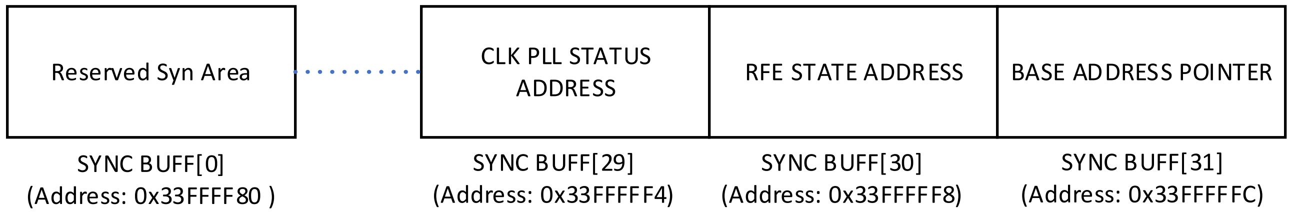
Sync Buffer


Reserved Syn Area
CLK PLL STATUS ADDRESS
RFE STATE ADDRESS
BASE ADDRESS POINTER
SYNC BUFF[0] (Address: 0x33FFFF80)
SYNC BUFF[29] (Address: 0x33FFFFF4)
SYNC BUFF[30] (Address: 0x33FFFFF8)
SYNC BUFF[31] (Address: 0x33FFFFFC)

In the shared system memory, 128 bytes section of memory starting from fixed address 0x33FFFF80 is reserved as Sync Buffer. The section of memory from address 0x33FFFF80 to 0x33FFFFF3 is reserved for future use. The 12 bytes section out of the Sync Buffer, starting from address 0x33FFFFF4 is used for synchronization. This 12 bytes section is divided into three 32 bit fields which are described below:

CLK PLL STATUS ADDRESS: This field is written by RFE FW as part of RFE CLK PLL initialization. The RFE FW writes 0xC0DE0001 when CLK PLL is ready. This status means that application running on APP-M7 can now initiate switching from FIRC clock to XOSC clock or to 640MHz (from CLK PLL). This means also that MIPICSI_0/Packet Processor Engine is ready to be configured, as this code also indicates the clock is enabled for those IPs. This field is of interest only during the boot, at boot FIRC is used as clock source because XOSC clock or 640MHz clock may not be stable then. This field must be zero initialized before the RFE FW is loaded and the Arm Cortex - M7 is released from reset.

RFE STATE ADDRESS: This field is initially cleared by RFE SW Driver as part of rfe_sync(). The field is updated with value as 0x01 by RFE FW during the initialization phase. This change is interpreted by RFE SW Driver as RFE FW is ready for synchronization.

BASE POINTER ADDRESS: This field is initially cleared by RFE FW during the synchronization phase. The field is then updated with value corresponding to shared memory command buffer address, by RFE SW Driver during rfe_sync(). This change is interpreted by RFE FW as RFE SW Driver is ready for synchronization. The RFE FW uses the address present in this field as the command buffer address, then derives the response buffer address (BASE POINTER ADDRESS + 256 bytes) and shared data buffer address (BASE POINTER ADDRESS + 512 bytes). After deriving the addresses, RFE FW transitions to INITIALIZED state, the state information is updated in shared data buffer which is read by RFE SW Driver. This completes the synchronization phase triggered via rfe_sync().

Remark: The address 0x33FFFF80 is chosen for the following reasons:

- It is located at the end of fixed memory section reserved for RADAR Application in Application System RAM and hence there is no effect of floater System RAM which can grow or shrink in size depending on configuration.
- Having sync buffer in this location helps to achieve low latency access by RFE FW and RADAR Application.

Shared Data Buffer

RFE FW STATE	RADAR CYCLE COUNT	CHIRP SEQUENCE COUNT	RESERVED FOR FUTURE USE	RESERVED FOR FUTURE USE	RESERVED FOR FUTURE USE	RESERVED FOR FUTURE USE	RESERVED FOR FUTURE USE
SHARED DATA_BUFF [0]	SHARED DATA_BUFF[1]	SHARED DATA_BUFF [2]	SHARED DATA_BUFF [3]	SHARED DATA_BUFF [4]	SHARED DATA_BUFF [5]	SHARED DATA_BUFF [6]	SHARED DATA_BUFF [7]

In the shared system memory, 32 bytes section of memory is marked for sharing the data from RFE FW. The data present here is written by RFE FW and read by RFE SW Driver. At present only 12 bytes of this memory section is used to represent RFE FW state, Radar Cycle Count, Chirp Sequence Count and the remaining memory is reserved for future use. The RFE SW Driver is responsible for initializing each of these buffers with value 0xFFFF0000 during synchronization as part of rfe_sync(). In each of the 32 bit value, the least significant 16 bits represent the actual value while remaining 16 bits are invert of first 16 bits. This is the data redundancy technique used. The RFE SW driver can compare the least significant 16 bits with most significant 16 bits to know if the data is not corrupted.

- RFE FW STATE: This field is updated by RFE FW on state change. Please refer the section "RFE Firmware FSM" described in SAF85xx RFE SW Reference Manual for details of state.

- RADAR CYCLE COUNT: This field is updated by RFE FW after completing each radar cycle (refer SAF85xx RFE SW Reference Manual for definition of radar cycle). The API rfe_getRadarCycleCount() makes use of this field to get the radar cycle count.

- CHIRP SEQUENCE COUNT: This field is updated by RFE FW after the chirp sequence during the active radar cycle. The API rfe_getRadarCycleCount() makes use of this field to get the chirp sequence count.

