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

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

- Endianness: Little endian
- Command Buffer:
  - CMD CNT / TRIGGER CNT: The 32 bit count which is incremented on every new command. 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 indirectly 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.
  - PAYLOAD: The payload (command parameters) to be shared to RFE FW via shared command buffer.
  - CRC: The 32 bit CRC calculated over CMD CNT, CMD ID, PAYLOAD LENGTH, PAYLOAD. The RFE FW calculates CRC locally over the command and compares against this value present in command buffer to know if command integrity is good. If the actual command data does not satisfy the 32 bit alignment, then the data contained in subsequent(/remaining) command buffer will be used as padding data to make it 32 bit aligned.
  - Response Buffer:
    - 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 first 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 consisting of API return values.**
    - ERROR CODE: The response which indicates if RFE FW accepted the command or not. Please refer SAF85xx RFE Abstract API Manual for error code values returned.
    - CRC: 32 bit CRC calculated over RSP CNT, RSP ID, RSP DATA LENGTH, RSP DATA. The CRC is always calculated on 32 bit aligned data. If the actual response data does not satisfy this, then the data contained in subsequent(/remaining) response buffer will be used as padding data to make it 32 bit aligned.

Remark:
1) If the size of entries in command or response format is not mentioned then it must be treated as 32 bit in size, if not the clear information of size will be captured with the entry.
2) If input for CRC calculation is not 32 bit aligned then padding value is used to make it 32 bit aligned. The content of buffer itself is used as padding data at present. The padding value is always referred with size in bits.
3) The payload length (command / response) does not include the size of padding value.
4) Orange Block is used to represent cases when the parameter is less than 32 bits or when there are two parameters corresponding to certain index of command / response buffer.

CHECK IF COMPLIMENT IS BETTER WORD COMPARED TO INVERT

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]

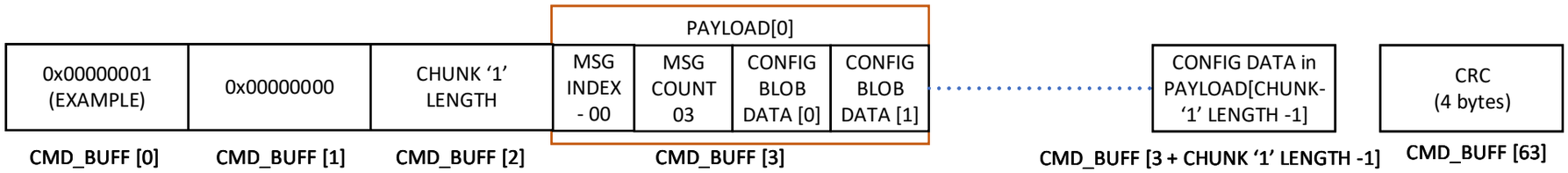
- In the shared system memory, 32 bytes of memory is marked for sharing the data from RFE FW. The data present here is read by RFE SW Driver. At present only 12 bytes are memory is used to represent RFE FW state, Radar Cycle Count, Chirp Sequence Count and the remaining memory is reserved for future use.
- In each of the 32 bit value, the first 16 bits represent the actual value while remaining 16 bits are invert of first 16 bits.

- **Command IDs**

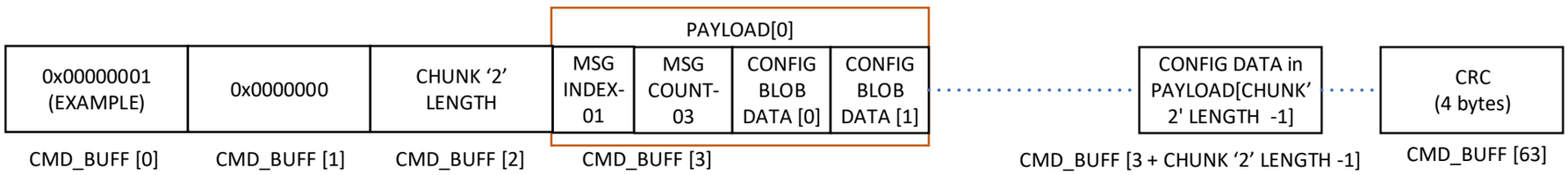
rfe_configure()	0x00000000
rfe_radarCycleStart()	0x00000001
rfe_radarCycleStop()	0x00000002
rfe_getError()	0x00000003
rfe_getTime()	0x00000004
rfe_getVersion()	0x00000005
rfe_monitorRead()	0x00000006
rfe_getNextRadarCycleStartTime()	0x00000007
rfe_setNextRadarCycleStartTime()	0x00000008
rfe_updatePush()	0x00000009
rfe_testContinuousWaveTransmissionStart()	0x0000000A
rfe_testContinuousWaveTransmissionStop()	0x0000000B
rfe_testSetParam()	0x0000000C

