Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	ARM7®
Core Size	16/32-Bit
Speed	55MHz
Connectivity	CANbus, Ethernet, I²C, SPI, SSC, UART/USART, USB
Peripherals	Brown-out Detect/Reset, DMA, POR, PWM, WDT
Number of I/O	62
Program Memory Size	256KB (256K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	64K x 8
Voltage - Supply (Vcc/Vdd)	1.65V ~ 1.95V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	100-LQFP
Supplier Device Package	100-LQFP (14x14)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/at91sam7x256c-au

9. System Controller

The System Controller manages all vital blocks of the microcontroller: interrupts, clocks, power, time, debug and reset.

The System Controller peripherals are all mapped to the highest 4 Kbytes of address space, between addresses 0xFFFF F000 and 0xFFFF FFFF.

Figure 9-1 on page 24 shows the System Controller Block Diagram.

Figure 8-1 on page 18 shows the mapping of the User Interface of the System Controller peripherals. Note that the Memory Controller configuration user interface is also mapped within this address space.

- Protect Mode
  - Easy debugging by preventing automatic operations
- Fast Forcing
  - Permits redirecting any interrupt source on the fast interrupt
- General Interrupt Mask
  - Provides processor synchronization on events without triggering an interrupt

9.5 Debug Unit

- Comprises:
  - One two-pin UART
  - One Interface for the Debug Communication Channel (DCC) support
  - One set of Chip ID Registers
  - One Interface providing ICE Access Prevention
- Two-pin UART
  - USART-compatible User Interface
  - Programmable Baud Rate Generator
  - Parity, Framing and Overrun Error
  - Automatic Echo, Local Loopback and Remote Loopback Channel Modes
- Debug Communication Channel Support
  - Offers visibility of COMMRX and COMMTX signals from the ARM Processor
- Chip ID Registers
  - Identification of the device revision, sizes of the embedded memories, set of peripherals
  - Chip ID is 0x275C 0A40 (MRL A) for SAM7X512
  - Chip ID is 0x275B 0940 (MRL A or B) for SAM7X256
  - Chip ID is 0x275B 0942 (MRL C) for SAM7X256
  - Chip ID is 0x275A 0740 (MRL A or B) for SAM7X128
  - Chip ID is 0x275A 0742 (MRL C) for SAM7X128

9.6 Periodic Interval Timer

- 20-bit programmable counter plus 12-bit interval counter

9.7 Watchdog Timer

- 12-bit key-protected Programmable Counter running on prescaled SLCK
- Provides reset or interrupt signals to the system
- Counter may be stopped while the processor is in debug state or in idle mode

9.8 Real-time Timer

- 32-bit free-running counter with alarm running on prescaled SLCK
- Programmable 16-bit prescaler for SLCK accuracy compensation

9.9 PIO Controllers

- Two PIO Controllers, each controlling 31 I/O lines
- Fully programmable through set/clear registers
- Multiplexing of two peripheral functions per I/O line
- For each I/O line (whether assigned to a peripheral or used as general-purpose I/O)
  - Input change interrupt

14.4.1 Real-time Timer Mode Register

Register Name: RTT_MR

Access Type: Read-write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	RTTRST	RTTINCIEN	ALMIEN

15	14	13	12	11	10	9	8
RTPRES							

7	6	5	4	3	2	1	0
RTPRES							

- **RTPRES: Real-time Timer Prescaler Value**

Defines the number of SLCK periods required to increment the real-time timer. RTPRES is defined as follows:

RTPRES = 0: The Prescaler Period is equal to 2^{16}

RTPRES ≠ 0: The Prescaler Period is equal to RTPRES.

- **ALMIEN: Alarm Interrupt Enable**

0 = The bit ALMS in RTT_SR has no effect on interrupt.

1 = The bit ALMS in RTT_SR asserts interrupt.

- **RTTINCIEN: Real-time Timer Increment Interrupt Enable**

0 = The bit RTTINC in RTT_SR has no effect on interrupt.

1 = The bit RTTINC in RTT_SR asserts interrupt.

- **RTTRST: Real-time Timer Restart**

1 = Reloads and restarts the clock divider with the new programmed value. This also resets the 32-bit counter.