Command Buffer

CMD CNT / TRIGGER CNT (4 bytes)	CMD ID (4 bytes)	PAYLOAD LENGTH (4 bytes)	PAYLOAD[0] (4 Bytes)	....	PAYLOAD[n] (4 Bytes)	CRC (4 bytes)
CMD_BUFF [0]	CMD_BUFF [1]	CMD_BUFF [2]	CMD_BUFF [3]		CMD_BUFF [62]	CMD_BUFF [63]

Response Buffer in case of No Error

RSP CNT / TRIGGER CNT (4 bytes)	RSP ID (4 bytes)	RSP DATA LENGTH (4 bytes)	RSP DATA[0] (ERROR CODE =0x00000000) (4 Bytes)	....	RSP DATA[n] (4 Bytes)	CRC (4 bytes)
RSP_BUFF [0]	RSP_BUFF[1]	RSP_BUFF [2]	RSP_BUFF [3]		RSP_BUFF [62]	RSP_BUFF [63]

Response Buffer in case of ERROR

RSP CNT / TRIGGER CNT (4 bytes)	RSP ID (4 bytes)	RSP DATA LENGTH = 0x00000004 (4 Bytes)	ERROR CODE (4 Bytes)	....	CRC (4 bytes)
RSP_BUFF [0]	RSP_BUFF[1]	RSP_BUFF [2]	RSP_BUFF [3]		RSP_BUFF [63]

- Endianness: Little endian
- Command Buffer: The command consists of CMD ID, Payload Length and Payload. Some of the commands may not have payload and for such commands the payload length is set as 0x00. The two fields CMD CNT and CRC are used by RFE FW for detecting new command and for command integrity respectively.
    - CMD CNT / TRIGGER CNT: The 32 bit count which is incremented after every new command is placed in the buffer. The RFE FW will see this counter to know if there is a new command by comparing against the previous command counter stored in it's local memory. RFE SW Driver also uses this counter to know if the response in response buffer belongs to the current command or not, please refer RSP CNT.
    - CMD ID: The 32 bit ID of the command to be processed by RFE FW.
    - PAYLOAD LENGTH: The length of the payload (command parameters) to be shared to RFE FW via shared command buffer. This length is included in CRC calculation.
    - PAYLOAD: The payload (command parameters) to be shared to RFE FW via shared command buffer.
    - CRC: The 32 bit CRC is calculated over CMD CNT/TRIGGER CNT, CMD ID, PAYLOAD LENGTH and PAYLOAD. The RFE FW calculates CRC locally over these fields and compares against CRC value present in command buffer to know if the command integrity is preserved. The 32 bit CRC is always present at fixed location (63rd index) of the command buffer.
- Response Buffer: The response consists of RSP ID, RSP data length and RSP data. Unlike the commands, every response has at least one response data which is the ERROR CODE. The two fields RSP CNT and CRC are used by RFE SW Driver for detecting new response and for response integrity respectively.
    - RSP CNT / TRIGGER CNT: The 32 bit count which is incremented by RFE FW on copying response into response buffer. This value will be equal to CMD CNT if the command was processed by RFE FW. This value is used by RFE SW Driver to know if the RFE FW has responded to command it shared.
    - RSP ID: This is 32 bit response ID which has least 31 bits same as that of CMD ID while the 32nd bit has the inverted value of 32nd bit of CMD ID.
    - RSP DATA LENGTH: The length of response data shared to RFE SW Driver via shared response buffer.
    - RSP DATA: The response data from RFE FW. The first 4 bytes of response data (at RSP DATA[0]) consists of ERROR CODE. In case of error, the response data length will be 0x04 bytes because only ERROR CODE will be present as part of response data.
        - ERROR CODE: The response which indicates if RFE FW accepted the command or not. Please refer for the type "rfe_error_t" in SAF85xx RFE SW Reference Manual for the error code values returned.
    - CRC: The 32 bit CRC is calculated over RSP CNT/TRIGGER CNT, RSP ID, RSP DATA LENGTH and RSP DATA. The RFE SW Driver calculates CRC locally over these fields and compares against CRC value present in response buffer to know if the response integrity is preserved. The 32 bit CRC is always present at fixed location (63rd index) of the response buffer.

Remark:
1) The CRC used is CRC 32 with Polynomial 0x04C11DB7h. If input for CRC calculation is not 32 bit aligned then padding value is used to make it 32 bit aligned. The subsequent content (after command payload or response data) of respective command or response buffer itself is used as padding data at present. The padding value is always referred in this documents with size in bits.

2) Since these fields are used as triggers to RFE FW and RFE SW Driver respectively, only when the operation of copying command or response in respective memory and CRC is completed, the trigger count is incremented. This approach avoids the situation where data in command or response buffer is read even before the copying operation is completed.

3) Orange Block in subsequent representations in this document is used to represent cases when the parameter is less than 32 bits or when there are multiple parameters corresponding to certain index of command / response buffer. If the parameter size in this case is not mentioned then the size of 8 bits must be considered.

4) The payload length (command / response) does not include the size of padding value.

5) In the representation of typical command and response format for each API, the ERROR CODE is set as "0x00000000" which indicates that there was no error.

6)There is a shadow command buffer maintained in RFE SW Driver. The RFE SW Driver first prepares the command and calculates CRC over complete command. Once the command is prepared, the command including CRC is copied from shadow buffer into command buffer. During this copy operation it is important to note that the CMD CNT / TRIGGER CNT is always copied at the last, this way it will be guaranteed that RFE FW sees the new command via CMD CNT/ TRIGGER CNT only when complete command has been written into command buffer. In the similar way, RFE FW uses shadow response buffer. During the copy operation from response shadow buffer to response buffer, the RSP CNT / TRIGGER CNT is copied at the last.

