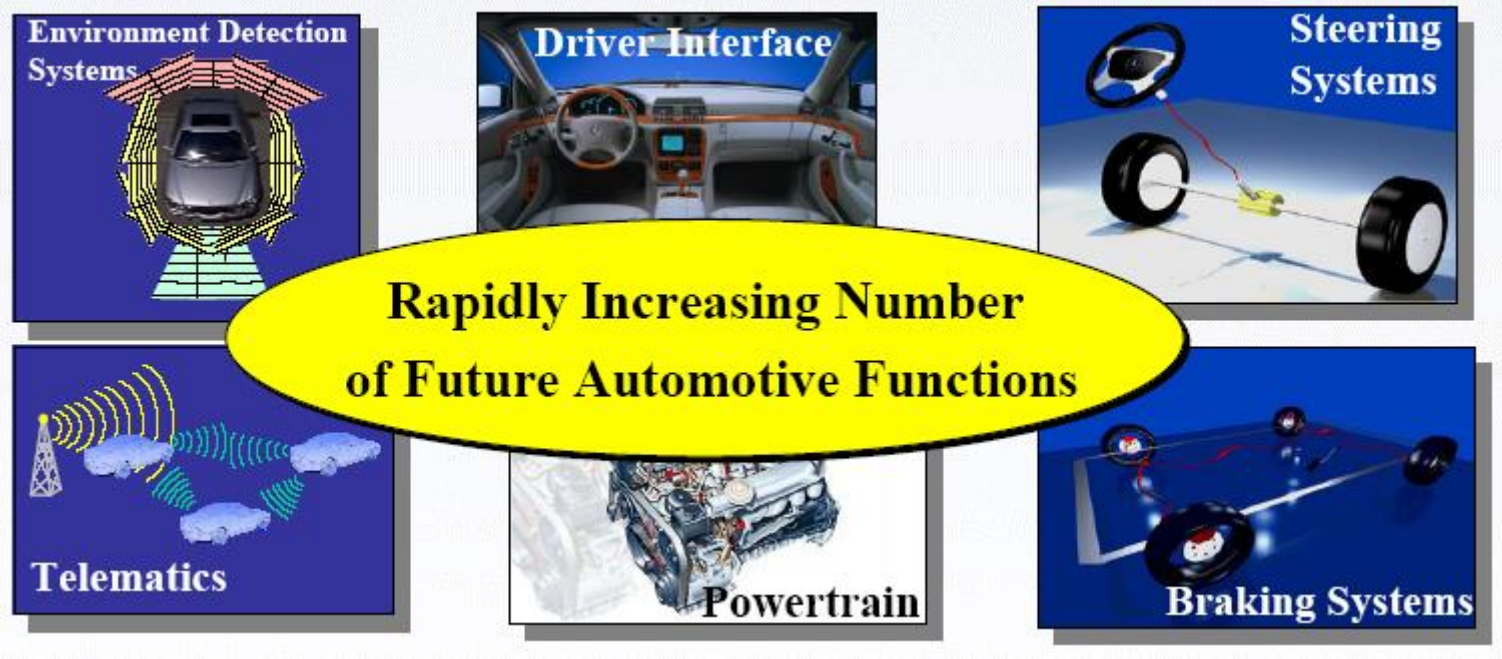
FlexRay Module Training

FlexRay Summary

FlexRay Overview	3
TI's FlexRay Implementation	27
FlexRay Transfer Unit (FTU)	45
FlexRay Interrupt Structure	59

FlexRay Overview

Future Needs for Networking



Future applications have additional requirements to a bus system

- dependability
- determinism
- robustness
- fault-tolerant

FlexRay Key Features

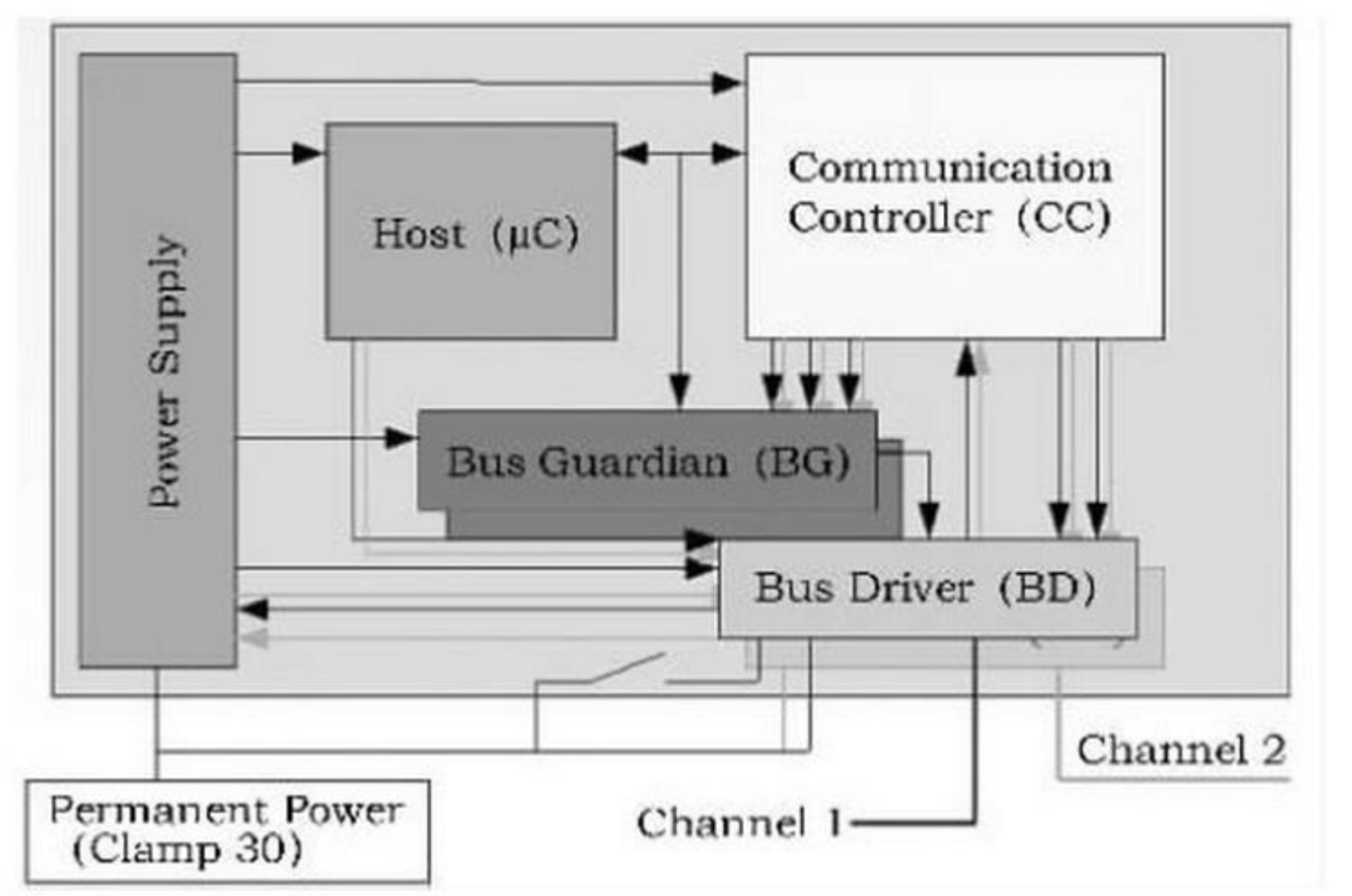
- Open Bus System
- Support of redundant transmission channels
- Data rate of 10 Mbit/sec per channel
- Support of a fault tolerant synchronized global time base
- Static and dynamic data transmission (scalable)
  - Deterministic data transmission
  - Arbitration free transmission
- Fault tolerant and time triggered services implemented in hardware
- Support of optical and electrical physical layers

FlexRay Protocol HardWare

Demo: There are 13 nodes in our network. 1 is TMS570LS3137, and others are TMS570LS20216

FlexRay Architecture -- ECU

- Flexray Node: ECU



FlexRay Architecture –Protocol Operation Control

FlexRay controller states:

- 1 – default config
- 2 – config
- 3 – ready
- 4 – wakeup
- 5 – startup
- 6 – normal active
- 7 – normal passive
- 8 – halt

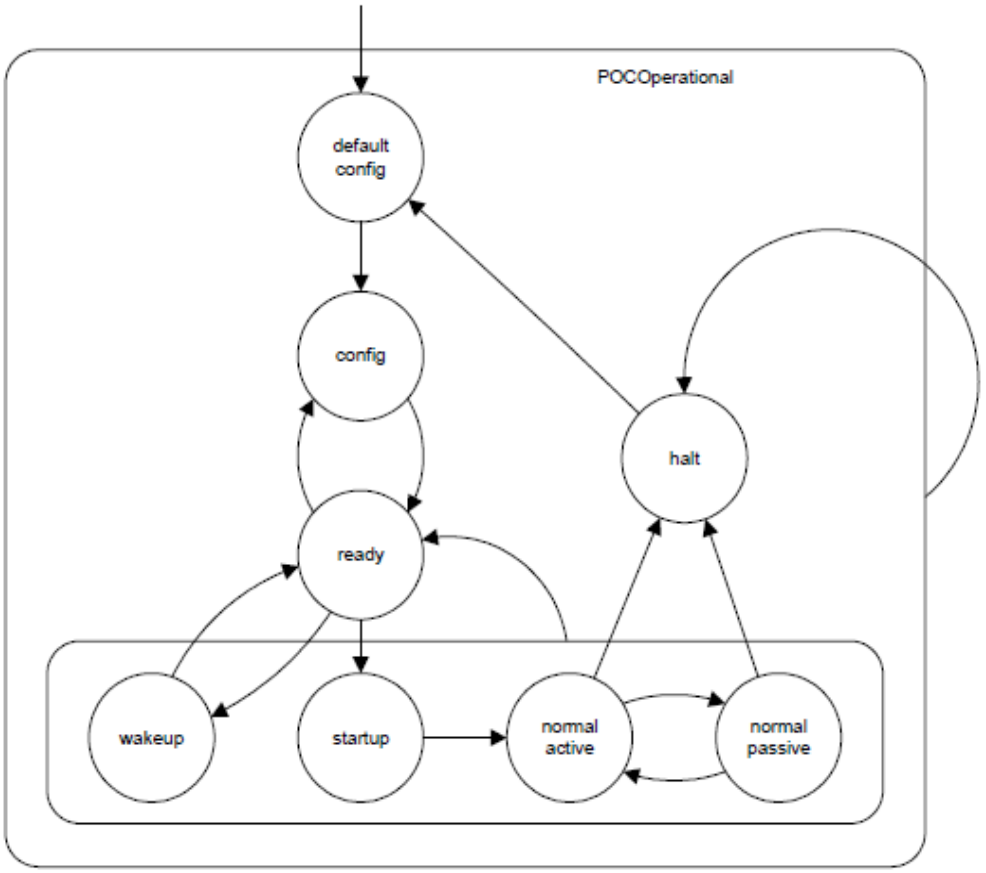


Figure 2-3: Overview of protocol operation control.

FlexRay Architecture – Network Topology

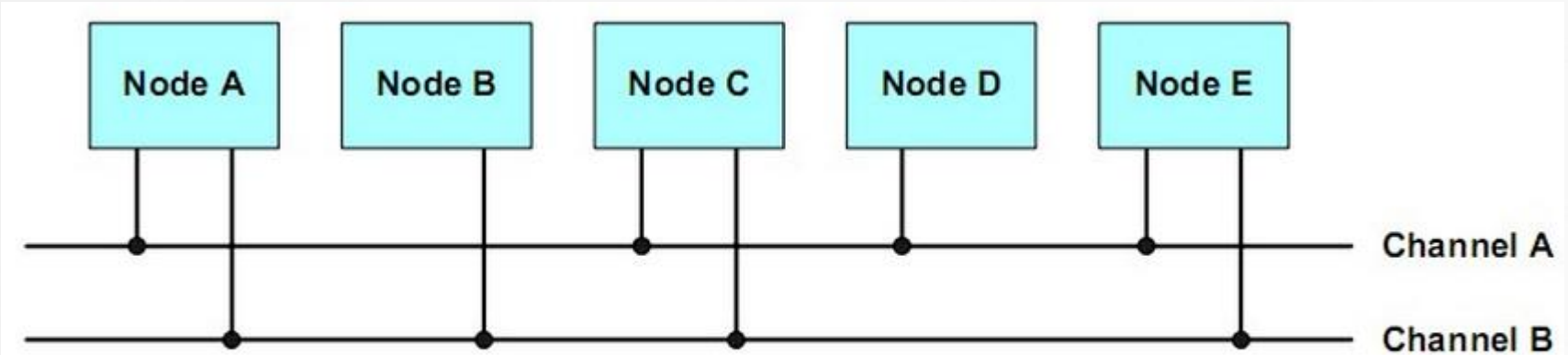


Figure 1-1: Dual channel bus configuration.

- A structure without rings and without active elements is called "linear passive bus".

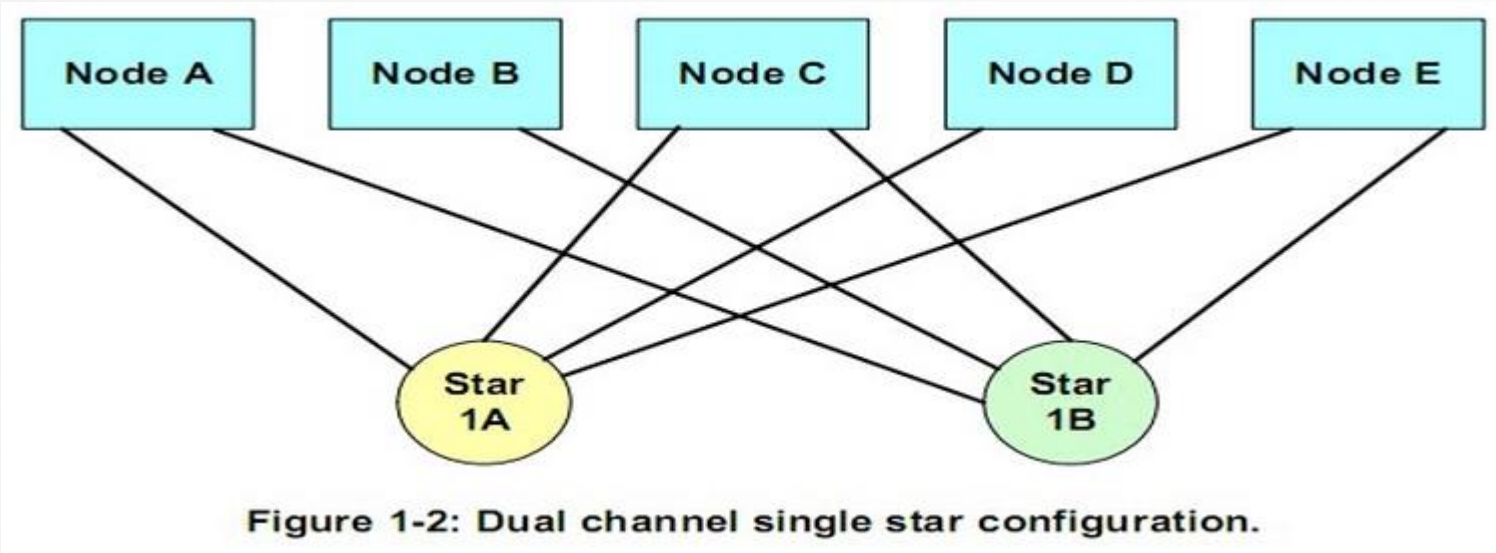


Figure 1-2: Dual channel single star configuration.

FlexRay Architecture – Network Topology

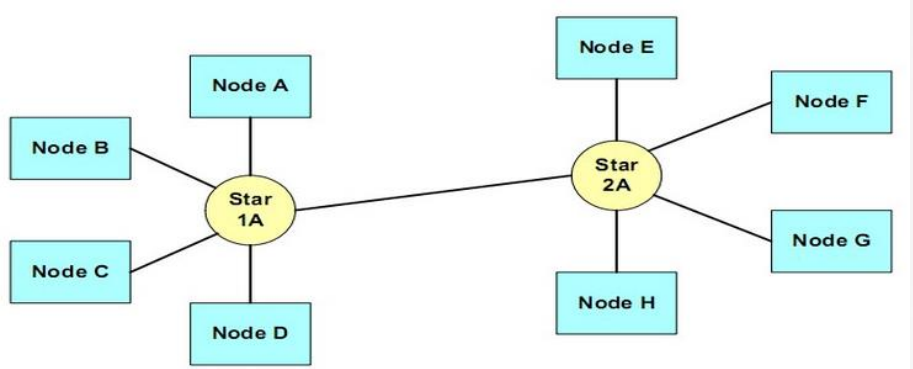


Figure 1-3: Single channel cascaded star configuration.

