

SIMULATION/MODELING AND AUDIO SYSTEMS
DELPHI AUTOMOTIVE SYSTEMS CORPORATION

System Analysis Interface Tool 2

SAINT 2

SD Card Set Up for Controller Re-Flash

Version 1.6

December 10, 2007

Copyright © Delphi Automotive Systems Corporation, 2005, 2006, 2007

Maintained by: Simulation and Modeling Group
Delphi Delco Electronics Systems
Mail Station CT30A
Kokomo, IN 46904

Table of Contents

1 <i>Introduction</i>	3
1.1 Scope	3
1.2 Definitions and Nomenclature	3
2 <i>Files Required for Controller Re-flash</i>	4
2.1 Unzipped DPS Archive Files	4
2.2 Encrypted Security Key File	4
2.3 SAINT2 Re-flash Script	4
2.3.1 Script Description	5
2.3.2 Script Structure	5
2.3.3 Script Example.....	6
2.3.4 Script Specifications	6
2.3.5 Script Keywords.....	7
3 <i>Re-flash Output Files</i>	14
4 <i>Error Reporting</i>	15
4.1 Archive Utility File Related Errors	15
4.2 Class 2 Errors	15
4.2.1 Class 2 Util File Errors	16
4.2.2 Class 2 Script Errors	16
4.2.3 Class 2 Bus Errors.....	16
4.3 GMLAN Errors	16

Standalone Flash Programming Language	12/10/07	Page 2 of 20
---------------------------------------	----------	---------------------

Revision Log

Version	Revisions	Date Released to Web
1.1	1. Initial release of the document.	NA
1.2	1. Noted Script Commands that are not yet supported 2. Listed supported CAN baud rate values and transceiver settings 3. Changed order in example script for exit diag command.	NA
1.3	1. Now Support Write Did commands and Read DTC command	NA
1.4	1. Added information for Class 2. Updated “not supported” items.	NA
1.5	1. Added Ford 2003.0 Software Download Strategy section 2. Added GMLAN Reflash Strategy section 3. Changed task block limit to unlimited 4. Allow 30 script commands 5. Allow up to 24 bytes of data to be stored 6. Changed Ford file and block reference to start with 1 7. Added Error codes 8. Removed Ford info 9. Allow 60 util file instructions (steps) 10. Specify max number of routines in util file – 40 11. Update Read DTCs description	NA
1.6	1. Added Archive utility file error codes. 2. Removed GMLAN and CAN references to a separate document 3. Reorganized and updated Universal Diagnostics commands	Dec 10, 2007

1 Introduction

1.1 Scope

This document describes how to set up the SD card to enable the SAINT2 to autonomously re-flash a controller.

This document describes the necessary product software files for re-flash, the SAINT2 re-flash script, the encrypted key file, and the reflash output files.

1.2 Definitions and Nomenclature

2 Files Required for Controller Re-flash

The following files may be needed on the SD card to successfully re-flash a controller:

The unzipped DPS archive file – The DPS archive file contains the software files needed to flash the controller. Each of the files in the archive must have an 8.3 filename.extension and must be in binary format.

An encrypted security key file – This file is used by the SAINT2 to determine the security key for a locked controller. This file must have an 8.3 filename.extension. This file is created using the SAINT2 Key Maker application. This file is not required if the controller is not locked or if security unlock routine 0 is used in the utility file.

The script.txt file – SAINT2 re-flash script – This file contains the commands to automate the re-flash of the controller. The file shall have the unique name “script.txt” to be recognized by the SAINT2 as re-flash instructions.

2.1 Unzipped DPS Archive Files

The unzipped DPS archive files contain the software files needed to flash the controller. Each of the files in the archive must have an 8.3 filename.extension and must be in binary format. The file names may not be changed after the creation of the archive because the files are referenced in the .tbl file. If the filenames are not in 8.3 format, the archive must be re-created with new file names. The archive file should include a .tbl file, a utility file, and necessary calibration files. If an opcode 53 is used within the utility file to compare VIT2 data, the software files must be named with a file name that can be converted to unsigned numeric.

2.2 Encrypted Security Key File

The encrypted security key file is used by the SAINT2 to determine the security key for a locked controller. This file must have an 8.3 filename.extension. This file is created using the SAINT2 Key Maker application. This file is not required if the controller is not locked or if security unlock routine 0 is used in the utility file. For information on how to use SAINT2 Key Maker, see the SAINT-2 Key Encryption User Notes.

