AN12233
MPC5744P FlexRay Interface in Pictures

Rev. 0 — May 2021 Application Note

by: NXP Semiconductors

Contents

1 Introduction

The FlexRay is an automotive network communication protocol developed by the FlexRay Consortium to govern on-board automotive computing. FlexRay interface uses Time Division Multiple Access (TDMA) in static segment and Flexible TDMA (FTDMA) in dynamic segment. The FlexRay Communications System is specified for a baud rate up to 10 Mbit/s (NXP devices supports baud rates of 2.5 Mbit/s, 5 Mbit/s, 8 Mbit/s and 10 Mbit/s).

1.1 FlexRay timing hierarchy

The communication on FlexRay is based on 64 communication cycles which are repeated.

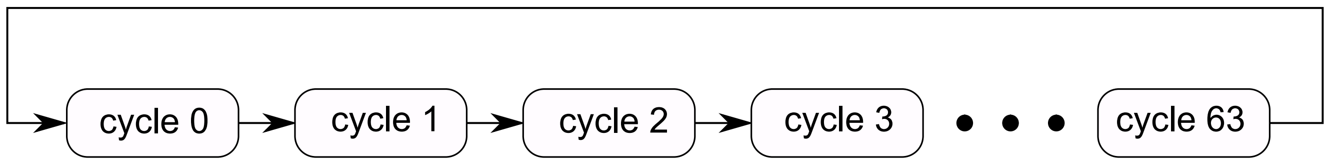


Figure 1. Base principle of communication on the FlexRay

The cycle is the highest level of the Flexray timing hierarchy which has four levels: communication cycle level, arbitration grid level, macrotick level and microtick level (see figure below). Each cycle is consisted from static segment, dynamic segment, symbol window and network idle timer (NIT). A cycle is composed of an integer number of macroticks.

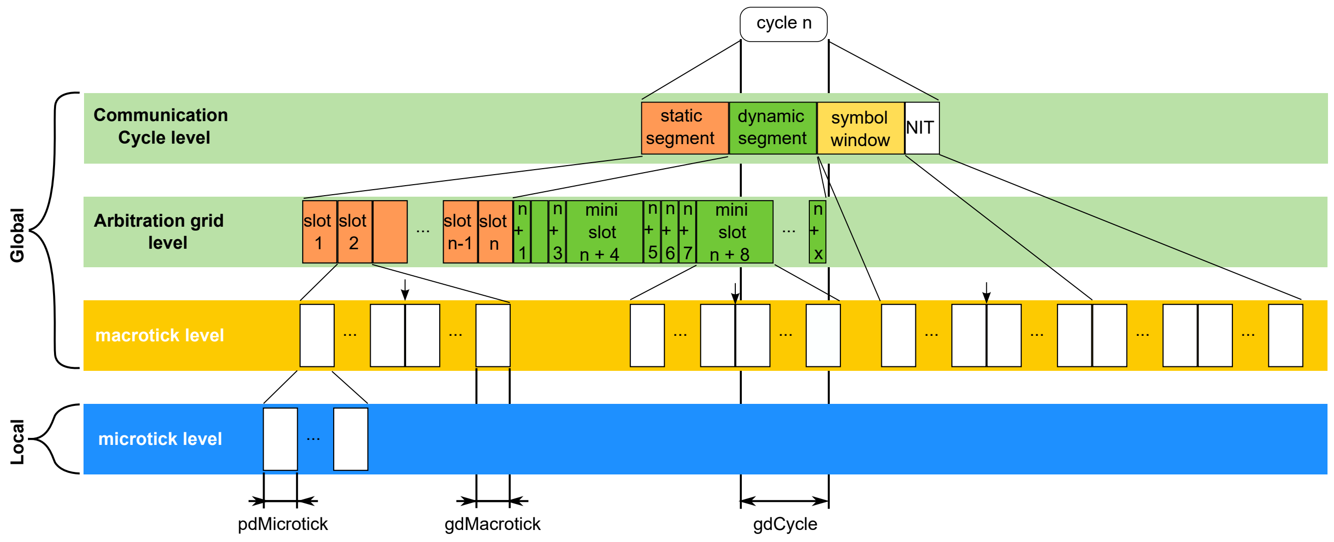


Figure 2. FlexRay timing

The Arbitration grid level dividing the static and dynamic segment on the data slots which are used for data transmitting. In static segments, they are called static slots and in dynamic segment they are called minislots. This level is based on the Macrotick “clock”. Macrotick “clock” is the time derived from the network-wide clock synchronization. It is the smallest granularity unit of the global time which means that this “clock” is the same on all nodes in the FlexRay network. Each Macrotick is composed from integer number of microticks.

All above levels of hierarchy are related to the global point of view (the same for all nodes). Microtic is the lowest clock level which is node specific. Microtick is the time derived from the Communication Controller Clock (CC). It is local time which means that it can be different on each node. It is the granularity of a node’s internal local time.

The detailed clocking relationship among the Macrotick, Microtick, Cycle time, sample time, bit clock and PE clock are described in the following figure. The base clock of the FlexRay interface implemented on NXP device is the PE clock (FlexRay clock) which can be either 40 MHz clock with stable duty cycle or 80 MHz with non-stable duty cycle. The module uses both edges when the clock has stable duty cycle (for example XOSC clock) for triggering the sample clock period. Compared to the nonstable duty cycle clock source (for example PLL) from which is used only rising edge for triggering the sample clock period as it is shown in the zoomed part of the figure below.

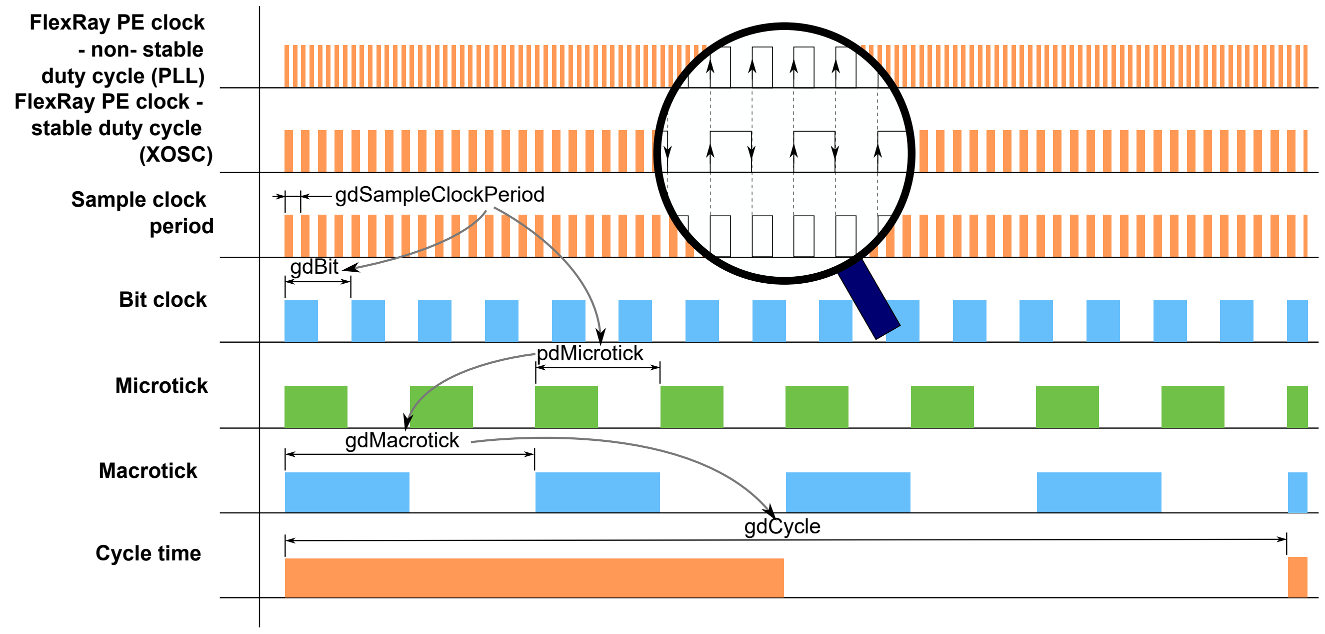


Figure 3. FlexRay interface timing diagram for one node

The bit clock (gdBit) is 8 (cSamplesPerBit) period of the sample clock period (gdSampleClockPeriod) which is set by the configured bitrate of FlexRay interface. The length of the gdBit is:

```
gdBit = cSamplesPerBit * gdSampleClockPeriod
```

The length of Microtick (pMicrotick) is also based on the Sample clock period as follow:

```
pdMicrotick = pSamplePerMicrotick * gdSampleClockPeriod
```

The length of the macrotick (gdMacrotick):

```
gdMacrotick =pMicroPerMacroNom * pdMicrotick
```

The length of the communication cycle (Variable gdCycle):

```
gdCycle = gMacroPerCycle * gdMacrotick,
```

where the gMacroPerCycle consist number of Macrotic in one cycle.

The following table summarizes the possible configuration of the above variables on the NXP devices. gdSamplesClockPeriod, pSamplesPerMicrotick, pdMicrotick and gdBit are set internally according to the configured bitrate.

Table 1. Main timing variables ranges

Parameter	Short-cut	Source of the parameter	Variable	Unit	Bitrate [Mb/s]			
					2.5	5	8	10
Sample clock	-	Automatically selected according to the bitrate	gdSampleClockPeriod	ns	50	25	12.5	12.5
Samples per Microtics	-		pSamplesPerMicrotick	-	1	1	2	2
Microtick	uT		pdMicrotick[1]	ns	50	25	25	25
Bit	-		gdBit	ns	400	200	100	100
Number of uT	-	User configuration	pMicroPerMacroNom	uT	40-240			
Macrotick	MT		gdMacrotick	us	2-6	1-6		
Number of MT	-		gMacroPerCycle	MT	10-16000			
Cycle	-	Calculated from other variables	gdCycle	us	10-16000			

1. NXP implement this parameter as global (It is not configurable but fixed) selected by the Bitrate.

NOTE

The data position is given by the cycle configuration (in which it is sent), the segment type and the slot number inside the segment.

1.2 FlexRay module

The FlexRay module has three main parts:

- Controller host interface (CHI),
- Protocol engine (PE),
- Clock domain crossing unit (CDC).

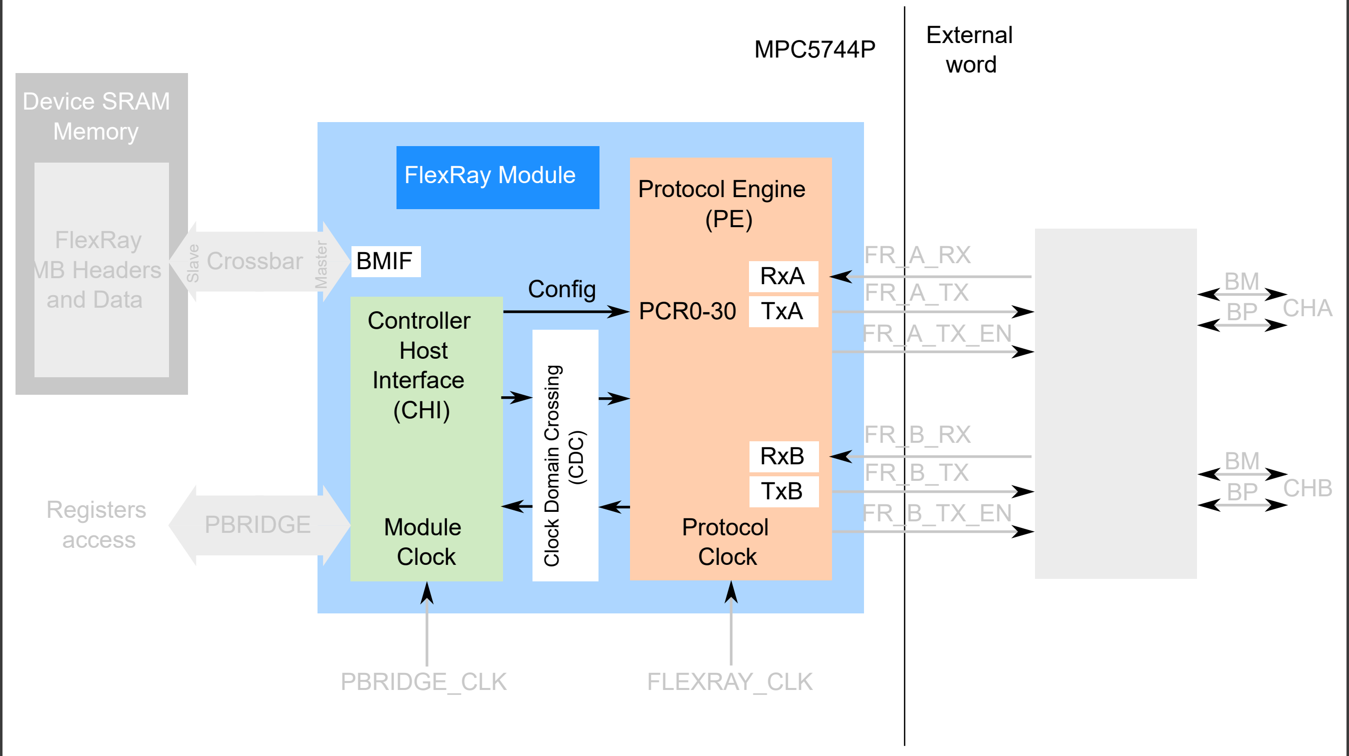


Figure 4. FlexRay block diagram

The PE ensures the timing on the physical interface. It has one transmitter unit and one receiver unit for each channel (TxA, RxA, TxB, RxB).

The CHI ensures module and message buffers (MB) configuration and also controls the communication with the CPU.

The CDC is necessary for communication between the PE and CHI parts because they are running with different clock.

The communication with the SRAM memory, where the MBs are stored, is done by the Bus master interface (BMIF) which is connected as master to the device crossbar.

1.3 Configuration

The FlexRay interface has global and local parameters configuration. All nodes have the same global parameters configuration (variable begin with “g”) as bitrade, number of cycles, number of static slots, payload of static slots etc. Each node has local parameters configuration (the variable begin with “p”) which is node specific. Some of them are related to the hardware of the node. The local parameters are number of Microtics per cycle, duration of the Microtick and maximum length of the dynamic segment data etc.

There are also couple of protocol constants (the constant begin with “c”) which have global scope as the channel A/B CRC polynomials, maximum cycle length and maximum number of cycles etc.

