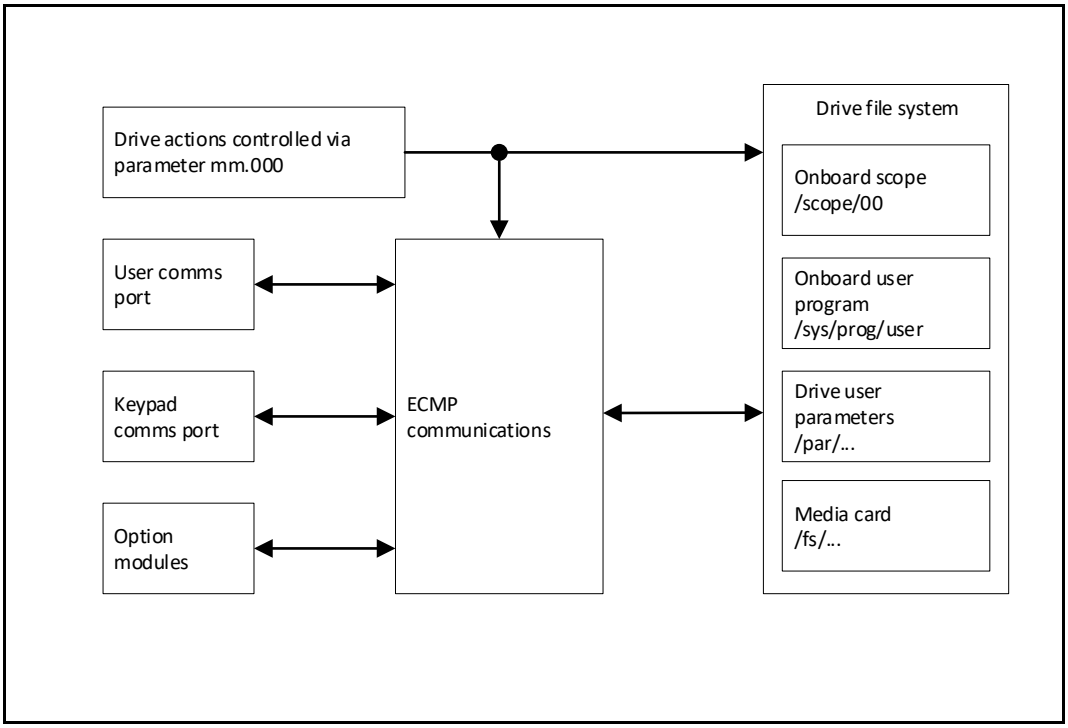
File System, SD and SMART Cards

Last updated: 11/11/2021

Introduction

The drive file system is used to access various files within the drive or on a media card (SD or SMART card) inserted into the drive.



Drive actions controlled by parameter mm.000 (parameter zero in any menu) can directly access the drive file system and can transfer parameters between each of the elements within the file system, i.e. from drive user parameters to a media card. ECMP requests can be sent via the user comms port (if present) and the keypad port to the file system to access each of the elements within the file system. This feature is also provided internally within the drive for option modules so that these can also access files within the drive. Drive actions controlled by parameter mm.000 can also create EMCP requests which allows this system to communicate with option modules. This feature is used to transfer parameter difference files to/from option module internal menus so that they can be appended to drive parameter differences files to create back-up files that contain both drive and option module parameters.

The following table shows the extent of the file system covered by this document.

File name	Type	Function	Access	Transfer methods
/par/all	Raw parameter file	Format and value of all user parameters in the drive.	Read-only	Drive comms
/par/NN	Raw parameter file	Format and value of all user parameters in one drive menu defined by NN.	Read-only	Drive comms
/par/diff	Parameter difference file	Values of any user parameters in the drive that can be copied (NC not copied attribute is NOT set) and are not at their default value or do not have a default. If the file is transferred to a media card by the drive, then option module internal menus are included. When this type of file is written to the drive user parameter defaults are loaded and the drive mode changed if required before the parameter values are written.	Read/write	Drive comms Drive actions controlled via parameter mm.000
/par/boot	Bootable parameter difference file	This is the same as a parameter difference file except that *Parameter Cloning* (11.042) is set to 4 in the file header. This is intended to be used internally within the drive to create a bootable file on a media card.	Read/write	Drive comms Drive actions controlled via parameter mm.000
/par/macro	Parameter macro file	Holds the same parameter data from the drive as a /par/diff file but does not include any parameter data from option module internal menus. When this type of file is written to the drive user parameter defaults are not loaded and the drive mode is not changed before the parameter values are written.	Read/write	Drive comms Drive actions controlled via parameter mm.000
/scope/00	Scope file	Onboard scope file.	Read-only	Drive comms Drive actions controlled via parameter mm.000

/sys/prog/user	User program file	User program files are a binary image of a PLC program stored in the drive. The file contents are beyond the scope of this document.	Read/write	Drive comms Drive actions controlled via parameter mm.000
/fs/…		SMART card or SD card file. The base path for all SMART card files is /fs/ with no subdirectories. The base path for all SD card files is /fs/ to give access to the file system structure on the card.	Read/write	Drive comms

All file types can be accessed via drive communications using ECMP. See the Parameter Reference Guide Communications document for details of the protocol used to perform file transfers. It should be noted that ECMP can be used with the communications interfaces on a drive (user port, if present, or keypad port) or it can be used between the drive and option modules via internal communications interfaces. For example, an applications module could write or read files to/from a media card using ECMP via an internal port within the drive. **Throughout this document where data transfer via comms is referred to, this means data transfer via physical communications ports on the drive or between the drive and option modules via an internal port.** The file paths given in the table above are those used to access the files via comms.

All file types shown with the media card transfer method can be transferred between the drive and a media card inserted into the drive. The transfer is initiated by entering a code into parameter zero in any menu (mm.000) and triggering a drive reset. The required file, related to the code used, is then transferred either to or from the media card. Further details are given later.

/par/diff files are used to save and restore user parameters that are in the drive menus, including option set-up menus (15 to 17 and 24) and option applications menus (25 to 28) when the file is accessed via a comms. When a /par/diff file is transferred to/from a media card by a drive action controlled via parameter mm.000 additional data is appended to the file so that it includes parameter differences for internal menus of any option module fitted to the drive which supports this feature.

All data is stored in files in Big Endian format (most significant byte first).

A 32-bit CRC is used in some files and is referred to as CRC32. This is defined by IEEE Std 802.3-2002 (Section 1) LOCAL AND METROPOLITAN AREA NETWORKS, section 3.2.8: Frame Check Sequence (FCS) Field. It uses an initial value of 0xFFFFFFFF and reverse polynomial 0xEDB88320.

An indication is provided on the drive keypad to show when a media card fitted to the drive is being accessed. This indication is given when a media card file is open whether it has been opened via comms or whether because of a media card transfer initiated from parameter mm.000.

When modifying an SD card in an external device (i.e. a PC), including formatting the card, it is important to eject the card before removing it. The drive does not carry out extensive error checking on the card and the card may appear to be read and written correctly by the drive even if it is corrupted by removing it from the external system. However, it may not work correctly when reinserted into the external system and the data structure on the card may be incorrect.

/par/ files

Files based on the /par/ path are used to transfer drive user parameters. (As mentioned previously /par/diff files can include option module internal menu parameters when transferred to/from a media card by drive actions controlled by parameter mm.000).

/par/all

This type of file is intended to allow the attributes and values of all user parameters in a drive to be read. It is not possible to write this type of file to a drive. The file has a header, followed by a parameter data block for each user parameter starting from parameter 00.000 going up each menu in turn to the highest parameter in the last menu (Menu 41).

File sections

Header
Drive parameter data

Header

Byte addresses	Data	Function
0-2	“RPF”	File type is “Raw Parameter File”
3	0	ECMP parameter addressing scheme is always 0.

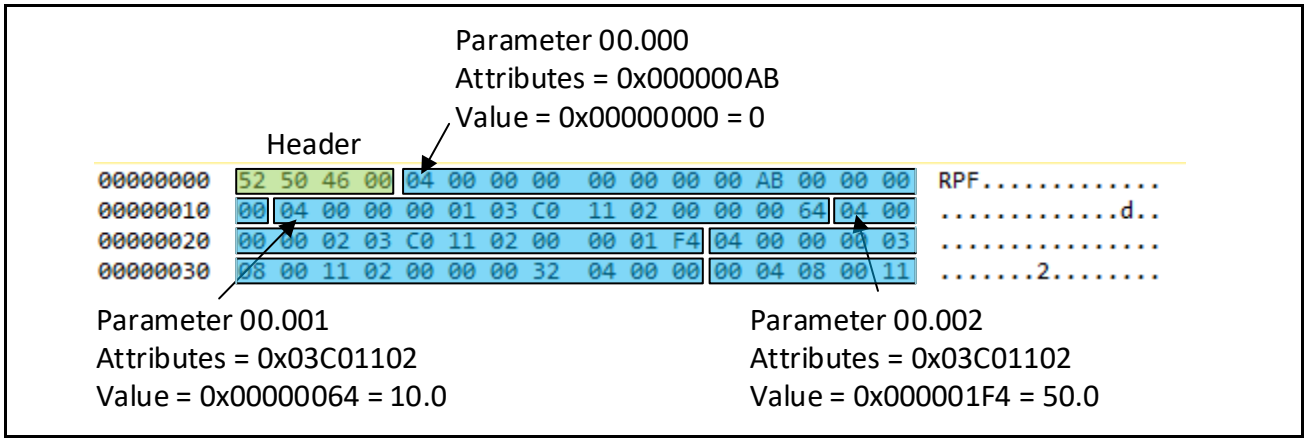
Drive parameter data for one parameter

Byte addresses	Data	Function
0	4	Number of bytes used to represent the parameter value is always 4.
1-2	Menu number	
3-4	Parameter number	
5-8	Parameter Attributes	See below.
9-12	Parameter value	Signed 32-bit value. 1-bit parameters are zero extended. 8 and 16-bit parameters are sign extended if their BU attribute is 0, otherwise they are zero extended.