2.3 SAINT2 Re-flash Script

2.3.1 Script Description

The script.txt file contains the instructions needed for the SAINT2 to automate the controller re-flash process. The script is divided up into a single or multiple task blocks. The multiple task blocks may be used to re-flash separate controllers or to perform conditional re-flashes. The script must be named “script.txt” in order to be recognized by the SAINT2 as re-flash instructions.

2.3.2 Script Structure

The script has the following structure:

Begin Statement for 1st Task Block

Non-Sequential Definitions

file name definitions

protocol definition

bus speed definition

Sequential Commands

conditional commands to determine need for flash

flash execution commands

commands to determine success of flash

conditional commands to write information to report files

controller data commands

report file commands

End Statement for 1st Task Block

Begin Statement for 2nd Task Block

Non-Sequential Definitions

file name definitions

protocol definition

bus speed definition

Sequential Commands

conditional commands to determine need for flash

flash execution commands

commands to determine success of flash

conditional commands to write information to report files

controller data commands

report file commands

End Statement for 2nd Task Block

.

.

.

Begin Statement for Nth Task Block

Non-Sequential Definitions

file name definitions

protocol definition

bus speed definition

Sequential Commands

conditional commands to determine need for flash

flash execution commands

commands to determine success of flash

controller data commands

report file commands

End Statement for Nth Task Block

2.3.3 Script Example

This script example has a single task block:

```

;“script.txt”
begin                                     ;statement to begin task block 1
;comment: non-sequential definitions
report file: podsrpt.txt                 ;report file
tbl file: podstbl.TBL                   ;tbl file
protocol: 0x52                           ;use protocol SWCAN
comm baud rate: 33.333                   ;baud rate = 33.333kHz
flash baud rate: 33.333                 ;baud rate = 33.333kHz
key file: EPodH00.key                   ;Key File for security unlock
report labels: C0                       ;set column labels
vin prod: 0x99                          ;obtain VIN from Prod 0x99

;comment: beginning of sequential script commands
enter diag
if did data exclude: 0x59, 0xC0, FFFFFFFF, 01453568 ; re-flash needed?
exec util                                ;execute DPS utility file
enter diag                                ;re-enter diagnostics
if did data include: 0x59, 0xC0, FFFFFFFF, 01453585 ;re-flash successful?
read did: 0x59, 0xC0, 1                 ;read data to report
exit diag                                ;exit diagnostics
report data: 1, ffffffff                 ;report data to the summary file
end                                       ;statement to end task block 1

```

2.3.4 Script Specifications

The following are script specifications:

Maximum number of Task Blocks: no limit

Maximum number of Commands per Task Block: 30

Maximum number of characters in comment field: 40

Standalone Flash Programming Language	12/10/07	Page 7 of 20
---------------------------------------	----------	---------------------

Maximum number of instructions in the utility file: 60 (not defined in the script file)

Maximum number of routines in the utility file: 40 (not defined in the script file)

Maximum delay time: 65535 milliSeconds (65.535 Seconds)

2.3.5 Script Keywords

2.3.5.1 File Keywords

report file: *filename.txt*

The **report file** is a text log of the controller re-flash process.

tbl file: *filename.bin*

The **tbl file** is the .tbl file in the DPS archive that defines the flash utility and the product calibration files. This file also defines the order of the calibration files. The tbl file must be in binary format. All files referenced by the tbl file must be in binary format.

key file: *filename.key*

The **key file** is an encrypted key file containing key values. This file is created by the application SAINT2 Key Maker.

filename description

Argument	Description	Example
<i>filename</i>	up to 8 char ASCII file name + 3 char extension	01345234.bin, podsprt.txt

2.3.5.2 Hardware Configuration Keywords

protocol: *0xAA* [, *0xLLLLLLLL_{opt}*]

The **protocol** statement specifies the serial protocol and any specific hardware configuration such as CAN node and transceiver.

comm baud rate: *0xBBBB* or *baud* Not applicable for Class 2

The **comm baud rate** statement specifies the normal communication baud rate of the controller. Baud rates from 33.333k to 500k are supported.

flash baud rate: *0xBBBB* or *baud* Not applicable for Class 2

The **flash baud rate** specifies the communication baud rate of the controller during the re-flash. Baud rates from 33.333k to 500k are supported.