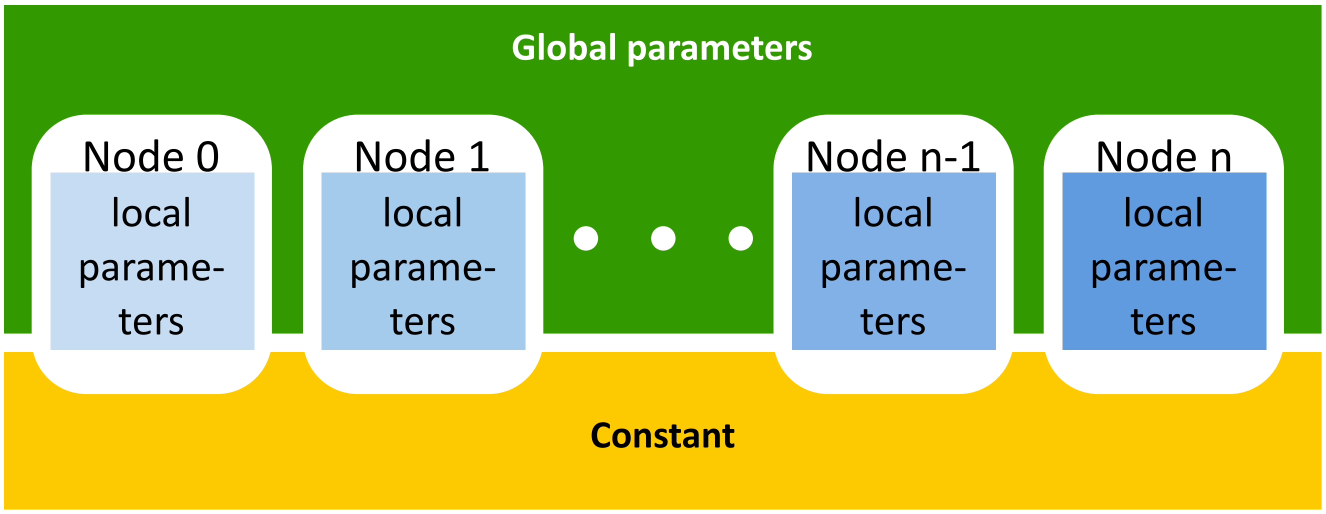


Figure 5. FlexRay levels of configuration

2 FlexRay segments of the cycle

Each cycle is composed from up to four parts Static Segment, Dynamic segment, symbol window and Network idle Time as shown in the following figure.

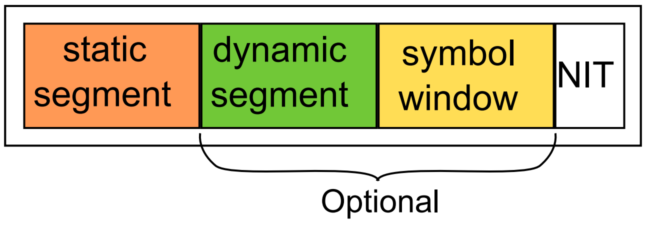


Figure 6. FlexRay cycle

2.1 FlexRay data segments

2.1.1 Static frame

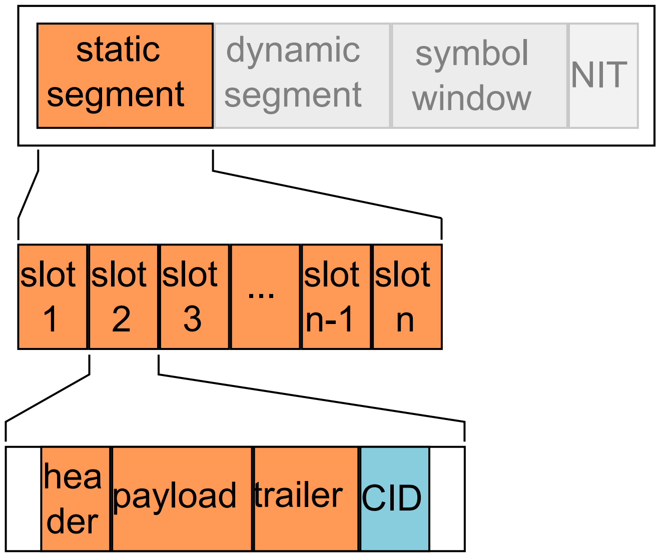


Figure 7. Static segment content and the position in the cycle

The Static segment used TDMA for bus access. The static frame/slot is always sent in the specified position with fixed payload length. When the payload length is smaller than reserved space, the rest of reserved space is lost because the payload length is fixed. But all static frames are always sent because they have reserved place.

When the alternative of ferry in static segment is used, every car has reserved space and the space for each car is the same (done by the biggest car which needs to be transported). Each car which has reserved space has guarantee that the car will be transported when it arrives. When the car do not arrive, the reserved space stays empty.

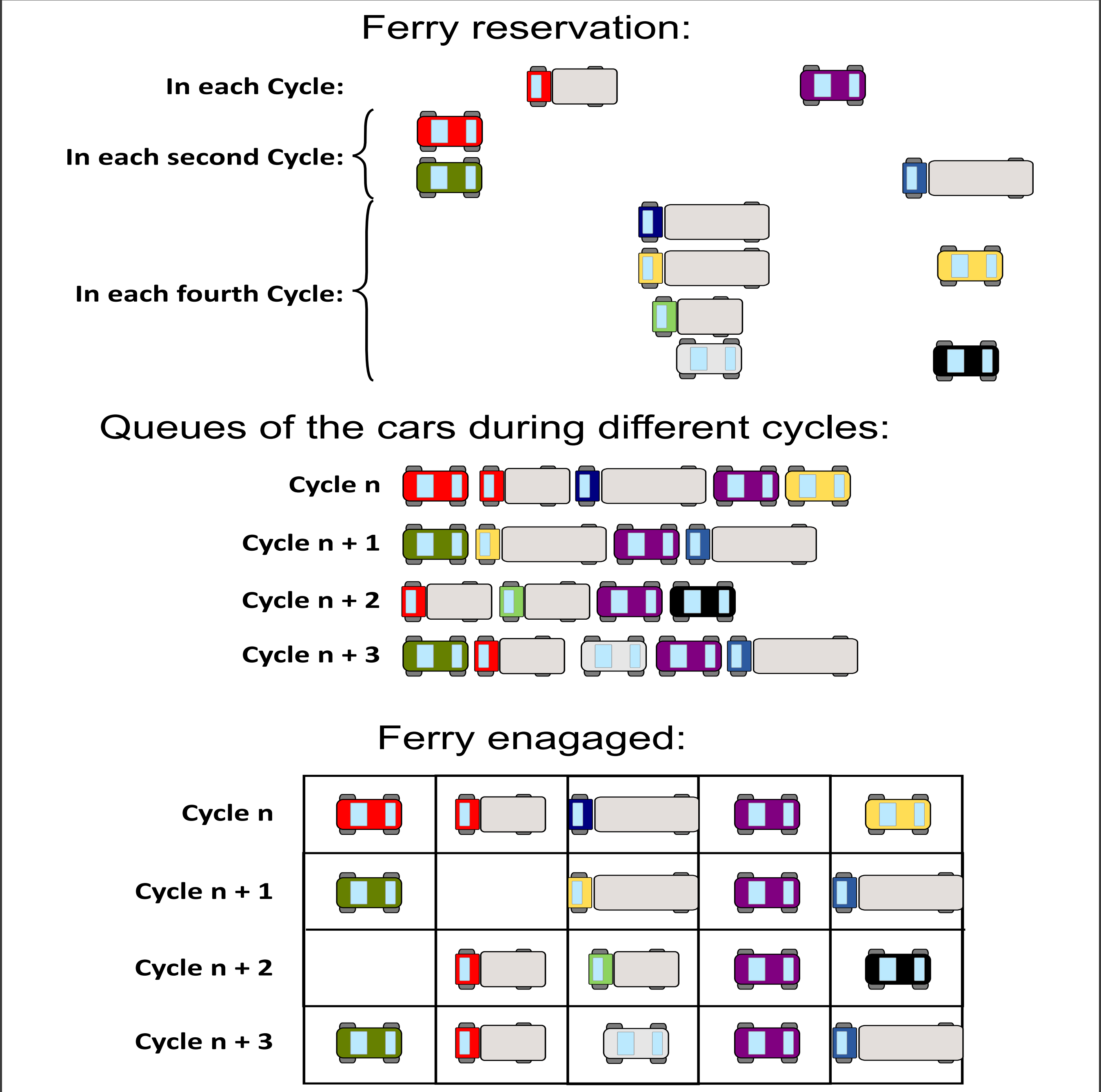


Figure 8. Ferry alternative of the static segment

2.1.1.1 Timing of the static segment

The timing of the static segment is based on the arbitration grid level of FlexRay hierarchy timing which means that it is based on the slots. When the static frame is not transmitting (data is not prepared, frame is not transmitted for the current cycle number etc.) the channel is in idle state for this slot. See below figure for better imagination.

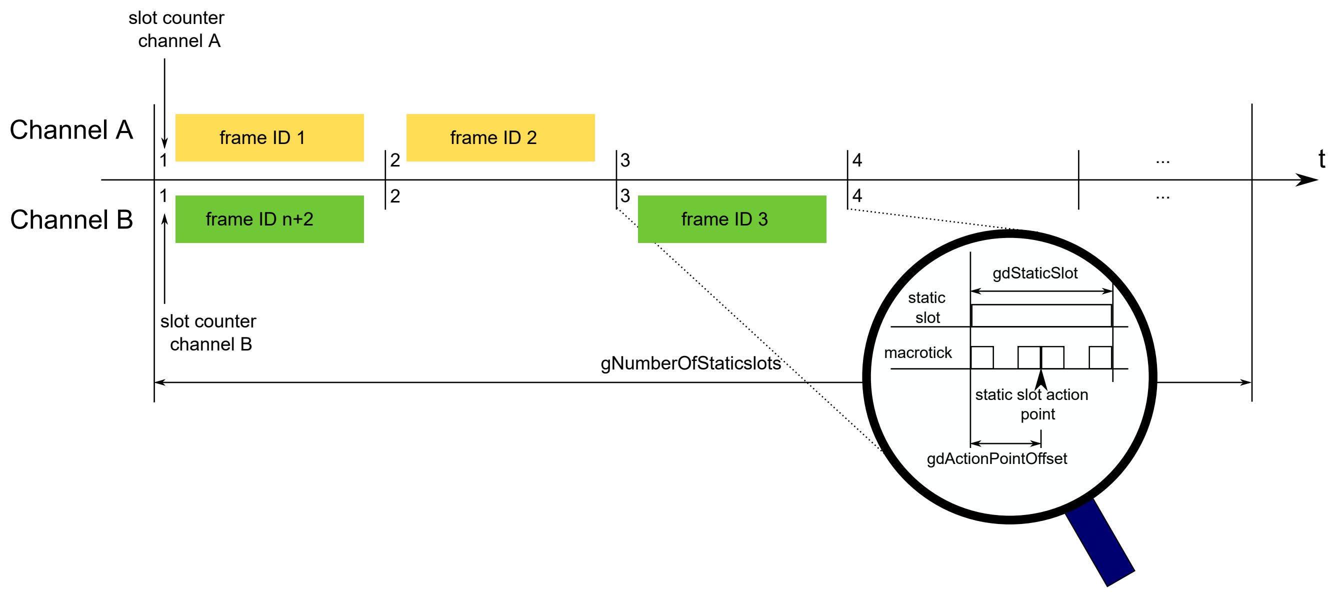


Figure 9. Detailed timing of the static segment

The channels slot counters for both channels are 1 on the beginning of the static segment and finished with the same number, because in the static segment there is always transmitted the same number of slots in both channels done by the gNumberOfStaticSlots.

The slot length is done by the gdStaticSlot number of Macrotics which is the same for all slots. All frames in the static segment has the length of 1 slot. Each frame transition start at the static slot action point which is configured by the gdActionPointOffset in number of macroticks and finish in the same slot.

2.1.2 Dynamic frame

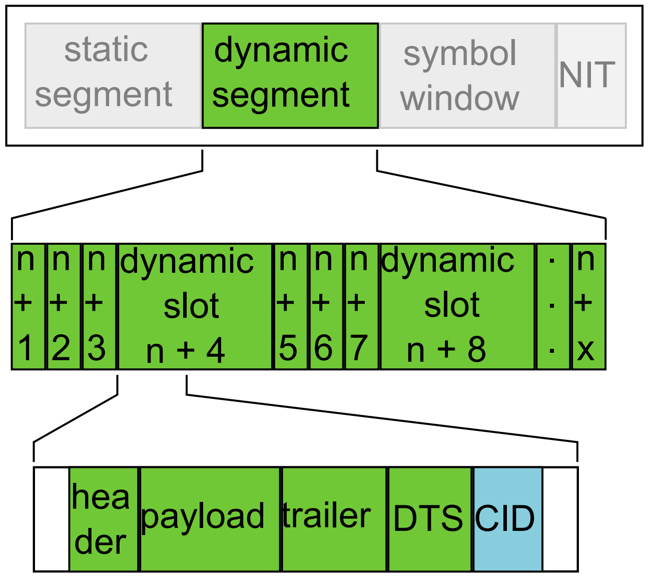


Figure 10. Dynamic segment content and the position in the cycle

The dynamic segment uses FTDMA for bus access and it is optional. When there is no data left which needs to be send in dynamic segment (gNumberOfMinislots = 0) it will not be there. The dynamic frame/slot position in the segment can change as well as the payload length which is limited only by the configured maximum length (pPayloadLengthDynMax). There is no guaranty for the transmission of dynamic frame. Only the list of the dynamic segments which should be transmitted is known but not their lengths. This is the reason why there is not guarantee of transmission for all frames/slots. The advantages is that the space is effectively used as there is no space between the slots (not any reserved spaces for frames which are not ready) and the length can be varied. The length of the dynamic segment is fixed (done by the gNumberOfMinislots). It means that there the transmission can vary from few long slots to a lot of small slots or any combination of the two.

When the alternative with ferry is used there is no reservation in dynamic segment. Who arrives first will be transported, because there is limited space (When there are big trucks, only few of them will be transported, but when there are small cars, it is possible to transport a lot of them).

When the car is arrives and there is enough space for it on the ferry it will be transported. The advantage is that there are not empty slots. Disadvantage is that there is not guaranteed that all the arrived cars will be transported because we do not know the length of the cars in advance. The cars before the red line will always be transported.

