	Local MAC address on originating port		
TLSAP	Local SAP on originating port		
TRMAC	Remote MAC address on originating port		
TRSAP	Remote SAP on originating port		
TFwdCnt	Count of messages forwarded towards originating node		
TRtryCnt	Count of retried message transmissions towards originating node		
TFailCnt	Count of failed message transmissions towards originating node		
TTstamp	Timestamp of last transmission towards originating node		

Table 73 Block parameters

CON_TBL: CONNECTION TABLE BLOCK

Block function

The CON_TBL block displays a summary of the file or database connections operating through a bridge.

Block parameters

Symbols used in this table are explained in *Table 1*. Parameter information is given in *Table 74* following.

Parameter	Function	Units	Status
Page	Selects a group of 16 routing table entries		
Alarms			  
Software	Block RAM data sumcheck error / network failure		
Combined	OR-ing of all Alarms bits		
SNodex	Address of the node originating the connection (Source)		
DNodex	Address of the node destined to receive the connection (Dest)		

Table 74 Block parameters

DB_DIAG: DATABASE DIAGNOSTIC BLOCK

Block function

The DB_DIAG block indicates the actual resource levels *used* by the current database (in the left half of the specification menu), and also the maximum resource levels allowed by the current software version (in the right half).

NOTE The displayed parameter values are valid only at runtime in the target instrument.

Block parameters

Symbols used in this table are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units*	Status*
Block	Database blocks in use		
Tmpl	Database templates in use		
Libs	Template libraries in use		
Edb	External databases in use (i.e. cached in remote databases)		
Featt	Qty of blocks in local database cached elsewhere		
Teatt	Qty of blocks in remote databases cached locally		
Sert	Server tasks in use		
Connect	Qty of field-to-field connections		
Alarms			  
Software	Block RAM data sumcheck error / network failure	T/F	
Combined	OR-ing of all Alarms bits	T/F	
MaxBlock	Maximum qty of database blocks		
MaxTmpl	Maximum qty of database templates		
MaxLibs	Maximum qty of template libraries		
MaxEdb	Max. qty of external databases		
MaxFeatt	Max. qty blocks in local Dbase cached elsewhere		
MaxTeatt	Max. qty blocks in remote Dbases cached locally		
MaxSert	Maximum qty of server tasks		
MaxConn	Maximum qty of field-to-field connections		
MaxDBSiz	Maximum database size in bytes/kbytes[1]		
Memory Page	Memory Diagnostics		
DBFree	Useable free memory space (ignoring small fragments)		
BigFrag	Largest free memory fragment		
DBSize	Current database size in bytes		
NumFrag	Number of useable memory fragments (ignoring small fragments)		
spare	Not used		
MaxDBSiz	Maximum database size in bytes/kbytes[1]		

*All parameters have integer format, and read-only status at runtime
[1] Shows either bytes or kbytes depending on instrument type

Table 75 Block parameters

Block specification menu

The following is given in addition to *Table 75*.

Dbase, Block, Type. See *Appendix D page 544* for details of these 'header' fields.

Featt. From external attachments. Quantity of blocks in the local database that are cached elsewhere - counted as one for each copy of the local block.

Teatt. To external attachments. Quantity of blocks in remote databases that are cached in the local database.

DDR_DIAG: DATA RECORDING FACILITY DIAGNOSTIC BLOCK

Block function

The DDR_DIAG block gives statistics on the data recording facility of the Eycon™ 10/20 Visual Supervisor and T2900 (Industrial Strategy Engine) in which the block is running.

Much of the DDR system is owned or controlled by the Group blocks in the ORGANISE template palette. Therefore there are two levels of DDR information - system-wide, and group-specific. To handle an arbitrary number of groups without becoming unwieldy, the DDR_DIAG block displays data for a single group, selected using the input fields *ShowArea* and *ShowGrp*.

Block parameters

Symbols used in this table are explained in *Table 1*. Parameter information is given in *Table 76* following.

Parameter	Function	Units	Status
Areas	Number of areas detected in LIN database	Integer	
Groups	Number of groups detected in LIN database	Integer	
Points	Number of recording points in LIN database	Integer	
ShowArea	The area of the group to display	Integer	
ShowGrp	The group to display	Integer	
SampRate	Sample interval for current group (seconds)	Eng	
Samples	No. of samples taken so far, for current group	Integer	
Mem_Used	Highest allocation from T2900 memory (bytes)	Integer	
Mem_Aloc	Number of blocks allocated at maximum use	Integer	
FlashReq	*No. of flash blocks allocated to current group	Integer	
FlashRcv	*No. of flash blocks recovered on hot startup for current group	Integer	
FlashRsz	*No. of flash blocks resized on hot startup for current group	Integer	
DataWdth	Size of each sample for current group (bytes)	Integer	
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
Combined	OR-ing of all Alarms bits	T/F	

*Units of 64Kb

Table 76 Block parameters

EDB_DIAG: EXTERNAL DATABASE DIAGNOSTIC BLOCK

Block function

The EDB_DIAG block displays information on connections to external databases running in remote instruments. One remote database is monitored at a time, selected via the *EDBIndex* field. Remote database entries can be examined to check that the name and address are as expected and that communication is occurring. The block also monitors the cached block update rate tuning algorithm.

Block parameters

Symbols used in this table are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
EDBIndex	Entry in EDB table; 1-MaxEdb (DB_DIAG block)	Hex	
NodeNo	Remote instrument node address	Hex	
NodeName	Remote instrument database name	Alphanumeric	
RetrySta	EDB attachment retry status number	Integer	
RetryTim	EDB attachment retry timeout (ticks)	Integer	
DSap	Remote node service access point	Hex	
LSap	Local node service access point	Hex	
TimeStmp	Time last message was sent	Integer	
Status	EDB status flags	ABCD Hex	
Status0	Current EDB status	T/F	D
Status1	Current EDB status	T/F	
Status2	Current EDB status	T/F	
Status3	Current EDB status	T/F	
Transprt	Comms transport type in use (0=old, 1=new)	T/F	C
Spare		T/F	
Spare		T/F	
External	Verify requested by other than network task	T/F	
1st_time	EDB connected at first attempt	T/F	B
Remote	EDB entry created by a remote node	T/F	
T_alarm	Teatt alarm (Teatts with no corresp remote Featts)	T/F	
F_alarm	Featts in alarm	T/F	
Verify	Network server is verifying featts	T/F	A
Sent_OK	Last change was sent successfully	T/F	
Discnect	Local disconnect prevents remote connect	T/F	
ReVerify	Reverify request after processor swap on duplex CPUs	T/F	
My ID	Local message identifier	Hex	
His ID	Remote message identifier	Hex	
ErrRate	Transmission error count to this EDB	Integer	
TxCount	Qty. of transmissions since TxBias was adjusted	Integer	
TxBias	Transmission timeout bias	Integer	
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
Combined	OR-ing of all Alarms bits	T/F	
Protocol	LIN database protocol version ('1' to '7')	Integer	

Table 77 Block parameters

Block specification menu

The following is given in addition to *Table 77*.

NodeName. Database name of the remote node. It is not addressable via LINOPC.

DSap. Number of the remote node's Service Access Point (SAP) in use.

LSap. Number of the local node's Service Access Point (SAP) in use.

My ID, His ID. Identifiers used by a station to keep track of the order of messages and replies to confirmed requests. These values should increment once for each message sent or received.

ErrRate. Provides information on the state of the block update rate tuning algorithm. *ErrRate* provides a measure of the transmission error rate to the remote node, i.e. the number of errors that occurred in the last *TxCount* transmission.

TxCount. Provides information on the state of the block update rate tuning algorithm. This shows a count of transmitted messages that have occurred after *TxBias* was edited.

TxBias. Provides information on the state of the block update rate tuning algorithm. Defines the maximum time permitted before an error is counted, increasing *ErrRate*. This value is derived from *ErrRate*, *TxCount*; tuned to give 0.5% - 1.0% target error rate.

EDB_TBL: EXTERNAL DATABASE TABLE BLOCK

Block function

The EDB_TBL block summarises all external databases currently in use.

Block parameters

Symbols used in this table are explained in *Table 1*. Parameter information is given in *Table 78* following.

Parameter	Function	Units	Status
Page	Selects a group of 10 external databases		
Noden ($n=1-10$)	External database node number - segment,MAC		
Alarms			   
Software	Block RAM data sumcheck error / network failure		
Combined	OR-ing of all error flags		
ENamen ($n=1-10$)	Corresponding external database name		

Table 78 Block parameters

EIO_DIAG: EURO THERM I/O DIAGNOSTIC BLOCK

Block function

The EIO_DIAG block provides a display for both the *expected* and *actual* I/O modules at each site, and the health of each of the sites. It can display a maximum of 16 I/O sites on one screen.

When operating on a coupled duplex system, the display of actually fitted I/O modules is derived by either the primary or secondary processor, as selected via the *AcSource* parameter, when operating on a simplex system, or a non-coupled duplex system, the display of actually fitted I/O modules is derived by the primary processor.

NOTE 'Non-coupled' is not the same as 'not sync'd', if the Duplex LED is steady, the processors are coupled, but may or may not be sync'd. If the Duplex LED is flashing or off, the processors are not coupled.

Block parameters

Symbols used in this table are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Page	Selects a group of 16 I/O sites		
MaxSites	Size of I/O Base Unit (slots)	Integer	
Expect_1 to Expect16	Expected I/O module type at this site (slot)	Alphanumeric	
PrFault			
Site_1 to Site16	Primary Processor fault detected		
	Primary can/cannot see I/O module	F	
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
PrMajFlt	Major Primary processor fault	T/F	
SeMajFlt	Major Secondary processor fault	T/F	
PrMinFlt	Minor Primary processor fault	T/F	
SeMinFlt	Minor Secondary processor fault	T/F	
Combined	OR-ing of all Alarms bits	T/F	
AcSource	'Actual' parameters source (Simplex/Primary/Secondary)	Menu	
Actual_1 to Actual16	Actual I/O module type at this site (slot)	Alphanumeric	
SeFault			
Site_1 to Site16	Secondary can/cannot see I/O module	F	
Options			
FaultsAs	Determines the method to derive health status	Menu	

Table 79 Block parameters

Block specification menu

The following is given in addition to *Table 79*.

Dbase, Block, Type. See *Appendix D page 544* for details of these 'header' parameters.

Page. This parameter provides switching between pages if an instrument supporting more than 16 sites. The number of pages are displayed in the units parameter to indicate exactly which I/O subsystem is referred to. Each page represents up to 16 I/O sites.

MaxSites. This parameter indicates the number of sites (slots) in the instrument.

Expect_1 to Expect16. None/AI2/AI3/AI4/ZI/AO2/DI16/DI4/DI8_LG/DI8_LGv2/DI8_CO/DI8_COv2/DI6_MV/DI6_HV/DO4_LG/DO4_LGv2/DO4_24/DO4_24v2/DO8/DO16/RLY4/RLY4v2/RLY8/FI2/AI8_TC/AI8_MA/AI8_RT/AI8_FMA. This is the module type which is expected to be fitted at a specific site number. For example, *Expect_1* is the module type expected to be physically fitted at site 1.

PrFault. Bit-fields indicating the status of the communications between the primary processor and the selected I/O site, as indicated by the *Page* parameter.

- **Site_1 to Site16.** Health status shown in *PrFault* of each sites is derived as per the table below.

Expected I/O Module	Actual I/O Module	Options.FaultAs set to 'Mismatch'	Options.FaultAs set to 'Redundcy'
None	None	HEALTHY	UNHLTHY
None	<any>	UNHLTHY	UNHLTHY
<specific>	<correct>	HEALTHY	HEALTHY
<specific>	<wrong>	UNHLTHY	HEALTHY
<specific>	None	UNHLTHY	UNHLTHY

Table 80 PrFault status according to expected vs actual fitted modules

Alarms. See *Appendix D page 545* for a general description of the Alarms parameter.

- **Software.** Sumcheck error in block’s RAM data.
- **PrMajFlt.** TRUE if the primary processor detects a major hardware fault. This is usually a hardware fault that affects the system.
- **SeMajFlt.** TRUE if the secondary processor detects a major hardware fault. This is usually a hardware fault that affects the system.
- **PrMinFlt.** If Options.FaultAs is set to ‘Mismatch’ then TRUE if any I/O module in *PrFault* parameter is set ‘UNHLTHY’ (unhealthy) according to the primary processor.

If Options.FaultAs is set to ‘Redundcy’ then TRUE if any I/O module in *PrFault* parameter is set ‘UNHLTHY’ (unhealthy) and if corresponding expected I/O module is anything other than ‘None’.

- **SeMinFlt.** If Options.FaultAs is set to ‘Mismatch’ then TRUE if any I/O module in *SeFault* parameter is set ‘UNHLTHY’ (unhealthy) according to the secondary processor.

If Options.FaultAs is set to ‘Redundcy’ then TRUE if any I/O module in *SeFault* parameter is set ‘UNHLTHY’ (unhealthy) and if corresponding expected I/O module is anything other than ‘None’.

- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

AcSource. Simplex/Primary/Secondary. Select to indicate the source of *Actual_1* to *Actual16* data. Secondary can only be selected for a coupled system

Actual_1 to Actual16. None/AI2/AI3/AI4/ZI/AO2/DI16/DI4/DI8_LG/DI8_LGv2/DI8_CO/DI8_COv2/DI6_MV/DI6_HV/DO4_LG/DO4_LGv2/DO4_24/DO4_24v2/DO8/DO16/RLY4/RLY4v2/RLY8/FI2/AI8_TC/AI8_MA/AI8_RT/AI8_FMA. Displays the I/O module actually fitted on a specific site. For example, *Actual_1* is the actual physical module type fitted at site 1. The value is derived by the processor indicated by the *AcSource* parameter.

SeFault. Bit-fields indicating the status of the communications between the secondary processor and the selected I/O site, as indicated by the *Page* parameter.

- **Site_1 to Site16.** When *AcSource* is set to ‘Secondary’, health status of each site is derived as per the table below. Any other value of *AcSource* would show ‘HEALTHY’.

Expected I/O Module	Actual I/O Module	Options.FaultAs set to 'Mismatch'	Options.FaultAs set to 'Redundcy'
None	None	HEALTHY	UNHLTHY
None	<any>	UNHLTHY	HEALTHY
<specific>	<correct>	HEALTHY	HEALTHY
<specific>	<wrong>	UNHLTHY	HEALTHY
<specific>	None	UNHLTHY	UNHLTHY

Table 81 SeFault status based on expected vs actual fitted modules (with AcSource set to ‘Secondary’)

Options.

- **FaultsAs.** Determines the method used to derive health status and impact how the *PrFault* and *SeFault* parameters behave.
 - **Mismatch:** This mode should usually be selected. Here, the fault bit-fields (*PrFault* and *SeFault*) reflect whether the expected and actual site contents match. If they do, then the appropriate bit-fields are set to HEALTHY; if they don't match, the bit-fields are set to UNHEALTHY.
 - **Redundcy:** This is the legacy mode for this block. In this mode, the *PrFault* bit-field is PRIMARY:HEALTHY (FALSE) if an I/O module is expected in the site and a module can be communicated with, or UNHEALTHY (TRUE) otherwise. This means that in this mode, a site is considered unhealthy if no I/O module is expected. The *SeFault* bit-field will be set to SECONDARY:HEALTHY if an I/O module can be communicated with.

Notes: 1. By default, legacy configurations act in 'Redundcy' mode. Newly created blocks default to the 'Mismatch' mode.

2. For base sizes with fewer than 16 sites, the unavailable I/O slots are reported as Unknown in the 'Actual' parameters. These unavailable slots are always reported as HEALTHY.

ELINDIAG: ETHERNET LIN DIAGNOSTIC BLOCK

Block function

The ELINDIAG block collects low-level statistics on the operation of the Ethernet Local Instrument Network (ELIN). The information concerns the Media Access Control (MAC) layer and the Logical Link Control (LLC) layer. The block can be cached in other instruments.

Block parameters

Symbols used in this table are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status*
LLCport	LLC port in use - first valid port is 0	Integer	
MACport	MAC port in use - first valid port is 0	Integer	
MACType	MAC hardware type displayed in the block	Menu	
MACstate	Operating mode of MAC state machine	Menu	
Procesor	Operating mode of MAC processor	Menu	
LastErr	Value of last returned error	Integer	
ThisNode	LIN address of this station	Integer	
ThisIP	IP address of this station		
ThisPort	Port address of this station	Hex	
RemEntry	Remote LIN Node IP address table conversion number	Integer	
RemNode	Remote LIN Node address and segment	Hex	
PrimIP	Primary Processor IP address of remote LIN Node		
PrimPort	Primary Processor port number of remote LIN Node	Integer	
ScndIP	Secondary Processor IP address of remote LIN Node	Hex	
ScndPort	Secondary Processor port number of remote LIN Node	Integer	
ScWeight	State of Secondary Processor interface	Integer	
NAT	Network Address Translation of IP address and Port number	T/F	
Remstate	Quality of mapping to remote device	Menu	
Alarms			
Software	Block RAM data sumcheck error/network failure	T/F	
BadMAC	MAC data failure	T/F	
Combined	OR-ing of all Alarms bits	T/F	
txCount	Quantity of attempted transmissions	Integer	
txTimOut	Quantity of failed (timed-out) transmissions	Integer	
txReject	Quantity of failed (rejected) transmissions	Integer	
txlmmRsp	Quantity of transmitted responses	Integer	
noLmmRsp	Quantity of unsuccessful immediate response requests	Integer	
rxCount	Quantity of responses received	Integer	
rxReq	Quantity of request buffers received	Integer	
rxRspOk	Quantity of valid request buffers received	Integer	
rxUnkwn	Quantity of invalid request buffers received	Integer	
rxRspErr	Quantity of un-identified responses received	Integer	
rxRspBsy	Quantity of busy responses received	Integer	
ClearCnt	TRUE resets the parameters above to 0 (zero)	T/F	
PrevNode	Previous logical station's LIN address	Hex	
NextNode	Next logical station's LIN address	Hex	

*Read-only unless otherwise stated

Table 82 ELINDIAG Block parameters

Block specification menu

The following is given in addition to Table ELINDIAG-1.

Dbase, Block, Type. See Appendix D-2 for details of these 'header' fields.

MACType. Specifies the type of Media Access Control used. This should read ELIN. If an attempt is made to operate this block with a different LIN type (e.g. ALIN) then the correct LIN type will be displayed, but the remainder of this block will not function.

MACstate. (OFFLINE/IDLE/DEMAND_IN/DEMAND_DL/CLAIM_TOK/AWAIT_IFM/CHK_TOK_P/AWAIT_RES). Current operating mode of the media access control (MAC) state machine. This should always read OnLine. Any other value indicates a serious internal error.

Procesor. This should always read Primary. ELIN is redundant processor aware, but it is not possible to view blocks running on a secondary processor.

ThisIP. IP address of this instrument.

LastErr. Four hexadecimal digits showing the code of the last database error. This should always read 0 (zero). Refer to the specific instrument manual for a list of error codes and their meanings (for non-obvious cases).

RemEntry. This field shows the index number into a table of remote LIN Node Number mappings. As entries are added and removed from the table, so individual entries can change their index. Setting this field to “-1” allows you to enter the required remote node number into the RemNode field (*SeWeight* field does not work in this mode).

NOTE ELIN only maintains mappings of nodes it actually wishes to talk to - if there is no LIN traffic between nodes, there will be no mapping.

PrimIP. Primary Processor IP address of remote LIN Node.

Remstate. This indicates the quality of the mapping to the selected remote node. The values “unresolved” and “known” have obvious meanings. The value “assumed” indicates traffic has been received from the specified node using the specified IP address and Port Number, but that the mapping has not been confirmed by PRP. LIN communications can successfully take place using an assumed mapping, but this is normally only a transitory state.

txTimOut. This indicates the number of individual transmission attempts for which an immediate response has not been received within the appropriate time out. This can reveal the loss of response to a single transmission which would otherwise be hidden by retries.

EMAPDIAG: ELIN MAPPING DIAGNOSTIC BLOCK

Block function

The EMAPDIAG block collects a complete map of all LIN Nodes on this LIN Network regardless of whether it specifically communicates with other nodes. The block can be cached in other instruments.

Block parameters

Symbols used in this table are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Seg0_map	Each 16-bit bitfied shows a single LIN Node on this LIN Network	bitfied	
Seg1_map			
Seg2_map			
Seg3_map			
Seg4_map			
Seg5_map			
Seg6_map			
Seg7_map			
Seg8_map			
Seg9_map			
SegA_map			
SegB_map			
SegC_map			
SegD_map			
SegE_map			
SegF_map			
Alarms			  
Software	Block RAM data sumcheck error/network failure	T/F	
Combined	OR-ing of all Alarms bits	T/F	
NodeStat	LIN Node Number of monitored LIN Node	Hex	
NodePrev	Previous logical LIN address prior to NodeStat	Hex	
NodeNext	Next logical LIN address following NodeStat	Hex	
TimeNow	Current system time	Hex	
Lastl-Am	System time NodeStat was last confirmed	Hex	

Table 83 Block parameters

Block specification menu

The following is given in addition to *Table 83*.

Dbase, Block, Type. See *Appendix D* page 544 for details of these 'header' fields.

NodeStat. Monitor another LIN Node in this LIN Network by entering the LIN Node Number.

Lastl-Am. Shows the system time the LIN Node specified in the NodeStat field was last confirmed.

ETH_RT_LIM: ETHERNET RATE LIMIT DIAGNOSTIC BLOCK

Block function

This block provides diagnostic information, and automatic Ethernet Rate Limit Protection against high levels of incoming Ethernet traffic for maintaining the performance of the local closed loop control. High levels of incoming Ethernet traffic may be experienced during .udp broadcast storms, invalid cable/hub configurations (rings) and rapidly repeated requests from other hosts when instruments do not offer, or deny, a requested service.

When Ethernet traffic is too high or a UDP broadcast storm is detected, Protection is automatically activated, and deactivated when an acceptable level is consistently received. While Rate Limit Protection is activated the types of message it will respond are limited to LIN communications. However, if the incoming traffic becomes too great, the external LIN communications will also be stopped.

NOTE Synchronised Redundant processors become desynchronised if Ethernet Rate Protection is activated. The processors will automatically attempt to synchronise when the Ethernet Rate Protection deactivates.

The values indicating excessive traffic, *ConA*, *ConB*, *ConC*, and *ConD*, are derived from instrument development and test, and held in the instrument firmware, see *Table 85*. Actual Ethernet message rates on any particular Ethernet segment to and from any node on that segment can be established through commercially available test tools and software packages for Computers connected to that Ethernet segment.

Block parameters

Symbols used in this table are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
ProtAct	Ethernet Rate Limit Protection active	T/F	 
NumTime	Ethernet Rate Limit Protection counter	Integer	
ITimeStr	Last Time Ethernet Rate Limit Protection was activated	Time	
IDateStr	Last Date Ethernet Rate Limit Protection was activated	Date	
ITimeFin	Last Time Ethernet Rate Limit Protection was deactivated	Time	
IDateFin	Last Date Ethernet Rate Limit Protection was deactivated	Date	
ResetTim	Reset time and date displays	T/F	
Alarms			  
Software	Block RAM data sumcheck error/network failure	T/F	
ProcActi	Ethernet rate protection active	T/F	
Combined	OR-ing of all Alarms bits	T/F	
IPAddr	IP address of instrument		
Subnet	Subnet mask of instrument		
ConA	Normal level of processable data	Integer	
ConB	Ticks before Ethernet Rate Protection is deactivated	Integer	
ConC	Undelayed data while Ethernet Rate Limit Protection is active	Integer	
ConD	High Level of processable data	Integer	

Table 84 Block parameters

Block specification menu

The following is given in addition to *Table 84*.

Dbase, Block, Type. See *Appendix D* page 544 for details of these ‘header’ fields.

ProtAct. TRUE, if Ethernet Rate Limit Protection active. This is set TRUE when too much information is received within 10 mS.

While *ProtAct* is set TRUE all non LIN messages are discarded. If the number of LIN messages received while Ethernet Rate Limit Protection is active exceeds the value in *ConC* ALL messages are discarded and a delay is added.

NumTime. Shows the number of times the Ethernet Rate Limit Protection has been activated, incrementing each time the *ProtAct* sets TRUE.

ITimeStr, IDateStr. Shows the Time and Date respectively, when the Ethernet Rate Limit Protection was last activated, *ProtAct* set TRUE.

ITimeFin, IDateFin. Shows the Time and Date respectively, when the Ethernet Rate Limit Protection was last deactivated, *ProtAct* changes from TRUE to FALSE.

ResetTim. Used to reset the Time and Date displays respectively. When set TRUE, *ITimeStr*, *IDateStr*, *ITimeFin*, and *IDateFin* reset to 00:00:00 and 00/00/00 as appropriate

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Asserted, if a sumcheck error in block's RAM data occurs.
- **ProcActi.** Asserted, if Ethernet Rate Limit Protection is activated, *ProtAct* set TRUE.
- **Combined.** Asserted, if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

IPAddr. Shows the IP Address of the instrument running the database file, .dbf containing this block.

Subnet. Shows the Subnet Mask value of the instrument running the database file, .dbf containing this block.

ConA. Shows the number of messages received in a single tick indicating normal operation of the instrument, see table below. This is used to determine when Ethernet Rate Protection mode can be exited.

ConB. Shows the number of consecutive ticks that the Ethernet Rate Limit Protection will remain active for unless excessive data still exists in the instrument, see table below.

ConC. Shows the number of messages that can be processed without delay while the Ethernet Rate Limit Protection is activated in the instrument, see table below.

ConD. Shows the number of Ethernet messages that can be received in one tick before the instrument will Watchdog, see table below.

Instrument And Version	ConA (messages per tick)	ConB (ticks)	ConC (messages per tick)	ConD (messages per sec)
T2550/V3.0	75 messages	10	10 messages	10,000 messages

Table 85 Ethernet Rate Limit Values

NOTE The actual values in a real system for comparison to the limits above can be established through commercially available test tools and software packages for Computers connected to the appropriate Ethernet segment.

FDDADIAG: FTP DISTRIBUTED DATA ARCHIVING DIAGNOSTIC BLOCK

Block function

The FDDADIAG block displays information and alarms about the File Transfer Protocol (FTP) logging file transfer for internal archiving on a remote computer. It shows a subset of the FTP configuration data and allows monitoring of the status of transfers to the FTP servers. It also allows for the generation of alarms related to the failure to transferred files.

Block parameters

Symbols used in this table are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
ServerNo	Number of selected server	Enum	
Host	IP address of selected server	Integer	
Status		(A)BCD Hex	
SingCopy	Single copy mode	T/F	D
Enabled	Server enabled	T/F	
Alive	Server available	T/F	
Pending	One or more files pending	T/F	
HiPend	HiPend threshold reached	T/F	C
Primary	TRUE, this is Primary Server	T/F	
Complete	TRUE, last transfer was successfully completed	T/F	
Missed	TRUE, last transfer was missed, file deleted	T/F	
Cancelled	TRUE, last transfer was cancelled	T/F	B
Failed	TRUE, last transfer failed	T/F	
		T/F	
		T/F	
		T/F	A
		T/F	
		T/F	
		T/F	
HiPend	Max files awaiting transfer	Integer	
ResetCnt	Reset counters excluding pending requests	T/F	
Req_Cnt	Count of requested file transfers	Integer	
Done_Cnt	Count of files transferred	Integer	
Miss_Cnt	Count of files deleted before transferred	Integer	
Cncl_Cnt	Count of files cancelled before transferred	Integer	
Fail_Cnt	Count of file transfer failures	Integer	
Pend_Cnt	Count of files awaiting transfer	Integer	
TotalXfer	Total transferred to date	Mbytes	
LastFile	Last file transferred or being transferred	Alphanumeric	
LastExt	Extension of file defined in LastFile field	Alphanumeric	
LastRes	Result of last transfer	Enum	
LastSize	Size of file defined in LastFile field	kbs	
Alarms			
Software	Block RAM data sumcheck error/network failure	T/F	
Comms	FTP server in comms error	T/F	
Disable	Server currently disabled	T/F	
HiPend	Primary server has reached HiPend threshold	T/F	
Combined	OR-ing of all Alarms bits	T/F	

Table 86 Block parameters

Block specification menu

The following is given in addition to *Table 86*.

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

ServerNo. (Server1/Server2/Server3) Shows the user specified Primary Server. If the *SingCopy* field is set TRUE, the instrument will attempt to copy the selected file to the specified FTP server. If the file transfer is unsuccessful, the instrument will attempt to copy the file to the next Server, until successful.

NOTE This is only applicable if the *SingCopy* field is set TRUE.

Host. Shows the IP address of the FTP Server defined in the *ServerNo* field.

Status. Indicates the status of the FTP Server specified by the *ServerNo* and *Host* field.

- **SingCopy.** TRUE, indicates the instrument will copy the selected file to a single FTP Server as defined in the *ServerNo* field. FALSE, shows that the instrument will copy the selected file to all FTP servers.
- **Enabled.** TRUE, indicates the FTP Server is enabled. FALSE, shows that the FTP server is disabled, sets the *Disable.Alarms*.
- **Alive.** TRUE, indicates the FTP Server is available. FALSE, shows that the FTP server is unavailable, sets the *Comms.Alarms*.
- **Pending.** TRUE, indicates there are one or more files pending transfer. FALSE shows that the number of files currently awaiting transfer is 0 (zero).
- **HiPend.** TRUE, indicates the HiPend threshold has reached the permitted number of pending file transfers, sets the *HiPend.Alarms*.
- **Primary.** TRUE, indicates that this is Primary FTP Server.
- **Complete.** TRUE, indicates that the last transfer was successfully completed. 1 is added to the number in the *Done_Cnt* count field.
- **Missed.** TRUE, indicates that the file does not exist and therefore the last transfer was missed. 1 is added to the number in the *Miss_Cnt* field.
- **Cancelled.** TRUE, indicates that the last transfer was cancelled. 1 is added to the number in the *Cncl_Cnt* count field.
- **Failed.** TRUE, indicates that the last transfer failed. 1 is added to the number in the *Fail_Cnt* count field.

HiPend. The user defined value indicating the maximum number of files that can be pending, prior to transfer. It is the *HiPend.Alarm* and *HiPend.Status* limit for high pending file transfers. *HiPend.Alarm* is set TRUE if number of files awaiting transfer is equal or more than this value.

ResetCnt. TRUE, resets all counters to zero, excluding the count of files awaiting transfer.

Req_Cnt. Shows the total number of files that have been requested for transfer.

Done_Cnt. Shows the total number of files that have been transferred.

Miss_Cnt. Shows the total number of files that have been missed.

Cncl_Cnt. Shows the total number of files that have been cancelled.

Fail_Cnt. Shows the total number of files that have failed.

Pend_Cnt. Shows the total number of files that are currently awaiting transfer.

TotalXfer. Shows the total size, in Mbytes, of all files transferred to date.

LastFile. Shows the filename of the last file transferred.

LastExt. Shows the type of file (file extension, .bmp, .uyt, etc.).

LastRes. NONE/IN PROG/OK/FAIL/CANCELED/MISSED. Shows the result of the last attempted file transfer.

LastSize. Shows the size in bytes of the last file transfer attempted.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Comms.** TRUE, indicates that the FTP server in communications error.
- **Disabled.** TRUE, indicates that the FTP server is currently disabled.
- **HiPend.** TRUE, indicates that the FTP Primary server has reached High Pending limit as defined in the *HiPend* field.

FSM_DIAG: FILE SYSTEM MANAGEMENT DIAGNOSTIC BLOCK

Block function

The FSM_DIAG block allows the examination of a single FSM resource. Although FSM resources tend to be active for only short periods of time, this block continues to examine a resource after termination of the FILE connection it was servicing, thus allowing post-mortem examination of the connection.

Block parameters

Symbols used in this table are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status*
FSMindex	Select FSM resource for examination	Menu	
Protocol	Protocol version being used	Integer	
Type	Indicates Node as client or server	Menu	
NodeNo	Node number of the File connection	Hex	
DSap	Remote node's Service Access Point (SAP) in use	Integer	
LSap	Local node's Service Access Point (SAP) in use	Integer	
TimeStmp	XEC timestamp of last transmission	Integer	
TXFail	Quantity of failed transmission attempts via this FSM resource	Integer	
State	FSM resource. Idle = not used, Active = used	ENUM	
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
Combined	OR-ing of all Alarms bits	T/F	
Init	TRUE initialises fields Open to ReadTime	T/F	
Open	Quantity of Open attempts via this FSM resource	Integer	
Close	Quantity of Close attempts via this FSM resource	Integer	
Read	Quantity of Read attempts via this FSM resource	Integer	
Write	Quantity of Write attempts via this FSM resource	Integer	
Delete	Quantity of Delete attempts via this FSM resource	Integer	
Scan	Quantity of Scan attempts via this FSM resource	Integer	
Freescan	Quantity of Freescan attempts via this FSM resource	Integer	
FileChk	Quantity of File check attempts via this FSM resource	Integer	
ScanEx	Quantity of Scan exchange attempts via this FSM resource	Integer	
ReadTime	Quantity of ReadTime attempts via this FSM resource	Integer	
Spare1 to Spare6	} For future development		

*All fields are Read-Only unless otherwise stated

Table 87 Block parameters

Block specification menu

The following is given in addition to *Table 87*.

Dbase, Block, Type. See *Appendix D page 544* for details of these 'header' fields.

FSMindex. Allows selection of which FSM resource is being examined. This field is writable.

Protocol. Indicates whether FILE protocol version 1, 2 or 3 is being used.

Type. Indicates if this node initiated the file connection (client), or is responding to another node (server).

NodeNo. The node number of the remote end of this file connection.

DSap. Number of the remote node's Service Access Point (SAP) in use.

LSap. Number of the local node's Service Access Point (SAP) in use.

TimeStmp. XEC task timestamp of last transmission via this FSM resource.

TxFail. A count of failed transmission attempts via this FSM resource.

State. (IDLE/ACTIVE). 'Idle' indicates the FSM resource is not currently in use, whereas 'Active' indicates that the FSM resource is in use.

Init. Setting this TRUE initialises the contents of the other fields.

Open.

Close.

Read.

Write.

Delete.

Scan.

Freescan.

FileChk.

ScanEx.

ReadTime.

} A count of either transmission or receipt of requests dependant on whether this node is a client or server respectively.

Spare1 - Spare6. For future development.

FTQ_DIAG: PRMT QUEUES DIAGNOSTIC BLOCK

Block function

Each process within a T2550/T2750/T940(X)/ T102 Unit Controller or T302 Unit Supervisor ‘talks’ to the PRMT (processor redundancy management task) via a set of queues. The FTQ_DIAG block provides access to the low-level diagnostic statistics about each queue as maintained by the PRMT.

Block parameters

Symbols used in this table are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Refresh	Time between block updates (0-60.00 secs)	Secs	
PrQueue	Primary processor queue selection, by name	Menu	
PrMode	Display mode for the selected queue statistics	Menu	
PrIdent, SeIdent	} <i>For manufacturing use only</i>		
PrSender, SeSender			
PrSndEvt, SeSndEvt			
PrSndRsn, SeSndRsn			
PrRcvr, SeRcvr			
PrRcvEvt, SeRcvEvt			
PrRcvRsn, SeRcvRsn			
PrTx_Cnt, SeTx_Cnt			
PrNo_Buf, SeNo_Buf			
PrTx_Err, SeTx_Err			
PrRx_Cnt, SeRx_Cnt			
PrNo_Msg, SeNo_Msg			
PrRx_Err, SeRx_Err			
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
Combined	OR-ing of all Alarms bits	T/F	
SeQueue	Secondary processor queue selection, by name	Menu	
SeMode	Display mode for the selected queue statistics	Menu	

Table 88 Block parameters

Block specification menu

The following is given in addition to *Table 88*.

PrMode, SeMode. User input fields, allowing continuous update (AUTO), or initialisation (INIT) of the currently selected queue data, and also ‘freezing’ the displays (HOLD), to aid event timing.

PrQueue, SeQueue. User input fields, allowing selection of which queue’s statistics to display.

PrIdent to SeRcvRsn. These fields give intricate details of the lowest-level workings of the queuing mechanism and are unlikely to be of use to most users. In case of persistent problems, the manufacturees support staff may request inspection of these values as an aid to diagnostic measures.

PrTx_Cnt to SeRx_Err. These fields display a particular set of counts, in a way specified by the *PrMode* and *SeMode* parameters.

FWD_DIAG: FORWARDING STATISTICS BLOCK

Block function

The FWD_DIAG block displays statistics on the forwarding process within a bridge.

Block parameters

Symbols used in this table are explained in *Table 1*. Parameter information is given in *Table 89* following.

Parameter	Function	Units	Status*
Requests	Number of connect requests handled		
Response	Number of connection responses		
EDBConn	Number of EDB connect requests		
FileConn	Number of file connect requests		
OthrConn	Number of unknown connect requests (should be 0)		
LINUFwd	Number of non-connection based forwards (ELIN/LIN)		
LINApp	Number of appl messages successfully forwarded (ELIN/LIN)		
LINRtry	Number of retries performed (ELIN/LIN)		
LINFail	Number of complete loss of message, due to TX failure (ELIN/LIN)		
LINRate	Rate of updates (ELIN/LIN)		
ALINUFwd	Number of non-connection based forwards (ALIN)		
ALINApp	Number of appl messages successfully forwarded (ALIN)		
ALINRtry	Number of retries performed (ALIN)		
ALINFail	Number of complete loss of message, due to TX failure (ALIN)		
ALINRate	Rate of updates (ALIN)		
Alarms			
Software	Block RAM data sumcheck error / network failure		
Combined	OR-ing of all Alarms bits		
CTimeout	Number of connection paths timed out		
CRouteF	Number of failed to route connect requests		
CLoop	Attempts to route out of same port		
CCreateF	Failed to create connection entity		
NoSAPs	Number of sap request failures		
NoBufs	Number of buffer request failures		
ExtraRsp	Number of responses without a connection		
TimeStmp	Timestamp for last pass		
PassTime	The amount of time to do one pass		
Period	The period		
HoldOff	Number of holdoffs performed		

*Read-only unless otherwise stated

Table 89 Block parameters

FWD_LOG: FORWARDING LOG BLOCK

Block function

The FWD_LOG block displays information on the results of attempting to forward packets.

Block parameters

Symbols used in this table are explained in *Table 1*. Parameter information is given in *Table 90* following.

Parameter	Function	Units	Status
Alarms			  
Software	Block RAM data sumcheck error / network failure		
Combined	OR-ing of all Alarms bits		
<i>For transmissions onto the LIN:</i>			
Ltx_ok	Transmission successful		
Lno_resp	No response frame received		
Lacm_err	Token stolen		
Lresp_lo	Response frame lost		
Lsetup	Tx buffer in wrong segment		
Lllc_nav	LLC is down		
Ltx_full	No entries left in TX_CNTRL_T array		
Ltx_busy	Transmission is already in progress from this SAP to given destination		
Lbuf_nav	There are no MAC Tx buffers left		
Loffline	The Port is not forwarding		
Lbuf_len	The buffer is too long to be transmitted		
Lsap_all	The SAP used is not yet allocated		
Llin_nav	The ELIN/LIN requested is not physically fitted		
Lmac_nav	The MAC requested is not physically fitted		
Lsap_ill	The SAP used is illegal		
Lrem_llc	The LLC at the remote node did not accept the transmission		
<i>For transmissions onto the ALIN:</i>			
Atx_ok	Transmission successful		
Ano_resp	No response frame received		
Aacm_err	Token stolen		
Aresp_lo	Rresponse frame lost		
Asetup	Tx buffer in wrong segment		
Allc_nav	LLC is down		
Atx_full	No entries left in TX_CNTRL_T array		
Atx_busy	Transmission is already in progress from this SAP to given destination		
Abuf_nav	There are no MAC Tx buffers left		
Aoffline	The Port is not forwarding		
Abuf_len	The buffer is too long to be transmitted		
Asap_all	The SAP used is not yet allocated		
Alin_nav	The ALIN requested is not physically fitted		
Amac_nav	The MAC requested is not physically fitted		
Asap_ill	The SAP used is illegal		
Arem_llc	The LLC at the remote node did not accept the transmission		

Table 90 Block parameters

ICM_DIAG: INTERPROCESSOR COMMS MECHANISM STATISTICS BLOCK

Block function

The ICM_DIAG block provides ICM (Interprocessor Comms Mechanism) diagnostic statistics on the numbers and types of message being sent by each processor in a redundant pair. The ICM is the module responsible for the physical transfer of messages between the two processors.

Block parameters

Symbols used in this table are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Refresh	Time between block updates (0-60.00 secs)	Secs	
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
Combined	OR-ing of all Alarms bits	T/F	
PrMode	Display mode for the selected 'Pr' (primary) statistics	Menu	
PrRx_Cnt	Count of received messages		
PrTx_Cnt	Count of sent messages		
PrNo_Buf	Count of buffers used		
PrFail	Count of failed messages		
PrTx_Bsy	Count of times ICM too busy to transmit		
PrTx_Err	Count of times ICM transmit failed		
SeMode	} <i>Correspond to 'Pr' parameters above</i>		
SeRx_Cnt			
SeTx_Cnt			
SeNo_Buf			
SeFail			
SeTx_Bsy			
SeTx_Err			
Class	Class of 'Tx' and 'Rx' message statistics (below)		
Cls_Mode	Display mode for the selected 'Tx' and 'Rx' statistics	Menu	
TxBusy	Count of transmit messages lost due to ICM busy		
TxFail	Count of transmit messages lost due to other faults		
RxMissng	Count of messages sent to which there was no reply		
RxDuplct	Count of duplicate replies received		

Table 91 Block parameters

Block specification menu

The following is given in addition to *Table 91*.

Dbase, Block, Type. See *Appendix D page 544* for details of these 'header' fields.

PrMode, SeMode. (AUTO/INIT/HOLD/INITBOTH/HOLDBOTH) User input fields, specifying the way the 'Pr' and 'Se' parameters display their statistics.

- **AUTO** is the normal mode, with the statistics updating regularly (limited by the *Refresh* field).
- **INIT** resets the statistics.
- **HOLD** 'freezes' the statistics displays, e.g. to allow examination. Returning to AUTO from HOLD results in the statistics 'jumping' to the values reached while HOLD was active.
- **HOLDBOTH** (*PrMode* only) freezes both 'Pr' and 'Se' statistics displays simultaneously.
- **INITBOTH** (*PrMode* only) resets both 'Pr' and 'Se' statistics simultaneously.

Class. (EXCEPTION/DB/DBMNGR/LINSYNC/FILING/STATUS/SYNC/CLASS_7 to CLASS_15) User input field selecting the class of messages displayed by the *TxBusy*, *TxFail*, *RxMissng*, and *RxDuplct* fields.

Cls_Mode. (AUTO/INIT/HOLD) User input field, applying to the *TxBusy*, *TxFail*, *RxMissng*, and *RxDuplct* fields. Select AUTO for continuous update of the statistics of the current class, INIT to reset them, and HOLD to 'freeze' their displays.

TxBusy, TxFail, RxMissng, RxDuplct. The messages displayed by these fields are in the class selected by the *Class* parameter. They are available only for the primary T920 CPU.

IDENTITY: INSTRUMENT IDENTIFICATION/STATUS DIAGNOSTIC BLOCK

Block function

The IDENTITY block is used to enable product build identity to be checked. It can be used by any TACTICIAN LIN product range, and the T940(X) instrument.

NOTE With the appropriate switch setting this block will be automatically generated for the TACTICIAN instruments.

Block parameters

Symbols used in this table are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status				
Type	Machine Name	Alphanumeric					
SubType	Extended Machine Name information	Alphanumeric					
SerialNo	Inst. serial number (held on Compact Flash card)						
InstVers	Instrument software information	Alphanumeric					
Special	Special instrument software information						
BootVers	Boot software information	Alphanumeric					
TmplVers	Template library set information	Alphanumeric					
HWvers	Hardware build version number	Alphanumeric					
HWbuild	Hardware options build	Alphanumeric					
OtherSer	Sec. Inst. serial number (held on Compact Flash card)						
Alarms							
Software	Block RAM data sumcheck error / network failure	T/F					
Combined	OR-ing of all Alarms bits	T/F					
Config	Instrument configuration	(A)BC(D) Hex					
HotStart	Hot Start is configured	T/F	<table border="1"> <tr><td>1</td></tr> <tr><td>2</td></tr> <tr><td>4</td></tr> <tr><td>8</td></tr> </table> D	1	2	4	8
1							
2							
4							
8							
ColdStrt	Cold Start is configured	T/F					
UpgdGood	TRUE indicates a successful recent firmware upgrade	T/F					
UpgdFail	TRUE indicates an unsuccessful recent firmware upgrade	T/F					
Protectd	TRUE encrypts/deciphers databases on save/load	T/F	<table border="1"> <tr><td>1</td></tr> <tr><td>2</td></tr> <tr><td>4</td></tr> <tr><td>8</td></tr> </table> A	1	2	4	8
1							
2							
4							
8							
LINfeat	LIN configuration	(A)BC(D) Hex					
OLIN	Original LIN protocol supported	T/F					
ALIN	LIN over ArcNet protocol supported	T/F					
SLIN	LIN using Serial port protocol supported	T/F	<table border="1"> <tr><td>1</td></tr> <tr><td>2</td></tr> <tr><td>4</td></tr> <tr><td>8</td></tr> </table> D	1	2	4	8
1							
2							
4							
8							
ELIN	LIN using Ethernet protocol supported	T/F					
NWaudit	Network Audit facility supported	T/F					
Consumer	A network audit consumer is configured	T/F					
Iconsume	Instrument operating as a network audit consumer	T/F	<table border="1"> <tr><td>1</td></tr> <tr><td>2</td></tr> <tr><td>4</td></tr> <tr><td>8</td></tr> </table> A	1	2	4	8
1							
2							
4							
8							
SupComms	Supported communications	A(BCD) Hex					
S6000M	S6000 master comms is supported	T/F					
S6000S	S6000 slave comms is supported	T/F					
GWmodM	GW Modbus master comms is supported	T/F	<table border="1"> <tr><td>1</td></tr> <tr><td>2</td></tr> <tr><td>4</td></tr> <tr><td>8</td></tr> </table> D	1	2	4	8
1							
2							
4							
8							
GWmodS	GW Modbus slave comms is supported	T/F					
DCMmodM	DCM Modbus comms is supported	T/F					
DCMprofM	DCM Profibus comms is supported	T/F	<table border="1"> <tr><td>1</td></tr> <tr><td>2</td></tr> <tr><td>4</td></tr> <tr><td>8</td></tr> </table> C	1	2	4	8
1							
2							
4							
8							
GWprofM	GW Profibus Master comms is supported	T/F					
GWprofS	GW Profibus Slave comms is supported	T/F					
AlarmSup	Alarm Suppression is supported	T/F	<table border="1"> <tr><td>1</td></tr> <tr><td>2</td></tr> <tr><td>4</td></tr> <tr><td>8</td></tr> </table> B	1	2	4	8
1							
2							
4							
8							
ActComms	Active communications	A(BCD) Hex					
S6000M	S6000 master comms is active	T/F					
S6000S	S6000 slave comms is active	T/F	<table border="1"> <tr><td>1</td></tr> <tr><td>2</td></tr> <tr><td>4</td></tr> <tr><td>8</td></tr> </table> D	1	2	4	8
1							
2							
4							
8							
GWmodM	GW Modbus master comms is active	T/F					
GWmodS	GW Modbus slave comms is active	T/F					

Continued...

Parameter	Function	Units	Status
<i>Continued...</i>			
DCMmodM	DCM Modbus comms is active	T/F	1
DCMprofM	DCM Profibus comms is active	T/F	2
GWprofM	Profibus Master comms is active	T/F	4
GWprofS	Profibus Slave comms is active	T/F	8
AlarmSup	Alarm Suppression is active	T/F	1
			2
			4
			8
Feature1		(AB)C(D) Hex	
Licence	Licence Control System is supported	T/F	1
Access	Access Control System is supported	T/F	2
EthRtPro	Ethernet rate protection is supported	T/F	4
EncrptFS	AES encrypted files on the GFS filing system supported	T/F	8
Cluster	Clustering is supported	T/F	1
Auditor	Auditor System is supported	T/F	2
TOD	LIN Time Of Day synchronisation is supported	T/F	4
TimeZone	Time Zone aware	T/F	8
OLReconf	On-line reconfiguration is supported	T/F	1
SNTP	Simple Network Time Protocol supported	T/F	2
			4
			8
Feature2		A(BCD) Hex	
Recipe	Recipes are supported	T/F	1
Reports	Reports are supported	T/F	2
Batch	Batch processes are supported	T/F	4
SFC	Sequences are supported	T/F	8
LD_PRGM	Ladder Programs are supported	T/F	1
UxgSW	Instrument software supports data recording	T/F	2
UxgHW	Instrument hardware supports data recording	T/F	4
			8
HMI	HMI is fitted	T/F	1
Printer	Printer is fitted	T/F	2
Barcode	BarCode Reader is supported	T/F	4
Archive	Internal Archiving is supported	T/F	8
Duplex		ABC(D) Hex	
Duplex	Configured for duplex operation	T/F	1
Secondary	Secondary processor	T/F	2
Syncd	Currently synchronised	T/F	4
Left	Database is running in this processor	T/F	8
Right	Database is running in this processor	T/F	1
GWPFsimpl	GW Profibus Master configured as Simplex Only	T/F	2
SecWorse	The secondary processor has worse subsystem health	T/F	4
			8
State		A(BCD) Hex	
Running	A database is running	T/F	1
Start1	Reason for start-up	T/F	2
Start2		T/F	4
Start3		T/F	8
Tepid		Tepid Start performed	T/F
DbfRunEr	DBF/RUN file inconsistency	T/F	2
DbfStatE			4
UxgStatE			8
ConfBusy	Configurator in use	T/F	1
OLRpCng	Tentative On-line reconfig. changes are present	T/F	2
OLRtry	Trying On-line Reconfig. changes	T/F	4
OLRpsav	DBF does not match Applied On-line Reconfig. changes	T/F	8
RtProcAc	Ethernet Rate protection is active	T/F	1
			2
			4
			8

Table 92 Block parameters

Block specification menu

The following is given in addition to *Table 92*.

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

Type. Identical to the MACHINE_NAME, e.g T2550, T940, etc..

SubType. This is the extended machine name information (e.g. “X” for a T940-X). This field generally remains blank on T2550.

SerialNo. Instrument serial number as held on the Compact Flash card. This field is set to zero (0) if unknown or unsupported.

InstVers. This is the main software issue/release/revision information.

Special. This is the numeric representation of the ‘specials number’ version of the software (issue/release/revision). Usually set to zero (0).

BootVers. This is the boot software issue/release/revision information.

TmplVers. This is the template library set name/issue/release/revision information. It is the same name as the template library directory used by LINtools when configuring databases for this instrument.

HWvers. This is the hardware build version number.

HWbuild. This is the hardware options build. The enumeration values indicate the hardware build type, such as RS-485.

OtherSer. Instrument serial number as held on the Compact Flash card of the secondary processor. This field is set to zero (0) if unknown or unsupported. Derived at block update time, because the secondary processor might change whilst we are running.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

Config. Bitfields display the start-up configuration of the instrument.

- **HotStart.** TRUE, indicates the Hot Start DIP switch is in the ON position, and a Hot Start will be attempted on start-up.
- **ColdStart.** TRUE, indicates the Cold Start DIP switch is in the ON position, and a Cold Start will be attempted on start-up.
- **UpgdGood.** TRUE, indicates a successful recent firmware upgrade. When a new firmware upgrade is performed, for the first power-up after the upgrade (until it is next powered down), a successful upgrade will set this bit to TRUE. Contact the manufacturer if in any doubt about performing a firmware upgrade.
- **UpgdFail.** TRUE, indicates an unsuccessful recent firmware upgrade. When a new firmware upgrade is performed, for the first power-up after the upgrade (until it is next powered down), an unsuccessful upgrade will set this bit to TRUE. Contact the manufacturer if in any doubt about performing a firmware upgrade.
- **Protectd.** TRUE indicates database strategies are ‘scrambled’ when saved.

LINfeat. All fields are derived at block update time.

NOTE The *Consumer* and *Iconsume* bitfields are not supported by the T2550.

- **OLIN.** TRUE, indicates the original LIN protocol is supported.
- **ALIN.** TRUE, indicates the ArcNet protocol is supported.
- **SLIN.** TRUE, indicates the Serial LIN protocol is supported.
- **ELIN.** TRUE, indicates the Ethernet LIN protocol is supported.
- **NWaudit.** TRUE, indicates the Network Audit facility is supported. This is required for 21CFR pt11 approval.

- **Consumer.** TRUE, indicates a network audit consumer is configured. This field can change at block update time.
- **Iconsume.** TRUE, indicates the instrument is operating as a network audit consumer. This field can change at block update time.

SupComms. All fields are derived at block update time.

- **S6000M, S6000S.** TRUE, indicates S6000 master communications, and S6000 slave communications is supported, respectively.
- **GWmodM, GWmodS.** TRUE, indicates GW Modbus master communications, and GW Modbus slave communications is supported, respectively.
- **DCMmodM.** TRUE, indicates DCM Modbus communications is supported.
- **DCMprofM.** TRUE, indicates DCM Profibus communications is supported.
- **GWProfM, GWProfS.** TRUE, indicates GW Profibus master communications, and GW Modbus slave communications is supported, respectively.
- **AlarmSup.** TRUE indicates Alarm Suppression is supported

ActComms. All fields are derived at block update time.

- **S6000M, S6000S.** TRUE, indicates S6000 master communications, and S6000 slave communications is active, respectively.
- **GWmodM, GWmodS.** TRUE, indicates GW Modbus master communications, and GW Modbus slave communications is active, respectively.
- **DCMmodM.** TRUE, indicates DCM Modbus communications is active.
- **DCMprofM.** TRUE, indicates DCM Profibus communications is active.
- **GWProfM, GWProfS.** TRUE, indicates GW Profibus master communications, and GW Profibus slave communications is active, respectively.
- **AlarmSup.** TRUE indicates Alarm Suppression is active

Feature1. All fields are derived at block update time.

- **Licence.** TRUE, indicates this instrument supports a Licence Control System.
- **Access.** TRUE, indicates this instrument supports an Access Control System.
- **EthRtPro.** TRUE, indicates this instrument supports an Ethernet rate protection. Ethernet rate protection is used to determine when the processor is not processing all Ethernet traffic presented, but is allowing LIN communications.
- **EncriptFS.** TRUE, indicates this instrument's GFS filing system supports AES encrypted files.
- **Clusters.** TRUE, indicates this instrument supports clustering.
- **Auditor.** TRUE, indicates this instrument supports an Auditor system.
- **TOD.** TRUE, indicates this instrument supports LIN TOD, Time Of Day, synchronisation.
- **TimeZone.** TRUE, indicates this instrument is Time Zone Aware.
- **OLReconf.** TRUE, indicates on-line reconfiguration is supported.
- **SNTP.** TRUE, indicates simple network time protocol is supported. This allows the time to be synchronised with other instruments or computers that support SNTP.

Feature2. All fields are derived at block update time.

- **Recipe.** TRUE, indicates Recipes are supported.
- **Reports.** TRUE, indicates Reports are supported.
- **Batch.** TRUE, indicates Batch processes are supported.
- **SFC.** TRUE, indicates Sequences are supported.

- **Ladder Programs.** TRUE, indicates Ladder Programs are supported.
- **UxgSW/UxgHW.** TRUE, indicates Data Recording is supported by the software and hardware respectively.
- **HMI.** TRUE, indicates HMI is supported.
- **Printer.** TRUE, indicates Printer is supported.
- **Barcode.** TRUE, indicates BarCode Reader is supported.
- **Archive.** TRUE, indicates Internal Archiving is supported.

Duplex. All fields are derived at block update time.

- **Duplex.** TRUE, indicates this processor is configured to operated in duplex mode.
- **Secondary.** TRUE, indicates this is currently the secondary processor. This field should always be FALSE when viewed via the block.
- **Syncd.** TRUE, indicates this instrument is currently synchronised.
- **Left, Right.** TRUE, indicates processors are fitted and running in the left and right positions on the base unit respectively. This denotes which processor is currently operating as the Primary.
- **GWPfSmpl.** TRUE, indicates GW Profibus Master has been configured for “Simplex Only” operation. This is set using the Instrument Options Editor and the Simplex option ticked in the Profibus tab.
- **SecWorse.** TRUE, indicates the secondary processor has worse subsystem health than the primary. This only applies to instruments which allow a duplex pair to remain synchronised even when the secondary has worse subsystem health compared to that of the primary (for example, T2750 V4/0 and later).

State. All fields are derived at block update time.

- **Running.** TRUE, indicates this instrument is currently running a database. This field should always be TRUE when viewed via the block.
- **Start1, Start2, Start3.** Together these indicate the start reason as shown in the table below.

Start3	Start2	Start1	Binary Value	Reason
0	0	1	1	Hot
0	1	0	2	Cold
0	1	1	3	TermCfg
1	0	0	4	Network/Remote

Table 93 Reason for Startup

- **Tepid.** TRUE, indicates this instrument performed a Tepid Start last.
- **DbfRunEr.** TRUE, indicates an error was detected when the database was loaded.
- **DbfStatE.** TRUE, indicates the loaded and running database configuration corresponds to the .dbf file in the instruments file system
- **UxgStatE.** TRUE, indicates the loaded and running data recording configuration corresponds to the .uxg file in the instruments file system.
- **ConfBusy.** TRUE, indicates the Configurator resource is in use, i.e. the terminal configurator is in use or the network invoked configuration operations are currently underway.
- **OLRpCng.** TRUE, indicates ‘Tentative’ Online Reconfiguration changes exist in the instrument, i.e. the database has changes that are yet to be applied.
- **OLRtry.** TRUE, indicates Trying Online Reconfiguration changes, i.e. the database being tested using the Online Reconfiguration changes.
- **OLRpSav.** TRUE, indicates Online Reconfiguration changes have been applied, but the .dbf has not yet been updated to match.
- **RtProcAc.** TRUE, indicates the Ethernet rate protection algorithm is active.

ISB_DEXT: INTERNAL SERIAL BUS DIAGNOSTIC EXTENSION BLOCK

Block function

The ISB_DEXT block is an extension to diagnostics provided by the ISB_DIAG block. It breaks down ISB errors into error type, rather than errors per node as provided by the ISB_DIAG block.

Block parameters

Symbols used in this table are explained in *Table 1*. Parameter information is given in *Table 94* following.

Parameter	Function	Units	Status
TimeSlot	Count of ISB timeslot overflow errors		
Timeout	Count of timeout errors		
Parity	Count of Rx parity errors		
Framing	Count of Rx framing errors		
Overrun	Count of ISB overrun errors		
DIC	Count of Rx DIC errors		
Address	Count of Rx address errors		
Sequence	Count of Rx sequence errors		
MRB_NAK	Count of NAK replies		
MRB_CAN	Count of CAN replies		
ACK	Count of invalid ACK replies		
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
Combined	OR-ing of all Alarms bits	T/F	
Mode	AUTO/INIT/HOLD	Menu	
FreeRAM	Unused ISB data store RAM		

Table 94 Block parameters

NOTE Errors counters are byte-sized, so they have a maximum count value of 255 and will overflow back to zero.

NOTE Mode allows the error values to be frozen (HOLD), or reset to zero (INIT).

ISB_DIAG: INTERNAL SERIAL BUS PERFORMANCE BLOCK

Block function

The ISB_DIAG block is to help analyse the performance of the ISB (Internal Serial Bus).

Block parameters

Symbols used in this table are explained in *Table 1*. Parameter information is given in *Table 95* following.

Parameter	Function	Units	Status
TimeSlot	Count of ISB timeslot overflow errors		
Usage n ($n=1-16$)	ISB traffic for this node	%	
Spare	Unused ISB bandwidth	%	
Alarms			  
Software	Block RAM data sumcheck error / network failure	T/F	
Combined	OR-ing of all Alarms bits	T/F	
Errors n ($n=1-16$)	Errors to this node	count	
Mode	AUTO/INIT/HOLD	Menu	

Table 95 Block parameters

NOTE Errors counters are byte-sized, so they have a maximum count value of 255 and will overflow back to zero.

NOTE *Mode* allows the error values to be frozen (HOLD), or reset to zero (INIT).

LIN_DEXT: LIN HIGH-LEVEL DIAGNOSTIC EXTENSION BLOCK

Block function

The LIN_DEXT block collects high (application) level statistics on the operation of the Local Instrument Network (LIN) - complementing the low-level activity of the LIN_DIAG block.

The block specification menu is divided into two main sections: well-known message counters in the left column, and database message counters in the right (plus some general parameters). The upper part of each column counts incoming messages, and the lower counts outgoing messages. Counters can be reset or suspended via the *Mode* field.

The LIN uses two message types:

- *unconfirmed messages*, which elicit an immediate response from the receiving node but no further acknowledgement. (Unconfirmed messages are marked with a dagger (†) in *Table 96* below.)
- *confirmed messages*, which are acknowledged immediately by the receiving node. Further processing then takes place before a confirmation with additional information is sent.

Block parameters

Symbols used in this table are explained in *Table 1*. Parameter information is given in *Table 96* following.

Parameter	Function	Units	Status*
Sparen	Used to display ThisNode, NextNode, PrevNode	Hex	
wkCReqIn	Well known service confirmed requests received	Integer	
wkCRspln	Well known service confirmations received	Integer	
wkRjctIn	Well known service rejects received	Integer	
wkUReqIn	Well known service unconfirmed requests received	Integer	
wkUknwIn	Well known service unrecognised messages recd.,	Integer	
wkConfrm	Number of Well Known Confirmations sent	Integer	
wkEDBCRq	External database connection requests sent	Integer	
wkFileRq	Remote filing system requests sent	Integer	
wkIdntRq	Remote identify requests sent	Integer	
wkStStRq	Remote database start/stop requests sent	Integer	
wkLoadRq	Remote database load requests sent	Integer	
wkSaveRq	Remote database save requests sent	Integer	
wkLRBcst	Number of LRA broadcasts sent	Integer	
wkNRBcst	Number of routing table broadcasts sent	Integer	
wkTxOK	Total well known transmissions sent successfully	Integer	
wkTxFail	Total well known transmissions sent unsuccessfully	Integer	
Alarms			  
Software	Block RAM data sumcheck error / network failure	T/F	
Combined	OR-ing of all Alarms bits	T/F	
Mode	Block operating mode	Menu	
dbCReqIn	Database service confirmed requests received	Integer	
dbCRspln	Database service confirmations received	Integer	
dbRjctIn	Database service rejects received	Integer	
dbUReqIn	Database service unconfirmed requests received	Integer	
dbUknwIn	Database service unrecognised messages recd.,	Integer	
dbConfrm	Database service confirmation messages sent	Integer	
dbDiscRq	Database disconnect requests sent	Integer	
dbUpdtRq†	Cached block update messages sent	Integer	
dbVrfyRq	Database verify requests sent	Integer	
dbAtchRq	Cached block attach requests sent	Integer	
dbChAtRq	Cached block change attachment requests sent	Integer	
dbWFldRq†	Remote write field (update connect)messages sent	Integer	
dbAAlmRq	Remote acknowledge alarm messages sent	Integer	
dbWAlmRq	Remote write alarm messages sent	Integer	
dbHeartB†	Database heartbeats sent	Integer	
dbTxOK	Total database transmissions sent successfully	Integer	
dbTxFail	Total database transmissions sent unsuccessfully	Integer	

*All parameters except Mode have read-only status at runtime.

†Unconfirmed messages.

Table 96 Block parameters

LINMAPD: LIN MAPPING DIAGNOSTIC BLOCK

Block function

The LINMAPD block will allow you to monitor the view of the ELIN network only, as seen by the Primary and Secondary processors, while operating in Duplex mode - the block will not function when operating in Simplex mode. PriFlags (Primary processor) and SecFlags (Secondary processor) subfields are then used to indicate which processor has a better comprehension of the network, e.g. which processor can communicate with more nodes or can successfully cache more blocks.

NOTE This block can only be used to monitor ELIN networks and not ALIN. It will not report for nodes on the ALIN side of an ELIN/ALIN bridge.

Using the LINMAPD block

By including the information gathered by this block in the LIN database, a changeover can be forced between Primary and Secondary processors, if the Secondary has a better comprehension of the network. Normally a changeover would only occur automatically due to failure occurring with or within the Primary processor, or if the Primary were to totally lose Ethernet communications.

To force a changeover to occur from within the LIN database, the Redundancy Control (RED_CTRL) Diagnostic block must be included in the LIN database. By wiring into the Change, Sync and Desync fields of this block the LIN database can cause a changeover to occur, and the processors to Sync/Desync as required.

It is also practical to wire the PriFlags (Primary processor) and SecFlags (Secondary processor) subfields from this block into appropriate alarm fields (i.e. using the DIGALARM block) to assert alarms when the Primary and Secondary processor network views differ.

NOTE In cases of potential network failure the control action of the LIN database can be halted or frozen, by using the PriFlags (Primary processor) and SecFlags (Secondary processor) subfields to force blocks (e.g. PID) into Hold etc.

This block may be used on networks where the Spanning Tree (STP) and/or Rapid Spanning Tree Protocols (RSTP) are being used. Failures in parts of the network protected by RSTP should not affect LINMAPD as it will quickly recover the network before the instrument has realized it can no longer see some nodes. However where failures occur on network segments protected only by STP, the LINMAPD block can indicate different network views for the Primary and Secondary processors. This is due to the time taken for STP to recover the network. The effect on LINMAPD will be determined by where the failure actually occurs within the network and if this affects nodes that the LINMAPD is monitoring, and the value of the STP parameter Forward Delay that ultimately determines the recovery time.

The LINMAPD block should quickly indicate the loss of nodes from the network if the instrument is actively communicating with those nodes via cached blocks in and/or out. Nodes that the instrument is not directly communicating with may take up to 1 minute to be removed from display by LINMAPD, once they can no longer see some nodes.

Block parameters

Symbols used in this table are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Seg_Mask	Remote node address. Respective bit MUST be TRUE	Bit	
Seg0Pmap to SegFPmap	Primary can communicate withsegment	Hex	
Timer			
Alarms	Period before asserting PriFlags or SecFlags	Mins.Secs	
Software	Block RAM data sumcheck error/network failure	T/F	
Combined	OR-ing of all Alarms bits	T/F	
Seg0Smap to SegFSmap	Secondary can communicate withsegment	Hex	

Continued...

Parameter	Function	Units	Status
<i>Continued...</i>			
PriFlags	Primary Instrument's comms status	(AB)CD Hex	
SegReach	Comms with segment of nodes possible	T/F 1	D
NodReach	Comms with node addresses possible	T/F 2	
SegBeter	Comms with nodes in the segment possible	T/F 4	
MorBkIn	Comms with blocks cached in at this node possible	T/F 8	
MorBkOut	Comms with blocks cached in at other nodes possible	T/F 1	C
MorBkTot	Comms with blocks regardless of where cached possible	T/F 2	
		4	
		8	
SecFlags	Secondary Instrument's comms status	(ABC)D Hex	
SegReach	Comms with segment of nodes possible	T/F 1	D
NodReach	Comms with node addresses possible	T/F 2	
SegBeter	Comms with nodes in the segment possible	T/F 4	
MorBkIn	Comms with blocks cached in at this node possible	T/F 8	
MorBkOut	Comms with blocks cached in at other nodes possible	T/F 1	C
MorBkTot	Comms with blocks regardless of where cached possible	T/F 2	
		4	
		8	

Table 97 Block parameters

Block specification menu

The following is given in addition to *Table 97*.

Dbase, Block, Type. See *Appendix D page 544* for details of these 'header' fields.

Seg_Mask. This field is used to control which LIN node 'segments' are to be considered when generating the PriFlags and SecFlags field. The respective bit must be TRUE for that segment to be considered. A LIN node segment is a group of 16 LIN node addresses, e.g. LIN node segment 0x06 is the LIN node addresses 0x60 through 0x6F (inclusive). *Seg_Mask* contains 1 bit per segment, bit 0 refers to LIN node segment 0x00 (addresses 0x00 through 0x0F), bit 15 to LIN node segment 0x0F (addresses 0xF0 through 0xFF).

NOTE LIN node addresses 0x00 and 0xFF are not legal values, e.g. to consider LIN node segment 0x06, *Seg_Mask* should be set to 0x0040.

SegnPmap, SegnSmmap. These fields indicate the ability of the primary and secondary processors to communicate with individual remote LIN nodes. Each field refers to the 16 instruments within a particular LIN node segment (e.g. Seg6Pmap refers to the primary's view of LIN node segment 0x06 = LIN node addresses 0x60 through 0x6F). Bit 0 refers to the lowest numbered node within that LIN node segment (e.g. 0x60 for segment 0x06), bit 15 to the highest numbered node within that LIN node segment (e.g. 0x6F for segment 0x06). If the bit is TRUE, communications to that node is possible, but it does not necessarily mean that communications to that node is actually occurring. If a bit is seen to flicker TRUE/FALSE it indicates that the remote node is attempting to cache invalid blocks out of the LIN node running the LINMAPD block (i.e. a configuration error in the remote LIN node).

PriFlags, SecFlags. These indicate the following conditions for either processor:

- **SegReach.** This is set TRUE, if the selected processor can communicate with more segments of a network than the other processor. If wired to the *Change, Sync* or *Desync* fields of the RED_CTRL block the appropriate action will be taken.
- **NodReach.** This is set TRUE, if the selected processor can communicate with more LIN nodes on a network than the other processor. If wired to the *Change, Sync* or *Desync* fields of the RED_CTRL block the appropriate action will be taken.
- **SegBeter.** This is set TRUE, if the selected processor can communicate with more LIN nodes in a segment than the other processor. If wired to the *Change, Sync* or *Desync* fields of the RED_CTRL block the appropriate action will be taken.

Example: If the Primary processor can see a single LIN Node on Segments A and B, but none on C, while the Secondary can only see 3 nodes on Segment C, the *SegReach* and *SegBeter* will be asserted for the Primary processor, but *NodReach* will be asserted for the Secondary processor. However, if the Secondary processor then saw a node appear on Segment B then it will assert its *SegBeter* flag and the Primary processor will clear its *SegBeter* flag.

- **MorBkIn.** This is set TRUE, if the selected processor can communicate with more blocks cached in the processor than the other processor.
- **MorBkOut.** This is set TRUE, if the selected processor can communicate with more blocks cached out of the processor than the other processor.
- **MorBkTot.** This is set TRUE, if the selected processor can communicate with more blocks cached in and out of the processor than the other processor.

Timer. This is the debounce in seconds applied before asserting the *PriFlags* and *SecFlags*. It can be adjusted so these flags do not change immediately if the Primary and Secondary views differ. This could be used to prevent changeovers occurring whilst the network is actually recovering from a failure or to avoid constantly initiating a changeover where momentary network failures occur.

LLC_DIAG: LOGICAL LINK CONTROL (LLC) DIAGNOSTIC BLOCK

Block function

The LLC_DIAG block is used to indicate the performance of the Logical Link Control (LLC) at a specified port.

Block parameters

Symbols used in this table are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status*
LLCport	LLC port in use - first valid port is 0	Integer	
MACport	MAC port in use - first valid port is 0	Integer	
MACType	MAC hardware type displayed in the block	Menu	
ThisNode	LIN address of this station	Integer	
NextNode	Next logical station's LIN address	Hex	
Alarms			
Software	Block RAM data sumcheck error/network failure	T/F	
Combined	OR-ing of all Alarms bits	T/F	
LLCstate	Operating mode of LLC state machine	Menu	
SAPsfree	Quantity of available Service Access Points	Integer	
SAPsbusy	Quantity of Service Access Points (SAPs) in use	Integer	
Tx_free	Quantity of spare transmit buffers	Integer	
Tx_alloc	Quantity of transmit buffers allocated	Integer	
Tx_busy	Quantity of transmit buffers in use	Integer	
Tx_ready	Quantity of transmit buffers ready for use	Integer	
Rx_free	Quantity of spare receive buffers	Integer	
Rx_alloc	Quantity of receive buffers allocated	Integer	
Rx_busy	Quantity of receive buffers in use	Integer	
Rx_bufav	Quantity of receive buffers available	Integer	
Rx_ready	Quantity of receive buffers ready for use	Integer	

*Read-only unless otherwise stated

Table 98 Block parameters

Block specification menu

The following is given in addition to *Table 98*.

Dbase, Block, Type. See *Appendix D page 544* for details of these 'header' fields.

LLCport. LLC port in use - first valid port is 0.

MACport. MAC port in use - first valid port is 0.

MACType. Specifies the type of Media Access Control (MAC) used. This should read ELIN. If an attempt is made to operate this block with a different LIN type (e.g. ALIN) then the correct LIN type will be displayed, but the remainder of this block will not function.

ThisNode. LIN address of this node.

NextNode. Next logical node's LIN address.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block's RAM data.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

LLCstate. (INIT/START/UP/STOP/DOWN/ABORT). Current operating mode of the Logical Link Control (LLC) state machine.

SAPsfree. Quantity of spare Service Access Points (SAPs). Equal to 1 for SAP zero + 1 for the well-known SAP + 1 for each remote database.

SAPsbusy. Quantity of Service Access Points (SAPs) in use. Equal to 1 for SAP zero + 1 for the well-known SAP + 1 for each remote database.

Tx_free. Quantity of spare transmit buffers. Equal to 1 for the well-known SAP + 1 for each remote database.

Tx_alloc. Quantity of transmit buffers allocated. Equal to 1 for the well-known SAP + 1 for each remote database.

Tx_busy. Quantity of transmit buffers in use. Equal to 1 for the well-known SAP + 1 for each remote database.

Tx_ready. Quantity of transmit buffers ready for use. Equal to 1 for the well-known SAP + 1 for each remote database.

Rx_free. Quantity of available receive buffers. Equal to 1 for the well-known SAP + 1 for each remote database.

Rx_alloc. Quantity of receive buffers allocated. Equal to 1 for the well-known SAP + 1 for each remote database.

Rx_busy. Quantity of receive buffers in use. Equal to 1 for the well-known SAP + 1 for each remote database.

Rx_bufav. Quantity of invalid request buffers received.

Rx_ready. Quantity of receive buffers ready for use. Equal to 1 for the well-known SAP + 1 for each remote database.

NATCDIAG: NETWORK AUDIT TRAIL CONSUMER DIAGNOSTIC BLOCK

Block function

The NATCDIAG block allows the display of Audit consumer configuration and diagnostics. The block can be cached in other instruments.

Block parameters

Symbols used in this table are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
ClientNo	Client Provider to view	Menu	
NodeNo	Node address of the specified Client Provider	Hex	
ResetCnt	Reset all counters	T/F	
Alarms			
Software	Block RAM data sumcheck error/network failure	T/F	
DisCnct	Specified Provider not connected	T/F	
Inactive	Specified Provider connected but not active	T/F	
Overflow	Overflow indication activated	T/F	
PowerUp	Specified Provider has power cycled	T/F	
MsgsLost	Lost messages indication activated	T/F	
Combined	OR-ing of all Alarms bits	T/F	
State	Current operating state of the specified Provider	Menu	
Recv_Cnt	Quantity of successful alarm transmissions recieved	Integer	
Bad_Cnt	Quantity of failed connections, Provider to Client	Integer	
n_Cnt	Quantity of connections, Provider to Consumer	Integer	
Act_t	Quantity of Client alarm transmissions from Provider	Integer	
Sync_Cnt	Quantity of Client re-synchronisations with Provider	Integer	
Over_Cnt	Quantity of Overflow indications	Integer	
Gap_Cnt	Quantity of undefined periods in Audit Trail	Integer	
Pwr_Cnt	Quantity of PowerUp indications	Integer	
Ign_Cnt	Quantity of failed (ignored) alarm transmissions	Integer	

Table 99 Block parameters

Block specification menu

The following is given in addition to *Table 99*.

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

State. (UNINIT/INIT/CONNECT/ACTIVE) Current operating state of the selected Client.

NATPDIAG: NETWORK AUDIT TRAIL PROVIDER DIAGNOSTIC BLOCK

Block function

The NATPDIAG block allows the display of information for a selected network Audit Trail provider. The block can be cached in other instruments.

Block parameters

Symbols used in this table are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
ProvNo	Provider to be viewed	Menu	
NodeNo	Node address of the specified Provider	Hex	
FiltType	Specifies the alarms to be included	(A)BCD Hex	
AlmAct	Include alarm active (default = TRUE)	T/F	<div style="border: 1px solid black; padding: 2px; display: inline-block;"> 1 2 4 8 </div> D
AlmClr	Include alarm clear (default = TRUE)	T/F	
AlmAck	Include alarm acknowledgements (default = TRUE)	T/F	
CacheAm	(Default = FALSE)	T/F	
EvtSys	Include system events (default = TRUE)	T/F	<div style="border: 1px solid black; padding: 2px; display: inline-block;"> 1 2 4 8 </div> C
EvtBlk	Include block events (default = TRUE)	T/F	
EvtNote	Include operator notes (default = TRUE)	T/F	
EvtChng	Include field change events (default = TRUE)	T/F	
MsgAct	Include message activation (default = FALSE)	T/F	<div style="border: 1px solid black; padding: 2px; display: inline-block;"> 1 2 4 8 </div> B
MsgClr	Include message clears (default = FALSE)	T/F	
MsgAck	Include message acknowledgements (default = FALSE)	T/F	
(Spare)	(Default = FALSE)	T/F	
FiltPriA	Min. alarm priority for inclusion (1 - 15)	Integer	
FiltPriE	Min. event priority for inclusion (1 - 15)	Integer	
ResetCnt	Reset all counters	T/F	
LoSpcLim	Low space alarm limit	Integer	
LoSpcHys	Hysteresis band applied to clear Low space alarm limit	Integer	
OverHys	Hysteresis band applied to clear Overflow indication	Integer	
Que_Size	Quantity of Alarm queue	Integer	
Alarms			
Software	Block RAM data sumcheck error/network failure	T/F	
DisCnnct	Specified Provider not connected	T/F	
Inactive	Specified Provider connected but not active	T/F	
Overflow	Overflow indication activated	T/F	
LoSpc	LoSpc indication activated	T/F	
Combined	OR-ing of all Alarms bits	T/F	
State	Current operating state of the specified Provider	Menu	
Xmit_Cnt	Quantity of successful alarm transmissions	Integer	
Rej_Cnt	Quantity of failed (rejected) alarm transmissions	Integer	
Con_Cnt	Quantity of connections from Provider to Client	Integer	
Act_Cnt	Quantity of Client alarm transmissions from Provider	Integer	
Sync_Cnt	Quantity of Client re-synchronisations with Provider	Integer	
LoSpc	Low space indication	T/F	
LoSpcCnt	Quantity of Low space alarm indications	Integer	
Overflow	Overflow indication	T/F	
Over_Cnt	Quantity of Overflow indications	Integer	
Que_Lag	Quantity of alarms yet to be transmitted to Client	Integer	
LagHiWm	Record of maximum Que_Lag since last reset	Integer	
Ign_Cnt	Quantity of failed (ignored) alarm transmissions	Integer	

Table 100 Block parameters

Block specification menu

The following is given in addition to *Table 100*.

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

ProvNo. (PROV_1/PROV_2/PROV_3) Monitor another LIN Node in this LIN Network by entering the LIN Node Number.

State. (UNINIT/INIT/CONNECT/ACTIVE) Current operating state of the specified Provider.

NET_DIAG: NETWORK DIAGNOSTIC BLOCK

Block function

The NET_DIAG block summarises the operating mode of a multi-segment network.

Block parameters

Symbols used in this table are explained in *Table 1*. Parameter information is given in *Table 101* following.

Parameter	Function	Units	Status*
LNode	LIN node address		
LMode	LIN operating mode (Full/Compatible)		
LFault	LIN fault		
LDupSeg	More than one seg. no. defined on the LIN cable (full mode)		
LSpltSeg	More than one cable defines the same seg. no.		
LSister	Got at least 1 sister redundant bridge		
LoListen	Low listen address for old instruments		
HiListen	High listen address for old instruments		
Alarms			
Software	Block RAM data sumcheck error / network failure		
Combined	OR-ing of all Alarms bits		
ANode	ALIN node address		
AMode	ALIN operating mode (Full/Compatible)		
AFault	ALIN fault		
ADupSeg	More than one seg. no. defined on the ALIN cable (full mode)		
ASpltSeg	More than one cable defines the same seg. no.		
ASister	Got at least 1 sister redundant bridge		

**Read-only unless otherwise stated*

Table 101 Block parameters

NETHOST: NETHOST DIANOSTIC BLOCK

Block function

The netHOST block summarises the properties, operating mode and diagnostics of connected Profibus Master netHOST devices. It provides access to the physical netHOSTs connected to the instrument to provide Profibus Master functionality. The block provides information on both the ‘left’ and ‘right’ netHOSTs, including their IP address, ADR decade switch settings and firmware information in addition to status and alarm information.

Block parameters

Symbols used in this table are explained in *Table 1*. Parameter information is given in *Table 102* following.

Parameter	Function	Units	Status*
MyIP	IP address of instrument to which the netHOSTs are connected		
L_Mode	Mode of left netHOST: primary or secondary		
R_Mode	Mode of right netHOST: primary of secondary		
L_IP	Configured IP address for the left netHOST		
R_IP	Configured IP address for the right netHOST		
L_ADR	Expected ADR address on left netHOST’s decade switches		
R_ADR	Expected ADR address on right netHOST’s decade switches		
L_Serial	Serial number of left netHOST connected to instrument		
R_Serial	Serial number on right netHOST connected to instrument		
L_Device	Device version of the left netHOST		
R_Device	Device version of the right netHOST		
R_FWMaj	Major firmware release of right netHOST		
L_FWMaj	Major firmware release of left netHOST		
R_FWMin	Minor firmware release of right netHOST		
L_FWMin	Minor firmware release of left netHOST		
L_FWBI	Build number of firmware of left netHOST		
R_FWBI	Build number of firmware of right netHOST		
L_FMRev	Firmware revision number of left netHOST		
R_FMRev	Firmware revision number of right netHOST		
L_status			
DBstop	Database stopped		
SubntBad	Left netHOST and instrument are on different subnets		
Missing	Left netHOST cannot be found		
IPwrong	Left netHOST has unexpected IP address		
ADRwrong	Left netHOST has unexpected ADR setting on decade switches		
DevBad	Left netHOST is of an unsupported model		
FirmBad	Left netHOST has an unsupported firmware installed		
R_status			
DBstop	Database stopped		
SubntBad	Right netHOST and instrument are on different subnets		
Missing	Right netHOST cannot be found		
IPwrong	Right netHOST has unexpected IP address		
ADRwrong	Right netHOST has unexpected ADR setting on decade switches		
DevBad	Right netHOST is of an unsupported model		
FirmBad	Right netHOST has an unsupported firmware installed		
Alarms			
Software	Block RAM data sumcheck error / network failure		
FaultL	Left netHOST has an alarm condition		
FaultR	Right netHOST has an alarm condition		
Combined	OR-ing of all Alarms bits		

*Read-only unless otherwise stated

Table 102 Block parameters

Block parameters

Symbols used in this table are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Block specification menu

The following is given in addition to *Table 102*.

- **Dbase, Block, Type.** See *Appendix D page 544* for details of these ‘header’ fields.
- **MyIP.** The IP address of the instrument to which the netHOSTs are connected (the T2750, for example).
- **Alarms.** text goes here
- **L_Mode, R_Mode.** The mode of the left and right netHOSTs: *primary* or *secondary*, which reflects the mode of the left processor on the instrument.
- **L_IP, R_IP.** The configured IP address of the left and right netHOSTs respectively, which makes up the netHOST’s identity when combined with the value of the ADR decade switches (*L_ADR* and *R_ADR* parameters).
- **L_ADR, R_ADR.** The expected value of the physical decade switches on the left and right netHOSTs respectively. When combined with the configured IP address of netHOST (*L_ADR* or *R_ADR* parameter), this makes up the netHOSTs’ identity.
- **L_Serial, R_Serial.** The serial number of the left and right netHOSTs respectively, connected to the instrument.
- **L_Device, R_Device.** The device version of the left and right netHOSTs respectively, connected to the instrument.
- **L_FWMaj, L_FWMin, L_FWBlid, L_FWRev.** The major (*L_FWMaj*), minor (*L_FWMin*), build (*L_FWBlid*) and revision (*L_FWRev*) firmware version for the left netHOST.
- **R_FWMaj, R_FWMin, R_FWBlid, R_FWRev.** The major (*R_FWMaj*), minor (*R_FWMin*), build (*R_FWBlid*) and revision (*R_FWRev*) firmware version for the right netHOST.
- **L_status, R_status.** A bitfield containing status and error information for the left (*L_status*) and right (*R_status*) netHOSTs.

DBstop. Database stopped. Set TRUE if the status of the netHOST cannot be derived because the database isn’t running. This is not an indication of a fault in the netHOST, but rather that the database is not running.

SubntBad. Bad subnet. Set TRUE if the instrument and netHOST are configured on different IP subnets.

Missing. Missing netHOST. Set TRUE if the netHOST cannot be found.

IPWrong. IP address wrong. This status bit is currently non-functional.

ADRWrong. ADR wrong. Set TRUE if the netHOST has a correct IP address but an unexpected ADR setting.

DevBad. Bad Device. Set TRUE if the netHOST device is unsupported (wrong hardware version).

FirmBad. Firmware Bad. Set TRUE if the netHOST’s firmware is unsupported (wrong version).

NODE_MAP: LIN NODE PROTOCOL BLOCK

Block function

The NODE_MAP block only applies to the Original OLIN. XLIN is an extension to the original LIN communications which added the ability to route via bridges. OLIN has since been superseded by ALIN and more recently ELIN, both of which have always used the XLIN extensions. Hence the term 'XLIN' has now fallen into disuse.

This block displays whether LIN or XLIN is supported by each node on the OLIN.

Block parameters

Symbols used in this table are explained in *Table 1*.

Parameter	Function	Units	Status
Node <i>n</i> _x (<i>n</i> =0-F)	LIN instrument group address	ABCD Hex	
<i>n</i> 0	Protocol supported by node <i>n</i> 0 (LIN/XLIN)	T/F	D
<i>n</i> 1	Protocol supported by node <i>n</i> 1 (LIN/XLIN)	T/F	
<i>n</i> 2	Protocol supported by node <i>n</i> 2 (LIN/XLIN)	T/F	
<i>n</i> 3	Protocol supported by node <i>n</i> 3 (LIN/XLIN)	T/F	
<i>n</i> 4	Protocol supported by node <i>n</i> 4 (LIN/XLIN)	T/F	C
<i>n</i> 5	Protocol supported by node <i>n</i> 5 (LIN/XLIN)	T/F	
<i>n</i> 6	Protocol supported by node <i>n</i> 6 (LIN/XLIN)	T/F	
<i>n</i> 7	Protocol supported by node <i>n</i> 7 (LIN/XLIN)	T/F	
<i>n</i> 8	Protocol supported by node <i>n</i> 8 (LIN/XLIN)	T/F	B
<i>n</i> 9	Protocol supported by node <i>n</i> 9 (LIN/XLIN)	T/F	
<i>n</i> A	Protocol supported by node <i>n</i> A (LIN/XLIN)	T/F	
<i>n</i> B	Protocol supported by node <i>n</i> B (LIN/XLIN)	T/F	
<i>n</i> C	Protocol supported by node <i>n</i> C (LIN/XLIN)	T/F	A
<i>n</i> D	Protocol supported by node <i>n</i> D (LIN/XLIN)	T/F	
<i>n</i> E	Protocol supported by node <i>n</i> E (LIN/XLIN)	T/F	
<i>n</i> F	Protocol supported by node <i>n</i> F (LIN/XLIN)	T/F	
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
Combined	OR-ing of all Alarms bits	T/F	

Table 103 Block parameters

Protocols supported

The protocols currently supported by various instruments are as follows:

Instrument	Revision	Protocol
T100/T1000	Up to 3/x	LIN
T231	Up to 4/1	LIN
PCLIN	Up to 4/x	LIN
QLIN	Up to 2p0x	LIN
T221	From 1/1	XLIN
All ALIN instruments	From 1/1	XLIN

Table 104 Protocols

OPT_DIAG: OPTIONS DIAGNOSTIC BLOCK

Block function

The OPT_DIAG block provides information about 10 licensable items at a time in the order of original registration, including attributes of the system that may impose some limit to the operation, i.e. database license violations. It can display the limits of memory regions (which are NOT items that can be increased by license) as well as items (e.g. number of standard blocks) which can be increased by license.

NOTE As memory region usage cannot exceed the limit, memory region licensable items can never invoke a license violation alarm.

Block parameters

Symbols used in this table are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status*
Page	Displays 10 licensable items in order of registration	Integer	
Type1 to Type10	Type of licensable item	Menu	
Usage1 to Usage10	Quantity used	Integer	
Limit1 to Limit10	Maximum limit permitted	Integer	
Alarms			  
Software	Block RAM data sumcheck error/network failure	T/F	
Combined	OR-ing of all Alarms bits	T/F	

*Read-only unless otherwise stated

Table 105 Block parameters

Block specification menu

The following is given in addition to *Table 105*.

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

Page. Page 0 provides a view of licensable items 1 - 10, Page 1 provides a view of licensable items 11 - 20; etc. Setting page to -1 will automatically advance to display the next page which contains a license violation, but will have no effect if there is not a license violation in the system.

Typen. The type of licensable item invoked due to imposed limits or license restrictions. Used in conjunction with *Usage* and *Limit* to show details about the licensable item shown. The type a item can be divided into categories, see table below.

Type	Description
RAM_*	This shows the details in Usage and Limit related to the displayed instrument region, i.e. RAM_DB (non-volatile DB RAM) RAM_DB_VOL (volatile DB RAM), RAM_TEMP (Template space), RAM_SFC (SFC resource table), RAM_ST (SFC ST space), RAM_GWMODBUS (GW modbus tables), RAM_ALARMS (alarms). The Usage value displays the amount of RAM (in kbytes) currently used, and the Limit value shows the total amount of RAM (in kbytes) in this region.
GW_MASTR_RAW	This shows the details in Usage and Limit related to the Modbus Master License item. The Usage value numerically indicates the current state of a Modbus Master configuration (0 = Modbus Master not configured, 1 = Modbus Master is configured). The Limit value numerically indicates the instrument has a valid license allowing it to be configured as a Modbus Master (0 = Modbus Master configuration is not permitted (not licensed), 1 = Modbus Master configuration is permitted). Note. Raw Comms is Licensed as part of Modbus Master

Continued...

Type	Description
<i>Continued...</i>	
RECORDING	This shows the details in the corresponding Usage and Limit related to the Recording Licencable item. The Usage value numerically indicates the current state of a Recording configuration (0 = Recording not configured, 1 = Recording is configured). The Limit value numerically indicates the instrument has a valid license allowing it to be configured as for recording (0 = Recording configuration is not permitted (not licensed), 1 = Recording configuration is permitted).
*_BLOCKS	This shows the details in the corresponding Usage and Limit related to the displayed block license category, i.e. STD_BLOCKS (Standard), CTRL_BLOCKS (Control) or ADV_BLOCKS (Advanced). The Usage value displays the number of blocks from the Standard and Control block license category currently used, and the Limit value shows the permitted number of blocks from the Standard and Control block license category. Blocks from the Advanced block license category are licensed and therefore the Usage value numerically indicates the current use of a Advanced block (0 = Advanced blocks not used, 1 = Advanced blocks are used). The Limit value numerically indicates the instrument has a valid license allowing it to use any Advanced block (0 = use of Advanced blocks is not be permitted (not licensed), 1 = use of Advanced blocks is permitted).

Table 106 Violation description

Usagen. The current usage of the type indicated in the *Typen* field.

Limitn. The limit of the type indicated in the *Typen* field.

PBUSDIAG: PROFIBUS DIAGNOSTIC BLOCK

Block function

The PBUSDIAG block gives communications diagnostics and statistics for a given Profibus master sub-system communications line, and diagnostics for a selected node on that line.

The 6 bytes of standard Profibus slave diagnostics are reported in interpreted form, adhering as closely as possible to the specified Profibus terminology. These 6 bytes are common across Profibus DP and DPv1 slaves, no DPv1 specific diagnostics are shown. Reporting of slave device specific extended diagnostics is also supported. Additionally other relevant diagnostics are provided to aid configuration of a Profibus DP network.

Block parameters

Symbols used in this table are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Port	Profibus master port that requires diagnostics	Menu	
ThisAdd	Profibus address of selected master port	Integer	
NextSlv	Go to next slave of type according to DefnNext	T/F	
DefnNext	Slave type definition	Menu	
Hold	Prevents changes to all fields except Hold	T/F	
Options	Reset all counters	(ABCD) Hex	
FastCapt	Diagnostics captured every block update	T/F	D
SlvLatch	Status Bit flags latch once TRUE	T/F	
SlvFetch	Get diagnostics from slave via Profibus network	T/F	
		T/F	
MstCfgEr	Master card config error code	(ABCD)	
CycRdMem	Total master cyclic read use	%	
CycWrMem	Total master cyclic write use	%	
CycTime	Cyclic exchange update time	ms	
MstDiag	Master's comms diagnostics	(ABCD) Hex	
AllSlvEr	True - All assigned slaves in error (Master card)	T/F	D
AnySlvEr	True - Any assigned slaves in error (Master card)	T/F	
		T/F	
		T/F	
GblError	Global errors	(AB)CD Hex	
AllOk	Master and all assigned slaves error-free	T/F	D
MstHwEr	Fault in master card	T/F	
MstNoTkn	Master not in logical token ring	T/F	
MstParam	Problem with master parameters	T/F	
SlvNoRes	Response not received by Master from slave at GblErAdd	T/F	C
SlvNoSup	Master's request not supported by slave at GblErAdd	T/F	
SlvParam	Unsuccessful parameterise of slave at GblErAdd by Master	T/F	
Unknown	Other undefined error	T/F	
GblErAdd	Address of slave with GblError	Integer	
BusErCnt	Quantity of serious bus errors events	Integer	
RjTgmCnt	Quantity of failed Profibus telegrams (serious bus errors)	Integer	
Alarms			
Software	Block RAM data sumcheck error/network failure	T/F	
Combined	OR-ing of all Alarms bits	T/F	
SlvAdd	Slave address detailed diagnostics	Integer	
SlvError	Error flags for slave at address SlvAdd	(A)BCD Hex	
NoSlvDcm	No DCM block in database for selected slave	T/F	D
NoSlvRes	No response from slave to diagnostics request	T/F	
CfgInMst	Host comms reports Slave config error during initialisation	T/F	
CfgInCrd	Master card explicitly reports Slave config error	T/F	
Global	Global error reported for the selected slave	T/F	C
MstAdd	Slave is incorrectly assigned	T/F	
SlvIdent	Slave's Profibus ID does not match that in the GSD.	T/F	
StdStat	Slave's "standard status" diagnostics reports error	T/F	

Continued...

Parameter	Function	Units	Status
<i>Continued...</i>			
NoCycRun	Slave not exchanging cyclic data with master	T/F	B
CycOflow	Overflowed Profibus cyclic space and acyclics failed	T/F	
CycFback	Overflowed Profibus cyclic space	T/F	
SlvCfgEr	Slave config error	(ABCD) Hex	
SlvCycRd	Size (in bytes) of slave's cyclic read data	bytes	
SlvCycWr	Size (in bytes) of slave's cyclic write data	bytes	
SlvGsd	Name of GSD file used slave	Alphanumeric	
ExtDgOff	Positions 8-byte window into slave's extended diagnostics	Integer	
ExtDgCnt	Size (in bytes) of slave's extended diagnostics data	Integer	
StdStat1	Profibus-DP slave standard station status 1	CD Hex	
NotExist	Slave not found on the bus (set by master)	T/F	D
NotReady	Slave not ready	T/F	
CfgFault	Slave rejected config. from master	T/F	
ExtDiag	Slave specific extended diagnostics supported	T/F	
NotSupp	Slave does not support selected function	T/F	C
InSlvRes	Invalid response received from slave	T/F	
ParamFlt	Last parameterisation faulty (GSD fault)	T/F	
MstLock	Slave already assigned to master	T/F	
StdStat2	Profibus-DP slave standard station status 2	CD Hex	
ParamReq	Master MUST reparameterise and reconfig slave	T/F	D
StatDiag	Master must refetch slave diagnostics	T/F	
SlvDev	DP-compatible slave sets this bit	T/F	
WdogOn	Slave watchdog control enabled	T/F	
FrzeMode	Slave input and input data in Freeze Mode	T/F	C
SyncMode	Slave output and output data in Sync mode	T/F	
reserve6	Reserved by Profibus master	T/F	
Deactiv	Slave deactivated, cyclic data exchanges suspended	T/F	
StdStat3	Profibus-DP slave standard station status 3	CD Hex	
reserve0		T/F	D
reserve1		T/F	
reserve2		T/F	
reserve3		T/F	
reserve4		T/F	C
reserve5		T/F	
reserve6		T/F	
ExtDiagOv	Extended diagnostics overflow	T/F	
MstAdd	Assigned master address of selected slave	Integer	
IdentNum	Slave Profibus ident number	(ABCD) Hex	
ExtDg0_1	1st two bytes of extended diagnostics data window	ABCD Hex	
byte0_b0		T/F	D
byte0_b1		T/F	
byte0_b2		T/F	
byte0_b3		T/F	
byte0_b4		T/F	C
byte0_b5		T/F	
byte0_b6		T/F	
byte0_b7		T/F	
byte1_b0		T/F	B
byte1_b1		T/F	
byte1_b2		T/F	
byte1_b3		T/F	
byte1_b4		T/F	A
byte1_b5		T/F	
byte1_b6		T/F	
byte1_b7		T/F	
ExtDg2_3	2nd two bytes of extended diagnostics data window	ABCD Hex	
byte2_b0	See ExtDg0_1, byte0_b0 to byte1_b7	T/F	
to			
byte3_b7			
		T/F	

Continued...

Parameter	Function	Units	Status
<i>Continued...</i>			
ExtDg4_5	3rd two bytes of extended diagnostics data window	ABCD Hex	 
byte4_b0	See ExtDg0_1, byte0_b0 to byte1_b7	T/F	
to			
byte5_b7		T/F	
ExtDg6_7	Last two bytes of extended diagnostics data window	ABCD Hex	 
byte6_b0	See ExtDg0_1, byte0_b0 to byte1_b7	T/F	
to			
byte7_b7		T/F	

Table 107 Block parameters

Block specification menu

The following is given in addition to *Table 107*.

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

NextSlv. True, go to next slave of the type defined in the DefnNext field.

DefnNext. (DbErOnly/DbOkOnly/DbAllSlv/AllAdd). TRUE, reset this field, then go to next slave Profibus address matching the selected criteria.

- **DbErOnly.** Only those slaves in LIN dB with SlvError ³ 0.
- **DbOkOnly.** Only those slaves in LIN dB with SlvError = 0.
- **DbAllSlv.** All slaves in LIN dB.
- **AllAdd.** All Profibus addresses 1 – 125.

Options. (FastCapt, SlvLatch, SlvFetch). True/False fields specifying the way diagnostics will be captured.

- **FastCapt.** FALSE, captures all blocks updated diagnostics approximately once per second.

Caution

This option can impact database efficiency, but judicious use may help debug some fleeting conditions, as long as they persist for a database update.

- **SLVLatch.** TRUE, “Status Bit” flags latch once. GblError and GblErAdd lock once, GblError not = ‘AllOk’.
- FALSE, flags do not latch.
- **SLVFetch.** FALSE, diagnostics are captured directly from the Master card’s internal buffer (if available), in the slave device via the Profibus network.

Caution

This impairs comms efficiency, but is required for slaves not successfully parameterised by any master. Note it will not always be possible to obtain diagnostics for a slave successfully parameterised by another master – Profibus disallows this, at least under some circumstances.

SlvAdd. Slave node number for diagnostics to be obtained from. If set ‘All’ subsequent fields refer to this node.

SlvCfgEr. Internal software error code indicating the reason for failing to configure the slave node.

SlvCycRd & SlvCycWr. Number of bytes of cyclic read, or write data configured on the slave node.

SlvGsd. Name of the GSD file used to configure this node.

ExtDgOff. If a slave node supports extended diagnostics, this field shows the offset into the extended diagnostics data when the fields *ExtDG0_1*, *ExtDg2_3*, *ExtDg4_5* and *ExtDg6_7* start to list the data.

ExtDgCnt. Number of bytes of extended diagnostics data on this node.

StdStat1. Profibus_DP slave standard station status 1.

- **NonExist.** Set by DP-Master if the respective DP-Slave fails to communicate. The state of the last diagnostic or initial value is contained if this bit is set. DP-Slave sets this bit to 0.

- **NotReady.** Set by DP-Slave if it is not ready for data transfer.
- **CfgFault.** Set by DP-Slave if it detects differences in configuration data from the DP-Master.
- **ExtDiag.** Set by DP-Slave. If set to 1, it indicates that a diagnostic entry exists in the slave specific diagnostic area, while if set to 0 an application specific message can exist in slave specific diagnostic area, but will not be fixed.
- **NotSupp.** Set by DP-Slave if an unsupported function is requested.
- **InSlvRes.** Set by DP-Master if an implausible response is received from an addressed slave. DP-Slave sets this bit to 0.
- **ParamFlt.** Set by DP-Slave if the last parameter frame was faulty, e.g. wrong length, wrong Ident_number or invalid parameter.
- **MstLock.** Set by DP-Master (Class 1) if address is different from 255 and own address. DP-Slave sets this bit to 0.

StdStat2. Profibus_DP slave standard station status 2.

NOTE If bit 1 and bit 0 are set, bit 0 takes priority.

- **ParamReq.** Set by DP-Slave if the respective DP-Slave is to be reparameterised and reconfigured. This bit remains set until reconfiguration is complete.
- **StatDiag.** Set by DP-Slave if it is not able to provide valid data.
- **SlvDev.** This bit is set to 1 by DP-Slave.
- **WdogOn.** Set by DP-Slave if the Watchdog control has been activated.
- **FrzeMode.** Set by DP-Slave if the respective DP-Slave receives a Freeze command.
- **SyncMode.** Set by DP-Slave if the respective DP-Slave receives a Sync command.
- **DeActiv.** Set by DP-Master if the DP-Slave is marked inactive in the DP-Slave parameter set and is removed from the cyclic processing. DP-Slave sets this bit to 0.

