41.3 Functional Description

This chapter describes the E-Ray implementation together with the related FlexRay™ protocol features. More information about the FlexRay™ protocol itself can be found in the FlexRay™ protocol specification v2.1.

Communication on FlexRay™ networks is based on Frames and symbols. The wakeup symbol (WUS) and the collision avoidance symbol (CAS) are transmitted outside the communication cycle to setup the time schedule. Frames and media access test symbols (MTS) are transmitted inside the communication cycle.

41.3.1 Definitions

FlexRay™ Frame: Header Segment + Payload Segment

Message Buffer: Header Section + Data Section

Message RAM: Header Partition + Data Partition

Data Frame: FlexRay™ Frame that is not a NULL Frame

41.3.2 Communication Cycle

A communication cycle in FlexRay™ consists of the following elements:

- Static Segment
- Dynamic Segment
- Symbol Window
- Network Idle Time (NIT)

Static segment, dynamic segment, and symbol window form the Network Communication Time (NCT). For each communication channel the slot counter starts at 1 and counts up until the end of the dynamic segment is reached. Both channels share the same arbitration grid which means that they use the same synchronized Macrotick.

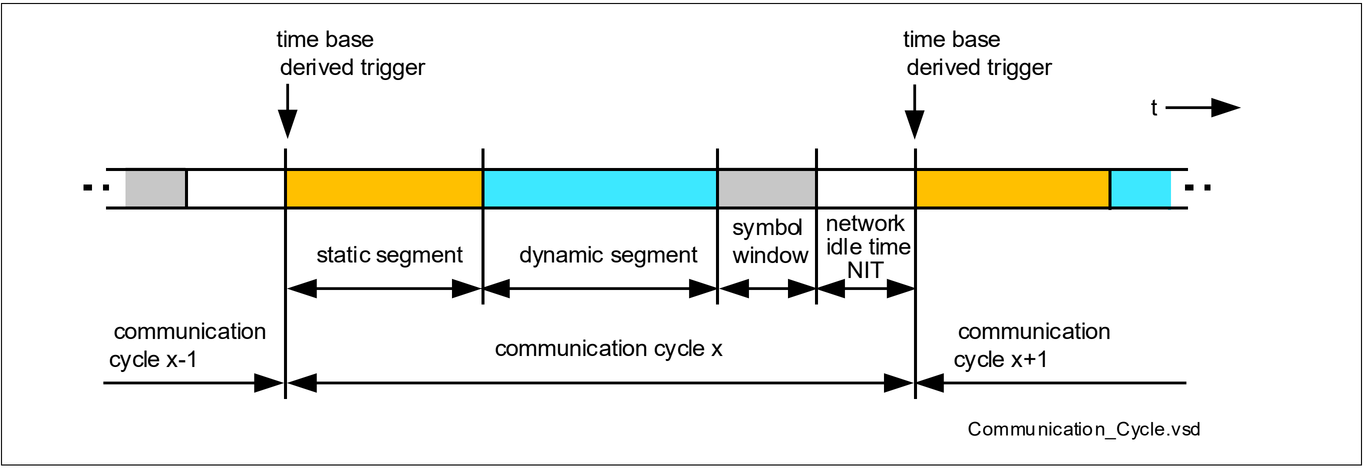

Figure 608 Structure of Communication Cycle

41.3.2.1 Static Segment

The Static Segment is characterized by the following features:

- Time slots of fixed length (optionally protected by bus guardian)
- Start of Frame transmission at action point of the respective static slot
- Payload length same for all Frames on both channel

Parameters: Number of Static Slots GTUC07.NSS, Static Slot Length GTUC07.SSL, Payload Length Static MHDC.SFDL, Action Point Offset GTUC09.APO.

41.3.2.2 Dynamic Segment

The Dynamic Segment is characterized by the following features:

- All controllers have bus access (no bus guardian protection possible)
- Variable payload length and duration of slots, different for both channels
- Start of transmission at minislot action point

Parameters: Number of Minislots GTUC08.NMS, Minislot Length GTUC08.MSL Minislot Action Point Offset GTUC09.MAPO, Start of Latest Transmit (last minislot) MHDC.SLT.

41.3.2.3 Symbol Window

During the symbol window only one media access test symbol (MTS) may be transmitted per channel. MTS symbols are send in “NORMAL_ACTIVE” state to test the bus guardian.

The symbol window is characterized by the following features:

- Send single symbol
- Transmission of the MTS symbol starts at the symbol windows action point

Parameters: Symbol Window Action Point Offset GTUC09.APO (same as for static slots), Network Idle Time Start GTUC04.NIT.

41.3.2.4 Network Idle Time (NIT)

During network idle time the Communication Controller has to perform the following tasks:

- Calculate clock correction terms (offset and rate)
- Distribute offset correction over multiple Macroticks
- Perform cluster cycle related tasks

Parameters: Network Idle Time Start GTUC04.NIT, Offset Correction Start GTUC04.OCS.

41.3.2.5 Configuration of Network Idle Time (NIT) Start and Offset Correction Start

The number of Macroticks per cycle (gMacroPerCycle) is assumed to be m. It is configured by programming GTUC02.MPC = m.

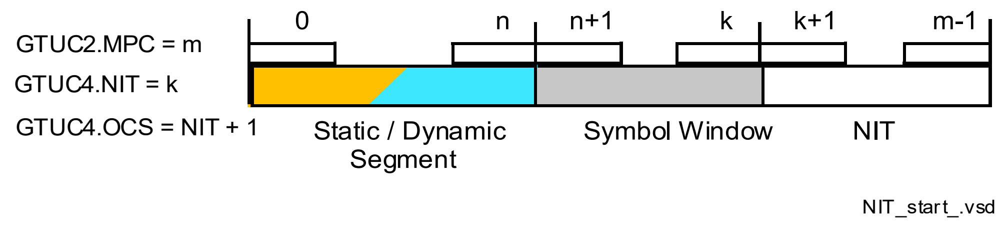

Figure 609 Configuration of network idle time (NIT) start and offset correction start

The static / dynamic segment starts with Macrotick 0 and ends with Macrotick n:
n = static segment length + dynamic segment offset + dynamic segment length - 1 Macrotick

n = gNumberOfStaticSlots • gdStaticSlot + dynamic segment offset
+ gNumberOfMinislots • gdMinislot - 1 Macroticks

The static segment length is configured by GTUC07.SSL and GTUC07.NSS.
The dynamic segment length is configured by GTUC08.MSL and GTUC08.NMS.

The dynamic segment offset is:

If gdActionPointOffset ≤ gdMinislotActionPointOffset:

dynamic segment offset = 0 MT

Else if gdActionPointOffset > gdMinislotActionPointOffset:

dynamic segment offset = gdActionPointOffset - gdMinislotActionPointOffset

The network idle time (NIT) starts with Macrotick k+1 and ends with the last Macrotick of cycle m-1. It has to be configured by setting GTUC04.NIT = k.

For the E-Ray the offset correction start is required to be

GTUC04.OCS ≥ GTUC04.NIT + 1 = k+1.

The length of symbol window results from the number of Macroticks between the end of the static / dynamic segment and the beginning of the NIT. It can be calculated by k - n.

41.3.3 Communication Modes

The FlexRay™ Protocol Specification v2.1 defines the Time-Triggered Distributed (TT-D) mode.

Time-triggered Distributed (TT-D)

In TT-D mode the following configurations are possible:

- **Pure static**: minimum 2 static slots + symbol window (optional)
- **Mixed static/dynamic**: minimum 2 static slots + dynamic segment + symbol window (optional)

A minimum of two coldstart nodes need to be configured for distributed time-triggered operation. Two fault-free coldstart nodes are necessary for the cluster startup. Each Startup Frame must be a SYNC Frame, therefore all coldstart nodes are sync nodes.

41.3.4 Clock Synchronization

In TT-D mode a distributed clock synchronization is used. Each node individually synchronizes itself to the cluster by observing the timing of received SYNC Frames from other nodes.

41.3.4.1 Global Time

Activities in a FlexRay™ node, including communication, are based on the concept of a global time, even though each individual node maintains its own view of it. It is the clock synchronization mechanism that differentiates the FlexRay™ cluster from other node collections with independent clock mechanisms. The global time is a vector of two values; the cycle (cycle counter) and the cycle time (Macrotick counter).

Cluster specific:

- Macrotick = basic unit of time measurement in a FlexRay™ network, a Macrotick consists of an integer number of Microticks
- Cycle length = duration of a communication cycle in units of Macroticks

41.3.4.2 Local Time

Internally, nodes time their behavior with Microtick resolution. Microticks are time units derived from the oscillator clock tick of the specific node. Therefore Microticks are controller-specific units. They may have different duration in different controllers. The precision of a node's local time difference measurements is a Microtick.

Node specific:

- Oscillator clock → prescaler → Microtick
- Microtick = basic unit of time measurement in a Communication Controller, clock correction is done in units of Microticks
- Cycle counter + Macrotick counter = nodes local view of the global time

41.3.4.3 Synchronization Process

Clock synchronization is performed by means of SYNC Frames. Only preconfigured nodes (sync nodes) are allowed to send SYNC Frames. In a two-channel cluster a sync node has to send its SYNC Frame on both channels.

For synchronization in FlexRay™ the following constraints have to be considered:

- Max. one SYNC Frame per node in one communication cycle
- Max. 15 SYNC Frames per cluster in one communication cycle
- Every node has to use all available SYNC Frames for clock synchronization
- Minimum of two sync nodes required for clock synchronization and startup

For clock synchronization the time difference between expected and observed arrival time of SYNC Frames received during the static segment, valid on both channels (two-channel cluster), is measured. The calculation of correction terms is done during network idle time (NIT) (offset: every cycle, rate: odd cycle) by using a FTA / FTM algorithm. For details see FlexRay™ protocol specification v2.1, chapter 8.

Offset (phase) Correction

- Only deviation values measured and stored in the current cycle used
- For a two channel node the smaller value will be taken
- Calculation during network idle time (NIT) of **every** communication cycle, value may be negative
- Offset correction value calculated in even cycles used for error checking only
- Checked against limit values (violation: “NORMAL_ACTIVE” → “NORMAL_PASSIVE” → “HALT”)
- Correction value is an integer number of Microticks
- Correction done in **odd** numbered cycles, distributed over the Macroticks beginning at offset correction start up to cycle end (end of network idle time (NIT)) to shift nodes next start of cycle (Macroticks lengthened / shortened)

Rate (frequency) Correction

- Pairs of deviation values measured and stored in even / odd cycle pair used
- For a two channel node the average of the differences from the two channels is used
- Calculated during network idle time (NIT) of **odd** numbered cycles, value may be negative
- Cluster drift damping is performed using global damping value
- Checked against limit values
- Correction value is a signed integer number of Microticks
- Distributed over Macroticks comprising the next **even / odd** cycle pair (Macroticks lengthened / shortened)

Synchronization Process

Clock synchronization is performed by means of SYNC Frames. Only preconfigured nodes (sync nodes) are allowed to send SYNC Frames. In a two-channel cluster a sync node has to send its SYNC Frame on both channels.

For synchronization in FlexRay™ the following constraints have to be considered:

- Max. one SYNC Frame per node in one communication cycle
- Max. 15 SYNC Frames per cluster in one communication cycle
- Every node has to use all available SYNC Frames for clock synchronization
- Minimum of two sync nodes required for clock synchronization and startup

For clock synchronization the time difference between expected and observed arrival time of SYNC Frames received during the static segment, valid on both channels (two-channel cluster), is measured. The calculation of correction terms is done during network idle time (NIT) (offset: every cycle, rate: odd cycle) by using a FTA / FTM algorithm. For details see FlexRay™ protocol specification v2.1, chapter 8.

SYNC Frame Transmission

SYNC Frame transmission is only possible from buffer 0 and 1. Message Buffer 1 may be used for SYNC Frame transmission in case that SYNC Frames should have different payloads on the two channels. In this case bit MRC.SPLM has to be programmed to 1.

Message Buffers used for SYNC Frame transmission have to be configured with the key slot ID and can be (re)configured in “DEFAULT_CONFIG” or “CONFIG” state only. For nodes transmitting SYNC Frames SUCC1.TXSY must be set to 1.

41.3.4.4 External Clock Synchronization

During normal operation, independent clusters can drift significantly. If synchronous operation across independent clusters is desired, external synchronization is necessary; even though the nodes within each cluster are synchronized. This can be accomplished with synchronous application of host-deduced rate and offset correction terms to the clusters.

- External offset / rate correction value is a signed integer
- External offset / rate correction value is added to calculated offset / rate correction value
- Aggregated offset / rate correction term (external + internal) is not checked against configured limits

41.3.5 Error Handling

The implemented error handling concept is intended to ensure that in case of a lower layer protocol error in a single node communication between non-affected nodes can be maintained. In some cases, higher layer program command activity is required for the Communication Controller to resume normal operation. A change of the error handling state will set bit EIR.PEMC in the Error Service Request Register and may trigger an service request to the Host if enabled. The actual error mode is signalled by CCEV.ERRM in the Communication Controller Error Vector register.

Table 389 Error Modes of the POC (Degradation Model)

Error Mode	Activity
ACTIVE (green)	**Full operation**, State: “NORMAL_ACTIVE” The Communication Controller is fully synchronized and supports the cluster wide clock synchronization. The host is informed of any error condition(s) or status change by interrupt (if enabled) or by reading the error and status interrupt flags from registers EIR and SIR.
PASSIVE (yellow)	**Reduced operation**, State: “NORMAL_PASSIVE”, Communication Controller self rescue allowed The Communication Controller stops transmitting Frames and symbols, but received Frames are still processed. Clock synchronization mechanisms are continued based on received Frames. No active contribution to the cluster wide clock synchronization. The host is informed of any error condition(s) or status change by interrupt (if enabled) or by reading the error and status interrupt flags from registers EIR and SIR.
COMM_HALT (red)	**Operation halted**, State: “HALT”, Communication Controller self rescue not allowed The Communication Controller stops Frame and symbol processing, clock synchronization processing, and the Macrotick generation. The host has still access to error and status information by reading the error and status interrupt flags from registers EIR and SIR. The bus drivers are disabled.

41.3.5.1 Clock Correction Failed Counter

When the Clock Correction Failed Counter reaches the maximum “without clock correction passive” limit defined by SUCC3.WCP, the POC transits from “NORMAL_ACTIVE” to “NORMAL_PASSIVE” state. When it reaches the “maximum without clock correction fatal” limit defined by SUCC3.WCF, it transits “NORMAL_ACTIVE” or “NORMAL_PASSIVE” to the “HALT” state. Both limits are defined in the SUC Configuration Register 3.

The Clock Correction Failed Counter CCEV.CCFC allows the Host to monitor the duration of the inability of a node to compute clock correction terms after the Communication Controller passed protocol startup phase. It will be incremented by one at the end of any **odd** numbered communication cycle where either the Missing Offset Correction signal SFS.MOCS nor the Missing Rate Correction signal SFS.MRCS flag is set. The two flags are located in the SYNC Frame Status register, while the Clock Correction Failed Counter is located in the Communication Controller Error Vector register.

The Clock Correction Failed Counter is reset to zero at the end of an **odd** communication cycle if neither the Missing Offset Correction signal SFS.MOCS nor the Missing Rate Correction signal SFS.MRCS flag is set.

The Clock Correction Failed Counter stops incrementing when the “maximum without clock correction fatal” value SUCC3.WCF as defined in the SUC Configuration Register 3 is reached (i.e. incrementing the counter at its maximum value will not cause it to “wraparound” back to zero). The Clock Correction Failed Counter is initialized to zero when the Communication Controller enters “READY” state or when “NORMAL_ACTIVE” state is entered.

41.3.5.2 Passive to Active Counter

The passive to active counter controls the transition of the POC from “NORMAL_PASSIVE” to “NORMAL_ACTIVE” state. SUCC1.PTA in the SUC Configuration Register 1 defines the number of consecutive even / odd cycle pairs that must have valid clock correction terms before the Communication Controller is allowed to transit from “NORMAL_PASSIVE” to “NORMAL_ACTIVE” state. If SUCC1.PTA is reset to zero the Communication Controller is not allowed to transit from “NORMAL_PASSIVE” to “NORMAL_ACTIVE” state.

41.3.5.3 HALT Command

In case the Host wants to stop FlexRay™ communication of the local node it can bring the Communication Controller into “HALT” state by asserting the HALT command. This can be done by writing SUCC1.CMD = 0110_B in the SUC Configuration Register 1. When called in “NORMAL_ACTIVE” or “NORMAL_PASSIVE” state the POC transits to “HALT” state at the end of the current cycle. When called in any other state SUCC1.CMD will be reset to 0000_B = “COMMAND_NOT_ACCEPTED” and bit EIR.CNA in the Error Service Request Register is set to 1. If enabled an service request to the Host is generated.

41.3.5.4 FREEZE Command

In case the Host detects a severe error condition it can bring the Communication Controller into “HALT” state by asserting the FREEZE command. This can be done by writing SUCC1.CMD = 0111_B in the SUC Configuration Register 1. The FREEZE command triggers the entry of the “HALT” state immediately regardless of the actual POC state.

The POC state from which the transition to HALT state took place can be read from CCSV.PSL.

41.3.6 Communication Controller States

This chapter introduces the states of the Communication Controller.

41.3.6.1 Communication Controller State Diagram

State transitions are controlled by externals the application reset or RXDA/B, by the POC state machine, and by the CHI Command Vector SUCC1.CMD located in the SUC Configuration Register 1.

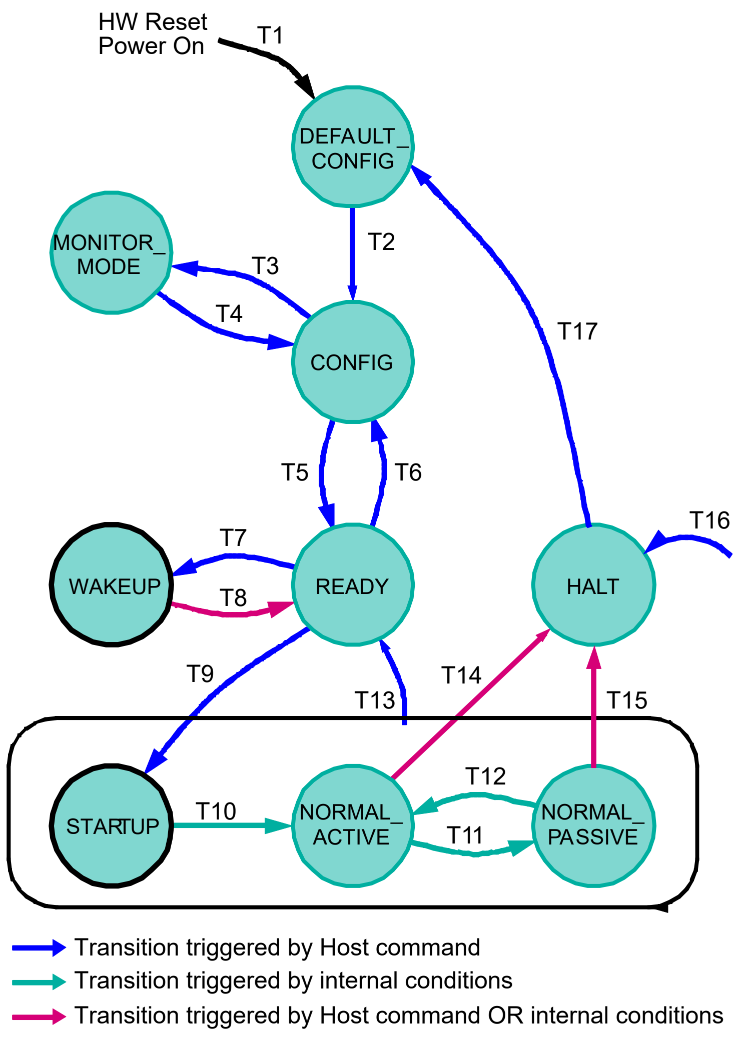

Figure 610 Overall State Diagram of E-Ray Communication Controller

The Communication Controller exits from all states to “HALT” state after application of the FREEZE command (SUCC1.CMD = 0111_B).

Table 390 State Transitions of E-Ray Overall State Machine

T#	Condition	From	To
1	application reset	HW Reset	DEFAULT_CONFIG
2	Command CONFIG, SUCC1.CMD = 0001_B	DEFAULT_CONFIG	CONFIG
3	Unlock sequence followed by command MONITOR_MODE, SUCC1.CMD = 1011_B	CONFIG	MONITOR_MODE
4	Command CONFIG, SUCC1.CMD = 0001_B	MONITOR_MODE	CONFIG
5	Unlock sequence followed by command READY, SUCC1.CMD = 0010_B	CONFIG	READY
6	Command CONFIG, SUCC1.CMD = 0001_B	READY	CONFIG
7	Command WAKEUP, SUCC1.CMD = 0011_B	READY	WAKEUP
8	Complete, non-aborted transmission of wakeup pattern OR received WUP OR received Frame Header OR command READY, SUCC1.CMD = 0010_B	WAKEUP	READY
9	Command RUN SUCC1.CMD = 0100_B	READY	STARTUP
10	Successful startup	STARTUP	NORMAL_ACTIVE
11	Clock Correction Failed counter reached Maximum Without Clock Correction Passive limit configured by WCP in SUC Configuration Register 3	NORMAL_ACTIVE	NORMAL_PASSIVE
12	Number of valid correction terms reached the Passive to Active limit configured by PTA in SUC Configuration Register 1	NORMAL_PASSIVE	NORMAL_ACTIVE
13	Command READY, SUCC1.CMD = 0010_B	STARTUP, NORMAL_ACTIVE, NORMAL_PASSIVE	READY
14	Clock Correction Failed counter reached Maximum Without Clock Correction Fatal limit configured by WCF in SUC Configuration Register 3 AND bit HCSE in the SUC Configuration Register 1 set to 1 OR command HALT, SUCC1.CMD = 0110_B	NORMAL_ACTIVE	HALT
15	Clock Correction Failed counter reached Maximum Without Clock Correction Fatal limit configured by WCF in SUC Configuration Register 3 AND bit HCSE in the SUC Configuration Register 1 set to 1 OR command HALT, SUCC1.CMD = 0110_B	NORMAL_PASSIVE	HALT

Table 390 State Transitions of E-Ray Overall State Machine (cont'd)

T#	Condition	From	To
16	Command FREEZE, SUCC1.CMD = 0111_B	All States	HALT
17	Command CONFIG, SUCC1.CMD = 0001_B	HALT	DEFAULT_CONFIG

41.3.6.2 DEFAULT_CONFIG State

In "DEFAULT_CONFIG" state, the Communication Controller is stopped. All configuration registers are accessible and the pins to the physical layer are in their inactive state.

The Communication Controller enters this state

- When leaving application reset
- When exiting from "HALT" state

To leave "DEFAULT_CONFIG" state the Host has to write SUCC1.CMD = 0001_B in the SUC Configuration Register 1. The Communication Controller transits to "CONFIG" state.

41.3.6.2.1 CONFIG State

In "CONFIG" state, the Communication Controller is stopped. All configuration registers are accessible and the pins to the physical layer are in their inactive state. This state is used to initialize the Communication Controller configuration.

The Communication Controller enters this state

- When exiting from "DEFAULT_CONFIG" state
- When exiting from "MONITOR_MODE" or "READY" state

When the state has been entered via "HALT" and "DEFAULT_CONFIG" state, the Host can analyze status information and configuration. Before leaving "CONFIG" state the Host has to assure that the configuration is fault-free.

To leave "CONFIG" state, the Host has to perform the unlock sequence as described on "LCK". Directly after unlocking the "CONFIG" state the Host has to write SUCC1.CMD in the SUC Configuration Register 1 to enter the next state.

Internal counters and the Communication Controller status flags are reset when the Communication Controller leaves "CONFIG".

Note: *The Message Buffer Status Registers (MHDS, TXRQ1 to TXRQ4, NDAT1 to NDAT4, MBSC1 to MBSC4) and status data stored in the Message RAM and are not affected by the transition of the POC from "CONFIG" to "READY" state.*

When the Communication Controller is in "CONFIG" state it is also possible to bring the Communication Controller into a power saving mode by halting the module clocks (f_{SCLK}, f_{CLC_ERAY}). To do this the Host has to assure that all Message RAM transfers have finished before turning off the clocks.