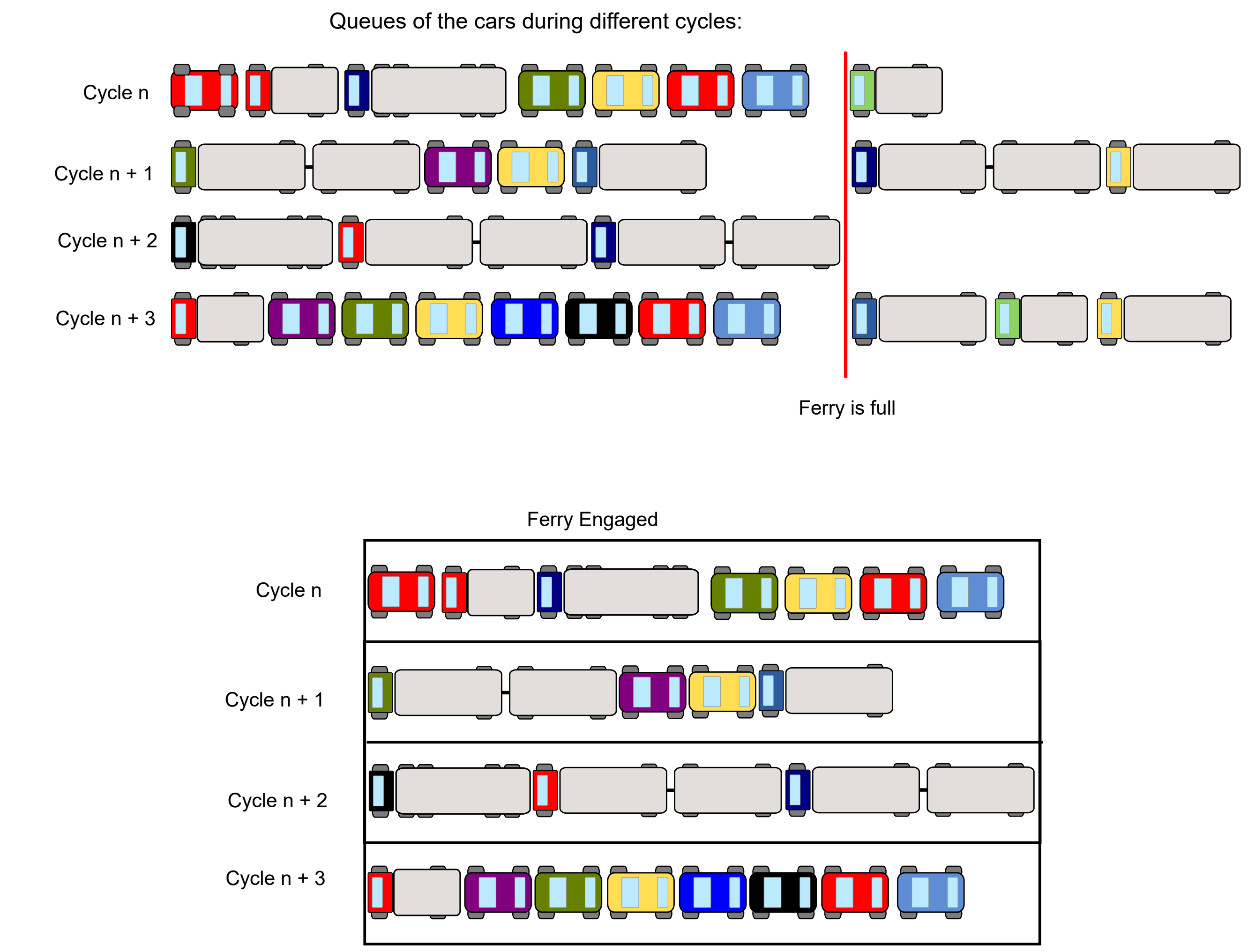


Figure 11. Ferry alternative of the dynamic segment

2.1.2.1 Timing of dynamic segment

The timing of the dynamic segment is based on the arbitration grid level of FlexRay hierarchy timing. It means that it is based on the minislots. When the dynamic frame is not transmitting (data is not prepared, frame is not transmitted for the current cycle number etc.) the channel is in idle state for one minislot in place of this dynamic frame. See the following figure for better understanding.

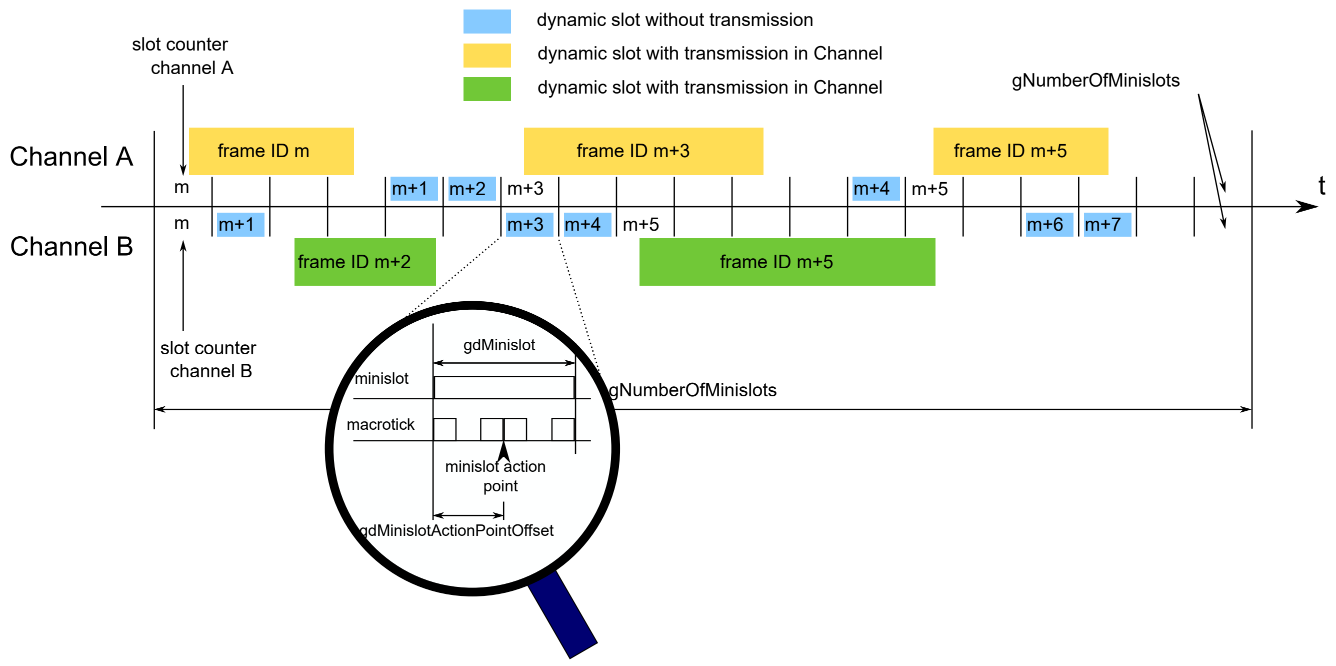


Figure 12. Detailed timing of the dynamic segment

The channels slot counters for both channels are equal only in the beginning of the dynamic segment (m = n) because in the dynamic segment there can be different number of frame transmitted in each channel, so the frame counters can be also different.

The minislot length is done by the *gdMinislot* number of Macrotics which is same for all minislots. Each frame transmission start at the minislot action point which is configured by the *gdMinislotActionPointOffset* in number of macroticks and finish at a minislot action point (the reason of the empty slot after the frames).

2.2 FlexRay protocol parts

2.2.1 Symbol window

The Symbol Window is optional and when gdSymbolWindow equals to zero and is not in the communication cycle. It is for sending the Media Access Test Symbol (MTS) which is used to test physical connection. The access arbitration is required for the symbol window among nodes. Software needs to do the arbitration, it is not provided by the FlexRay protocol.

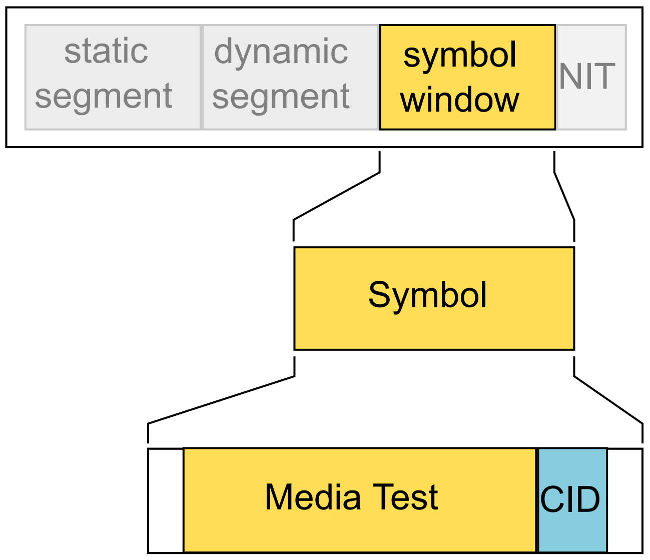


Figure 13. Symbol window content and the position in the cycle

2.2.2 Network Idle Time (NIT)

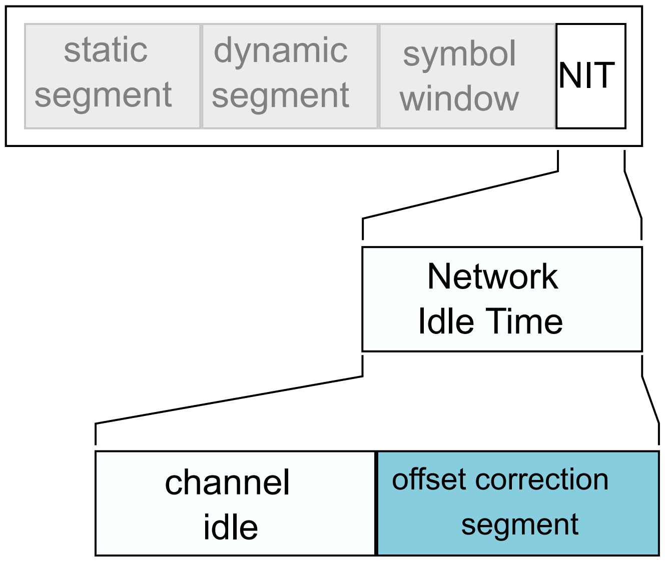


Figure 14. Network Idle Time content and the position in the cycle

The network idle time serves as a phase during which the node calculates and applies clock correction terms (More information can be found in offset Synchronization) and adds the remaining number of Macrotics within the communication cycle which were not allocated for the static segment, dynamic segment or symbol window.

3 Frame structure

3.1 Header segment

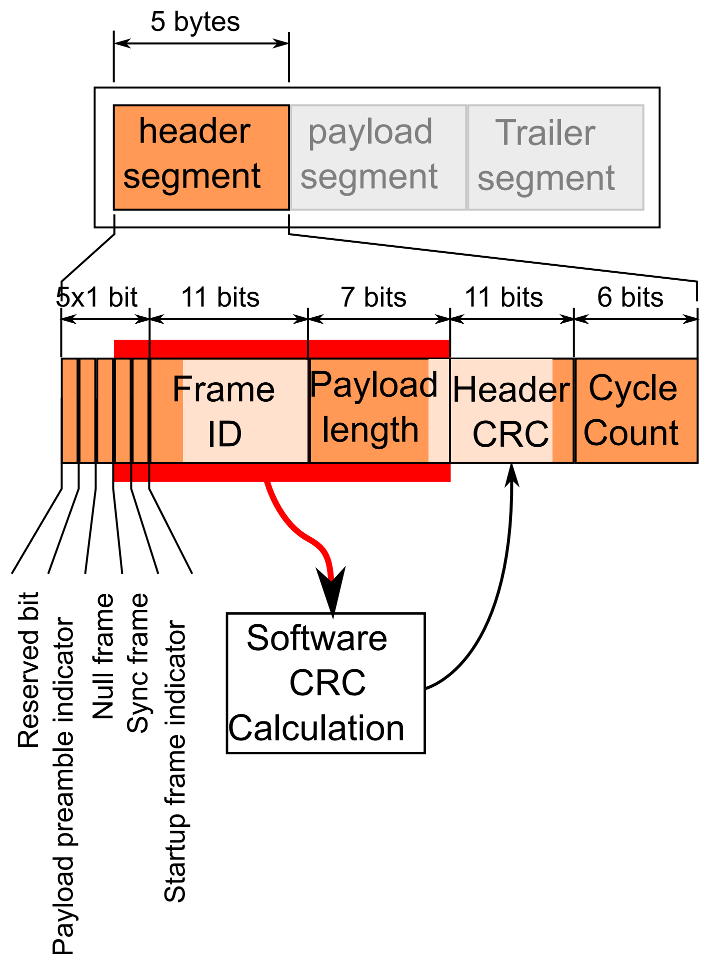


Figure 15. Frame common header segment content

3.1.1 Reserved

The reserved bit is reserved for future protocol use. It shall not be used by the application.

3.1.2 Payload preamble indicator

This bit indicates if the first bytes of the payload are used for another purpose other than for transmitting data. The number of bytes and concrete purpose is segment dependent.

0 The payload segment beginning of the frame contain a normal data.

1 The payload segment beginning of the frame contains a Network Management Vector (NMVector) in static slot/frame and Message ID (MID) in the dynamic slot/frame. More information is in Payload segment.

3.1.3 Null frame indicator

This bit informs if the frame contains valid data (The null frame indicator indicates only whether payload data was available to the communication controller at the time the frame was sent).

0 The payload segment contains no valid data. All bytes in the payload section are set to zero.

1 The payload segment contains data.

3.1.4 Sync frame indicator

This bit informs if the received frame is synchronization frame. The synchronization frame can be sent only in the static segment.

0 The frame is not for synchronization or synchronization related tasks.

1 The frame is for synchronization or synchronization related tasks, if it meets other acceptance criteria (see below).

3.1.5 Startup frame indicator

This bit informs if the frame is the startup frame. Startup frames serve a special role in the startup mechanism (see Communication startup). Only coldstart nodes are allowed to transmit startup frames.

0 The frame is not a startup frame.

1 The frame is a startup frame.

The startup frame indicator shall only be set to one in the sync frames of coldstart nodes. Therefore, a frame with the startup frame indicator set to one shall also have the sync frame indicator set to one. Since the startup frame indicator can only be set to one in sync frames, every coldstart node can transmit exactly one frame per communication cycle and channel with the startup frame indicator set to one.

3.1.6 Frame ID (11 bits)

The Frame ID defines the slot in which the frame should be transmitted. The Frame ID is unique in communication cycle with the range 1-2047 (Frame ID 0 is an invalid).

3.1.7 Payload length (7 bits)

The payload length defines how many 2 bytes data is in the Frame (When the payload length is 22 there is 44 bytes of data). The range of this field is from 0 to *cPayloadLengthMax*. The meaning of this value is different in static and dynamic slot/frame. See Payload segment for more information.

3.1.8 Header CRC (11 bits)

The Header CRC uses the following CRC polynomial:

$X^{11}+X^{9}+X^{8}+X^{7}+X^{2}+1$

The CRC is calculated from the following part of the frame header:

- Sync frame indicator
- Startup frame indicator
- Frame ID
- Payload length

3.1.9 Cycle count (6 bits)

It is the value of the internal counter vCycleCounter at the time of frame transmitting.

3.2 Payload segment

The maximum payload segment length is done by the constant cPayloadLengthMax (254B). The payload length is configured by global variables gPayloadLengthStatic in the static segment. The maximum payload length is configured by the variable pPayloadLengthDynMax in the dynamic segment. It means that the data length in the dynamic slot can vary from 0 to pPayloadLengthDynMax.