StdStat3. Profibus_DP slave standard station status 3.

- **ExtDiagOv.** Set by respective node, DP-Master or DP-Slave, if the extended diagnostics data exceeds its diagnostic buffer.

ExtDG0_1, ExtDg2_3, ExtDg4_5 and ExtDg6_7. 2 (two) bytes, 16-bit subfields, providing Profibus extended diagnostics support.

PMC_DIAG: PROFIBUS LINE DIAGNOSTIC BLOCK

Block function

The PMC_DIAG block provides database access to the diagnostic information for a Profibus line. The Profibus system makes diagnostics available via a simple direct call that fills in a number of diagnostics structures. This call is made at update time, and the returned data is copied into the block fields. The block's *Port* parameter specifies the Profibus line for which diagnostics are being collected.

NOTE Currently there are two Profibus lines available (PROFDP_1 and PROFDP_2).

Block parameters

Symbols used in this table are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Port	Profibus port to be diagnosed (PROFDP_1/PROFDP_2)	Menu	
Status	Port status	Menu	
Go_NoGo	'Go' = port working OK; 'NoGo' = Problem	Menu	
COM	TRUE = port receiving cyclic data from • 1 configured node	T/F	
CyclRate	Actual cycle rate as % of requested cycle rate	Integer	
LkErrCnt	Count of errors on the link	Integer	
BdMsgCnt	Count of rejected messages on the bus	Integer	
MstState	Instrument's Profibus Master interface state	(AB)CD Hex	
NoError	Normal error-free state	T/F	D
NoAccess	Diagnostics inaccessible	T/F	
HwFault	Hardware fault	T/F	
SlvFault	Slave fault	T/F	
SPmFault	Invalid slave parameter configuration	T/F	C
MPmFault	Invalid master parameter configuration	T/F	
UnkFault	Unknown fault	T/F	
Spare		T/F	
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
Combined	OR-ing of all Alarms bits	T/F	
Mode	Latching mode of block bitfields	Menu	
Slave_No	Slave node no. applying to State_1, State_2, Read, Write	Integer	
State_1	Diagnostics for slave specified by Slave_No	ABCD Hex	
Configed	Master has successfully parameterised slave	T/F	D
CyclcRun	Master engaged in cyclic comms with slave	T/F	
DiagAval	Slave has diagnostics available (message sent to master)	T/F	
MstrLock	Slave is locked to this master	T/F	
ParamFlt	Fault in parameters (parameters rejected by slave/master)	T/F	C
BadResps	Invalid response from slave	T/F	
UnknwnCd	Slave does not support command	T/F	
GotExtDg	Extended diagnostics available	T/F	
ConfigFlt	Configuration fault	T/F	B
NotReady	Slave not ready	T/F	
NoResps	No/invalid response	T/F	
Deactvtd	Slave deactivated	T/F	
SynMdAct	Sync mode is active	T/F	A
FreMdAct	Freeze mode is active (slave told to freeze outputs)	T/F	
WdogAct	Watchdog is activated	T/F	
DgOnly	Only diagnostics data can be fetched from slave	T/F	
State_2	Diagnostics for slave specified by Slave_No (contd.)	(ABC)D Hex	
ParamReq	Slave requires parameterisation	T/F	D
ExDiagOv	Slave has more diagnostics than it can send	T/F	
		T/F	
		T/F	
Read	Number of bytes of read cyclic data for specified Slave_No	Integer	
Write	Number of bytes of write cyclic data for specified Slave_No	Integer	

Table 108 Block parameters

Block specification menu

The following is given in addition to *Table 108*.

Dbase, Block, Type. See *Appendix D page 544* for details of these 'header' fields.

Status. (OK/Not_Prof/Not_Init/No_Diag/NoSIDiag). Indicates the status of the Profibus port specified by the *Port* parameter.

- **OK.** Port working normally.
(The *State_1* and *State_2* parameters are valid only if *Status* is 'OK'.)
- **Not_Prof.** Not a Profibus port.
- **Not_Init.** Port is not initialised.
- **No_Diag.** Diagnostics are not available.
- **NoSIDiag.** Slave diagnostics are not available.

NOTE The block's fields are valid only if *Status* is either 'OK' or 'NoSIDiag'.

Mode. (Auto/Hold). Specifies whether or not the block's bitfields are latching. In 'Hold' mode the bitfields latch any bit values that become TRUE. This enables fleeting error conditions to be stored. Setting *Mode* to 'Auto' unlatches any existing latched bits, and prevents further latching.

PNL_DIAG: FRONT-PANEL DIAGNOSTIC BLOCK

Block function

The PNL_DIAG block gives information about the front-panel fitted to the Eycon™ 10/20 Visual Supervisor and T2900 (Industrial Strategy Engine) in which the block is running.

Block parameters

Symbols used in this table are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
State	Panel state	Menu	
MemUse	Percentage of panel memory used (%)	Eng	
Access	Access level	Integer	
StatusId	Current active agent in status pane	Integer	
MainId	Current active agent in main pane	Integer	
PopuId	Current active agent in popup pane	Integer	
DialogId	Current active agent in dialog pane	Integer	
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
Combined	OR-ing of all Alarms bits	T/F	
Active	Popup & dialog pane status	(ABCD) Hex	
PopUp	TRUE = popup currently active	T/F	D
Dialog	TRUE = dialog currently active	T/F	
Spare		T/F	
Signatur ^[1]	TRUE = signature currently active	T/F	

^[1]Applicable from Version 4.0 onwards.

Table 109 Block parameters

Block specification menu

The following is given in addition to *Table 109*.

State. (UNINIT/NONE/BARE/APP). Panel state.

PRPDIAG: PORT RESOLUTION PROTOCOL DIAGNOSTIC BLOCK

Block function

The PRPDIAG block displays the characteristics of the selected Port Resolution Protocol (PRP). One PRP is monitored at a time, selected via the *SIndex* field. PRP entries can be examined to check that the name and address are as expected and that communication is occurring.

Block parameters

Symbols used in this table are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status*
SIndex	Entry in PRP table; 1-MaxEdb (DB_DIAG block)	Integer	
Status	Status of Port Resolution Protocol	Menu	
Serial	PRP re-initialisation counter	Integer	
NetNo	Network used (multi-homed applications only)	Integer	
SecsToGo	Countdown until next 'I am' transmission	Integer	
Prot_S	Network Name (normally LIN.NET)	Menu	
Node	LIN node address	Menu	
Comment	First 8 characters of the 'I am' transmission	Alphanumeric	
IP	IP address of node selected in SIndex field	Hex	
Port	UDP port of node selected in SIndex field	Alphanumeric	
Weight	State of Processor interface	Integer	
CIndex		Integer	
Prot_C		Menu	
Inst1			
Inst2			
Alarms			  
Software	Block RAM data sumcheck error/network failure	T/F	
Combined	OR-ing of all Alarms bits	T/F	
TxTS	Timestamp of last PRP message from this node	Hex	
String1T		Menu	
String2T		Hex	
String3T		Integer	
String4T		Alphanumeric	
RxTS	Timestamp of last PRP message received	Hex	
String1R		Menu	
String2R		Hex	
String3R		Integer	
String4R		Alphanumeric	
TimeNow	Current time of node	Hex	

*Read-only unless otherwise stated

Table 110 Block parameters

Block specification menu

The following is given in addition to *Table 110*.

Prot_S. This identifies the Protocol Name. It always starts with "LIN." but the remaining characters shows the configured Protocol Name (Default - "NET").

NOTE Only LIN Nodes with an identical Protocol Name are considered to be part of the LIN Network.

IMPORTANT *If more than one user is configuring ELIN Networks on a shared Ethernet Network, a conflict of LIN Node numbers could occur if the Default Protocol name, NET, is used by everyone.*

PS_TASK: T940X TASK DIAGNOSTIC BLOCK

Block function

The PS_TASK block monitors the approximate CPU usage of each of the tasks only in the T940X (not T940) instrument, including each one of the four User Task server parameters (UserT1 to UserT4), the cached server task (CacheSrv) and the cached connections task (CacheCon). Server data is shown as milliseconds per period (server cycle time). The prioritised nature of the User Tasks should be allowed for when adjusting repeat rates (1 is the highest priority, 4 the lowest). The reported execution time for a User Task may include a period of suspension whilst higher priority tasks execute.

NOTE A 'server' is a specific process that updates control strategy blocks (e.g. synchronising cached blocks). One or more servers may be scheduled to run by each 'task', and the tasks are themselves scheduled by the runtime executive (XEC).

Block parameters

Symbols used in this table are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units*	Status*
OpSystem	CPU usage of tasks external to XEC tasks	%	
Idle	Idle time - no tasks being executed (per 1000)		
Bgnd	Bgnd task CPU usage (per 1000)		
BatLoad	BATCH/record loading task CPU usage (per 1000)		
Config	Serial Terminal Configurator task usage (per 1000)		
AMC1	Application Master Comms task 1 usage (per 1000)		
AMC2	Application Master Comms task 2 usage (per 1000)		
Load	Database load (Network initiated) task CPU usage (per 1000)		
Pr_Maint	PRP database CPU usage (per 1000)		
LLC	LIN LLC layer and MACmanager CPU usage (per 1000)		
CacheCon	Cached connections task CPU usage (per 1000)		
CacheSrv	Block server for cached blocks CPU usage (per 1000)		
UserT4	User task 4 CPU usage		
UserT3	User task 3 CPU usage		
UserT2	User task 2 CPU usage		
UserT1	User task 1 CPU usage		
ModServ	Modbus slave server task CPU usage (per 1000)		
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
Combined	OR-ing of all Alarms bits	T/F	
Mod_Rx	Modbus slave receive task CPU usage (per 1000)		
FileSync	File synchronisation task CPU usage (per 1000)		
PMC	Profibus Master Comms task usage (per 1000)		
NFS	NFS task (an ICMmanager on duplex systems) CPU usage (per 1000)		
Network	Network task CPU usage (per 1000)		
EDBserv1	External Database server1 task CPU usage (per 1000)		
EDBserv2	External Database server2 task CPU usage (per 1000)		
Pr_Rx	PRP receive buffers processing task CPU usage (per 1000)		
PRMT	PRMT task CPU usage (per 1000)		
ICM_Mgr	ICM manager task CPU usage (per 1000)		
Rx_LIN	LIN receive buffers processing task CPU usage (per 1000)		
Rx_ICM	ICM receive buffers processing task CPU usage (per 1000)		
ARCirp	ALIN interrupt processing task usage (per 1000)		
ICMirp	ICM interrupt processing task usage (per 1000)		
TickTask	Timer tick task CPU usage (per 1000)		

Table 111 Block parameter

Block specification menu

The following is given in addition to *Table 111*.

Dbase, Block, Type. See *Appendix D page 544* for details of these 'header' fields.

OpSystem. Proportion of time used by tasks not scheduled by the runtime executive (XEC).

Config. Proportion of time used by the Terminal Configurator task when accessed via the serial port.

Pr_Maint. Proportion of time used by the PRP database task.

ModSrv. Proportion of time used by the Modbus slave server task.

Mod_Rx. Proportion of time used by the Modbus slave receive task.

LLC. Proportion of time used by the Logical Link Control (LLC) task, which is responsible for the low-level network functions and the Media Access Control (MAC) manager, responsible for the Inter-processor Communications Mechanism (ICM).

ICM_Mgr. Proportion of time used by the ICM manager (Media Access Control, MAC manager), responsible for the Inter-processor Communications Mechanism (ICM).

Rx_LIN. Proportion of time used to receive LIN communications.

Rx_ICM. Proportion of time used to receive ICM communications.

ARCirp. Proportion of time used by the ALIN interrupt processing task.

ICMirp. Proportion of time used by the ICM interrupt processing task.

TickTask. Proportion of time used by the Timer tick task responsible for providing a system check.

PS_TUNE: T940 PERFORMANCE BLOCK

Block function

The PS_TUNE block monitors the performance of the three block servers in the T940 (and older versions of the T940X), Local (S2), Cached blocks' synchronisation and outgoing connections (S3), Cached blocks' incoming connections (S4), and also the proportion of time used by each task in the system. Note that a 'server' is a specific process that updates control strategy blocks (e.g. synchronising cached blocks). One or more servers may be scheduled to run by each 'task', and the tasks are themselves scheduled by the runtime executive (XEC).

Note: The PS_TUNE block should only be used for the T940 or the T940X V4 and earlier. The T940X V5 introduced multiple user tasks and therefore PS_TUNE does not give a complete task view for the T940X V5. The PS_TASK block should instead be used for the T940X V5 and above.

The left-hand half of the displayed block specification menu shows the server data. A 'period' is the server cycle time, i.e. the time between subsequent executions of the first block on the update list. It is never less than 100ms and can reach about 1500ms in big strategies. The right-hand side of the specification menu shows the task data; the proportions are expressed on a scale of 0 to 1000, i.e. x 0.1%. Tasks at the top of the list have the lowest priority and those at the bottom have the highest.

Parameter	Function	Units*	Status*
S2used	Local block server (S2) activity (ms per period)		
S2period	Last S2 cycle time (ms)		
S2delay	S2 forced suspension (ms per period)		
S3used	Cached blocks synchronisation server (S3) activity (ms per period)		
S3period	Last S3 cycle time (ms)		
S3delay	S3 forced suspension (ms per period)		
S4used	Cached blocks incoming connections server (S4) activity (ms per period)		
S4period	Last S4 cycle time (ms)		
S4delay	S4 forced suspension (ms per period)		
OpSystem	Operating System usage (per 1000)		
Idle	Idle time — no tasks being executed (per 1000)		
Bgnd	Background task usage (per 1000)		
BatLoad	BATCH/record loading task usage (per 1000)		
Config	Configurator task usage (per 1000)		
PMC1	Profibus Master Comms task 1 usage (per 1000)		
PMC2	Profibus Master Comms task 2 usage (per 1000)		
AMC1	Application Master Comms task 1 usage (per 1000)		
AMC2	Application Master Comms task 2 usage (per 1000)		
AMC3	Application Master Comms task 3 usage (per 1000)		
AMC4	Application Master Comms task 4 usage (per 1000)		
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
Combined	OR-ing of all Alarms bits	T/F	
Load	Database load task usage (per 1000)		
LLC	Logical link control task usage (per 1000)		
CacheCon	Cached connections task usage (per 1000)		
CacheSrv	Block server for cached blocks, usage (per 1000)		
NFS	Network filing system task usage (per 1000)		
LocalSrv	Block server for local blocks, usage (per 1000)		
ModServ	Modbus slave server task, usage (per 1000)		
Mod_Rx	Modbus slave receive task, usage (per 1000)		
FileSync	File synchronisation task usage (per 1000)		
Network	Network task usage (per 1000)		
PRMT	PRMT task usage (per 1000)		
Unused1	(Unused)		
Rx_ALIN	ALIN receive buffers processing task usage (per 1000)		
Rx_ICM	ICM receive buffers processing task usage (per 1000)		
ARCirp	ALIN interrupt processing task usage (per 1000)		
ICMirp	ICM interrupt processing task usage (per 1000)		
Unused2	(Unused)		
LIOleds	Local I/O LEDs display task usage (per 1000)		
TickTask	Timer tick task usage (per 1000)		

*All parameters have integer format, and read-only status at runtime.

Table 112 Block parameters

Block parameters

Symbols used in this table are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Block specification menu

The following is given in addition to *Table 112*.

Dbase, Block, Type. See *Appendix D page 544* for details of these 'header' fields.

BatLoad. Proportion of time used by the batch/record loading task.

Config. Proportion of time used by the configurator task (interface to terminal).

LLC. Proportion of time used by the Logical Link Control (LLC) task, which is responsible for the low-level network functions.

CacheCon. Proportion of time used by the server responsible for generating ALIN writes when data on connections into cached blocks changes.

CacheSrv. Proportion of time used by the cached blocks server. Because of the way redundancy management works, cached blocks are synchronised between primary and secondary T940 processors by this separate dedicated server.

NFS. Proportion of time used by the network file server task, which is responsible for handling requests for file-copying across the network.

RARCDIAG: DATA RECORD ARCHIVE DIAGNOSTIC BLOCK

Block function

This block provides monitoring and diagnostics information for archiving existing recorded data. The previously recorded data will be archived to a specified FTP Server via FTP, File Transfer Protocol.

A single RARCDIAG block provides support for both primary and secondary modules of an I/O subsystem configured for duplex operation. However, only fields associated with the primary module are relevant in an instrument configured for simplex operation, and fields associated with the secondary module remain at the default value.

Data recording. Data recording applies to the capturing and storing of process data into internal flash memory of the instrument.

Data archiving. Data archiving applies to the automatic copying of previously stored data, to ensure secure backup of process data, from the instrument to a defined archive host, typically across a network via FTP.

Block parameters

Symbols used in this table are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status				
ArcNow	On-demand archiving command	T/F	↔				
RstDiags	Reset error indications	T/F	↔				
Host1 to Host3	IP address of configured FTP Server host		📖				
PrFtpSt1 to PrFtpSt3				Integer	📖		
PrError1 to PrError3						CD Hex	📖
Connect	Connection to FTP Server failure	T/F					
Login	Login to FTP Server failure	T/F					
Dir	Archive directory error in FTP Server	T/F					
Access	Write access to FTP Server denied	T/F					
Write	Error writing to archive directory in FTP Server	T/F					
DataLost	Data archiving failure	T/F					
BadPing	Network failure	T/F					
PrArchv1 to PrArchv3	Current archiving status of Primary	Enum		📖			
ArcStrat	Current archiving strategy	Enum	📖				
Alarms			📖 📖 🗣️				
Software	Block RAM data sumcheck error/network failure	T/F					
Config	Incorrect configuration detected	T/F					
PrHst1Er	Error occurred during archiving	T/F					
PrHst2Er		T/F					
PrHst3Er		T/F					
SeHst1Er		T/F					
SeHst2Er	T/F						
SeHst3Er	T/F						
Combined	OR-ing of all Alarms bits	T/F					
Status		(AB)CD Hex	📖 📖				
Config	Incorrect configuration detected	T/F	<table border="1"> <tr><td>1</td></tr> <tr><td>2</td></tr> <tr><td>4</td></tr> <tr><td>8</td></tr> </table> D	1	2	4	8
1							
2							
4							
8							
PrHst1Er	-Error occurred during archiving	T/F					
PrHst2Er		T/F					
PrHst3Er		T/F					
SeHst1Er		T/F					
SeHst2Er	T/F	<table border="1"> <tr><td>1</td></tr> <tr><td>2</td></tr> <tr><td>4</td></tr> <tr><td>8</td></tr> </table> C	1	2	4	8	
1							
2							
4							
8							
SeHst3Er	T/F						
SeHst3Er	T/F						

Continued...

Parameter	Function	Units	Status
<i>Continued...</i>			
Reset	Reset flags	(AB)CD Hex	
PrHst1Er	-Reset specified error indication	T/F	D
PrHst2Er		T/F	
PrHst3Er		T/F	
SeHst1Er		T/F	
SeHst2Er		T/F	C
SeHst3Er		T/F	
Dir1	Path to archive directory on FTP Server	Integer	
to Dir3			
SeFtpSt1			
to SeFtpSt3	Status of Secondary module communicating with FTP Server	Integer	
SeError1			
to SeError3	Secondary module error status bits, see PrError1 to PrError3	CD Hex	
SeArchv1			
to SeArchv3	Current archiving status of Secondary	Enum	

Table 113 Block parameters

Block specification menu

The following descriptions are given in addition to *Table 113*.

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

ArcNow. Set TRUE to request an additional on-demand archive of the recorded data. This copies the recorded data to all configured FTP Servers defined by *Host1*, *Host2*, and *Host3*. It will automatically reset unless wired to.

RstDiags. Set TRUE to reset all error status bits derived from *PrError1*, *SeError1*, *PrError2*, *SeError2*, *PrError3*, and *SeError3*. It will automatically reset unless wired to.

Host1 to Host3. Shows the configured IP address, in dot notation, of the FTP Server. This identifies the FTP Server used to archive the recorded data that currently exists in the primary or secondary module. Recorded data will not be archived to any FTP Server configured at an invalid IP address, e.g. 0.0.0.0.

PrFtpSt1 to PrFtpSt3, SeFtpSt1 to SeFtpSt3 . Shows the FTP status/error code from primary module and/or secondary module following the last FTP transaction with *Host1*, *Host2*, and *Host3*, as applicable.

PrError1 to PrError3, SeError1 to SeError3. 8-bit digital outputs showing archiving errors derived from the primary or secondary module archiving to *Host1*, 2 or 3 as appropriate. These conditions remain active until reset when *RstDiags* is set TRUE. This allows the system to be left unattended knowing that any change in condition since *RstDiag* was last set TRUE will remain indicated.

- **Connect.** TRUE if an FTP connection (from primary or secondary respectively to the FTP Server defined by *Host1*, *Host2*, and/or *Host3*) could not be established. This may occur if FTP Server configured in *Host1*, *Host2*, and/or *Host3* is disabled in the FTP Server configuration.
- **Login.** TRUE if the login attempt failed. This will occur if the FTP Server rejects User Name and Password, configured in the Instrument Properties dialog. To resolve this error ensure the instrument archiving configuration, configured in the Instrument Properties dialog, and the configured FTP Server archive directory, correspond.
- **Dir.** TRUE if directory path configured in *Dir1*, *Dir2*, and/or *Dir3* is invalid. This may occur if the archive directory shown in *Dir1*, *Dir2*, and/or *Dir3* does not exist. To resolve this error ensure the instrument archiving configuration, shown in the Instrument Properties dialog, and the configured FTP Server archive directory, correspond.
- **Access.** TRUE if the archive directory defined in *Dir1*, *Dir2*, and/or *Dir3* is configured as read-only. This may occur if write access to the validated archive directory has been denied in the FTP Server.

- **Write.** TRUE if an error occurred while attempting to archive (write) the recorded data to the archive directory defined in *Dir1*, *Dir2*, and/or *Dir3*. This may occur if the FTP operation failed while writing recorded data to the configured FTP Servers defined by *Host1*, *Host2*, and/or *Host3*.
- **DataLost.** TRUE if previously recorded data has been deleted from the instrument memory before the .uhh files could be archived. This may occur if the data is written to the .uhh files quicker than the .uhh files are being archived to the configured FTP Server, *Host1*, *Host2*, and/or *Host3*. This can be caused by other archiving errors that are slowing the archive process.
- **BadPing.** TRUE if a network failure occurred when attempting to connect to the IP address of the configured FTP Servers defined by *Host1*, *Host2*, and/or *Host3*.

ArcStrat (ToAllSrvrs) Defines the current archiving strategy. **ToAllSrvrs** indicates that recorded data will be archived to all configured FTP Servers defined by *Host1*, *Host2*, and/or *Host3*.

PrArchv1 to PrArchv3, SeArchv1 to SeArchv3. (Disabled/Inactive/Active) Shows the current state of the primary or secondary module as appropriate while archiving to the configured FTP Servers defined by *Host1*, *Host2*, and/or *Host3*.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block's RAM data.
- **Config.** Asserted, when the corresponding *Status.Config* is TRUE. Indicates data recording configuration failure.
- **PrHstEr1 to PrHstEr3, SeHstEr1 to SeHstEr3.** Asserted, when the corresponding *Status.PrHstEr1* to *Status.PrHstEr3*, *Status.SeHstEr1* to *Status.SeHstEr3* is TRUE. Indicates the primary or secondary module detected an error while archiving to the corresponding FTP Server.
- **Combined.** Asserted, if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

Status. Bitfield indicating comms./hardware error conditions.

- **Config.** TRUE, if the data recording and/or archiving configuration is invalid.
- **PrHstEr1 to PrHstEr3, SeHstEr1 to SeHstEr3.** TRUE, if the primary or secondary module detected an error while archiving to the corresponding FTP Server. These conditions can be caused if any change in condition derived from *PrError1*, *SeError1*, *PrError2*, *SeError2*, *PrError3*, and *SeError3*.

Reset. Digital inputs used to reset errors conditions.

- **PrHstEr1 to PrHstEr3, SeHstEr1 to SeHstEr3.** Set TRUE to clear *Status.PrHstEr1* to *Status.PrHstEr3*, and *Status.SeHstEr1* to *Status.SeHstEr3* fields, as appropriate. This will automatically clear *Alarms.PrHstEr1* to *Alarms.PrHstEr3*, and *Alarms.SeHstEr1* to *Alarms.SeHstEr3*.

Dir1 to Dir3. Shows the configured archive destination directory path of the corresponding FTP Server, *Host1*, *Host2*, and *Host3* as applicable.

RED_CTRL: REDUNDANCY CONTROL BLOCK

Block function

The RED_CTRL block provides a means of controlling the redundancy states of a pair of CPUs operating in Duplex mode. As well as showing various diagnostic parameters, the block can trigger processor synchronisation, de-synchronisation, and primary/secondary exchange.

The management of synchronisation for two processors in the instrument(s) is handled by a dedicated task, called the Processor Redundancy Management Task (PRMT). It controls all actions between the two processors and maintains the statistics accessed by the diagnostic blocks.

NOTE Only one RED_CTRL block must be configured in a LIN Database.

Block parameters

Symbols used in this table are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Refresh	Time between block updates (0-10s)	Secs	
HW_IntLk	Hardware interlocks register	(ABC)D Hex	
PrIntlck	Software interlocks; primary CPU's view-point	ABCD Hex	
PrHWstat	Primary CPU hardware status bits	ABCD Hex	
PrSWstat	Primary CPU software status bits	ABCD Hex	
PrFilSt	Primary's file synchronisation status	Menu	
PrEdbSt	ALIN synchronisation status; primary's view-point	Menu	
PrSYNCst	Overall synchronisation status; primary's view-point	Menu	
PrMCStat	Overall operating state of primary CPU	Menu	
PrPRM_DB	Operating state of primary's database	Menu	
PrRedMod	Redundancy mode; primary CPU's view-point	Menu	
SYNC	TRUE starts synchronisation process (auto-resets)	T/F	
DeSYNC	TRUE starts de-synchronisation process (auto-resets)	T/F	
PrLINnam	Primary LIN Name	Alphanumeric	
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
PrIntlck	Primary has interlock bit(s) set	T/F	
SeIntlck	Secondary has interlock bit(s) set	T/F	
PrHard	Primary has hardware status bit(s) reset	T/F	
SeHard	Secondary has hardware status bit(s) reset	T/F	
PrSoft	Primary has software status bit(s) set	T/F	
SeSoft	Secondary has software status bit(s) set	T/F	
PrMCfail	Primary m/c operating state = FAILED	T/F	
SeMCfail	Secondary m/c operating state = FAILED	T/F	
PrSync	Primary sync status is an error state	T/F	
SeSync	Secondary sync status is an error state	T/F	
PrEdbSyn	Primary EDB (ALIN) synchronisation failed	T/F	
SeEdbSyn	Secondary EDB (ALIN) synchronisation failed	T/F	
SecWorse	Secondary processor has worse subsystem health	T/F	
Combined	OR-ing of all Alarms bits	T/F	
PwrData	Power-up data stored in realtime clock chip	(AB)CD Hex	
A_Sync		F	D
A_Primry		F	
A_Scndry		F	
A_Spare		F	
B_Sync		F	C
B_Primry		F	
B_Scndry		F	
B_Spare		F	

Continued...

Parameter	Function	Units	Status
<i>Continued...</i>			
SeIntLck	Correspond to 'Pr' parameters above for the secondary processor		
SeHWstat			
SeSWstat			
SeFilSt			
SeEdbSt			
SeSYNCst			
SeMCStat			
SePRM_DB			
SeRedMod			
Change	TRUE starts primary/secondary exchange process	T/F	<input type="checkbox"/>
DB_Path	Path/device of file specified in New_DB (unused)	Alphanumeric	
New_DB	Dbase loaded by new primary after 'Change' (unused)	Alphanumeric	
SeLINnam	Secondary LIN Name	Alphanumeric	

Table 114 Block parameters

Block specification menu

The following is given in addition to *Table 114*.

Dbase, Block, Type. See *Appendix D page 544* for details of these 'header' fields.

Refresh. This parameter can be used to lessen processor block-running overhead, and also to adjust the time you have to examine intermediate field values. To see parameter changes with less than the default 10 seconds' delay, reduce *Refresh* accordingly.

HW_IntLk. The main task of the interlocks register is to control which processor has, or is requesting, access to the SDX/IDLIC interface to the I/O modules.

PrIntLck, SeIntLck. These are the software interlocks as derived from the PRMT.

The parameter prefixes **A_** and **B_** refer to 'this processor' and 'the other processor'. The combination of these prefixes with **Pr** and **Se** allow you to determine what each processor thinks *it* is doing and what it thinks the *other* one is doing. *PrIntLck* reports the primary CPU's viewpoint, and *SeIntLck* the secondary's.

Example

- If the subfield *PrIntLck.A_IsPrim* is TRUE, then you know that the primary believes it is the primary.
- If *PrIntLck.B_IsPrim* is FALSE, then you know that the primary believes the secondary is *not* primary (as obvious as it sounds).
- By examining the field *SeIntLck* you can determine what the secondary perceives as being the truth. In the correct state you should see *SeIntLck.A_IsPrim* as FALSE, indicating that the secondary knows that it is the secondary.

PrFilSt, SeFilSt. (unSYNCd/SYNCrqst/SYNCing/in_SYNC/SYNCfail/unSYNCrq). The state of file synchronisation. For redundant control it is essential that the secondary has in its filing system (EEPROM E:) all the files that exist on the primary. These parameters indicate what point the file synchronisation process has reached.

All fields derived from the secondary processor, i.e. all fields that commence 'Se', will stop updating when the two processors are decoupled, indicated by the flashing 'Duplex LED'. These fields continue to display their last valid values when decoupled. Note that desyncing the processors via the RED_CTRL block effects a decouple, so these fields will continue to show the last valid values for the secondary prior to the desync.

PrEdbSt, SeEdbSt. (unSYNCd/SYNCing/in_SYNC/SYNCfail). LIN database protocol synchronisation state. To maintain cached connections to/from remote nodes, the secondary must duplicate all EDBs, TEATTs and FEATTs that exist on the primary. This field indicates the success or otherwise of the process of synchronising these resources.

PrSYNCst, SeSYNCst. (NotSYNCd/SyncStrt/FileSync/FileDone/DBLoadSt/DBLoadDn/DBRunSt/DBRunDn/DBSyncSt/DBSyncDn/SyncDone/SyncErr/FileErr/DBLoadEr/DBRunEr/DBSyncEr/DSyncing). These fields indicate the point reached by the overall synchronisation process. If synchronisation fails, you should examine this field to determine more specifically what failed to sync.

NOTE Owing to the reconnection mechanism already present in the LIN database protocol (for block caching), a failure to synchronise the EDB resources between the two processors is not regarded as fatal and will therefore not prevent overall synchronisation. Hence EDB sync state is not present in this field.

PrMCStat, SeMCStat. (POST/INIT/OPERATNL/FAILED). Machine status. The overall running state of the instrument.

SYNC. A TRUE input to this field is a request for the duplex pair of processors to synchronise their databases. Note that setting *SYNC* clears any entry in the *New_DB* field, whether the synchronisation command was successful or not.

NOTE For Tactician instruments, use the *PriFlags* (Primary processor) and *SecFlags* (Secondary processor) subfields of the LINMAPD block to initiate this action.

DESYNC. A TRUE input to this field is a request for the duplex pair of processors to synchronise their databases. Note that setting *DESYNC* clears any entry in the *New_DB* field, whether the synchronisation command was successful or not.

NOTE For Tactician instruments, use the *PriFlags* (Primary processor) and *SecFlags* (Secondary processor) subfields of the LINMAPD block to initiate this action.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **PrIntlck, SeIntlck.** Corresponding processor has one of the following interlock bits set: *A_FrcdPr*, *A_MajFlt*, *A_MinFlt*, *B_MajFlt*, *B_MinFlt*.
- **PrHard, SeHard.** Corresponding processor has one of the following hardware status bits reset, *PrHWStat* or *SeHWStat*: *Alin_Ok*, *ICM_Ok*, *RTC_Bat*, *Ext_Bat*. If the instrument supports the battery not fitted option, e.g. *Options.BattFit* is FALSE in the T2550 TACTICIAN header block (v4.0 and later), the *PrHWStat.Ext_Bat* and/or *SeHWStat.Ext_Bat* will be ignored.
- **PrSoft, SeSoft.** Corresponding processor has one of the following software status bits set: *SW_Fault*, *FngrFlt*.
- **PrMCfail, SeMCfail.** Corresponding processor machine operating state is FAILED.
- **PrSync, SeSync.** Corresponding processor sync status is one of: *SyncErr*, *FileErr*, *DBLoadEr*, *DBRunEr*, or *DBSyncEr*.
- **PrEdbSyn, SeEdbSyn.** Edb (LIN database protocol resource) synchronisation failed.
- **SecWorse.** TRUE indicates the secondary processor has worse subsystem health than the primary. This only applies to instruments which allow a duplex pair to remain synchronised even when the secondary has worse subsystem health compared to that of the primary (for example, T2750 V4/0 and later).

Change. A TRUE input to this parameter causes the processors to swap roles i.e. the primary processor becomes the secondary, and *vice versa*. This request can succeed only if the processors are synchronised. Note that after the swap, synchronisation is *not* automatically initiated. To re-synchronise the instrument, a TRUE input to the SYNC field (or front-panel intervention) is required.

NOTE For Tactician instruments, use the *PriFlags* (Primary processor) and *SecFlags* (Secondary processor) subfields of the LINMAPD block to initiate this action.

New_DB. (*Unused at current issue*) If a database filename is specified in this field then after a successful processor swap the new primary attempts to run this new database. This is cleared after a *SYNC* or *DESYNC* command (successful or otherwise).

RMEMDIAG: DATA RECORD MEMORY DIAGNOSTIC BLOCK

Block function

This block provides monitoring and diagnostics information concerning the flash memory in the instrument used for historical storage of recorded data. The main function of the block is to provide life expectancy information for the flash memory used to store Data Recording files, .uhh. The block will be automatically generated when the first point for data recording is defined using LINtools Data Recording Configurator.

A single RMEMDIAG block provides support for both primary and secondary modules of an I/O subsystem configured for duplex operation. However, only fields associated with the primary module are relevant in an instrument configured for simplex operation, and fields associated with the secondary module remain at the default value.

IMPORTANT Both primary and secondary modules contain an internal flash memory device, providing extremely high reliability for data recording. Flash memory slowly wears out with use, once the memory life has expired its reliability is not guaranteed, and the appropriate module(s) must be replaced.

Data recording. Data recording applies to the capturing and storing of process data into the instruments internal flash memory.

Block parameters

Symbols used in this table are explained in Table 1. Additional parameter information is given in the Block specification menu section following.

Parameter	Function	Units	Status
PrSize	Recording memory space available on Primary	Mbyte	
PrWearHi	Current wear indication (peak) on Primary	%	
PrWearAv	Current wear indication (average) on Primary	%	
SeSize	Recording memory space available on Secondary	Mbyte	
SeWearHi	Current wear indication (peak) on Secondary	%	
SeWearAv	Current wear indication (average) on Secondary	%	
Alarms			
Software	Block RAM data sumcheck error/network failure	T/F	
PrNoMem	Recording memory on Primary unavailable	T/F	
PrThresh	PrWearHi exceeds WearThr	T/F	
PrLifeEx	PrWearHi exceeds 100%	T/F	
PrMemErr	Error detected in Primary data recording memory	T/F	
SeNoMem	Recording memory on Secondary unavailable	T/F	
SeThresh	SeWearHi exceeds WearThr	T/F	
SeLifeEx	SeWearHi exceeds 100%	T/F	
SeMemErr	Error detected in Secondary data recording memory	T/F	
Combined	OR-ing of all Alarms bits	T/F	
Status		(AB)CD Hex	
PrNoMem	Recording memory on Primary unavailable	T/F - 1	D
PrThresh	PrWearHi exceeds WearThr	T/F - 2	
PrLifeEx	PrWearHi exceeds 100%	T/F - 4	
PrMemErr	Error detected in Primary data recording memory	- 8	
SeNoMem	Recording memory on Secondary unavailable	T/F - 1	C
SeThresh	SeWearHi exceeds WearThr	T/F - 2	
SeLifeEx	SeWearHi exceeds 100%	T/F - 4	
SeMemErr	Error detected in Secondary data recording memory	T/F - 8	
Reset	Reset flags	(ABC)D Hex	
PrMemErr	Reset primary memory error command		
SeMemErr	Reset secondary memory error command		
WearThr	User defined wear threshold	%	

Table 115 Block parameters

Block specification menu

The following descriptions are given in addition to *Table 115*.

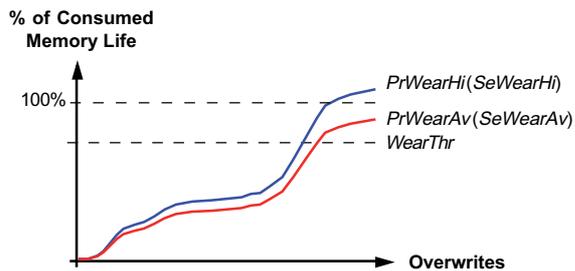
Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

PrSize, SeSize. Shows the memory available for storing recorded data history in the primary or secondary module respectively. If recording memory is unavailable this shows 0 (zero), and sets *Alarms.PrNoMem* and *Status.PrNoMem*, or *Alarms.SeNoMem* and *Status.SeNoMem* TRUE.

NOTE When this shows 0 (zero) all primary fields and secondary fields show the default values, as appropriate.

PrWearHi, SeWearHi. Shows the approximate peak level of wear that has occurred in the flash memory on the primary or secondary module respectively. This value is calculated using the device manufacturers specified life expectancy. If this value reaches 100%, data recording is not prohibited, but recorded data may be compromised. *SeWearHi* remains at the default value when the instrument is configured for simplex operation.

NOTE The amount of flash memory life remaining reduces as data is overwritten.



PrWearAv, SeWearAv. Shows the approximate average level of wear that has occurred in the flash memory on the primary or secondary module respectively. This value is calculated using the device manufacturers specified life expectancy. If this value reaches 100%, data recording is not prohibited, but recorded data may be compromised. *SeWearAv* remains at the default value when the instrument is configured for simplex operation.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.
- **PrNoMem, SeNoMem.** Asserted, if insufficient data recording memory exists in the primary and secondary module respectively. Indicates insufficient or incorrect data recording memory exists in the module.
- **PrThresh, SeThresh.** Asserted, if *PrWearHi* or *SeWearHi* on the primary or secondary respectively has exceeded 100%.
- **PrLifeEx, SeLifeEx.** Asserted, if the data recording memory in the primary and secondary module respectively, has exceeded 100% of the life expectancy limit specified by the flash memory manufacturer.

NOTE The module that has exceeded its life expectancy limit must be replaced to ensure continued reliable data recording.

- **PrMemErr, SeMemErr.** Asserted, if an error was detected when accessing the data recording memory. Indicates an error was detected when accessing the flash memory on the primary or secondary respectively.

IMPORTANT *If asserted, live data may fail to record and previously recorded data may be lost.*

- **Combined.** Asserted, if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

Status. Bitfield indicating memory/hardware error conditions.

- **PrNoMem, SeNoMem.** TRUE, when *Alarms.PrNoMem* or *Alarms.SeNoMem* is TRUE respectively. It is caused if the hardware does not contain appropriate memory, indicated by *PrSize* or *SeSize* set as 0 (zero), as appropriate. This remains TRUE until a module containing flash memory device that supports data recording is installed.

- **PrThresh, SeThresh.** TRUE, when *Alarms.PrThresh* or *Alarms.SeThresh* is TRUE respectively. Indicates the *PrWearHi* or *SeWearHi* on the primary or secondary respectively has exceeded the value defined in *WearThr*. This remains TRUE until a primary and/or secondary module with appropriate flash memory is installed, or *WearThr* is increased to exceed the value shown in *PrWearHi* or *SeWearHi*.
- **PrLifeEx, SeLifeEx.** TRUE, when *Alarms.PrLifeEx* or *Alarms.SeLifeEx* is TRUE respectively. Indicates the level of used data recording memory in the primary or secondary module, specified in *PrWearHi*, or *SeWearHi* respectively, has attained the value defined in *WearThr*.
- **PrMemErr, SeMemErr.** TRUE, when *Alarms.PrMemErr* or *Alarms.SeMemErr* is TRUE respectively. This automatically provides a Severity level entry to the Event Log file, *event_l.udz*, on the instrument via the *Alarms.EventLog* and *Status.EventLog* in the Header block. It can be caused if the primary or secondary module, as appropriate, is removed without first synchronising, desynchronising and then stopping the database, when the data recording memory was starting or operating. It should never normally be seen, because the most likely cause is a hardware failure. This remains TRUE until the *Reset.PrMemErr* or *Reset.SeMemErr* is set TRUE. This will immediately return to show TRUE if the hardware remains invalid.

Reset. Digital inputs used to reset errors indicators.

- **PrMemErr, SeMemErr.** Set TRUE to clear *Status.PrMemErr* and *Status.SeMemErr*, and *Alarms.PrMemErr* or *Alarms.SeMemErr* as appropriate.

WearThr. Defines the flash memory threshold. When this value is obtained, *Status.PrThresh* and *Alarms.PrThresh* or *Status.SeThresh* and *Alarms.SeThresh*, as appropriate, are set TRUE. The default value of 75% provides sufficient time to plan corrective action before the flash memory reaches 100% wear and potentially becomes unreliable. This applies to both primary and secondary modules if the instrument is configured for duplex operation.

ROUTETBL: ROUTING TABLE BLOCK

Block function

This block displays the routing table.

Block parameters

Symbols used in this table are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Page	Selects a group of 16 routing table entries		
SegCost n ($n=0-F$)	Routing table entry segment number, cost to get to segment		
Alarms			  
Software	Block RAM data sumcheck error / network failure		
Combined	OR-ing of all Alarms bits		
PortMAC n ($n=0-F$)	Routing table entry port number, MAC destination address		

Table 116 Block parameters

Block specification menu

The following is given in addition to *Table 116*.

SegCost0 to SegCostF. Shows the routing table entry segment number (high 8 bits) and cost to get to that segment (low 8 bits).

PortMAC0 to PortMACF. The top 4 bits are set to 1 if this is a backup entry, ie. there is another entry for the same segment which is currently in use. The next 4 bits show the routing table entry port number (0=LIN, 1=ALIN on bridge) and the bottom 8 bits give the MAC address to send messages to for this segment (FF=permanent entry).

RSRCDIAG: RESOURCE DIAGNOSTIC BLOCK

Block function

The RSRCDIAG block allows read access to any database resource. A resource is selected via the *Resource* field, and the *SubType* field selects a particular portion of that resource. The start address along with the first 32 bytes of the selected resource are displayed in the *Segment*, *Offset*, *Base_0* to *Base_1E* fields respectively. The *Timer* field lets you scroll automatically through all allocated resources of the type selected at a rate set by *Timer* (seconds).

The block also lets you explore any memory region in the instrument - by setting *Resource* to MEM_DIAG and specifying the relevant *Segment* and *Offset* values.

Block parameters

Symbols used in this table are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Resource	Selects the required resource	Menu	
SubType	Selects sub-type of selected resource	Menu	
Index	Selects resource number of selected resource	ABCD Hex	
Timer	Selects scroll rate (secs) via allocated resources (0 disables)	ABCD Hex	
Segment	Segment containing selected resource	ABCD Hex	[1]
Offset	Offset (within Segment) of selected resource	ABCD Hex	[1]
Max[2]	Max. no. of specified resources able to be allocated	ABCD Hex	
Alloc[2]	Number of specified resources currently allocated	ABCD Hex	
HWM[2]	High water mark (max. no. resources allocated since start)	ABCD Hex	
Free	Number of specified resources still available	ABCD Hex	
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
Combined	OR-ing of all Alarms bits	T/F	
Region	Memory area containing selected resource/memory	Menu	
String	First 8 bytes of resource/memory displayed as a string	Alphanumeric	
Base_0 - 1E	First 32 bytes of the selected resource (or memory region)	ABCD Hex	

[1] Read/write if Resource set to MEM_DIAG

[2] Not applicable if Resource = MEM_DIAG

Table 117 Block parameters

Block specification menu

The following is given in addition to *Table 117*.

Resource. (BLOCK/TEMPLATE/LIBRARY/ENT/EDB/LATT/FEATT/TEATT/USERT/SERT/CONNECT/MEM_DIAG) Selects the resource (or memory location) required to be investigated. The menu items are:

- **BLOCK.** Database blocks.
- **TEMPLATE.** Database templates.
- **LIBRARY.** Database libraries.
- **ENT.** (*Not implemented.*)
- **EDB.** External database connections.
- **LATT.** (*Not implemented.*)
- **FEATT.** Attachments from external database blocks.
- **TEATT.** Attachments to external database blocks.
- **USERT.** (*Not implemented.*)
- **SERT.** Servers.
- **CONNECT.** Database connections ('wires').

- **MEM_DIAG.** Allows the *Segment* and *Offset* fields to be used to specify a memory region start address for display in the *Base_0* to *Base_1E* fields. Note that *Segment* and *Offset* are normally read-only.

SubType. (RESOURCE/HEADER/DYNAMIC/LOCAL/POOL/MEM_DIAG) Selects the resource sub-type required. The menu items are:

- **RESOURCE.** The actual resource structure.
- **HEADER.** The Block/Template header.
- **DYNAMIC.** Dynamic data area of a block.
- **LOCAL.** Local data area of a block.
- **POOL.** Pool data area for Block/Template.
- **MEM_DIAG.** Selected automatically if *Resource* = 'MEM_DIAG'.

Index. Lets you select the resource *number* of the particular resource type that was selected in the *Resource* field. E.g. setting *Resource* to **BLOCK** and *Index* to **4** displays the contents of BLOCK resource number 4.

Timer. When *Timer* is set to a non-zero number, the display scrolls through all allocated resources of the type selected via *Resource* and *SubType*, at a rate of *Timer* seconds per resource. Resetting *Timer* to zero stops the scrolling process.

Segment, Offset. Together these fields display the start address of the selected resource. If *Resource* = MEM_DIAG, these fields become read/write and let you select a start address for a memory region to be displayed in the *Base_0* to *Base_1E* fields.

Region. (ROM/RAM/E2ROM/UNUSED) Shows the memory device in which the selected resource resides. When the *Resource* field has been selected as **MEM_DIAG**, *Region* shows the location of the selected memory region. The menu items are:

- **ROM.** ROM (read-only memory).
- **RAM.** RAM (random access memory).
- **E2ROM.** E2ROM (electrically erasable read-only memory).
- **UNUSED.** No memory device used.

RTB_DIAG: ROUTING TABLE DIAGNOSTIC BLOCK

Block function

This block displays diagnostic information on the routing table.

Block parameters

Symbols used in this table are explained in *Table 1*. Parameter information is given in *Table 118* following.

Parameter	Function	Units	Status
Entries	Number of routing table entries in use		
Search	Average search length for lookups		
SearchF	Number of comparisons on failed search		
Lookup	Number of lookups		
LookupF	Number of failed lookups		
Delete	Number of deletions		
DeleteF	Number of failed deletions		
Add	Number of additions		
AddF	Number of failed additions		
Garbage	Number of garbage collections		
Reclaim	Number of entries reclaimed		
Alarms			  
Software	Block RAM data sumcheck error / network failure		
Combined	OR-ing of all Alarms bits		
Searchx	Index of searches by search length		
SearchFx	Index of failed searches by search length		

Table 118 Block parameters

SD_DIAG: SD CARD DIAGNOSTIC BLOCK

Block function

The SD_DIAG block provides diagnostic information about the SD card within a T2750 instrument.

Block parameters

Symbols used in this table are explained in *Table 1*. Parameter information is given in *Table 119* following.

Parameter	Function	Units	Status
PwrOnTim	Time taken between CPU reset and firmware running	Secs	
PwrOnExp	Expected time between CPU reset and firmware running	Secs	
PwrOnTol	Permitted tolerance between <i>PwrOnTim</i> and <i>PwrOnExp</i>	Secs	
State	Status of the SD card driver	Enum	
RESET	Start-up condition indicating SD card not ready yet		
INIT	Normal condition of SD card when not being accessed		
BUSY	Normal condition of SD card during read/write activity		
REQ_SHUTDOWN	SD card error detected and transitioning to QUARANTINE		
QUARANTINE	SD card in quarantine mode and preventing further access		
UNKNOWN	SD card in an unrecognised state. SD card not accessible.		
BlocksRd	Number of blocks read from SD card since last reset		
BlocksWr	Number of blocks written to SD card since last reset		
GoodTran	Number of successful read/write transactions to SD card		
Alarms			
Software	Block data checksum error/network failure	T/F	
PwrOnTim	Time taken for firmware to load and run too long	T/F	
Combined	OR-ing of all Alarms bits	T/F	
ManfId	Manufacturer ID		
OedId	OEM/Application ID		
ProdName	(up to) 5 ASCII character string		
SerialNo	Serial Number		
ManfMnth	Month of manufacture		
ManfYear	Year of manufacture		
HwRev	Hardware revision (manufacturer specific)		
FwRev	Firmware revision (manufacturer specific)		

Table 119 Block parameters

Block specification menu

Main page

The following is given in addition to *Table 119*.

PwrOnTim. The time (in seconds) from the CPU being reset (as power applied) until the main firmware started running. Because the firmware is loaded from the SD card, this is an indirect measure of the SD card speed.

PwrOnExp. The value expected for *PwrOnTim* parameter. The recommended value is 14.0 seconds. This parameter, combined with *PwrOnTol* is used to determine whether the *PwrOnTim* alarm is asserted.

PwrOnTol. The permitted tolerance, as a absolute value of Power On Time (*PwrOnTim*) minus the Power On Expected time (*PwrOnExp*). If the tolerance is exceeded, the *PwrOnTim* alarm is asserted. The recommended value for *PwrOnTol* is 3.0 seconds.

State. Indicates the status of the SD card driver. In normal operation it should usually be INIT. Possible values are:

- **RESET.** The start-up condition. In this state, the SD card is not yet operational; it is still starting up.
- **INIT.** The normal, at rest, condition. In this state, the SD card is not currently being accessed and is ready for use.
- **BUSY.** The card is currently busy reading or writing. In this state, a client is currently in the process of reading from or writing to the card. This state is very brief and is unlikely to be observed very often when inspecting this block.

- **REQ_SHUTDN.** The card is about to enter quarantine mode. In this state, an error has been detected with the SD card and the driver is in the process of performing an orderly transfer to QUARANTINE. This state only lasts briefly.
- **QUARANTINE.** The card is in quarantine mode. In this state, the driver has detected a fault with the SD card and has entered a state where it will no longer access the card. This is to prevent possible (further) corruption of the SD card contents.
- **UNKNOWN.** The SD card driver has entered an unrecognised state. The SD card is not accessible whilst in this state.

BlocksRd and BlocksWr. The number of blocks of data read and written from/to the SD card since the last power-on/reset. The SD card data is always read/written in blocks of 512 bytes.

GoodTran. The total number of successful read and write transactions from/to the SD card since the last power-on/reset. Note that a single transaction may consist of 1 or more blocks of 512 bytes, so it is expected that $GoodTran < (BlocksRd + BlocksWr)$

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block's RAM data.
- **PwrOnTim.** TRUE indicates that the time taken for the firmware to load and run after a power-up/reset (*PwrOnTim*) has completed more slowly or more quickly than expected (*PwrOnExp*) and has exceeded the tolerance set by *PwrOnTol*.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

Additional static, read-only information is available from the following parameters giving details about the SD card.

Manfld. The manufacturer's identification.

OedID. The card's OEM/Application ID.

ProdName. Product name, up to five ASCII characters.

SerialNo. The card's serial number.

ManfMnth. Month of manufacture (1-12).

ManfYear. Year of manufacture (displayed as actual year number).

HwRev. Hardware revision number.

FwRev. Firmware revision number.

ErrorCounts page

The ErrorCounts page consists of a set of read-only fields which display a count of error types since the last power-on/reset. Most values are expected to usually be zero. The SD card driver has retry mechanisms to allow it to recover from occasional errors.

The information provided on this page is not intended for general use and is of only use to the manufacturer of the instrument and/or during a support call.

SFC_DIAG: SEQUENCE DIAGNOSTIC BLOCK

Block function

The SFC_DIAG block indicates the actual resource levels *used* by the current sequence (in the left half of the specification menu as it appears in LINtools), and also the maximum resource levels *allowed* by the current software version (in the right half).

NOTE The displayed parameter values are valid only at runtime in the target instrument.

Block parameters

Symbols used in this table are explained in *Table 1*. Parameter information is given in *Table 120* following.

Parameter	Function	Units*	Status*
SFC	Total number of SFCs (root & sub-SFCs) loaded		
ROOT	Total number of root SFCs loaded		
STEP	Total number of steps loaded		
STEPLINK	Total number of 'wires' going into and out of steps		
TRAN	Total number of transitions loaded		
TRANLINK	Total number of 'wires' going into transitions		
ASSOCIAT	Total number of action associations loaded		
ACTION	Total number of actions referred to		
EVENT	Total number of timed events currently scheduled (by active steps)		
POINT	Total number of points selected for display via LINtools connect		
SFCsize	Total amount of ST space used (bytes)		
Alarms			  
Software	Block RAM data sumcheck error / network failure	T/F	
NonSeq	Target instrument does not support SFCs	T/F	
Combined	OR-ing of all Alarms bits	T/F	
MaxSFC	Maximum number of SFCs (root & sub-SFCs) loadable		
MaxROOT	Maximum number of root SFCs loadable		
MaxSTEP	Maximum number of steps loadable		
MaxSTLK	Maximum number of 'wires' permitted going into and out of steps		
MaxTRAN	Maximum number of transitions loadable		
MaxTRLK	Maximum number of 'wires' permitted going into transitions		
MaxASSOC	Maximum number of action associations loadable		
MaxACTN	Maximum number of actions referable to		
MaxEVENT	Maximum number of timed events currently schedulable (by active steps)		
MaxPOINT	Maximum number of points selectable for display via LINtools connect		
MaxSFCsz	Maximum amount of ST space available (bytes/Kbytes) ^[1]		

*All parameters have integer format, and read-only status at runtime

[1] Shows either bytes or kbytes depending on instrument type

Table 120 Block parameters

SUM_DIAG: SUMMARY DIAGNOSTIC BLOCK

Block function

This block collects together and summarises data on network and database faults within a running instrument. Much of the displayed data is also available via other diagnostic blocks, but SUM_DIAG highlights the most important information and presents it in a simplified form. The block does some ‘expert’ analysis to yield meaningful fault indications, which help you diagnose LIN and database faults as rapidly as possible.

Block parameters

Symbols used in this table are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
InstType	Instrument type	Alphanumeric	
Version	Instrument software version	Alphanumeric	
Database	Name of currently-running database	Alphanumeric	
SegNode	Segment (AB) & node address (CD) of this node	ABCD Hex	
LastErr	Last database error code (see instrument manual)	Hex	
LRA	LIN Redundancy Algorithm status	Menu	
MAC	Media Access Control layer status	Menu	
LLC	Logical Link Control layer status	Menu	
NET	NET layer status (unused currently)	Menu	
EDB	EDB (external databases) layer status	Menu	
EDBName	Name of faulty external database (see EDB)	Alphanumeric	
EDBNode	Node no. of faulty external database (see EDB)	Hex	
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
LRA	LRA status alarm. TRUE if LRA status not ‘OK’	T/F	
MAC	MAC status alarm. TRUE if MAC status not ‘OK’	T/F	
LLC	LLC status alarm. TRUE if LLC status not ‘OK’	T/F	
NET	NET status alarm. TRUE if NET status not ‘OK’	T/F	
EDB	EDB status alarm. TRUE if EDB status not ‘OK’	T/F	
Status	Status alarm. TRUE if any status bits are TRUE	T/F	
Spare (4)	(Spare alarms)	T/F	
Combined	OR-ing of all Alarms bits	T/F	
Mode	Operating mode of database	Menu	
Status	General status information	(ABC)D Hex	
Memory	TRUE if <2% database memory free	T/F	D 1 2 4 8
ScanRate	Local block server (or User Task 1) repeat time	ms	
Spare1 to Spare7	(unused)		
Spare8	(unused)	Alphanumeric	

Table 121 Block parameters

Block specification menu

The following is given in addition to *Table 121*.

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

SegNode. Four hexadecimal digits showing the LIN segment (first two digits) and node address (last two digits) of this node.

LastErr. Four hexadecimal digits showing the code of the last database error. Refer to the specific instrument manual for a list of error codes and their meanings (for non-obvious cases).

LRA. (OK/‘N/A’/SelfTstB/SelfTstA/FaultOnB/FaultOnA). Shows the LRA (LIN Redundancy Algorithm) status. Please refer to LRA section for details of this algorithm.

- **OK.** Shown if no LRA faults currently detected.

- **N/A.** Shown if the LRA is not appropriate (ALIN instruments).
- **SelfTstB.** Shown if LRA self-test failed on LIN B and LIN B is fitted. Indicates a hardware fault in the local station.
- **SelfTstA.** Shown if LRA self-test failed on LIN A. Indicates a hardware fault in the local station.
- **FaultOnB.** Shown if LRA fault detected on LIN B and LIN B is fitted. Indicates that a message was detected on LIN A that did not appear on LIN B. May be caused by cabling or connector faults, excessive noise, or a fault in the hardware of one of the stations.
- **FaultOnA.** Shown if LRA fault detected on LIN A. Indicates that a message was detected on LIN B that did not appear on LIN A. May be caused by cabling or connector faults, excessive noise, or a fault in the hardware of one of the stations.

NOTE The menu items are in order of ascending priority, e.g. a *FaultOnA* indication would override any other status indication.

MAC. (OK/'N/A'/LoclFail/ToknPass/ToknRotn/ALINConn/LINConn/ALINOffl/LINOffl). Shows the MAC (Media Access Control) layer status. Menu items are in ascending order of priority, as for the previous section.

- **OK.** Shown if no MAC faults currently detected.
- **N/A.** Shown if MAC is not appropriate (*not currently used*).
- **LoclFail.** (*Not applicable to ALIN.*) Shown if failures in the local station's hardware are detected.
- **ToknPass.** (*Not applicable to ALIN.*) Shown if excessive problems occur when passing the token to next station on the LIN. Indicates a possible fault in the local hardware timing, the next logical station's hardware, or the cabling. This menu item should not be indicated during normal operation unless stations are plugged in and out.

ToknRotn. (*Not applicable to ALIN.*) Shown if token rotation is in trouble on the LIN, i.e. problems are detected maintaining the logical ring. Indicates that stations are appearing and disappearing from the LIN, which may be caused by nodes being plugged in and out, or by a possible cabling fault. This menu item should not be indicated during normal operation unless stations are plugged in and out.

ALINConn. Shown if the ALIN is not connected, i.e. MAC state is IDLE, but no token rotation is possible. Indicates that cabling is incorrect or broken, or that no other ALIN nodes are connected.

LINConn. Shown if the LIN is not connected, i.e. MAC state is IDLE, but no token rotation is possible. Indicates that cabling is incorrect or broken, or that no other LIN nodes are connected.

ALINOffl. Shown if the ALIN MAC state is 'offline'.

LINOffl. Shown if the LIN MAC state is 'offline', for example due to an illegal node number (00 or FF), a duplicate node number, or possibly a faulty transmitter.

LLC. (OK/'N/A'/ALINBuff/LINBuff/ALINIdle/LINIdle) Shows the LLC (Logical Link Control) layer status. Menu items are in ascending order of priority, as for the previous section.

- **OK.** Shown if no LLC faults currently detected.
- **N/A.** Shown if LLC is not appropriate (*not currently used*).
- **ALINBuff.** Shown if the ALIN LLC buffers are exhausted, i.e. *SAPsfree*, *Tx_Free* or *Rx_Free* are zero. Generally caused by attempting to connect too many EDBs.
- **LINBuff.** Shown if the LIN LLC buffers are exhausted, i.e. *SAPsfree*, *Tx_Free* or *Rx_Free* are zero. Generally caused by attempting to connect too many EDBs.
- **ALINIdle.** Shown if the ALIN LLC layer is not functioning. (This is unlikely ever to occur, and would indicate a serious fault.)
- **LINIdle.** Shown if the LIN LLC layer is not functioning. This is unlikely ever to occur.

NET. (OK/'N/A') Shows the NET layer status. Menu items are in ascending order of priority, as for the previous section.

- **OK.** Shown if no NET faults currently detected.
- **N/A.** Shown if NET is not appropriate (*not currently used*).

NOTE The *NET* field is included for future expansion and is currently unused.

EDB. (OK/'N/A'/Teatt/Featt/Verify/Disconn/Idle). Shows the EDB (External Database) layer status. *EDB* hunts through EDBs and displays fault status information on the first EDB it discovers with a problem. Menu items are in ascending order of priority, as for the previous section.

- **OK.** Shown if no EDB faults currently detected in any EDB.
 - **N/A.** Shown if EDBs are not appropriate (*not currently used*).
 - **Teatt.** Shown if at least one *Teatt* alarm is active in the EDB, i.e. local cached blocks are not being updated. Generally caused by misnamed cached blocks. The block(s) causing the *Teatt* alarm will themselves be in software alarm.
 - **Featt.** Shown if a *Featt* alarm is active, i.e. the local node is not sending cached block updates. Generally caused by a node that is caching local blocks going offline. Otherwise, the fault may be caused by a remote node requesting a block name that does not exist in this instrument. The remote node will flag a software alarm.
 - **Verify.** Shown if the EDB has not completed attachment, i.e. if the EDB state is INIT or VERIFYING. This is generally a transient state entered during connection of remote databases.
 - **Disconn.** Shown if the EDB is not connected, i.e. the EDB state is DORMANT, DISCONNECTED or DISCONNECTING. Generally caused by a remote database which does not exist or has halted. If the EDB was created by the remote node, this is a transient state terminating when the local EDB is deleted. This is a non-transient state if the remote node is disconnected or not running.
 - **Idle.** Shown if EDB layer is not functioning, i.e. the EDB state is OFFLINE. (This fault rarely occurs.)
- Mode.** (Auto/Init/Latch/Hold) Used to set and indicate the database operating mode.
- **Auto.** Normal automatic operating mode.
 - **Init.** Resets block values and returns to *Auto* mode.
 - **Latch.** Latches highest priority status of each field until cleared by setting *Mode* to *Auto*.
 - **Hold.** Freezes current values.
- Status.** Bitfield showing general status information, see *Table 121*.

NOTE Only one bit is used currently - the other 15 are reserved for future use.

ScanRate. The local block server repeat time in milliseconds, or the repeat time of User Task 1 in instruments which support this feature.

T600TUNE: T600 PERFORMANCE BLOCK

Block function

The T600TUNE block monitors the performance of the T600 Series instrument's block servers and the proportion of time used by each task in the system. Server data is shown as milliseconds per period (server cycle time). Task data is expressed on a scale of 0 to 1000, i.e. x 0.1%.

Block parameters

Symbols used in this table are explained in *Table 1*. Parameter information is given in *Table 122* following.

Parameter	Function	Units	Status
T1used	Execution time user task 1	ms	
T1period	Repeat time of user task 1	ms	
T2used	Execution time user task 2	ms	
T2period	Repeat time of user task 2	ms	
T3used	Execution time user task 3	ms	
T3period	Repeat time of user task 3	ms	
T4used	Execution time user task 4	ms	
T4period	Repeat time of user task 4	ms	
CacheUse	Execution time cache blocks	ms	
CachePer	Repeat time of cache blocks	ms	
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
Combined	OR-ing of all Alarms bits	T/F	
Bgnd	Bgnd task CPU usage	0.1%	
Scan	Scan task CPU usage	0.1%	
Modbus	Modbus task CPU usage	0.1%	
NFS	NFS task CPU usage	0.1%	
Load	Load task CPU usage	0.1%	
LLC	LLC task CPU usage	0.1%	
CacheSrv	Cache block server CPU usage	0.1%	
UserT4	User task 4 CPU usage	0.1%	
UserT3	User task 3 CPU usage	0.1%	
UserT2	User task 2 CPU usage	0.1%	
UserT1	User task 1 CPU usage	0.1%	
Fpanel	Front panel task CPU usage	0.1%	
Network	Network task CPU usage	0.1%	
BinModRx	Binary comms task CPU usage	0.1%	
RX	Network Rx task CPU usage	0.1%	

**Read-only unless otherwise stated*

Table 122 Block parameters

The prioritised nature of the user tasks, and the meanings of the 'used' and 'period' fields is shown by *Table 122*.

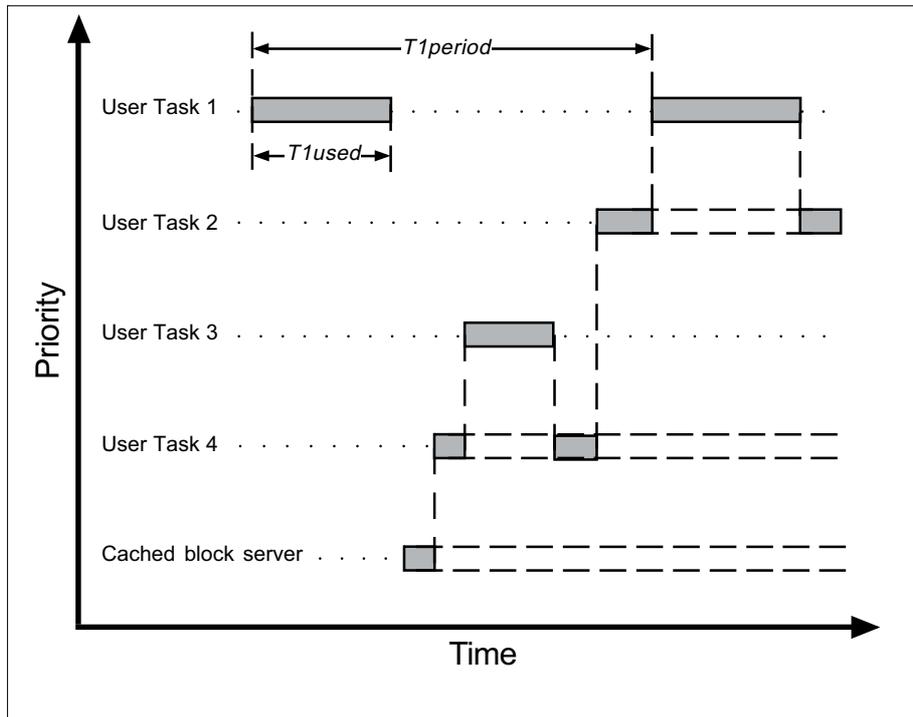


Figure 47 User task priorities

TACTTUNE: TACTICIAN TASK SUMMARY BLOCK

Block function

The TACTTUNE block monitors the approximate CPU usage of each of the tasks in the TACTICIAN instrument, including each one of the four User Task server parameters (UserT1 to UserT4), the cached server task (CacheSrv) and the cached connections task (CacheCon).

Block parameters

Symbols used in this table are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units*	Status*
Other	CPU usage of tasks external to XEC tasks	%	
Bgnd	Bgnd task CPU usage (per 1000)		
BatLoad	BATCH/record loading task CPU usage (per 1000)		
Config	Serial Terminal Configurator task usage (per 1000)		
Panel	HMI panel task CPU usage (per 1000)		
Load	Database load (Network initiated) task CPU usage (per 1000)		
Pr_Maint	PRP database CPU usage (per 1000)		
TTermcfg	Telnet Terminal Configurator task CPU usage (per 1000)		
LLC	LIN LLC layer and MACmanager CPU usage (per 1000)		
CacheCon	Cached connections task CPU usage (per 1000)		
CacheSrv	Block server for cached blocks CPU usage (per 1000)		
UserT4	User task 4 CPU usage		
UserT3	User task 3 CPU usage		
UserT2	User task 2 CPU usage		
UserT1	User task 1 CPU usage		
Spare1 to Spare4	For future development		
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
Combined	OR-ing of all Alarms bits	T/F	
ModSrv	Modbus slave server task CPU usage (per 1000)		
Mod_Rx	Modbus slave receive task CPU usage (per 1000)		
FileSync	File synchronisation task CPU usage (per 1000)		
NFS	NFS task (an ICMmanager on duplex systems) CPU usage (per 1000)		
Network	Network task CPU usage (per 1000)		
EDBserv1	External Database server1 task CPU usage (per 1000)		
EDBserv2	External Database server2 task CPU usage (per 1000)		
Pr_Rx	PRP receive buffers processing task CPU usage (per 1000)		
PRMT	PRMT task CPU usage (per 1000)		
ICM_Mgr	ICM manager task CPU usage (per 1000)		
Rx_LIN	LIN receive buffers processing task CPU usage (per 1000)		
Rx_ICM	ICM receive buffers processing task CPU usage (per 1000)		
TickTask	Timer tick task CPU usage (per 1000)		
PROFDP_1	Profibus Master gateway task CPU usage (per 1000)		
netHOST	netHOST driver task CPU usage (per 1000)		
Spare5 to Spare9	} For future development		

*All parameters have integer format, and read-only status at runtime.

Table 123 Block parameters

Block specification menu

The following is given in addition to *Table 123*.

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

Other. Proportion of time used by tasks not scheduled by the run-time executive (XEC).

Config. Proportion of time used by the Terminal Configurator task when accessed via the serial port.

Panel. Proportion of time used by the HMI panel task.

Pr_Maint. Proportion of time used by the PRP database task.

TTermcfg. Proportion of time used by the Terminal Configurator task when accessed via a Telnet session.

ModSrv. Proportion of time used by the Modbus slave server task.

Mod_Rx. Proportion of time used by the Modbus slave receive task.

LLC. Proportion of time used by the Logical Link Control (LLC) task, which is responsible for the low-level network functions and the Media Access Control (MAC) manager.

ICM_Mgr. Proportion of time used by the ICM manager (Media Access Control, MAC manager), responsible for the Inter-processor Communications Mechanism (ICM).

Rx_LIN. Proportion of time used to receive LIN communications.

Rx_ICM. Proportion of time used to receive ICM communications.

TickTask. Proportion of time used by the Timer tick task.

PROFDP_1. Proportion of time used by the Profibus Master gateway tasks.

netHOST. Proportion of time used by the netHOST driver tasks.

TOD_DIAG: TIME-OF-DAY DIAGNOSTIC BLOCK

Block function

This block provides a mechanism for controlling and monitoring network time synchronisation. The block's modes determine the operation of time synchronisation by allowing you to specify whether an instrument is to broadcast (MASTER mode) or receive (SLAVE mode) the time-of-day, or is isolated from the network's synchronisation efforts (ISOLATED mode). If the instrument supports Simple Network Time Protocol (SNTP) the synchronised time-of-day can be monitored.

NOTE To make an instrument act as a TOD slave to a cross-subnet master, configure the mapping to the cross-subnet master. Refer to *ELIN Comms User Guide* (Part No. HA082429).

Block parameters

Symbols used in this table are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Mode	Current operating mode.	Menu	
Brdcasts	Number of time broadcasts made to the network	Integer	
RcvdOk	No. of valid time broadcasts recd. from network	Integer	
RvcdRjct	No. of rejected time broadcasts/	Integer	
Alarms			  
Software	Block RAM data sumcheck error / network failure	T/F	
Own Time	Instrument not yet recd. MASTER's regular b/cast	T/F	
NoTime	This instrument's real-time clock is not set up	T/F	
Muffled	This MASTER instrument has been muffled	T/F	
Config	Configuration error	T/F	
Combined	OR-ing of all Alarms bits	T/F	
NeedTime	Node has b/cast a time request & awaits response	T/F	
ReqsIn	Number of requests received by this node	Integer	
ReqsOut	Number of requests broadcast by this node	Integer	

Table 124 Block parameters

Block specification menu

The following is given in addition to *Table 124*.

Dbase, Block, Type. See *Appendix D page 544* for details of these 'header' fields.

Mode. (MASTER/SLAVE/ISOLATED/SNTP) Current operating mode. This can also be determined by how a database was started. If this block exists in a database when it coldstarts, the mode shown in this field is adopted. If a database hotstarts the mode held in non-volatile memory is adopted. If the non-volatile memory is bad, ISOLATED mode is adopted.

NOTE If this block does not exist in the database, the mode held in non-volatile memory is adopted, irrespective of how the database is started.

- **MASTER.** When MASTER mode is selected the instrument broadcasts an initial time request. If there is no response from a lower-numbered node within a set time (currently ~2 secs.) the time is broadcast. Broadcasts are sent at regular intervals (currently ~15 mins.) and also in response to time requests. If a response to a time request or a regular time broadcast is received from a lower-numbered node, the local node becomes 'muffled' and the clock is set to the time received. When muffled, a MASTER node acts as a SLAVE node except that, if the regular time broadcast is overdue (currently by ~20 secs.), the node unmuffles and assumes MASTER status. A MASTER also muffles if its Real-Time Clock has not been set up.
- **SLAVE.** When SLAVE mode is selected the instrument broadcasts an initial time request. On receipt of a time broadcast the clock is set. Broadcasts are expected at regular intervals (~15 mins.) and a time request is broadcast if the regular time broadcast is overdue (by ~20 secs.). Time requests received in SLAVE mode are ignored.

NOTE When using ELIN, if a node is required to act as a slave to a master that exists on a different subnet, cross subnet mapping must be appropriately configured. Refer to *ELIN Comms User Guide* (Part No. HA082429).

- **ISOLATED.** The node neither broadcasts the time nor responds to time broadcasts and requests received from other nodes.

In all modes except ISOLATED, changing the time of the local clock causes a time broadcast, and receiving nodes update their clocks, unless they are in ISOLATED mode. No change of mode results.

NOTE The interval between regular broadcasts is not affected by these local clock-change broadcasts.

- **VALIDATE.** Not supported.
- **SNTP.** When SNTP mode is selected this block will display diagnostic data for SNTP server and client functionality. If SNTP server and client functionality is not configured in the Time category of the Instrument Properties dialog *Alarms.Config* is set TRUE.

Brdcasts. When *Mode* is not SNTP, this increments whenever a time update broadcast is transmitted from this node. When *Mode* is SNTP and the SNTP server is configured, this will increment for every outgoing response to a time request.

NOTE These are responses are not broadcasts.

RcvdOk. When *Mode* is not SNTP, this increments whenever an incoming broadcast is accepted and used to update the Real-Time Clock. When *Mode* is SNTP and the SNTP client is configured, this will increment whenever an incoming response to a time request is received and successfully actioned.

RcvdRjct. When *Mode* is not SNTP, this increments whenever an incoming broadcast is rejected, e.g. because we are a MASTER with higher priority than the sending MASTER. When *Mode* is SNTP and the SNTP client is configured, this will increment whenever an incoming response to a time request is an explicit rejection (not a time-out), e.g. the targeted SNTP server has not been correctly configured.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **OwnTime.** Asserted when the time has not been successfully updated within the expected period (approx. 15 mins).
- **NoTime.** Asserted when Real-Time Clock has failed when *Mode* is MASTER, or if the SNTP Server is enabled but is unable to function.
- **Muffled.** Asserted when the instrument is configured as a MASTER but has been muffled by a MASTER on a lower-numbered node, or as a result of a *NoTime* alarm (i.e. its Real-Time Clock has not been set up). See *Mode* for the definition of ‘muffled’.
- **Config.** Asserted when the method for applying time synchronisation is incorrectly configured. When *Mode* is not SNTP this may be caused by multiple TOD_DIAG blocks. This alarm will be asserted in all TOD_DIAG blocks. This may also be caused by a conflict between the *Mode* configuration of this block and the configured Time Sync category of the Instrument Properties dialog, i.e. if *Mode* is SLAVE, but the Time Synchronisation Method in the Instrument Properties dialog is SNTP. (Default Alarm Priority = 4).

NeedTime. When *Mode* is SLAVE this is set to TRUE if the time has never been updated, or has not been successfully updated within the expected period (approx. 15 mins). In ISOLATED mode it is always FALSE. In any other mode it remains unchanged. When *Mode* is SNTP this is set to TRUE if the SNTP Client is enabled, but has not been able to obtain a time update from the configured servers (Server IP 1 or Server IP 2 parameter in the Time Sync category of the Instrument Properties dialog), after 3 successive attempts with each.

ReqsIn. When *Mode* is MASTER this increments whenever a ‘request for time broadcast’ message is received from another node. This message is typically sent when a TOD SLAVE has just started up, and therefore is requesting its time to be updated as soon as possible. When *Mode* is SNTP this will increment for every incoming time request (even if SNTP Server is disabled).

ReqsOut. When *Mode* is not SNTP, this increments whenever a ‘request for time broadcast’ is transmitted from this node. When *Mode* is SNTP and the SNTP client is configured, this will increment whenever a time request is issued.

USERTASK: USER TASK DIAGNOSTIC BLOCK

Block function

The USERTASK block provides a mechanism for monitoring the task related timing information for each one of the four User Task server parameters (T1 to T4), the cached sync task (CSync) and the cached conn task (CConn), and also the proportion of time used by each task in the system. A ‘server’ is a specific process that updates strategy blocks (e.g. synchronising cached blocks). One or more servers may be scheduled to run by each ‘task’, and the tasks are themselves scheduled by the runtime executive (XEC).

NOTE Instruments that support this block, and the T640, use multiple block servers. This offers a 5ms xec Tick. Instruments that do not support this block use a single block server, and run a 4ms xec Tick.

The left-hand half of the displayed block specification menu shows the server data. A ‘period’ is the server cycle time, i.e. the time between subsequent executions of the first block on the update list. It is never less than 100ms and can reach about 1500ms in big strategies. The right-hand side of the specification menu shows the task data; the proportions are expressed on a scale of 0 to 1000, i.e. x 0.1%. Tasks at the top of the list have the lowest priority and those at the bottom have the highest.

Block parameters

Symbols used in this table are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status*
T1used	Duration of User Task 1	ms	
T1period	Total time between starting User Task 1	ms	
T2used	Duration of User Task 2	ms	
T2period	Total time between starting User Task 2	ms	
T3used	Duration of User Task 3	ms	
T3period	Total time between starting User Task 3	ms	
T4used	Duration of User Task 4	ms	
T4period	Total time between starting User Task 4	ms	
CSyncUse	Duration of cached sync task	ms	
CSyncPer	Total time between starting cached sync task	ms	
CConnUse	Duration of cached conn task	ms	
CConnPer	Total time between starting cached conn task	ms	
Alarms			  
Software	Block RAM data sumcheck error / network failure	T/F	
Combined	OR-ing of all Alarms bits	T/F	
Stretch	Server period values relative to the min. repeat values		
LastScan	Duration of previous database scan - not task	Secs	
ThisScan	Time elapsed of current database scan	Secs	
Suspend1	User Task 1 suspended using TaskHalt	F	
Suspend2	User Task 2 suspended using TaskHalt	F	
Suspend3	User Task 3 suspended using TaskHalt	F	
Suspend4	User Task 4 suspended using TaskHalt	F	

*All parameters have read-only status.

Table 125 Block parameters

Block specification menu

The following is given in addition to *Table 125*.

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

Stretch. This field is the value derived by the User Task Tuning algorithm. It represents how much the server period values have increased relative to their minimum repeat values as set in the header block (or derived as defaults if no values were set in the header block). This value is chosen to ensure that all system tasks get adequate CPU usage. A value of less than 1 indicates that the configured server period values may be able to be reduced without the task tuning extending them. (Values of *Stretch* less than 1 are only indicated on version 7.2 or later of the T2550, and on all versions of the T2750).

LastScan. This field gives the duration (in seconds) of the last ‘database scan’. The database scan is performed by the background task, and principally performs the block sumchecking. It is derived from `srv_ref_scan`, which is the duration of the last scan in xec ticks.

ThisScan. This field gives the duration (so far) of the current database scan. It is derived from `srv_ref_time`, which is the raw xec timestamp of the start of the current scan.

NOTE Stretch, LastScan and ThisScan are extremely useful when testing the effectiveness of the User Task Tuning algorithm if the instrument.

Suspendn. These fields indicate if the relative User Task has been suspended via the *TaskHalt* of the header block.

XEC_DIAG: TASK DIAGNOSTIC BLOCK

Block function

The XEC_DIAG block displays information on the tasks within the instrument's operating system (XEC).

Block parameters

Symbols used in this table are explained in *Table 1*. Parameter information is given in *Table 126* following.

Parameter	Function	Units	Status
LocTUID	Selects the task		
CurrPrty	Current task priority		
BasePrty	Base task priority		
TCB_Lock	Task locks		
TUID	Task ID		
SuspMsk	Task suspend mask		
TSuspMsk	Timed task suspend mask		
PendMsk	Pending wakeups		
AlrmTime	Next wakeup time		
CPUTime	CPU usage counter		
WDCount	Watchdog counter		
Alarms			  
Software	Block RAM data sumcheck error / network failure		
StackErr	Stack overflow alarm		
Combined	OR-ing of all Alarms bits		
CurrFree	Current free stack		
MinFree	Minimum free stack		

Table 126 Block parameters

CHAPTER 10 HISTORIAN FUNCTION BLOCKS

This block is used only with the legacy instrument T1000. For current instruments refer to *Chapter 17*, 'RECORDER' for further information.

The HISTORIAN category of Function Block Templates provides the control strategy with functions for collecting, filtering, processing, compressing and recording analogue or digital values for use on a runtime Trend page.

Refer to the LIN Block Reference Manual HA082375U003 Issue 15 (Vintage) for details.

CHAPTER 11 I/O FUNCTION BLOCKS

The I/O category of Function Block Templates provides the Strategy with functions for defining the connection from the Strategy to the real-time input/output hardware. Typically, this can be Analogue In and Out, Digital In and Out, Thermocouple, Frequency, etc.

Refer to the LIN Block Reference Manual HA082375U003 Issue 15 (Vintage) for details of blocks that are not described here.

AN_IP: ANALOGUE INPUT BLOCK

Block function

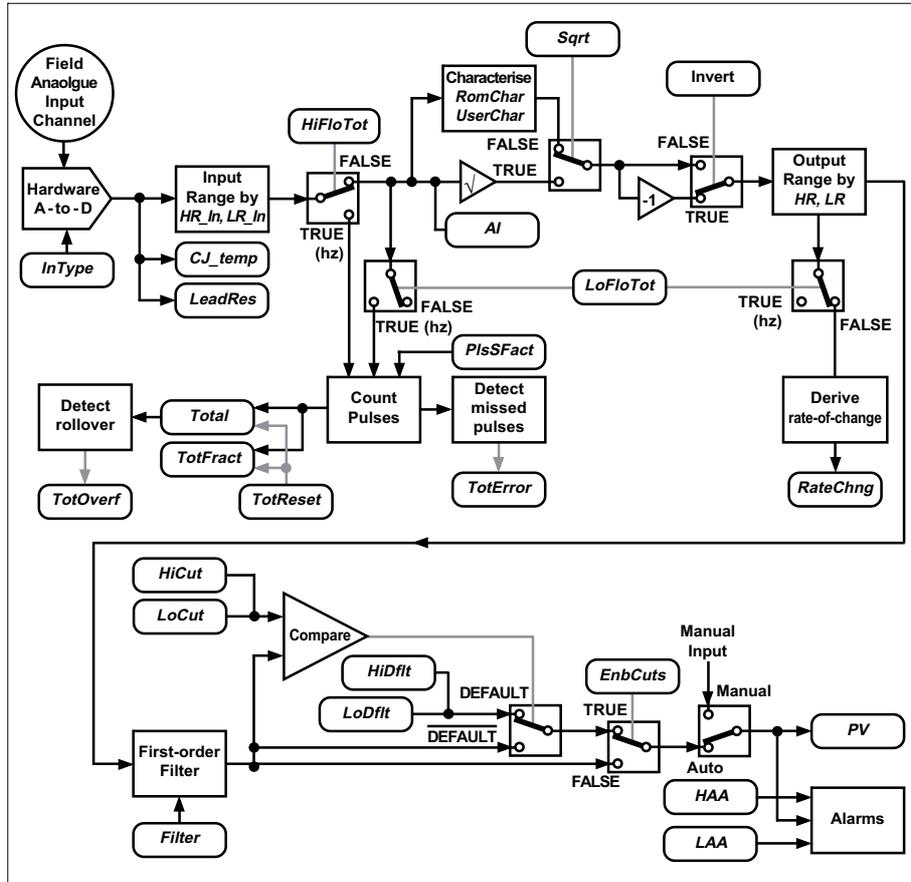


Figure 48 Block schematic

Please refer to the schematic in *Figure 48*. The AN_IP block converts voltage, frequency, or resistance measurement inputs to T600 Series instruments into floating point values in engineering units. In addition the block provides signal conditioning, calibration, filtering and alarm capability.

The LIN function block can be used in a Manual mode where the process variable *PV* can be forced to a specified value for testing purposes. Normal operation of *PV* resumes when the block is in Auto mode.

Pulse totalisation

With the frequency function selected (*InType* = Hz) and either *LoFloTot* or *HiFloTot* options TRUE the block counts, scales, and totalises input pulses. Setting *LoFloTot* enables pulse-rates up to 1kHz to be accurately totalised and also provides unprocessed and processed frequency measurement outputs (*AI* and *PV*, respectively). With *HiFloTot* set, frequency outputs are not provided but the block can handle pulse-rates up to around 30kHz. Setting both option bits FALSE disables pulse counting and allows frequency and rate-of-change of frequency measurements to be made. (Refer to *Figure 48*.)

Open-circuit detection /protection

Some I/O modules support break detection on their analogue inputs, while others support only break protection. Please refer to specific I/O module documentation for details. The AN_IP block can operate in either break detection mode, or break protection mode, but not both at the same time. Making *Options.BrkDetct* TRUE selects break detection; making it FALSE selects break protection. If you try to operate a break mode which is not supported by the particular I/O channel, the *Status.BadBrk* bit is asserted.

Break detection. Modules supporting break detection can detect a break in the input before any bad readings have been made. As soon as this is detected the *BrkDtctd* status bit is asserted. The *OCctdel* alarm bit is also asserted, subject to the delay specified in the *Delay* field. Delay can be applied either at the start of the break (i.e. the alarm bit going TRUE is delayed), or at the end of the break (i.e. the alarm bit going FALSE is delayed), or both, using the *Options.OCdelSt* and *OCdelEnd* bits, respectively.

The *Options.InitFilt* bit, when TRUE, causes the block’s first-order filter to be initialised whilst *OCctdel* or *BrkDtctd* is TRUE. This means that already-filtered bad readings do not corrupt the subsequent good readings. Note that the *OCctdel* alarm does not need to be enabled for *InitFilt* to act.

The effect on *PV* when a break is detected depends on the setting of two *Options* bits. *HoldDect*, if TRUE, causes the last valid value of *PV* to be held. If FALSE, *PV* moves either to *HR* (if *BreakUp* is TRUE) or to *LR* (*BreakUp* FALSE). These movements are reversed when the *Options.Invert* bit is TRUE.

Caution

A 0.5sec delay may occur before a break is detected, which means that the value of *PV* held may differ from the correct value.

Break protection. In I/O modules supporting only break protection, when a break occurs the value of *PV* drifts either up (*BreakUp* TRUE) or down (*BreakUp* FALSE). The direction is reversed if the *Options.Invert* bit is TRUE.

Execution of input blocks

Plant input blocks are normally executed at the start of a user task, before other blocks in the task. Because of this, connections into an input block from other blocks in the same user task (e.g. feedback from an output block) transmit the values resulting from the *previous* execution of the user task. Input connections from other user tasks, in this or other nodes, are always executed before the current user task is run, and are guaranteed to result from a completed execution of the source user task.

Block parameters

Symbols used in *Table 127* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
MODE	Operating mode	Menu	
PV	Processed input	Eng	
RateChng	Rate of change of unfiltered PV value	Eng/Sec	
HR, LR	High & low range for PV	Eng	
HAA, LAA	High & low absolute alarm limits	Eng	
Filter	First order filter time constant	Secs	
HiCut	High cutoff threshold	Eng	
HiDflt	Default PV-value when PV > HiCut	Eng	
LoCut	Low cutoff threshold	Eng	
LoDflt	Default PV-value when PV < LoCut	Eng	
RomChar	Fixed characterisation	Menu	
UserChar	User characterisation	Block name	
Alarms			
Software	Block RAM data sumcheck error/network failure	T/F	
Hardware	Fault with I/O hardware	T/F	
HiLevel	In high alarm	T/F	
LoLevel	In low alarm	T/F	
OutRange	PV outside LR-HR range (>5%)	T/F	
OCctdel	OpenCct delayed	T/F	
UCharErr	S/C error in used user char	T/F	
Combined	OR-ing of all Alarms bits	T/F	
SiteNo	Linked I/O module’s T600 ISB address	1-8	
Channel	Specifies channel number within I/O module	1-n	
InType	Specifies input type	Menu	
HR_in, LR_in	Plant signal HR, LR equivalents		
AI	Unprocessed input		
CJ_temp	Cold junction temperature value		
LeadRes	Lead resistance	Ohms	

Continued...

Parameter	Function	Units	Status
<i>Continued...</i>			
Delay	Delay before OCctdel alarm trips	Secs	
Options		ABCD hex	 
Invert	Input signal inverter	T/F	D
Sqrt	Square root input signal	T/F	
InitFilt	Initialise filter whilst OCctdel OR BrkDtctd TRUE	T/F	
OCdelSt	Apply O/C delay at start	T/F	
OCdelEnd	Apply O/C delay at end	T/F	C
BrkDetct	Use break detection	T/F	
BreakUp	PV to HR (TRUE), or LR (FALSE) on break	T/F	
TAbsolut	Temperature base (K or R)	T/F	
DsblFilt	Disables low-level filter in I/O module	T/F	B
HoldDect	On break: TRUE, PV holds; FALSE, PV to range	T/F	
EnbCuts	Enables high & low cutoff defaults	T/F	
LoFloTot	Enables totalising (<1000Hz)	T/F	
HiFloTot	Enables totalising (>1000Hz)	T/F	A
TotReset	Set Total.TotFract to 0.0	T/F	
TotOverf	Total has overflowed (rollover)	T/F	
Status		ABCD hex	 
Reset	I/O module has reset	T/F	D
Missing	No I/O module at site	T/F	
CommsErr	I/O module comms fail	T/F	
BadType	No such channel on I/O module	T/F	
TotError	Missed input pulses (invalidates Total)	T/F	C
BadCal	Corrupted calibration data	T/F	
PSUShort	Transmitter PSU overload	T/F	
BadRef	Internal reference fault	T/F	
BlockErr	Block data corruption (insufficient RAM)	T/F	B
BrkDtctd	TRUE if open-cct. or out-of-range detected	T/F	
BadBrk	'Break' configuration unsupported by I/O h/ware	T/F	
BadRange	Bad range selected	T/F	
BadSetup	Setup invalid for connected I/O module	T/F	A
HwFault	Fault in I/O module	T/F	
TmStmpOv	Time stamp overflow (invalidates RateChng)	T/F	
BrkWarn	RTD partial break	T/F	
Total	Integral part of scaled pulse count	Integer	
TotFract	Fractional part of scaled pulse count		
PlsSFact	Pulse scale factor (divisor)		

Table 127 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these 'header' fields.

Task. This field (default value = 0) specifies the user task the block is allocated to run in (1 - 4). Note that certain normally read-write parameters in this block cannot be configured unless *Task* has first been set to a non-zero value.

MODE. (AUTO/MANUAL/CALIB). Current operating mode. MANUAL allows *PV* to be forced to a specified value at runtime. CALIB (calibration) mode cannot be directly selected, but is 'forced' when an AI_CALIB block attaches to the AN_IP block. CALIB mode is very similar to AUTO, except that all fields that can affect the derivation of *AI* become read-only (*InType*, *SiteNo*, *Channel*, *HR_in*, *LR_in*). Please refer to the AI_CALIB block.

PV. Process variable (processed input), i.e. block output. Read-only in Auto mode.

RateChng. Rate of change of the unfiltered *PV* value (dPV/dt) calculated between the sample that was used by the previous execution of this task, and the sample being used by the current execution of the task. dPV/dt is particularly useful in loss-in-weight control applications. However, if *Status.TmStmpOv*, is TRUE, dPV/dt cannot be derived and *RateChng* holds its last good value.

NOTE. Set *Options.DsblFilt* TRUE when using *RateChng*. This disables the low-level filter in the A-to-D hardware (not shown in *Figure 49*).

HR, LR. High and low range of *PV* in engineering units. *HR* and *LR* define two points on a linear engineering units scale that normally map to points *HR_in* and *LR_in*, respectively, on a linear (millivolts) scale. This pair of scales is used by the block to derive output *PV* from the input measured signal.

The equivalent conversion equation is:

$$PV = LR + \left[\frac{\text{Input} - LR_{in}}{HR_{in} - LR_{in}} \times (HR - LR) \right]$$

NOTE. The *Options.Invert* bit affects the derivation of *PV*. See *Options*.

HAA, LAA. High and low absolute alarm limits applied to *PV*, with built-in 0.5% hysteresis as the alarm state is left. If a limit is crossed, the appropriate alarm bit (*HiLevel* or *LoLevel*) is asserted.

Filter. Specifies the time constant (0-250 seconds) of a simple first-order filter applied to the value in *PV*. Set *Filter* equal to zero to disable the filter.

HiCut. Specifies a high cutoff value for *PV*, whereupon *PV* sets to *HiDflt* (provided *Options.EnbCuts* is TRUE).

HiDflt. Default value adopted by *PV* when its calculated value is more than *HiCut*. See under *HiCut*.

LoCut. Specifies a low cutoff value for *PV*, whereupon *PV* sets to *LoDflt* (provided *Options.EnbCuts* is TRUE).

LoDflt. Default value adopted by *PV* when its calculated value is less than *LoCut*. See under *LoCut*.

RomChar. (None/J/K/T/S/R/E/B/N/W/W5/W3/MoRe/PRT/CU10). The value in *AI* can be processed by one of three mutually exclusive characterising processes, selected via *RomChar*, *UserChar* and *Sqrt*. *RomChar* selects a characterisation type from the ROM library. See *Table 127* for details. The AN_IP block automatically adjusts the ROM tables to suit the selected temperature base of the block.

NOTE To avoid conflict, specifying a value in *RomChar* other than 'None' automatically deletes any entry in the *UserChar* field. Similarly, writing a tagname in *UserChar* deletes any *RomChar* entry.

UserChar. Specifies the name of a UCHAR block defining a 16-point (*x,y*) custom linearisation/characterisation table to be applied to *AI*. (See note above.) The AN_IP block applies no temperature base correction to the UCHAR table, so it must be defined in units that match the prevailing temperature base.

NOTE A sumcheck error detected in the UCHAR block trips the AN_IP block's *Alarms.UCharErr* bit

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block's RAM data.
- **Hardware.** A hardware alarm is generated when any of the bits in the *Status* parameter are set, e.g. in the event of a fault in the I/O hardware.
- **HiLevel, LoLevel.** These alarm limits are set by parameters *HAA* and *LAA*. A 0.5% hysteresis bandwidth applied on leaving each alarm permits clean transitions into and out of the alarm condition.
- **OutOfRange.** This alarm is tripped as soon as *PV* reaches a value of 5% above or below *HR* or *LR* respectively. There is a 0.5% hysteresis band on each range value.
- **OCctdel.** 'Open-circuit delay'. With *OCdelSt* (Options) TRUE, this alarm trips after a delay specified by the *Delay* parameter, if an open-circuit is detected in the block input from the field. With *OCdelSt* FALSE, the alarm trips without delay. Similarly, with *Options.OCdelEnd* TRUE, *OCctdel* resets *Delay* seconds after the open-circuit input has been restored. With *OCdelEnd* FALSE, the alarm resets immediately. (See *Delay*.)

NOTE. Use of burden resistors may affect open-circuit detection. Refer to specific instrument product manual for details.

- **UCharErr.** This alarm is tripped if a sumcheck error is detected in the user characterisation file specified in the *UserChar* parameter.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

SiteNo. Specifies the address (1 - 8) on the T600 Series instrument's internal serial bus where the I/O module corresponding to this AN_IP block is located.

NOTE An invalid *SiteNo* sets the *Missing* status bit, and also that you cannot configure *SiteNo* if the *Task* parameter is still at its default value of zero.

Channel. Defines the channel number (1 - *n*) within the I/O module specified by the *SiteNo* parameter, to which this AN_IP block is attached. The number of channels available (*n*) depends on the I/O module type.

NOTE An invalid *Channel* sets the *Status.BadType* bit, and configuration of the *Channel* is not permitted if the *Task* parameter is still at its default value of zero.

InType. (Volts/mV_Int/mV_Ext/4 WIRE/3 WIRE/2 WIRE/Hz) Specifies input type. *Volts* configures the block as a high-level voltage input with maximum range of -10 to +10V. For millivolt inputs, *mV_Int* and *mV_Ext* select internal and external cold-junction compensation, respectively. For direct millivolt measurements, specify external CJC and set the *CJ_temp* parameter to zero. The maximum mV range is -100 to +100mV. The options *4 WIRE*, *3 WIRE*, and *2 WIRE*, support 4-, 3-, and 2-wire resistance measurements (ohms), respectively, with a maximum range of 0 to 1000Ω. *Hz* selects frequency input mode with a maximum range of 0 to 30kHz.

The input ranges given above are LIN function block maxima only - you should consult the I/O module documentation for specific limits.

NOTE Where certain options are not available on certain channels, the *Status.BadSetup* bit is set.

HR_in, LR_in. High & low range of the measured input signal, in units appropriate to the currently selected *InType* (V, mV, Ω, Hz). *HR_in* and *LR_in* define two points on a linear scale that map to points *HR* and *LR*, respectively, on a linear engineering units *PV* scale. Nominally, when the real input is at the value in *HR_in*, *PV* equals *HR*; when at *LR_in*, *PV* equals *LR*. Please refer to the section on *HR, LR* above for details.

NOTE Whenever *InType* is set to Volts, or when no characterisation is being applied, *HR_in* and *LR_in* can be set independently of *HR* and *LR*. In all other cases the block automatically derives *HR_in* and *LR_in* from *HR* and *LR* via the characterisation. The block compares the entered values of *HR_in* and *LR_in* with the list of available hardware ranges supported by the I/O module, and chooses the ‘best fit’ range to maximise the resolution of the input. Scaling to the exact values of *HR_in* and *LR_in* is done in software.

AI. The unprocessed input value, from which *PV* is derived.

NOTE The block normally derives *AI* from the average of the last four input samples. This averaging, together with the internal filtering provided by some I/O modules, yields an increased resolution but a reduced response speed. If speed of response is more important, both the I/O module’s hardware filter and the AN_IP block’s input-averaging can be disabled by setting *Options.DsblFilt* TRUE.

CJ_temp. Cold junction temperature, only valid in the ‘mV’ input modes. With *InType* set to *mV_Int*, *CJ_temp* is read-only and reports the temperature of the I/O module connector block as measured by its inbuilt sensor (if present). With *mV_Ext* selected, you can write a known cold junction temperature in *CJ_temp*, or input a value via a connection from the Strategy. Making *CJ_temp* zero disables cold-junction compensation and allows the block to function as a pure mV input.

NOTE All temperatures are reported, and must be entered, in the units jointly specified by the *TAbsolut* parameter, and the corresponding T600 block’s *IP_type* parameter. That is: °C, °F, K, or R.

NOTE Where certain options are not available on certain channels, the *Status.BadSetup* bit sets.

LeadRes. Specifies lead resistance value (Ω) for 2-wire measurements only. In 2-wire mode, *LeadRes* is read-write and a previously-measured value must be written to this field. For RTDs being used in 3- or 4-wire modes, *LeadRes* is read-only and is back-calculated by the block. Hence, should a ‘partial break’ to 2-wire operation occur, the input continues to function (and *Status.BrkWarn* is asserted).

Delay. With *OCdelSt* TRUE, *Delay* specifies the delay (secs) before the *Alarms.OCctdel* bit is set after detection of an open-circuit in the field input, and also, with *OCdelEnd* TRUE, the delay before the alarm is reset after restoration of the input. Making *OCdelSt* and/or *OCdelEnd* FALSE cancels these delays independently. Figure 49 shows schematically how *Delay* is implemented. The Delay facility is useful when short-term interruptions of the field input are to be ignored as ‘noise’, and also when restoration of an open-circuit input is liable to be ‘noisy’.

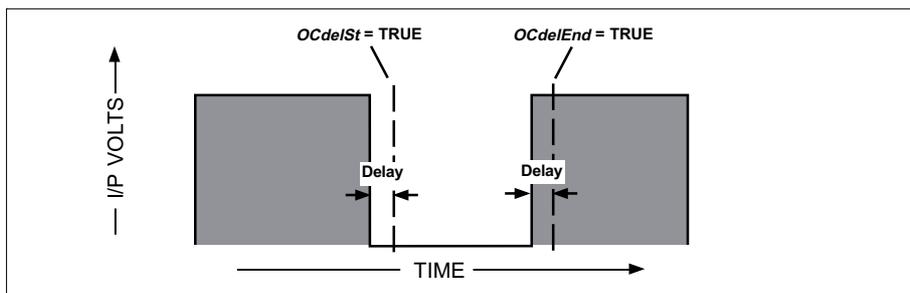


Figure 49 Delay parameter implementation (schematic)

Options. Bitfield selecting a variety of signal-processing options. The following information is given in addition to the summary in *Table 127*.

- **Invert.** If TRUE, this has the effect of mapping *HR* to *LR_in*, and *LR* to *HR_in*, i.e. of inverting the input signal, see *HR*, *LR* section.
- **Sqrt.** If TRUE, square root characterisation is applied to *AI*, i.e.

$$\text{Result} = \sqrt{\frac{AI - LR_{in}}{HR_{in} - LR_{in}}} \times (HR_{in} - LR_{in}) + LR_{in}$$

- **TAbsolut.** Used in conjunction with *IP_type.Imperial* field of the associated T600 block to select the AN_IP block's prevailing temperature units. The T600 block globally selects the Imperial (°F/R) or SI (°C/K) system, but each individual AN_IP block can specify either absolute (K/R) or relative (°C/°F) temperature units within that system. The temperature units apply to *CJ_temp*, and to any *RomChar* characterisation (i.e. the fixed tables). No temperature base corrections are applied to user-specified characterisations in UCHAR blocks.
- **DsblFilt.** TRUE disables the low-level filter in the I/O module's A-to-D hardware, this is not the AN_IP block's first-order filter. The characteristics of I/O modules vary and not all analogue inputs support low-level filters. Please refer to the specific module documentation for details.
- **LoFloTot.** Selects 'low' flow-rate totalisation, involving pulse input rates not exceeding 1kHz. Valid only when *InType* = Hz. When TRUE, the block counts input pulses, scales the count by *PlsSFact*, and stores the running total in *Total* plus *TotFract*. *AI* and *PV* report valid frequency values, but *RateChng* is disabled.
- **HiFloTot.** Selects 'high' flow-rate totalisation, involving pulse input rates from 1kHz to around 30kHz. Valid only when *InType* = Hz. When TRUE, the block counts and totalises input pulses as for *LoFloTot*, but *AI*, *PV*, *RateChng*, and the rest of the block functions are disabled. Refer to the schematic in *Figure 48*. Note that if TRUE, *HiFloTot* overrides *LoFloTot*.
- **TotOverf.** Latches TRUE if Total 'rolls over' after reaching 99 999 999.