41.3.6.3 MONITOR_MODE

After unlocking "CONFIG" state and writing SUCC1.CMD = 1011_B the Communication Controller enters "MONITOR_MODE". In this mode the Communication Controller is able to receive FlexRay™ Frames and to detect wakeup pattern. The temporal integrity of received Frames is not checked, and therefore cycle counter filtering is not supported. It is not possible to distinguish between static and dynamic frames, because limited functions in Monitor Mode (FRF.RSS will be ignored, filtering not functional).This mode can be used for debugging purposes in case e.g. that startup of a FlexRay™ network fails. After writing SUCC1.CMD = 0001_B the Communication Controller transits back to “CONFIG” state.

In MONITOR_MODE the pick first valid mechanism is disabled. This means that a receive Message Buffer may only be configured to receive on one channel. Received Frames are stored into Message Buffers according to Frame ID and receive channel. NULL Frames are handled like Data Frames. After Frame reception only status bits MBS.VFRA, MBS, MBS.MLST, MBS.RCIS, MBS.SFIS, MBS.SYNS, MBS.NFIS, MBS.PPIS, MBS.RESS have valid value.

In “MONITOR_MODE” the Communication Controller is not able to distinguish between CAS and MTS symbols. In case one of these symbols is received on one or both of the two channels, the flags SIR.MTSA resp. SIR.MTSB are set. SIR.CAS has no function in “MONITOR_MODE”.

41.3.6.4 READY State

After unlocking “CONFIG” state and writing SUCC1.CMD = 0010_B the Communication Controller enters “READY” state. From this state the Communication Controller can transit to WAKEUP state and perform a cluster wakeup or to “STARTUP” state to perform a coldstart or to integrate into a running communication.

The Communication Controller enters this state

- When exiting from “CONFIG”, “WAKEUP”, “STARTUP”, “NORMAL_ACTIVE”, or “NORMAL_PASSIVE” state by writing SUCC1.CMD = 0010_B (READY command).

The Communication Controller exits from this state

- To “CONFIG” state by writing SUCC1.CMD = 0001_B (CONFIG command)
- To “WAKEUP” state by writing SUCC1.CMD = 0011_B (WAKEUP command)
- To “STARTUP” state by writing SUCC1.CMD = 0100_B (RUN command)

Internal counters and the Communication Controller status flags are reset when the Communication Controller enters “STARTUP” state.

Note: Status bits MHDS, registers TXRQ1 to TXRQ4, and status data stored in the Message RAM are not affected by the transition of the POC from “READY” to “STARTUP” state.

41.3.6.5 WAKEUP State

The description below is intended to help configuring wakeup for the E-Ray IP-module. A detailed description of the wakeup procedure together with the respective SDL diagrams can be found in the FlexRay™ protocol specification v2.1, section 7.1.

The Communication Controller enters this state

- When exiting from “READY” state by writing SUCC1.CMD = 0011_B (WAKEUP command).

The Communication Controller exits from this state to “READY” state

- After complete non-aborted transmission of wakeup pattern
- After WUP reception
- After detecting a WUP collision
- After reception of a Frame Header
- By writing SUCC1.CMD = 0010_B (READY command)

The cluster wakeup must precede the communication startup in order to ensure that all mechanisms defined for the startup work properly. The minimum requirement for a cluster wakeup is that all bus drivers are supplied with power. A bus driver has the ability to wake up the other components of its node when it receives a wakeup pattern on its channel. At least one node in the cluster needs an **external** wakeup source.

The Host completely controls the wakeup procedure. It is informed about the state of the cluster by the bus driver and the Communication Controller and configures bus guardian (if available) and Communication Controller to perform the cluster wakeup. The Communication Controller provides to the Host the ability to transmit a special wakeup pattern on each of its available channels separately. The Communication Controller needs to recognize the wakeup pattern only during "WAKEUP" state.

Wakeup may be performed on only one channel at a time. The Host has to configure the wakeup channel while the Communication Controller is in "CONFIG" state by writing bit SUCC1.WUCS in the SUC Configuration Register 1. The Communication Controller ensures that ongoing communication on this channel is not disturbed. The Communication Controller cannot guarantee that all nodes connected to the configured channel awake upon the transmission of the wakeup pattern, since these nodes cannot give feedback until the startup phase. The wakeup procedure enables single-channel devices in a two-channel system to trigger the wakeup, by only transmitting the wakeup pattern on the single channel to which they are connected. Any coldstart node that deems a system startup necessary will then wake the remaining channel before initiating communication startup.

The wakeup procedure tolerates any number of nodes simultaneously trying to wakeup a single channel and resolves this situation such that only one node transmits the pattern. Additionally the wakeup pattern is collision resilient, so even in the presence of a fault causing two nodes to simultaneously transmit a wakeup pattern, the resulting collided signal can still wake the other nodes.

After wakeup the Communication Controller returns to "READY" state and signals the change of the wakeup status to the Host by setting bit SIR.WST in the Status Service Request Register. The wakeup status vector can be read from the Communication Controller Status Vector register CCSV.WSV. If a valid wakeup pattern was received also either flag SIR.WUPA or flag SIR.WUPB in the Status Service Request Register is set.

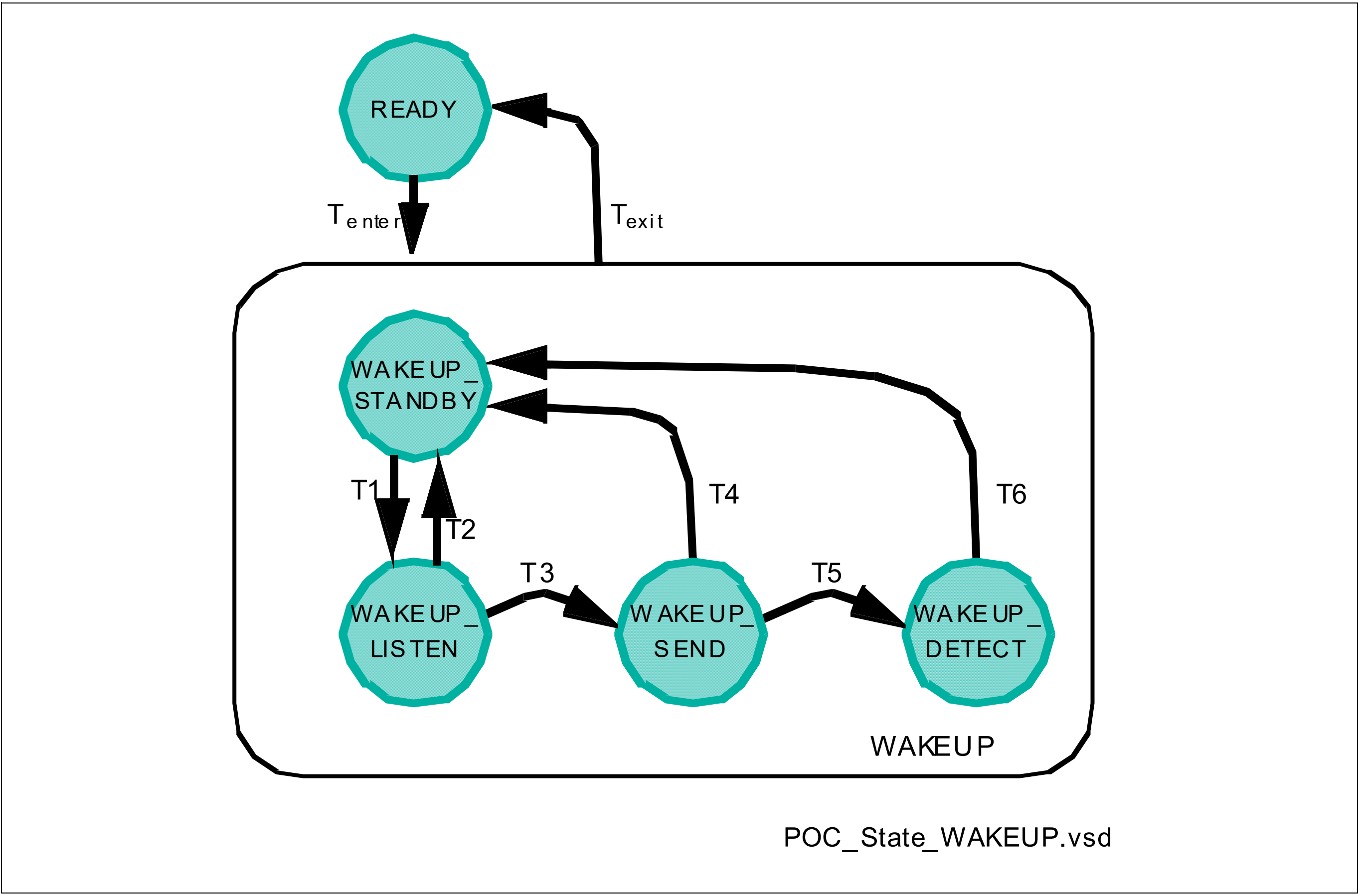

Figure 611 Structure of POC State WAKEUP

Table 391 State Transitions WAKEUP

T#	Condition	From	To
enter	Host commands change to “WAKEUP” state by writing SUCC1.CMD = 0011_B (WAKEUP command)	READY	WAKEUP
1	CHI command WAKEUP triggers wakeup FSM to transit to “WAKEUP_LISTEN” state	WAKEUP_ STANDBY	WAKEUP_LISTE N
2	Received WUP on wakeup channel selected by flag SUCC1.WUCS in the SUC Configuration Register 1 OR Frame Header on either available channel	WAKEUP_ LISTEN	WAKEUP_STAN DBY
3	Timer event	WAKEUP_ LISTEN	WAKEUP_SEND
4	Complete, non-aborted transmission of wakeup pattern	WAKEUP_ SEND	WAKEUP_STAN DBY
5	Collision detected	WAKEUP_ SEND	WAKEUP_DETE CT
6	Wakeup timer expired OR WUP detected on wakeup channel selected by flag SUCC1.WUCS in the SUC Configuration Register 1 OR Frame Header received on either available channel	WAKEUP_ DETECT	WAKEUP_STAN DBY
exit	Wakeup completed (after T2 or T4 or T6) OR Host commands change to “READY” state by writing SUCC1.CMD = 0010_B (READY command). This command also resets the wakeup FSM to “WAKEUP_STANDBY” state	WAKEUP	READY

The “WAKEUP_LISTEN” state is controlled by the wakeup timer and the wakeup noise timer. The two timers are controlled by the parameters listen time-out SUCC2.LT and listen time-out noise SUCC2.LTN. Both values can be configured in the SUC Configuration Register 2. listen time-out enables a fast cluster wakeup in case of a noise free environment, while listen time-out noise enables wakeup under more difficult conditions regarding noise interference.

In “WAKEUP_SEND” state the Communication Controller transmits the wakeup pattern on the configured channel and checks for collisions. After return from wakeup the Host has to bring the Communication Controller into “STARTUP” state by CHI command RUN.

In “WAKEUP_DETECT” state the Communication Controller attempts to identify the reason for the wakeup collision detected in “WAKEUP_SEND” state. The monitoring is bounded by the expiration of listen time-out as configured by SUCC2.LT in the SUC Configuration Register 2. Either the detection of a wakeup pattern indicating a wakeup attempt by another node or the reception of a Frame Header indication existing communication, causes the direct transition to “READY” state. Otherwise WAKEUP_DETECT is left after expiration of listen time-out; in this case the reason for wakeup collision is unknown.

The Host has to be aware of possible failures of the wakeup and act accordingly. It is advisable to delay any potential startup attempt of the node having instigated the wakeup by the minimal time it takes another coldstart node to become awake and to be configured.

The FlexRay™ Protocol Specification v2.1 recommends that two different Communication Controllers shall awake the two channels.

Host activities

The host must coordinate the wakeup of the two channels and must decide whether, or not, to wake a specific channel. The sending of the wakeup pattern is initiated by the Host and generated by the Communication Controller. The wakeup pattern is detected by the remote BDs and signalled to their local Hosts.

Wakeup procedure controlled by Host (single-channel wakeup):

- Configure the Communication Controller in "CONFIG" state
  - Select wakeup channel by programming bit SUCC1.WUCS
- Check local BDs whether a WUP was received
- Activate BD of selected wakeup channel
- Command Communication Controller to start wakeup on the configured channel by writing SUCC1.CMD = 0011_B
  - Communication Controller enters "WAKEUP
  - Communication Controller returns to "READY" state and signals status of wakeup attempt to Host
- Wait predefined time to allow the other nodes to wakeup and configure themselves
- Coldstart node: wait for WUP on the other channel
  - In a dual channel cluster wait for WUP on the other channel
  - Reset coldstart inhibit flag CCSV.CSI by writing SUCC1.CMD = 1001_B (ALLOW_COLDSTART command))
- Reset Coldstart Inhibit flag CCSV.CSI in the CCSV register by writing SUCC1.CMD = 1001_B (ALLOW_COLDSTART command), coldstart node only
- Command Communication Controller to enter startup by writing SUCC1.CMD = 0100_B (RUN command)

Wakeup procedure triggered by BD:

- Wakeup recognized by BD
- BD triggers power-up of Host (if required)
- BD signals wakeup event to Host
- Host configures its local Communication Controller
- If necessary Host commands wakeup of second channel and waits predefined time to allow the other nodes to wakeup and configure themselves
- Host commands Communication Controller to enter "STARTUP" state by writing SUCC1.CMD = 0100_B (RUN command)

Wakeup pattern (WUP)

The wakeup pattern is composed of at least two wakeup symbols (WUS). Wakeup symbol and wakeup pattern are configured by the PRT Configuration Registers PRTC1 and PRTC2.

- Single channel wakeup, wakeup symbol may not be sent on both channels at the same time
- Wakeup symbol collision resilient for up to two sending nodes (two overlapping wakeup symbols still recognizable)
- Wakeup symbol must be configured identical in all nodes of a cluster
- Wakeup symbol transmit low time configured by PRTC2.TXL
- Wakeup symbol idle time used to listen for activity on the bus, configured by PRTC2.TXI
- A wakeup pattern composed of at least two Tx-wakeup symbols needed for wakeup
- Number of repetitions configurable by PRTC1.RWP (2 to 63 repetitions)
- Wakeup symbol receive window length configured by PRTC1.RXW
- Wakeup symbol receive low time configured by PRTC2.RXL
- Wakeup symbol receive idle time configured by PRTC2.RXI

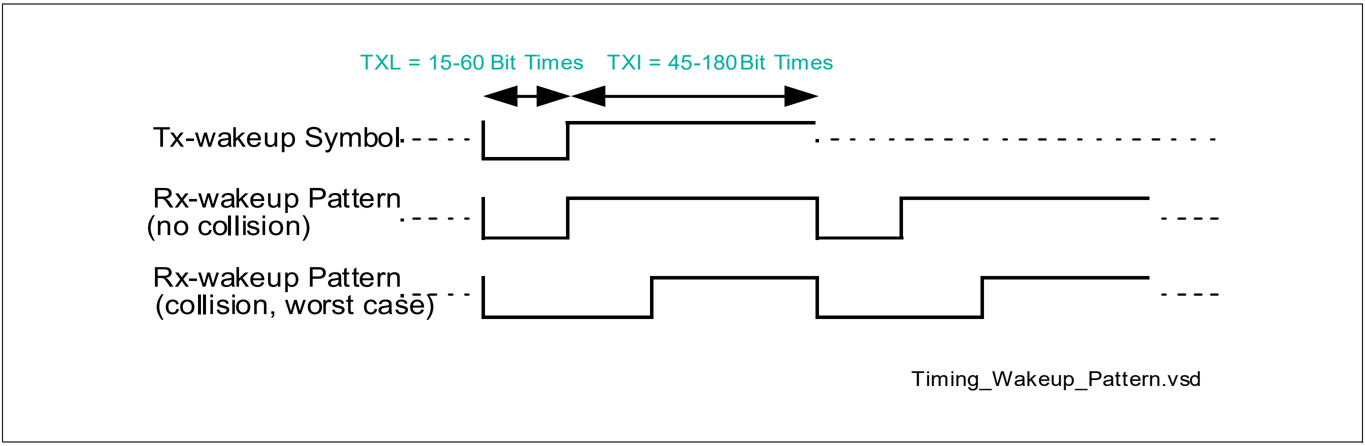
TXL = 15-60 Bit Times
TXI = 45-180 Bit Times
Tx-wakeup Symbol
Rx-wakeup Pattern (no collision)
Rx-wakeup Pattern (collision, worst case)
Timing_Wakeup_Pattern.vsd

Figure 612 Timing of Wakeup Pattern

41.3.6.6 STARTUP State

The description below is intended to help configuring startup for the E-Ray IP-module. A detailed description of the startup procedure together with the respective SDL diagrams can be found in the FlexRay™ protocol specification v2.1, section 7.2.

Any node entering "STARTUP" state that has coldstart capability should assure that both channels attached have been awakened before initiating coldstart.

It cannot be assumed that all nodes and stars need the same amount of time to become completely awake and to be configured. Since at least two nodes are necessary to start up the cluster communication, it is advisable to delay any potential startup attempt of the node having instigated the wakeup by the minimal amount of time it takes another coldstart node to become awake, to be configured and to enter startup. It may require several hundred milliseconds (depending on the hardware used) before all nodes and stars are completely awakened and configured.

Startup is performed on all channels synchronously. During startup, a node only transmits startup Frames.

A fault-tolerant, distributed startup strategy is specified for initial synchronization of all nodes. In general, a node may enter "NORMAL_ACTIVE" state via (see **Figure 613**):

- Coldstart path initiating the schedule synchronization (leading coldstart node)
- Coldstart path joining other coldstart nodes (following coldstart node)
- Integration path integrating into an existing communication schedule (all other nodes)

A coldstart attempt begins with the transmission of a collision avoidance symbol (CAS). Only a coldstart node that had transmitted the CAS transmits Frames in the first four cycles after the CAS, it is then joined firstly by the other coldstart nodes and afterwards by all other nodes.

A coldstart node has the Transmit Startup Frame in Key Slot bit SUCC1.TXST and Transmit SYNC Frame in Key Slot bit SUCC1.TXSY in the SUC Configuration Register 1 set to 1. The Message Buffer 0 holds the key slot ID which defines the slot number where the Startup Frame is send. In the Frame Header of the Startup Frame the Startup Frame indicator bit is set.

In clusters consisting of three or more nodes, at least three nodes shall be configured to be coldstart nodes. In clusters consisting of two nodes, both nodes must be coldstart nodes. At least two fault-free coldstart nodes are necessary for the cluster to startup.

Each Startup Frame must also be a SYNC Frame; therefore each coldstart node will also be a sync node. The number of coldstart attempts is configured by SUCC1.CSA in the SUC Configuration Register 1.

A non-coldstart node requires at least two startup Frames from distinct nodes for integration. It may start integration before the coldstart nodes have finished their startup. It will not finish its startup until at least two coldstart nodes have finished their startup.

Both non-coldstart nodes and coldstart nodes start passive integration via the integration path as soon as they receive SYNC Frames from which to derive the TDMA schedule information. During integration the node has to adapt its own clock to the global clock (rate and offset) and has to make its cycle time consistent with the global schedule observable at the network. Afterwards, these settings are checked for consistency with all available network nodes. The node can only leave the integration phase and actively participate in communication when these checks are passed.

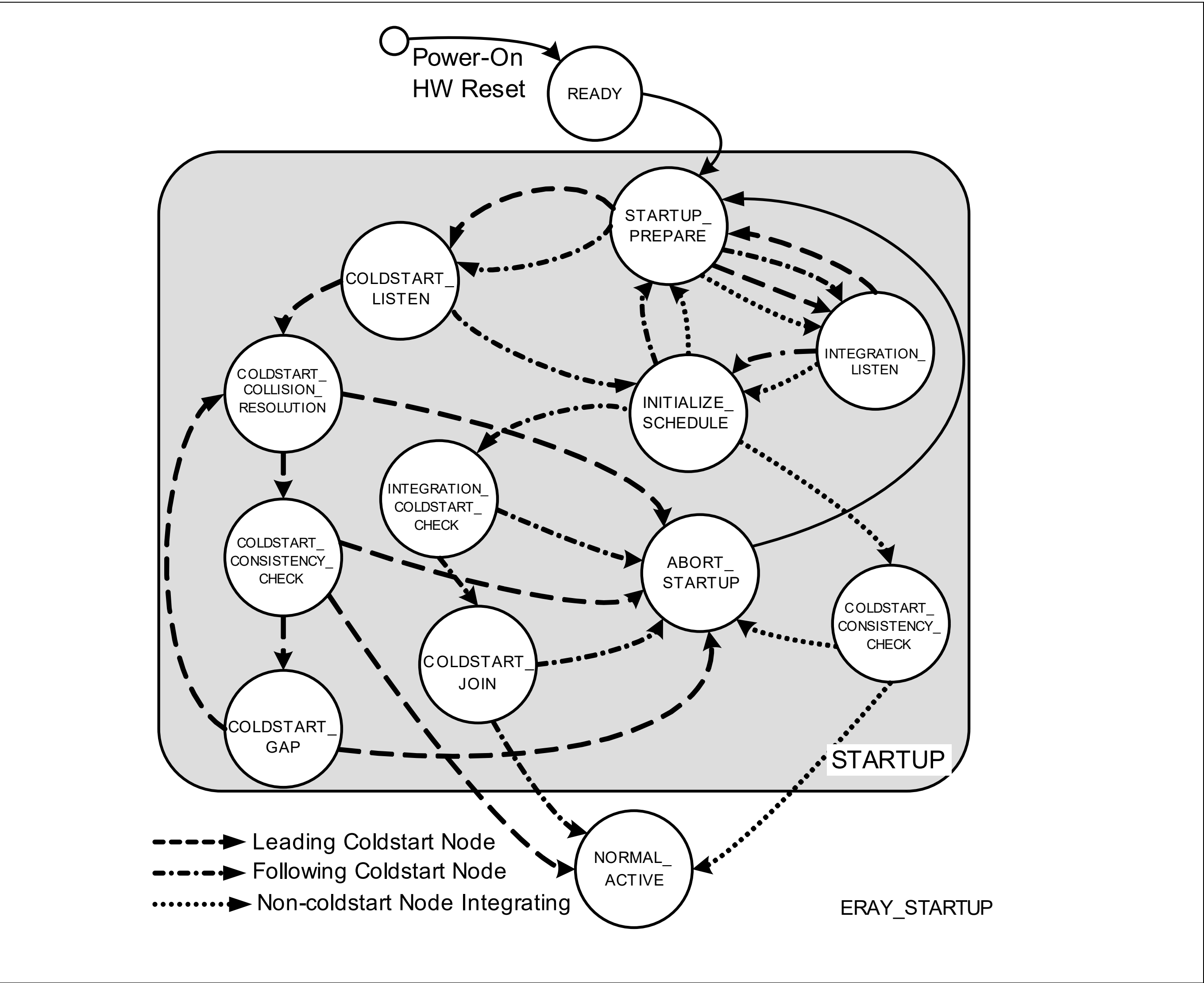

Figure 613 State Diagram Time-Triggered Startup

Coldstart Inhibit Mode

In coldstart inhibit mode the node is prevented from initializing the TDMA communication schedule. If bit CCSV.CSI in the Communication Controller Status Vector register is set, the node is not allowed to initialize the cluster communication, i.e. entering the coldstart path is prohibited. The node is allowed to integrate to a running cluster or to transmit startup Frames after another coldstart node started the initialization of the cluster communication.

The coldstart inhibit bit CCSV.CSI is set whenever the POC enters “READY” state. The bit has to be cleared under control of the Host by CHI command ALLOW_COLDSTART (SUCC1.CMD = 1001_B)

41.3.6.7 Startup Time-outs

The Communication Controller supplies two different Microtick timers supporting two time-out values, startup time-out and startup noise time-out. The two timers are reset when the Communication Controller enters the “COLDSTART_LISTEN” state. The expiration of either of these timers causes the node to leave the initial sensing phase (“COLDSTART_LISTEN” state) with the intention of starting up communication.

Note: The startup and startup noise timers are identical with the wakeup and wakeup noise timers and use the same configuration values SUCC2.LT and SUCC2.LTN from the SUC Configuration Register 2.