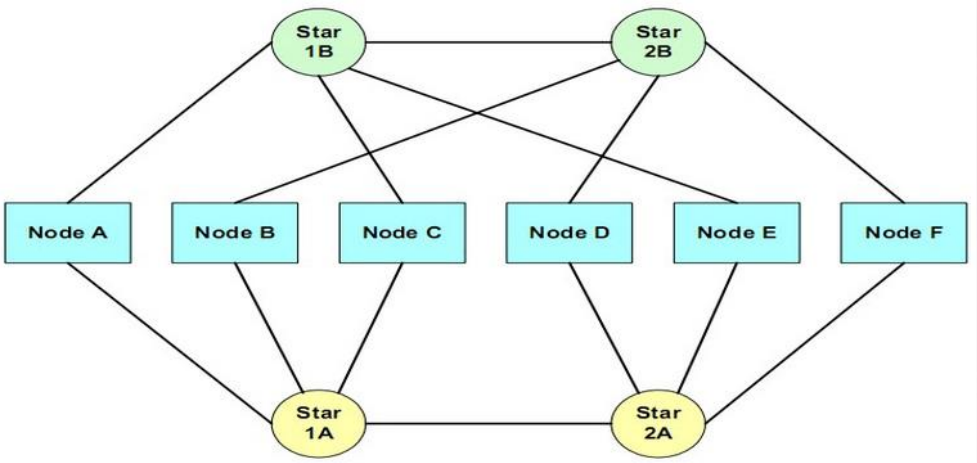


Figure 1-4: Dual channel cascaded star configuration.

- **Active star simplifies some aspects of distributed coordination**

Maximum delay through star is 250 ns, so it does not buffer full messages

In active star networks, one or more branches of the active star(s) may be built as a linear passive bus or as a passive star.

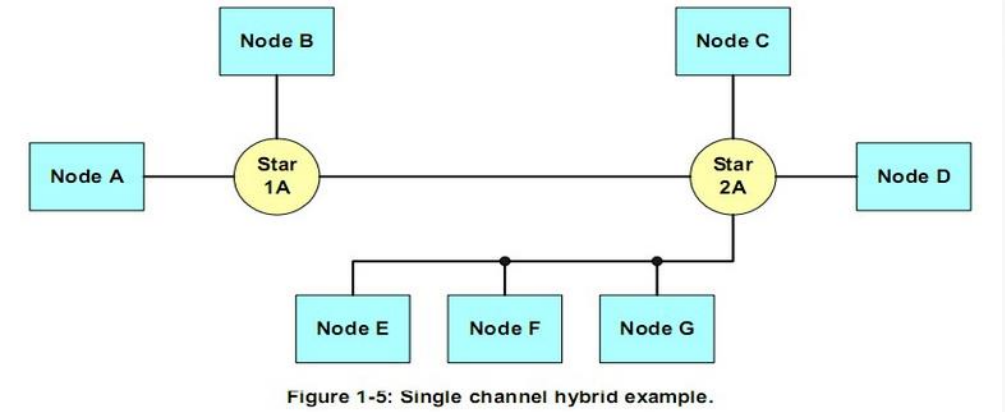


Figure 1-5: Single channel hybrid example.

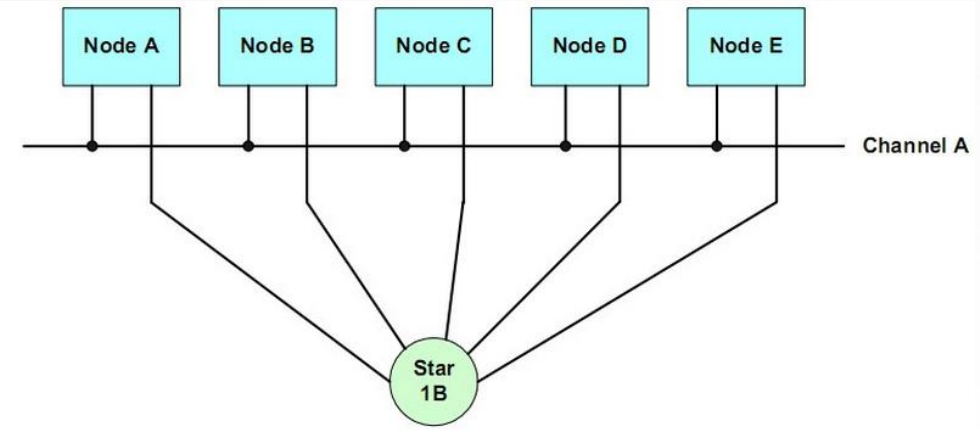


Figure 1-6: Dual channel hybrid example.

FlexRay Architecture – Network Topology

- **Intended to eliminate single-point failures for critical systems**
  - This seems the most likely configuration for FlexRay X-by-Wire

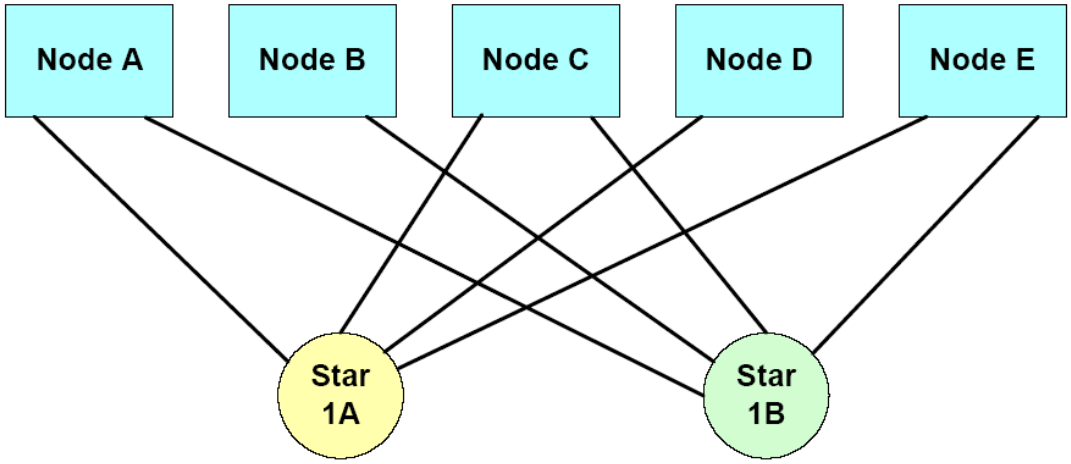


Figure 1-2: Dual channel single star configuration.

- **TTP was found to have some distributed bus guardian issues**
  - Problems related to nodes listening to faulty network startup messages
  - Latest proposal is to move to dual channel star configuration for TTP as well

FlexRay – Signal on FR BUS

- **Differential NRZ encoding**

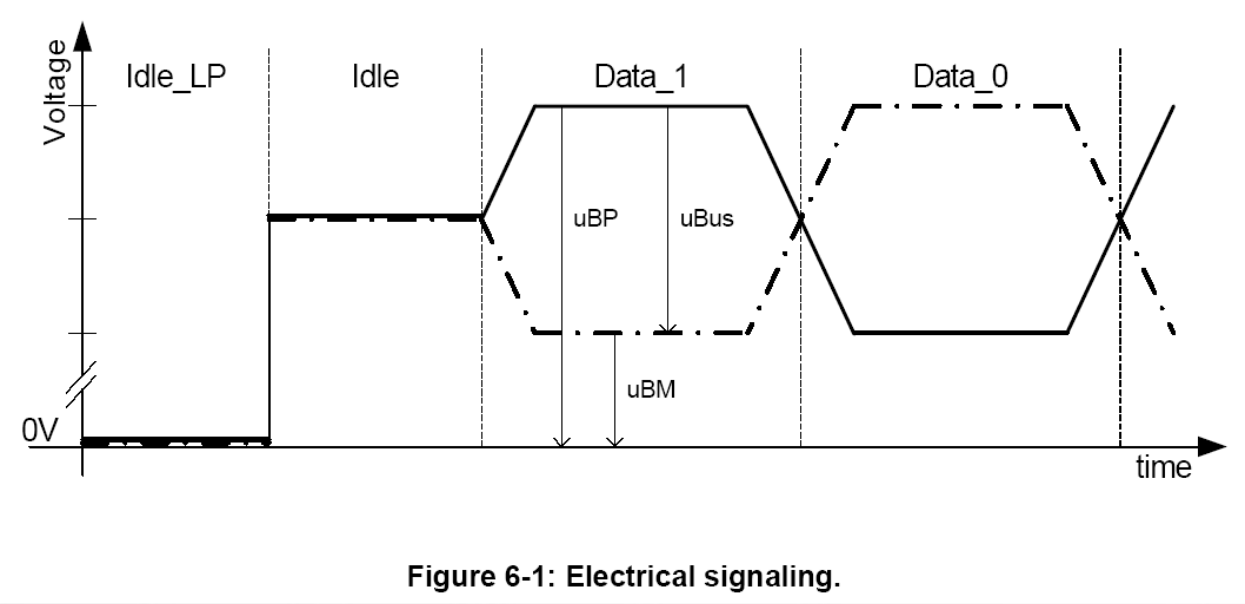
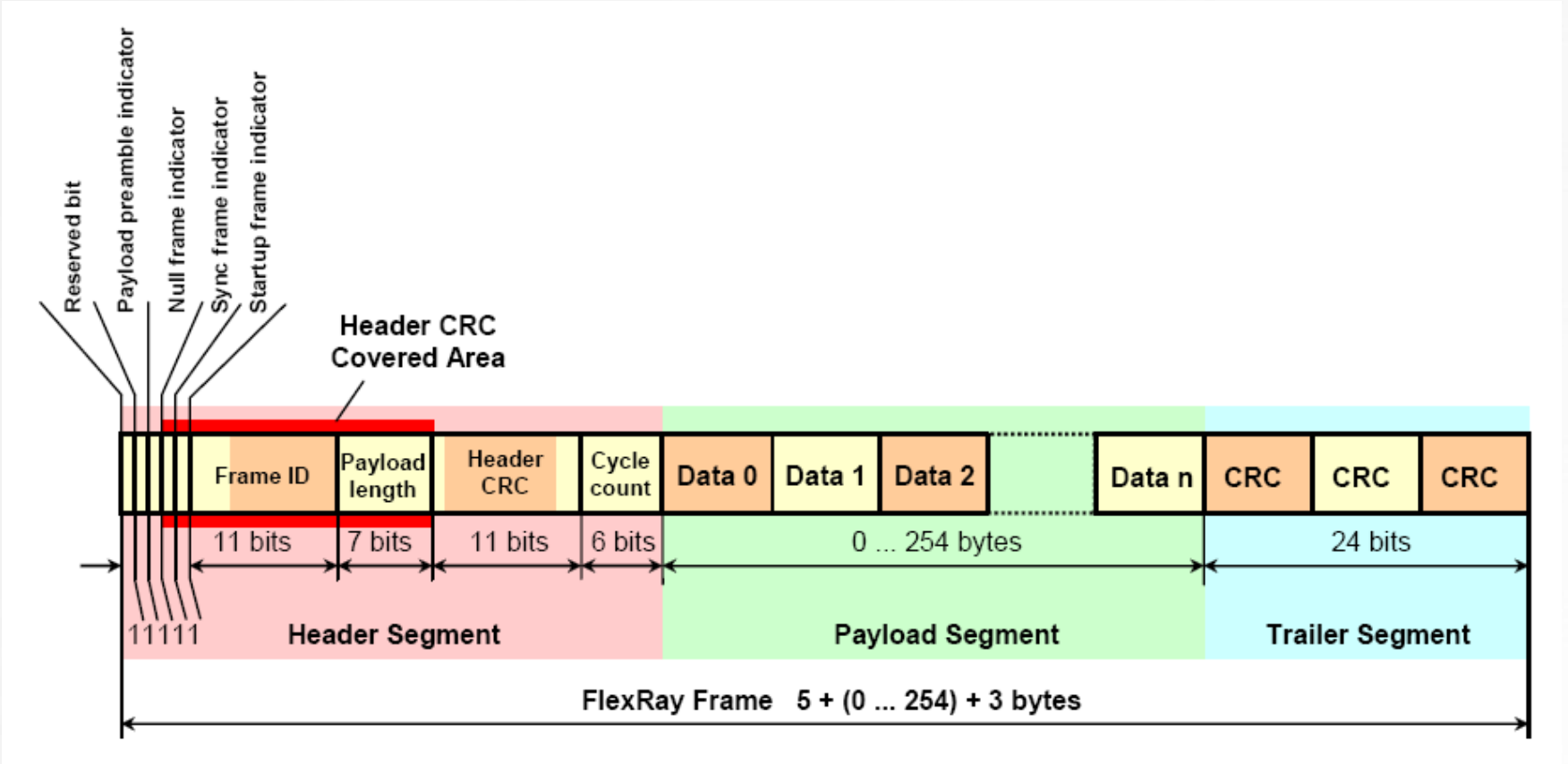


Figure 6-1: Electrical signaling.

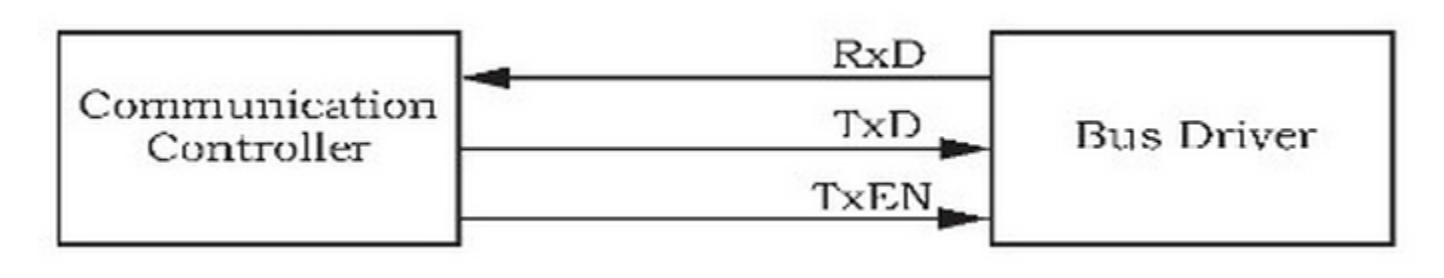
- **10 Mbps operating speed**
  - Independent of network length because, unlike CAN, doesn't use bit arbitration

FlexRay Protocol -- Frame Format



•• This data is encoded into NRZ bytes per the encoding format

FlexRay Protocol -- Frame Encoding



FlexRay Protocol -- Coding

- **Data sent as NRZ bytes**
  - TSS = Transmit Start Sequence (LOW for 5-15 bits)
  - FSS = Frame Start Sequence (one HI bit)
  - BSS = Byte Start Sequence (similar to start/stop bits in other NRZ)
  - FES = Frame End Sequence (END symbol for frame – LO + HI)