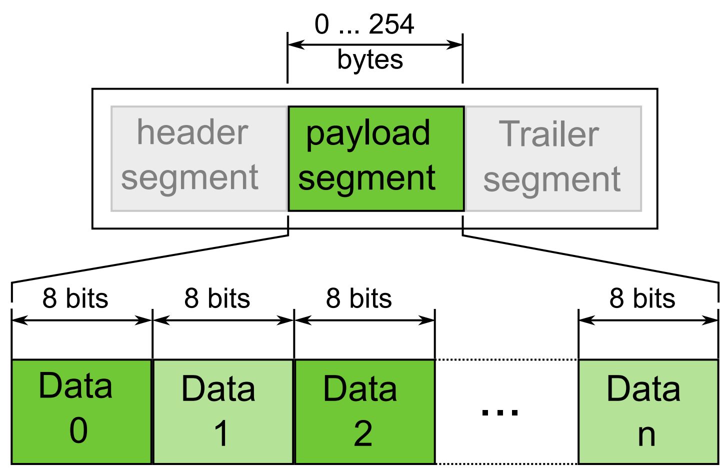


Figure 16. Frame common payload segment content

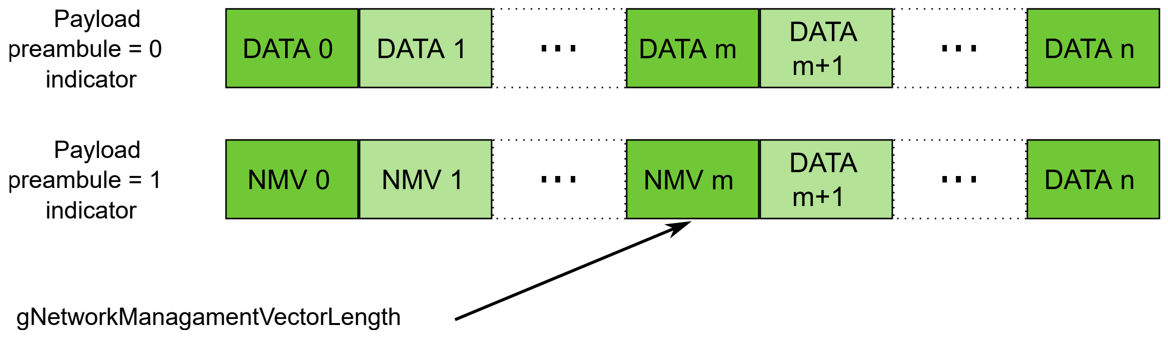
When the Payload preamble Indicator bit is set in the frame header the first data in the payload segment has different functionality based on the frame segment.

3.2.1 Static segment – NMVector

The length of the NMVector is configured by the *gNetworkManagementVectorLength*. The content of this field is written by the application and not by the communication controller.

gNetworkManagementVectorLength range is from 0 to 12 bytes.

Figure 17. Network Management Vector


Payload preamble = 0 indicator
DATA 0
DATA 1
...
DATA m
DATA m+1
...
DATA n
Payload preamble = 1 indicator
NMV 0
NMV 1
...
NMV m
DATA m+1
...
DATA n
gNetworkManagamentVectorLength

All NMVectors received in one cycle from all static frames and both channels are ORed into NMVRn registers. The NMVector values are stored into the CHI registers NMVRn. The *pNMVectorEarlyUpdate* variable defines when the NMVRn registers are updated. When *pNMVectorEarlyUpdate* = 0 registers are updated at the end of the communication cycle. When *pNMVectorEarlyUpdate* = 1 registers are updated at the end of static segment.

3.2.2 Dynamic segment – MID

The payload preamble indicator indicates the presence of a Message ID (MID) at the beginning of the payload. The MID is always 16 bits (See below figure). The content of this field is written by the application and not by the communication controller. On the receiving side, the MID can be used for filtering the dynamic frames received. For more information check the FIFO filtering.

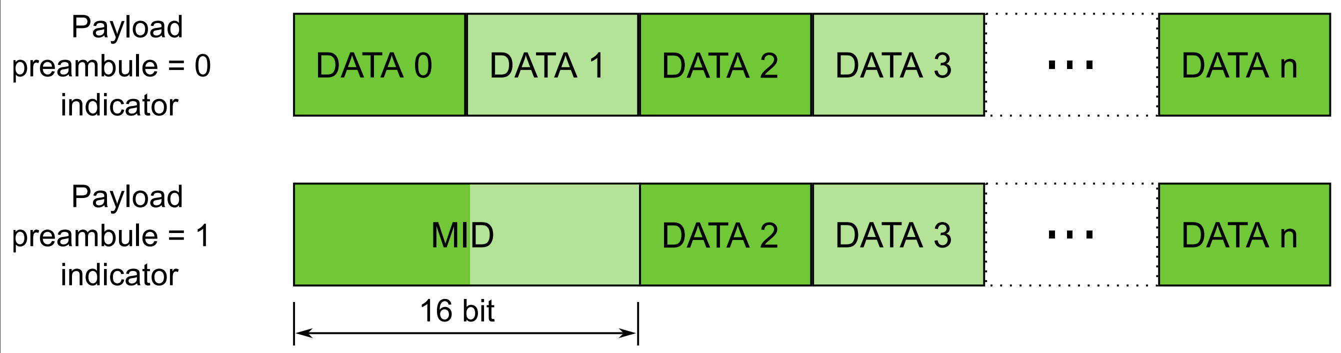


Figure 18. Message ID

3.3 Trailer segment

This field consists of 24 bit CRC which is computed over whole header and payload segment of the frame.

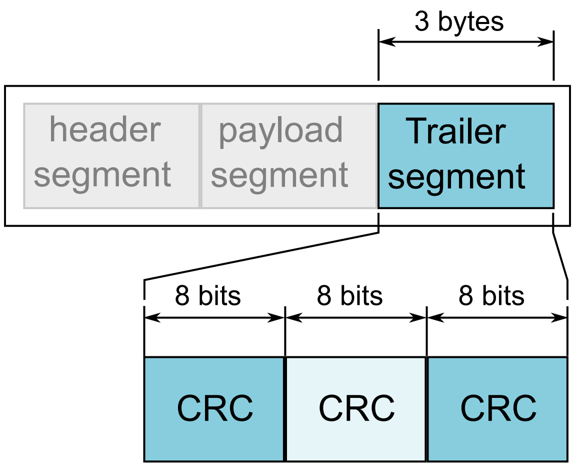


Figure 19. Frame common Trailer segment content

The CRC uses following CRC polynomial:

$X^{24}+X^{22}+X^{20}+X^{19}+X^{18}+X^{16}+X^{14}+X^{13}+X^{11}+X^{10}+X^{8}+X^{7}+X^{6}+X^{3}+X+1$

There are different initialization word for the channel A (0xFEDCBA) and channel B (0xABCDEF).

4 Memory structure

4.1 Organization of the data

FlexRay module is used for receiving and transmitting frames message buffers (MBs). The MB has three parts:

- MB Control data – stored in the FlexRay LRAM memory space
- MB Header – stored in the device system RAM memory space
- MB Data – stored in the device system RAM memory space

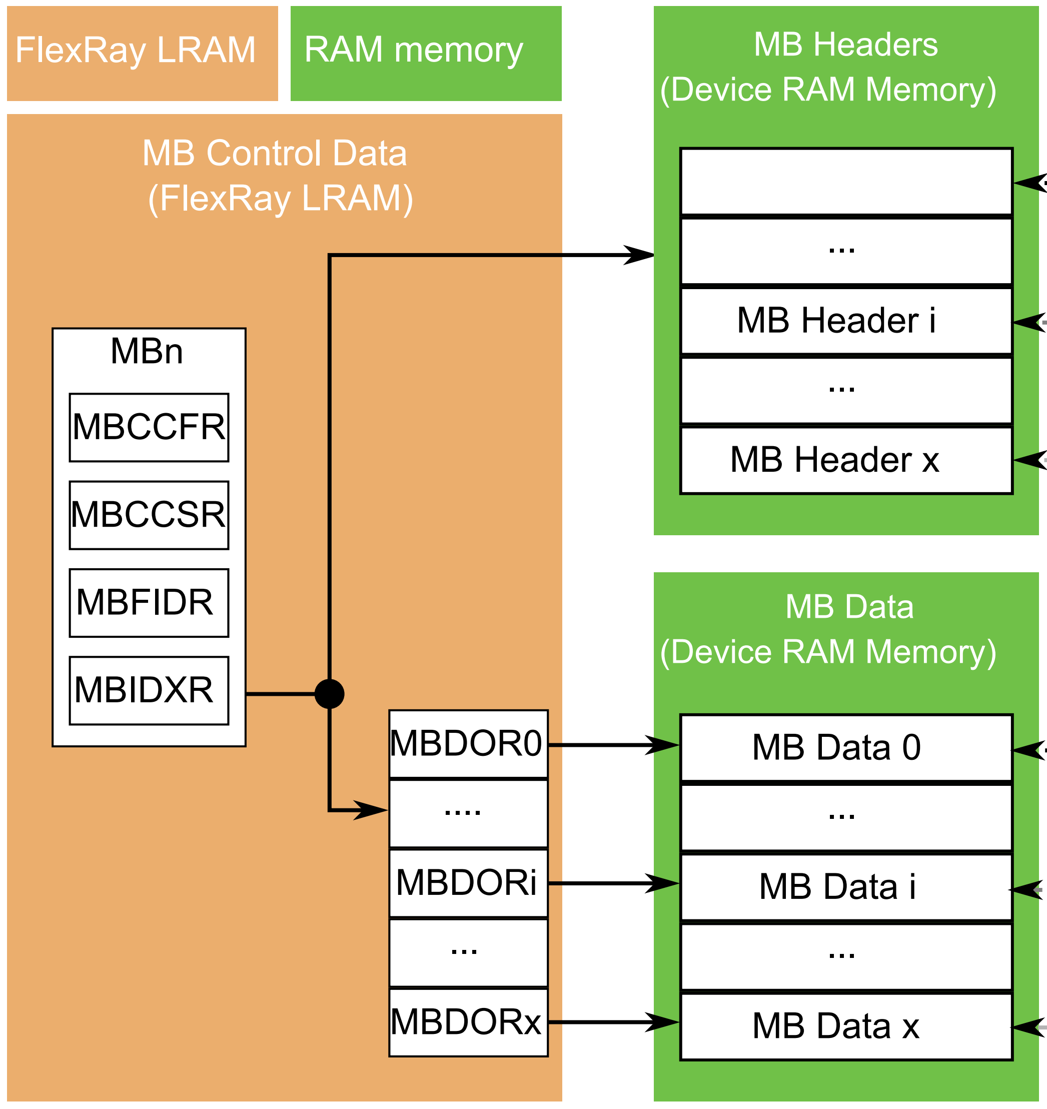


Figure 20. MB structure

The x means the number of MB which consists of Receive shadow buffers + Number of configurable MB implemented on the device. There is one Receive shadow buffer for each segment and channel. It means that there are 4 of them (two channels [A,B] and 2 segments [static, dynamic]).

How to get the address of the MB Header:

SADR_MBHF[n] = SMBA + MBIDXR(n)[MBIDX]*8

How to get the address of the MB data:

SADR_MBDF[n] = SMBA + MBDOR(i) = SMBA + MBDOR(MBIDXR(n)[MBIDX])

4.2 MB control data

They are stored in the dedicated register of the FlexRay. The control data of each MB are stored in four FlexRay registers:

- MBCCFR (Message Buffer Cycle Counter Filter Register)
- MBCCSR (Message Buffer Configuration, Control, Status Register)
- MBFIDR (Message Buffer Frame ID Register)

- MBIDX (Message Buffer Index Register)

4.3 MB header

It is stored in the RAM memory space and the length is 8 Bytes. The header has two parts:

- 6 Bytes of the Frame Header
- 2 bytes Slot Status

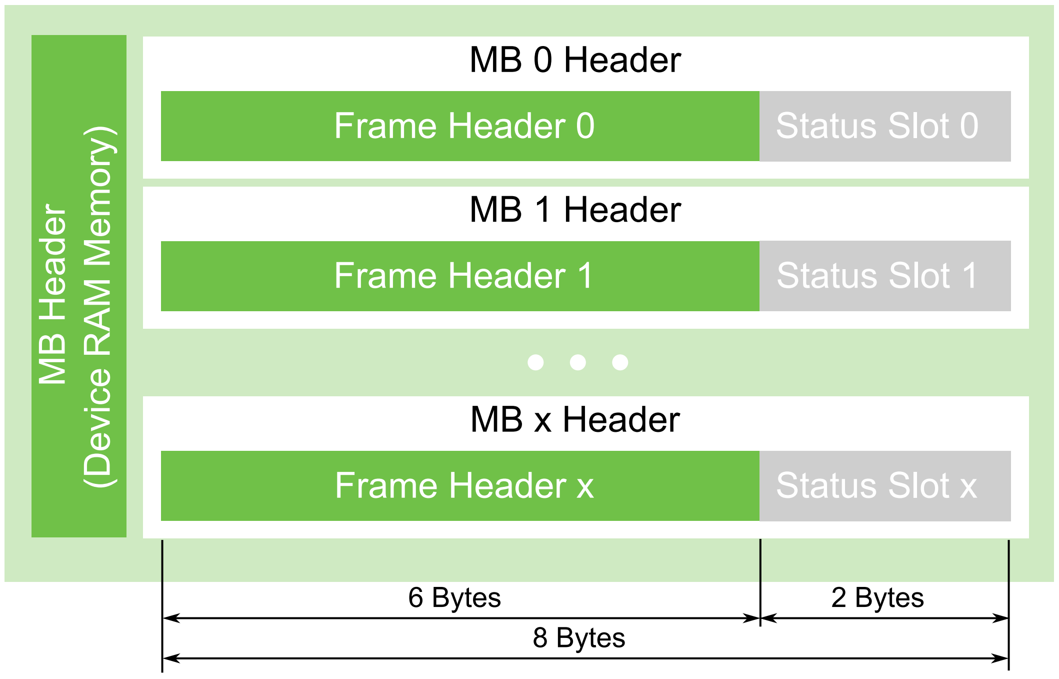


Figure 21. Higher level of the MB header structure

4.3.1 Frame header

Correspond with the data transmitted in the protocol frame header. The relationship between the frame header and MB header is shown in the following figure:

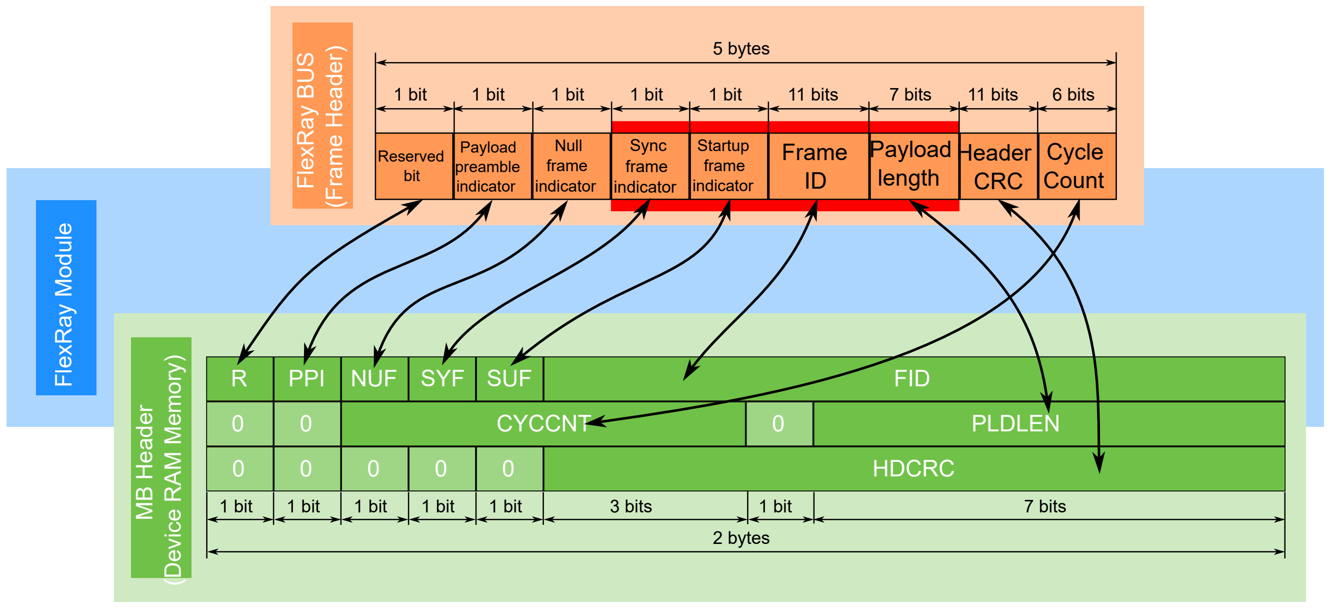


Figure 22. Relationship between frame header on BUS and MB header stored in the RAM

4.3.2 Slot status

The application can only read the status slot and the CC can only write to it. The meaning of the Slot status data depends on the MB type.

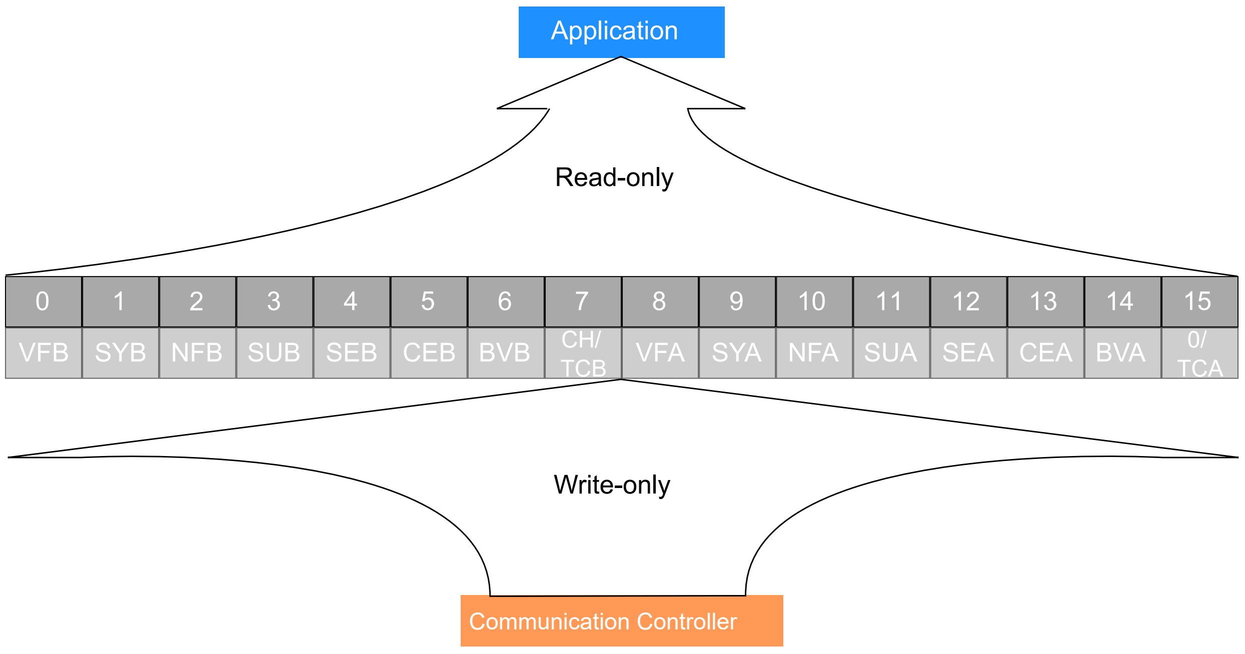


Figure 23. Access to the slot status field

4.3.2.1 Receive MB + FIFO

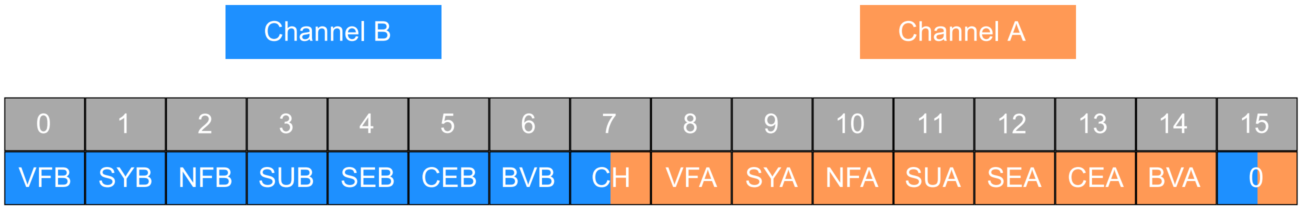


Figure 24. Receive MB Slot status structure

Table 2. Receive MB Slot Status field description

Bit	Field	Description
0,8	VFn	Valid Frame on Channel n
1,9	SYn	Sync Frame Indicator Channel n
2,10	NFn	Null Frame Indicator Channel n
3,11	SUn	Startup Frame Indicator Channel n
4,12	SEn	Syntax Error on Channel n
5,13	CEn	Content Error on Channel n
6,14	BVn	Boundary Violation on Channel n
7	CH	Channel first valid received
15	-	Used only for transmit MB

4.3.2.2 Transmit MB

The following figure shows the transmit MB slot status structure.

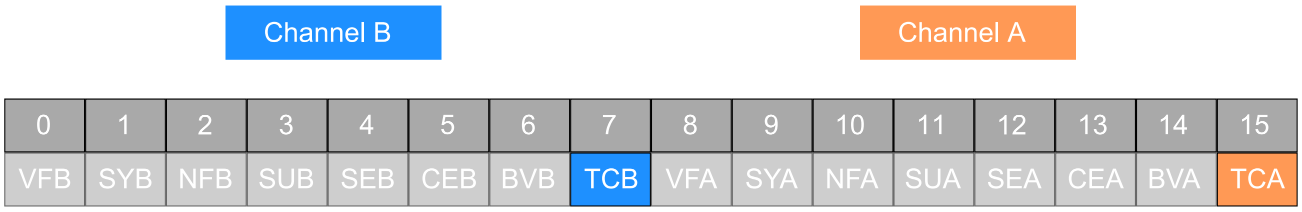


Figure 25. Transmit MB slot status structure

Table 3. Transmit MB slot status field description

Bit	Field	Description
0-6, 8-14	-	Used only for receive MB
7, 15	TCn	Transmission Conflict on Channel n

4.4 MB data – stored in the RAM memory space

The data length which can be receive/transmit in one frame is set according to the segment where the MB is stored. There are 2 segments configured by the MBDSR and MBSSUTR registers. All MB in one segment have reserved the same space for the data. There is not possibility to configure different data length within the segment. However each segment can have different data length. Standard configuration is the Segment 1 (Static segment) and is used for static slots and the Segment 2 (Dynamic Segment) is used for the dynamic slots. THe following figure shows the above described MBs Data structure. Where x is number of MBs, n is Segment 1 number of bytes and m is Segment 2 number of bytes.

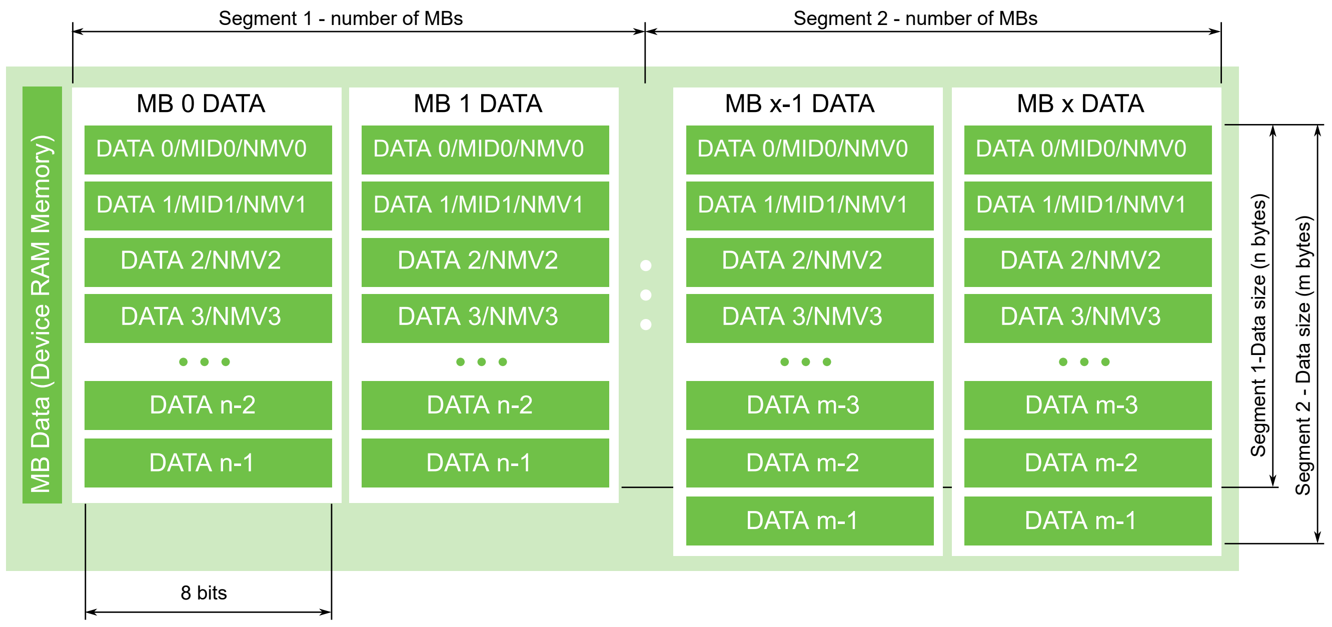


Figure 26. MB data structure

4.5 Configuration of the MB

4.5.1 Receive shadow MB

There are four receive shadow MB. One for each channel and sector.

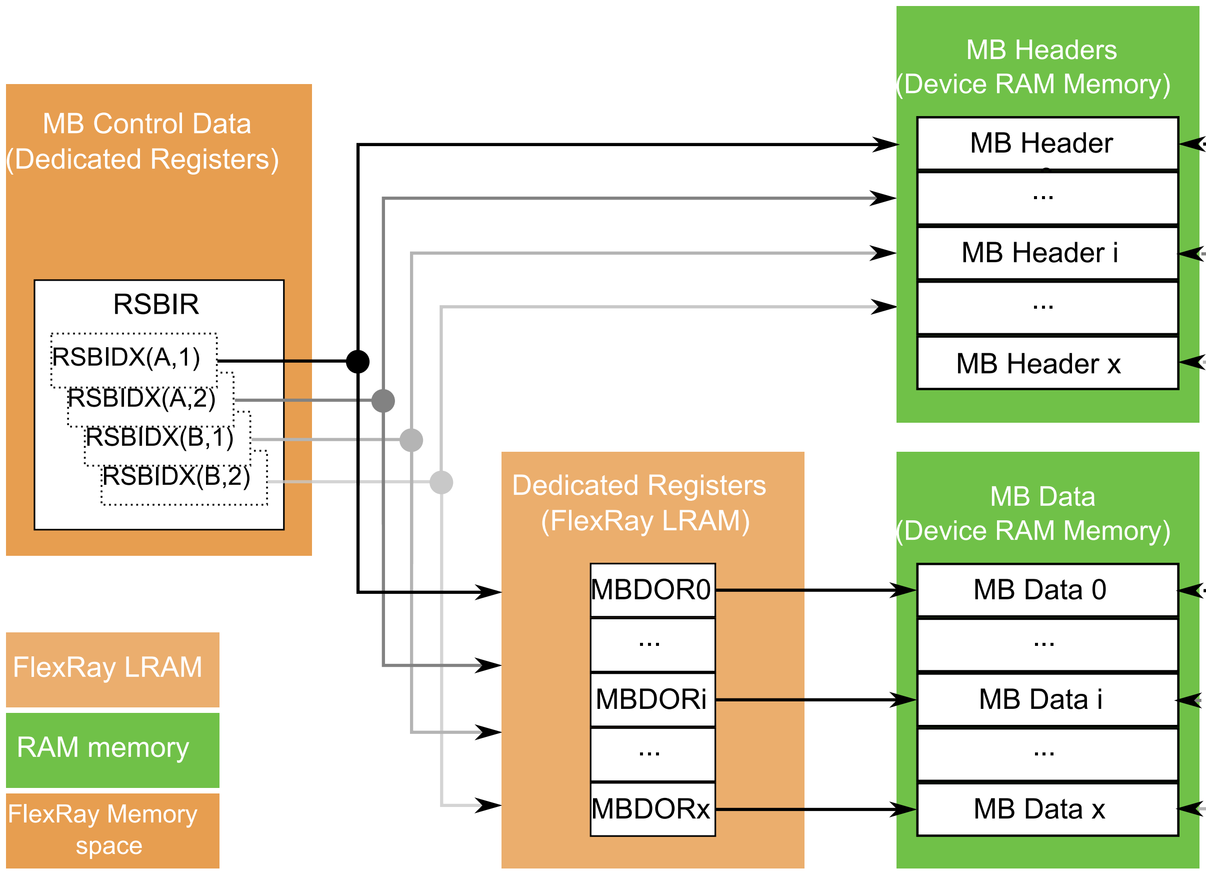


Figure 27. Receive shadow MB structure

4.5.2 Receive MB

The used MB structure is described in Organization of data and the MB is configured as Receive MB (MBCCSRn[MTD] = 0x0).