Startup Time-out

The startup time-out limits the listen time used by a node to determine if there is already communication between other nodes or at least one coldstart node actively requesting the integration of others.

The startup timer is configured by programming SUCC2.LT (pdListenTimeout) in the SUC Configuration Register 2.

The startup timer is restarted upon:

- Entering the “COLDSTART_LISTEN” state
- Both channels reaching idle state while in “COLDSTART_LISTEN” state

The startup timer is stopped:

- If communication channel activity is detected on one of the configured channels while the node is in the “COLDSTART_LISTEN” state
- When the “COLDSTART_LISTEN” state is left

Once the startup time-out expires, neither an overflow nor a cyclic restart of the timer is performed. The timer status is kept for further processing by the startup state machine.

Startup Noise Time-out

At the same time the startup timer is started for the first time (transition from “STARTUP_PREPARE” state to “COLDSTART_LISTEN” state), the startup noise timer is started. This additional time-out is used to improve reliability of the startup procedure in the presence of noise.

The startup noise timer is configured by programming SUCC2.LTN (gListenNoise - 1) in the SUC Configuration Register 2 (see “**SUCC2**).

The startup noise time-out is:
pdListenTimeout • ´gListenNoise = SUCC2.LT • (SUCC2.LTN + 1)

The startup noise timer is restarted upon:

- Entering the “COLDSTART_LISTEN” state
- Reception of correctly decoded Headers or CAS symbols while the node is in “COLDSTART_LISTEN” state

The startup noise timer is stopped when the “COLDSTART_LISTEN” state is left.

Once the startup noise time-out expires, neither an overflow nor a cyclic restart of the timer is performed. The status is kept for further processing by the startup state machine. Since the startup noise timer won’t be restarted when random channel activity is sensed, this time-out defines the fall-back solution that guarantees that a node will try to start up the communication cluster even in the presence of noise.

41.3.6.8 Path of leading Coldstart Node (initiating coldstart)

When a coldstart node enters “COLDSTART_LISTEN”, it listens to its attached channels.

If no communication is detected, the node enters the “COLDSTART_COLLISION_ RESOLUTION” state and commences a coldstart attempt. The initial transmission of a CAS symbol is succeeded by the first regular cycle. This cycle has the number zero.

From cycle zero on, the node transmits its startup Frame. Since each coldstart node is allowed to perform a coldstart attempt, it may occur that several nodes simultaneously transmit the CAS symbol and enter the coldstart path. This situation is resolved during the first four cycles after CAS transmission.

As soon as a node that initiates a coldstart attempt receives a CAS symbol or a Frame Header during these four cycles, it re-enters the “COLDSTART_LISTEN” state. Thereby, only one node remains in this path. In cycle four, other coldstart nodes begin to transmit their startup Frames.

After four cycles in “COLDSTART_COLLISION_RESOLUTION” state, the node that initiated the coldstart enters the “COLDSTART_CONSISTENCY_CHECK” state. It collects all startup Frames from cycle four and five and performs the clock correction. If the clock correction does not deliver any errors and it has received at least one valid Startup Frame pair, the node leaves “COLDSTART_CONSISTENCY_CHECK” and enters “NORMAL_ACTIVE” state.

The number of coldstart attempts that a node is allowed to perform is configured by SUCC1.CSA in the SUC Configuration Register 1. The number of remaining coldstarts attempts CCSV.RCA can be read from Communication Controller Status Vector register. The number of remaining attempts is reduced by one for each attempted coldstart. A node may enter the “COLDSTART_LISTEN” state only if this value is larger than one and it may enter the “COLDSTART_COLLISION_RESOLUTION” state only if this value is larger than zero. If the number of coldstart attempts is one, coldstart is inhibited but integration is still possible.

Path of following Coldstart Node (responding to leading Coldstart Node)

When a coldstart node enters the “COLDSTART_LISTEN” state, it tries to receive a valid pair of startup Frames to derive its schedule and clock correction from the leading coldstart node.

As soon as a valid Startup Frame has been received the “INITIALIZE_SCHEDULE” state is entered. If the clock synchronization can successfully receive a matching second valid Startup Frame and can derive a schedule from this startup Frames, the “INTEGRATION_COLDSTART_CHECK” state is entered.

In “INTEGRATION_COLDSTART_CHECK” state it is assured that the clock correction can be performed correctly and that the coldstart node from which this node has initialized its schedule is still available. The node collects all SYNC Frames and performs clock correction in the following double-cycle. If clock correction does not signal any errors and if the node continues to receive sufficient Frames from the same node it has integrated on, the “COLDSTART_JOIN” state is entered.

In “COLDSTART_JOIN” state integrating coldstart nodes begin to transmit their own startup Frames. Thereby the node that initiated the coldstart and the nodes joining it can check if their schedules agree to each other. If for the following three cycles the clock correction does not signal errors and at least one other coldstart node is visible, the node leaves “COLDSTART_JOIN” state and enters “NORMAL_ACTIVE” state. Thereby it leaves “STARTUP” at least one cycle after the node that initiated the coldstart.

Path of Non-coldstart Node

When a non-coldstart node enters the INTEGRATION_LISTEN state, it listens to its attached channels and tries to receive FlexRay™ Frames.

As soon as a valid Startup Frame has been received the “INITIALIZE_SCHEDULE” state is entered. If the clock synchronization can successfully receive a matching second valid Startup Frame and derive a schedule from this, the INTEGRATION_CONSISTENCY_ CHECK state is entered.

In “INTEGRATION_CONSISTENCY_CHECK” state it is verified that the clock correction can be performed correctly and that enough coldstart nodes (at least 2) send startup Frames that agree to the nodes own schedule. Clock correction is activated, and if any errors are signalled, the integration attempt is aborted.

During the first even cycle in this state, either two valid startup Frames or the Startup Frame of the node that this node has integrated on must be received; otherwise the node aborts the integration attempt.

During the first double-cycle in this state, either two valid Startup Frame pairs or the Startup Frame pair of the node that this node has integrated on must be received; otherwise the node aborts the integration attempt.

If after the first double-cycle less than two valid startup Frames are received within an even cycle, or less than two valid Startup Frame pairs are received within a double-cycle, the startup attempt is aborted.

Nodes in this state need to see two valid Startup Frame pairs for two consecutive double-cycles each to be allowed to leave STARTUP and enter NORMAL_ACTIVE. Consequently, they leave startup at least one double-cycle after the node that initiated the coldstart and only at the end of a cycle with an odd cycle number.

41.3.6.9 NORMAL_ACTIVE State

As soon as the node that transmitted the first CAS symbol (resolving the potential access conflict and entering “STARTUP” via coldstart path) and one additional node have entered the “NORMAL_ACTIVE” state, the startup phase for the cluster has finished. In the “NORMAL_ACTIVE” state, all configured messages are scheduled for transmission. This includes all Data Frames as well as the SYNC Frames. Rate and offset measurement is started in all even cycles (even/odd cycle pairs required).

In “NORMAL_ACTIVE” state the Communication Controller supports regular communication functions

- The Communication Controller performs transmissions and reception on the FlexRay™ bus as configured
- Clock synchronization is running
- The Host interface is operational

The Communication Controller exits from that state to

- “HALT” state by writing SUCC1.CMD = 0110_B (HALT command, at the end of the current cycle)
- “HALT” state by writing SUCC1.CMD = 0111_B (FREEZE command, immediately)
- “HALT” state due to change of the error state from “ACTIVE” to “COMM_HALT”
- “NORMAL_PASSIVE” state due to change of the error state from “ACTIVE” to “PASSIVE”
- “READY” state by writing SUCC1.CMD = 0010_B (READY command)

41.3.6.10 NORMAL_PASSIVE State

“NORMAL_PASSIVE” state is entered from “NORMAL_ACTIVE” state when the error state changes from ACTIVE (green) to PASSIVE (yellow).

In “NORMAL_PASSIVE” state, the node is able to receive all Frames (node is fully synchronized and performs clock synchronization). In comparison to the “NORMAL_ACTIVE” state the node does not actively participate in communication, i.e. neither symbols nor Frames are transmitted.

In “NORMAL_PASSIVE” state

- The Communication Controller performs reception on the FlexRay™ bus
- The Communication Controller does not transmit any Frames or symbols on the FlexRay™ bus
- Clock synchronization is running
- The Host interface is operational

The Communication Controller exits from this state to

- “HALT” state by writing SUCC1.CMD = 0110_B (HALT command, at the end of the current cycle)
- “HALT” state by writing SUCC1.CMD = 0111_B (FREEZE command, immediately)

- “HALT” state due to change of the error state from “PASSIVE” to “COMM_HALT”
- “NORMAL_ACTIVE” state due to change of the error state from “PASSIVE” to “ACTIVE”. The transition takes place when CCEV.PTAC from the Communication Controller Error Vector register equals SUCC1.PTA - 1.
- “READY” state by writing SUCC1.CMD = 0010_B (READY command)

41.3.6.11 HALT State

In this state all communication (reception and transmission) is stopped.

The Communication Controller enters this state

- By writing SUCC1.CMD = 0110_B (HALT command) while the Communication Controller is in “NORMAL_ACTIVE” or “NORMAL_PASSIVE” state
- By writing SUCC1.CMD = 0111_B (FREEZE command) from all states
- When exiting from “NORMAL_ACTIVE” state because the clock correction failed counter reached the “maximum without clock correction fatal” limit
- When exiting from “NORMAL_PASSIVE” state because the clock correction failed counter reached the “maximum without clock correction fatal” limit

The Communication Controller exits from this state to “DEFAULT_CONFIG” state

- By writing SUCC1.CMD = 0001_B (CONFIG command)

When the Communication Controller enters “HALT” state, all configuration and status data is maintained for analyzing purposes.

When the Host writes SUCC1.CMD = 0110_B (HALT command), the Communication Controller sets bit CCSV.HRQ in the Communication Controller Status Vector register and enters “HALT” state after the current communication cycle has finished.

When the Host writes SUCC1.CMD = 0111_B (FREEZE command), the Communication Controller enters “HALT” state immediately and sets the CCSV.FSI bit in the Communication Controller Status Vector register.

The POC state from which the transition to HALT state took place can be read from CCSV.PSL.

41.3.7 Network Management

The accrued Network Management (NM) vector is located in the Network Management Register 1 to Network Management Register 3 (NMVx (x = 1-3)). The Communication Controller performs a logical OR operation over all Network Management (NM) vectors out of all received valid Network Management (NM) Frames with the Payload Preamble Indicator (PPI) bit set. Only a static Frame may be configured to hold Network Management (NM) information. The Communication Controller updates the Network Management (NM) vector at the end of each cycle.

The length of the Network Management (NM) vector can be configured from 0 to 12 byte by NML in the NEM Configuration Register. The Network Management (NM) vector length must be configured identically in all nodes of a cluster.

To configure a transmit buffer to send FlexRay™ Frames with the PPI bit set, the PPIT bit in the Header Section of the respective transmit buffer has to be set via WRHS1.PPIT. In addition the Host has to write the Network Management (NM) information to the Data Section of the respective transmit buffer.

The evaluation of the Network Management (NM) vector has to be done by the application running on the Host.

Note: *In case a Message Buffer is configured for transmission / reception of Network Management Frames, the payload length configured in Header 2 of that Message Buffer should be equal or greater than the length of the NM Vector configured by NEMC.NML.*
When the Communication Controller transits to “HALT” state, the cycle count is not incremented and therefore the NM Vector is not updated. In this case NMV1 to NMV3 holds the value from the cycle before.

41.3.8 Filtering and Masking

Filtering is done by checking specific fields in a received Frame against the corresponding configuration constants of the valid Message Buffers and the actual slot and cycle counter values (acceptance filtering), or by comparing the configuration constants of the valid Message Buffers against the actual slot and cycle counter values (transmit filtering). A Message Buffer is only updated / transmitted if the required matches occur.

Filtering is done on the following fields:

- Channel ID
- Frame ID
- Cycle Counter

The following filter combinations for acceptance / transmit filtering are allowed:

- Frame ID + Channel ID
- Frame ID + Channel ID + Cycle Counter

In order to store a received message in a Message Buffer all configured filters must match.

Note: *For the FIFO the acceptance filter is configured by the FIFO Rejection Filter and the FIFO Rejection Filter Mask.*

A message will be transmitted in the time slot corresponding to the configured Frame ID on the configured channel(s). If cycle counter filtering is enabled the configured cycle filter value must also match.

41.3.8.1 Frame ID Filtering

Every transmit and receive buffer contains a Frame ID stored in the Header Section. This Frame ID is used differently for receive and transmit buffers.

Receive Buffers

A received message is stored in the first receive buffer where the received Frame ID matches the configured Frame ID, provided channel ID and cycle counter criteria are also met.

Transmit Buffers

For transmit buffers the configured Frame ID is used to determine the appropriate slot for message transmission. The Frame will be transmitted in the time slot corresponding to the configured Frame ID, provided channel ID and cycle counter criteria are also met.

41.3.8.2 Channel ID Filtering

There is a 2-bit channel filtering field (CHA, CHB) located in the Header Section of each Message Buffer in the Message RAM. It serves as a filter for receive buffers, and as a control field for transmit buffers (see Table below).

Table 392 Channel Filtering Configuration

CHA	**CHB**	**Transmit Buffer**transmit Frame	**Receive Buffer**store valid receive Frame
1	1	on both channels (static segment only)	received on channel A or B (store first semantically valid Frame, static segment only)
1	0	on channel A	received on channel A
0	1	on channel B	received on channel B
0	0	no transmission	ignore Frame

Note: *If a Message Buffer is configured for the dynamic segment and both bits of the channel filtering field are set to 1, no Frames are transmitted resp. received Frames are ignored (same function as CHA = CHB = 0)*

Receive Buffers

Valid received Frames are stored if they are received on the channels specified in the channel filtering field. Only in static segment a receive buffer may be setup for reception on both channels (CHA and CHB set). Other filtering criteria must also be met.

If a valid Header Segment was stored, the respective MBC flag in the Message Buffer Status Changed register is set. If a valid Payload Segment was stored, the respective NDn (n = 0-31) to NDn (n = 96-127) flag in the New Data NDAT1 to NDAT4 register is set. In both cases, if bit RDHS1.MBI in the Header Section of the respective Message Buffer is set, the RXI flag in the Status Service Request Register is set to 1. If enabled an service request is generated.

Transmit Buffers

The content of the buffer is transmitted only on the channels specified in the channel filtering field when the Frame ID filtering and cycle counter filtering criteria are also met. Only in static segment a transmit buffer may be setup for transmission on both channels (CHA and CHB set). After transmission has completed, and if bit WRHS1.MBI in the Header Section of the respective Message Buffer is set, the TXI flag in the Status Service Request Register is set to 1. If enabled an service request is generated.

41.3.8.3 Cycle Counter Filtering

Cycle counter filtering is based on the notion of a cycle set. For filtering purposes, a match is detected if any one of the elements of the cycle set is matched. The cycle set is defined by the cycle code field in the Header Section of each Message Buffer.

If Message Buffer 0 is configured to hold the startup / SYNC Frame or the single slot Frame by bits TXST, TXSY, and TSM in the SUC Configuration Register 1, cycle counter filtering for Message Buffer 0 should be disabled.

Note: Sharing of a static time slot via cycle counter filtering between different nodes of a FlexRay™ network is ***not*** *allowed.*

The set of cycle numbers belonging to a cycle set is determined as described in Table below.

Table 393 Definition of Cycle Set

Cycle Code	Matching Cycle Counter Values		
$000000x_B$	all Cycles		
$000001c_B$	every second Cycle	at (Cycle Count)mod2	= c
$00001cc_B$	every fourth Cycle	at (Cycle Count)mod4	= cc
$0001ccc_B$	every eighth Cycle	at (Cycle Count)mod8	= ccc
$001cccc_B$	every sixteenth Cycle	at (Cycle Count)mod16	= cccc
$01ccccc_B$	every thirty-second Cycle	at (Cycle Count)mod32	= ccccc
$1cccccc_B$	every sixty-fourth Cycle	at (Cycle Count)mod64	= cccccc

Table below gives some examples for valid cycle sets to be used for cycle counter filtering:

Table 394 Examples for Valid Cycle Sets

Cycle Code	Matching Cycle Counter Values
0000011_B	1-3-5-7- ….-63 ↲
0000100_B	0-4-8-12- ….-60 ↲
0001110_B	6-14-22-30- ….-62 ↲
0011000_B	8-24-40-56 ↲
0100011_B	3-35 ↲
1001001_B	9 ↲

Receive Buffers

The received message is stored only if the received cycle counter matches an element of the receive buffer's cycle set. Channel ID and Frame ID criteria must also be met.

Transmit Buffers

The content of the buffer is transmitted on the configured channels when an element of the cycle set matches the current cycle counter value and the Frame ID matches the slot counter value.

41.3.8.4 FIFO Filtering

For FIFO filtering there is one rejection filter and one rejection filter mask available. The FIFO rejection filter consists of 20 bits for **Channel** (2 bits), **Frame ID** (11 bits), and **Cycle Code** (7 bits). Rejection filter and rejection filter mask can be configured in "DEFAULT_CONFIG" or "CONFIG" state only. The filter configuration in the Header Sections of Message Buffers belonging to the FIFO is ignored.

A valid received Frame is stored in the FIFO if channel ID, Frame ID, and cycle counter are not rejected by the configured rejection filter and rejection filter mask, and if there is no matching dedicated receive buffer.

41.3.9 Transmit Process

The transmit process is described in the following sections.

41.3.9.1 Static Segment

For the static segment, if there are several messages pending for transmission, the message with the Frame ID corresponding to the next sending slot is selected for transmission.

The Data Section of transmit buffers assigned to the static segment can be updated until the end of the preceding time slot. This means that a transfer from the Input Buffer has to be started by writing to the Input Buffer Command Request register latest at this time.

41.3.9.2 Dynamic Segment

In the dynamic segment, if several messages are pending, the message with the highest priority (lowest Frame ID) is selected next. Only Frame ID's which are higher than the largest static Frame ID are allowed for the dynamic segment.

In the dynamic segment different slot counter sequences are possible (concurrent sending of different Frame ID's on both channels). Therefore pending messages are selected according to their Frame ID and their channel configuration bit.

The Data Section of transmit buffers assigned to the dynamic segment can be updated until the end of the preceding slot. This means that a transfer from the Input Buffer has to be started by writing to the Input Buffer Command Request register latest at this time.

The start of latest transmit configured by SLT in the MHD Configuration Register 1 defines the maximum minislot value allowed before inhibiting new Frame transmission in the dynamic segment of the current cycle.

41.3.9.3 Transmit Buffers

A portion of the E-Ray Message Buffers can be configured as transmit buffers by programming bit CFG in the Header Section of the respective Message Buffer to 1. This can be done via the Write Header Section 1 register.

There exist the following possibilities to assign a transmit buffer to the Communication Controller channels:

- Static segment: channel A **or** channel B, channel A **and** channel B
- Dynamic segment: channel A **or** channel B

Message Buffer 0 is dedicated to hold the startup Frame, the SYNC Frame, or the designated single slot Frame as configured by TXST, TXSY, and TSM in the SUC Configuration Register 1. In this case it can be reconfigured in "DEFAULT_CONFIG" or "CONFIG" state only. This ensures that any node transmits at most one startup / SYNC Frame per communication cycle. Transmission of startup / SYNC Frames from other Message Buffers is not possible.

All other Message Buffers configured for transmission in static or dynamic segment are reconfigurable during runtime. Due to the organization of the Data Partition in the Message RAM (reference by data pointer), reconfiguration of the configured payload length and the data pointer in the Header Section of a Message Buffer may lead to erroneous configurations. If a Message Buffer is reconfigured during runtime it may happen that this Message Buffer is not send out in the respective communication cycle.

The Communication Controller does not have the capability to calculate the Header CRC. The Host is supposed to provide the Header CRCs for all transmit buffers. If Network Management is required the Host has to set the PPIT bit in the Header Section of the respective Message Buffer to 1 and write the Network Management information to the Data Section of the Message Buffer.

The payload length field configures the data payload length in 2-byte words. If the configured payload length of a static transmit buffer is shorter than the payload length configured for the static segment by SFDL in the Message Handler Configuration Register 1, the Communication Controller generates padding byte to ensure that Frames have proper physical length. The padding pattern is logical zero.

Each transmit buffer provides a transmission mode flag TXM that allows the Host to configure the transmission mode for the transmit buffer in the static segment. If this bit is set, the transmitter operates in the single-shot mode. If this bit is cleared, the transmitter operates in the continuous mode. In dynamic segment the transmitter always works in single-shot mode.

If a Message Buffer is configured in the continuous mode, the Communication Controller does not reset the transmission request flag TXR after successful transmission. In this case a Frame is sent out each time the Frame ID and cycle counter filter match. The TXR flag can be reset by the Host by writing the respective Message Buffer number to the Input Buffer Command Request register while bit STXRH in the Input Buffer Command Mask register is reset to 0.

If two or more transmit buffers are configured with the same Frame ID **and** cycle counter filter value, the transmit buffer with the lowest Message Buffer number will be transmitted in the respective slot.

41.3.9.4 Frame Transmission

To prepare a transmit buffer for transmission the following steps are required:

- Configure the Message Buffer as transmit buffer by writing bit CFG = 1 in the Write Header Section 1 register
- Write transmit message (Header and Data Section) to the Input Buffer.
- To transfer a transmit message from Input Buffer to the Message RAM proceed as described on "Data Transfer from Input Buffer to Message RAM".
- If configured in the Input Buffer Command Mask register the Transmission Request flag for the respective Message Buffer will be set as soon as the transfer has completed, and the Message Buffer is ready for transmission.
- Check whether the Message Buffer has been transmitted by checking the TXR bits (TXR = 0) in the Transmission Request 1 to 4 registers (single-shot mode only).

In single-shot mode the Communication Controller resets the TXR flag after transmission has been completed. Now the Host may update the transmit buffer with the next message. The Communication Controller does not transmit the message before the Host has indicated that the update is completed by setting the Transmission Request flag TXR again. The Host can check the actual state of the TXR flags of all Message Buffers by reading the Transmission Request registers. After successful transmission, if bit WRHS1.MBI in the Header Section of the respective Message Buffer is set, the transmit service request flag in the Status Service Request Register is set (TXI = 1). If enabled an service request is generated.

41.3.9.5 NULL Frame Transmission

If in static segment the Host does not set the transmission request flag before transmit time, and if there is no other transmit buffer with matching filter criteria (matching Frame ID and cycle counter filter), the Communication Controller transmits a NULL Frame with the NULL Frame indication bit reset to 0 and the payload data reset to zero.

In the following cases the Communication Controller transmits a NULL Frame with the NULL Frame indication bit reset to 0, and the rest of the Frame Header and the Frame length unchanged (payload data is reset to zero):

- All transmit buffers configured for the slot have cycle counter filters that do not match the current cycle
- There are matching Frame ID's and cycle counter filters, but none of these transmit buffers has the transmission request flag TXR set

NULL Frames are not transmitted in the dynamic segment.

41.3.10 Receive Process

The receive process is described in the following sections.

41.3.10.1 Frame Reception

To prepare or change a Message Buffer for reception the following steps are required:

- Configure the Message Buffer as receive buffer by writing bit CFG = 0 in the Write Header Section 1 register
- Configure the receive buffer by writing the configuration data (Header Section) to the Input Buffer
- Transfer the configuration from Input Buffer to the Message RAM by writing the number of the target Message Buffer to the Input Buffer Command Request register.

Once these steps are performed, the Message Buffer functions as an active receive buffer and participates in the internal acceptance filtering process, which takes place every time the Communication Controller receives a message. The first matching receive buffer is updated from the received message. If the Message Buffer holds an unprocessed Data Section (ND = 1) it is overwritten with the new message and the MLST bit in the respective Message Buffer Status register is set.

If the payload length of a received Frame PLC is longer than the value programmed by PLC in the Header Section of the respective Message Buffer, the data field stored in the Message Buffer is truncated to that length.

If no Frame, a NULL Frame, or a corrupted Frame is received in a slot, the Data Section of the Message Buffer configured for this slot is not updated. In this case only the flags in the Message Buffer Status register are updated to signal the cause of the problem. In addition the respective MBC flag in the Message Buffer Status Changed 1,2,3,4 registers is set.