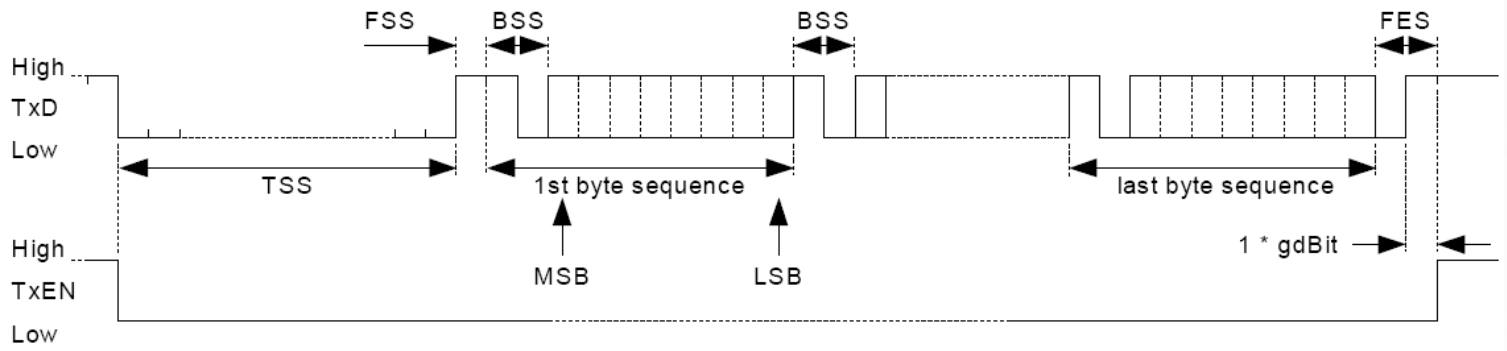
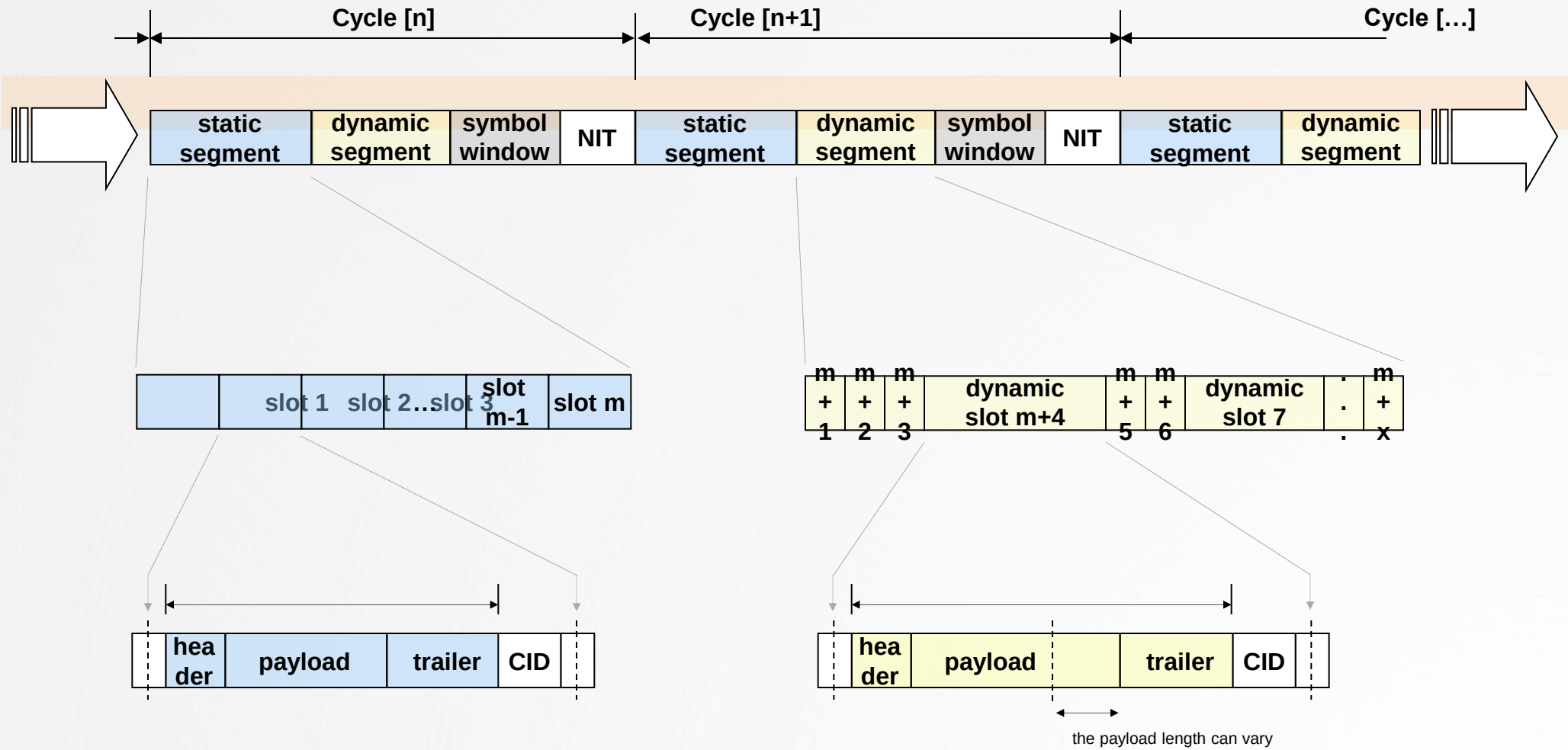


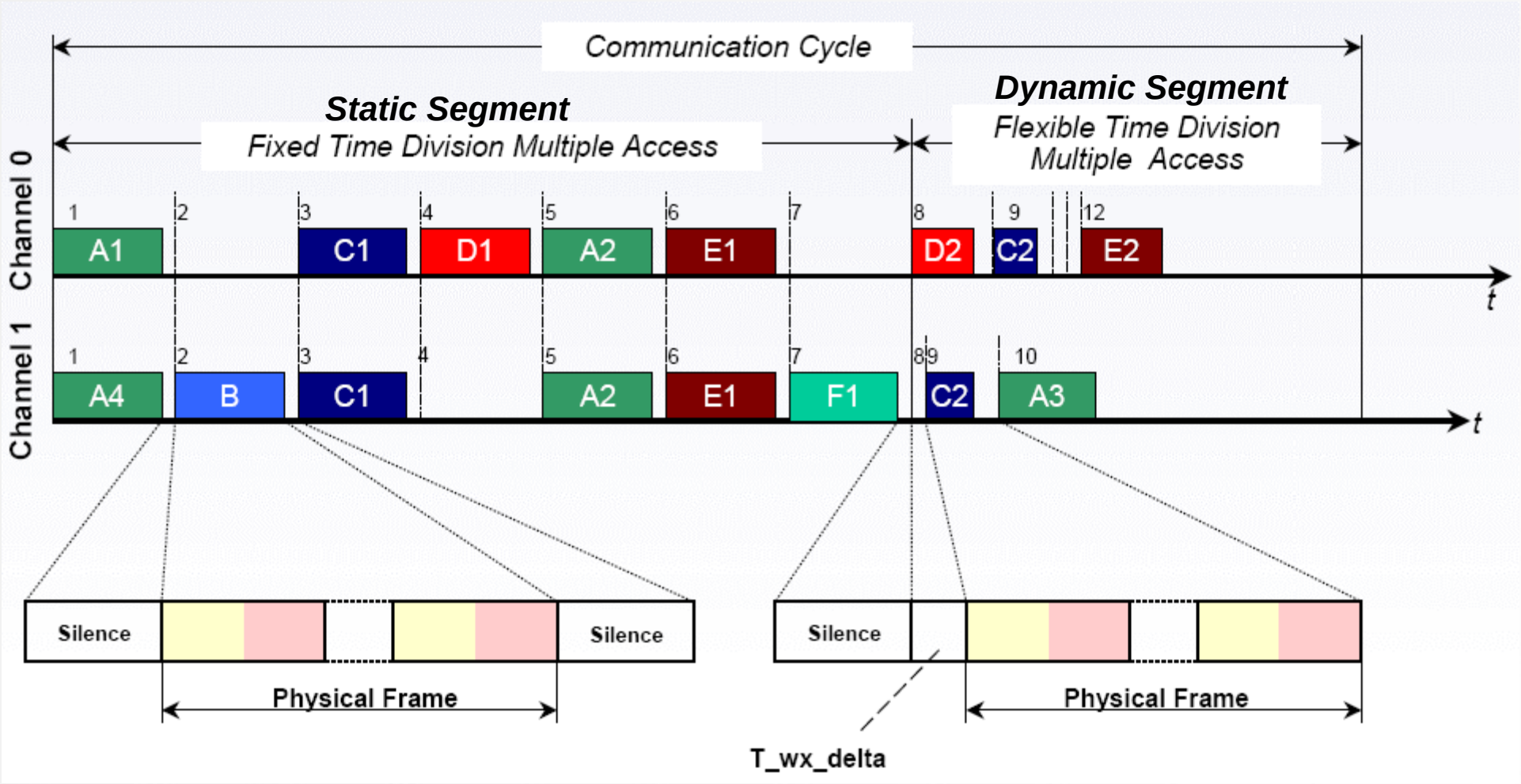
Figure 3-2: Frame encoding in the static segment.

- **Dynamic segment frames are similar**
  - Adds a DTS = dynamic trailing sequence field; helps line up minislots

FlexRay Communication Structure



FlexRay Communication Cycle



FlexRay Message Cycle

- **Two main phases: static & dynamic**
  - "Temporal firewall" – partition between phases protects timing of each phase

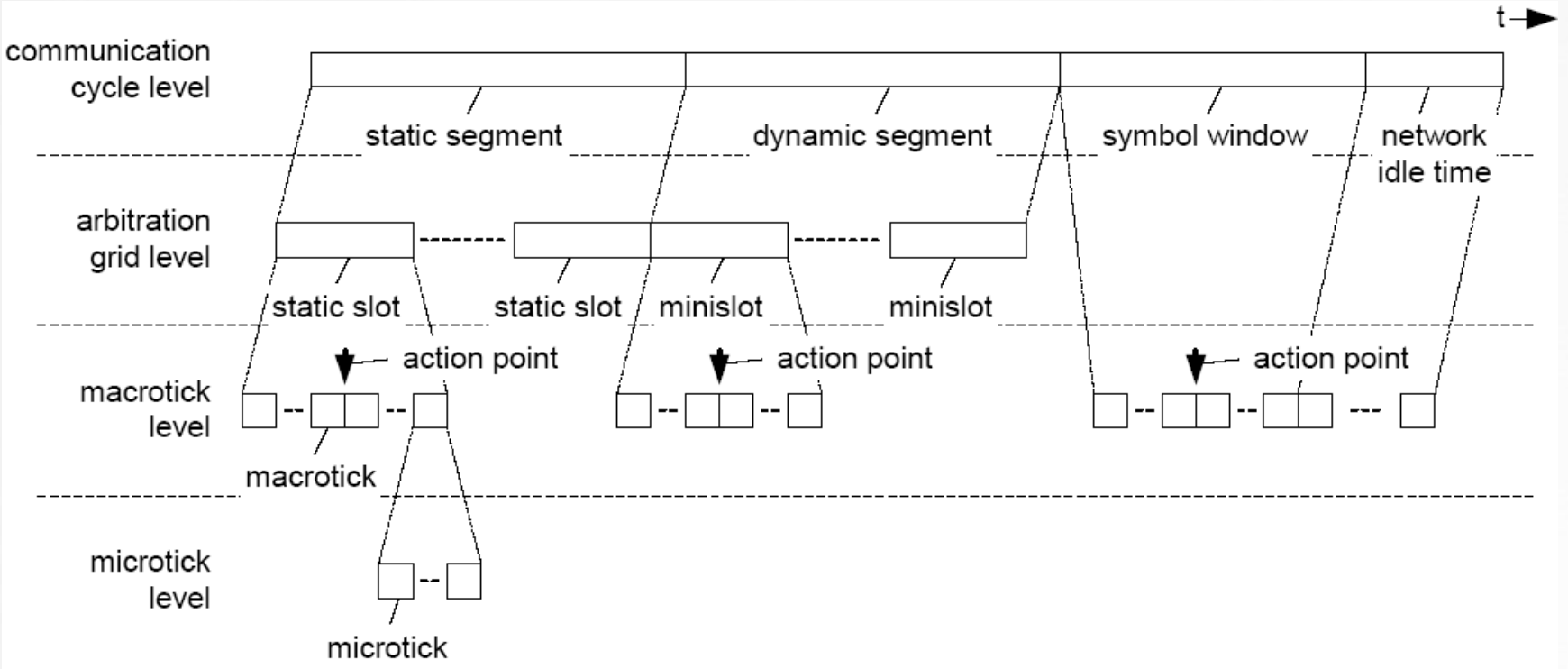


Figure 5-1: Timing hierarchy within the communication cycle.

Microtick & Macrotick

- **Microtick level**
  - Node's own internal time base
  - Direct or scaled value from a local oscillator or counter/timer
  - Not synchronized with rest of system – local free-running oscillator
- **Macrotick level**
  - Time interval derived from cluster-wide clock sync algorithm
  - Always an integral number of microticks
  - BUT, not necessarily the same number of microticks per node
  - Number of microticks varies at run time to implement clock sync
- **Designated macrotick boundaries are "action points"**
  - Transmissions start here – static; dynamic; symbol window
  - Transmissions end here – dynamic segment

Static Segment

TDMA messages, most likely used for critical messages

- All static slots are the same length in microticks
- All static slots are repeated in order every communication cycle
- All static slot times are expended in cycle whether used or not
- Number of static slots is configurable for system ; up to 1023 slots

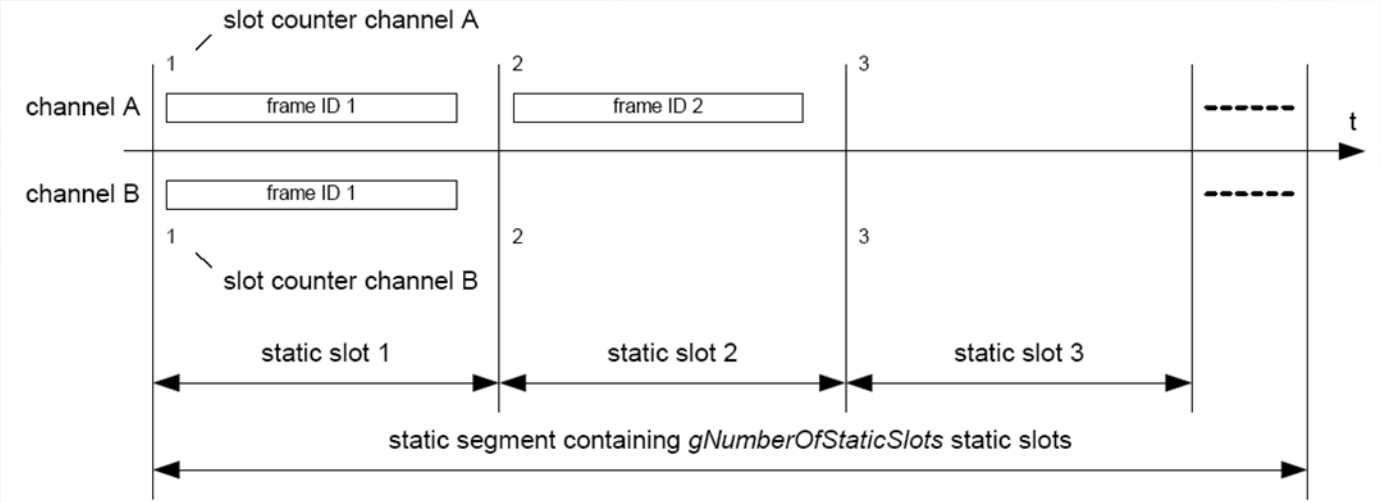


Figure 5-3: Structure of the static segment.

Static Segment Details

Two-channel operation

- Sync frames on both channels; other frames optionally 1 or 2 channels
- Less critical/less expensive nodes might only connect to one channel
- Slots are lock-stepped in order on both channels

TDMA order is by ascending frame ID number

- Frame number used to determine slot # by software
  1. It is NOT a binary countdown arbitration mechanism – only one xmitter at a time
  2. Optionally, there is a Message ID in the payload area that can be unrelated to slot number
  3. Example use: each node uses its node # as frame # and multiplexes its messages onto a single time slot, distinguished by Message ID
- In contrast, TTP has a MEDL that can have sub-cycles
  1. Need neither a Frame ID nor a Message ID
  2. Extra information to be managed and coordinated

Dynamic Segment

High-level idea is event-based communication channel

- Want arbitration, but must be deterministic
- Binary countdown not used (among other things, restricts possible media)

“Minislot” approach

- Can be thought of as a time-compressed TDMA approach (details on next slide)
- Two channels can use independent message queues

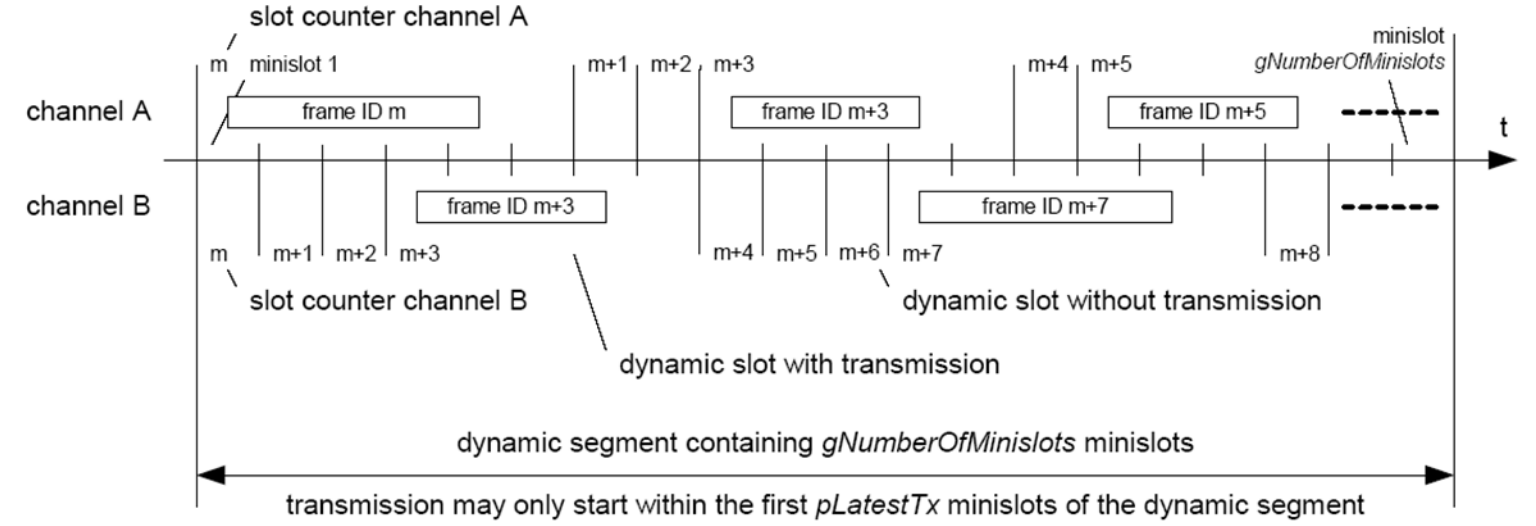


Figure 5-5: Structure of the dynamic segment.