Standalone Flash Programming Language	12/10/07	Page 8 of 20
---------------------------------------	----------	---------------------

Protocol Arguments

Argument	Description	Example
<i>0xAA</i> : bit7-bit3	SAINT Header used for the desired protocol	0x50 = 01010xxx _b = CAN node 1
<i>0xAA</i> : bit2	Not used	
<i>0xAA</i> : bit1-bit0	node configuration if applicable for a specific protocol	xxxxxx10 _b = SWCAN
		Values Currently Supported
<i>0xAA</i>	SWCAN1 - Single Wire CAN node 1	0x52
<i>0xAA</i>	CAN_1_H, CAN_1_L – Dual Wire High Speed CAN node 1	0x50
<i>0xAA</i>	Class 2	0x60
<i>0LLLLLLLL</i>	Optional – Sub-protocol information 32 bit	0x00 – GMLAN 0x04 – GMLAN for APM

0xB BBB or baud description

Argument	Description	Example
<i>0xB BBB</i>	baud rate configuration value for a specific protocol	0x8DAF = 33.333k
<i>baud</i>	ASCII string specifying a common baud rate in kHz	83 = 83.333k

Supported CAN *baud* values

<i>baud</i>	Baud Rate	Sample Point	Equivalent <i>0xB BBB</i>
33	33.333k	85%	0x8DAF
83	83.333k	83.3%	0xB12D
95	95.2k	75%	0xF03A
100	100k	73.3%	0xF139
125	125k	80%	0xDD3E
200	200k	73.3%	0xD839
250	250k	80%	0xCE3E
500	500k	73.3%	0xC939

See the SAINT2 Programmer's Reference Document for information on how to calculate *0xB BBB* for other baud rates.

2.3.5.3 Flash Execution Commands

begin

The **begin** command defines the beginning of a series of script commands belonging to a single task block

end

The **end** command defines the beginning of a series of script commands belonging to a single task block

Standalone Flash Programming Language	12/10/07	Page 9 of 20
---------------------------------------	----------	---------------------

exec util

The **exec util** command specifies where, during the script, the DPS utility file should be executed. For GMLAN re-flashes, this command will exit diagnostics after the utility file has executed.

enter diag [*0xCC_{opt}*]

The **enter diag** command specifies where, during the script, diagnostics mode should be entered. If this command is specified with a product ID argument only the specified product is commanded to enter diagnostics. Otherwise, if a product ID is not specified, diagnostics is commanded to all devices (0xFE for CAN and Class 2). This command is also useful if you need to re-enter diagnostics mode after the issuance of an exit diag command.

0xCC description

Argument	Description	Example
<i>0xCC</i>	Physical Address of the target ECU	0x80 = target address of radio

enter prog: *0xCC*

The **enter prog** specifies where, during the script, programming mode should be entered for the specified target ID.

unlock: *0xCC, 0xSA, 0flag*

The **unlock** command specifies where, during the script, the specified target ID should be unlocked. This command is not needed unless the target needs to be unlocked for the script to read or write protected values before or after reprogramming. The utility file should contain an opcode \$27 to unlock the target during utility file execution.

Argument	Description	Example
<i>0xSA</i>	Security Algorithm	
<i>0flag</i>	Flag to indicate whether key should be sent if seed is 0x0000	0=do not send key, 1=send key

exit diag

The **exit diag** command specifies where, during the script, diagnostics mode should be exited. Both regular diagnostics mode and manufacturing diagnostics mode will be exited.

delay: *time*

The **delay** command will cause the script execution to pause for a given time

time description

Argument	Description	Example
<i>Time</i>	amount of time to pause script in milliseconds (0 – 65535ms)	2000 = 2000ms

vin prod: *0xCC*

The **vin prod** command identifies to the SAINT2 the Product ID of target device that will provide VIN data at the end of script execution. If this argument is not specified, VINPRODNOTDEFINED will be written to the summary file in place of the VIN. A script will not fail due to an error associated with reading the VIN.