When the Data Section of a receive buffer has been updated from a received Frame, the respective New Data NDn (n = 0-31) to NDn (n = 96-127) flag in the New Data NDAT1 to NDAT4 registers is set. When the Message Handler has updated the Message Buffer status, the respective MBC flag in the Message Buffer Status Changed 1,2,3,4 registers is set. If bit RDHS1.MBI in the Header Section of the respective Message Buffer is set, the receive service request flag in the Status Service Request Register is set (RXI = 1). If enabled an service request is generated.

To read a receive buffer from the Message RAM via the Output Buffer proceed as described on “Data Transfer from Message RAM to Output Buffer”.

Note: The ND and MBC flags are automatically cleared by the Message Handler when the received message has been transferred to the Output Buffer.

41.3.10.2 NULL Frame reception

The Payload Segment of a received NULL Frame is **not** copied into the matching receive buffer. If a NULL Frame has been received, the Header Section of the matching Message Buffer is updated from the received NULL Frame. The NULL Frame indication bit in the Header Section 3 of the respective Message Buffer is reset (NFI = 0) and the respective MBC flag in the Message Buffer Status Changed 1,2,3,4 registers is set.

In case that bit ND and / or MBC were already set before this event because the Host did not read the last received message, bit MLST in the Message Buffer Status register of the respective Message Buffer is also set.

41.3.11 FIFO Function

A group of the Message Buffers can be configured as a cyclic First-In-First-Out (FIFO). The group of Message Buffers belonging to the FIFO is contiguous in the register map starting with the Message Buffer referenced by FFB and ending with the Message Buffer referenced by LCB in the Message RAM Configuration register. Up to 128 Message Buffers can be assigned to the FIFO.

41.3.11.1 Description

Every valid incoming message not matching with any dedicated receive buffer but passing the programmable FIFO filter is stored into the FIFO. In this case Frame ID, payload length, receive cycle count, and the status bits of the addressed FIFO Message Buffer are overwritten with Frame ID, payload length, receive cycle count, and the status from the received message and can be read by the Host for message identification. Bit RFNE in the Status Service Request Register shows that the FIFO is not empty, bit RFCL in the Status Service Request Register is set when the last available Message Buffer belonging to the FIFO is written, bit RFO in the Error Service Request Register shows that a FIFO overrun has been detected. If enabled, service requests are generated.

There are two index registers associated with the FIFO. The PUT Index Register (PIDX) is an index to the next available location in the FIFO. When a new message has been received it is written into the Message Buffer addressed by the PIDX register. The PIDX register is then incremented and addresses the next available Message Buffer. If the PIDX register is incremented past the highest numbered Message Buffer of the FIFO, the PIDX register is loaded with the number of the first (lowest numbered) Message Buffer in the FIFO chain. The GET Index Register (GIDX) is used to address the next Message Buffer of the FIFO to be read. The GIDX register is incremented after transfer of the contents of a Message Buffer belonging to the FIFO to the Output Buffer. The PUT Index Register and the GET Index Register are not accessible by the Host.

The FIFO is completely filled when the PUT index (PIDX) reaches the value of the GET index (GIDX). When the next message is written to the FIFO before the oldest message has been read, both PUT index and GET index are incremented and the new message overwrites the oldest message in the FIFO. This will set FIFO overrun flag RFO in the Error Service Request Register.

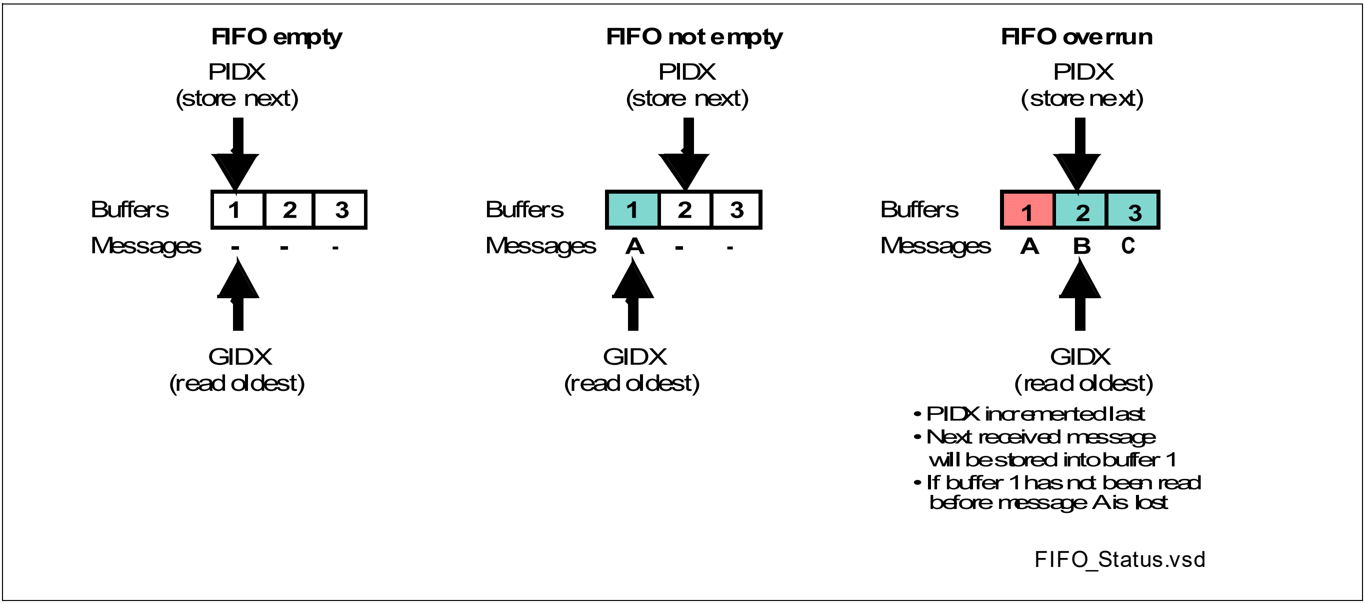

Figure 614 FIFO Status: Empty, Not Empty, Overrun

A FIFO non empty status is detected when the PUT index (PIDX) differs from the GET index (GIDX). In this case flag RFNE is set. This indicates that there is at least one received message in the FIFO. The FIFO empty, FIFO not empty, and the FIFO overrun states are explained in **Figure 614** for a three Message Buffer FIFO.

There is a programmable FIFO rejection filter for the FIFO. The FIFO Rejection Filter register (FRF) defines a filter pattern for messages to be rejected. The FIFO rejection filter consists of channel filter, Frame ID filter, and cycle counter filter. If bit RSS is set to 1 (default), all messages received in the static segment are rejected by the FIFO. If bit RNF is set to 1 (default), received NULL Frames are not stored in the FIFO.

The FIFO Rejection Filter Mask register (FRFM) specifies which bits of the Frame ID filter in the FIFO Rejection Filter register are marked "don't care" for rejection filtering.

41.3.11.2 Configuration of the FIFO

For all Message Buffers belonging to the FIFO the data pointer to the first 32-bit word of the Data Section of the respective Message Buffer in the Message RAM has to be configured via the Write Header Section 3 register. All information required for acceptance filtering is taken from the FIFO rejection filter and the FIFO rejection filter mask and needs not be configured in the Header Sections of the Message Buffers belonging to the FIFO.

When programming the data pointers for the Message Buffers belonging to the FIFO, the payload length of all Message Buffers should be programmed to the same value.

Note: It is recommended to program the MBI bits of the Message Buffers belonging to the FIFO to 0 via WRHS1.MBI to avoid generation of RX interrupts.
If the payload length of a received Frame is longer than the value programmed by WRHS2.PLC in the Header Section of the respective Message Buffer, the data field stored in a Message Buffer of the FIFO is truncated to that length.

41.3.11.3 Access to the FIFO

To read from the FIFO the Host has to trigger a transfer from the Message RAM to the Output Buffer by writing the number of the first Message Buffer of the FIFO (referenced by FFB) to the Output Buffer Command Request register. The Message Handler then transfers the Message Buffer addressed by the GET Index Register (GIDX) to the Output Buffer. After this transfer the GET Index Register (GIDX) is incremented.

41.3.12 Message Handling

The Message Handler controls data transfers between the Input / Output Buffer and the Message RAM and between the Message RAM and the two Transient Buffer RAMs. All accesses to the internal RAM's are 32 bit accesses.

Access to the Message Buffers stored in the Message RAM is done under control of the Message Handler state machine. This avoids conflicts between accesses of the two protocol controllers and the Host to the Message RAM.

Frame IDs of Message Buffers assigned to the static segment have to be in the range from 1 to NSS as configured in the GTU Configuration Register 7. Frame IDs of Message Buffers assigned to the dynamic segment have to be in the range from NSS + 1 to 2047.

Received messages with no matching dedicated receive buffer (static or dynamic segment) are stored in the receive FIFO (if configured) if they pass the FIFO rejection filter.

41.3.12.1 Host access to Message RAM

The message transfer between Input Buffer and Message RAM as well as between Message RAM and Output Buffer is triggered by the Host by writing the number of the target / source Message Buffer to be accessed to the Input or Output Buffer Command Request register.

The Input / Output Buffer Command Mask registers can be used to write / read Header and Data Section of the selected Message Buffer separately. If bit STXRS in the Input Buffer Command Mask register is set (STXRS = 1), the transmission request flag TXR of the selected Message Buffer is automatically set after the Message Buffer has been updated.

If bit STXRS in the Input Buffer Command Mask register is reset (STXRS = 0), the transmission request flag TXR of the selected Message Buffer is reset. This can be used to stop transmission from Message Buffers operated in continuous mode.

Input Buffer (IBF) and the Output Buffer (OBF) are build up as a double buffer structure. One half of this double buffer structure is accessible by the Host (IBF Host / OBF Host), while the other half (IBF Shadow / OBF Shadow) is accessed by the Message Handler for data transfers between IBF / OBF and Message RAM.

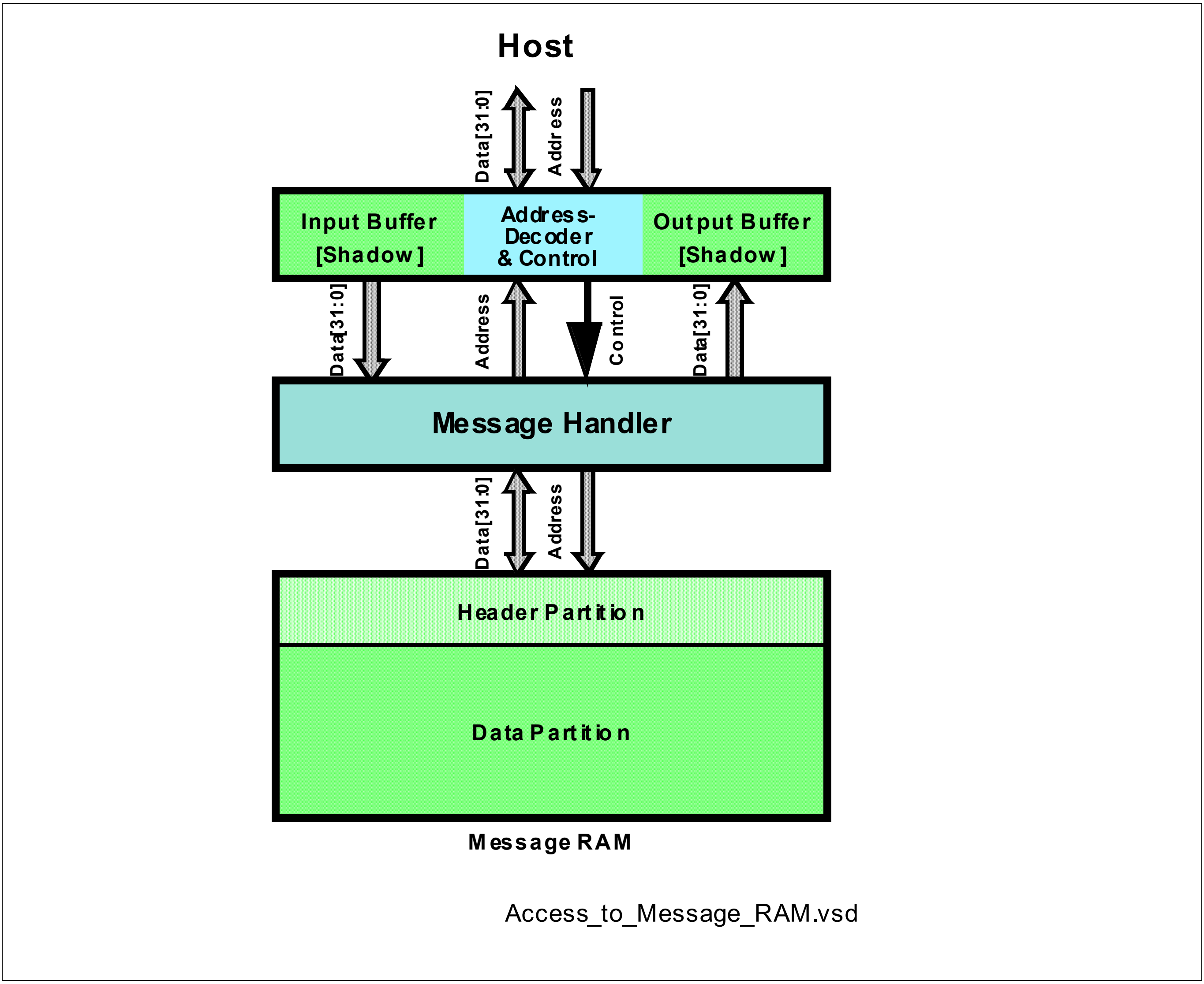

Figure 615 Host Access to Message RAM

Data Transfer from Input Buffer to Message RAM

To configure / update a Message Buffer in the Message RAM, the Host has to write the data to WRDSnn (nn = 01-64) and the Header to WRHS1, WRHS2, WRHS3. Two sets of WRDSnn (nn = 01-64) are available in parallel and selected by CUST1.IBF1PAG and CUST1.IBF2PAG. CUST1.IBFS shows which Input Buffer is currently used as Input Shadow Buffer and which as Input Host Buffer. WRHS1, WRHS2, and WRHS3 does only exist once. The specific action is selected by configuring the Input Buffer Command Mask IBCM.

When the Host writes the number of the target Message Buffer in the Message RAM to IBRH in the Input Buffer Command Request register IBCR, IBF Host and IBF Shadow are swapped (see **Figure 615**).

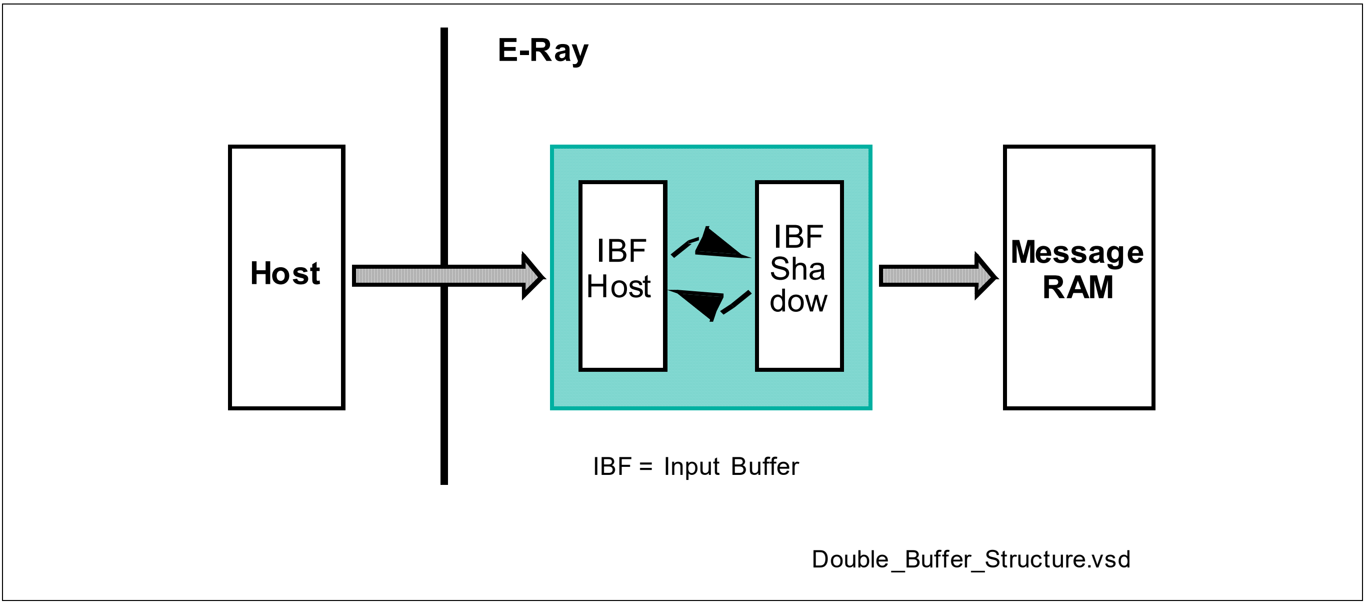

Figure 616 Swapping of IBCM and IBCR Bit

In addition the bits in the Input Buffer Command Mask and Input Buffer Command Request registers are also swapped to keep them attached to the respective IBF section (see Figure below).

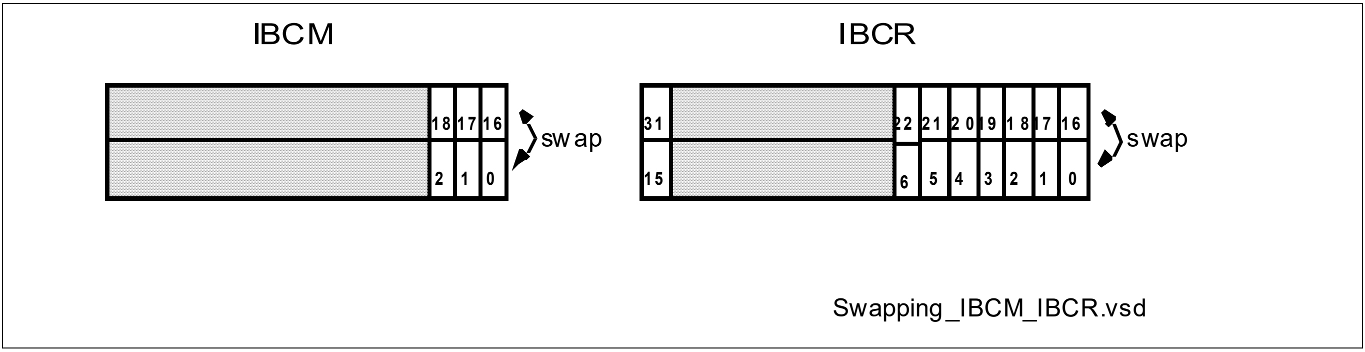

Figure 617 Swapping of IBCM and IBCR Bit

With this write operation the IBSYS bit in the Input Buffer Command Request register is set to 1. The Message Handler then starts to transfer the contents of IBF Shadow to the Message Buffer in the Message RAM selected by IBRS.

While the Message Handler transfers the data from IBF Shadow to the target Message Buffer in the Message RAM, the Host may write the next message to IBF Host. After the transfer between IBF Shadow and the Message RAM has completed, the IBSYS bit is set back to 0 and the next transfer to the Message RAM may be started by the Host by writing the respective target Message Buffer number to IBRH in the Input Buffer Command Request register.

If a write access to IBRH occurs while IBSYS is 1, IBSYH is set to 1. After completion of the ongoing data transfer from IBF Shadow to the Message RAM, IBF Host and IBF Shadow are swapped, IBSYH is reset to 0, IBSYS remains set to 1, and the next transfer to the Message RAM is started. In addition the Message Buffer numbers stored under IBRH and IBRS and the Command Mask flags are also swapped.

Table 395 Assignment of Input Buffer Command Mask Bit

Pos.	Access	Bit	Function
18	rh	STXRS	Set Transmission Request Shadow
17	rh	LDSS	Load Data Section Shadow
16	rh	LHSS	Load Header Section Shadow
2	rwh	STXRH	Set Transmission Request Host
1	rwh	LDSH	Load Data Section Host
0	rwh	LHSH	Load Header Section Host

Table 396 Assignment of Input Buffer Command Request Bit

Pos.	Access	Bit	Function
31	rh	IBSYS	**IBF Busy Shadow,** signals ongoing transfer from IBF Shadow to Message RAM
22…16	rh	IBRS	**IBF Request Shadow,** number of Message Buffer currently / last updated
15	rh	IBSYH	**IBF Busy Host,** transfer request pending for Message Buffer referenced by IBRH
6-0	rwh	IBRH	**IBF Request Host,** number of Message Buffer to be updated next

Data Transfer from Message RAM to Output Buffer

To read a Message Buffer from the Message RAM, the Host has to write to Command Request register OBCR to trigger the data transfer as configured in Output Buffer Command Mask OBCM register. After the transfer has completed, the Host can read the transferred data from RDDSnn (nn = 01-64), RDHS1, RDHS2, RDHS3, and MBS.

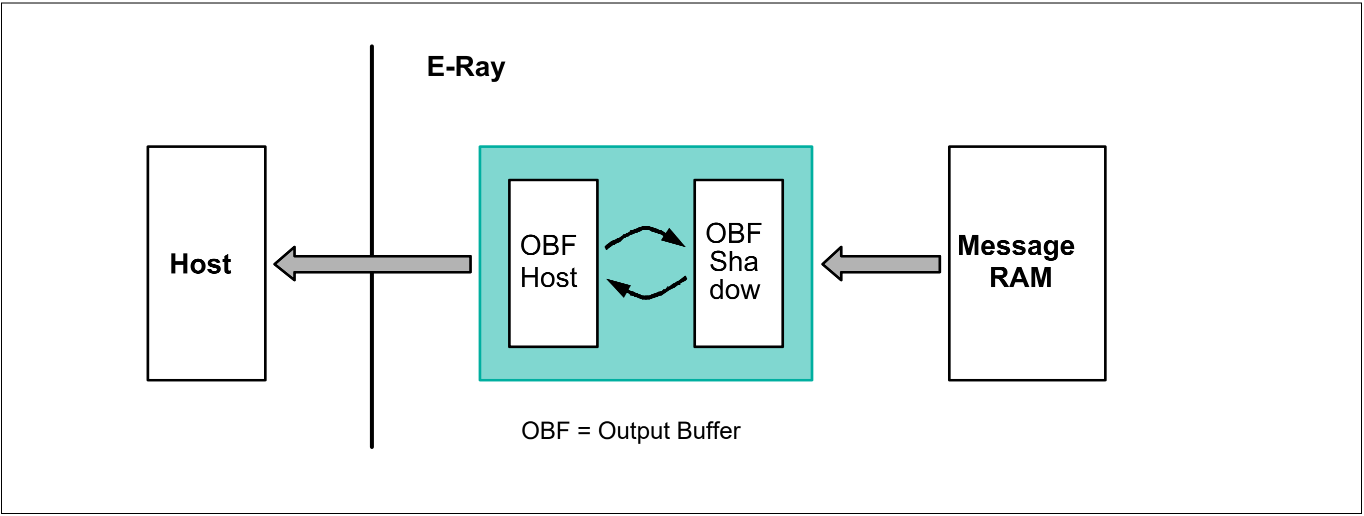

Figure 618 Double Buffer Structure Output Buffer

OBF Host and OBF Shadow as well as bits OBCM.RHSS, OBCM.RDSS, OBCM.RHSH, OBCM.RDSH and bits OBCR.OBRS, OBCR.OBRH are swapped under control of bits OBCR.VIEW and OBCR.REQ.

Writing bit OBCR.REQ to 1 copies bits OBCM.RHSS, OBCM.RDSS and bits OBCR.OBRS to an internal storage (see **Figure 619**).

After setting OBCR.REQ to 1, OBCR.OBSYS is set to 1, and the transfer of the Message Buffer selected by OBCR.OBRS from the Message RAM to OBF Shadow is started. After the transfer between the Message RAM and OBF Shadow has completed, the OBCR.OBSYS bit is set back to 0. Bits OBCR.REQ and OBCR.VIEW can only be set to 1 while OBCR.OBSYS is 0.