Dynamic Segment Details

High-level idea is each minislot is an opportunity to send a message

- If message is sent, minislot expands into a message transmission
- If message isn't sent, minislot elapses unused as a short idle period
- All transmitters watch whether a message is sent so they can count minislots

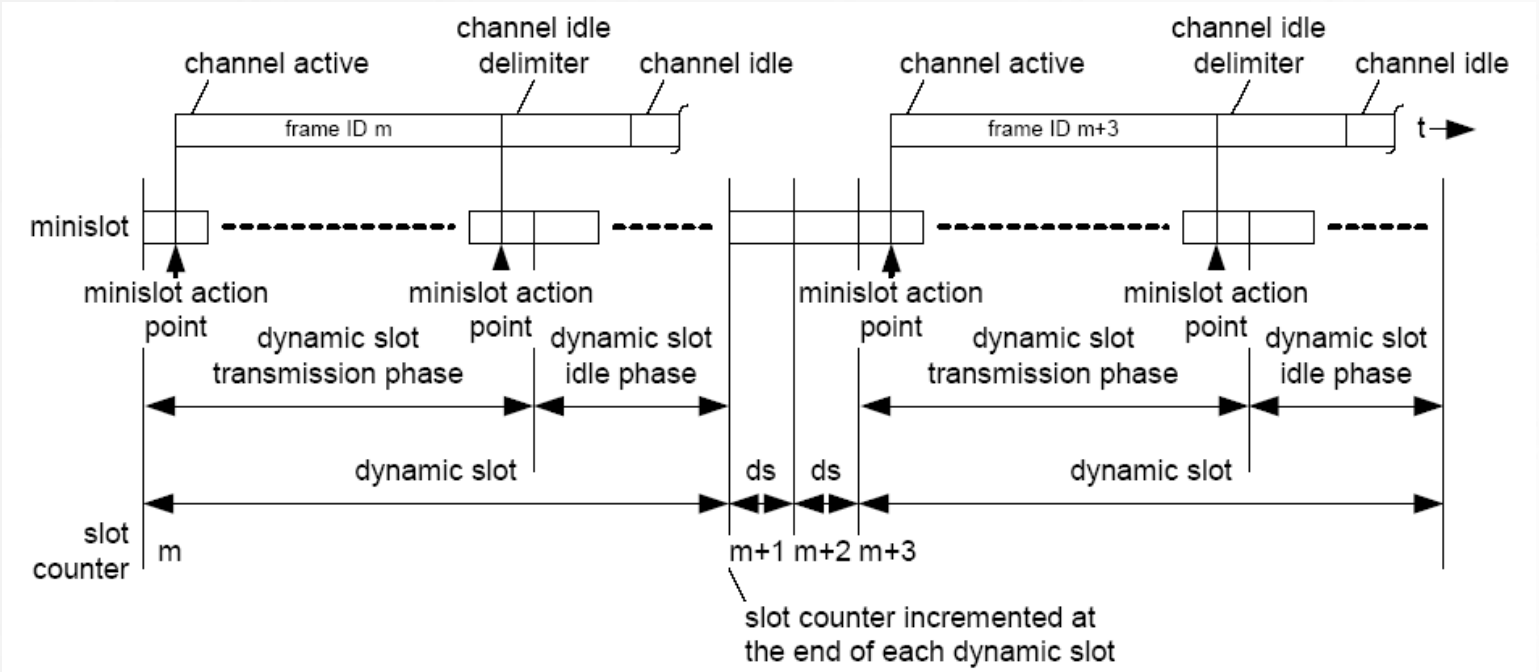


Figure 5-7: Timing within the dynamic segment.

Minislot Performance

Frame ID # is used for slot numbering

- First dynamic Frame ID = last static Frame ID + 1

Dynamic segment has a fixed amount of time

- Fixed number of macroticks, divided up into minislots
- There might or might not be enough time for all dynamic messages to be sent
- When dynamic segment time is up, unsent messages wait for next cycle

Net effect: event-triggered messages

- Messages with the lowest Frame ID are sent first
- Each Frame ID # can only send ONE message per cycle
- As many message as will fit in dynamic segment are sent
- This means that only highest priority messages queued are sent in each cycle
- Note that idle minislots consume dynamic segment bandwidth
  - But minislots are a lot smaller than messages

System Startup of FlexRay

First, a wakeup procedure

- Transition controllers from “sleep” to “wake”
- (Not necessarily initiated by a coldstart node)

Then, a coldstart, that actually starts the TDMA operation

- At least two coldstart nodes in system
- For systems with 3 or more nodes, ideal number is three coldstart nodes
  - ✓ More than 3 nodes might have problems with forming a single group (clique avoidance on startup works only for 3 nodes)
- Coldstart nodes are pre-designated in system

Wakeup Pattern

Wakeup pattern sent on one channel to alert other nodes to wake up

- Pattern sent multiple times in general – example shows sending it twice
- Only sent on one channel at a time
  - ✓ Have to handle case where sending wakeup is a single-point fault
- Wakeup isn't acknowledged by nodes
  - ✓ That is dealt with during startup

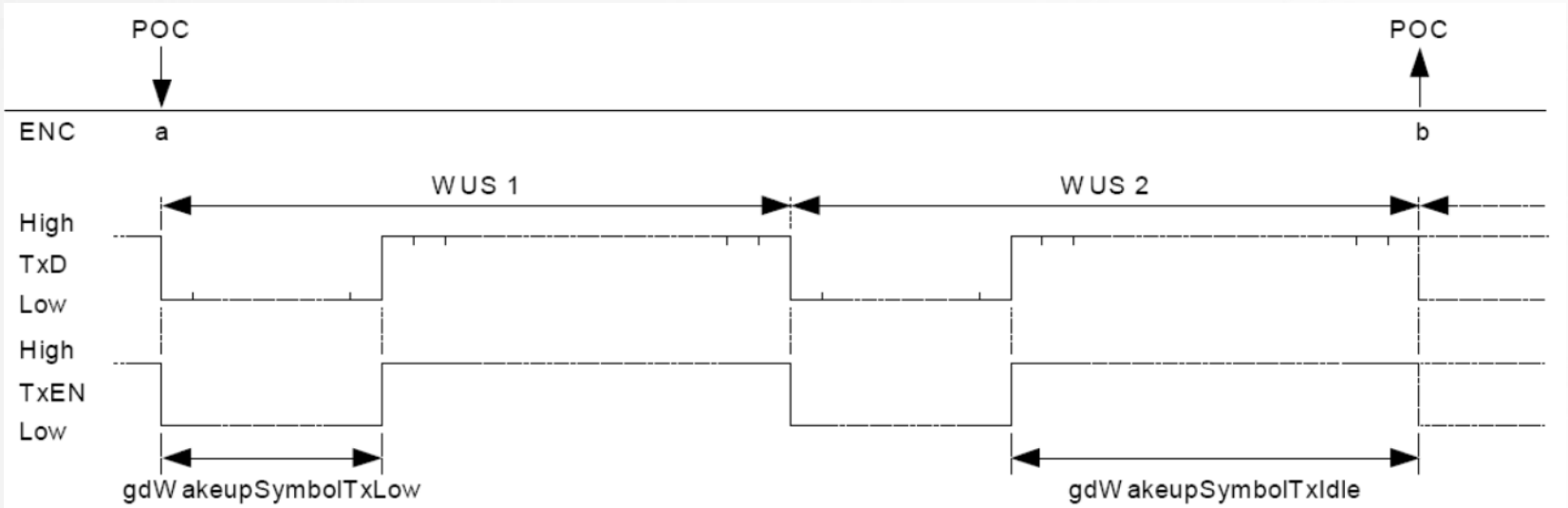


Figure 3-5: Wakeup pattern consisting of two wakeup symbols.

TI's FlexRay Implementation

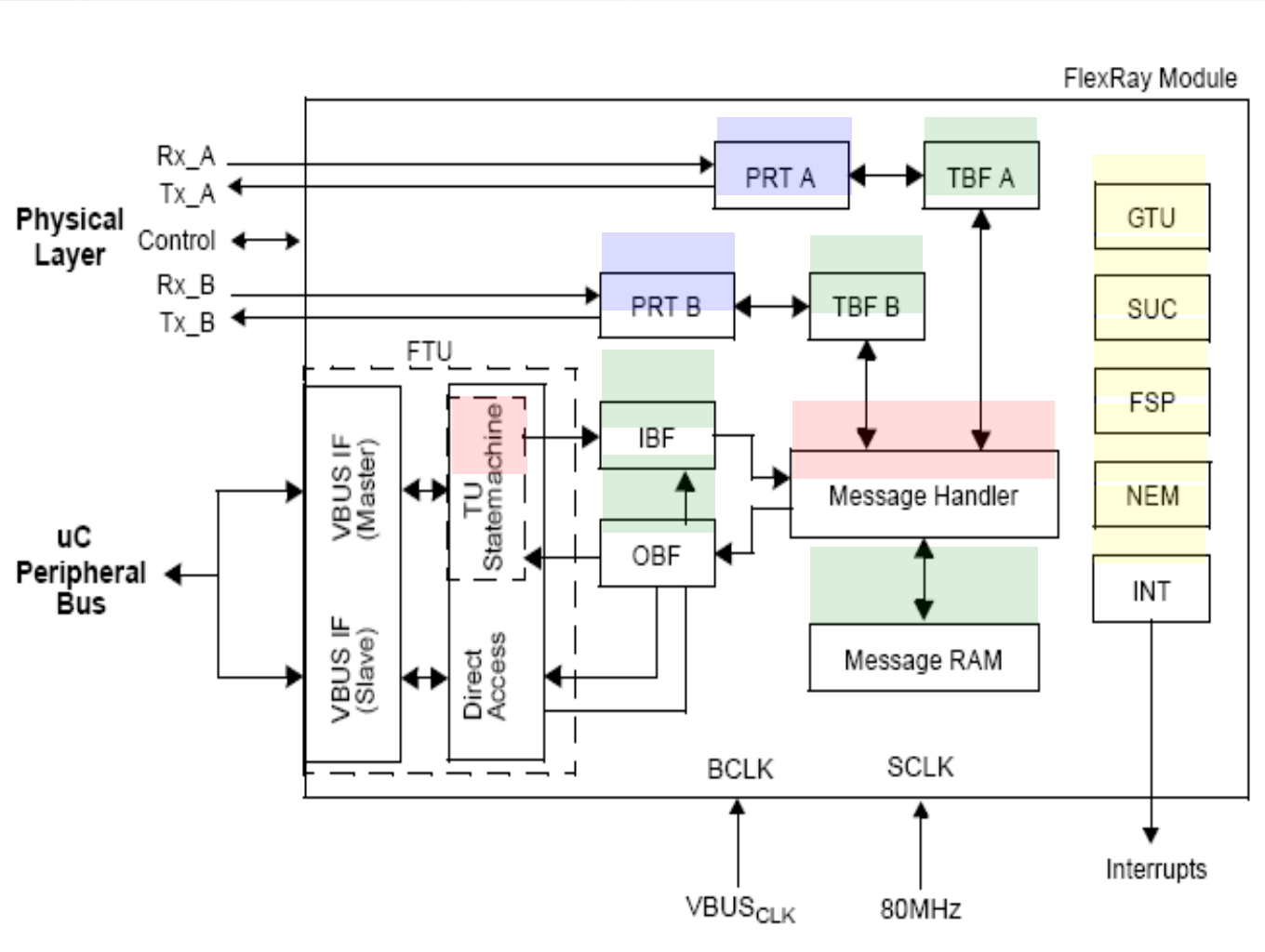
Key Features (1/2)

- FlexRay Core (E-Ray)
- Conform to FlexRay Protocol Specification V2.1 RevA
- Data rates of up to 10 Mbit/s on each of the 2 channels
- 8 Kbyte of Message RAM for storage of e.g.
  - 128 message buffers with max. 48 byte data section or
  - 30 message buffers with 254 byte data section
  - Different payload lengths possible
- Parity Protection of Message RAM
- Message Handler controls
  - Message RAM access arbitration
  - Acceptance Filtering
  - Maintaining the transmission schedule
  - Providing status information

Key Features (2/2)

- Each message buffer can be configured as
  - Receive buffer
  - Transmit buffer
- Each message buffer can be assigned to
  - Static segment of the Communication Cycle
  - Dynamic segment of the Communication Cycle
  - Part of a receive FIFO
- 2 Module Interfaces
  - Direct CPU access to message buffers via input and output buffer
  - Transfer Unit for automatic data transfer between data memory and message buffers without CPU interaction
- Filtering for frame ID, channel ID and cycle counter
- Maskable module interrupts
- Network Management supported

Block Diagram



FTU:
FlexRay Transfer Unit

IBF:
Input Buffer (relative to msg handler)

OBF:
Output Buffer

INT:
Interrupt Control

TBF A/B:
Transient Buffer RAM

PRT A/B:
FlexRay Channel Protocol Controller

GTU:
Global Time Unit

SUC:
System Universal Control

FSP:
Frame and Symbol Processing

NEM:
Network Management

Message Buffers in the Message RAM

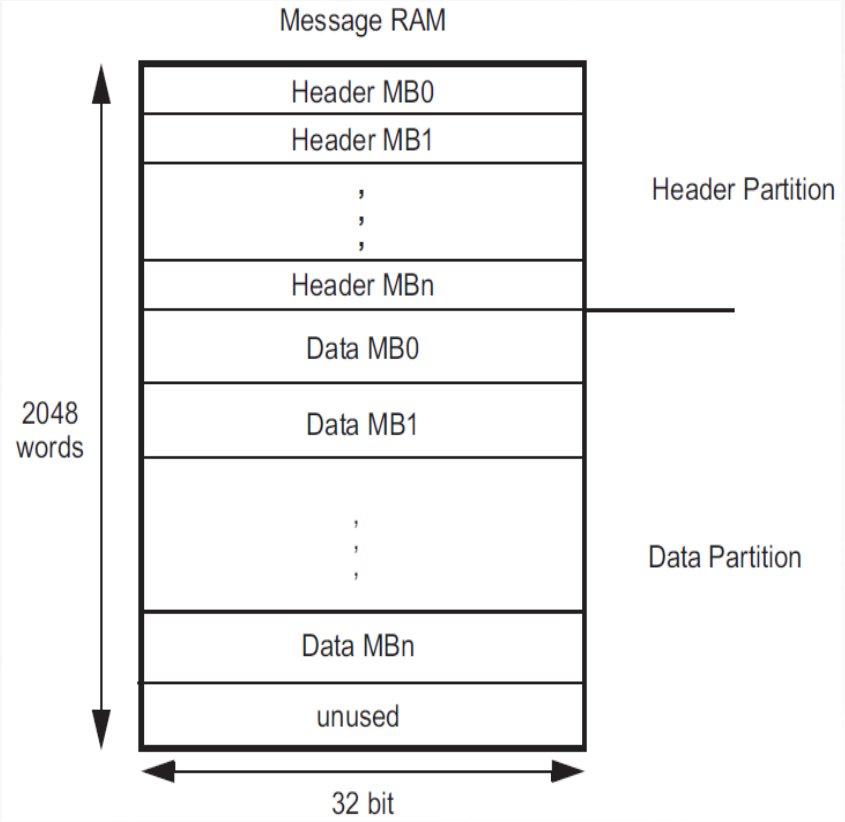
The message RAM has a structure as shown in right diagram

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

Header Partition: Stores header segments of FlexRay frames. Each message buffer has a header of four 32 bit words.