rfe_configure()

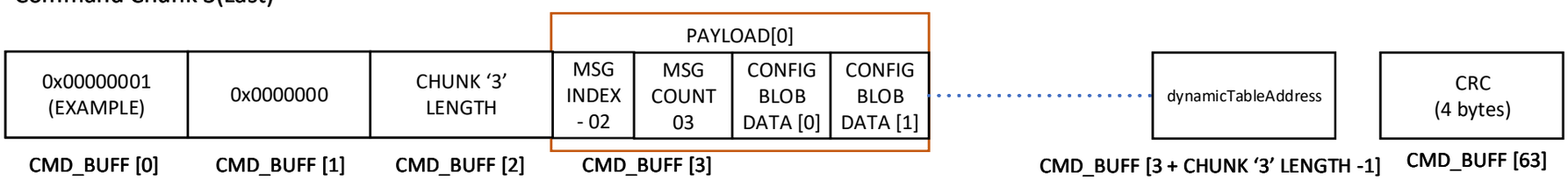
Command Chunk 1



Command Chunk 2



Command Chunk 3(Last)

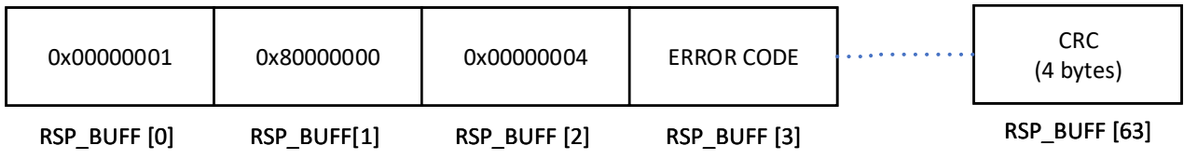


The configuration blob data consists of data more than 256 bytes, hence it cannot be sent in single attempt. The entire blob data is divided and sent in chunks. The RFE FW excepts command even if the chunks are of smaller size as long as it has valid format, size and parameters. The last chunk contains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

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 during the transfer of chunks any error is detected, the RFE SW Driver must begin sending the chunks right from beginning. The RFE FW discards the successfully received data.

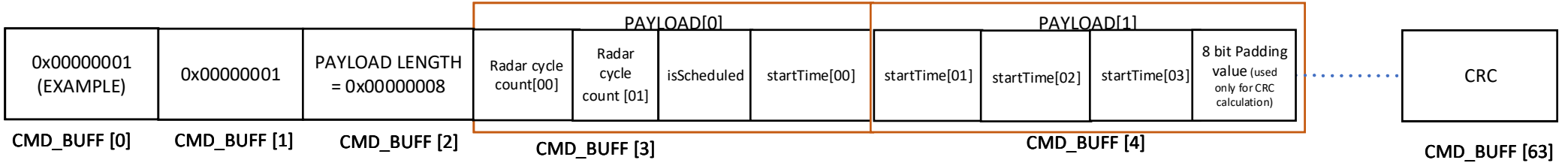
Typical Response



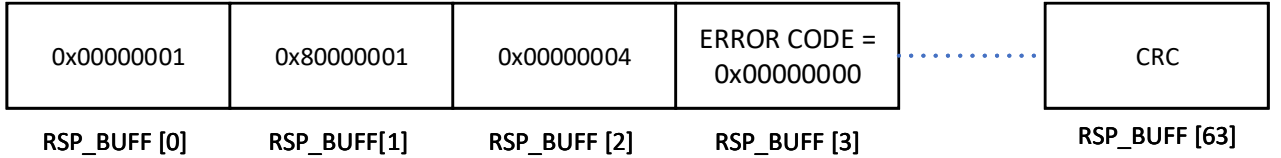
The Response for each of the chunks will be in same format as above.

