Chapter 4

FILE SYSTEM

Edition 3

Overview

The FileSystem function block class contains a set of function blocks for the definition and control of the PC3000 File System. The File System is used by the Programmer function blocks for the storage of data and can also be used for simple data logging applications. Facilities are available in the DOS programming tools for uploading and downloading files from/to the PC3000.

FILE STORE FORMAT FUNCTION BLOCK

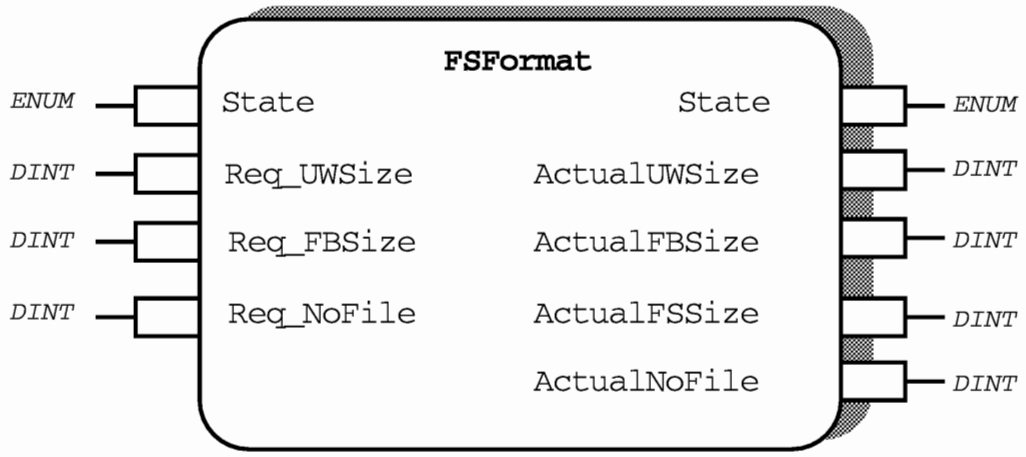

Figure 4-1 File Store Format Function Block

Functional Description

The PC3000 File Store is created in spare RAM space within the LCM. The RAM space within the LCM is also required for user programs, downloaded user source, and downloadable function blocks. Therefore the precise amount of space that can be assigned to the file store depends on the amount of memory required for these other uses, and the amount of RAM fitted to the LCM. Refer to the technical documentation for the LCM for valid memory configurations.

The FSFormat function block allows a file store to be set up within spare RAM. Once this file store has been created it remains intact with all files present until explicitly re-formatted by means of an instance of the FSFormat function block.

The amount of memory required for the user program can be determined from the On-Line Status information provided by the PC3000 Programming Software after a program has been downloaded. Enough space must be allowed for the largest user program that will be downloaded to the PC3000 together with any associated downloadable function blocks that are required. The LCM will go into an idle state when a download is attempted if insufficient space is available for the user program and downloadable function blocks.

Remember that the File Store is only deleted from memory if power is lost to the LCM and the battery is disconnected. Its size may be adjusted (although this means its contents will be deleted) by means of the File Store Format Function Block.

An attempt to create a file store larger than permitted by other memory uses whilst a program is running causes a file store of the largest size possible to be created with the current program and downloadable function blocks remaining intact. However, this would mean that a larger program or one requiring more downloadable function block space could not be downloaded until the file store was reformatted.

Function Block Attributes

Type: D80

Class: ..... FILESYSTEM

Default Task: ..... Task_2

Short List: ..... State, ActualUWSize, ActualFBSize,
..... ActualFSSize.

Memory Requirements: ..... 36 Bytes

Parameter Descriptions

State (S)

When driven as an input, this parameter allows the formatting of the file store to be carried out. It also displays the result of the format operation.

Ok:	The quiescent state of this input/output - either no format action has yet been undertaken, or the last format was succesful.
Write:	Used to initiate a format operation. The State parameter automatically returns to either Ok or Error depending on the success or otherwise of the format operation.
Error:	The last format action was unsuccesful.

Req_UWSize (RUW)

The size of the available RAM that the user wishes to reserve for user programs. If this is less than the current user program size then the current user program size is used.

Req_FBSize (RFB)

The size of the available RAM that the user wishes to reserve for downloadable function block libraries. If this is less than the current ActualFBSize then the current downloadable function block library size is used.