2.3.5.4 Comparison Commands

clear dtc: *0xCC*

The **clear dtc** command is used to clear diagnostic trouble codes with a Clear Diagnostic Mode diagnostic service command.

read dtc: *0xCC, 0xDD*

The **read dtc** command is used to read diagnostic trouble codes by status mask. DTCs are immediately written to the line associated with each task block in the summary file. DTCs are not permitted to be stored to a storage buffer location.

read did: *0xCC, 0xEE, F* (Read did = Mode \$3C for Class 2)

The **read did** command is used to read a specified data identifier and store the returned data in a given SAINT2 data buffer.

write did: *0xCC, 0xEE, F* (Write did = Mode \$3B for Class 2)

The **write did** command is used to write a specified data identifier with data from a given SAINT2 data buffer.

write did data: *0xCC, 0xEE, data* (Write did = Mode \$3B for Class 2)

The **write did data** command is used to write a specified data identifier with data specified in the script.

read dpid: *0xCC, 0xGG, F*

The **read dpid** command is used to read a single dpid and store the returned data in a given SAINT2 data buffer.

read mem: *0xCC, address, 0xJJ, F* – not yet supported for C2

The **read mem** command is used to read the specified memory from the controller and store it in a given SAINT2 data buffer. In version 3.6.0, argument *F* represents both the storage location and the number of bytes stored in memory. This value can be 1 to 9. In other words if you read 4 bytes of memory, the memory read will be stored in location 4. This argument limitation will be changed in the future.

if did include: *0xCC, 0xEE, F, mask* (Read did = Mode \$3C for Class 2)

The **if did include** command is used to compare a specified data identifier with the value in a given SAINT2 data buffer. If a true is returned, the script will continue to execute. If the command is called before the exec util command, a return of false will cause the task block to exit with a flash not needed state. If the command is called after the exec util command, a return value of false will cause the task block to exit with a flash failure state.

if did exclude: *0xCC, 0xEE, F, mask* (Read did = Mode \$3C for Class 2)

The **if did exclude** command is used to compare a specified data identifier with the value in a given SAINT2 data buffer. If a true is returned, the script will continue to execute. If the command is called before the exec util command, a return of false will cause the task block to exit with a flash not needed state. If the command is called after the exec util command, a return value of false will cause the task block to exit with a flash failure state.

if did data include: *0xCC, 0xEE, mask, data[, data2]* (Read did = Mode \$3C for Class 2)

The **if did data include** command is used to compare a specified data identifier with a range of data values specified in the script. If a true is returned, the script will continue to execute. If the command is called before the exec util command, a return of false will cause the task block to exit with a flash not needed state. If the command is called

Standalone Flash Programming Language	12/10/07	Page 11 of 20
---------------------------------------	----------	---------------

after the exec util command, a return value of false will cause the task block to exit with a flash failure state. If data2 is omitted, data2 is assumed to be equal to data.

if did data exclude: *0xCC, 0xEE, mask, data[, data2]* (Read did = Mode \$3C for Class 2)

The **if did data exclude** command is used to compare a specified data identifier with a range of data values specified in the script. If a true is returned, the script will continue to execute. If the command is called before the exec util command, a return of false will cause the task block to exit with a flash not needed state. If the command is called after the exec util command, a return value of false will cause the task block to exit with a flash failure state. If data2 is omitted, data2 is assumed to be equal to data.

if dpid include: *0xCC, 0xGG, F, mask*

The **if dpid include** command is used to compare a specified dpid with the value in a given SAINT2 data buffer. If a true is returned, the script will continue to execute. If the command is called before the exec util command, a return of false will cause the task block to exit with a flash not needed state. If the command is called after the exec util command, a return value of false will cause the task block to exit with a flash failure state.

if dpid exclude: *0xCC, 0xGG, F, mask*

The **if dpid exclude** command is used to compare a specified dpid with the value in a given SAINT2 data buffer. If a true is returned, the script will continue to execute. If the command is called before the exec util command, a return of false will cause the task block to exit with a flash not needed state. If the command is called after the exec util command, a return value of false will cause the task block to exit with a flash failure state.

if dpid data include: *0xCC, 0xGG, mask, data[, data2]*

The **if dpid data include** command is used to compare a specified dpid with a range of data values specified in the script. If a true is returned, the script will continue to execute. If the command is called before the exec util command, a return of false will cause the task block to exit with a flash not needed state. If the command is called after the exec util command, a return value of false will cause the task block to exit with a flash failure state. If data2 is omitted, data2 is assumed to be equal to data.

if dpid data exclude: *0xCC, 0xGG, mask, data[, data2]*

The **if pid data exclude** command is used to compare a specified dpid with a range of data values specified in the script. If a true is returned, the script will continue to execute. If the command is called before the exec util command, a return of false will cause the task block to exit with a flash not needed state. If the command is called after the exec util command, a return value of false will cause the task block to exit with a flash failure state. If data2 is omitted, data2 is assumed to be equal to data.

if no dtc: *0xCC, 0xDD* – not yet supported for Class 2 Not applicable for Class 2

The **if no dtc** command is used to confirm that no diagnostic trouble codes are currently set.

