Flash Sequence Syntax Document

Version	Description	Author By
1.0	Initial Version	Mukund Sutrave

Contents

1 Introduction

This document explains the FSQ Syntax that would be used by the eFLASH flash interpreter.

1. FSQ Syntax

1.1 Key Variables

The FSQ interpreter goes thru the FSQ file line by line and executes them. The following are the key variables that are created by the interpreter before starting to run the interpreter.

The eFLASH tool converts the HEX / S19 files to a JSON file before using it with the interpreter. The JSON file created would be of the following format:


```
{"NoOfSectors":4,"SectorData":[{"JsonStartAddress":"80004000","JsonEndAddress
":"80013FFF","ECUMemMapStartAddress":"80004000","ECUMemMapEndAddress":"80013F
FF","JsonCheckSum":"21CE","JsonData":"xxxx"},{"JsonStartAddress":"80000000","
JsonEndAddress":"80003FFF","ECUMemMapStartAddress":"80000000","ECUMemMapEndAd
dress":"80003FFF","JsonCheckSum":"DCA7","JsonData":"xxxx"},{"JsonStartAddress
":"80080000","JsonEndAddress":"8017FFFF","ECUMemMapStartAddress":"80080000","
ECUMemMapEndAddress":"8017FFFF","JsonCheckSum":"A317","JsonData":"xxxx"},{"Js
onStartAddress":"80020000","JsonEndAddress":"8007FFFF","ECUMemMapStartAddress
":"80020000","ECUMemMapEndAddress":"8007FFFF","JsonCheckSum":"4F95","JsonData
":"xxxx"}]}
```


The following variables related to this json would be made available to the fsq file:

1. `NoOfSectors ---- (type – Integer)`
2. `JsonStartAddress[0.. NoOfSectors-1] ---- (type – U8 array)`
3. `JsonEndAddress[0.. NoOfSectors-1] ---- (type – U8 array)`
4. `JsonCheckSum[0.. NoOfSectors-1] ---- (type – U8 array)`
5. `JsonData[0.. NoOfSectors-1] ---- (type – U8 array)`

Every line of the fsq file has the following format:

<command>:<parameters>

Any FSQ file would have 3 sections namely:

1. Datafile Conversion Configuration
2. Dongle Configuration
3. Flashing Sequence

Each section can have the following commands supported:

1.2 Datafile Conversion Configuration

1.2.1 Command – ECUMapFile

	Cmd
Syntax	ECUMapFile:start_address_in_hex,end_address_in_hex,priority

	- This input is used by the FBDatasetJSONBuilder function. We can have one or more of these records in the beginning of the FSQ file and these records mention the sectors addresses that need to be considered for the json creation. If these records are not found in the FSQ file, sectors would be created completely based on the data that is available in the dataset.hex / s19 file. If these records are present, the sectors are created strictly based on the start addresses + end address mentioned here. Any data in the dataset file which is not within any of these start to end addresses would not be considered for the sector creation.
Example	EcuMapFile: 090c0000,0917FFFF,1 EcuMapFile: 09080000090BFFFF,2

1.2.2 Command – chksum

	Cmd
Syntax	chksum:chksumalgoENUM - This input is used by the FBDatasetJSONBuilder function and mentions the checksum algorithm that needs to be used to calculate the JsonCheckSum[i].
Example	chksum:FLETCHER chksum:CRCCCITT16

1.3 Dongle Configuration

1.3.1 Command – protocol

This command is used to set the protocol in the dongle:

	Configuration Cmd
Syntax	protocol:protocol_enum
Example	protocol:ISO15765_500K_11BIT_CAN

protocol_enum can have the following values:

ISO15765-250KB-11BIT-CAN
ISO15765-250Kb-29BIT-CAN
ISO15765-500KB-11BIT-CAN
ISO15765-500KB-29BIT-CAN
ISO15765-1MB-11BIT-CAN
ISO15765-1MB-29BIT-CAN
250KB-11BIT-CAN
250Kb-29BIT-CAN
500KB-11BIT-CAN

500KB-29BIT-CAN
1MB-11BIT-CAN
1MB-29BIT-CAN
OE-IVN-250KBPS-11BIT-CAN
OE-IVN-250KBPS-29BIT-CAN
OE-IVN-500KBPS-11BIT-CAN
OE-IVN-500KBPS-29BIT-CAN
OE-IVN-1MBPS-11BIT-CAN
OE-IVN-1MBPS-29BIT-CAN

1.3.2 Command – txid

This command is used to set the TXID in the dongle:

	Configuration Cmd
Syntax	txid:txid2byte4byte
Example	txid:07E0 txid:18DA00F1

1.3.3 Command – rxid

This command is used to set the RXID in the dongle:

	Configuration Cmd
Syntax	rxid:rxid_2byte_4 byte
Example	rxid:07E0 rxid:18DA00F1

1.3.4 Command – startpadding

This command is used to start padding of CAN Messages:

	Configuration Cmd
Syntax	strtpadding:paddingbyte_in_hex
Example	strtpadding:00

1.3.5 Command – stoppadding

This command is used to start padding of CAN Messages:

	Configuration Cmd
Syntax	Stoppadding
Example	Stoppadding

1.3.6 Command – strtTP

This command is used to start padding of CAN Messages:

	Configuration Cmd

Syntax	strtTP
Example	strtTP

1.3.7 Command – stopTP

This command is used to start padding of CAN Messages:

	Configuration Cmd
Syntax	stopTP
Example	stopTP

1.3.8 Command – setT2MAX

This command is used to set the max time between response (valid positive / negative response except for NACK78) to the next request message, after which the dongle will send a no message response to the tool.