Status. Bitfield indicating general comms./hardware error conditions. The following information is given in addition to the summary in *Table 127*.

- **TotError.** Asserts during periods when the block detects that input pulses are being missed, and hence *Total* is not incrementing correctly. Note that the *TotError* flag is non-latching and must therefore be captured by the Strategy if required.
- **BlockErr.** Sets if there is insufficient RAM left to run the block. The block is then not updated, and other blocks must be deleted to free up RAM.

Total. Integral part of the scaled total pulse count.

TotFract. Fractional part of the scaled total pulse count.

PlsSFact. Pulse scale factor. Number by which the input pulse frequency (Hz) is divided to produce the scaled pulse count, *Total* plus *TotFract*. E.g. 1000 pulses input per second with *PlsSFact* = 100.000 causes *Total* to be incremented by 10 counts per second.

AN_OUT: ANALOGUE OUTPUT BLOCK

Block function

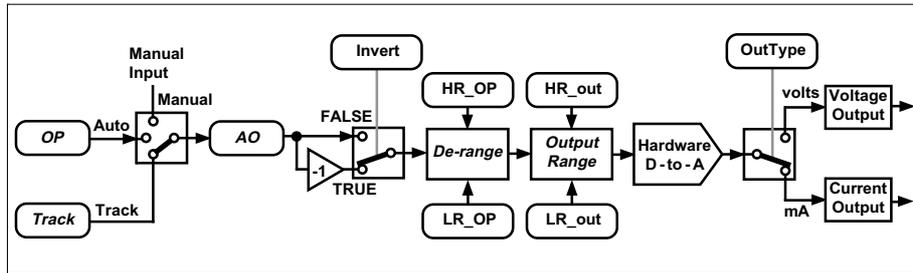


Figure 50 Block schematic

Please refer to *Figure 50*. The AN_OUT analogue output block converts ranged analogue variables (floating point numbers) from a Strategy running in a T600 Series controller into voltage or current outputs (depending on the capabilities of the associated I/O hardware). The block provides Auto/Manual/Track control, Inversion and Alarms.

Outputs from different instruments can be cross-coupled in redundant configurations to enhance the level of systems integrity. In such configurations a ‘master’ output normally controls the plant; if the master instrument fails, a ‘slave’ automatically takes over. Please refer to the specific hardware documentation for details.

Block parameters

Symbols used in *Table 128* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

Task. This field (default value = 0) specifies the user task the block is allocated to run in (1 - 4). Certain normally read-write parameters in this block cannot be configured unless *Task* has first been set to a non-zero value.

Mode. (AUTO/MANUAL/TRACK/CALIB). Current operating mode. AUTO is the normal operating mode of the block. MANUAL allows *AO* to be forced to a specified value at runtime. In TRACK mode (*SelTrack* TRUE), *AO* tracks the value in the *Track* parameter. CALIB (calibration) mode cannot be directly selected, but is ‘forced’ when an *AO_CALIB* block attaches to the AN_OUT block. CALIB mode is very similar to AUTO, except that all fields that can affect the derivation of *AO* become read-only (*Mode, SelTrack, Invert, OutType, SiteNo, Channel, HR_out, LR_out*), see *AO_CALIB* block.

Parameter	Function	Units	Status
Mode	Operating mode	Menu	
FallBack	Fallback operating mode	Menu	
OP	Requested output	Eng	
HR_OP, LR_OP	High and low range for PV	Eng	
AO	Actual output	Eng	
Track	Track output value	Eng	
SelTrack	TRUE selects Track mode (AO = Track)	T/F	
Invert	Selects output signal inversion	T/F	
NotAuto	TRUE if mode is Manual or Track	T/F	
Alarms			
Software	Block RAM data sumcheck error/network failure	T/F	
Hardware	I/O module failure / transmitter PSU faults	T/F	
CctFault	Output is open/short circuit	T/F	
OvrDrive	Output being overdriven	T/F	
Combined	OR-ing of all Alarms bits	T/F	
SiteNo	Linked I/O module's T600 ISB address	1-8	
OutType	Specifies output type	Menu	
HR_out, LR_out	Plant signal HR, LR equivalents		

Continued...

Parameter	Function	Units	Status
<i>Continued...</i>			
Status		ABCD hex	 
Reset	I/O module has reset	T/F	D
Missing	No I/O module at site	T/F	
CommsErr	I/O module comms fail	T/F	
BadType	No such channel on I/O module	T/F	
WrongCal	Wrong calibration data version (EEPROM)	T/F	C
BadCal	Corrupted calibration data	T/F	
FaultCct	Open- or short-circuit	T/F	B
BadRange	Bad range selected	T/F	
BadSetup	Setup invalid for connected I/O module	T/F	
HwFault	Fault in I/O module	T/F	A
OverDrv	Output being overdriven	T/F	
Killed	Output has been killed	T/F	
Options		(C)D hex	 
PwrFLo	TRUE = output low on power-up	T/F	D
CPUFLo	TRUE = output low on CPU failure	T/F	
Channel	Specifies channel number within I/O module	1-n	

Table 128 Block parameters

Fallback. (AUTO/MANUAL). The next (suppressed) operating mode, that the LIN function block adopts if the current mode is disabled or deselected.

OP. Requested output. Input to the block from the Strategy, and the source of *AO* in *AUTO* mode.

HR_OP, LR_OP. High and low range of *AO* (and *OP*) in engineering units. *HR_OP* and *LR_OP* define two points on a linear engineering units scale that normally map to points *HR_out* and *LR_out*, respectively, on a linear (volts or milliamps) scale. This pair of scales is used by the block to derive the output signal from *AO*.

The equivalent conversion equation is:

$$Output = LR_out + \left[\frac{AO - LR_OP}{HR_OP - LR_OP} \times (HR_out - LR_out) \right]$$

NOTE The *Invert* bit affects the derivation of the output. See *Invert*.

AO. Analogue output. Actual output (as opposed to the demanded output *OP* or *Track* value) from the block before any inversion and de-ranging. *AO* is a special parameter, in that it is guaranteed to be correctly set up before any other blocks start to execute (including power fail low options, if applicable). This feature allows *AO* to be connected into the *InitDmnd* field of the *MAN_STAT* block.

Track. Source of *AO* in Track mode.

SelTrack. TRUE selects Track mode.

Invert. If TRUE, this has the effect of mapping *HR_OP* to *LR_out*, and *LR_OP* to *HR_out*, which inverts the sense of the output signal, e.g. for fail-safe operation or for reverse-acting valves, see *HR_OP*, *LR_OP* section.

NotAuto. TRUE when the block is not operating in Auto mode.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block's RAM data.
- **Hardware.** A hardware alarm is generated when any of the bits in the *Status* parameter are set, e.g. in the event of a power interruption, incorrect module type, circuit hardware fault, etc.
- **CctFault.** Circuit fault. Reported when the module detects an open circuit (on current outputs), or a short circuit (on voltage outputs).

- **OvrDrive.** Output overdrive condition. Reported when the output signal is being overdriven by another output module. Used in redundant control strategies. Note that this facility is not supported by all I/O hardware; please refer to specific hardware documentation.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

SiteNo. The address on the T600 Series instrument's Internal Serial Bus (ISB) of the I/O module to which the AN_OUT block is linked. Valid ISB addresses depend on the T600 I/O hardware in use. E.g., in a T640 instrument, I/O sites 1 and 2 are allocated *SiteNos* of 1 and 2 respectively, and any external I/O modules take *SiteNos* of 3 onwards. An invalid *SiteNo* sets the *Status.Missing* bit.

NOTE You cannot configure *SiteNo* if the *Task* parameter is still at its default value of zero.

OutType. (Volts/mA). Selects voltage or current output type. *Volts* configures the block as a high level voltage output with a maximum range of -10 to +10V. *mA* configures the block as a current output with a maximum range of -20 to +20mA.

The output ranges given above are block maxima only - you should consult the I/O module documentation for specific limits.

NOTE. Where certain options are not available on certain channels, the *Status.BadSetup* bit is set.

HR_out, LR_out. High & low range of the output signal, in units appropriate to the currently selected *OutType* (Volts or mA). *HR_out* and *LR_out* define two points on a linear scale that map to points *HR_OP* and *LR_OP*, respectively, on a linear engineering units *AO* scale. Nominally, when *AO* equals *HR_OP*, the real output is at the value in *HR_out*; when at *LR_OP*, the real output equals *LR_out*. Please refer to the section on *HR_OP, LR_OP* above for details. Note that *HR_out* cannot exceed 10V or 20mA when *OutType* = Volts or mA, respectively. *HR_out* and *LR_out* can always be set independently of *HR_OP* and *LR_OP*.

The block compares the entered values of *HR_out* and *LR_out* with the list of available hardware ranges supported by the I/O module, and chooses the 'best fit' range to maximise the resolution of the output. Scaling to the exact values of *HR_out* and *LR_out* is done in software.

Status. Bitfield indicating general comms./hardware error conditions, see *Table 128*.

Options. Bitfield whose bits specify - if TRUE - what happens to the block's analogue output in the event of a power-up (after a power failure), or a CPU failure. If FALSE, the bits have no effect on the output, i.e. it holds its existing level in the event of a warm or 'tepid' start, or adopts its configured value under cold start conditions, see *Table 128*.

Channel. Specifies a channel number within the I/O module to which the AN_OUT block is linked. Valid channel numbers depend on the type of I/O hardware in use. Please refer to the I/O module documentation for details.

NOTE An invalid *Channel* sets the *Status.BadType* bit, and configuration of the *Channel* is not permitted if the *Task* parameter is still at its default value of zero.

DG_IN: 8-CHANNEL DIGITAL INPUT BLOCK

Block function

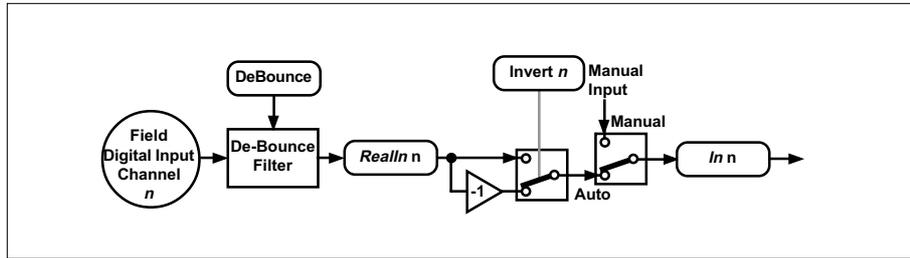


Figure 51 Block schematic

Please refer to *Figure 51*, which schematises one of the eight digital channels. The DG_IN block allows eight digital signals to be input to the Strategy, and provides auto/manual control, inversion, and debounce. Inputs can be configured as either all contact or all voltage sensing. Contacts can use the output from the transmitter power supply (which has an overload alarm), provided it is not also supplying analogue transmitters. Alternatively, the external pullup voltage terminal (1X) can be used for hardware pullup of up to eight digital inputs. Voltage sensing inputs have variable switching threshold.

Alarm fields are not provided directly on digital input channels. The DIGALARM block should be used for alarms on each input if required.

Block parameters

Symbols used in *Table 129* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Mode	Current operating mode	Menu	
Realln	Field input connections states (actual)	CD hex	
Bit0	Channel 1 field input state	T/F - 1	D
Bit1	Channel 2 field input state	T/F - 2	
Bit2	Channel 3 field input state	T/F - 4	
Bit3	Channel 4 field input state	T/F - 8	
Bit4	Channel 5 field input state	T/F - 1	C
Bit5	Channel 6 field input state	T/F - 2	
Bit6	Channel 7 field input state	T/F - 4	
Bit7	Channel 8 field input state	T/F - 8	
Invert	field input invert states	CD hex	
Bit0	Channel 1 field input inverted	T/F - 1	D
Bit1	Channel 2 field input inverted	T/F - 2	
Bit2	Channel 3 field input inverted	T/F - 4	
Bit3	Channel 4 field input inverted	T/F - 8	
Bit4	Channel 5 field input inverted	T/F - 1	C
Bit5	Channel 6 field input inverted	T/F - 2	
Bit6	Channel 7 field input inverted	T/F - 4	
Bit7	Channel 8 field input inverted	T/F - 8	
In	Strategy input states	CD hex	
Bit0	Channel 1 strategy input state	T/F - 1	D
Bit1	Channel 2 strategy input state	T/F - 2	
Bit2	Channel 3 strategy input state	T/F - 4	
Bit3	Channel 4 strategy input state	T/F - 8	
Bit4	Channel 5 strategy input state	T/F - 1	C
Bit5	Channel 6 strategy input state	T/F - 2	
Bit6	Channel 7 strategy input state	T/F - 4	
Bit7	Channel 8 strategy input state	T/F - 8	
Alarms			
Software	Block RAM data sumcheck error/network failure	T/F	
Hardware	I/O module failure / transmitter PSU faults	T/F	
Combined	OR-ing of all Alarms bits	T/F	

Continued...

Parameter	Function	Units	Status				
<i>Continued...</i>							
SiteNo	Linked I/O module's T600 ISB address	Integer					
InType	Input type	Menu					
Thresh	Threshold voltage	Numeric					
DeBounce	Debounce time (secs)	Numeric					
Status	Comms/hardware status	ABCD hex	 				
Reset	Transient flag Set at power-up	T/F	<table border="1"> <tr><td>1</td></tr> <tr><td>2</td></tr> <tr><td>4</td></tr> <tr><td>8</td></tr> </table> D	1	2	4	8
1							
2							
4							
8							
Missing	Invalid SiteNo value	T/F					
CommsErr	Comms fault (T600 CPU to module)	T/F					
BadType	Wrong module type at site	T/F					
PSUShort	Transmitter PSU overload	T/F	<table border="1"> <tr><td>1</td></tr> <tr><td>2</td></tr> <tr><td>4</td></tr> <tr><td>8</td></tr> </table> C	1	2	4	8
1							
2							
4							
8							
BadThr	Thresh value not supported by I/O module	T/F					
BadDeb	DeBounce time not supported by I/O module	T/F	<table border="1"> <tr><td>1</td></tr> <tr><td>2</td></tr> <tr><td>4</td></tr> <tr><td>8</td></tr> </table> B	1	2	4	8
1							
2							
4							
8							
BadSetup	Configured option unavailable in I/O hardware	T/F					
HwFault	Hardware fault in I/O module	T/F	<table border="1"> <tr><td>1</td></tr> <tr><td>2</td></tr> <tr><td>4</td></tr> <tr><td>8</td></tr> </table> A	1	2	4	8
1							
2							
4							
8							
BitCount	Number of bits supported by I/O hardware	Integer					

Table 129 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these 'header' fields.

Task. This field (default value = 0) specifies the user task the block is allocated to run in (1 - 4). Note that certain normally read-write parameters in this block cannot be configured unless *Task* has first been set to a non-zero value.

Mode. (AUTO/MANUAL). In AUTO mode the status field *In* follows the input stimuli from field connections. In MANUAL mode *In* can be manipulated independently of field stimuli.

RealIn. Bitfield indicating actual states of the eight field input connections.

Invert. Bitfield selecting inversion of any of the eight inputs to the Strategy.

In. Bitfield indicating the states of the eight inputs to the Strategy. These may be the true or inverted form of *RealIn* depending on the states of the *Invert* bits. If the block is in manual mode, *In* is independent of *RealIn*.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block's RAM data.
- **Hardware.** A hardware alarm is generated when any of the bits in the *Status* parameter are set, e.g. in the event of a power interruption, incorrect module type, PSU overload, circuit hardware fault, etc.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

SiteNo. Specifies the address (1 - 8) on the T600 Series instrument's internal serial bus where the I/O module corresponding to this DG_IN block is located. Note that an invalid *SiteNo* sets the *Missing* status bit, and that you cannot configure *SiteNo* if the *Task* parameter is still at its default value of zero.

NOTE For the high-level I/O board in a T640 instrument, each of its two sites (1 and 2) can handle only four bits (bits 0 to 3). Therefore two DG_IN blocks - with their respective *SiteNo* parameters set to 1 and 2 - are required to handle the eight bits available within the instrument.

InType. (Volts/Contact). Specifies type of input.

Thresh. Threshold voltage. Specifies threshold for voltage input option.

DeBounce. Debounce time, in seconds. This is the minimum time a change in the digital input signal must persist before the *RealIn n* parameter is allowed to switch to the new state. The filtering action of *DeBounce* is shown in *Figure 52*. (Minimum debounce time is dependent on scan time; maximum time is 5 seconds.)

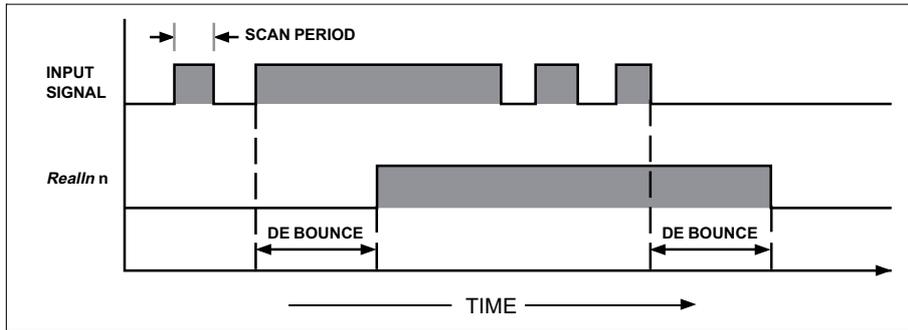


Figure 52 Effects of Debounce on digital input signal

Status. Bitfield indicating general comms./hardware error conditions. Please refer to *Table 129* for details.

BitCount. Reports the number of bits (digital inputs) supported by the associated hardware module.

DG_OUT: 8-CHANNEL DIGITAL OUTPUT BLOCK

Block function

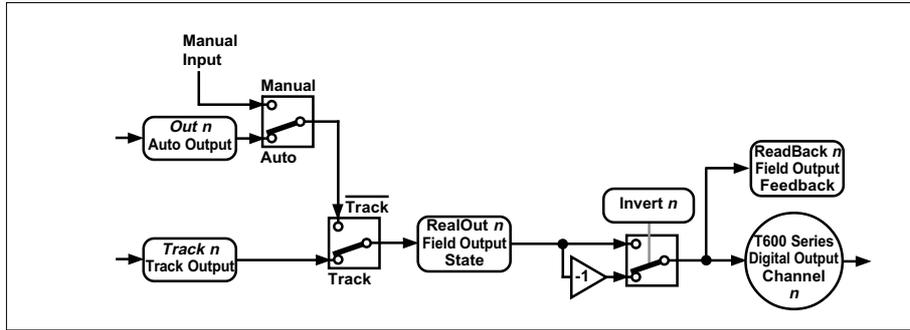


Figure 53 Block schematic

Please see *Figure 53*, it refers to one of the eight digital channels. The DG_OUT block transfers logic signals to digital output channels on a T600 Series digital output module. The block provides Auto/Manual/Track control, inversion and discrepancy checking of output feedback.

Block parameters

Symbols used in *Table 130* are explained in *Table 1*. Additional parameter information is given in the Block specification menu section following.

Parameter	Function	Units	Status
Mode	Current operating mode	Menu	
FallBack	Suppressed operating mode	Menu	
Out	Output states from Strategy	CD hex	
Bit0	Channel 1 output state	T/F - 1	D
Bit1	Channel 2 output state	T/F - 2	
Bit2	Channel 3 output state	T/F - 4	
Bit3	Channel 4 output state	T/F - 8	
Bit4	Channel 5 output state	T/F - 1	C
Bit5	Channel 6 output state	T/F - 2	
Bit6	Channel 7 output state	T/F - 4	
Bit7	Channel 8 output state	T/F - 8	
Track	Track output states	CD hex	
Bit0	Channel 1 track output state	T/F - 1	D
Bit1	Channel 2 track output state	T/F - 2	
Bit2	Channel 3 track output state	T/F - 4	
Bit3	Channel 4 track output state	T/F - 8	
Bit4	Channel 5 track output state	T/F - 1	D
Bit5	Channel 6 track output state	T/F - 2	
Bit6	Channel 7 track output state	T/F - 4	
Bit7	Channel 8 track output state	T/F - 8	
RealOut	Output states (before inversion)	CD hex	
Bit0	Channel 1 output state	T/F - 1	D
Bit1	Channel 2 output state	T/F - 2	
Bit2	Channel 3 output state	T/F - 4	
Bit3	Channel 4 output state	T/F - 8	
Bit4	Channel 5 output state	T/F - 1	C
Bit5	Channel 6 output state	T/F - 2	
Bit6	Channel 7 output state	T/F - 4	
Bit7	Channel 8 output state	T/F - 8	
Invert	Field output invert states	CD hex	
Bit0	Channel 1 field output inverted	T/F - 1	D
Bit1	Channel 2 field output inverted	T/F - 2	
Bit2	Channel 3 field output inverted	T/F - 4	
Bit3	Channel 4 field output inverted	T/F - 8	

Continued...

Parameter	Function	Units	Status	
<i>Continued...</i>				
Bit4	Channel 5 field output inverted	T/F	C	
Bit5	Channel 6 field output inverted	T/F		
Bit6	Channel 7 field output inverted	T/F		
Bit7	Channel 8 field output inverted	T/F		
Readback	Feedback of low-level I/O module output states	CD hex	 	
Bit0	Channel 1 feedback	T/F	D	
Bit1	Channel 2 feedback	T/F		
Bit2	Channel 3 feedback	T/F		
Bit3	Channel 4 feedback	T/F		
Bit4	Channel 5 feedback	T/F	C	
Bit5	Channel 6 feedback	T/F		
Bit6	Channel 7 feedback	T/F		
Bit7	Channel 8 feedback	T/F		
Alarms			  	
Software	Block RAM data sumcheck error/network failure	T/F		
Hardware	I/O module failure / transmitter PSU faults	T/F		
CctFault	RealOut/Readback discrepancy detected	T/F		
Combined	OR-ing of all Alarms bits	T/F		
SiteNo	Linked I/O module's T600 ISB address	Integer		
Pullup	High logic status output voltage	Menu		
SelTrack	Track mode select	T/F		
NotAuto	NOT Auto mode	T/F	 	
Status	Comms/hardware status	ABCD hex	 	
Reset	Transient flag set at power-up	T/F	D	
Missing	Invalid SiteNo value	T/F		
CommsErr	Comms fault (T600 to module)	T/F		
BadType	Wrong module type at site	T/F		
BadPulup	Pullup value not supported by I/O module	T/F	C	
BadSetup	Configured option unavailable in I/O hardware	T/F	A	
HwFault	Hardware fault in I/O module	T/F		
Options	Block options	(C)D hex	 	
PwrFIlo	TRUE = output low on power-up	T/F	D	
CPUFIlo	TRUE = output low on CPU failure	T/F		
BitCount	Number of bits supported by I/O hardware	Integer		

Table 130 Block parameters

Block specification menu

Dbase, Block, Type. See Appendix D page 544 for details of these ‘header’ fields.

Task. This field (default value = 0) specifies the user task the block is allocated to run in (1 - 4). Certain normally read-write parameters in this block cannot be configured unless *Task* has first been set to a non-zero value.

Mode. (AUTO/MANUAL/TRACK). The current operating mode. TRACK must be selected via the *SelTrack* parameter.

Fallback. (AUTO/MANUAL). The next (suppressed) operating mode, that the block adopts if the current mode is disabled or deselected.

Out. In AUTO mode, this bitfield parameter indicates the status of output channels set by the Strategy. In MANUAL mode it has read/write status. This field may indicate the true or inverted form of actual output channels depending on the status of the *Invert* fields.

Track. Bitfield controlling the states of outputs in Track mode.

RealOut. Bitfield indicating the status (before any inversions) of the field output connections.

Invert. Bitfield specifying inversions of the field outputs.

Readback. Bitfield showing feedback of the actual output states from the T600 Series high-level and/or low-level (TC) digital output module (before any inversions). Used for discrepancy checking by comparison with *RealOut*, which should be identical (before any inversions).

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block's RAM data.
- **Hardware.** A hardware alarm is generated when any of the bits in the *Status* parameter are set, e.g. in the event of a power interruption, incorrect module type, circuit hardware fault, etc.
- **CctFault.** This alarm occurs when a discrepancy is detected between the status of *RealOut* and *Readback*, indicating a fault in an output channel.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm

SiteNo. Specifies the address (1 - 8) on the T600 Series instrument's internal serial bus where the I/O module corresponding to this DG_OUT block is located. An invalid *SiteNo* sets the *Missing* status bit, and that you cannot configure *SiteNo* if the *Task* parameter is still at its default value of zero.

NOTE For the high-level I/O board in a T640 instrument, each of its two sites (1 and 2) can handle only four bits (bits 0 to 3). Therefore two DG_OUT blocks, with the respective *SiteNo* parameters set to 1 and 2, are required to handle the eight bits available within the instrument.

Pullup. (External/5/15/24). Defines the output voltage used for the high logic status.

SelTrack. Selects Track mode.

NotAuto. Indicates that the block is not operating in Automatic mode.

Status. Bitfield indicating general comms./hardware error conditions. Please refer to *Table 130* for details.

Options. Bitfield for selecting block operating options, see *Table 130*.

BitCount. Reports the number of bits (digital outputs) supported by the associated hardware module.

DGPULS_4: 4-channel Digital Pulse Block

Block function

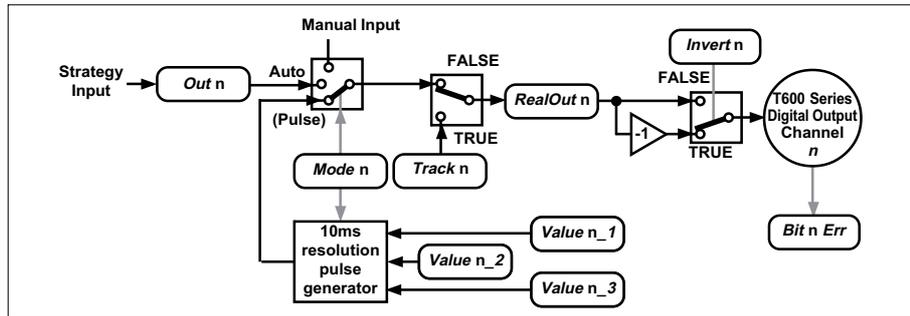


Figure 54 DGPULS_4 block schematic

The schematic in *Figure 54* shows one of the four channels of the DGPULS_4 block. Each channel's mode can be independently selected. In AUTO mode a channel transfers digital outputs from the strategy to the field via the associated T600 Series digital output module. Manual, track, and four pulse modes, as well as inversion and discrepancy-checking, are also provided. Note that for a T640 instrument with high-level I/O boards in both sites, only Site 1 can support a DGPULS_4 block.

Pulse modes

NOTE The digital output hardware must support a *timing function* on each channel used by a DGPULS_4 channel in a pulse mode, see hardware specification in the instrument's manual.

In the pulse modes the block channels can output a single pulse (ONE_SHOT), a time-proportioned rectangular wave (TPO), a train of pulses (PLS_TRN), or, by connecting together a pair of channels a dual pulse (DUAL_PLS). The set of *Value n_1* to *Value n_3* parameters specify timings and mark/space ratios of the pulses.

In all pulse modes, the task execution of the block derives integer values that represent units of either 10ms or 20ms (depending on the I/O module). These integer values are downloaded to the I/O module so that it can perform the necessary high speed timing.

In the simplified schematic of *Figure 54* the *RealOut n* parameter is shown as driving the output hardware, but in reality this field is updated by reading back its state from the I/O module. Similarly, as both the requested digital state and its readback value are possibly running considerably faster in the hardware than in the block, they are compared at the I/O module, and a healthy value (*Bit n Err* flag) returned to the block. *Bit n Err* flags are latching, so that fleeting mis-matches are not lost. The I/O module asserts a healthy flag every time the value has been successfully communicated to the main processor. Inversion also is implemented in the I/O module (not as suggested in the schematic) for similar reasons.

ONE_SHOT mode

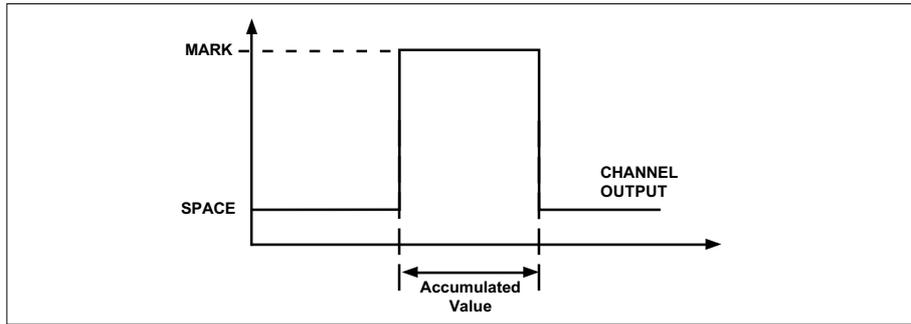


Figure 55 ONE_SHOT mode output

A typical application for the one-shot pulse output is in on/off motor-driven valves. In this mode the block generates a single pulse (see *Figure 55*). At each execution of the block the value currently in *Valuen_1* is added to an accumulator that represents the remaining pulse length in seconds. Whenever the accumulator is non-zero, *RealOutn* = TRUE, i.e. the output is in the ‘mark’ state, and the accumulator counts down in real time. Whenever the accumulator is zero, *RealOutn* = FALSE, i.e. the output is in the ‘space’ state, and the accumulator is ‘idle’. The ONE_SHOT accumulator is unsigned.

At any time, making the *Accn_Rst (Options)* bit TRUE resets channel n’s accumulator to zero.

TPO mode

TPO (time-proportioned output) mode generates a continuous rectangular wave. *Valuen_1* specifies the period in seconds (see *Figure 56*), *Valuen_2* specifies the percentage of the cycle at mark state, and *Valuen_3* specifies both a minimum mark duration and a minimum space duration (in seconds). If the requested pulse length - either mark or space - is less than the specified minimum, the pulse is not generated, i.e. the output holds at a steady space or mark, respectively. If the length of the mark state (*Valuen_2*) is changed whilst the output is in that state, the change takes effect immediately, without waiting until the end of the current period.

Resolution-compensation is applied, so that if the requested mark length cannot be represented within the resolution of the I/O module’s pulse length, the actual mark length varies about the requested one. E.g. if the mark state is specified as 25ms long, but the I/O module can only resolve to the nearest 10ms, the actual mark length will alternate between 20ms and 30ms, averaging 25ms. Resolution-compensation is accurate to 12 bits.

(The *Options.Accn_Rst* bit has no application in TPO mode.)

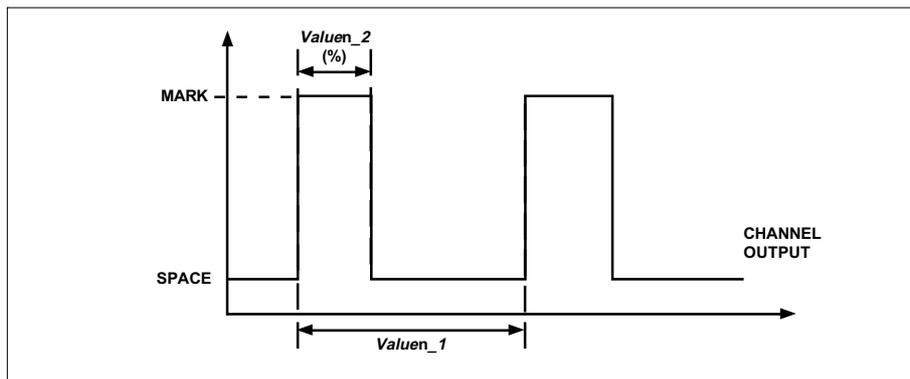


Figure 56 TPO mode output

PLS_TRN mode

Pulse-train mode generates a train of pulses, useful in stepper motor and choke valve applications, etc. *Valuen_2* (see *Figure 57*) specifies the pulse period. At each execution of the block, the value currently in *Valuen_1* is added to an accumulator which contains a count of the number of pulses remaining to be output. Only whole cycles can be output, so if the accumulator value is non-integral, the actual number of pulses output is the next lowest integer value. The accumulator decrements for every pulse output.

A new pulse train cannot be started until the previous one has finished, this being defined as one half-period duration after the last falling edge.

Pulses have a mark/space ratio of one (to within the 10ms/20ms resolution). Any change to *Valuen_2* (period) takes immediate effect.

At any time, making the *Accn_Rst (Options)* bit TRUE resets channel n's accumulator to zero. The PLS_TRN accumulator is unsigned.

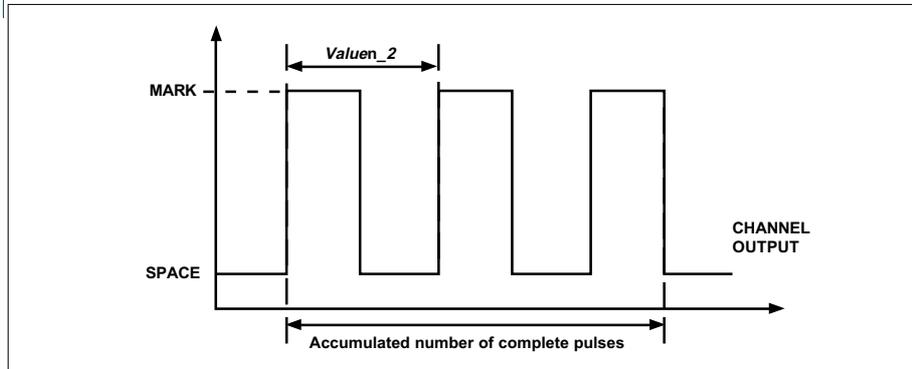


Figure 57 PLS_TRN mode output

DUAL_PLS mode

The Dual_Pls mode combines two channels as a pair - channels 1 and 2 form one pair, channels 3 and 4 another. A channel-pair outputs a 'linked' pair of pulses, one via each channel (see Figure 58).

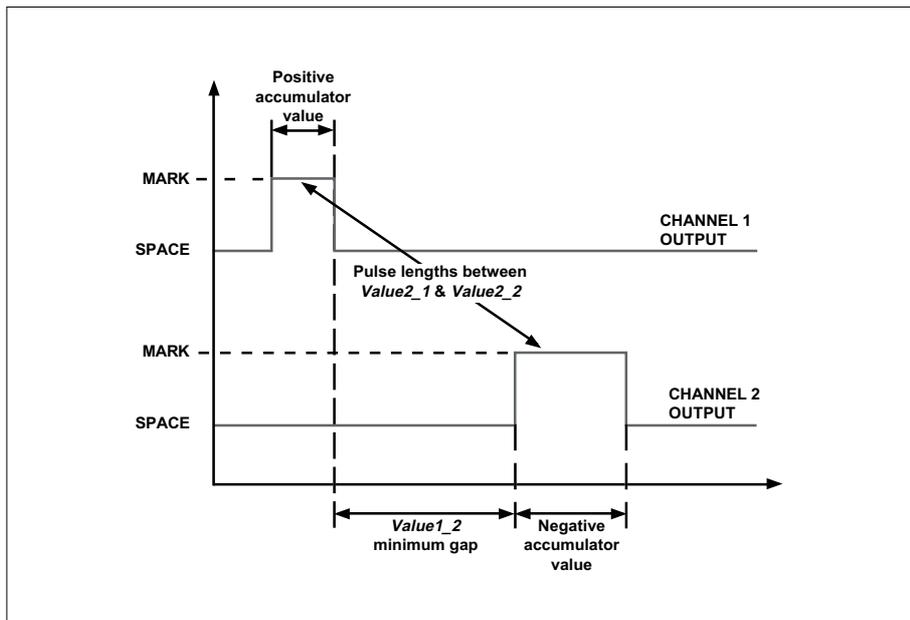


Figure 58 DUAL_PLS mode output

Considering the Ch1-Ch2 pair, at each execution of the block the value currently in *Value1_1*, which may be positive or negative, is added to an accumulator that represents the required pulse length in seconds. When the accumulator is zero, both channels output a space condition. When the accumulator is positive (and greater than *Value2_1*) a mark condition is output on channel 1, and the accumulator counts down in real time. When the accumulator is negative (and of absolute value greater than *Value2_1*) a mark condition is output on channel 2, and the accumulator counts up in real time. *Value2_1* therefore represents a minimum permitted pulse length. *Value2_2* specifies a maximum pulse length that cannot be exceeded. *Value1_2* specifies a minimum gap that must exist between a mark on channel 1 and a mark on channel 2.

At any time, setting the *Acc1_Rst (Options)* bit TRUE resets a positive accumulator value, and *Acc2_Rst* TRUE resets a negative accumulator value, these bits are not self-resetting.

Block parameters

Symbols used in *Table 131* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Out			
Chan1	Channel 1 selected states	T/F	
Chan2	Channel 2 selected states	T/F	
Chan3	Channel 3 selected states	T/F	
Chan4	Channel 4 selected states	T/F	
Track			
Chan1	Channel 1 track states	T/F	
Chan2	Channel 2 track states	T/F	
Chan3	Channel 3 track states	T/F	
Chan4	Channel 4 track states	T/F	
Invert			
Chan1	Channel 1 invert states	T/F	
Chan2	Channel 2 invert states	T/F	
Chan3	Channel 3 invert states	T/F	
Chan4	Channel 4 invert states	T/F	
RealOut			
Chan1	Channel 1 output states	T/F	
Chan2	Channel 2 output states	T/F	
Chan3	Channel 3 output states	T/F	
Chan4	Channel 4 output states	T/F	
SelTrack			
Chan1	Select channel 1 track mode	T/F	
Chan2	Select channel 2 track mode	T/F	
Chan3	Select channel 3 track mode	T/F	
Chan4	Select channel 4 track mode	T/F	
NotAuto			
Chan1	If channel 1 in Manual or Track	T/F	
Chan2	If channel 2 in Manual or Track	T/F	
Chan3	If channel 3 in Manual or Track	T/F	
Chan4	If channel 4 in Manual or Track	T/F	
Mode1-Mode4	Channels 1-4 operating modes, resp.	Menu	
Fallbak1-Fallbak4	Channels 1-4 fallback modes, resp.	Menu	
Valuen_1 to 3	Channel n, pulse mode values 1 to 3, resp.	Float Value	
Alarms			
Software	Block RAM data sumcheck error/network failure	T/F	
Hardware	I/O module failure / transmitter PSU faults	T/F	
Bit1Err	Channel 1 readback error	T/F	
Bit2Err	Channel 2 readback error	T/F	
Bit3Err	Channel 3 readback error	T/F	
Bit4Err	Channel 4 readback error	T/F	
Combined	OR-ing of all Alarms bits	T/F	
SiteNo	Linked I/O module's T600 ISB address	Integer	
BitCount	Number of bits supported by I/O hardware	Integer	
Options			
PwrFLo	Power fail low	T/F	
CPUFLo	CPU fail low	T/F	
Acc1_Rst	Reset bit 1 accumulators	T/F	
Acc2_Rst	Reset bit 2 accumulators	T/F	
Acc3_Rst	Reset bit 3 accumulators	T/F	
Acc4_Rst	Reset bit 4 accumulators	T/F	

Continued...

Parameter	Function	Units	Status
<i>Continued...</i>			
Status		ABCD hex	 
Reset	I/O module has reset	T/F	D
Missing	No I/O module at site	T/F	
CommsErr	I/O module comms fail	T/F	
BadType	No such channel on I/O module	T/F	
BadPulup	Selected PullUp not supported on I/O module	T/F	C
BadPulse	Unsupported pulse mode	T/F	
			B
BadSetup	Invalid setup this channel	T/F	A
HwFault	Hardware fault in I/O module	T/F	
Pullup	Specifies logic 1 output voltage	Menu	

Table 131 Block parameters

Block specification menu

Dbase, Block, Type. See Appendix D page 544 for details of these ‘header’ fields.

Task. This field (default value = 0) specifies the user task the block is allocated to run in (1 - 4). Note that certain normally read-write parameters in this block cannot be configured unless *Task* has first been set to a non-zero value.

Out. This bitfield parameter can be connected directly to the Strategy. In automatic mode, *Out* drives the output channels, (subject to the status of the *Invert* fields). In manual and pulse modes, *Out* is disconnected from the outputs. See Figure 55.

Track. Bitfield controlling the states of the four output channels in track mode.

Invert. Bitfield specifying inversions of the four field output channels.

RealOut. Bitfield indicating the status (before any inversions) of the four field output channels.

SelTrack. A TRUE bit selects track mode for that channel.

NotAuto. TRUE if the channel is NOT operating in AUTO mode.

Mode1 to Mode4. (AUTO/MANUAL/TRACK/ONE_SHOT/TPO/PLS_TRN/DUAL_PLS). Selects the current operating mode for channels 1 to 4, respectively.

NOTE. TRACK must be selected via the *SelTrack* parameter. In AUTO mode the field outputs are driven by *Out*; in MANUAL, by a manual input to *RealOut*; and in TRACK, by the *Track* parameter - all subject to inversion.

For a detailed description of the four pulse modes, see the *Pulse modes* section above.

Fallbak1 to Fallbak4. (AUTO/MANUAL/ONE_SHOT/TPO/PLS_TRN/DUAL_PLS). The operating mode that the block adopts if TRACK mode is deselected.

Valuen_m, (n=1 to 4, m=1 to 3). A set of three values (*m*), for each of the four channels (*n*). These values specify timings (in seconds) and percentages connected with the generation of output pulses in one of the four pulse modes, as selected by the corresponding channel’s *Mode(n)* parameter, see *Pulse modes* section.

Alarms. See Appendix D page 545 for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.
- **Hardware.** A hardware alarm is generated when any of the bits in the *Status* parameter are set, e.g. in the event of an invalid channel setup, incorrect module type, circuit hardware fault, etc.

- **Bit1Err to Bit4Err.** These alarms occur for channels 1 to 4, respectively, when a discrepancy is detected between the desired channel output *RealOutn* and a readback output, indicating a fault in an output channel. Any inversion configured for the channel is allowed for in the discrepancy checking.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

SiteNo. Specifies the address (1 - 8) on the T600 Series instrument's Internal Serial Bus where the I/O module corresponding to this DGPULS_4 block is located.

NOTE An invalid *SiteNo* sets the *Status.Missing* bit, and configuring the *SiteNo* is not permitted if the *Task* parameter is still at its default value of zero.

BitCount. Reports the number of bits (digital outputs) supported by the associated hardware module.

Options. Bitfield for selecting block operating options, see *Table 131*.

Status. Bitfield indicating general comms./hardware error conditions, see *Table 131*.

Pullup. (External/5/15/24). Defines the output voltage used for the high logic status.

AI_CALIB: ANALOGUE INPUT CALIBRATION BLOCK

Block function

The AI_CALIB block is used to calibrate analogue input channels in the T600 Series of I/O modules. A database is created in the T600 Series instrument that contains the AN_IP block associated with the I/O module to be calibrated, plus an AI_CALIB block. The calibration block effectively ‘attaches’ itself to the named AN_IP block, forces it into CALIB mode, then works through it to calibrate the specified I/O channel. Via its *Action* field(s), the calibration block displays one of a sequence of instructions for the operator to perform, who then sets the *Status.CONTINUE* field when ready to move on to the next instruction.

In a remote instrument, a cached copy of the calibration block is created so that the calibration process can be driven via a full screen interface to the block.

Details of how to calibrate the various types of analogue input module are given in the *Calibration procedures* section below.

Block parameters

Symbols used in *Table 132* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Action (invisible field)	Give calibration instructions to operator	Menu	
Value		Menu	
Status	Current status of calibration process (& abort I/P)	Menu	
Alarms			
Software	Block RAM data sumcheck error/network failure	T/F	
Failed	Calibration process aborted (default priority=6)	T/F	
Combined	OR-ing of all Alarms bits	T/F	
SiteNo	Linked I/O module's T600 ISB address	Integer	
Channel	Specifies channel number within I/O module	Integer	
InType	Specifies type of analogue input being calibrated	Menu	
Range	Reports hardware range being calibrated	Menu	
AN_IP	Name of AN_IP block to be attached to	Alphanumeric	
AI_Alarm	Shows if channel config. valid/AN_IP in alarm	Menu	

Table 132 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

Task. This field (default value = 0) specifies the user task the block is allocated to run in (1 - 4).

NOTE Certain normally read-write parameters in this block cannot be configured unless *Task* has first been set to a non-zero value.

Action (part 1 – visible). (Done/Apply/Enter/Write) Displays one of four short text messages which, in combination with a message displayed by the second (invisible) part of the *Action* field immediately below it, enables a variety of instructions to be given to the operator.

‘Done’ appears when the block is in idle mode, i.e. before calibration has started or when calibration is complete. See next paragraph, *Action (part 2 - invisible)*.

Action (part 2 – invisible). ([blank]/High rng/Low rng/E2 data?/Test 1/Test 2/R/V1/V3/Vcjc/Vbase/switches) This (invisible) field supplies the second part of an operator instruction, e.g. **Apply Low rng**. The operator performs the given instruction, and then selects *CONTINUE* in the *Status* parameter to tell the block that the instruction has been duly carried out. Refer to the section *Calibration procedures* for details of the calibration process.

The *Action* instructions are necessarily very brief, and it is assumed that the operator is referring to this manual for a full description of the required actions.

Value. When the operator is instructed to ‘Enter’ a value, it should be entered into this field.

Status. (WAITING/CONTINUE/PROCESSING/ABORT) Shows the operator the current status of the calibration process, and also allows him to abort it.

- **WAITING.** Shown whenever the AI_CALIB block is awaiting an action by the operator.
- **CONTINUE.** Must be entered by the operator when the awaited action is completed.
- **PROCESSING.** Displayed when the AI_CALIB block is in the process of carrying out an action.
- **ABORT.** Entering this field abandons the calibration process.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block's RAM data.
- **Failed.** Indicates an abort from the calibration process. This alarm's priority of 6 is automatically set by the block and means that, if it occurs, the alarm must be acknowledged before continuing.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

SiteNo. Specifies the address (1 - 8) on the T600 Series instrument's internal serial bus of the I/O module whose analogue input is being calibrated. The AI_CALIB block automatically copies this field from the corresponding one in the AN_IP block when first attached, but *SiteNo* may be altered subsequently.

NOTE Configuration of the *SiteNo* is not permitted if the *Task* parameter is still at its default value of zero.

Channel. Specifies the number (1 - 8) of the analogue input channel being calibrated. The AI_CALIB block automatically copies this field from the corresponding one in the AN_IP block when first attached, but *Channel* may be altered subsequently.

NOTE Configuration of the *Channel* is not permitted if the *Task* parameter is still at its default value of zero.

InType. (HiLevel/Therm/RTD/Freq) Specifies the type of analogue input to be calibrated. The AI_CALIB block automatically copies this field from the corresponding one in the AN_IP block when first attached, but *InType* may be altered subsequently.

NOTE The entire calibration process for a particular *InType* must be run to completion for all supported ranges, or aborted. It is not possible to calibrate only some of the ranges.

Range. (0-1.25 V/0-2.5 V/0-5 V/0-10 V/0-12.5 mV/0-25 mV/0-50 mV/0-100 mV/0-50 Ohm/0-100 Ohm/0-500 Ohm/0-1k Ohm/0-30k Hz) Read-only field reporting which one of the specified channel's hardware ranges is currently being calibrated. See the note in the *InType* section above.

AN_IP. Specifies the name of the AN_IP block to be attached to. Once a valid block name has been entered, the AI_CALIB block attaches to it, and forces it into CALIB mode.

AI_Alarm. (Invalid/Bad Loop/In alarm/OK) Indicates the current state of the calibration block.

- **Invalid.** Means an invalid configuration for the channel, or no AN_IP block found.
- **Bad Loop.** Means that the AN_IP and AI_CALIB blocks are not running in the same user task ('loop'), which they must be for calibration to operate.
- **In alarm.** Means that the attached AN_IP block is in alarm.
- **OK.** Displayed in all other cases.

Calibration procedures

Caution

Only authorised personnel using equipment of an approved standard should carry out hardware re-calibration.

To start the calibration process, enter the 'parent' I/O block name in the AI_CALIB block's *AN_IP* field. Assuming the parent block can be found, the calibration block attaches itself to it and then copies other fields as applicable, *SiteNo*, *Channel*, *InType*, *Range*. The *Action* field reports **Done**, and the *Status* field reports **WAITING**. Begin the calibration process by setting *Status* to **CONTINUE**.

This section tabulates each of the commands given by the block and the actions required of the operator in response. *InType* must be set to the required hardware type before the calibration process begins, as each type is calibrated as a separate operation.

High-level analogue input calibration sequence (*InType* = HiLevel)

Step	Action message	Action required of operator, before entering CONTINUE
1	Apply Low rng	Apply to the input the value equivalent to the low end of the range as given in the Range field. (This is always 0V for this <i>InType</i>).
2	Enter Low rng	Enter into the Value field the EXACT voltage applied, in volts.
3	Apply High rng	Apply to the input the value equivalent to approximately 100% of the high end of the range as given in the Range field.
4	Enter High rng	Enter into the Value field, the EXACT voltage applied, in volts.
5	<i>Steps 1 to 4 are repeated for each range supported by the connected hardware</i>	
6	Write E2 data?	CONTINUE only if you want the calibration data updated in the I/O module. Stopping before this point abandons this calibration, with no changes to the setup of the unit.

Thermocouple/mV analogue input calibration sequence (*InType* = Therm)

Step	Action message	Action required of operator, before entering CONTINUE
1	Apply Low rng	Apply to the input the value equivalent to the low end of the range as given in the Range field — always 0mV for this <i>InType</i> .
2	Enter Low rng	Enter into the Value field, the EXACT voltage applied, in mV.
3	Apply High rng	Apply to the input the value equivalent to approximately 100% of the high end of the range as given in the Range field. OR: if this hardware type is limited to 77% of nominal range (as is e.g. the Direct I/O board), apply 70% of the high end of the range (Either 8mV, 17mV, 35mV, or 70mV).
4	Enter High rng	Enter into the Value field, the EXACT voltage applied, in mV.
5	<i>Steps 3 & 4 are repeated for each range supported by the connected hardware</i>	
6	Apply switches	Connect the input as shown in the <i>Figure 59</i> , with both switches open.
7	Enter Vcjc	Refer to <i>Figure 59</i> . Measure Vcjc and enter the value into the Value field, in volts.
8	Apply switches	Close switch SW1 and leave SW2 open.
9	Enter Vbase	Measure Vcjc (which is now 'Vbase' of the CJC transistor) and enter the value into the Value field, in volts
10	Apply switches	Open switch SW1 and close SW2.
11	Enter Vcjc	Measure Vcjc and enter the value into the Value field, in volts
12	Write E2 data?	CONTINUE only if you want the calibration data updated in the I/O module. Stopping before this point abandons this calibration, with no changes to the setup of the unit.

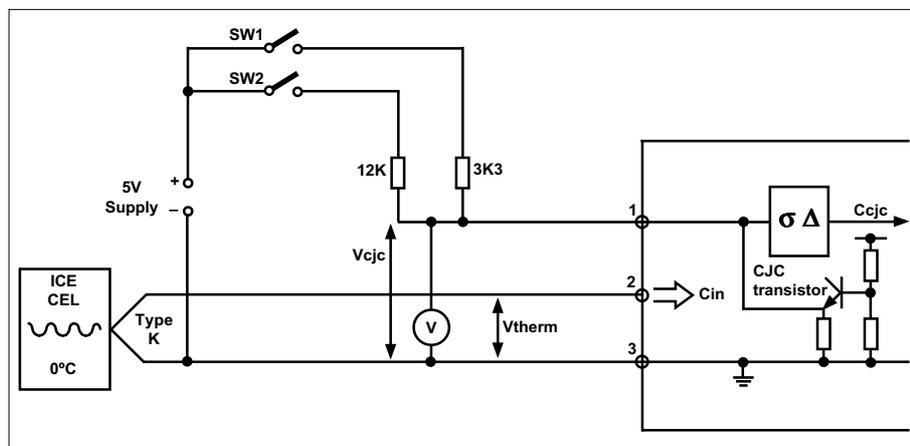


Figure 59 Thermocouple/mV calibration setup

RTD analogue input calibration sequence (InType = RTD)

Step	Action message	Action required of operator, before entering CONTINUE
1	Apply Test 1	Connect the input as shown in <i>Figure 60</i> .
2	Enter R	Measure R across C1-C2 and enter the value, in ohms.
3	Enter V1	Enter the measured value of V1, in mV.
4	Enter V3	Enter the measured value of V3, in mV.
5	Apply Test 2	Connect the input as shown in <i>Figure 61</i> . Use a resistor value Rhi equal to the high end of the range (= Range field).
6	Enter R	Measure R across C1-C2 and enter the value, in ohms.
7	Enter V1	Enter the measured value of V1, in mV.
8	Enter V3	Enter the measured value of V3, in mV.
9	Write E2 data?	CONTINUE only if you want the calibration data updated in the I/O module. Stopping before this point abandons this calibration, with no changes to the setup of the unit.
10	<i>Steps 1 to 9 are repeated for each range supported by the connected hardware</i>	

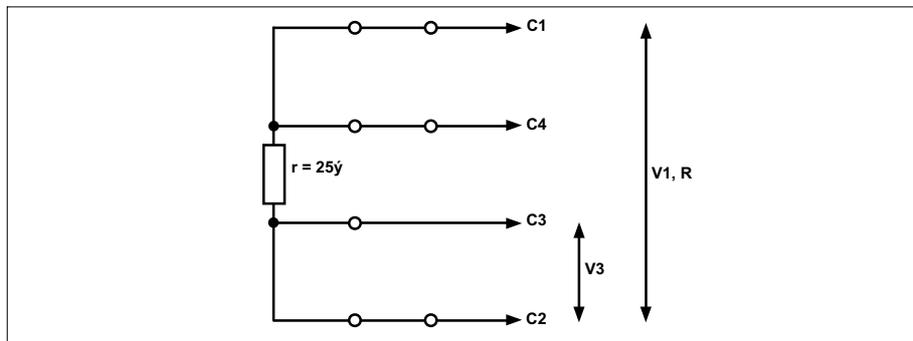


Figure 60 RTD calibration setup - Test 1

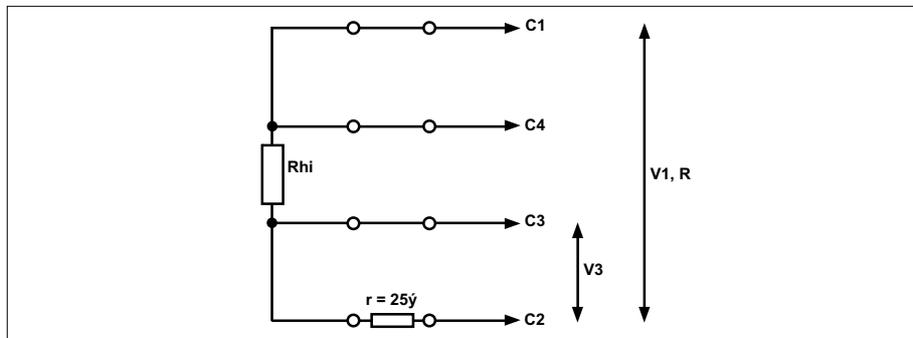


Figure 61 RTD calibration setup - Test 2

Frequency analogue input calibration sequence (InType = Freq)

Step	Action message	Action required of operator, before entering CONTINUE
1	Apply High rng	Apply a frequency equal to the high end of the range. (This is always 30kHz).
2	Enter High rng	Enter the exact value of the applied frequency, in Hz.
3	Write E2 data?	CONTINUE only if you want the calibration data updated in the I/O module. Stopping before this point abandons this calibration, with no changes to the setup of the unit.

AO_CALIB: ANALOGUE OUTPUT CALIBRATION BLOCK

Block function

The AO_CALIB block is used to calibrate analogue output channels in the T600 Series of I/O modules. A database is created in the T600 Series instrument that contains the AN_OUT block associated with the I/O module to be calibrated, plus an AO_CALIB block. The calibration block effectively ‘attaches’ itself to the named AN_OUT block, forces it into CALIB mode, then works through it to calibrate the specified I/O channel. Via its *Action* field(s), the calibration block displays one of a sequence of instructions for the operator to perform, who then sets the *Status.CONTINUE* field when ready to move on to the next instruction.

In a remote instrument, a cached copy of the calibration block is created so that the calibration process can be driven via a full screen interface to the block.

Details of how to calibrate the various types of analogue output module are given in the *Calibration procedures* section below.

Block parameters

Symbols used in *Table 133* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Action (invisible field)	Give calibration instructions to operator	Menu	
Value		Menu	
Status	Current status of calibration process (& abort I/P)	Menu	
Alarms			
Software	Block RAM data sumcheck error/network failure	T/F	
Failed	Calibration process aborted (default priority=6)	T/F	
Combined	OR-ing of all Alarms bits	T/F	
SiteNo	Linked I/O module's T600 ISB address	Integer	
Channel	Specifies channel number within I/O module	Integer	
OutType	Defines type of analogue output being calibrated	Menu	
Range	Indicates hardware range being calibrated	Menu	
AN_OUT	Name of AN_OUT block to be attached to	Alphanumeric	
AO_Alarm	Shows if channel config. valid/AN_OUT in alarm	Menu	

Table 133 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

Task. This field (default value = 0) specifies the user task the block is allocated to run in (1 - 4).

NOTE Certain normally read-write parameters in this block cannot be configured unless *Task* has first been set to a non-zero value.

Action (part 1 - visible). (Done/Supplying/Enter/Write) Displays one of four short text messages which, in combination with a message displayed by the second (invisible) part of the *Action* field immediately below it, enables a variety of instructions to be given to the operator.

‘Done’ appears when the block is in idle mode, i.e. before calibration has started or when calibration is complete. See next paragraph, *Action (part 2 - invisible)*.

Action (part 2 - invisible). ([blank]/High rng/Low rng/E2 data?) This (invisible) field supplies the second part of an operator instruction, e.g. **Supplying Low rng**. The operator performs the given instruction, and then selects *CONTINUE* in the *Status* parameter to tell the block that the instruction has been duly carried out. Refer to the section *Calibration procedures* for details of the calibration process.

The *Action* instructions are necessarily very brief, and it is assumed that the operator is referring to this manual for a full description of the required actions.

Value. When the operator is instructed to ‘Enter’ a value, it should be entered into this field.

Status. (WAITING/CONTINUE/PROCESSING/ABORT) Shows the operator the current status of the calibration process, and also allows him to abort it.

- **WAITING.** Shown whenever the AO_CALIB block is awaiting an action by the operator.
- **CONTINUE.** Must be entered by the operator when the awaited action is completed.
- **PROCESSING.** Displayed when the AO_CALIB block is in the process of carrying out an action.
- **ABORT.** Entering this field abandons the calibration process.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block's RAM data.
- **Failed.** Indicates an abort from the calibration process. This alarm's priority of 6 is automatically set by the block and means that, if it occurs, the alarm must be acknowledged before continuing.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

SiteNo. Specifies the address (1 - 8) on the T600 Series instrument's internal serial bus of the I/O module whose analogue output is being calibrated. The AO_CALIB block automatically copies this field from the corresponding one in the AN_OUT block when first attached, but *SiteNo* may be altered subsequently. Note that you cannot configure *SiteNo* if the *Task* parameter is still at its default value of zero.

Channel. Specifies the number (1 - 8) of the analogue output channel being calibrated. The AO_CALIB block automatically copies this field from the corresponding one in the AN_OUT block when first attached, but *Channel* may be altered subsequently. Note that you cannot configure *Channel* if the *Task* parameter is still at its default value of zero.

OutType. (Volts/mA) Specifies the type of analogue output to be calibrated. The AO_CALIB block automatically copies this field from the corresponding one in the AN_OUT block when first attached, but *OutType* may be altered subsequently.

NOTE The entire calibration process for a particular *OutType* must be run to completion for all supported ranges, or aborted. It is not possible to calibrate only some of the ranges.

Range. (0-1.25 V/0-2.5 V/0-5 V/0-10 V/0-2.5 mA/0-5 mA/0-10 mA/0-20 mA) Read-only field reporting which one of the specified channel's hardware ranges is currently being calibrated. See the note in the *OutType* section above.

AN_OUT. Specifies the name of the AN_OUT block to be attached to. Once a valid block name has been entered, the AO_CALIB block attaches to it, and forces it into CALIB mode.

AO Alarm. (Invalid/Bad Loop/In alarm/OK) Indicates the current state of the calibration block.

- **Invalid** means an invalid configuration for the channel, or no AN_OUT block found.
- **Bad Loop** means that the AN_OUT and AO_CALIB blocks are not running in the same user task ('loop'), which they must be for calibration to operate.
- **In alarm** means that the attached AN_OUT block is in alarm.
- **OK** is displayed in all other cases.

Calibration procedures

Caution

Only authorised personnel using equipment of an approved standard should carry out hardware re-calibration.

To start the calibration process, enter the 'parent' I/O block name in the AO_CALIB block's *AN_OUT* field. Assuming the parent block can be found, the calibration block attaches itself to it and then copies other fields as applicable, *SiteNo*, *Channel*, *OutType*, *Range*. The *Action* field reports **Done**, and the *Status* field reports **WAITING**. Begin the calibration process by setting *Status* to **CONTINUE**.

This section tabulates each of the commands given by the block and the actions required of the operator in response. *OutType* must be set to the required hardware type before the calibration process begins, as each type is calibrated as a separate operation.

High-level analogue output calibration sequence

Step	Action message	Action required of operator, before entering CONTINUE
1	Supplying Low rng	Measure the output voltage/current now being generated.
2	Enter Low rng	Enter into the Value field, the EXACT voltage or current as measured, in volts or mA.
3	Supplying High rng	Measure the output voltage/current now being generated.
4	Enter High rng	Enter into the Value field, the EXACT voltage or current as measured, in volts or mA.
5	<i>Steps 1 to 4 are repeated for each range supported by the connected hardware.</i>	
6	Write E2 data?	CONTINUE only if you want the calibration data updated in the I/O module. Stopping before this point abandons this calibration, with no changes to the setup of the unit.

MOD_UIO: MODULE INPUT/OUTPUT BLOCK

Block function

These blocks represent each of the hardware modules fitted in the base unit. This is because each MOD_UIO block defines the Module's type and position of a hardware module in the base unit, of the node indicated by the *Node* field.

Block parameters

Symbols used in *Table 134* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status								
Expect	Requested operating module type	Menu									
Node	LIN Node address of the base unit	Hex									
SiteNo	Module's position on the base unit	Integer									
Alarms											
Software	Block RAM data sumcheck error/network failure	T/F									
Hardware	Fault with I/O hardware	T/F									
Combined	Logical OR of all Alarm bits	T/F									
Actual	Actual Module fitted	Alphanumeric									
Version	Actual Module Version fitted										
Status		(ABC)D hex									
Missing	No I/O module at site	T/F	<table style="border-collapse: collapse;"> <tr><td style="border-right: 1px solid black; padding: 2px;">1</td><td style="padding: 2px;">D</td></tr> <tr><td style="border-right: 1px solid black; padding: 2px;">2</td><td style="padding: 2px;"></td></tr> <tr><td style="border-right: 1px solid black; padding: 2px;">4</td><td style="padding: 2px;"></td></tr> <tr><td style="border-right: 1px solid black; padding: 2px;">8</td><td style="padding: 2px;"></td></tr> </table>	1	D	2		4		8	
1	D										
2											
4											
8											
BadType	Incorrect module type fitted	T/F									
BadSite	SiteNo exceeds I/O capacity of the Node	T/F									
BadTask	Block allocated to an unsynchronised task	T/F									

Table 134 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these 'header' fields.

Expect. (None/AI2/AI3/AI4/AI8_TC/AI8_MA/AI8_RT/AI8_FMA/AO2/DI4/DI6_MV/DI6_HV/DI8_LG/DI8_LGv2/DI8_CO/DI8_COv2/DO4_LG/DO4_LGv2/DO4_24/DO4_24v2/DO8/RLY4/RLY4v2). This is the Module Type which is expected to be fitted at the specified *Node* and *SiteNo*. The type of available channel blocks and the block's allocated task are decided on selection of the Module Type. A Module block can be allocated to any task, but the 'BadTask' is set TRUE, if allocated to an unsynchronised task.

Node. Specifies the LIN node number, in hex format, of the base unit where the module is fitted. This is configurable for multiple nodes only.

NOTE Associated channel blocks will be orphaned if the Node number changes.

SiteNo. Specifies the position on the base unit where the I/O module corresponding to this MOD_UIO block is located.

NOTE An invalid *SiteNo* sets the *Status.Missing* bit setting the *Alarms.Hardware* bit TRUE.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block's RAM data.
- **Hardware.** A hardware alarm is generated when any *Status* bit is set TRUE, e.g. in the event of a fault in the I/O hardware the I/O module would be considered as missing.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

Actual. This is the actual Module Type fitted at the specified *Node* and *SiteNo*. *Unknown* indicates the I/O Module is not compatible with the current I/O subsystem configuration, e.g. when configuring a redundant T2550 subsystem,

particular Version 1 I/O Module hardware is not compatible. Version 2 I/O Module hardware must be fitted in a redundant subsystem, where appropriate.

Version. The version number of the module fitted at the selected *SiteNo*.

Status. Bitfield indicating general comms./hardware error conditions. The following information is given in addition to the summary in *Table 134*.

- **Missing.** If TRUE, the hardware is not located at address defined by *Node* and *SiteNo*, sets *Alarms.Hardware* TRUE.
- **BadType.** If TRUE expected and fitted modules are not compatible, sets *Alarms.Hardware* TRUE.
- **BadSite.** If TRUE specified *SiteNo* exceeds the total number of slots on the base unit, sets *Alarms.Hardware* TRUE.
- **BadTask.** If TRUE, the specified *Task Rate* is not synchronised to the I/O subsystem, sets *Alarms.Hardware* TRUE.

AI_UIO: ANALOGUE INPUT BLOCK

Block function

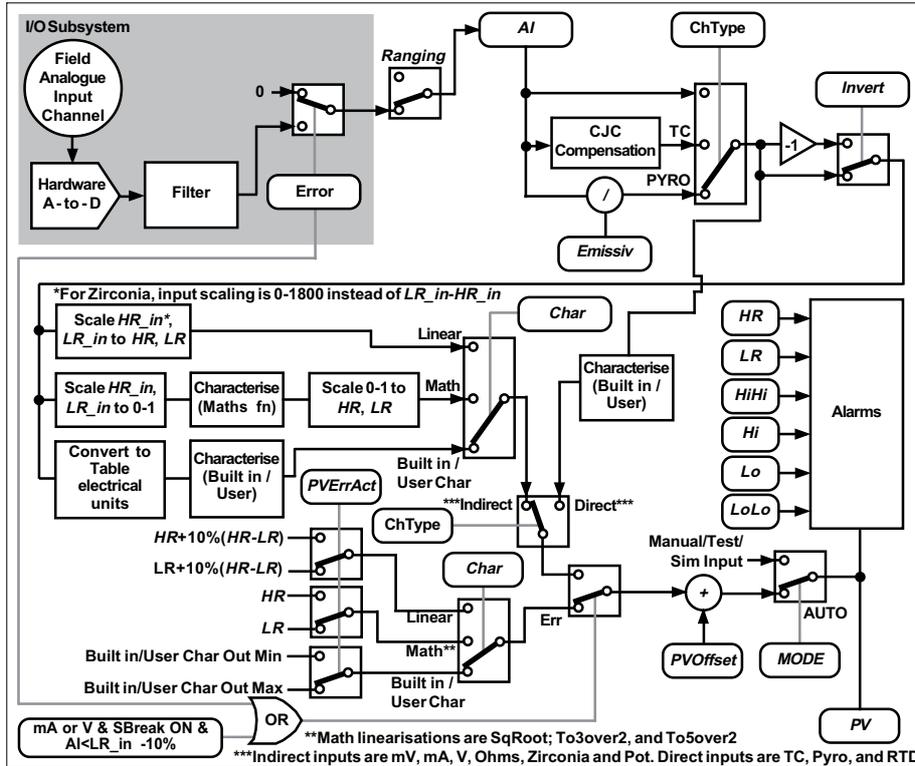


Figure 62 Block schematic

Please refer to the schematic in *Figure 62*. This block converts voltage, current, or resistance measurement units into floating point values in engineering units. In addition the block provides signal conditioning, calibration, filtering and alarm capability.

The function block can be used in a MANUAL mode where the process variable *PV* can be forced to a specified value for testing purposes. Normal operation of *PV* resumes when the function block is in AUTO mode.

NOTE Not all fields are applicable to every channel type, and are therefore ‘greyed out’ in LINTools configuration software for clarity.

Open-circuit detection/protection

Some I/O modules support break detection on their analogue inputs, while others support only break protection, see Tactician Series Instrument documentation for details.

Break detection. Modules supporting break detection detect a break in the input before any bad readings have been made. When detected the *Status.BrkDtctd* bit is set. The *Alarm.OCctDel* bit is also set, subject to any delay specified in the *Delay* field. Delay can be applied either at the start of the break (i.e. the alarm bit going TRUE is delayed), or at the end of the break (i.e. the alarm bit going FALSE is delayed), or both, using the *Options.OCDelSt* and *Options.OCDelEnd* bits, respectively.

The *Options.InitFilt* bit, when TRUE, causes the block’s first-order filter to be initialised whilst *Alarm.OCctDel* or *Status.BrkDtctd* is TRUE. This means that already-filtered bad readings do not corrupt the subsequent good readings. The *Alarm.OCctDel* bit does not need to be enabled for *InitFilt* to act.

The behaviour of *PV* under break conditions will depend on the value of *PVErrAct* (UP, DOWN or HOLD).

Caution

A delay may occur before a break is detected, which means that the value of *PV* held may differ from the correct value.

Break protection. In I/O modules only supporting break protection, a break causes *PV* values to drift either up or down. For break protection, sensor break (*SBreak*) actions can be configured to adopt an action as a result of a sensor break. *SBreak* UP initiates a *Drive high* action to the hardware, *SBreak* DOWN, a *Drive low* action, and *SBreak* NONE issues a take no action. The UP/DOWN directions are reversed if the *Options.Invert* bit is TRUE.

NOTE True Sensor Break on mA or V *InType* cannot be directly detected, as 0 (zero) mA or V is considered a valid measurement on bi-polar inputs.

Execution of input blocks

Plant input LIN function blocks are normally executed at the start of a user task, before any other blocks in the task. As a result, connections to an input block from other blocks in the same user task (e.g. feedback from an output block) transmit the values resulting from the *previous* execution of the user task. Input connections from other user tasks, in this or other nodes, are always executed before the current user task is run, and will result from a completed execution of the source user task.

Block parameters

Symbols used in *Table 135* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
MODE	Operating mode	Menu	
Fallback	Fallback operating mode	Menu	
PV	Process variable	Eng	
HR, LR	High & low range for PV	Eng	
HiHi, Hi, Lo, LoLo	High & low absolute alarm limits	Eng	
Hyst	Hysteresis bandwidth, applied inside all alarm levels	Eng	
Filter	First order filter time constant	Secs	
Char	Fixed characterisation	Menu	
UserChar	User characterisation	Block name	
AlmOnTim	Time Delay for Alarm On	Secs	
AlmOffTim	Time Delay for Alarm Off	Secs	
PVOffset	Simple offset	Eng	
Alarms			
Software	Block RAM data sumcheck error/network failure	T/F	
Hardware	Fault with I/O hardware	T/F	
HiHi	In high high alarm	T/F	
Hi	In high alarm	T/F	
Lo	In low alarm	T/F	
LoLo	In low low alarm	T/F	
OutOfRange	PV outside LR-HR range (>10%)	T/F	
PVError	PV determined by PVErrAct	T/F	
OCctDel	OpenCct detected (If H/W supports Break detection)	T/F	
CharErr	Characterisation table error, or limits exceeded	T/F	
NotAuto	Block not in normal operation	T/F	
ModBlock	Module block error	T/F	
Combined	Logical OR of all Alarm bits	T/F	
Node	LIN Node address of the base unit	Hex	
SiteNo	Module's position on the base unit	Integer	
Channel	Specifies channel number within I/O module (1-n)	Integer	
InType	Specifies input type	Menu	
HR_in, LR_in	Plant signal HR, LR equivalents		
AI	Unprocessed input		
Res	User defined Lead resistance for ohms type or Shunt value for mA type	Ohms	
CJ_type	Defines CJ_temp source	Menu	
CJ_temp	Cold junction temperature value		
LeadRes	Lead resistance	Ohms	
Emissiv	Pyrometer emissivity		
Delay	Delay before OCct alarm trips (If H/W supports Break detection)	Secs	
SBreak	Hardware action on sensor break	Menu	
PVErrAct	Action on out of range measured value	Menu	

Continued...

Parameter	Function	Units	Status
<i>Continued...</i>			
Options		(AB)CD hex	
Invert	Input signal inverter	T/F	1
InitFilt	Initialise filter whilst OCctDel OR BrkDtctd TRUE	T/F	2
OCDelSt	Apply O/C delay at start	T/F	4
OCDelEnd	Apply O/C delay at end	T/F	8
T/Absolute		T/F	1
SqrtOver	Allows over-range PV values for SqRoot, To3over2 & To5over2	T/F	2
			4
			8
Status		ABCD hex	
Missing	No valid I/O block found	T/F	1
BadType	No such channel on I/O module	T/F	2
Ranging	Channel initialising	T/F	4
BadSetup	Hardware capabilities exceeded	T/F	8
HwFlt		T/F	1
NotAuto	Instrument not operating in AUTO mode	T/F	2
UsrCalib	Channel has been calibrated by the user	T/F	4
CalEr	Corrupted measured value calibration data	T/F	8
AuxCalEr		T/F	1
RngEr	Input exceeds themeasurement circuit range	T/F	2
HwAuxFlt	Fault in I/O module auxiliary measurement	T/F	4
BrkWarn	RTD partial break	T/F	8
BrkDtctd	TRUE if open-cct. detected (If H/W supports Break detection)	T/F	1
AuxRngEr	Cj_temp exceeds the measurement circuit range	T/F	2
ManPCal	Manual Pot input type calibration	T/F	4
BadTask	Invalid task	T/F	8

Table 135 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

Mode. (AUTO/MANUAL/TEST/SIMULATE/CALIB). Current operating mode. AUTO normal block operation. MANUAL allows *PV* to be forced to a specified value at runtime and calculates alarms. TEST, enabled when *IO_Mode.EnaTest* in the Header block is set TRUE, functions the same as in Manual operation, but will cause all blocks currently in TEST to revert to the mode defined in *FallBack* when disabled. During SIMULATE, enabled when *IO_Mode.EnaSim* in the Header block is set TRUE, only the *Status* and *PV* fields are writeable. Alarms are calculated, but all other aspects of the block do not update. CALIB (calibration) mode cannot be directly selected, but is ‘forced’ when a CALIB_UIO block is attached. In CALIB mode all the fields that can affect the derivation of *AI* become read-only (*InType*, *SiteNo*, *Channel*, *HR_in*, *LR_in*). See CALIB_UIO block for details.

Fallback. (AUTO/MANUAL). Normal operating mode of the block. Updated if AUTO or MANUAL is selected in MODE field. MODE field adopts this value on deselection of CALIB mode and global exit of TEST or SIMULATE mode.

PV. Process Variable (processed input), i.e. block output. In AUTO mode the PV is derived from measured value, auxiliary value and status returned from I/O subsystem, and is Read-Only. In all other modes this field is writeable.

HR, LR. High and low range of *PV* in engineering units. An alarm is raised while *PV* is more than 10% outside the *HR-LR* range. *HR*, and *LR* are also used in defining scaling of inputs and/or the range over which a characterisation operates. The following table explains how the scaling, limits and hardware range selection relates to the configuration of the channel.

Input Type	Characterisation	Hardware Range Selection Criteria	Range of Characterisation		Rng Alm	Scaling
			Input	PV		
TC	User	X0 to X15 from the UCHAR block	X0 to X15 from the UCHAR block	Y0 to Y15 from the UCHAR block	Available subject to config	None
	Built in thermocouple characterisation	mV equivalent of characterisation table limits	mV equivalent of characterisation table limits	Characterisation table limits (see char)	Available subject to config	None
mV/V/mA	Linear	HR_in to LR_in	-	-	Available maps onto HR, LR	HR_in, LR_in
	Maths function	HR_in to LR_in	HR_in to LR_in	HR to LR	Not available	HR_in, LR_in maps onto HR, LR
	User	HR_in to LR_in	X0 to X15 from the UCHAR block	Y0 to Y15 from the UCHAR block	Available subject to config	None
RTD	Built in characterisation	HR_in to LR_in	HR_in to LR_in	HR to LR	Available subject to config	HR_in, LR_in maps onto HR, LR
	User	X0 to X15 from the UCHAR block	X0 to X15 from the UCHAR block	Y0 to Y15 from the UCHAR block	Available subject to config	None
Ohms	Built in RTD characterisation	ohm equivalent of characterisation table limits	ohm equivalent of characterisation table limits	Characterisation table limits (see char)	Available subject to config	None
	Linear	HR_in to LR_in	-	-	Available	HR_in, LR_in maps onto HR, LR
Pyrometer	Maths function	HR_in to LR_in	HR_in to LR_in	HR to LR	Not available	HR_in, LR_in maps onto HR, LR
	User	HR_in to LR_in	X0 to X15 from the UCHAR block	Y0 to Y15 from the UCHAR block	Available subject to config	None
	Linear	HR_in to LR_in	-	-	Available	HR_in, LR_in maps onto HR/emissivity, LR/emissivity
Potentiometer	User	HR_in to LR_in	X0 to X15 from the UCHAR block	Y0 to Y15 from the UCHAR block	Available subject to config	Input to character'tion scaled by 1/emissivity
	Linear	0 to 100%	-	-	Not available except on PVError action	0,100% maps onto HR, LR
	Maths function	0 to 100%	0 to 100%	HR to LR	Not available	0,100% maps onto HR, LR
Zirconia	User	0 to 100%	X0 to X15 from the UCHAR block	Y0 to Y15 from the UCHAR block	Available subject to config	None
	Linear	0 to 1800mV	-	-	Available	0,1800mV maps onto HR, LR
	User	0 to 1800mV	X0 to X15 from the UCHAR block	Y0 to Y15 from the UCHAR block	Available subject to config	None

Table 136 Scaling, limits and hardware range selection

In the table above:

- **Hardware Range Selection Criteria.** This is the range used to select the input channel's hardware configuration. For example, for a J-type thermocouple which has table limits of -8.096mV to 69.536mV, a -100mV to 100mV range might be selected by the hardware for best resolution over the range. Refer to Appendix D of *HA028898 - T2550 Eurotherm PAC Handbook*, and *HA030047 - T2750 Eurotherm PAC User Guide* for details of ranges available and corresponding input equivalent circuits.
- **Range of Characterisation.** This is the working range of the input imposed by processing of the input value, for example, characterisation table limits. Outside these limits, the value will be clipped and a CharErr alarm raised.
- **Range Alarm.** PV values outside HR +10%(HR-LR), LR-10%(HR-LR) will cause an out of range alarm. This alarm may not occur due to clipping of the input to the working range. When it is not possible for the range alarm to occur due to clipping to working range the table shows "Not Available" in the Range Alarm column. When range

alarm may not operate if the configuration is such that the clipping to the limits of the linearisation table precludes it, then the column is marked “Available subject to config”.

NOTE. The *Options.Invert* bit affects the derivation of *PV*. See *Options* below.

Hi, Lo. High and Low absolute alarm limits applied to *PV*. If a limit is crossed, the appropriate alarm bit (*Hi* or *Lo*) is asserted.

HiHi, LoLo. HighHigh and LowLow absolute alarm limits applied to *PV*. If a limit is crossed, the appropriate alarm bit (*HiHi* or *LoLo*) is asserted.

Hyst. Hysteresis bandwidth value, applied inside all alarm levels.

Filter. Specifies the time constant (0-1000 seconds) of a simple first-order filter applied to the value in *PV*. Set *Filter* equal to zero to disable the filter.

Char. (Linear/User/B/C/D/E/G/J/K/L/N/R/S/T/U/NiMoNiCo/PL2/NiNiMo/PtRe24/MoRe/Cu10/Cu53/PT100/PT100a/Jpt100/PT1000/Ni100/Ni120/SqRoot/t038/To3over2/To5over2). *Char* selects a characterisation type to be applied to the AI to produce the *PV*, see the following table for details. This block automatically adjusts the standard tables to suit the selected temperature units of the block.

	Type	Range (°C)	Accuracy(°C)
Linear	Linear input		
User	Custom downloaded table.		
B	B type thermocouple ^[1]	0 to 1820	
C	C type thermocouple	0 to 2314	0.12
D	D type thermocouple	0 to 2495	0.08
E	E type thermocouple	-270 to 1000	0.03
G2	G2 type thermocouple	0 to 2314	0.07
J	J type thermocouple	-210 to 1200	0.02
K	K type thermocouple	-270 to 1372	0.04
L	L type thermocouple	-200 to 900	0.02
N	N type thermocouple	-270 to 1300	0.04
R	R type thermocouple	-49.9 to 1767	0.04
S	S type termocouple	-50 to 1767	0.04
T	T type thermocouple	-270 to 400	0.02
U	U type thermocouple	-199.9 to 599.8	0.08
NiMoNiCo	Ni 18% Molybdenum vs Ni 0.9% Cobalt	-49.99 to 1410	0.06
PL2	Platinel type thermocouple	0 to 1369.6	0.02
NiNiMo	Ni/NiMo thermocouple	0 to 1405.8	0.14
PtRe24	Pt20%Rh/Pt40%Rh thermocouple	0 to 1888	0.08
MoRe	MoRe thermocouple	0 to 1990	0.32
Cu10	Cu10 resistance thermometer	-19.9 to 249.9	0.02
Cu53	Cu53 resistance thermometer	-69.999 to 200	0.01
PT100	Pt100 platinum resistance thermometer	-200 to 850.0	0.01
PT100a	Pt100 platinum resistance thermometer	-199.9 to 629.8	0.09
Jpt100	Pt100 J platinum resistance thermometer	-200 to 630	0.01
PT1000	Pt1000 platinum resistance thermometer	-199.9 to 850	0.01
Ni100	Ni100 resistance thermometer	-59.9 to 249.9	0.01
Ni120	Ni120 resistance thermometer	-59.9 to 249.9	0.01
SqRoot	Square root		
t038	Bucose W5%Re/W26%Re thermocouple	0 to 2000	
To3over2	Power 3/2		
To5over2	Power 5/2		

^[1] Measurements in the range 0-250°C are not recommended, see NOTE below.

Table 137 Characterisation table

NOTE The linearisation for the B Type Thermocouple has been specifically designed to provide a safety margin at lower temperatures. This ensures that large temperature changes in the range 0-250°C will not occur if the Voltage is only very slightly adjusted, e.g. .1mV.

UserChar. Only available if *Char* is User. Specifies the name of a UCHAR block defining a 16-point (x,y) custom linearisation/characterisation table to be applied to *AI*. This block applies no temperature base correction to the UCHAR table, so it must be defined in units that match the prevailing temperature base.

PVOffset. A single offset added equally over the whole range of the input.

Alarms. See *Appendix D* page 545 for a general description of the Alarms field.

- **Software.** Sumcheck error in block's RAM data.
- **Hardware.** A hardware alarm is generated when the *Status.HwFlt*, *Status.HwAuxFlt*, or *Status.BadTask* bit is set TRUE, e.g. in the event of a fault in the I/O hardware.
- **HiHi, LoLo.** These alarm limits are set by parameters *HiHi*, and *Hi* fields.
- **Hi, Lo.** These alarm limits are set by parameters *Lo*, and *LoLo* fields.
- **PVError.** The PV has been determined from *PVErrAct* field. The PV error action is performed for the following reasons:
Status.RngEr or
(PV > PVErrValue & PVErrAct = Up) or
(PV < PVErrValue & PVErrAct = Down) or
Status.BadSetup or
Status.HwFault or
Status.CalEr or
Status.BrkDtctd or
Status.HwAuxFlt or
Status.AuxCalEr or
(SBreak != None & (Type=mA or V) & AO < LR_in - 10%(HR_in - LR_in)) or
*Status.Missing** or
*Status.BadType** or
*Status.BadTask** or
CharErr alarm

*Prior to Tick 7, Missing, BadType and BadTask caused the input to default to 0.

- **OutOfRange.** This is tripped if *PV* is greater than or equal to *HR* +10% of range or *PV* less than or equal to *LR* -10% of range.
 NOTE. It may not occur if the input is limited by the Characterisation table limits.
- **OCctDel.** 'Open-circuit delay'. With *Options.OCDelSt* TRUE, this alarm trips after a delay specified by the *Delay* parameter, if an open-circuit is detected in the block input from the field. With *Options.OCDelSt* FALSE, the alarm trips without delay. Similarly, with *Options.OCDelEnd* set TRUE, *Options.OCctDel* resets *Delay* seconds after the open-circuit input has been restored. With *Options.OCDelEnd* FALSE, the alarm resets immediately, see *Delay*.
 NOTE Use of burden resistors may affect open-circuit detection. Refer to the instrument product manual for details.
- **CharErr.** The selected characterisation is not supported for this input type or the cold junction temperature or the input exceeds the range of this table.
- **NotAuto.** TRUE if operating in MANUAL, TEST, SIMULATE or CALIB mode.
- **ModBlock.** TRUE if there is no corresponding MOD_UIO block.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

Node. Specifies the LIN node number in hex format of the base unit where the module is fitted. The *Node* and *SiteNo* identifies the MOD_UIO block to which this channel block applies. The channel block must be allocated to the same task as its' accompanying MOD_UIO block.

SiteNo. Specifies the position on the Base Unit where the I/O module corresponding to this block is located. The *Node* and *SiteNo* identify the MOD_UIO block to which this channel block applies. The channel block must be allocated to the same task as its associated MOD_UIO block. An invalid *SiteNo* sets the *Status.Missing* bit.

Channel. Defines the channel number within the I/O module specified by the *SiteNo* parameter, to which this AI_UIO block is attached. The number of channels available depends on the I/O module type. An invalid *Channel* sets the *Status.BadType* bit.

InType. (Off/TC/PYRO/RTD2/RTD3/RTD4/Pot/Zirconia/mV/V/mA/Ohms2/Ohms3/Ohms4) Specifies input type. **Off** indicates that the input type is not yet defined. **TC** (Thermocouple), **PYRO** (Optical Pyrometer), and **Zirconia** (Zirconia probe). The options **RTD2**, **RTD3**, and **RTD4**, support 2-, 3-, and 4-wire resistance thermometers, respectively. **Pot** indicates a potentiometer input, and automatically sets *Status.ManPCal* TRUE. **mV** (millivolt) measurements, Consult the I/O module documentation for specific limits. *Volts* configures the block as a high-level voltage input. For milliamps inputs, select mA. The options **Ohms2**, **Ohms3**, and **Ohms4**, support 2-, 3-, and 4-wire resistance measurements (ohms), respectively. Consult the I/O module documentation for specific limits.

NOTE Where certain options are not available on certain channels, the *Status.BadSetup* bit is set.

HR_in, LR_in. High and low range of the measured input signal, in units appropriate to the currently selected *InType* (V, mV, Ohm, and mA). *HR_in* and *LR_in* are not used for Potentiometer or Zirconia inputs as the measured range is fixed at 0-100% and 0-1800mV respectively, or for direct inputs, i.e. TC, RTD2, RTD3, and RTD4, as the measurement range is derived from the input range of the characterisation. Refer to Appendix D of *HA028898 - T2550 Eurotherm PAC Handbook*, and *HA030047 - T2750 Eurotherm PAC User Guide* for details of ranges available and corresponding input equivalent circuits.

HR_in and *LR_in* are also used in the scaling of the linear inputs, maths function inputs and characterised inputs on the mV, mA, Ohm, and Pyrometer ranges, see *HR, LR* section for scaling details.

AI. The unprocessed input value, from which *PV* is derived if operating in AUTO mode. This value will be updated if operating in CALIB mode, but *PV* will not. Holds the last value while the channel is initialising.

Res. Field depending on the *InType* setting. *mA InType* setting supports configurability of the shunt resistor. Lead Resistance is supported via the *Ohms2* and *RTD2 InType* setting, *Ohms3*, *Ohms4*, *RTD3* and *RTD4* supports the fallback lead resistance when the lead resistance cannot be measured.

CJ_type. (Auto/UsrSet/OFF). *Auto* indicates the temperature is measured by a sensor on the terminal unit used to compensate for the cold junction. *UsrSet* allows the user defined value set in the *CJ_temp* field to define the external reference temperature.

CJ_temp. Cold junction temperature, only valid in the ‘Thermocouple’ input modes (*TC*). With *CJ_type* set to *AUTO*, *CJ_temp* is read-only and reports the temperature of the I/O module connector block as measured by the built-in sensor (if present). With *CJ_type* set to *UsrSet*, a known cold junction temperature can be written in *CJ_temp*, or input a value via a connection from the Strategy.

NOTE. All temperatures are reported, and must be entered, in the units jointly specified by the *TAbsolut* parameter, and the corresponding header block’s *IP_Type* parameter. That is: °C, °F, K, or R.

NOTE. Where certain options are not available on certain channels, the *Status.BadSetup* bit sets.

LeadRes. For RTDs being used in 3- or 4-wire modes, *LeadRes* is read-only and is the Lead Resistance currently being measured. Should a ‘partial break’ to 2-wire operation occur, the input continues to function, and *Status.BrkWarn* is asserted and the value entered in *Res* is used as the Lead Resistance.

Emissiv. This applies to *PYRO InType* only. It is the correction factor applied to compensate for the emissivity of the object whose temperature is currently being measured.

Delay. With *Options.OCDelSt* TRUE, *Delay* specifies the delay (secs) before the *Alarms.OCDel* bit is tripped after detection of an open-circuit in the field input, and also, with *Options.OCDelEnd* TRUE, the delay before the alarm is reset after restoration of the input. Making *Options.OCDelSt* and/or *Options.OCDelEnd* FALSE cancels these delays independently. *Figure 63* shows schematically how *Delay* is implemented. The *Delay* facility is useful when short-term interruptions of the field input are to be ignored as ‘noise’, and also when restoration of an open-circuit input is liable to be ‘noisy’.

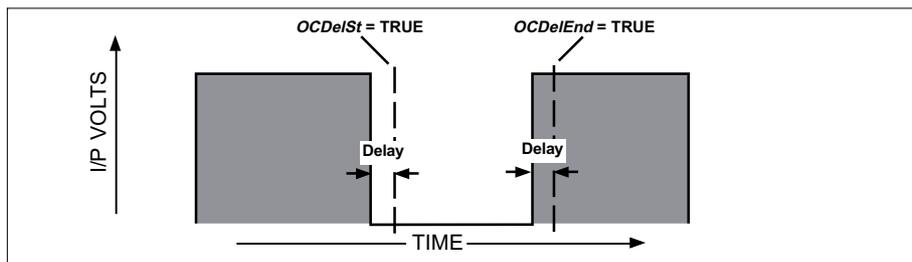
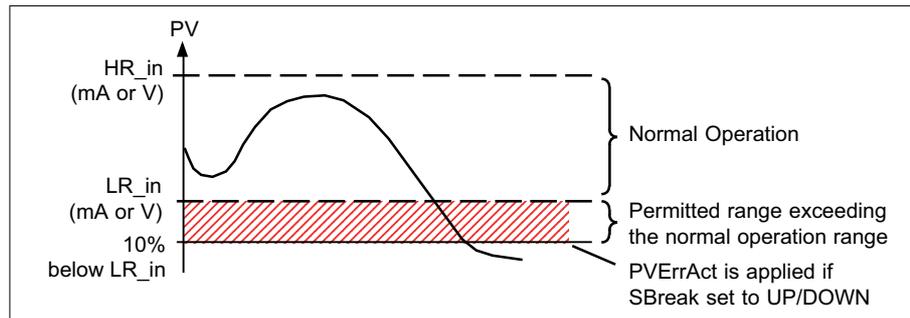


Figure 63 Delay parameter implementation (schematic)

SBreak. (UP/DOWN/DETECT/NONE). Only actions supported by the hardware are available. UP configures a *Drive high* hardware action on a sensor break. DOWN configures a *Drive low* hardware action. DETECT configures the action as determined by *PVErrAct* on a sensor break. NONE configures the take no hardware action instruction.

NOTE True Sensor Break on mA or V *InType* cannot be directly detected, as 0 (zero) mA or V is considered a valid measurement on bi-polar inputs. *Figure 64* is an example of the response to a measurement more than 10% below the *LR_in* value of the normal operating range. Selection of the *SBreak* UP/DOWN operation has no effect on the I/O hardware, however, the resulting action of the block is derived from *PVErrAct* and will be implemented when the I/O subsystem returns a measured value more than 10% below *LR_in* if *SBreak* is configured to either UP or DOWN. Under these circumstances, a *PVErr* alarm is raised but *Status.BrkDtctd* is not set.

**Figure 64 mA or V InType response example**

PVErrAct. (UP/DOWN/HOLD). Specifies how PV is derived when a good reading is not available from the I/O (that is, when it is not GOOD or CALIBRATING). UP sets PV to the top of the range defined in the following table when there is an error. DOWN sets PV to the bottom of the range defined in the following table when there is an error. HOLD causes PV to freeze when a break condition is detected (option only available if hardware supports break detect).

For Linear inputs, the PV error values cause a Range alarm when the PV Error Action is taken. For UCHAR and built in characterisations, the PV error value is set at the limit of the table and therefore the Range alarm only operates on PV error if HR/LR are at least 10% inside the limits of the characterisation. For maths characterisations, the range alarm is never raised under PV error conditions.

Input Type	Characterisation	Range Alarm	PV Error Action Value
TC	User	Available subject to config	Y15/Y0
	Built in thermocouple characterisations	Available subject to config	TC Table Range High/Low
mV/V/mA	Linear	Available	$HR+10\%(HR-LR) / LR-10\%(HR-LR)$
	Maths function	Not available	HR/LR
RTD	User	Available subject to config	Y15/Y0
	Built in characterisation	Available subject to config	TC Table Range High/Low
Ohms	User	Available subject to config	Y15/Y0
	Built in characterisation	Available subject to config	TC Table Range High/Low
Pyrometer	Linear	Available	$HR+10\%(HR-LR) / LR-10\%(HR-LR)$
	User	Available subject to config	Y15/Y0
Potentiometer	Linear	Not available except on PVError action	$HR+10\%(HR-LR) / LR-10\%(HR-LR)$
	Maths function	Not available	HR/LR
Zirconia	User	Available subject to config	Y15/Y0
	Linear	Available	$HR+10\%(HR-LR) / LR-10\%(HR-LR)$
	User	Available subject to config	Y15/Y0

Table 138 PV Error Action Values

In the table on the previous page, the Range Alarm column options are:

- **Available.** Ranging alarms always function.
- **Not available.** Ranging alarms never function, even on *PVErr* action.

- **Available subject to config.** Ranging alarm operation depends on the configuration.
- **Not available except on PVError action.** For potentiometer input with no characterisation, normal operation can never cause a range error because of clipping. If there is a fault, a Ranging alarm is raised because the PVError value will be sufficient to cause a Range alarm.

Options. Bitfield selecting a variety of signal-processing options. The following information is given in addition to the summary in *Table 135*.

- **Invert.** If TRUE, this has the effect of mapping *HR* to *LR_in*, and *LR* to *HR_in*, i.e. of inverting the input signal. See above, in the *HR, LR* section.
- **InitFilt.** If TRUE, it causes the block's first-order filter to be initialised whilst *Alarms.OCctDel* or *Status.BrkDtctd* is TRUE. This means that already-filtered bad readings do not corrupt the subsequent good readings.

NOTE The *Alarms.OCctDel* bit does not need to be enabled for *InitFilt* to act.

- **OCDelSt.** If TRUE, this has the effect of delaying the starting of an open circuit alarm, subject to the delay specified in the *Delay* field, i.e. the alarm bit going TRUE is delayed, see *Figure 63*.
- **OCDelEnd.** If TRUE, this has the effect of delaying the ending of an open circuit alarm, subject to the delay specified in the *Delay* field, i.e. the alarm bit going FALSE is delayed, see *Figure 63*.
- **TAbsolut.** Used in conjunction with *IP_type.Imperial* field of the associated header block to select this block's prevailing temperature units. The header block globally selects the Imperial (°F/R) or SI (°C/K) system, but individual AI_UIO blocks can specify either absolute (K/R) or relative (°C/°F) temperature units within that system. The temperature units apply to *CJ_temp*, and to any *Char* characterisation, i.e. the fixed tables. No temperature base corrections are applied to user-specified characterisations in UCHAR blocks.
- **SqrtOver.** Only used when the analogue input characterisation, *Char*, is set to *SqRoot*, *To3over2* or *To5over2*. When *Options.SqrtOver* is TRUE, *PV* can exceed the range between *LR* and *HR*, and therefore allow the "10% out of range" alarm to be raised (*Alarms.OutRange*). If *Options.SqrtOver* is FALSE, *PV* is limited to a range between *LR* and *HR* and cannot exceed *HR*. If, however, *PV = HR*, then a "PV error" alarm is raised (*Alarms.PVError*) as it could be indicative of an over-ranged *PV* value.

Status. Bitfield indicating general comms./hardware error conditions. The following information is given in addition to the summary in *Table 135*.

- **Missing.** If TRUE, the module block is missing or assigned to an unsupported task.
- **BadType.** If TRUE, the channel number is greater than the number of AI channels on the module, or the channel has already been allocated on another channel block. Sets *Alarms.Hardware* TRUE.
- **Ranging.** If TRUE, the *Channel* is currently initialising.
- **BadSetup.** If TRUE, an invalid Setup for the connected I/O module is detected, e.g. if the user configured input range exceeds the instrument input range.
- **HwFlt.** If TRUE, a fault is detected in the I/O module, sets *Alarms.Hardware* TRUE.
- **NotAuto.** If TRUE, the I/O module is not operating in AUTO mode, sets *Alarms.NotAuto* TRUE.
- **UsrCalib.** If TRUE, the user re-calibration constants are in use, FALSE if factory calibration constants are in use.
- **CalEr.** If TRUE, corrupt measured value calibration data is detected.
- **AuxCalEr.** If TRUE, corrupt auxiliary value calibration data is detected.
- **RngEr.** If TRUE, the input value is not measurable by the hardware, sets *Alarms.PVError* TRUE.

NOTE When this block is used in an AI3 Module, an AI (unprocessed input) value that exceeds the limits of the range may not cause *Status.RngEr* to set TRUE.

- **HwAuxFlt.** If TRUE, a fault is detected in the I/O module auxiliary measurement.
- **BrkWarn.** If TRUE, a 'partial break' to 2-wire operation is detected. The input continues to function, and the value entered in *Res* is used as the Lead Resistance.

- **BrkDtctd.** If TRUE, a break in the input is detected. Breaks can be detected before any bad readings have been made if the first order filtering has been initialised.
- **AuxRngEr.** If TRUE, the *CJ_temp* value exceeds the measurement circuit range.
- **ManPCal.** If TRUE, it indicates the values of *HR* and *LR* have been set manually, and resets FALSE on successful completion of an automated Pot calibration process via a connected LOOP_PID block. Only used when *ChType* shows **Pot**, in all other input types it is always FALSE.
- **BadTask.** If TRUE, the block is not on the same task as the parent module block (MOD_UIO).

AO_UIO: ANALOGUE OUTPUT BLOCK

Block function

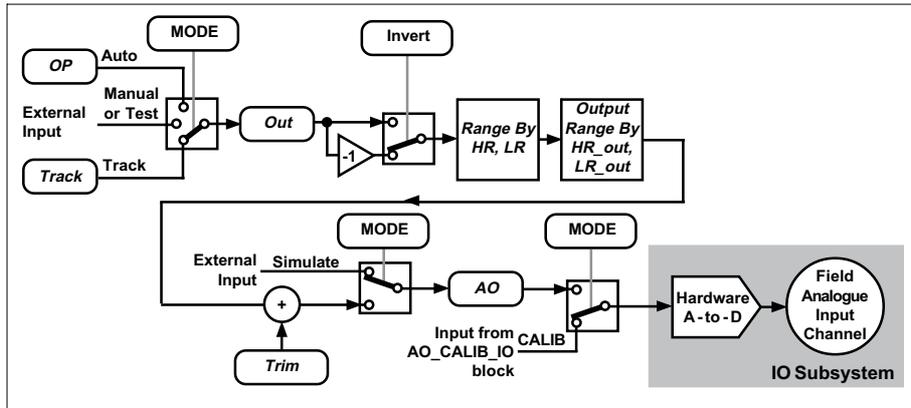


Figure 65 Block schematic

Please refer to *Figure 65*. The analogue output block converts ranged analogue variables (floating point numbers) from a Strategy into voltage or current outputs (software selectable). The block provides Auto/Manual/Track control, Inversion and Alarms.

Outputs from different instruments can be cross-coupled in redundant configurations to enhance the level of systems integrity. In such configurations a ‘master’ output normally controls the plant; if the master instrument fails, a ‘slave’ automatically takes over. Please refer to the specific hardware documentation for details.

NOTE Not all fields are applicable to every channel type, and are therefore ‘greyed out’ in LINTools configuration software for clarity.