Req_NoFile (RNF)

Permits the user to limit the number of files that can be kept in the file store at any one time.

ActualUWSize (AUW)

The amount of the available RAM that is currently reserved for user programs. This does not represent the actual size of the current user program.

ActualFBSize (AFB)

The amount of the available RAM that is currently reserved for downloadable function block libraries. This does not represent the actual size of the currently downloaded function block library.

ActualFSSize (AFS)

The amount of the available RAM that is currently reserved for the PC3000 File Store.

ActualNoFile (ANF)

The actual number of files that can be kept in the file store at any one time.

Parameter Attributes

Name	Type	Cold Start	Read Access	Write Access	Type Specific Information	
State	**ENUM**	Ok (0)	Oper	Oper	See Parameter List	
Req_UWSize	**DINT**	0	Oper	Oper	High Limit Low Limit	2147483647 0
Req_FBSize	**DINT**	0	Oper	Oper	High Limit Low Limit	2147483647 0
Req_NoFile	**DINT**	0	Oper	Oper	High Limit Low Limit	255 20
ActualUWSize	**DINT**	Memory in LCM, RAM1, RAM2	Oper	Block	High Limit Low Limit	Depends on LCM RAM 0
ActualFBSize	**DINT**	Memory in LCM, RAM3,	Oper	Block	High Limit Low Limit	Depends on LCM RAM 0
ActualFSSize	**DINT**	0	Oper	Block	High Limit Low Limit	Depends on LCM RAM 0
ActualNoFile	**DINT**	0	Oper	Oper	High Limit Low Limit	255 20

Table 4-2 File Store Format Function Block

FILE STORE FILE HANDLING FUNCTION BLOCK

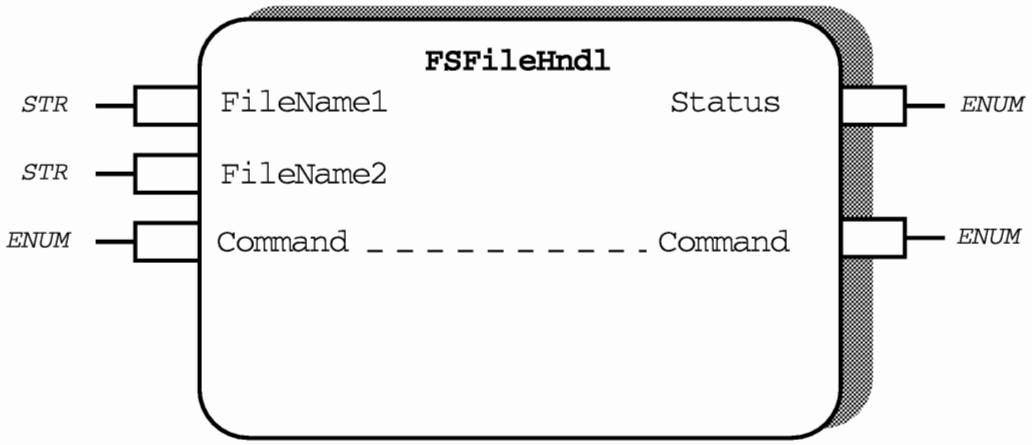

Figure 4-2 File Store File Handling Function Block

Functional Description

This block enables simple file handling operations to be carried out within a pre-formatted file store with files present. To carry out operations which require a knowledge of file names, prior use of FSDirectry function block may be necessary.

Function Block Attributes

Type: D82
Class: .....FILESYSTEM
Default Task: .....Task_2
Short List: .....FileName1, FileName2, Command, Status.
Memory Requirements: .....42 Bytes

Parameter Descriptions

FileName1 (FN1)

The name of the first file to be used in conjunction with Command. The drive letter R: for RAM or E: for ROM must be inserted in front of the file name. Drive and file names are case sensitive.

FileName2 (FN2)

The name of the second file to be used in conjunction with Command if required. Drive and file names are case sensitive.

Command (CMD)

The control parameter for this function block.

Ok: The quiescent state of this input/output - either no file handling action has yet been undertaken, or the last requested action has been completed or aborted.