Argument description

Argument	Description	Example
<i>0xCC</i>	Physical Address of the target ECU	0x80 = target address of radio
<i>0xDD</i>	DTC Status Mask	0x12 = status mask of 0x12
<i>0xEE</i>	DID (data identifier value)	0x90 = DID 0x90 = VIN

Standalone Flash Programming Language	12/10/07	Page 12 of 20
---------------------------------------	----------	----------------------

<i>F</i>	24 byte Data buffer number (0-9)	5 = data buffer 5
<i>0xGG</i>	DPID/PID number	0x02 = DPID 2
<i>0xJJ</i>	Memory address length in bytes (1-4)	0x03 = 3 byte address
<i>address</i>	Address to read (enter as hex without 0x)	00FE6082 = read at 0x00FE6082
<i>mask</i>	Up to 24 byte bit mask for data comparison	00000F00 = compare low nibble of the 3 rd byte with <i>data</i> and <i>data2</i>
<i>data</i>	Low end of the data range - up to 24 bytes of data (enter as hex bytes without 0x)	00475800
<i>data2</i>	High end of the data range - up to 24 bytes of data (enter as hex bytes without 0x)	004758FF

2.3.5.5 Universal Diagnostic Commands

enter manf diag: *0xXX*

The enter manf diag command is used to enter the manufacturing diagnostics mode.

read pid: *0x2XXYZZZZ, udata, F*

For C2 the read dpid command is used to read a single pid and store the returned data in a given SAINT2 data buffer.

if pid include: *0x2XXYZZZZ, udata, F, mask*

Not applicable for Class 2

The **if pid include** command is used to compare a specified pid with the value in a given SAINT2 data buffer. If a true is returned, the script will continue to execute. If the command is called before the exec util command, a return of false will cause the task block to exit with a flash not needed state. If the command is called after the exec util command, a return value of false will cause the task block to exit with a flash failure state.

if pid exclude: *0x2XXYZZZZ, udata, F, mask*

Not applicable for Class 2

The **if pid exclude** command is used to compare a specified pid with the value in a given SAINT2 data buffer. If a true is returned, the script will continue to execute. If the command is called before the exec util command, a return of false will cause the task block to exit with a flash not needed state. If the command is called after the exec util command, a return value of false will cause the task block to exit with a flash failure state.

if pid data include: *0x2XXYZZZZ, udata, mask, data[, data2]* Not applicable for Class 2

The **if pid data include** command is used to compare a specified pid with a range of data values specified in the script. If a true is returned, the script will continue to execute. If the command is called before the exec util command, a return of false will cause the task block to exit with a flash not needed state. If the command is called after the exec util command, a return value of false will cause the task block to exit with a flash failure state. If data2 is omitted, data2 is assumed to be equal to data.

if pid data exclude: *0x2XXYZZZZ, udata, mask, data[, data2]* Not applicable for Class 2

The **if pid data exclude** command is used to compare a specified pid with a range of data values specified in the script. If a true is returned, the script will continue to execute. If the command is called before the exec util command, a return of false will cause the task block to exit with a flash not needed state. If the command is called

Standalone Flash Programming Language	12/10/07	Page 13 of 20
---------------------------------------	----------	----------------------

after the exec util command, a return value of false will cause the task block to exit with a flash failure state. If data2 is omitted, data2 is assumed to be equal to data.

Argument	Description	Example
<i>0x2XX</i>	The controller's CAN diagnostic address	0x245 for a controller that uses 0x245 and 0x645
<i>0xY</i>	Opcode	
<i>0xZZZ</i>	Command	
<i>udata</i>	4 bytes of data	
<i>F</i>	24 byte Data buffer number (0-9)	5 = data buffer 5
<i>mask</i>	Up to 24 byte bit mask for data comparison	00000F00 = compare low nibble of the 3 rd byte with <i>data</i> and <i>data2</i>
<i>data</i>	Low end of the data range - up to 24 bytes of data (enter as hex bytes without 0x)	00475800
<i>data2</i>	High end of the data range - up to 24 bytes of data (enter as hex bytes without 0x)	004758FF

2.3.5.6 Report File Commands

report labels: *Label1, Label 2...*

The **report labels** command defines the column labels for the data being written to the summary file. The VIN, Product Address and Pass\Fail labels are added automatically. The labels are only written to the summary file when it is created. Data will be appended to the summary file columns with every task block. You will need to delete the summary file from the SD card each time you need to change the summary file format.

report data: *F, print enable mask*

The **report data** writes the contents of a SAINT2 data buffer to the summary file. A 20 byte mask is applied to the data. Any non-zero byte mask value will cause the corresponding data byte to be written to the report file. Note: VIN, product address and Pass/Fail status is always reported at the end of each line. Each task block correlates to one line in the summary file. A new line is written at the end of each task block.