Block parameters

Symbols used in *Table 139* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
MODE	Operating mode	Menu	
Fallback	Fallback operating mode	Menu	
OP	Requested output	Eng	
HR, LR	High & low range for OP	Eng	
Out	Actual output	Eng	
Track	Track output value	Eng	
Trim	Offset trim	mA/V	
Alarms			
Software	Block RAM data sumcheck error/network failure	T/F	
Hardware	Fault with I/O hardware / fault with transmitter PSU	T/F	
CctFault	Output is open / short circuit	T/F	
OvrDrive	Output being overdriven	T/F	
BadDmnd	A non-numeric demand signal has been encountered	T/F	
NotAuto	Block not in normal operation	T/F	
ModBlock	Module block error	T/F	
Combined	Logical OR of all Alarm bits	T/F	
Node	LIN Node address of base unit	Hex	
SiteNo	Module's position on base unit	Integer	
Channel	Specifies channel number within I/O module (1-n)	Integer	
OutType	Specifies output type	Menu	
HR_out, LR_out	Plant signal HR, LR equivalents	mA/V	
AO	Electrical output	mA/V	

Continued...

Parameter	Function	Units	Status
<i>Continued...</i>			
Options		(ABC)D hex	
Invert	Output signal inverter	T/F	D
SelTrack	Select a Track mode, Out = Track.	T/F	
Status		(A)BCD hex	
Missing	No valid I/O block found	T/F	D
BadType	No such channel on I/O module	T/F	
Ranging	Channel initialising	T/F	
BadSetup	Hardware capabilities exceeded	T/F	
HwFlt	Fault in I/O module	T/F	C
NotAuto	Instrument not operating in AUTO mode	T/F	
UsrCalib	Channel was calibrated by the user	T/F	
CalEr	Corrupted calibration data	T/F	
FaultCct	Open / short circuit	T/F	B
HiClip	Clipped high	T/F	
LoClip	Clipped low	T/F	
OvrDrive	Output being overdriven	T/F	
BadDmnd	A non-numeric demand signal has been encountered	T/F	A
BadTask	Invalid task	T/F	

Table 139 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D* page 544 for details of these ‘header’ fields.

Mode. (AUTO/MANUAL/TEST/TRACK/SIMULATE/CALIB). Current operating mode. AUTO is the normal operating mode of the block. MANUAL allows *Out* to be forced to a specified value at runtime. TEST, enabled when *IO_Mode.EnaTest* in the Header block is set TRUE, functions the same as in Manual operation, but will cause all blocks currently in TEST to revert to the mode defined in *FallBack* when disabled. In TRACK mode (*SelTrack* TRUE), *Out* tracks the value in the *Track* parameter. SIMULATE, enabled when *IO_Mode.EnaSim* in the Header block is set TRUE, functions the same as in Manual operation, but will ignore *Status*. CALIB (calibration) mode cannot be directly selected, but is ‘forced’ when an CALIB_UIO block attaches to the AO_UIO block. In CALIB mode all the fields that can affect the derivation of *Out* become read-only (*Mode*, *SelTrack*, *Invert*, *OutType*, *SiteNo*, *Channel*, *HR_out*, *LR_out*), see CALIB_UIO block.

Fallback. (AUTO/MANUAL). Read Only field, updated whenever AUTO or MANUAL is selected in the *Mode* field. The *Mode* field adopts this value on deselection of TRACK or CALIB mode and global exit of TEST or SIMULATE modes.

OP. Requested output. Input to block from strategy, and the source of *Out* in Auto mode.

NOTE. If the *BadDmnd* alarm bit is set TRUE, the electrical output (*AO* parameter) is set to 0.

HR, LR. High and low range of *Out* in engineering units. *HR* and *LR* define two points on a linear engineering units scale that map to points *HR_out* and *LR_out*, respectively, on a linear (volts or milliamps) scale. This pair of scales is used by the block to derive the output signal from *Out*. If inversion is not being applied (*Invert* FALSE), the equivalent conversion equation is:

$$\text{Output} = LR_out + \left[\frac{Out - LR}{HR - LR} \times (HR_out - LR_out) \right] + \text{Trim}$$

With *Invert* TRUE, the conversion used is:

$$\text{Output} = HR_out - \left[\frac{Out - LR}{HR - LR} \times (HR_out - LR_out) \right] + \text{Trim}$$

Out. Analogue output in engineering units. Output from *OP* or *Track* or directly set by the User before inversion (if specified) and de-ranging.

Track. Source of *Out* in Track mode.

Trim. An offset trim in electrical units.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block's RAM data.
- **Hardware.** A hardware alarm is generated when the *Status.HwFlt* or *Status.BadTask* bit is set TRUE, e.g. in the event of a fault in the I/O hardware.
- **CctFault.** Circuit fault. Reported when the module detects an open circuit (on current outputs), or a short circuit (on voltage outputs).
- **OvrDrive.** Output overdrive condition. This alarm reports when the output signal is being overdriven by another output module. Used in redundant Strategy.
- **BadDmnd.** TRUE if a non-numeric internal (to the block) or external (*OP* parameter, for example) demand signal has been encountered.
- **NotAuto.** TRUE if operating in MANUAL, TEST, SIMULATE or CALIB mode.
- **ModBlock.** TRUE if there is no corresponding MOD_UIO block.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

Node. Specifies the LIN node number in hex format of the base where the module is fitted. The *Node* and *SiteNo* identify the MOD_UIO block to which this channel block applies. The channel block must be allocated to the same task as its accompanying MOD_UIO block.

SiteNo. Specifies the position on the Base Unit where the I/O module corresponding to this block is located. The *Node* and *SiteNo* identify the MOD_UIO block to which this channel block applies. The channel block must be allocated to the same task as its associated MOD_UIO block. An invalid *SiteNo* sets the *Status.Missing* bit.

Channel. Defines the channel number within the I/O module specified by the *SiteNo* parameter, to which this AO_UIO block is attached. The number of channels available depends on the I/O module type. An invalid *Channel* sets the *Status.BadType* bit.

OutType. (OFF/V/mA). Selects voltage or current output type for *HR_out*, *LR_out*, *Out*, and *Trim*.

Consult the I/O module documentation for specific limits.

NOTE. Where certain options are not available on certain channels, the *Status.BadSetup* bit is set.

HR_out, LR_out. High and low range of the output signal, in units appropriate to the currently selected *OutType* (V or mA). *HR_out* and *LR_out* define two points on a linear scale that map to points *HR* and *LR*, respectively, on a linear engineering units *Out* scale. Nominally, when *Out* equals *HR*, the real output is at the value in *HR_out*; when at *LR*, the real output equals *LR_out*. Please refer to the section on *HR*, *LR* above for details.

NOTE. The entered values of *HR_out* and *LR_out* are compared with the list of available hardware ranges supported by the I/O module, and chooses the 'best fit' range to maximise the resolution of the output. Scaling to the exact values of *HR_out* and *LR_out* is done in software.

AO. Actual Output (as opposed to the demanded output *OP* or *Track* value) from the block before any inversion and de-ranging. This is derived from *Out*, unless in SIMULATE mode, when it can be set by the user.

Options. Bitfield selecting a variety of signal-processing options. The following information is given in addition to the summary in *Table 139*.

- **Invert.** If TRUE, this has the effect of mapping *HR* to *LR_out*, and *LR* to *HR_out*, i.e. of inverting the output signal. See above, in the *HR*, *LR* section.
- **SelTrack.** TRUE selects Track mode. *Out* is derived from the *Track* parameter.
- **CPUFILO.** If TRUE the I/O Module outputs *LR_out* on the loss of communications with higher level software.

NOTE Only offered if communications failure detection is supported by the I/O Module.

Status. Bitfield indicating general comms./hardware error conditions. The following information is given in addition to the summary in *Table 139*.

- **Missing.** If TRUE, the module block is missing or assigned to an unsupported task.

- **BadType.** If TRUE expected and fitted modules are not compatible, sets *Alarms.Hardware* TRUE.
- **Ranging.** If TRUE, the *Channel* is currently initialising.
- **BadSetup.** If TRUE, invalid *Setup* for the connected I/O module.
- **HwFlt.** If TRUE, a fault has been detected in the I/O module, sets the *Alarms.Hardware* field TRUE.
- **NotAuto.** If TRUE, the I/O module is not operating in AUTO mode, sets the *Alarms.NotAuto* TRUE.
- **UsrCalib.** If TRUE, the user re-calibration constants are in use, FALSE if factory calibration constants are in use.
- **CalEr.** If TRUE, corrupt calibration data has been detected.
- **FaultCct.** If TRUE, an open / short circuit has been detected, sets the *Alarms.CctFault* field TRUE.
- **HiClip.** If TRUE, *AO* exceeds the range of the hardware and has been clipped at the high end of the hardware range.

NOTE This applies to the range which the hardware is operating, not the *LR_out* - *HR_out* range.

- **LoClip.** If TRUE, *AO* exceeds the range of the hardware and has been clipped at the low end of the hardware range.
NOTE This applies to the range which the hardware is operating, not the *LR_out* - *HR_out* range.
- **OvrDrive.** If TRUE, *AO* exceeds the range of the hardware, sets *Alarms.OvrDrive* TRUE. Used in a redundant Strategy.
- **BadDmnd.** If TRUE, a non-numeric internal (to the block) or external (*OP* parameter, for example) demand signal has been encountered.
- **BadTask.** If TRUE, the block is not on the same task as the parent module block (MOD_UIO).

CALIB_UIO: ANALOGUE INPUT/OUTPUT CALIBRATION BLOCK

Block function

This block is used to calibrate analogue input and output channels on modules that support user calibration. A database is created that contains the AI_UIO or AO_UIO block associated with the I/O module to be calibrated, plus an CALIB_UIO block. The calibration block effectively ‘attaches’ itself to the named AI_UIO or AO_UIO block, forces it into CALIB mode, then works through it to calibrate the specified I/O channel. Via its *Action* field(s), the calibration block displays one of a sequence of instructions for the operator to perform, who then sets the *State.CONTINUE* field when ready to move on to the next instruction.

Details of how to calibrate analogue modules are given in the *Calibration procedures* section.

Block parameters

Symbols used in *Table 140* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Action	Displays instructions to operator	Menu	
(invisible field)	Displays instructions to operator	Menu	
CalValue	Nominal calibration value of calibration point	Float	
Value	Operator-entered values as requested by ‘Action’	Float	
State	Current status of calibration process (& abort I/P)	Menu	
Target	Channel config. valid or channel block in alarm	Menu	
Alarms			  
Software	Block RAM data sumcheck error/network failure	T/F	
Failed	Calibration process aborted (default priority=6)	T/F	
Combined	Logical OR of all Alarm bits	T/F	
Node	LIN Node address of the base unit	Hex	
Type	Calibrated channel type	Menu	
SiteNo	Module’s position on the base unit	Integer	
Channel	Specifies channel number within I/O module	Integer	
RangeNo	Hardware range selected for calibration	Integer	
LowLim	Low limit of range selected	Float	
HighLim	High limit of range selected	Float	
Units	Units of selected calibration range	Menu	

Table 140 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

Action (part 1 - visible). (Done/Supplying/Apply/Enter/Save/Sure?) Displays one of six text messages which, in combination with a message displayed by the second (invisible) part of the *Action* field immediately below it, enables a variety of instructions for the operator.

‘Done’ appears when the block is in idle mode, i.e. before calibration has started or when calibration is complete. See next paragraph, *Action (part 2 - invisible)*.

Action (part 2 - invisible). (Value/VRef/Calib?) This (invisible) field supplies the second part of an operator instruction, e.g. *Calib?*. The operator performs the given instruction, and then selects *CONTINUE* in the *State* parameter to tell the block that the instruction has been duly carried out, see *Calibration procedures*.

The *Action* instructions are necessarily very brief, and it is assumed that the operator is referring to this manual for a full description of the required actions.

CalValue. It is the responsibility of the user to ensure that sensible CalValues are used.

- **Calibrating an output.** This specifies the nominal output value produced when calibrating the current calibration point, e.g. the T2550 AO2 module has two calibration points for each range as follows.

RangeNo	OutType	LowLim	HiLim	Units	CalValue Low point	CalValue High point
1	Volts	-500.0	10500.0	mV	1000.0	9000.0
2	mA	-0.1	20.5	mA	2.0	18.0

Table 141 Output range details

- **Calibrating an input.** This field displays the write protected nominal calibration point value for the point and RangeNo being calibrated. For example, the T2550 AI2 module has two calibration points for each range, as shown in *Table 142*.

NOTE During AI2 Module resistance range calibration, lead resistance cannot be fully compensated. Lead resistance should be as low as practicably possible, and MUST never be greater than 5 Ohms.

RangeNo	OutType	LowLim	HiLim	Units	CalValue Low point	CalValue High point
1	Volts	-500.0	10500.0	mV	1000.0	9000.0
2	mA	-0.1	20.5	mA	2.0	18.0
AI2 - Channel 1						
1	Volts	-10300.0	10300.0	mV	2000.0	8000.0
2	mV	-150.0	150.0	mA	20.0	80.0
3	Ohms4	0.0	464.0	Ohms	200.0	400.0
4	Ohms4	0.0	7000.0	Ohms	1200.0	4800.0
AI2 - Channel 2						
1	Volts	-10300.0	10300.0	mV	2000.0	8000.0
2	mV	-150.0	150.0	mV	20.0	80.0
3	Zirconia	0.0	1800.0	mV	800.0	1200.0
4	Ohms4	0.0	464.0	Ohms	200.0	400.0
5	Ohms4	0.0	7000.0	Ohms	1200.0	4800.0
AI3 - Channel 1, 2, & 3						
1	mA	-28.0	28.0	mA	4.0	16.0
AI4 - Channel 1, & 3						
NOTE. CalValues for Channels 2 and 4 are taken from Channels 1 and 3.						
1	mV	-150.0	150.0	mV	20.0	80.0

Table 142 Input range details

Value. When instructed to ‘Enter’ a value, it must be entered in this field.

State. (WAITING/CONTINUE/PROCESSING/ABORT/RESTORE/DISCONNECT) Shows the operator the current status of the calibration process, allows it to be aborted and returned to a defined state.

- **WAITING.** Shown whenever the CALIB_UIO block is awaiting an action by the operator.
- **CONTINUE.** Must be entered by the operator when the awaited action is completed.
- **PROCESSING.** Displayed when the CALIB_UIO block is in the process of carrying out an action.
- **ABORT.** Entering this field abandons the calibration process.
- **RESTORE.** Selected to restore factory calibration settings.
- **DISCONNECT.** Selected to remove the channel block from the Calibration Mode.

Target. (Invalid/Bad task/In Alarm/OK) Specifies the current state of the calibration block. Invalid indicates that the calibration block detected an invalid configuration or channel block not found. *Status.BadTask* specifies that the CALIB_UIO block and the channel blocks are not running in the same task. In Alarm indicates the the attached block is in alarm. OK indicates Calibration block is OK.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.
- **Failed.** Indicates an abort from the calibration process. This alarm’s priority of 6 is automatically set by the block and means that, if it occurs, the alarm must be acknowledged before continuing.

- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.
- Node.** Specifies the LIN node number of the base where the module is fitted in hex format.
NOTE Associated channel blocks will be orphaned if the Node number changes.
- Type.** (Input/Output) Specifies whether the channel to be calibrated is an input or an output channel.
- SiteNo.** Specifies the position on the base of the I/O module whose analogue input or output is being calibrated. The *SiteNo* cannot be configured if the *Task* parameter is at its default value of zero.
- Channel.** Specifies the number (1 - 4) of the analogue channel being calibrated. Not all channels can be calibrated, for example the AI4 module, only channels 1 and 3 can be calibrated, as channels 1, 2 and 3, 4 share the same calibration data.
- RangeNo.** (Menu derived from the attached analogue block) This field reflects which one of the specified channel's hardware ranges is currently being calibrated. The initial value is derived from the *OutType* or *InType* of the analogue block to which it is attached. Each separate I/O module type that can be calibrated has a value of *RangeNo* corresponding to the lowest range of that type. If the I/O module type has more than one range, the higher ranges can be selected by increasing the value of *RangeNo*.
- LowLim.** Reports the low limit of the currently selected calibration range.
- HighLim.** Reports the high limit of the currently selected calibration range.
- Units.** Shows the units of measurement for the currently selected calibration range. It is not addressable via LINOPC.

Calibration procedures

Caution

Only authorised personnel using equipment of an approved standard should carry out hardware re-calibration.

To start the calibration process, set the *SiteNo*, *Channel* and *Type* fields to specify the I/O Module to be calibrated. Assuming the parent block can be found, the calibration block attaches itself to it and then derives other fields as applicable, *LowLim*, *HighLim* and *CalValue*. The *Action* field reports **Done**, and the *State* field reports **WAITING**. The *RangeNo* must then be set to select the required range, as each type is calibrated as a separate operation.

NOTE Some modules, such as the T2550 AI3 and AI4 modules, only have a single range to be calibrated.

Analogue Input Calibration

This section tabulates each of the commands given by the block and the actions required of the operator in response.

Analogue input calibration sequence

Step	Action field message	Action required of operator, before entering CONTINUE
1	Enter VRef*	If the module has a reference voltage that requires measuring and entering, measure and enter the value of the Voltage Reference.
2	Apply Value	Apply to the input the calibration point value as given in the <i>CalValue</i> field. The <i>State</i> field will finally display WAITING , after first showing the PROCESSING indication.
3	Enter Value*	If the module allows calibration of points that deviate from the nominal value, enter into the <i>Value</i> field, the EXACT value applied.
4	<i>Steps 2 and 3 are repeated for each calibration point supported by the connected hardware</i>	
5	Save Calib?	At this point the parent block <i>AI</i> field shows the effect of the new calibration. Select CONTINUE only if you want the calibration data updated in the I/O module. Stopping before this point abandons this calibration, with no changes to the setup of the unit. When all ranges are calibrated set <i>State</i> field to DISCONNECT allowing selection of a different module or channel.

Start the calibration process by setting the *State* to **CONTINUE**. The *Action* field will read **Apply Value** or **Enter VRef** while the *CalValue* field shows the low calibration point.

NOTE. *, Steps ignored if not applicable for the type of input module being calibrated.

Analogue Output Calibration

Step	Action field message	Action required of operator, before entering CONTINUE
1	Supplying Value	Measure the output voltage/current now being generated.
2	Enter Value	Enter into the Value field, the EXACT voltage or current as measured, in volts or mA. If the module allows calibration of points that deviate from the nominal value, CalValue can be changed causing the electrical output value to change.
3	<i>Steps 1 and 2 are repeated for each calibration point supported by the connected hardware.</i>	
4	Supplying Value	Measure the output voltage / current now being generated. This output now shows the effect of the new calibration allowing you to decide whether or not to update the calibration data.
5	Save Calib?	Select CONTINUE only if you want the calibration data updated in the I/O module. Stopping before this point abandons this calibration, with no changes to the setup of the unit. When all ranges are calibrated set State field to DISCONNECT allowing selection of a different module or channel.

This section tabulates each of the commands given by the block and the actions required of the operator in response.

Analogue output calibration sequence

Start the calibration process by setting the *State* to CONTINUE. The Action field will read *Supplying Value* while the *CalValue* field shows the low calibration point.

Restore Factory Calibration

Step	Action field message	Action required of operator, before entering CONTINUE
1	Sure?	Select CONTINUE only if you want the calibration data for all ranges on the selected channel to be restored to the factory settings.

Start the process by setting the State to RESTORE.

DI_UIO: DIGITAL INPUT BLOCK

Block function

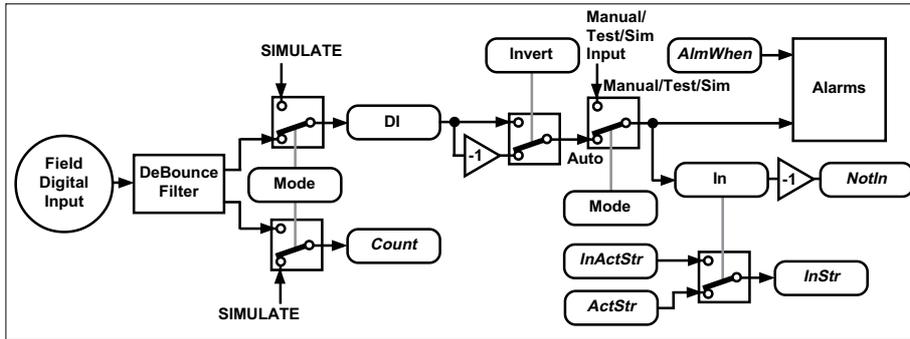


Figure 66 Block schematic

Please refer to *Figure 66*. This block allows digital signals to be input to the Strategy, and provides Auto/Manual control, inversion, and debounce. The input can be configured as either contact or voltage sensing.

NOTE Not all fields are applicable to every channel type, and are therefore ‘greyed out’ in LINTools configuration software for clarity.

Block parameters

Symbols used in *Table 143* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
MODE	Current operating mode	Menu	
Fallback	Fallback operating mode	Menu	
InactStr	8-character string copied to InStr if In is FALSE	Alphanumeric	
ActStr	8-character string copied to InStr if In is TRUE	Alphanumeric	
AlmWhen	In state causing Input alarm	Menu	
Thresh	Threshold voltage for module	Integer	
Debounce	Channel debounce period	Integer	
Count	Number of debounced pulses	Long	
Alarms			
Software	Block RAM data sumcheck error	T/F	
Hardware	I/O module failure	T/F	
Input	In state = AlmWhen	T/F	
NotAuto	Block not in normal operation	T/F	
ModBlock	Module block error	T/F	
Combined	Logical OR of all Alarm bits	T/F	
Node	LIN Node address of the base unit	Hex	
SiteNo	Module's position on the base unit	Integer	
Channel	Specifies channel number within I/O module (1-n)	Integer	
DI	Plant input signal	T/F	
In	Input state as Boolean	T/F	
NotIn	Inverse of In	T/F	
InStr	Input state as String	Alphanumeric	
Options		(A)BCD hex	
Invert	Input signal inverter	T/F	
Status	Comms/hardware status	(A)BCD hex	
Missing	No valid I/O block found	T/F	
BadType	No such channel on I/O module	T/F	
Ranging	Channel initialising	T/F	
BadSetup	Hardware capabilities exceeded	T/F	

Continued...

Parameter	Function	Units	Status
<i>Continued...</i>			
HwFlt	Fault in I/O module	T/F	C
NotAuto	Instrument not operating in AUTO mode	T/F	
GoneTrue	In changed, False to True	T/F	
GoneFals	In changed, True to False	T/F	
PulsTrue	Input has experienced a rising edge since last task cycle	T/F	B
PulsFals	Input has experienced a falling edge since last task cycle	T/F	
			A
BadTask	Invalid task	T/F	

Table 143 Block parameters

Block specification menu

The following is given in addition to *Table 143*.

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

MODE. (AUTO/MANUAL/TEST/SIMULATE). Current operating mode. AUTO normal block operation. MANUAL allows *In* to be forced to a specified value at runtime and calculates alarms. TEST, enabled when *IO_Mode.EnaTest* in the Header block is set TRUE, functions the same as in Manual operation, but will cause all blocks currently in TEST to revert to the mode defined in *FallBack* when disabled. SIMULATE, enabled when *IO_Mode.EnaSim* in the Header block is set TRUE, functions the same as in Manual operation, but only the *Status* and *In* fields are writeable. Alarms are calculated, but all other aspects of the block do not update.

Fallback. (AUTO/MANUAL). Normal operating mode of the block. Updated if AUTO or MANUAL is selected in MODE field. MODE field adopts this value on global exit of TEST or SIMULATE mode.

ActStr, InactStr, InStr. *InactStr* and *ActStr* define a pair of 8-character (max) strings that correspond with the FALSE and TRUE states, respectively, of the *In* parameter. The *InStr* parameter adopts the string value corresponding to the current state of *In*.

AlmWhen. (Never/FALSE/TRUE) Specifies the *In* state that will raise an *Input* alarm. ‘Never’ means that the *Input* alarm is never raised.

In. Input state as a boolean, wireable to the Strategy. In MANUAL, TEST and SIMULATE mode, *In* can be written to and wired into. In AUTO, *In* can only be wired out of.

NotIn. The inverse of *In*, above.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.
- **Hardware.** A hardware alarm is generated when the *Status.HwFlt*, or *Status.BadTask* bit is set TRUE, e.g. in the event of a fault in the I/O hardware.
- **Input.** Raised when the *In* state equals the state specified in *AlmWhen*. If ‘Never’ has been specified, this alarm is never raised.
- **NotAuto.** TRUE if operating in MANUAL, TEST, or SIMULATE mode.
- **ModBlock.** TRUE if there is no corresponding MOD_UIO block
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

Node. Specifies the LIN node number in hex format of the base where the module is fitted. The *Node* and *SiteNo* identify the MOD_UIO block to which this channel block applies. The channel block must be allocated to the same task as its accompanying MOD_UIO block.

SiteNo. Specifies the position on the Base Unit where the I/O module corresponding to this block is located. The *Node* and *SiteNo* identify the MOD_UIO block to which this channel block applies. The channel block must be allocated to the same task as its associated MOD_UIO block. An invalid *SiteNo* sets the *Status.Missing* bit.

Channel. Defines the channel number within the I/O module specified by the *SiteNo* parameter, to which this DI_UIO block is attached. The number of channels available depends on the I/O module type. An invalid *Channel* sets the *Status.BadType* bit.

DI. Debounced input state (Plant signal) as a boolean, wireable to the Strategy. In the event of a module failure, this field sets *False*.

Thresh. Threshold Voltage. Defines threshold for voltage input option. Any values exceeding the I/O Range are clipped to the limits of the target hardware.

Debounce. Debounce time, in seconds. This is the minimum time a change in the digital input signal must persist before the *In* parameter is allowed to switch to the new state. The filtering action of *Debounce* is shown in *Figure 66*. This value can be changed to reflect the actual debounce time if the module cannot achieve the requested time.

Count. Number of low to high transitions on the input.

Options. Bitfield selecting a variety of signal-processing options. The following information is given in addition to the summary in *Table 143*.

■ **Invert.** If TRUE, this has the effect of inverting the input signal, i.e. *In* is the inverse *DI*.

Status. Bitfield indicating general comms./hardware error conditions. Please refer to *Table 143* for details.

■ **Missing.** If TRUE, the module block is missing or assigned to an unsupported task.

■ **BadType.** If TRUE expected and fitted modules are not compatible, sets *Alarms.Hardware* TRUE.

■ **Ranging.** If TRUE, the *Channel* is currently initialising.

■ **BadSetup.** If TRUE, an invalid *Setup* for the connected I/O module.

■ **HwFlt.** If TRUE, a fault has been detected in the I/O module, sets *Alarms.Hardware* TRUE.

■ **NotAuto.** If TRUE, the I/O module is not operating in AUTO mode, sets *Alarms.NotAuto* TRUE.

■ **GoneTrue.** If TRUE, the Input state, *In*, has changed from *FALSE* to *TRUE*.

NOTE Not evaluated until a complete cycle after the first successful reading from the I/O subsystem.

■ **GoneFals.** If TRUE, the Input state, *In*, has changed from *TRUE* to *FALSE*.

NOTE Not evaluated until a complete cycle after the first successful reading from the I/O subsystem.

■ **PulsTrue/PulsFals.** If TRUE, a rising or falling edge was detected in the last task cycle.

NOTE The following table shows how *GoneTrue*, *GoneFals*, *PulsTrue*, *PulsFals* are derived.

	In	GoneTrue	GoneFals	PulsTrue	PulsFals
	T	T	F	T	F
	F	F	F	T	T
	F	F	F	F	F
	F	F	T	F	T
	T	F	F	T	T
	T	F	F	F	F

■ **BadTask.** If TRUE, the block is not on the same task as the parent module block (MOD_UIO).

DO_UIO: DIGITAL OUTPUT BLOCK

Block function

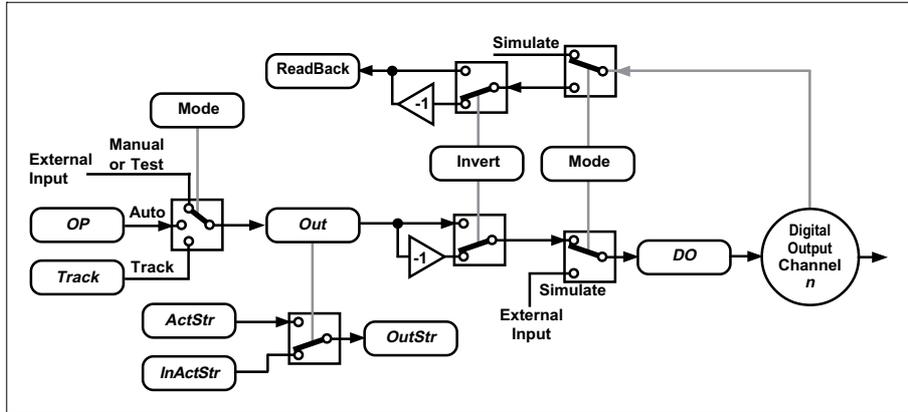


Figure 67 Block schematic

Please refer to the schematic in *Figure 67*, illustrating the digital output channels. The Digital output block transfers a logic signal to a digital output channel. The block provides Auto/Manual/Track control, inversion and discrepancy checking of output feedback.

NOTE Not all fields are applicable to every channel type, and are therefore ‘greyed out’ in LINtools configuration software for clarity.

Block parameters

Symbols used in *Table 144* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
MODE	Current operating mode	Menu	
FallBack	Fallback operating mode	Menu	
OP	Output state from Strategy	T/F	
Track	Track output state	T/F	
Out	Output state (before inversion)	T/F	
Out Str	Output state as a string	Alphanumeric	
InactStr	8-character string copied to OutStr if Out = FALSE	Alphanumeric	
ActStr	8-character string copied to OutStr if Out = TRUE	Alphanumeric	
Alarms			
Software	Block RAM data sumcheck error	T/F	
Hardware	I/O module failure or transmitter PSU faults	T/F	
CctFault	RealOut/readback discrepancy detected	T/F	
NotAuto	Block not in normal operation	T/F	
ModBlock	Module block error	T/F	
Combined	Logical OR of all Alarm bits	T/F	
Node	LIN Node address of the base unit	Hex	
SiteNo	Module's position on the base unit	Integer	
Channel	Specifies channel number within I/O module (1-n)	Integer	
DO	Plant output signal	T/F	
Pullup	High logic status output voltage	Volts	
Readback	Feedback of low-level I/O module output state	T/F	
Options		(ABC)D hex	
Invert	Output signal inverter	T/F	
SelTrack	Select a Track mode, Out = Track	T/F	

Continued...

- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

Node. Specifies the LIN node number in hex format of the base where the module is fitted. The *Node* and *SiteNo* identify the MOD_UIO block to which this channel block applies. The channel block must be allocated to the same task as its accompanying MOD_UIO block.

SiteNo. Specifies the position on the Base Unit where the I/O module corresponding to this block is located. The *Node* and *SiteNo* identify the MOD_UIO block to which this channel block applies. The channel block must be allocated to the same task as its associated MOD_UIO block. An invalid *SiteNo* sets the *Status.Missing* bit.

Channel. Defines the channel number within the I/O module specified by the *SiteNo* parameter, to which this DO_UIO block is attached. The number of channels available depends on the I/O module type. An invalid *Channel* sets the *BadType* status bit.

Pullup. Specifies the output voltage used for high logic status, if supported by the hardware.

Readback. Indicates feedback of low-level I/O module output state, if supported by the hardware. If the *Options.Invert* field is TRUE the actual state is inverted before being displayed in *Readback*. If the output state does NOT correspond to the *Out* field, the *Alarms.CctFault* sets TRUE, raising an alarm.

Options. Bitfield selecting a variety of signal-processing options. The following information is given in addition to the summary in *Table 144*.

- **Invert.** If TRUE, *Out* is inverted before transmission to the I/O subsystem, i.e. of inverting the output signal.
- **SelTrack.** TRUE selects Track mode. *Out* is derived from the *Track* parameter.

Status. Bitfield indicating general comms./hardware error conditions., see *Table 144* for details.

- **Missing.** If TRUE, the module block is missing or assigned to an unsupported task.
- **BadType.** If TRUE expected and fitted modules are not compatible, sets *Alarms.Hardware* TRUE.
- **Ranging.** If TRUE, the *Channel* is currently initialising.
- **BadSetup.** If TRUE, an invalid Setup for the connected I/O module.
- **HwFlt.** If TRUE, the a fault has been detected in the I/O module, sets *Alarms.Hardware* TRUE.
- **NotAuto.** If TRUE, the I/O module is not operating in AUTO mode, sets *Alarms.NotAuto* TRUE.
- **BadTask.** If TRUE, the block is not on the same task as the parent module block (MOD_UIO).

TPO_UIO: TIME PROPORTIONING OUTPUT BLOCK

Block function

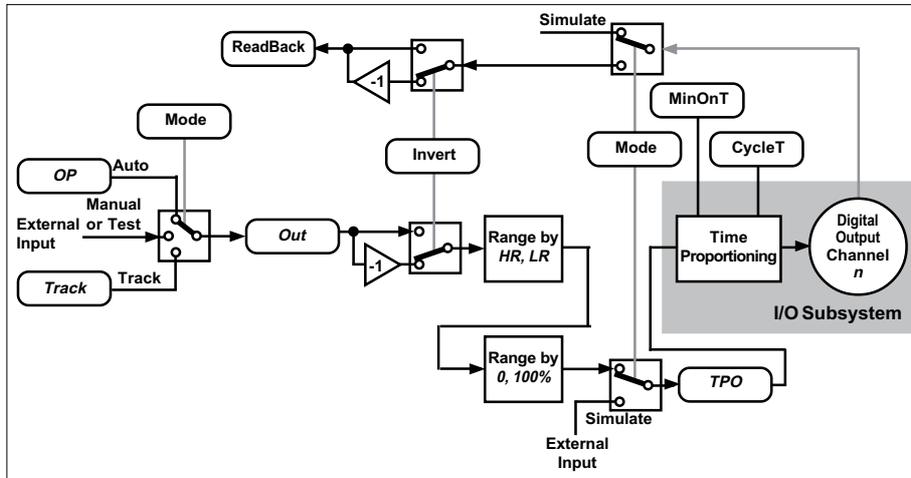


Figure 68 Time-Proportioning Block schematic

Please refer to the schematic in *Figure 68*, illustrating one of the eight digital channels. The Digital output block transfers one logic signal to a digital output channel. The block provides Auto/Manual/Track control, inversion and discrepancy checking of output feedback.

Block parameters

Symbols used in *Table 145* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
MODE	Current operating mode	Menu	
FallBack	Fallback operating mode	Menu	
OP	Output state from Strategy	Eng	
HR, LR	Range of units in Engineering units	Eng	
MinOnT	Minimum pulse on / off duration	Secs	
CycleT	Cycle time duration	Secs	
Out	Output (before scaling inversion)	Eng	
Track	Track output	Eng	
Alarms			
Software	Block RAM data sumcheck error	T/F	
Hardware	I/O module failure or transmitter PSU faults	T/F	
BadDmnd	A non-numeric demand signal has been encountered	T/F	
NotAuto	Block not in normal operation	T/F	
ModBlock	Module block error	T/F	
Combined	Logical OR of all Alarm bits	T/F	
Node	LIN Node address of the base unit	Hex	
SiteNo	Module's position on the base unit	Integer	
Channel	Specifies channel number within I/O module (1-n)	Integer	
TPO	Plant output signal	%	
Pullup	High logic status output voltage	Volts	
Readback	Feedback of low-level I/O module output state	T/F	
Options		(ABC)D hex	
Invert	Output signal inverter	T/F	<input type="checkbox"/>
SelTrack	Select a Track mode, Out = Track	T/F	<input type="checkbox"/>
		<input type="checkbox"/> 1 <input type="checkbox"/> 2 <input type="checkbox"/> 4 <input type="checkbox"/> 8	D
Status	Comms/hardware status	(AB)CD hex	
Missing	No valid I/O block found	T/F	<input type="checkbox"/>
BadType	No such channel on I/O module	T/F	<input type="checkbox"/>
Ranging	Channel initialising	T/F	
BadSetup	Hardware capabilities exceeded	T/F	
		<input type="checkbox"/> 1 <input type="checkbox"/> 2 <input type="checkbox"/> 4 <input type="checkbox"/> 8	D

Parameter	Function	Units	Status
<i>Continued...</i>			
HwFlt	Fault in I/O module	T/F	C
NotAuto	Instrument not operating in AUTO mode	T/F	
Clipped	TPO outside LR-HR range	T/F	
			B
			A
BadDmnd	A non-numeric demand signal has been encountered	T/F	
BadTask	Invalid task	T/F	

Table 145 Block parameters

Block specification menu

Dbase, Block, Type. See Appendix D page 544 for details of these ‘header’ fields.

Mode. (AUTO/MANUAL/TEST/TRACK/SIMULATE). Current operating mode. AUTO is the normal operating mode of the block. MANUAL allows *Out* to be forced to a specified value at runtime. TEST, enabled when *IO_Mode.EnaTest* in the Header block is set TRUE, functions the same as in Manual operation, but will cause all blocks currently in TEST to revert to the mode defined in *FallBack* when disabled. In TRACK mode (*SelTrack* TRUE), *Out* tracks the value in the *Track* parameter. SIMULATE, enabled when *IO_Mode.EnaSim* in the Header block is set TRUE, functions the same as in Manual operation, but only the *Status* and *TPO* fields are writeable. Alarms are calculated, but all other aspects of the block do not update.

Fallback. (AUTO/MANUAL). The normal operating mode of the block. Updated when AUTO or MANUAL is selected in the MODE field. MODE field adopts this value on deselection of TRACK mode (*SelTrack* FALSE), and global exit of TEST or SIMULATE mode.

OP. Requested output. Input to the block from the Strategy, and the source of *Out* in Auto mode. This field indicates the actual output channel state if *Options.Invert* is False, but if *Options.Invert* is TRUE an inverted form of the output channel state is displayed.

NOTE. If the *BadDmnd* alarm bit is set TRUE, the plant output signal (*TPO* parameter) is set to 0.

HR, LR. High and low range of *Out* and *OP* in engineering units. *HR* and *LR* define two points on a linear engineering units scale that normally map to 0% and 100%, respectively. This pair of scales is used by the block to derive the output signal from *Out*.

The equivalent conversion equation is:

$$TPO = \left[\frac{Out - LR}{HR - LR} \times 100 \right]$$

With *Invert* TRUE, the conversion used is:

$$TPO = 100 - \left[\frac{Out - LR}{HR - LR} \times 100 \right]$$

MinOnT. The Minimum On / Off Time in seconds. It defines the minimum duration of the On and OFF pulse and is used to minimise the frequency of the switch on and switch off. This value determines how accurately a device can be positioned. The shorter the time, the more precise the control. However, if the time is set too short, process noise will cause an excessively busy valve.

CycleT. The cycle time of the output in seconds.

NOTE Only offered if comms failure detection is supported by the I/O Module.

Out. The status (before inversion) of the field output connection. Sourced from *OP*, *TRACK* or directly set by user depending on current operating mode.

Track. Source of *Out* in TRACK mode.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block's RAM data.
- **Hardware.** A hardware alarm is generated when the *Status.HwFlt*, or *Status.BadTask* bit is set TRUE, e.g. in the event of a fault in the I/O hardware.
- **BadDmnd.** TRUE if a non-numeric internal (to the block) or external (*OP* parameter, for example) demand signal has been encountered.
- **NotAuto.** TRUE if operating in MANUAL, TEST, TRACK, or SIMULATE mode.
- **ModBlock.** TRUE if there is no corresponding MOD_UIO block.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

Node. Specifies the LIN node number in hex format of the base where the module is fitted. The *Node* and *SiteNo* identify the MOD_UIO block to which this channel block applies. The channel block must be allocated to the same task as its' accompanying MOD_UIO block.

SiteNo. Specifies the position on the Base Unit where the I/O module corresponding to this block is located. The *Node* and *SiteNo* identify the MOD_UIO block to which this channel block applies. The channel block must be allocated to the same task as its associated MOD_UIO block. An invalid *SiteNo* sets the *Status.Missing* bit.

Channel. Defines the channel number within the I/O module specified by the *SiteNo* parameter, to which this block is attached. The number of channels available depends on the I/O module type. An invalid *Channel* sets the *Status.BadType* bit.

TPO. Specifies the % demand being sent to the module driver.

Pullup. Specifies the output voltage used for high logic status, if supported by the hardware.

Readback. Indicates feedback of low-level I/O module output state, if supported by the hardware. If the *Options.Invert* field is TRUE the actual state is inverted before being displayed in *Readback*.

Options. Bitfield selecting a variety of signal-processing options. The following information is given in addition to the summary in *Table 145*.

- **Invert.** If TRUE, this has the effect of mapping *Out* to *DO*, i.e. of inverting the output signal.
- **SelTrack.** TRUE selects Track mode. *Out* is derived from the *Track* parameter.

Status. Bitfield indicating general comms./hardware error conditions, see *Table 145* for details.

- **Missing.** If TRUE, the module block is missing or assigned to an unsupported task.
- **BadType.** If TRUE expected and fitted modules are not compatible, sets *Alarms.Hardware* TRUE.
- **Ranging.** If TRUE, the *Channel* is currently initialising.
- **BadSetup.** If TRUE, an invalid *Setup* for the connected I/O module.
- **HwFlt.** If TRUE, a fault has been detected in the I/O module, sets *Alarms.Hardware* TRUE.
- **NotAuto.** If TRUE, the I/O module is not operating in AUTO mode, sets *Alarms.NotAuto* TRUE.
- **Clipped.** If TRUE, the *TPO* value has been clipped to be within the range 0 - 100%.
- **BadDmnd.** If TRUE, a non-numeric internal (to the block) or external (*OP* parameter, for example) demand signal has been encountered.
- **BadTask.** If TRUE, the block is not on the same task as the parent module block (MOD_UIO).

Block parameters

Symbols used in *Table 146* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
MODE	Operating mode	Menu	
Fallback	Fallback operating mode	Menu	
PV	Processed input	Eng1	
HR, LR	High & low range for PV	Eng1	
HiHi, Hi	High & High high absolute alarm limits	Eng1	
Lo, LoLo	Low & Low low absolute alarm limits	Eng1	
Hyst	Hysteresis Bandwidth	%	
Filter	First order filter time constant	Secs	
UserChar	User characterisation	Block name	
CutOff	PV low threshold	Eng1	
Default	PV default value (PV < CutOff)	Eng1	
Count	Scaled pulse count	Eng2	
Carry	Carry flag	T/F	
Overflow	Overflow flag	T/F	
SetCount	Start/End count value	Eng2	
Scale	Pulse/Frequency scale factor		
Control		Bitfield	
Init	Counter initialise	T/F	
Disable	Stop count at end value	T/F	
Clear0v	Overflow flag clear	T/F	
UpCount	Up/down count select	T/F	
InputOff	Halt count immediately	T/F	
AlmOnTim	Alarm ON time delay	Secs	∅ ∅ ∅ ∅ ∅ ∅ ∅
AlmOffTim	Alarm OFF time delay	Secs	
Alarms			
Software	Block data sumcheck error/network failure	T/F	
Hardware	Fault with I/O hardware	T/F	
HiHi	In high high alarm	T/F	
Hi	In high alarm	T/F	
Lo	In low alarm	T/F	
LoLo	In low low alarm	T/F	
OutOfRange	PV outside LR-HR range (>10%)	T/F	
CutOff	Measured frequency below CutOff Value	T/F	
CctFault	Volt-free contact open or short circuit	T/F	
CharErr	Characterisation table error, or limits exceeded	T/F	
TaskRate	Actual Task rate slower than time permitted by Instrument	T/F	
NotAuto	TRUE if not in normal operation	T/F	
ModBlock	Module block error	T/F	
Combined	OR-ing of all Alarms bits	T/F	
Node	LIN Node address of the base unit	Hex	
SiteNo	Module's position on the base unit	Integer	
Channel	Specifies channel number within I/O module (1-n)	Integer	
InType	Specifies input type	Menu	
HR_in, LR_in	Frequency I/P high & low range	Hz	
FI	Unprocessed frequency input	Hz	
Thresh	Trigger threshold	mA/Off/V	
Burden	Burden resistor value (Units are InType dependant)	Ohms	
Debounce	Debounce filter select for Logic I/P	mS	
PSU	Sensor power supply voltage	Volts	
Options		(ABCD) hex	
ScaleUp	If TRUE Scale is a multiplier, FALSE = divisor	T/F	
SBreak	Sensor break enable	T/F	
ShortCct	Short circuit detect enable	T/F	
Status		ABCD hex	
Missing	No valid I/O block found	T/F	
BadType	Incorrect module type fitted	T/F	
Ranging	Channel initialising	T/F	
BadSetup	Hardware capabilities exceeded	T/F	

Continued...

Parameter	Function	Units	Status
<i>Continued...</i>			
HwFlt	Fault in I/O module	T/F	C
NotAuto	Instrument not operating in AUTO mode	T/F	
OverRng	Input value is above the measurement circuit range	T/F	
UnderRng	Input value is below the measurement circuit range	T/F	
OpenCct	Volt-free contact open circuit	T/F	B
ShortCct	Volt-free contact short circuit	T/F	
BadHwSet	Hardware and requested configuration incompatible	T/F	
CutOff	Measured frequency is below CutOff Value	T/F	
			A
BadTask	Invalid task	T/F	

Table 146 Block parameters

Block specification menu

Dbase, Block, Type. See Appendix D page 544 for details of these ‘header’ fields.

MODE. (AUTO/MANUAL/TEST/SIMULATE). Current operating mode. AUTO normal block operation. MANUAL allows *PV* and *Count* to be forced to a specified value at runtime. TEST, enabled when *IO_Mode.EnaTest* in the Header block is set TRUE, functions the same as in Manual operation, but will cause all blocks currently in TEST to revert to the mode defined in *FallBack* when disabled. SIMULATE, enabled when *IO_Mode.EnaSim* in the Header block is set TRUE, functions the same as in Manual operation, but only the *Status* and *PV* fields are writeable. Alarms are calculated, but all other aspects of the block do not update.

Fallback. (AUTO/MANUAL). The normal operating mode of the block. Updated when AUTO or MANUAL is selected in the MODE field. MODE field adopts this value on exit of TEST or SIMULATE mode.

PV. The Process Variable. Its coldstart value is 0. When *MODE* is set AUTO *PV* is derived from the measured frequency value and status returned from the I/O subsystem, and is read-only. In all other modes it is writeable.

HR, LR. Range of *PV* in engineering units. *HR* and *LR* define two points on the engineering units scale (*PV*) that map from points *HR_in* and *LR_in*, respectively, on the input. *HR_in/LR_in* to *HR/LR* scaling is not applied if *UserChar* is defined. The equivalent conversion equation is:

$$PV = LR + \left[\frac{\text{Frequency} - LR_in}{HR_in - LR_in} \times (HR - LR) \right]$$

NOTE that *Scale* is applied to *FI* before the *HR_in/LR_in* to *HR/LR* scaling. If *Scale* is set to 1 then *HR_in/LR_in* will be in Hz, otherwise they will be in scaled units per second, for example, litres per second (l/s), revolutions per second (rev/s), and so on.

HiHi, Hi, Lo, LoLo. Absolute alarm levels defining when the corresponding alarms trip, with user configurable hysteresis (*Hyst*) in all modes.

Hyst. Hysteresis bandwidth value, applied inside all alarm levels.

Filter. Time constant (seconds) of a simple first-order filter applied to the measured frequency value (*FI/PV*), to remove noise from the signal.

UserChar. Specifies the name of a UCHAR block defining a 16-point (x,y) custom linearisation/characterisation table to be applied to *FI*.

NOTE that *Scale* is applied to *FI* before user characterisation is applied. If *Scale* is set to 1 then the input to the characterisation will be in Hz, otherwise they will be in scaled units per second, for example, litres per second (l/s), revolutions per second (rev/s), and so on.

CutOff. Low threshold value for *PV*. If *PV* falls below *CutOff*, *PV* automatically adopts the value specified by the *Default* parameter and the *Status.CutOff* is set TRUE.

Default. Default value of *PV*, automatically adopted if *PV* falls below the value specified by the *CutOff* parameter.

Count. Scaled counter output. Relates to the number of pulses input to the block, multiplied (or divided) by the *Scale* parameter. *Count* can increment or decrement with pulses, depending on the *Control.UpCount* parameter, and automatically resets to a ‘start value’ after reaching an ‘end value’, as specified by the *SetCount* parameter.

Carry. Sets TRUE when the total in *Count* has reached its ‘end value’ (defined by *Control.UpCount* and *SetCount*), and has then been reset to its defined ‘start value’. *Carry* automatically resets to FALSE in the following update cycle. Initialising the block via *Control.Init* also resets *Carry*.

Overflow. Set TRUE when the counter has reached its ‘end value’ and has been reset to its ‘start value’. The *Overflow* parameter is reset when *Control.Init* is TRUE, or if *Control.ClearOv* is TRUE.

SetCount. Defines ‘start values’ and ‘end values’ for the *Count* parameter. When the block is configured as an up-counter (*Control.UpCount* TRUE) the ‘end value’ equals *SetCount* and the ‘start value’ equals *Count* – *SetCount*.

NOTE This ‘start value’ (*Count* - *SetCount*) does not necessarily equal zero, as the total in *Count*, increased by the last batch of pulses collected in the buffer, usually exceeds *SetCount* when the reset is triggered. When the block is configured as a down-counter (*Control.UpCount* set FALSE) the ‘end value’ equals zero and the ‘start value’ equals *Count* + *SetCount*.

Scale. Applied to the raw delta count and raw frequency measurement, *FI*. Specifies the value added to (or subtracted from) *Count* for each pulse detected, e.g. with *Control.UpCount* set TRUE and *Scale* = 2, if 15 pulses are detected between block updates and *Option.ScaleUp* set TRUE, then 30 is added to *Count*. If *Option.ScaleUp* set FALSE, 7 is added to *Count* and 1 pulse is held over until the next block update.

FI is multiplied by *Scale* before ranging or linearisation if *Option.ScaleUp* is TRUE. If *Option.ScaleUp* is FALSE, then *FI* is divided by *Scale*.

Control. Bitfield parameters can be connected to digital inputs from the strategy. These only apply to the counter function, and have no effect on the measured frequency function.

- **Init.** TRUE initialises the counter, i.e. resets *Carry*, *Overflow*, and *Count*, and will automatically reset to FALSE.
- **Disable.** If TRUE the counter stops, but only when the *Count* total has reached its ‘end value’, then reset to its ‘start value’, and the *Carry* and *Overflow* parameters set TRUE. After that, input pulses are not added to *Count*, until *Disable* is reset to FALSE.

NOTE Resetting *Control.Disable* does not reset the *Overflow* parameter.

- **ClearOv.** A TRUE input resets the *Overflow* parameter, then automatically resets this bitfield FALSE.
- **UpCount.** TRUE selects an up-counter function: scaled pulses add to the *Count* total. FALSE selects a down-counter function: scaled pulses subtract from the *Count* total. The *Count* value is not re-initialised when *Control.UpCount* is switched, so the counter function can be changed in mid-count, before an ‘end value’ is reached.
- **InputOff.** TRUE stops pulses altering the *Count* total immediately, unlike *Control.Disable*. Other *Control* functions are unaffected (e.g. *Control.Init*).

AlmOnTim, AlmOfTim. Each time the block is processed the alarm conditions are evaluated. When the state of a time-delayed alarm changes, the corresponding internal timer is loaded with the *AlmOnTim* if the state is now TRUE, or with *AlmOfTim* if FALSE. There is an independent timer for each of the time-delayed alarms. When a timer reaches zero, the state of the corresponding Alarms field is set or cleared.

The time-delayed alarms are *HiHi*, *Hi*, *Lo*, *LoLo* and *OutRange*.

NOTE If an alarm state changes continuously within the values specified in *AlmOnTim* and *AlmOfTim*, the state of the reported alarm does not change.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.
- **Hardware.** TRUE when the *Status.HwFlt*, *Status.BadSetup* or *Status.BadTask* bit is set TRUE, e.g. in the event of a power interruption, incorrect module type, transmitter PSU failure, circuit hardware fault, etc.
- **HiHi, Hi, Lo, LoLo.** Set TRUE by parameters *HiHi*, *Hi*, *Lo*, and *LoLo*, respectively. The alarms do not clear until *PV* has returned within these levels by more than the hysteresis value.

- **OutOfRange.** TRUE when *PV* exceeds a value of 10% above or below *HR* or *LR* respectively. The alarm does not clear until *PV* has returned within these levels by more than the hysteresis value.
- **CutOff.** TRUE when the measured frequency falls below value in *CutOff* parameter. This alarm has 0.5% hysteresis.
- **CctFault.** Circuit fault. TRUE when the module detects an open circuit (on current outputs), or a short circuit (on voltage outputs).
- **CharErr.** The UChar block specified in the *UserChar* field was not detected or the input exceeds the range of this table.
- **TaskRate.** TRUE if a Task Rate exceeds the time permitted by the I/O Subsystem, e.g. T2550/T2750 is 30 seconds. If this time update rate is not achieved then counts may be lost.
- **NotAuto.** TRUE if operating in MANUAL, TEST, or SIMULATE mode.
- **ModBlock.** TRUE if there is no corresponding MOD_UIO block.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

Node. Specifies the LIN node number in hex format of the base unit where the module is fitted. The *Node* and *SiteNo* identify the MOD_UIO block to which this channel block applies. The channel block must be allocated to the same task as its' associated MOD_UIO block.

SiteNo. Specifies the position on the Base Unit where the I/O module corresponding to this block is located. The *Node* and *SiteNo* identify the MOD_UIO block to which this channel block applies. The channel block must be allocated to the same task as its associated MOD_UIO block. An invalid *SiteNo* sets the *Status.Missing* bit.

Channel. Defines the channel number within the I/O module specified by the *SiteNo* parameter, to which this block is attached. The number of channels available depends on the Base Unit. An invalid *Channel* sets the *Status.BadType* bit.

InType. (Off/Magnetic/mA/Volts). Specifies the type of input connected to the I/O module associated with this block. Defines the input signal type and conditioning path. Only the options supported by the hardware are available. As *InType* changes, the appropriate electrical units are defined for *Thresh*.

HR_in, LR_in. *HR_in* and *LR_in* define two points on the input that map to points *HR* and *LR*, respectively, on the engineering units scale (*PV*). *HR_in/LR_in* to *HR/LR* scaling is not applied if *UserChar* is defined. Note that *Scale* is applied to *FI* before the *HR_in/LR_in* to *HR/LR* scaling. If *Scale* is set to 1, then *HR_in/LR_in* will be in Hz, otherwise they will be in scaled units per second, such as litres per second (l/s), revolutions per second (rev/s), and so on.

FI. The measured frequency in its raw state. *FI* is scaled by *Scale* in the direction defined by *Option.ScaleUp* then *HR_in/LR_in* to *HR/LR* scaling applied to produce *PV*. If *UserChar* is defined then it is used instead of *HR_in, LR_in* to *HR, LR* scaling. In this case, *HR* and *LR* are still used for the range alarm.

Thresh. Sets the trigger threshold, using the electrical units defined by the *InType*.

Burden. Sets the burden resistor value for *InType* = mA only. When the I/O subsystem supports user configurability, this field selects the burden resistor value. Otherwise it displays the setting of the fitted burden resistor.

Debounce. (0/5/10/20/50ms) Debounce filter for mA/V inputs. Inhibits any apparent change on the input until it has been stable for the specified time. Because the debounce is applied to rising and falling edges, the maximum square wave frequency which can be measured is $1/(2 \times \text{debounce})$. For example, 100Hz for a *Debounce* of 5ms.

Caution

The *Alarms.OverRange* may not trigger when debounce inhibits a signal. Control loops based on PV should NOT apply Debounce without provision for protecting against the consequences of the measured frequency dropping to 0 due to the debounce.

PSU. Voltage select parameter for the programmable power supply, allowing it to be used with current preamplifiers, proximity detectors, and volt-free contacts. The PSU can supply 8, 12 or 24V. If any other value is written then the field will snap to the closest of these values.

NOTE Only the T2550 FI2 module supports 8V, 12V and 24V.

Options. Bitfield selecting a variety of signal-processing options. The following information is given in addition to the summary in *Table 146*.

- **ScaleUp.** If TRUE, *Scale* is a multiplier, applying to both counter and measured frequency functions. If FALSE, *Scale* is a divisor.
- **SBreak.** If TRUE, sensor break detection is enabled.
- **SCct.** If TRUE, short circuit detection is enabled.

Status. Bitfield indicating general comms./hardware error conditions. Please refer to *Table 146* for details.

- **Missing.** If TRUE, the module block is missing or assigned to an unsupported task.
- **BadType.** If TRUE expected and fitted modules are not compatible, sets *Alarms.Hardware* TRUE.
- **Ranging.** If TRUE, the *Channel* is currently initialising.
- **BadSetup.** If TRUE, an invalid Setup for the connected I/O module is detected, e.g. if the user configured input range exceeds the instrument input range.
- **HwFlt.** If TRUE, a fault is detected in the I/O module, sets *Alarms.Hardware* TRUE.
- **NotAuto.** If TRUE, the I/O module is not operating in AUTO mode, sets *Alarms.NotAuto* TRUE.
- **OverRng, UnderRng.** If TRUE, the input value is not measurable by the hardware, sets *Alarms.OutRange* TRUE.
- **OpenCct, ShortCct.** If TRUE, a logic sensor open circuit or short circuit is detected, respectively, sets *Alarms.CctFault* TRUE.
- **BadHwSet.** If TRUE, the hardware configuration does not correspond to the requested configuration, defined in the block, i.e. a burden resistor is fitted (*InType* = mA), but the block is configured as *InType* = mV. This will set *Alarms.Hardware* TRUE when asserted.
- **CutOff.** If TRUE, the measured frequency falls below the value specified in *CutOff*.
- **BadTask.** If TRUE, the block is not on the same task as the parent module block (MOD_UIO).

Examples of setting up scaling in the FI_UIO block

The following three scenarios provide real-life examples of how one could approach the setting up of the scaling in the FI_UIO block.

Example 1: Turbo Machinery Control

In this scenario, the input comes from a sensor pointed at a gear wheel with a defined number of teeth. The requirement is to normalise from the 49-tooth gear wheel to revolutions, and then display revolutions per minute and count revolutions. The *Scale* parameter acts as a divide function (*ScaleUp* in the *Options* field = FALSE) to normalise the pulses to revolutions. *Count* is used to count the number of revolutions. The *PV* is scaled from revolutions per second to revolutions per hour using *HR_in* and *LR_in* (rev/s) and *HR* and *LR* (rev/min). Trending of *PV* is scaled by *HR/LR*, 0-600 RPM for example.

The setup for this scenario is shown in the following figure.

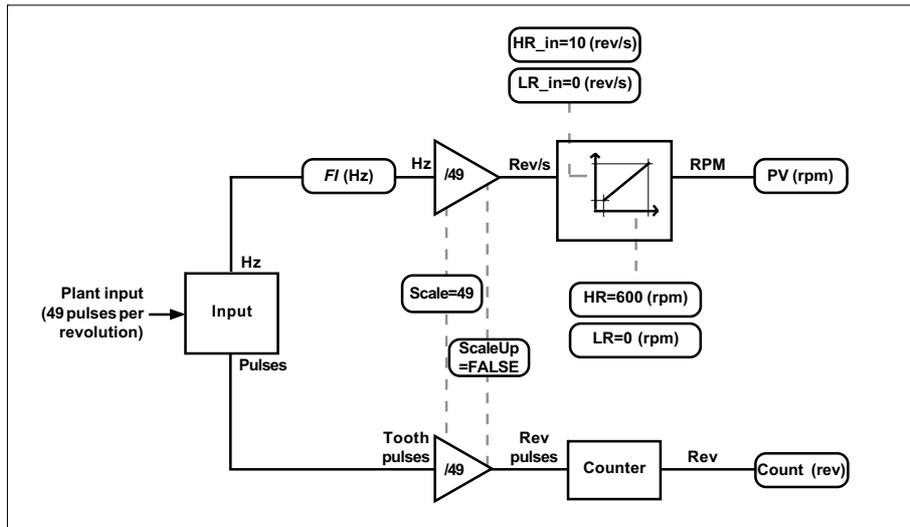


Figure 70 Turbo Machinery Control

Example 2: Canning Plant

In this scenario, the end of a production line has a sensor counting cases of beer cans. The requirement is to have a rate and count display of the number of cans produced. In this scenario, *Scale* is used as a multiplier (*ScaleUp* in the *Options* field = TRUE) to multiply by 24 cans per case. The count field then counts bottles. In the example, *PV* is then scaled from cans per second to cans per minute using *HR_in* and *LR_in* (cans/s) and *HR* and *LR* (cans/min). Trending of *PV* is scaled by *HR/LR*, 0-300 cans/min, for example.

The setup for this scenario is shown in the following figure.

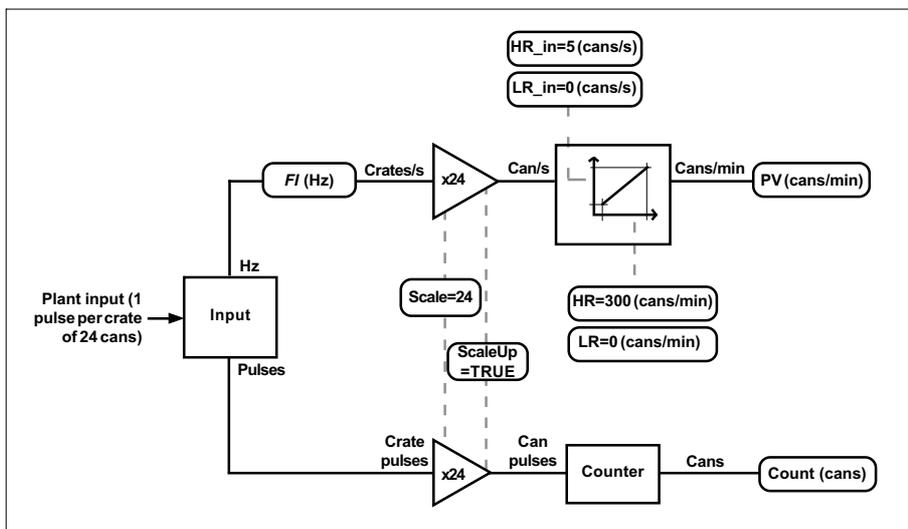


Figure 71 Example 2 - Canning Plant

Example 3: Flow Meter

In this scenario, a flow meter produces pulses indicating the rate of flow of a fluid. The requirement is to measure the quantity of fluid and the rate of flow. *Scale* is used to convert the incoming pulses to represent a volume. In this scenario, the flow meter produces 2 pulses per litre, so *Scale* is set to 2 and the *ScaleUp* bit in the *Options* field is set to FALSE. *Count* accumulates the number of litres. In this example, *PV* is then scaled litres per second to litres per hour using *HR_in* and *LR_in* (litre/s) to *HR* and *LR* (litres/hour). Trending of *PV* is scaled by *HR/LR*, 0-9100 litres/hour, for example.

The setup for this scenario is shown in the following figure.

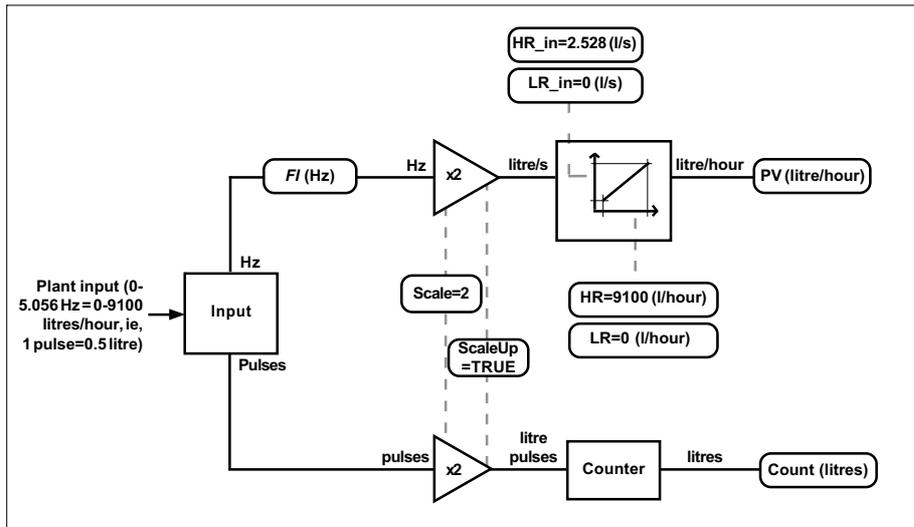


Figure 72 Example 3 - Flow Meter

MOD_DI_UIO: MULTI-CHANNEL DIGITAL INPUT MODULE BLOCK

Block function

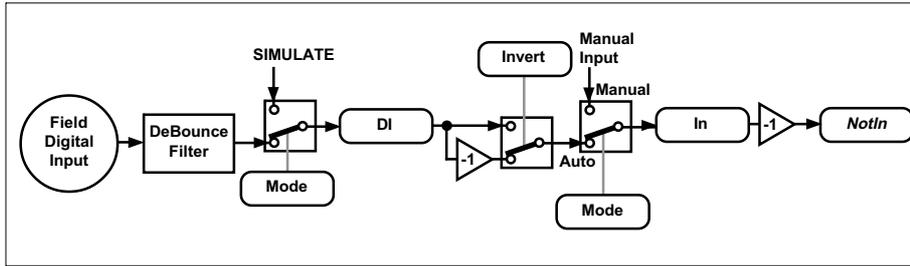


Figure 73 Block schematic

Please refer to *Figure 73*, showing a single channel only. This block allows upto 16 (sixteen) digital signals to be input to the strategy, and provides Auto/Manual control, inversion, and debounce. The input can be configured as either contact or voltage sensing.

Block parameters

Symbols used in *Table 147* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Node	LIN Node address of the base unit	Hex	
SiteNo	Module's position on the base unit	Integer	
Mode	Current operating mode of the base unit	Menu	
Fallback	Fallback operating mode of the base unit	Menu	
ChanMask	Channel available	ABCD hex	
Chan1 to Chan16	TRUE, if channel is available	T/F	
DI		Plant input signal	ABCD hex
Chan1 to Chan16	TRUE, if channel is used	T/F	
Invert	Invert channel Input signal	ABCD hex	
Chan1 to Chan16	TRUE, inverts Input signal	T/F	
In	Channel Input state (after inversion)	ABCD hex	
Chan1 to Chan16	Input state of channel	T/F	
NotIn	Inverse of In channel	ABCD hex	
Chan1 to Chan16	Inverse of Input channel state	T/F	
Thresh	Threshold voltage for module	Volts	
Debounce	Channel debounce period	Secs	
Alarms			
Software	Block RAM data sumcheck error	T/F	
Hardware	I/O module failure	T/F	
NotAuto	Block not in normal operation	T/F	
Combined	Logical OR of all Alarm bits	T/F	
Expect	Requested operating module type	Menu	
Actual	Actual Module fitted	Menu	
Version	Actual Module Version fitted		
HwFlt	Fault in I/O module	(ABC)D hex	
Chan1 to Chan16	I/O Module failure	T/F	

Continued...

Parameter	Function	Units	Status
<i>Continued...</i>			
GoneTrue Chan1 to Chan16	In changed, False to True	(ABC)D hex	<input type="checkbox"/> <input type="checkbox"/>
	In channel <i>n</i> changed, False to True	T/F	
GoneFals Chan1 to Chan16	In changed, True to False	(ABC)D hex	<input type="checkbox"/> <input type="checkbox"/>
	In channel <i>n</i> changed, True to False	T/F	
PulsTrue Chan1 to Chan16	In pulsed to True	(ABC)D hex	<input type="checkbox"/>
	In channel <i>n</i> pulsed to True	T/F	
PulsFals Chan1 to Chan16	In pulsed to False	(ABC)D hex	<input type="checkbox"/>
	In channel <i>n</i> pulsed to False	T/F	
Status	Comms/hardware status	(AB)CD hex	<input type="checkbox"/>
Missing	No I/O module at site	T/F	D
BadType	Incorrect module type fitted	T/F	
BadSite	SiteNo value exceeds base unit capacity	T/F	
BadTask	Invalid task	T/F	
Ranging	Channel initialising	T/F	C
BadSetup	Hardware capabilities exceeded	T/F	
NotAuto	Instrument not operating in AUTO mode	T/F	
		T/F	B
		T/F	
		T/F	
		T/F	
		T/F	A
		T/F	
		T/F	
		T/F	

Table 147 Block parameters

Block specification menu

The following is given in addition to *Table 147*.

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

Node. Specifies the LIN node number in hex format of the base where the module is fitted.

SiteNo. Specifies the position on the Base Unit where the I/O module corresponding to this block is located.

Mode. (AUTO/MANUAL/TEST/SIMULATE). Current operating mode of all channels configured in the block. AUTO normal block operation. MANUAL allows *In* to be forced to a specified value at runtime and calculates alarms. TEST, enabled when *IO_Mode.EnaTest* in the Header block is set TRUE, functions the same as in Manual operation, but will cause all blocks currently in TEST to revert to the mode defined in *FallBack* when disabled. SIMULATE, enabled when *IO_Mode.EnaSim* in the Header block is set TRUE, only the *Status* and *In* fields are writeable. Alarms are calculated, but all other aspects of the block do not update.

Fallback. (AUTO/MANUAL). Normal operating mode of the block. Updated if AUTO or MANUAL is selected in MODE field. MODE field adopts this value on global exit of TEST or SIMULATE mode.

ChanMask. Identifies available channels from an I/O Module.

- **Chan1 to Chan16.** Each bit represents a single input. Defines each available channel within the I/O module derived from the *Actual* parameter, to which this block is attached.

DI. Input signal (Plant signal), wireable to the Strategy.

- **Chan1 to Chan16.** Each bit represents a single input plant signal.

Invert. Invert the Input signal control.

- **Chan1 to Chan16.** Each bit represents the control for a single input plant signal. Any bit set TRUE, causes the corresponding *In.Chanxx* to show the inverse of the associated *DI.Chanxx*, i.e. *In* is the inverse *DI*.

In. Input state, wireable to the Strategy.

- **Chan1 to Chan16.** Each bit represents a single input. In MANUAL, TEST and SIMULATE mode, individual bits can be written to and wired into. In AUTO, individual bits can only be wired out of. This shows the inverse of the corresponding *DI.Chanxx* if the corresponding *Invert.Chanxx* is set TRUE.

NotIn. The inverse of *In*, above.

- **Chan1 to Chan16.** Each bit represents the inverse of the corresponding single input, *In.Chanxx*.

Thresh. Threshold Voltage. Defines threshold for voltage input option. Any values exceeding the I/O Range are clipped to the limits of the target hardware.

Debounce. Debounce time, in seconds. This is the minimum time a change in the digital input signal must persist before the *In.Chanxx* parameter is allowed to switch to the new state.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.
- **Hardware.** A hardware alarm is generated when the *Status.HwFlt*, or *Status.BadTask* bit is set TRUE, e.g. in the event of a fault in the I/O hardware.
- **NotAuto.** TRUE if operating in MANUAL, TEST, or SIMULATE mode.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

Expect. (None/DI4/DI6_MV/DI6_HV/DI8_LG/DI8_LGv2/DI8_CO/DI8_COv2). This is the Module Type which is expected to be fitted at the specified *Node* and *SiteNo*. The type of available channel blocks and the block’s allocated task are decided on selection of the Module Type. A Module block can be allocated to any task, but the *Status.BadTask* is set TRUE, if allocated to an unsynchronised task.

Actual. (Unknown/None/AI2/AI3/AI4/AO2/DI4/DI6_MV/DI6_HV/DI8_LG/DI8_LGv2/DI8_CO/DI8_COv2/DO4_LG/DO4_LGv2/DO4_24/DO4_24v2/DO8/RLY4/RLY4v2/FI2/ZI). This is the actual Module Type fitted at the specified *Node* and *SiteNo*. *Unknown* indicates the I/O Module is not compatible with the current I/O subsystem configuration, e.g. when configuring a redundant T2550 subsystem, Version 1 I/O Module hardware is not compatible. Version 2 I/O Module hardware must be fitted in a redundant subsystem, where appropriate.

Version. The version number of the module fitted at the selected *SiteNo*. When used in conjunction with *Actual*, this will assist with diagnosing module incompatibilities.

HwFlt. Hardware fault detected.

- **Chan1 to Chan16.** Each bit represents a single input. If TRUE, a fault has been detected in the I/O module communicating via the configured channel, sets *Alarms.Hardware* TRUE.

GoneTrue/GoneFals. Shows corresponding *In.Chanxx* has changed.

- **Chan1 to Chan16.** Each bit represents a single input. If TRUE, the Input state, *In*, has changed from *FALSE* to *TRUE* or *TRUE* to *FALSE*, respectively.

PulsTrue/PulsFals. Shows corresponding *In.Chanxx* has changed.

- **Chan1 to Chan16.** Each bit represents a single input. If TRUE, a transition in the last task cycle is detected.

	DI _n	DI _{n+1}	PulsTrue	PulsFals
	F	T	T	F
	F	F	T	F
	F	F	F	F
	T	F	F	T
	T	T	F	T
	T	T	F	F

Status. Bitfield indicating general comms./hardware error conditions.

- **Missing.** If TRUE, the module block is missing or assigned to an unsupported task.

- **BadType.** If TRUE expected and fitted modules are not compatible, sets *Alarms.Hardware* TRUE.
- **BadSite.** If TRUE, an invalid *SiteNo* has been entered e.g. *SiteNo* equals 16, but *Model* in the header block shows an 8-way base unit.
- **BadTask.** If TRUE, the block is operating on a non-I/O task.
- **Ranging.** If TRUE, the channel is currently initialising.
- **BadSetup.** If TRUE, an invalid setup for the connected I/O module.
- **NotAuto.** If TRUE, the I/O module is not operating in AUTO mode, sets *Alarms.NotAuto* TRUE.

MOD_DO_UIO: MULTI-CHANNEL DIGITAL OUTPUT MODULE BLOCK

Block function

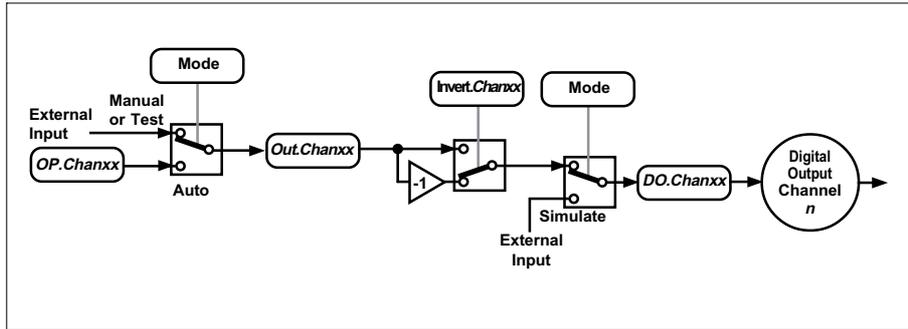


Figure 74 Block schematic

Please refer to *Figure 74*, illustrating a single digital output channel. The Digital output block transfers upto 16 (sixteen) logic signals to digital output channels. The block provides Auto/Manual/Track control, and inversion.

Block parameters

- Symbols used in *Table 148* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Node	LIN Node address of the base unit	Hex	
SiteNo	Module's position on the base unit	Integer	
Mode	Current operating mode	Menu	
FallBack	Fallback operating mode	Menu	
ChanMask	Channel available	ABCD hex	
Chan1 to Chan16	TRUE, if channel is available	T/F	
OP		Output state from Strategy	ABCD hex
Chan1 to Chan16	TRUE, if channel is used	T/F	
Out		Channel output state (before inversion)	ABCD hex
Chan1 to Chan16	Output state of channel	T/F	
Invert		Invert channel Input signal	ABCD hex
Chan1 to Chan16	TRUE, inverts Input signal	T/F	
DO		Plant output signal	ABCD hex
Chan1 to Chan16	TRUE, if channel is used	T/F	
Alarms			
Software	Block RAM data sumcheck error	T/F	
Hardware	I/O module failure or transmitter PSU faults	T/F	
NotAuto	Block not in normal operation	T/F	
Combined	Logical OR of all Alarm bits	T/F	
Expect	Requested operating module type	Menu	
Actual	Actual Module fitted	Menu	
Version	Actual Module Version fitted		
HwFlt	Fault in I/O module	(ABC)D hex	
Chan1 to Chan16	I/O Module failure	T/F	

Continued...

Parameter	Function	Units	Status
<i>Continued...</i>			
Status	Comms/hardware status	(AB)CD hex	 
Missing	No I/O module at site	T/F	D
BadType	Incorrect module type fitted	T/F	
BadSite	SiteNo value exceeds base unit capacity	T/F	
BadTask	Invalid task	T/F	
Ranging	Channel initialising	T/F	C
BadSetup	Hardware capabilities exceeded	T/F	
NotAuto	Instrument not operating in AUTO mode	T/F	
			B
			A

Table 148 Block parameters

Block specification menu

The following is given in addition to *Table 148*.

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

Node. Specifies the LIN node number in hex format of the base where the module is fitted.

SiteNo. Specifies the position on the Base Unit where the I/O module corresponding to this block is located.

Mode. (AUTO/MANUAL/TEST/SIMULATE). Current operating mode of all channels configured in the block. AUTO normal block operation. MANUAL allows *Out* to be forced to a specified value at runtime and calculates alarms. TEST, enabled when *IO_Mode.EnaTest* in the Header block is set TRUE, functions the same as in Manual operation, but will cause all blocks currently in TEST to revert to the mode defined in *FallBack* when disabled. During SIMULATE, enabled when *IO_Mode.EnaSim* in the Header block is set TRUE, only the *Status* and *DO* fields are writeable. Alarms are calculated, but all other aspects of the block do not update.

Fallback. (AUTO/MANUAL). Normal operating mode of the block. Updated if AUTO or MANUAL is selected in MODE field. MODE field adopts this value on global exit of TEST or SIMULATE mode.

ChanMask. Identifies available channels from an I/O Module.

- **Chan1 to Chan16.** Each bit represents a single input. Defines each available channel within the I/O module derived from the *Actual* parameter, to which this block is attached.

OP. Requested output from Strategy.

- **Chan1 to Chan16.** Each bit represents a single input to the block from the Strategy, and is the source of *Out.Chanxx* in AUTO mode. Each field indicates the actual output channel state if the corresponding *Invert.Chanxx* is False, but if *Invert.Chanxx* is TRUE an inverted form of the output channel state is displayed.

Out. The status (before inversion) of the field output connection, wireable to the Strategy.

- **Chan1 to Chan16.** Each bit represents a single output, sourced from *OP* or directly set by user depending on current operating mode.

Invert. Invert the Output signal control.

- **Chan1 to Chan16.** Each bit represents the control for a single output plant signal. Any bit set TRUE, causes the corresponding *Out.Chanxx* to show the inverse of the associated *DO.Chanxx* before transmission to the I/O subsystem, i.e. of inverting the output signal..

DO. Output signal (Plant signal), wireable from the strategy.

- **Chan1 to Chan16.** Each bit represents a single output plant signal.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block's RAM data.
- **Hardware.** A hardware alarm is generated when the *Status.HwFlt*, or *Status.BadTask* bit is set TRUE, e.g. in the event of a fault in the I/O hardware.
- **NotAuto.** TRUE if operating in MANUAL, TEST, or SIMULATE mode.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

Expect. (None/DO4_LG/DO_LGv2/DO4_24/DO4_24v2/DO8/RLY4/RLYv2). This is the Module Type which is expected to be fitted at the specified *Node* and *SiteNo*. The type of available channel blocks and the block's allocated task are decided on selection of the Module Type. A Module block can be allocated to any task, but the *Status.BadTask* is set TRUE, if allocated to an unsynchronised task.

Actual. (Unknown/None/AI2/AI3/AI4/AO2/DI4/DI6_MV/DI6_HV/DI8_LG/DI8_LGv2/DI8_CO/DI8_COv2/DO4_LG/DO4_LGv2/DO4_24/DO4_24v2/DO8/RLY4/RLY4v2/FI2/ZI). This is the actual Module Type fitted at the specified *Node* and *SiteNo*. *Unknown* indicates the I/O Module is not compatible with the current I/O subsystem configuration, e.g. when configuring a redundant T2550 subsystem, Version 1 I/O Module hardware is not compatible. Version 2 I/O Module hardware must be fitted in a redundant subsystem, where appropriate.

Version. The version number of the module fitted at the selected *SiteNo*. When used in conjunction with *Actual*, this will assist with diagnosing module incompatibilities.

HwFlt. Hardware fault detected.

- **Chan1 to Chan16.** Each bit represents a single output. If TRUE, a fault has been detected in the I/O module communicating via the configured channel, sets *Alarms.Hardware* TRUE.

Status. Bitfield indicating general comms./hardware error conditions.

- **Missing.** If TRUE, the module block is missing or assigned to an unsupported task.
- **BadType.** If TRUE expected and fitted modules are not compatible, sets *Alarms.Hardware* TRUE.
- **BadSite.** If TRUE, an invalid *SiteNo* has been entered e.g. *SiteNo* equals 16, but *Model* in the header block shows an 8-way base unit.
- **BadTask.** If TRUE, the block is operating on a non-I/O task.
- **Ranging.** If TRUE, the *Channel* is currently initialising.
- **BadSetup.** If TRUE, an invalid *Setup* for the connected I/O module.
- **NotAuto.** If TRUE, the I/O module is not operating in AUTO mode, sets *Alarms.NotAuto* TRUE.

VP_UIO: VALVE POSITIONER BLOCK

Block function

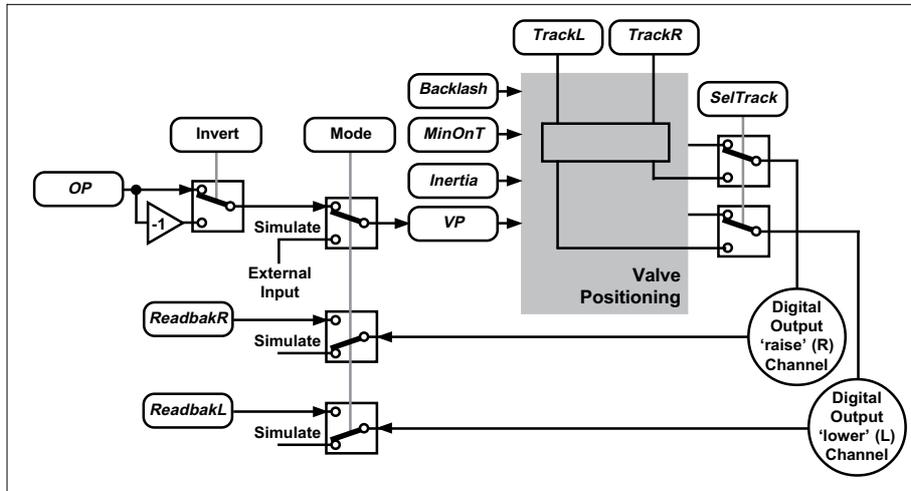


Figure 75 Valve Positioner Block schematic

Please refer to the schematic in *Figure 75*. This block is designed specifically for controlling the position of a valve via two digital output channels from a single I/O module. The block represents two points from a single point in the plant/system to effect the ‘raise’ (R) and ‘lower’ (L) valve position output.

NOTE Valve control can be used as an alternative to the standard PID control algorithm.

Block parameters

Symbols used in this table are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
MODE	Current operating mode	Menu	
OP	Output state from Strategy	Eng	☐➔
MinOnT	Minimum on/off pulse duration	Secs	☐➔
Inertia	Valve positioning time overrun	Secs	☐➔
Backlash	Time before valve positioning starts	Secs	☐➔
TrackR	Track output for raise	T/F	☐➔
TrackL	Track output for lower	T/F	☐➔
Alarms			☐📖🔊
Software	Block RAM data sumcheck error	T/F	
Hardware	I/O module failure or transmitter PSU faults	T/F	
BadDmnd	A non-numeric demand signal has been encountered	T/F	
NotAuto	Block not in normal operation	T/F	
ModBlock	Module block error	T/F	
Combined	Logical OR of all Alarm bits	T/F	
Node	LIN Node address of the base unit	Hex	📖
SiteNo	Module's position on the base unit	Integer	
ChannelR	Specifies I/O module channel number for raise	Integer	
ChannelL	Specifies I/O module channel number for lower	Integer	
VP	Plant output signal	%	☐➔
Pullup	High logic status output voltage	Volts	
ReadbakR	Feedback of low-level I/O module output state	T/F	☐📖
ReadbakL	Feedback of low-level I/O module output state	T/F	☐📖
Options		(ABC)D hex	☐➔
Invert	Output signal inverter	T/F	<div style="border: 1px solid black; padding: 2px; display: inline-block;"> 1 2 4 8 </div> D
SelTrack	Track mode select	T/F	

Continued...

Parameter	Function	Units	Status
<i>Continued...</i>			
Status	Comms/hardware status	ABCD hex	 
Missing	No valid I/O block found	T/F 1	D 
BadType	No such channel on I/O module	T/F 2	
Ranging	Channel initialising	T/F 4	
BadSetup	Hardware capabilities exceeded	T/F 8	
HwFlt	Fault in I/O module	T/F 1	C 
NotAuto	Instrument not operating in AUTO mode	T/F 2	
Clipped	OP exceeded -100% to +100%	T/F 4	
		8	B 
		1	
		2	
		4	
		8	A 
BadDmnd	A non-numeric demand signal has been encountered	T/F 1	
BadTask	Invalid task	T/F 2	
		4	
		8	

Table 149 Block parameters

Block specification menu

Dbase, Block, Type. See Appendix D page 544 for details of these ‘header’ fields.

Mode. (AUTO/TRACK/SIMULATE). Current operating mode. AUTO is the normal operating mode of the block, in which the wired OP value is passed to the VP algorithm. In TRACK mode (*Options.SelTrack* TRUE), the VP algorithm is bypassed and the R and L digital output points are driven directly from *TrackR* and *TrackL* respectively, subject to enforcing the normal minimum off time. During SIMULATE operation the *Status* field is writeable. VP is held at 0%, alarms are calculated, but all other aspects of the block do not update.

OP. The demanded valve velocity expressed as a percentage. A positive value opens the valve, ‘raise’ (R), and a negative value closes the valve, ‘lower’ (L). An OP value of ±100% instructs the valve to move at its maximum speed, i.e. the raise or lower digital output, will be on constantly when the absolute value of OP is 100. Typically, this field is wired from the LOOP block, *Ch1Outpt* or *Ch2Outpt* field.

NOTE. If the *BadDmnd* alarm bit is set TRUE, the demanded valve velocity (*VP* parameter) is set to 0.

MinOnT. The Minimum On Time, is also used as minimum off time. It defines the minimum duration of the ‘raise’ and ‘lower’ pulse and is used to minimise the frequency of the switch on and switch off valve motors. This value determines how accurately the valve can be positioned. The shorter the time, the more precise the control. However, if the time is set too short, process noise will cause an excessively busy valve.

Inertia. The time for which the valve continues to move after the ‘raise’ or ‘lower’ digital output has reverted to FALSE.

Backlash. The time for which the ‘raise’ or ‘lower’ digital output needs to be TRUE before the valve actually starts to move following a change of direction.

TrackR, TrackL. When *Options.SelTrack* is TRUE, these fields will be copied directly to the digital output points, bypassing the Valve Positioning algorithm, subject to enforcing the normal minimum off time.

Alarms. See Appendix D page 545 for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.
- **Hardware.** Asserted if the *Status.HwFlt*, or *Status.BadTask* bit is set TRUE, e.g. in the event of a fault in the I/O hardware.
- **BadDmnd.** TRUE if a non-numeric internal (to the block) or external (*OP* parameter, for example) demand signal has been encountered.
- **NotAuto.** Asserted if operating in TRACK, or SIMULATE mode.
- **ModBlock.** Asserted if there is no corresponding MOD_UIO block.

- **Combined.** Asserted if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

Node. Specifies the LIN node number in hex format of the base where the module is fitted. The *Node* and *SiteNo* identify the MOD_UIO block to which this channel block applies. The channel block must be allocated to the same task as its accompanying MOD_UIO block.

SiteNo. Specifies the position on the Base Unit where the I/O module corresponding to this block is located. The *Node* and *SiteNo* identify the MOD_UIO block to which this channel block applies. The channel block must be allocated to the same task as its associated MOD_UIO block. An invalid *SiteNo* sets the *Status.Missing* bit.

ChannelR, ChannelL. Defines the channel numbers within the I/O module specified by the *SiteNo* parameter, to which this block is attached. The number of channels available depends on the I/O module type. *ChannelL* must be the channel associated with *ChannelR* in the I/O subsystem. Setting one of these fields automatically sets the other. An invalid *ChannelR/ChannelL* sets the *Status.BadType* bit TRUE.

VP. The demanded valve velocity currently sent to the module driver valve positioning algorithm. Can be written directly in SIMULATE mode. In TRACK mode it displays **-100%/0%/+100%** to represent **Lower/Rest/Raise**, otherwise it is copied from *OP*. The value copied from *OP* is negated if *Options.Invert* is TRUE. If the value exceeds +/-100%, *VP* will be clipped and *Status.Clipped* set TRUE.

Pullup. Specifies the output voltage used for high logic status, if supported by the plant/system hardware.

ReadbakR, ReadbakL. Indicates feedback of low-level I/O module output state, if supported by the hardware. If the *Options.Invert* field is TRUE the actual state is inverted before being displayed in *ReadbakR*, or *ReadbakL* as appropriate.

Options. Bitfield selecting a variety of signal-processing options. The following information is given in addition to the summary in *Table 149*.

- **Invert.** If TRUE the value of *OP* negates when copied to *VP*. This reverses the allocation of the 'raise' and 'lower' digital outputs.

NOTE This is not applicable while operating in TRACK mode.

- **SelTrack.** If TRUE the 'raise' and 'lower' digital outputs are driven directly from the *TrackR* and *TrackL* fields, bypassing the valve positioning algorithm.

Status. Bitfield indicating general comms./hardware error conditions. The following information is given in addition to the summary in *Table 149*.

- **Missing.** If TRUE, the module block is missing or assigned to an unsupported task.
- **BadType.** If TRUE expected and fitted modules are not compatible, sets *Alarms.Hardware* TRUE.
- **Ranging.** If TRUE, the channels are currently initialising.
- **BadSetup.** If TRUE, an invalid setup for the connected I/O module.
- **HwFlt.** If TRUE, a fault has been detected in the I/O module, sets *Alarms.Hardware* TRUE.
- **NotAuto.** If TRUE, the I/O module is not operating in AUTO mode, sets *Alarms.NotAuto* TRUE.
- **Clipped.** If TRUE, the *VP* value has been clipped to be within the range -100% to +100%.
- **BadDmnd.** If TRUE, a non-numeric internal (to the block) or external (*OP* parameter, for example) demand signal has been encountered.
- **BadTask.** If TRUE, the block is not on the same task as the parent module block (MOD_UIO).

CHAPTER 12 LOGIC FUNCTION BLOCKS

The LOGIC category of Function Block Templates provides the control strategy with functions for logical calculations.

PULSE: PULSE BLOCK

Block function

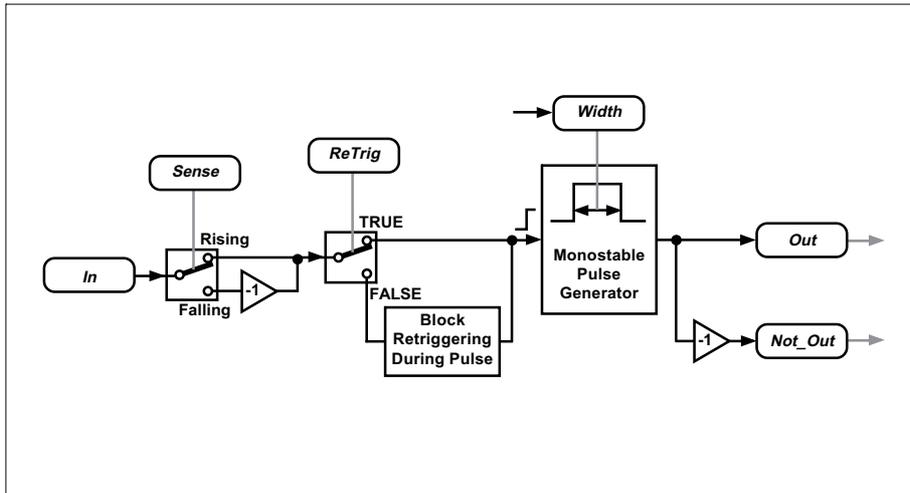


Figure 76 Block schematic

Please refer to *Figure 76*. The PULSE block acts as a monostable pulse generator, producing a rectangular pulse when triggered via the input. Rising or falling edges may be specified as the trigger (via the *Sense* parameter), and the output is available in true and complementary form. The *ReTrig* parameter can be set to prevent input edges retriggering the monostable during its 'on' period.

NOTE The maximum generated pulse length is dependant on the configured Task rate and instrument type, i.e. 46 hours if the block is running on the 10ms Task and 21 days if the block is running on 110ms Task.

Figure 77 shows examples of how these parameters affect the output of pulses.

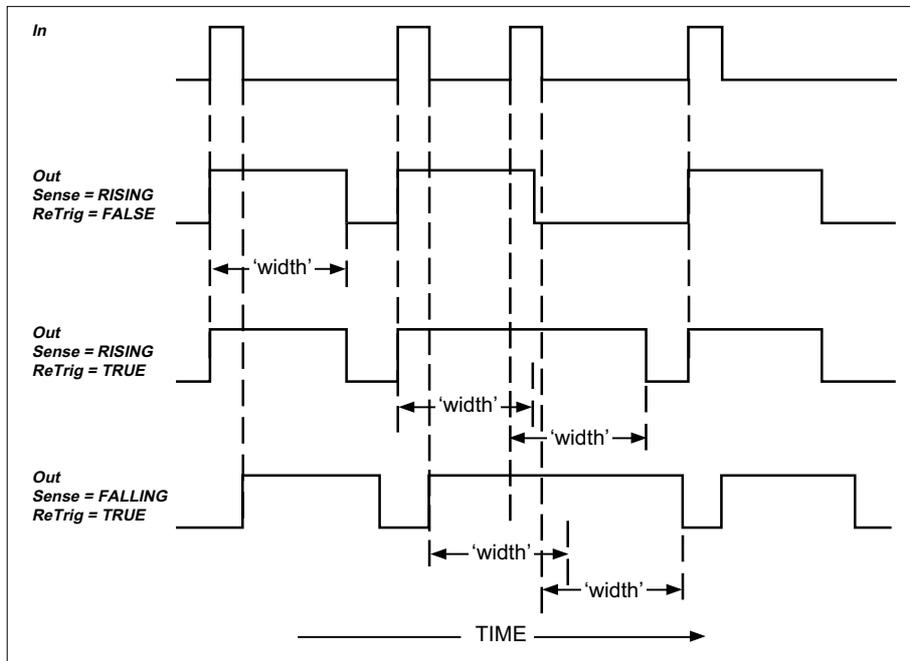


Figure 77 Block output (examples)

Block parameters

Symbols used in *Table 150* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
In	Triggering input	T/F	
Width	Monostable pulse width	Secs	 
Sense	Input trigger edge select	Menu	
ReTrig	Pulse timer retrigger enable	T/F	
Alarms			  
Software	Block RAM data sumcheck error/network failure	T/F	
Combined	OR-ing of all Alarms bits	T/F	
Out	Output	T/F	
Not_Out	Complementary output	T/F	

Table 150 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

In. The triggering input edge, either falling or rising, as specified in the *Sense* parameter. (If the block is run without an input connection, *In* returns automatically to its rest state when the triggered pulse resets to FALSE.)

Width. Monostable pulse width, in seconds. *Width* can be altered whilst the monostable is ‘on’.

Sense. (FALLING/RISING). Specifies the edge required to trigger the output pulse. *Falling* = TRUE-to-FALSE input transition, and *Rising* = FALSE-to-TRUE input transition.

ReTrig. Re-trigger enable. With *ReTrig* TRUE the block emulates a conventional monostable where the timer can be re-triggered at any moment. With *ReTrig* FALSE, re-triggering inputs are ignored whilst the output pulse is ‘on’ (i.e. *Out* = TRUE). *Figure 77* shows these effects.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

Out. Output from the monostable function, TRUE when the pulse is ‘on’.

Not_Out. Equal to logical NOT *Out*. Complementary form of the block output, FALSE when the pulse is ‘on’.

LOGIC FUNCTION BLOCKS
AND4: AND BLOCK
OR4: OR BLOCK
XOR4: EXCLUSIVE-OR BLOCK

Block function

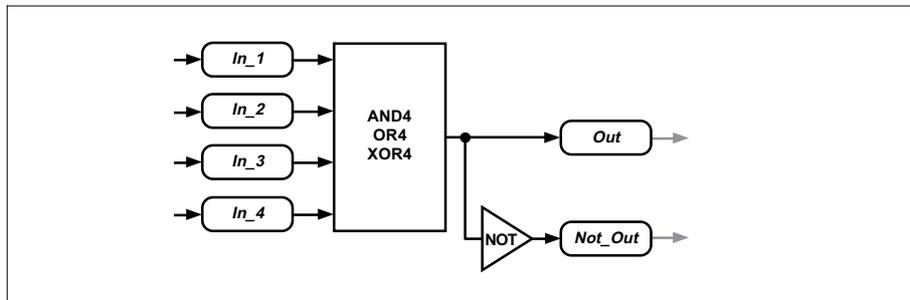


Figure 78 Block schematics

Please refer to the schematic in *Figure 78*, which applies to all three blocks.

Each block has four digital inputs and performs the corresponding Boolean function to generate the output, which is available in both true (*Out*) and complementary (*Not_Out*) forms.

In the AND4 block, *Out* is TRUE only if all four inputs are TRUE. In the OR4 block, *Out* is TRUE only if at least one of the inputs is TRUE. In the XOR4 block, *Out* is TRUE if, and only if, an odd number (1 or 3) of the inputs is TRUE. In all other cases the *Out* parameter is FALSE.

Block parameters

Symbols used in *Table 151* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
In_1	Input 1	T/F	
In_2	Input 2	T/F	
In_3	Input 3	T/F	
In_4	Input 4	T/F	
Out	Output	T/F	
Not_Out	Complementary output	T/F	
Alarms			
Software	Block RAM data sumcheck error/network failure	T/F	
Combined	OR-ing of all Alarms bits	T/F	

Table 151 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

In_1 to In_4. Inputs 1 to 4, respectively.

Out. Output from Boolean operation.

Not_Out. Equal to logical NOT *Out*. Complementary form of the block output.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

NOT: NOT BLOCK

Block function

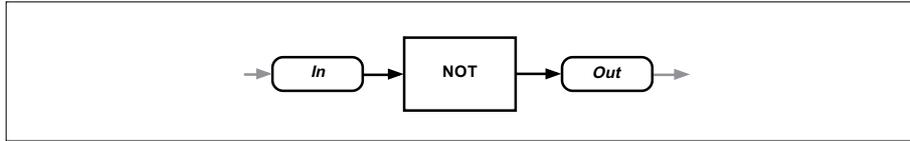


Figure 79 Block schematic

Please refer to *Figure 79*. The NOT block inverts a logic input, i.e. if *In* is TRUE, *Out* is FALSE; if *In* is FALSE, *Out* is TRUE.

Block parameters

Symbols used in *Table 152* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
In	Input	T/F	▶◻▶
Out	Output	T/F	◻▶ ◻▶
Alarms			◻▶ ◻▶ ◻▶
Software	Block RAM data sumcheck error/network failure	T/F	
Combined	OR-ing of all Alarms bits	T/F	

Table 152 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

In. Block input.

Out. Output from NOT operation.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

LATCH: LATCH BLOCK

Block function

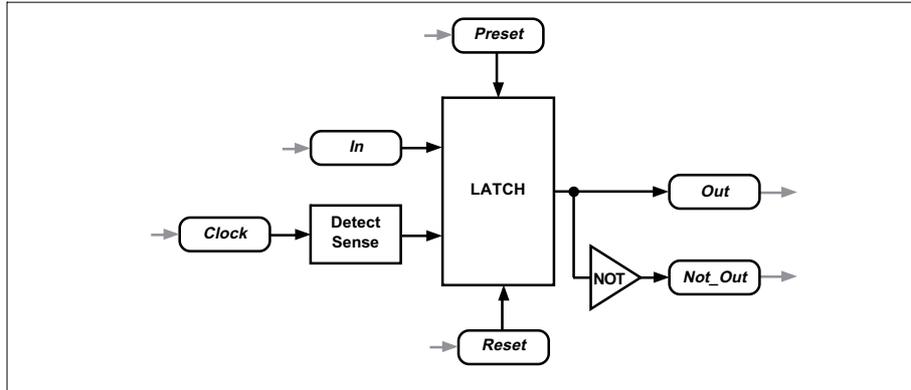


Figure 80 Block schematic

Please refer to *Figure 80*. The LATCH block provides a general purpose D-type flip-flop function, with a clock input that can be set to operate the latch on rising, falling, or either edges (specified by the *Sense* parameter). The *Clock* input sets *Out* equal to *In*, unless *Preset* or *Reset* are active. *Preset* forces *Out* to TRUE, overriding *In*. *Reset* forces *Out* to FALSE, overriding both *Preset* and *In*. The truth table in *Table 153* summarises the action of the block.

In	Clock	Preset	Reset	Out	Not_Out
X	X	TRUE	FALSE	TRUE	FALSE
X	X	X	TRUE	FALSE	TRUE
FALSE	↗/↘	FALSE	FALSE	FALSE	TRUE
TRUE	↗/↘	FALSE	FALSE	TRUE	FALSE

KEY: X = Any state /↗ = Rising or falling defined by *Sense*

Table 153 Block truth table

Block parameters

Symbols used in *Table 154* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
In	Data input	T/F	☑
Clock	Clock input	T/F	☑
Preset	Preset input	T/F	☑
Reset	Reset input	T/F	☑
Sense	Clock input trigger edge select	Menu	
Alarms			☑ ☑ ☑
Software	Block RAM data sumcheck error/network failure	T/F	
Combined	OR-ing of all Alarms bits	T/F	
Out	Output	T/F	☑ ☑
Not_Out	Complementary output	T/F	☑ ☑

Table 154 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

In. Data input.

Clock. A *Clock* input transition sets the *Out* field to the same state as the *In* field, unless *Preset* or *Reset* are TRUE. The *Clock* input responds either to rising or to falling edges, or both, as specified by the *Sense* parameter.