4.5.3 Transmit MB_1

The used MB structure is described in the Organization of the data and the MB is configured as Transmit MB (MBCCSRn[MTD] = 0x1).The

4.5.4 FIFO structure

There are two independent FIFO memories, one per each channel. Both FIFO memories shares two registers Global Interrupt Flag and Enable Register (GIFER) and Receive FIFO Fill Level and POP Count Register (RFFLPCR). Each FIFO has its own Read Index register (RFARIR/RFBRIR) which point to the next available FIFO entry that application can read. The rest of the FIFO registers are used either for configuration of FIFO A or FIFO B depending on the configuration of the Watermark and Selection Register bitfield SEL (RFWMSR[SEL]=0 FIFO for channel A, RFWMSR[SEL]=1 FIFO for channel B):

- Register for FIFO Configuration:
  - Receive FIFO Start Index Register (RFSIR)
    - Index of the first MB header
  - Receive FIFO Watermark and Selection Register (RFWMSR)
    - FIFOs watermarks
    - Selection which FIFO is configured

- Receive FIFO Depth and Size Register (RFDSR)
  - Depth of the FIFO
  - Size of FIFO item
- Receive FIFO Start Data Offset Register (RFSDOR)
  - Offset of the first MB data

• Registers for filtering the MB for the FIFO:

- Receive FIFO Message ID Acceptance Filter Value Register (FR_RFMIDAFVR)
- Receive FIFO Message ID Acceptance Filter Mask Register (FR_RFMIDAFMR)
- Receive FIFO Frame ID Rejection Filter Value Register (FR_RFFIDRFVR)
- Receive FIFO Frame ID Rejection Filter Mask Register (FR_RFFIDRFMR)
- Receive FIFO Range Filter Configuration Register (FR_RFRFCFR)
- Receive FIFO Range Filter Control Register (FR_RFRFCTR)

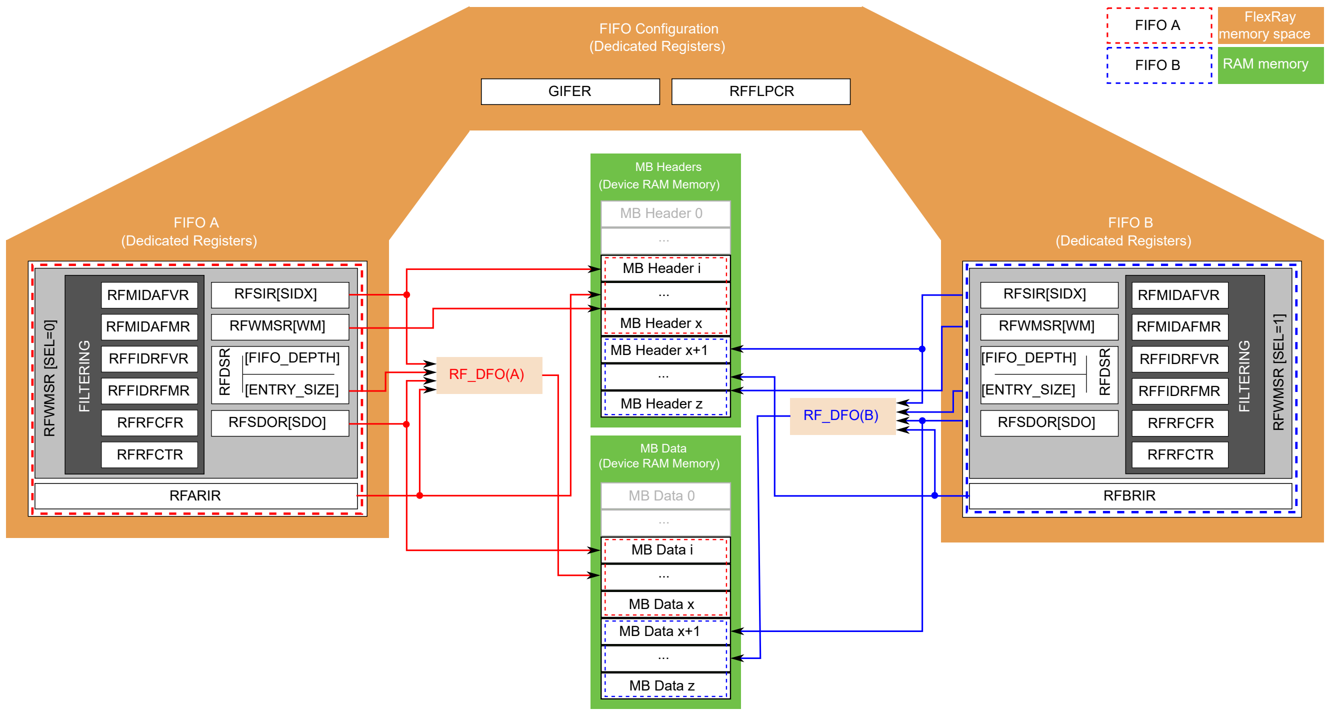


Figure 28. Receive FIFO structure

Data pointer is calculated as below:

RF_DFO(?) = RFSDOR[SDO] + (RFDSR[ENTRY_SIZE * 2)*(RF?RIR[RDIDX]-RFSIR[SIDX]),where “?” is A or B depends on the selected FIFO channel by the RFWMSR[SEL] bit.

4.5.5 Filtering the MBs

The filtering is applicable for all the receivied MBs. There are two different ways how the MBs are filtered based on where the receive MBs will be stored. The first way is for the Individual MBs and the second for FIFO.

4.5.5.1 Individual MBs filtering

Receiving Filtering is based on the position (in which channel/channels and in which cycle/cycles) where the Frame with concrete ID is received.

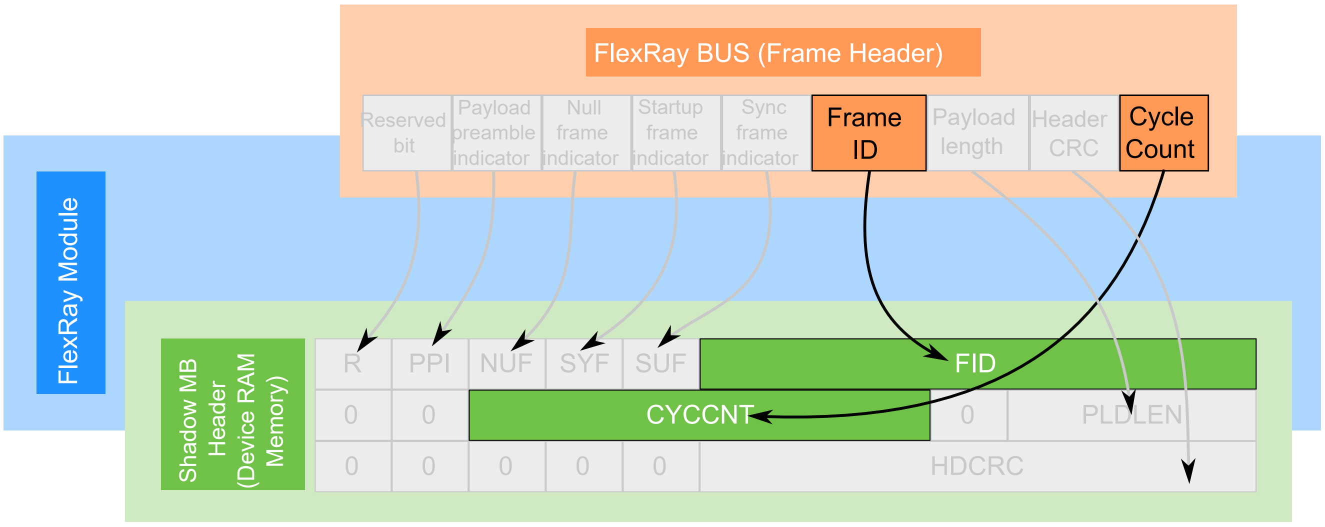


Figure 29. Frame/MB fields used for receiving filtering

When the frame ID of the received frame stored in the shadow MB is equal to ID configured in some individual MB (Register MBFIDRn), the received frame is stored (connected) into this individual MB when the cycle filtering is not enabled (MBCCFRn[CCFE] = 0x0). When the cycle filtering is enabled (MBCCFRn[CCFE] = 0x1) the MB needs to pass the cycle filter before it is stored into the individual MB. When the MB does not pass the filter, it is ignored. When the FlexRay module doesn't find the Frame ID in any individual MB, it check if it passes to FIFO.

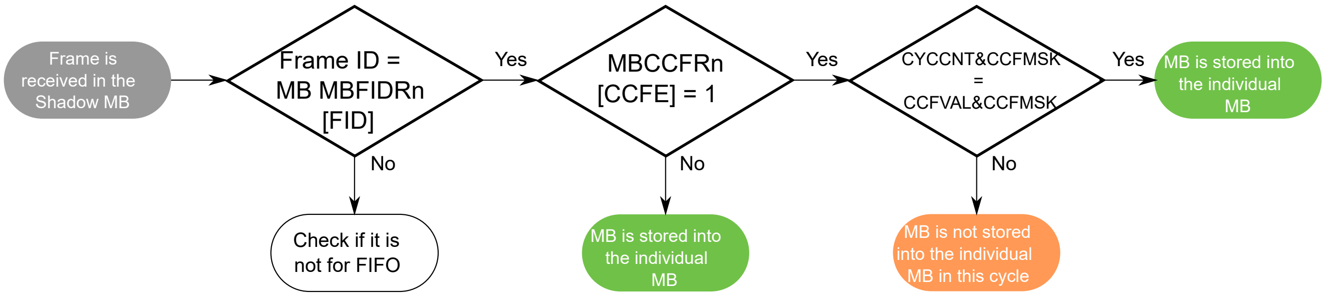

Frame is received in the Shadow MB
Frame ID = MB MBFIDRn [FID]
Yes
No
Check if it is not for FIFO
MBCCFRn [CCFE] = 1
Yes
No
MB is stored into the individual MB
CYCCNT&CCFMSK = CCFVAL&CCFMSK
Yes
No
MB is not stored into the individual MB in this cycle
MB is stored into the individual MB

Figure 30. Individual MB receiving filtering

The cycle filtering defines in which cycle the frame with concrete ID is received. This is configured by MBCCFRn register bitfield CCFE, CCFMSK and CCVFVAL bitfield. When the frame passes the filter (fulfill the equation 1) it is received and stored (connected) into the individual MB with this ID.

$$CYCCNT\&CCFMSK = CCFVAL\&CCFMS$$

Figure 31. Equation 1

Example: The application wants to receive frame in every 4th cycle, start in cycle 1.

1. The periodicity is configurable by the CCFMSK – cycle counter filtering mask CCFMSK = Perioda-1 = 4-1=3
2. The start offset is done by configuring the CCFVAL – cycle counter filtering mask CCFVAL = in which cycle to start = 1
3. We need to enable the cycle filtering by setting the CCFE bit

The channel/channels in which the frame is received, is configured in the MBCCFRn register.

Table 4. Channel “filtering” configuration

MBCCFRn		Transmitting MB		Receiving MB	
CHA	CHB	Static segment	Dynamic segment	Static segment	Dynamic segment
0	0	No frame transmission		No frame stored	
0	1	channel B		channel B	
1	0	channel A		channel A	
1	1	channel A channel B	channel A only	either channel A or channel B	channel A, ignore channel B

Transmitting Filtering: The transmitting filtering is based on position (in which channel/channels and in which cycle/cycles) where the MB with concrete ID is transmitted.

When the Slot counter value of the current frame is equal to ID configured in some individual MB (Register MBFIDRn) and the cycle filtering is not enabled (MBCCFRn[CCFE] = 0x0) this individual MB is transmitted. When the cycle filtering is enabled (MBCCFRn[CCFE] = 0x1) the MB needs to pass the cycle filter before it is transmitted. When the MB doesn’t pass the filter, it is not transmitted.

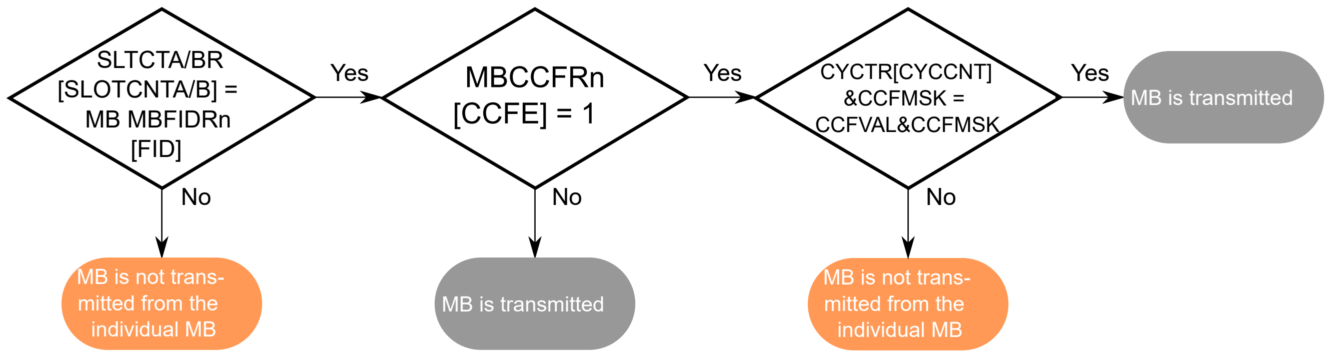

SLTCTA/BR [SLOTCNTA/B] = MB MBFIDRn [FID]
Yes
No
MB is not trans-mitted from the individual MB
MBCCFRn [CCFE] = 1
Yes
No
MB is transmitted
CYCTR[CYCCNT] &CCFMSK = CCFVAL&CCFMSK
Yes
No
MB is not trans-mitted from the individual MB
MB is transmitted

Figure 32. Individual MB transmitting filtering

The cycle filtering defines in which cycle the frame with concrete ID is transmitted. This is configured by MBCCFRn register bitfields CCFE,CCFMSK and CCVFVAL bitfield. When the frame pass the filter (fulfill the equation 2) it is transmitted.

CYCTR[CYCCNT]&CCFMSK = CCFVAL&CCFMS

Figure 33. Equation 2

The channel/channels in which the frame is transmitted, is configured in the MBCCFRn register. See Main timing variables ranges for more information.

4.5.5.2 FIFO filtering

The FIFO filtering is based on the Frame ID.