Parameter attributes

Bit	Code	Function
31-30		Not assigned
29-22	UNITS	Keypad units
21	FL	Floating point value (always 0)
20	DF3	Keypad display format parameter
19	DF2	
18	DF1	
17	DF0	
16	PR	Pseudo read-only
15	FI	Filtered when displayed by keypad
14	DE	Destination set-up parameter
13	TE	String parameter
12	VM	Variable maximum and minimum
11	DP3	Number of decimal places
10	DP2	
9	DP1	
8	DP0	
7	ND	Parameter has no default
6	RA	Voltage or current rating dependant
5	NC	Not copied
4	NV	Not visible
3	PT	Protected from destinations
2	NR	Read not allowed
1	W	Write allowed
0	BU	Bit default/Unipolar

The example below shows the first 64 bytes of a /par/all file.



/par/NN
This type of file is intended to allow the attributes and values of all user parameters in one menu in a drive to be read. It is not possible to write this type of file to a drive. The file format is the same as a /par/all file, but the file only contains the parameters in the defined menu starting from parameter NN.000 going up the menu to the highest parameter number.

/par/diff (or /par/boot)
This type of file (Parameter Difference File) is intended to be used to save and restore the user parameters in a drive, or transfer parameters from one drive to another. This can be done via comms, or by saving and restoring the file to/from a media card with parameter mm.000. Note that a /par/boot file is identical to a /par/diff file except that *Parameter Cloning* (11.042) in the header is set to 0 for a /par/diff file and to 4 for a /par/boot file, and a /par/boot file is read-only via comms.

File sections

Header
Drive parameter data
Option Slot S delimiter
Option Slot S data header
Option Slot S parameter data
Option Slot T delimiter
Option slot T data header
Option Slot T parameter data

All the sections related to option modules are only present if the file is stored on a media card (and not if obtained via comms). The file sections shown above include two option modules sections. There can be between 0 and 4 option module sections as these are only created if an option module is present and supplies its parameter data when requested.

Header

Byte addresses	Data	Function
0-2	“PAR”	File type is “Parameter”
3	0	ECMP parameter addressing scheme is always 0.
4-7	*NV Media Card Required Version* (11.077)	
8-11	*Drive Mode* (11.084)	
12-15	*Parameter Cloning* (11.042)	0 in a /par/diff file or 4 in a /par/boot file.
16-19	*Maximum Heavy-Duty Rating* (11.032)	
20-23	*Maximum Rated Current* (11.060)	
24-27	*Full Scale Current* (11.061)	
28-31	*Drive Rated Voltage* (11.033)	
32-35	*Defaults Previously Loaded* (11.046)	
36-39	*Drive Derivative* (11.028)	
40-43	*Module ID* (15.001)	
44-47	*Module ID* (16.001)	
48-51	*Module ID* (17.001)	
52-55	*Module ID* (24.001)	
56-59	*Product Type* (11.063)	Product type is always 0.
60-71	Reserved.	
72-79	Not used (all zeros)	

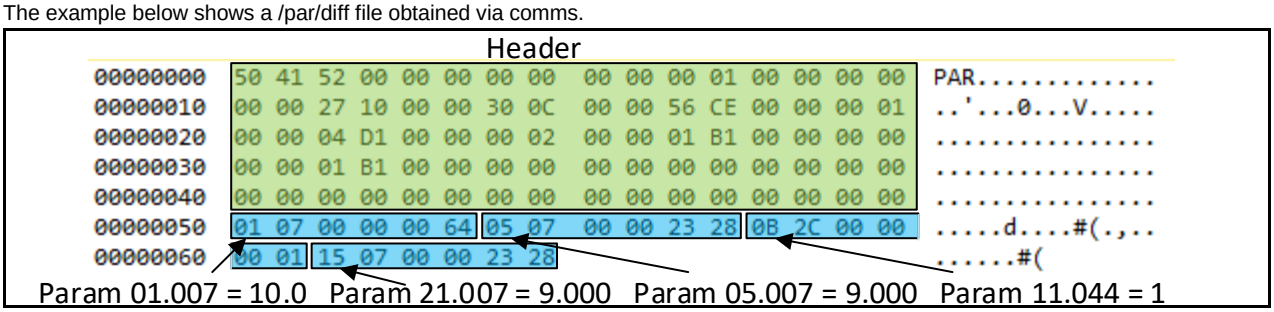
Drive parameter data for one parameter

Byte addresses	Data	Function
0	Menu number	
1	Parameter number	
2-5	Parameter value	Signed 32-bit value. 1-bit parameters are zero extended. 8 and 16-bit parameters are sign extended if their BU attribute is 0, otherwise they are zero extended.

The parameter data can be stored in any order. Menu 0 parameters are duplicates of other parameters, and so only the value of parameters in Menus 1 to 41 are included in the file. Parameters are only stored if the not copied (NC) attribute is NOT set. They must either be different from their default value or NOT have a default value (ND attribute set). From V01.22.00.00 firmware onwards *Rated Current* (05.007) and *M2 Rated Current* (21.007) are always included even if they have their default values. This ensures that the motor rated current is correct if the source and destination

drives are different sizes and the motor rated current was set to the default value in the source drive. If there is no option module data, the file can end after the last drive parameter data, however if the file is required to end on a 16-byte page boundary (i.e. it is being stored on a SMART card), any additional data must be zeros.

The example below shows a /par/diff file obtained via comms.



Option slot delimiter

Byte addresses	Data	Function
0-2	Zeros	As Menu 0 parameters are not included in the file the first zero, which would be the menu number for the next parameter, can be used to detect the start of this delimiter.
3	Option slot number	
4-5	Data size	Combined size of the following option slot data header section and option slot parameter data sections in bytes.

Above is the first option slot delimiter after the drive parameter data. The number of zeros is adjusted in any subsequent option slot delimiter so that the option slot number is at the same location within a page of 16 bytes as the option slot number in the previous delimiter.