16.4.3 Watchdog Timer Status Register

Register Name: WDT_SR

Access Type: Read-only

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
–	–	–	–	–	–	WDERR	WDUNF

- **WDUNF: Watchdog Underflow**

0: No Watchdog underflow occurred since the last read of WDT_SR.

1: At least one Watchdog underflow occurred since the last read of WDT_SR.

- **WDERR: Watchdog Error**

0: No Watchdog error occurred since the last read of WDT_SR.

1: At least one Watchdog error occurred since the last read of WDT_SR.

After each transfer, the relevant PDC memory pointer is incremented and the number of transfers left is decremented. When the memory block size is reached, a signal is sent to the peripheral and the transfer stops.

The same procedure is followed, in reverse, for transmit transfers.

22.3.5 Priority of PDC Transfer Requests

The Peripheral DMA Controller handles transfer requests from the channel according to priorities fixed for each product.These priorities are defined in the product datasheet.

If simultaneous requests of the same type (receiver or transmitter) occur on identical peripherals, the priority is determined by the numbering of the peripherals.

If transfer requests are not simultaneous, they are treated in the order they occurred. Requests from the receivers are handled first and then followed by transmitter requests.

22.4 Peripheral DMA Controller (PDC) User Interface

Table 22-1. Register Mapping

Offset	Register	Name	Access	Reset
0x100	Receive Pointer Register	PERIPH(1)_RPR	Read-write	0x0
0x104	Receive Counter Register	PERIPH_RCR	Read-write	0x0
0x108	Transmit Pointer Register	PERIPH_TPR	Read-write	0x0
0x10C	Transmit Counter Register	PERIPH_TCR	Read-write	0x0
0x110	Receive Next Pointer Register	PERIPH_RNPR	Read-write	0x0
0x114	Receive Next Counter Register	PERIPH_RNCR	Read-write	0x0
0x118	Transmit Next Pointer Register	PERIPH_TNPR	Read-write	0x0
0x11C	Transmit Next Counter Register	PERIPH_TNCR	Read-write	0x0
0x120	PDC Transfer Control Register	PERIPH_PTCR	Write-only	-
0x124	PDC Transfer Status Register	PERIPH_PTSR	Read-only	0x0

Note: 1. PERIPH: Ten registers are mapped in the peripheral memory space at the same offset. These can be defined by the user according to the function and the peripheral desired (DBGU, USART, SSC, SPI, MCI, etc.).

23.8.6 AIC FIQ Vector Register

Register Name: AIC_FVR

Access Type: Read-only

Reset Value: 0

31	30	29	28	27	26	25	24
FIQV							

23	22	21	20	19	18	17	16
FIQV							

15	14	13	12	11	10	9	8
FIQV							

7	6	5	4	3	2	1	0
FIQV							

- **FIQV: FIQ Vector Register**

The FIQ Vector Register contains the vector programmed by the user in the Source Vector Register 0. When there is no fast interrupt, the FIQ Vector Register reads the value stored in AIC_SPU.

23.8.7 AIC Interrupt Status Register

Register Name: AIC_ISR

Access Type: Read-only

Reset Value: 0

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
–	–	–	IRQID				

- **IRQID: Current Interrupt Identifier**

The Interrupt Status Register returns the current interrupt source number.

23.8.16 AIC Spurious Interrupt Vector Register

Register Name: AIC_SPU

Access Type: Read-write

Reset Value: 0

31	30	29	28	27	26	25	24
SIVR							
23	22	21	20	19	18	17	16
SIVR							
15	14	13	12	11	10	9	8
SIVR							
7	6	5	4	3	2	1	0
SIVR							

- **SIVR: Spurious Interrupt Vector Register**

The user may store the address of a spurious interrupt handler in this register. The written value is returned in AIC_IVR in case of a spurious interrupt and in AIC_FVR in case of a spurious fast interrupt.

23.8.17 AIC Debug Control Register

Register Name: AIC_DEBUG

Access Type: Read-write

Reset Value: 0

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
–	–	–	–	–	–	GMSK	PROT

- **PROT: Protection Mode**

0 = The Protection Mode is disabled.

1 = The Protection Mode is enabled.

- **GMSK: General Mask**

0 = The nIRQ and nFIQ lines are normally controlled by the AIC.

1 = The nIRQ and nFIQ lines are tied to their inactive state.