NOTE. On database start the ‘previous’ value of *Clock* is always considered to be FALSE; hence if *Clock* is TRUE when the database starts this will be treated as a rising edge.

Preset. When *Preset* is TRUE, *Out* is forced to TRUE, and *Clock* and *In* are overridden. *Preset* is overridden by *Reset*.

Reset. When *Reset* is TRUE, *Out* is forced to FALSE, and all other inputs are overridden.

Sense. (FALLING/RISING/EITHER). Specifies the triggering edge the *Clock* input is to respond to. FALLING = TRUE-to-FALSE transition; RISING = FALSE-to-TRUE transition; EITHER = both transitions.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

■ **Software.** Sumcheck error in block's RAM data.

■ **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

Out. Output from latch function.

Not_Out. Equal to logical NOT *Out*. Complementary form of the block output.

COUNT: COUNT BLOCK

Block function

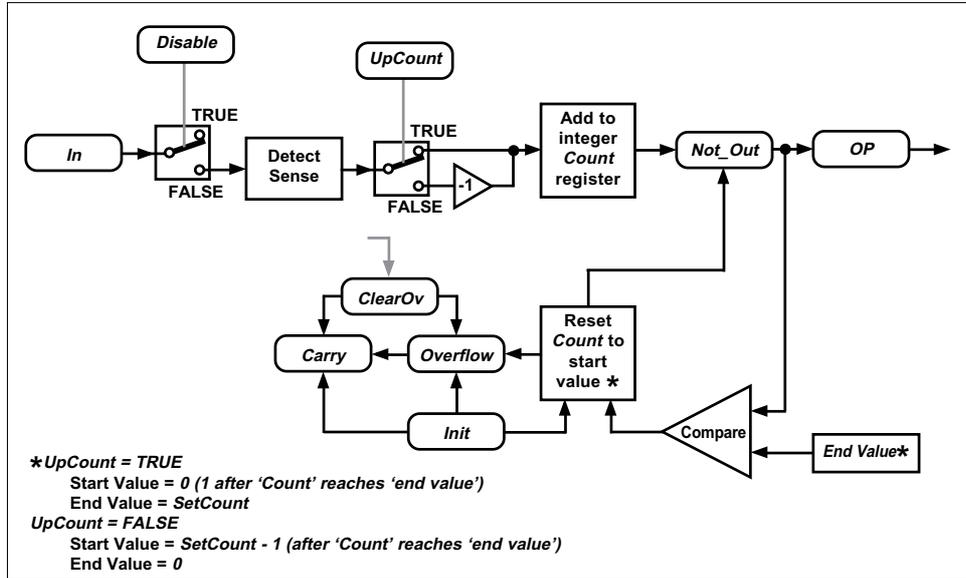


Figure 81 Block schematic

Please refer to *Figure 81*. The COUNT block counts input pulses, either rising or falling edges (as specified by the *Sense* parameter). Totals of up to 8 digits are held in the *Count* parameter as integers. The count total is also available as an analogue output connection *OP*, with 7-digit floating point accuracy.

The counter can be configured (or re-configured at any time) as either an up- or a down-counter, via the *UpCount* parameter. When configured as an up-counter, input pulses increment the total in *Count* until it reaches an 'end value' specified by the *SetCount* parameter. At the next pulse, *Count* is returned to a 'start value' of '1', an *Overflow* flag is latched on, and a *Carry* flag is temporarily set (usually for one more count) to allow cascading of counters. When configured as a down-counter, input pulses decrement the *Count* total until it reaches its 'end value', this time equal to zero. At the next pulse, *Count* is returned to 1 less than the 'start value', *SetCount*, and the *Overflow* and *Carry* flags are set as before. This value is 1 less to ensure the correct number of pulses are counted.

NOTE *Initialise*, *Disable* and *Overflow* flags reset inputs are also provided.

The timing diagram below is the result of the following Count block configuration, *SetCount* = 5, *Sense* = *Rising*, and *Disable* = *False*.

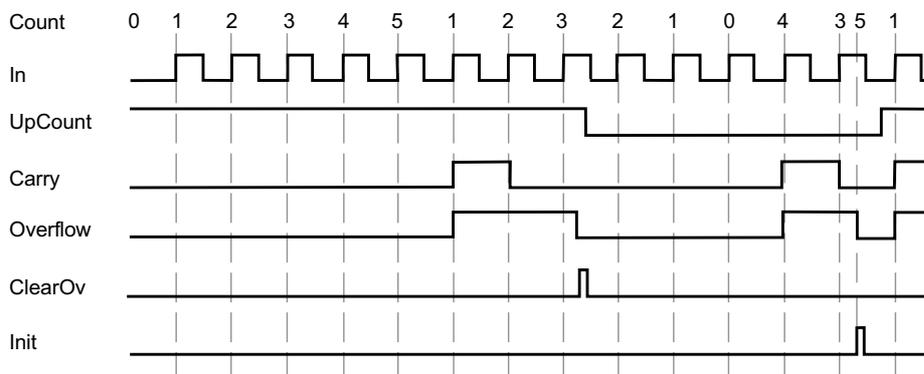


Figure 82 Timing example

Block parameters

Symbols used in *Table 155* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
In	Pulse input	T/F	
SetCount	Start/end count value	Integer	
Sense	Counter trigger edge select	Menu	
ClearOv	Overflow (& carry) flag clear	T/F	
Disable	Stop count at start value (after rolling over from end value)	T/F	
UpCount	Up/down count select	T/F	
Init	Counter initialise	T/F	
Alarms			
Software	Data corruption/communications fault		
Combined	OR-ing of all Alarms bits		
OP	Counter output value (floating point)	Eng	
Count	Count total (integral)	Integer	
Carry	Carry flag	T/F	
Overflow	Overflow flag	T/F	
HR_OP, LR_OP	High and Low graphics range of OP	Eng	
Options			
CanDsable	Determines whether the disable feature is enabled		

Table 155 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

In. Digital input to counter.

SetCount. Defines the ‘end value’ or the ‘start value’ for the *Count* parameter when the block is configured as an up- or a down-counter, respectively.

NOTE If *SetCount* is configured to up-count, the counter value, *Count*, will reset to 1 when the defined ‘end value’ is reached.

Sense. (FALLING/RISING) Specifies the triggering edge the counter input (*In*) is to respond to. FALLING = TRUE-to-FALSE transition; RISING = FALSE-to-TRUE transition.

ClearOv. A TRUE input resets the *Overflow* and *Carry* flags; then *ClearOv* resets to FALSE automatically.

Disable. With *Disable* TRUE the counter stops, but only after the *Count* total has reached its ‘end value’ and then been reset to its ‘start value’ (either *SetCount*-1 or 1), and the *Carry* and *Overflow* flags set. After that, input pulses are not added to *Count*, and the flags are ‘frozen’, until *Disable* is reset to FALSE.

NOTE To achieve correct operation of the Disable feature, refer to the *Options.CanDsable* field. If the *Options.CanDsable* field is not TRUE, the disable functionality will not ‘freeze’ the *Count* value.

UpCount. TRUE selects an *up*-counter function: pulses increment the count total which runs from zero to *SetCount*. FALSE selects a *down*-counter function: pulses decrement the total which runs in the reverse direction. Note that the *Count* value is not re-initialised when *UpCount* is switched, so the counter function can be changed in ‘mid-count’, before an end value is reached.

Init. A TRUE value initialises the counter, i.e. resets *Carry*, *Overflow*, and *Count*. *Init* then itself resets to FALSE.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.

- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

OP. Counter output, in 7-digit floating point format. Available for connecting to blocks in a strategy and for linking to bargraphs in graphical displays. The accuracy of *OP* is restricted to 7 digits, compared to the 8 digits offered by the integral *Count* parameter.

Count. Counter value, in integer format. The value displayed has full 8-character integer resolution and can be linked to graphics displays.

Carry. Sets TRUE when the total in *Count* has reached its 'end value' (defined by *UpCount* and *SetCount*), and then been reset to its defined 'start value'. When *Count* has moved away from its 'start value' (after one more input pulse), *Carry* automatically resets to FALSE. Initialising the block via *Init* also resets *Carry*.

Overflow. Latches to TRUE when the counter has reached its 'end value' and been reset to its 'start value'. The flag is reset when *Init* is TRUE, or when *ClearOv* is TRUE.

HR_OP, LR_OP. High and low range for graphic objects linked to *OP* (Bar, Trend). *HR_OP* and *LR_OP* define the 100% and 0% displays, respectively.

Options.

- **CanDsable.** Determines whether the disable feature operates. If TRUE, the disable feature operates normally. If FALSE, the disable feature will only stop counting when *Count* is either equal to *SetCount* (when counting down) or 0 (when counting up) - that is, the value that *Count* attains when *Init* is asserted. In this case, the disable feature is effectively turned off because in normal operation, *Count* will never reach *SetCount* or 0 (refer to the *Disable* field) due to the way the rollover operation works (resetting *Count* to *SetCount-1* or 1).

COMPARE: COMPARE BLOCK

Block function

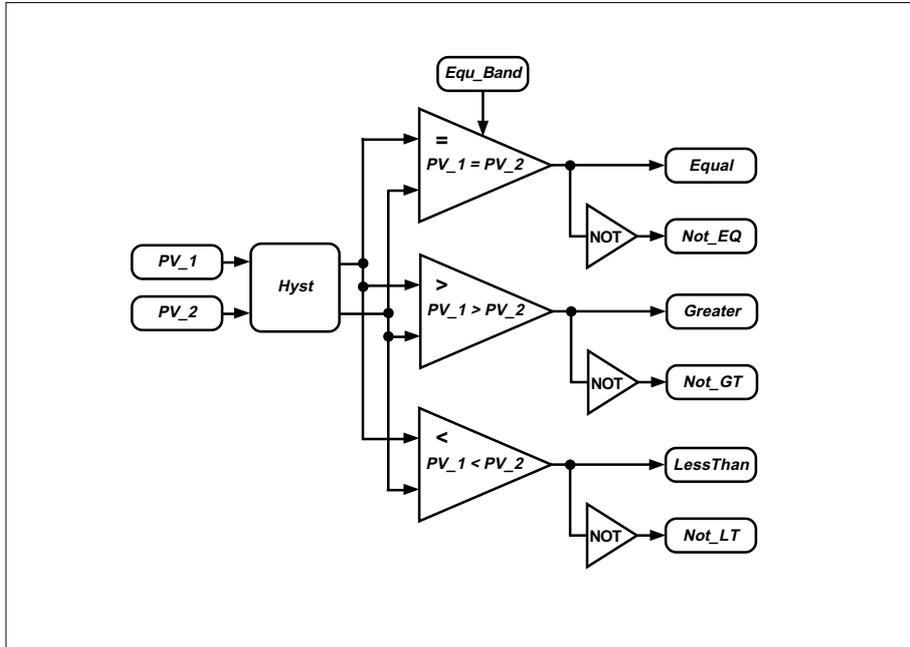


Figure 83 Block schematic

Please refer to *Figure 83*. The COMPARE block compares a pair of analogue inputs, performing three logical functions: Equal To, Greater Than, and Less Than. Three pairs of outputs indicate the logical results of the comparisons in both true and complementary form.

Hysteresis & Equal Bands. A user-set hysteresis band (*Hyst*) operates at each boundary, and an ‘equal band’ (*Equ_Band*) operates additionally in the Equal To function. These parameters act as ‘tolerance bands’ used in the comparisons of the two inputs. *Figure 84*, *Figure 85*, and *Figure 86* show how *Hyst* and *Equ_Band* are applied.

NOTE. In the Figures, the curves must be followed from left to right along the horizontal time axis. They are not valid in the reverse direction.

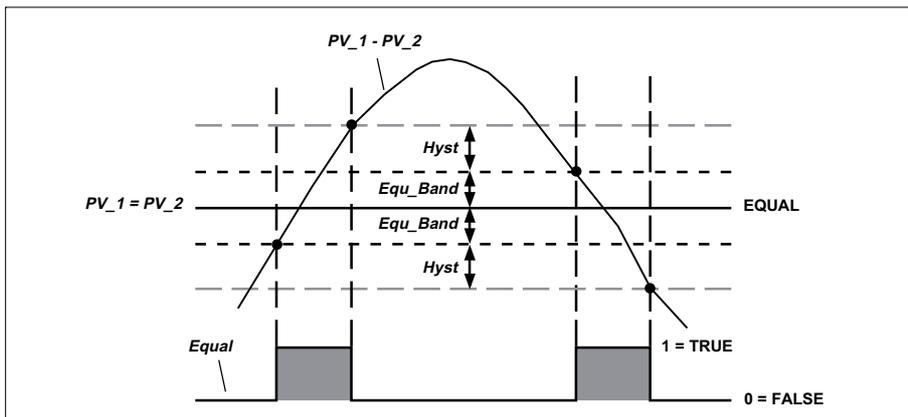


Figure 84 ‘Equal’ action

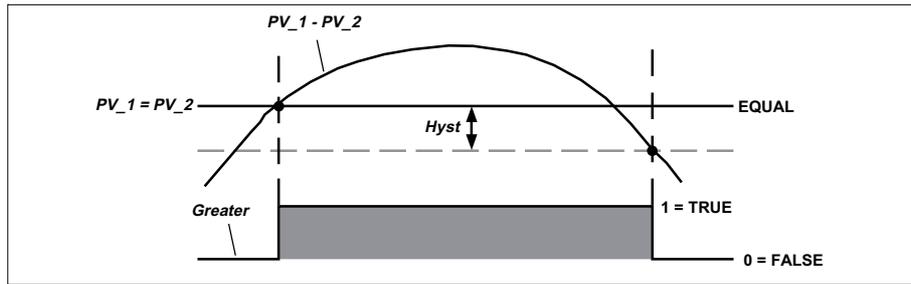


Figure 85 'Greater' action

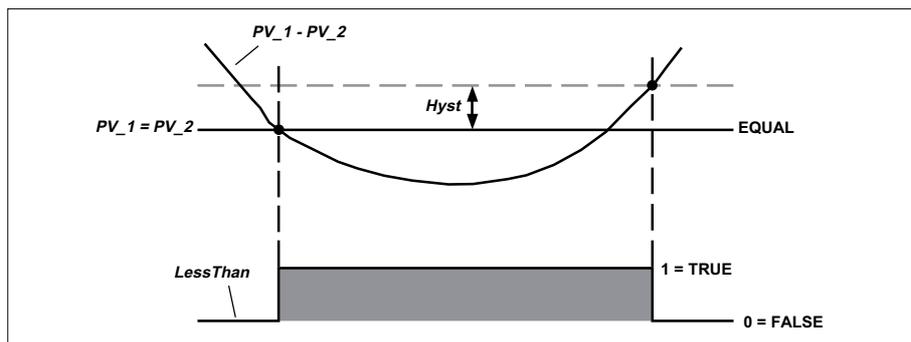


Figure 86 'LessThan' action

Block parameters

Symbols used in *Table 156* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
PV_1	Input 1	Eng	
PV_2	Input 2	Eng	
Hyst	Hysteresis band	Eng	
Equ_Band	Equal band	Eng	
HR, LR	High & low graphics range of PV_1, PV_2	Eng	
Alarms			
Software	Block RAM data sumcheck error/network failure	T/F	
Combined	OR-ing of all Alarms bits	T/F	
Equal	$PV_1 = PV_2$	T/F	
Not_EQ	$PV_1 \neq PV_2$	T/F	
Greater	$PV_1 > PV_2$	T/F	
Not_GT	$PV_1 \nless PV_2$	T/F	
LessThan	$PV_1 < PV_2$	T/F	
Not_LT	$PV_1 \ngtr PV_2$	T/F	

Table 156 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these 'header' fields.

PV_1, PV_2. Inputs 1 and 2, respectively.

Hyst. Hysteresis band. *Figure 84, Figure 85* and *Figure 86* show how *Hyst* is applied.

Equ_Band. Equal band. Specifies a symmetrical 'tolerance' band where *PV_1* and *PV_2* are defined as being equal. See *Figure 85*.

HR, LR. High and low range for graphic objects linked to *PV_1* and *PV_2* (Bar, Trend). *HR* and *LR* define the 100% and 0% displays, respectively.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

■ **Software.** Sumcheck error in block's RAM data.

■ **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

Equal. TRUE indicates that *PV_1* and *PV_2* are within the *Equal* band, subject to hysteresis.

Not_EQ. Logical NOT *Equal*. Complementary output form of *Equal*.

Greater. TRUE indicates that $PV_1 > PV_2$, subject to hysteresis.

Not_GT. Logical NOT *Greater*. Complementary output form of *Greater*.

LessThan. TRUE indicates that $PV_1 < PV_2$, subject to hysteresis.

Not_LT. Logical NOT *LessThan*. Complementary output form of *LessThan*.

CHAPTER 13 MATHS FUNCTION BLOCKS

The MATHS category of Function Block Templates provides the control strategy with functions for mathematical calculations.

ADD2: ADD BLOCK
SUB2: SUBTRACT BLOCK
MUL2: MULTIPLY BLOCK
DIV2: DIVIDE BLOCK

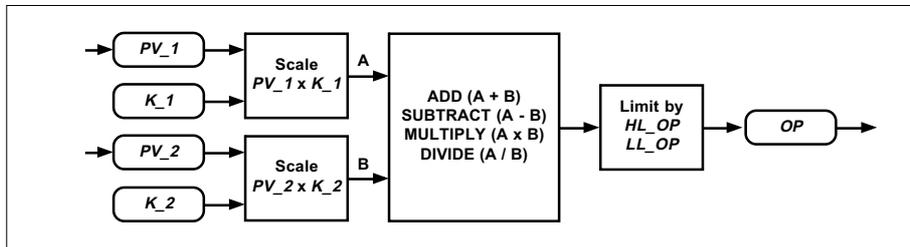


Figure 87 Block schematic

Block function

Please refer to the schematic in *Figure 87*, which applies to all four blocks. Maths blocks carry out the basic arithmetic functions on a pair of scaled analogue inputs. The resulting output *OP* is subjected to high and low limits.

Block parameters

The parameters are the same for all four blocks, listed together in *Table 157*. Symbols used in the table are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
PV_1	Input 1	Eng1	☐
K_1	Scaling constant for PV_1		☐☐
PV_2	Input 2	Eng2	☐
K_2	Scaling constant for PV_2		☐☐
OP	Output	Eng3	☐☐☐
HL_OP, LL_OP	High & low output limits	Eng3	☐☐☐ ?
Alarms			☐☐☐ ☐ ?
Software	Block RAM data sumcheck error/network failure	T/F	
Combined	OR-ing of all Alarms bits	T/F	

Table 157 Maths Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

PV_1, PV_2. Input values 1 and 2, respectively. May be positive or negative.

K_1, K_2. Scaling constants for inputs 1 and 2, respectively. May be positive or negative.

OP. Output value from the calculation.

NOTE. In the DIV2 block, 0/0 is calculated as 0.

HL_OP, LL_OP. High and low output limits, applied to *OP*.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

EXPR: EXPRESSION BLOCK

Block function

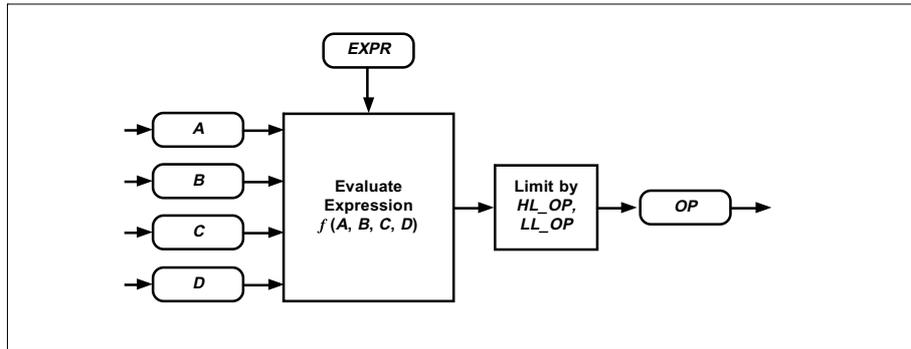


Figure 88 Block schematic

Please refer to *Figure 88*. The Expression block allows entry (via, e.g., an Eycon on-screen alphanumeric keyboard) of an expression of up to 78 characters long, manipulating up to four input variables *A*, *B*, *C*, and *D*. The result is subjected to high and low limits before being output as *OP*. *Table 158* lists in descending priority order the mathematical operators that can be used in the expression.

Evaluation order

The mathematical precedence is from left to right of an expression, then as follows:

- () brackets - *highest priority*
- – (negate), NOT (logical invert), standard function (e.g. TRUNC, FLOAT, etc.)
- ** (exponentiate)
- * (multiply), / (divide), MOD (modulo)
- + (add), – (subtract)
- =, <, >, <=, >=, <> (comparators)
- AND (logical AND)
- XOR (exclusive OR)
- OR (logical OR) - *lowest priority*

For example, $A * B + C / D$ is processed as $(A * B) + (C / D)$, but $A + B * C - D$ is processed as $A + (B * C) - D$.

Operator	Purpose	Format
()	Brackets (alter precedence)	(A+B)*(C+D)
–	Negate	–A
NOT	Logical invert	NOT(A)
TRUNC	Truncate (to an integer)	TRUNC(A)[3]
FLOAT	Convert to Floating Point	FLOAT(A)
ROUND	Round (up or down)	ROUND(A)
ABS	Absolute	ABS(A)
DEG	Degrees (from Radians)	DEG(A)
RAD	Radians (from degrees)	RAD(A)
SIN	Sine (Radians)	SIN(A)
COS	Cosine (Radians)	COS(A)
TAN	Tangent (Radians)	TAN(A)
ASIN	Arc Sine (Radians)	ASIN(A)
ACOS	Arc Cosine (Radians)	ACOS(A)
ATAN2	Arc Tangent (Radians)	ATAN2(A,B) [Tangent=A/B]
SQRT	Square Root	SQRT(A)
LN	Natural Logarithm (base e)	LN(A)
LOG	Logarithm (base 10)	LOG(A)
EXP	Exponentiation (eA)	EXP(A)
MIN	Minimum value	MIN(A,B,C,...)

Continued...

Operator	Purpose	Format
<i>Continued...</i>		
MAX	Maximum value	MAX(A,B,C,...)
AVG	Average (arithmetic mean)	AVG(A,B,C,...)
RANDOM	Random value	RANDOM(A) [A=max modulus]
SWITCH	Selects A (integer C=0) else B	SWITCH(A,B,C[2])
**	Exponentiate (Power AB)	A**B[1]
*	Multiply (x)	A*B
/	Divide (+)	A/B
MOD	Modulo (remainder of A/B as integer)	TRUNC(A) MOD TRUNC(B)[2][3]
+	Add	A+B
-	Subtract	A-B
=	Equals	A=B
<	Less Than	A	Greater Than	A>B
<=	Less Than Or Equal To	A<=B
>=	Greater Than Or Equal To	A>=B
<>	Not Equals	A<>B
AND	Logical AND	A AND B
XOR	Logical XOR (exclusive OR)	A XOR B
OR	Logical OR	A OR B

[1] 'A' must exceed zero.

[2] 'A' and 'B' must be integers - using TRUNC ensures this.

[3] Returns integer; see Number Formats below.

Table 158 Block operators (in order of precedence)

Number formats

All inputs and outputs between the expression block and other blocks in the strategy must be in floating point format. If required, numbers can be converted into integers using TRUNC and then converted back to floating point using FLOAT, after evaluation of the expression.

For example, TRUNC returns an integer, if used to remove a decimal, the resulting number must be converted to floating point FLOAT(TRUNC(A)). A value of A=3.4 would result in OP=3.

Trigonometric functions

The radian is the default unit, but trigonometric functions can be evaluated in degrees using expressions such as:

SIN(RAD(A)) for Sine (degrees), and DEG(ASIN(A)) for Arc Sine (degrees).

Calculating limits

Brackets can be nested up to 38 levels, and up to 19 variables may be included in an expression (repetitions of A, B, C, and D). The power calculation x**y is performed using logarithms and so does not work for negative x-values.

Calculation errors

If the calculation cannot be performed Alarms.Software is set TRUE, and OP takes the value of the last valid result calculated.

Block parameters

Symbols used in Table 159 are explained in Table 1. Additional parameter information is given in the Block specification menu section following.

Parameter	Function	Units	Status
EXPR	Expression	Alphanumeric	
A	Input A	EngA	▶□
B	Input B	EngB	▶□
C	Input C	EngC	▶□
D	Input D	EngD	▶□
Alarms			▶□
Software	Block RAM data sumcheck error/network failure	T/F	▶□
Combined	OR-ing of all Alarms bits	T/F	▶□
OP	Output		▶□
HL_OP, LL_OP	High & low output limits		▶□ ?

Table 159 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

EXPR. The alphanumeric mathematical expression evaluated by the block. Expressions can contain up to 78 characters, including *A, B, C,* and *D,* together with operators and numbers.

NOTE The *EXPR* field as displayed in the block Specification Menu (before entering) shows only the first ten characters in the expression.

A, B, C, D. Inputs *A, B, C, D.* The arguments of the expression defined in the *EXPR* parameter.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

■ **Software.** Sumcheck error in block’s RAM data.

■ **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

OP. Output from the expression, after limiting by *HL_OP* and *LL_OP*.

HL_OP, LL_OP. High and low output limits, applied to the result of the expression before it becomes *OP*.

ACTION: ACTION BLOCK

Block function

The ACTION block allows a sequence-type ‘action’, separately created in the LINtools package, to be run in the control strategy. Please refer to the *T500 LINtools Product Manual* (Part no. HA 082 377 U999) for detailed information on configuring Structured Text actions.

The block provides sets of floating-point variables, integers, and digitals that can be used by the action, and fields specifying the action’s qualifier and associated qualifier time (where appropriate). A digital input provides the means to enable the action, and the elapsed time since the action was enabled is available as an output.

Block parameters

Symbols used in *Table 160* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

ActName. Specifies the name of an action contained in a file called *FileName.STO*; *FileName* is defined in the next section.

Actions are stored in a compiled and resolved format in the .STO file.

FileName. Specifies the root name (with extension .STO) of the file containing the action specified by the *ActName* parameter. *FileName* can be any valid filename. Note that the .STO file must reside on the E: drive of the instrument.

A0 to A7. Eight floating-point variables that can be incorporated into the structured text defining the action. These variables can be inputs or outputs via the control strategy.

D. Bitfield containing 16 digital variables, *Bit0* to *Bit15*, that can be incorporated into the structured text defining the action. These variables can be inputs or outputs via the control strategy.

T. Time elapsed since the action enable parameter (*EN*) became TRUE, in units specified by the *TimeBase* parameter. The timer controlling *T* stops running when *EN* goes FALSE, and restarts from zero when *EN* is set to TRUE again. The block uses *T* to time the starting and stopping of *L*-, *D*-, and *E*-qualified actions by comparing its current value with *QualTime*.

EN. Action enabling input. Setting *EN* TRUE zeroes and starts the *T* timer, and setting it FALSE stops the *T* timer, which holds its value until restarted. The state of *EN* controls the running of the action, as shown in *Figure 89*.

NOTE A FALSE-to-TRUE transition on EN is parallel to the activation of a step in a sequence, and a TRUE-to-FALSE transition is parallel to the step’s de-activation.

Parameter	Function	Units	Status
ActName	8-character (max.) action name	Alphanumeric	
FileName	8-character (max.) filename	Alphanumeric	
A0 to A7	Floating point variables	Eng	◀▶
D	Digital point variables	ABCD hex	◀▶
Bit0	Digital 0	T/F	D
Bit1	Digital 1	T/F	
Bit2	Digital 2	T/F	
Bit3	Digital 3	T/F	
Bit4	Digital 4	T/F	C
Bit5	Digital 5	T/F	
Bit6	Digital 6	T/F	
Bit7	Digital 7	T/F	
Bit8	Digital 8	T/F	B
Bit9	Digital 9	T/F	
Bit10	Digital 10	T/F	
Bit11	Digital 11	T/F	
Bit12	Digital 12	T/F	A
Bit13	Digital 13	T/F	
Bit14	Digital 14	T/F	
Bit15	Digital 15	T/F	

Continued...

DIGACT: DIGITAL ACTION BLOCK

Block function

The DIGACT block allows a user-defined ‘action’, separately created in the LINtools Action configurator, to be run in the control strategy. Please refer to the *T500 LINtools Product Manual* (Part no. HA 082 377 U999) for detailed information on configuring Structured Text Actions.

NOTE. The DIGACT block template is stored in a sub-directory called DEVICES, accessible via *SelDir* in the LINtools MATHS category menu.

The block provides sets of digital variables, individual and grouped (into 8-bit bytes and 16-bit words), that can be used by the action, and fields specifying the action’s qualifier and associated qualifier time (where appropriate). A digital input provides the means to enable the action, and the elapsed time since the action was enabled is available as an output.

Note that the DIGACT block is very similar to the ACTION block, described in the previous section, and has several identical parameters.

Block parameters

Symbols used in *Table 161* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

ActName. Specifies the name of an action contained in a file called *FileName.STO*; *FileName* is defined in the next section.

Actions are stored in a compiled and resolved format in the .STO file.

FileName. Specifies the root name (with extension .STO) of the file containing the action specified by the *ActName* parameter. *FileName* can be any valid filename. Note that the .STO file must reside on the E: drive of the instrument.

T. Time elapsed since the action enable parameter (*EN*) became TRUE, in units specified by the *TimeBase* parameter. The timer controlling *T* stops running when *EN* goes FALSE, and restarts from zero when *EN* is set to TRUE again. The block uses *T* to time the starting and stopping of *L*-, *D*-, and *E*-qualified actions by comparing its current value with *QualTime*.

EN. Action enabling input. Setting *EN* TRUE zeroes and starts the *T* timer, and setting it FALSE stops the *T* timer, which holds its value until restarted. The state of *EN* controls the running of the action, as shown in *Table 160* (previous page).

Note that a FALSE-to-TRUE transition on *EN* is parallel to the activation of a step in a sequence, and a TRUE-to-FALSE transition is parallel to the step’s de-activation.

Parameter	Function	Units	Status
ActName	8-character (max.) action name	Alphanumeric	
FileName	8-character (max.) filename	Alphanumeric	
T	Elapsed time since EN parameter TRUE		
EN	Action-enabling input	T/F	
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
NoAction	Unable to find named action	T/F	
BadActn	Runtime evaluation error	T/F	
Combined	OR-ing of all Alarms bits	T/F	
Qual	Action qualifier	Menu	
QualTime	Qualifier time		
TimeBase	Time units for T and QualTime	Menu	
Bool_A to Bool_X	Digital variables	T/F	

Continued...

Parameter	Function	Units	Status
<i>Continued...</i>			
Word0 to Word3	16-bit digital variables	ABCD hex	↔ □
Bit0	Digital 0	T/F	D
Bit1	Digital 1	T/F	
Bit2	Digital 2	T/F	
Bit3	Digital 3	T/F	
Bit4	Digital 4	T/F	C
Bit5	Digital 5	T/F	
Bit6	Digital 6	T/F	
Bit7	Digital 7	T/F	
Bit8	Digital 8	T/F	B
Bit9	Digital 9	T/F	
Bit10	Digital 10	T/F	
Bit11	Digital 11	T/F	
Bit12	Digital 12	T/F	A
Bit13	Digital 13	T/F	
Bit14	Digital 14	T/F	
Bit15	Digital 15	T/F	
Byte0 to Byte3	8-bit digital variables	CD hex	↔ □
Bit0	Digital 0	T/F	D
Bit1	Digital 1	T/F	
Bit2	Digital 2	T/F	
Bit3	Digital 3	T/F	
Bit4	Digital 4	T/F	C
Bit5	Digital 5	T/F	
Bit6	Digital 6	T/F	
Bit7	Digital 7	T/F	

Table 161 Block parameters

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.
- **NoAction.** Asserted if the block cannot find the action named in *ActName*.
- **BadActn.** Asserted if an evaluation error occurs during the running of the action.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

Qual. (P(Initial)/N(Normal)/L(Limited)/D(Delayed)/E(Event)/F(Final)) The action qualifier. *Qual* specifies when the action is to run during the time that the enabling input *EN* is TRUE, as shown in *Table 160*.

QualTime. Time value associated with the *L*, *D*, and *E* action qualifiers, with units specified by *TimeBase*. *QualTime* specifies a running time or delay for the action, and operates during the period that the *EN* parameter is TRUE, as shown in *Table 160*.

TimeBase. (Secs/Mins/Hours) Specifies time units for the *T* and *QualTime* parameters.

Bool_A to Bool_X. 24 digital variables that can be incorporated into the structured text defining the action. These variables can be inputs or outputs via the control strategy.

Word0 to Word3. Four bitfields each containing 16 digital variables, *Bit0* to *Bit15*, that can be incorporated into the structured text defining the action. These variables can be inputs or outputs via the control strategy.

Byte0 to Byte3. Four bitfields each containing 8 digital variables, *Bit0* to *Bit7*, that can be incorporated into the structured text defining the action. They can be inputs or outputs via the control strategy.

ACT_2A2W3T: ACTION BLOCK WITH GATED DOWNTIMERS

Block function

The ACT_2A2W3T ACTION block type allows a sequence-type ‘action’, created separately in LINtools, to be run in the strategy. Please refer to the *LINtools Help file* (Part no. RM 263 001 U055) for detailed information on configuring Structured Text actions.

This block type provides 2 analogue floating-point variables, 2 Word fields, 3 Down Timers with individual enables and disables, and 6 User Alarms that can be used by the action. A digital input provides the means to enable the action.

Block parameters

Symbols used in *Table 162* are explained in *Table 1* of the *LIN Block Reference Manual* (Part no. HA 082 375 U003). Additional parameter information is given in the *Block specification menu* section following.

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* of the *LIN Block Reference Manual*, Part No. HA 082 375 U003, for details of these ‘header’ fields.

ActName. Specifies the 8 character name of an action contained in a file called *FileName.sto*; *FileName* is defined in the next section.

Actions are stored in a compiled and resolved format in the .sto file.

FileName. Specifies the 8 character root name (with extension .sto) of the file containing the action specified by the *ActName* parameter. *FileName* can be any valid filename.

NOTE When using this block in the LIN database, the .sto file specified in this field must also be downloaded to instrument.

A1, A2. Two floating-point variables that can be incorporated into the Structured Text defining the action. These variables can be inputs or outputs via the control strategy.

Parameter	Function	Units	Status
ActName	8-character (max.) action name	Alphanumeric	
FileName	8-character (max.) filename	Alphanumeric	
A1, A2	Floating point variables		
Word1, Word2	Word variable bits	ABCD hex	
Bit0	Digital 0	T/F	D
Bit1	Digital 1	T/F	
Bit2	Digital 2	T/F	
Bit3	Digital 3	T/F	
Bit4	Digital 4	T/F	C
Bit5	Digital 5	T/F	
Bit6	Digital 6	T/F	
Bit7	Digital 7	T/F	
Bit8	Digital 8	T/F	B
Bit9	Digital 9	T/F	
Bit10	Digital 10	T/F	
Bit11	Digital 11	T/F	
Bit12	Digital 12	T/F	A
Bit13	Digital 13	T/F	
Bit14	Digital 14	T/F	
Bit15	Digital 15	T/F	
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
NoAction	Unable to find named action	T/F	
BadActn	Runtime evaluation error	T/F	
UserAlm1	User alarms initiated from ST	T/F	
UserAlm2		T/F	
UserAlm3		T/F	
UserAlm4		T/F	
UserAlm5		T/F	
UserAlm6		T/F	
Combined	OR-ing of all Alarms bits	T/F	

Continued...

Parameter	Function	Units	Status
<i>Continued...</i>			
EN	Action enabling input	T/F	<input type="checkbox"/>
DwnTmr1	3 Down Timers	Secs	<input type="checkbox"/>
DwnTmr2			
DwnTmr3			
TmrEnabl		(ABC)D hex	<input type="checkbox"/>
DwnTmr1	DwnTmr1 enabled when TRUE	T/F	D 1 2 4 8
DwnTmr2	DwnTmr2 enabled when TRUE	T/F	
DwnTmr3	DwnTmr3 enabled when TRUE	T/F	
		T/F	

Table 162 Block parameters

Word1, Word2. Bitfields containing 16 variables, *Bit0* to *Bit15*, that can be incorporated into the Structured Text defining the action. These variables can be inputs or outputs via the strategy.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.
- **NoAction.** Asserted if the block cannot find the action named in *ActName*.
- **BadActn.** Asserted if an evaluation error occurs during the running of the action.
- **UserAlm1 to UserAlm6.** User alarms, asserted, set TRUE, and cleared from the Structured Text action specified in *ActName*.
- **Combined.** Asserted if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

EN. Action enabling input. The state of *EN* controls the running of the action. TRUE, runs the configured action, FALSE stops the action, see *Figure 90*.

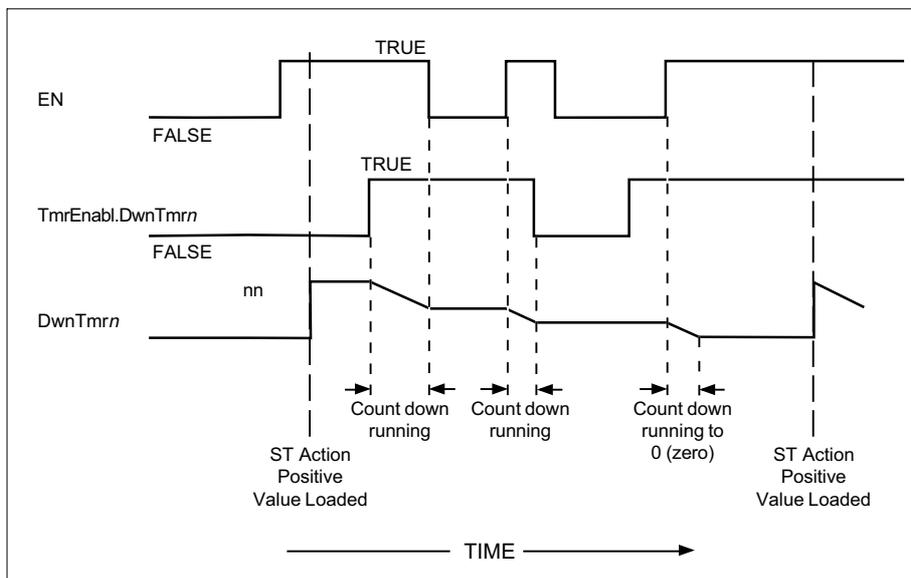


Figure 90 DwnTmr operation

DwnTmr1, DwnTmr2, DwnTmr3. *DwnTmr1*, *DwnTmr2*, and *DwnTmr3*, can be loaded with a value from the Structured Text (ST) Action or from a Sequential Function Chart (SFC) Structured Text (ST) Action, User Screen Editor, etc.. These fields count down to 0 (zero) when loaded with a positive value. Individual enabling subfields, *TmrEnabl.DwnTmrn*, allow these fields to run when set TRUE, but will stop when the corresponding *TmrEnabl.DwnTmrn* is set FALSE.

TmrEnabl. Bitfield containing, *DwnTmr1*, *DwnTmr2*, and *DwnTmr3*, that are used to enable, set TRUE, or stop, set FALSE, the corresponding Downtimer. *DwnTmr1*, *DwnTmr2*, and *DwnTmr3* can only countdown when *EN* is set TRUE.

CHAPTER 14 OPERATOR FUNCTION BLOCKS

The OPERATOR category of Function Block Templates provides the control strategy with functions for controlling the operator interface related activity in the supported instrument(s), e.g. the Visual Supervisor.

- The PNL_CMD (panel command) block takes command of the panel, e.g. forces a jump to a specified page.
- The PNL_DICT (panel dictionary) block provides write access to the Writeable Text dictionary of the Eycon™ Visual Supervisor.
- The PNL_MSG (panel message) block creates a message.
- The PNL_DLG (panel dialogue) block creates a dialogue box.
- The PNL_ACC (panel access) block enables and monitors user logons.
- The READER block enables the use of barcode readers and similar devices.
- The EVENT block indicates the occurrence of a Visual Supervisor event, e.g. disqualification of an access account.

PNL_CMD: PANEL COMMAND BLOCK

Block function

The PNL_CMD block is used typically to force the Visual Supervisor panel to jump to a specified page. This may be a *user page* (Id 1-999) or a built-in *agent* (Id > 999).

The block's *Action* parameter includes the following options:

- **GOTO.** Jump to the specified page.
- **DESCEND.** ‘Descend’ to the specified page, with an ‘escape’ key allowing return to the original page.
- **ESCAPE.** Undo a ‘descend’, or clear a popup window. (Equivalent to the ‘page up’ key).

Other actions are also available (see *Block specification menu* section below) which could be used, for example, to construct rolling demonstrations.

Performing a user interface command

A particular sequence of parameter value changes must be followed for successful user interface interaction. The required changes in the relevant PNL_CMD block can be carried out using a Sequence (SFC), a user action list, or by wired inputs from the running control strategy.

A typical one-shot page-change, ensuring the new page is reached, could proceed as follows:

1. Set **Option.Lock** TRUE to suppress navigation driven directly from the panel itself, and also to suppress timeouts.
2. Set **Acquire** TRUE to initiate acquisition of command of the user interface.
3. Wait for **Acquired** to become TRUE. This confirms that no other PNL_CMD function block is already in command of the interface.
4. NOTE You can also make acquisition conditional on the user's access level and/or status, via the *Access* subfields.
5. Set **Action** to *GoTo*, and set **Id** to the appropriate value for the desired page.
6. Set **Trigger** from FALSE to TRUE, to trigger the selected action.
7. Wait for **Triggered** to set, then (optionally) check for any alarms.
8. Reset **Acquire** and **Trigger** to relinquish command and prepare for further triggering.

In applications having only one PNL_CMD block and not requiring locking, it is simpler to leave the block permanently in command (*Acquire* held TRUE), preconfigure *Action* and *Id*, and pulse *Trigger* TRUE then FALSE whenever the action is required. Alternatively, you can leave *Trigger* TRUE and pulse *Acquire* to trigger the action.

Block parameters

Symbols used in *Table 163* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
PanelId	Id of panel (<i>Default = 1. Not currently implemented</i>)	Integer	
Acquire	TRUE acquires user interface command, FALSE releases it	T/F	☐
Trigger	TRUE executes the selected action (latching)	T/F	☐
Action	Selects the required action	Menu	
Id	Id of destination page or agent	Integer	☐
Context	Block context, where applicable	Blockname	
DispMode	Display mode, where applicable (0, 1-255)	Integer	
Access	Access/user status	(AB)CD hex	
LOCKED	TRUE enables acquisition in LOCKED access level	T/F	D
OPERATOR	TRUE enables acquisition in OPERATOR access level	T/F	
COMMISSI	TRUE enables acquisition in COMMISSION access level	T/F	
ENGINEER	TRUE enables acquisition in ENGINEER access level	T/F	

Continued...

- **PREV.** Equivalent to the ‘page left’ key, when this causes a page change.
- **HOME.** Jump to the home page.
- **ROOT.** Jump to the root menu. Equivalent to the ‘root’ (menu) key.

Id. Id (identity number) of destination page or agent. User pages, that you build with the User Screen Editor, have *Id* in the range 1-999; preconfigured Built-in Agents have *Id* >999. For Built-in Agents, there are some cases where contextual information (e.g. block name, display mode) may be needed to fully specify the target page. (See the *Context* and *DispMode* parameters, next.)

Context. Block context (where applicable). This block name field may further specify the destination page, in addition to the selected screen *Id*.

Id	Agent	Context parameter value
1500	System View	<i>(Not applicable)</i>
1501	Area View	AREA blockname <i>(Not currently applicable)</i>
1502	Group View*	GROUP blockname
1503	Point View	Point to be displayed

**DispMode also applicable*

Table 164 Context parameter values for area/group/point agents

Table 164 lists components of the Visual Supervisor area/group/point agent where the *Context* parameter may generally be applied, and the block types providing the context.

Id	Operator message agent display format
9020	Ordered list format. Lists only messages specified by Context
9021	Ordered list format. Lists all messages <i>(Context not applicable)</i>
9022	Array format. Shows only messages specified by Context
9023	Array format. Shows all messages <i>(Context not applicable)</i>

Table 165 Operator message agent Id values and formats

The *Context* parameter can also be applied to Operator Message agents to control what messages are to be included in the displays. Table 165 lists *Id* values and the corresponding message formats, and shows how *Context* works.

EXAMPLE. With *Id* = **9022** and *Context* = **FIC-010** (blockname), the destination page would display in array format only those operator messages associated with the block called ‘FIC-010’. If the block specified by *Context* is an AREA or GROUP block, messages associated with all blocks included in the area/group are displayed.

DispMode. Display mode required (where applicable) for the specified *Id* and *Context*. Display modes correspond to the different views of the same set of data, e.g. the various trend formats of the Visual Supervisor.

DispMode	Display
0	Default view
1	Mimic
2	Faceplate
3	Numeric
4	Vertical bargraphs
5	Horizontal bargraphs
6	Vertical trend with faceplates
7	Vertical trend without faceplates
8	Horizontal trend with faceplates
9	Horizontal trend without faceplates

Table 166 DispMode parameter values

Table 166 lists available Visual Supervisor display mode numbers currently in use, and their meanings. (*DispMode* can take values from 1 to 255. ‘0’ indicates ‘default view’.)

NOTE Currently, the *DispMode* parameter applies only to Visual Supervisor’s *Group View* page, which has a default *Id* of 1502.

EXAMPLE. With *Id* = **1502** (Group View page), *Context* = **preplot** (GROUP category blockname), and *DispMode* = **8**, the destination page would display a horizontal trend with faceplates for the points in the group defined by the GROUP block called ‘preplot’.

Access. Bitfield controlling user access to acquisition of the user interface. The following is given in addition to *Table 163*.

- **UserRef.** If TRUE, acquisition is enabled only if the *UserRef* field value equals the ‘User Reference’ of the user currently logged in. This condition must be met *in addition to* any access level limitations imposed by the other *Access* subfields.

NOTE The Visual Supervisor has two security access systems, *Standard Access* and *User ID Access*. In the User ID Access system, every user is assigned a (non-unique) ‘User Reference’ number. This is the number referred to here.

Options. Bitfield controlling user interface navigation options. The following is given in addition to *Table 163*.

- **Lock.** If TRUE when *Acquire* is also TRUE, *Lock* suppresses navigation driven directly from the panel itself, e.g. via the root menu. It also suppresses panel timeouts, e.g. the timeout to the home page. There may be an additional delay in acquisition in this case, because a few built-in pages also use locking (e.g. during cloning or upgrade.)

NOTE This option must be used with great care to avoid locking up the user interface irretrievably! Also that *Option.Lock* is relevant only at the time of acquisition, toggling it subsequently has no effect.

PNL_DICT: PANEL DICTIONARY BLOCK

Block function

This block is an interface between the Eycon™ Visual Supervisor panel and the dictionary files. It provides a view to the dictionaries and allows the ‘W’ (Writable Text) dictionary to be written to. The availability is this function allows long text strings used in Batch data to be read and written to by a remote computer.

Block parameters

Symbols used in *Table 167* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Dict	Dictionary used	String	
Index	Index into dictionary	Integer	
Lang	Language of defined dictionary	Integer	
Alarms			  
Software	Block RAM data sumcheck error / network failure	T/F	
Invalid	Invalid Dict/Index configuration	T/F	
Combined	OR-ing of all Alarms bits	T/F	
Text1 to Text16	Text strings in defined dictionary	String	

Applicable to Version 3.0 onwards

Table 167 Block parameters

Block specification menu

The following is given in addition to *Table 167*.

Dict. This defines the dictionary that will be read from and/or written to by the instrument.

NOTE Only supports writes to the ‘W’ (Writable Text) dictionary.

Index. Index number of the text string in the dictionary defined in *Dict*.

Lang. Defines the language variant of the dictionary defined in *Dict*. The default values indicating the language variant are as shown.

Language	Numeric reference
English	0
French	1
German	2
Italian	3
Spanish	4
Portugese	5
Dutch	6

Alarms. See *page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.
- **Invalid.** Asserted if invalid configuration of *Dict* and/or *Index* is detected. This could be caused if the defined dictionary does not exist.
- **Combined.** Asserted if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

Text1 to Text16. Sixteen consecutive text string fields in dictionary starting at the number defined by *Index*. *Text1* relates to the number defined in *Index*, *Text2* to *Text16* relate to the next text strings configured in the dictionary. If a *Textn* field shows ‘’, no text exists in the defined dictionary. Each field supports 255 characters maximum per field.

PNL_MSG: PANEL MESSAGE BLOCK

Block function

PNL_MSG blocks let you configure and queue messages that the application can present to the operator for viewing on Visual Supervisor's front panel. These messages can for example be used to prompt the operator for a particular response, or simply to inform him/her of something that needs only an acknowledgement. The strategy itself can be configured to respond to messages in the absence of operator responses. The block allows the strategy to record, and if necessary act on, all operator responses.

Message setup & queuing. A message is set up via the block's *Title* field, which specifies the message banner, and the *Body* field, specifying the message contents, both as dictionary entries.

NOTE All text messages for this block are stored in a special *message* ('M') *dictionary*. A dictionary entry value of '0' is interpreted as 'no text'. This does not prevent references to other dictionaries within an associated 'form'.

A TRUE input to *Trigger* adds the configured message to the message queue, where it remains 'active' and available for operator viewing and response until automatically deleted after *Timeout* seconds have elapsed. (*Timeout* can be disabled to allow the message to remain queued indefinitely.) The message is immediately deleted if the operator responds to it by pressing a message 'button', or if the application itself responds automatically.

Messages are queued (by default) in creation-time order, with the oldest message first in the queue for the operator's attention. The *Priority* parameter can however be used to override this and move a message up the queue.

Operator buttons. A maximum of four operator 'buttons' can appear at the foot of the message. These are selected by the *Buttons* parameter and include 'OK', 'CANCEL', 'YES', 'NO', etc. plus four free-format user-defined buttons with dictionary-defined legends (allowing for multi-lingual applications). The *Response* parameter records which button has been pressed by the operator, or if the application has responded automatically, which button(s) have been activated. The *SelfResp* parameter specifies which button(s) will be activated by the application when triggered by a TRUE input to the *TrigSelf* field.

Message context. Each message can be given a LIN function block 'context', via the *Context* parameter. This allows the operator to view, and the application to use, the queued messages selectively, filtered by context, as an alternative to globally. Note that other function blocks, e.g. the PNL_CMD block, may also use this context to filter messages.

NOTE An application can have several PNL_MSG block running, each block being able to support one active message in the message queue at one time.

Block parameters

Symbols used in *Table 168* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
PanelId	Id of panel (<i>Default = 1. Not currently implemented</i>)	Integer	
Trigger	TRUE adds the message to the queue (latching)	T/F	<input checked="" type="checkbox"/>
Title	Dictionary entry for message title (20 chars. max.)	Integer	
Body	Dictionary entry for message text	Integer	
FormFile	(Not currently implemented)	Alphanumeric	
FormId	(Not currently implemented)	Integer	<input checked="" type="checkbox"/>
Options	Optional features	(A)BCD hex	<input checked="" type="checkbox"/>
Panel	TRUE (default) enables onscreen message box generation	T/F	1 2 4 8 D
Event	TRUE (default) enables message/response event logging	T/F	
TimeStmp	TRUE includes timestamp in displayed message	T/F	
Sign ^[1]	TRUE = signature required in Auditor option to acknowledge	T/F	
Buttons	Selects buttons to appear on message, enables timeout	(A)BCD hex	
OK	'OK' button (Default = TRUE)	T/F	1 2 4 8 D
CANCEL	'CANCEL' button	T/F	
YES	'YES' button	T/F	
NO	'NO' button	T/F	

Continued...

Parameter	Function	Units	Status
<i>Continued...</i>			
ABORT	'ABORT' button	T/F	1
UButton1	User button 1	T/F	2
UButton2	User button 2	T/F	4
UButton3	User button 3	T/F	8
UButton4	User button 4	T/F	1
Timeout	Timeout enable (Default = TRUE)	T/F	2
		T/F	4
		T/F	8
UButton1 - 4	Dictionary legends for user buttons 1 - 4	Integer	
Timeout	Time message remains in queue (secs); 0 = disable	Integer	
Priority	Message queue-order priority, 1 - 15. (Default = 1)	Integer	
Emphasis	Emphasis applied to message title (to get attention)	Menu	
Context	Function block context of message (if any)	Blockname	
TrigSelf	Triggers 'self-response' & clears message. (Latching)	T/F	<input type="checkbox"/>
SelfResp	Specifies self-response (all buttons) when <i>TrigSelf</i> TRUE	(A)BCD hex	<input type="checkbox"/>
OK	'OK' button (Default = TRUE)	T/F	1
CANCEL	'CANCEL' button	T/F	2
YES	'YES' button	T/F	4
NO	'NO' button	T/F	8
ABORT	'ABORT' button	T/F	1
UButton1	User button 1	T/F	2
UButton2	User button 2	T/F	4
UButton3	User button 3	T/F	8
UButton4	User button 4	T/F	1
Timeout	Force message timeout (when <i>TrigSelf</i> asserted)	T/F	2
		T/F	4
		T/F	8
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	<input type="checkbox"/>
Nbuttons	Too many buttons requested (limit = 4)	T/F	
Dictnry	Dictionary error	T/F	
TextLen	Piece of text in the dictionary was too long for context	T/F	
Form	Form file error (see <i>FormErr</i> , <i>FormErLn</i> , <i>FormErCh</i>)	T/F	
Failure	Failed to queue the message	T/F	
Combined	OR-ing of all Alarms bits	T/F	
Triggered	Trigger acknowledgement	T/F	<input type="checkbox"/>
Response	Flags operator 'pressed' buttons or self-response bits	(A)BCD hex	<input type="checkbox"/>
OK	'OK' button 'pressed'	T/F	1
CANCEL	'CANCEL' button 'pressed'	T/F	2
YES	'YES' button 'pressed'	T/F	4
NO	'NO' button 'pressed'	T/F	8
ABORT	'ABORT' button 'pressed'	T/F	1
UButton1	User button 1 'pressed'	T/F	2
UButton2	User button 2 'pressed'	T/F	4
UButton3	User button 3 'pressed'	T/F	8
UButton4	User button 4 'pressed'	T/F	1
Timeout	Message was cleared due to a timeout	T/F	2
		T/F	4
		T/F	8
Status	Message status	(ABC)D hex	<input type="checkbox"/>
Active	TRUE = message active, whether viewed or not	T/F	1
OnView	TRUE = message active & being viewed onscreen	T/F	2
		T/F	4
		T/F	8
FormErr	Form error code	Menu	
FormErLn	Line on which error flagged by <i>FormErr</i> occurred	Integer	
FormErCh	Character position of error in line flagged by <i>FormErLn</i>	Integer	

[1] Applicable to Version 4.1 onwards

Table 168 Block parameters

Block specification menu

The following is given in addition to *Table 168*.

Trigger. Asserting the *Trigger* input generates a message (according to your specifications) and adds it to the queue of active messages awaiting operator attention. *Trigger* is latching, and must be reset FALSE before re-use. Note that setting *Trigger* to TRUE resets all bits in the *Response* bitfield.

Title. Dictionary entry for message title, up to 20 characters. This appears as the message ‘banner’ and can be given more or less emphasis to attract operator attention (via the *Emphasis* parameter).

Buttons. Selects what buttons will appear at the foot of the message window (and also enables the message timeout). The following is given in addition to *Table 168*.

- **UButton 1-4.** These bits let you select up to four free-format user buttons with legends specified by the corresponding *UButton1* to *UButton4* dictionary fields.
- **Timeout.** Set this bit TRUE (the default) to enable the message timeout, i.e. the time the unacknowledged message remains in the queue, specified by the *Timeout* field. With the *Timeout* bit FALSE, the timeout is disabled and the message remains indefinitely in the queue until acknowledged (by the operator or the application).

NOTE You can use this bit to disable the timeout without deleting the timeout value from the *Timeout* parameter.

Timeout. Time (seconds) that the message remains in the queue, available for operator (or application) acknowledgement. When this time has elapsed the message is automatically deleted. Note that the message is immediately deleted if the operator responds to it by pressing a message ‘button’ (or if the application itself responds to the message). To disable, set *Timeout* to zero. The message now remains indefinitely in the queue until acknowledged.

Priority. Message queue-order priority (1 - 15, default = 1, the lowest priority). Messages are normally queued in creation-time order, with the oldest message first in the queue for the operator’s attention. The *Priority* parameter can however be used to override this and move a newer message up the queue, because priority takes precedence over age. If messages have equal priorities (the default case), the oldest message takes precedence.

Emphasis. (NONE/WEAK/STRONG/ATTN) Selects the emphasis applied to the message title, to get operator attention.

- **NONE.** No emphasis (‘Logo green’ title).
- **WEAK.** Weak emphasis (green title).
- **STRONG.** Strong emphasis (red title).
- **ATTN.** Maximum ‘attention’ emphasis (flashing red title).

Context. Lets you specify a LIN function block tagname as the context of the message, i.e. to be associated with the message. A blank field denotes ‘no particular context’. This allows the operator to view, and the application to use, the queued messages selectively, filtered by context, as an alternative to globally. Note that other function blocks, e.g. the PNL_CMD block, may also use this context to filter messages.

TrigSelf. This input allows the application (strategy) to respond to the buttons in a message instead of relying on operator responses. Specifically, a TRUE input to *TrigSelf* copies the bit-pattern of the *SelfResp* parameter onto the *Response* parameter. This is equivalent to ‘pressing’ automatically all the buttons specified in *SelfResp*. When *TrigSelf* is asserted the message is deleted from the queue (as it is when responded to by an operator).

NOTE *TrigSelf* is latching, and must be reset FALSE before re-use.

SelfResp. Specifies the message’s ‘self-response’, i.e. what button(s) will effectively be ‘pressed’ when the application sets the *TrigSelf* input TRUE. Specifically, a TRUE input to *TrigSelf* copies the bit-pattern of the *SelfResp* parameter onto the *Response* parameter. This is equivalent to automatically ‘pressing’ all the buttons specified in *SelfResp*.

Self-response can be used by the application as an alternative to waiting for operator response. Note that, unlike operator response, self-response can ‘press’ several buttons simultaneously, and can also activate buttons that have not been enabled by the *Buttons* parameter, and so are not actually seen in the message box.

Triggerd. This output sets TRUE after *Trigger* has been asserted and the resulting new message has been successfully added to the queue. It can be used as a handshake to confirm message-creation.

Response. This read-only bitfield records operator (or automatic) message box responses. The corresponding bit(s) set(s) TRUE when the operator presses a button, or when an automatic self-response is triggered by an input to *TrigSelf*. The bits latch TRUE until the *Trigger* input is asserted, when all bits reset FALSE (and a new message is generated).

PNL_DLG: PANEL DIALOGUE BLOCK

Block function

PNL_DLG blocks let you configure and queue dialogues that the application forces the operator to view and respond to on Visual Supervisor's front panel. These dialogues can for example be used to prompt the operator for a particular response, or simply to inform him/her of something that needs only an acknowledgement. The strategy itself can be configured to respond to dialogues in the absence of operator responses. The block allows the strategy to record, and if necessary act on, all operator responses.

Dialogue setup & queuing. A dialogue is set up via the block's *Title* field, which specifies the dialogue banner, and the *Body* field, specifying the dialogue contents, both as dictionary entries.

NOTE All text messages for this block are stored in a special *message* ('M') *dictionary*. A dictionary entry value of '0' is interpreted as 'no text'. This does not prevent references to other dictionaries within an associated 'form'.

A TRUE input to *Trigger* adds the configured dialogue to the dialogue queue, where it remains 'active' and ready to appear on the screen for the operator to respond to. The currently-displayed dialogue is immediately deleted if the operator responds to it by pressing a dialogue 'button', or if the application itself responds automatically.

Dialogues are queued (by default) in creation-time order, with the oldest dialogue (if any) currently on display. The *Priority* parameter can however be used to override this and move a dialogue up the queue. For dialogues of equal priority, the oldest takes precedence. Unless the current dialogue is responded to by the operator (or automatically), **no other operator actions** may be carried out. When a dialogue is cleared, the next dialogue in the queue (if one exists) pops up.

Operator buttons. A maximum of four operator 'buttons' can appear at the foot of the dialogue. These are selected by the *Buttons* parameter and include 'OK', 'CANCEL', 'YES', 'NO', etc. plus four free-format user-defined buttons with dictionary-defined legends (allowing for multilingual applications). The *Response* parameter records which button has been pressed by the operator, or if the application has responded automatically, which button(s) have been activated. The *SelfResp* parameter specifies which button(s) will be activated by the application when triggered by a TRUE input to the *TrigSelf* field.

NOTE An application can have several PNL_DLG block running, each block being able to support one active dialogue in the dialogue queue at one time.

Block parameters

Symbols used in *Table 169* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
PanelId	Id of panel (<i>Default = 1. Not currently implemented</i>)	Integer	
Trigger	TRUE adds the dialogue to the queue (latching)	T/F	☐
Title	Dictionary entry for dialogue title (20 chars. max.)	Integer	
Body	Dictionary entry for dialogue text	Integer	
FormFile	(<i>Not currently implemented</i>)	Alphanumeric	
FormId	(<i>Not currently implemented</i>)	Integer	☐
Options	Optional features	(ABC)D hex	☐
PlsWait	TRUE adds flashing 'Please wait...' text to dialogue	T/F	D 1 2 4 8
Progress	TRUE adds progress bar, driven by <i>Progress</i> input	T/F	
		T/F	
Event ^[1]	TRUE (default) enables message/response event logging	T/F	
Buttons	Selects buttons to appear on dialogue	(A)BCD hex	
OK	'OK' button (<i>Default = TRUE</i>)	T/F	D 1 2 4 8
CANCEL	'CANCEL' button	T/F	
YES	'YES' button	T/F	
NO	'NO' button	T/F	

Continued...

Parameter	Function	Units	Status
<i>Continued...</i>			
ABORT	'ABORT' button	T/F	1
UButton1	User button 1	T/F	2
UButton2	User button 2	T/F	4
UButton3	User button 3	T/F	8
UButton4	User button 4	T/F	1
		T/F	2
		T/F	4
		T/F	8
UButton1 - 4	Dictionary legends for user buttons 1 - 4	Integer	
Progress	Optional progress indicator I/P (<i>Options.Progress</i> enables)	Eng	
Priority	Dialogue queue-order priority, 16 - 31. (Default = 16)	Integer	
Emphasis	Emphasis applied to dialogue title (indicates urgency)	Menu	
TrigSelf	Triggers 'self-response' & clears dialogue. (Latching)	T/F	<input type="checkbox"/>
SelfResp	Specifies self-response (all buttons) when <i>TrigSelf</i> TRUE	(A)BCD hex	<input type="checkbox"/>
OK	'OK' button (Default = TRUE)	T/F	1
CANCEL	'CANCEL' button	T/F	2
YES	'YES' button	T/F	4
NO	'NO' button	T/F	8
ABORT	'ABORT' button	T/F	1
UButton1	User button 1	T/F	2
UButton2	User button 2	T/F	4
UButton3	User button 3	T/F	8
UButton4	User button 4	T/F	1
		T/F	2
		T/F	4
		T/F	8
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
Nbuttons	Too many buttons requested (limit = 4)	T/F	
Dictnry	Dictionary error	T/F	
TextLen	Piece of text in the dictionary was too long for context	T/F	
Form	Form file error (see <i>FormErr</i> , <i>FormErLn</i> , <i>FormErCh</i>)	T/F	
Failure	Failed to queue the dialogue	T/F	
Combined	OR-ing of all Alarms bits	T/F	
Triggered	Trigger acknowledgement	T/F	<input type="checkbox"/> <input type="checkbox"/>
Response	Flags operator 'pressed' buttons or self-response bits	(A)BCD hex	<input type="checkbox"/> <input type="checkbox"/>
OK	'OK' button 'pressed'	T/F	1
CANCEL	'CANCEL' button 'pressed'	T/F	2
YES	'YES' button 'pressed'	T/F	4
NO	'NO' button 'pressed'	T/F	8
ABORT	'ABORT' button 'pressed'	T/F	1
UButton1	User button 1 'pressed'	T/F	2
UButton2	User button 2 'pressed'	T/F	4
UButton3	User button 3 'pressed'	T/F	8
UButton4	User button 4 'pressed'	T/F	1
		T/F	2
		T/F	4
		T/F	8
Status	Dialogue status	(ABC)D hex	<input type="checkbox"/> <input type="checkbox"/>
Active	TRUE = dialogue active, whether viewed or not	T/F	1
OnView	TRUE = dialogue active & being viewed onscreen	T/F	2
		T/F	4
		T/F	8
FormErr	Form error code	Menu	<input type="checkbox"/>
FormErLn	Line on which error flagged by <i>FormErr</i> occurred	Integer	<input type="checkbox"/>
FormErCh	Character position of error in line flagged by <i>FormErLn</i>	Integer	<input type="checkbox"/>

Table 169 Block parameters

[1] Applicable to Version 4.2.

Block specification menu

The following is given in addition to *Table 169*.

Trigger. Asserting the *Trigger* input generates a dialogue (according to your specifications) and adds it to the queue of active dialogues awaiting operator attention. *Trigger* is latching, and must be reset FALSE before re-use. Note that setting *Trigger* to TRUE resets all bits in the *Response* bitfield.

Title. Dictionary entry for dialogue title, up to 20 characters. This appears as the dialogue ‘banner’ and can be given more or less emphasis to indicate the level of urgency to the operator (via the *Emphasis* parameter).

Options. Optional dialogue features.

- **PlsWait.** TRUE adds a flashing ‘Please wait...’ legend to the dialogue, or whatever text is specified by system dictionary entry 89 (S89). This can be used, for example, in a dialogue that forces the operator to wait until a particular process has completed before allowing him/her to proceed. No operator actions, other than responding to a dialogue button (if any are enabled), can be carried out while a dialogue is on display. It can help to include a ‘progress bar’ (see next) on the dialogue together with *PlsWait*, if this is possible.
- **Progress.** TRUE adds a horizontal ‘progress bar’, driven by *Progress* input. If you want this feature, wire a suitable input from the strategy. The indication ranges from 0.0 to 100.0.

Buttons. Selects what buttons will appear at the foot of the dialogue window. The following is given in addition to *Table 169*.

- **UButton1-4.** These bits let you select up to four free-format user buttons with legends specified by the corresponding *UButton1* to *UButton4* dictionary fields.

Priority. Dialogue queue-order priority (16 - 31, default = 16, the lowest dialogue priority). Dialogues are normally queued in creation-time order, with the oldest dialogue first in the queue for the operator’s attention. The *Priority* parameter can however be used to override this and move a newer dialogue up the queue, because priority takes precedence over age. If dialogues have equal priorities (the default case), the oldest dialogue takes precedence.

Emphasis. (NONE/WEAK/STRONG/ATTN) Selects the emphasis applied to the dialogue title, to get operator attention and indicate the level of urgency of the dialogue.

- **NONE.** No emphasis (‘Logo green’ title).
- **WEAK.** Weak emphasis (green title).
- **STRONG.** Strong emphasis (red title).
- **ATTN.** Maximum ‘attention’ emphasis (flashing red title).

TrigSelf. This input allows the application (strategy) to respond to the buttons in a dialogue instead of relying on operator responses. Specifically, a TRUE input to *TrigSelf* copies the bit-pattern of the *SelfResp* parameter onto the *Response* parameter. This is equivalent to ‘pressing’ automatically all the buttons specified in *SelfResp*. When *TrigSelf* is asserted the dialogue is deleted from the queue (as it is when responded to by an operator).

NOTE *TrigSelf* is latching, and must be reset FALSE before re-use.

SelfResp. Specifies the dialogue’s ‘self-response’, i.e. what button(s) will effectively be ‘pressed’ when the application sets the *TrigSelf* input TRUE. Specifically, a TRUE input to *TrigSelf* copies the bit-pattern of the *SelfResp* parameter onto the *Response* parameter. This is equivalent to automatically ‘pressing’ all the buttons specified in *SelfResp*.

Self-response can be used by the application as an alternative to waiting for operator response. Note that, unlike operator response, self-response can ‘press’ several buttons simultaneously, and can also activate buttons that have not been enabled by the *Buttons* parameter, and so are not actually seen in the dialogue box.

Triggerd. This output sets TRUE after *Trigger* has been asserted and the resulting new dialogue has been successfully added to the queue. It can be used as a handshake to confirm dialogue-creation.

Response. This read-only bitfield records operator (or automatic) dialogue box responses. The corresponding bit(s) set(s) TRUE when the operator presses a button, or when an automatic self-response is triggered by an input to *TrigSelf*. The bits latch TRUE until the *Trigger* input is asserted, when all bits reset FALSE (and a new dialogue is generated).

PNL_ACC: PANEL ACCESS BLOCK

Block function

The PNL_ACC block allows an application to control the logging on/off process, both for Visual Supervisor’s ‘standard’ access system and its ‘multi-user’ access system.

For the standard system, the logon is set up using the *ReqLevel* parameter to select the required access level, e.g. ENGINEER, and the *Password* parameter to store the relevant password. For the multi-user system, *ReqID* and *Password* store the required user ‘Identity’ and password respectively. (Access level is not required for multi-user logon.)

A TRUE input to the *LogOn* field then ‘logs on’ the preconfigured user or sets the required access level. The *LogOff* input is used to ‘log off’ the current user or reset the access level to LOCKED.

Please refer to the instrument’s Product Manual for full details of the access systems.

Block parameters

Symbols used in *Table 170* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
PanelId	Id of panel (<i>Default = 1. Not currently implemented</i>)	Integer	
LogOn	TRUE triggers logon (latching)	T/F	▶ <input type="checkbox"/>
LogOff	TRUE triggers logoff (latching)	T/F	▶ <input type="checkbox"/>
ReqID	Requested user ID (multi-user strategy) (2 - 8 chars.)	Alphanumeric	
ReqLevel	Requested access level (standard strategy)	Menu	
Password	Password for new user (displayed as “”) (0 - 8 chars.)	Alphanumeric	
Options	Optional features	(ABCD) hex	▶ <input type="checkbox"/>
LOnHome	Return to home page at logon	T/F	D
LOffHome	Return to home page at logoff	T/F	
		T/F	
		T/F	
Alarms			▶ <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
Software	Block RAM data sumcheck error / network failure	T/F	
InvLogon	Invalid logon attempted, e.g. incorrect password	T/F	
Conflict	Logon rejected because of a conflict	T/F	
Combined	OR-ing of all Alarms bits	T/F	
LoggedOn	TRUE if logged on (level > LOCKED), else FALSE	T/F	▶ <input type="checkbox"/> <input type="checkbox"/>
CurID	Current user ID, or blank if logged off	Alphanumeric	<input type="checkbox"/> <input type="checkbox"/>
CurLevel	Current access level name	Menu	<input type="checkbox"/> <input type="checkbox"/>
CurLvlNo	Current access level number (1 - 5)	Integer	<input type="checkbox"/> <input type="checkbox"/>
CurRef	Current user reference number (0 - 65535)	Integer	<input type="checkbox"/> <input type="checkbox"/>
CurAttr ^[1]	Current user attributes	ABCD hex	▶ <input type="checkbox"/>
Sign	TRUE = can sign	T/F	D
Authorse	TRUE = can authorise	T/F	
ViewOnly	TRUE = View Only access	T/F	
MaxExpiry	TRUE = Password expiry period set to maximum	T/F	
AdmOnly ^[2]	TRUE = edit Administrator access rights ONLY	T/F	C
		T/F	
		T/F	
		T/F	
		T/F	B
		T/F	
		T/F	
		T/F	
FTP ^[2]	TRUE = can access FTP	T/F	A
Remote ^[2]	TRUE = can login to this database from a remote node	T/F	
		T/F	
		T/F	
CurAttr ^[2]	Additional User attributes	(ABCD) hex	▶ <input type="checkbox"/>
User1		T/F	D
User2		T/F	
User3		T/F	
User4		T/F	

[1] Block parameters Applicable to Version 4. onwards. [2] Applicable to Version 5.0 onwards

Table 170 Block parameters

Block specification menu

The following is given in addition to *Table 170*.

ReqLevel, CurLevel. (LOCKED/OPERATOR/COMMISSI/ENGINEER/ADMIN) Requested access level (standard access system), and current access level (both systems), respectively.

- **LOCKED.** Locked access level.
- **OPERATOR.** Operator access level.
- **COMMISSI.** Commissioning access level.
- **ENGINEER.** Engineer access level.
- **ADMIN.** Administrator access level (multi-user access system only).

Options. Optional access features.

- **LOnHome.** Return to the home page when logging on. In the standard strategy, logging on means changing the access level from LOCKED to a higher level.
- **LOffHome.** Return to the home page when logging off. In the standard strategy, logging off means changing the access level from a higher level to LOCKED.

CurLvlNo. Outputs the current access level (*CurLevel*) as an integer in the range 1 - 5. *Table 171* lists the *CurLvlNo* values and their corresponding *CurLevel* names.

CurLvlNo	CurLevel
1	LOCKED
2	OPERATOR
3	COMMISSI(ON)
4	ENGINEER
5	ADMIN

Table 171 CurLvlNo values and equivalent access level names

CurRef. Outputs the current user reference number (multi-user access system only) as an integer.

CurAttr. Current user attributes (multi-user access system only).

- **SIGN.** Outputs confirmation that the current user has Sign access rights.
- **AUTHORSE.** Outputs confirmation that the current user has Authorise access rights.
- **VIEWONLY.** Outputs confirmation that the current user has View Only access rights.
- **MAXEXPRY.** Outputs confirmation that the current user password is set to the maximum expiry period.
- **ADMONLY.** Outputs confirmation that the current user, if an Administrator, has View Only access rights to functions not within the constraints of the Administrator access rights.
- **FTP.** Outputs confirmation that the current user can access the FTP (File Transfer Protocol).
- **REMOTE.** Outputs confirmation that the current user password can access this database from a remote node using Telnet communications.

CurAttr2. Selected user attributes (multi-user access system only).

- **USER1 to USER4.** Outputs selected users specific attributes.

- **LINE_LEN.** A line in the file was too long (limit is 255 excluding CR and/or LF).
- **NEWLINE.** File does not end with a newline (possibly indicating corruption).
- **MEMORY.** File is too big for the memory allocated to it.
- **SYNTAX.** There is a syntax error of some kind.
- **RANGE.** A numeric value is out of its permitted range.
- **NAME.** A named object cannot be found (possibly misspelled).
- **DICTNRY.** A dictionary entry cannot be found.
- **TYPE.** An object type is inappropriate in the context in which it is being used.
- **ACTION.** (*Not implemented.*)
- **CONTEXT.** A construct was used which is invalid in the context of Reader Interface Language files.
- **OTHER.** Other error.

EVENT: EVENT BLOCK

Block function

The EVENT block flags up the occurrence of an Eycon™ 10/20 Visual Supervisor, e.g. disqualification of an access account, and provides a means for the strategy to respond to the event. Events are recorded in the instrument’s alarm history/audit trail.

The block traps events based on either event *number* (flagging a single specified number) or else event *priority* (matching all events at or above a specified priority). It may also trap events associated with specific function blocks.

If the event occurs, the block’s output Boolean *Out* latches TRUE. Setting the block’s *Reset* input TRUE resets *Out*. While *Reset* remains TRUE, all new events are ignored (but buffered). When both *Out* and *Reset* are again FALSE, the block is ready to match another event.

The block also matches and buffers events that occur while the database is stopped or unloaded. When the LIN Database restarts the buffered events are flagged up.

A particular use for the EVENT block is to trigger an alarm on an event such as a failed login. A second EVENT block can then be used to generate the clear signal on successful login, which is itself another event. In this way an alert is created on a failed login that persists until a login is successful.

Block parameters

Symbols used in *Table 173* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Disable	TRUE = event matching disabled	T/F	
FiltType	Selects types of event to match	(AB)CD hex	
(Unused)		T/F	D
(Unused)		T/F	
(Unused)		T/F	
(Unused)		T/F	
EvtSys	Match system (non-block) events [Default = TRUE]	T/F	C
EvtBlk	Match block events (specified by <i>FiltBlk</i>) [Def. = TRUE]	T/F	
EvtNote	Match notepad events [Default = FALSE]	T/F	
EvtChng	Match block change events (specified by <i>FiltBlk</i>) [Def. = FALSE]	T/F	
FiltEvNo	Number of event to match [0 = match all events]	Integer	
FiltPri	Minimum priority of matched event [0-15, default 1]	Integer	
FiltBlk	Name of block in which events are to be matched	Alphanumeric	
Reset	TRUE clears <i>Out</i> , ready to match another event	T/F	
Init	TRUE resets all counters, excluding 'Total', to zero	T/F	
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
Out	Follows the <i>Out</i> parameter	T/F	
Missed	TRUE = block of events missed. Clears on matched event	T/F	
Combined	OR-ing of all Alarms bits	T/F	
Out	Sets TRUE when an event is matched	T/F	
Date	Date of matched event	dd/mm/yy	
Time	Time of matched event	hh:mm:ss	
EventNo	Event number of matched event	Integer	
EventPri	Priority of matched event	Integer	
Block	Block associated with event	Alphanumeric	
Name1	First eight characters of matched event’s context	Alphanumeric	
Name2	Second eight characters of matched event’s context	Alphanumeric	
WhoBy	User ID of instigator of event	Alphanumeric	
WhoAuth	User ID of authorisor of event	Alphanumeric	
Count	Count of matches	Integer	
Missed	Number of events missed (not processed)	Integer	
Total	Total number of events since powerup	Integer	

Table 173 Block parameters

Block specification menu

The following is given in addition to *Table 173*.

FiltEvNo. Number of event to be matched. If left blank (0), *all* events are allowed through for matching. Only matched event numbers are allowed through.

NOTE A full list of possible events and corresponding event numbers are available in the *Eycon™ 10/20 Visual Supervisor Handbook, Part No. HA029280*.

FiltBlk. Name of block in which events are to be matched. If left blank, *all* events are allowed through for matching. If a block is named, only events associated with the specified block are allowed through. If the named block is a GROUP block, only events in the GROUP block or in a block associated with any of its *Disp1-16* or *Chan1-16* fields are allowed through. If the named block is an AREA block, only events in the AREA block or in any of its GROUP blocks or groups are allowed through.

NOTE AREA/GROUP blocks specified in *FiltBlk* may or may not be used for display purposes.

WhoBy. User ID of the instigator of the event. This is either the ID of the logged-in user, or if signed for, the ID of the person signing.

CHAPTER 15 ORGANISE FUNCTION BLOCKS

The ORGANISE category of Function Block Templates provides the control strategy with functions for organizing system screens and grouping data for logging. The blocks fall into two groups, *area/group* blocks comprising the AREA and GROUP blocks, and *data logging* blocks comprising the LOGDEV, LGROUP, LOGRPEX, LPTDEV, and PGROUP blocks.

Area/Group blocks

The AREA and GROUP blocks are used to configure the area and group database needed for display navigation and flash memory data recording in the Eycon™ 10/20 Visual Supervisor.

The block forms a two-tier hierarchy and can contain up to 16 groups. Each group consists of up to 16 display items and 16 recording channels. Overlapping groups and areas are possible, i.e. a point may appear in more than one group and even more than once within a single group.

The blocks include alarm summary outputs so that the alarm status of the group/area can be monitored.

Figure 91 illustrates this hierarchy. The example shows an area (*Area1*) which has two groups (*Zone1* and *Zone2*) each with two PIDs in the display group, and recording the SP and PV of both PIDs.

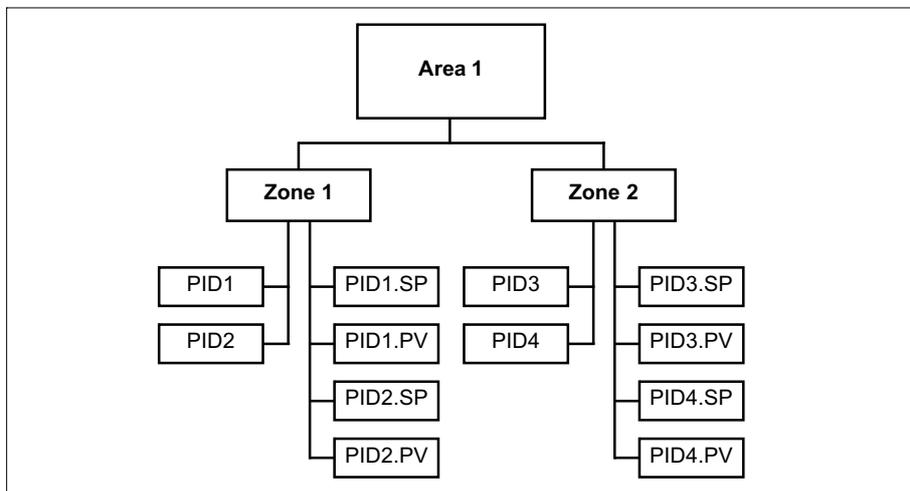


Figure 91 Example of the Area/Group hierarchy

Eycon™ 10/20 Visual Supervisor preplots (Legacy Programmer Only)

The legacy programmer is described in more detail in Chapter 16 - refer to page 435 for further details.

An important function of the AREA and GROUP blocks is to organise setpoints for display in the ‘preplot’ view offered by the Eycon™ 10/20 Visual Supervisor. Refer to the *Eycon™ 10/20 Visual Supervisor Handbook*, (Part No. HA029280, Appendix D4.2 for details on the use of this display.) Configuring the blocks for this task is described in this chapter under *Using GROUP block channels to organise Eycon preplot* page 425.

Data logging blocks

The LOGDEV and LGROUP blocks organise analogue and digital point blocks into log groups for data logging to a local archive medium. Up to 16 points can be associated with each log group, and up to 16 log groups can be logged onto a single logging device.

Figure 93 shows schematically how the ORGANISE blocks are arranged to collect data from the point blocks (DR_DGCHP and DR_ANCHP) and route it to the logging device.

AREA: AREA BLOCK

Block function

The AREA block can associate up to 16 GROUP blocks within it and is also where the available group display modes are configured. The block also generates a set of area alarms. *Figure 91* showed an example area/group hierarchy, and *Figure 92* shows in more detail how the block is linked to other blocks in the ORGANISE and RECORDER categories.

Displays. The ‘Area Overview’ user screen is a representation of all the GROUP blocks associated within the selected AREA block’s *Group1* to *Group16* fields. At the bottom of the display hierarchy, the ‘Group Overview’ and ‘Point/Loop View’ screens are determined by what display block types are linked to the GROUP blocks via their *Disp1* to *Disp16* fields. See *Figure 92*.

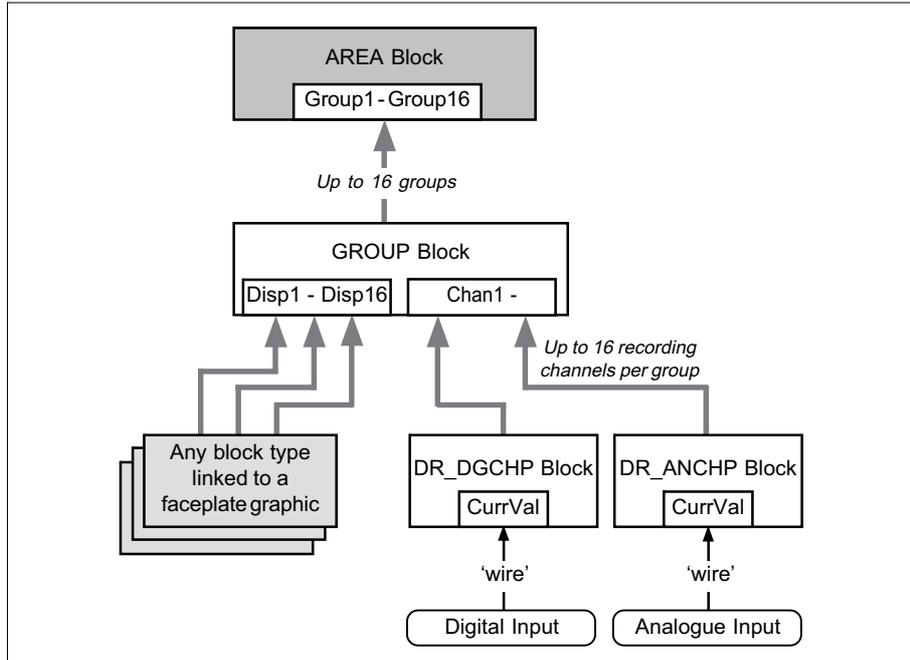


Figure 92 Organisation of recording & display blocks into groups and areas

Block parameters

Symbols used in *Table 174* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Id	Area number 0 or 1 (0 = disable, the default)	Integer	
Group1-Group16	Name of associated GROUP block	Alphanumeric	
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
Config	TRUE = configuration error (e.g. invalid Id or Groupn)	T/F	
Area	TRUE = GROUP block is in Group alarm	T/F	
Combined	OR-ing of all Alarms bits	T/F	
AlmAct	TRUE = GROUP block is in alarm	T/F	<input type="checkbox"/> <input type="checkbox"/>
AlmUnack	TRUE = GROUP block has unacknowledged alarm	T/F	<input type="checkbox"/> <input type="checkbox"/>
DispMode	Specifies the available group display modes	(A)BCD hex	
V_Trend	Vertical trend display (default=TRUE)	T/F	D
H_Trend	Horizontal trend display (default=FALSE)	T/F	
FV_Trend	Full width vertical trend display (default=FALSE)	T/F	
V_Bars	Vertical bars display (default=FALSE)	T/F	
H_Bars	Horizontal bars display (default=TRUE)	T/F	C
Numeric	Numeric faceplate display (default=FALSE)	T/F	
Faceplat	Instrument faceplate display (default=TRUE)	T/F	
Mimic	User mimic display (default=FALSE)	T/F	

Continued...

Parameter	Function	Units	Status
<i>Continued...</i>			
FH_Trend	Full width horizontal trend display (default=FALSE)	T/F	B
		T/F	
		T/F	
		T/F	
Mimic	Mimic associated with area (1-999, 0=none, the default)	Integer	
TrendOpt	Specifies trend display presentation options	(AB)CD hex	
DkBgLive	Use dark background when trend live (default=FALSE)	T/F	D
DkBgRevw	Dark background when in review (default=FALSE)	T/F	
Markers	Display time/date markers (default=FALSE)	T/F	
SmoothSc	Use smooth scroll for panning (default=FALSE)	T/F	
Messages	Display chart messages (default=TRUE)	T/F	C
ThickTrc	Display thick traces (default=FALSE)	T/F	
RecrdOpt	Specifies recording options	(ABC)D hex	
AlarmCol	Alarms records are colour-coded (default=TRUE)	T/F	D
MsgCol	Messages are colour-coded (default=FALSE)	T/F	
		T/F	
		T/F	
RecrdAlm	DR_ALARM block to identify alarms/events for recording	Alphanumeric	

Table 174 Block parameters

Block specification menu

The following is given in addition to *Table 174*.

Id. Allocates an Area number (1) to this Area.

Group1 to Group16. Specify the GROUP blocks to be associated with this Area, that will appear in the ‘Area Overview’ display on the Eycon™ 10/20 Visual Supervisor.

Alarms. The area alarm status is available as the output *AlmAct*, and also as the alarm *Alarms.Area*. If you want to monitor the status without generating an alarm, disable *Area*.

DispMode. Specifies which display modes of the group are available.

GROUP: GROUP BLOCK

Block function

The GROUP block lets you associate (up to) 16 display-type blocks and 16 Recorder channel blocks with a group, specify the recording sample rate and mode, and link a mimic to the group. The block also provides a set of group alarms. *Figure 92* shows how the block is associated with others in the ORGANISE and RECORDER block categories.

Block parameters

Symbols used in *Table 175* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units*	Status*
Update	Recording sample rate (1-60 secs, default=0 sec)	Eng	
RecMode	Recording mode (default = NORMAL)	Menu	
Mimic	Mimic associated with group (1-999, 0=none, default)	Integer	
Disp1-Disp16	Name of block sourcing the display point	Alphanumeric	
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
Config	TRUE = configuration error (e.g. invalid Chann or Disp)	T/F	
Group	TRUE = Display source block is in alarm	T/F	
Combined	OR-ing of all Alarms bits	T/F	
AlmAct	TRUE = Display source block is in alarm	T/F	
AlmUnack	TRUE = Display source block has unacknowledged alarm	T/F	
Chan1-Chan16	Name of recorder channel block forming the recording group	Alphanumeric	

Table 175 Block parameters

Block specification menu

The following is given in addition to *Table 175*.

Update. Recording sample rate, in the range 1 - 60 seconds. At the default of zero, no recording occurs.

RecMode. (NORMAL) Recording mode to be used.

Disp1 to Disp16. Specify the display-type blocks to be associated with this Group, that will appear in the 'Group Overview' and 'Loop/Point View' displays on the Eycon™ 10/20 Visual Supervisor.

Alarms. The display block alarm status is available as the output *AlmAct*, and also as the alarm *Alarms.Group*. If you want to monitor the status without generating an alarm, disable *Group*. (Note: This block can also be used to run locally on a T2550 for alarm grouping)

Chan1 to Chan16. Specify the recorder channel blocks, DR_ANCHP and DR_DGCHP, to be associated with this Group. These blocks feed in the analogue and/or digital data to be recorded in the ISE's flash memory area.

Using GROUP block channels to organise Eycon preplot

An important function of the *Chan1* to *Chan16* parameters is to organise setpoints for display in the 'preplot' view offered by the Eycon™ 10/20 Visual Supervisor legacy programmer (refer to *Chapter 16* for further information). Refer to the *Eycon™ 10/20 Visual Supervisor, (Part No. HA029280)*, for details on the use of this display.)

Figure 93 shows an example of how a group of two analogue setpoints and one digital setpoint must be associated with a GROUP block (via DR_ANCHP and DR_DGCHP blocks) to create a valid preplot display, see *Recorder Function Blocks*.

For a valid preplot display, the GROUP block channels must be associated in strict numerical order with the setpoints generated by the running setpoint program. This is done as follows:

1. First, the analogue PVs must be associated in numerical order as defined by the setpoint program. That is, Analogue PV1 (resulting from Analogue Setpoint 1) must be associated with *Chan1*, Analogue PV2 with *Chan2*, etc., until all the analogue PVs have been dealt with, in this example only two.

- Then, any digital setpoints must be associated in program order with the next of the GROUP block's channels. In this example the single Digital Setpoint 1 must be associated with *Chan3*.

NOTE In the preplot display, digital setpoints are displayed as single trends throughout, unlike analogues which appear as PV/SP pairs in the 'past' region. You can in fact choose to display *any* digital(s) via the 'digital' channels, either locally or remotely derived.

- When all the analogue PVs and digitals have been dealt with, the analogue SPs are associated with the remaining channels in order. In this example, Analogue SP1 is associated with *Chan4*, and Analogue SP2 is associated with *Chan5*, to complete the set.

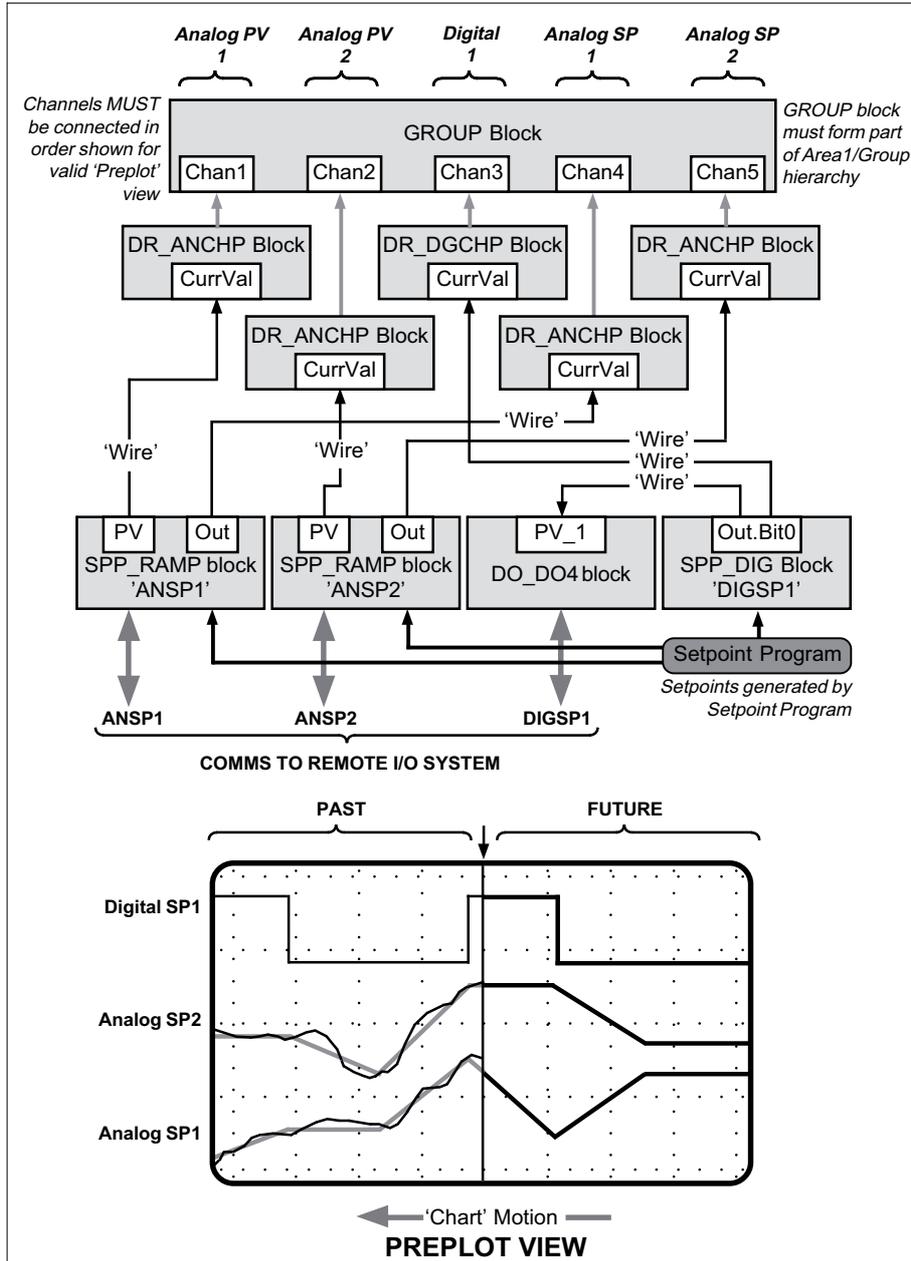


Figure 93 Organising setpoint channels for the EYCON 'preplot' view

LOGDEV: LOGGING DEVICE BLOCK

Block function

This block lets you specify and control access to a specified archive medium, H: (Internal archive) as defined in the Device field, and specify up to 16 groups of points (log groups) that are to be logged on that medium.

IMPORTANT Only one LOGDEV block may be included in any database.

Each log group is in turn specified via a LGROUP block, which collects data from up to 16 DR_DGCHP, DR_ANCHP, DR_ALARM, and DR_REPRT point blocks.

Figure 94 shows how the ORGANISE blocks (shaded) interconnect the point data and logging device.

NOTE The hierarchy shown in Figure 94 coexists with, but is not the same as, the Area/Group hierarchy used by the RECORDER category for flash recording of data (see Figure 92).

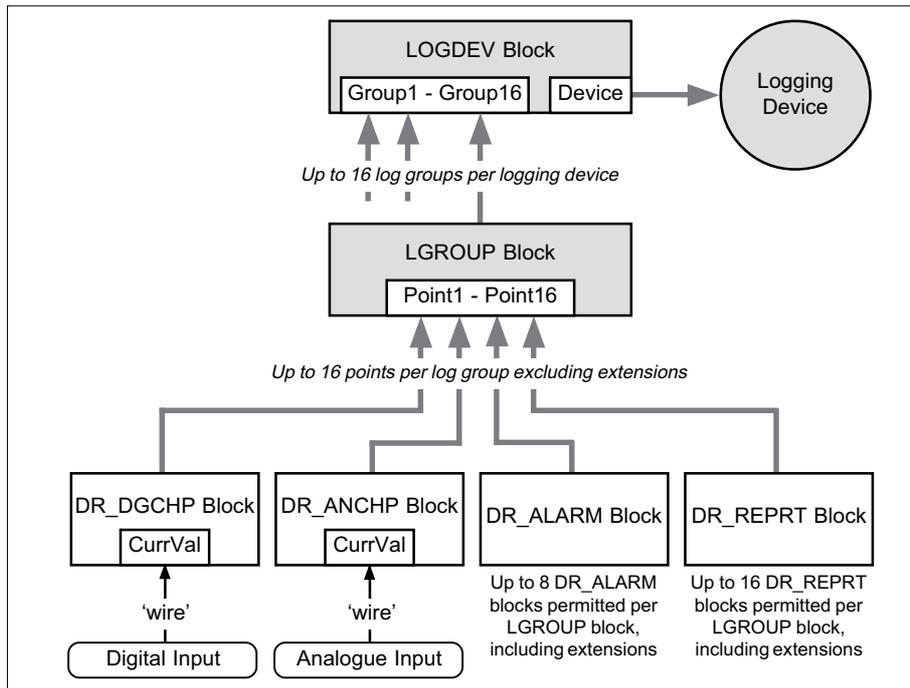


Figure 94 Organising point blocks for data logging

Block parameters

Symbols used in Table 176 are explained in Table 1. Additional parameter information is given in the Block specification menu section following.

Parameter	Function	Units	Status
State	Specify when the medium may be accessed	Menu	
Status	Report current status of logging	Menu	<input type="checkbox"/> <input type="checkbox"/>
Trigger	TRUE enables logging if State = TRIGGER	T/F	<input checked="" type="checkbox"/>
Options	Specify logging options	(A)BCD hex	<input type="checkbox"/>
Del_ASC	TRUE = Delete .ASC files as deletion strategy (default)	T/F	D
Del_PKD	TRUE = Delete .PKD files as deletion strategy (default)	T/F	
Del_UHH	TRUE = Delete .UHH files as deletion strategy (default)	T/F	
		T/F	
New_File	TRUE stops deletion of current logging file on coldstart	T/F	C
		T/F	
		T/F	
		T/F	

Continued...

Parameter	Function	Units	Status
<i>Continued...</i>			
NoCusUHH	TRUE = disable custom batch entries	T/F	<div style="border: 1px solid black; padding: 2px; display: inline-block;"> 1 2 4 8 </div> B
UHHPHase	TRUE = enables inclusion of batch phase records	T/F	
Device	Specify logging device name ('H' - default)	Alphanumeric	
Group1-Group16	Specify LGROUP block names	Alphanumeric	
Alarms 📄 📖 🔊			
Software	Block RAM data sumcheck error / network failure	T/F	
Config	TRUE=configuration error (e.g. invalid log group Id or device name)	T/F	
DiskFull	TRUE = not enough space on archive medium	T/F	
No Disk	TRUE = no archive medium present	T/F	
Disk Err	TRUE = any other disk error occurs	T/F	
Combined	OR-ing of all Alarms bits	T/F	
FreeSpac	Percentage free space on the medium (%)	Eng	📄 📖
FreeTime	Logging time remaining at current rate (hours)	Eng	📄 📖
DiskFull	% of disk full to trigger DiskFull alarm (default=95%)	Eng	

Table 176 Block parameters

Block specification menu

The following is given in addition to *Table 176*.

State. (ON/OFF/TRIGGER) Determines when the medium may be accessed by specifying when logging is permitted. 'ON' (the default) enables, and 'OFF' disables logging. 'TRIGGER' allows the *Trigger* digital input to enable/disable logging.

Status. (INACTIVE/ACTIVE/FLUSHING/OFF_LINE) Reports current status of logging to disk. *Table 177* explains the options.

Option...	...Means
INACTIVE	No files are open
ACTIVE	Files are open
FLUSHING	Actual flush to disk
OFF_LINE	Logging terminated

Table 177 Meanings of Status options

Options. Bitfields used to configure logging options.

- **Del_ASC, Del_PKD, Del_UHH.** See *DiskFull*.
- **New_File.** Setting this bit prevents the deletion of the current .ASC or .PKD logging file that would normally occur at cold startup. Instead, the current files are closed, and new logging files are opened at coldstart, with .ASC extensions modified to AS1 ... AS9, A10 ... A99, and .PKD extensions modified to PK1 ... PK9, P10 ... P99. New .UHH files are automatically created regardless of this bit.
- **NoCusUHH.** Setting this bit disables custom batch entries in the .uhh file. .uhh files that support this can only be opened using Review 3.3.10 or later.
- **UHHPHase.** Setting this bit enables the addition of batch phase record to the .uhh file, when using the batch system, BAT_CTRL block. .uhh files that support this can only be opened using Review 3.10 or later.

Device. Specify the logging storage medium which must be set to 'H' (Internal Archive).

Group1 to Group16. Specify up to sixteen LGROUP blocks to be associated with the logging device specified in the *Device* field. Enter the block names in the relevant fields. (*Figure 94* shows the way the LGROUP and LOGDEV blocks are interlinked.)

DiskFull. Triggers the deletion of defined old file types but does not initiate an alarm. Full alarm is asserted when the device is 100% full.

LGROUP: LOG GROUP BLOCK

Block function

The LGROUP block collects data from up to sixteen DR_DGCHP and DR_ANCHP point blocks, extendible to a total of 80, and allows it to be archived onto a medium specified by the associated LOGDEV block. A total of up to eight DR_ALARM blocks, and up to 16 DR_REPRT blocks, can also be attached to a single LGROUP block including any LOGGRPEX block extensions.

Figure 94 shows how the LGROUP block mediates between the point data in the log group and the logging device.

NOTE The hierarchy shown in Figure 94 coexists with, but is not the same as, the Area/Group hierarchy used by the RECORDER block category for flash recording of data (see Figure 92).

Block parameters

Symbols used in Table 178 are explained in Table 1. Additional parameter information is given in the Block specification menu section following.