rfe_radarCycleStart()

Command

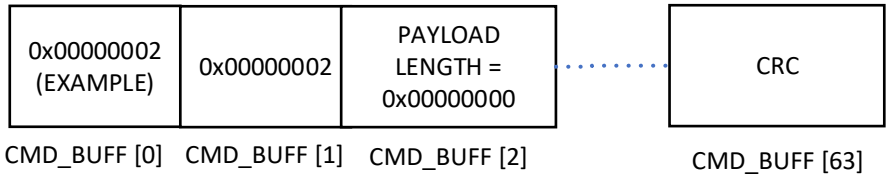


Response

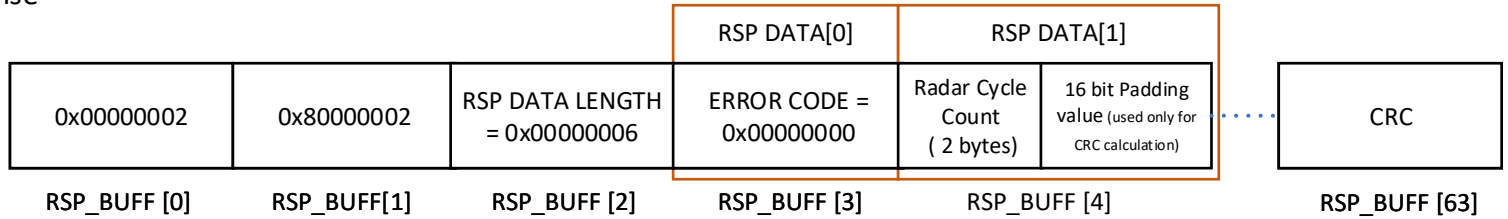


rfe_radarCycleStop()

Command

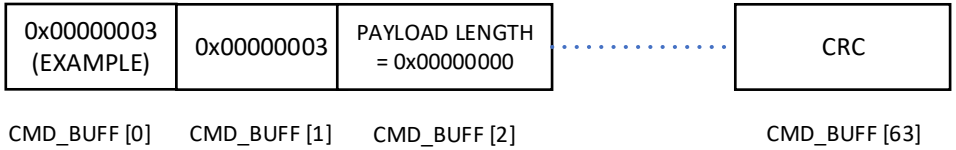


Response

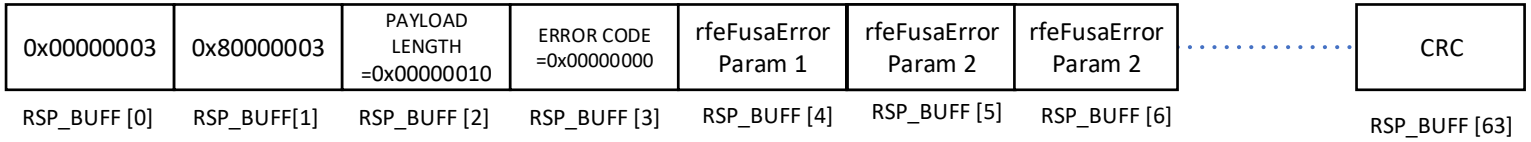


rfe_getError()