Copy: Copy the file whose name is given by FileName1 to the file whose name is given by FileName2. If FileName2 does not exist, it will be created. If it does exist its contents will be overwritten. Command will return to Ok when the copy has either completed succesfully or is known to have failed.

Delete: Delete the file whose name is given by FileName1. The filewhose name is given by FileName2 is not affected. Command will return to Ok when the deletion has either completed succesfully or is known to have failed.

CloseAl: All files in the file store are examined to see which files are believed to be open. Any file which is open but does not have a function block currently accessing it is closed.

Status (ST)

This output indicates the status of the current operation or the completion status of the previous operation if no current operation is in progress.

Ok:	Last operation completed succesfully.
Copying:	A copy operation is in progress.
Error:	An undefined error occured in the last operation.
OpenErr:	An error occured when attempting to open a file.
ClsErr:	An error occured when attempting to close a file.
CopyErr:	An error occured when attempting to copy a file.
ReadErr:	An error occured when attempting to read a file.
WrtErr:	An error occured when attempting to write a file.

Parameter Attributes

Name	Type	Cold Start	Read Access	Write Access	Type Specific Information
FileName1	**STR**	"	Oper	Oper	Max 12 char
FileName2	**STR**	"	Oper	Oper	Max 12 char
Command	**ENUM**	Ok(0)	Oper	Oper	See Parameter List
Status	**ENUM**	Ok(0)	Oper	Block	See Parameter List

Table 4-2 File Store File Handling Function Block

FILE STORE FREE SPACE FUNCTION BLOCK

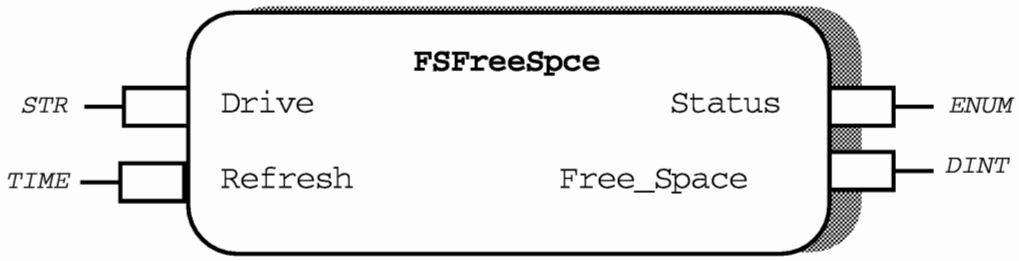

Figure 4-3 File Store Free Space Function Block

Functional Description

This function block provides a monitor on the current free space within the PC3000 File Store. The refresh rate may be varied to limit the overhead of this function block on the system.

Different PC3000 File Store drives may be examined. The only drive types currently valid are 'R', indicating a RAM-based file store and 'E' indicating a ROM based file store.

Function Block Attributes

Type:.....D84

Class: .....FILESYSTEM

Default Task: .....Task_2

Short List: .....Drive, Refresh, Status, Free_Space.

Memory Requirements: .....20 Bytes

Parameter Descriptions

Drive (DRV)

The drive (in other words, the memory type) to be examined. Valid entries are'R', indicating a RAM-based file system and 'E', indicating a ROM-based file system.

Refresh (R)

The frequency with which the memory space is examined to update the Free_Space output.

Status (ST)

Indicates the status of the selected drive.

OK:	The drive has been formatted and is functional.
NoSuprt:	The drive type selected is not recognised or is not valid for the current file system.
NoFile:	Memory is allocated for a file store but it has lost its directory structure. A re-format wil be required.
NoFrmat:	The drive is not formatted. This does not necessarily indicate that the drive could be formatted or is valid for the current system.

Free_Space (FRE)

The amount of space free on the selected drive. This is the space available for new files.