Data Partition: Flexible storage of data sections with different length.

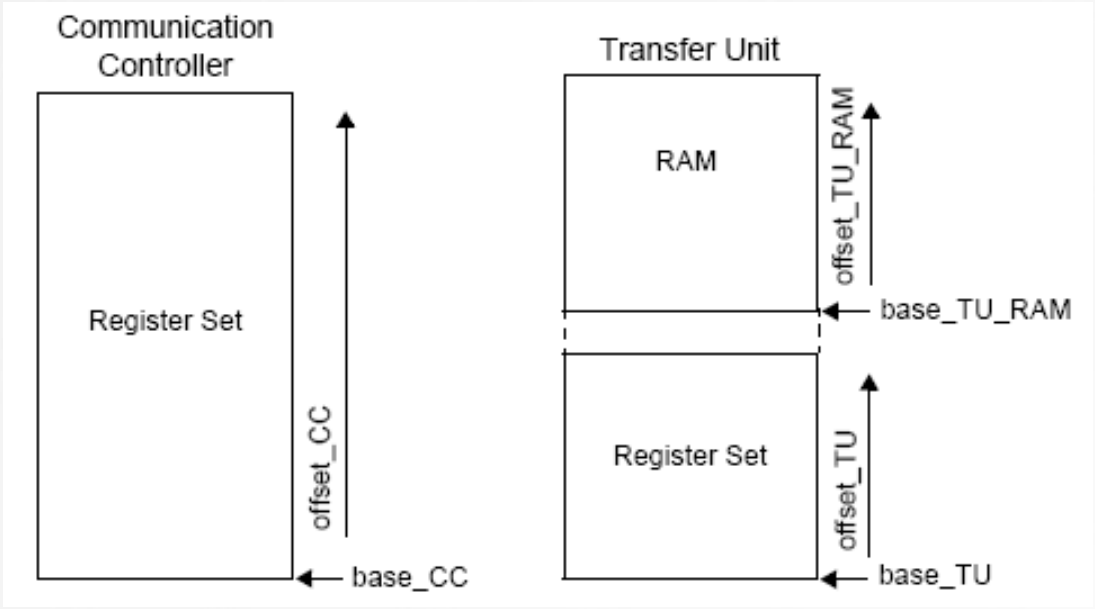
Restriction: header partition + data partition may not occupy more than 2048 x 32 bit words.



Note:

The Message RAM of the Communication Controller is not memory mapped

Module Block Memory Map

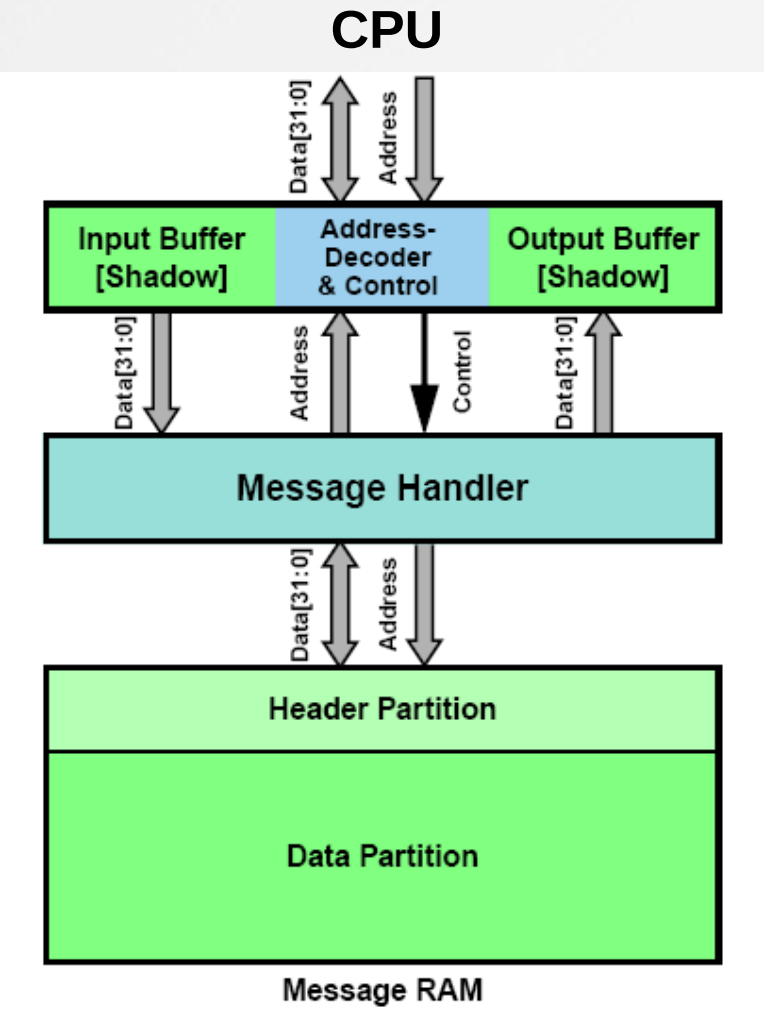


Module	Address Range
FlexRay Communication Controller	0xFFF7_C800 - 0xFFF7_CFFF
FlexRay TU	0xFFF7_A000 - 0xFFF7_A1FF
FlexRay TU RAM	0xFF50_0000 - 0xFF51_FFFF

Note:

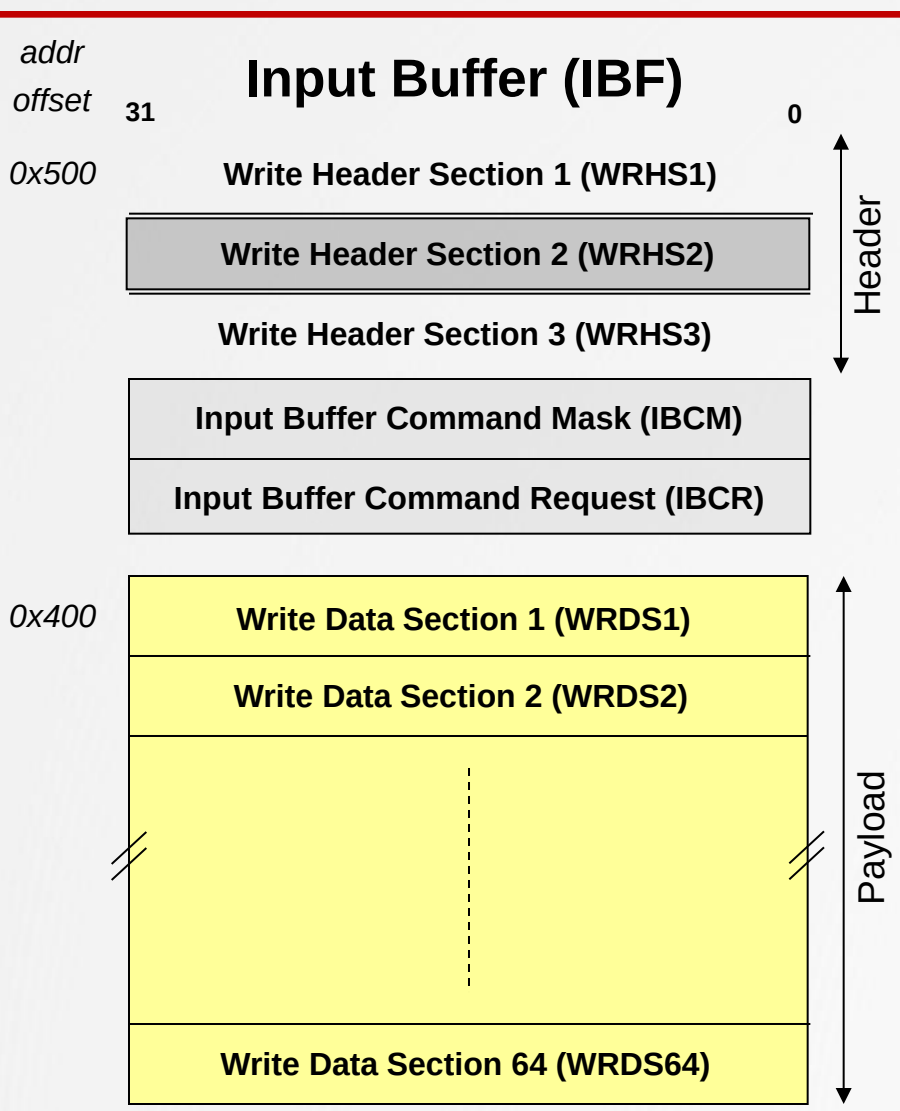
The Message RAM of the Communication Controller is not memory mapped

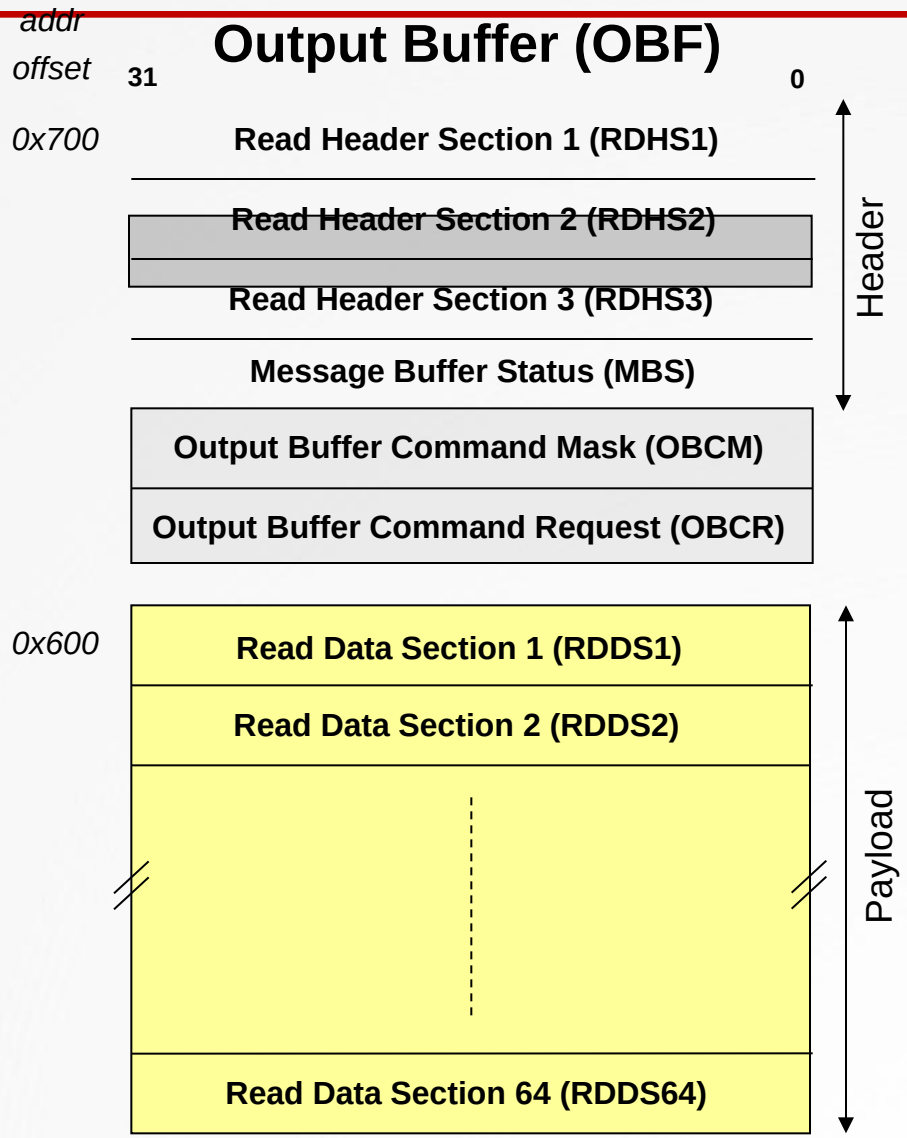
CPU Access to Message RAM



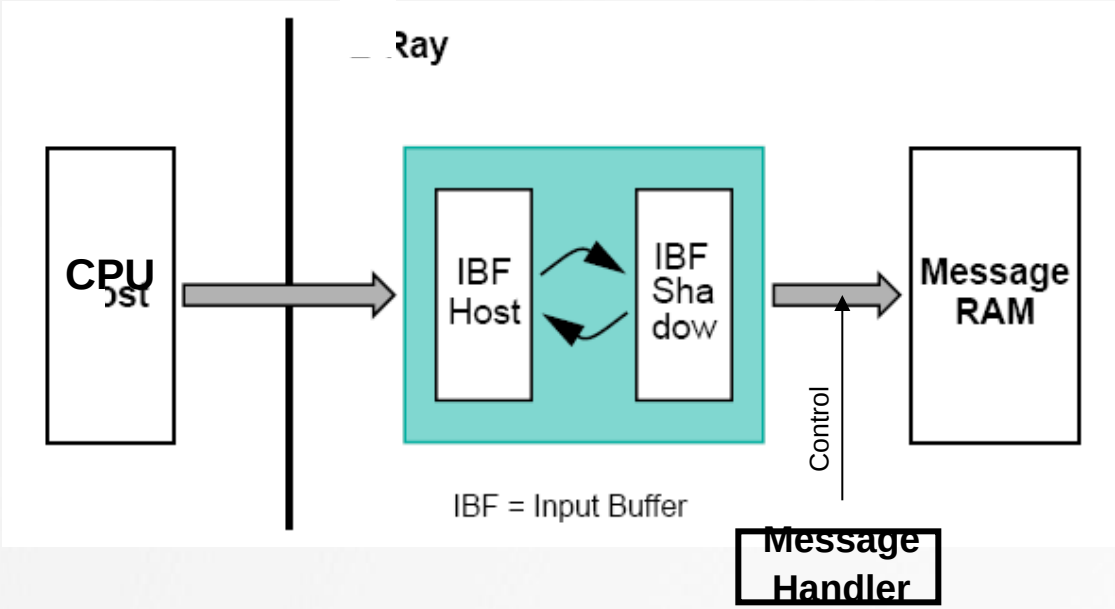
- The data transfer between CPU and Message Buffer is done via a Input- and Output Buffer register set
- Each, Input- and Output Buffer register set, can hold one message frame (Header + Payload)
- Input Buffer (IBF) and Output Buffer (OBF) are build up as a double buffer structure
- The message transfer is triggered by the CPU

Data Organization in Interface Registers



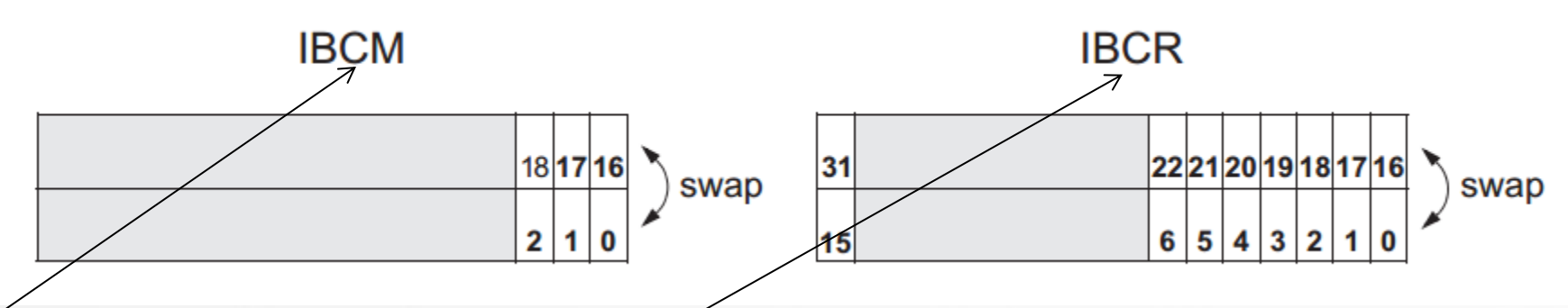


Data Transfer from Input Buffer to Message RAM 1/3



- When the Host writes the number of the target message buffer in the Message RAM to Input Buffer Command Register, IBF Host and IBF Shadow are swapped
- While the Message Handler transfers the data from IBF Shadow to the target message buffer in the Message RAM, the CPU may write the next message to IBF Host.