Command IDs

rfe_configure()	0x00000000
rfe_radarCycleStart()	0x00000001
rfe_radarCycleStop()	0x00000002
rfe_getFuSaFaults()	0x00000003
rfe_getTime()	0x00000004
rfe_getVersion()	0x00000005
rfe_monitorRead()	0x00000006
rfe_getNextRadarCycleStartTime()	0x00000007
rfe_setNextRadarCycleStartTime()	0x00000008
rfe_updatePush()	0x00000009
rfe_continuousWaveTransmissionStart()	0x0000000A
rfe_continuousWaveTransmissionStop()	0x0000000B
rfe_testSetParam()	0x0000000C
rfe_getFuSaFaultStatistics()	0x0000000D
rfe_getBistZeroHourReferenceData()	0x0000000E
rfe_testGetInternalError()	0x0000000F
rfe_configureInterrupt()	0x00000010

rfe_sync()

The rfe_sync() function is the first function of RFE SW Driver to be called by the application. This function does not result in any command exchange with RFE FW, instead it performs actions which are required to Synchronize with RFE FW. The following are the actions which happen in RFE SW Driver and in RFE FW during the synchronization process:

- RFE SW Driver initializes the shared data buffer with value 0xFFFF0000 , sets the “RFE STATE ADDRESS” field to 0x00 and keeps polling for this field to check if RFE FW has changed the value to 0x01.

- During the synchronization process within the RFE FW, the “BASE ADDRESS POINTER” field is set to 0x00. The RFE FW keeps updating the “RFE STATE ADDRESS” as 0x01 in loop till “BASE ADDRESS POINTER” has any non zero value.

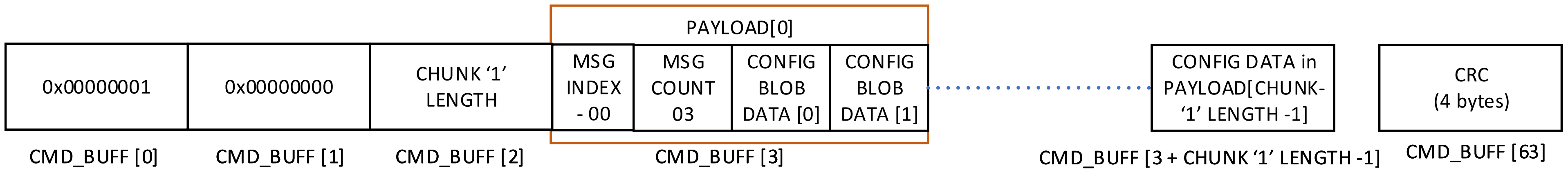
- The change of “RFE STATE ADDRESS” value to 0x01 by RFE FW, is interpreted by RFE SW Driver as “RFE FW is ready for Synchronization”. The RFE SW Driver sets the “BASE ADDRESS POINTER” to value which corresponds to the Shared Data Buffer Address.

- The RFE FW sees the “BASE ADDRESS POINTER” as non zero value and uses this value to derive locally the addresses: Command Buffer, Response Buffer and Shared Data Buffer. Upon deriving addresses, RFE FW updates the state in shared data buffer as “Initialized”.

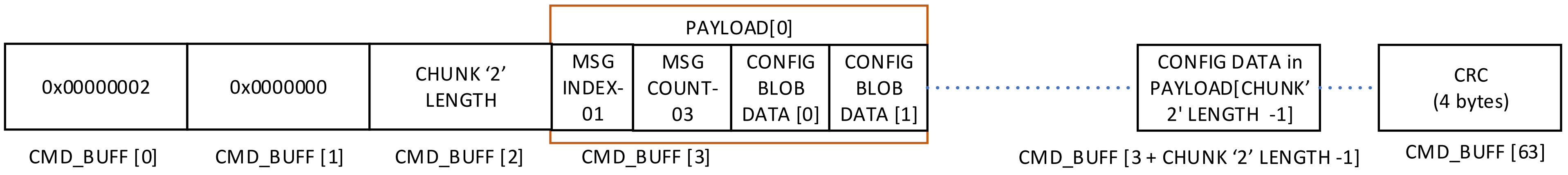
- The RFE SW Driver looks for the state change and upon seeing the state as Initialized, considers the Synchronization process as complete.

rfe_configure(): An example as per the configure command is shown below

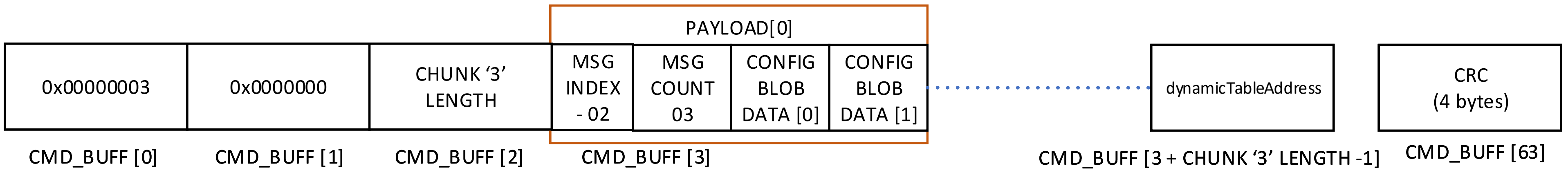
Command Chunk 1



Command Chunk 2



Command Chunk 3(Last)



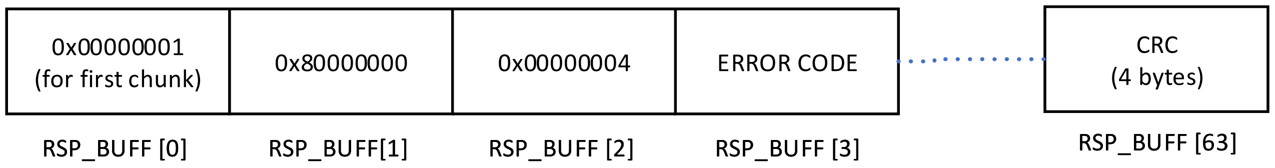
The configuration BLOB data consists of data more than 256 bytes, hence it cannot be sent in single transfer. The entire BLOB data is divided and sent in chunks. The RFE FW accepts command even if the chunks are of smaller size as long as it has valid format, size and pa rameters. The last 32 bits of the last chunk contain the shared memory address of Dynamic Table. The Dynamic table data is directly fetched by RFE FW from shared memory buffer whose address is indicated in command.