Option slot data header

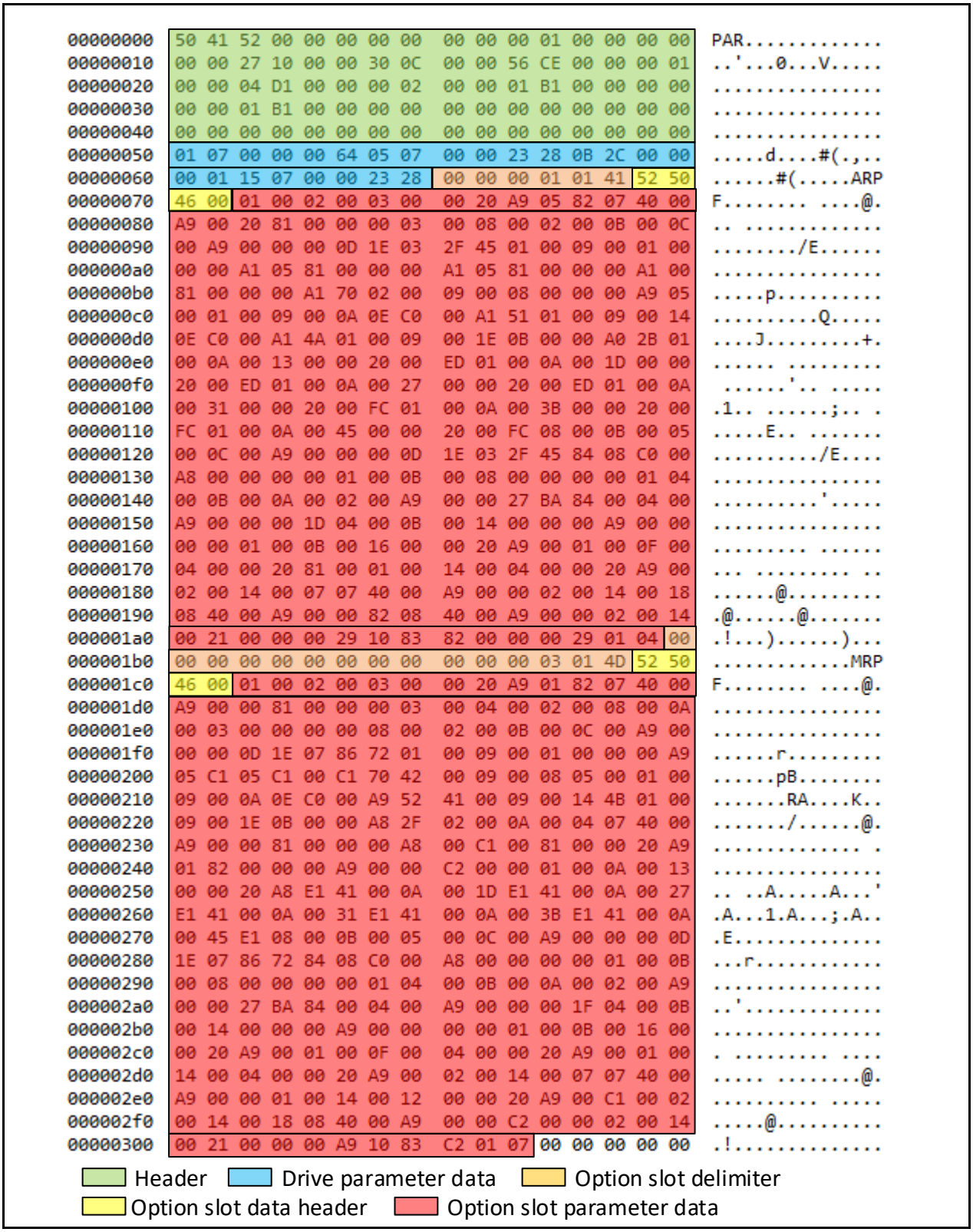
Byte addresses	Data	Function
0-2	“RPF”	File type is “Raw parameter File”. This is different to an RPF file produced by a drive.
3	0	ECMP parameter addressing scheme is always 0.

Option slot parameter data for one parameter

Byte addresses	Data	Function
0	Data format	See below.
1-2	Menu number	
3-4	Parameter number	
5-8	Parameter attributes	See /par/all file description.
9 onwards	Data	The size of the data is specified in the data format.

The data format is defined as follows. Bit 7 indicates that the menu and parameter number are omitted, and that this parameter has the same menu number as the previous parameter, but with the parameter number incremented by one. Bit 6 indicates that the parameter attributes are omitted because they are the same as the previous parameter. Bits 3-0 give the size of the parameter in bytes. The file can end after the last parameter data for the last option slot. However, if the file is to be extended (i.e. to the end of a 16-byte page on a SMART card) the remainder of the file should be zeros.

The example below shows a /par/diff file stored on a media card when a drive has two option modules fitted in Slots 1 and 3 which both provide difference from default data.



The following sequence is followed when this type of file is read from a drive via comms or transferred by the drive to a media card.

1. The header is created using the relevant parameter values read from the drive.
2. An entry is added for each drive parameter that does not have the NC (not copied) attribute set and fulfils the following conditions: Is different from default or does not have a default or the parameter is *Rated Current* (05.007) and *M2 Rated Current* (21.007). It should be noted that a parameter does not need to be a User Save or Power-down Save parameter for an entry to be included in the file. This means for example, that entries can be provided for Menu 20 parameters.
3. If the file is being written to a media card by the drive, it will then request difference from default data from each option module that is fitted. If the module supplies this data, it is appended to the end of the file.

The following sequence is followed when this type of file is written to the drive via comms or transferred by the drive from a media card.

1. If any of the following are true, then if the file is being written via comms an “Incompatible” file status is returned when the file is opened, or if the file is being read from a media card by the drive a “Card product” failure is detected, and the process is aborted: The *Product Type* (11.063) stored in the file header is different to the value in the drive (i.e. not 0). The product identifier in the card header of a SMART card indicates this file was created by another product (i.e. not 5). The product identifier in the /fs/MCDF/000 file on an SD card indicates this file was created by another product (i.e. not 5).
2. If *Drive Mode* (11.084) stored in the file header is different to the actual drive mode, then the drive will attempt to change to the new mode. If the new drive mode is invalid then if the file is being written via comms an “Incompatible” file status is returned, or if the file is being read from a media card by the drive a “Card file data” failure is detected, and the process is aborted.
3. The drive will then attempt to load the defaults defined by *Defaults Previously Loaded* (11.046) from the file header. The value for the defaults previously loaded is the value that would be written to parameter mm.000 to load the required defaults. If the value from the header is not valid then if the file is being written via comms an “Incompatible” file status is returned, or if the file is being read from a media card by the drive a “Card file error” failure is detected, and the process is aborted.
4. Each of the drive parameter data sections is then used to write the value from the file to the relevant parameter. If the parameter does not exist, it is not written. If the parameter value is outside the allowed range, it is still written but is limited by the range. If the parameter has the RA (voltage rating dependent) attribute set and *Drive Voltage Rating* (11.033) in the file header is different to the value in the drive the parameter is not written. (See table below of voltage rating dependent parameters.) If the *Full-Scale Current Kc* (11.061) stored in the file header is different to the value for the drive, then the speed and current loop gains will be scaled if they are not at their default values (i.e. included in the file). See table below for list of gain parameters and the applied scaling. Option set-up or applications menu parameters (Menu 15-17 and 24-28) in the file are only written to the drive if the option module fitted in each option slot has the same module ID as the corresponding module ID in the file header. During this phase the drive position feedback interfaces are held in the un-initialised state so that loading defaults and modifying their parameters do not initiate trips. Once this phase is complete the position feedback interfaces are then allowed to initialise themselves. (Note that the file cannot contain Menu 0 parameters: if the file is being written via comms, and the menu and parameter number are zero the returned file status value from the write is a “custom success” code of 0x6E if the parameter value is zero, or 0x6F if the parameter value is non-zero. All the parameters are written up to the point where the entry with the menu and parameter number of zero is detected.)
5. If the file is being written via comms the process is now complete. If *Drive Voltage Rating* (11.033), *Drive Derivative* (11.028) or the module ID’s in the file header are different from those in the drive, the returned file status from the file close request is a “custom success” code of 0x70 ORed with the following bits to indicate what is different: Bit 0 - *Drive Voltage Rating* (11.033), Bit 1 - *Drive Derivative* (11.028), Bit 2 - module ID. Otherwise the returned file status is “Okay”.
6. If the file is being read from a media card by the drive the process continues. If the module ID of any module fitted in a drive option slot is the same as the corresponding module ID in the file header and there is additional option parameter data in the file for an option fitted in that slot, an attempt is made to send the data to the option module. If this process fails a “Card slot” failure is detected and the transfer of data to the option module is aborted.
7. A parameter save is initiated, and so the parameters that have been written to the drive are saved in drive EEPROM. Option modules will also save their internal menu parameters.
8. The process is now complete, failures can be detected at this point to act as a warning that the some of the data may not have been transferred from the media card to the drive as follows. “Card rating” failure indicates that *Drive Voltage Rating* (11.033) in the file header is different to the value in the drive. “Card option” failure indicates that at least one option module fitted to the drive has a different module ID to the relevant value in the file header. “Card derivative” failure indicates that *Drive Derivative* (11.028) in the file header is different to the value in the drive. These failures will cause trips, but these may be suppressed if the warning suppression flag on the card is set. See later section on media card storage and recovery for details of this flag and how it can be set or cleared.

Parameters with RA (voltage rating dependent) attribute set.

User parameters	**Modes**
Standard Ramp Voltage (02.008)	Open-loop, RFC-A and RFC-S
Voltage Set-point (03.005)	Regen
Regen Supply Loss a.c. Level (03.023)	Regen
Regen Minimum Voltage (03.026)	Regen
Regen Maximum Voltage (03.027)	Regen
Supply Voltage (03.028)	Regen
Supply Loss Detection Level (06.048)	Open-loop, RFC-A and RFC-S
Standard Under-voltage Threshold (06.065)	All
Low Under-voltage Threshold (06.066)	All
Braking IGBT Lower Threshold (06.073)	All
Braking IGBT Upper Threshold (06.074)	All
Low Voltage Braking IGBT Threshold (06.075)	All

Scaling applied when *Full Scale Current Kc* (11.061) in the file header is different to the value in the drive when writing a file to the drive.

User parameters	**Scaling**
Speed Controller Proportional Gain Kp1 (03.010) *Speed Controller Integral Gain Ki1* (03.011) *Speed Controller Proportional Gain Kp2* (03.013) *Speed Controller Integral Gain Ki2* (03.014) *M2 Speed Controller Proportional Gain Kp* (21.017) *M2 Speed Controller Integral Gain Ki* (21.018)	Kc from file header / Kc for drive
Current Controller Kp Gain (04.013) *Current Controller Ki Gain* (04.014) *M2 Current Controller Kp Gain* (21.022) *M2 Current Controller Ki Gain* (21.023)	Kc for drive / Kc from file header

/par/macro

This type of file (Macro File) is intended to be used to write a limited number of parameters to a drive to set it up for a specific application. The differences between a /par/diff file and a /par/macro file are as follows.