Data Transfer from Input Buffer to Message RAM 2/3



Position	Access	Bit	Function
18	r	STXRS	Set Transmission Request shadow ongoing or finished
17	r	LDSS	Load Data Section shadow ongoing or finished
16	r	LHSS	Load Header Section shadow ongoing or finished
2	r/w	STXRH	Set Transmission Request Host
1	r/w	LDSH	Load Data Section Host
0	r/w	LHSH	Load Header Section Host

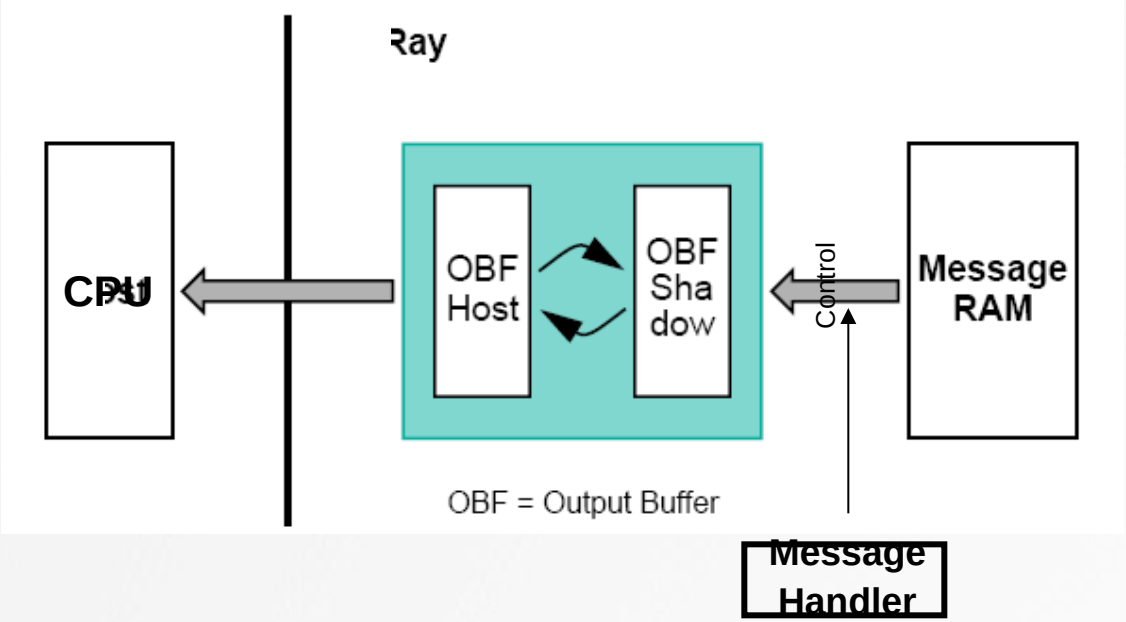
Pos.	Access	Bit	Function
31	r	IBSYS	IBF Busy Shadow, signals ongoing transfer from IBF shadow to message RAM
22…16	r	IBRS(6-0)	IBF Request Shadow, number of message buffer currently / last updated
15	r	IBSYH	IBF Busy Host, transfer request pending for message buffer referenced by IBRH(6-0)
6…0	r/w	IBRH(6-0)	IBF Request Host, number of message buffer to be updated next

Data Transfer from Input Buffer to Message RAM 3/3

- **Example of a 8/16/32-bit ost access sequence:**
  1. Configure / update n-th message buffer through IBF
  2. Wait until IBCR.IBSYH is reset
  3. Write data section to WRDSn
  4. Write header section to WRHS1,2,3
  5. Write command mask: write IBCM.STXRH, IBCM.LHSH, IBCM.LDSH
  6. Demand data transfer to target message buffer: write IBCR.IBRH(6-0)
- **Configure / update further message buffer through IBF in the same**

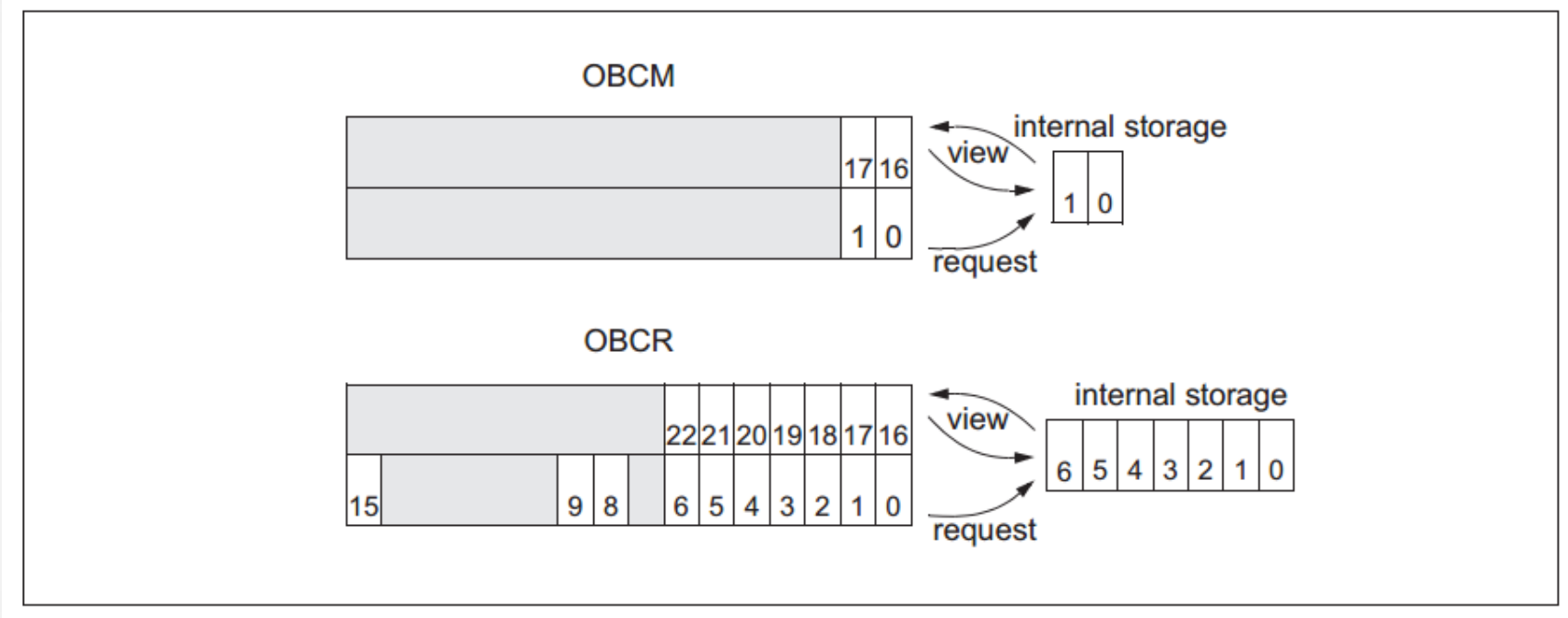
```
write_buffer->stxrh= 1;     // set transmission request
write_buffer->ldsh = 1;     // load data section
write_buffer->lhsh = 0;     // load header section
write_buffer->ibsys = 0;    // check for input buffer busy shadow
write_buffer->ibsyh = 1;    // check for input buffer busy host

// wait for cycle start interrupt flag
Fray_PST->SIR_UN.SIR_UL = 0xFFFFFFFF;              // clear all status int. flags
while ((Fray_PST->SIR_UN.SIR_UL & 0x4) == 0x0);    // wait for CYCS interrupt flag
Fray_PST->SIR_UN.SIR_UL = 0xFFFFFFFF;              // clear all status int. flags


// write payload for buffers
if(node <= SYNC_NODE)
{
    // buffer #0
    write_buffer->ibrh = 0;                                  // input buffer number, sync buffer
    (Fray_PST->WRDS[0] = sync_data[node-1][0]);              // Data 1
    (Fray_PST->WRDS[1] = sync_data[node-1][1]);              // Data 2
    Fr_TransmitTxLPdu(Fray_PST, write_buffer);
}
```

Data Transfer from Message RAM to Output Buffer 1/3



- For reading a message buffer from the Message RAM, the CPU has to write to the Output Buffer Command Register to trigger the data transfer from the target message buffer to the OBF Shadow
- After the transfer between the Message RAM and OBF Shadow has completed, the CPU has to set a **VIEW bit** to swap OBF Shadow and OBF Host
- While the CPU reads the transferred message buffer from OBF Host, the Message Handler may transfer the next message from the Message RAM to OBF Shadow.

Data Transfer from Message RAM to Output Buffer 2/3



- Writing **bit OBCR.REQ** in the output buffer command request register to 1 copies bits OBCM.RHSS, OBCM.RDSS from the output buffer command mask register and bits OBCR.OBRS(6-0) from the output buffer command request register to an internal storage

Data Transfer from Message RAM to Output Buffer 3/3

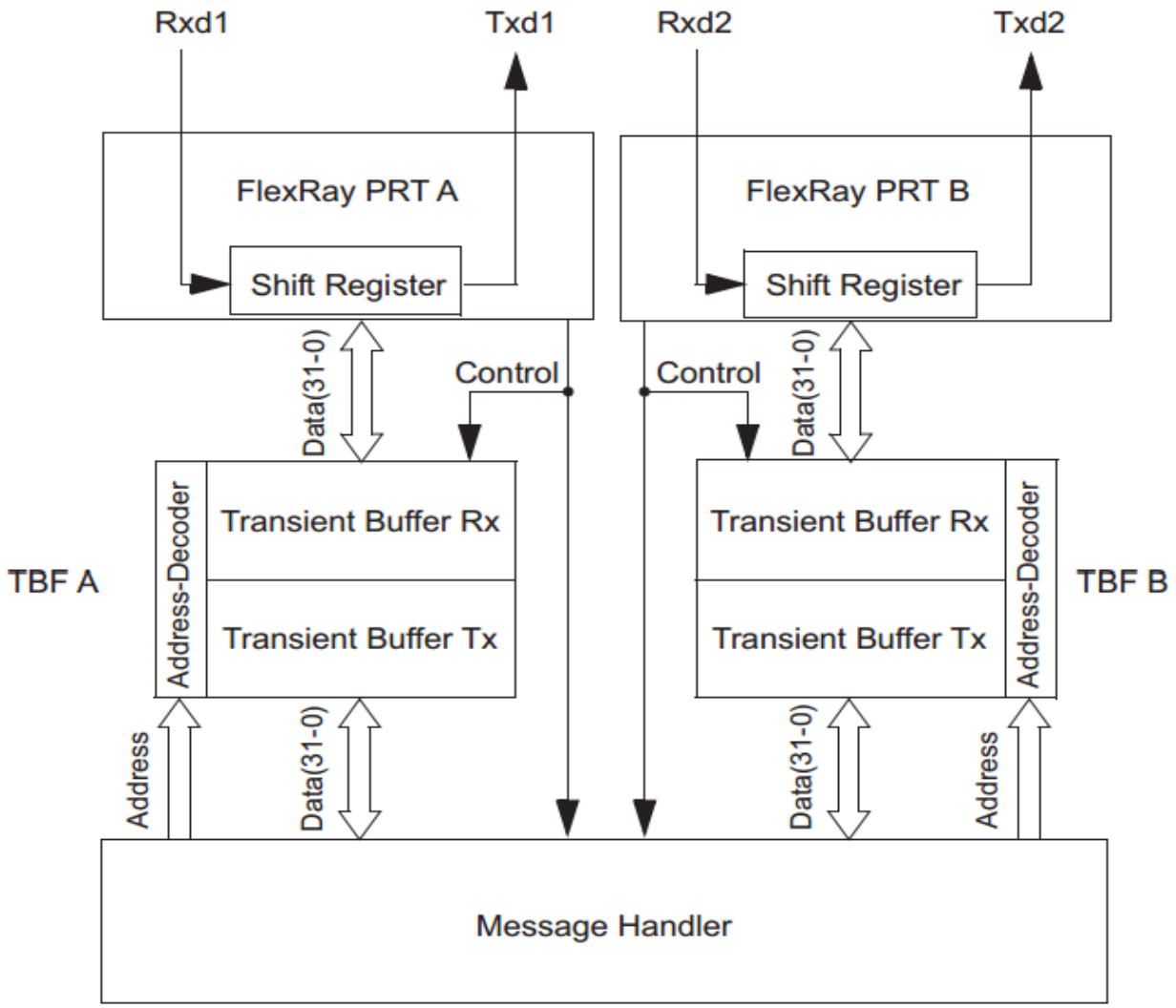
Example of host access to a single message buffer:

If a single message buffer has to be read out, two separate write accesses to OBCR.REQ and OBCR.VIEW are necessary:

- Wait until OBCR.OBSYS is reset
- Write Output Buffer Command Mask OBCM.RHSS, OBCM.RDSS
- Request transfer of message buffer to OBF Shadow by writing OBCR.OBRS(6-0) and OBCR.REQ (in case of and 8-bit Host interface, OBCR.OBRS(6-0) has to be written before OBCR.REQ).
- Wait until OBCR.OBSYS is reset
- Toggle OBF Shadow and OBF Host by writing OBCR.VIEW = 1
- Read out transferred message buffer by reading RDDSn, RDHS1,2,3, and MBS

```
void Fr_ReceiveRxLPdu(FRAY_ST *Fray_PST, bc *Fr_LSduPtr)
{
    // ensure no transfer in progress on shadow registers
    while (((Fray_PST->OBCR_UN.OBCR_UL) & 0x00008000) != 0);
    Fray_PST->OBCM_UN.OBCM_UL=(((Fr_LSduPtr->rdss & 0x1) << 1) | (Fr_LSduPtr->rhss & 0x1));
    Fray_PST->OBCR_UN.OBCR_UL=((1 << 9) | (Fr_LSduPtr->obrs & 0x7F)); //req=1, view=0
    // wait for completion on shadow registers
    while (((Fray_PST->OBCR_UN.OBCR_UL) & 0x00008000) != 0);

    Fray_PST->OBCM_UN.OBCM_UL=(((Fr_LSduPtr->rdss & 0x1) << 1) | (Fr_LSduPtr->rhss & 0x1));
    Fray_PST->OBCR_UN.OBCR_UL=((1 << 8) | (Fr_LSduPtr->obrs & 0x7F)); //req=0, view=1
}
```

FlexRay Protocol Controller Access to Message RAM



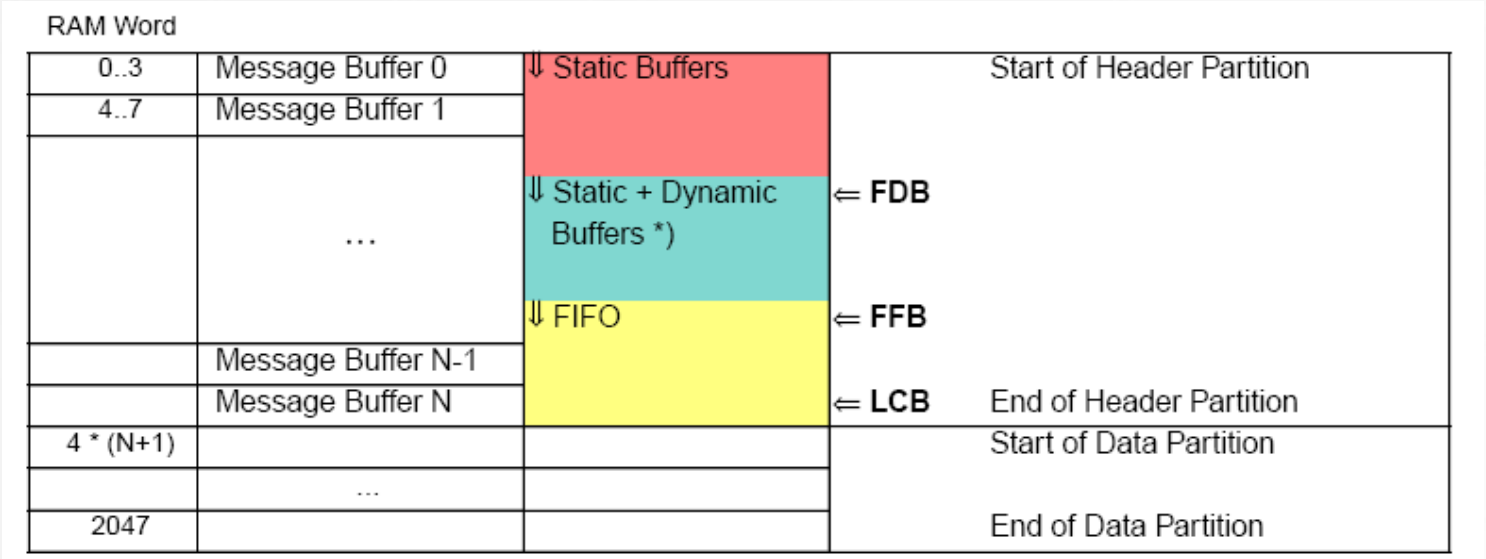
The FlexRay module contains the following RAM portions:

1. Message RAM
2. Transient Buffer RAM Channel A (TBF A)
3. Transient Buffer RAM Channel B (TBF B)
4. Input Buffer (IBF)
5. Input Buffer Shadow (IBFS)
6. Output Buffer (OBF)
7. Output Buffer Shadow (OBFS)
8. Transfer Configuration RAM (TCR)

All RAMs except the TCR are part of the Communication Controller core.