Command



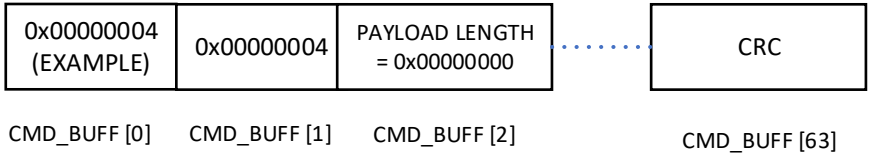
Response



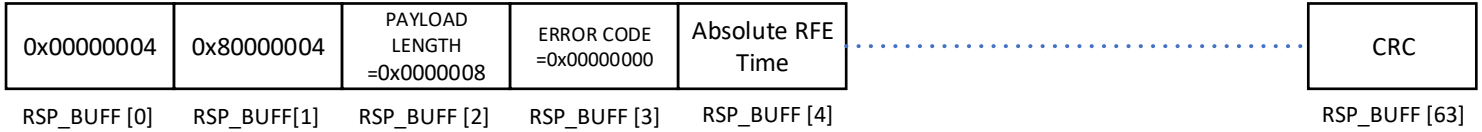
Remark: In RFE SW Release 0.4.0 Code Drop, the output parameters – rfeFusaError 1 , 2 and 3 are not present

rfe_getTime()

Command

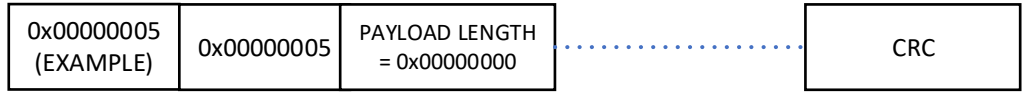


Response

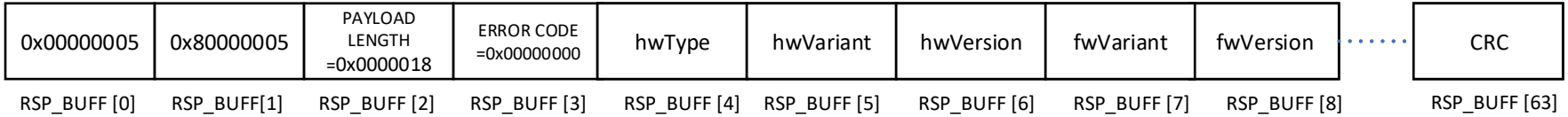


rfe_getVersion()

Command

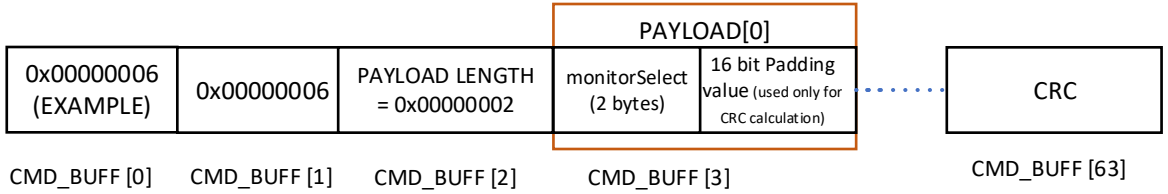


Response

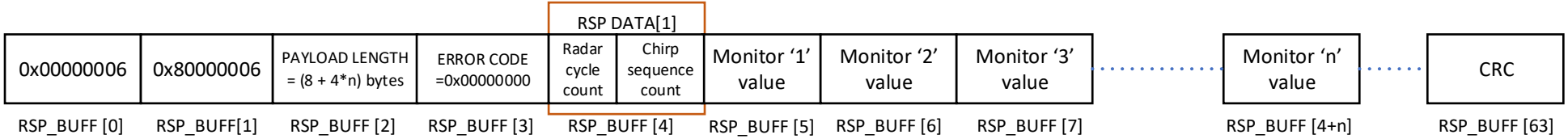


rfe_monitorRead()