1. The first 3 bytes of the header are “MAC” instead of “PAR”.
2. Only the header and drive parameter data (and not option module internal menu data) are included in the file even if written to a media card by the drive.
3. When this type of file is written to a drive the drive mode is not changed and defaults are not loaded before the parameter data is used to modify the user parameters.
4. Parameters are only stored in this type of file if the not copied (NC) attribute is NOT set and the parameter is not at its default value. Note that parameters that have no default are not stored in this type of file.

/scope/ files

The on-board scope function (see Menu 9) produces a file that can be read via comms or transferred to a media card. Normally the full file path via comms is /scope/00, but any path that starts /scope/ with more than 7 characters will access this file.

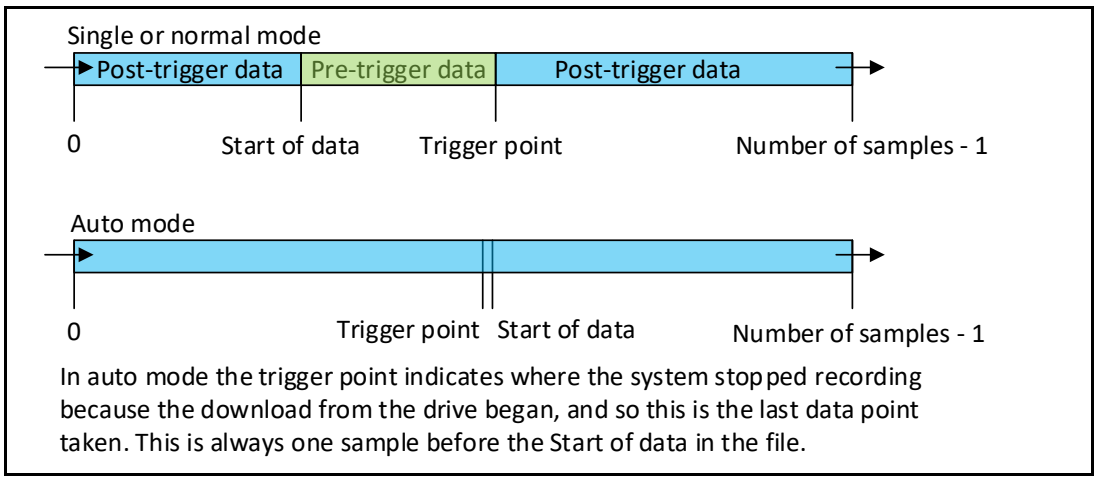
File sections

Header
Trace 1 header
Trace 1 data
Trace 2 header
Trace 2 data
Trace 3 header
Trace 3 data
Trace 4 header
Trace 4 data
Additional data

Header

Byte addresses	Data	Function
0-3	File size	File size in bytes
4-7	CRC	CRC32 applied to the whole file excluding the file size and CRC.
8-11	Start of data	Location of the oldest data within each trace.
12-15	Trigger point	Location of data within each trace where the trigger was detected.
16-19	Number of samples	Number of data samples in each trace.
20-23	Time base	Time base for samples in micro-seconds.
24-27	*Date* (06.016)	Date at the trigger point.
28-31	*Time* (06.017)	Time at the trigger point.
32-35	Recording mode	0 = single, 1 = normal, 2 = auto.
36-39	4	Number of traces is always 4.
40-43	1	File format is always 1.
44-47	*Serial Number MS* (11.053)	Drive serial number.
48-51	*Serial Number LS* (11.052)	
52-55	Start of additional data area	Offset from the start of the file to the start of the additional data in bytes.
56-63	All zeros	Reserved for future use.

The following diagram shows how start of data and trigger point relate to the data.



Trace N header

Byte addresses	Data	Function
0-3	Trace size	Number of bytes in trace including this header.
4-5	Size of trace data point	0: 32-bit signed, 1: 32-bit unsigned, 2: 16-bit signed, 3: 16-bit unsigned, 4: 8-bit signed, 5: 8-bit unsigned.
6-7	Decimal places	Decimal places of the user parameter being monitored.

If there is no data in the trace the 8-byte header is still present and filled with zeros except the trace size which is 8, indicating a header but no data.

Trace N data

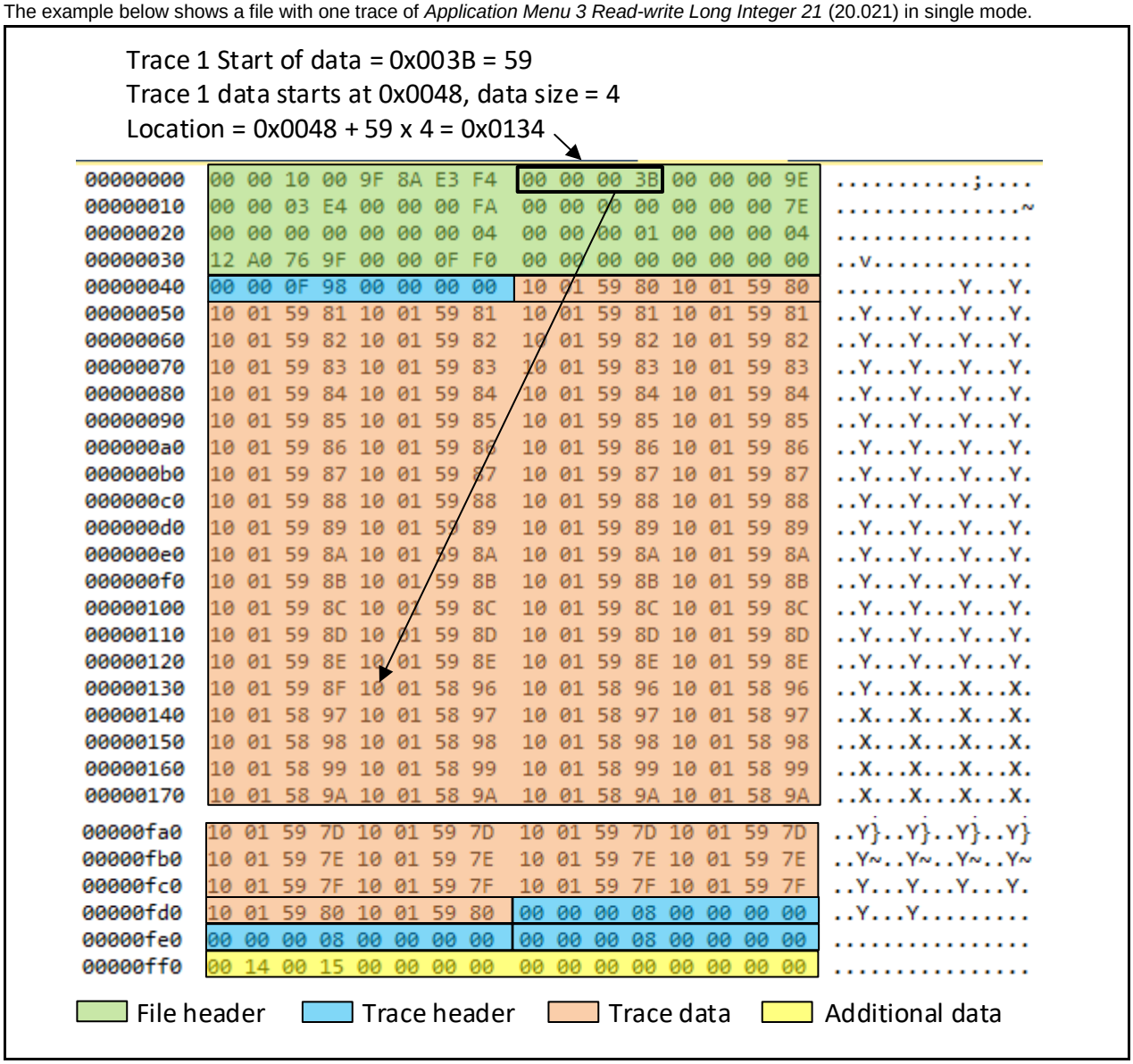
Byte addresses	Data	Function
0	Start of trace data	
---	---	
Trace size – 1	End of trace data	

Additional data

Byte addresses	Data	Function
0	0	Trace 1 menu/parameter numbers.
1	Trace 1 Menu	
2-3	Trace 1 Parameter	
4	0	Trace 2 menu/parameter numbers.
5	Trace 2 Menu	
6-7	Trace 2 Parameter	
8	0	Trace 3 menu/parameter numbers.
9	Trace 3 Menu	
10-11	Trace 3 Parameter	
12	0	Trace 4 menu/parameter numbers.
13	Trace 4 Menu	
14-15	Trace 4 Parameter	

The additional data is present for each trace even if the trace is not present, but the data is all zeros.