25.9.6 PMC Peripheral Clock Status Register

Register Name: PMC_PCSR

Access Type: Read-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	–	–

- **PIDx: Peripheral Clock x Status**

0 = The corresponding peripheral clock is disabled.

1 = The corresponding peripheral clock is enabled.

Note: PID2 to PID31 refer to identifiers as defined in the section “Peripheral Identifiers” in the product datasheet.

27.6 Parallel Input/Output Controller (PIO) User Interface

Each I/O line controlled by the PIO Controller is associated with a bit in each of the PIO Controller User Interface registers. Each register is 32 bits wide. If a parallel I/O line is not defined, writing to the corresponding bits has no effect. Undefined bits read zero. If the I/O line is not multiplexed with any peripheral, the I/O line is controlled by the PIO Controller and PIO_PSR returns 1 systematically.

Table 27-2. Register Mapping

Offset	Register	Name	Access	Reset
0x0000	PIO Enable Register	PIO_PER	Write-only	–
0x0004	PIO Disable Register	PIO_PDR	Write-only	–
0x0008	PIO Status Register	PIO_PSR	Read-only	(1)
0x000C	Reserved			
0x0010	Output Enable Register	PIO_OER	Write-only	–
0x0014	Output Disable Register	PIO_ODR	Write-only	–
0x0018	Output Status Register	PIO_OSR	Read-only	0x0000 0000
0x001C	Reserved			
0x0020	Glitch Input Filter Enable Register	PIO_IFER	Write-only	–
0x0024	Glitch Input Filter Disable Register	PIO_IFDR	Write-only	–
0x0028	Glitch Input Filter Status Register	PIO_IFSR	Read-only	0x0000 0000
0x002C	Reserved			
0x0030	Set Output Data Register	PIO_SODR	Write-only	–
0x0034	Clear Output Data Register	PIO_CODR	Write-only	
0x0038	Output Data Status Register	PIO_ODSR	Read-only or(2) Read-write	–
0x003C	Pin Data Status Register	PIO_PDSR	Read-only	(3)
0x0040	Interrupt Enable Register	PIO_IER	Write-only	–
0x0044	Interrupt Disable Register	PIO_IDR	Write-only	–
0x0048	Interrupt Mask Register	PIO_IMR	Read-only	0x00000000
0x004C	Interrupt Status Register(4)	PIO_ISR	Read-only	0x00000000
0x0050	Multi-driver Enable Register	PIO_MDER	Write-only	–
0x0054	Multi-driver Disable Register	PIO_MDDR	Write-only	–
0x0058	Multi-driver Status Register	PIO_MDSR	Read-only	0x00000000
0x005C	Reserved			
0x0060	Pull-up Disable Register	PIO_PUDR	Write-only	–
0x0064	Pull-up Enable Register	PIO_PUER	Write-only	–
0x0068	Pad Pull-up Status Register	PIO_PUSR	Read-only	0x00000000
0x006C	Reserved			

27.6.11 PIO Controller Clear Output Data Register

Name: PIO_CODR

Access Type: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: Set Output Data**

0 = No effect.

1 = Clears the data to be driven on the I/O line.