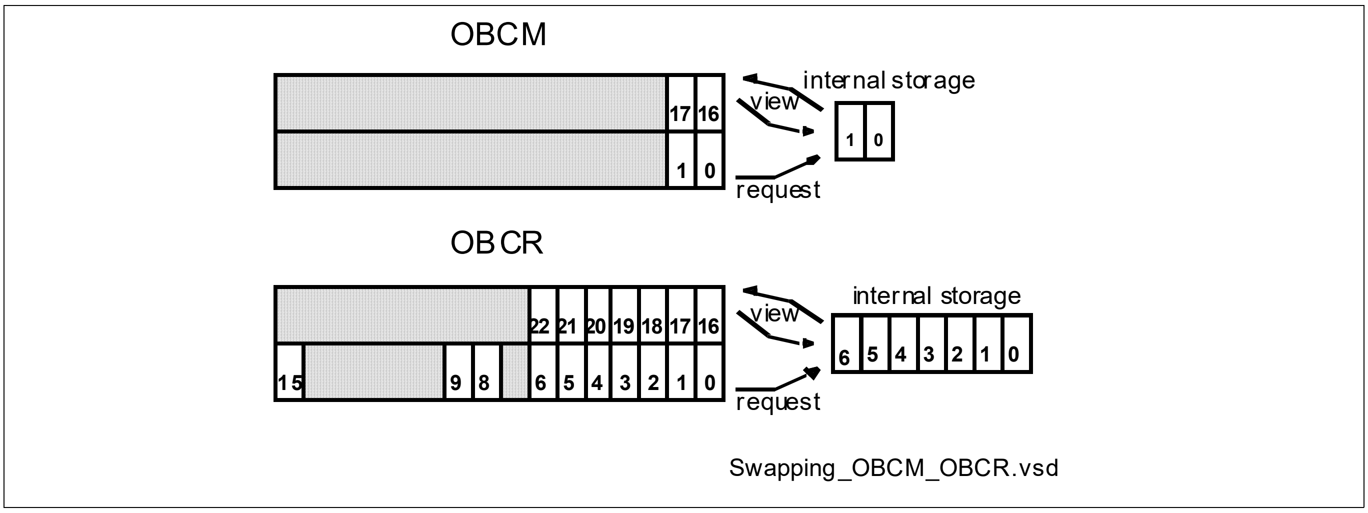

Figure 619 Swapping of OBCM and OBCR Bit

OBF Host and OBF Shadow are swapped by setting bit OBCR.VIEW to 1 while bit OBCR.OBSYS is 0 (see **Figure 619**).

In addition bits OBCR.OBRH and bits OBCM.RHSH, OBCM.RDSH are swapped with the registers internal storage thus assuring that the Message Buffer number stored in OBCR.OBRH and the mask configuration stored in OBCM.RHSH, OBCM.RDSH matches the transferred data stored in OBF Host (see **Figure 619**).

Now the Host can read the transferred Message Buffer from OBF Host while the Message Handler may transfer the next message from the Message RAM to OBF Shadow.

Table 397 Assignment of Output Buffer Command Mask Bit

Pos.	Access	Bit	Function
17	rh	RDSH	Data Section available for Host access
16	rh	RHSH	Header Section available for Host access
1	rwh	RDSS	Read Data Section Shadow
0	rwh	RHSS	Read Header Section Shadow

Table 398 Assignment of Output Buffer Command Request Bit

Pos.	Access	Bit	Function
22…16	rh	OBRH	**OBF Request Host** number of Message Buffer available for Host access
15	rh	OBSYS	**OBF Busy Shadow** signals ongoing transfer from Message RAM to OBF Shadow
9	rw	REQ	**Request Transfer from Message RAM to OBF Shadow**
8	rw	VIEW	**View OBF Shadow, swap OBF Shadow, and OBF Host**
6…0	rw	OBRS	**OBF Request Shadow** number of Message Buffer for next request

41.3.12.2 Data Transfers between IBF / OBF and Message RAM

This document uses the following terms and abbreviations:

Table 399 Terms and Abbreviations

Term	Meaning
MHD	Message Handler
IBF	Input Buffer 1 or 2 RAM
OBF	Output Buffer 1 or 2 RAM
MBF	Message Buffer RAM
TBF	Transient Buffer RAM Channel A (TBF1) or Channel B (TBF2)
IBF⇒ MBF	Transfer from IBF to MBF
MBF ⇒ OBF	Transfer from MBF to OBF
MBF⇒ TBF	Transfer from MBF to TBF
TBF⇒ MBF	Transfer from TBF to MBF
SS	Slot Status
SS⇒ MBF	Transfer SS to MBF

Message Handler functionality

The MHD controls the access to the MBF. It manages data-transfer between MBF and IBF, OBF, TBF1, TBF2. The data-path are shown in Figure below.

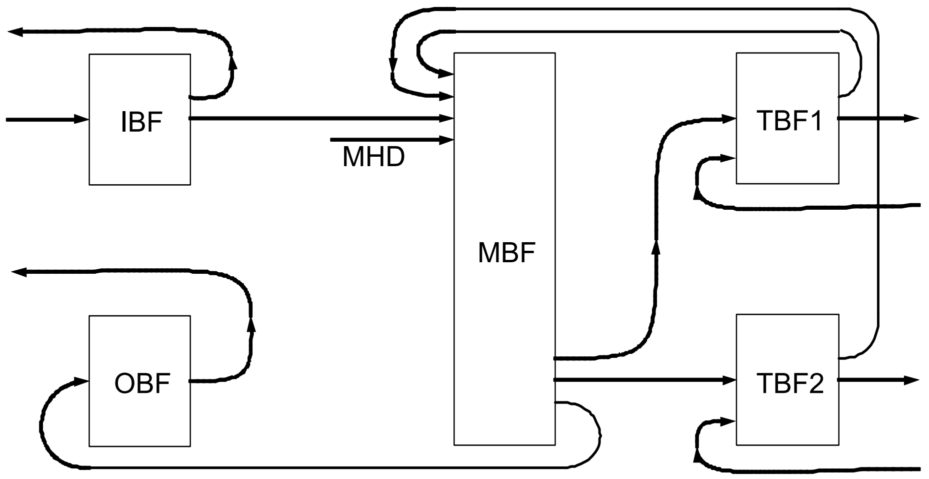

Figure 620 Interconnection of RAMs

Furthermore a search-algorithm allows to find the next valid message object in the MBF for transmission or reception.

Each transfer consists of a setup-time, four time steps to transfer the Header-section and a payload-length-dependent number of time steps to transfer the data-section. The internal data-busses have a width of 32 bits. Thereby it is possible to transfer two 2-byte words in one time step. If the payload consists of an odd number of

2-byte words the last time step of the data-section contains only 16 bit of valid data. If the Payload-Length (PL) is e.g. 7, the data-section consists of 4 time steps.

The maximum length for the data-section is 64 time steps, the minimum length is zero time steps.

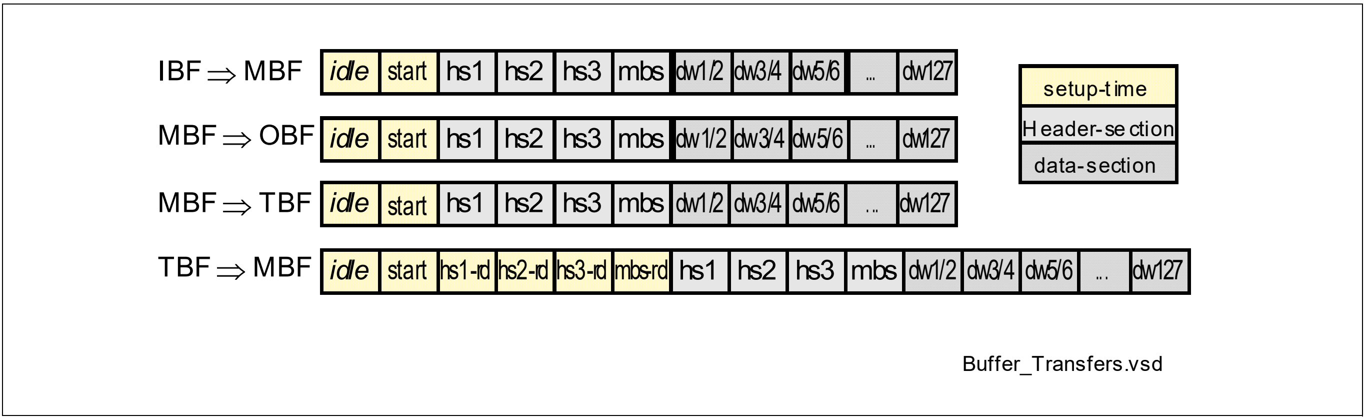
IBF ⇒ MBF
idle start hs1 hs2 hs3 mbs dw1/2 dw3/4 dw5/6 ... dw127
MBF ⇒ OBF
idle start hs1 hs2 hs3 mbs dw1/2 dw3/4 dw5/6 ... dw127
MBF ⇒ TBF
idle start hs1 hs2 hs3 mbs dw1/2 dw3/4 dw5/6 ... dw127
TBF ⇒ MBF
idle start hs1-rd hs2-rd hs3-rd mbs-rd hs1 hs2 hs3 mbs dw1/2 dw3/4 dw5/6 ... dw127
setup-time
Header-section
data-section
Buffer_Transfers.vsd

Figure 621 Different Possible Buffer Transfers

The update of the Slot-Status consists of a setup-time and one time-step to write the new Slot-Status.

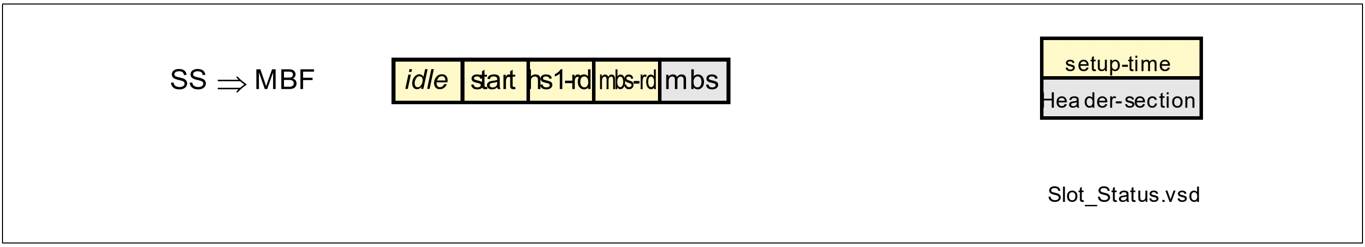

Figure 622 Update of Slot Status

The length of a time step depends on the number of concurrent tasks.

The following concurrent tasks are executed under control of the Message Handler:

- Data transfer between IBF or OBF and MBF
- Data transfer between TBF1 and MBF, search next TX / RX Message Buffer CHA
- Data transfer between TBF2 and MBF, search next TX / RX Message Buffer CHB

Thereby the time step length can vary between one and three f_{CLC_ERAY}periods.

Under certain conditions it is possible that a transfer is stopped or interrupted for a number of time steps until it is continued.

When a IB FÞ MBF is started short after a TBF Þ MBF or SS Þ MBF the transfer from IBF has to wait until the setup-time of the internal transfer has finished (see Figure below)

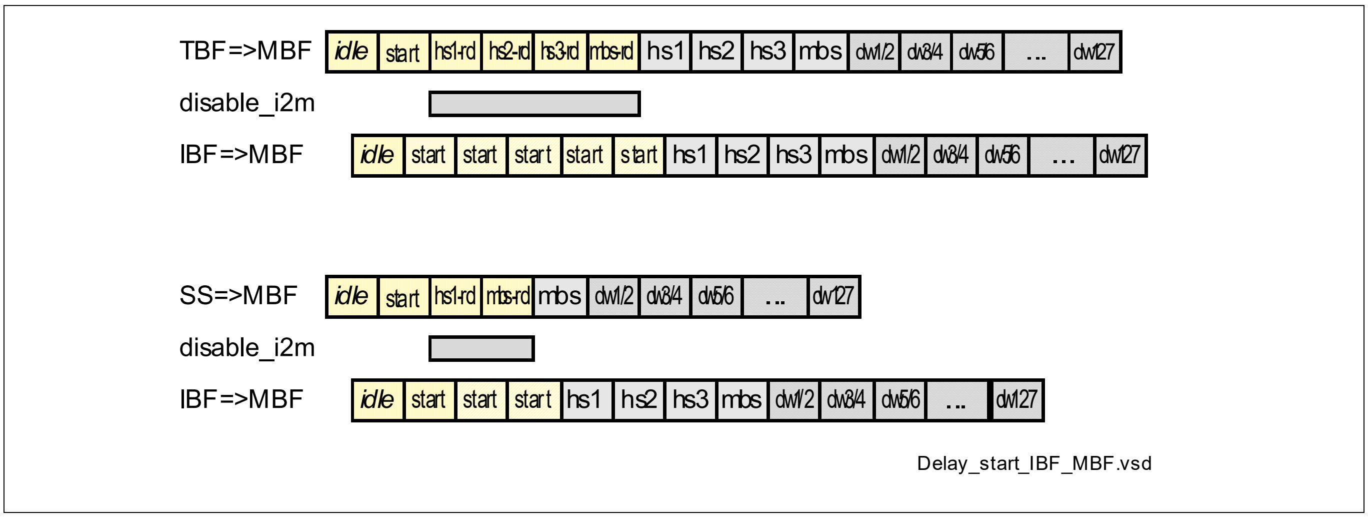

Figure 623 Delay start of IBF => MBF

The internal signal "disable_i2m" is always active when the TBF ⇒ MBF is in state "hs1-rd", "hs2-rd", "hs3-rd" or "mbs-rd" and when the SS⇒ MBF is in state "hs1-rd" or "mbs-rd".

The IBF ⇒ MBF is hold in state "start" until the internal signal "disable_i2m" gets inactive.

These additional time-steps are independent of any address-counter-values. This means, the IBF ⇒ MBF has to wait even if it writes to another buffer than the internal transfer.

Multiple requests of transfers between IBF/OBF and Message RAM

The time required to transfer the contents of a Message Buffer between IBF / OBF and Message RAM depends on the number of 4-byte words to be transferred, the number of concurrent tasks to be managed by the Message Handler, and in special cases the type and address range of the internal transfer. The number of 4-byte words varies from 4 (Header Section only) to 68 (Header + maximum Data Section) plus a short setup time to start the first transfer, while the number of concurrent task varies from one to three. The 4 Header words have to be included in calculation even if only the Data Section is requested for transfer.

The following concurrent tasks are executed under control of the Message Handler:

- Data transfer between IBF or OBF and MBF
- Data transfer between TBF1 and MBF, search next TX / RX Message Buffer CHA
- Data transfer between TBF2 and MBF, search next TX / RX Message Buffer CHB

Transfers between IBF and MBF respectively MBF and OBF can only be handled one after another. In case that e.g. a IBF⇒ MBF has been started shortly before a MBF ⇒ OBF is requested, the MBF ⇒ OBF has to wait until the IBF ⇒ MBF has completed.

In case that e.g. a second IBF⇒ MBF is requested, a MBF⇒ OBF is requested and a IBF⇒ MBF is ongoing, the MBF⇒ OBF has to wait until the first IBF⇒ IBF has completed. The second IBF=MBF has to wait until the MBF⇒ OBF has completed (see figure below) independent whether MBF⇒ OBF or second IBF⇒ MBF is requested first.

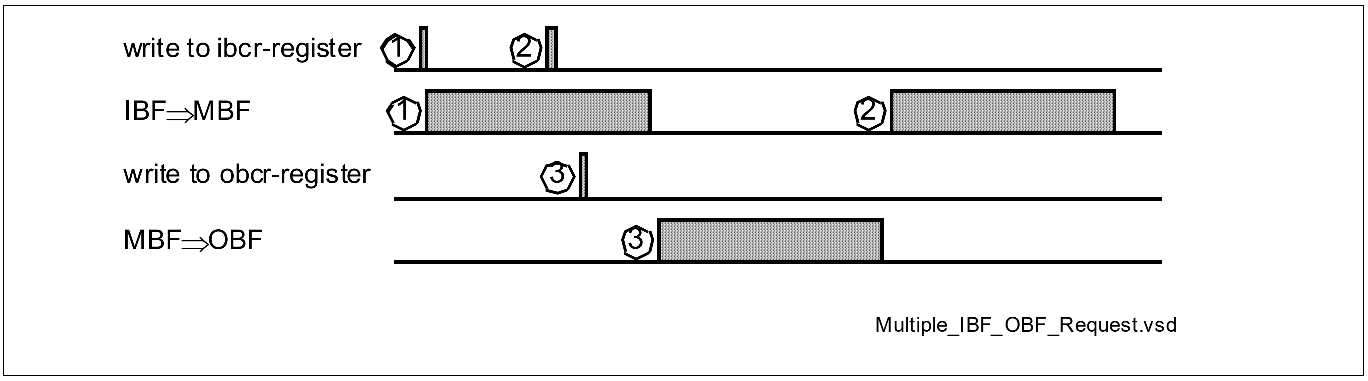

Figure 624 Multiple IBF/OBF Request

Worst case for single request

When a message with a large payload length is received the TBF⇒ MBF is started at the begin of the next slot (n+1). If the next slot is a dynamic slot without transmission/reception (minislot), it may happen that the TBF⇒ MBF has not finished until begin of the next but one slot (n+2). In this case the TBF⇒ MBF will be service requested (break) to start a transmission in the next but one slot (MBF⇒ TBF) and/or to update the slot status (SS⇒ MBF) for the RX-buffer corresponding with next slot (n+1). After this interruption the TBF⇒ MBF is continued.

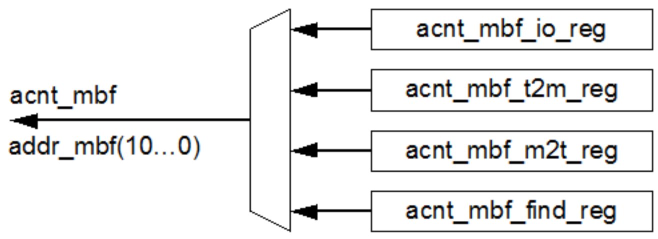

Figure 625 Address Counter Scheme of Message RAM (simplified)

For the transfers IBF⇒ MBF / MBF⇒ OBF, TBF⇒ MBF and MBF⇒ TBF separate address-counter are implemented (see **Figure 625**).

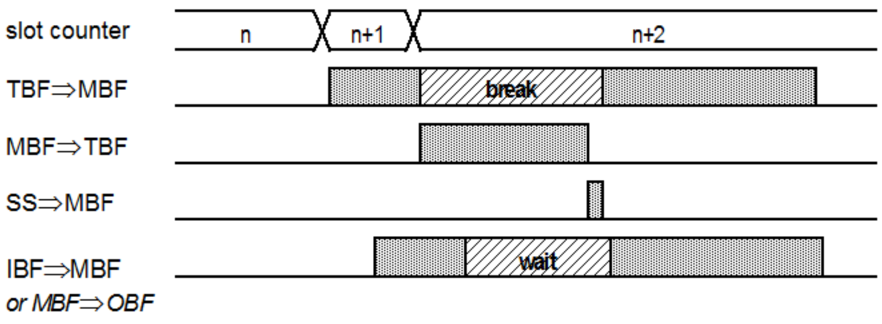

Figure 626 interruption of TBF to MBF

If the address-counter for IBF$\Rightarrow$ MBF / MBF$\Rightarrow$ OBF (acnt_mbf_io_reg) reaches the address of the interrupted TBF=>MBF (acnt_mbf_t2m_reg) the IBF$\Rightarrow$ MBF / MBF$\Rightarrow$ OBF has to wait until the TBF$\Rightarrow$ MBF is continued (see **Figure 626**).

The relative time is measured in f_{CLC_ERAY}cycles. Absolute time depends on the actual f_{CLC_ERAY}cycle period.

tbf_to_mbf_break time$_{max}$ =	(setup time + mbf_to_tbf time$_{max}$) + (setup time + ss_to_mbf)
cycles$_{req}$ =	(number of concurrent tasks) x ((setup time + (number of 4-byte words)$_{req}$) + tbf_to_mbf_break time)
setup time =	2 f_{CLC_ERAY}cycles

Worst case for one IBF$\Rightarrow$ MBF or MBF$\Rightarrow$ OBF:

Max. break time: tbf_to_mbf_break time$_{max}$ =	(2+68) + (4+1) = 75
Max. number of f_{CLC_ERAY}cycles: cycles$_{req}$ =	3 x (6 + 68 + 75) = 435

Worst case for multiple transfers

If a second IBF⇒ MBF and a MBF⇒ OBF (see **Figure 624**) is requested directly after the first IBF⇒ MBF has started following worst case timing could appear:

$cycles_{trans}$ =	(remaining cycles of transfer running) + (cycles of second requested transfer) + (cycles of third requested transfer)
$cycles_{trans}$ =	$cycles_{rem} + cycles_{req\_2} + cycles_{req\_3}$
Max. number of f_{CLC_ERAY} cycles: $cycles_{trans}$ =	447 + 435 + 447 = 1329

41.3.12.3 Minimum f_{CLC_ERAY}

To calculate the minimum f_{CLC_ERAY} the worst case scenario has to be considered.

The worst case scenario depends on the following parameters

- maximum payload length
- minimum minislot length
- number of configured Message Buffers (excluding FIFO)
- used channels (single/dual channel)

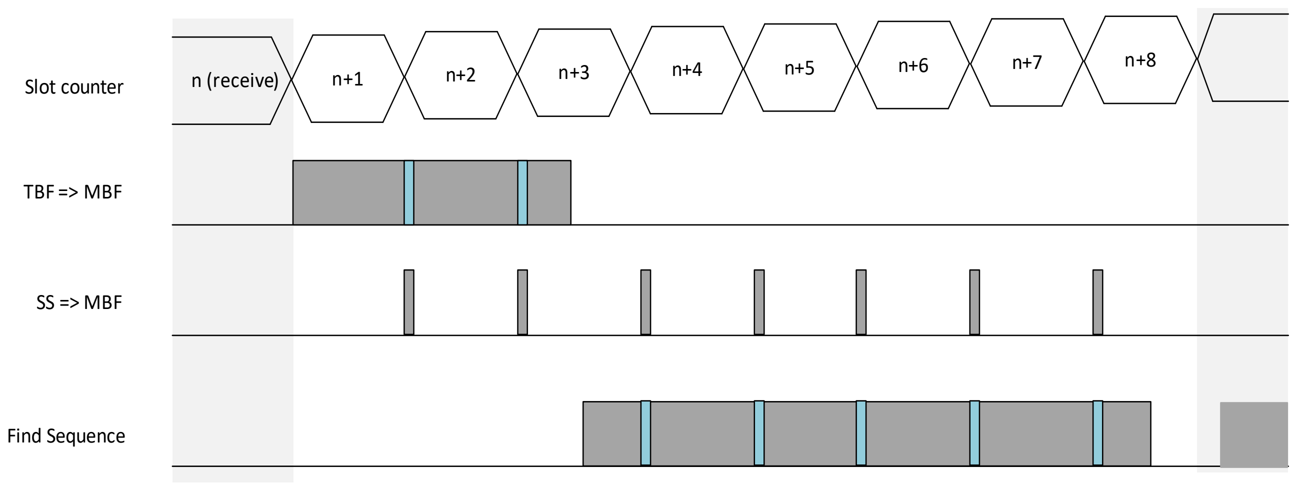

Figure 627 worst case scenario

Worst case scenario:

- reception of message with a maximum payload length in Slot n (n is 7,15,23,31,39,…)
- slot n+1 to n+7 are empty dynamic slots (minislot) and configured as receive buffer
- the find-sequence (usually started in slot 8,16,24,32,40,…) has to scan the maximum number of configured buffers
- the number of concurrent tasks has its maximum value of three

The find-sequence is executed each 8 Slots (slot 8,16,24,32,40,…). It has to be finished until the next find-sequence is requested.