Message RAM Configuration

Each FlexRay Buffer consists of a Header and a Data section



RAM Word				
0..3	Message Buffer 0	⇓ Static Buffers		Start of Header Partition
4..7	Message Buffer 1			
	…	⇓ Static + Dynamic Buffers *)	⇐ **FDB**	
		⇓ FIFO	⇐ **FFB**	
	Message Buffer N-1			
	Message Buffer N		⇐ **LCB**	End of Header Partition
4 * (N+1)				Start of Data Partition
	…			
2047				End of Data Partition

Header:

FDB: First Dynamic Buffer

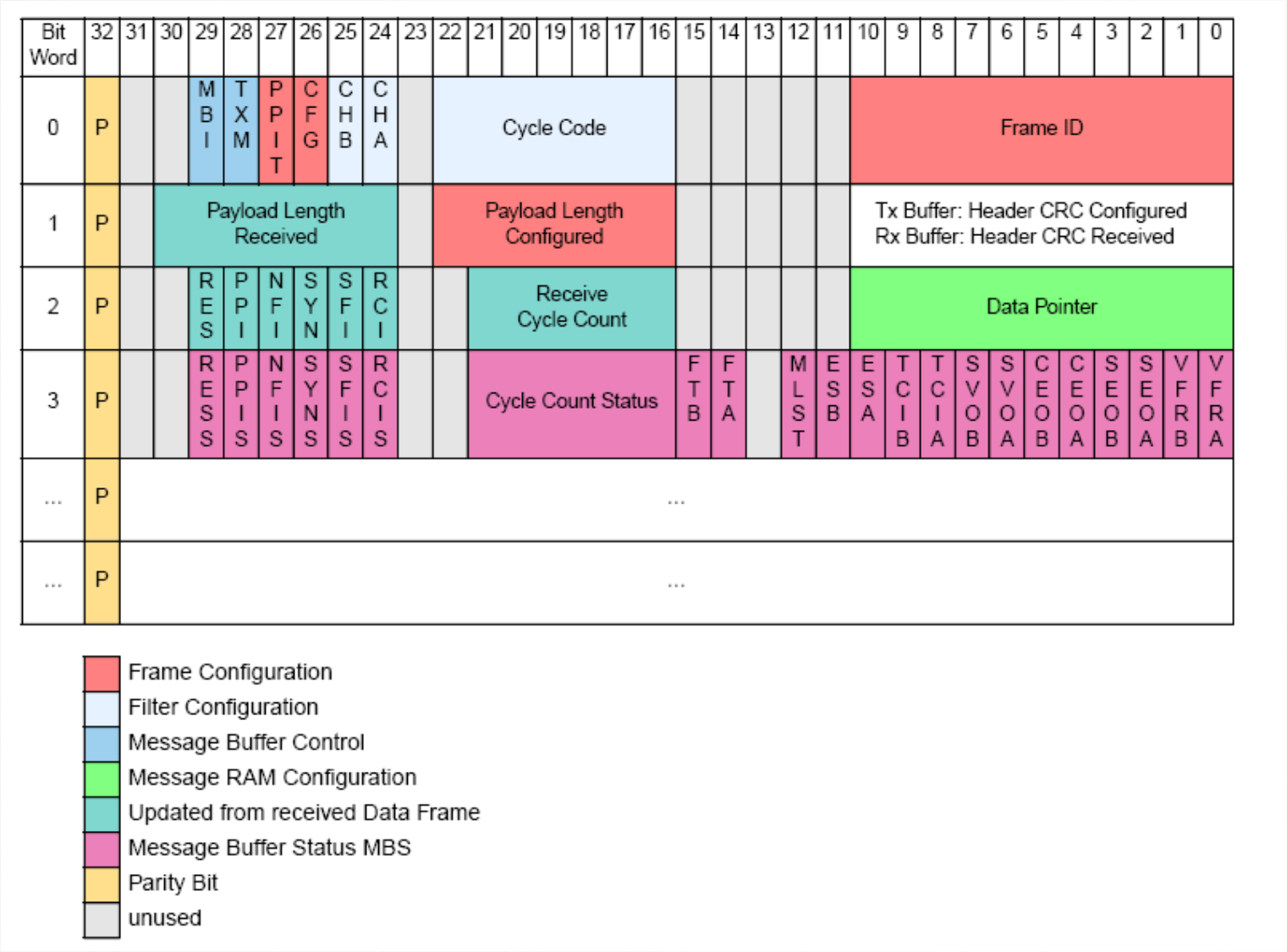
FFB: First Buffer of FIFO

LCB: Last Configured Buffer

Data:

The data section of a message buffer is referenced by the data pointer configured in the header section.

Header Partition



Data (Payload) Partition

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
:		MB0 Data3								MB0 Data2								MB0 Data1								MB0 Data0							
:		unused								unused								MB0 Data5								MB0 Data4							
:		MB1 Data3								MB1 Data2								MB1 Data1								MB1 Data0							
:		unused								unused								MB1 Data5								MB1 Data4							
:		MB2 Data3								MB2 Data2								MB2 Data1								MB2 Data0							
:		unused								unused								MB2 Data5								MB2 Data4							
:		MB3 Data3								MB3 Data2								MB3 Data1								MB3 Data0							
:		o								o								o								o							
:		MB3 Data(k)								MB3 Data(k-1)								MB3 Data(k-2)								MB3 Data(k-3)							
:		MBn Data3								MBn Data2								MBn Data1								MBn Data0							
:		o								o								o								o							
:		o								o								o								o							
:		MBn Data(m)								MBn Data(m-1)								MBn Data(m-2)								MBn Data(m-3)							
:		unused								unused								unused								unused							
:		unused								unused								unused								unused							
2046		unused								unused								unused								unused							
2047		unused								unused								unused								unused							

- The data partition starts after the last word of the header partition
- 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
- The programmer has to assure that the data pointers point to addresses within the data partition

Filtering and Masking of FlexRay Frames

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

- Slot Counter + Channel ID
- Slot Counter + Cycle counter + Channel ID

Receive Filtering:

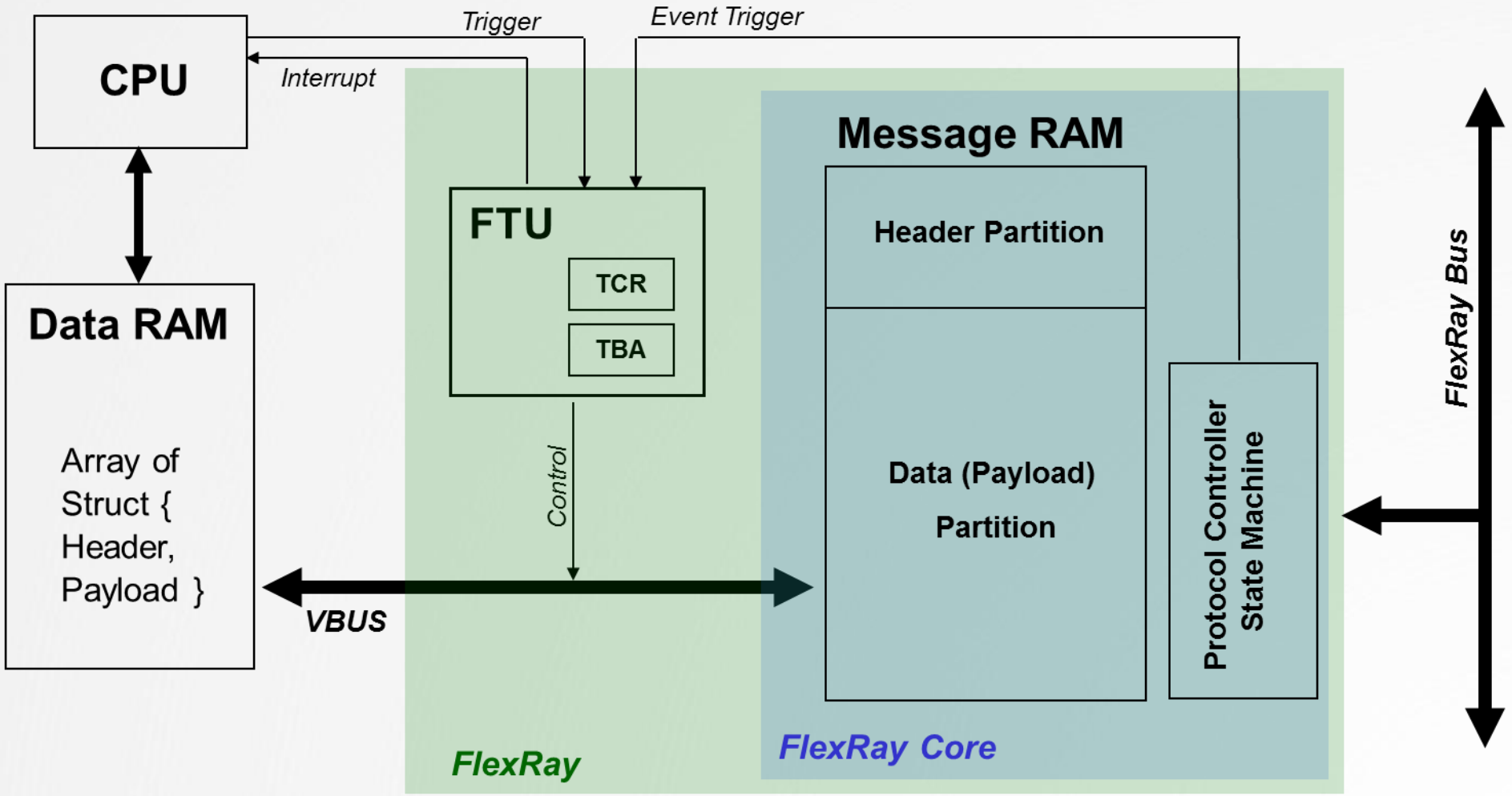
In order to store a received message in a message buffer all configured filters must match.

Transmit Filtering:

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.

FlexRay Transfer Unit (FTU)

FTU Data Transfer Scheme



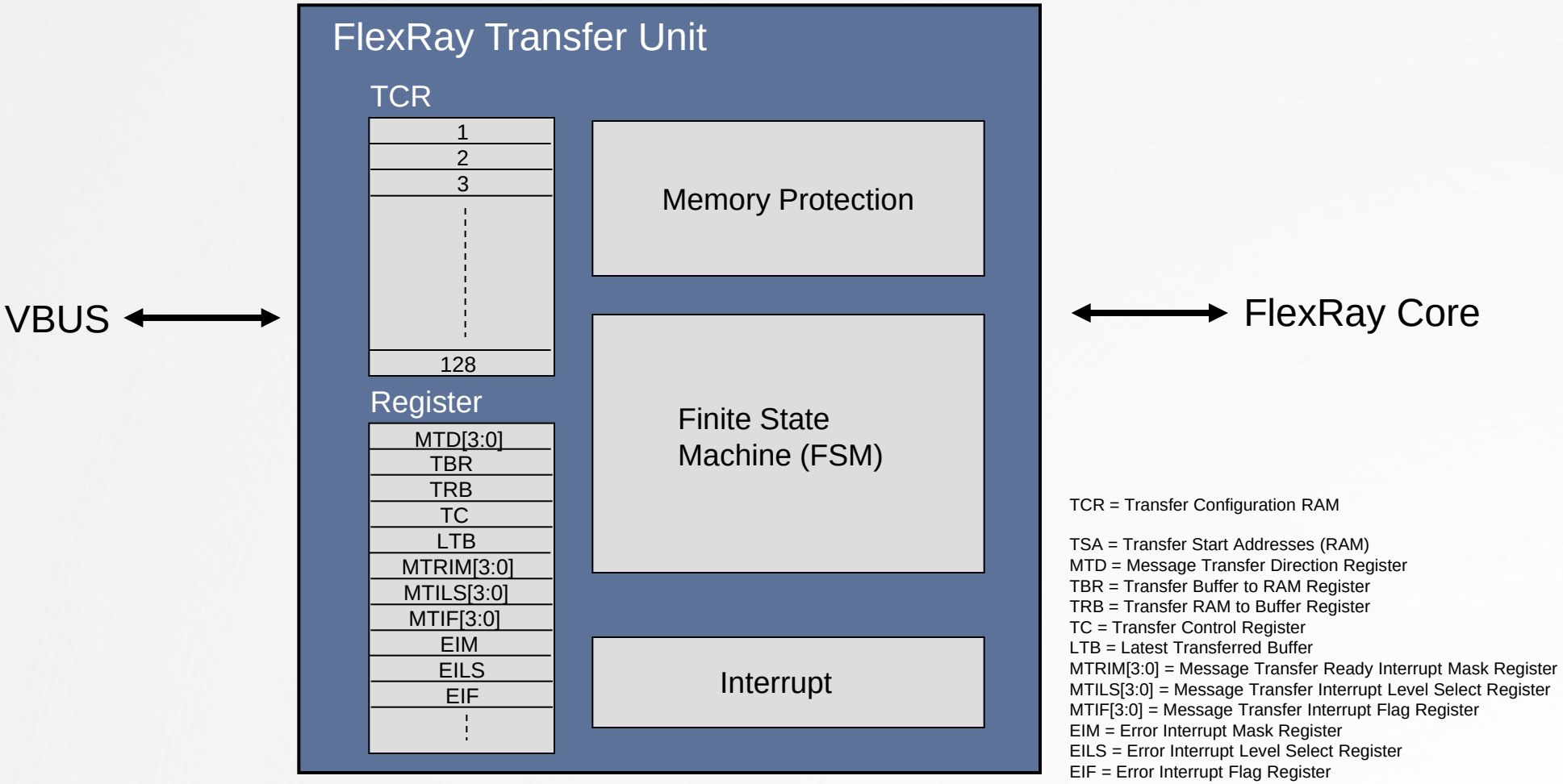
FlexRay Transfer Unit Key Features (1/2)

- Data Transfer without CPU interaction
  - From FlexRay Message RAM to Data RAM (Read)
  - From Data RAM to FlexRay Message RAM (Write)
- Transfer Types
  - data and header section
  - header section only
  - data section only
- Transfer Configuration RAM (with Parity)
  - Configures the transfer sequence
  - Parity protection
- Triggers to Start a Transfer
  - CPU driven (single transfer sequence)
  - Event driven (single or continuous transfer sequence)

FlexRay Transfer Unit Key Features (2/2)

- Different Transfer Conditions
  - If the status flags (header section) of the respective message buffer has been updated
  - If the data section of the respective message buffer has been updated
  - Always
- Maskable interrupt generation when Message Buffer transfer is finished
- Memory Protection Unit
  - One memory section (start- and end address) can be defined
  - No memory section is setup after reset

FlexRay Transfer Unit (FTU) Sub Blocks

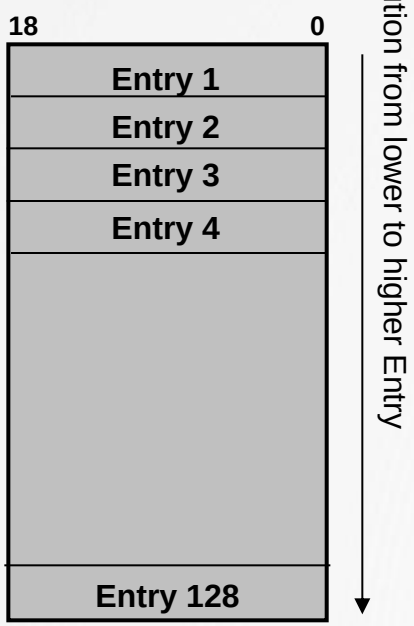


TCR = Transfer Configuration RAM

TSA = Transfer Start Addresses (RAM)
MTD = Message Transfer Direction Register
TBR = Transfer Buffer to RAM Register
TRB = Transfer RAM to Buffer Register
TC = Transfer Control Register
LTB = Latest Transferred Buffer
MTRIM[3:0] = Message Transfer Ready Interrupt Mask Register
MTILS[3:0] = Message Transfer Interrupt Level Select Register
MTIF[3:0] = Message Transfer Interrupt Flag Register
EIM = Error Interrupt Mask Register
EILS = Error Interrupt Level Select Register
EIF = Error Interrupt Flag Register

Transfer Configuration RAM (TCR)