Argument description

Argument	Description	Example
<i>Label1, Label 2</i>	Column label to be printed in the summary file	SWID for "SWID"
<i>F</i>	20 byte Data buffer number (0-9)	5 = data buffer 5
<i>print enable mask</i>	Up to 20 byte print enable mask	FFFFFFFF to print the first 4 bytes

3 Re-flash Output Files

The Re-flash process creates two output files: the defined report file and a summary file.

The report file is created with the name defined in the report file command in the script. This file contains some high level information about the re-flash process. Currently, it is mainly used for debug. This file is overwritten at the start of each new task block.

The summary file, *summary.csv* is created during the re-flash process when it does not exist on the card. The summary file is used to report any script defined data (report data commands), the VIN, the re-flashed controller's target address, and the result of the re-flash. The data column labels (report labels command) will be written when the file is created. The summary file will be appended with re-flash information during each task block. Data will continue to be appended unless the *summary.csv* file is deleted from the SD card so that a new summary file will be created.

Standalone Flash Programming Language	12/10/07	Page 15 of 20
---------------------------------------	----------	----------------------

4 Error Reporting

All errors are written to the designated report file. If an error occurs and the report file is not determined the error information is written to a default report file. Error reporting is done in two levels, a top level generic error ID (Error Code) and a more detailed specific value (Sub Code).

4.1 Archive Utility File Related Errors

The following error codes apply to both Class2 and CAN. They relate to the SAINT2 processing of the utility file used within the DPS archive.

Error Code	Sub Code	Definition
0x03	interpreter step	Process exited the utility file through the \$EE opcode
0x21	header protocol	Bad syntax in interpreter header definition of protocol - check your utility file
0x22	interpreter opcode	This interpreter opcode is incorrect or is unsupported - check your utility file
0x28	interpreter step	Bad syntax in interpreter step goto fields (G0 - G9) - check your utility file
0x29	interpreter step	Bad syntax in interpreter step action fields (AC0 - AC3) - check your utility file
0x30	interpreter step	Software error in the utility engine goto function - contact the SAINT2 team
0x31	index	Software error in the utility engine - contact the SAINT2 team
0x5A	device ID	The target device does not respond on the bus - check your bus connections

4.2 Class 2 Errors

The following errors can occur when the specified protocol is Class2. The defined report file will contain textual information explaining which script command was being executed when the error occurred. An error code number is also reported. This error code serves as a diagnostic as to what exactly went wrong.

Standalone Flash Programming Language	12/10/07	Page 16 of 20
---------------------------------------	----------	----------------------

4.2.1 Class 2 Util File Errors

Errors associated with the processing of the utility file commands and Class messages have a generic ID of 0x25.

Specific Error Codes

Value	Description
0x12	Unable to Open Util File. See report file for specific SD card error.
0x41	Unknown Opcode
0x42	No matching calibration file
0x43	Unable to Get VIN
0x71	Seed and Key Error
0x81	Error reading util file routine info
0x82	Error in Mode \$3B reading SD card. See report file for specific SD card error.
0x83	Error in Mode \$3B file seeking in SD card. See report file for specific SD card error.
0x84	Error in Mode \$AE with SD card access. See report file for for specific SD card error.
0x85	Error in Mode \$40 with SD card access. See report file for for specific SD card error.

4.2.2 Class 2 Script Errors

Errors associated with script file have a generic ID of 0x20.

Specific Error Codes

Value	Description
0x44	Invalid Function
0x46	Incorrect number of function arguments

4.2.3 Class 2 Bus Errors

Errors associated with the messages on the Class2 message bus have a generic ID of 0x7F.