RFE FW estimates the configuration data size based on the number of chirp profile and number of chirp sequence to configure i n metadata section of RFE config BLOB. If the actual received data size does not match the estimated data size, RFE FW will return "rfe_error_api_invalidConfigurationSize_e" error.

The MSG Index indicates the chunk number and MSG COUNT indicates the total number of chunks to be expected by RFE FW. The MSG Index is incremented on every chunk while MSG COUNT remains same. The RFE FW validates the correctness of MSG INDEX and MSG CNT, if its not valid then an error response is reported. This also means, if chunks are out of order they will be rejected. If RFE FW receives less number of chunks than indicated, it continues to stay in Initialized state and handle any new commands which are allowed in Initialized state. The RFE SW Driver must send complete Configuration Data to RFE FW, for the Configuration step to begin.

Remark:
1) If RFE FW is in Configured state, on receiving the first chunk of Configure command, RFE FW transitions to Initialized state.

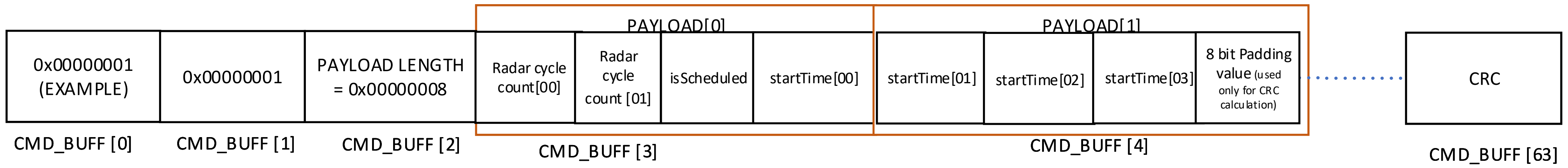
Typical Response of rfe_configure()



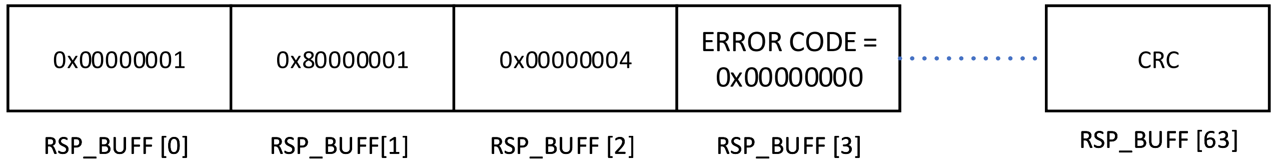
The Response for each of the chunks will be in format as above.

rfe_radarCycleStart()

Command

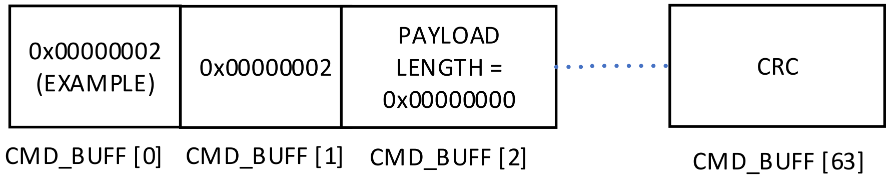


Response

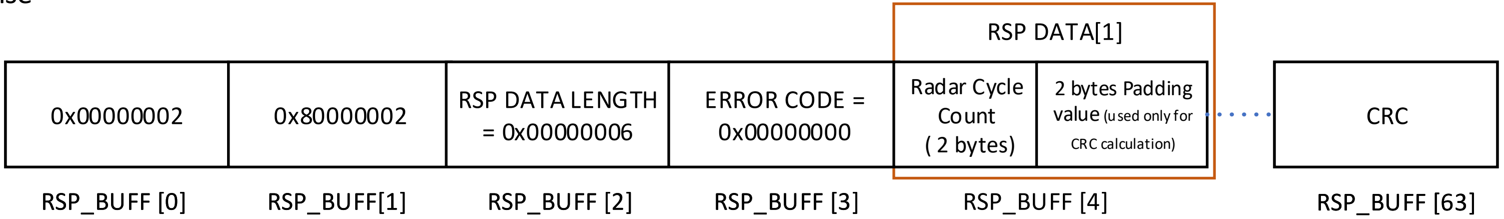


rfe_radarCycleStop()

Command

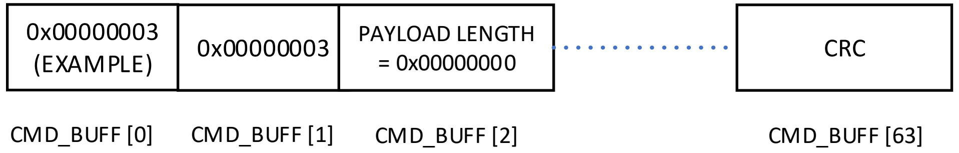


Response

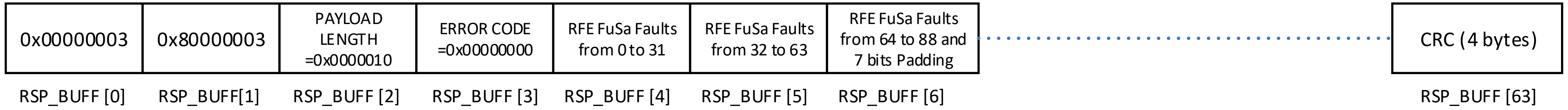


rfe_getFuSaFaults()