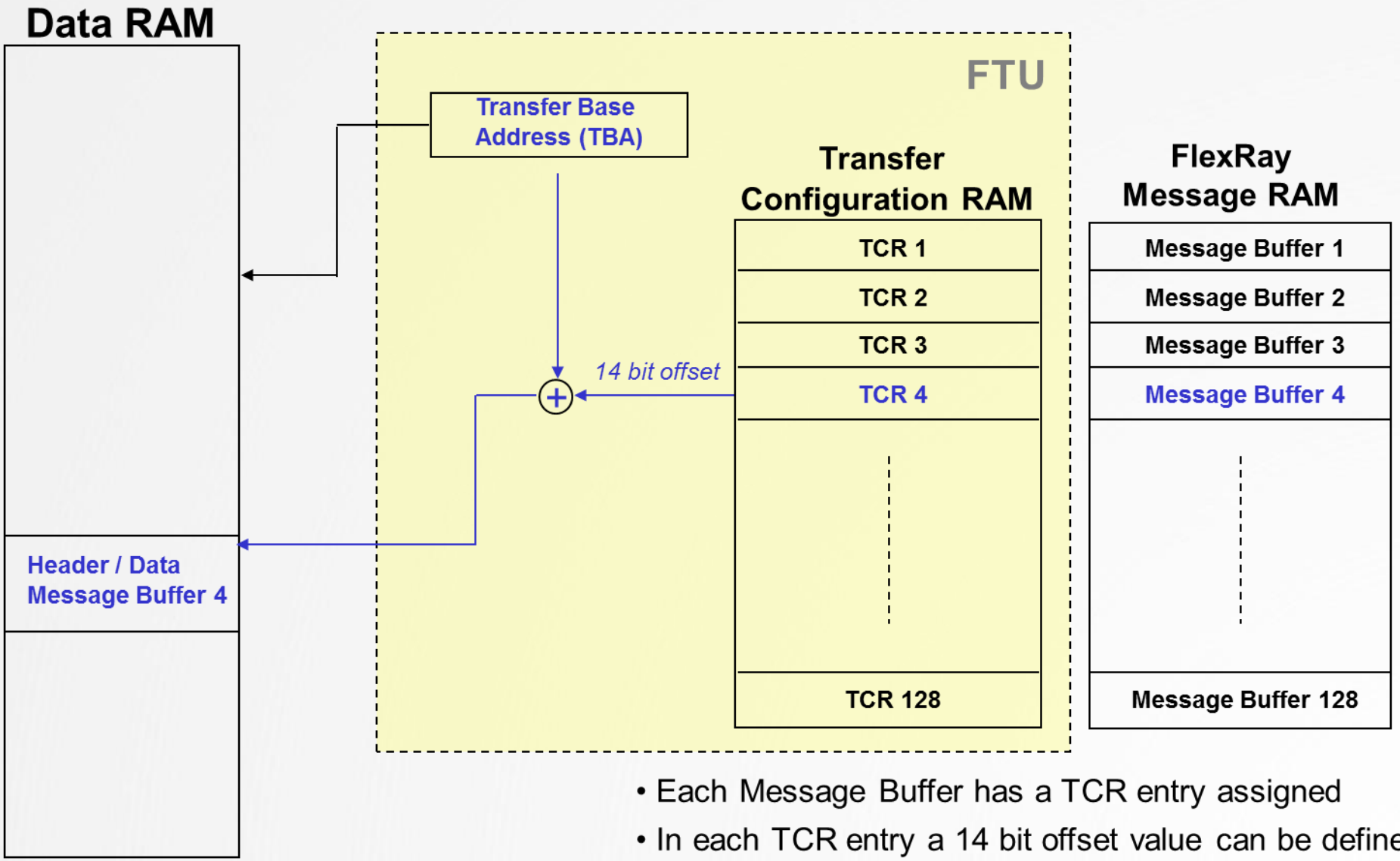
- Used to setup a transfer sequence
- Entry number is equal to the message buffer number of the FlexRay
- A transfer cycle is executed from lower to higher entry number
- The Transfer Configuration RAM is parity protected.



Each Entry defines:

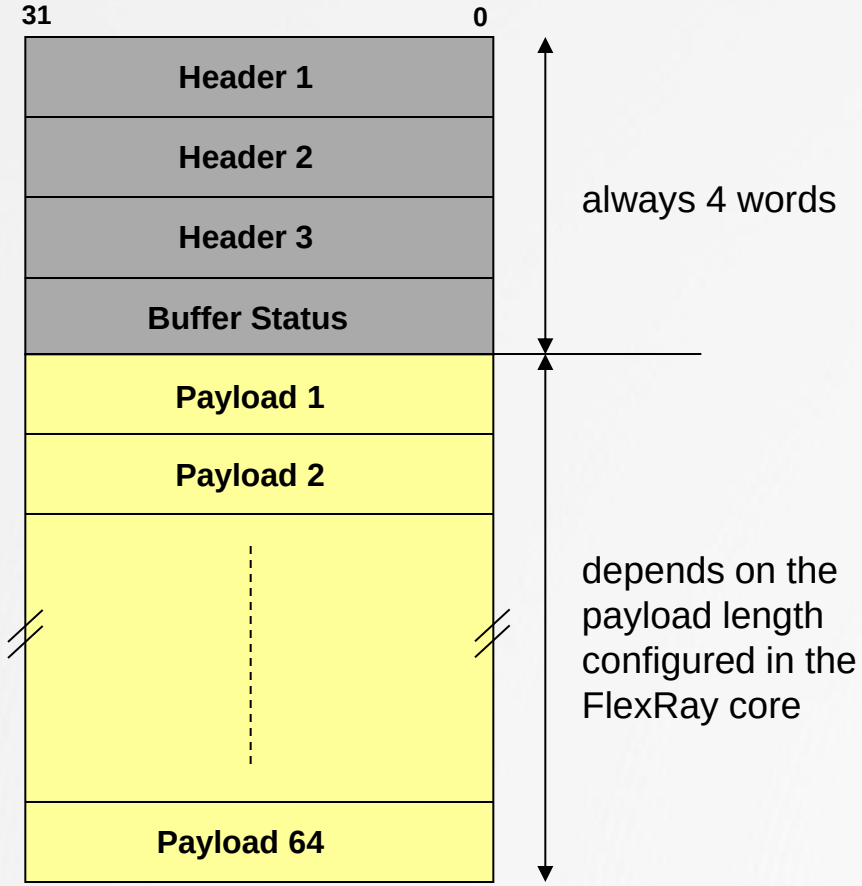
- Transfer Address offset
- Transfer direction
- Transfer of Header and/or Payload
- Automatic send trigger to FlexRay core

FTU Data Addressing



- Each Message Buffer has a TCR entry assigned
- In each TCR entry a 14 bit offset value can be defined

Data Package Storage Order in Data RAM



- Header segments are always 4 words.
- Payload Length is configured in the FlexRay core
- The data transferred by the TU can be selected as:
  - data and header section
  - header section only
  - data section only
- The Buffer Status is transferred only from Communication Controller to System Memory

Possible Transfer Sequence Modes

- Manually
  by setting the according bit in the Trigger Transfer to System Memory Register (TTSM) or the Trigger Transfer to Communication Controller Register (TTCC)
- Event-Driven
  using the Enable Transfer on Event to System Memory Register (ETESM)
  - Only transfers from FlexRay Communication Controller to the System Memory
  - Transfer trigger occurs upon completion of reception or transmission of a frame via the FlexRay bus
- Continuously
  by using the Clear on Event to System Memory (CESM)

Transfer Status Indication

There are 3 registers indicating the transfer status

- Transfer Status Current Buffer (TSCB)
  shows the current transfer buffer status
- Last Transferred Buffer to Communication Controller (LTBCC)
  shows the last completed buffer transfer to the communication controller
- Last Transferred Buffer to System Memory (LTBSM)
  shows the last completed buffer transfer to system memory

Transfer Priority

- The TU will transfer the message buffers from low to high message buffer numbers.
- In case the same buffer is pending in both
  - the Trigger Transfer to Communication Register (TTCC)
  - the Trigger Transfer to System Memory Register (TTSM)

  ⇨ the priority between TTCC and TTSM is determined by the Transfer Priority bit (GC.PRIO) in the TU Global Control Register.

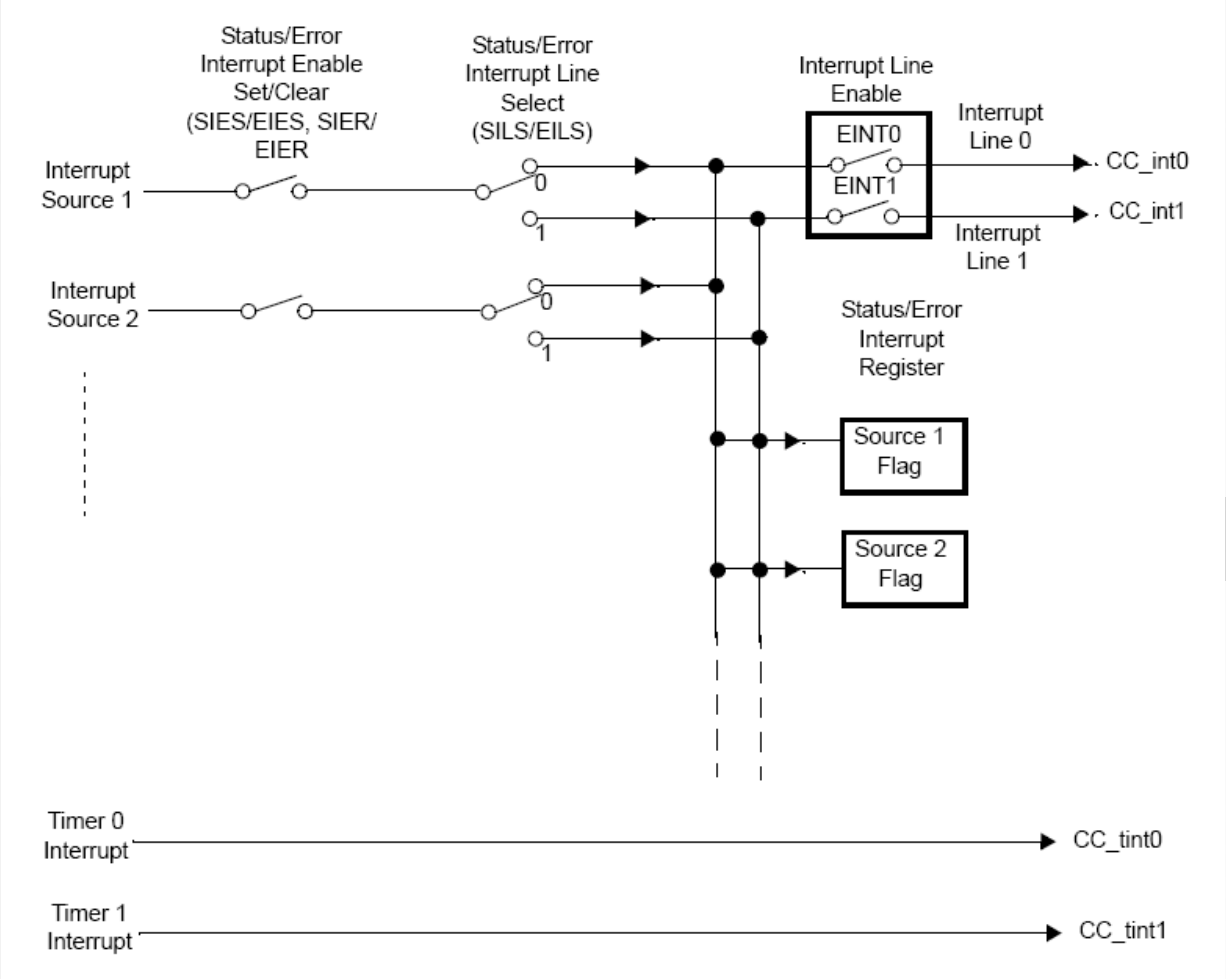
Memory Protection Mechanism

- One memory section can be defined in the data RAM, which allows read and write accesses for the Transfer Unit State Machine.
- In case of a protection violation
  - ➢ the transfer will not be performed
  - ➢ a flag will be set
  - ➢ the Memory Protection Violation interrupt will be activated
  - ➢ The Transfer Unit State Machine will be disabled
- Default the setting of the protection address range:
  - 0x00000000 for startaddress
  - 0x00000000 for endaddress

  ⇨ a valid address range must be setup, before the Transfer Unit can be used.

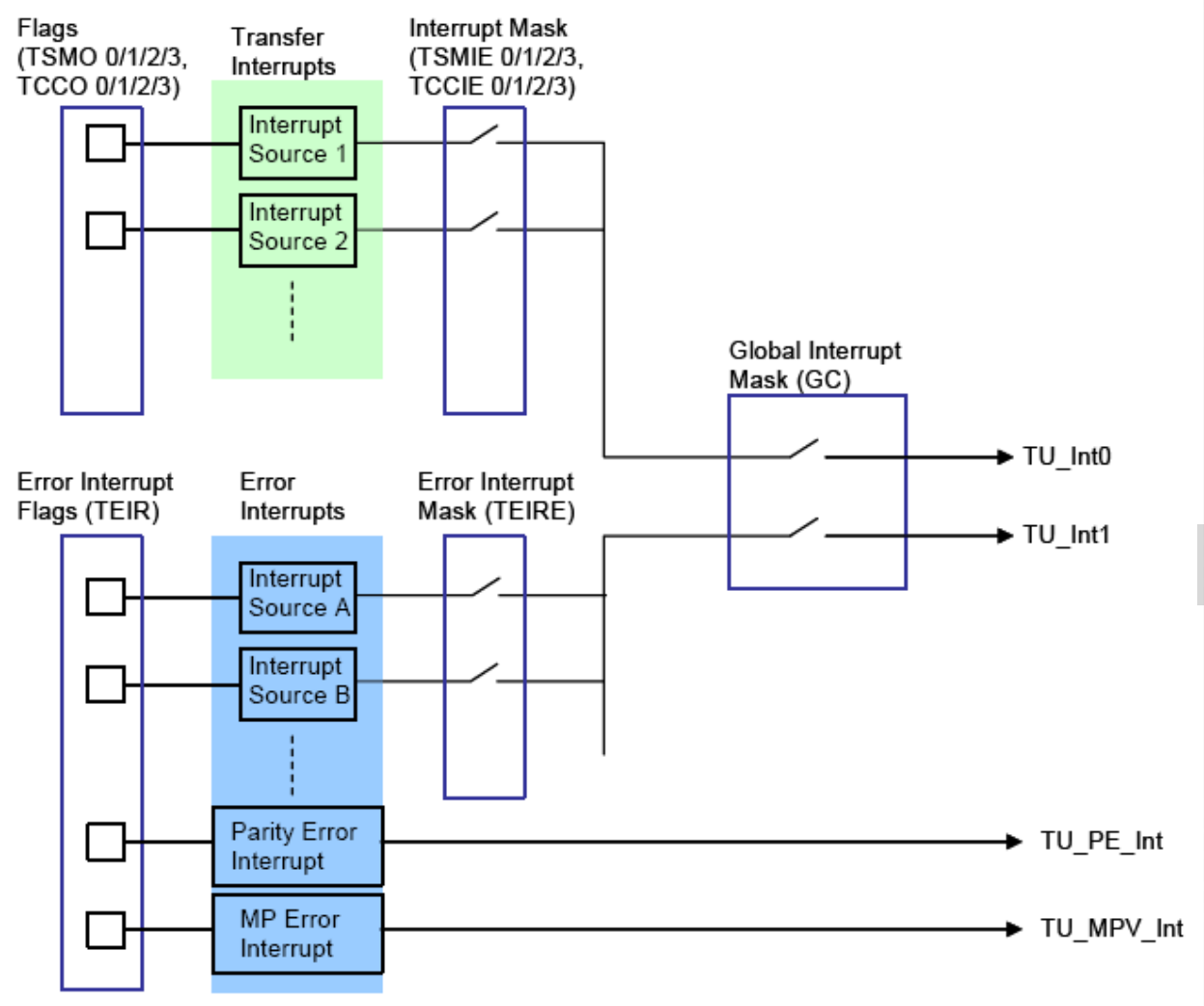
FlexRay Interrupt Structure

FlexRay Core Interrupt Structure



- Error and Status interrupts can be generated
- Each interrupt source can be mapped to one of the 2 interrupt lines
- Timer0 and Timer1 interrupt have private interrupt lines
- Interrupt flag for each interrupt source

FTU Interrupt Structure



- Each transfer channel can generate a transfer interrupt
- Parity and Memory Protection error have private interrupt lines
- Further error interrupts can be:
  - Read transfer error
  - Write transfer error
  - Transfer not ready
  - Forbidden access of CPU to IBF or OBF
- Interrupt flag for each interrupt source

SPRT718

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers’ products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers’ products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265