Parameter	Function	Units	Status
State	Specify when the group is to be logged	Menu	
UpdateA	Logging interval, in seconds. (Default = 0.0s)	Eng	
UpdateB	Alternative logging interval. (Default = 0.0s)	Eng	
UpdB_On	TRUE enables UpdateB (default = FALSE)	T/F	☐
LogNow	TRUE logs single sample, then resets	T/F	☐
Trigger	TRUE enables logging if State = TRIGGER	T/F	☐
Extend	Name of LOGGRPEX block adding logged points	Alphanumeric	
Point1-Point16	Specify attached point block names	Alphanumeric	
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
Config	TRUE=configuration error (e.g. invalid log group Id etc.)	T/F	
Combined	OR-ing of all Alarms bits	T/F	
NameType	Specify how file is named	Menu	
FileType	Specify file format generated by log group	Menu	
FileName	Specify base name of file to be generated	Alphanumeric	
ColTitle	Specify if ASCII column titles to appear	Menu	
DateFmt	Specify how ASCII date &/or time column appears	Menu	
ComRatio	Specify if packed data is compressed or normal	Menu	
DataRate	Logging rate (Kbytes/hour)	Eng	

Table 178 Block parameters

Block specification menu

The following is given in addition to Table 178.

State. (ON/OFF/TRIGGER) Determines when the log group is to be logged. ‘ON’ (the default) enables, and ‘OFF’ disables logging. ‘TRIGGER’ allows the *Trigger* digital input to enable/disable logging.

Extend. Name of a Log Group Extension (LOGGRPEX) block that extends the possible number of logged points by a further 16. You can add more logged points, up to a total of 80, by referring to another extension block from the within each LOGGRPEX block (via the block’s *Next* field).

NameType. (Text/Hourly/Daily/Sequence) Specifies how the file is named.

FileType. (ASCII/Packed/UHH) Specifies the type of file format generated by the log group.

FileName. String specifying the base name of the file to be generated by logging. Only the first two characters are applicable, irrespective of the *NameType* configuration.

ColTitle. (Standard/None) With ASCII column titles, *ColTitle* set to ‘Standard’ specifies that the titles are to be added to the log files (the default). ‘None’ means titles omitted.

DateFmt. (DateTime/Sprdsht/Integer/Duration/Days/D,H,M,S) With ASCII date and/or time columns, *DateFmt* specifies their presentation.

Table 179 lists examples of the options.

ComRatio. (Normal/High) ‘Compression ratio’. With *FileType* set to ‘Packed’, *ComRatio* set to ‘Normal’ specifies that data is uncompressed (the default), and *ComRatio* set to ‘High’ specifies that data is compressed.

Option	Example
DateTime	01/01/88 00:02:26
Sprdsht	32143.001690
Integer	980605123030 (=5/6/98 @ 12:30:30)
Duration	00:02:26
Days	08
D,H,M,S	08:00:02:26

Table 179 Examples of DateFmt options

LOGGRPEX: LOG GROUP EXTENSION BLOCK

Block function

The LOGGRPEX block is used to extend the number of points that can be logged by a LGROUP block by an additional 16 points. You can log further points, up to a total of 80, by 'daisy-chaining' more extension blocks via their *Next* fields.

Block parameters

Symbols used in *Table 180* are explained in *Table 1*.

Parameter	Function	Units	Status
Next	Name of further LOGGRPEX block adding logged points	Alphanumeric	
Point1-Point16	Specify attached point block names	Alphanumeric	
Alarms			  
Software	Block RAM data sumcheck error / network failure	T/F	
Config	TRUE=configuration error (e.g. invalid log group Id etc.)	T/F	
Combined	OR-ing of all Alarms bits	T/F	

Table 180 Block parameters

LPTDEV: PRINTER DEVICE BLOCK

Block function

The LPTDEV block defines access to a printer device, and specifies which print groups apply to that device. *Figure 95* shows how the ORGANISE blocks (shaded) interconnect the point data and printing device.

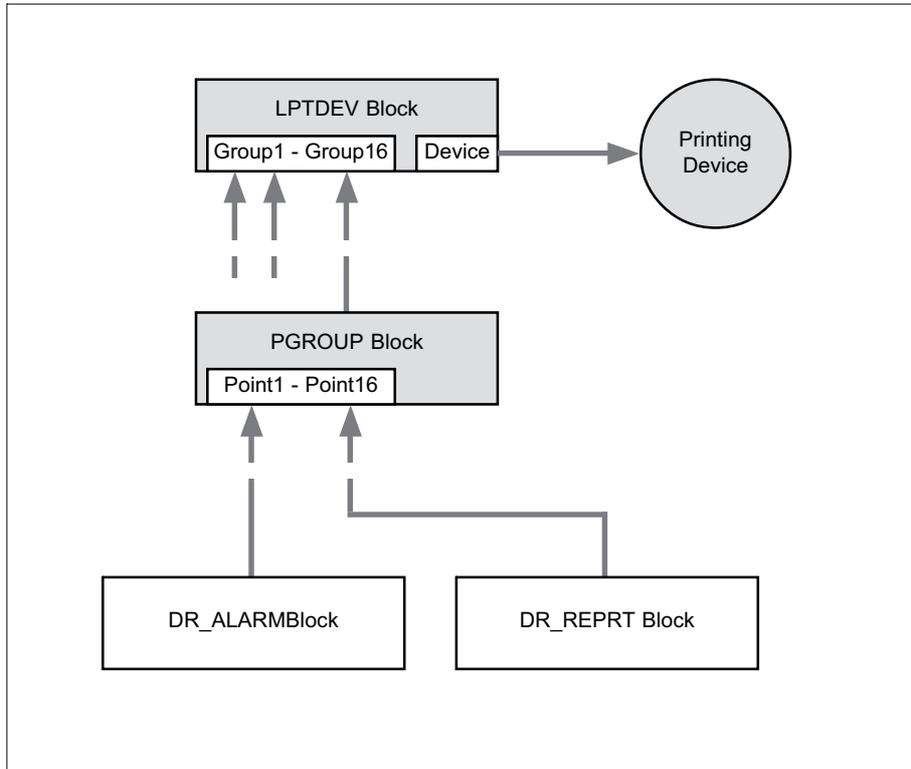


Figure 95 Organising DR_ALARM & DR_REPRT point blocks for printing

Block parameters

Symbols used in *Table 181* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
State	Specify when the device may be accessed	Menu	
Status	Report current status of printing	Menu	
Trigger	TRUE enables printing if State = TRIGGER	T/F	
Options	Specify printing options (currently unused)	ABCD hex	
Device	Specify printing device name (default = 'PRINTER1')	Alphanumeric	
AlmForm	Name of optional .UYT printer form file for alarm O/P	Alphanumeric	
Group1-Group16	Specify PGROUP (print group) block names	Alphanumeric	
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
Config	TRUE=config. error (e.g. invalid printer group or device name)	T/F	
Form	TRUE = errors in AlmForm (.UYT)	T/F	
PaperEnd	TRUE = printer has reported out of paper	T/F	
ErrorLPT	TRUE = printer error	T/F	
Combined	OR-ing of all Alarms bits	T/F	
Tx_Cnt	Number of bytes sent to printer since started	Integer	
FormErr	Form error code	Menu	
FormErLn	Line number on which FormErr error occurred	Integer	
FormErCh	Character position of error in line flagged by FormErLn	Integer	

Table 181 Block parameters

Block specification menu

The following is given in addition to *Table 181*.

State. (ON/OFF/TRIGGER) Determines when the device may be accessed by specifying when printing is permitted. ‘ON’ (the default) enables, and ‘OFF’ disables printing. ‘TRIGGER’ allows the *Trigger* digital input to enable/disable printing.

Status. (INACTIVE/ACTIVE/PRINTING/OFF_LINE) Reports current status of printing. *Table 182* explains the options.

Option...	...Means
INACTIVE	No printing permitted
ACTIVE	Printing permitted
PRINTING	Printing occurring
OFF_LINE	Printer offline

Table 182 Meanings of Status options

Device. (PRINTER1/PRINTER2) The device to be used for printing.

AlmForm. Name of an optional printer form file (extension .UYT) which may be used to customise the text layout of alarm output.

Group1 to Group16. Specify the up to sixteen print (PGROUP) blocks to be associated with the printing device specified in the *Device* field. Enter the block names in the relevant fields. (*Figure 95* shows the way the PGROUP and LPTDEV blocks are interlinked.)

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.
- **Config.** Asserted if a configuration error was detected. This can be caused if an incorrect Protocol is configured in the Serial category of the Instrument Properties dialog, or an invalid printer group or device name was selected.
- **Form.** Asserted if an error in the AlmForm, .uyt, was detected.
- **PaperEnd.** Asserted if the printer is reporting it is out of paper.
- **ErrorLPT.** Asserted if the printer is reporting an error.
- **Combined.** Asserted if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

FormErr. (OK /BAD_FILE /LINE_LEN /NEWLINE /MEMORY /SYNTAX /RANGE /NAME /DICTNRY /TYPE /ACTION /FORM /OTHER) Form error code. The options have the following meanings:

- **OK.** No error.
- **BAD_FILE.** File cannot be found, or cannot be read.
- **LINE_LEN.** A line in the file was too long (limit is 255 excluding CR and/or LF).
- **NEWLINE.** File does not end with a newline (possibly indicating corruption).
- **MEMORY.** Form is too big for the memory allocated to it.
- **SYNTAX.** There is a syntax error of some kind.
- **RANGE.** A numeric value is out of its permitted range.
- **NAME.** A named object cannot be found (possibly misspelled).
- **DICTNRY.** A dictionary entry cannot be found.
- **TYPE.** An object type is inappropriate in the context in which it is being used.
- **ACTION.** (*Not implemented.*)
- **FORM.** A construct was used which is invalid in a form file.
- **OTHER.** Other error.

PGROUP: PRINTER GROUP BLOCK

Block function

The PGROUP block collects data from up to sixteen DR_ALARM and DR_REPRT point blocks and allows it to be printed via a printing device specified by the associated LPTDEV block.

The block may be used to group together periodic logged values (analogous to trending) and also to specify reports that may be generated on the printer.

Figure 95 shows how the PGROUP block mediates between the point data in the print group and the printing device.

Block parameters

Symbols used in Table 183 are explained in Table 1. Additional parameter information is given in the Block specification menu section following.

Parameter	Function	Units	Status
State	Specify when the group is to be printed	Menu	
UpdateA*	Printing interval, in seconds. (Default = 0.0s)	Eng	
UpdateB*	Alternative printing interval. (Default = 0.0s)	Eng	
UpdB_On*	TRUE enables UpdateB (default = FALSE)	T/F	☐
PrintNow*	TRUE prints single sample, then resets	T/F	☐
Trigger	TRUE enables printing if State = TRIGGER	T/F	☐
Point1-Point16	Specify attached point block names	Alphanumeric	
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
Config	TRUE=configuration error (e.g. invalid print group Id etc.)	T/F	
Combined	OR-ing of all Alarms bits	T/F	
FrqTitle*	Title printing frequency (periodically printed values)	Menu	
ColTitle*	Select if ASCII column titles to appear	Menu	
ColWidth*	Column width in characters (5 - 16; default = 8)	Integer	
DateFmt*	Specify how ASCII date &/or time column appears	Menu	
DataRate	Printing rate (Kbytes/hour)	Eng	

*Not currently implemented

Table 183 Block parameters

Block specification menu

The following is given in addition to Table 183.

State. (ON/OFF/TRIGGER) Determines when the print group is to be printed. ‘ON’ (the default) enables, and ‘OFF’ disables printing. ‘TRIGGER’ allows the *Trigger* digital input to enable/disable printing.

Alarms. This information is given in addition to Table 183:

- **Config.** TRUE if there is a configuration error, e.g. invalid printer group Id, invalid point block reference, or invalid characters in file names.

FrqTitle. (Never/Start/Always) The frequency with which titles are printed for periodically printed values. (*Not currently implemented.*)

ColTitle. (Standard/None) With ASCII column titles, *ColTitle* set to ‘Standard’ specifies that the titles are to be added to the printouts (the default). ‘None’ means titles omitted. (*Not currently implemented.*)

DateFmt. (DateTime/Sprdsht/Integer/Duration/Days/D,H,M,S) With ASCII date and/or time columns, *DateFmt* specifies their presentation. (*Not currently implemented.*)

Table 184 lists examples of the options.

Option	Example
DateTime	01/01/88 00:02:26
Sprdsht	32143.001690
Integer	980605123030 (=5/6/98 @ 12:30:30)
Duration	00:02:26
Days	08
D,H,M,S	08:00:02:26

Table 184 Examples of DateFmt options

CHAPTER 16 PROGRAMMER FUNCTION BLOCKS

The PROGRAMMER category of Function Block Templates provide the control strategy with functions for controlling, monitoring and scheduling setpoint programs. There are two versions of the Program Editor as follows:

- the Programmer Editor, for use with the Eycon™ 10/20 Visual Supervisor and Tactician T2550 instruments. This version is considerably enhanced over the legacy programmer and allows for a devolved multi-node application and is configured using the LINtools ‘Programmer Wizard’ and ‘Programmer Editor’.

The *PROGCTRL: Programmer Control Block* allows control of the currently-loaded program, scheduling of the next program to run, and control over the programmer state machine.

The *PROGCHAN: Programmer Channel Block* allows configuration of the data and options of one channel being profiled by the Program.

The *SEGMENT: Programmer Segment Block* allows the display of up to 4 Segments for a single channel being profiled by the Program.

NOTE Alarms asserted in the PROGCTRL block, PROGCHAN block and any other process critical block can be automatically added to the Programmer Alarms displayed on the Eycon by using the AREA block and the GROUP Block in the Recorder category.

- the legacy Setpoint Programmer Editor, for use only at the supervisory level, i.e. the Eycon™ 10/20 Visual Supervisor and T800 Visual Supervisor. This version of Setpoint Programmer is configured using the ‘T800 Setpoint Programmer’ editor.

The *SPP_CTRL: Setpoint Programmer Control Block* allows monitoring of the currently-loaded program, scheduling of the next program to run, and control over the programmer state machine.

The *SPP_DIG: Setpoint Programmer Digital Block* provides a means of wiring out digital setpoints from the setpoint programmer.

The *SPP_RAMP: Setpoint Programmer Local Ramp Block* allows local ramping of setpoints in the Visual Supervisor.

The *SPP_EXT: Setpoint Programmer Extension Block* allows simple control of the program without the need for Sequences (SFCs).

SPP_CTRL: SETPOINT PROGRAMMER CONTROL BLOCK

Block function

The SPP_CTRL (setpoint programmer control) block can access the status and programming/scheduling facilities of the setpoint programmer, for use by User Screens, SFC, FBD, and across communications. This block must be run if all aspects of programmer functionality are required.

It provides the same facilities as the programmer panel agent, except that it cannot report full text descriptions (other than LIN database 8-character names), owing to the 8-character LIN Database STRING restriction.

Block parameters

Symbols used in *Table 185* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Id	ID no. of program (1 to 4)	Integer	
State	Current state of program	Enum	
Hold	Program in HOLD	T/F	
Manual	Programmer is in Manual mode	T/F	[1]
NxtRdy	Next requested program is ready	T/F	[2]
RqNxtPrg	Requested filename of next program to run	Alphanumeric	
RqStrtDt	Requested earliest date to start next program	Date	
RqStrtTm	Requested earliest time of day to start next program	Time	
RqNumIt	Requested number of iterations (0-999; 0=continuous)	Integer	
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
Holdback	Program now in holdback	T/F	
Combined	OR-ing of all Alarms bits	T/F	
CurrProg	Name of current .UYS file (setpoint program)	Alphanumeric	
EndDate	Date currently-executing program iteration will end	Date	
EndTime	Time of day currently-executing program iteration will end	Time	
NumIt	Original no. of iterations requested (0-999; 0=continuous)	Integer	
ItRemain	No. of program iterations left, incl. current (0-999; 0=cont.)	Integer	
CurrSeg	Currently-executing segment-no. (0=no current segment)	Integer	
SegTime	Time remaining in current segment	Time	
ProgDur	Expected total program iteration duration	Time	

[1] Read Only prior to Version 4.0. [2] Applicable to Version 4.0

Table 185 Block parameters

Block specification menu

The following is given in addition to *Table 185*.

Id. Identification number of program, for use with multiple programmers. Defaults to a value of '1'.

State. (RUN/HELDBACK/RESET/IDLE/COMPLETE) Current state of program.

Manual. TRUE means the program is in manual mode. In manual, scheduling and control cannot be done by the SPP_CTRL block.

NxtRdy. TRUE means the next requested program is ready. You should set *NxtRdy* to FALSE while modifying the '*Rq...*' fields, then reset it to TRUE when all the fields are correctly configured. These *NxtRdy* and '*Rq...*' settings would normally be done via a sequence (SFC), or a cold start definition.

RqStrtDt. The requested earliest date at which to start the next program, in the date format dd/mm/yy. Entering '??/??/??' in this field means 'today'.

NOTE Any entry in this field is overridden if '??:??:??' has been entered in the *RqStrtTm* field (see next).

RqStrtTm. The requested earliest time-of-day to start the next program, in the time format hh:mm:ss. Entering '??:??:??' in this field means 'start immediately, ignoring the entry in the *RqStrtDt* field' (see previous).

RqNumIt. The requested number of iterations (0 = 'continuous').

NOTE. The main use of the *NxtRdy*, *RqStrtDt*, *RqStrtTm*, and *RqNumIt* fields is to force a program to run or cold start.

EndDate. Date at which currently-executing program iteration will terminate. *EndDate* changes only if the program goes into HOLD or HOLDBACK.

EndTime. Time-of-day at which currently-executing program iteration will terminate. *EndTime* changes only if the program goes into HOLD or HOLDBACK.

ProgDur. Expected total program iteration duration. *ProgDur* does not change once the program has started.

SPP_DIG: SETPOINT PROGRAMMER DIGITAL BLOCK

Block function

This block groups together digital setpoints generated by the setpoint program running in the Visual Supervisor, and provides corresponding digital outputs that can be ‘wired’ to remote target instruments via devolved control module blocks. The digital outputs can also be used locally, e.g. by wiring them to DR_DGCHP blocks for flash memory recording within the Visual Supervisor.

Figure 96 shows an example of an SPP_DIG block linked to other blocks in a Visual Supervisor. The scheme provides control of a remote digital setpoint ‘Blower On’ across the comms (via the D25_DO4 block), and also local recording of the setpoint (via the DR_DGCHP block). Note that the DR_DGCHP block must form part of a ‘Group/Area’ hierarchy for recording to happen, this is not shown in the figure, see *Recorder Function Blocks*.)

NOTE There is no equivalent block for handling the analogue setpoints generated by the setpoint program. These are routed directly to their targets via D25_RAMP (devolved control module) blocks. See Chapter 8, *DCM Function Blocks*, for details of using this block.

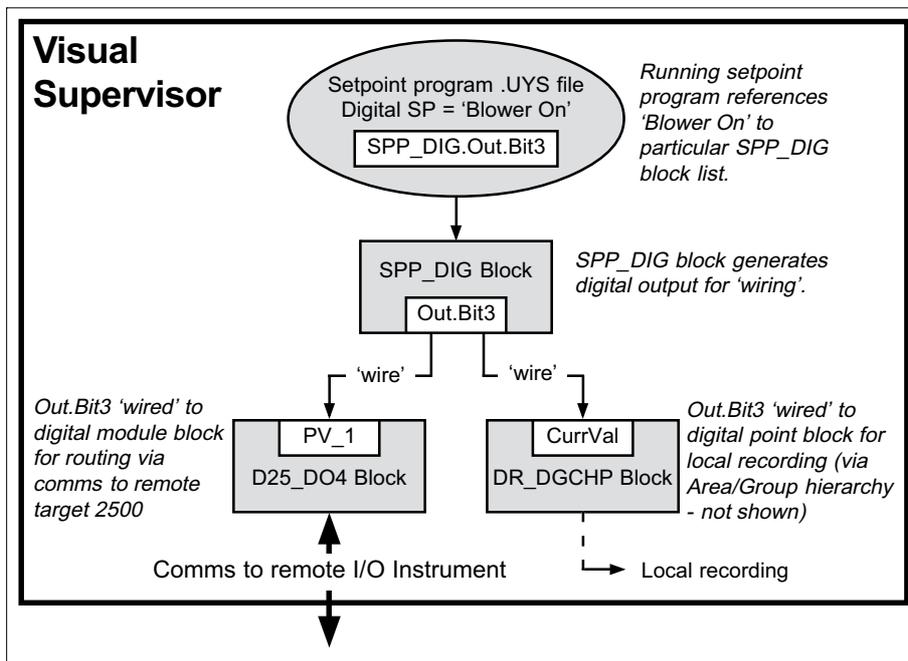


Figure 96 Using the SPP_DIG block to route digital setpoints - example

Block parameters

Symbols used in *Table 186* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Out	Values of digital setpoints	ABCD hex	<input type="checkbox"/> →
Bit0	TRUE = assigned digital setpoint 'ON'	T/F	D
Bit1		T/F	
Bit2		T/F	
Bit3		T/F	
Bit4		T/F	C
Bit5		T/F	
Bit6		T/F	
Bit7		T/F	
Bit8		T/F	B
Bit9		T/F	
Bit10		T/F	
Bit11		T/F	
Bit12		T/F	A
Bit13		T/F	
Bit14		T/F	
Bit15	T/F		
Alarms			<input type="checkbox"/> → <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
Software	Block RAM data sumcheck error / network failure	T/F	
Combined	OR-ing of all Alarms bits	T/F	

Table 186 Block parameters

Block specification menu

The following is given in addition to *Table 186*.

Out. The bits of this field are associated with digital setpoints generated by the setpoint program. This can be done for each setpoint by entering the relevant SPP_DIG block *Out* bit in the *Hardware reference* field of the Setpoint program editor's *Setpoint Properties* dialog, using the format **Blockname.Out.Bit*n***. See the *Setpoint Program Editor Handbook* (Part No. HA261134U005), for details.

SPP_RAMP: SETPOINT PROGRAMMER LOCAL RAMP BLOCK

Block function

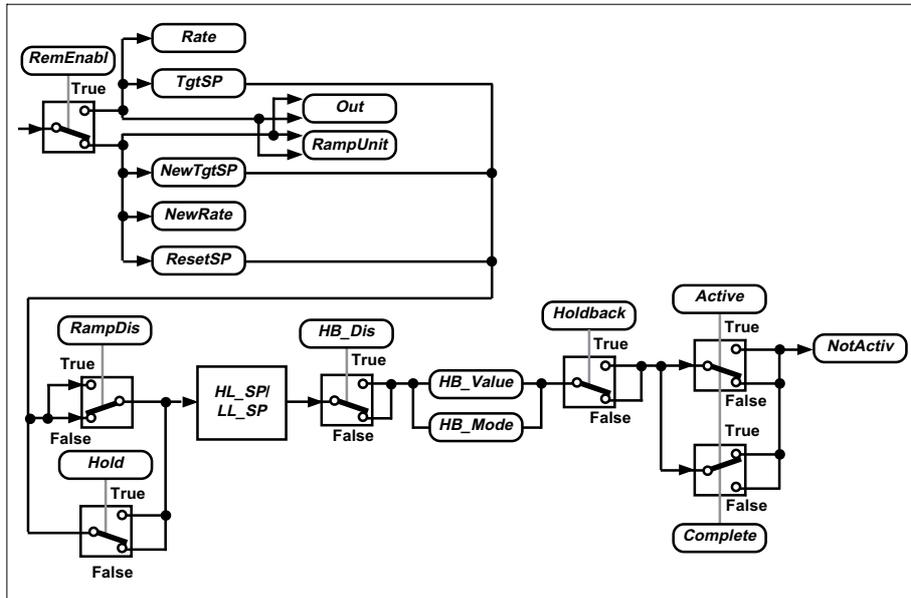


Figure 97 Block function

This block mirrors the functionality provided in the I/O system units by allowing a local setpoint to be ramped locally (within the Visual Supervisor), as an alternative to executing the ramp in the remote I/O instrument.

Block parameters

Symbols used in Table 187 are explained in Table 1. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
ResetSP*	Setpoint used in 'reset', i.e. when RemEnabl FALSE	Eng	<input type="checkbox"/>
HL_SP*	Upper limit of SP (in all forms)	Eng	
LL_SP*	Lower limit of SP (in all forms)	Eng	
Track*	Track PV - Not Implemented	T/F	<input type="checkbox"/>
RampDis*	Ramp disable	T/F	<input type="checkbox"/>
PV	Process variable	Eng	<input type="checkbox"/>
TgtSP	Target setpoint	Eng	<input type="checkbox"/>
Rate	Requested rate of change	Eng	
RampUnit	Units of rate (SEC/MIN/HOUR)	Enum	
Out	Current setpoint value	Eng	<input type="checkbox"/>
NewTgtSP	Next target setpoint	Eng	<input type="checkbox"/>
NewRate	Rate of next segment	Eng	<input type="checkbox"/>
Sync	Triggers load of NewTgtSP & NewRate values	T/F	
SyncPV	Servo to PV at start of ramp	T/F	<input type="checkbox"/>
Alarms			<input type="checkbox"/>
Software	Block RAM data sumcheck error / network failure	T/F	
Holdback	Ramp in holdback	T/F	
Underflo		T/F	
Combined	OR-ing of all Alarms bits	T/F	
Complete	Ramp complete	T/F	<input type="checkbox"/>
Active	Ramping is in progress	T/F	<input type="checkbox"/>
NotActiv	Time ramp has not been actively ramping	Time	<input type="checkbox"/>
Holdback	Ramp is currently held back	T/F	<input type="checkbox"/>
Hold	Ramp in HOLD	T/F	<input type="checkbox"/>
HB_Mode	Holdback type (NONE/LOW/HIGH/DEV)	Enum	
HB_Value	Holdback value	Eng	

Continued...

Parameter	Function	Units	Status
<i>Continued...</i>			
HB_Dis	Holdback disabled	T/F	
MinHback[N]	(Reserved function)	Eng	
OorHback[N]	Ramp in holdback & 'rate-failed' (Not implemented)	T/F	
RemEnabl	Set by SPP at program start, reset at termination	T/F	
StLocRmt	Setting of Local/Remote at startup	Enum	
StWspCh	Setting of WSP at startup	Enum	
StHold	Setting of Hold at startup	Enum	
SP_Servo*	TRUE = SP servo'd. (User resettable)	T/F	

[N] No communications with this field are implemented currently.

Table 187 Block parameters

Block specification menu

The following is given in addition to *Table 187*.

Dbase, Block, Type. See *Appendix D page 544* for details of these 'header' fields.

ResetSP. This shows the Setpoint value used in 'reset', i.e. when *RemEnabl* is FALSE.

HL_SP, LL_SP. High and low Setpoint limit values operational in all forms but clips the configured *TgtSP* value.

Track. (TRUE/FALSE). For future use. Used to control the Setpoint ramp. Set TRUE to allow the Setpoint ramp to track *PV*.

RampDis. (TRUE/FALSE). Used to control the Setpoint ramp. Set TRUE to temporarily disable the Setpoint ramp.

PV. This shows the current value of the control loop process variable.

TgtSP. Used to define the Setpoint value required by the Program.

Rate. Used to define the requested rate of change when ramping to the value specified in *TgtSP*.

RampUnit. (SEC/MIN/HOUR). Used to define the units of rate when ramping to the value specified in *TgtSP* if *RemEnabl* is TRUE or *NewTgtSP* if *RemEnabl* is FALSE.

Out. Shows the current Setpoint value of the process.

NewTgtSP. Used to define next Target Setpoint value when an External master is controlling the ramp. This will be limited by the value configured in *HL_SP* and *LL_SP*. If *RemEnabl* is FALSE this value is ignored and *ResetSP* is used.

NewRate. Used to define the rate of next Segment for when an External master is controlling the ramp.

Sync. (TRUE/FALSE). Used to control the values used when an External master is controlling the ramp. TRUE loads the *NewTgtSP* and *NewRate* values to *TgtSP* and *Rate* respectively.

SyncPV. (TRUE/FALSE). Used to control the how the Program will start. TRUE indicates the Program will advance to the start of the ramp derived from *PV*.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block's RAM data.
- **Holdback.** Asserted, if the *PV* has exceeded the rules specified by *HB_Mode*.
- **Underflo.** Asserted, if *PV* did not obtain the minimum value in measuring range.
- **Combined.** Asserted if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

Complete. (TRUE/FALSE). Used to indicate the Setpoint has completed ramping to the value defined in *TgtSP* if *RemEnabl* is TRUE or *NewTgtSP* if *RemEnabl* is FALSE.

Active. (TRUE/FALSE). Used to indicate the Setpoint is currently ramping to the value defined in *TgtSP* if *RemEnabl* is TRUE or *NewTgtSP* if *RemEnabl* is FALSE.

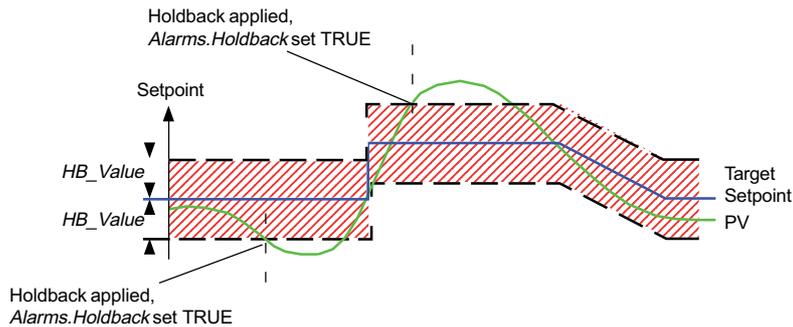
NotActiv. Shows the amount of time the Setpoint ramp has not been active.

Holdback. (TRUE/FALSE). Used to control and indicate the Setpoint ramp is currently held back because *PV* is unable to keep up with the changing Setpoint within the value defined in *HB_Value*.

Hold. (TRUE/FALSE). Used to control the Setpoint ramp. Set TRUE to HOLD the current Setpoint ramp. Set FALSE to continue with normal operation.

HB_Mode. (NONE/LOW/HIGH/DEV). Shows the cause of the Holdback applied to the Setpoint ramp. **NONE** indicates the Setpoint ramp is operating normally. **LOW** indicates the Setpoint ramp is held back, *Holdback* shows TRUE, when the *PV* is below the Setpoint by the Holdback value. **HIGH** indicates the Setpoint ramp is held back when the *PV* is above the Setpoint by the Holdback value. **DEV** indicates the Setpoint ramp is held back when the *PV* is below or above the Setpoint by the Holdback value.

HB_Value. The configured deviation value between *TgtSP* and *PV* used to stop the Setpoint ramp when the *PV* is unable to keep up with the changing Setpoint. This value is only applied if *HB_Dis* is FALSE.



HB_Dis. (TRUE/FALSE). Used to control the Holdback strategy of the Setpoint ramp. TRUE enables the Holdback strategy using the value defined in *HB_Value*.

MinHback. For future use.

OorHback. For future use. Shows that Setpoint ramp is in Holdback and ‘rate-failed’.

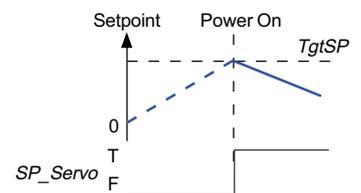
RemEnabl. (TRUE/FALSE). Used to control the use of the ‘Remote’ Setpoint. TRUE when the Local Setpoint (*TgtSP*) is being used by the PID, and FALSE when the ‘Remote’ Setpoint (*NewTgtSP*) is being used by the PID.

StLocRmt. (NoChange/StLocal/StRemote). Defines the Setpoint Mode when the controller powers up. **NoChange** indicates the Setpoint selection remains as it was when the controller was powered down. **StLocal** indicates the controller uses the local Setpoint when it powers up. **StRemote** indicates the controller uses the Remote Setpoint when it powers up.

StWspCh. (NoChange/GotoPV/GotoTSP). Defines the Setpoint strategy when the controller powers up. **NoChange** indicates the Setpoint remains the same as when the controller was last used. **GotoPV** indicates the Setpoint takes the same value as the process variable, *PV*. **GotoTSP** indicates the Setpoint takes the same value as the Target Setpoint, *TgtSP*.

StHold. (NoChange/Hold/NoHold). Defines the Hold strategy when the controller powers up. **NoChange** indicates the Hold remains the same as when the controller was last used. **Hold** indicates the controller powers up in Hold mode. **NoHold** indicates the controller powers up in normal ramp mode.

SP_Servo. (TRUE/FALSE). Used to control the start point of the Setpoint ramp. Set TRUE to start the Setpoint ramp from the current value derived from *TgtSP*.



SPP_EXT: SETPOINT PROGRAMMER EXTENSION BLOCK

Block function

The SPP_EXT block is an optional block that may be used to provide additional control over a running program, or to perform actions at given points in the programmer execution. The same functionality could be achieved using a Sequence (SFC), but for applications that would otherwise need no sequence this block can be a simpler solution.

The block provides as inputs a number of programmer conditions: the current running program (*CurrProg*), a range of segment numbers (*FirstSeg* to *LastSeg*), a set of states (*Inhibit*) that can be individually selected to inhibit the requested programmer action, and an external stimulus (*Trigger*). If all the conditions are satisfied and the *Trigger* input is asserted, then the *Triggerd* output sets and the requested programmer action, *Action* or *GotoSeg*, is performed.

Block parameters

Symbols used in *Table 188* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Id	ID number of program	Integer	 
CurrProg	Name of current .SPP file to act on	Alphanumeric	
FirstSeg	First segment number to match	Integer	
LastSeg	Last segment number to match (0 = End segment)	Integer	
Inhibit	States inhibiting programmer action	(AB)CD hex	
RUNNING	TRUE inhibits action when this state active	T/F	D
HOLD		T/F	
HELD		T/F	
BACK COMPLETE		T/F	
ERROR		T/F	C
Spare		T/F	
Spare		T/F	
Spare		T/F	
Trigger	TRUE = external condition satisfied	T/F	 
Alarms			  
Software	Block RAM data sumcheck error / network failure	T/F	
Config	Id invalid	T/F	
Segment	GOTO segment illegal AND Triggerd = TRUE	T/F	
Combined	OR-ing of all Alarms bits	T/F	
Triggerd	TRUE = action conditions satisfied	T/F	
Action	Program action done when Triggerd = TRUE	Enum	
GoToSeg	Segment number to go to if Action = GOTO	Integer	
NumIt	(Not used)	Integer	

Table 188 Block parameters

Block specification menu

The following is given in addition to *Table 188*.

Id. Identification number of program, for use with multiple programmers. Defaults to a value of '1'. *Id* must correspond to the *Id* value in the appropriate SPP_CTRL block.

CurrProg. Name of current .spp file (without the extension) on which the action is to be performed. Leaving this field blank (null string) means 'any program'.

FirstSeg, LastSeg. These fields specify the block of consecutive program segment numbers on which the action is to be performed. Setting respective values of:

- 0, 0 means all segments
- *n*, *n* means only segment *n*
- *m*, *n* means segments *m* to *n* inclusive
- 0, *n* means all segments up to and including *n*
- *n*, 0 means all segments from *n* to the end of the program.

Alarms. The *Segment* alarm trips if the segment number specified in *GoToSeg* is illegal when the other input conditions are satisfied (i.e. when *Triggerd* is TRUE).

Triggerd. This bit becomes TRUE only when all programmer conditions are satisfied, whereupon the specified action is performed. *Triggerd* is asserted only if:

- The current program name matches that specified in *CurrProg*
- The segment currently running is included in the block specified by *FirstSeg* and *LastSeg*.
- States specified as TRUE in the *Inhibit* parameter are not currently active
- The *Trigger* input is TRUE.

Action. (NONE/HOLD/ABORT/SKIP/GOTO) Action to be performed on the program when the conditions are satisfied, i.e. when *Triggerd* = TRUE. With HOLD selected, the program running is resumed when *Triggerd* resets to FALSE. If GOTO is defined, the program proceeds to the segment number specified by *GoToSeg* after the currently executing step has completed.

PROGCTRL: PROGRAMMER CONTROL BLOCK

Block function

This block is used to control the overall execution of a single Program configured using the Programmer Editor, i.e. one PROGCTRL block per Programmer. It will allow the user to load and save a Program file, manipulate segments of a Program, control the operation of the Program and indicate the current status of the Program.

It is one block type in a suite of blocks that includes the PROGCHAN and SEGMENT blocks, held in a **PROG_WIZ** compound when automatically created using the Programmer Wizard. Together these blocks provide access to all parameters in the Setpoint Program used in the Program file, created via the Programmer Editor.

IMPORTANT Always use the Programmer Wizard to configure the PROG_WIZ compound. This automatically combines the required blocks, and generates the complete Program Template file that must be downloaded to the instrument.

NOTE This block functions using a number of pages. Fields can be located using <Page>.<Field>.<Subfield> convention.

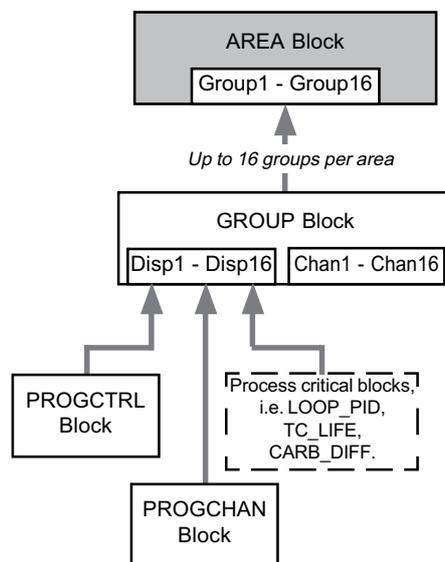
Alarm indication

Alarms asserted in the Programmer block can be automatically added to the list of Alarms shown on the Monitor display of an Eycon by using the AREA block and the GROUP block in the Recorder category, as shown.

NOTE If this AREA block is not configured, the Eycon Alarms button on the Monitor display will only show PROGCTRL block alarms and events.

When *Disp1 - Disp16* fields in a GROUP block are populated with the PROGCTRL block an asserted alarm will appear on the Monitor display of the Eycon when the Alarms button is pressed.

NOTE This also applies to the PROGCHAN block and any other process critical block.



Parameter	Function	Units	Status
<i>Continued...</i>			
PwrFail ^[W]	Hot start power fail recovery strategy	Enum	
TestT1 ^[W]	Power fail recovery control time	Time	
TestT2 ^[W]			
StartSeg	Starting segment of first cycle	Integer	
StartOff	Offset time for starting segment of first cycle	Time	
Cycles	Number of times the Program repeats	Integer	
RateUnit	Units of rate (Single channel Programs only)	Enum	
Alarms			
See File Page			
State	Current state of program	Enum	
StateFlg	Bitfields showing current state of Program	ABCD hex	
Unloaded	TRUE, Program is not loaded or failed to load	T/F	D
Loading	TRUE, during transition from idle to loading	T/F	
Idle	TRUE, Program is loaded but not started	T/F	
Starting	TRUE, during transition from idle to running	T/F	
Running	TRUE, Program is running	T/F	C
Complete	TRUE, Program is complete	T/F	
Holding	TRUE, during transition from running to held	T/F	
Held	TRUE, Program is held	T/F	
Restart	TRUE, during transition from held to running	T/F	B
Aborting	TRUE, during transition to aborted	T/F	
Aborted	TRUE, Program is aborted	T/F	
Reseting	TRUE, during transition to idle	T/F	
Schedule	TRUE, Program is scheduled to start	T/F	A
Saving	TRUE, Program is being saved	T/F	
Inhibit	TRUE, Program execution/running is prevented	T/F	
CycleNo	Current iteration of Program	Integer	
ProgLeft	Time remaining in current Program	Time	
SegLeft	Time remaining in current Segment	Time	
Config Page			
ChanAsyn	Synchronous/asynchronous channel configuration	AB(CD) hex	
Chan_01	For Future use	T/F	D
Chan_02		T/F	
Chan_03		T/F	
Chan_04		T/F	
Chan_05	TRUE, if channel is running asynchronously	T/F	C
Chan_06		T/F	
Chan_07		T/F	
Chan_08		T/F	
Chan_01 to Chan_08	PROGCHAN blocks associated with this Program	Block	
Alarms			
See File Page			
AlgBlk	Associated block providing Program algorithm	Block	

[W] Configured by the Programmer Wizard

Table 189 Block parameters

Block specification menu

The following is given in addition to *Table 189*.

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

NOTE These fields automatically appear in each of the pages in this block.

File Page

This page is used to define the files used by this Program.

TmplFile. This value is automatically configured using Programmer Wizard. This is the File name used to identify which Program Template file will be used for this Programmer.

ProgFile. Used to configure the File name identifying which Program file will be used. The Program file is created using the Programmer Editor.

AlgFile. File name identifying the Algorithm file used in the Program.

Filepath. Specifies the location of the Program Template file and Program file. See “Filepath parameter” on page 547. for details.

Important Both Program Template file and Program file must be at the location defined in Filepath.

Load. (TRUE/FALSE). This is used to load the selected Program file. If *Config.AlgBlk* is configured *Monitor.AlgFile* is populated with the filename specified by the file derived from *Monitor.ProgFile*. If *Config.AlgBlk* is not configured *FileErr.AlgBlk* is set.

Save. (TRUE/FALSE). This is used to save the edited Program file.

SchFile. This is the file name of the Program scheduled to start at the configured date and time, see *File.SchDate*, and *File.SchTime*.

SchDate. Shows the date when the configured Program file is next scheduled to run.

SchTime. Shows the time on the configured date, *File.SchDate*, when the configured Program file is next scheduled to run.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data or network failure.
- **BadFile.** Asserted if a file failed to load because the file being loaded contains errors.
- **Oob.** Asserted if the *PV* value from one or more channels configured in the Program has exceeded a defined boundary during execution. This remains asserted until the Program is reset, see *PROGCHAN.Monitor.HiOob* and *PROGCHAN.Monitor.LoOob*.
- **Config.** Asserted if the Program configuration is invalid. This will not allow the Program to be executed or edited. This is caused if the Program Template file is not available when the database is started, or the configuration of the blocks in the **PROG_WIZ** compound is invalid.
- **BadStart.** Asserted on a user start if the offset into the selected segment is beyond the end of the segment. Cleared on a START or LOAD.
- **Combined.** Asserted if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

FileErr. (NONE/Format/TooBig/GoBack/AlgBlk/TmplFile/NoFile/ReadOnly/Other). This shows the cause of a failure to load the file.

- **Format** indicates a mismatch between the number of channels in Program and the number of channels configured for the PROGCTRL block.
- **TooBig** indicates the number of segments, user values, digital events in the Program exceeds the number supported in the database.
- **GoBack** indicates the configured Go back sequence is invalid. This may be caused if the specified Go back loop is nested or overlaps another Go back loop.
- **AlgBlk** indicates the file defined in *AlgFile* failed to load. This is caused if a suitable block type is not specified in *Config.AlgBlk*.
- **TmplFile** indicates the file defined in *TmplFile* failed to load. This may be caused if the defined Program Template file contains errors.
- **NoFile** indicates the file defined in *ProgFile* failed to load. This is caused if the defined Program file does not exist.
- **ReadOnly** indicates the file defined in *ProgFile* was not saved because the defined Program file was previously saved as read-only, *FileAttr.ReadOnly* is TRUE.
- **Other** indicates the file failed to load because of another form of error.

NOTE Information may be recorded in the instruments EVENTLOG.udz.

FileAttr. The bitfield shows the File attributes.

- **ReadOnly.** TRUE, if the loaded Program file is write protected.

ProgName. This is the user defined text string that identifies the Program when displayed via the visualisation software.

Edited. (TRUE/FALSE). When this shows TRUE, data in the loaded Program file has been changed, but is currently unsaved.

Important In Auditor systems the Program may not be run or restarted while this remains TRUE.

FileVer. This shows the version of the loaded Program file. The version number increments each time the loaded Program file is saved, i.e. *File.Save* set TRUE.

SaveDate, SaveTime. These fields show the date and time when the Program file was last saved, and are automatically updated when *File.Save* is set TRUE.

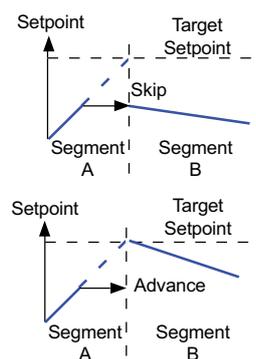
Monitor Page

This page provides parameters used to monitor the operation of the loaded Program.

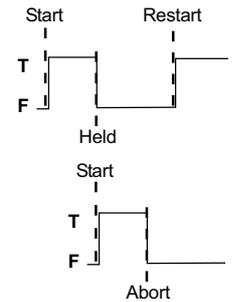
Mode. (AUTO/MAN). The current operating mode of the Program. It can only be changed when the Program is Idle or Running (*PROGCTRL.Monitor.State* shows Idle or Running). AUTO is automatically selected whenever the Program is started. In AUTO, outputs from the PROGCHAN block are controlled by the Program. In MAN, the Program can be controlled for commissioning purposes. This allows the user to step through the Program manually or define a starting segment to check events configured in each segment of the Program until AUTO is selected. This causes the Program to run in AUTO from the current position. During MAN mode *Monitor.StartSeg* and *Monitor.StartOff* are ignored, and all outputs of PROGCHAN block may be overwritten (*Monitor.SegNo*, *Monitor.SP*, *Monitor.UsrVal*, *Monitor.Event* etc.)

Command. Bitfields providing control for the currently active Program.

- **Start.** Used to start the Program. TRUE applies the Start command to the Program defined in *File.ProgFile*.
- **Hold.** Used to hold the Program. TRUE applies the Hold command to the currently active Program. Use the *Restart* or *StrtHold* commands to resume from a Hold state.
- **Restart.** Used to restart the Program. TRUE applies the Restart command to the currently active Program. This command also restarts a Program that has been paused using the *Hold* command.
- **Abort.** Used to abort the Program. TRUE applies the Abort command to the currently active Program.
- **Reset.** Used to reset the Program. TRUE applies the Reset command to the Program, causing it to return to the Idle state.
- **Skip.** Used to skip the current segment. TRUE applies the Skip command to the current segment. This will start the next segment from the current Setpoint, ignoring Wait Conditions.
- **Advance.** Used to advance to the next segment. TRUE shows the advance command has been applied to the current segment. This will set the Program Setpoint equal to the Target Setpoint and advance to the next segment, ignoring any existing Wait Conditions.
- **ReDo.** Used to repeat to the current segment. TRUE shows the redo command has been applied to the current Dwell or Step segment. This can only be applied if the Program is Held.
- **Jog.** Used to manually increase or decrease the duration of a current dwell segment. TRUE shows the current segments will jog for the period defined in *JogTime* of the associated PROGCHAN block.
- **Schedule.** Used to schedule the Program defined in *File.SchFile*. TRUE shows that a Program is already scheduled, the existing schedule will be replaced with this one. If *File.SchFile* is empty, this will cancel an existing schedule.



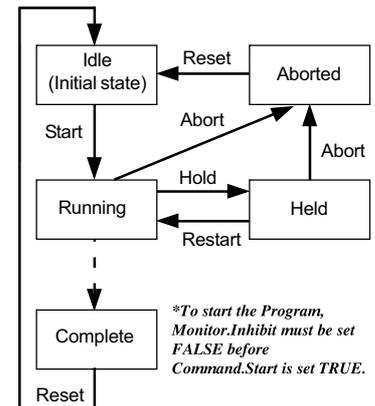
- **StrtHold.** Used to start or restart and hold the Program. Set TRUE to start the Program on the rising edge, and then set to FALSE to hold the Program on the falling edge. If the Program is currently Held, it will be restarted on the rising edge.



- **StrtAbtrt.** Used to start and terminate. Set TRUE to start the Program on the rising edge, and then set to FALSE to abort the Program on the falling edge.

CmndHshk. Bitfields allowing the transition between commands controlling the current Program. Each bitfield is set TRUE by default to allow the Program to run automatically without any intervention from the user.

- **Started.** TRUE allows the Program to complete the transition from Idle to Running, initiated via *Monitor.Command.Start*. FALSE prevents the Program from completing this transition allowing the user to manually enable the continuation of the Program.
- **Held.** TRUE allows the Program to complete the transition from Running to Held, initiated via *Monitor.Command.Restart*. FALSE prevents the Program from completing this transition allowing the user to manually enable the continuation of the Program.
- **Restartd.** TRUE allows the Program to complete the transition from Held to Running, initiated via *Monitor.Command.Restart*. FALSE, prevents the Program from completing this transition allowing the user to manually enable the continuation of the Program.
- **Aborted.** TRUE allows the Program to complete the transition from Running or Held to Aborted, initiated via *Monitor.Command.Abort*. FALSE prevents the Program from completing this transition allowing the user to manually enable the continuation of the Program.
- **Idle.** TRUE allows the Program to complete the transition from Complete or Aborted to Idle, initiated via *Monitor.Command.Reset*. FALSE prevents the Program from completing this transition allowing the user to manually enable the continuation of the Program. While Idle is TRUE, Aborted is a transitory state only.



CMND

Inhibit. Set TRUE to prevent the next Program from starting.

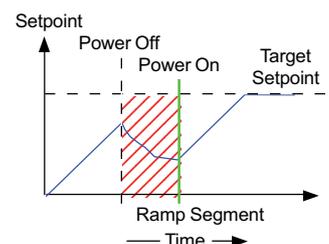
Options. Bitfield for optional configuration.

- **FastRun.** TRUE, causes a Setpoint Program to be illustrated over an accelerated period of 10 x normal speed.

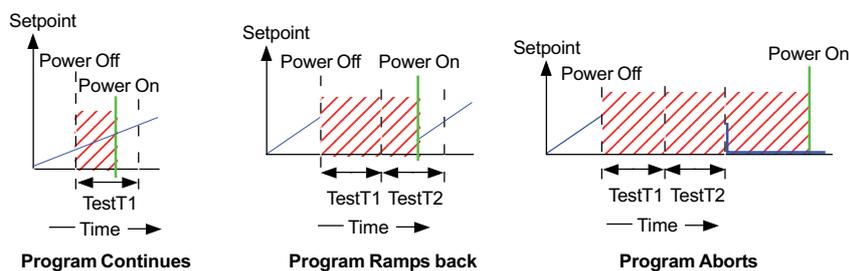
NOTE Generally of use for demonstration purposes.

PwrFail. (RampBack/Abort/Continue/Hold/TestTime). This value should be configured and edited using the Programmer Wizard. Shows the action that will be taken if a power failure occurs and the instrument hot-starts.

- **RampBack.** Indicates that when power is restored the PV will ramp to the Target Setpoint for the Program. If the Segment is configured as RampRate or RampTime, the PV will Ramp to the Target Setpoint, if configured as Step the PV will Step to the Target Setpoint, and if configured as Dwell, the Dwell time will not resume until the Target Setpoint is obtained at the last rate used in the Program.



- **Abort.** Indicates that when power is restored the Program will be aborted. The Program will abort and enter the Idle state if *Monitor.CmndHshk.Idle* is TRUE.
- **Continue.** Indicates that when power is restored the Program continues from where it was interrupted when power was lost. All parameters, such as the Setpoint and time remaining in the active segment will be restored to their power-down values.
 NOTE Recommended for applications that need to bring the measured PV to the Setpoint as soon as possible.
- **Hold.** Indicates that when power is restored the program will hold and enter the Held state, if *Monitor.CmndHshk.Held* TRUE.
- **TestTime.** Indicates that if power is restored before the period configured in *Monitor.TestT1* the Program will continue. However, if power is restored after the period configured in *Monitor.TestT1* but before *Monitor.TestT2* the Program will RampBack using the rules specified, see *PwrFail.RampBack*. If power is not restored until after the period configured in *Monitor.TestT2* the Program will abort and enter the Idle state if *Monitor.CmndHshk.Idle* is TRUE.



TestT1. This value should be configured and edited using Programmer Wizard. Used in conjunction with *Monitor.PwrFail* and *Monitor.TestT2*. This defines the time limit, 00:00:01 to 23:59:59, used to allow the Program to apply the Continue strategy after a power failure. Using 00:00:00 will not allow the Program to apply the Continue strategy, see *Monitor.TestT2*.

TestT2. This value should be configured and edited using Programmer Wizard. Used in conjunction with *Monitor.PwrFail* and *Monitor.TestT1*. It defines the time limit, 00:00:01 to 23:59:59, used to allow the Program to apply the RampBack strategy after a power failure. Using 00:00:00 will not allow the Program to apply the RampBack strategy causing the Program to abort and enter the Idle state if *Monitor.CmndHshk.Idle* TRUE.

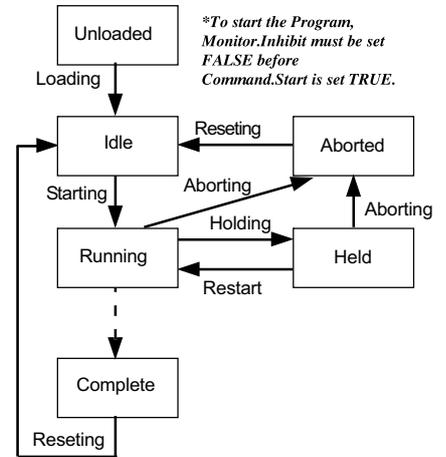
StartSeg. Used to define the first segment of a Program using a value between 0 and one less than the value shown in *PROGCHAN.NumSegs*. This value can only be set during an Idle state when *Monitor.Mode* shows AUTO and is automatically set to 0 (zero - disabled) when *Monitor.Mode* is set to MAN.

StartOff. Used to apply an offset start time period (23:59:59 max) to the first segment of a Program. It is automatically disabled (00:00:00) when *Monitor.Mode* is set to MAN.

Cycles. Used to define the number (1 - 999) of times the Program will run. Using 0 will cause the Program to repeat continuously. If multiple channels are running in a Program, each channel will complete before the next cycle commences. This implies a synchronisation point at the end of each cycle.

RateUnit. (Secs/Mins/Hours/Days). Used to define the time unit applied, by default, to all RampRate segment types in the Program.

State. (Unloaded/Loading/Idle/Starting/Running/Complete/Holding/Held/Restart/Aborting/Aborted/Resetting/Inhibit). Shows an enumerated description of the current state of the Program, as shown. **Unloaded** indicates a Program file has not yet been loaded. **Loading** indicates the Program file is in the process of being loaded. **Idle** indicates that the Program file has been loaded, but is not yet running. **Starting** indicates the Program has loaded and has been requested to run but is not yet running. **Running** indicates the Program is in operation. **Complete** indicates the Program has finished and the segment is held in an indefinite dwell at the setpoint value shown in *PROGCHAN.Monitor.Curr.SP*. **Holding** indicates that a command to hold the Setpoint has been requested, but it is not yet in a Held state. **Held** indicates that the Setpoint and remaining time are frozen at the current value. **Restart** indicates that a command to restart the currently held Program has been requested, but it has not yet returned to a Running state. **Aborting** indicates that a command to terminate the Program has been requested, but it has not yet in an Aborted state. **Aborted** indicates that the Program has terminated. **Resetting** indicates that a command to return to the start of the Program has been requested. **Inhibit** indicates that a loaded Program has been prevented from starting via *Monitor.Inhibit*.



StateFlg. Bitfields showing a digital representation of the current state of the Program, see *Monitor.State*.

NOTE All corresponding bitfield details are described in *Monitor.State*.

- **Schedule.** Set TRUE when a Program is scheduled.
- **Saving.** Shows TRUE when the currently loaded Program is being saved. When the save is complete, *File.FileVer* is increased by 1.
- CycleNo.** Shows the cycle number currently in operation.
- ProgLeft, SegLeft.** Shows the Program time remaining in this cycle, and the Segment time remaining.

NOTE The time displayed may be incorrect if the duration of the Program or Segment exceeds 36 hours, but this does not affect the execution of the Program.

Config Page

This page provides parameters used to show the blocks used in the Program, either associated with the Channel Profile or to populate files.

ChanAsyn. For Future Use - Bitfield indicating asynchronous operation of each channel.

- **Chan_01 to Chan_08.** For Future Use - Set TRUE to configure corresponding channel operation.

Chan_01 to Chan_08. These values are configured and edited using Programmer Wizard. Each field shows the PROGCHAN block associated with each channel in the Program.

AlgBlk. Shows the block name used to load the Algorithm file required by the Program. Select the block name of either a, SFC_CON, RECORD, RCP_SET, or DR_REPRT. These blocks hold the data used to populate the required algorithm.

Parameter	Function	Units	Status
<i>Continued...</i>			
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
HoldBack	Channel in Holdback	T/F	
Hi	Channel PV exceeds high limit	T/F	
Lo	Channel PV exceeds low limit	T/F	
Oob	Channel PV out of bounds during execution	T/F	
Waiting	Current segment held by external conditions	T/F	
Combined	OR-ing of all Alarms bits	T/F	
Cond	Segment condition	ABCD hex	
Exit1 to Exit8	Exit condition exists if TRUE	T/F	
Wait1 to Wait8			
Wait1 to Wait8	Wait condition exists if TRUE	T/F	
InvCond	Inverts the Segment condition state	ABCD hex	
Exit1 to Exit8	Inverted Exit condition state	T/F	
Wait1 to Wait8			
Wait1 to Wait8	Inverted Wait condition state	T/F	
CurrCond	Current condition logic	ABCD hex	
Exit1 to Exit8	Exiting current segment logic	T/F	
Wait1 to Wait8			
Wait1 to Wait8	Hold current segment logic	T/F	
GoSeg	First segment in GoBack sequence (0 = no GoBack)	Integer	
GoMax	Number of times GoBack will be executed	Integer	
GoNum	GoBack count	Integer	
SyncTo	For future use - Segment used to synchronise with	Integer	
StateFlg	Condition indicators	ABCD	
Wait1 to Wait8	Indicates Wait condition of channel	T/F	
ChActive			
End	Channel started, Program operating normally	T/F	
End	Segment complete	T/F	
Config Page			
Options	Bitfield for optional Program configuration	ABC(D) hex	
DisChan ^[W]	Disable the channel	T/F	
NumSegs	Total number of segments in Program	Integer	
MaxSegs ^[W]	Maximum number of segments in Program	Integer	
MaxEvents ^[W]	Maximum number of Events in Program	Integer	
MaxExit ^[W]	Maximum number of exit conditions in Program	Integer	
MaxWait ^[W]	Maximum number of wait conditions in Program	Integer	
MaxUval ^[W]	Maximum number of User Values in Program	Integer	
MaxSP ^[W]	Maximum Target Setpoint of all segments	Eng	
MinSP ^[W]	Minimum Target Setpoint of all segments	Eng	
HL_SP ^[W]	High limit of Setpoint range	Eng	
LL_SP ^[W]	Low limit of Setpoint range	Eng	
DP_PV ^[W]	Number of Decimal places for PV and TgtSP	Enum	
DP_Hback ^[W]	Number of Decimal places for Holdback	Enum	
DP_Rate ^[W]	Number of Decimal places for Rate	Enum	

Continued...

Parameter	Function	Units	Status
<i>Continued...</i>			
Alarms			
See Monitor Page			
Segments	First Segment block	Block	
Servo	Startup control for Setpoint in channel	Enum	
PVStart	Defines PV starting point in Program	Enum	
HbMode ^[W]	Defines Holdback mode of channel	Enum	
EndCond	Defines operation used when Program is complete	Enum	
HL_Uval1 ^[W] to HL_Uval4 ^[W]	High limit of User defined values	Eng	
LL_Uval1 ^[W] to LL_Uval4 ^[W]			
DP_UV1 ^[W] to DP_UV4 ^[W]	Number of Decimal places for User defined values	Integer	
Names Page			
PVName ^[W]	Displays name of PV currently being profiled		
EvName1 ^[W] to EvName16 ^[W]	Names assigned to Digital Events		
Alarms			
See Monitor Page			
UvName1 ^[W] to UvName4 ^[W]	Names assigned to User Values		
Names2 Page			
ExName1 ^[W] to ExName8 ^[W]	Names assigned to channel Exit Condition		
WaName1 ^[W] to WaName8 ^[W]	Names assigned to channel Wait Condition		
Enums Page			
Ev1Off ^[W] to Ev16Off ^[W]	Off Text assigned to channel Digital Events		
Ev1On ^[W] to Ev16On ^[W]	On Text assigned to channel Digital Events		
Alarms			
See Monitor Page			
Enums2 Page			
Ex1Off ^[W] to Ex8Off ^[W]	Off Text assigned to channel Exit Condition		
Ex1On ^[W] to Ex8On ^[W]	On Text assigned to channel Exit Condition		
Alarms			
See Monitor Page			
Wa1Off ^[W] to Wa8Off ^[W]	Off Text assigned to channel Wait Condition		
Wa1On ^[W] to Wa8On ^[W]	On Text assigned to channel Wait Condition		

[W] Configured by the Programmer Wizard

Table 190 Block parameters

Block specification menu

The following is given in addition to *Table 190*.

Dbase, Block, Type. See *Appendix D page 544* for details of these 'header' fields.

NOTE These fields automatically appear in each of the pages in this block.

Monitor Page

This page provides parameters used to monitor the operation of the loaded Program.

SegName. This shows which segment in the Program is currently being executed.

SegID. For future use.

SegNo. This shows the number of the segment currently being executed.

SegType. This shows the type of the segment currently being executed.

SegDur. This shows the period of time that the segment currently being executed will take to complete.

TgtSP. This shows the Setpoint value required by the Program channel at the end of the currently executing segment. It is derived from each specific SEGMENT block corresponding to a particular segment in the Program (see note below).

Rate. This shows the current rate at which the Setpoint in the Segment currently being executed is achieving the target value (see note below).

NOTE If the Program is operating, TgtSP and Rate values cannot be edited unless the Program is Held, *PROGCTRL.Monitor.StateFlg* is TRUE. Changes to the Target Setpoint value in other segments will take effect only when the segment is executed.

CurrSP. This shows the current Setpoint value limited by *Config.HL_SP* and *Config.LL_SP*. This field can only be edited when *PROGCTRL.Mode* is set to Man. To provide setpoint control for a PID control loop, wire this to the *LOOP_PID.SP.AltSP* field in the strategy.

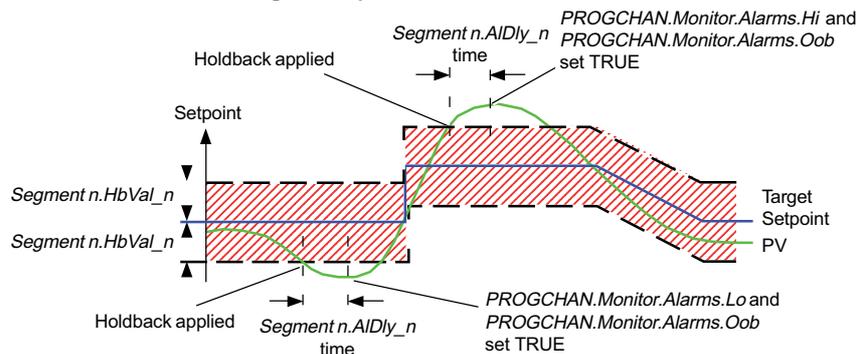
PV. This shows the current process variable input to this block. This should be wired to the *LOOP_PID.Main.PV* to control the setpoint of the PID control loop.

SP. This shows the current value of the Setpoint being used by the control loop and represents the operator Setpoint for the control loop. This should be wired from the *LOOP_PID.Main.WSP* of the PID control loop.

HbType. Used to configure the criteria for applying Holdback and asserting associated alarms if the priority is set greater than 0 (zero).

NOTE Holdback extends the duration of a Setpoint ramp allowing the measured PV to 'catch up' with its' SP (setpoint), if it has deviated from it by more than a specified amount. If PV deviates from its' SP by more than a specified amount (*Monitor.HbVal*), the associated *Monitor.Alarms* TRUE are asserted and the configured Holdback operation is initiated.

- **HiHback, LoHback.** If TRUE and if *Monitor.PV* is greater than $Monitor.CurrSP + Monitor.HbVal$, or less than $Monitor.CurrSP - Monitor.HbVal$ ramping is held. If this persists for longer than the time defined in *Monitor.AIDly*, *Monitor.Alarms.Holdback* is set TRUE.
- **HiAlarm, LoAlarm.** If TRUE and if *Monitor.PV* is greater than $Monitor.CurrSP + Monitor.HbVal$, or less than $Monitor.CurrSP - Monitor.HbVal$ for longer than the time defined in *Monitor.AIDly*, *Monitor.Alarms.Hi* or *Monitor.Alarms.Lo* are set TRUE, respectively.



- **HiOob, LoOob.** If TRUE and if *Monitor.PV* is greater than $Monitor.CurrSP + Monitor.HbVal$, or less than $Monitor.CurrSP - Monitor.HbVal$, for longer than the time defined in *Monitor.AIDly*, *Monitor.Alarms.Oob* is set TRUE.

HbVal. This shows the deviation value between *Monitor.CurrSP* and *Monitor.PV* used to stop the Setpoint ramp when the PV is unable to keep up with the changing Setpoint. The value is configured in *SEGMENT.Segment n.HbValn*, where n is the Segment page identifier. If *PROGCHAN.HbMode* is set to PerChan, the value is stored in the first Segment and applies to all segments in the channel.

NOTE The HbVal field may be edited whilst the program is operating

AIDly. Used to define the delay in time, up to 12 hours, between detecting an alarm condition and asserting the corresponding alarm bitfield. This time delay period provides a filter for nuisance alarm conditions.

EventOut. Bitfield showing the output events currently being executed in this segment of the Program. It is derived from each specific SEGMENT block corresponding to a particular segment in the Program.

NOTE The number of digital events is not limited by the number of channels entered. If more digital events are declared than the number of given channels can provide, additional PROGCHAN blocks will be created. However, these additional blocks will not be profiled as only the digital events are being used.

- **Event1 to Event16.** TRUE, if the digital event is currently active in the Segment. Wire a *Monitor.Eventn* to an output block to enable/disable a connected digital field as the Segment is started.

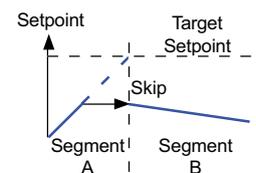
NOTE If the Program is operating these values cannot be edited unless the Program is Held, *PROGCTRL.Monitor.StateFlg* is TRUE. Changes to the output event values in other segments will only take effect when the segment is executed.

UVal1 to UVal4. This shows the current value of the corresponding User Values, derived from SEGMENT blocks that correspond to each particular segment in the Program. The number of User Values is not limited by the number of channels entered. If more User Values are declared than the number of given channels can provide, additional PROGCHAN blocks will be created. However, these additional blocks will not be profiled as only the User Values are being used. These values can be wired to a destination within the instrument for use in a particular application. A different value may be set in each segment via the Programmer Editor. Wire a *Monitor.UValn* to an output block to write the configured User Value to the connected analogue value field as the Segment is started.

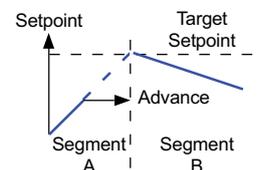
NOTE If the Program is operating with these values, they cannot be edited unless the Program is Held, *PROGCTRL.Monitor.StateFlg* is TRUE. Changes to the User Values in other segments will only take effect when the segment is executed.

Command. For Future Use - Bitfield controlling the operation of the segment currently being executed.

- **Skip.** TRUE, will move the Program to the next Segment immediately and start it using the current Setpoint value, *Monitor.CurrSP*.



- **Advance.** TRUE, will set the current Setpoint value, *Monitor.CurrSP* to equal the Target Setpoint, *Monitor.TgtSP* and move the Program to the next Segment.



JogTime. This defines the amount of time in minutes a dwell segment will jog when requested, i.e. *PROGCTRL.Monitor.Command.Jog* is TRUE. Set +ve to jog forwards, or -ve to jog backwards.

EditCmnd. (NoAction/Delete/Insert). Shows the command performed on the Segment specified in *Monitor.EditSeg*. When set to **NoAction** the Segment defined in *Monitor.EditSeg* no other action will occur. If set to **Delete** the Segment defined in *Monitor.EditSeg* will be deleted from the Program, and all subsequent Segments are moved down by 1 and the total number of Segments decrements. Segments can only be deleted when the program is not running. If set to **Insert** a Segment will be added to the Program before the Segment defined in *Monitor.EditSeg*, and all subsequent Segments are moved up by 1. The inserted Segment is configured as a zero duration dwell and if left in this condition, it will have no effect on the Program when run. **NoAction** is automatically reset when the defined command is complete.

SegLeft. Shows the Segment time remaining for this segment.

Alarms. See Appendix D page 545 for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.
- **Holdback.** Asserted, if the *Monitor.HbType.HiHback* or *Monitor.HbType.LoHback* bitfield is TRUE and *Monitor.PV* has exceeded the rules specified by *Monitor.HbType*.
- **Hi, Lo.** Asserted, if the *Monitor.HbType.HiAlarm* or *Monitor.HbType.LoAlarm* bitfield is TRUE and *Monitor.PV* has exceeded the rules specified by *Monitor.HbType*.
- **Oob.** Asserted, if the *Monitor.PV* value from this channel has exceeded a defined boundary during execution. This remains asserted until the Program is reset.
- **Waiting.** Asserted, if this Segment will not complete because of an active Wait condition.
- **Combined.** Asserted if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

Cond. Bitfield showing the current condition of the active Segment in each of the channels.

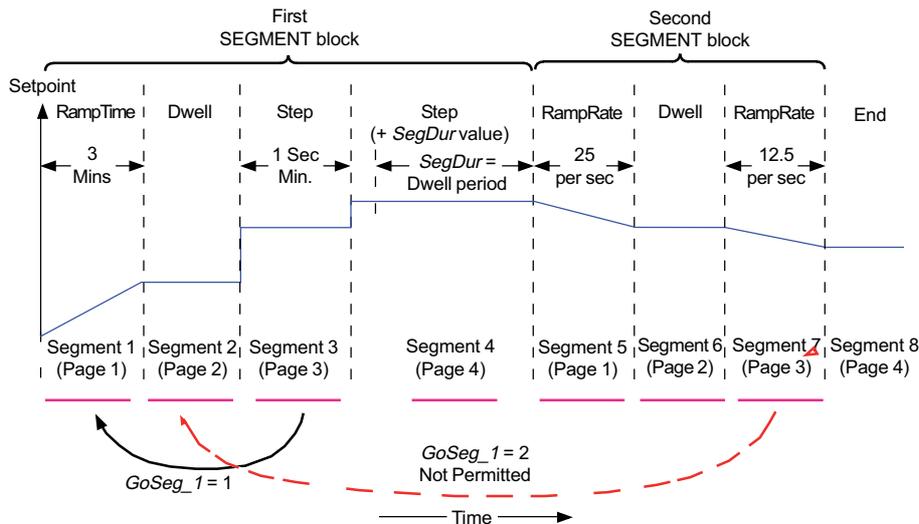
NOTE The table below shows the relationship of control between *Cond_n* bitfields, *InvCnd_n* bitfields and the *CurrCond* bitfields.

InvCnd. Bitfields used to show an inverted Segment condition state.

CurrCond. Bitfield showing the current condition of the current Segment in each of the channels.

Cond_n	InvCnd_n	Segment will ...
Exit_n is FALSE		continue as normal. An Exit condition does not exist.
Exit_n is TRUE	Exit_n is FALSE	proceed to next segment before it has reached its normal completion time, if <i>PROGCHAN.CurrCond.Exitn</i> is TRUE.
Exit_n is TRUE	Exit_n is TRUE	proceed to next segment before it has reached its normal completion time, if <i>PROGCHAN.CurrCond.Exitn</i> is FALSE.
Wait_n is FALSE		continue as normal. A Wait condition does not exist.
Wait_n is TRUE	Wait_n is FALSE	do not proceed to the next Segment, if <i>PROGCHAN.CurrCond.Waitn</i> is TRUE.
Wait_n is TRUE	Wait_n is TRUE	do not proceed to the next Segment, if <i>PROGCHAN.CurrCond.Waitn</i> is FALSE.

GoSeg. This defines the next non-sequential Segment that will be executed when this current Segment is complete. It is the number of the first Segment in a Go Back sequence when set to a value more than 0 (zero). The segment initiating the Go Back sequence is considered as the last segment in the sequence. This allows the implied segments to be repeated the number of times specified in *GoMax_1*. If multiple Go Back sequences are configured *GoSeg* must not return to a segment before the previous Go Back sequence, see below.



GoMax. This defines the total number of times (1-999) the next non-sequential portion of the Program will be executed when the Segment is complete.

GoNum. This shows the current iteration.

SyncTo. For Future Use - This defines the Segment number in the Program used to synchronise the execution of the Segment.

StateFlg. Bitfields showing the state of the current channel.

- **Wait1 to Wait8.** TRUE shows which Wait condition(s) are preventing the current segment from finishing.
- **ChActive.** TRUE shows that a channel is currently being Profiled.
- **End.** TRUE shows a Segment has completed and any Wait conditions have been successfully obtained.
- **Holdback.** TRUE shows the measured PV (*Monitor.PV*) has deviated from the setpoint (*Monitor.CurrSP*) by more than a specified amount (*Monitor.HbVal*) causing a Holdback action to occur in a Segment. When set TRUE, the configured Holdback Type and Holdback Mode are applied.

EditSeg. This defines the Segment that will be edited according to *Monitor.EditCmnd*.

ProgLeft. Shows the Program time remaining in this cycle for this channel.

Config Page

This page provides parameters used to show Programmer configuration.

Options. Bitfield for optional channel configuration.

- **DisChan.** TRUE, disables the Profiling of this channel. This is automatically set TRUE in PROGCHAN blocks that provide additional digital events and user values for the last enabled PROGCHAN block.

NumSegs. This shows the number of Segments in the channel, including the END Segment. It automatically increments when a Segment is added, and decrements when a Segment is deleted.

MaxSegs. This shows the maximum number of Segments required by the Program as specified by the Programmer Wizard.

NOTE This description will also apply to *Config.DP_UV1* to *Config.DP_UV4*.

MaxEvent. This shows the maximum number of output events required by the Program as specified by the Programmer Wizard. This avoids scrolling through unwanted events used by other Segments.

MaxExit, MaxWait. This shows the maximum number of Exit conditions and Wait conditions in the Program respectively.

MaxUval. This value should be configured and edited using Programmer Wizard. This shows the maximum number of User Values required by the Program.

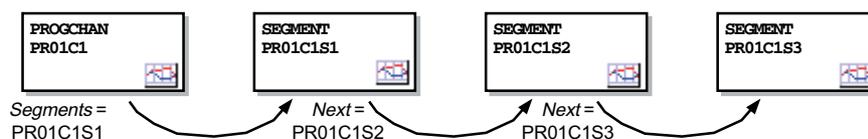
MaxSP, MinSP. This shows the maximum and minimum Target Setpoint values detected from all Segments in the channel respectively. *PROGCTRL.Config.MaxSP* can be wired to a TC_LIFE block to determine if a Program may violate the permitted maximum temperature of a thermocouple.

HL_SP, LL_SP. These high and low Setpoint limit values should be configured and edited using Programmer Wizard. Setpoint limits are active in all modes of operation, and restrict the target setpoint values configured in the Programmer Editor.

DP_PV, DP_Hback, DP_Rate. (SCI/0/1/2/3/4). This value should be configured and edited using Programmer Wizard. Defines the number of decimal places used when displaying the value defined in *PROGCTRL.Monitor.PV* and *PROGCTRL.Monitor.TgtSP*, *PROGCTRL.Monitor.HbVal*, and *Monitor.Rate*. **SCI** indicates that scientific notation will be used, e.g. 1.31e+1. **0, 1, 2, 3, and 4** indicate the number of decimal places used.

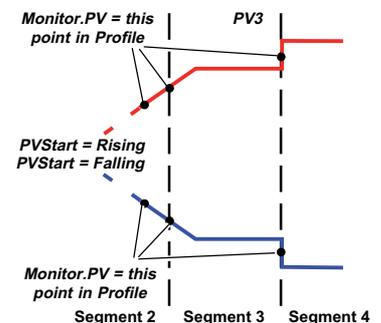
NOTE This description will also apply to *Config.DP_UV1* to *Config.DP_UV4*.

Segments. This shows the name of the first SEGMENT block containing Program segment data.



Servo. (ServoSP/ServoPV). Defines the starting value of the Program used when *PROGCTRL.Monitor.Command.Start* is set TRUE. **ServoPV** indicates that the Program will start from the current value derived from *PROGCTRL.Monitor.PV*. This option is used in conjunction with *PROGCTRL.Config.PVStart*. **ServoSP** indicates that the Program will start from the current value derived from *PROGCTRL.Monitor.SP*. This is ignored if *Config.PVStart* is set to **Rising** or **Falling**.

PVStart. (Off/Rising/Falling). This determines the starting point of the Program used when *PROGCTRL.Monitor.Command.Start* is set TRUE. This allows the Program to advance to a point in any segment of the Profile corresponding to the current PV. **Off** indicates that the Program will start at the first Segment. **Rising** and **Falling** indicates that the Program will advance to the point in the Program, possibly in the middle of a segment, where the value reaches or exceeds *Monitor.PV*.



HbMode. (Off/PerChan/PerSeg). **Off** indicates that Holdback will not be applied in the Program. **PerChan** indicates that the value in *PROGCHAN.Monitor.HbVal* of the first segment in the channel applies to all other segments in which *PROGCTRL.Monitor.HbType* is not Off. **PerSeg** indicates that each segment has its own configured Holdback value.

EndCond. (Reset/Dwell). Defines the action taken when the last Segment in a Program has completed. **Reset** indicates the Program will automatically return to the Idle state, *PROGCTRL.Monitor.State* shows Idle. **Dwell** indicates the value shown in *PROGCTRL.Monitor.CurrSP* will remain at the same value until the Program is reset.

HL_Uval1 to HL_Uval4, LL_Uval1 to LL_Uval4. This shows the high limit and low limit for each of the configured User Values derived from the Programmer Wizard. This is the value at which the corresponding User Value, *SEGMENT.Segment n.UVal1* to *SEGMENT.Segment n.UVal4*, will be clipped.

DP_UV1 to DP_UV4. (SCI/0/1/2/3/4). This value should be configured and edited using Programmer Wizard. Defines the number of decimal places used when displaying the corresponding User Value defined in *PROGCTRL.Monitor.UVal1* to *PROGCTRL.Monitor.UVal4*, see *PROGCTRL.Config.DP_PV*, *PROGCTRL.Config.DP_Hback*, and *PROGCTRL.Config.DP_Rate*.

Names Page

This page provides parameters used to identify the Profile currently in operation.

PVName. This field is used to identify the channel. It shows the name of the variable currently being Profiled and corresponds to the string entered in the Name field on the Profiled Channel page of the Programmer Wizard.

EvName1 to EvName16. These fields are used to identify a Digital Event. Each field shows the name of a digital event output that exists in the Program and corresponds to one of the strings entered in the Digital Event Name field on the Digital Event page of the Programmer Wizard.

UvName1 to UvName4. These fields are used to identify a User Value. Each field shows the name of a User Value that exists in the Program and corresponds to one of the strings entered in the User Value Name field on the User Value page of the Programmer Wizard.

Names2 Page

This page provides parameters used to identify the condition of each channel in the Program.

ExName1 to ExName8, WaName1 to WaName8. These fields are used to identify an Exit Condition or a Wait Condition. Each field shows the name of an Exit condition and/or a Wait condition that exists in the Program and corresponds to one of the strings entered in the Condition Name field on the Condition page of the Programmer Wizard.

Enums Page

This page provides parameters used to show the Off and On values corresponding to each Digital Event Output included in the Profile of this channel.

Ev1Off to Ev16Off, Ev1On to Ev16On. These fields are used to identify the enumeration configured for each Digital Event condition. Each field shows the text string associated with each digital event output Off and digital event output On value and corresponds to one of the strings entered in the Off Text field or On Text field on the Digital Event page of the Programmer Wizard.

Enums2 Page

This page provides parameters used to show the Off and On values corresponding to each Exit and Wait condition included in the Profile of this channel.

Ex1Off to Ex8Off, Wa1Off to Wa8Off. These fields are used to identify the enumeration configured for each Exit or Wait condition. Each field shows the text string associated with each Exit and Wait Off value and corresponds to one of the strings entered in the Off Text field on the Condition page of the Programmer Wizard.

Ex1On to Ex8On, Wa1On to Wa8On. These fields are used to identify the enumeration configured for each Exit or Wait condition. Each field shows the text string associated with each Exit and Wait On value and corresponds to one of the strings entered in the On Text field on the Condition page of the Programmer Wizard.

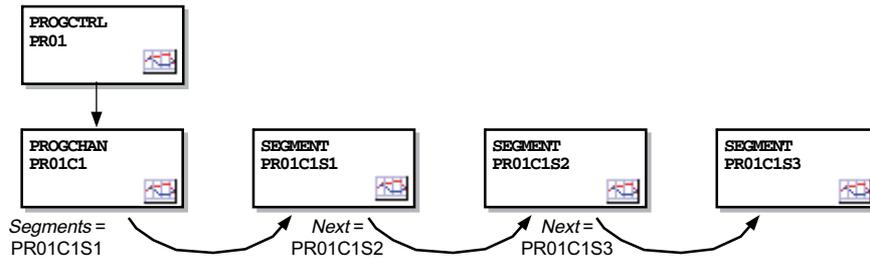
SEGMENT: PROGRAMMER SEGMENT BLOCK

Block function

This block provides parameters for up to 4 Segments for a single channel being profiled by the Program configured using LINTools. A maximum of 8 SEGMENT blocks, 4 Segments per block, provide up to 32 segments per Program.

Each of the 4 Segment pages of a block correspond to a configured Setpoint Program segment displayed in the Programmer Editor. If more than 4 Setpoint Program segments are configured the next SEGMENT block is referenced. The first SEGMENT block is linked to the PROGCHAN block.

NOTE An END Segment will account for one segment in a Program, and will therefore use one Segment page of a SEGMENT block.



When the Program is operating only the corresponding page of the SEGMENT block configuration is applied. All other SEGMENT block configuration will not take effect until the specific segment is executed. However, the current Target Setpoint, output Events and User Values must be edited in the *PROGCHAN.Monitor.TgtSP*, *PROGCHAN.Monitor.EventOut* bitfields and *PROGCHAN.Monitor.UVal1* to *PROGCHAN.Monitor.UVal4* as appropriate.

It is one block type in a suite of blocks that includes the PROGCTRL block and PROGCHAN block, held in a **PROG_WIZ** compound when automatically created using the Programmer Wizard. Together these blocks provide access to all parameters in the Setpoint Program.

Important Always use the Programmer Wizard to configure the PROG_WIZ compound. This automatically combines the required blocks, and generates the complete Program Template file that must be downloaded to the instrument.

NOTE This block functions using a number of pages. Fields can be located using <Page>.<Field>.<Subfield> convention.

Block parameters

Symbols used in Table 191 are explained in Table 1. Additional parameter information is given in the Block specification menu section.

Parameter	Function	Units	Status*
Segment 1 Page			
SegNam_1	Name of segment	String	
SegId_1	For future use - 4 character string identifying segment	String	
SegTyp_1	Segment type	Enum	
SegDur_1	Duration of Segment	Integer	
TgtSP_1	Required Target Setpoint for the Segment	Eng	
Rate_1	Current rate to target	Integer	
HbType_1	Current Holdback type	Enum	
HbVal_1	Current Holdback value	Eng	
AlDly_1	Time delay before asserting alarm	Time	
Events_1	Event output indicators	ABCD hex	
Event1 to Event16	Digital event output	T/F	
UVal1_1 to UVal4_1			

Continued...

Parameter	Function	Units	Status*
<i>Continued...</i>			
Alarms ☐➔☒🔊			
Software	Block RAM data sumcheck error / network failure	T/F	
Combined	OR-ing of all Alarms bits	T/F	
Cond_1	Segment condition	ABCD hex	
Exit1 to Exit8	} Exit condition exists if TRUE	T/F	
Wait1 to Wait8			
InvCnd_1	Inverts the Segment condition state	ABCD hex	
Exit1 to Exit8	} Inverted Exit condition state	T/F	
Wait1 to Wait8			
GoSeg_1	Segment to GoBack to when current segment completes	Integer	
GoMax_1	Maximum number of times to GoBack	Integer	
SyncTo_1	Segment used to synchronise with	Integer	
Segment 2, Segment 3, and Segment 4 Page			
See Segment 1 Page			
Alarms ☐➔☒🔊			
See Monitor Page			
Next	Segment block extending Profile (Segment 4 page only)	Block	

* All fields are read-only unless otherwise stated.

Table 191 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

NOTE These fields automatically appear in each of the pages in this block.

Segment 1 Page

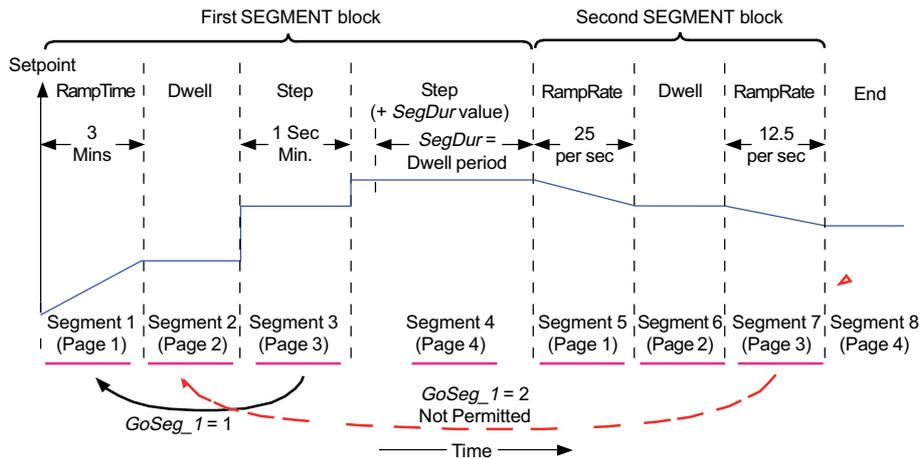
This page provides parameters used to show the operational conditions of the Program segment corresponding to this Segment page in the SEGMENT block.

SegNam_1. Used to show the Segment name.

SegId_1. For future use - Shows a 4 character ID string.

SegTyp_1. (STEP/DWELL/RAMPTIME/RAMPRATE/END). Used to show the type of the segment to be executed. Set to **STEP** to change the operating Setpoint value to the configured Target Setpoint value. Set to **DWELL** to inherit the Setpoint from the previous Segment and retain it at a constant value for a specified period. Set to **RAMPTIME** to increase the operating Setpoint to a defined Target Setpoint over segment duration. Set to **RAMPRATE** to increase the operating Setpoint at a desired rate of change, see *Rate_1*, until a defined Target Setpoint is obtained indicating the end of the Segment. Set to **END** to indicate the last Segment of the channel.

NOTE **RAMPRATE** is only available in Single Channel Programs.



SegDur_1. Used to define the amount of time the Segment will take to complete. Configured in conjunction with the **RAMPTIME** Segment type to define the duration of the Ramp, and the **DWELL** Segment type. It can also be used with a **STEP** Segment type to define a dwell period once the Target Setpoint is achieved and includes the time taken to process the **STEP** Segment.

NOTE The time displayed when connected to the instrument may be incorrect if the duration of the Program or Segment exceeds 36 hours, but this does not affect the execution of the Program.

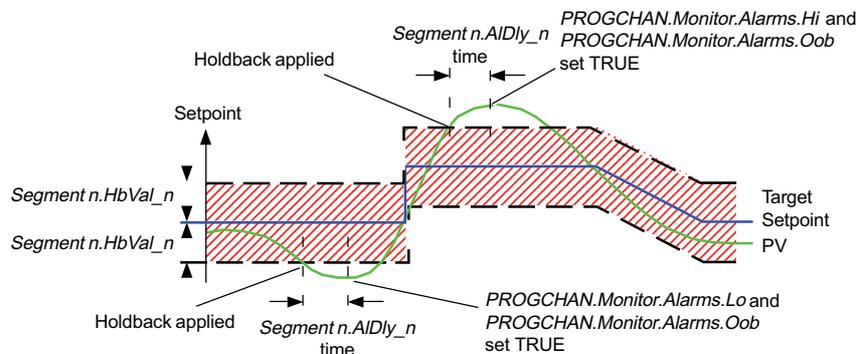
TgtSP_1. Used to show the Setpoint value required by the Program channel at the end of the Segment for starting the next segment. Unavailable when *Segment n.SegTyp_n* shows **DWELL**.

Rate_1. Used to show the rate at which the operating Setpoint in the Segment will achieve the Target Setpoint value, *SEGMENT n.TgtSP_n* in Programs running a single channel. However, in Programs running multiple channels, this shows the calculated rate at which the operating Setpoint will attain the Target Setpoint in a **RAMPTIME** Segment.

HbTyp_1. Used to show the criteria for applying Holdback and asserting associated alarms if the priority is set greater than 0 (zero).

NOTE Holdback extends the duration of a Setpoint ramp allowing the measured PV to 'catch up' with its' SP (setpoint), if it has deviated from it by more than a specified amount. If PV deviates from its' SP by more than a specified amount (*Monitor.HbVal*), the associated *Monitor.Alarms* **TRUE** are asserted and the configured Holdback operation is initiated.

- **HiHback, LoHback.** If **TRUE** and if *Monitor.PV* is greater than $Monitor.CurrSP + Monitor.HbVal$, or less than $Monitor.CurrSP - Monitor.HbVal$ ramping is held. If this persists for longer than the time defined in *Monitor.AIDly*, *Monitor.Alarms.Holdback* is set **TRUE**.
- **HiAlarm, LoAlarm.** If **TRUE** and if *Monitor.PV* is greater than $Monitor.CurrSP + Monitor.HbVal$, or less than $Monitor.CurrSP - Monitor.HbVal$ for longer than the time defined in *Monitor.AIDly*, *Monitor.Alarms.Hi* or *Monitor.Alarms.Lo* are set **TRUE**, respectively.



- **HiOob, LoOob.** If **TRUE** and if *Monitor.PV* is greater than $Monitor.CurrSP + Monitor.HbVal$, or less than $Monitor.CurrSP - Monitor.HbVal$, for longer than the time defined in *Monitor.AIDly*, *Monitor.Alarms.Oob* is set **TRUE**.

HbVal_1. Used to the deviation value between *Monitor.CurrSP* and *Monitor.PV* used to stop the Setpoint ramp when the PV is unable to keep up with the changing Setpoint. The value is configured in *SEGMENT.Segment n.HbValn*, where *n* is the Segment page identifier. If *PROGCHAN.HbMode* is set to *PerChan*, the value is stored in the first Segment and applies to all segments in the channel.

AIDly_1. Used to show the delay in time, up to 12 hours, between detecting an alarm condition and asserting the corresponding alarm bitfield. This time delay period provides a filter for nuisance alarm conditions.

Events_1. Bitfield showing the current state of the digital event outputs of the Program corresponding to this segment.

NOTE The number of digital event outputs is not limited by the number of channels entered. If more digital events are declared than the number of given channels can provide, additional PROGCHAN blocks (8 maximum) will be created. However, these additional blocks will not be profiled as only the digital events are being used.

- **Event1 to Event16.** TRUE shows a digital event output applies in a segment of the Program corresponding to this Segment page in the SEGMENT block.

UVal1_1 to UVal4_1. Used to show a user specified value for the corresponding User Value Names configured using Programmer Wizard.

NOTE The number of User Values is not limited by the number of channels entered. If more User Values are declared than the number of given channels can provide, additional PROGCHAN blocks (8 maximum) will be created. However, these additional blocks will not be profiled as only the User Values are being used.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.
- **Combined.** Asserted if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

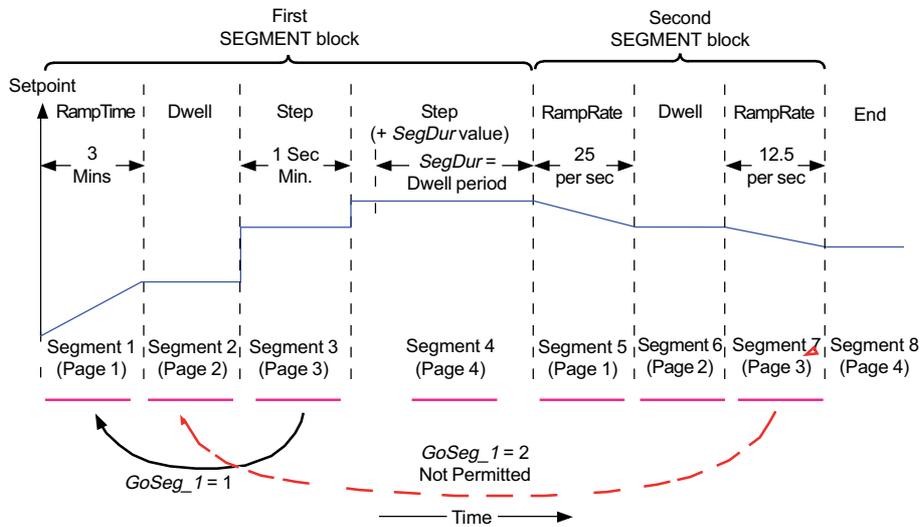
Cond_1. Bitfield showing the current condition of the active Segment in each of the channels.

NOTE The table below shows the relationship of control between *Cond_n* bitfields, *InvCnd_n* bitfields and the *CurrCond* bitfields.

InvCnd_1. Bitfields used to show an inverted Segment condition state.

Cond_n	InvCnd_n	Segment will ...
Exit_n is FALSE		continue as normal. An Exit condition does not exist.
Exit_n is TRUE	Exit_n is FALSE	proceed to next segment before it has reached its normal completion time, if <i>PROGCHAN.CurrCond.Exitn</i> is TRUE.
Exit_n is TRUE	Exit_n is TRUE	proceed to next segment before it has reached its normal completion time, if <i>PROGCHAN.CurrCond.Exitn</i> is FALSE.
Wait_n is FALSE		continue as normal. A Wait condition does not exist.
Wait_n is TRUE	Wait_n is FALSE	do not proceed to the next Segment, if <i>PROGCHAN.CurrCond.Waitn</i> is TRUE.
Wait_n is TRUE	Wait_n is TRUE	do not proceed to the next Segment, if <i>PROGCHAN.CurrCond.Waitn</i> is FALSE.

GoSeg_1. Used to show the number of the first Segment in a Go Back Sequence when set to a value more than 0 (zero). The segment initiating the Go Back Sequence is considered as the last segment in the sequence. This allows the implied segments to be repeated the number of times specified in *GoMax_1*. If further Go Back Sequences are configured this value must not return to a segment before the previous Go Back Sequence, see below.



GoMax_1. Used to show the total number of times the Go Back Sequence will be applied. Go Back is not applied when this shows 0 (zero).

SyncTo_1. For future use - Used to show the Segment number in the Program used to synchronise the execution of the Segment.

Segment 2, Segment 3 and Segment 4 Page

These pages provide identical parameters used to show the operational conditions of each loaded Program segment; see Segment 1 page.

Next. (Segment 4 page only) Used to show the next SEGMENT block in the Program that provides up to 4 additional Program segments.

CHAPTER 17 RECORDER FUNCTION BLOCKS

The RECORDER category of Function Block Templates provides the control strategy with functions for data recording. The DR_ANCHP and DR_DGCHP blocks allow flash memory recording of analogue and digital points, in video chart recorders. The analogue and digital points are 'wired' as inputs to the blocks from the control strategy. For flash recording to happen, the blocks must be organised into groups and areas via GROUP and AREA blocks (in the ORGANISE category), as shown in *Figure 98*.

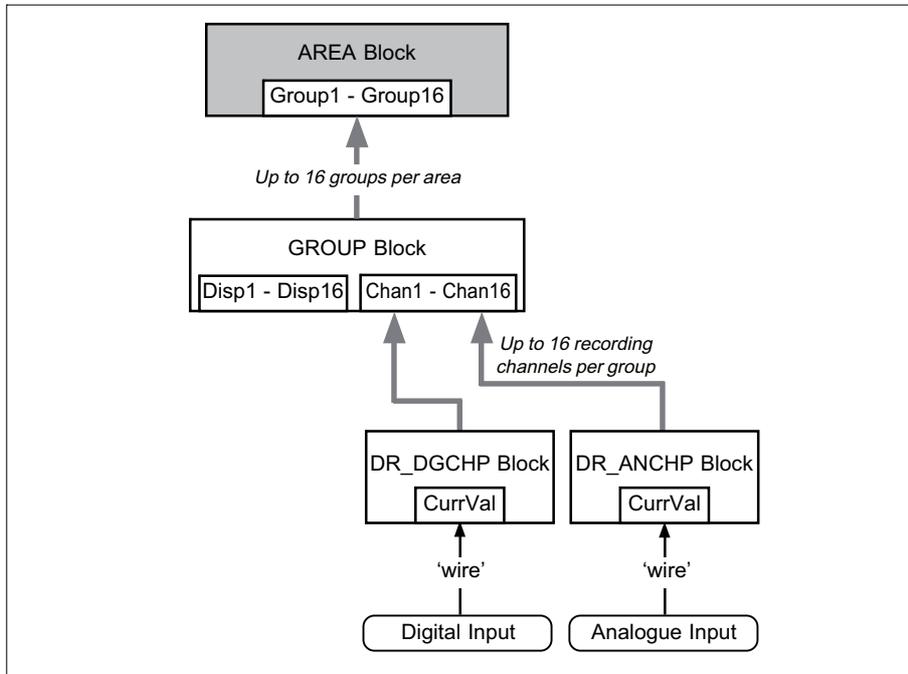


Figure 98 Organisation of recorder blocks into groups and areas

NOTE The DR_ANCHP and DR_DGCHP point blocks also allow data to be permanently archived onto removable media for replay offline. This is data logging, as opposed to flash recording which is replayed in the instrument. For data logging, the point blocks must be organised into *logging groups* via special ORGANISE category blocks, LOGDEV, LGROUP, and LOGRPEX, see Organise function block category for details.

DR_ANCHP: DATA RECORDING ANALOGUE CHANNEL POINT BLOCK

Block function

The DR_ANCHP block lets you record the history of a set of analogue data values onto the local flash storage system of the instrument running the block. The record can be subsequently replayed on the instrument.

The block specifies how the analogue value is to be reviewed and recorded. The functionality of both measured and derived analogue channels is provided by the block.

The analogue point to be recorded is ‘wired’ in the control database to the block’s *CurrVal* input field. The block must be attached to a GROUP block, via a *Chann* field, for recording to take place. A total of up to 16 data recording blocks (DR_ANCHP and DR_DGCHP blocks) can be attached to a given GROUP block.

Block parameters

Symbols used in *Table 192* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
CurrVal	Current point value	Eng	☑
Quality	Quality of CurrVal — Recorder definitions	(A)BCD hex	☑
GOOD	GOOD	T/F — 1	D
PV_OFF	PV_OFF	T/F — 2	
O_RANGE	OVER_RANGE	T/F — 4	
U_RANGE	UNDER_RANGE	T/F — 8	
HW_ERROR	HARDWARE_ERROR	T/F — 1	C
RANGING	RANGING	T/F — 2	
OVERFLOW	OVERFLOW	T/F — 4	
BAD_PV	BAD_PV	T/F — 8	
HW_EXCED	HARDWARE_EXCEEDED	T/F — 1	B
NO_DATA	NO_DATA	T/F — 2	
LIN_TABL	TOO_MANY_LIN_TABLES	T/F — 4	
		T/F — 8	
Trace_On	TRUE = trace on (default), FALSE = disabled	T/F	☑
Format	The display & logging format of CurrVal	Menu	
ColourA	Colour to be used for the trace	Menu	
ColourB	Alternative colour to be used for the trace	Menu	
ColB_On	TRUE selects colour B, FALSE selects A (default)	T/F	☑
ZoneA_HI	Zone A high value (0-100% of chart width, 100%=default)	%	
ZoneA_LO	Zone A low value (0-100% of chart width, 0%=default)	%	
ZoneB_HI	Zone B high value (0-100% of chart width, 100%=default)	%	
ZoneB_LO	Zone B low value (0-100% of chart width, 0%=default)	%	
ZoneB_On	TRUE selects zone B, FALSE selects A (default)	T/F	☑
Alarms			☑ ☑ ☑ ☑
Software	Block RAM data sumcheck error / network failure	T/F	
<AlmTyp_1>	User alarm defined by AlmTyp_1	T/F	
<AlmTyp_2>	User alarm defined by AlmTyp_2	T/F	
<AlmTyp_3>	User alarm defined by AlmTyp_3	T/F	
<AlmTyp_4>	User alarm defined by AlmTyp_4	T/F	
Combined	OR-ing of all Alarms bits	T/F	
AlmAct	Active user analogue alarms		☑
Alm_1	TRUE = AlmTyp_1 alarm	T/F	
Alm_2	TRUE = AlmTyp_2 alarm	T/F	
Alm_3	TRUE = AlmTyp_3 alarm	T/F	
Alm_4	TRUE = AlmTyp_4 alarm	T/F	
AlmTyp_1-AlmTyp_4	Alarm types for Alm_1 - 4	Menu	
SpanA_HI	Maximum value at zone high (default = 100.0)	Eng	
SpanA_LO	Maximum value at zone low (default = 0.0)	Eng	
SpanB_HI	Maximum value at zone high (default = 100.0)	Eng	
SpanB_LO	Maximum value at zone low (default = 0.0)	Eng	

Table 192 Block parameters

Block specification menu

The following is given in addition to *Table 192*.

Quality. Shows the PV state enumeration used by the manufacturers Recorder products. You can wire any suitable digitals to any *Quality* bitfield, with the output being written to the .uhh file (.pkd) and interpreted by Recorder products and Review accordingly.

- **PV_OFF.** This bitfield can be used to control when a trace is drawn if wired to a condition. When *Quality.PV_OFF* is TRUE a trace will not be drawn on the Trend screen of an Eycon or a chart in Review that has imported the .uhh file from an Eycon, i.e. no data is received from the trace (*Quality.NO_DATA* is TRUE).

NOTE The *Quality.NO_DATA* is controlled automatically. If wired from a cached block with an asserted *Alarm.Software*, the trace will be disabled until the *Alarm.Software* is cleared. This feature cannot be turned off.

Format. Specifies the number format used for the display and logging of the *CurrVal* analogue input. The table below lists the available options with examples.

Option	Example
XXXX	12345
XXXX.X	1234.5
XXX.XX	123.45
XX.XXX	12.345
X.XXXX	1.2345
XXXXXXXX	12345678
XXXXXXXX.X	123456.7
XXXXX.XX	12345.67
XXXX.XXX	1234.567
XXX.XXXX	123.4567
XX.XXXXX	12.34567
X.XXXXXX	1.234567
X.XXXE+XX	1.234E+56

Table 193 Format parameter options with examples

ColourA. (Default/Violet/Blue/Green/Brown/Red/Black/DkViolet/DkBlue/DkGreen/DkBrown/DkRed/DkGrey/LiViolet/LiBlue/LiGreen/LiBrown/LiRed/LiGrey). The colour to be used for the trace. ‘Dk’ and ‘Li’ prefixes represent dark and light versions of the base colours, and dark/light grey. Specifying *Default* causes the colour to be determined by the block’s associated *Chann* number in the GROUP block.