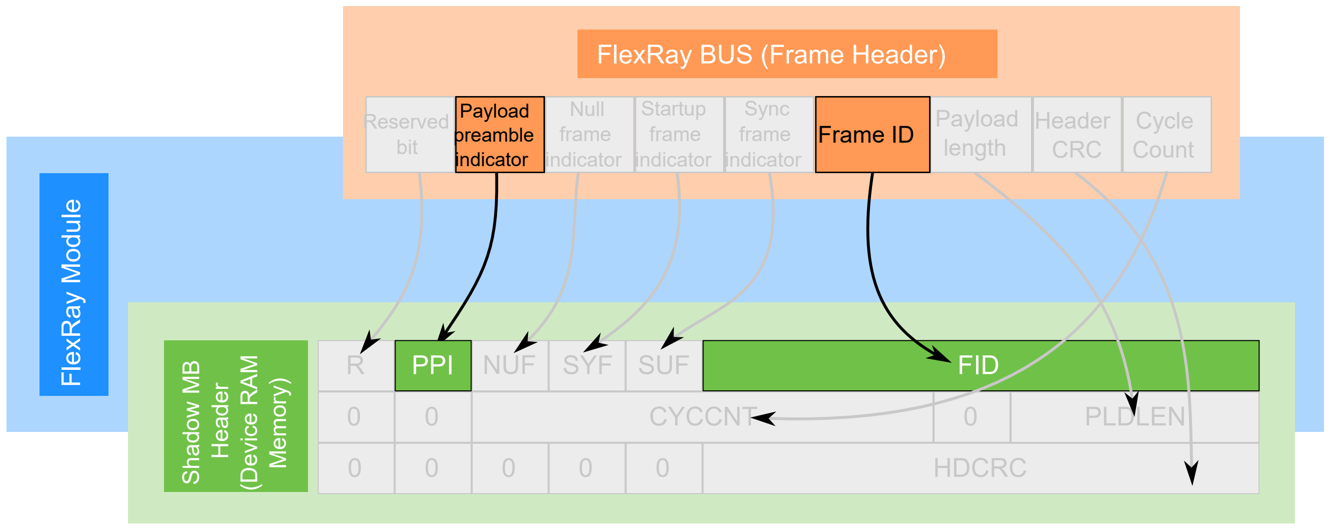


Figure 34. Frame/MB fields used for FIFO filtering

The frame is received when it is not null frame and it passes the four FIFO filters mentioned below:

1. Frame ID Value-Mask Rejection filter
2. Frame ID Range Rejection filter
3. Frame ID Range Acceptance filter
4. Message ID Acceptance filter

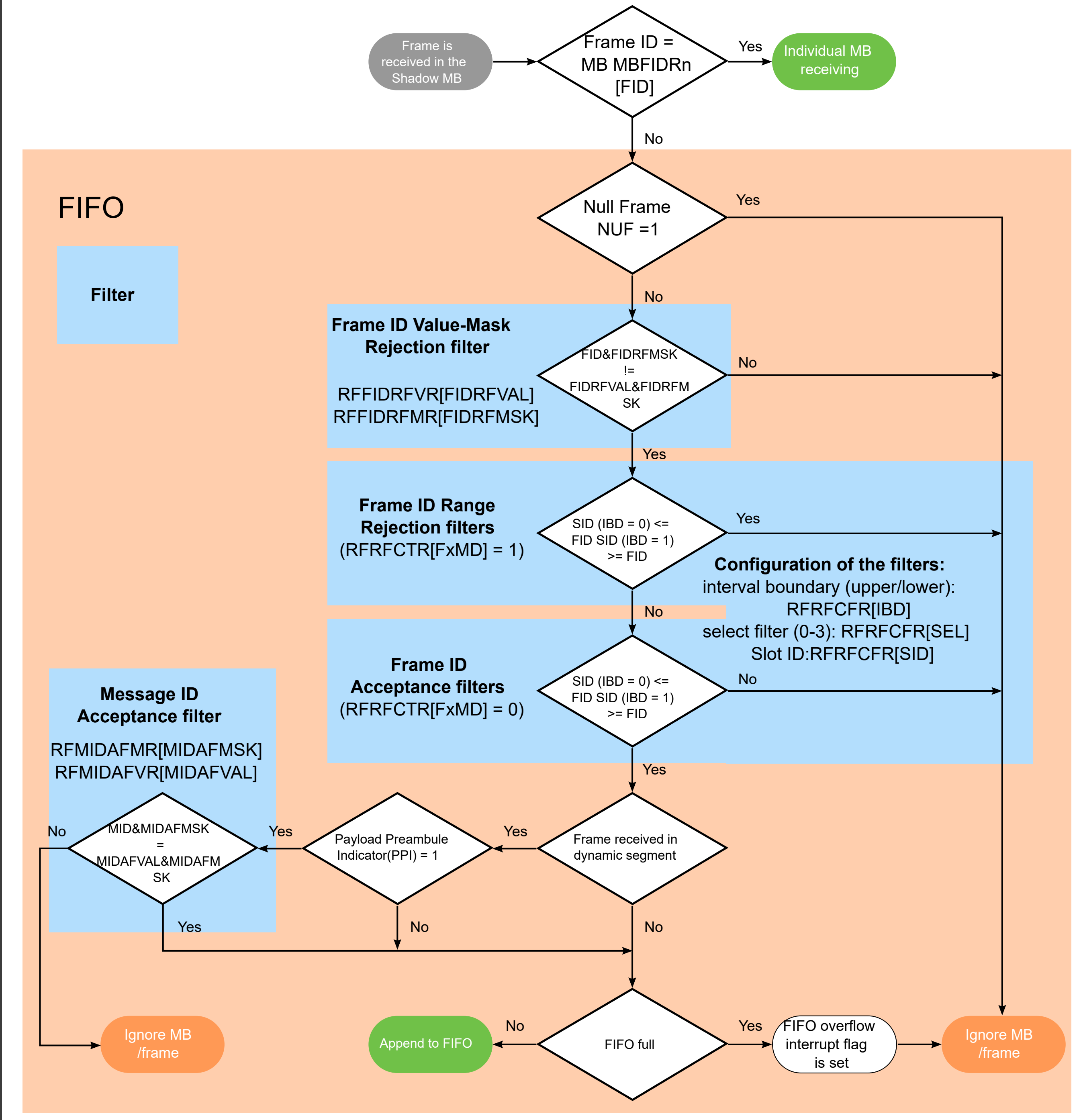

Frame is received in the Shadow MB
Frame ID = MB MBFIDRn [FID]
Yes
Individual MB receiving
No
FIFO
Filter
Null Frame NUF =1
Yes
No
Frame ID Value-Mask Rejection filter
RFFIDRFVR[FIDRFVAL]
RFFIDRFMR[FIDRFMSK]
FID&FIDRFMSK != FIDRFVAL&FIDRFMSK
No
Yes
Frame ID Range Rejection filters (RFRFCTR[FxMD] = 1)
SID (IBD = 0) <= FID SID (IBD = 1) >= FID
Yes
No
Configuration of the filters:
interval boundary (upper/lower): RFRFCFR[IBD]
select filter (0-3): RFRFCFR[SEL]
Slot ID:RFRFCFR[SID]
Frame ID Acceptance filters (RFRFCTR[FxMD] = 0)
SID (IBD = 0) <= FID SID (IBD = 1) >= FID
No
Yes
Message ID Acceptance filter
RFMIDAFMR[MIDAFMSK]
RFMIDAFVR[MIDAFVAL]
MID&MIDAFMSK = MIDAFVAL&MIDAFMSK
No
Yes
Payload Preambule Indicator(PPI) = 1
Yes
Frame received in dynamic segment
Yes
No
No
FIFO full
No
Yes
Append to FIFO
FIFO overflow interrupt flag is set
Ignore MB /frame
Ignore MB /frame

Figure 35. FIFO filtering

There are four Frame ID Range filters (2) a 3)). Each of them can be configure either as rejection filter or acceptance filter by the RFRFCTR[FxMD] bit.

Message ID (MID) Acceptance filter is valid only for dynamic segment where MID can be stored in the begining of the payload segment which is signalized by the Payload Preambule Indicator (PPI) bit in the Frame header. The MID is 16 bit long optional bitfiled which occupies the first two bytes of the Payload segment. The content of the MID is fully configured by the application which put there the information about the data in the payload segment.

5 Receiving frame

Receive shadow buffers are used for receiving data. There is one Receive shadow buffer for each segment and channel. It means there are four of them (two channels [A,B] and two segments [static, dynamic]).

The configuration of the receive shadow buffers is stored in RSBIR register.

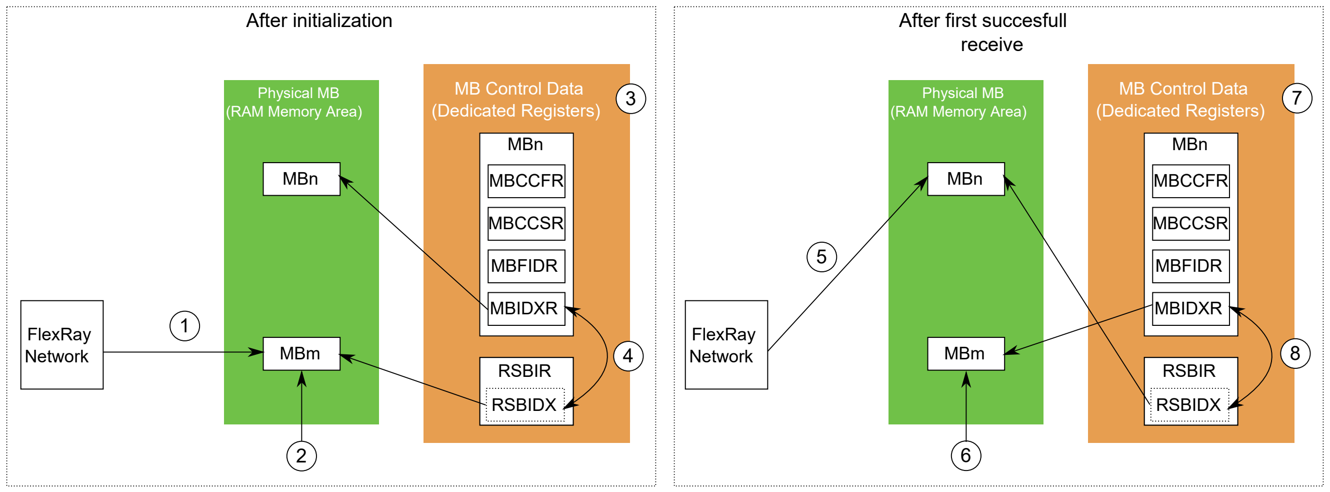

After initialization
Physical MB (RAM Memory Area)
MB Control Data (Dedicated Registers)
3
MBn
MBn
MBCCFR
MBCCSR
MBFIDR
MBIDXR
FlexRay Network
1
MBm
4
RSBIR
RSBIDX
2
After first succesfull receive
Physical MB (RAM Memory Area)
MB Control Data (Dedicated Registers)
7
MBn
MBn
MBCCFR
MBCCSR
5
MBFIDR
MBIDXR
FlexRay Network
MBm
8
RSBIR
RSBIDX
6

Figure 36. Receiving frame/MB

NOTE

The physical MB consists of two parts: Header and Data (See figure: MB structure). Both are represented by MBx labels for simplification in the above figure.

1. The data is received to the MBm whose index is stored in the Receive Shadow Buffer Index Register RSBIR[RSBIDX]. Each channel (A,B) and segment (1, 2) has its own receive shadow buffer register.
2. As soon as the data is received, they are check for validity.
3. When the data is valid, FlexRay module check if there is any MB configured for receiving at the slot in which has been received data by the Receive shadow register.
4. The receive MBn index (stored in the MBIDXR register of the MB) is swapped with the Receive shadow buffer index (stored in the RSBIR[RSBIDX]). On the right side of the figure we can see the pointers after successfully receiving the data.
5. When the new data is received, it is stored into the MBn whose index is stored in the RSBIR[RSBIDX].
6. Data is check if it is valid.
7. When the data is valid, FlexRay module check if there is any MB configured for receiving the slot which has been received by the Receive shadow register.
8. The receive MBn index (stored in the MBIDXR register) is swapped with the Receive shadow buffer index (stored in the RSBIR[RSBIDX]).
9. And so on.

6 Communication startup

There must be minimally two nodes to start up the communication on FlexRay interface. Both these nodes needs to be configured as coldstart nodes to do the start up sequence. The startup sequence is described in Communication startup diagram. The following is the summary of the start up sequence:

1. When the coldstart node is prepared [ready], it tries to receive frame on the channel [coldstart listen].

2. When the node do not receive any frame. The node transmits the collision avoidance symbol (CAS) in case there is not any transition. The node which send the CAS is called leading coldstart node.
3. After the CAS was sent, only the leading coldstart node is allowed to send start up frame for four cycles [coldstart collision resolution].
4. Other coldstart nodes begin to transmit their startup frames in fourth cycle.
5. The leading coldstart node collects all startup frames from cycle four and five and perform clock correction [coldstart consistency check].
6. The leading coldstart node leaves the startup and enters operation [normal active] when following are true:
    a. Clock correction doesn't signal any errors
    b. The node receives at least one valid startup frames pair
7. The other coldstart node when they are prepared [ready] try to receive frame on channel [coldstart listen]. During that time, they found that one node sent the CAS symbol.
8. When communication is received, it tries to receive a valid pair of startup frames to derivate its schedule and clock correction from the leading coldstart node [initialize schedule].
9. When the previous step is complete. The node performs clock correction in the following double-cycle [integration coldstart check].
10. The following coldstart node begins to transmit its startup frame when following are true [coldstart join]:
    a. Clock correction doesn't signal any errors
    b. The node continues receiving sufficient frames from the leading coldstart node.
11. The following node leaves the startup and enters operation [normal active] when following are true:
    a. Clock correction doesn't signal any errors for following three cycles.
    b. One other coldstart node is visible.
12. The following coldstart node leaves startup at least one cycles later than the leading coldstart node.

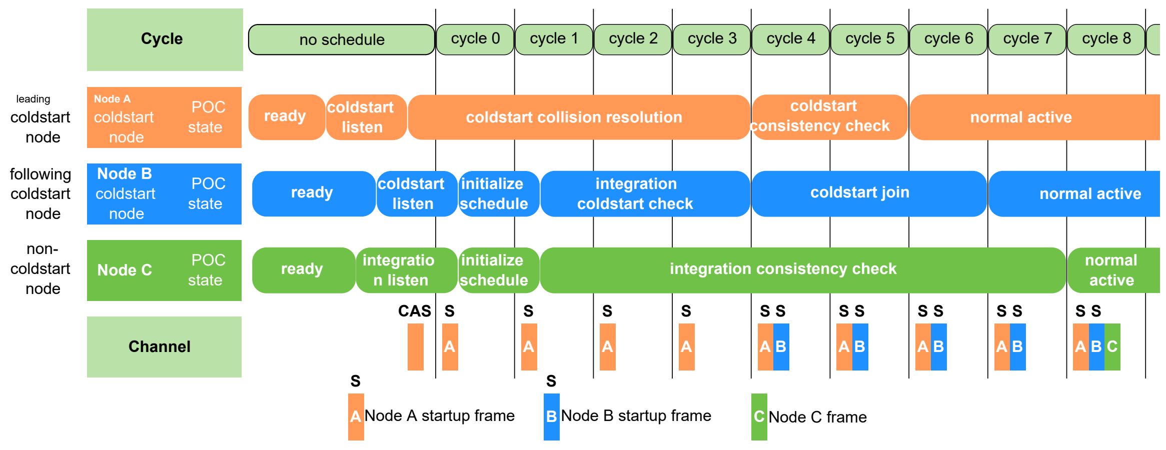


Figure 37. Communication startup diagram

13. When the non-coldstart node is prepared [ready], it tries to receive frame [integration listen].
14. When communication is received, it tries to receive a valid pair of startup frames to derivate its schedule and clock correction from the leading coldstart node [initialize schedule].
15. In the following cycles, the node tries to find at least two startup frames from different coldstart nodes [integration consistency check].