Command

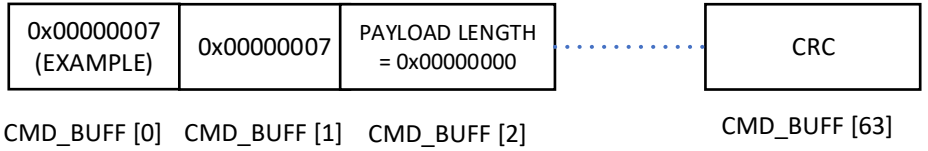


Response

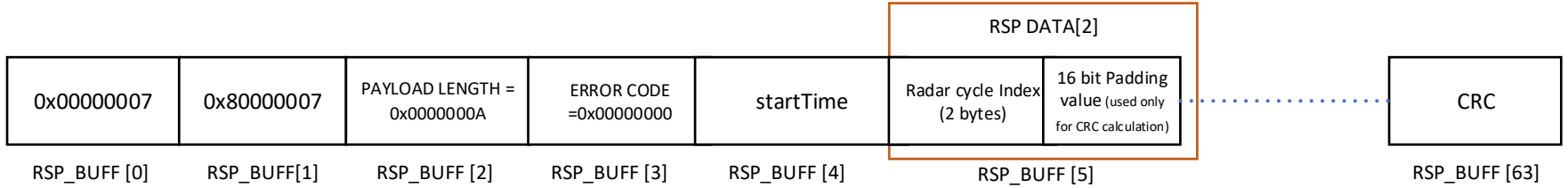


rfe_getNextRadarCycleStartTime()

Command

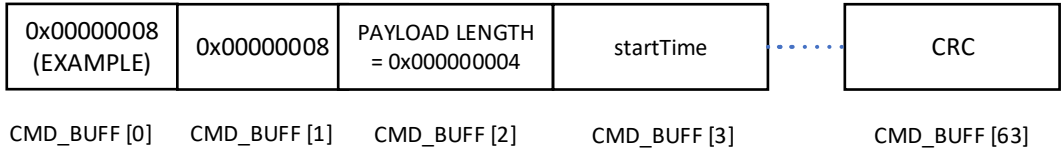


Response

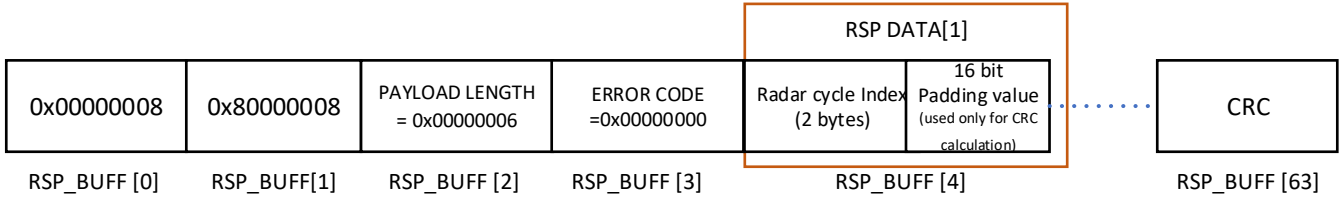


rfe_setNextRadarCycleStartTime()

Command

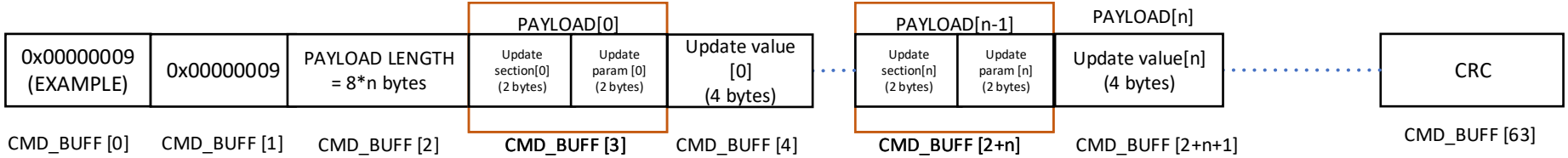


Response

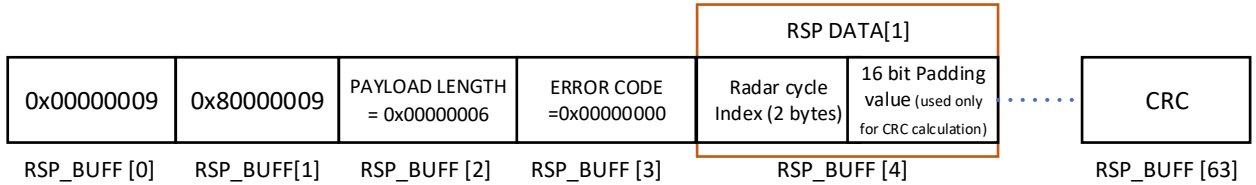


rfe_updatePush()