ColourB. (Default/Violet/Blue/Green/Brown/Red/Black/DkViolet/DkBlue/DkGreen/DkBrown/DkRed/DkGrey/LiViolet/LiBlue/LiGreen/LiBrown/LiRed/LiGrey/FIViolet/FIBlue/FIGreen/FIBrown/FIRed/FIblack). As for *ColourA* above, but flashing versions of the base colours and black are also selectable (represented by the ‘FI’ prefix).

ZoneA_LO, ZoneA_HI. The area of the ‘chart’ (perpendicular to the time base) occupied by a trend is called the *zone*. Specified as a pair of values marking the start and end of the zone, each in the range 0% to 100%. *ZoneA_LO* and *ZoneA_HI* specify the low and high values, respectively, for zone A.

ZoneB_LO, ZoneB_HI. (*Zone B specification is not currently implemented.*)

ZoneB_On. (*Zone B specification is not currently implemented.*)

Alarms. <AlmTyp_1> to <AlmTyp_4> alarms are specified by the corresponding parameters (see next), and driven by the respective *Alm_1* to *Alm_4* inputs to the *AlmAct* bitfield.

AlmTyp_1 to AlmTyp_4. (ABS LOW /ABS HIGH /DEV IN /DEV OUT /ROC RISE /ROC FALL) These parameters let you specify four user alarms to appear in the *Alarms* field, which should match the alarm inputs wired into the *AlmAct* bitfield. The possible options are shown in the table below.

Parameter	Function
ABS LOW	Low absolute
ABS HIGH	High absolute
DEV IN	Low deviation
DEV OUT	High deviation
ROC RISE	Rate of increase
ROC FALL	Rate of decrease

Table 194 Alarm type options (AlmTyp_1 to AlmTyp_4)

SpanA_HI, SpanA_LO. ‘Span’ specifies the range of values that are to be displayed in the trend. (The range of values *recorded* is not affected.) It is specified for zone A as a pair of values, in engineering units, for the high - *SpanA_HI* - and low - *SpanA_LO* - of the zone. The trend is effectively ‘clipped’ between these two values.

SpanB_HI, SpanB_LO. (*Zone B specification is not currently implemented.*)

DR_DGCHP: DATA RECORDING DIGITAL CHANNEL POINT BLOCK

Block function

The DR_DGCHP function block lets you record the history of a set of digital data values onto the local flash storage system of the instrument running the block. The record can be subsequently replayed on the instrument.

The block specifies how the digital value is to be reviewed and recorded.

The digital point to be recorded is 'wired' in the control database to the block's *CurrVal* input field. The block must be attached to a GROUP block, via a *Chann* field, for recording to take place. A total of up to 16 data recording blocks (DR_DGCHP and DR_ANCHP blocks) can be attached to a given GROUP block.

Block parameters

Symbols used in *Table 195* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
CurrVal	Current point value	T/F	<input type="checkbox"/>
Quality	Quality of CurrVal — Recorder definitions	(A)BCD hex	<input type="checkbox"/>
PV_OFF	PV_OFF	T/F	D
O_RANGE	OVER_RANGE	T/F	
U_RANGE	UNDER_RANGE	T/F	
HW_ERROR	HARDWARE_ERROR	T/F	
RANGING	RANGING	T/F	C
OVERFLOW	OVERFLOW	T/F	
BAD_PV	BAD_PV	T/F	
HW_EXCED	HARDWARE_EXCEEDED	T/F	
NO_DATA	NO_DATA	T/F	B
LIN_TABL	TOO_MANY_LIN_TABLES	T/F	
Spare		T/F	
Spare		T/F	
Trace_On	TRUE = trace on (default), FALSE = disabled	T/F	<input type="checkbox"/>
ColourA	Colour to be used for the trace	Menu	
ColourB	Alternative colour to be used for the trace	Menu	
ColB_On	TRUE selects colour B, FALSE selects A (default)	T/F	<input type="checkbox"/>
ZoneA_HI	Zone A high value (0-100% of chart width, 100%=default)	%	
ZoneA_LO	Zone A low value (0-100% of chart width, 0%=default)	%	
ZoneB_HI	Zone B high value (0-100% of chart width, 100%=default)	%	
ZoneB_LO	Zone B low value (0-100% of chart width, 0%=default)	%	
ZoneB_On	TRUE selects zone B, FALSE selects A (default)	T/F	<input type="checkbox"/>
Alarms			<input type="checkbox"/>
Software	Block RAM data sumcheck error / network failure	T/F	
DIGITAL1	User digital alarm 1	T/F	
DIGITAL2	User digital alarm 2	T/F	
DIGITAL3	User digital alarm 3	T/F	
DIGITAL4	User digital alarm 4	T/F	
Combined	OR-ing of all Alarms bits	T/F	
AlmAct	Active user digital alarms		<input type="checkbox"/>
Alm_1	TRUE = DIGITAL1 alarm	T/F	
Alm_2	TRUE = DIGITAL2 alarm	T/F	
Alm_3	TRUE = DIGITAL3 alarm	T/F	
Alm_4	TRUE = DIGITAL4 alarm	T/F	

Table 195 Block parameters

Block specification menu

The following is given in addition to *Table 195*.

Quality. Shows the PV state enumeration used by the manufacturers Recorder products. You can wire any suitable digitals to any *Quality* bitfield, with the output being written to the .uhh file (.pkd) and interpreted by Recorder products and Review accordingly.

- **PV_OFF.** This bitfield can be used to control when a trace is drawn if wired to a condition. When *Quality.PV_OFF* is TRUE a trace will not be drawn on the Trend screen of an Eycon or a chart in Review that has imported the .uhh file from an Eycon, i.e. no data is received from the trace (*Quality.NO_DATA* is TRUE).

NOTE The *Quality.NO_DATA* is controlled automatically. If wired from a cached block with an asserted *Alarm.Software*, the trace will be disabled until the *Alarm.Software* is cleared. This feature cannot be turned off.

ColourA. (Default/Violet/Blue/Green/Brown/Red/Black/DkViolet/DkBlue/DkGreen/DkBrown/DkRed/DkGrey/LiViolet/LiBlue/LiGreen/LiBrown/LiRed/LiGrey). The colour to be used for the trace. ‘Dk’ and ‘Li’ prefixes represent dark and light versions of the base colours, and dark/light grey. Specifying *Default* causes the colour to be determined by the block’s associated *Cham* number in the GROUP block.

ColourB. (Default/Violet/Blue/Green/Brown/Red/Black/DkViolet/DkBlue/DkGreen/DkBrown/DkRed/DkGrey/LiViolet/LiBlue/LiGreen/LiBrown/LiRed/LiGrey/FIViolet/FIBlue/FIGreen/FIBrown/FIRed/FIBlack). As for *ColourA* above, but flashing versions of the base colours and black are also selectable (represented by the ‘FI’ prefix).

ZoneA_LO, ZoneA_HI. The area of the ‘chart’ (perpendicular to the time base) occupied by a trend is called the *zone*. Specified as a pair of values marking the start and end of the zone, each in the range 0% to 100%. *ZoneA_LO* and *ZoneA_HI* specify the low and high values, respectively, for zone A.

ZoneB_LO, ZoneB_HI, ZoneB_On. (*Zone B is not currently implemented.*)

Alarms. The *DIGITAL_1* to *DIGITAL_4* alarms are driven by the respective *Alm_1* to *Alm_4* inputs to the *AlmAct* bitfield.

DR_REPRT: REPORT POINT BLOCK

Block function

The DR_REPRT function block generates user-defined text reports and sends them to text devices (printers, or for messages to UHH recording files), and cannot be shared between PGROUP, LGROUP or RGROUP blocks, i.e. it may be referenced only once in the database.

Block parameters

Symbols used in *Table 197* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
FormFile	Name of a form file (extension .UYF)	Alphanumeric	
FormId	Id within FormFile.UYF to use for the report form	Integer	☐
Trigger	Rising edge generates report	T/F	☐
Triggerd	TRUE while report request accepted but not completed	T/F	☐☐
Active	TRUE while report is being generated	T/F	☐☐
Alarms			☐☐☐
Software	Block RAM data sumcheck error / network failure	T/F	
Form	Error in FormFile.UYF form file	T/F	
Combined	OR-ing of all Alarms bits	T/F	
FormErr	Form error code	Menu	☐
FormErLn	Line on which FormErr occurred	Integer	☐
FormErCh	Character position of error in line FormErLn	Integer	☐

Table 197 Block parameters

Block specification menu

The following is given in addition to *Table 197*.

FormFile. A form file is used to define textual output. The .UYF file specifies a text layout, refer to your instrument Product Manual for details of these files. This form is then ‘filled out’ on demand to generate the required textual output.

Triggerd. TRUE while report request accepted but not completed. Another report cannot be initiated (via a *Trigger* input) until *Triggerd* is FALSE again. *Triggerd* returns to FALSE when report generation is complete, at the same time as the *Active* field goes FALSE.

FormErr. (OK /BAD_FILE /LINE_LEN /NEWLINE /MEMORY /SYNTAX /RANGE /NAME /DICTNRY /TYPE /ACTION /FORM /OTHER) Form error code. The options have the following meanings:

- **OK.** No error.
- **BAD_FILE.** File cannot be found, or cannot be read.
- **LINE_LEN.** A line in the file was too long (limit is 255 excluding CR and/or LF).
- **NEWLINE.** File does not end with a newline (possibly indicating corruption).
- **MEMORY.** Form is too big for the memory allocated to it.
- **SYNTAX.** There is a syntax error of some kind.
- **RANGE.** A numeric value is out of its permitted range.
- **NAME.** A named object cannot be found (possibly misspelled).
- **DICTNRY.** A dictionary entry cannot be found.
- **TYPE.** An object type is inappropriate in the context in which it is being used.
- **ACTION.** (*Not used.*)
- **FORM.** A construct was used which is illegal in a form.
- **OTHER.** Other error.

RGROUP: DATA RECORDING GROUP BLOCK

Block function

This block provides run-time control and monitoring of the Data Recording, specifically

- Control of recording, e.g. enabling/disabling, rate, etc.
- Monitoring of recording, e.g. expected duration of locally stored history
- Monitoring of memory life expectancy
- Graphical User Interface support, e.g. alarm collection

A single block must be created for each Data Recording Group that is to be run. This means that each Data Recording Group index requires an RGROUP block.

Block parameters

Symbols used in this table are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status				
Id	Associated Data Recording Group Index number	Integer					
Name	Data Recording Group name						
Enable	Provides Data Recording control	T/F					
UpdateA	Data Recording period A	Secs					
UpdateB	Data Recording period B	Secs					
SelectB	Data Recording period selector	T/F					
SamplNow	On-Demand Data Recording	T/F					
Note	Remote HMI operator note	Alphanumeric					
Alarms		ABCD Hex					
Software	Block RAM data sumcheck error / network failure	T/F					
Config	Incorrect configuration detected	T/F					
Update	Update rate failure	T/F					
Archive	Archiving History file failure	T/F					
LostData	CPU overload causing Data Recording failure	T/F					
MemHW	Data Recording memory specification exceeded	T/F					
Group	OR-ing of all Alarms contained in the Group	T/F					
Combined	OR-ing of all Alarms bits	T/F					
Status	Configuration/recording status	(AB)CD Hex					
Config	Incorrect configuration detected	T/F	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr><td>1</td></tr> <tr><td>2</td></tr> <tr><td>4</td></tr> <tr><td>8</td></tr> </table> D	1	2	4	8
1							
2							
4							
8							
Update	Update rate failure	T/F					
Archive	Archiving History file failure	T/F					
LostData	CPU overload causing Data Recording failure	T/F					
MemHW	Data Recording memory specification exceeded	T/F	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr><td>1</td></tr> <tr><td>2</td></tr> <tr><td>4</td></tr> <tr><td>8</td></tr> </table> C	1	2	4	8
1							
2							
4							
8							
GrpAct	Active alarm detected in Data Recording Group	T/F					
GrpUnack	Unacknowledged alarm detected in Data Recording Group	T/F					
Recordng	Data Recording in progress	T/F	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr><td>1</td></tr> <tr><td>2</td></tr> <tr><td>4</td></tr> <tr><td>8</td></tr> </table> B	1	2	4	8
1							
2							
4							
8							
Batching	Recording start/stop in sync with Batch start/stop	T/F					
Reset	Reset flags	(ABC)D Hex					
Update	Clear Update alarm	T/F	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr><td>1</td></tr> <tr><td>2</td></tr> <tr><td>4</td></tr> <tr><td>8</td></tr> </table> D	1	2	4	8
1							
2							
4							
8							
Archive	Clear Archive alarm	T/F					
LostData	Clear LostData alarm	T/F					
ActUpdat	Time taken to complete the update	Secs					
Duration	Estimated recording time available in instrument	Hours					
Options	For future use						
Default	TRUE if this is the default recording group.	T/F					

Table 198 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

Id. This index number identifies the Data Recording Group in the instrument with this block. If set to 0 (zero) data recording from this block is disabled.

Name. This is the name of the Data Recording Group in the instrument, as specified in the Data Recording Group configuration file (.uxg file).

Enable. Used to control Data Recording in the instrument. TRUE, enables the Data Recording function. FALSE, terminates the Data Recording function.

NOTE A new .uhh file is started when this is set TRUE.

UpdateA, UpdateB. Defines the intended time period between recording values, in the range 1 sec to 3600 secs. The active recording period is defined by *SelectB*. Values are not recorded when this is set to 0 (zero), but alarms and messages will be. This value will be automatically synchronised to record at regular intervals during run-time, e.g. recording values will be synchronised on the hour if set to 600 secs (10mins).

SelectB. Used to select the data recording period. When FALSE, recording period A (*UpdateA*) is operational and if TRUE, recording period B (*UpdateB*) is operational.

SamplNow. Set TRUE to request an additional on-demand sample when data recording is enabled (*Enable* is TRUE) irrespective of *UpdateA* or *UpdateB*. An additional sample will not be recorded if this request coincides with the periodic sampling identified by *UpdateA* or *UpdateB*, specified in *SelectB*.

Note. Allows an operator at a remote HMI to record a note comprising of up to 50 characters. Provided the note is signed and has the “record:yes” flag set, the note’s content will be copied as a signed message into the UHH file. Writing to this field without the “record:yes” flag set causes no action. Note that this field is write-only and therefore never actually visibly appears in the field.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.
- **Config.** Asserted if the Data recording configuration failed. This may occur if *Id* is configured incorrectly, e.g. more than one group has the same *Id*, the *Id* does not correspond to an existing Data Recording Group, or the instrument hardware version does not support Data Recording, i.e. T2550v3 or earlier.

This may also be asserted if data recording has not been licensed.

- **Update.** Asserted if the configured update rate has not been obtained, i.e. *ActUpdat* does not correspond to the selected recording period, *UpdateA* or *UpdateB*, derived from *SelectB*. This remains active until *Reset.Update* is set TRUE, *Enable* is set FALSE, or the recording is reconfigured.
- **Archive.** Asserted if the Data Recording file, .uhh, relating to the group identified in *Id*, was not successfully archived. This occurs if the file was deleted before it was archived. This remains active until *Reset.Archive* is set TRUE, or the recording is reconfigured.
- **LostData.** Asserted if an internal queue overflow causes the loss of recorded data. This may occur if a surge of asynchronous data occurs during the recording, i.e. alarms and messages. This remains active until *Reset.LostData* is set TRUE, *Enable* is set FALSE, or the recording is reconfigured.

NOTE Alarms and messages are not associated with periodic data recording but may affect the *ActUpdat* times.

- **MemHW.** Asserted if the recording memory device has exceeded its specified usage parameter.
- **Group.** Asserted if any alarm within the group is active, identified when *Status.GrpAct* is TRUE.
- **Combined.** Asserted, if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

Status. Bitfield indicating general comms./hardware error conditions.

- **Config.** TRUE, when *Alarms.Config* is asserted. Indicates Data recording configuration failure.
- **Update.** TRUE, when *Alarms.Update* is asserted. Indicates configured update rate has not been obtained.

- **Archive.** TRUE, when *Alarms.Archive* is asserted. Indicates a .uhh file was deleted before it was archived.
- **LostData.** TRUE, when *Alarms.LostData* is asserted. Indicates an internal queue overflow causing the loss of recorded data.
- **MemHW.** TRUE, when *Alarms.MemHW* is asserted. Indicates the recording memory device has exceeded the specified usage parameter.
- **GrpAct.** TRUE, when any alarm within the group is active.
- **GrpUnack.** TRUE, when any alarm within the group is unacknowledged.
- **Recording.** TRUE, when data recording is in progress.
- **Batching.** TRUE, when the current recording was initiated automatically by the start of a batch. Recording will also stop automatically when the batch terminates.

Reset. Digital inputs used to reset errors conditions.

- **Update.** Set TRUE to clear *Alarms.Update* and set *Status.Update* FALSE.
- **Archive.** Set TRUE to clear *Alarms.Archive* and set *Status.Archive* FALSE.
- **LostData.** Set TRUE to clear *Alarms.LostData* and set *Status.LostData* FALSE.

ActUpdat. Shows the currently achieved update period, and should correspond to the selected recording period, *UpdateA* or *UpdateB*, derived from *SelectB*. If insufficient CPU time was available for recording this period will not correspond to the selected recording period and will set *Status.Update* and *Alarms.Update* TRUE.

Duration. Shows a current estimate of the recording time, in hours, that this group is able to maintain on the instrument.

IMPORTANT *To ensure no loss of data, archiving should be set up at a period much lower period than this.*

Options. For future use.

Default. TRUE if this is the default recording group (can only be TRUE in one RGROUP block). When a message is to be recorded for a signed write to a block which is not in a recording group - then the message will be written to the recording group which has the Default field set to TRUE. If no recording group has Default set to TRUE, then such a message will be written to ALL recording groups.

CHAPTER 18 S6000 FUNCTION BLOCKS

This block is used only with legacy instruments.

The S6000 category of Function Block Templates provides the control strategy with functions for defining the connection from the LIN database to Series-6000 Bisynch instruments, e.g. S6350, S6358, S6360, etc.. These function blocks fall into two groups: the S6000 series 'master' blocks, and the SL6000 series 'slave' blocks.

Refer to the LIN Block Reference Manual HA082375U003 Issue 15 (Vintage) for details.

CHAPTER 19 SELECTOR FUNCTION BLOCKS

The SELECTOR category of Function Block Templates provides the control strategy with functions for signal selection.

SELECT: SELECTOR BLOCK

Block function

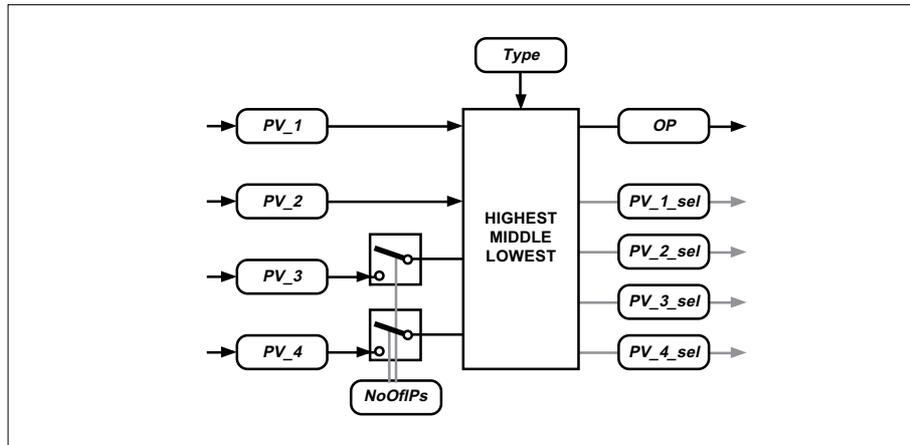


Figure 99 Block schematic

Please refer to *Figure 99*. The SELECT block has four input connections and can be configured to select the highest or lowest value of two, three, or four inputs, or the middle value of three inputs. The input selected is indicated by a digital output and its value is retransmitted as an analogue output *OP*.

Block parameters

Symbols used in *Table 200* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Type	Specifies block function	Menu	
NoOfIPs	Specifies number of inputs (2 - 4)	Integer	
PV_1 to PV_4	Inputs 1 to 4, respectively	Eng	
Alarms			
Software	Block RAM data sumcheck error/network fail	T/F	
Combined	OR-ing of all Alarms bits	T/F	
OP	Selected input value	Eng	
PV_1_sel to PV_4_sel	PV_1 to PV_4, respectively, selected	T/F	
HR_OP, LR_OP	High & low graphics range (PV_n, OP)	Eng	

Table 200 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

Type. (HIGHEST/MIDDLE/LOWEST). Specifies block function.

NoOfIPs. Specifies number of inputs for HIGHEST and LOWEST functions. Automatically defaults to three inputs when the MIDDLE function is selected.

PV_1 to PV_4. Analogue input values 1 to 4, respectively.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.

- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

OP. Current analogue value of selected input.

PV_1_sel to PV_4_sel. Indicate which one of the corresponding inputs *PV_1* to *PV_4*, respectively, is currently selected. TRUE = selected.

HR_OP, LR_OP. High and low range for graphic objects linked to any input *PV_n*, or to the output *OP* (Bar, Trend). *HR_OP* and *LR_OP* define the 100% and 0% displays, respectively.

SWITCH: SWITCH BLOCK

Block function

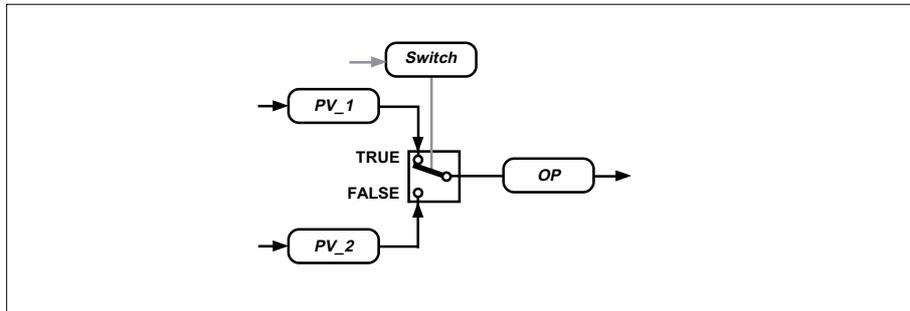


Figure 100 Block schematic

Please refer to *Figure 100*. The SWITCH block acts as a single-pole double-throw switch. When the *Switch* parameter is set to TRUE, the output *OP* transmits *PV_1*. When *Switch* is FALSE, *OP* transmits *PV_2*.

Block parameters

Symbols used in *Table 201* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Switch	Input select	T/F	↔
PV_1	Input 1	Eng	↔
PV_2	Input 2	Eng	↔
HR, LR	High & low graphics range (PV_n, OP)	Eng	↔ ?
Alarms			☐ 📖 🔔
Software	Block RAM data sumcheck error/network failure	T/F	
Combined	OR-ing of all Alarms bits	T/F	
OP	Re-transmitted selected input value	Eng	↔

Table 201 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

Switch. Input select. TRUE switches output *OP* to input *PV_1*, and FALSE switches *OP* to *PV_2*.

PV_1, PV_2. Analogue input values 1 and 2, respectively.

HR, LR. High and low range for graphic objects linked to any input *PV_n*, or to the output *OP* (Bar, Trend). *HR* and *LR* define the 100% and 0% displays, respectively.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.

- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

OP. Current analogue value of selected input. Note that *OP* can be written to (e.g. via the LIN). This allows a latch to be created, by wiring *OP* back into *PV_1* or *PV_2*.

ALC: ALARM COLLECTION BLOCK

Block function

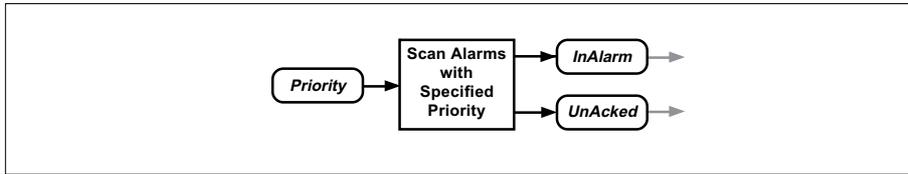


Figure 101 Block schematic

Please refer to *Figure 101*. The Alarm Collection block simplifies alarm monitoring by allowing alarms of the same specified priority to be collected together to generate a single collected alarm logic output.

Block parameters

Symbols used in *Table 202* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Priority	Priority of collected alarms (0-15)	Integer	
InAlarm	Collected current alarm output	T/F	
UnAcked	Collected unacknowledged alarm output	T/F	
Alarms			
Software	Block RAM data sumcheck error/network failure	T/F	
Combined	OR-ing of all Alarms bits	T/F	

Table 202 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

Priority. Specifies priority level of alarms to be collected. A zero priority disables the block.

InAlarm. TRUE indicates a current alarm with the specified priority level.

UnAcked. TRUE indicates an unacknowledged alarm with the specified priority level. For alarms with priority 6 or more, the *UnAcked* flag latches at TRUE until the alarm has been acknowledged, even if it is no longer current. Alarms of priority 5 or less do not need acknowledging, and so for these the *UnAcked* flag remains FALSE.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

2OF3VOTE: BEST-AVERAGE BLOCK

Block function

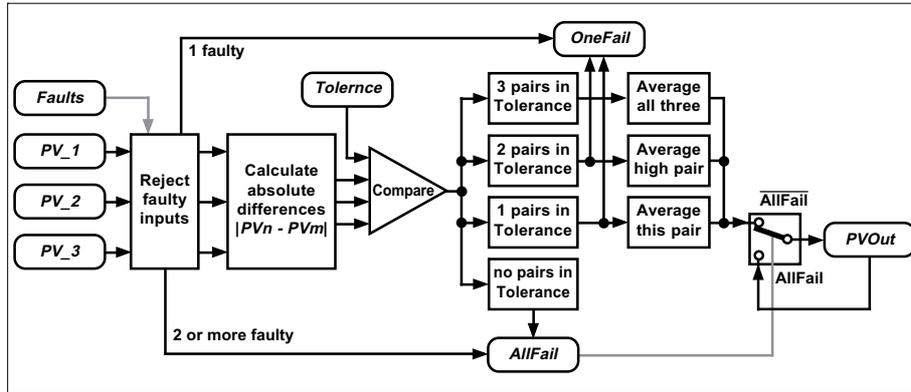


Figure 102 Block schematic

Please refer to *Figure 102*. The 2OF3VOTE block outputs a ‘best’ average (arithmetic mean) value *PVout* derived from three analogue inputs *PV1*, *PV2*, and *PV3*.

If any one of the *PVn* inputs is faulty, as declared by inputs to the *Faults* bitfield, it is rejected and is not used in the average. If two or more *PVn* inputs are faulty, all three are rejected and *PVout* is not recalculated but holds its last good value.

The non-faulty *PVn* inputs are also tested for ‘spread’ by comparing their differences with a user-specified tolerance value (*Tolerance*). The average is formed only from *PVn* values that differ from each other by amounts less than or equal to *Tolerance*. If no pairs of inputs are in-tolerance, all are rejected and *PVout* again holds its last good value.

Appropriate alarm bits set when inputs are faulty or out of tolerance.

Block parameters

Symbols used in *Table 203* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

PV1, PV2, PV3. Analogue inputs from which *PVout* is derived.

HR, LR. High and low range for the ‘Eng’ units parameters (see *Table 203*). Also, high/low range for graphics objects linked to these parameters (Bar, Trend). *HR* and *LR* define the 100% and 0% displays, respectively.

Parameter	Function	Units	Status
PV1	Input 1	Eng	
PV2	Input 2	Eng	
PV3	Input 3	Eng	
HR, LR	PVn, PVout, Tolerance high & low graphics range	Eng	
Faults			
PV1fault	Fault on PV1	T/F	
PV2fault	Fault on PV2	T/F	
PV3fault	Fault on PV3	T/F	
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
AllFail	All inputs rejected	T/F	
OneFail	One input rejected	T/F	
Combined	OR-ing of all Alarms bits	T/F	
Tolerance	Drift tolerance	Eng	
PVout	Resultant PV	Eng	

Table 203 Block parameters

Faults. 3-bit bitfield defining the existence of faults on each of the inputs $PV1$, $PV2$, $PV3$. A TRUE PVn fault bit signifies a ‘fault’ on input PVn . A faulty input is not used in the derivation of $PVout$.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.
- **AllFail.** This alarm is flagged if two or more of the PVn inputs are defined as faulty (by the *Faults* parameter), or if no pair of PVn values differs by an amount less than or equal to *Tolerance*. A TRUE *AllFail* alarm prevents $PVout$ from being recalculated, holding it at its last good value.
- **OneFail.** This alarm is flagged if only one of the PVn inputs is defined as faulty (by the *Faults* parameter), or if only one or two pairs of PVn values differ by an amount less than or equal to *Tolerance*.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

Tolerance. Specifies the maximum amount by which pairs of PVn inputs may differ before they are regarded as being ‘out of tolerance’. The number of pairs of non-faulty inputs that are in-tolerance determines how, or if, $PVout$ is calculated. See *Figure 102*, and the $PVout$ section next.

PVout. This is the ‘best’ average derived from the three inputs, $PV1$ to $PV3$.

Subject to the rejection of faulty input(s), the block calculates the absolute differences between the three possible input pairs as $|PV1-PV2|$, $|PV1-PV3|$, and $|PV2-PV3|$. These values are compared with *Tolerance* to determine how many pairs are in-tolerance, and $PVout$ is then calculated accordingly. *Table 204* shows how this works, gives simple examples, and indicates the alarm bits that are set.

Pairs in-tolerance	Alarms	PVout average of:	Example (Tolerance = 2.0)			
			PV1	PV2	PV3	PVout
3	(None)	PV1, PV2, PV3	<u>7.0</u> *	<u>6.0</u>	<u>5.0</u>	6.0
2	OneFail	Higher-valued pair only	4.0	<u>6.0</u>	<u>8.0</u>	7.0
1	OneFail	This pair only	<u>4.0</u>	<u>6.0</u>	9.0	5.0
0	AllFail	(not re-calculated)	3.0	6.0	9.0	(Held)

*Only underlined values used to form average

Table 204 Derivation of Pvout

TAG: TAG BLOCK

Block function

The TAG block provides a character string that normally appears in the T600 front-panel tag display when the user task containing the block is occupying the main loop display. There should be one TAG block per user task. If a TAG block does not exist in the user task currently active on the front panel, the tag display is generated according to the rules given in the *T640 User Guide* (Part no. HA082468U005), under *Operator Displays & Controls*.

Block parameters

Symbols used in *Table 205* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
SelTag			
Tag1	TRUE selects Tag1 field (lowest priority)	T/F	
Tag2	TRUE selects Tag2 field	T/F	
Tag3	TRUE selects Tag3 field	T/F	
Tag4	TRUE selects Tag4 field	T/F	
Tag5	TRUE selects Tag5 field	T/F	
Tag6	TRUE selects Tag6 field	T/F	
Tag7	TRUE selects Tag7 field	T/F	
Tag8	TRUE selects Tag8 field (highest priority)	T/F	
Tag1 to Tag8	8-character text strings or field names	Alphanumeric	
TAG	Actual tag display when loop on front panel	Alphanumeric	
Alarms			
Software	Block RAM data sumcheck error / network failure	T/F	
Combined	OR-ing of all Alarms bits	T/F	
Block1 to Block8	Block references, as sources of Tag1-Tag8, resp.	Name	
Options			
NoMsg	Suppress normal front-panel messages	T/F	
NoInval	Suppress the 'INVALID' front-panel message	T/F	

Table 205 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

SelTag. An 8-bit bitfield selecting which of the *Tag1* to *Tag8* parameters are possible sources for the T600 front-panel tag display. If no bits are TRUE, the TAG block name becomes the default tag for this user task. If more than one bit is TRUE, the most significant bit selects the corresponding *Tagn* field as the source of tag for this user task.

Tag1 to Tag8. These are 8-character strings providing eight possible tags for the tag display.

TAG. Always displays the current tag for the loop, i.e. what actually appears in the T600 front panel’s tag display when this user task occupies the main loop display.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

Block1 to Block8. These fields refer to other blocks, or even this TAG block, by name (*Block* parameter). If a *Blockn* fields contains a valid block reference, the corresponding *Tagn* field is taken to be a field name, and the contents of the referenced block/field is displayed as the tag for this user task. If the field name is invalid, ‘**BadField**’ is displayed in the tag.

Options.

- **NoMsg.** TRUE disables all standard tag-display messages, thereby retaining the tag at all times except during database inspect mode.
- **NoInval.** TRUE suppresses the ‘INVALID’ message, useful when the strategy uses normally-invalid key-combinations to generate special functions.

CHAPTER 20 TAN FUNCTION BLOCKS

This block is used only with legacy instruments.

The TAN category of Function Block Templates provides the control strategy with functions for controlling and monitoring the communications via an RS485 TCS Asynchronous Network (TAN).

Refer to the LIN Block Reference Manual HA082375U003 Issue 15 (Vintage) for details.

CHAPTER 21 TIMING FUNCTION BLOCKS

The TIMING category of Function Block Templates provides the control strategy with functions for timing and simple Programmer operations.

SEQ: SEQUENCE BLOCK

Block function

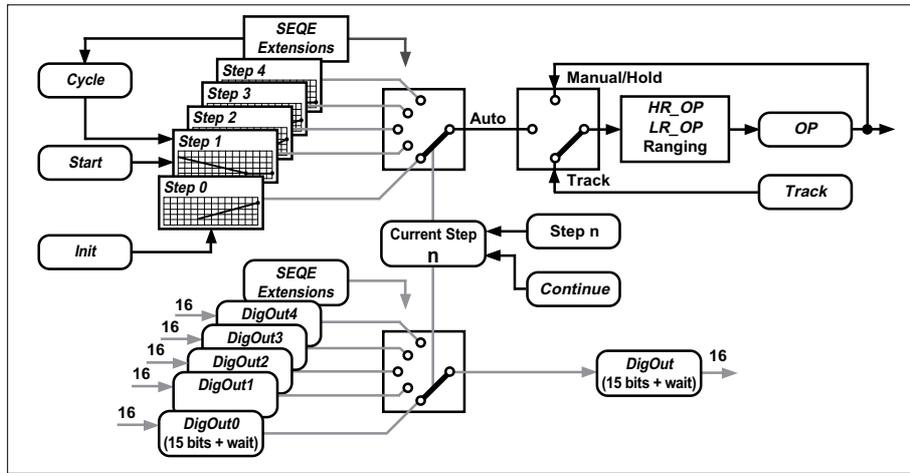


Figure 103 Block schematic

Please refer to *Figure 103*. This block lets you profile analogue setpoints and/or time-sequence 15 digital output channels as a series of four *steps*, plus an initialisation step. More steps can be added to the sequence in multiples of eight using Sequence Extension blocks (SEQE). SEQE blocks are controlled from the main SEQ block, and contain the parameter database for each step. Sequences can be cycled, interlinked and cascaded in series.

Each step defines a setpoint slope, end value (target) and dwell time, together with the status of the 15 digital outputs for that step. The digitals can also be written to (via the *DigOut_n* fields) from the runtime screen or via connections from the control strategy.

The Sequence block has Auto/Manual/Track/Hold control with fallback. Sequence interlocks control the execution of the sequence allowing a conditional Wait and Continue at each step. The whole sequence can be re-cycled automatically or conditionally on detection of an input.

Operating Modes. Execution of the sequence occurs only when the block is operating in Automatic mode. In Manual, Track or Hold modes the ramp or timing functions are interrupted, allowing derivation of the output value *OP* from other sources. In this way, *OP* can be moved away from the value set by the sequence.

However, these actions merely defer the real-time duration of the sequence, since ramp rates are defined as a slope and target value in each step, and the dwell timer is frozen. Normal execution of the sequence resumes at the ‘deferred’ time as soon as the operating mode is switched back to Automatic.

Priority 1.

Hold Mode. Hold mode is selected from the input of *SelHold*. In Hold mode, *OP* is locked and cannot be changed.

Priority 2.

Track Mode. Track mode is selected from the input of *SelTrack*. In Track mode *OP* is controlled by *Track*.

Priority 3.

Auto/Manual Mode. In Automatic mode *OP* is controlled by the sequence function. In Manual mode *OP* can be adjusted independently.

Block parameters

Symbols used in Table 206 are explained in Table 1. Additional parameter information is given in the Block specification menu section following.

Parameter	Function	Units	Status
Mode	Current Operating Mode	Menu	
FallBack	Suppressed Operating Mode	Menu	
MaxSteps	Number of Last Step in Sequence	Integer	
CurrStep	Currently Active Step	Integer	
OP	Block Output	Eng	☐➔
HR_OP	Output High Range	Eng	➔☐➔
LR_OP	Output Low Range	Eng	➔☐➔
Slope	Ramp Rate of Current Step		☐📖
Target	Target Value of Current Step	Eng	☐📖
Slope_0	Ramp Rate in Step 0		
Slope_1 to Slope_4	Ramp Rates in Steps 1 to 4, resp.		➔☐
StartVal	Target Value in Step 0	Eng	
EndVal_1 to EndVal_4	Target Values in Steps 1 to 4, resp.	Eng	➔☐
Alarms			☐➔📖🔊
Software	Data Corruption/Communications Fault	T/F	
Combined	OR-ing of all Alarms bits	T/F	
Extend	Name of Extension Block	Alphanumeric	
TimeBase	Time Units for Slope, Dwell Parameters	Menu	
Track	Defines Output (OP) in Track Mode	Eng	➔☐
StatIn	Input Sequence Status	Bitfield	➔☐
Init	Initialise Sequence (Force Return to Step 0)	T/F	☐🔧
Start	Start Sequence (Step 0 to Step 1)	T/F	☐🔧
Continue	Continue after Wait Flag	T/F	☐
Cycle	Continuously Cycles Sequence	T/F	☐
Track_0	Selects Track Mode at Step 0	T/F	☐
SelHold	Hold Mode Select	T/F	☐
SelTrack	Track Mode Select	T/F	☐
			0 0 0 0 0 0 0 0
StatOut	Output Sequence Status	Bitfield	☐➔📖
Ramping	Step Ramping	T/F	☐
StepDone	End of Step (Waiting) or Sequence	T/F	☐
CycDone	Sequence Finished (Not Cycling)	T/F	☐
NotTrack	Track Mode Not Active Flag	T/F	☐
			0 0 0 0 0 0 0 0
TimeLeft	Remaining Dwell Time in Current Step	Eng	☐➔📖
DigOut	Digital Outputs & Wait Field	ABCD hex	
Bit0	Bit 0 Digital Output	T/F	☐ 1
Bit1	Bit 1 Digital Output	T/F	☐ 2
Bit2	Bit 2 Digital Output	T/F	☐ 4
Bit3	Bit 3 Digital Output	T/F	☐ 8
			D
Bit4	Bit 4 Digital Output	T/F	☐ 1
Bit5	Bit 5 Digital Output	T/F	☐ 2
Bit6	Bit 6 Digital Output	T/F	☐ 4
Bit7	Bit 7 Digital Output	T/F	☐ 8
			C
Bit8	Bit 8 Digital Output	T/F	☐ 1
Bit9	Bit 9 Digital Output	T/F	☐ 2
BitA	Bit A Digital Output	T/F	☐ 4
BitB	Bit B Digital Output	T/F	☐ 8
			B
BitC	Bit C Digital Output	T/F	☐ 1
BitD	Bit D Digital Output	T/F	☐ 2
BitE	Bit E Digital Output	T/F	☐ 4
Wait	Wait Field State	T/F	☐ 8
			A
DigOut_n (n = 0 to 4)	Step n Digital Outputs & Wait Field	ABCD hex	☐➔
Bit0	Bit 0 Digital Output	T/F	☐ 1
Bit1	Bit 1 Digital Output	T/F	☐ 2
Bit2	Bit 2 Digital Output	T/F	☐ 4
Bit3	Bit 3 Digital Output	T/F	☐ 8
			D

Continued...

Parameter	Function	Units	Status
<i>Continued...</i>			
Bit4	Bit 4 Digital Output	T/F	C
Bit5	Bit 5 Digital Output	T/F	
Bit6	Bit 6 Digital Output	T/F	
Bit7	Bit 7 Digital Output	T/F	
Bit8	Bit 8 Digital Output	T/F	B
Bit9	Bit 9 Digital Output	T/F	
BitA	Bit A Digital Output	T/F	
BitB	Bit B Digital Output	T/F	
BitC	Bit C Digital Output	T/F	A
BitD	Bit D Digital Output	T/F	
BitE	Bit E Digital Output	T/F	
Wait	Wait Field	T/F	
Dwell_n (n = 1 to 4)	Dwell Time for Step n	Eng	☐

Table 206 Block parameters

Block specification menu

Dbase, Block, Type. See Appendix D page 544 for details of these ‘header’ fields.

Mode. (AUTO/MANUAL/TRACK/HOLD) Current operating mode.

Fallback. (AUTO/MANUAL) Indicates next (suppressed) mode.

MaxSteps. Total number of steps in sequence (excluding the initialisation step 0). Sequences with more than four steps need SEQE Extension blocks, which add steps in multiples of eight. MaxSteps can be any number, not necessarily a multiple of 4 or 8.

CurrStep. Current Step. Indicates the step currently active in the sequence. This number will not correspond with the step number used in the Sequence Extension block.

OP. Output value. The analogue output from the Sequence block. Often called the ‘Setpoint’ since the output from a Sequence block is frequently used as a Remote Setpoint to a controller. Parameter *OP* controls the output in Manual mode only. In Auto and Track modes, *OP* is controlled by the sequence and Track inputs respectively. In Hold mode, *OP* is locked.

HR_OP, LR_OP. *OP* High & Low Graphics Ranges, respectively. These parameters are also used to define the value of a step slope in the special case when *Slope_n* is set to zero. (See *Slope_n*.)

Slope. Slope of current step (absolute value).

Target. End value of current step.

Slope_0. Ramp rate in step 0. This parameter is used when re-initialising a sequence. It defines the ramp rate (absolute value) to the target specified by *StartVal*. Setting *Slope_0* equal to 0.0 causes *OP* to jump immediately (i.e. in one scan time) to the *StartVal* value.

StartVal. Target value in step 0, used when a sequence is re-initialised.

Slope_n. Ramp Rate. Defines the ramp rate to the target specified by *EndVal*, where *n* represents the block step number. Ramp rates are defined as gradients which can have either positive or negative slopes, depending on the position of the target value.

It is therefore unnecessary to allocate a sign (-/+) to this value. The time units are defined in the *TimeBase* field.

Setting *Slope_n* equal to 0.0 causes *OP* to ramp to *EndVal* at a rate given by (*HR_OP* – *LR_OP*) engineering units per *TimeBase* unit. E.g. with the block default values of *HR_OP* = 100.0, *LR_OP* = 0.0, *TimeBase* = Secs, and *Slope_1* = 0.0, the slope of step 1 equals 100 eng. units per second.

NOTE. Avoid setting *Slope_n* to a value so small that arithmetic rounding errors at each block update leave *OP* unchanged, i.e. with an effective gradient of zero. *Example:* with a slope of 0.1/hr and block update rate of 0.1s, *OP* increments (or decrements) by $0.1 \times 0.1/3600 \approx 2.8 \times 10^{-6}$ each scan. If *OP* = 1000, say, the result of the addition is approximately 1000.0000028, which needs at least 10 significant figures to express it. In practice, with the 7 to 8 figures available, *OP* would remain unchanged at 1000.

If very small slopes are unavoidable, do not use a SEQ block. Instead, use a timer to increment *OP* less frequently than the scan rate, by proportionately larger quantities.

EndVal_n. Target Value. This value represents the target value for the step, where *n* is the block step number. This value can be changed even when the step is active.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block's RAM data.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

Extend. Used to enter the name of the SEQE Extension block. An entry in this field automatically links the control and database of the extension block to the main sequence block. Subsequent extension blocks are linked using the *Extend* fields in the preceding block.

TimeBase. (Secs/Mins/Hours/Days). Specifies time units for *Slope* and *Dwell*.

Track. Controls *OP* in Track mode.

StatIn. This bitfield is used to control the sequence via digital inputs from the strategy.

- **Init.** Initialise Sequence. Initialises sequence by forcing a return to step 0 irrespective of the operating mode. The initialisation takes place whenever *Init* is set to TRUE, and on completion, the field automatically returns to FALSE. In Auto mode, the initialisation causes *OP* to ramp towards a value defined by *StartVal* at a rate set by *Slope_0*. The sequence remains in Step 0 until *Start* is set to TRUE.
- **Start.** Starts execution of the sequence by incrementing forward from Step 0 to 1 after initialisation. The *Start* field automatically returns to FALSE. Note that sequence programs which continuously cycle only require to be started after a 'Cold Start' condition. It is also possible to automatically start sequences following a 'Cold Start' by changing the default settings in the following fields during configuration:

<i>CurrStep</i>	=	1 (or any other step)
<i>Start</i>	=	TRUE
<i>Init</i>	=	FALSE

- **Continue.** Used in conjunction with the *Wait* flag which can be set individually for each step in the *DigOut* fields. If the *Wait* flag in a particular step is set to TRUE, the sequence will wait at the end of the respective step until either *Continue* is set to TRUE, or the respective *Wait* flag is set FALSE.

The *StepDone* flag is also set TRUE during the period that the *Wait* function is active.

- **Cycle.** Allows a sequence program to be continuously cycled. Therefore, when *Cycle* is set to TRUE, a sequence automatically returns to Step 1 on completion of the final step (defined in *MaxSteps*). Note that re-initialisation of a cyclic sequence using the *Init* input interrupts the cycle by forcing the sequence program into Step 0.

A sequence program remains in the last step on completion when *Cycle* is set to FALSE. The *CycDone* output is set to TRUE in this condition to indicate that the sequence program has finished. If the *Cycle* input is subsequently set to TRUE, the sequence restarts at Step 1.

- **Track_0.** When this input is TRUE, Track mode is automatically selected when the sequence block enters Step 0, irrespective of the status of *SelTrack*. Consequently, *OP* follows *Track* unless Hold mode is selected. Track mode is also automatically deselected when the sequence program is started (*Start* = TRUE), unless *SelTrack* = TRUE.

The operating mode is not affected in Step 0 when *Track_0* is FALSE.

- **SelHold.** Selects Hold mode.
- **SelTrack.** Selects Track mode.

StatOut. This bitfield indicates the current status of the sequence.

- **Ramping.** Indicates the operating phase of the step. A Ramping phase is indicated by TRUE and a Dwell time phase is indicated by FALSE. Note that operation in Manual, Track, or Hold modes does not affect the state of this field.

- **StepDone.** Used in conjunction with the Wait / Continue facility at the end of each step. *StepDone* is TRUE if the sequence is held by the Wait condition at the end of a step, and returns to FALSE as soon as the sequence continues, or is re-initialised. *StepDone* is also TRUE at the end of the last step in a sequence when *Cycle* is FALSE.

- **CycDone.** Indicates when the sequence has finished and is set TRUE on completion of the last step when *Cycle* is FALSE. The field returns to FALSE as soon as the sequence is re-initialised or *Cycle* is set to TRUE.

- **NotTrack.** FALSE when Track mode is active.

TimeLeft. Countdown timer indicating the remaining dwell time in the current step. Its value freezes when either Hold, Track or Manual modes are active.

DigOut. This bitfield indicates the current digital output status, i.e. the state of each of the block's 15 digital output bits, and of the *Wait* field for the current step. *DigOut* is read-only in HOLD and TRACK modes.

DigOut_n ($n = 0$ to 4). These five bitfields define the digital output states and *Wait* fields for each of the block steps 0 to 4, respectively.

- **Bit 0 - E.** Define the output state of each bit in the respective step.

- **Wait.** Causes the sequence to wait at the end of the step for a condition. When the *Wait* field is set to TRUE, execution of the sequence stops until either *Continue* = TRUE, or *Wait* is set to FALSE. The sequence continues at the next step as soon as one of these conditions is satisfied.

Dwell_n ($n = 1$ to 4). Dwell Time for step n . Specifies the time during which the output *OP* is maintained at the target value in each step. The time units are defined in the *TimeBase* field. Dwell time can be adjusted dynamically but is ignored if the value is changed whilst timing is in progress.

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

PV. Input value to be totalised. (Negative *PV* values are treated as zero.)

CutOff. If $PV < CutOff$ the value in the *Default* parameter is totalised instead of the *PV* input.

Default. Value totalised when the *PV* input falls below *CutOff* value. (Negative *Default* values decrement *Total* and *OP*, and the pulse accumulator.)

Scale. Scales *PV* by making the variable being totalised equal to $PV/Scale$.

Total. Totaliser value in integer format with full 8-character resolution. It can be linked into graphic displays (via the READOUT graphic).

OP. Totaliser value output in 7-digit floating point format.

HR_OP, LR_OP. High & Low range for graphic objects linked to *PV* or *OP* (Bar, Trend). *HR_OP* and *LR_OP* define the 100% and 0% displays, respectively.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

Target. Sets the maximum (integral) value for *Total*. When *Total* is about to exceed the value in *Target* the *Overflow* output is set TRUE and integration continues from a value of zero. Being an integer, *Target* cannot be linked to a bargraph. Note that altering *Target* during operation initialises the Totalisation block and resets *Total*.

Overflow. Overflow flag, set TRUE when the *Total* value ‘rolls over’ after having reached the value specified in *Target*. *Overflow* can be reset via the *ClearOv* and *Init* parameters.

ClearOv. Resets the *Overflow* flag and returns to FALSE automatically.

Disable. Disables totalisation by halting the integration function. The output of pulses remaining in the pulse accumulator is not affected.

Pulse. This output field is driven by pulses stored in the Pulse Accumulator, which decrements as each pulse is output until empty. For every stored pulse, *Pulse* sets TRUE and then resets automatically after two block updates (~0.2 sec). The accumulator itself is incremented (or decremented) whenever *Total* increments (or decrements). (Negative *Default* values can decrement *Total*.)

NOTE. Pulses are output only when the accumulator holds a positive number.

At high *PV* integration rates the relatively slow pulses cannot keep pace, but none are missed - they accumulate and are eventually output. In this way the pulse total can ‘catch up’ in quieter periods. Although the accumulator ensures that no pulses are lost within the block itself, they may be lost when transmitted over the LIN if the scan times of the systems involved are different. It may be necessary to slow down the pulse rate by increasing the *Scale* parameter value.

The pulse output accumulator is re-initialised by *Init*, and whenever the *Timebase* parameter is written to.

Init. Initialises the *Total* and *OP* values to zero, clears the pulse accumulator, and resets *Overflow* to FALSE.

Timebase. (Secs/Mins/Hours/Days) Specifies time units of *PV* and of the integration function.

DTIME: DEADTIME BLOCK

Block function

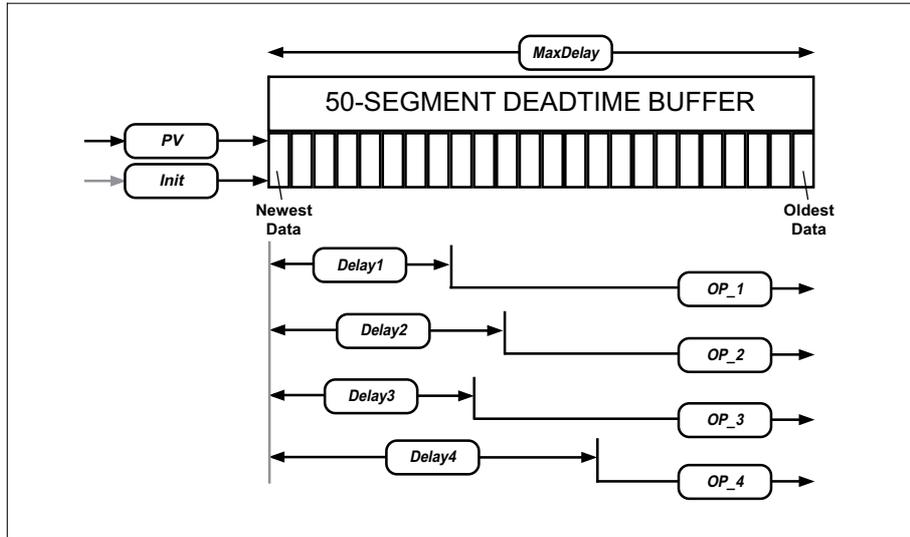


Figure 105 Block schematic

Please refer to *Figure 105*. The Deadtime block introduces time delays into the measurement signal *PV*. The block buffer contains 50 storage locations and 4 pointers for output extraction. At each sample update, a new measurement is inserted and the oldest measurement is discarded. The buffer sample rate is calculated as $MaxDelay/50$, where *MaxDelay* is the maximum delay time in seconds.

Each pointer can extract an output at any delay time up to *MaxDelay*. When the required point lies between two buffer samples, the output is estimated by linear interpolation between the samples. *Figure 106* shows as an example the delayed outputs from a rectangular *PV* pulse input. The amount of ‘slowing’ of the output signals, due to the averaging action between buffer points, depends on the value of *MaxDelay*.

The block is initialised via the *Init* parameter, which sets all elements equal to the current value of *PV*.

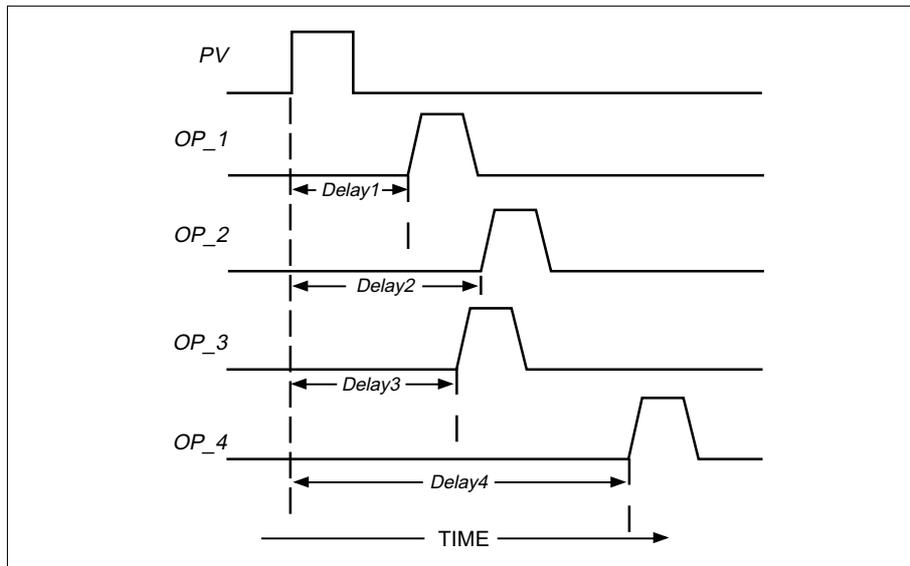


Figure 106 DTIME block output (example)

Block parameters

Symbols used in *Table 209* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
PV	Process Variable (Block Input)	Eng	
OP_1	Output Value at Delay 1	Eng	
OP_2	Output Value at Delay 2	Eng	
OP_3	Output Value at Delay 3	Eng	
OP_4	Output Value at Delay 4	Eng	
HR, LR	High & Low Graphics Range (PV, O/Ps)	Eng	
Alarms			
Software	Data Corruption/Communications Fault	T/F	
Combined	OR-ing of all Alarms bits	T/F	
Delay1	Delay 1 Time	Eng	
Delay2	Delay 2 Time	Eng	
Delay3	Delay 3 Time	Eng	
Delay4	Delay 4 Time	Eng	
MaxDelay	Maximum Delay Time	Eng	
Init	Initialise Delay Buffer	T/F	

Table 209 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these 'header' fields.

PV. Input value.

OP_1 to OP_4. Output values at *Delay1* to *Delay4*, respectively. (The outputs are modified by the effects of linear interpolation, to an extent determined by the value of *MaxDelay*.)

HR, LR. High & Low range for graphic objects linked to *PV* or *OP_1* to *OP_4* (Bar, Trend). *HR* and *LR* define the 100% and 0% displays, respectively.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

■ **Software.** Sumcheck error in block's RAM data.

■ **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

Delay1 to Delay4. Specify time periods (seconds) of delay 1 to delay 4, respectively. Note that delay values cannot be entered that exceed the value of *MaxDelay*. Also, entering a *MaxDelay* value that is less than existing delays automatically reduces them to the new *MaxDelay* value.

MaxDelay. Specifies maximum delay time (i.e. length of the delay buffer in seconds). Entered *MaxDelay* values are automatically rounded up to the next multiple of 5 seconds.

Init. Initialises buffer by setting all elements equal to the current value of *PV*.

TIMER: TIMER BLOCK

Block function

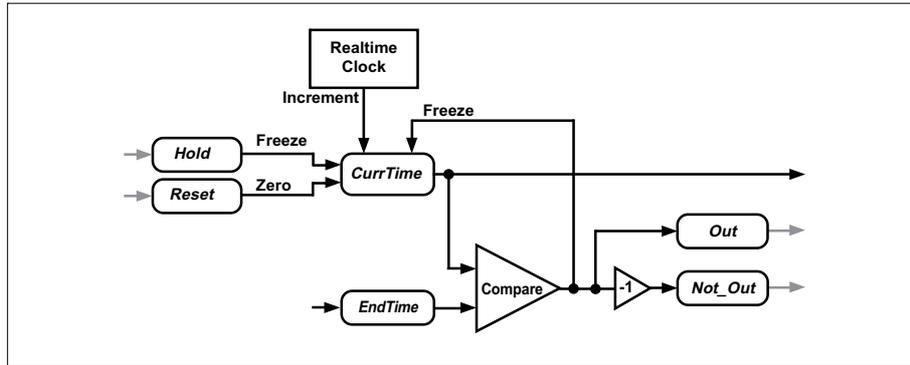


Figure 107 Block schematic

Please refer to *Figure 107*. The TIMER block is a resettable incremental ‘up’ timer, continuously outputting the current time and setting digital flags on reaching a preset end time value, when the timing stops. A *Hold* digital input allows the timer to be ‘frozen’. Seconds, minutes, or hours may be selected via the *Timebase* parameter.

Timing Precision. The precision of the current time output depends on the rate at which the block is being updated, i.e. the scan time. At best, this is every 0.1 seconds, but for a medium-sized strategy (about 75 to 100 blocks) scan times are nearer 0.2 to 0.3 seconds.

Block parameters

Symbols used in *Table 210* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
EndTime	End time value (in Timebase units)	Eng	<input type="checkbox"/>
Reset	Resets timer (CurrTime) to zero	T/F	<input type="checkbox"/>
Hold	Freezes timer (CurrTime)	T/F	<input type="checkbox"/>
Timebase	Time units for EndTime and CurrTime	Menu	
Alarms			<input type="checkbox"/>
Software	Data Corruption/Communications Fault	T/F	
Combined	OR-ing of all Alarms bits	T/F	
CurrTime	Current timer value (in Timebase units)	Eng	<input type="checkbox"/>
Out	End time reached flag	T/F	<input type="checkbox"/>
NotOut	End time not reached flag (= NOT-Out)	T/F	<input type="checkbox"/>

Table 210 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

EndTime. The time value at which the timer stops. Its time units are specified by the *Timebase* parameter. If *EndTime* is set to a value that exceeds *CurrTime*, with *Reset* and *Hold* FALSE, *CurrTime* increments until it reaches the *EndTime* value. At this point *CurrTime* freezes and the *Out/NotOut* flags are set.

If *EndTime* is set to a value less than *CurrTime* (whatever the states of *Reset* and *Hold*), *CurrTime* immediately adopts this *EndTime* value.

Reset. Setting *Reset* TRUE zeroes *CurrTime*, or makes it equal to *EndTime* if this is negative. Whilst *Reset* is TRUE, *CurrTime* cannot increment.

NOTE *Reset* is not self-resetting.

Hold. Setting *Hold* TRUE freezes *CurrTime*. When *Hold* is reset to FALSE, *CurrTime* continues incrementing where it left off. Whilst *Hold* is TRUE, the state of *Reset* is ignored.

Timebase. (Secs/Mins/Hours) Specifies the value of the timer units (i.e. of *EndTime* and *CurrTime*).

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block's RAM data.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

CurrTime. Current timer value, in units specified by *Timebase*. (See the *EndTime* section above.)

Out. Sets TRUE when the timed period has elapsed, and remains TRUE if endtime is increased, unless a reset is carried out. Specifically, *Out* is FALSE when *EndTime* exceeds *CurrTime*, and TRUE otherwise.

NotOut. Sets FALSE when the timed period has elapsed, i.e. inverse of the *Out* parameter. Specifically, *NotOut* is TRUE when *EndTime* exceeds *CurrTime*, and FALSE otherwise.

TIMEDATE: TIME/DATE EVENT BLOCK

Block function

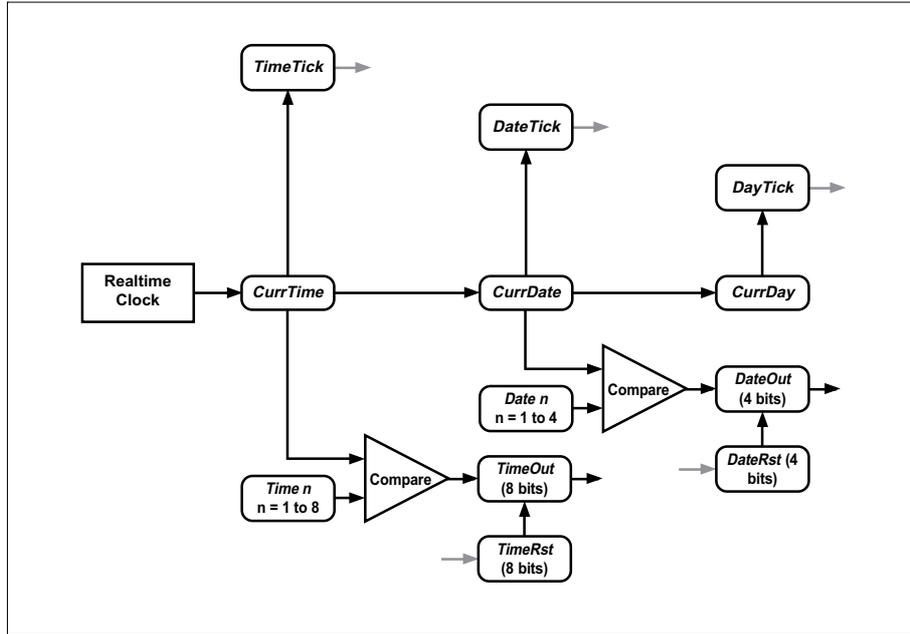


Figure 108 Block schematic

Please refer to *Figure 108*. The Time/Date Event block acts as a clock/calendar, deriving the time, date and day from the Realtme clock of the target instrument. It emits digital pulses (‘ticks’) at each time/date increment, as well as the current weekday value. Weekends are specially indicated. The block also acts as a programmable ‘alarm clock’, setting output bits (*TimeOut* and *DateOut*) when specified times and dates occur. Up to eight different event times and four event dates can be programmed.

The system clock can be reset, as well as monitored, via the TIMEDATE block.

Block parameters

Symbols used in *Table 211* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
CurrTime	Current Time (System Clock)	hh:mm:ss	
TimeTick	Time Ticks (secs, mins, hrs)	Bitfield	
SecsTick	Seconds Tick Output	T/F	
MinsTick	Minutes Tick Output	T/F	
HrsTick	Hours Tick Output	T/F	
0 0 0 0 0 0 0 0			
Time1 to Time8	Event Times	nn:nn:nn	
TimeOut	Time Event Output Bits	Bitfield	
Time1out	Time Event 1 Output Bit	T/F	
Time2out	Time Event 2 Output Bit	T/F	
Time3out	Time Event 3 Output Bit	T/F	
Time4out	Time Event 4 Output Bit	T/F	
Time5out	Time Event 5 Output Bit	T/F	
Time6out	Time Event 6 Output Bit	T/F	
Time7out	Time Event 7 Output Bit	T/F	
Time8out	Time Event 8 Output Bit	T/F	
0 0 0 0 0 0 0 0			

Continued...

Parameter	Function	Units	Status
<i>Continued...</i>			
TimeRst	Time Event Output Bits Resets	Bitfield	
Time1rst	Time Event 1 Output Bit Reset	T/F	
Time2rst	Time Event 2 Output Bit Reset	T/F	
Time3rst	Time Event 3 Output Bit Reset	T/F	
Time4rst	Time Event 4 Output Bit Reset	T/F	
Time5rst	Time Event 5 Output Bit Reset	T/F	
Time6rst	Time Event 6 Output Bit Reset	T/F	
Time7rst	Time Event 7 Output Bit Reset	T/F	
Time8rst	Time Event 8 Output Bit Reset	T/F	
Alarms			
Software Combined	Data Corruption/Communications Fault OR-ing of all Alarms bits	T/F	
CurrDate	Current Date	dd:mm:yy	
CurrDay	Current Day	Menu	
DateTick	Date Ticks (day, week, month, year)	Bitfield	
DayTick	Daily Tick Output	T/F	
WeekTick	Weekly Tick Output	T/F	
MnthTick	Monthly Tick Output	T/F	
YearTick	Yearly Tick Output	T/F	
DayTick	Current Day	Bitfield	
Monday	Day of Week	T/F	
Tuesday		T/F	
Wednesda(y)		T/F	
Thursday		T/F	
Friday		T/F	
Saturday		T/F	
Sunday	Saturday or Sunday	T/F	
Weekend		T/F	
Date1 to Date4	Event Dates	dd:mm:yy	
DateOut	Date Event Output Bits	Bitfield	
Date1out	Date Event 1 Output Bit	T/F	
Date2out	Date Event 2 Output Bit	T/F	
Date3out	Date Event 3 Output Bit	T/F	
Date4out	Date Event 4 Output Bit	T/F	
DateRst	Date Event Output Bits Resets	Bitfield	
Date1rst	Date Event 1 Output Bit Reset	T/F	
Date2rst	Date Event 2 Output Bit Reset	T/F	
Date3rst	Date Event 3 Output Bit Reset	T/F	
Date4rst	Date Event 4 Output Bit Reset	T/F	

Table 211 Block parameters

Block specification menu

Dbase, Block, Type. See Appendix D page 544 for details of these ‘header’ fields.

CurrTime. Current time indicated by the system clock, in hr:min:sec. The system clock can be reset via this parameter.

TimeTick. Bitfield outputting pulses for each time increment (‘ticks’).

- **SecsTick.** Bit sets TRUE for one block update period, then returns to FALSE, when the seconds time-value increments.
- **MinsTick.** Bit sets TRUE for one block update period, then returns to FALSE, when the minutes time-value increments.
- **HrsTick.** Bit sets TRUE for one block update period, then returns to FALSE, when the hours time-value increments.

Time1 to Time8. Eight independent programmable ‘alarm clock’ event times expressed as hr:min:sec. When one of these times is reached, the corresponding *TimeOut* parameter bit is set TRUE.

TimeOut. Bitfield of eight bits, *Time1out* to *Time8out*. A bit sets when the system clock time equals the time specified in the corresponding *Time1* to *Time8* parameter. *TimeOut* bits can be individually reset via the *TimeRst* parameter.

TimeRst. Bitfield of eight bits, *Time1rst* to *Time8rst*. When set from FALSE to TRUE, each bit resets the corresponding *TimeOut* bit (*Time1out* to *Time8out*).

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

CurrDate. Current date indicated by system clock, in dd:mm:yy format. The system calendar can be reset via this parameter.

CurrDay. (MONDAY/ TUESDAY/WEDNESDAY/THURSDAY/FRIDAY/SATURDAY/SUNDAY). Current day, derived from the system calendar.

DateTick. Bitfield outputting pulses for each date increment (‘ticks’).

- **DayTick.** Bit sets TRUE for one block update period, then returns to FALSE, when the day-value increments.
- **WeekTick.** Bit sets TRUE for one block update period, then returns to FALSE, when the week-value increments.
- **MnthTick.** Bit sets TRUE for one block update period, then returns to FALSE, when the month-value increments.
- **YearTick.** Bit sets TRUE for one block update period, then returns to FALSE, when the year-value increments.

DayTick. Bitfield whose eight bits indicate the current day. The first seven bits indicate (TRUE) Monday to Sunday, respectively, and the eighth ‘Weekend’ bit is TRUE on both Saturday and Sunday.

Date1 to Date4. Four independent programmable ‘alarm clock’ event dates expressed as dd:mm:yy. When one of these dates is reached, the corresponding *DateOut* parameter bit is set TRUE.

DateOut. Bitfield of four bits, *Date1out* to *Date4out*. A bit sets when the system calendar date equals the date specified in the corresponding *Date1* to *Date4* parameter. *DateOut* bits can be individually reset via the *DateRst* parameter.

DateRst. Bitfield of four bits, *Date1rst* to *Date4rst*. When set from FALSE to TRUE, each bit resets the corresponding *DateOut* bit (*Date1out* to *Date4out*).

TPO: TIME-PROPORTIONING OUTPUT BLOCK

Block function

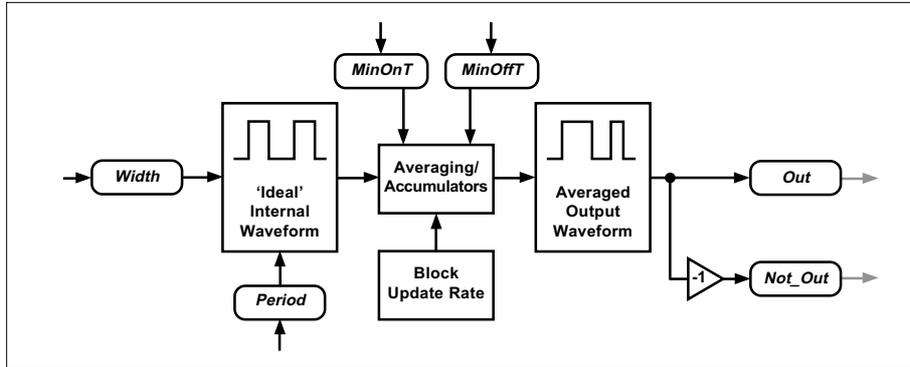


Figure 109 Block schematic

The Time-Proportioning Output block converts an analogue input (*Width*) into a rectangular wave digital output (*Out*), with variable mark/space ratio proportional to the input. A typical application for the TPO block is where a continuous analogue signal is to control power input to a process in which the power can only be ‘on’ or ‘off’, for example, in electrical heating of furnaces.

Please refer to the schematic in *Figure 109*. The block generates an internal ‘ideal’ waveform having a period set by the *Period* input, and a mark/space ratio set by the *Width* input. Specifically, *Width* is the percentage of the period that the output is high (TRUE). See *Figure 110*.

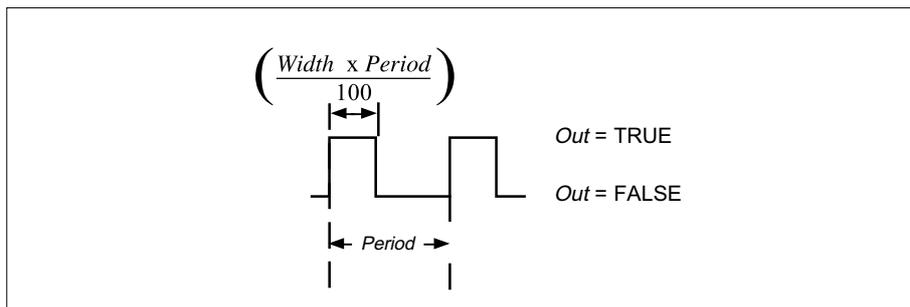


Figure 110 TPO block internal waveform parameters

Minimum ‘On’ and ‘Off’ times. The ideal waveform generated by the block can be subjected to automatic modification before being transmitted via the *Out* connection. This happens if the *MinOnT* or *MinOffT* inputs specify minimum ‘on’ or ‘off’ times that conflict with the calculated ideal times. Minimum on/off times may be required when plant could be damaged or destabilised by very short switching periods. The block algorithm resolves these conflicts by holding the output ‘on’ or ‘off’ for longer than the ideal values, storing the resulting time-errors in accumulators so that they can be applied in a subsequent period, see the *Block specification menu* section below for further details.

Output precision. The ideal waveform is also automatically adjusted whenever the calculated mark/space cannot in practice be output precisely, because the block update rate is large compared with the TPO block period. In these cases the algorithm averages the output over successive cycles. For example, with a *Period* of 1 second, *Width* of 25%, and block update rate of 0.1 seconds, the block outputs alternate cycles of 0.2 and 0.3 seconds (to average 0.25 seconds).

Block parameters

Symbols used in *Table 212* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Width	'On' as percentage of 'Ideal' Period (0-100)	%	▶□
Period	'Ideal' Internal Waveform Period (seconds)	Eng	▶□
MinOnT	Minimum Output 'ON' time (seconds)	Eng	▶□
MinOffT	Minimum Output 'OFF' time (seconds)	Eng	▶□
Alarms			▶□ 📖 🔊
Software	Data Corruption/Communications Fault	T/F	
Combined	OR-ing of all Alarms bits	T/F	
Out	Output Rectangular Waveform	T/F	▶□ 📖
NotOut	Inverse of Out (= NOT-Out)	T/F	▶□ 📖

Table 212 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these 'header' fields.

Width. Percentage of the period that the internal waveform is 'on'. See *Figure 110*. Specifically:

$$\text{'On' time} = \text{Width} \times \text{Period} \div 100 \text{ (seconds)}.$$

Note that the actual block output *Out* may be 'on' for times other than this, owing to automatic averaging effects. Inputs to *Width* should be ranged between 0 and 100; inputs outside this range are truncated.

Period. Period of the internal waveform in seconds. See *Figure 110*. Note that the waveform of the block output *Out* may may have an apparent period other than this, owing to automatic averaging effects.

MinOnT & MinOffT. Specify minimum 'on' and 'off' times, respectively, for the *Out* signal. These may be required to avoid damage to actuators, etc. If the 'on' time of the internal waveform is less than *MinOnT*, the output remains *off* for that period but the unused 'on' time adds to an error accumulator. At the next period, or a subsequent one, when the accumulated error eventually exceeds *MinOnT*, the output goes high for the total error time and the accumulator is reset. *MinOffT* works in a similar way, with the output remaining *on* until the value of the 'off time' error accumulator exceeds *MinOffT*. The nett effect of these adjustments is to increase the apparent period of the output waveform (by an integral multiple) to accommodate *MinOnT* and *MinOffT*, whilst maintaining the mark/space ratio at the correct *average* level.

NOTE. To avoid 'error windup', the accumulators cannot hold errors bigger than the period of the internal waveform. *Errors that build up beyond this level are truncated, resulting in an output waveform having an incorrect average mark/space ratio.* This becomes a possibility when *MinOnT* or *MinOffT* are greater than half the *Period*, but also depends on the value of *Width*. Specifically, truncation can occur if

$$50\% < \text{Width} < (\text{MinOnT}/\text{Period})\%$$

and/or if
$$50\% < (100 - \text{Width}) < (\text{MinOffT}/\text{Period})\%.$$

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block's RAM data.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

Out. The digital waveform actually output by the block. This is a processed version of the 'ideal' internal waveform (see *Figure 110*) having the same average mark/space ratio, but possibly a different apparent period, depending on the values of *MinOnT*, *MinOffT*, and the block update frequency. Please refer to the Note in the section *MinOnT* and *MinOffT*, above.

NotOut. Inverse of the *Out* parameter, i.e. NOT-*Out*.

DELAY: DELAY BLOCK

Block function

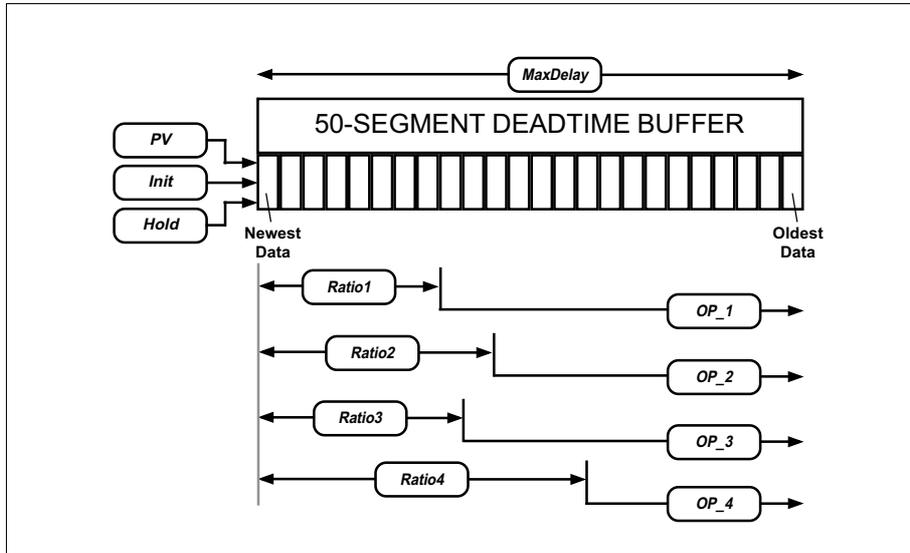


Figure 111 Block schematic

Please refer to *Figure 111*. The DELAY block introduces time delays into the measurement signal *PV*. The block buffer contains 50 storage locations and 4 pointers for output extraction. At each sample update the existing measurements all move one location along the buffer, a new measurement is inserted, and the oldest measurement discarded. The buffer sample rate is calculated as $MaxDelay/50$, where *MaxDelay* is the maximum delay time in seconds.

Each pointer can extract an output at any delay time up to 100% of *MaxDelay*. When the required point lies between two buffer samples, the output is estimated by linear interpolation between the samples.

The block is initialised via the *Init* parameter, which sets all elements equal to the current value of *PV*. While the *Hold* parameter is TRUE the block action is ‘frozen’, i.e. data flow along the buffer halts and so outputs remain unchanged.

Block parameters

Symbols used in *Table 213* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
PV	Process Variable (Block Input)	Eng	
OP_1	Output Value at Delay 1	Eng	
OP_2	Output Value at Delay 2	Eng	
OP_3	Output Value at Delay 3	Eng	
OP_4	Output Value at Delay 4	Eng	
HR, LR	High & Low Graphics Range (PV, O/Ps)	Eng	
Alarms			
Software	Data Corruption/Communications Fault	T/F	
Combined	OR-ing of all Alarms bits	T/F	
Ratio1	Delay 1 Time (% of MaxDelay)	%	
Ratio2	Delay 2 Time (% of MaxDelay)	%	
Ratio3	Delay 3 Time (% of MaxDelay)	%	
Ratio4	Delay 4 Time (% of MaxDelay)	%	
MaxDelay	Maximum Delay Time	Secs	
Init	Initialise Delay Buffer	T/F	
Hold	Freeze Delay Buffer	T/F	

Table 213 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

PV. Input value.

OP_1 to OP_4. Output values at delays of *Ratio1* percent to *Ratio4* percent of *MaxDelay*, respectively.

HR, LR. High & Low range for graphic objects linked to *PV* or *OP_1* to *OP_4* (Bar, Trend). *HR* and *LR* define the 100% and 0% displays, respectively.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block’s highest priority active alarm.

Ratio1 to Ratio4. Specify time periods (seconds) of delays in *OP_1* to *OP_4*, respectively, as percentages of *MaxDelay*. Note that values greater than 100% cannot be entered.

MaxDelay. Specifies maximum delay time (i.e. length of the buffer) in seconds. Note that *MaxDelay* can be altered while the block is running without loss of data. This simply changes the rate at which data flows along the buffer and appears at *OP_1* to *OP_4*.

Init. A TRUE input initialises the buffer by setting all elements equal to the current value of *PV*. *Init* is write-only, i.e. it self-resets to FALSE after use.

Hold. With *Hold* TRUE, movement of data along the buffer is halted and so the *OP_1* to *OP_4* outputs remain constant. Initialisation via *Init* is not affected by the state of *Hold*.

RATE_ALM: RATE ALARM BLOCK

Block function

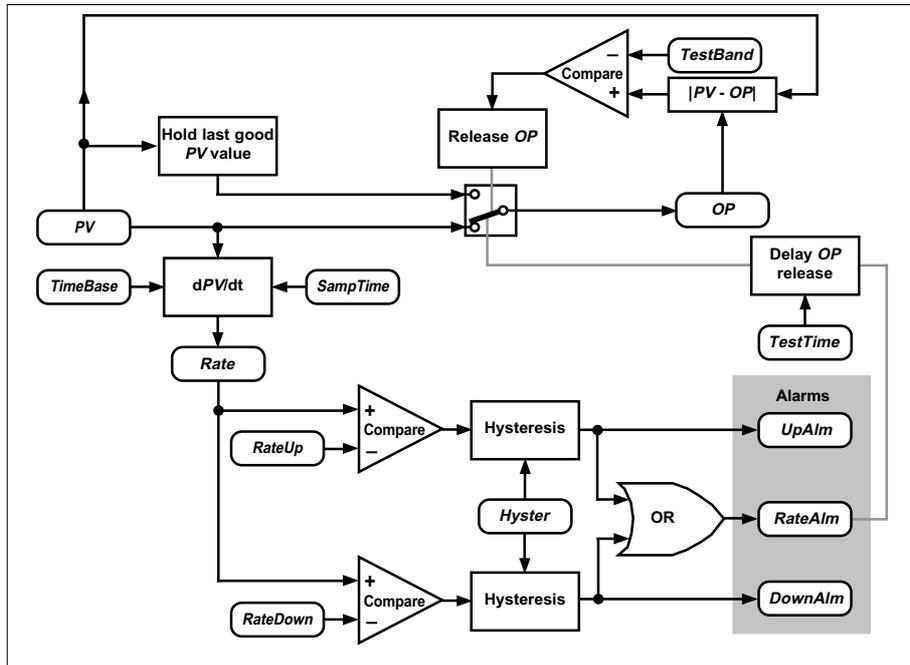


Figure 112 Block schematic

Please refer to the schematic in *Figure 112*. The RATE_ALM block monitors the rate of change of an analogue input signal *PV*, and copies the signal to output *OP*. Should *PV* change at a rate that exceeds user-set limits, *OP* is held at *PV*'s last 'good' (i.e. not in alarm) value and appropriate alarms trip.

The release of *OP* after a rate alarm clears is not immediate but occurs after a specified time period (*TestTime*), or when *PV* returns to within a specified value (*TestBand*) of *OP*, whichever happens sooner. Status bit *OPrelsd* flags *OP*'s release by setting TRUE for one task iteration only. *Figure 113* and *Figure 114* illustrate how *OP*, *OPrelsd*, and the rate alarm bit *RateAlm* behave as *dPV/dt* goes in and out of limits.

Block parameters

Symbols used in *Table 214* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

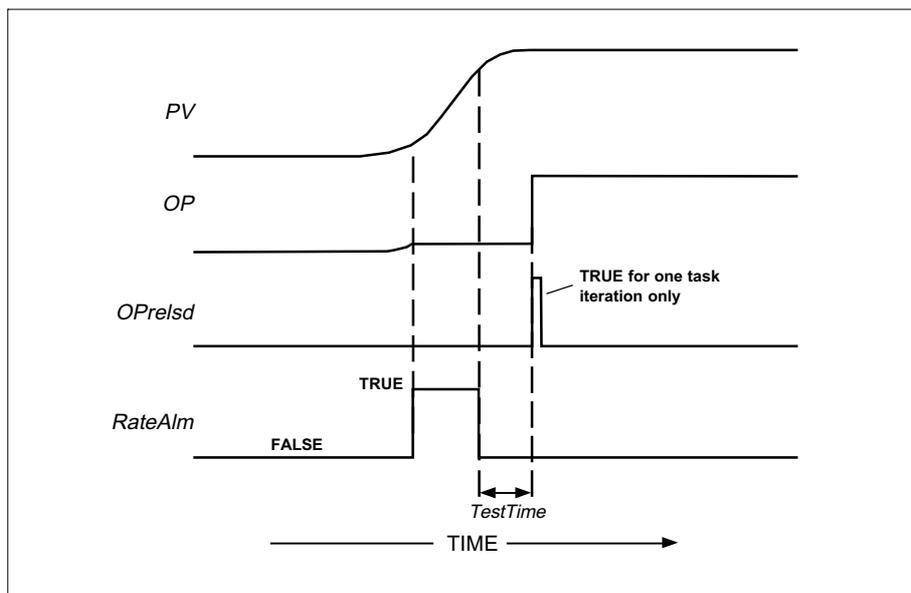


Figure 113 Block parameter actions, OP release after TestTime

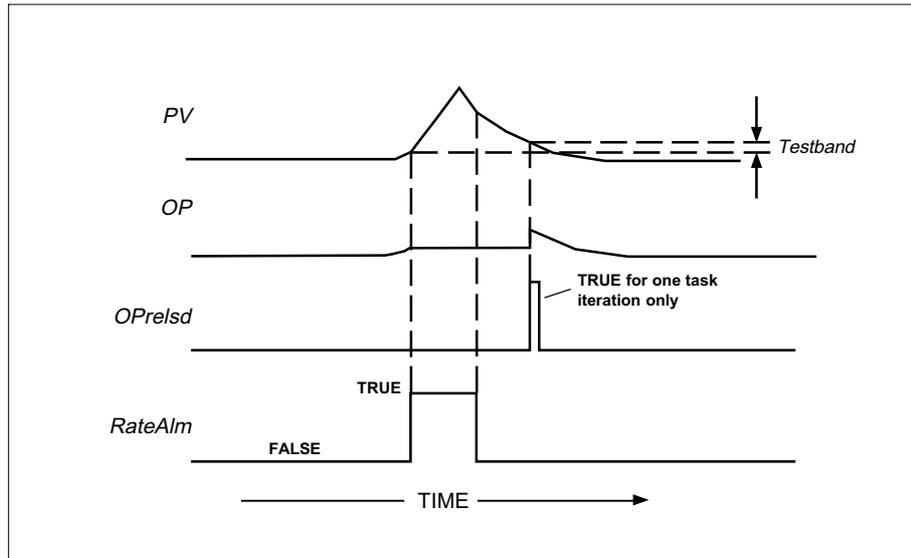


Figure 114 Block parameter actions, OP release controlled by TestBand

Parameter	Function	Units	Status
PV	Value to be rate alarmed	Eng	↔ ↔
OP	Value after rate alarm	Eng	↔ ↔
TestBand	PV error release value	Eng	↔ ↔
HR	High range	Eng	↔ ↔
LR	Low range	Eng	↔ ↔
Alarms			↔ ↔
Software	Block RAM data sumcheck error	T/F	↔ ↔
RateAlm	UpAlm OR DownAlm	T/F	↔ ↔
UpAlm	Rising rate alarm	T/F	↔ ↔
DownAlm	Falling rate alarm	T/F	↔ ↔
Combined	OR-ing of all Alarms bits	T/F	↔ ↔
RateUp	Rising rate alarm value	Eng/Time	↔ ↔
RateDown	Falling rate alarm value	Eng/Time	↔ ↔
Hyster	Rate alarm hysteresis	Eng/Time	↔ ↔
Rate	Actual rate	Eng/Time	↔ ↔
SampTime	PV sample time	Time	↔ ↔
TestTime	OP release time	Time	↔ ↔
TimeBase	Rate calculation time units	Menu	↔ ↔
Status			↔ ↔
OPrelsd	OP released flag (one-shot)	T/F	↔ ↔

Table 214 Block parameters

Block specification menu

Dbase, Block, Type. See Appendix D page 544 for details of these ‘header’ fields.

PV. Input signal. Value to be rate alarmed.

OP. Output signal. Value after being rate alarmed. Normally tracks *PV* unless being held at the last ‘good’ (i.e. not in alarm) *PV*-value during and after a rate alarm.

TestBand. The absolute difference between *PV* and *OP* when *OP* is released from the ‘held’ state, if this occurs before *TestTime* has expired after *RateAlm* has reset FALSE. See Figure 114.

HR, LR. High & Low range for graphic objects linked to *PV* or *OP* (Bar, Trend). *HR* and *LR* define the 100% and 0% displays, respectively.

Alarms. See Appendix D page 545 for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.

- **RateAlm.** *UpAlm* OR *DownAlm*.
- **UpAlm.** Rising rate alarm.
- **DownAlm.** Falling rate alarm
- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.

RateUp. Rising rate alarm limit. If *PV* increases at more than this rate, an *UpAlm* alarm occurs. The alarm clears only when *PV*'s rate of change has fallen below *RateUp* by at least *Hyster* (the alarm hysteresis value).

RateDown. Falling rate alarm limit. If *PV* decreases at more than this rate, a *DownAlm* alarm occurs. The alarm clears only when *PV*'s rate of change has fallen below *RateDown* by at least *Hyster* (the alarm hysteresis value).

Hyster. Specifies the rate alarm hysteresis value, operating only as an alarm state is being cleared. See the *RateUp* and *RateDown* sections above.

Rate. Actual computed rate of change of *PV*, i.e. dPV/dt , which may be positive or negative. *Rate*'s time units are set by the *TimeBase* parameter.

SampTime. *PV* sample time, in units specified by *TimeBase*. The interval between *PV* samples being taken for the dPV/dt calculation. Choose a *SampTime* value that is small enough to follow all significant *PV*-variations, but not so small that zero rates result from the calculation (due to rounding errors).

TestTime. Normal time interval between the clearance of a rate alarm and the release of *OP* from the held state. Units are specified by *TimeBase*. *OP*-release may be earlier than this, depending on the behaviour of *PV* and the value of *TestBand*, see the *TestBand* section above.

TimeBase. (Secs/Mins/Hours). Specifies the time units used throughout the block.

Status. Bitfield reporting the block status.

- **OPrelsd.** This bit becomes TRUE for one task iteration, then resets to FALSE, when *OP* is released from the held state after a rate alarm has occurred. See *Figure 113* and *Figure 114* for illustrations of the action of the *OPrelsd* flag.

RATE_LMT: RATE LIMIT BLOCK

Block function

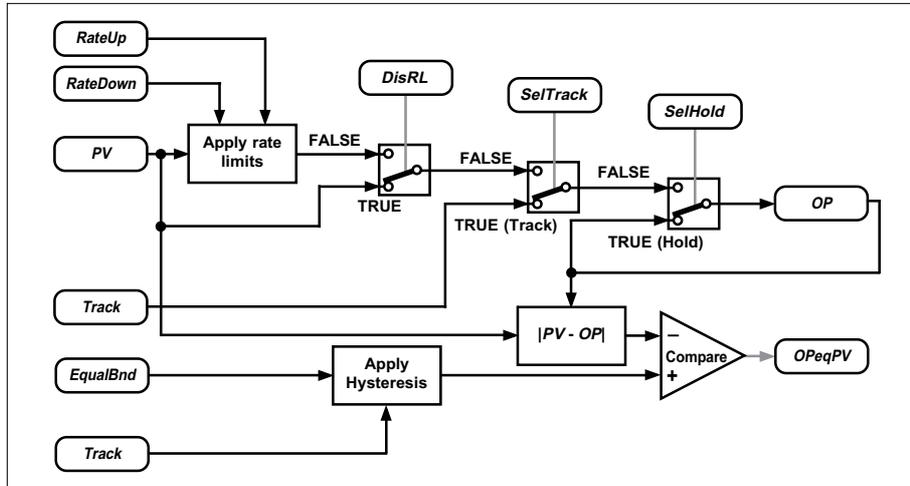


Figure 115 Block schematic

Please refer to the schematic in *Figure 115*. The RATE_LMT block limits the rate of change of an output *OP* as it tries to follow a dynamic ‘target value’ input *PV*. Both *OP*’s increase and decrease rates can be independently specified.

A status bit *OPeqPV* flags when *OP* becomes equal to *PV* to within a tolerance specified by *EqualBnd*, with an extra hysteresis tolerance *Hyst* applied as the state of equality is left. *Figure 116* shows how *EqualBnd* and *Hyst* are applied. Note that the curve must be followed along the arrows and is not valid in the reverse direction.

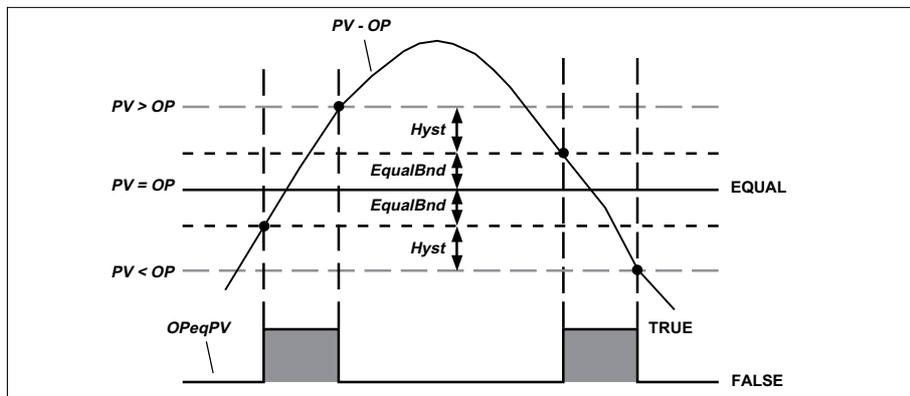


Figure 116 EqualBnd & Hyst action

Block parameters

Symbols used in *Table 215* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
Mode	Operating mode of block	Menu	
FallBack	Fallback operating mode of block	Menu	
PV	Target Value	Eng	
Track	Track Output Value	Eng	
OP	Ramp Output	Eng	
HR	High Range	Eng	
LR	Low Range	Eng	
EqualBnd	Permitted error PV – OP	Eng	
Hyst	Hysteresis for OP = PV		
Alarms			
Software	Block RAM data sumcheck error	T/F	
Combined	OR-ing of all Alarms bits	T/F	
RateUp	Rising rate limit	Eng/Time	
RateDown	Falling rate limit	Eng/Time	
Timebase	Time units used by block	Menu	
Options			
SelTrack	Track Selected	T/F	
SelHold	Hold Selected	T/F	
DisRL	Rate limiting disable (TRUE)	T/F	
Status			
OPeqPV	Ramp done flag	T/F	

Table 215 Block parameters

Block specification menu

Dbase, Block, Type. See *Appendix D page 544* for details of these ‘header’ fields.

Mode, FallBack. (Auto/Track/Hold) Operating and fallback modes, respectively, of block, selected via the corresponding *Options* parameters. The schematic in *Figure 115* shows the priorities of these modes and how they act on *PV* and *OP*. Note that more than one mode can be selected at a time, but only the mode of highest priority can be in operation. The next-highest priority mode selected (or Auto by default) becomes the fallback mode.

- **Auto.** This is the normal operating mode of the block. The block is in Auto mode when neither Track nor Hold has been selected. In Auto mode, *OP* ramps towards *PV* at the specified rate limits, unless *Options/DisRL* is TRUE, when *OP* tracks *PV*. Auto mode has the lowest priority.
- **Track.** In Track mode, *OP* tracks the *Track* parameter value.
- **Hold.** In Hold mode, *OP* maintains the value it had when Hold was selected. Hold mode has the highest priority.

PV. Target value for *OP*.

Track. Value that *OP* tracks when the block is operating in Track mode.

OP. Block output. In the normal rate-limited automatic mode, *OP* ramps towards *PV* at specified rates. See the *Mode, FallBack* section for other modes.

HR, LR. High & Low range for graphic objects linked to *PV* or *OP* (Bar, Trend). *HR* and *LR* define the 100% and 0% displays, respectively.

EqualBnd. Equal band. Specifies a symmetrical ‘tolerance’ band in which *PV* and *OP* are defined as being equal. See *Figure 116*.

Hyst. Hysteresis band. Specifies an asymmetrical ‘tolerance’ band added to the equal band. See *Figure 116*.

Alarms. See *Appendix D page 545* for a general description of the Alarms field.

- **Software.** Sumcheck error in block’s RAM data.

- **Combined.** TRUE if any alarm is active in the block. Adopts the same status message and priority number as the block's highest priority active alarm.
- RateUp.** Rate limit applied to *OP* as it ramps up towards a higher-valued *PV*.
- RateDown.** Rate limit applied to *OP* as it ramps down towards a lower-valued *PV*.
- TimeBase.** (Secs/Mins/Hours). Specifies the time units used throughout the block.
- Options.** Bitfield selecting block operating modes. The mode actually operating is the one with highest priority.
- **SelTrack.** TRUE selects Track mode.
- **SelHold.** TRUE selects Hold mode.
- **DisRL.** TRUE disables rate limiting in Auto mode, and *OP* tracks *PV*.
- Status.** Bitfield reporting block status.
- **OPeqPV.** This bit is TRUE whenever *OP* equals *PV* within the tolerance specified by *EqualBnd* and subject to hysteresis specified by *Hyst*. Otherwise *OPeqPV* is FALSE.

TOT_CON: TOTALISATION CONNECTIONS BLOCK

Block function

This block is intended as a placeholder for 32-bit parameter values, in particular when the values are acquired over a Modbus link using the 2-register 32-bit number facility. Such large numbers are often generated by totalisation processes. It has the same parameter complement as the TOTAL block, but with one extra field, *UserAlrm*. However, the TOT_CON block has virtually no update routine, and most of its fields are general-purpose read/write data stores. The block's 'dummy' parameters can be used to hold totals and associated values for communication with other LIN nodes.

Because of their similar structures, TOT_CON and TOTAL blocks are cache-overlayable, i.e. either block can act as a cached image of the other over the LIN communications. This can be useful in instruments that support TOTAL but not TOT_CON blocks.

Block parameters

Symbols used in *Table 216* are explained in *Table 1*. Additional parameter information is given in the *Block specification menu* section following.

Parameter	Function	Units	Status
PV	32-bit floating-point (general-purpose)	Eng	↔
CutOff	32-bit floating-point (place-holder)	Eng	↔
Default	32-bit floating-point (place-holder)	Eng	↔
Scale	32-bit floating-point (place-holder)	Eng	↔
Total	32-bit unsigned (general-purpose)	Integer	↔
OP	32-bit floating-point (=Total)	Eng	↔
HR_OP, LR_OP	32-bit floating-point (range for OP)	Eng	↔
Alarms			↔  
Software	Data Corruption/Communication Fault	T/F	
Combined	OR-ing of all Alarms bits	T/F	
Target	32-bit unsigned (general-purpose)	Integer	↔
Overflow	Boolean (general-purpose)	T/F	↔
Clear0v	Boolean (general-purpose)	T/F	↔
Disable	Boolean (general-purpose)	T/F	↔
Pulse	Boolean (general-purpose)	T/F	↔
Init	Boolean (general-purpose)	T/F	↔
Timebase	Enumeration (general-purpose)	Menu	
UserAlm	Boolean (TRUE triggers Software alarm)	T/F	↔

Table 216 Block parameters

Block specification menu

The following is given in addition to *Table 216*.

Dbase, Block, Type. See *Appendix D page 544* for details of these 'header' fields.

OP. 32-bit floating-point version of *Total*, written by update routine. Ranged by *HR_OP* and *LR_OP*.

UserAlrm. TRUE trips the block's Software alarm.

NOTE. The *UserAlrm* field can be wired to the *TableOfI* field of the MDBDIAG block so that a SCADA system can detect when the Modbus communications have failed.

TOTAL2: TOTALISATION BLOCK

Block function

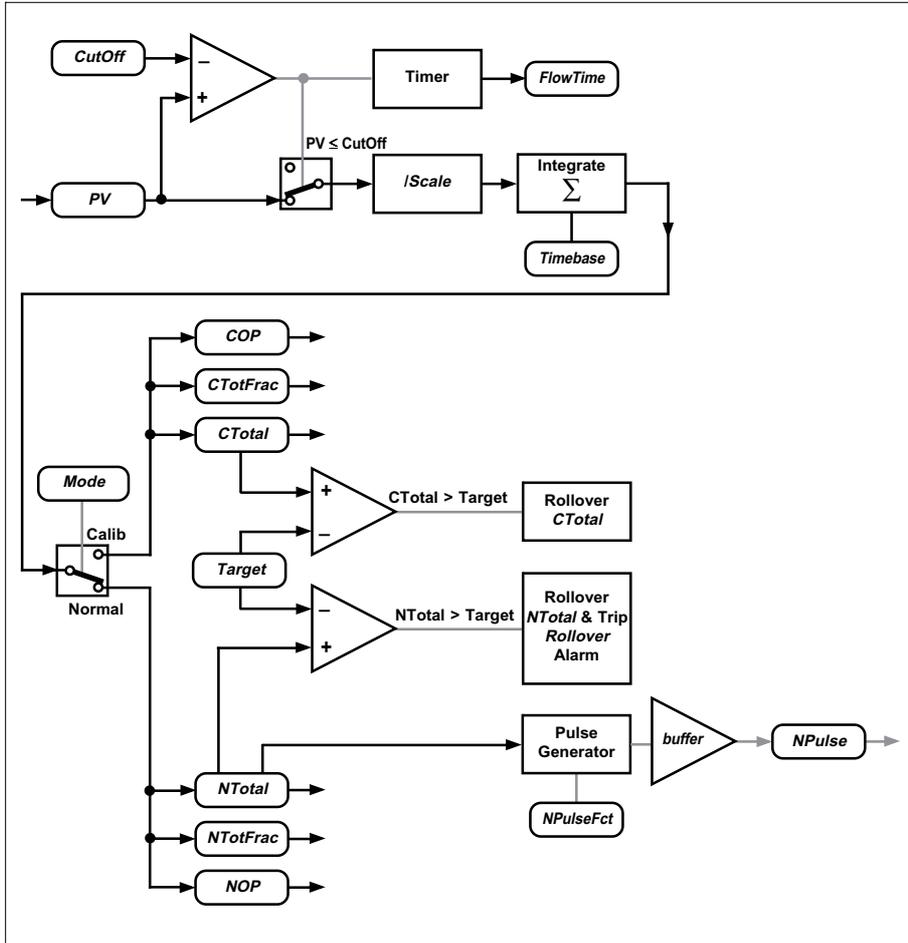


Figure 117 Block schematic

Please refer to *Figure 117*. In ‘Normal’ mode, the TOTAL2 block integrates an analogue input *PV*, storing the integer part of the result in the *NTotal* parameter, the fractional part in *NTotFrac*, and the whole total in floating-point format in *NOP*. These ‘normal’ totals are non-resettable. *Scale* specifies the size of the units being integrated, and *Timebase* specifies the *PV* and integration time units.