Parameter Attributes

Name	Type	Cold Start	Read Access	Write Access	Type Specific Information	
Drive	**STR**	'R'	Oper	Oper	Only 'R 'and 'E' are currently valid	
Refresh	**TIME**	1s	Oper	Oper	High Limit Low Limit	23d 23h 59m 59s 999ms 0ms
Status	**ENUM**	Ok(0)	Oper	Block	See Parameter List	
Free_Space	**DINT**	Format configuration	Oper	Block	Range depends on LCM memory size	

Table 4-3 File Store Free Space Function Block

FILE STORE ACCESS FUNCTION BLOCK

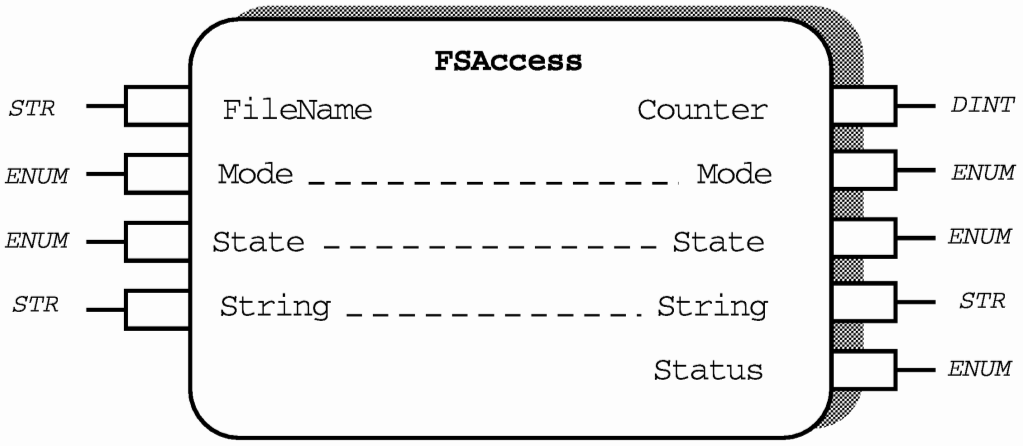

Figure 4-4 File Store Access Function Block

Functional Description

This block allows new files to be created and the contents of existing files to be read or written either one line at a time or one character at a time.

Note: These are two special versions of this block available as downloadable function blocks.

These will search the initial character(s) of each line of the file for:

(a) an alphanumeric

(b) a number

Function Block Attributes

Type: D86

Class: ..... FILESYSTEM

Default Task: ..... Task_2

Short List: ..... FileName, Mode, State, String.

Memory Requirements: ..... 252 Bytes

Parameter Descriptions

FileName (FNM)

The name of the file which will be read or written. The drive letter R: for RAM or E: for ROM must be inserted in front of the file name. Drive and file names are case sensitive.

Mode (M)

Determines the operation which will be carried out by means of the State and String input/outputs.

Ok	The quiescent state of the function block. The State parameter has no effect when mode is set to Ok.
Read	The State parameter can be used to read line by line or character by character from the file. The information read will appear on the String input/output parameter. When the end of the file is reached, Mode will return to Ok automatically - no action by the user program is required to achieve this.
Write	The State parameter can be used to write line by line or character by character to the file. The information to be written is input via the String input/output parameter and will commence with the first line of the file - any information in the file at the start of the Write operation will be overwritten.

Writing can only commence when no other function blocks are accessing the file in question for either read or write. A useful way of checking this is the parameter FSDirectry.FileRdOpen which should be zero before writing can commence.

Append	The State parameter can be used to append line by line or character by character to the file. The information to be appended is input via the String input/output parameter and this will be placed after the information already in the file.

Note that the first character or line to begin a file must be inserted using Mode=Write. Thereafter characters and lines may be appended.

Appending can only commence when no other function blocks are accessing the file in question for either read or write. A useful way of checking this is the parameter FSDirectry.FileRdOpen, which should be zero before appending can commence.

State (S)

Enables information to be read or written from the file whose file name is given by FileName one line or one character at a time.

Ok	The quiescent state of the function block. State returns to Ok automatically. No action by the user program is required.
NextLn	If the Mode parameter is set to Read, this will cause the next line of the file to be read and the result shown at the String input/output. The line terminator (<CR><LF>) will not be shown. If the Mode parameter is set to Write, this will cause the string shown at the String input/output to be written as the next line of the file, and terminated with 'RL' i.e. <CR><LF>.
NextChr	If the Mode parameter is set to Read, this will cause the next character of the file to be read and the result shown at the String input/output. The line terminator (<CR><LF>) will not be shown. If the Mode parameter is set to Write, this will cause the string shown at the String input/output to be written as the next character of the file. A line may be terminated by writing in $R then $L as the last two characters written or appended.