	Configuration Cmd
Syntax	setT2MAX:time_in_hex_ms
Example	setT2MAX:2710

1.3.9 Command – setstmin

	Cmd
Syntax	setstmin:time_in_ms_hex - This is the time gap that the dongle would maintain between two consecutive frames. This stmin if set, would override the STMIN coming from the ECU (as a part of the flow control frame) - Default value of STMIN would be 0xFF. If no STMIN is set or if it is set to 0xFF, the ECU STMIN would be considered as the interframe deley. - If the STMIN is set to any value other than 0xFF, the STMIN from ECU would be overridden by the value set. - STMIN <= 127 (separation time in ms) - 0xF1 <= STMIN <= 0xF9 corresponds to 100 – 900 microsecs
Example	setstmin:14 – sets it to 20ms setstmin:F2 – sets it to 200microsecs setstmin:FF – sets to default. Dongle would consider the stmin provided by ECU

1.4 Flashing Sequence

1.4.1 Command – send

This command is used to send data send request to the dongle:

	Data Cmd
Syntax	send:payload
Example	send:1002

	send 1002 to the ECU send:3101020201+<i,1> Here the variable i of 1 byte in hex is sent. 3101FF01+<json_strt_addr[i],4>+<json_end_addr[i],4>+<json_checksum[i],2> send:3B99+<DAYBCD,1>+<MONTHBCD,1>+<YEARBCD,1> - DAYBCD, MONTHBCD, YEARBCD are constants available to be used in fsq. These would be replaced with the BCD values of the current date parameters

1.4.2 Command – repeatstart and repeatend

This is a loop feature that can be executed inside the fsq file. We can have multiple loops in a file referenced by the loop handle

	Configuration Cmd
Syntax	repeatstart:repeathandle,var,var_index_init,var_index_max - used to start the loop - loop starts with variable with initial index value as var_index_init until variable has reached the max index value of var_index_max - The var would be available to be used within repeatstart and repeatend for any kind of calculation or operation such as below: - send:3101020201+<i,1>
Example	repeatstart:1,i,0,no_of_sectors …… repeatend:1

	Configuration Cmd
Syntax	repeatend:repeathandle - used to end the loop with repeathandle
Example	repeatstart:1,i,0,no_of_sectors …… repeatend:1

1.4.3 Command – function

This command is used to execute some standard functions that are already known to the runtime

	Configuration Cmd
Syntax	function:functionname[parameters]
Example	function:CalculateKeyFromSeed[SEEDKEYINDEX, seedkeylen] parameters are predefined for functions. For example, there are 2 parameters for the CalculateKeyFromSeed function. This function is call in between a getseed (send:27xx) and sendkey (send:27xxkkkk..). It takes the seed from the response of previous send, calculates key and sends it in the succeeding send.

1.4.4 Command – sendbulkdata

	Cmd
Syntax	sendbulkdata:seqvar,seqvar_init_value,sendarray [total_array_size_in_hex,blksize_in_hex]>> - ***blksize_in_hex*** number of bytes to be transferred from ***json_sectordata*** in one go, until ***total_array_size_in_hex*** number of bytes have completely been transferred. - ***seqvar*** would be incremented by 1 upon every bulk transfer. - ***seqvar*** would be initiated with the ***seqvar_initvalue*** to start with - sendarray can have seqvar and json_sectordata used as variables
Example	sendbulkdata:bsc,1,36+bsc+json_sectordata[8877ff, 212] sendbulkdata:bsc,1,36+json_sectordata[8877ff, 212]

1.5 Keywords used in Flash Sequence Section

Here are the keywords that can be used in the Flash sequence section to calculate values and send to ECU

1. DAYBCD – returns the current date in BCD format
2. MONTHBCD – returns the current month in BCD format
3. YEARLOWERBCD – returns the LS byte of the year in BCD format
4. YEARUPPERBCD – returns the MS byte of the year in BCD format
5. HOUR – returns the current hour in BCD format
6. MINBCD - returns the current minute in BCD format
7. SECBCD - returns the current second in BCD format

2. Annexure: Example FSQ File

// Datafile Conversion Configuration

EcuMapFile:<start_address,010c0000>+<end_address,0117FFFF>

EcuMapFile:<start_address,01080000>+<end_address,010BFFFF>

chksum:CRCC16

// Dongle Configuration

protocol:ISO15765_11BIT_500KBPS_CAN

txid:07e0

rxid:07e8

startpadding

// Flashing Sequence

send:1002

send:2705

function:CalculateKeyFromSeed[2,8,SEEDKEYINDEX,8]

send:2706+<key,8>

repeatstart:1,i,1,noofsectors

send:3101FF01+<json_strt_addr[i],4>+<json_end_addr[i],4>+<json_checksum[i],2>

send:3101FF0001+<i,1>

send:340044+<json_strt_addr[i],4>+<json_sector_len[i],4>

setstmin:7D0

setstmin:3C

sendbulkdata:bsc,1,36+bsc+json_sectordata [json_sector_len, 100]

send:37

send:3101020201+<i,1>

repeatend:1

send:2EF1990E070E00

send:1101

send:3B99+<DAYBCD,1>+<MONTHBCD,1>+<YEARBCD,1>