The block includes a low threshold *PV* cutoff with zero default totalisation, a flow timer output *FlowTime*, a buffered pulse output for counters, a rollover value (*Target*) and overflow flag (*RollOver*).

In ‘Calib’ (calibration) mode the scaled integrated *PV* is added to *CTotal*, *CTotFrac*, and *COP* instead of to the corresponding ‘normal’ parameters. Entering calibration mode resets these totals, but not the normal totals.

The block provides calibration mode, low flow, and rollover alarms in addition to the usual software and combined alarms.

- **NoFlow.** Trips if *PV* is less than or equal to *CutOff*. No hysteresis is applied.
- **RollOver.** Whenever *NTotal* exceeds *Target* and rolls over, this alarm is set. Note that when *RollOver* is raised the operator can acknowledge the alarm as usual, but it remains set until *CleaRol* is written to (e.g. by a metering technician).
- **Combined.** OR-ing of all Alarms bits.

Target. This integer specifies the value at which *NTotal* and *CTotal* roll over, i.e. reset to zero counts. The default *Target* value is set at 999 999 999. Being an integer, *Target* cannot be linked to a bargraph.

NOTE. Altering the *Target* during operation does not reset any totalisation counts, unless the new *Target* is less than the current total.

NTotal. The integral part of the ‘normal’ total, incrementing only when *Mode* is ‘Normal’. *NTotal* rolls over when it reaches the value specified in *Target*. Note that *NTotal* is a signed long integer with a maximum theoretical value of 2147483647. Being signed, *NTotal* has good connectivity via the LIN to other similar fields, e.g. *I0* to *I7* in ACTION blocks. *NTotal* cannot be written to once the block is configured or while the database is running.

NOTE. If the database is saved, *NTotal* is held as a LIN Database field value. Therefore on cold start the indicated total starts from this saved value, unless the database has never been saved. In this case *NTotal* starts at zero counts.

NTotFrac. Single-precision floating-point number representing the fractional part of the ‘normal’ total (*NTotal*). Increments only when *Mode* is ‘Normal’.

NOP. Floating-point form of *NTotal*. *NOP* can be used to drive trend displays or T3000 Series Process Databases and command scripts that do not support integers. Note that for very large *NTotal* values, *NOP* has less resolution than *NTotal*.

HR_NOP, LR_NOP. High & Low range for graphic objects linked to *NOP* (Bar, Trend). *HR_NOP* and *LR_NOP* define the 100% and 0% displays, respectively.

NPulsFct. Unsigned long integer specifying the number of counts by which *NTotal* must increment to output one ‘pulse’ via the *NPulse* field. E.g. with *NPulsFct* set to 1000 and totalisation in kg, the pulse output will represent the total in tonnes. (See *NPulse* next.)

NPulse. This boolean sets TRUE for one database cycle when *NTotal* has incremented by *NPulsFct* counts. (See *NPulsFct* previous.) If more than one pulse is required per database cycle, the extra pulses are held in a buffer and emitted in ‘quieter’ periods so that none are lost. Note that altering *Timebase* or *Scale* does not affect the number of pulses currently in the buffer. Outstanding pulses continue to be output from the buffer, and any new (possibly differently-scaled) pulses are added to the buffer.

NOTE. The buffer ensures that no pulses are lost within the block itself, but some may be lost when transmitted over the LIN communications if the scan times of the systems involved are different. Although slowing down the pulse rate, by increasing *NPulsFct*, may alleviate the problem, monitoring pulses over the LIN communications is not in general recommended.

CTotal. The integral part of the ‘calibration’ total, incrementing only when *Mode* is ‘Calib’. *CTotal* rolls over when it reaches the value specified in *Target*. *CTotal* cannot be written to once the block is configured or while the database is running. But *CTotal*, *CTotFrac*, and *COP* are zeroed every time *Mode* is set to ‘Calib’.

CTotFrac. Single-precision floating-point number representing the fractional part of the ‘calibration’ total (*CTotal*). Increments only when *Mode* is ‘Calib’. *CTotFrac* is zeroed on entering the calibration mode.

COP. Floating-point form of *CTotal*. Note that for very large *CTotal* values, *COP* has less resolution than *CTotal*.

HR_COP, LR_COP. High & Low range for graphic objects linked to *COP* (Bar, Trend). *HR_COP* and *LR_COP* define the 100% and 0% displays, respectively.

FlowTime. Increments to show the time in seconds since the flow last rose above the *CutOff* value, irrespective of the *Mode* selected. When the measured flow falls to less than or equal to *CutOff*, the timer stops incrementing. As the flow rises again to exceed *CutOff*, *FlowTime* resets and resumes timing.

ClearRol. TRUE resets the *RollOver* alarm, after which *ClearRol* returns to FALSE automatically.

APPENDIX A CONTROL LOOP OPERATING MODES

The PID Control block can operate in one of several different *control modes*, each having its own way of controlling the loop. The characteristics of these modes are described in the PID block section. The present chapter explains how control modes are selected and interact with each other. It also details the 3-term PID control algorithm used in the PID block, and explains the techniques of Integral Balance and Integral Desaturation and their implementation.

THE ACTIVE CONTROL MODE

Control loop modes are selected, but not activated, via the *SelMode* or *Mode* parameter. It is possible to have several modes selected at the same time, but there can only be one active mode. A 'selected' mode is therefore only *potentially* active. The active mode is the selected mode with the highest priority. If the active mode becomes deselected, the selected mode with the next highest priority takes over.

Control mode priorities

The order of mode priority is:

Priority	Mode	Fallback Mode*
1 (Highest)	HOLD	—
2	TRACK	—
3	FORCED MANUAL	MANUAL
4	MANUAL	MANUAL
5	AUTO	AUTO
6	REMOTE	REMOTE
7 (Lowest)	FORCED AUTO	FORCED AUTO

*See next section

Fallback modes

The Fallback mode is the control mode that takes over from the currently active mode if all modes become deselected by the *SelMode* bitfield, i.e. if all the bits go FALSE (*SelMode* = 00000000). This is not necessarily the same as the mode that takes over if the currently active mode is deselected; that is determined by mode priorities.

Each control mode has its own specified Fallback mode (usually itself). Fallback modes are listed in the table above. HOLD and TRACK are the only exceptions - when either of these becomes active the Fallback mode stays at its previous value.

NOTE. If all the *SelMode* bits become zero *simultaneously* the resulting control mode that the loop 'falls back' to is undefined. This is because in reality each bit is reset in a scanning order usually unknown to the user, and so the very last currently active mode, and its Fallback, will be unpredictable.

Selecting control modes

Control modes can be selected/activated either by wiring digital connections to the bits of the *SelMode* parameter from other parts of the control strategy, or by writing directly to these bits, or by 'pressing' pushbuttons on 'controller' graphics faceplates. Wired inputs to *SelMode* always override written inputs, which in turn override pushbutton presses. The pushbuttons interact directly with the *Mode* parameter, and are an alternative to writing directly to *Mode*.

Remote mode

Remote mode is unusual because it cannot become the active mode unless it is enabled (*EnaRem* = TRUE) as well as being the current top priority selected mode (*SelRem* = TRUE). The modes that can be adopted by the loop as determined by these two *SelMode* parameter bits are:

Priority	Mode	Fallback Mode*
TRUE	TRUE	REMOTE (if highest priority)
TRUE	FALSE	FORCED AUTO (if highest priority)
FALSE	TRUE	current mode
FALSE	FALSE	current mode

PID BLOCK 3-TERM CONTROL ALGORITHM

Analogue 3-term control equation

The classical 3-term (PID) control equation, implemented by conventional analogue controllers using operational amplifiers, is usually written as:

$$OP = - \frac{100}{XP} \left[ER + \frac{1}{TI} \int ER dt + TD \frac{dER}{dt} \right]$$

where: OP = controller output
 XP = proportional band
 TI = integral time constant
 TD = derivative time constant
 ER = error (PV-SP).

This equation may be rewritten in the Y(s) terminology of the Laplace transformation:

$$\frac{OP(s)}{ER} = - \frac{100}{XP} \left(1 + \frac{1}{sTI} + sTD \right)$$

Limiting of the high frequency response introduces a digital limit filter, typically chosen to have a time constant equal to a quarter of the derivative time. The complete transfer function is then.

$$\frac{OP(s)}{ER} = - \frac{100}{XP} \left(1 + \frac{1}{sTI} + sTD \right) \left(\frac{1}{1 + sTD/4} \right)$$

Digital control algorithm

In microprocessor-based instruments (e.g. T2550) sampling techniques must be used to calculate the terms of the control equation. It is also more convenient to rewrite the transfer function in terms of difference equations rather than the Y(s) Laplace transform terminology. Thus the 3-term calculated output after n samples is given by:

$$OP_n = - \frac{100}{XP} \left[ER_n + \frac{TS^{r-n}}{TI_{r=1}} \sum ER_r + \frac{TD}{TS} \Delta PV_n \right] + FF$$

where (additionally):

$$\Delta PV_n = \Delta PV_{n-1} + \frac{4TS}{TD} \left(dN - \Delta PV_{n-1} \right)$$

FF = feed-forward term
 OP_n = controller output after n samples
 ER_n = value of error at sample n
 ER_r = value of error at sample r
 ΔPV_n = change in filtered process variable value between samples n and n-1

ΔPV_n is obtained after first-order filtering with an effective time constant TD/4, thus:

$$dN = PV_n - PV_{n-1}$$

where (additionally):

The process variable PV is itself a filtered version of a sampled analogue input value, MV:

$$PV_n = PV_{n-1} + \frac{TF}{IF} \left(MV_n - PV_{n-1} \right)$$

where (additionally):

MV_n = value of analogue input at sample n
 TF = effective first-order time constant
 IF = input channel filter constant.

NOTE. The FF offset is apparent at zero error under proportional-only control action with the integral term disabled. This allows the output to respond to both positive and negative errors, if required.

Equivalence between analogue & digital equations

When the setpoint is constant, the digital algorithm of equation (4) may be written as the following equivalent continuous transfer function:

$$\frac{OP(s)}{ER} = - \frac{100}{XP} \left(1 + \frac{1}{sTI} + \frac{sTD}{1 + sTD/4} \right) \quad (7)$$

This can now be compared with the classical $Y(s)$ version of the analogue controller shown in equation (2). The proportional (P) and integral (I) terms are identical, but the derivative (D) term is slightly modified. This is because the additional first-order filtering is applied to the derivative value DPV, rather than to the error directly.

NOTE. The response to local setpoint (SL) changes is determined by the value of the *IntBalSL* bit in the control block's *Options* parameter, which can be set to disable integral term balancing on SL changes. Similarly, the *IntBalXP* bit determines the response to XP changes.

Integral balance is *automatically* done whenever the loop mode is changed to AUTO, REMOTE, or FORCED AUTO.

INTEGRAL BALANCE & INTEGRAL DESATURATION

These are two calculating techniques, triggered by specific conditions, applied to the *integral term* of the PID algorithm to improve controller output behaviour when these conditions arise. Although the techniques are generally beneficial in a control loop, they do distinctively modify its response characteristics - a point that must be considered when designing and tuning a control system.

Integral balance

Integral balance is applied to prevent abrupt changes - 'bumps' - in the output which might otherwise occur after control mode changes, and step changes in certain control parameters (e.g. setpoint). Output bumps are undesirable because they can damage valves and destabilise the process.

In essence, integral balance works by adjusting the integral accumulator ($TS/TI \cdot \sum ER_i$) in the PID calculation to keep the new output (almost) equal to the value it had before the triggering event. Mathematical details are given in a later section.

Effect of integral balance on dynamics

Figure 118 and Figure 119 show as an example the output (*OP*) and process variable (*PV*) responses to an abrupt setpoint (*SP*) change, without and with integral balance being applied, respectively. In this case applying integral balance markedly reduces the slope and peak value of the *OP* response, whilst at the same time bringing *PV* to the new *SP* just as rapidly but with half the overshoot.

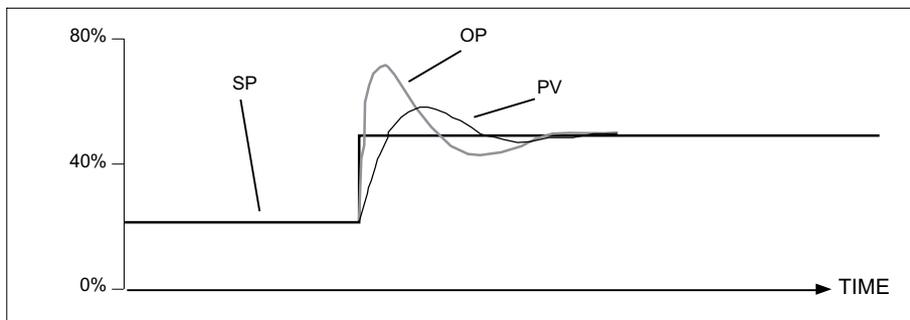


Figure 118 Response to step setpoint change - without integral balance

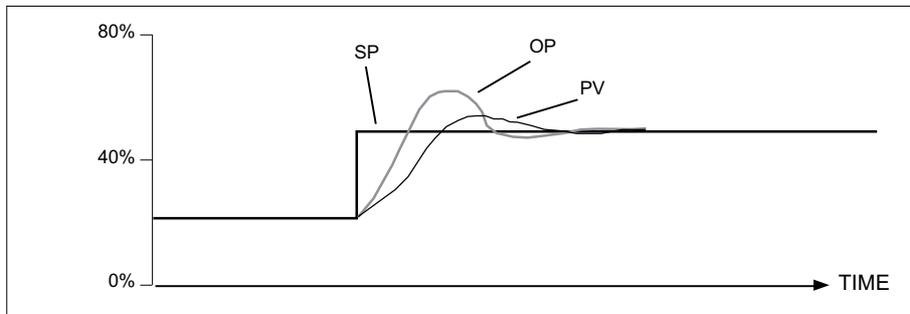


Figure 119 Response to step setpoint change - with integral balance

Clearly, integral balance modifies the loop's response to setpoint changes, and this must be considered when a loop is being tuned using these changes. The response of a well-tuned loop with integral balance generally has less overshoot than one without.

Conditions triggering integral balance application

- Changing the local setpoint (*SL*) in AUTO mode. (Optional, via the control block's *Options.IntBalSL* parameter).
- Changing the proportional band (*XP*) in AUTO or REMOTE. (Optional, via the *Options.IntBalXP* parameter).
- Changing operating mode to any automatic mode (AUTO, REMOTE, FORCED AUTO).
- Inputting a rising edge to the *IntBal* bit of the PID block's *Options* parameter.

Integral balance & adaptive gain control

Integral balance is particularly useful in plants where several sets of tuning constants are needed to cope with different process dynamics at different operating bands (e.g. pH control). Here, integral balance prevents bumps in the output each time the constants are changed at a boundary.

In some control applications the value of the proportional band XP must be varied *continuously* ('adaptive gain'). In these cases *IntBalXP* should be set FALSE to prevent an integral balance being performed at every iteration of the PID algorithm, which would interfere with proper control.

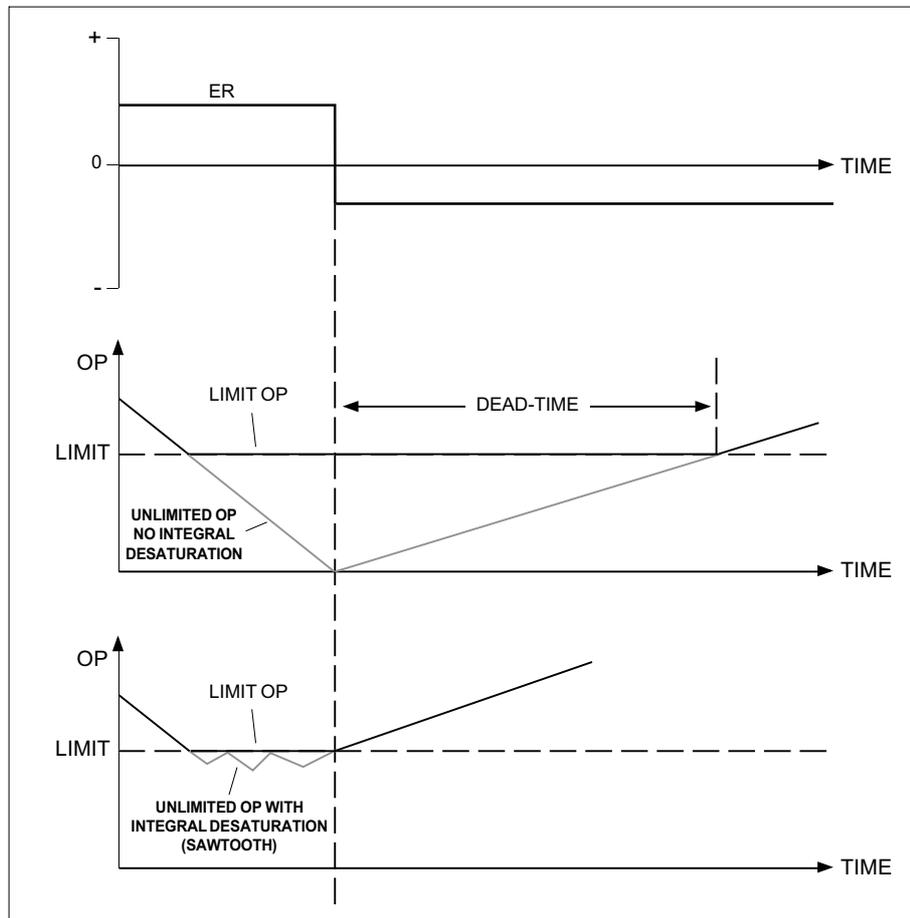


Figure 120 Effect of integral desaturation (integral action)

Integral desaturation

With a persistent error (ER), the PID algorithm's integral term ($TS/TI \cdot \sum ER_i$) can accumulate a large value ('integral term wind-up'), which begins to reduce only after ER has reversed. When large enough the integral accumulator holds the controller output against an output limit, and even whilst reducing it continues to do so for a further *dead-time* period until it has emptied sufficiently. The upper part of *Figure 120* shows this effect schematically.

Integral desaturation eliminates this dead-time, allowing the output to move away from the limit as soon as the error reverses, and so avoids large PV overshoots. It does this by suspending integral action when an output limit is exceeded, and 'bleeding off' the integral accumulator until the calculated PID output is back on limit. At this point normal integral action is resumed. Integral desaturation is repeatedly activated if necessary to keep the output near the limit, until the error reverses. This is shown schematically in the lower part of *Figure 121*.

In this example the control action is *Integral only*, for clarity. The 'sawtooth' effect in the unlimited output arises from the repeated application of integral desaturation to bring it back to the limit. This is of course not seen in the limited output passed on to the plant.

Limit detection

Approach to a limit is detected by comparing (at each iteration) the calculated output value *OP* with the corresponding feedback value *FB*, which has been limited in the manual station. If these differ by more than a small margin a limit is judged to have been reached and integral desaturation starts. *Figure 121* shows this schematically. (Mathematical details are given later.)

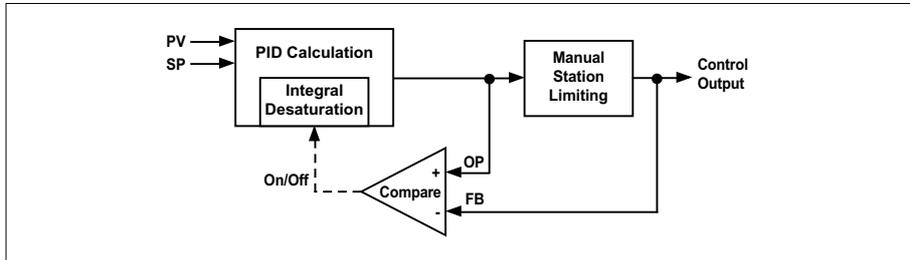


Figure 121 Application of integral desaturation (schematic)

The margin between *OP* and *FB* needed to trigger integral desaturation is very small (.006% approx.), so the feedback signal should come directly from the manual station output whenever possible. *FB* can originate elsewhere provided excessive time delays in the feedback loop are avoided. Delays longer than the sample time (*TS*) could cause the feedback to differ from the last output enough to trigger unwanted integral desaturation, which would slow down the controller action.

Effect of integral desaturation on dynamics

Figure 122 shows a more realistic example where integral desaturation is preventing excessive overshoot in a full PID control application. For comparison, *Figure 123* shows the same setup *without* integral desaturation.

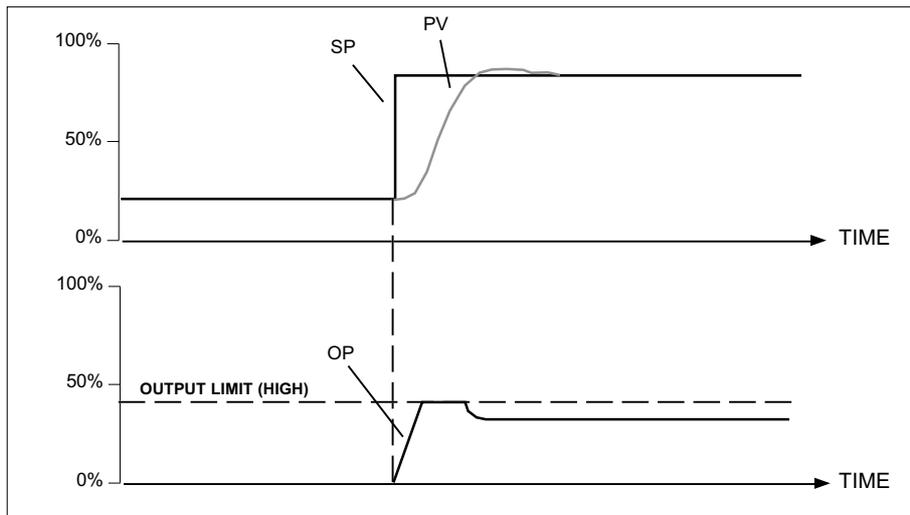


Figure 122 Application with integral desaturation (PID action)

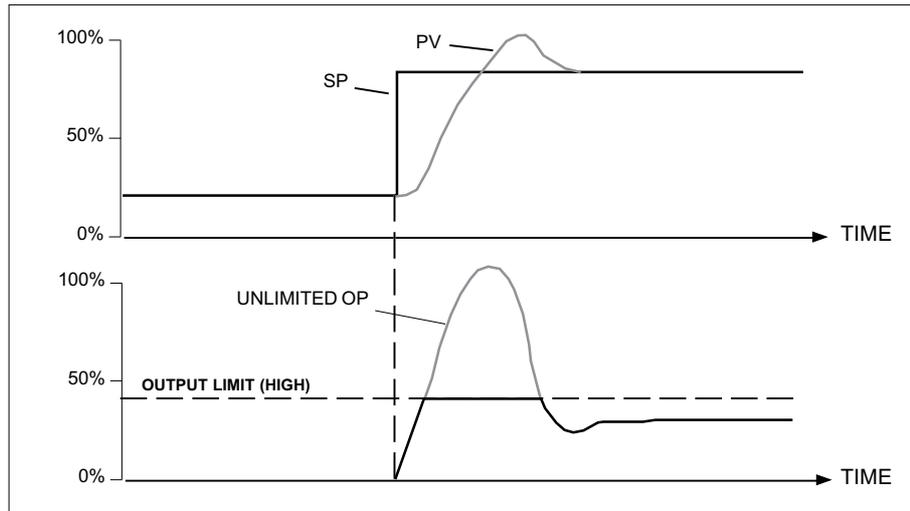


Figure 123 Application without integral desaturation (PID action)

Mathematical implementation

Definition of terms

OP = PID output

XP = Proportional band

ER = Error (PV-SP)

TS = Sample time (algorithm iteration period)

TI = Integral time constant

TD = Derivative time constant

FF = Feed-forward value

FB = Feedback value (normally $FB_n = OP_{n-1}$)

$\Delta PV = PV_n - PV_{n-1}$ (filtered)

PD = Combined Proportional and Derivative term

I = Integral term

(*Subscripts* denote algorithm iteration number)

In the LIN instruments, the PID output algorithm is usually implemented in general terms as:

$$OP = \frac{-100}{XP} \left[ER + \frac{TS}{TI} \sum ER_i + \frac{TD}{TS} \Delta PV \right] + FF$$

The flow diagram in *Figure 124* shows how integral balance and integral desaturation are actually applied in the PID algorithm at each iteration. In the diagram, the proportional and derivative terms are combined for clarity into a single term:

$$PD_n = ER_n + \frac{TD_n}{TS_n} \Delta PV_n$$

and the standard integral term is called:

$$I_n = I_{n-1} + \frac{TS_n}{TI_n} \cdot ER_n$$

Balancing the integral term

Figure 124 shows the ‘balancing’ of the integral term as setting:

$$I_n = \frac{-XP_n}{100} \left[FB_n - FB_n \right] - PD_n + \frac{TS_n}{TI_n} ER_n$$

To see the effect this has on the calculated output, this balanced In can be substituted into the equation defining OPn, which gives the result:

$$OP_n = FB_n - \left[\frac{100}{XP_n} \frac{TS_n}{TI_n} ER_n \right]$$

that is,

$$OP_n \text{ (balanced)} = OP_{n-1} \text{ (limited)} - \text{integral increment}$$

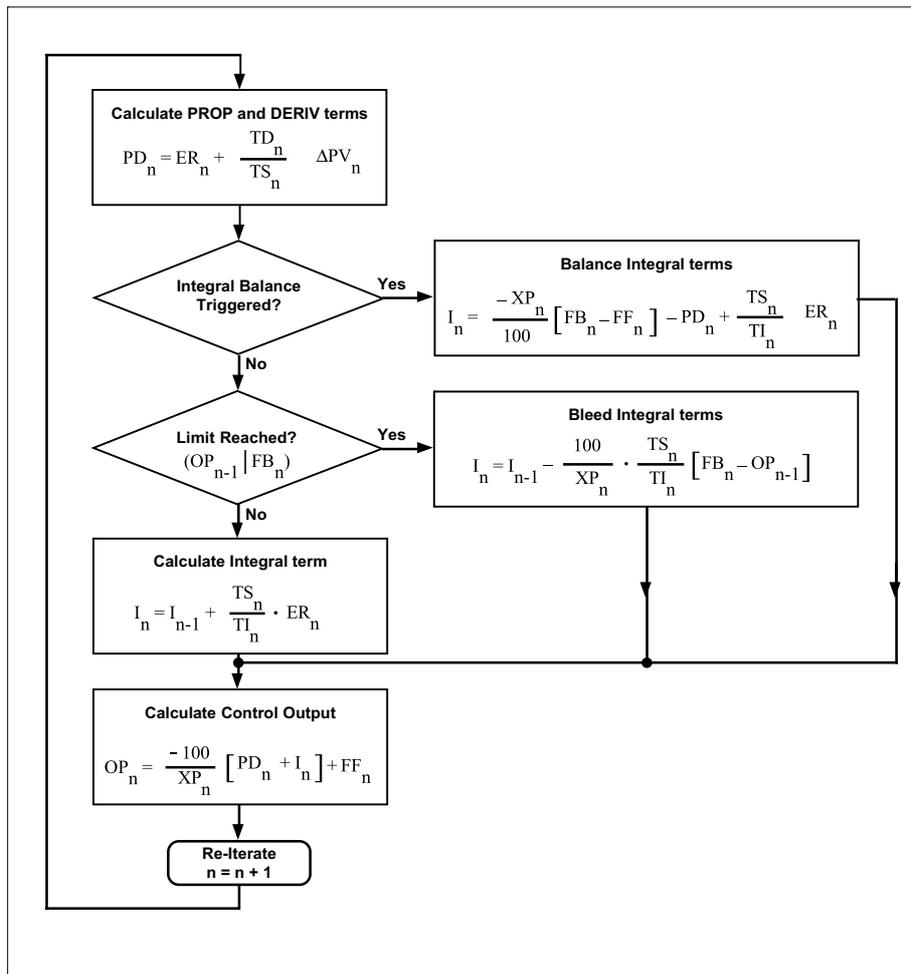


Figure 124 Implementation of integral balance & integral desaturation

(Because of the way the algorithm is executed the feedback at the nth iteration, FBn, is derived from the output at the previous iteration, OPn-1.) So balancing the integral term in this way results in the new output differing from the previous (limited) output by the integral term increment only, with no contribution from the proportional or derivative terms.

'Bleeding' the integral term

Figure 124 shows that each time integral desaturation is applied the integral term is reduced by the quantity:

$$\frac{XP_n}{100} \cdot \frac{TS_n}{TI_n} \left[FB_n - OP_{n-1} \right]$$

that is, by an amount proportional to $FB_n - OP_{n-1}$, the margin the calculated output is over limit. This causes the output to approach the limit exponentially until the margin is small enough (<0.006% approx.) not to trigger integral desaturation.

APPENDIX B THE LIN APPLICATION LAYER

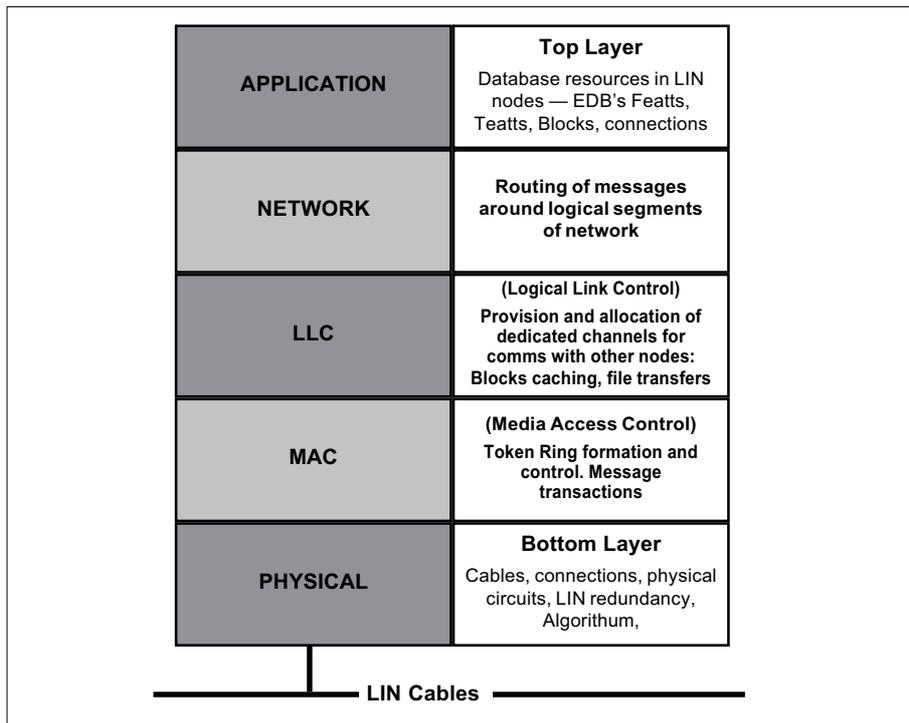


Figure 125 Layer structure of the LIN

The LIN (Local Instrument Network) is a complex system in its entirety, but it can be broken down into simpler 'layers'. Each layer is essential to the correct operation of the LIN communications, with the upper layers relying on the correct working of the lower layers for normal operation. *Figure 125* schematises these layers and summarises their functions.

This appendix concentrates on the top (Application) layer of the model, using a series of diagrams to explain schematically what happens when example function blocks are cached across the LIN to allow communications between them. The steps being illustrated are:

1. Configuring a cached block - the initial steps (*Figure 126*)
2. Block-caching at runtime (*Figure 127*)
3. Caching a second block between the same databases (*Figure 128*)
4. Caching another block in the opposite direction (*Figure 129*)
5. Caching blocks between three nodes (*Figure 130*).

1 CONFIGURING A CACHED BLOCK

In *Figure 126*, two nodes (instruments) are connected on the LIN - called 'EYCON_80' and 'T2550_20' at addresses 80 and 20 respectively. The aim of this step is to cache the function block 'LIC001' running in node T2550_20, into the EYCON_80's database. This is effectively creating in the EYCON_80 an 'image' of the remote block, allowing communications across the LIN.

In the figures, local blocks are drawn with bold outlines, while remote (cached) blocks have feint outlines. A **Teatt** (To External Attachment) and an **LEDB** (Locally Requested External Database) table entry are created in node 80's database, which are stored when the database is saved. Nothing yet happens in the T2550_20 database.

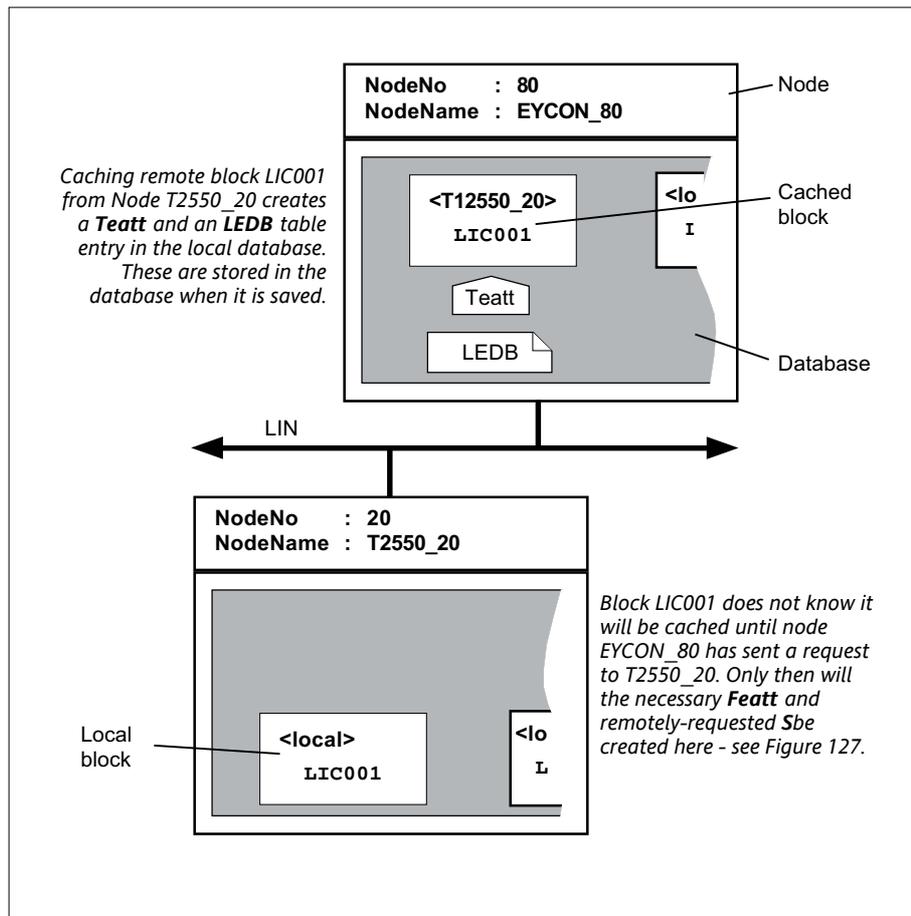


Figure 126 Configuring a cached block — initial steps

2 BLOCK-CACHING AT RUNTIME

Figure 127 shows what happens when the configured databases are running.

The T2550 database receives a request from the EYCON to cache block LIC001. The T2550 responds by creating an **REDB** (Remotely Requested External Database) table entry and a **Featt** (From External Attachment) in its database. These structures allow data to flow from the 'real local' LIC001 block to its remote cached 'image' in the EYCON.

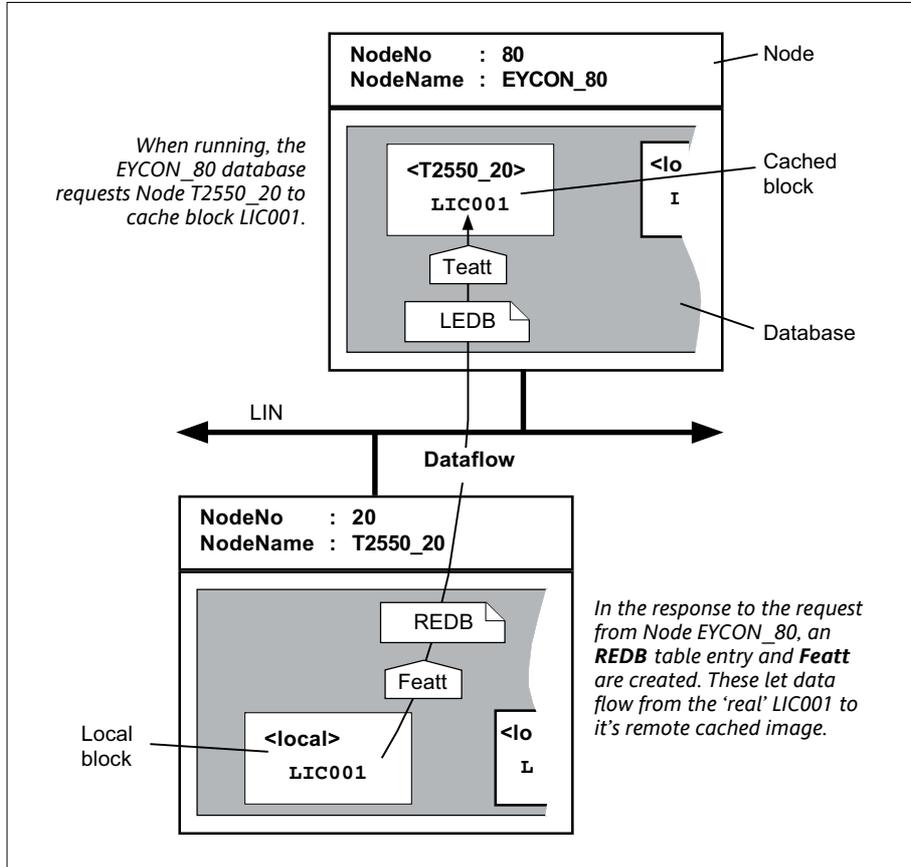


Figure 127 Block-caching at runtime

3 CACHING A SECOND BLOCK

In *Figure 128*, EYCON’s database has been configured to cache another block (FIC002) from the same remote T2550 node. This creates a second Teatt, but a further LEDB entry is not needed because one already exists linking to the remote T2550 database. This LEDB can be shared by all blocks cached between the two nodes.

At runtime the T2550 responds to the request to cache block FIC002 by creating a new Featt to allow FIC002 to send data. Note that the EDB connection between the two nodes handles all cached block data between them, but that multiple Featts and Teatts are required.

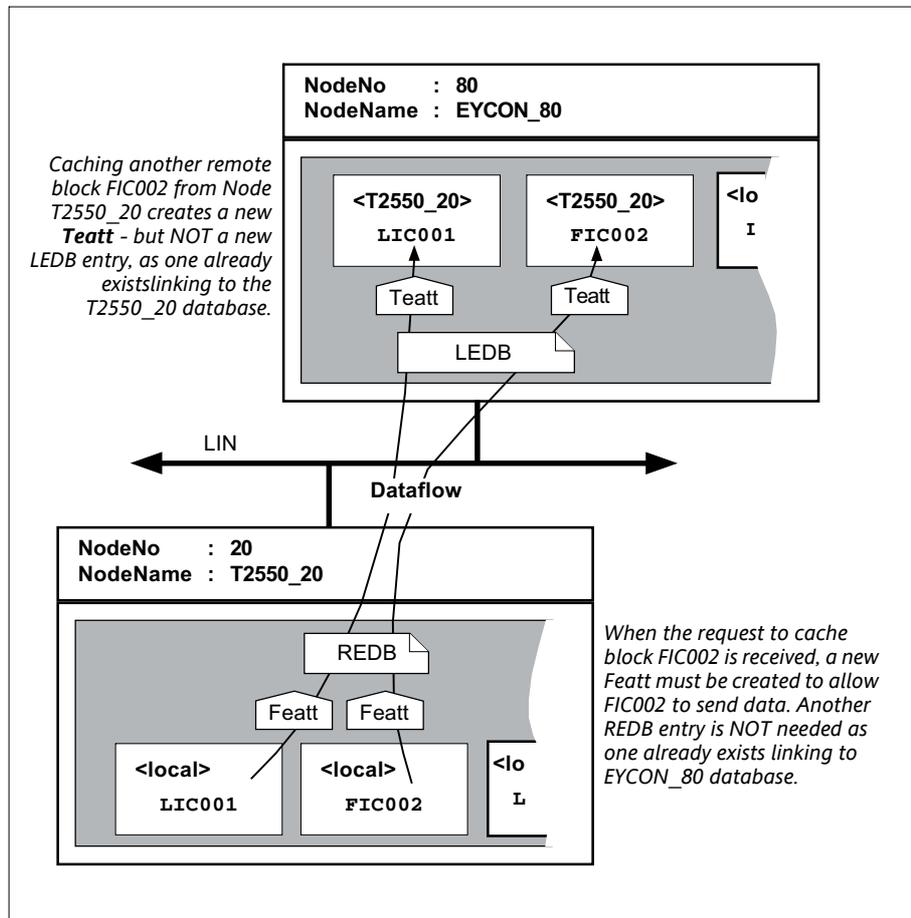


Figure 128 Caching a second block between the same databases

4 CACHING ANOTHER BLOCK IN THE OPPOSITE DIRECTION

Figure 129 shows what happens when a block from the EYCON_80 database is cached into the T2550_20 node, i.e. in the opposite direction.

As soon as the T2550 database is configured with cached block LRA_80, a Teatt and LEDB entry appear as usual and are saved in the database. LEDBs can handle cached block data flows in both directions, so at runtime an REDB will not be needed.

Similarly, in the EYCON node an REDB is not required (though a Featt is) in response to the request to cache LRA_80, because the existing LEDB can cope with the bidirectional data flow.

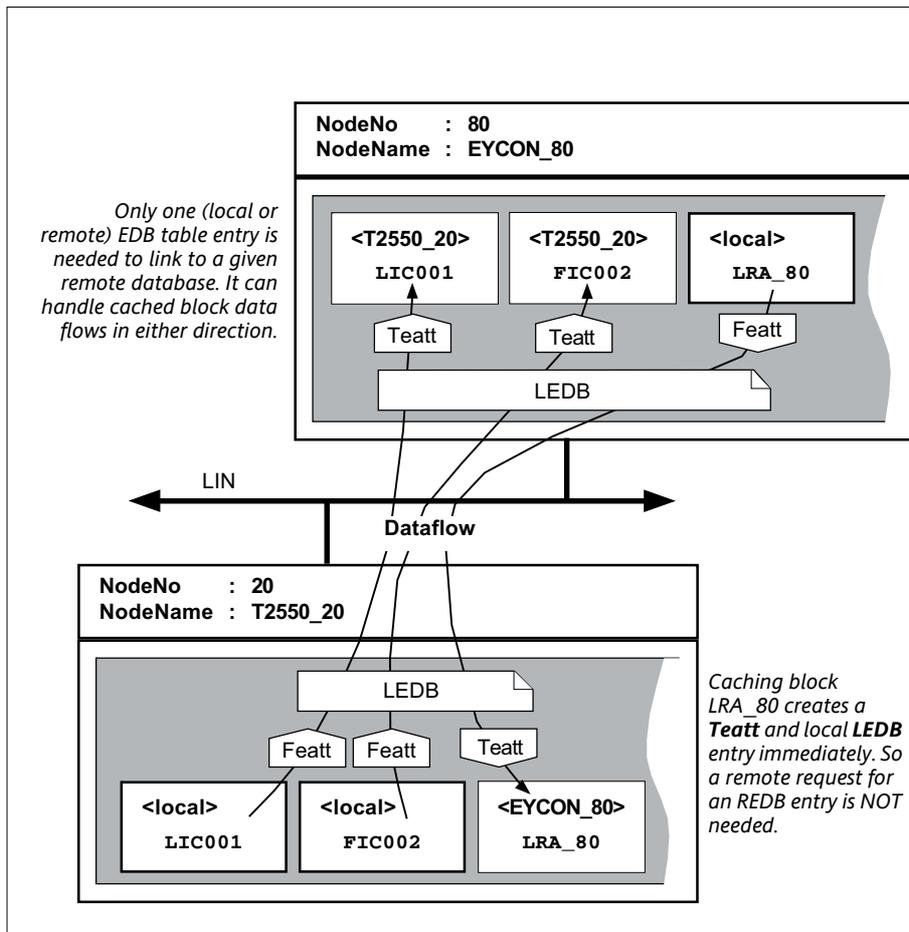


Figure 129 Caching another block in the opposite direction

5 CACHING BLOCKS BETWEEN THREE NODES

In *Figure 130* a third node has been connected to the LIN, T2550_60 at address 60. This node has been configured to cache the LIC001 block from the T2550_20 node, i.e. this block is being cached in two different nodes at once. As usual, a new Teatt and LEDB table entry are created at configuration time and saved in the T2550_60 database.

At runtime, T2550_60's request to cache the LIC001 block causes a new Featt and REDB to appear in T2550_20. One Featt is needed for each cached 'image' of the block requested, and the REDB entry is needed to allow data flow to the new node.

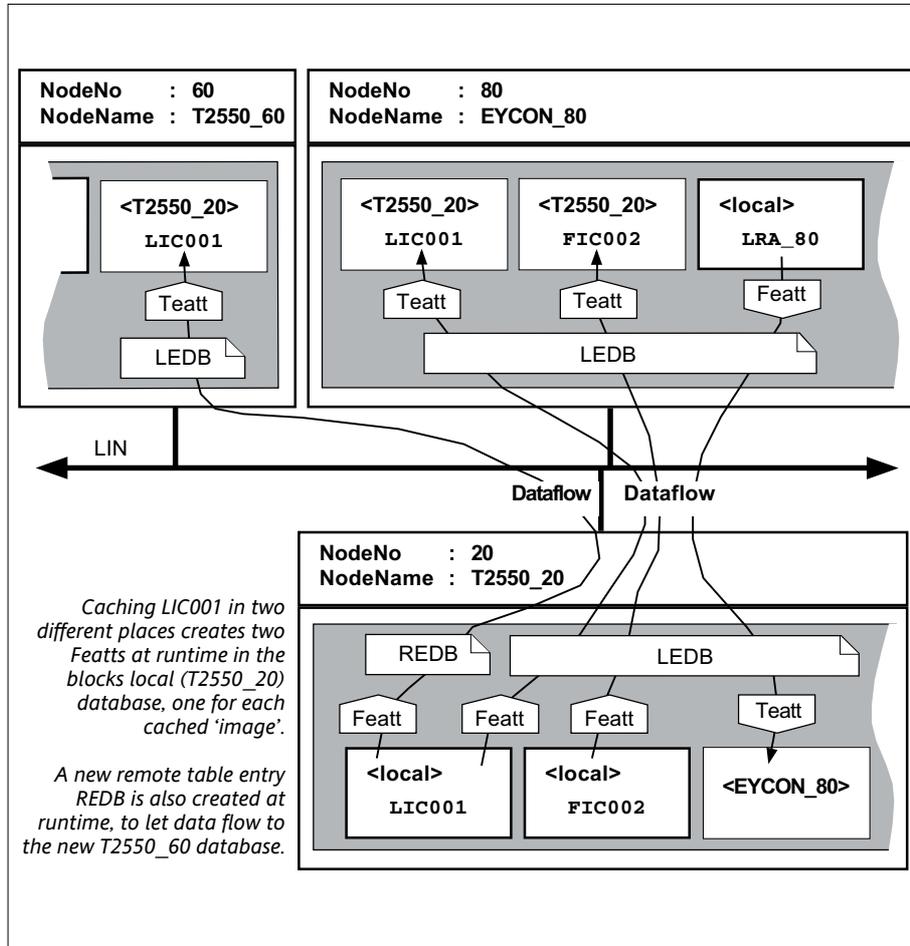


Figure 130 Caching blocks between three nodes

APPENDIX C OBSOLETE PRODUCT SUPPORT

This section provides an overview of all the LIN function blocks available for running in obsolete instruments that can be connected to the Local Instrument Network (LIN).

Full details for these function blocks are given in the Lin Blocks Reference Manual HA082375U003 issue 15 (Vintage).

INSTRUMENT SUPPORT OF FUNCTION BLOCKS

Note that not all the function blocks described in this manual can be run in all LIN instruments. *Table 218* indicates the degree of support currently offered for each LIN function block by a range of instruments.

Please refer to the appropriate document accompanying a particular instrument for details of any individual LIN function block parameters that are not supported, or only partially supported, or that have been special functions when used in that instrument.

Category	Block	Function	T100[1]	T231	T1000	T2001[2]	T4000[2]	T800/2900
BATCH	BAT_CTRL	Loading & control of a batch						3.0✓
	DISCREP	Transmitted/received digital signal-matching to diagnose plant faults	3/1✓	✓	3/1✓			✓
	RCP_LINE	Control downloading of recipes from a .UYR file in associated RCP_SET block						2.7✓
	RCP_SET	Control the recipe set (i.e. file to be used) for a number of recipe lines						2.7✓
	RECORD	Storage/retrieval of analogue/digital values for runtime use	3/1✓	✓	3/1✓		X	✓
	SFC_CON	Sequence (SFC) control, selection, and running	3/1S	S	3/1S		X	✓
	SFC_DISP	Display/monitoring/control of remotely running sequence (SFC)	3/1S	S	3/1S			✓
	SFC_MON	Sequence (SFC) runtime monitoring	3/1S	S	3/1S		X	✓
COMM	GW_CON	Control the Communications subsystem						
	GW_TBL	Show the Modbus configuration diagnostic table information						
	GWProfM_CON	Control the Profibus Master Communications device						
	GWProfS_CON	Control the Profibus Slave Communications device						
CONDITN	AGA8DATA	American Gas Association Report #8 calculations						✓
	AN_ALARM	Alarm, with absolute/Deviation/Rate alarms	✓	✓			✓	✓
	CHAR	16-point analogue characteriser	✓	✓	✓			✓
	DIGALARM	Digital alarm	✓	✓	✓			✓
	FILTER	First-order filter	✓	✓	✓			✓
	FLOWCOMP	Computes flow-rate, corrected for pressure, temperature, and density	5/1✓	5/1✓	5/1✓			✓
	GASCONC	Natural gas concentration data storage and validation						✓
	INVERT	Inverts signal about HR, LR limits	✓	✓	✓			✓
	LEADLAG	Lead-lag	✓	✓	✓			✓
	LEAD_LAG	Lead-lag	5/1✓	5/1✓	5/1✓			✓
	RANGE	Re-ranges an analogue input	5/1✓	5/1✓	5/1✓			✓
UCHAR	16-point characteriser for analogue input blocks	✓	✓	✓			✓	
CONFIG	PROGRAM	System block						
	PROGT600	System block						
	T100	System block	✓	✓	✓	P		
	T1000	System block			✓			
	T102	System block						
	T221	System block						
	T231	System block		✓	5/1✓			
	T302	System block						
	T600	System block						
	T932	System block						
	T800/T2900	System block						✓
	T940(X)	System block						
	Eycon-10/20	System block						
Tactician	System block							

Category	Block	Function	T100[1]	T231	T1000	T2001[2]	T4000[2]	T800/2900
CONTROL	3_TERM	Incremental form of the PID block						✓
	ANMS	Analogue manual station	✓	✓	✓	X	X	✓
	AN_CONN	Analogue connections	✓	✓	✓	X	X	✓
	DGMS	Digital manual stations	✓	✓	✓	X	X	✓
	DG_CONN	Digital connections	✓	✓	✓	X	X	✓
	MAN_STAT	Manual station, with connections to front panel displays						✓
	MODE	Control mode selection, with pushbutton masking						✓
	PID	PID control function	✓	✓	✓	X	X	✓
	PID_CONN	'Faceplate' for SETPOINT/3_TERM/MAN_STAT/MODE combination						✓
	PID_LINK	'Faceplate' for SETPOINT/3_TERM/MAN_STAT/MODE combination						✓
	SETPOINT	Generates a setpoint, with bias, limits, and alarms						✓
	SIM	Simulates two first order lags or capacity, with noise	✓	✓	✓			✓
	TP_CONN	Specifies up to 9 fields as EEPROM 'tepid data' at power-down						✓
CONVERT	ENUMENUM	Convert an enumerated value from one enumeration to another						2.7✓
	UINTENUM	Convert an integer into an enumeration						2.7✓
	ENUMUINT	Convert an enumeration into an integer						2.7✓
DCM...	D2X_LOOP	Access PID control loop in 2200/2400/2500						✓
	D2X_TUNE	Tune PID loop in 2200/2400/2500						✓
	D25_LOOP	Access PID control loop in 2500C						✓
	D25eLOOP	Access PID control loop in 2500E						2.6✓
	D25_TUNE	Tune PID loop in 2500C						✓
	D25eTUNE	Tune PID loop in 2500E						2.6✓
	D25_RAMP	Ramp remote setpoint in 2500C						✓
	D25eRAMP	Ramp remote setpoint in 2500E						2.6✓
	D25_MOD	Access single I/O physical module in 2500						✓
	D25_AI2	Access 2-channel analogue input module in 2500						✓
	D25_AI3	Access 3-channel analogue input module in 2500						✓
	D25_AI4	Access 4-channel analogue input module in 2500						2.7✓
	D25_AO2	Access 2-channel analogue output module in 2500						✓
	D25_AO4	Access 4-channel analogue output module in 2500						✓
	D25_DI4	Access 4-channel digital input module in 2500						✓
	D25_DI6	Access 6-channel digital input module in 2500						2.7✓
	D25_DI8	Access 8-channel digital input module in 2500						✓
	D25_DO4	Access 4-channel digital output module in 2500						✓
	D25_AICH	Access single analogue input channel in 2500						✓
	D25_AOCH	Access single analogue output channel in 2500						✓
D25_DICH	Access single digital input channel in 2500						✓	
D25_DOCH	Access single digital output channel in 2500						✓	

Category	Block	Function	T100[1]	T231	T1000	T2001[2]	T4000[2]	T800/2900	
...DCM	D25_R_CV	Access up to 8 user wiring real calculated values in 2500						✓	
	D25_B_CV	Access up to 8 user wiring boolean calculated values in 2500						✓	
	D25_R_UV	Access the 8 real user values in 2500						✓	
	D25_UALM	Access the user analogue or digital alarms in 2500						✓	
	DCM_I8	Access up to 8 integer (16-bit) parameters in 2500						✓	
	DCM_UI8	Access up to 8 unsigned integer (16-bit) parameters in 2500						✓	
	DCM_US8	Access up to 8 unsigned integer (8-bit) parameters in 2500						2.3✓	
	DCM_R8	Access up to 8 real number (32-bit) parameters in 2500						✓	
	DCM_W8	Access up to 8 'ABCD' hex word (16-bit) parameters in 2500						✓	
	DCM_Y8	Access up to 8 'AB' hex byte (8-bit) parameters in 2500						2.3✓	
	DCM_B8	Access up to 8 boolean parameters in 2500						✓	
	DCM_D8	Access up to 8 double-precision integer (32-bit) parameters in 2500						✓	
	DCM_S8	Access up to 8 signed integer (8-bit) parameters in 2500						2.3✓	
	DCM_T8	Access up to 8 time duration (32-bit) parameters in 2500						2.3✓	
	D2500	Provide overall view of remote 2500 instrument							✓
	D2000	Provide overall view of remote 2200 or 2400 instrument							✓
	D25_AI	Access single-channel analogue input module in 2500							✓
	D25_AI_T	Access single-channel analogue input module in 2500, with time-delayed alarms							3.1✓
	D25_AO	Access single-channel analogue output module in 2500							✓
	D25_DI	Access single-channel digital input module in 2500							✓
D25_DI_T	Access single-channel digital input module in 2500, with time-delayed input alarm							3.1✓	
D25_DO	Access single-channel digital output module in 2500							✓	
DIAG...	AGA8DIAG	AGA8 calculation diagnostics						2.4✓	
	ALH_DIAG	T800/T2900/T940 alarm history diagnostics						✓	
	ALINDIAG	Arcnet local instrument network (ALIN) diagnostics		✓				4.0✓	
	AMC_DIAG	Application master comms diagnostics						✓	
	BCS_DIAG	Routing broadcast diagnostics							
	CON_DIAG	Connection diagnostics							
	CON_ENT	Connection entry							
	CON_TBL	Connection table							
	DB_DIAG	Database resource information	✓	✓	3/1✓	P	✓	✓	
	DDR_DIAG	T800/T2900 data recording facility diagnostics							✓
	DTU_DIAG	Data transfer unit (DTU) diagnostics			3/1✓				
	EDB_DIAG	External remote database information	✓	✓	3/1✓	P	✓	4.0✓	
	EDB_TBL	External database diagnostics	✓	5/1✓	5/1✓	5/1P		4.0✓	
ELINDIAG	Ethernet LIN diagnostic block								

Category	Block	Function	T100[1]	T231	T1000	T2001[2]	T4000[2]	T800/2900
...DIAG	EIO_DIAG	Ethernet I/O diagnostic block						
	FDDADIAG	FTP Distributed Data Archiving diagnostic block				5/1✓		
	FSM_DIAG	File System Management diagnostics						
	FTQ_DIAG	PRMT queues statistics and status						
	FT_TUNE	Performance statistics						
	FWD_DIAG	Forwarding statistics						
	FWD_LOG	Forwarding log						
	GW_DIAG	Gateway file diagnostics						
	ICM_DIAG	ICM diagnostics						
	IDENTITY	Instrument Identification						
	ISB_DEXT	ISB (internal serial bus) high-level performance statistics						
	ISB_DIAG	ISB (internal serial bus) diagnostics						
	ISE_DIAG	Industrial strategy engine (T800/T2900) diagnostics						✓
	ISE_TUNE	Industrial strategy engine (T800/T2900) processor usage & server cycle times						2.4✓
	LIN_DEXT	LIN high level performance statistics	✓	✓	3/1✓	P	✓	
	LIN_DIAG	Local instrument network (LIN) diagnostics	✓	✓	3/1✓	P	✓	4.0✓
	LINMAPD	Local instrument network (LIN) mapping diagnostics						
	LLC_DIAG	Local Link Control diagnostics						
	LRA	Controls LIN redundancy algorithm operating modes	3/1✓	✓	3/1✓	P	✓	
	MDTUNE	Performance statistics	✓	✓	X	P		
	MDBDIAG	Modbus diagnostics table						
	NET_DIAG	Network diagnostics						
	NODE_MAP	LIN node protocol	✓	✓	5/1✓	5/1✓	5/1P	
	OPT_DIAG	Options/Licences Control diagnostics						
	PBUSDIAG	Profibus diagnostic						5.0✓
	PMC_DIAG	Profibus line diagnostics						2.7✓
	PNL_DIAG	T800/T2900 front panel diagnostics						✓
	POL_DIAG	Poll plant diagnostics						
	POL_ENT	Poll plant entry						
	POL_TBL	Poll table						
	PRPDIAG	Port resolution protocol diagnostics						
	PS_TASK	T940 task diagnostics				1/3✓		
	PS_TUNE	T940 performance						
	RED_CTRL	Redundancy control						
	ROUTETBL	Routing table	✓	✓	5/1✓	5/1✓	5/1P	
	RSRCDIAG	Resource & memory diagnostics						
	RTB_DIAG	Routing table diagnostics	✓		5/1✓	5/1✓	5/1 P	
	S6_DIAG	S6000 binary synchronous comms driver setup/diagnostics			✓	5/1✓		

Category	Block	Function	T100[1]	T231	T1000	T2001[2]	T4000[2]	T800/2900
...DIAG	SDX_IDLC	I/O data link control						
	SDX_RSRC	I/O interface status						
	SFC_DIAG	Sequence diagnostics and statistics (resources in use and available)						
	SUM_DIAG	Summary diagnostics	✓	5/1✓	5/1 ✓	5/1 P		
	T221TUNE	Tuning						
	T231TUNE	Performance statistics		✓	5/1✓			
	T600TUNE	Performance statistics						
	TOD_DIAG	Time-of-day diagnostics & control						
	TACTTUNE	Tactician Tack Summary			5/1✓			4.0✓
	UCTUNE	Performance statistics			3/1✓			
	USERTASK	User Task diagnostics						
	XEC_DIAG	Task diagnostics	✓	5/1✓	5/1✓	5/1P		✓
HIST	HIST	Collects and files data for longterm historic trends			✓			
I/O...	AI_CALIB	Guides operator through analogue input calibration						
	AN8_OUT	Analogue output channels, 8-way	5/1✓	5/1 X				
	ANIN	Analogue input channels	✓	X	X			
	ANIN_6	Analogue input channels, 6-way						
	ANOP	Analogue output channels	✓	X	X			
	AN_IP	Analogue input channels						
	AN_OUT	Analogue output channels						
	AO_CALIB	Guides operator through analogue output calibration						
	DGIN_1	Digital input channel, single						
	DGIN_8	Digital input channels, 8-way	✓	X	X			
	DGOUT_8	Digital output channels, 8-way	✓	X	X			
	DGPULS_4	Pulse outputs (1-shot, pulse train, dual pulse, time-proportioned output)						
	DG_IN	Digital input channels						
	DG_OUT	Digital output channels						
	FREQIN	Frequency input	2/1✓	X	X			
	FULL_TC8	Thermocouple input	2/1✓	X	X			
	RTD	Temperature reading analogue inputs	✓	X	X			
	RTD_6	Temperature reading analogue inputs, 6-way						
	TCOUPLE	Temperature reading analogue inputs	✓	X	X			
	MOD_UIO	Tactician module input/output block						
	AI_UIO	Tactician analogue input block						
	AO_UIO	Tactician analogue output block						
	CALIB_UIO	Tactician calibration block						
	DI_UIO	Tactician digital input block						
DO_UIO	Tactician digital output block							

Category	Block	Function	T100[1]	T231	T1000	T2001[2]	T4000[2]	T800/2900
.../O	TPO_UIO	Tactician time proportioning output block						
	FI_UIO	Tactician frequency input block						
LOGIC	AND4	4-input AND boolean function	✓	✓	✓			✓
	COMPARE	Indicates greater/less than/equal of 2 inputs	✓	✓	✓			✓
	COUNT	UP/DOWN pulse counter with START/END count target	✓	✓	✓			✓
	LATCH	D-type flip-Flop function	✓	✓	✓			✓
	NOT	NOT boolean function	✓	✓	✓			✓
	OR4	4-input OR boolean function	✓	✓	✓			✓
	PULSE	Pulse output, (monostable) function	✓	✓	✓			✓
	XOR4	4-input exclusive-OR boolean function	✓	✓	✓			✓
MATHS	ACTION	Action control, with use of stored variables and elapsed time	5/1✓	5/1✓	5/1✓			✓
	ADD2	Adds 2 inputs	✓	✓	✓			✓
	DIGACT	Action control, with use of stored digital variables and elapsed time	✓	✓	5/1✓			✓
	DIV2	Divides 2 inputs	✓	✓	✓			✓
	EXPR	Free-format maths expression with up to 4 inputs	✓	✓	✓			✓
	MUL2	Multiplies 2 inputs	✓	✓	✓			✓
	SUB2	Subtracts 2 inputs	✓	✓	✓			✓
OPERATOR	PNL_ACC	Panel access block - enable & monitor user logons (Visual Supervisor)						3.0✓
	PNL_CMD	Panel command block - take command of the panel (Visual Supervisor)						3.0✓
	PNL_DLG	Panel dialogue block - create dialogue box (Visual Supervisor)						3.0✓
	PNL_MSG	Panel message block - create message (Visual Supervisor)						3.0✓
	READER	Reader block - enable use of barcode readers, etc. (Visual Supervisor)						3.0✓
	EVENT	Event block - enable use of an action if an event occurs (Visual Supervisor)						4.1/- ✓
ORGANISE	AREA	Associate GROUP blocks with an area						✓
	GROUP	Associate T800/T2900 display and recorder channels into a group						✓
	LGROUP	Collect data from point blocks for archiving						✓
	LOGDEV	Specify and control access to an archive medium						✓
	LOGRPEX	Extend number of points logged by LGROUP block						2.7✓
	LPTDEV	Specify and control access to a printing device						2.7✓
	PGROUP	Collect data from point blocks for printing						2.7✓
PROGRAMMER	SPP_CTRL	Monitor, schedule, & control the setpoint programmer (T800/T2900)						✓
	SPP_DIG	Wire out digital setpoints from the setpoint programmer (T800/T2900)						✓
	SPP_EXT	Provide additional control over running setpoint program (T800/T2900)						2.7✓
	SPP_RAMP	Allow local ramping of setpoints in the setpoint programmer (T800/T2900)						2.7✓
RECORDER	DR_ALARM	Alarm filtering when recording to log files and/or printers						2.7✓
	DR_ANCHP	Data recording analogue channel point block						✓
	DR_DGCHP	Data recording digital channel point block						✓
	DR_REPRT	Generate user-defined text reports and send them to printers						2.7✓

Category	Block	Function	T100[1]	T231	T1000	T2001[2]	T4000[2]	T800/2900
S6000	6432 AI	Analogue input template		✓	✓		X	
	6432 AO	Analogue output template		✓	✓		X	
	6432 DI	Digital input template		✓	✓		X	
	6432 DO	Digital output template		✓	✓			
	GEN_COMM	Generic template		✓	3/2✓			
	CONN32AI	Analogue input connection block						
	CONN32AO	Analogue output connection block						
	S6350	Controller template						
	S6358	Controller template						
	S6360	Controller template			✓	✓	X	X
	S6366	Controller template			✓	✓	X	X
	SL0832	Controller template						
	SL6360	Controller template						
	SL6360R	Controller template						
	SL6366	Controller template						
	SL6366EX	Controller template (extrn.)						
	SL6366R	Controller template						
	SL6432 AI	Analogue input template						
	SL6432 AO	Analogue output template						
	SL6432 DI	Digital input template						
SL6432 DO	Digital output template							
SL6437	Controller template			2/1✓				
SELECTOR	2OF3VOTE	Selects 'best' input from three, by averaging only the within-tolerance inputs	✓	5/1✓	5/1✓			✓
	ALC	Alarm collection producing a common logic O/P	✓	✓	✓			✓
	PAGE	Selects up to 8 display pages through page pending feature						
	SELECT	Outputs highest, middle, lowest, or median of 2, 3, or 4 inputs	✓	✓	✓			✓
	SWITCH	Single-pole double-throw switch for analogue signals	✓	✓	✓			✓
	TAG	Specifies a user task (loop) tagname, selected from list of 8 tags						
TAN	RS485_NODE	TCS asynchronous network (TAN) configuration			2/1✓			
	RX_AN	Analogue receives			2/1✓			
	RX_DG	Analogue and digital receives			2/1✓			
	TAN_DIAG	TAN diagnostics			2/1✓			
	TX_AN	Analogue transmits			2/1✓			
	TX_DG	Digital transmits			2/1✓			
TIMING...	DELAY	Delay for deadtime applications	✓	5/1✓	5/1✓			✓
	DTIME	Delay, for deadtime applications	✓	✓	✓			
	RATE_ALM	Up- and down-rate alarm applied to PV, with OP held at last non-alarm value	✓	5/1✓	5/1✓			✓
	RATE_LMT	Rate-limiter and ramp generator	✓	5/1✓	5/1✓			✓

Category	Block	Function	T100[1]	T231	T1000	T2001[2]	T4000[2]	T800/2900
...TIMING	SEQ	Multi-segment slope/level/time, 15 O/P digitals	✓	✓	✓			✓
	SEQE	SEQ extender	✓	✓	✓			✓
	TIMEDATE	Clock and calendar event	✓	✓	✓			✓
	TIMER	Timer	3/2✓	✓	3/2✓			✓
	TOTAL	Totaliser (integrator) for analogue variable	✓	✓	✓			✓
	TOTAL2	Totaliser (integrator) for analogue variable, with calibration mode						✓
	TOT_CON	Placeholder for 32-bit parameter values (e.g. totalisations)						
	TPO	Time-Proportioning output	3/2✓	✓	3/2✓			✓

✓ = fully supported; S = fully supported in Sequence version only; P = supported locally on PCLIN card;
A = supported in 'Advanced' option only.

Revision number (e.g. 2/1) denotes software version where block first appeared.

X = partially supported by the hardware — may be used only for simulation or caching. [1]And T241.

[2]May have special template views that differ from those of regular LIN instruments — refer to product manuals.

†Supported only in T920 v3/1 onwards & T921 v2/1 onwards.

Table 218 Function block categories & instrument support

APPENDIX D COMMON FIELD PARAMETERS

This Appendix contains information that is common to most function blocks and is referenced by each appropriate block to the relevant page within this Appendix. These Common Field Parameters are as follows:

- **Block Specification Menu** Page 544
- **Alarms** Page 545

This Appendix also contains information about Filepaths that are defined as parameters in the following blocks: BAT_CTRL, BATCHCONTROL, PROGCTRL, RECORD, and SFC_CON.

- **Filepath parameter** Page 547

BLOCK SPECIFICATION MENU

This section, specifically the Tagname, Type, Task, LIN Name, DBase, Rate fields apply to all blocks except as indicated.

Tagname, Type, Task. These are block related parameters used to control the operation of the block within the LIN Database.

- **TagName.** This shows the user-defined (16-character max.) block Tagname, default 'NoName', which identifies and distinguishes it from other blocks of the same *Type*. The Tagname appears as a label on the block icon, below its *Type* identifier. The Tagname is usually related to the LIN Name unless it exceeds 8 characters, but can differ entirely by changing the Tag settings configuration using **View > Options > Settings > Tags** in *LINtools Engineering Studio*.

Note: A Strategy cannot be saved unless the TagName has been entered for all blocks in the LIN Database.

- **Type.** This read-only field shows the Template Function block category mnemonic, e.g. ANIN, PID, SIM, etc. which also appears as a label on the block icon, above its Tagname. This corresponds to the various chapters in this manual.
- **Task.** This is used to define the Task number on which this block will be run. It corresponds to a specific repeat rate, defined by the *Period ms* field. All blocks operating on this Task number will be updated at the intervals defined by the *Period ms* parameter. Task numbers are not used on the Eycon™ 10/20 Visual Supervisor.

LIN Name, DBase, Rate. These are LIN Database related parameters used to control the communications between LIN Instruments.

- **LIN Name.** This shows the maximum 8-character unique network LIN Name of the block for identification via the LIN. The unique network LIN Name can be derived from TagName or be entirely different by changing the Tag settings configuration using **View > Options > Settings > Tags** in *LINtools Engineering Studio*.
- **DBase.** (**Local/Layer/Remote**) This is used to define the location of the LIN Database that any block, other than the Header block, will run, **local** or **cached**. A cached block is a local 'image' of a remote block, i.e. a block running in another instrument on the LIN, that allows interaction with the remote block.

Note: In a cached block, Dbase specifies the name of the remote LIN Database containing the 'real' block.

The *Local* option indicates that this block will operate in the LIN Database in this Instrument, e.g. an Eycon™ 10/20 Visual Supervisor or T2550. The *Layer* option indicates that the block is local to the **base.dbf** in this LIN Database, but is on another layer and only referenced to make connections. The *Remote* option indicates that this block operates in a LIN Database in a LIN Instrument at a different LIN Node address. It is referenced to provide access to field values and make connections. When selected, **DBase** and **Node Address** parameters become available.

- **Rate.** This is used to defined the minimum update rate (cycle time in milliseconds) of a single or group of remote (cached) function blocks, at which an individual cached block is transmitted across the LIN. The default is 10ms minimum, i.e. 100Hz maximum. *Rate* can be set between 10ms and 64s. These rate values are minimum update times only, and heavily loaded networks may not be able to reach the faster update rates.

Note: For the system to run correctly, a database of the selected name must reside at the specified node, and must contain a corresponding real block of the same type as the local cached block.

ALARMS.

This field displays the name of the highest priority alarm which is either active and/or unacknowledged.

Alarms can also be subject to a suppression condition, refer to the *Alarm Suppression User Guide*, HA030272. An alarm which is suppressed masks the active status.

The Alarms field also accesses a window* with three columns - Name, Value, and Priority. The *Name* column lists the names of all the block alarms, e.g. Software, Hardware, etc. The *Value* column displays for each current alarm condition a message dependant upon 'Alarm Priority Number' setting as follows, 'In Ack', 'In Unack' or 'Out UnAck' where 'In' indicates an active Alarm condition and 'Out UnAck' indicates an Alarm that has cleared awaiting Acknowledgement. By 'right-clicking' in this field, unacknowledged Alarms can be Acknowledged. The *Priority* column shows the user-specified priority number for each alarm and is the only read/write field in the Alarms window.

Alarm Priority Number. There are four types of alarm priority. *Priority 0*, the lowest priority, switches the alarm off. *Priorities 1 to 5* specify 'self-acknowledging' alarms. For these, the 'In Ack' message appears when the alarm condition occurs, and disappears when the alarm has cleared without the need to acknowledge the alarm. *Priorities 6 to 15* specify 'acknowledging' alarms. For these the 'In UnAck' message appears when the alarm condition occurs. To acknowledge the alarm, 'right-click' in the field and select 'Acknowledge' from the drop down menu and the field now displays 'In Ack'. When the alarm condition clears the 'In Ack' message disappears. However, if the alarm condition clears and the Alarm has not been acknowledged ('In UnAck'), the 'Out UnAck' message appears indicating that the Alarm has cleared but is awaiting acknowledgement. *Priorities 11 to 15* for certain legacy instruments set a special bit in the Configuration (Header) block and activate a hardware alarm relay. (Not supported in current instruments.)

**NOTE In instruments without integral displays, the Alarms window can be seen using the Terminal Configurator attached to the instrument via Ethernet (ELIN), via a block cached in a display-type instrument (e.g. Eycon™ 10/20 Visual Supervisor) or via the LIN on a PC running LINtools' 'Connect' facility.*

- **Software.** All LIN function blocks have a Software (and a Combined) alarm, as a minimum and cannot be suppressed. A Software alarm (default priority 1) is generated upon checksum failure in a block, i.e. corruption of its database. For cached, S6000, and TAN blocks, failure of communications to the principal block also activates the alarm.
- **Hardware.** A hardware alarm is generated when any of the bits in the *Status* parameter are set, e.g. in the event of a power interruption, incorrect module type, hardware alarm generated by any of the fitted I/O modules, etc.
A hardware alarm also causes the respective I/O module LED to turn off, located at the top of each module. Also any other relevant LEDs fitted to a module will also indicate a Hardware fault condition, e.g. AI2 module.
- **Alarm Suppression.** All Alarms (except software and combined) can be subjected to Alarm Suppression functionality as appropriate. For further information see the *Alarm Suppression User Guide*, HA030272.
- **Combined.** All blocks have a Combined (and a Software) alarm, as a minimum. A Combined alarm is generated as a copy of the most significant alarm in the block and inherits the same Value message and Priority number.

Combined Alarms Logic with Multiple Alarms

There are two methods of determining the most significant alarm (and therefore the Combined alarm) when more than one alarm exists. To switch between these two methods, there is an option for the inclusion of a parameter in the `_system.opt` configuration file for the instrument (COMBINED_ALARM_METHOD keyword). When several alarms exist, the most significant alarm is chosen in accordance to the two basic rules:

- Legacy - unacknowledged alarms are chosen in preference to any acknowledged alarms. Thus, the ordering for determining the combined alarm is:
 1. Any unacknowledged Alarm is chosen in preference to any Acknowledged Alarm.
 2. If there is more than one Alarm as described in step 1 above, then the Alarm with the highest priority number is chosen.
 3. For Unacknowledged Alarms only; If there is more than one, then alarms that are currently active are chosen in preference to those Alarms currently not active.

4. If there is more than one Alarm (Acknowledged or Unacknowledged) with the same priority number, then the lowest numbered alarm (highest/first position in the alarms field list of names) is chosen.
- Modern - any current active alarms are chosen in preference to unacknowledged alarms. Thus, the ordering for determining the combined alarm is:
 1. Any active alarm is chosen in preference to one that is no longer active.
 2. If there is more than one alarm in step 1 above, then the alarm with the highest priority number is chosen.
 3. For active alarms only, if there is more than one, then alarms that are currently unacknowledged are chosen in preference to those Alarms which have been acknowledged.
 4. If there is more than one Alarm (Acknowledged or Unacknowledged) with the same priority number, then the lowest numbered alarm (highest/first position in the alarms field list of names) is chosen.

If the `_system.opt` file contains the `COMBINED_ALARM_METHOD` keyword set to a value of 1, the Modern method of determining the combined alarm is used. If the `COMBINED_ALARM_METHOD` keyword has a value of 0, or the does not appear in the `_system.opt` file at all, then the legacy method is used.

The Instrument Options Editor can be used to modify the status of the `COMBINED_ALARM_METHOD` by toggling the check box for the *Combine active alarms* field in the *Startup* tab. If ticked, the Modern method of active alarms having higher priority than unacknowledged alarms is used. If un-ticked, the legacy method of unacknowledged alarms having higher priority over active alarms is used.

FILEPATH PARAMETER

The Filepath field exists in the BAT_CTRL, BatchControl, PROGCTRL, RECORD and SFC_CON block types. It is always paired with a field called Filename. When concatenated together <Filepath>:<Filename> provide a full path to a file which the block will either load or save depending on other actions taken in the block context.

In many cases Filepath can be left blank – in this case the file is accessed via the ‘default drive’ which in all current LIN instruments is the ‘E:’ drive.

Note: LIN filing does not support the concept of directories – hence the file will always be in the root of the specified path.

The syntax of the Filepath field contents can be expressed as follows:

```
[<node_address>::][<drive_letter>:]
```

That is, the filepath may consist of a node address (identified as such by a trailing ‘::’) followed by a drive letter (identified as such by a trailing ‘:’). Both elements are optional, but if both are included they must be in the order specified above.

Node Address

The node address is a four-digit hexadecimal representation of the LIN ‘node number’.

On many products the LIN node number is set via an eight-way DIP switch somewhere on the device – this equates to only two hexadecimal digits. If the LIN node number is ‘XY’ the four-digit hexadecimal representation should be ‘0XXY’ (that is, the 2nd & 3rd digits should be the same). This four digit representation is known as ‘16-bit addressing’.

The 16-bit version of addressing was introduced in the early 1990s when the T221 (OLIN/ALIN) bridge was released. Instruments supporting only OLIN comms (identifiable via the thick coax comms cable) only support the eight-bit version of addressing. OLIN-only instruments include the T100, T1000, T231. These older products used a two-digit hexadecimal representation of the node address in the filepath field.

The 16-bit addressing was introduced to allow for the possibility of an increased address space – although this has never been implemented.

Drive Letter

The drive letter must be in the range ‘A’ to ‘Z’ (uppercase). It must also be a valid drive letter for the target device. All LIN instruments support ‘E’ as their default drive. Other drive letters are supported on different devices (for example, if a USB stick is inserted into the T2750, this is assigned ‘B’ as the drive letter).

The most common example of a drive letter reference is thus ‘E:’

Examples for the ‘Filepath’ field

The following are example formats for the Filepath field:

- ‘’ (that is, an empty string) – a blank filepath, so the file should be located on the local instrument’s ‘E:’ drive.
- ‘E:’ – the local instrument’s ‘E:’ drive (i.e. the same effect as an empty string).
- ‘B:’ – the local instrument’s ‘B:’ drive (e.g. the T2750’s USB stick – note: this is not recommended).
- ‘0112::’ – on the ‘E:’ drive of remote LIN instrument with node number 0x12.
- ‘0112::E:’ – on the ‘E:’ drive of remote LIN instrument with node number 0x12.
- ‘12::E:’ – on the ‘E:’ drive of remote LIN instrument with node number 0x12 (this shorter node number syntax is only valid on OLIN-only instruments – e.g. T100).

Index

LIN BLOCKS REFERENCE MANUAL

Symbols

.asc file	428
.ASn to .A99	428
.bmp file	233
.cpf file	128, 132, 133, 139
.dbf file	54, 59, 120, 124, 134
.gsd file	61
.gwf file	47, 61
.Lnn file	122
.pkd file	428
.PKn to .P99	428
.rcd file	28
.run file	121, 124, 132, 137
.sbf file	20, 138
.spp file	443
.sto file	397, 399, 401
.tpf file	126
.txt file	34
.udb file	126, 133
.uhh file	428
.upb file	55
.uyb file	36, 40
.uyr file	34
.uyt file	233, 433
.uyy file	445

Numerics

2OF3VOTE block	484
32-bit parameter values	514
3_TERM block	162, 163, 168, 172, 176, 178
4-channel	
digital pulse block	319
8-channel	
digital input block	313
digital output block	316

A

ACT_2A2W3T block	401
Action	399
ACTION block	397, 399
Action Block with Gated Downtimers block	401
Action qualifier	397, 400
Active control mode	519
Adaptive gain control	523
ADD2 block	393
AGA8 diagnostic block	210
AGA8DATA block	84, 88, 210
AGA8DIAG block	210
AI_CALIB block	306, 325
AI_UIO block	186, 334

Alarm

Acknowledging	545
Area	423
Collection block	483
Functionality	545
History diagnostic block	212
Point Block	473
Priority Numbers	545
Suppression	545
UnSupAll (Tactician)	133
UnSupAll (Visual Supervisor)	138
ALC block	483
ALH_DIAG block	212
ALIN	227, 291
Network	125, 127
Node address	125
ALINDIAG block	213
AMC_DIAG block	214
American Gas Association Report #8 Data block	88
AN_ALARM block	75
Analogue data block	180
Analogue input block	304, 334, 344
Analogue input calibration block	325
Analogue input/output calib block	348
Analogue inversion block	68
Analogue output block	344
Analogue output calibration block	329
AN_CONN block	120, 154, 158
AND block	382
AND4 block	382
AN_DATA block	94
ANIN block	154
AN_IP block	304
ANMS block	148
AN_OUT block	310, 329
AO_CALIB block	310, 329
AO_UIO block	344
Application layer	528
Application master comms diagnostic block	214
Archive medium	427, 467
AREA block	406, 421, 422, 423
Area Overview user screen	423
Audit Trail	253, 254
Auditor	137
Auto mode	266, 519

- B**
- Bar 68, 69
 - Bar Chart 149, 153
 - Barcode readers 418
 - Bargraph 161, 168
 - Base gas 90
 - Batch 137
 - Batch control block (Eycon) 36
 - Batch control block (T2750) 40
 - BATCH function blocks 12, 535
 - BATCHCONTROL 40
 - BAT_CTRL 36
 - BAT_CTRL block 36, 40, 428
 - Battery 132, 139
 - BattFail 132
 - BattLow 132
 - BCS_DIAG block 215
 - Best-average block 484
 - Binary/hexadecimal/decimal conversion chart 11
 - Bleeding the integral term 527
 - Block Specification Menu 544
 - Bumps 522
- C**
- CacheCon 269, 272, 295
 - Cached 210
 - block 21, 120, 126, 133, 138, 210, 248, 250, 271, 544
 - block update rate tuning algorithm 221
 - connections task 269, 295, 299
 - RECORD block 29
 - server task 269, 295
 - sync task 299
 - CacheSrv 269, 272, 295
 - Calibration 325, 326, 330, 350
 - block 325, 329
 - procedures 326, 330, 350
 - CALIB_UIO block 336, 348
 - CARB_DIFF block 92, 98, 180
 - Carbon Diffusion Calculation Block 92
 - CConn 299
 - CHAR block 69
 - Characterisation block 69
 - Choke valve 320
 - ColdStart 132, 492
 - ColdStart Parameter File, .cpf 128, 132, 139
 - Combined alarm 545
 - COMMS function blocks 12, 45, 535
 - Communications
 - Profibus 54, 61
 - statistics 214, 261
 - Communications diagnostic block 214
 - COMPARE block 389
 - Compensated flow block 81
 - Compression ratio of data 430
 - CON_DIAG block 216
 - CONDITION function blocks 12, 67, 535
 - CON_ENT block 217
 - CONFIG function blocks 12, 535
 - Connection diagnostics block 216
 - Connection entry block 217
 - Connection table block 218
 - CON_TBL block 218
 - Continuous transfer function 521
 - Control block 143, 518
 - CONTROL function blocks 140
 - Control Loop operation 518
 - Control mode 519
 - Control Strategy. *See* Strategy
 - CONVERT block 202
 - CONVERT function blocks 13, 202, 536
 - COUNT block 386
 - CSync 299
- D**
- D25_D04 block 438
 - D25_RAMP block 438
 - Data compression ratio 430
 - Data logging 422, 467
 - Data Recording 468, 471
 - Analogue channel point block 468
 - Digital channel point block 471
 - Facility diagnostic block 220
 - Flash memory 438, 467
 - Database diagnostics block 219
 - Date Format 434
 - Date/time Format 434
 - DB_DIAG block 219
 - DCM 208, 536
 - DCM Function blocks 208
 - DDR_DIAG block 220
 - Deadband 188
 - Deadtime 523
 - Deadtime block 497
 - Debounce 315, 354, 369
 - Decimal/hexadecimal/binary conversion chart 11
 - DELAY block 506
 - Derivative Time, TD 520
 - DEVICES sub-directory 399
 - Devolved Control Module blocks, DCM 208
 - DG_CONN 156
 - DG_CONN block 29, 120, 156, 158
 - DG_IN block 313
 - DGIN block 156
 - DGIN_8 block 156
 - DGMS block 150, 151
 - DG_OUT block 316
 - DGPULS_4 block 319
 - DIAG function blocks 209, 537
 - Dialogues 413
 - Differential pressure 82
 - DIGACT block 399
 - DIGALARM block 77, 248, 313
 - Digital control algorithm 520

- Digital input block 157, 352
 Digital output block 355
 DISCREP 30
 DISCREP block 30
 Discrepancy batch block 30
 Display items 422
 Display type blocks 425
 DI_UIO block 352
 DIV2 block 393
 DO_UIO block 103, 106, 355
 DR_ALARM block 427, 429, 434, 473, 478
 DR_ANCHP block 422, 427, 429, 468
 DR_DGCHP block 422, 427, 429, 438, 471
 DR_REPRT block 427, 429, 434, 452, 474
 DTIME block 497
 Duplex mode 248
- E**
- E2PROM 158
 E2ROM 121
 EDB_DIAG block 221
 EDB_TBL block 223
 EIO_DIAG block 224
 ELIN 138, 227
 Network 126, 127, 133, 137, 227
 Node address 126, 133
 ELIN I/O diagnostic block 224
 ELINDIAG block 227
 EMAPDIAG block 229
 ENUMENUM block 203
 Enumerated to enumerated converter block 203
 Enumerated to integer converter block 205
 ENUMUINT block 205
 ER 520
 Error 523
 Ethernet LIN diagnostic block 227
 Ethernet Local Instrument Network. *See* ELIN
 Ethernet Mapping diagnostic block 229
 Ethernet Rate Limit diagnostic block 230
 ETH_RT_LIM block 230
 EVENT block 403, 420
 Event block 420
 Exclusive-OR block 382
 EXPR block 104, 158, 394
 Expression block
 Logical OR 394
 Logical XOR 395
 External database diag block 221, 268
 External database table block 223
 EYCON-10 136
 Configuration Block 136
 Eycon-10
 PROGCHAN 445, 453, 462
 PROGCTRL 445, 453, 462
 SEGMENT 445, 453, 462
 EYCON-20 136
 Configuration Block 136
- Eycon-20
 PROGCHAN 445, 453, 462
 PROGCTRL 445, 453, 462
 SEGMENT 445, 453, 462
- F**
- Faceplate block 176, 178
 Fallback modes 519
 FDDADIAG block 232
 Featt 530
 Feedback value FB 524
 FF offset 520
 File System Management diag block 234
 File Transfer Protocol (FTP) 232, 417
 FTP distributed data archiving diagnostic block 232
 Logging 232
 Filter 67, 78, 79
 Debounce 315
 Digital limit 520
 First-order 307, 361, 521
 Input/Output 79
 LagTime 79
 LeadTime 79
 FILTER block 71
 FI_UIO block 361
 FLOWCOMP block 81
 Forced
 Auto 519
 Manual 519
 Forwarding Log block 238
 Forwarding Statistics block 237
 Frequency
 Input block 361
 Input calib. sequence 328
 Frequency value 361
 Front-panel diagnostic block 267
 FSM_DIAG block 234
 FTQ_DIAG block 236
 FULL_TC8 block 118
 Function block
 Category 9, 208, 403, 467
 Instrument support 10, 534
 FWD_DIAG block 237
 FWD_LOG block 238
- G**
- Gas 88, 210
 CO2 88
 Endothermic Gas Reference 106
 Gas supercompressibility 90
 GASCONC block 84, 88, 89
 GateWay
 Profibus 53
 GateWay Configuration block 46
 GateWay Profibus Master Configuration block 53
 GateWay Profibus Slave Configuration block 58
 GateWay Table block 50

GROUP block 406, 421, 422, 425, 478
 channels for T2900 preplot 425
 Group display modes 423
 Group Overview 423
 GW_CON 46
 GW_CON block 46, 52
 GWProfM_CON block 53
 GWProfS_CON block 58
 GW_TBL 50
 GW_TBL block 50

H

Handshake 20
 Header block 375
 Hexadecimal/decimal/binary conversion chart 11
 High-level analogue
 input calibration sequence 327
 output calibration sequence 331
 HISTDATA block 478
 HISTORIAN function blocks 16, 302, 539
 HOLD mode 519
 Hysteresis 141, 143, 161, 166, 188, 196, 338, 363, 510, 511

I

I/O function blocks 16, 303, 539
 I/O system units 440
 ICM manager 296
 ICM_DIAG block 239
 IDENTITY block 240
 Impedance measurement 106
 Incremental PID block 163
 Input block
 execution 305, 335
 INS button 11, 119
 T640 instrument 119
 T640 series instrument 11
 Instrument Identification/Status Diagnostic block 240
 Integer
 to enumerated converter block 204
 Integral
 accumulator 523
 balance 518, 521
 desaturation 518, 523
 term wind-up 523
 Integral Time, TI 520
 Internal Serial Bus (ISB) 111, 245, 287
 diagnostic extension block 245
 ISB performance block 246
 Interprocessor Comms Mechanism (ICM) 239, 270
 Interprocessor Comms Mechanism (ICM)statistics block
 239
 INVERT block 68
 ISB_DEXT block 245
 ISB_DIAG block 245, 246

L

Laplace transformation 520
 LATCH block 384
 Lead/Lag (Filter) block 78
 Lead/Lag block 73
 LEAD_LAG block 78
 LEADLAG block 73
 LEDB 529
 LGROUP block 422, 427, 429, 431
 License 259
 Product licence 132
 Violation 259
 Limit detection 524
 LIN 270, 291, 296, 514
 Application layer 528
 block 409
 block category 9
 block name 158
 block type 203
 block update rate 504
 cached block 544
 Database faults 290
 ELIN 137
 instrument support 10, 534
 Network 125, 127, 247
 Operating 209
 Redundancy algorithm 290
 Segment 290
 Sequence 20
 Statistics 247
 LIN High-level diagnostics extension block 247
 LIN Mapping diagnostic block 248
 LIN Node Protocol block 258
 LIN_DEXT block 247
 LIN_DIAG block 247
 Line gas 90
 Linearisation 67
 LINMAPD block 248
 LINtools 22, 119, 289, 334, 344, 352, 545
 LLC_DIAG block 251
 Local
 server 271
 setpoint 521
 LOG group 422, 467
 Log Group Extension block 431
 Log Group Organisation block 429
 LOGDEV block 422, 427, 429
 Logfile 122
 Logging Device Organisation block 427
 LOGGRPEX block 429, 431
 LOGIC function blocks 540
 Logical Link Control (LLC) 251, 270, 296
 LLC Diagnostic block 251
 LLC layer 227
 LLC performance 213
 LOOP block 343
 Loop Break 201

- Loop operating mode518
 Loop Proportional, Integral, Derivative block182
 LOOP_PID block182, 198
 LPTDEV block432, 433, 434
 LRA block290
- M**
- MAN_STAT block168, 172, 176, 178, 311
 MANUAL519
 Manual station output524
 Manual Station Block148, 150
 Master blocks
 S6000 Series479
 MASTER node297
 MATHS function blocks16, 392, 540
 MDBDIAG block514
 Media Access Control (MAC)251
 MAC layer227
 MAC manager270, 296
 MAC performance213
 Messages239, 409, 411
 operator406
 quantity239
 type239
 Modbus communications49, 52, 137
 MOD_DI_UIO block369
 MOD_DO_UIO block373
 Mode518
 Auto 144, 236, 239, 245, 246, 292, 306, 310, 314, 317, 489, 519
 Calib306, 310
 Dual_Pls321
 Duplex248
 Forced Auto144, 519
 Forced Manual144, 519
 Hold236, 239, 245, 246, 266, 489, 519
 Holdboth239
 Init236, 239, 245, 246
 InitBoth239
 Isolated297
 Manual144, 306, 310, 314, 317, 489, 519
 One-shot pulse320
 Pulse319
 Pulse-train320
 Remote144, 519
 Simplex248
 TPO (time-proportioned output)320
 Track143, 310, 317, 489, 519
 MODE block172, 176, 178
 Mode control block172
 MOD_UIO block332
 Module Input/Output Block332
 Molar gas density90
 Muffled297
 MUL2 block393
 Multi-Channel Digital Input Module block369
 Multi-Channel Digital Output Module block373
- N**
- NATCDIAG block253
 NATPDIAG block254
 Natural Gas Concentration Data block84
 NET_DIAG block255
 netHOST block256
 Network
 ALIN127
 ELIN126, 133, 137, 138
 LIN125, 127
 Modbus49, 137
 Profibus53, 54, 61
 SLIN129, 138
 Network Audit Trail Consumer Diagnostic block253
 Network Audit Trail Provider Diagnostic block254
 Network Diagnostics block255, 256
 Node address
 ALIN125
 ELIN126, 133
 NODE_MAP block258
 NOT block383
 Number format469
- O**
- OP520
 response522
 Open-circuit detection /protection304
 OPERATOR function blocks17, 403, 540
 Operator message406
 OPT_DIAG block259
 Options Diagnostic block259
 OR block382
 OR4 block382
 ORGANISATION function blocks17, 540
 ORGANISE function blocks422
 Output
 bumps522
 limits523
 Overshoot522, 524
- P**
- Panel Access block416
 Panel Command block404
 Panel Diagnostics block267
 Panel Dialogue block413
 Panel Dictionary block408
 Panel Message block409
 Parameter
 Options469
 Status11
 PBUSDIAG block261
 PGROUP block433, 434

PID	518	Rate	
3-Term control algorithm	141	Alarm block	508
algorithm	163	Limit block	511
connection block	176	RATE_ALM block	508
control block . 140, 141, 159, 163, 168, 172, 176, 518		RATE_LMT block	511
control equation	520	RAW_COM block	62
linking block	178	RCP_LINE	34
loop	120	RCP_LINE block	34
PID control	13, 198, 201	RCP_SET	33
PID Tuning Set block	198	RCP_SET block	33, 34, 452
PID_CONN block	176	READER block	403, 418
PID_LINK block	178	Reader Interface Language file	418, 419
PMC_DIAG block	265	Real and Time Converter Block	206
PNL_ACC block	403, 416	REALTIME block	206
PNL_CMD block	403, 404, 411	Real-Time Clock ... 117, 118, 124, 127, 130, 131, 136, 297, 298	
PNL_DIAG block	267	RTCFail	132
PNL_DICT block	408	Recipe	137
PNL_DLG block	403, 413	line block	34
PNL_MSG block	403, 409	set block	33
Point/Loop View	423	RECORD	27
Port Resolution Protocol (PRP) diagnostic block	268	Record Batch block	27
Preplot	425	RECORD block	27, 452
Pressure	82	.rcd file	28
Priority	519	Recorder channel blocks	425
Processor Redundancy Management Task (PRMT) ... 276		RECORDER function blocks	17, 467, 540
PRMT Queues Diagnostic Block	236	Recording channels	422
Profibus communication	54, 61	Recording facility diagnostic block	220
Profibus Diagnostic Block	261	REDB	530
Profibus GateWay	53, 58	RED_CTRL block	248, 249, 276
Profibus Line Diagnostic Block	265	Redundancy control block	276
PROGCHAN block	445, 453, 462	Redundant system	
PROGCTRL block	445, 453, 462	Chngovr	135
PROGRAM		CldStPri	134
Configuration block	116	Product Licence for Secondary processor	132
PROGRAM block	116	Red	135
PROGRAMMER function blocks	435, 540	SavActiv	134
Programmer Wizard	445, 453, 462	Remote	519
PROGT600	116	Mode	519
Configuration block	116	Removable archive medium	422
Proportional Control	520, 523	Report	137
PRPDIAG block	268	Report Point Block	474
PS_TASK block	269	Resource Diagnostic Block	283
PS_TUNE block	271	RGROUP Block	475
Pulse		RMEMDIAG block	279
output	495	ROUTETBL block	282
totalisation	304	Routing broadcast diagnostic block	215
PULSE Block	380	Routing table block	282
PV overshoot	191, 523	Routing table diagnostic block	285
PwrFail	129, 135, 138	RSRCDIAG block	283
Q		RTB_DIAG block	285
QueueS	236	RTD analogue input calibration sequence	328
R		RTD block	118
RANGE block	80	Runtime Executive (XEC)	270, 295
RARCDIAG block	273	S	
		S6000 function blocks	17, 479, 541

Sample time (TS)	524
Sawtooth effect	523
SCADA system	514
Scaled pulse count	361
SD_DIAG block	287
SEGMENT block	445, 453, 462
SELECT block	481
SELECTOR function blocks	17, 481, 541
Self-acknowledging alarms	545
Sensor Break	305, 335
detection	305, 334
protection	305, 335
Sensor break	335, 341
SEQ block	489, 494
SEQE block	489, 494
Sequence	118, 404, 435, 443
block	489
diagnostics block	289
extension block	494
Setpoint	143, 198, 200, 201, 445, 489, 522
control block	159, 436
Local (SL)	143, 521
Program	435
Program digital block	438
Program Editor	137, 436, 439
Program extension block	443
Program local ramp block	440
Program panel agent	436
Resultant (SP)	143, 522
SETPOINT block	159, 166, 172, 176, 178
SFC	20, 23, 404, 435, 436
Control block	20
Display block	24, 26
Monitor block	23
SFC_CON	20
SFC_CON block	20, 23, 24, 452
SFC_DIAG block	289
SFC_DISP block	22, 24, 26
SFC_DISP_EX	26
SFC_MON block	23
Signal conditioning	67
SIM block	152
Simplex mode	248
Simulation block	152
Slave blocks	
S6000 Series	479
Slave mode	297
SLIN	
Network	129, 138
Software	
Alarm	545
SPP_CTRL block	435, 436
SPP_DIG block	435, 438
SPP_EXT block	435, 443
SPP_RAMP block	435, 440
Remote I/O instrument	440
Status symbols	11
Steel Composition Block	98
STEEL_SPEC	98
Strategy 27, 30, 123, 127, 130, 140, 147, 202, 210, 269, 271, 310, 313, 317, 340, 344, 347, 352, 364, 369, 399, 401, 488, 519	
Structured Text	399
SUB2 block	393
SUM_DIAG block	290
Summary diagnostic block	290
SWITCH block	482
T	
T100	
Characterisation block	258, 535
T225	127
Configuration block	127
T2550	130, 520
Configuration block	130
PROGCHAN	445, 453, 462
PROGCTRL	445, 453, 462
SEGMENT	445, 453, 462
Task Summary block	295
T2900	
Data recording facility	220
DCM block	208
Display navigation	422
Flash memory data recording	422
Multi-setpoint programmer & data logger	422
Preplot	422, 425
T600	117, 310, 318, 324
Configuration block	158
Performance block	293
T600TUNE block	293
T800	
DCM block	208
T820	130
Configuration block	130
Task Summary block	295
T940 block	123
T940(X)	123
Configuration block	123
DCM block	208
Performance block	271
Task diagnostic block	269
Tactician	130
Configuration block	130
TACTTUNE block	295
TAG block	486
Tagname	22, 24, 121, 178, 411, 494
Talk-Thru	47
TAN function blocks	17, 487, 541
Task Diagnostic block	301
Task field	306
TC_LIFE	
Extension block	111
TC_LIFE block	107, 459
TC_LIFE_EX	111

TC_LIFE_EX block 111
TCOUPLE block 70, 118
TCP-master 47
TCP-slave 47
TC_SEL block 113
Teatt 529
Telnet 417
Temperature 82
Tepid Data 120, 126, 158
 connection block 158
 Tepid Data file 126
Thermocouple Life Expectancy block 107, 113
Time
 Broadcasts 297
 Requests 297
 Synchronisation 297
Time/Date event block 501
TIMEDATE block 501
Time-Of-Day Diagnostic block 297
Time-proportioning output block 358, 504
TIMER block 499
Time-sequence 489
TIMING function blocks 488, 541
TOD_DIAG block 297
TOTAL block 495, 514
TOTAL2 block 515
Totalisation block 515
TOT_CON block 48, 52, 514
TP_CONN block 120, 121, 158
TPO block 504
TPO_UIO block 358
TRACK 519
Trend 68, 69, 72, 73, 75, 79, 149, 153, 388, 391, 406, 469, 472
TUNE_SET block 198
Tune_Set block 189, 190, 191
Tuning 198, 522

U

UCHAR block 70, 118, 307, 342
UINTENUM block 204
UnSupAll 133, 138
User Screen Editor 406
 Area/Group/Point agent 406
 Built-in agents 406
 User Screens 436
User Task diagnostic block 299
USERTASK block 299

V

Valve Positioner block 376
Video data recorders 467
Visual Supervisor agents 406
VP_UIO block 376

X

XEC_DIAG block 301
XOR4 block 382
XP 520

Y

Yes 194

Z

Zirconia block 99
 Zirconia Control block 99
Zone 469



Scan for local contents

Eurotherm Ltd

Faraday Close
Durrington
Worthing
West Sussex
BN13 3PL
Phone: +44 (0) 1903 268500
www.eurotherm.co.uk

Schneider Electric, Life Is On, Eurotherm, EurothermSuite, Wonderware, InTouch, eCAT, EFit, EPack, EPower, Eycon, Eyris, Chessell, Mini8, nanodac, optivis, piccolo, and versadac are trademarks of Schneider Electric SE, its subsidiaries and affiliated companies. All other trademarks are the property of their respective owners.

HA082375U003 Issue 33 (CN37603)

© 2019 Schneider Electric. All Rights Reserved.