The length of a TBF⇒ MBF varies from 4 (Header Section only) to 68 (Header + maximum Data Section) time step plus a setup time of 6 time steps.

f_{CLC_ERAY}cycles$_{t2m}$ =	number of concurrent tasks) x (setup time$_{t2m}$ + (number of 4-byte words)$_{t2m}$)

A SS⇒ MBF has a fixed length of 1 time steps plus a setup time of 4 time steps.

f_{CLC_ERAY} cycles$_{ss2m}$ =	(number of concurrent tasks) x 5

The find sequence has a maximum length of 128 (maximum number of buffers) time steps plus a setup time of 2 time steps.

f_{CLC_ERAY} cycles$_{find}$ =	(number of concurrent tasks) x (setup time$_{find}$ + (number of configured buffers))

A minislot has a length of 2 to 63 Macrotick (gdMinislot). The minimum nominal Macrotick period (cdMinMTNom) is 1 µs. A sequence of 8 minislots has a length of

time$_{8minislots}$ =	8 x gdMinislot x cdMinMTNom

The maximum period T$_{CLC\_ERAY}$ = 1/f_{CLC_ERAY} can be calculated as followed:

time$_{8minislots}$ ≥ (f_{CLC_ERAY} period in µs) x ((f_{CLC_ERAY} cycles$_{t2m}$) + 7 x (f_{CLC_ERAY} cycles$_{ss2m}$) + (f_{CLC_ERAY}cycles$_{find}$))

f_{CLC_ERAY}period in ms ≤

$$\frac{\mathrm{time}_{8\mathrm{minislots}}}{(\mathrm{cycles}_{\mathrm{t2m}}) + 7\times(\mathrm{cycles}_{\mathrm{ss2m}}) + (\mathrm{cycles}_{\mathrm{find}})}$$

minimum time$_{8minislots}$ =	8 x 2 x 1 µs = 16 µs
maximum f_{CLC_ERAY} cycles$_{t2m}$ =	3 x (6 + 68) = 222
maximum f_{CLC_ERAY} cycles$_{ss2m}$ =	3 * 5 = 15
maximum f_{CLC_ERAY} cycles$_{find}$ =	3 * (2 + 128) = 390

f_{CLC_ERAY} period in ms ≤

$$\frac{16\mu s}{222+7\times15+390} \approx 22.315ns$$

The minimum f_{CLC_ERAY} frequency for this worst case scenario is 44.8125 MHz.
A too low f_{CLC_ERAY} frequency can cause a malfunction of the E-Ray.
The E-Ray can detect several malfunctions and reports this by setting the corresponding flag in the Message Handler Constraints Flags (MHDF) register.

41.3.12.3.1 Minimum f_{CLC_ERAY} for various maximum payload length

Table below summarizes the minimum required f_{CLC_ERAY} frequency for various maximum payload length assuming:

- a minimum minislot length of 2 µs.
- a maximum of 128 configured Message Buffers.

- dual channels in use.

Table 400 Minimum f_{CLC_ERAY} for different maximum payload length

Maximum payload length of 32 bit words	**4**	**8**	**16**	**32**	**64**
minimum f_{CLC_ERAY}	32,82 MHz	33,57 MHz	35,07 MHz	38,07 MHz	44,1 MHz

41.3.12.3.2 Minimum f_{CLC_ERAY} for various minimum minislot length

Table below summarizes the minimum required f_{CLC_ERAY} frequency for various minimum minislot length assuming:

- a maximum payload length of 254 bytes / 64 four-byte-words.
- a maximum 128 configured Message Buffers.
- dual channels in use.

Table 401 Minimum f_{CLC_ERAY} for different minimum minislot length

gdMinislot at dMinMTNom = 1 µs	**2 µs**	**3 µs**	**4 µs**	**7 µs**	**8 µs**
minimum f_{CLC_ERAY}	44,82 MHz	29,88 MHz	22,412 MHz	12,8 MHz	9,96 MHz

41.3.12.3.3 Minimum f_{CLC_ERAY} for various amount of configured Message Buffers

Table below summarizes the minimum required f_{CLC_ERAY} frequency for various amount of configured Message Buffers assuming:

- a maximum payload length of 254 bytes / 64 four-byte-words.
- a minimum minislot length of 2 µs.
- dual channels in use.

Table 402 Minimum f_{CLC_ERAY} for different amount of configured Message Buffers

Configured maximum amount of Message Buffers	**128**	**64**	**32**
minimum f_{CLC_ERAY}	44,82 MHz	32,82 MHz	26,82 MHz

41.3.12.3.4 Minimum f_{CLC_ERAY} for a typical configuration

´The minimum required f_{CLC_ERAY} frequency for various assuming the following typical E-Ray configuration:

- a maximum payload length of 32 bytes / 8 four-byte-words.
- a minimum minislot length of 7 µs.
- a maximum 128 configured Message Buffers.
- dual channels in use

The minimum f_{CLC_ERAY} frequency for this typical example would be 10 MHz.

41.3.12.4 FlexRay™ Protocol Controller access to Message RAM

The two Transient Buffer RAMs (TBF 1, TBF 2) are used to buffer the data for transfer between the two FlexRay™ Protocol Controllers and the Message RAM.

Each Transient Buffer RAM is build up as a double buffer, able to store two complete FlexRay™ messages. There is always one buffer assigned to the corresponding Protocol Controller while the other one is accessible by the Message Handler.

If e.g. the Message Handler writes the next message to be send to Transient Buffer Tx, the FlexRay™ Channel Protocol Controller can access Transient Buffer Rx to store the message it is actually receiving. During transmission of the message stored in Transient Buffer Tx, the Message Handler transfers the last received message stored in Transient Buffer Rx to the Message RAM (if it passes acceptance filtering) and updates the respective Message Buffer.

Data transfers between the Transient Buffer RAMs and the shift registers of the FlexRay™ Channel Protocol Controllers are done in words of 32 bit. This enables the use of a 32 bit shift register independent of the length of the FlexRay™ messages.

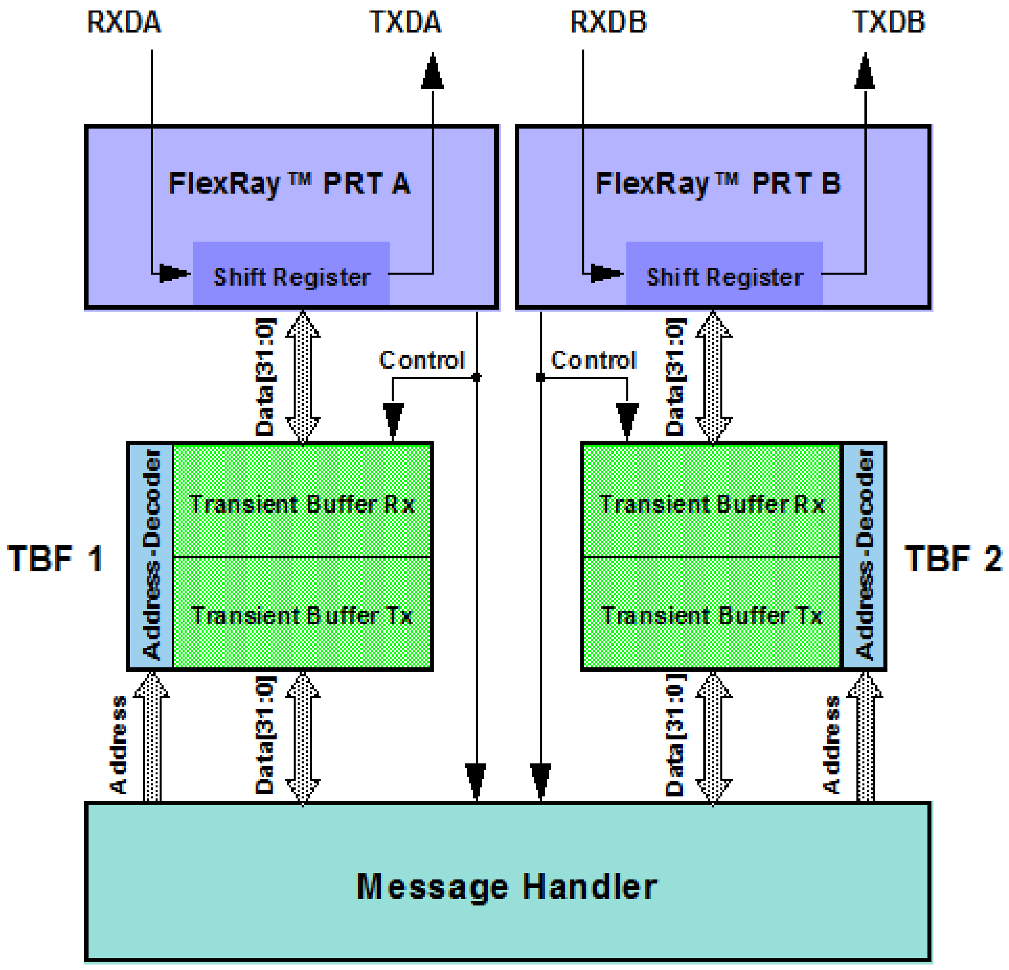

Figure 628 Access to Transient Buffer RAMs

41.3.13 Message RAM

To avoid conflicts between Host access to the Message RAM and FlexRay™ message reception / transmission, the Host cannot directly access the Message Buffers in the Message RAM. These accesses are handled via the Input and Output Buffers. The Message RAM is able to store up to 128 Message Buffers depending on the configured payload length.

The Message RAM is organized in 2048 32-bit words. To achieve the required flexibility with respect to different numbers of data byte per FlexRay™ Frame (0 to 254), the Message RAM has a structure as shown in **Figure 629**.

The Data Partition is allowed to start at Message RAM word number: (MRC.LCB + 1) • 4

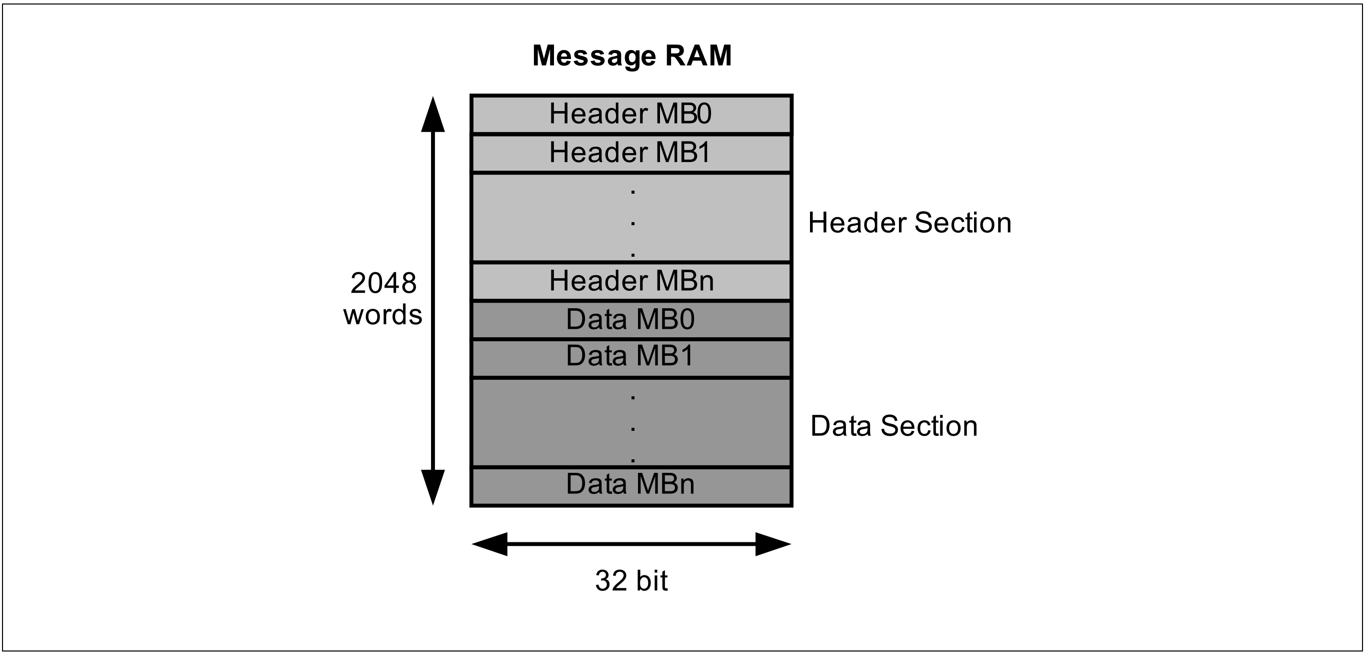

Figure 629 Structure of Message RAM

Header Partition

Stores Header Segments of FlexRay™ Frames:

- Supports a maximum of 128 Message Buffers
- Each Message Buffer has a Header of four 32 bit words
- Header 3 of each Message Buffer holds the11 bit pointer to the respective Data Section in the Data Partition

Data Partition

Flexible storage of Data Sections with different length. Some maximum values are:

- 30 Message Buffers with 254 byte Data Section each
- Or 56 Message Buffers with 128 byte Data Section each
- Or 128 Message Buffers with 48 byte Data Section each

Restriction: Header Partition + Data Partition may not occupy more than 2048 32-bit words.

41.3.13.1 Header Partition

The Header of each Message Buffer occupies four 32-bit words in the Header Partition of the Message RAM. The Header of Message Buffer 0 starts with the first word in the Message RAM.

For transmit buffers the Header CRC has to be calculated by the Host.

Payload Length Received PLR, Receive Cycle Count RCC, Received on Channel Indication RCI, Startup Frame Indication bit SFI, Sync bit SYN, NULL Frame Indication bit NFI, Payload Preamble Indication bit PPI, and Reserved bit RES are only updated from received valid Frames (including valid NULL Frames).

Header word 4 of each configured Message Buffer holds the respective Message Buffer Status MBS information.

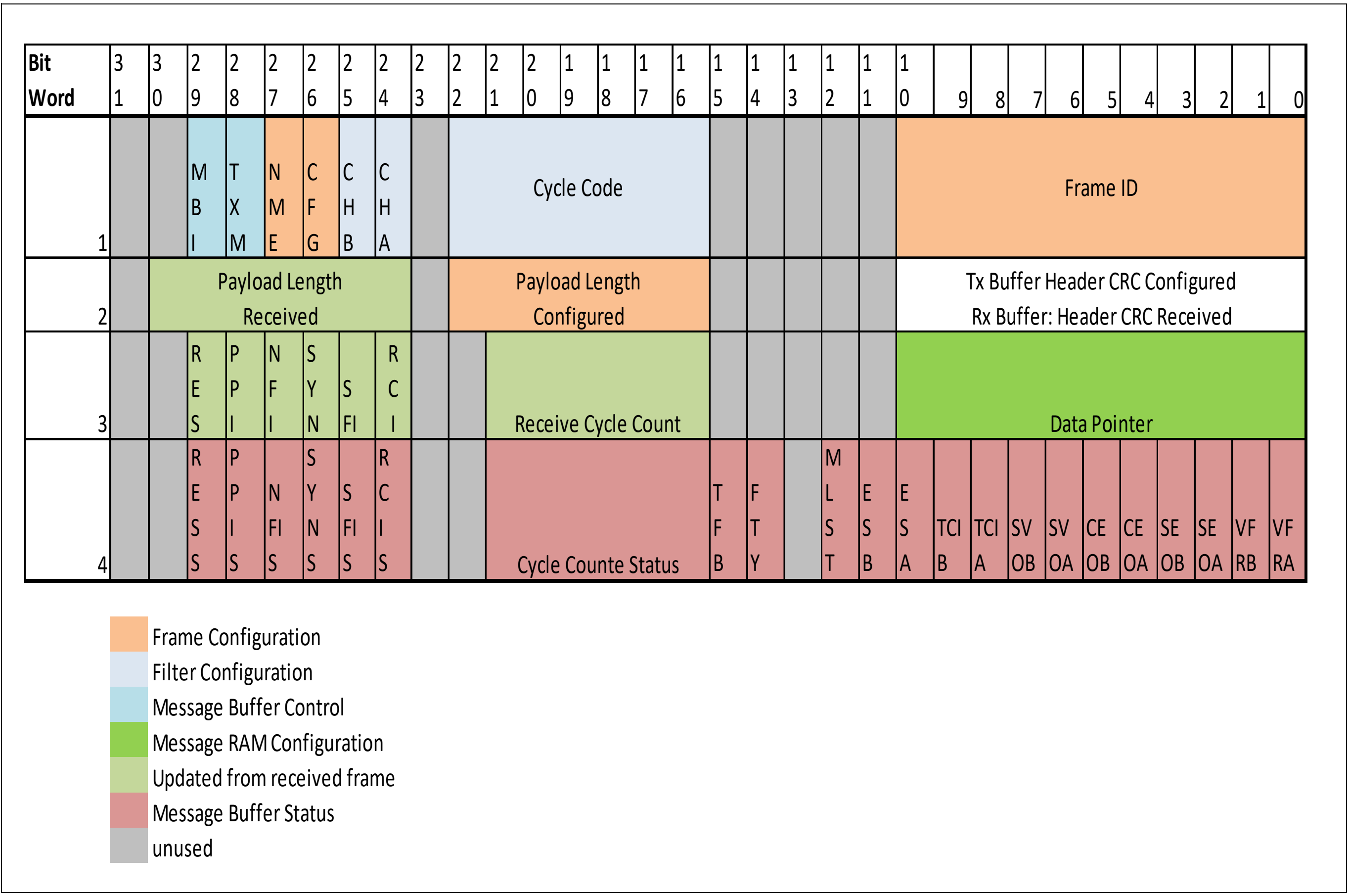

Figure 630 Header Section of a Message Buffer in the Message RAM

Header 1 (word 0)

Write access via WRHS1, read access via RDHS1:

- Frame ID: Slot counter filtering configuration
- Cycle Code: Cycle counter filtering configuration
- CHA, CHB: Channel filtering configuration
- CFG: Message Buffer configuration: receive / transmit
- PPIT: Payload Preamble Indicator Transmit
- XMI: Transmit mode configuration: single-shot / continuous
- MBI: Message Buffer receive / transmit service request enable

Header 2 (word 1)

Write access via WRHS2, read access via RDHS2:

- Header CRC
  - Transmit Buffer: Configured by the Host (calculated from Frame Header Segment)
  - Receive Buffer: Updated from received Frame
- Payload Length Configured
  - Length of Data Section (2-byte words) as configured by the Host
- Payload Length Received
  - Length of Payload Segment (2-byte words) stored from received Frame

Header 3

Write access via WRHS3, read access via RDHS3:

- Data Pointer
  - Pointer to the beginning of the corresponding Data Section in the Data Partition

Read access via RDHS3, valid for receive buffers only, updated from received Frames:

- Receive Cycle Count: Cycle count from received Frame
- RCI: Received on Channel Indicator
- SFI: Startup Frame Indicator
- SYN: SYNC Frame Indicator
- NFI: NULL Frame Indicator
- PPI: Payload Preamble Indicator
- RES: Reserved bit

Message Buffer Status MBS (word 3)

Read access via MBS, updated by the Communication Controller at the end of the configured slot.

- VFRA: Valid Frame Received on channel A
- VFRB: Valid Frame Received on channel B
- SEOA: Syntax Error Observed on channel A
- SEOB: Syntax Error Observed on channel B
- CEOA: Content Error Observed on channel A
- CEOB: Content Error Observed on channel B
- SVOA: Slot boundary Violation Observed on channel A
- SVOB: Slot boundary Violation Observed on channel B
- TCIA: Transmission Conflict Indication channel A
- TCIB: Transmission Conflict Indication channel B
- ESA: Empty Slot Channel A
- ESB: Empty Slot Channel B
- MLST: Message LoST
- FTA: Frame Transmitted on Channel A
- FTA: Frame Transmitted on Channel B
- Cycle Count Status: Actual cycle count when status was updated
- RCIS: Received on CHannel Indicator Status
- SFIS: Startup Frame Indicator Status
- SYNS: SYNC Frame Indicator Status
- NFIS: NULL Frame Indicator Status
- PPIS: Payload Preamble Indicator Status
- RESS: Reserved Bit Status

41.3.13.2 Data Partition

The Data Partition of the Message RAM stores the Data Sections of the Message Buffers configured for reception / transmission as defined in the Header Partition. The number of data bytes for each Message Buffer can vary from 0 to 254. To optimize the data transfer between the shift registers of the two FlexRay™ Protocol Controllers and the Message RAM as well as between the Host interface and the Message RAM, the physical width of the Message RAM is set to 4 bytes.

The Data Partition starts after the last word of the Header Partition. When configuring the Message Buffers in the Message RAM the programmer has to assure that the data pointers point to addresses within the Data Partition. **Table 403** below shows an example how the Data Sections of the configured Message Buffers can be stored in the Data Partition of the Message RAM.

The beginning and the end of a Message Buffer's Data Section is determined by the data pointer and the payload length configured in the Message Buffer's Header Section, respectively. This enables a flexible usage of the available RAM space for storage of Message Buffers with different data length.

If the size of the Data Section is an odd number of 2-byte words, the remaining 16 bits in the last 32-bit word are unused (see Table below)

Table 403 Example for Structure of the Data Section in the Message RAM

Bit Word	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
…	unused								unused								unused								unused							
…	unused								unused								unused								unused							
…	**MB1** Data3								**MB1** Data2								**MB1** Data1								**MB1** Data0							
…	…								…								…								…							
…	…								…								…								…							
…	**MB1** Data(m)								**MB1** Data(m-1)								**MB1** Data(m-2)								**MB1** Data(m-3)							
…	…								…								…								…							
…	…								…								…								…							
…	…								…								…								…							
…	**MBn** Data3								**MBn** Data2								**MBn** Data1								**MBn** Data0							
…	…								…								…								…							
…	**MBn** Data(k)								**MBn** Data(k-1)								**MBn** Data(k-2)								**MBn** Data(k-3)							
2046	**MB80** Data3								**MB80** Data2								**MB80** Data1								**MB80** Data0							
2047	unused								unused								**MB80** Data5								**MB80** Data4							

41.3.13.3 ECC Check

There is an ECC checking mechanism implemented in the E-Ray module to assure the integrity of the data stored in the seven RAM blocks of the module. The RAM blocks have an ECC generator / checker attached as shown in **Figure 631**. When data is written to a RAM block, the local ECC generator generates the ECC data. The ECC data is stored together with the respective data word. The ECC data is checked each time a data word is read from any of the RAM blocks.

If an ECC error is detected, the respective error flag is set. The ECC error flags MHDS.EIBF, MHDS.EOBF, MHDS.EMR, MHDS.ETBF1, MHDS.ETBF2, and the faulty Message Buffer indicators MHDS.FMBD, MHDS.MFMB, MHDS.FMB are located in the Message Handler Status register. These error flags control the error interrupt flag EIR.EERR.

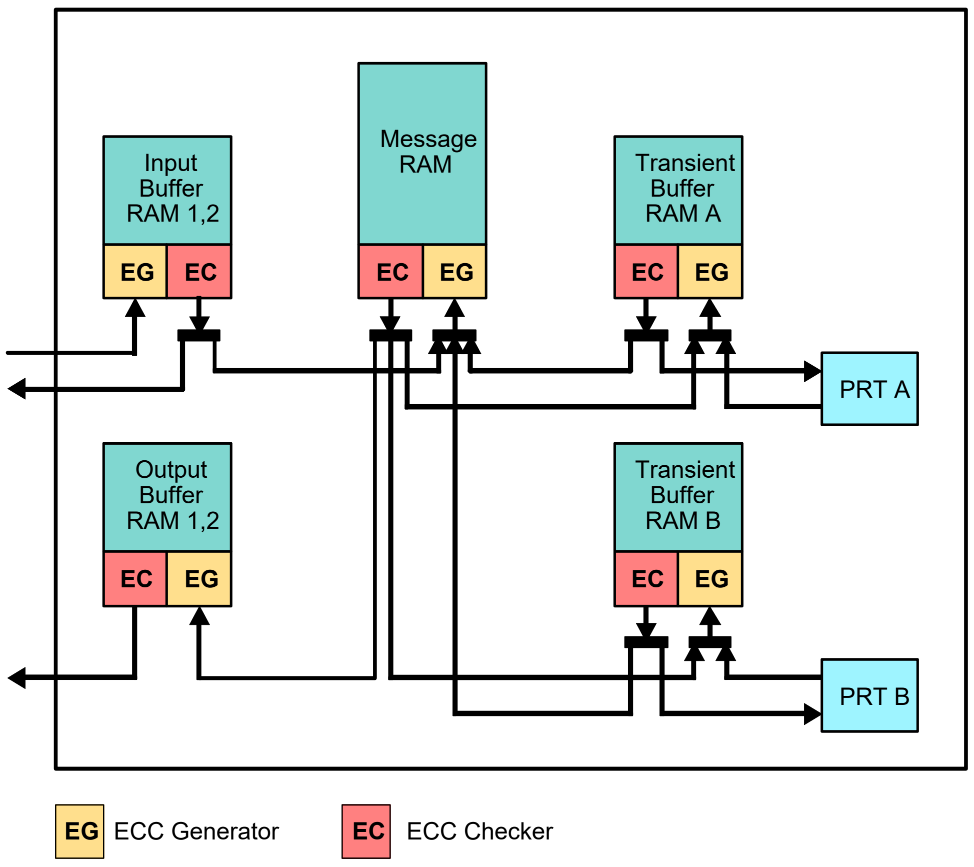

Figure 631 ECC Generation and Check

When an ECC error has been detected the following actions will be performed:

In all cases

- The respective ECC error flag in the Message Handler Status MHDS register is set
- The ECC error flag EIR.EERR in the Error Service Request Register is set, and if enabled, a module service request to the Host will be generated.