Command

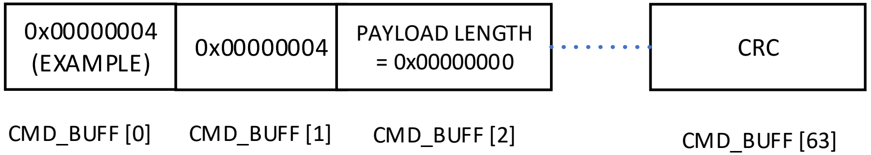


Response

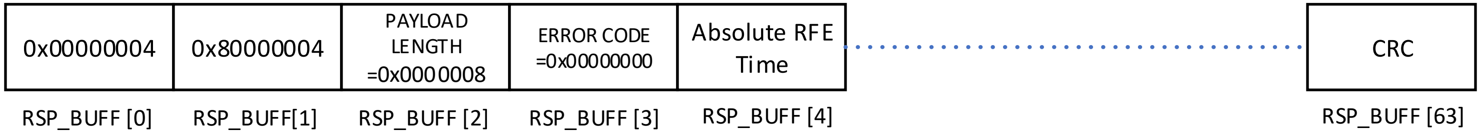


rfe_getTime()

Command

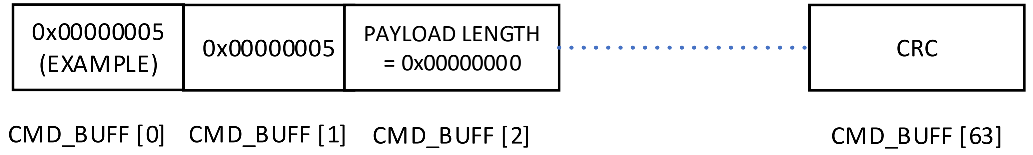


Response

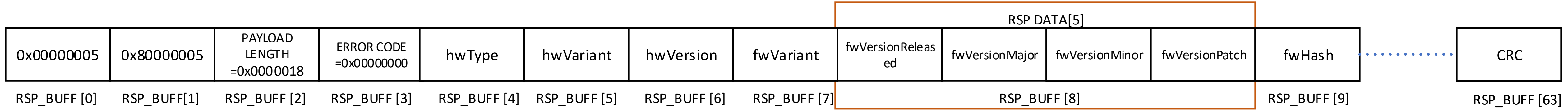


rfe_getVersion()

Command

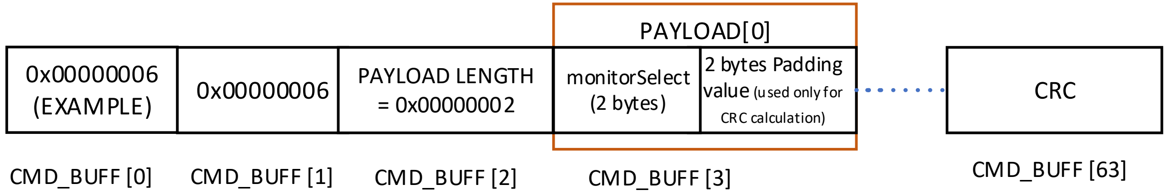


Response

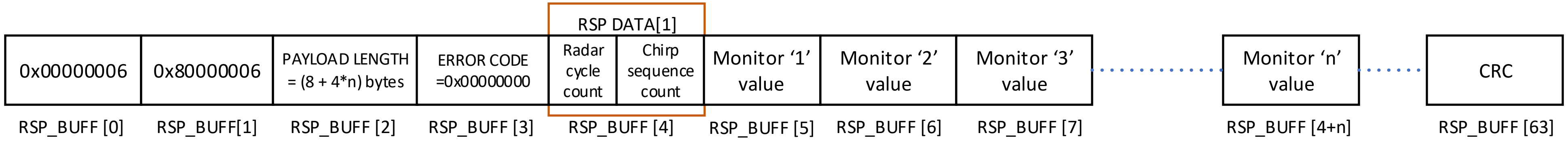


rfe_monitorRead()

Command



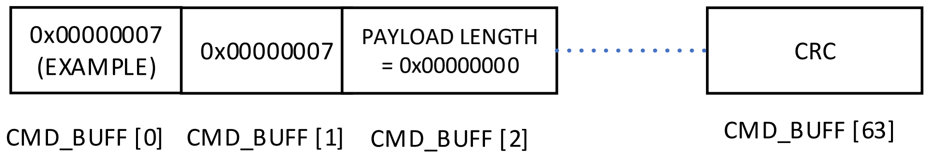
Response



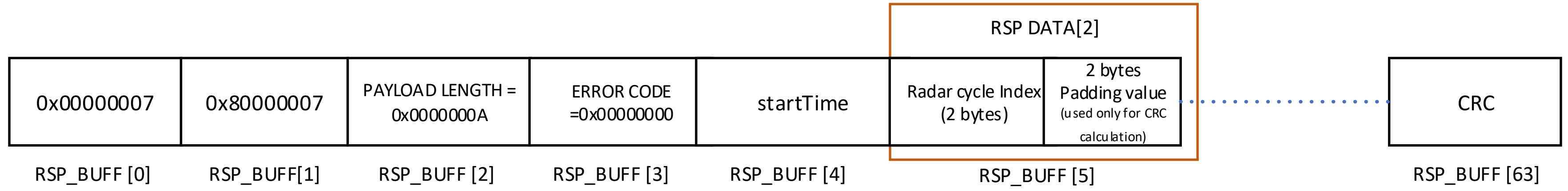
Please refer SAF85xx RFE SW Reference Manual for details on supported Monitor values. The “monitorSelect” uses bit field representation for the monitors. Depending on this value there can be “n” monitor values whose value will be placed in response buffer by RFE FW.