The example below shows a file with one trace of *Application Menu 3 Read-write Long Integer 21* (20.021) in single mode.



/sys/ files

/sys/prog/user
This file path is used for user program files which are a binary image of a PLC program stored in the drive. The file contents are beyond the scope of this document.

/fs/ files

The file system on a media card is accessed via comms through the /fs/ folder which is a virtual folder. There is no folder structure on a SMART card, but a series of file blocks which all appear as though they are in the /fs/ folder. The /fs/ folder on an SD card corresponds to the root of the card. If the card is read directly by a PC all the folders and files in the /fs/ folder appear in the root folder.

/fs/ on a SMART card

/fs/ is the base path for all files stored on a SMART card. This folder cannot include any sub-folders and should only contain Parameter Difference Files or User Program Files. An example of the file structure is given below.

```
/fs
  /001  Open-loop Parameter Difference File
  /002  User Program File
  /007  RFC-S Parameter Difference File
  /501  Open-loop Parameter Difference File
  /999  Regen Parameter Difference File
```

/fs/ on an SD card

The file system on an SD card is accessed via comms through the /fs/ folder which contains sub-folders. If an SD card is inserted into a drive and the folder/file structure shown below does not exist, it is automatically created. If any part of this process fails, the drive does not indicate a failure or initiate a trip, but it is likely to give a failure indication if any further operations are performed on the card. For an SD card to be used with a drive it must be formatted using FAT32. All folder and file names will be upper case and the file names will conform to 8.3 format. It is possible to include other data on the card if required, but this must be outside the MCDF folder.

```
/fs
  /MCDF
    /DATA
      /S1
      /S2
      /S3
      /S4
    /KEYPAD
    /SCOPE
    /000
```

The /MCDF/KEYPAD and /MCDF/DATA folders are not used by the drive. The /MDCF/SCOPE folder is used by the onboard scope auto-save function to write Scope Files starting with SCP00XY.DAT where XY is the current value of *Scope Auto-save File Number* (09.071).

/000 is a binary file containing 16 byes with the functions given below which relate to the whole of the MCDF folder.

Byte addresses	Function
0	Read-only flag.
1	Warning suppression flag.
2	Product Identifier. Always 5 indicating Unidrive M and Digitax HD products.
3 - 15	Not used, always zero.

The root of the /MCDF/ folder can also contain files. These should only be Parameter Difference Files or User Program Files. An example of the file structure is given below.

```
/fs
  /MCDF
    /DATA
    /KEYPAD
    /SCOPE
    /000
    /001  Open-loop Parameter Difference File
    /002  Drive user Program File
    /007  RFC-S Parameter Difference File
    /501  Open-loop Parameter Difference File
    /999  Regen Parameter Difference File
```

Read-only flag

The read-only flag stored in the card header on a SMART card or in file /000 on an SD card makes the card read-only when the drive attempts to write to the card in response to parameter mm.000 commands, but not when accessed via comms. Also, files numbered 500 or more are read-only when the drive attempts to delete or replace them in response to parameter mm.000 commands, but not when deleted via comms. When the read-only function prohibits an action, it will cause a “Card Read-only” failure to be detected.

Media Card Storage and Recovery

Storage and recovery actions

The table below gives the parameter mm.000 values used to initiate data transfers between a drive and a media card. The process is started by setting up the value and initiating a drive reset. Other parameter mm.000 values that are not related to media card data transfer are not included. When the required action is completed successfully parameter mm.000 is reset to zero.

mm.000 parameter value	Function	Description	Possible failure indications
2	Load file 1	Transfers /fs/001 from a SMART card or /fs/MCDF/001 from an SD card to the drive. Equivalent to 6001.	See 6XXX.
3	Save to file 1	Transfers drive parameters to parameter difference file /fs/001 on a SMART card or /fs/MCDF/001 on an SD card. Equivalent to 4001.	See 4XXX.
4	Load file 2	Transfers /fs/002 from a SMART card or /fs/MCDF/002 from an SD card to the drive. Equivalent to 6002.	See 6XXX.
5	Save to file 2	Transfers drive parameters to parameter difference file /fs/002 on a SMART card or /fs/MCDF/002 on an SD card. Equivalent to 4002.	See 4XXX.
6	Load file 3	Transfers /fs/003 from a SMART card or /fs/MCDF/003 from an SD card to the drive. Equivalent to 6003.	See 6XXX.
7	Save to file 3	Transfer drive parameters to parameter difference file /fs/003 on a SMART card or /fs/MCDF/003 on an SD card. Equivalent to 4003.	See 4XXX.
15	Update file 1	Transfers drive parameters to parameter difference file /fs/001 on a SMART card or /fs/MCDF/001 on an SD card and overwrites the file if it exists. Equivalent to 3001.	See 3XXX.
16	Update file 2	Transfers drive parameters to parameter difference file /fs/002 on a SMART card or /fs/MCDF/002 on an SD card and overwrites the file if it exists. Equivalent to 3002.	See 3XXX.
17	Update file 3	Transfers drive parameters to parameter difference file /fs/003 on a SMART card or /fs/MCDF/003 on an SD card and overwrites the file if it exists. Equivalent to 3003.	See 3XXX.
18	Clear warn flag	Clear the media card warning suppression flag. Equivalent to 9555.	See 9555.
19	Set warn flag	Set the media card warning suppression flag. Equivalent to 9666.	See 9666.
20	Clear read-only	Clear the media card read-only flag. Equivalent to 9777.	See 9777.
21	Set read-only	Set the media card read-only flag. Equivalent to 9888.	See 9888.
22	Erase card	Erases a SMART card or resets the fs/MCDF/ folder on an SD card. Equivalent to 9999.	See 9999.
2001	Save parameter boot file	Create a /par/boot file as /fs/001 on a SMART card or /fs/MCDF/001 on an SD card. This has the effect of creating a parameter difference file which can be used to boot the drive parameters on power-up. If file 001 already exists on the card, then no action is taken and parameter mm.000 is not reset to zero.	Card busy, Card data exists, Card full, Card error, Card product, Card read-only, Card slot S
2002	Save full boot file	Create a user program file /fs/002 on a SMART card or /fs/MCDF/002 on an SD card. If there is no user program in the drive, then no action is taken and parameter mm.000 is not reset to zero. If the user program file is created successfully, then create a /par/boot file as /fs/001 on a SMART card or /fs/MCDF/001 on an SD card. This has the effect of creating parameter difference and user program files which can be used to boot the drive parameters and user program on power-up. If either file 001 or 002 already exists, then no action is taken and parameter mm.000 is not reset to zero.	Card busy, Card data exists, Card full, Card error, Card product, Card read-only, Card slot S, Card user prog
2003	Update parameter boot file	Equivalent to 2001 except that the file is over-written if it already exists.	Card busy, Card full, Card error, Card product, Card read-only, Card slot S
3001 to 3499	Save parameters to file XXX with over-write	3XXX transfers drive parameters to parameter difference file /fs/XXX on a SMART card or /fs/MCDF/XXX on an SD card and overwrites the file if it exists.	Card busy, Card full, Card error, Card product, Card read-only, Card slot S
4001 to 4499	Save parameters to file XXX	4XXX transfers drive parameters to parameter difference file /fs/XXX on a SMART card or /fs/MCDF/XXX on an SD card. If the file already exists, then no action is taken and parameter mm.000 is not reset to zero.	Card busy, Card data exists, Card full, Card error, Card product, Card read-only, Card slot S
5001 to 5499	Save user program to file XXX	5XXX transfer a drive user program to user program file /fs/XXX on a SMART card or /fs/MCDF/XXX on an SD card. If the file already exists, then no action is taken and parameter mm.000 is not reset to zero. If there is no user program in the drive, then no action is taken and parameter mm.000 is not reset to zero	Card busy, Card data exists, Card full, Card error, Card product, Card read-only, Card user prog
6001 to 6999	Load file XXX	6XXX transfers /fs/XXX from a SMART card or /fs/MCDF/XXX from an SD card to the drive. This action is completed whether the file is a parameter difference file or a user program file. If the drive is already active when it is reset to start the transfer, no action is taken and parameter mm.000 is not reset to zero. If the drive is enabled during the transfer process a “Data Changing” trip is initiated by the drive.	Card busy, Card derivative, Card error, Card file error, Card file data, Card no data, Card option, Card product, Card rating, Card slot S
7001 to 7499	Erase file XXX	7XXX erases /fs/XXX on a SMART card or /fs/MCDF/XXX on an SD card. If the file does not exist, then parameter mm.000 is not reset to zero.	Card busy, Card error, Card no data, Card product Card read-only,
9555	Clear warning suppression	Clear the media card warning suppression flag. Whatever the initial state of this flag parameter mm.000 is reset provided the card can be accessed.	Card busy, Card error, Card product, Card read-only