16. The non-coldstart node leaves the startup and enters operation [normal active] when it receives two valid startup frame pairs with different frame IDs (from different nodes) for two consecutive double cycles.The non-coldstart node leaves startup at least two cycles later than the leading coldstart node.

7 Synchronization

FlexRay interface is time synchronous, so it is necessary to do the synchronization. The following two synchronization are necessary:

1. Offset synchronization
2. Rate(frequency) synchronization

Offset synchronization: The cycle is in wrong place than it should be (it is earlier or later). See the following figure.

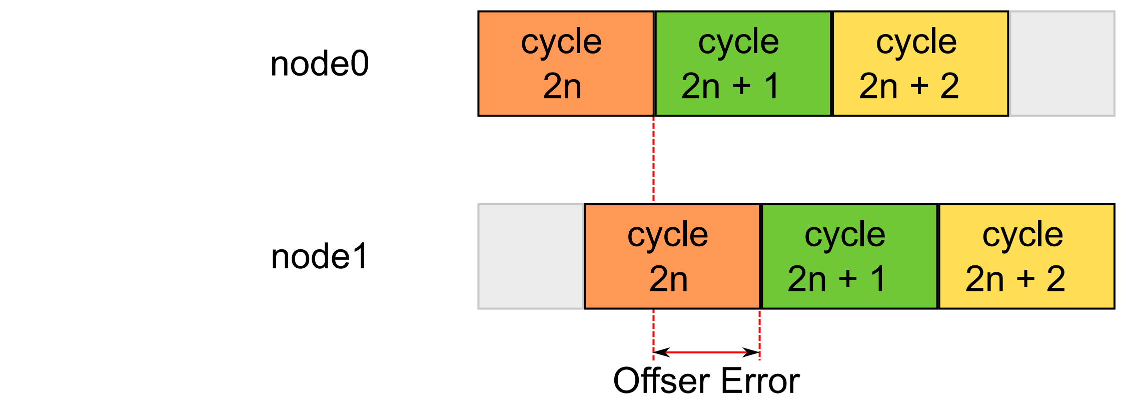


Figure 38. Offset error

How the offset synchronization is done:

The Measuring is executed every cycle during the static segment. The offset correction (vOffsetCorrection) is calculated between the static segment and the offset correction segment (gOffsetCorrectionStart). The offset correction is applied every odd cycle. See the figure below for better understanding. The offset correction is done by measuring the value calculated during the same cycle. The offset correction value can also be negative.

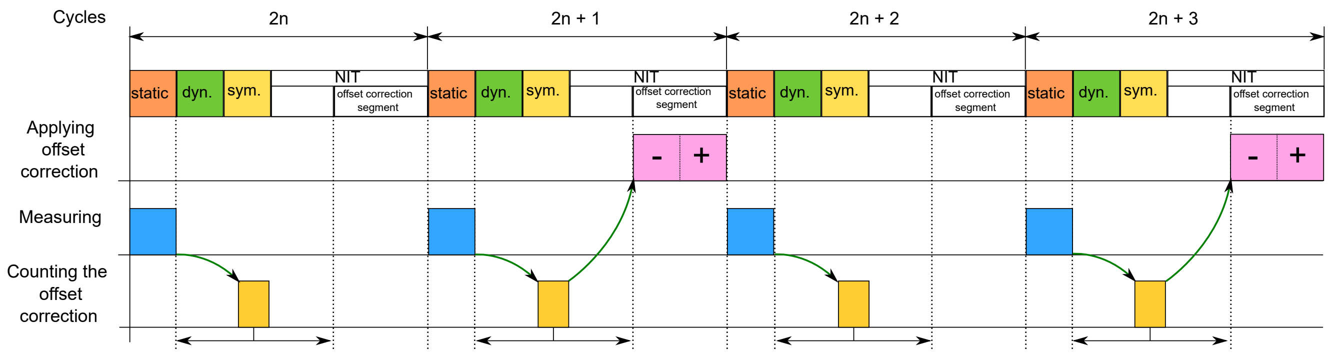

Cycles
2n
2n + 1
2n + 2
2n + 3
static
dyn.
sym.
NIT
offset correction segment
Applying offset correction
-
+
Measuring
Counting the offset correction

Figure 39. Description of the offset synchronization

Rate synchronization:The length of each cycle is shorter/longer than it should be. See figure below.

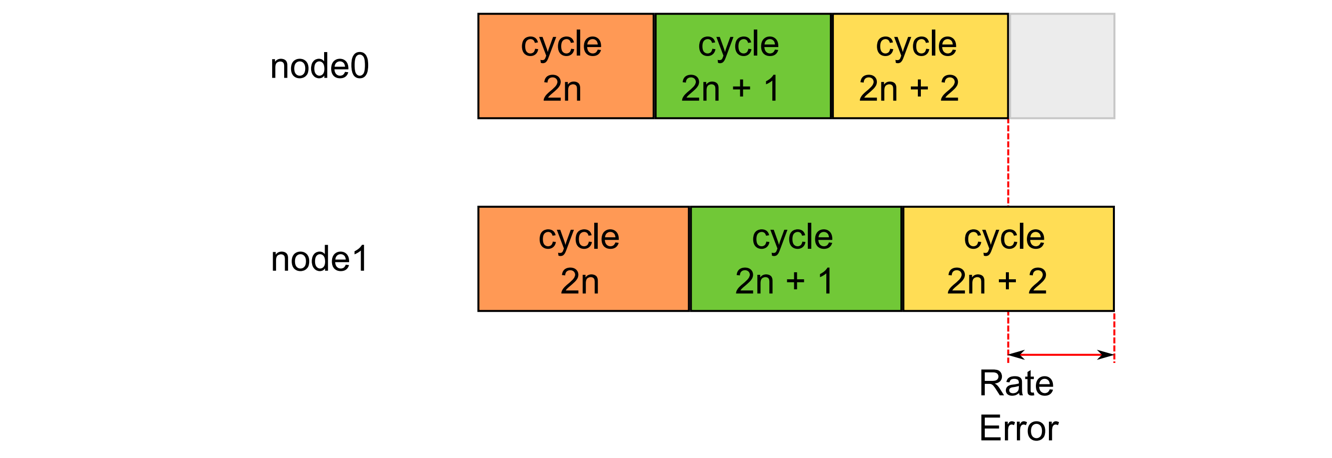


Figure 40. Rate error

How rate synchronization is done: The Measuring is executed every cycle during the static segment. The offset correction (vRateCorrection) is calculated from two cycles in queue. The calculation needs to be finished before the new cycle starts. This correction value is used for two cycles until the new value is calculated. The calculation is executed only during the odd cycle. See the figure below for better understanding. The rate correction value can also be negative.

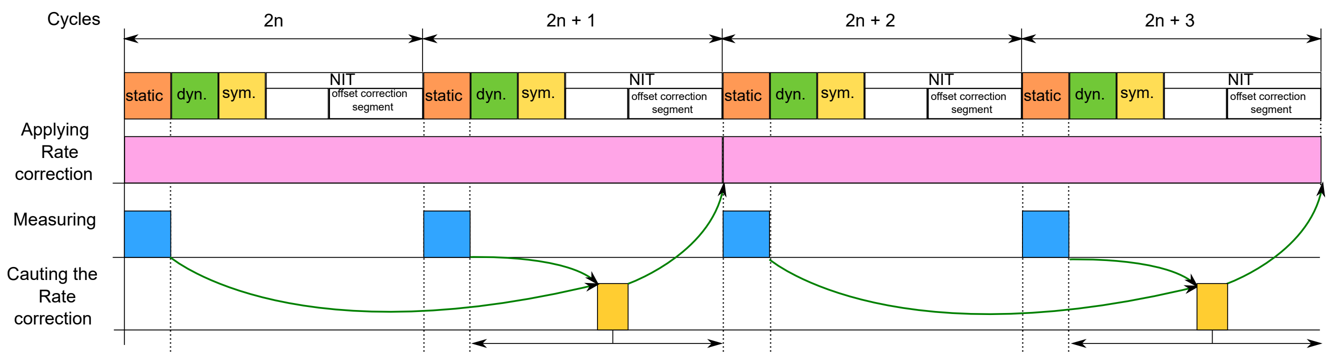


Figure 41. Description of the rate synchronization

8 How to configure FlexRay

8.1 MPC5744P

8.1.1 Features

Number of configurable MBs = 64

Total number of MBs = 64 + 4 (Receive shadow MB) = 68 MBs

The FIFO number of entries = up to 255

8.1.2 FlexRay clock configuration

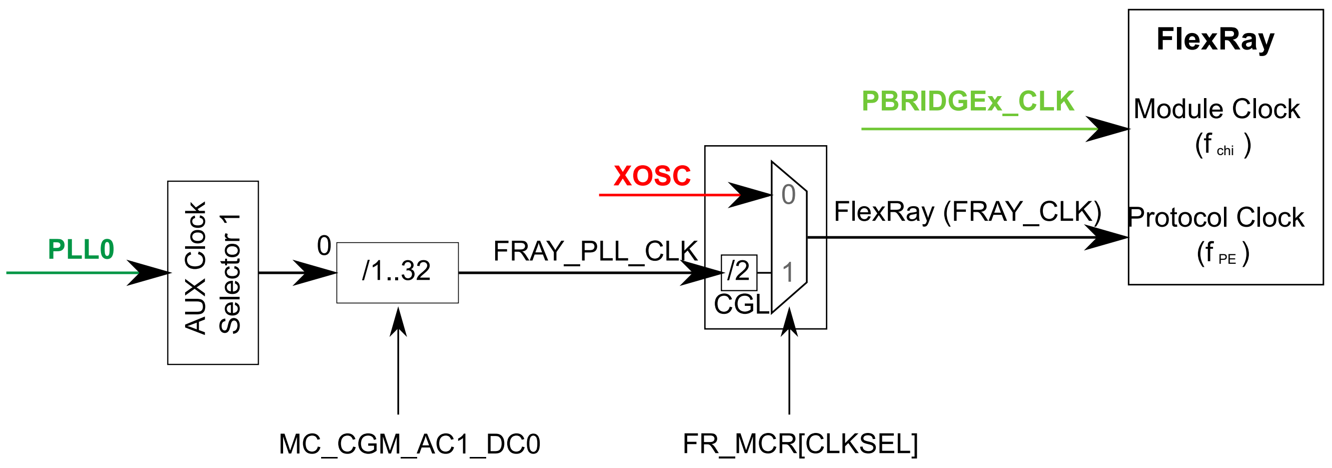


Figure 42. Clocking scheme for MPC5744P

The FRAY_CLK clock (fPE) can be either supplied by 40 MHz oscillator clock or the 80 MHz PLL clock. The reason is that the module use both edges of the 40 MHz crystal signal which has stable frequency and duty cycle compared to the PLL which has only the stable frequency. It means that the PLL must be 2 times faster than oscillator frequency.

fchi frequency has following restriction:

fchi [MHz]>=(FR_MBSSUTR[LAST_MB_UTIL]+27)/(39*pdMicrotick [us]*gdMinislot [MT]), where

Table 5. Ranges of pdMicrotick, gdMinislot and FR_MBSSUTR[LAST_MB_UTIL]

Range	Min	Max
FR_MBSSUTR[LAST_MB_UTIL]	0	63
pdMicrotick [us]	0.0125	0.1
gdMinislot [MT]	2	63

pdMicrotick depends on the interface communication speed:

Table 6. pdMicrotick value based on Bit Rate

Bit Rate [Mb/s]	2.5	5	8	10
pdMicrotick [us]	**0.050**	**0.025**	0.0125	0.0125
	0.100	0.050	**0.0250**	**0.0250**
		0.100	0.0500	0.0500

8.1.3 FlexRay memory configuration

Configure the memory space for the module in the RAM memory:

- It is necessary to set the FlexRay RAM base address into the SYMBADHR and SYMBADLR registers (Also RFSYMBADHR and RFSYMBADLR registers when MCR[FAM] = 1). The base address must be aligned to 32 bytes.
- Cache must be inhibited for RAM memory which is used for FlexRay module. (Use CMPU/SMPU module for it).

Memories initialization:

- The system RAM which is used for the FlexRay MBs - Needs to be initialized before using.

- LRAM (Look Up Table RAM. Module internal memory to store message buffer configuration data and data field offsets for individual message buffers and receive shadow buffers) – It is initialized by the module when it leaves the disable mode.

8.1.4 FlexRay pins configuration

Table 7. FlexRay pin muxing for the MPC5744P device

Channel	Signal name	direction	MSCR	IMCR	SSSS	Pin Name	LQFP	BGA
A	TXEN	O	47	-	1	C[15]	124	A8
	TX	O	48	-	1	D[0]	125	B8
	RX	I	49	136	1	D[1]	3	E3
B	TXEN	O	52	-	1	D[4]	129	B7
	TX	O	51	-	1	D[3]	128	A5
	RX	I	50	137	1	D[2]	140	B4
Debug	DBG[0]	O	104	-	1	G[8]	81	N14
	DBG[1]	O	105	-	1	G[9]	79	P14
	DBG[2]	O	106	-	1	G[10]	77	R17
	DBG[3]	O	107	-	1	G[11]	75	T15

8.1.5 FlexRay Interrupts configuration

Table 8. FlexRay interrupts for the MPC5744P device

Interrupt number	Interrupt name	Source of interrupt
453	LRAM Non-Corrected Error interrupt	LRAM ECC
	DRAM Non-Corrected Error interrupt	DRAM ECC
454	LRAM Corrected Error interrupt	LRAM ECC
	DRAM Corrected Error interrupt	DRAM ECC
455	RX FIFO A Almost Full interrupt	RX FIFO A
456	RX FIFO B Almost Full interrupt	RX FIFO B
457	Wakeup interrupt	Protocol Engine (PE)
458	Protocol interrupt	Protocol Engine (PE)
459	CHI interrupt	Controller host Interface (CHI)
460	Transmit Message Buffer interrupt	Controller host Interface (CHI)
461	Receive Message Buffer interrupt	Controller host Interface (CHI)
462	Module Interrupt	ORed interrupts number 455-461

9 Links

- MPC5744P RM
- FlexRay Communications System Protocol Specification Version 2.1
- Vector FlexRay Protocol Reference Chart: know-how, Embedded Software, and Services for FlexRay

Date of release: May 2021
Document identifier: AN12233