Additionally in specific cases

1. ECC error in data transfer from Input Buffer RAM 1,2 ⇒ Message RAM (Transfer of Header and Data Section)
   a) MHDS.EIBF bit is set
   b) MHDS.FMBD bit is set to indicate that MHDS.FMB has been updated
   c) MHDS.FMB indicates the number of the faulty Message Buffer
   d) Transmit buffer: Transmission request for the respective Message Buffer is not set
2. ECC error in data transfer from Input Buffer RAM 1,2 ⇒ Message RAM (Transfer of Data Section only)
   a) MHDS.EMR bit is set
   b) MHDS.FMBD bit is set to indicate that MHDS.FMB points to a faulty Message Buffer

   c) MHDS.FMB indicates the number of the faulty Message Buffer
   d) The Data Section of the respective Message Buffer is not updated
   e) Transmit buffer: Transmission request for the respective Message Buffer is not set
3. ECC error during host reading Input Buffer RAM
   a) • MHDS.EIBF bit is set
4. ECC error during scan of Header Sections in Message RAM
   a) MHDS.EMR bit is set
   b) MHDS.FMBD bit is set to indicate that MHDS.FMB points to a faulty Message Buffer
   c) MHDS.FMB indicates the number of the faulty Message Buffer
   d) Ignore Message Buffer (Message Buffer is skipped)
5. ECC error during data transfer from Message RAM $\Rightarrow$ Transient Buffer RAM A, B
   a) MHDS.EMR bit is set
   b) MHDS.FMBD bit is set to indicate that MHDS.FMB points to a faulty Message Buffer
   c) MHDS.FMB indicates the number of the faulty Message Buffer
   d) Frame not transmitted, Frames already in transmission are invalidated by setting the Frame CRC to zero
6. ECC error during data transfer from Transient Buffer RAM A, B $\Rightarrow$ Protocol Controller 1, 2
   a) MHDS.ETBF1, MHDS.ETBF2 bit is set
7. ECC error in data transfer from Transient Buffer RAM A, B $\Rightarrow$ Message RAM
   (ECC error when reading Header Section of respective Message Buffer from Message RAM)
   a) MHDS.EMR bit is set
   b) MHDS.FMBD bit is set to indicate that MHDS.FMB points to a faulty Message Buffer
   c) MHDS.FMB indicates the number of the faulty Message Buffer
   d) The Data Section of the respective Message Buffer is not updated
8. ECC error in data transfer from Transient Buffer RAM A, B $\Rightarrow$ Message RAM
   (ECC error when reading Transient Buffer RAM A, B)
   a) MHDS.ETBF1, MHDS.ETBF2 bit is set
   b) MHDS.FMBD bit is set to indicate that MHDS.FMB points to a faulty Message Buffer
   c) MHDS.FMB indicates the number of the faulty Message Buffer
9. ECC error during data transfer from Message RAM $\Rightarrow$ Output Buffer RAM
   a) MHDS.EMR bit is set
   b) MHDS.FMBD bit is set to indicate that MHDS.FMB points to a faulty Message Buffer
   c) MHDS.FMB indicates the number of the faulty Message Buffer
10. ECC error during Host reading Output Buffer RAM
   a) • MHDS.EOBF bit is set
11. ECC error during data read of Transient Buffer RAM A, B

If an ECC error occurs while the Message Handler reads a Frame with Network Management information (PPI = 1) from the Transient Buffer RAM A, B the corresponding Network Management vector registers NMV1 to NMV3 are not updated from that Frame.

41.3.14 Host Handling of Errors

An ECC error caused by transient bit flips can be fixed by:

41.3.14.1 Self-Healing

ECC errors located in

- Input Buffer RAM 1,2
- Output Buffer RAM 1,2
- Data Section of Message RAM
- Transient Buffer RAM A
- Transient Buffer RAM B

are overwritten with the next write access to the disturbed bit(s) caused by Host access or by FlexRay communication.

41.3.14.2 CLEAR_RAMS Command

After Power-On Reset, the E-Ray RAMs hold arbitrary values which causes ECC errors (MHDS) when a read operation is performed in a E-Ray RAM location. Hence the ERAY RAMs should be initialized always after a Power-On reset. When called in DEFAULT_CONFIG or CONFIG state POC command CLEAR_RAMS initializes all module-internal RAMs to zero. Inorder to perform a safe initialization of RAM blocks, the following programming sequence is suggested:

1. Remove EINIT protection for the writing of the CLC register
2. Enable the clock in the CLC register
3. Read the CLC register back to ensure that the clocks are enabled
4. Enable the EINIT protection
5. Enable the test mode. Take care of the unlock sequence. Refer to description of LCK.TMK and TEST1.WRTEN
6. Check if CCSV.POCS is either 0x0 (DEFAULT_CONFIG) or 0xF (CONFIG). If not in any of these states, perform the according command to get to CONFIG state
7. Check if SUCC1.PBSY is equal to 0x0. If 0x1 wait until 0x0.
8. Set SUCC1.CMD to 0xC meaning that the CLEAR_RAMS command is entered
9. Read SUCC1.CMD. If 0x0 the command has not been accepted. Repeat up from step 7. Otherwise continue.
10. Wait for 1024 module cycles.
11. Enable RAM Test mode: TEST1.TMC = 01b. This mode enables access of all RAM blocks in E-Ray module to host.
12. CUST1.IBF1PAG = 1b
13. CUST1.IBF2PAG = 1b
14. Repeat steps 7 to 10 and then proceed to step 15
15. Read atleast one address in all the E-Ray RAM blocks
16. Switch off Test Mode: TEST1.WRTEN = 0b
17. Clear ECC error flags in MHDS and EIR registers

41.3.14.3 Temporary Unlocking of Header Section

An ECC error in the header section of a locked message buffer can be fixed by a transfer from the Input Buffer to the locked buffer Header Section. For this transfer, the write-access to the IBCR (specifying the message buffer number) must be immediately preceded by the unlock sequence normally used to leave CONFIG state (see “Lock Register”).For that single transfer the respective message buffer header is unlocked, regardless whether it belongs to the FIFO or whether its locking is controlled by MRC.SEC[1:0], and will be updated with new data.

41.3.15 Module Service Request

In general, service requests provide a close link to the protocol timing as they are triggered almost immediately when an error or status change is detected by the controller, a Frame is received or transmitted, a configured timer service request is activated, or a stop watch event occurred. This enables the Host to react very quickly on specific error conditions, status changes, or timer events. On the other hand too many service requests can cause the Host to miss deadlines required for the application. Therefore the Communication Controller supports disable / enable controls for each individual service request source separately.

An service request may be triggered when

- An error was detected
- A status flag is set
- A timer reaches a preconfigured value
- A message transfer from Input Buffer to Message RAM or from Message RAM to Output Buffer has completed
- A stop watch event occurred

Tracking status and generating service requests when a status change or an error occurs are two independent tasks. Regardless of whether an service request is enabled or not, the corresponding status is tracked and indicated by the Communication Controller. The Host has access to the actual status and error information by reading the Error Service Request Register EIR and the Status Service Request SIR Register.

Table 404 Module Service Request Flags and Service Request Line Enable

Register	Bit	Function
SIR	WST	Wakeup Status
	CAS	Collision Avoidance Symbol
	CYCS	Cycle Start Service Request
	TXI	Transmit Service Request
	RXI	Receive Service Request
	RFNE	Receive FIFO not Empty
	RFCL	Receive FIFO Critical Level
	NMVC	Network Management Vector Changed
	TI0	Timer Service Request 0
	TI1	Timer Service Request 1
	TIBC	Transfer Input Buffer Completed
	TOBC	Transfer Output Buffer Completed
	SWE	Stop Watch Event
	SUCS	Startup Completed Successfully
	MBSI	Message Buffer Status Interrupt
	SDS	Start of Dynamic Segment
	WUPA	Wakeup Pattern Channel A
	MTSA	MTS Received on Channel A
	WUPB	Wakeup Pattern Channel B
	MTSB	MTS Received on Channel B
ILE	EINT0	Enable Service Request Line 0
	EINT1	Enable Service Request Line 1

Table 404 Module Service Request Flags and Service Request Line Enable (cont'd)

Register	Bit	Function
EIR	PEMC	Protocol Error Mode Changed
	CNA	Command Not Valid
	SFBM	SYNC Frames Below Minimum
	SFO	SYNC Frame Overflow
	CCF	Clock Correction Failure
	CCL	CHI Command Locked
	EERR	ECC Error
	RFO	Receive FIFO Overrun
EIR	EFA	Empty FIFO Access
	IIBA	Illegal Input Buffer Access
	IOBA	Illegal Output Buffer Access
	MHF	Message Handler Constraints Flag
	EDA	Error Detected on Channel A
	LTVA	Latest Transmit Violation Channel A
	TABA	Transmission Across Boundary Channel A
	EDB	Error Detected on Channel B
	LTVB	Latest Transmit Violation Channel B
	TABB	Transmission Across Boundary Channel B

The interrupt lines to the Host INT0SR and INT1SR are controlled by the enabled interrupts. In addition each of the two interrupt lines can be enabled / disabled separately by programming bit ILE.EINT0/INT0SRC.SRE and ILE.EINT1/ INT1SRC.SRE.

The interrupt lines to the Host NDAT0SR and NDAT1SR are controlled by the enabled new data interrupts (NDIC1 to NDIC4). In addition each of the two interrupt lines can be enabled / disabled separately by programming bit NDAT0SRC.SRE and NDAT1SRC.SRE.

The interrupt lines to the Host MBSC0SR and MBSC1SR are controlled by the enabled message buffer status changed interrupts (MSIC1 to MSIC4). In addition each of the two interrupt lines can be enabled / disabled separately by programming bit MBSC0SRC.SRE and MBSC1SRC.SRE.

The two timer service requests generated by service request timer 0 and 1 are available on pins TINT0SR and TINT1SR. They can be configured via the Timer 0 and Timer 1 Configuration register. In addition each of the two interrupt lines can be enabled / disabled separately by programming bit TINT0SRC.SRE and TINT1SRC.SRE.

A stop watch event may be triggered via input pin **STPWn**.

The status of the data transfer between IBF / OBF and the Message RAM is signalled on signals IBUSY and OBUSY. When a transfer has completed bit SIR.TIBC or SIR.TOBC is set.

41.3.16 Restrictions

The following restrictions have to be considered when programming the E-Ray IP-module. A violation of these restrictions may lead to an erroneous behavior of the E-Ray IP-module.

41.3.16.1 Message Buffers with the same Frame ID

If two or more Message Buffers are configured with the same Frame ID, and if they have a matching cycle counter filter value for the same slot, then the Message Buffer with the lowest Message Buffer number is used.

Sharing of a static time slot via cycle counter filtering between different nodes of a FlexRay™ network is **not** allowed.

41.3.16.2 Data Transfers between IBF / OBF and Message RAM

The time required to transfer the contents of a Message Buffer between IBF / OBF and Message RAM depends on the setup time to start the first transfer, the number of 4-byte words to be transferred, and the number of concurrent tasks to be managed by the Message Handler. The number of 4-byte words varies from 4 (Header Section only) to 68 (Header + maximum Data Section) while the number of concurrent task varies from one to three.

The following concurrent tasks are executed under control of the Message Handler:

- Data transfer between IBF or OBF and Message RAM
- Data transfer between TBF1 and Message RAM, search next TX / RX Message Buffer CHA
- Data transfer between TBF2 and Message RAM, search next TX / RX Message Buffer CHB

Transfers between IBF and Message RAM respectively Message RAM and OBF can only be handled one after another. In case that e.g. a transfer between IBF and Message RAM has been started shortly before a transfer between Message RAM and OBF is requested, the OBF transfer has to wait until the IBF transfer has completed.

The relative time is measured in f_{CLC_ERAY} cycles. Absolute time depends on the actual f_{CLC_ERAY} cycle period.

cyclestrans = (remaining cycles of transfer running) + (cycles of requested transfer)

cyclestrans = cyclesrem + cyclesreq

cyclesrem = (number of concurrent tasks) * (setup time + (number of 4-byte words)rem)

cyclesreq = (number of concurrent tasks) * (setup time + (number of 4-byte words)req)

setup time = 2 f_{CLC_ERAY} cycles

Under worst case conditions a transfer is requested directly after the previous transfer started:

Max. number of f_{CLC_ERAY} cycles: cyclestrans = (3 * (2 + 68)) + (3 * (2 + 68)) = 420

Worst case timing: timetrans(40MHz) = 420 * 25ns = 10.5 µs

41.3.17 E-Ray Module Implementation

This section describes the E-Ray interfaces as implemented in AURIX™ TC3xx Platform with the clock control, port and DMA connections, interrupt control, and address decoding.

Figure below shows a detailed view of the E-Ray interface.

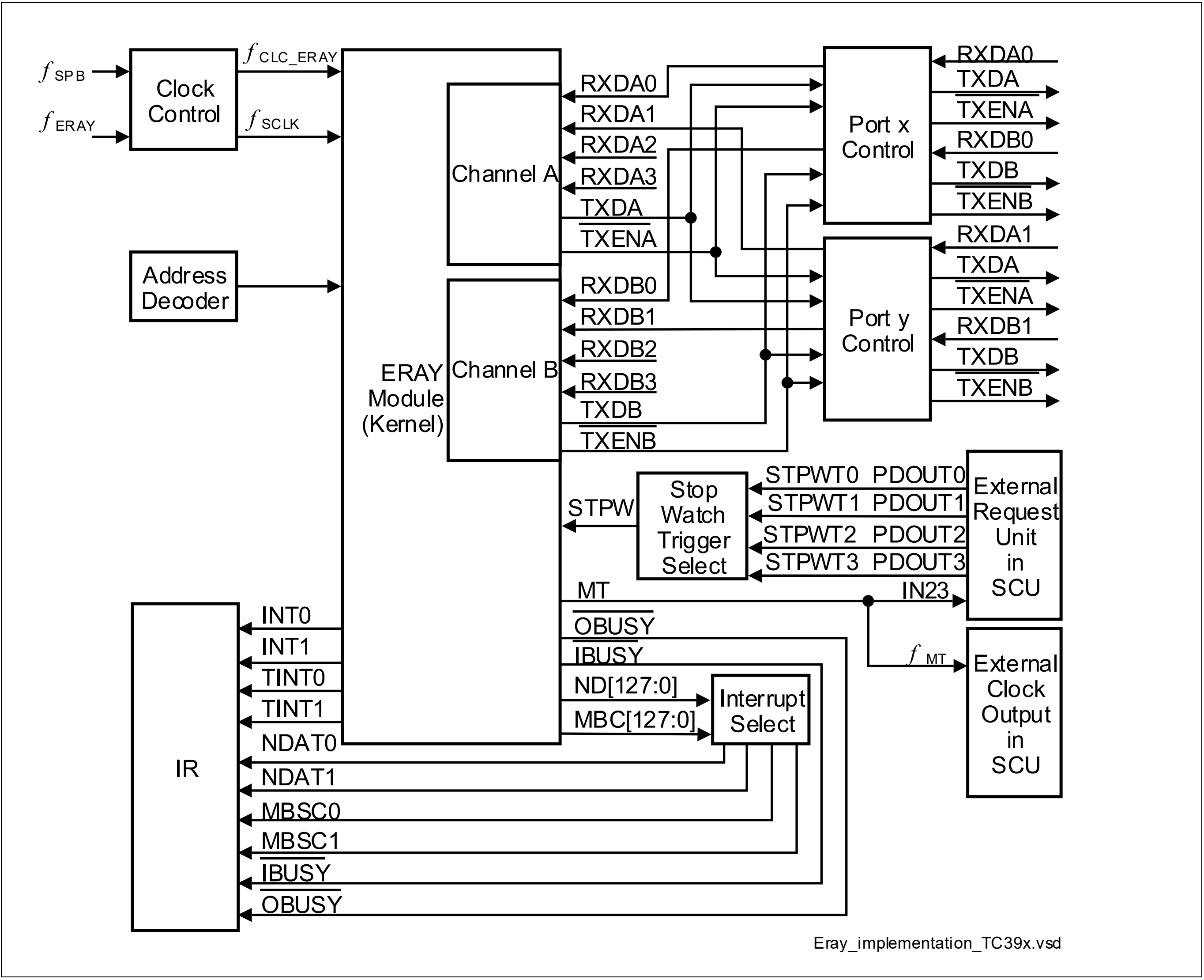

Figure 632 Detailed Block Diagram of the E-Ray Interface

41.3.17.1 Interconnections of the E-Ray Module

The E-Ray module has 2 FlexRay™ communication channels, channel A and channel B. Each channel provides a set of signals to drive a bus driver. The E-Ray module requires two different clocks, a sampling clock of the FlexRay™ bus f_{SCLK}. f_{SCLK} has to be 8 times the baud rate of the FlexRay™ communication. A second clock f_{CLC_ERAY} is used for the main protocol controller state machine and the customer interface logic. To enable deactivation of the E-Ray Module, f_{CLC_ERAY} and f_{SCLK} may be disabled (clock gated) by the CLC.DISR Enable E-Ray (Clock Gating) bit.The following items are described in this section:

- E-Ray module (kernel) external registers
- Port control and connections
  - I/O port line assignment
  - I/O function selection

  - Pad driver characteristics selection
- On-chip connections
  - SCU Connections
  - DMA connections
- Module clock generation
- Interrupt registers
- E-Ray address map

41.3.17.2 Port Control and Connections

This section describes the I/O connections of the E-Ray module.

41.3.17.2.1 Input/Output Function Selection

Table below shows how bits and bit fields must be programmed for the required I/O functionality of the E-Ray I/O lines. This table also shows the values of the peripheral input select registers.

Table 405 E-Ray I/O Control Selection and Setup

Port Lines	Input Select Register	Input/Output Control Register Bits	I/O
FlexRay™ Channel 0A			
RXD0A0/ P14.8	ERAY_CUST1.RISA = 00_B	P14_IOCR8.PC8 = $0XXXX_B$	In
RXD0A1/ P11.9	ERAY_CUST1.RISA = 01_B	P11_IOCR8.PC9 = $0XXXX_B$	In
RXD0A2/ P02.1	ERAY_CUST1.RISA = 10_B	P02_IOCR0.PC1 = $0XXXX_B$	In
RXD0A3/ P14.1	ERAY_CUST1.RISA = 11_B	P14_IOCR0.PC1 = $0XXXX_B$	In
TXD0A/ P02.0	not applicable	P02_IOCR0.PC0 = $1X110_B$	Out
TXD0A/ P11.3	not applicable	P11_IOCR0.PC3 = $1X100_B$	Out
TXD0A/ P14.10	not applicable	P14_IOCR8.PC10 = $1X110_B$	Out
TXD0A/ P14.0	not applicable	P14_IOCR0.PC0 = $1X011_B$	Out
$\overline{\text{TXEN0A}}$/ P02.4	not applicable	P2_IOCR4.PC4 = $1X110_B$	Out
$\overline{\text{TXEN0A}}$/ P11.6	not applicable	P11_IOCR4.PC6 = $1X100_B$	Out
$\overline{\text{TXEN0A}}$/ P14.9	not applicable	P14_IOCR8.PC9 = $1X110_B$	Out
FlexRay™ Channel 0B			
RXD0B0/ P14.7	ERAY_CUST1.RISB = 00_B	P14_IOCR4.PC7 = $0XXXX_B$	In
RXD0B1/ P11.10	ERAY_CUST1.RISB = 01_B	P11_IOCR8.PC10 = $0XXXX_B$	In
RXD0B2/ P02.3	ERAY_CUST1.RISB = 10_B	P02_IOCR0.PC3 = $0XXXX_B$	In
RXD0B3/ P14.1	ERAY_CUST1.RISB = 11_B	P14_IOCR0.PC1 = $0XXXX_B$	In
TXD0B/ P02.2	not applicable	P02_IOCR0.PC2 = $1X110_B$	Out
TXD0B/ P14.0	not applicable	P14_IOCR0.PC0 = $1X100_B$	Out
TXD0B/ P14.5	not applicable	P14_IOCR4.PC5 = $1X110_B$	Out
TXD0B/ P11.12	not applicable	P11_IOCR12.PC12 = $1X100_B$	Out
$\overline{\text{TXEN0B}}$/ P02.5	not applicable	P2_IOCR4.PC5 = $1X110_B$	Out
$\overline{\text{TXEN0B}}$/ P14.6	not applicable	P14_IOCR4.PC6 = $1X110_B$	Out
$\overline{\text{TXEN0B}}$/ P14.9	not applicable	P14_IOCR8.PC9 = $1X101_B$	Out

Table 405 E-Ray I/O Control Selection and Setup (cont'd)

Port Lines	Input Select Register	Input/Output Control Register Bits	I/O
$\overline{\text{TXEN0B}}$/ P11.11	not applicable	P11_IOCR8.PC11 = $1X110_B$	Out
$\overline{\text{TXEN0B}}$/ P11.6	not applicable	P11_IOCR4.PC6 = $1X010_B$	Out
FlexRay™ Channel 1A			
RXD1A0/ P14.8	ERAY_CUST1.RISA = 00_B	P14_IOCR8.PC8 = $0XXXX_B$	In
RXD1A1/ P01.1	ERAY_CUST1.RISA = 01_B	P01_IOCR0.PC1 = $0XXXX_B$	In
RXD1A2/ NC	ERAY_CUST1.RISA = 10_B		In
RXD1A3/ NC	ERAY_CUST1.RISA = 11_B		In
TXD1A/ P01.12	not applicable	P01_IOCR12.PC12 = $1X110_B$	Out
TXD1A/ P14.10	not applicable	P14_IOCR8.PC10 = $1X111_B$	Out
$\overline{\text{TXEN1A}}$/ P01.14	not applicable	P1_IOCR12.PC14 = $1X110_B$	Out
$\overline{\text{TXEN1A}}$/ P14.9	not applicable	P14_IOCR8.PC9 = $1X111_B$	Out
FlexRay™ Channel 1B			
RXD1B0/ P14.7	ERAY_CUST1.RISB = 00_B	P14_IOCR4.PC7 = $0XXXX_B$	In
RXD1B1/ P01.8	ERAY_CUST1.RISB = 01_B	P01_IOCR8.PC8 = $0XXXX_B$	In
RXD1B2/ NC	ERAY_CUST1.RISB = 10_B		In
RXD1B3/ NC	ERAY_CUST1.RISB = 11_B		In
TXD1B/ P01.13	not applicable	P01_IOCR12.PC13 = $1X110_B$	Out
TXD1B/ P14.5	not applicable	P14_IOCR4.PC5 = $1X111_B$	Out
$\overline{\text{TXEN1B}}$/ P02.15	not applicable	P2_IOCR12.PC15 = $1X110_B$	Out
$\overline{\text{TXEN1B}}$/ P14.6	not applicable	P14_IOCR4.PC6 = $1X111_B$	Out

41.3.17.3 On-Chip Connections

This section describes all on-chip interconnections of the E-Ray modules except the connections to I/O ports.

41.3.17.3.1 E-Ray Connections with IR

The E-Ray module of the AURIX™ TC3xx Platform has several on-chip interconnections to the IR. Table below shows these interconnections. These enable the IR to handle different service request of E-Ray module via the DMA or Interrupt Service Routine.