rfe_getNextRadarCycleStartTime()

Command

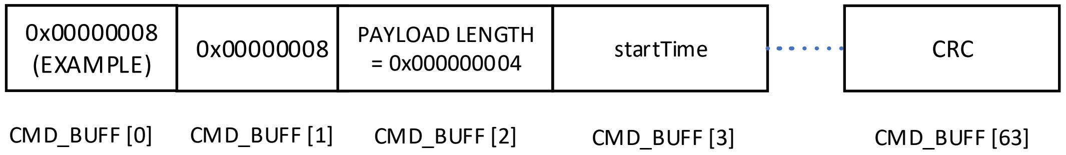


Response

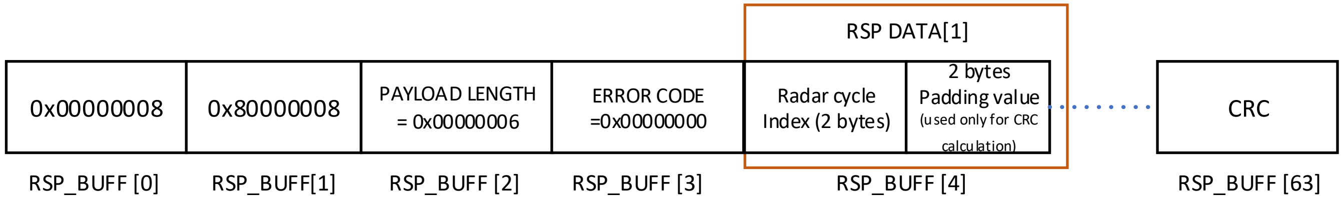


rfe_setNextRadarCycleStartTime()

Command

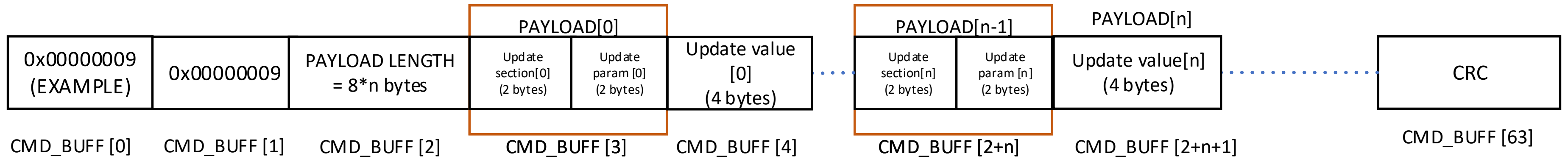


Response

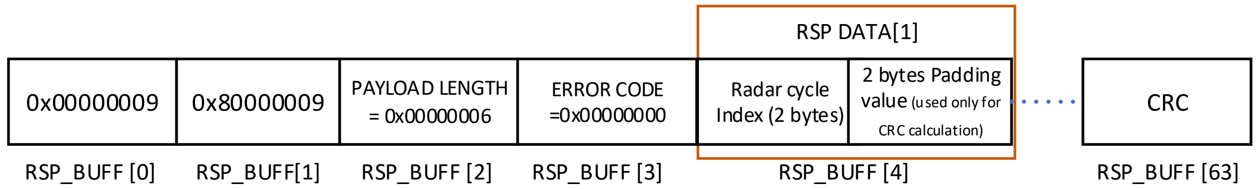


rfe_updatePush()

Command


0x00000009 (EXAMPLE)
0x00000009
PAYLOAD LENGTH = 8*n bytes
PAYLOAD[0]
Update section[0] (2 bytes)
Update param [0] (2 bytes)
Update value [0] (4 bytes)
PAYLOAD[n-1]
Update section[n] (2 bytes)
Update param [n] (2 bytes)
PAYLOAD[n]
Update value[n] (4 bytes)
CRC
CMD_BUFF [0]
CMD_BUFF [1]
CMD_BUFF [2]
CMD_BUFF [3]
CMD_BUFF [4]
CMD_BUFF [2+n]
CMD_BUFF [2+n+1]
CMD_BUFF [63]

Response



Remarks:
1) In one command it is possible to update maximum of 16 parameters.

2) If there is active radar cycle the RFE FW does not apply these updates immediately on receiving the rfe_updatePush command, instead these are applied at a fixed time interval during the end of radar cycle. Until the updates are applied, next update push is not possible unless rfe_radarCycleStop command is called.

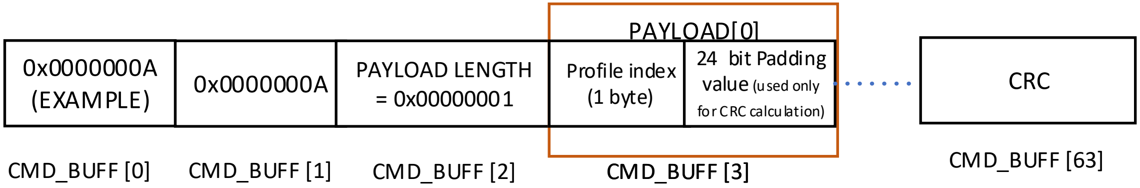
3) If there is no active radar cycle (example: Main FSM is in configured state), then the updates are applied upon receiving the rfe_updatePush command.

4) The rfe_radarCycleStop command clears the pending updates on RFE FW side.