String (STR)

The string value to be written to the file when in Write or Append Mode, or the string value read from the file whilst in Read Mode.

Counter (CNT)

The current line number within the file when using NextLn mode.

Status (ST)

Indicates the success or otherwise of the last operation requested through the State parameter to the file whose name is given by FileName.

Ok	The last operation was succesful.
Opn_Err	There was an error opening the file.
Cls_Err	There was an error closing the file.
EOF	The last operation caused the end of the file to be reached.
LnToLng	The String supplied via the String as an input to be written to the file was too long. Note that when writing using State=NextChr String should only be a single character.
WrtErr	An error occured when attempting to write to the file.

Parameter Attributes

Name	Type	Cold Start	Read Access	Write Access	Type Specific Information	
FileName	**STR**	"	Oper	Oper	Maximum 12 characters	
Mode	**ENUM**	Ok(0)	Oper	Oper	See Parameter List	
State	**ENUM**	Ok(0)	Oper	Oper	See Parameter List	
String	**STR**	"	Oper	Oper	Maximum 80 characters	
Counter	**DINT**	0	Oper	Block	High Limit Low Limit	2147483647 0
Status	**ENUM**	Ok(0)	Oper	Block	See Parameter List	

Table 4-4 File Store Access Parameter Attributes

FILE STORE DIRECTORY ACCESS FUNCTION BLOCK

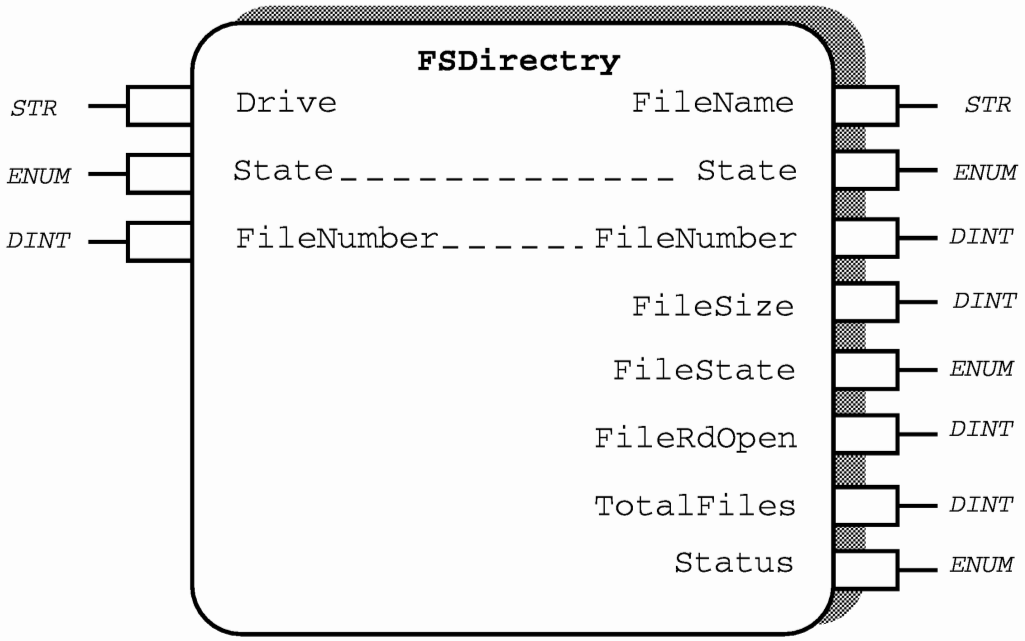

Figure 4-5 File Store Directory Access Function Block

Functional Description

This function block gives the ability to determine what are the names and sizes of the files currently in the file store. The amount of free space in the file store is also given.

Information on the outputs does not update automatically - it must be updated by the user program by means of the State input.

Function Block Attributes

Type: D88

Class: ..... FILESYSTEM

Default Task: ..... Task_2

Short List: ..... Drive, State, FileNumber, FileName

Memory Requirements: ..... 54 Bytes

Parameter Descriptions

Drive (DRV)

Specifies which type of memory to examine for the file store information.

Possible memory types are RAM, designated by 'R', and EPROM, designated by 'E'.

State (S)

This parameter enables the user program to control the examination of the directory on the chosen Drive.