Specific Error Codes

Value	Description
0x45	Error Exting Diagnostics
Values than 0x45	Actual Class 2 \$7F response code

4.3 GMLAN Errors

Standalone Flash Programming Language	12/10/07	Page 17 of 20
---------------------------------------	----------	----------------------

Protocol Definition Errors

Error Code	Sub Code	Definition
0x26		The protocol specified in the second argument of the protocol statement is unsupported.
0x27		The protocol specified in the second argument of the protocol statement is disabled for this hardware.
	0x00	GMLAN
	0x04	CustomAPM
	0xLL	reserved

Script Command Errors

Error Code	Sub Code	Definition
0x20		Error in the script command
	0x20	This command is not supported for the protocol specified in the script
	0x21	This command contains an invalid argument

SD Card File Access Errors

Error Code	Sub Code	Definition
0x10		Error opening a file on the SD card
0x11		Error reading a file on the SD card
0x12		Error opening the utility file on the SD card
0x13		Error reading the utility file on the SD card
	0x02	No such file or directory
	0x05	I/O Error
	0x09	Bad file number
	0x13	Permission denied
	0x17	File Exists
	0x19	No such device
	0x22	Invalid Argument
	0x24	Too many files open
	0x28	No space left on device
	0x30	Read only file system
	0x80	Data file has bad format

Script Command Errors

Error Code	Sub Code	Definition
0x41		Error during CAN communication
0x50		Error during Diagnostic Mode \$04
0x51		Error during Enter Programming Mode
0x52		Error during Diagnostic Mode \$1A - DID
0x53		Error during Diagnostic Mode \$27 - Security Unlock

Standalone Flash Programming Language	12/10/07	Page 18 of 20
---------------------------------------	----------	---------------

0x57		Error during Diagnostic Mode \$3B - Write DID
0x59		Error during Enter Diagnostic commands
0x5B		Error during Diagnostic Mode \$14 - Read DTCs
0x5C		Error during Enter Diagnostics before executing the utility file
0x5D		Error when reading diagnostic addresses (\$1A B0) after executing utility file
0x5F		Error during Read DPID
0x61		Error during read memory command
0x64		Error during command to enter manufacturing diagnostics or a PID request
0x67		Error during Enter Manufacturing Diagnostics Command
	0x02	CAN RX FIFO is not configured properly - contact the SAINT2 team
	0x03	The CAN FIFO has been overrun
	0x04	The CAN FIFO has been corrupted
	0x10	The number of DTCs reported overflows the buffer
	0x5A	The requested target ID is not on the bus
	0x90	Check for CAN bus errors - Is script baud rate correct? Is product powered? Is bus connected?
	0x91	Controller responded with a \$7F - negative response
	0x92	Controller responded with an incorrect or unexpected response
	0x93	Controller did not respond
	0x94	SAINT2 software error

Universal Diagnostic Command Errors

Error Code	Sub Code	Definition
0x62		Error during a Universal Diagnostics Command
	0x00	No error
	0x01	Command not supported
	0x02	Command failed
	0x03	Data out of range
	0x04	Data not writeable
	0x90	Check for CAN bus errors - Is script baud rate correct? Is product powered? Is bus connected?
	0x91	Controller responded with a \$7F - negative response
	0x92	Controller responded with an incorrect or unexpected response
	0x93	Controller did not respond
	0x94	SAINT2 software error

Utility File Execution Errors

Error Code	Sub Code	Definition
0x03	interpreter step	Process exited the utility file through the \$EE opcode
0x21	header protocol	Bad syntax in interpreter header definition of protocol - check your utility file

Standalone Flash Programming Language	12/10/07	Page 19 of 20
---------------------------------------	----------	----------------------

0x22	interpreter opcode	This interpreter opcode is incorrect or is unsupported - check your utility file
0x28	interpreter step	Bad syntax in interpreter step goto fields (G0 - G9) - check your utility file
0x29	interpreter step	Bad syntax in interpreter step action fields (AC0 - AC3) - check your utility file
0x30	interpreter step	Software error in the utility engine goto function - contact the SAINT2 team
0x31	index	Software error in the utility engine - contact the SAINT2 team
0x5A	device ID	The target device does not respond on the bus - check your bus connections
0x90	interpreter step	Check for CAN bus errors - Is script baud rate correct? Is product powered? Is bus connected?
0x91	interpreter step	Controller responded with a \$7F - negative response
0x92	interpreter step	Controller responded with an incorrect or unexpected response
0x93	interpreter step	Controller did not respond
0x94	interpreter step	SAINT2 software error