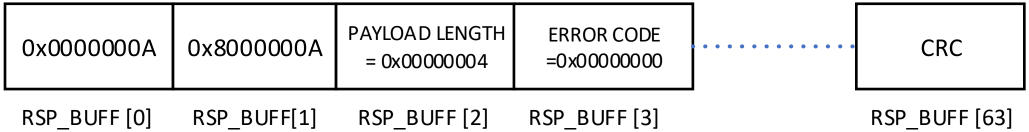
5) In case the last radar cycle is being executed, the rfe_updatePush command will report error.

rfe_continuousWaveTransmissionStart()

Command

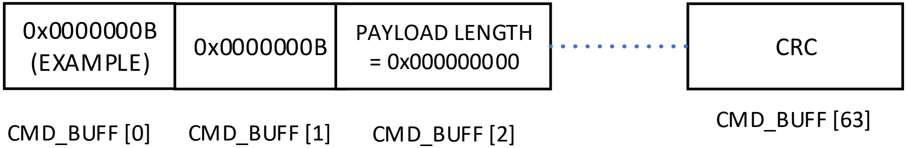


Response

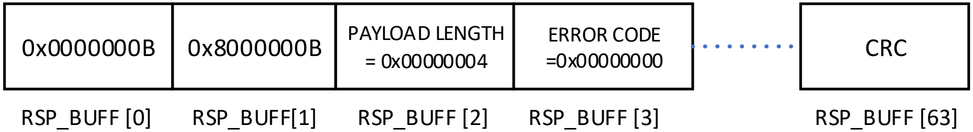


rfe_continuousWaveTransmissionStop()

Command

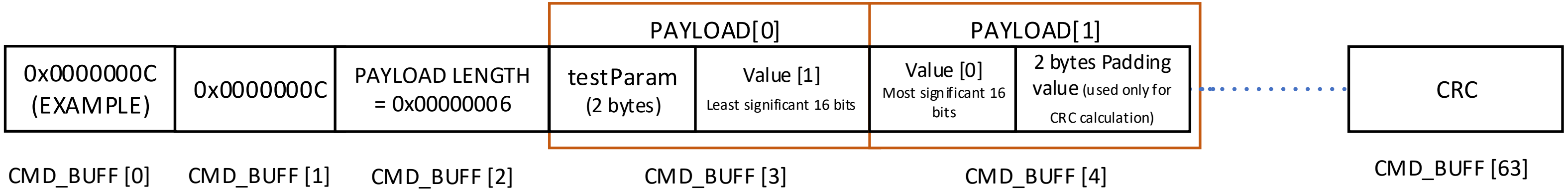


Response

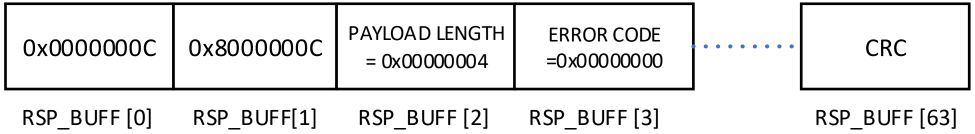


rfe_testSetParam()

Command

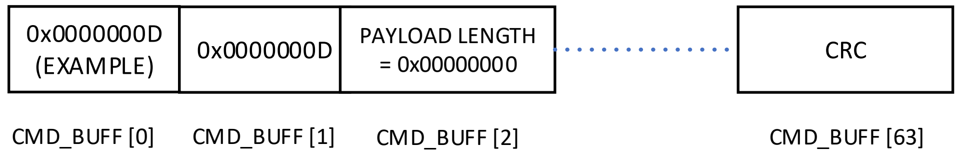

PAYLOAD[0]
PAYLOAD[1]
0x0000000C (EXAMPLE)
0x0000000C
PAYLOAD LENGTH = 0x00000006
testParam (2 bytes)
Value [1] Least significant 16 bits
Value [0] Most significant 16 bits
2 bytes Padding value (used only for CRC calculation)
CRC
CMD_BUFF [0]
CMD_BUFF [1]
CMD_BUFF [2]
CMD_BUFF [3]
CMD_BUFF [4]
CMD_BUFF [63]

Response

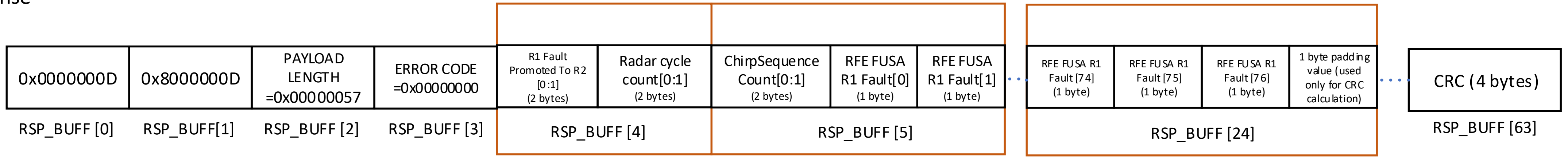


rfe_getFuSaFaultStatistics()