9666	Set warning suppression	Set the media card warning suppression flag. Whatever the initial state of this flag parameter mm.000 is reset provided the card can be accessed.	Card busy, Card error, Card product, Card read-only
9777	Clear read-only	Clear the media card read-only flag. Whatever the initial state of this flag parameter mm.000 is reset provided the card can be accessed.	Card busy, Card error, Card product
9888	Set read-only	Set the media card read-only flag. Whatever the initial state of this flag parameter mm.000 is reset provided the card can be accessed.	Card busy, Card error, Card product
9999	Erase card	Erases a SMART card or resets the fs/MCDF/ folder on an SD card. When a SMART card has been erased it contains no files and the card header is reset so that the product type is correct, and the read-only and warning suppression flags are reset. When the fs/MCDF/ folder is reset on an SD the following actions are performed: Files 000 to 999 are deleted from the fs/MCDF/ folder. Files SCP0000 to SCP0099 are deleted from the fs/MCDF/SCOPE/ folder. File fs/MCDF/000 is reset so that that the product type is correct, and the read-only and warning suppression flags are reset.	Card busy, Card error, Card read-only
15010 to 15499 16010 to 16499 17010 to 17499	Save option user program	15XXX transfers a user program from an option module in Option Slot 1 to file /fs/MCDF/XXX on an SD card. If there is no SD card inserted in the drive or a failure occurs parameter mm.000 is not reset. 16XXX is used for Option Slot 2 and 17XXX is used for Option Slot 3. **NOTE: The drive cannot differentiate between drive user program files and option user program files. Care should be taken not to transfer these files into the drive using the 6XXX request code.**	Card busy, Card data exists, Card full, Card error, Card product, Card read-only, Card slot S
18001 to 18999 19001 to 19999 20001 to 20999	Restore option user program	18XXX transfers file /fs/MCDF/XXX to an option module in Option Slot 1. If there is no SD card inserted in the drive or a failure occurs parameter mm.000 is not reset. 19XXX is used for Option Slot 2 and 20XXX is used for Option Slot 3.	Card busy, Card error, Card no data, Card product, Card slot S

The failure indications related to transfer of data from a card to the drive (and option modules) are shown in red. These are critical failures because they could result in a drive malfunction, and so these failures always initiate a drive trip if they are detected. If parameters are being transferred to the drive these are not saved until the whole transfer has been completed successfully, and so the original parameter set-up can be recovered after a partial transfer by cycling the power. Failure indications related to transfer of data to a card are shown in black. These are non-critical failures and cannot result in a drive malfunction, but simply mean that no data was transferred, or the data transferred to the media card is corrupt. Non-critical failures do not cause a drive trip as this would disable the drive and may cause problems in the wider system. When a failure is detected (critical or non-critical) parameter mm.000 is not reset which indicates that the required action is not complete, but because non-critical failures do not trip the drive it would not be possible to determine what caused the transfer to be aborted. *NV Media Card Status* (11.078) is set to 1 (Active) when any of the actions given in the table above are started. If the action is completed successfully parameter mm.000 and *NV Media Card Status* (11.078) are reset to 0 and no further action is taken. If a failure is detected, whether the failure is critical or non-critical, *NV Media Card Status* (11.078) is set to the value associated with the failure as given in the table below. *NV Media Card Status* (11.078) is mapped to the top of Menu 0 (i.e. *Parameter 00.080 Set-up* (22.080) = 11.078), and so the simple action of scrolling down one parameter from *Parameter mm.000* (00.000) will show why the action failed.

***NV Media Card Status* (11.078)**	**Text String**	**Equivalent trip value in Trip 0 (10.020) etc.**	**Cause of failure**	**Failure when reading or writing to card**
0	None			
1	Active			
2	Card slot 1	174 (sub-trip 1)	A failure has occurred in the communications between the drive and an option module in Slot 1 when reading or writing data.	Read or Write
3	Card slot 2	174 (sub-trip 2)	A failure has occurred in the communications between the drive and an option module in Slot 2 when reading or writing data.	Read or Write
4	Card slot 3	174 (sub-trip 3)	A failure has occurred in the communications between the drive and an option module in Slot 3 when reading or writing data.	Read or Write
5	Card slot 4	174 (sub-trip 4)	A failure has occurred in the communications between the drive and an option module in Slot 4 when reading or writing data.	Read or Write
6	Card product	175	The card is from another drive product and is not compatible.	Read
7	Card user prog	177	An attempt has been made to read a write-only user program. An attempt has been made to transfer a user program from the drive to a card, but there is no user program present in the drive.	Write
8	Card busy	178	The card is already being accessed via comms or by an option module, or a file cannot be opened in the drive (i.e. /par/diff file) because it has already been opened via comms.	Read or write
9	Card data exists	179	An attempt is being made to write to a file that already exists.	Write
10*	Card option	180	A parameter difference file is being written to a drive, but one or more of the options that are present in the drive are different to those stored in the file.	Read
11	Card read-only	181	An attempt is being made to write to a read-only card or file.	Write
12	Card error	182	Failure to be able to access the card either because there is no card inserted in the drive, or because of a communication failure with the card or because the file system on the card is corrupted.	Read or write
13	Card no data	183	An attempt has been made to read a file that is not present on the card.	Read
14	Card full	184	An attempt is being made to write more data to the card, but it is full.	Write

15	Card file error	185	There is an error in the file being read from the card.	Read
16*	Card rating	186	A parameter difference file is being written to a drive, but the value of *Drive Rated Voltage* (11.033) in the file header is different to the value in the drive.	Read
17	Card file data	187	The drive mode, defaults previously loaded or product type in a parameter difference file are incorrect or outside the allowed range.	Read
18*	Card derivative	188	A parameter difference file is being written to the drive, but the value of *Drive Derivative* (11.028) in the file header is different to the value in the drive.	Read

* These failures are not detected if the card warning suppression flag Is set.

The following actions are taken at power-up or when a media card is inserted into a drive:

1. If it appears a SMART card is present, then it is accessed to check if communication is possible and to determine the size. If the action fails, no failure is indicated, and it is assumed the card is not present.
2. If it appears an SD card is present, then it is accessed to make sure the required folder/file structure is present. If the action fails, no failure is indicated.
3. A check is made to determine if the card is bootable if *NV Media Card Disable Booting* (11.041) is at its default value of 0. If a failure occurs during this check then no failure is indicated, and it is assumed that the card is not bootable.
4. If booting is not disabled and the check shows the card is bootable then an attempt is made to transfer the data from the card to the drive and to option modules. The following failures could then occur: Card slot S, Card product, Card option, Card error, Card file error, Card rating, Card file data or Card derivative. If a failure is detected it will always initiate a trip (unless it is suppressed by the card suppression warning flag).

Media card file access via comms does not cause a trip if a failure is detected, but it is reflected in the status returned in the ECMP responses. As already mentioned, the warnings are signalled in the response status when the file is closed.

This system is designed to minimise the number of trips initiated by reading or writing from/to a media card. If *NV Media Card Disable Booting* (11.041) is stored on a bootable card as 1 it will mean that the drive will only boot once at power-up and then not attempt to use any card inserting into the drive for booting again unless defaults are loaded. This provides a level of security in case an unauthorised bootable card is subsequently inserted into the drive.

Drive back-up methods

This section covers different methods that can be used to create a back-up of a drive and its option modules so that they can be restored later in the same drive or copied to another drive. Care should be taken when writing files to a media card via comms to ensure that the file does not cause a problem for the drive to interpret the file type. The only files that should be written to a SMART card or to root of the /MCDF/ folder on an SD card should be parameter difference files, macro files or drive user program files. Files that contain “PAR” as the first three bytes are assumed to be parameter difference files and files that contain “MAC” as the first three bytes are assumed to be macro files. Files that contain any other data as the first three bytes are assumed to be drive user program files.

Parameter difference file

A parameter difference file can be created using parameter mm.000 = 3XXX or 4XXX to initiate data transfer from the drive to the card. This will create a file on the card that can be restored again using parameter mm.000 = 6XXX in the destination drive to initiate the restore. Before the difference from default data in the file is transferred to the destination drive, defaults are loaded using the same mm.000 value that was used on the source drive. If the drive mode in the source drive is different to the drive mode in the destination drive the drive mode is changed. If the transfer is successful, the drive parameters are saved. If the destination drive has the same voltage and current rating, the same option modules are fitted, and each drive is the same type (i.e. *Drive Derivative* (11.028) is the same) then the parameters in the destination drive and its option modules will be the same as those in the source drive. If any of these are different then the parameters in the destination drive and option modules may be different as follows.

1. If the voltage or current rating are different some parameters will have different variable maximums, i.e. Rated Current (05.007). The parameter value from the source drive may be outside the range in the destination drive and will be limited.
2. If the voltage ratings of the source and destination drives are different any parameters with the RA (voltage rating dependent) attribute set will be left at default in the destination drive. A “Card rating” failure will be detected but can be supressed if required by setting the warning suppression bit on the card.
3. If the option modules fitted in the source and destination drives are different the parameters in the set-up menus (15 to 17 and 24) and applications menus (25 to 28) will be left at their default values for any option modules that is different. Any option module internal menu parameters stored in the file on the card will not be transferred to any option modules that are different. A “Card option” failure will be detected but can be supressed if required by setting the warning suppression bit on the card.
4. If the source and destination drive have different derivatives the parameters in the destination drive may have different defaults or ranges, and some parameter may not exist. A “Card derivative” failure will be detected but can be supressed if required by setting the warning suppression bit on the card.