Table 406 Request Assignment for IR

Line #	ERAY Output Signal	IR Request Input Line
00	INT0	SRC_ERAY0INT0
01	INT1	SRC_ERAY0INT1
02	TINT0	SRC_ERAY0TINT0
03	TINT1	SRC_ERAY0TINT1
04	NDAT0	SRC_ERAY0NDAT0
05	NDAT1	SRC_ERAY0NDAT1
06	MBSC0	SRC_ERAY0MBSC0
07	MBSC1	SRC_ERAY0MBSC1
08	OBUSY	SRC_ERAY0OBUSY
09	IBUSY	SRC_ERAY0IBUSY
10	INT0	SRC_ERAY1INT0
11	INT1	SRC_ERAY1INT1
12	TINT0	SRC_ERAY1TINT0
13	TINT1	SRC_ERAY1TINT1
14	NDAT0	SRC_ERAY1NDAT0
15	NDAT1	SRC_ERAY1NDAT1
16	MBSC0	SRC_ERAY1MBSC0
17	MBSC1	SRC_ERAY1MBSC1
18	OBUSY	SRC_ERAY1OBUSY
19	IBUSY	SRC_ERAY1IBUSY

41.3.17.3.2 E-Ray Connections with SMU

The E-Ray module of the AURIX™ TC3xx Platform provides to the SMU the following alarms (ALMx): SRAM ECC single bit correction, SRAM ECC uncorrectable error, SRAM address error, SRAM buffer address error.

41.3.17.3.3 E-Ray Connections with the External Request Unit of SCU

The E-Ray module of the AURIX™ TC3xx Platform has several on-chip interconnections to the External Request Unit (ERU) in the SCU to externally trigger stop watch events and to provide a global time e.g. to the on chip timers. Table 24-29 and Table below show these interconnections.

STPWT0_ are the inputs of ERAY0, STPWT1_ of ERAY1.

Table 407 External Stop Watch Request Assignment

ERAY Input Signal	ERU Request Output Line	Selected by
STPWT0_0/1_0	ERU_PDOUT0	CUST1.STPWTS = 00_B
STPWT0_1/1_1	ERU_PDOUT1	CUST1.STPWTS = 01_B
STPWT0_2/1_2	ERU_PDOUT2	CUST1.STPWTS = 10_B
STPWT0_3/1_3	ERU_PDOUT3	CUST1.STPWTS = 11_B

Table 408 Global Macrotick Connection to ERU

ERAY Output Signal	ERU Request Input Line	Selected by
MT0 (from ERAY0)	ERU_IN23	ERU_EICR1.EXIS0 = 11_B
MT1 (from ERAY1)	ERU_IN73	ERU_EICR3.EXIS1 = 11_B

41.3.17.3.4 E-Ray Connections to GTM

The E-Ray module of the AURIX™ TC3xx Platform has several on-chip interconnections to the Generic Timer Module (GTM). Table below show these interconnection s.

Table 409 Global Macrotick Connection to GTM

ERAY Output Signal	TIM Input Line
MT0	TIM0_7
MT0	TIM1_7
MT0	TIM2_7
MT0	TIM3_7
MT1	TIM4_7
MT1	TIM5_7

41.3.17.3.5 E-Ray Connections with the External Clock Output of SCU

The E-Ray module of the AURIX™ TC3xx Platform has one on-chip interconnections to the External Clock Output Unit in the SCU to distribute externally as also internally the Macro Tick as time base for distributed system control. Table below shows this interconnection.

Table 410 Global Macrotick Connection to External Clock Output

ERAY Output Signal	External Clock Output	Selected by
MT0 (from ERAY0)	f_{MT0}	SCU_EXTCON.SEL0 = 1111_B

41.3.17.4 OCDS Trigger Bus (OTGB) Interface

The E-Ray module has two 16 bit and one 32 bit Trigger Sets (**Table 411**) which are selected with the OTSS register.

Table 411 E-Ray Trigger Sets

Trigger Set	Details
TS16_SEP Service Requests, Errors and POC State	**Table 413**

Table 411 E-Ray Trigger Sets (cont'd)

Trigger Set	Details
TS16_MC Macrotick Counter	Table 414
TS32_SCSC State, Cycle and Slot Counter	Table 415

Table 412 shows all possible Trigger Set mapping options. If OTGB0 and/or OTGB1 is not used for E-Ray, Trigger Sets of other sources can be added in the OTGM module.

Table 412 Trigger Set Mapping Options

Width	OTGB0	OTGB1	OTGB2
32 Bit	TS16_SEP/MC		
		TS16_SEP/MC	
	TS16_SEP/MC	TS16_SEP/MC	
64 Bit			TS32_SCSC
	TS16_SEP/MC		TS32_SCSC
		TS16_SEP/MC	TS32_SCSC
	TS16_SEP/MC	TS16_SEP/MC	TS32_SCSC

Table 413 TS16_SEP Service Requests, Errors and POC State

Bits	Description
0	Interrupt 0 Service Request (INT0SRC)
1	Interrupt 1 Service Request (INT1SRC)
2	Timer Interrupt 0 Service Request (TINT0SRC)
3	Timer Interrupt 1 Service Request (TINT1SRC)
4	New Data 0 Service Request (NDAT0SRC)
5	New Data 1 Service Request (NDAT1SRC)
6	Message Buffer Status Changed 0 Service Request (MBSC0SRC)
7	Message Buffer Status Changed 1 Service Request (MBSC1SRC)
8	Output Buffer Busy Service Request (OBUSYSRC)
9	Input Buffer Busy Service Request (IBUSYSRC)
10	Reserved
11	Error on Channel A (EIR.EDA)
12	Error on Channel B (EIR.EDB)
[15:13]	POC State bits[2:0] (CCSV.POCS)

Table 414 TS16_MC Macrotick Counter

Bits	Description
[13:0]	Macrotick Value (MTCCV.MTV)
[15:14]	Reserved

Table 415 TS32_SCSC State, Cycle and Slot Counter

Bits	Description
[10:0]	Slot Counter Channel A (SCV.SCCA)
[22:12]	Slot Counter Channel B (SCV.SCCB)
[29:24]	Cycle Counter Value (MTCCV.CCV)
30	Transfer Input Buffer Completed (SIR.TIBC)
31	Transfer Output Buffer Completed (SIR.TOBC)
11,23	Reserved

TS32_SCSC becomes valid (posedge on otgb2_valid_o) on a change of any Slot Counter or Transfer Buffer state. This covers also the Cycle Counter which will always change with a slower rate.

41.3.17.5 OTGB E-Ray Registers

41.3.17.5.1 OCDS Trigger Bus (OTGB)

The OTGB control register is cleared by Debug Reset. Write access is 32 bit wide only and requires Supervisor Mode.

OCDS Trigger Set Select

OTSS
OCDS Trigger Set Select **(0870$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															**OTGB 2**
r															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0						**OTGB1**		**0**						**OTGB0**	
r						rw		r						rw	

Field	Bits	Type	Description
OTGB0	1:0	rw	**Trigger Set for OTGB0** 00$_B$ No Trigger Set selected 01$_B$ Trigger Set TS16_SEP 10$_B$ Trigger Set TS16_MC 11$_B$ reserved
OTGB1	9:8	rw	**Trigger Set for OTGB1** 00$_B$ No Trigger Set selected 01$_B$ Trigger Set TS16_SEP 10$_B$ Trigger Set TS16_MC 11$_B$ reserved
OTGB2	16	rw	**Trigger Set for OTGB2** 0$_B$ No Trigger Set selected 1$_B$ Trigger Set TS32_SCSC
0	7:2, 15:10, 31:17	r	**Reserved** Read as 0; must be written with 0.

41.3.17.6 BPI_FPI Module Registers

Note: *Register bits marked “r” in the following register description are virtual registers and do not contain flip-flops. They are always read as 0.*

Clock Control Register

The Clock Control Register CLC allows the programmer to adapt the functionality and power consumption of the module to the requirements of the application. The description below shows the clock control register functionality which is implemented in the BPI_FPI for the module. Where a module kernel is connected to the CLC clock control interface, CLC controls the module clock, sleep mode and disable mode for the module.

Note: *The number of module clock cycles (wait states) which are required by the kernel to execute a read or write access depends on the selected CLC clock frequency, which is selected via bit field RMC in the CLC register. Therefore, increasing CLC.RMC may result in a longer FPI Bus read cycle access time for kernel registers and can also slow down the write throughput to the kernel registers.*

CLC
Clock Control Register **(0000_H)** **Application Reset Value: $0000\ 0003_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					**RMC**			**0**				**EDIS**	**0**	**DISS**	**DISR**
r					rw			r				rw	r	rh	rw

Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module. Note: *This bit disables the kernel clocks f_{CLC_ERAY} and the sampling clock f_{SCLK}.*
DISS	1	rh	**Module Disable Status Bit** Bit indicates the current status of the module.
EDIS	3	rw	**External Sleep Mode Request Disable Bit** Used to control module’s sleep mode. Note: *If this bit is cleared the kernel clock f_{CLC_ERAY} and the sampling clock f_{SCLK} are disabled during System Sleep Mode.*

Field	Bits	Type	Description
RMC	10:8	rw	**Clock Divider in Run Mode** *Note: This bit field is not affected by an application reset.* *Note: This bit field only controls the kernel clock f_{CLC_ERAY} and not the sampling clock f_{SCLK}.* 000_B No clock signal f_{CLC_ERAY} generated (default after reset) 001_B Clock $f_{CLC\_ERAY} = f_{SPB}$ selected 010_B Clock $f_{CLC\_ERAY} = f_{SPB}$ /2 selected 011_B reserved, do not use! 100_B Clock $f_{CLC\_ERAY} = f_{SPB}$ /4 selected 101_B reserved, do not use! … 111_B reserved, do not use!
0	2, 7:4, 15:11, 31:16	r	**Reserved** Read as 0; should be written with 0.

OCDS Control and Status

The OCDS Control and Status (OCS) register is cleared by Debug Reset. The OCS register can only be written when the OCDS is enabled. If OCDS is being disabled, the OCS register value will not change. When OCDS is disabled the OCS suspend control is ineffective.

OCS

OCDS Control and Status **($08E8_H$)** **Debug Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		**SUSSTA**	**SUS_P**	**SUS**				**0**							
r		rh	w	rw				r							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															
r															

Field	Bits	Type	Description
SUS	27:24	rw	**OCDS Suspend Control** Controls the sensitivity to the suspend signal coming from the OCDS Trigger Switch (OTGS) 0_H Will not suspend 1_H Hard suspend. Clocks f_{CLC_ERAY} and the sampling clock f_{SCLK} are switched off immediately. No read or write access to any registers. 2_H Soft suspend. This bit forces the module into freeze state. **others**, reserved

Field	Bits	Type	Description
SUS_P	28	w	**SUS Write Protection** SUS is only written when SUS_P is 1, otherwise unchanged. Read as 0.
SUSSTA	29	rh	**Suspend State** 0_B Module is not (yet) suspended 1_B Module is suspended
0	23:0, 31:30	r	**Reserved** Read as 0; must be written with 0.

Access Enable Register 0

The Access Enable Register 0 controls write access. [1]for transactions with the on chip bus master TAG ID 000000B to 011111B (see On Chip Bus Master TAG Assignments Chapter). The BPI_FPI is prepared for a 6-bit TAG ID. The registers ACCEN0 / ACCEN1 are providing one enable bit for each possible 6-bit TAG ID encoding.

Mapping of TAG IDs to ACCEN0.ENx: EN0 -> TAG ID 000000B, EN1 -> TAG ID 000001B, … , EN31 -> TAG ID 011111B.

ACCEN0

Access Enable Register 0 **($08FC_H$)** **Application Reset Value: FFFF $FFFF_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN31	**EN30**	**EN29**	**EN28**	**EN27**	**EN26**	**EN25**	**EN24**	**EN23**	**EN22**	**EN21**	**EN20**	**EN19**	**EN18**	**EN17**	**EN16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN15	**EN14**	**EN13**	**EN12**	**EN11**	**EN10**	**EN9**	**EN8**	**EN7**	**EN6**	**EN5**	**EN4**	**EN3**	**EN2**	**EN1**	**EN0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ENn (n=0-31)	n	rw	**Access Enable for Master TAG ID n** This bit enables write access to the module kernel addresses for transactions with the Master TAG ID n 0_B Write access will not be executed 1_B Write access will be executed

Kernel Reset Register 0

The Kernel Reset Register 0 is used to reset the related module kernel. Kernel registers related to the Debug Reset (Class 1) are not influenced. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset Registers.The RST bit will be re-set by the BPI with the end of the BPI kernel reset sequence.

Kernel Reset Register 0 includes a kernel reset status bit that is set to ´1´ by the BPI_FPI in the same clock cycle the RST bit is re-set by the BPI_FPI. This bit can be used to detect that a kernel reset was processed. The bit can be re-set to ´0´ by writing ´1´ to the KRSTCLR.CLR register bit.

Note: During the execution of the kernel reset until RSTSTAT is set, access to the kernel registers will result in an error acknowledge.

1) The BPI_FPI Access Enable functionality controls only write transactions to the CLC, OCS, KRSTx and the kernel registers. Read transactions are not influenced. SW has to take care for destructive/modifying read functionality in kernel registers

KRST0

Kernel Reset Register 0 **(08F4$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0														**RSTSTAT**	**RST**
r														rh	rwh

Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel registers are set. The RST bit will be cleared (re-set to ´0´) by the BPI_FPI after the kernel reset was executed. 0_B No kernel reset was requested 1_B A kernel reset was requested
RSTSTAT	1	rh	**Kernel Reset Status** This bit indicates wether a kernel reset was executed or not. This bit is set by the BPI_FPI after the execution of a kernel reset in the same clock cycle both reset bits. This bit can be cleared by writing with ´1´ to the CLR bit in the related KRSTCLR register. 0_B No kernel reset was executed 1_B Kernel reset was executed
0	31:2	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Register 1

The Kernel Reset Register 1 is used to reset the related module kernel. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset Registers (KRST1.RST and KRST0.RST) related to the module kernel that should be reset (kernel 0 or kernel 1). The RST bit will be re-set by the BPI with the end of the BPI kernel reset sequence.

KRST1

Kernel Reset Register 1 **(08F0$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															**RST**
r															rwh

Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel reset registers is set. The RST bit will be cleared (re-set to ´0´) by the BPI_FPI after the kernel reset was executed. 0_B No kernel reset was requested 1_B A kernel reset was requested
0	31:1	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Status Clear Register

The Kernel Reset Register Clear register is used to clear the Kernel Reset Status bit (KRST0.RSTSTAT).

KRSTCLR

Kernel Reset Status Clear Register **($08EC_H$)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															**CLR**
r															w

Field	Bits	Type	Description
CLR	0	w	**Kernel Reset Status Clear** Read always as 0. 0_B No action 1_B Clear Kernel Reset Status KRST0.RSTSTAT
0	31:1	r	**Reserved** Read as 0; should be written with 0.

41.3.17.7 Interrupt Registers

Two different type of Interrupt Registers are described within this chapter.
The Interrupt Control register enable the selection of the Service Request used to signal an event. The Interrupt Control registers NDIC1 to NDIC4 select the service request node used for New Data Events.The Interrupt Control registers MSIC1 to MSIC4 select the service request node used for Message Buffer Status Changed Events.
The Interrupt Service Request Control Registers control the eight service request nodes.

New Data Interrupt Control 1

This New Data Interrupt Control register controls the interrupt that becomes active (NDAT0SRC or NDAT1SRC) on a ND flag turning active of all configured Message Buffers 0 to Message Buffers 31.

NDIC1
New Data Interrupt Control 1 **($03A8_H$)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NDIP31	NDIP30	NDIP29	NDIP28	NDIP27	NDIP26	NDIP25	NDIP24	NDIP23	NDIP22	NDIP21	NDIP20	NDIP19	NDIP18	NDIP17	NDIP16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NDIP15	NDIP14	NDIP13	NDIP12	NDIP11	NDIP10	NDIP9	NDIP8	NDIP7	NDIP6	NDIP5	NDIP4	NDIP3	NDIP2	NDIP1	NDIP0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
NDIPn (n=0-31)	n	rw	**New Data Interrupt Pointer n (n = 0-31)** NDIPn determines the interrupt (NDAT0SRC or NDAT1SRC) of the service request output that becomes active on a New Data Flag becoming active. 0_B NDAT0SRC selected for New Data Service Request 1_B NDAT1SRC selected for New Data Service Request

New Data Interrupt Control 2

This New Data Interrupt Control register controls the interrupt that becomes active (NDAT0SRC or NDAT1SRC) on a ND flag turning active of all configured Message Buffers 32 to Message Buffers 63.

NDIC2
New Data Interrupt Control 2 **($03AC_H$)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NDIP63	NDIP62	NDIP61	NDIP60	NDIP59	NDIP58	NDIP57	NDIP56	NDIP55	NDIP54	NDIP53	NDIP52	NDIP51	NDIP50	NDIP49	NDIP48
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NDIP47	NDIP46	NDIP45	NDIP44	NDIP43	NDIP42	NDIP41	NDIP40	NDIP39	NDIP38	NDIP37	NDIP36	NDIP35	NDIP34	NDIP33	NDIP32
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
NDIPn (n=32-63)	n-32	rw	**New Data Interrupt Pointer n (n = 32-63)** NDIPn determines the interrupt (NDAT0SRC or NDAT1SRC) of the service request output that becomes active on a New Data Flag becoming active. 0_B NDAT0SRC selected for New Data Service Request 1_B NDAT1SRC selected for New Data Service Request

New Data Interrupt Control 3

This New Data Interrupt Control register controls the interrupt that becomes active (NDAT0SRC or NDAT1SRC) on a ND flag turning active of all configured Message Buffers 64 to Message Buffers 95.

NDIC3
New Data Interrupt Control 3 **(03B0$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NDIP95	**NDIP94**	**NDIP93**	**NDIP92**	**NDIP91**	**NDIP90**	**NDIP89**	**NDIP88**	**NDIP87**	**NDIP86**	**NDIP85**	**NDIP84**	**NDIP83**	**NDIP82**	**NDIP81**	**NDIP80**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NDIP79	**NDIP78**	**NDIP77**	**NDIP76**	**NDIP75**	**NDIP74**	**NDIP73**	**NDIP72**	**NDIP71**	**NDIP70**	**NDIP69**	**NDIP68**	**NDIP67**	**NDIP66**	**NDIP65**	**NDIP64**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
NDIPn (n=64-95)	n-64	rw	**New Data Interrupt Pointer n (n = 64-95)** NDIPn determines the interrupt (NDAT0SRC or NDAT1SRC) of the service request output that becomes active on a New Data Flag becoming active. 0_B NDAT0SRC selected for New Data Service Request 1_B NDAT1SRC selected for New Data Service Request

New Data Interrupt Control 4

This New Data Interrupt Control register controls the interrupt that becomes active (NDAT0SRC or NDAT1SRC) on a ND flag turning active of all configured Message Buffers 96 to Message Buffers 127.

NDIC4
New Data Interrupt Control 4 **(03B4$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NDIP127	**NDIP126**	**NDIP125**	**NDIP124**	**NDIP123**	**NDIP122**	**NDIP121**	**NDIP120**	**NDIP119**	**NDIP118**	**NDIP117**	**NDIP116**	**NDIP115**	**NDIP114**	**NDIP113**	**NDIP112**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NDIP111	**NDIP110**	**NDIP109**	**NDIP108**	**NDIP107**	**NDIP106**	**NDIP105**	**NDIP104**	**NDIP103**	**NDIP102**	**NDIP101**	**NDIP100**	**NDIP99**	**NDIP98**	**NDIP97**	**NDIP96**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
NDIPn (n=96-127)	n-96	rw	**New Data Interrupt Pointer n (n = 96-127)** NDIPn determines the interrupt (NDAT0SRC or NDAT1SRC) of the service request output that becomes active on a New Data Flag becoming active. 0_B NDAT0SRC selected for New Data Service Request 1_B NDAT1SRC selected for New Data Service Request

Message Buffer Status Changed Interrupt Control 1

This Message Buffer Status Change Interrupt Control register controls the interrupt that becomes active (MBSC0SRC or MBSC1SRC) on a MBC flag of all configured Message Buffer 0 to Message Buffer 31 turning active.

MSIC1

Message Buffer Status Changed Interrupt Control 1(03B8$_H$) **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSIP31	**MSIP30**	**MSIP29**	**MSIP28**	**MSIP27**	**MSIP26**	**MSIP25**	**MSIP24**	**MSIP23**	**MSIP22**	**MSIP21**	**MSIP20**	**MSIP19**	**MSIP18**	**MSIP17**	**MSIP16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSIP15	**MSIP14**	**MSIP13**	**MSIP12**	**MSIP11**	**MSIP10**	**MSIP9**	**MSIP8**	**MSIP7**	**MSIP6**	**MSIP5**	**MSIP4**	**MSIP3**	**MSIP2**	**MSIP1**	**MSIP0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
MSIPn (n=0-31)	n	rw	**Message Buffer Status Changed Interrupt Pointer n (n = 0-31)** MSIPn determines the interrupt (MBSC0SRC or MBSC1SRC) of the service request output that becomes active on a Message Buffer Status Changed Flag becoming active. 0$_B$ MBSC0SRC selected for Message Buffer Status Changed Service Request 1$_B$ MBSC1SRC selected for Message Buffer Status Changed Service Request

Message Buffer Status Changed Interrupt Control 2

This Message Buffer Status Change Interrupt Control register controls the interrupt that becomes active (MBSC0SRC or MBSC1SRC) on a MBC flag of all configured Message Buffer 32 to Message Buffer 63 turning active.

MSIC2

Message Buffer Status Changed Interrupt Control 2(03BC$_H$) **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSIP63	**MSIP62**	**MSIP61**	**MSIP60**	**MSIP59**	**MSIP58**	**MSIP57**	**MSIP56**	**MSIP55**	**MSIP54**	**MSIP53**	**MSIP52**	**MSIP51**	**MSIP50**	**MSIP49**	**MSIP48**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSIP47	**MSIP46**	**MSIP45**	**MSIP44**	**MSIP43**	**MSIP42**	**MSIP41**	**MSIP40**	**MSIP39**	**MSIP38**	**MSIP37**	**MSIP36**	**MSIP35**	**MSIP34**	**MSIP33**	**MSIP32**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
MSIPn (n=32-63)	n-32	rw	**Message Buffer Status Changed Interrupt Pointer n (n = 32-63)** MSIPn determines the interrupt (MBSC0SRC or MBSC1SRC) of the service request output that becomes active on a Message Buffer Status Changed Flag becoming active. 0_B MBSC0SRC selected for Message Buffer Status Changed Service Request 1_B MBSC1SRC selected for Message Buffer Status Changed Service Request

Message Buffer Status Changed Interrupt Control 3

This Message Buffer Status Change Interrupt Control register controls the interrupt that becomes active (MBSC0SRC or MBSC1SRC) on a MBC flag of all configured Message Buffer 64 to Message Buffer 95 turning active.

MSIC3
Message Buffer Status Changed Interrupt Control 3(03C0$_H$) **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSIP95	**MSIP94**	**MSIP93**	**MSIP92**	**MSIP91**	**MSIP90**	**MSIP89**	**MSIP88**	**MSIP87**	**MSIP86**	**MSIP85**	**MSIP84**	**MSIP83**	**MSIP82**	**MSIP81**	**MSIP80**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSIP79	**MSIP78**	**MSIP77**	**MSIP76**	**MSIP75**	**MSIP74**	**MSIP73**	**MSIP72**	**MSIP71**	**MSIP70**	**MSIP69**	**MSIP68**	**MSIP67**	**MSIP66**	**MSIP65**	**MSIP64**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
MSIPn (n=64-95)	n-64	rw	**Message Buffer Status Changed Interrupt Pointer n (n = 64-95)** MSIPn determines the interrupt (MBSC0SRC or MBSC1SRC) of the service request output that becomes active on a Message Buffer Status Changed Flag becoming active. 0_B MBSC0SRC selected for Message Buffer Status Changed Service Request 1_B MBSC1SRC selected for Message Buffer Status Changed Service Request

Message Buffer Status Changed Interrupt Control 4

This Message Buffer Status Change Interrupt Control register controls the interrupt that becomes active (MBSC0SRC or MBSC1SRC) on a MBC flag of all configured Message Buffer 96 to Message Buffer 127 turning active.