Ok — The quiescent state of the block.

Top — Causes the top of the directory to be examined. The **FileNumber** will be set to zero. and State will return immediately to Ok. No action is required by the user program to achieve this.

If there are files in the file store this top entry will bear the **FileName** <FREE-SPACE> and FileSize will equate to the amount of memory storage available in the file store excluding space already used by files. This is the same figure as is given by FSFreeSpce.**Free_Space**, but not the same as FSFormat.**ActualFSSize.**

If there no files in the file store this top entry will have the **FileName** 'FREE'. **FileSize** will equate to the amount of memory storage available in the file store less the space required to maintain the file store. This is the same figure as is given by FSFreeSpce.**Free_Space,** but not the same as FSFormat.ActualFSSize even when the file store is empty.

ReadNxt — Information about the next entry in the directory list from the file whose FileNumber is indicated is presented on the outputs. State will return immediately to Ok - no action is required by the user program to achieve this. **FileNumber** will automatically increment.

If the directory end has been reached (indicated by the Status) then **FileNumber** will continue to increment but none of the other outputs will change.

ReadAgn — Information about the entry in the directory list whose **FileNumber** is indicated is presented on the outputs. State will return immediately to Ok - no action is required by the user program to achieve this. This allows the ouptuts to be updated with the current state of a particular file.

FileNumber (FNB)

The function block will automatically update this input/output parameter if the State parameter is used to scroll through the directory using ReadNxt.

FileNumber may however be set to a specific value and details about the the chosen file displayed on the outputs by setting State to ReadAgn.

FileName (FNM)

The name of the file last updated by the actions of the State and **FileNumber** inputs. Four built-in file names are used:

" No file store is set up for this Drive.

'FREE' No files are in the file store

'<FREE-SPACE>' The top of the directory is being examined. **FileSize** will show the amount of free space for files on the Drive indicated, and FileState will show Unused.

'<END-OF_DIR>' **FileNumber i**s the number of actual files plus one, and Status shows DirEnd.

Apart from these, file names will appear for files that have been loaded by the user or user program.

FileSize (FSZ)

The size in bytes of the file indicated by **FileName.**

FileState (S)

Indicates the current state of the file indicated by **FileName.**

Unused This space is not currently used by a file - normally only seen at the top and end of the directory.

Closed	The file is closed - no other function block is accessing the file.
WrtOpen	A function block is writing to the file, or the file being accessed over a comms link.
RdOpen	One or more function blocks are reading from the file. The number of function blocks currently reading from the file is shown by **FileRdOpen.**
RdWtOpn	One or more function blocks are reading from the file and a function block is writing to the file. The number of function blocks currently reading from the file is shown by **FileRdOpen.**
NoSpce	An error has occured in the file system due to the file store being full.
NoSpCls NoSpWrt NoSpRd NoSpRdWt	[Low level system errors - Please report to Eurotherm Controls Limited.]

FileRdOpen (FRO)

The number of function blocks currently reading from the file whose name is given by FileName.

TotalFiles (TFL)

The total number of files in the directory the last time this function block read the directory from the top.

Status (ST)

Indicates the completion status of the last operation requested by State.

Ok	The last operation completed succesfully
DirEnd	The last operation caused a read to the end of the directory.

Error	An error occured in attempting the last operation. The outputs will not be showing reliable information.

Parameter Attributes

Name	Type	Cold Start	Read Access	Write Access	Type Specific Information	
Drive	**STR**	'R'	Oper	Oper	Only 'R' and 'E' are currently valid	
State	**ENUM**	Ok(0)	Oper	Oper	See Parameter List	
FileNumber	**DINT**	0	Oper	Oper	High Limit Low Limit	2147483647 0
FileName	**STR**	"	Oper	Block	Max 12 characters	
FileSize	**DINT**	"	Oper	Block	High Limit Low Limit	2147483647 0
FileState	**ENUM**	Unused(0)	Oper	Block	See Parameter List	
FileRdOpen	**DINT**	0	Oper	Block	High Limit Low Limit	30 0
TotalFiles	**DINT**	0	Oper	Block	High Limit Low Limit	2147483647 0
Status	**ENUM**	Ok(0)	Oper	Block	See Parameter List	

Table 4-5 File Store Directory Access Parameter Attributes