A parameter difference file can be read from the source drive and then written to the destination drive via comms. This has the same effect as transferring the data via a media card except that:

1. No option module internal parameter data is transferred.
2. *NV Media Card Status* (11.078) is not used and trips are not initiated during the file read or write. The status in the EMCP responses indicates any failures.

Macro file
If *NV Media Card Create Special File* (11.072) is set to 1 when parameter mm.000 is set to 3XXX or 4XXX to transfer parameter data from the drive to a card, the file created on the card is a macro file instead of a parameter difference file.

User program file
A user program file can be created using parameter mm.000 = 5XXX to initiate the data transfer to the card. This will create a file on the card that can be restored again using parameter mm.000 = 6XXX in the destination drive to initiate the transfer to the drive. Once the user program has been written to the drive the processor resets and the drive restarts. If the transfer process to the drive is started, but not completed successfully the user program is deleted.

It is possible to create a write-only user program which prevents the program from being transferred on to a media card. If this is attempted a “Card user prog” failure is detected

User program files can be read from the source drive and then written to the destination drive via comms. This has the same effect as transferring the data via a media card except that *NV Media Card Status* (11.078) is not used and trips are not initiated during the file read or write. The status in the EMCP responses indicates any failures.

Bootable media card with parameter difference file
If *NV Media Card Disable Booting* (11.041) = 0 and a bootable media card is present in a drive at power-up the drive will attempt to use the data on the card to set up its user parameters and option module parameters. A bootable media card must contain a parameter difference file as /fs/001 on a SMART card or /fs/MDCF/001 on an SD card and the value of *Parameter Cloning* (11.042) in the file header must be set to 4. If this file is present it is used to set-up the drive and option module parameters and then the drive parameters are saved. If the transfer is not successful, a drive is trip is initiated and the process is aborted. If the option modules fitted to a drive are either removed or changed since the last user parameter save the drive normally produces a “Slot Not Fitted” or “Slot Different S” trip. These trips are suppressed during the booting process, so that provided the option modules fitted match those used to create the booting file, these trips will not be produced.

If the media card was created in a drive with a different voltage rating, different option modules fitted or with a different derivative the transfer process will be completed, but a trip will be initiated to warn the user unless the warning suppression bit is set on the card.

A bootable media card that contains a parameter difference file as /fs/001 on a SMART card or /fs/MDCF/001 on an SD card can be created using parameter mm.000 = 2001.

Bootable media card with parameter difference file and user program
If a bootable media card also includes a file that is not a parameter difference file as /fs/002 on a SMART card or /fs/MDCF/002 on an SD card, then the drive will read the CRC from this file in the location where it would expect it to be. If this is different from the CRC of the user program stored in the drive the user program will be transferred from the card to the drive. The drive processor will be reset, and the drive will again detect the bootable media card. Because the CRC of the program on the card will now be the same as the CRC in the program in the drive, the program will not be transferred again. The parameters will be transferred from the parameter difference file to complete the process.

A bootable media card with an additional user program file as /fs/002 on a SMART card or /fs/MDCF/002 on an SD card can be created using parameter mm.000 = 2002.

Automatically creating a parameter file backup on a media card
If *Parameter Cloning* (11.042) = 3 (Auto) the drive operates in automatic back-up mode and maintains a parameter difference file as /fs/001 on a SMART card or /fs/MDCF/001 on an SD card which holds a copy of the drive and option module parameters. At power-up, each time a parameter save is initiated and each time a Menu 0 parameter is changed via a keypad the file on the card is written or over-written (i.e. the same action as initiated with mm.000 = 3001). The value of *Parameter Cloning* (11.042) stored in the file header will be zero and the card will not be a bootable media card. If there is no card in the drive *Parameter Cloning* (11.042) is held at zero. If a new card is inserted and automatic back-up mode is required *Parameter Cloning* (11.042) should be set to 3 (Auto) and a drive reset should be initiated.

Automatically creating a bootable media card
If *Parameter Cloning* (11.042) = 4 (Boot) the drive still creates an automatic back-up as described above, but the value of *Parameter Cloning* (11.042) stored in the file header will be 4 and the card will be bootable. If the card is present in the drive at power-up, the boot process is followed to transfer the user parameters (and user program if present) to the drive instead of creating the back-up file. Otherwise the system operates in the same way as when *Parameter Cloning* (11.042) = 3 (Auto).

SMART Card Data Structure and Access

SMART cards are I2C memory cards. For example, a card containing a Microchip AT24C512 device. Card sizes from 4K bytes to 128K bytes are supported with a minimum page write size of 16 bytes. A clock rate of 250kHz is used to access the card.

Below is a repeat of the example of the file structure seen on a SMART card via comms.

```
/fs
  /001   Open-loop Parameter Difference File
  /002   Drive user Program File
  /007   RFC-S Parameter Difference File
  /501   Open-loop Parameter Difference File
  /999   Regen Parameter Difference File
```

The structure on the actual card cannot contain any sub-folders and is a card header followed by a series of file blocks each containing a file.

SMART card data structure.

Card header
File block 1 header
File 1

File block N header
File block N

Card header.

Byte addresses	Data	Function
0	0x00, 0x01 or 0xFF	Read-only flag. Any value except 0x01 = flag not set. 0x01 = flag set.
1	0x00, 0x01, 0xFF	Warning suppression flag. Any value except 0x01 = flag not set. 0x01 = flag set.
2	5	Product identifier always set to 5.
3 to 15	0xFF	Not used.

If the card header, except the product identifier, does not conform to the specification above then a “Card error” failure is detected. If the product identifier is not 5 then a “Card product” failure is detected. For reference the table below gives the expected product identifiers that might be found in SMART cards from other products.

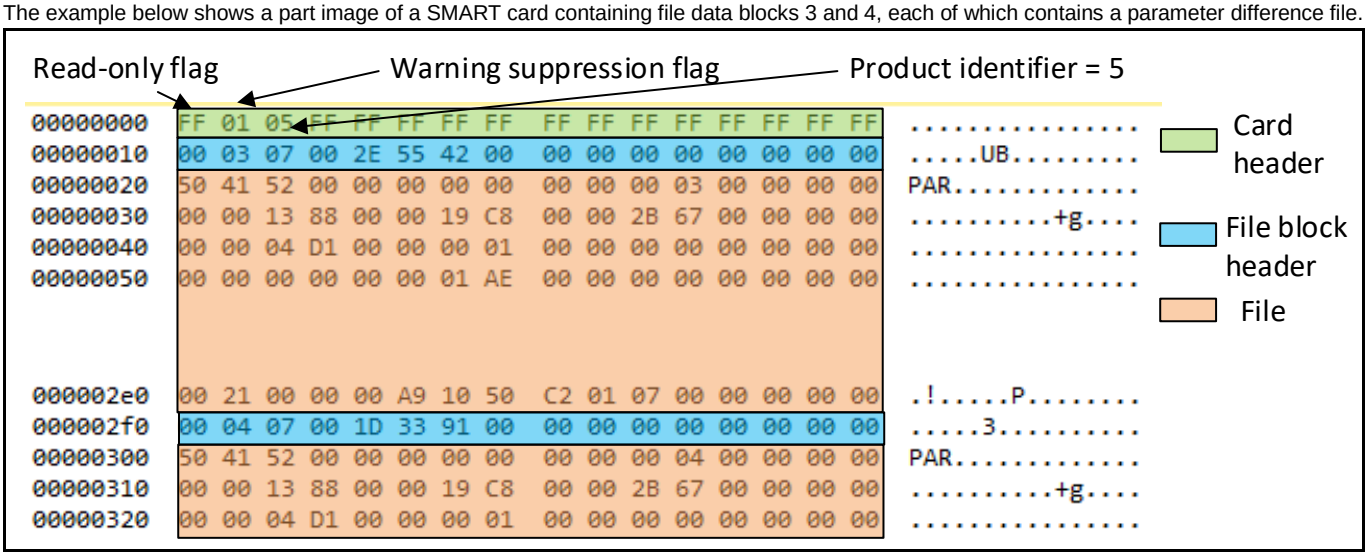
Product identifier	Product
0	Not used
1	Unidrive GP20
2	Digitax ST
3	Unidrive Affinity
4	Mentor MP
5	Unidrive M or Digitax HD
6 - 254	Not used
255	Unidrive SP

File block header

Byte addresses	Data	Function
0 - 1	File block number	*NV Media Card File Number* (11.037) corresponding to this file block.
2	7	File block type is always 7.
3 - 4	Number of pages	Number of 16-byte pages in the file.
5 - 6	Checksum	See below.
7 - 15	0x00	Always zero.