Command



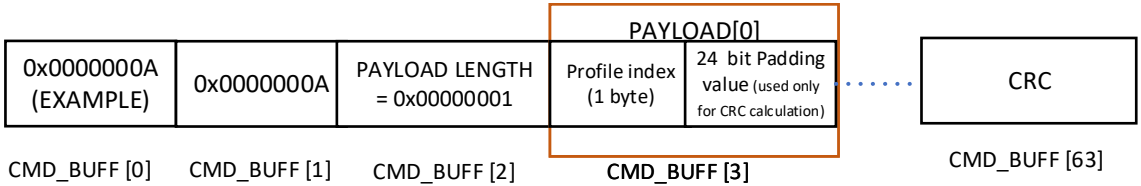
Response



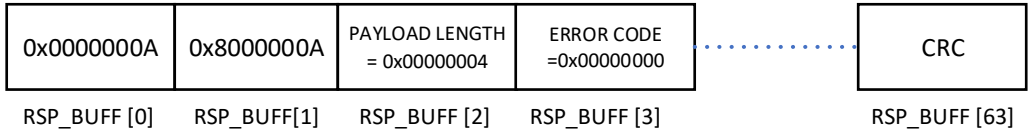
Remark: In one command it is possible to update maximum of 16 parameters. Until the updates are applied, next update push is not possible unless rfe_radarCycleStop() is called, which clears the pending updates on RFE FW side. In case the last radar cycle is being executed, the rfe_updatePush() will report error.

rfe_testContinuousWaveTransmissionStart()

Command

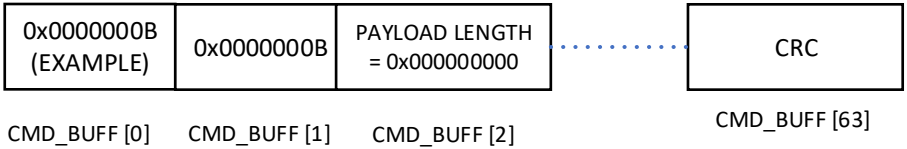


Response

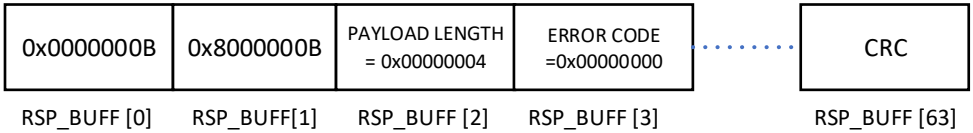


rfe_testContinuousWaveTransmissionStop()

Command

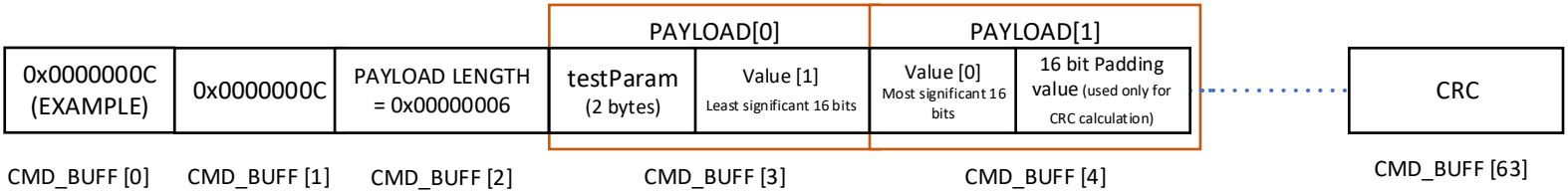


Response



rfe_testSetParam()

Command



Response