Command



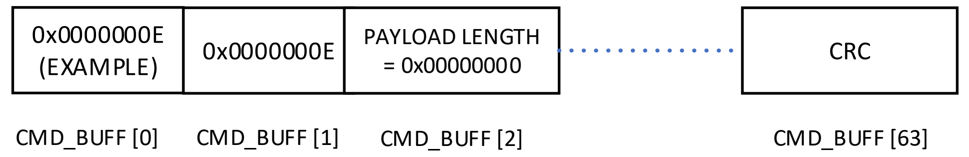
Response



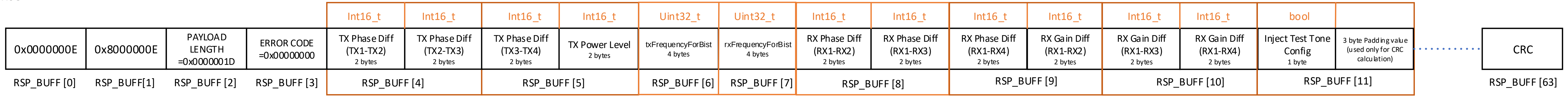
0x0000000D
0x8000000D
PAYLOAD LENGTH =0x00000057
ERROR CODE =0x00000000
R1 Fault Promoted To R2 [0:1] (2 bytes)
Radar cycle count[0:1] (2 bytes)
ChirpSequence Count[0:1] (2 bytes)
RFE FUSA R1 Fault[0] (1 byte)
RFE FUSA R1 Fault[1] (1 byte)
RFE FUSA R1 Fault [74] (1 byte)
RFE FUSA R1 Fault [75] (1 byte)
RFE FUSA R1 Fault [76] (1 byte)
1 byte padding value (used only for CRC calculation)
CRC (4 bytes)
RSP_BUFF [0]
RSP_BUFF[1]
RSP_BUFF [2]
RSP_BUFF [3]
RSP_BUFF [4]
RSP_BUFF [5]
RSP_BUFF [24]
RSP_BUFF [63]

rfe_getBistZeroHourReferenceData()

Command

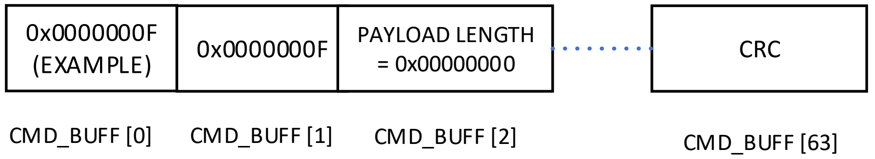


Response

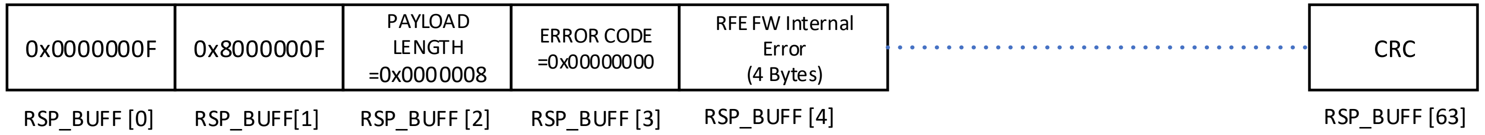


rfe_testGetInternalError()

Command

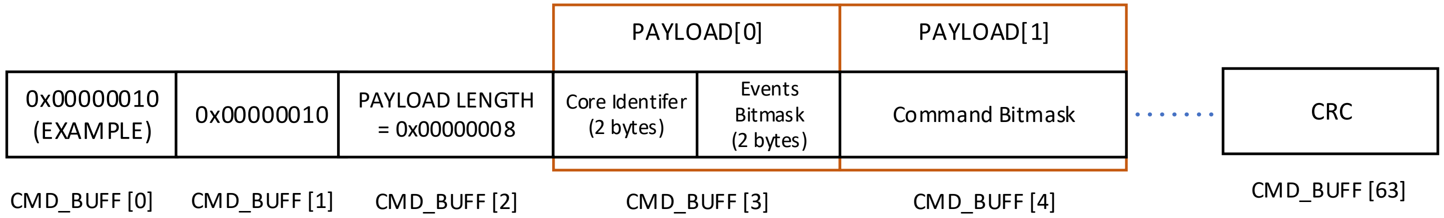


Response

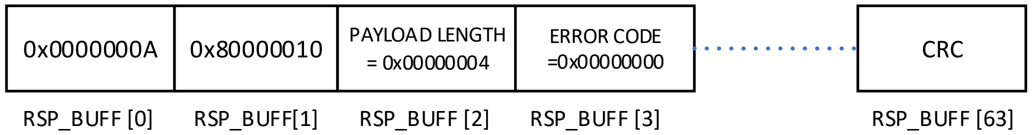


rfe_configureInterrupt()

Command



Response



Core Identifier: Core which will receive interrupts from RFE-FW:

0 : None (Disables RFE-FW interrupts).

1 : RFE-FW sends configured interrupts to appM7 core.

2 : RFE-FW sends configured interrupts to a53 core.

Events Bitmask:

Bitmask of events that will make RFE-FW raise an interrupt, each bit is mapped to an element of the shared data buffer.

Command Bitmask:

Bitmask of commands that will make RFE-FW raise an interrupt upon response, each bit is mapped to the respective command id.

Example : Requesting interrupts for rfe_configure() and rfe_monitorRead(): 0x00000041

Recommendations

- CMD CNT /TRIGGER CNT must always be updated for every new command in steps of 0x01. If an out of order CMD CNT is seen by RFE FW, an error is reported with expected CMD CNT. The RFE SW Driver must send the command with the expected CMD CNT for it to be back in Sync with RFE FW.

- On detecting CRC errors the RFE SW Driver must send the command again with updated CMD CNT / TRIGGER CNT.

- To be sure data is read from buffers only after completing write operation, use non-cached and volatile buffers.

- The RFE FW does not have command queue hence the RFE SW Driver must send next command only when the response for current command is received. In scenario where RFE SW Driver sends the command even before response is sent, RFE FW will handle this new command after responding to current command.

- The dynamic table is accessed during rfe_configure() and rfe_updatePush(). The Application has means to know if the updates were applied, until the updates are applied the dynamic table must not be touched by Application .
  - If rfe_configure() is called then Application must not touch this table until it receives successful response.
  - If rfe_updatePush() is called when there is no radar cycle, Application must not touch this table until it receives successful response
  - If rfe_updatePush() is called when there is active radar cycle, Application must not touch until it sees an update in radar cycle count.