1. The file blocks can be stored in any order on the card and do not need to be in the order of the file block numbers.
2. File blocks must contain an integer number of whole 16-byte pages and be at least 1 page long. The length of each file block is given in its header as the number of 16-byte pages. This is the length including the header.
3. The file data can be a parameter difference, macro or a drive user program file. Any space beyond the end of the file in the last 16-byte page must be all zeros.
4. The checksum is the modulo 65536 sum of bytes in the file block. This includes all 16-byte pages including the file header block but excluding the checksum itself (i.e. bytes 5 and 6 in the header). Although any data beyond the end of the file is included up to the end of the last 16-byte page, as the data beyond the end of the file is all zeros, it will not affect the checksum. The checksum is always present in files that are created on the card, however due to the time required to check this with large files when a file is opened the checksum is not used by the drive.
5. The file blocks must be contiguous. If a file is deleted the subsequent blocks must be moved to remove the gap left on the card.
6. All space beyond the last file block (if there is any) must be 0xFF.

The example below shows a part image of a SMART card containing file data blocks 3 and 4, each of which contains a parameter difference file.



Accessing a SMART card via comms
Accessing a SMART card via comms gives more flexible access than operations triggered via parameter mm.000. The following should be noted.

1. The file number must be between 001 and 999 for a valid file, however it is also possible to use File number 000 in a request via comms. Reading File /000 reads the whole card as a binary image. Deleting File /000 formats the whole card. Writing File /000 begins writing at the first location on the card (i.e. the start of the card header) and data can be written to the whole card. No data is deleted from the card before writing begins. This means that writing File /000 can be used to modify the card header without affecting the rest of the card data if up to 16 bytes are written before the file is closed. Alternatively, the whole card can be written if required.
2. The card read-only flag has no effect, and so the card can always be formatted or written via comms. Files 500 and above are read/write via comms.
3. Blocking or non-blocking operation can be specified via ECMP communications. It is recommended that non-blocking operation is used because some operations can take some time to complete. For example, deleting a file which is towards the start of the card when there are many present, formatting the card by deleting File /000 or writing a large amount of data with a write request. If blocking operation is used there may be a significant delay before a response is given to the request or the drive background task may be stalled for a significant time. Note that when non-blocking operation is used, the response from the request will indicate whether action can be taken (i.e. there is enough space for the required data to be written to the card). It is then possible to determine when the action is complete using the ECMP FileState request.
4. It is only possible to open a file for writing in create mode and not append mode. Opening a file for writing will delete the file if it exists and the new file will be added immediately after all the other files on the card.
5. Because opening a file for writing will delete the file if it exists, if a delete request is used to delete a file, the file must be open for reading before it is deleted. If non-blocking operation is used the file must be closed when the file has been deleted. If non-blocking operation is not used, then the file is automatically closed after being deleted.
6. It is possible to change the location for reading or writing within the file. As already mentioned, the checksum for the file is created, but not used. If the location is changed to over-write data within the file that has already been written the checksum is not first reduced by the data value that is over-written. This means that the checksum will only be correct if the data over-written is zero.

Blank cards
The following are notes related to blank SMART cards.

1. When a blank SMART card (all locations are 0xFF) is inserted into a drive location 15 is modified by the algorithm that detects the size of the card. All other locations are left at 0xFF.
2. File 0 can be read or written to a blank SMART card via comms.
3. If any file other than File 0 written to a blank SMART card via comms the card header is automatically created on the card.
4. If any file is written to the card via parameter mm.000 the card header is automatically created on the card.
5. A “Card Product” failure is initiated if any of the following actions are performed via parameter mm.000: reading a file, changing the warning suppression flag, erasing a file. This indicates that the card is not formatted for this product.
6. The read-only flag can be modified via parameter mm.000 on a blank card.
7. A blank card can be erased via parameter mm.000 independent of the state of the read-only flag. If header exists on the card and the read-only flag is set, the card cannot be erased via parameter mm.000. In this case the read-only flag needs to be cleared first.

SD Card Data Structure and Access

SD or SDHC cards can be used and must be preformatted with the FAT32 file system. As already mentioned, when the card is inserted into a drive the following folders and file with be created if not already present.

```
/fs
  /MCDF
    /DATA
      /S1
      /S2
      /S3
      /S4
    /KEYPAD
    /SCOPE
    /000
```

File /fs/MCDF/000 contains 16 bytes and provides the same features as the header on a SMART card. This file can be read or written in the same way as any other file via comms. However, deleting this file is the same as using the erase request via parameter mm.000 and resets the fs/MCDF/ folder. This function does not affect any of the folders, but simply deletes files 001 to 999 in the fs/MCDF/ folder and files SCP0000 to SCP0099 in the fs/MCDF/SCOPE/SCOPE folder if present. It then overwrites fs/MCDF/000 with default settings: read-only flag = 0x00, warning suppression flag = 0x00 and product identifier = 5. All file operations via comms, except deleting fs/MCDF/000, can be blocking or non-blocking, but the implementation is always blocking in that the required action is taken and then a response is given. The request to delete fs/MCDF/000 via comms must be a non-blocking request otherwise an Invalid request response is given. This is because this can take several seconds even if no files are deleted.

The maximum length file path is 64 characters and all file names must conform to 8.3 format (maximum of 8 characters in the name and a maximum of 3 characters in the extension). This means that the total file path and file name must not exceed 76 characters.

It is possible to open a file for appending. The location pointer is set to the end of the file.

Blank cards
An SD card does not need to be blank to be used with the drive. If the MCDF folder or any of the required folder and file structure is not present when the card is inserted into a drive, then it is created automatically.

ECMP Error Codes

The tables below give the possible ECMP status values and their meaning when accessing a media card via comms.

FileOpen

Error code	Error	Description
1	Okay	File opened successfully.
-1	File handle	The file handle used in the request is out of range.
-2	Blocked	Non-blocking mode was used for the previous request and the required process has not finished yet.
-5	Not found	SMART card: Open with any access mode and the file name is not a decimal value between 000 and 999. Open for read and the requested file cannot be found. SD card: Open for read and the requested file cannot be found.
-11	Too big	SMART card: Open for writing and not enough space for a file header (16 bytes)
-15	Too many open	A file handle is not available.
-16	File invalid	SMART card: The card header is not valid (i.e. from a different product). Note this error is not given when File 000 is opened. Open for read and the requested file is not valid: the product identifier is not 5.
-17	Invalid request	SMART card: There is no card present, there has been a failure to communication with the card or the card file system is corrupted. SD card: There is not card present, there has been a failure to communication with the card or the file path and file name is too long.
-18	Append not allowed	SMART card: Not supported.

FileRead

Error code	Error	Description
3	Okay end of file	The requested amount of data will return all the remaining data from the file.
2	Okay more data	The requested amount of data will not return all the remaining data from the file.
-1	File handle	The file handle used in the request is out of range.
-2	Blocked	Non-blocking mode was used for the previous request and the required process has not finished yet.
-9	No more data	The end of the file has already been reached and there is no more data to read.
-10	Wrong mode	The file is not open for reading.
-17	Invalid request	There is no card present, there has been a failure to communication with the card, or the file is not open for reading.

FileWrite

Error code	Error	Description
1	Okay	Data written to the file successfully.
-1	File handle	The file handle used in the request is out of range.
-2	Blocked	Non-blocking mode was used for the previous request and the required process has not finished yet.
-10	Wrong mode	The file is not open for writing.
-11	Too big	SMART card: There is not enough space left on the card to write the required data. No data will be written. The incomplete file written so far will be delated.
-17	Invalid request	There is no card present, there has been a failure to communication with the card, or the file is not open for writing.

FileClose

Error code	Error	Description
1	Okay	File closed successfully.
-1	File handle	The file handle used in the request is out of range.
-2	Blocked	Non-blocking mode was used for the previous request and the required process has not finished yet.
-17	Invalid request	There is no card present or there has been a failure to communication with the card.

FileDelete

Error code	Error	Description
1	Okay	File closed successfully.
-1	File handle	The file handle used in the request is out of range.
-2	Blocked	Non-blocking mode was used for the previous request and the required process has not finished yet.
-17	Invalid request	SMART card: There is no card present, there has been a failure to communication with the card, or the file is not open for reading. SD card: There is no card present, there has been a failure to communication with the card, the file is not open for reading or the request has not been made as non-blocking.

FileState

Error code	Error	Description
1	Okay	File closed successfully.
-1	File handle	The file handle used in the request is out of range.
-2	Blocked	Non-blocking mode was used for the previous request and the required process has not finished yet.
-17	Invalid request	There is no card present, there has been a failure to communication with the card.

FilePos

Error code	Error	Description
1	Okay	Position has been set successfully. The required position may have been limited to ensure it is within the open file.
-1	File handle	The file handle used in the request is out of range.
-2	Blocked	Non-blocking mode was used for the previous request and the required process has not finished yet.
-10	Wrong mode	The file is not open.
-17	Invalid request	This function is not supported.