27.6.12 PIO Controller Output Data Status Register

Name: PIO_ODSR

Access Type: Read-only or Read-write

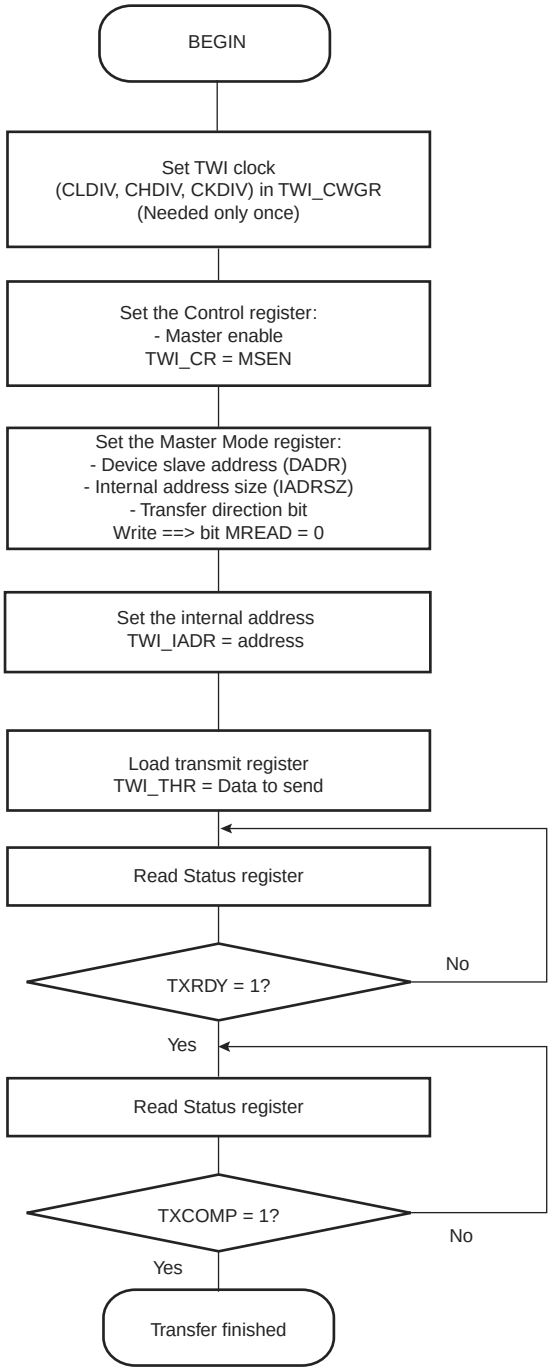
31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: Output Data Status**

0 = The data to be driven on the I/O line is 0.

1 = The data to be driven on the I/O line is 1.

Figure 29-14. TWI Write Operation with Single Data Byte and Internal Address



0	1	6 bits
1	0	7 bits
1	1	8 bits

- **SYNC: Synchronous Mode Select**

0: USART operates in Asynchronous Mode.

1: USART operates in Synchronous Mode.

- **PAR: Parity Type**

PAR			Parity Type
0	0	0	Even parity
0	0	1	Odd parity
0	1	0	Parity forced to 0 (Space)
0	1	1	Parity forced to 1 (Mark)
1	0	x	No parity
1	1	x	Multidrop mode

- **NBSTOP: Number of Stop Bits**

NBSTOP		Asynchronous (SYNC = 0)	Synchronous (SYNC = 1)
0	0	1 stop bit	1 stop bit
0	1	1.5 stop bits	Reserved
1	0	2 stop bits	2 stop bits
1	1	Reserved	Reserved

- **CHMODE: Channel Mode**

CHMODE		Mode Description
0	0	Normal Mode
0	1	Automatic Echo. Receiver input is connected to the TXD pin.
1	0	Local Loopback. Transmitter output is connected to the Receiver Input..
1	1	Remote Loopback. RXD pin is internally connected to the TXD pin.

- **MSBF: Bit Order**

0: Least Significant Bit is sent/received first.

1: Most Significant Bit is sent/received first.

- **MODE9: 9-bit Character Length**

0: CHRL defines character length.

1: 9-bit character length.

33.6.10 PWM Channel Duty Cycle Register

Register Name: PWM_CDTY[0..X-1]

Access Type: Read-write

31	30	29	28	27	26	25	24
CDTY							

23	22	21	20	19	18	17	16
CDTY							

15	14	13	12	11	10	9	8
CDTY							

7	6	5	4	3	2	1	0
CDTY							

Only the first 16 bits (internal channel counter size) are significant.

- **CDTY: Channel Duty Cycle**

Defines the waveform duty cycle. This value must be defined between 0 and CPRD (PWM_CPRx).

- Timestamp interrupt: This interrupt is generated after the reception or the transmission of a start of frame or an end of frame. The value of the internal counter is copied in the CAN_TIMESTP register.

All interrupts are cleared by clearing the interrupt source except for the internal timer counter overflow interrupt and the timestamp interrupt. These interrupts are cleared by reading the CAN_SR register.

36.7.3 CAN Controller Message Handling

36.7.3.1 Receive Handling

Two modes are available to configure a mailbox to receive messages. In ***Receive Mode***, the first message received is stored in the mailbox data register. In ***Receive with Overwrite Mode***, the last message received is stored in the mailbox.

36.7.3.2 Simple Receive Mailbox

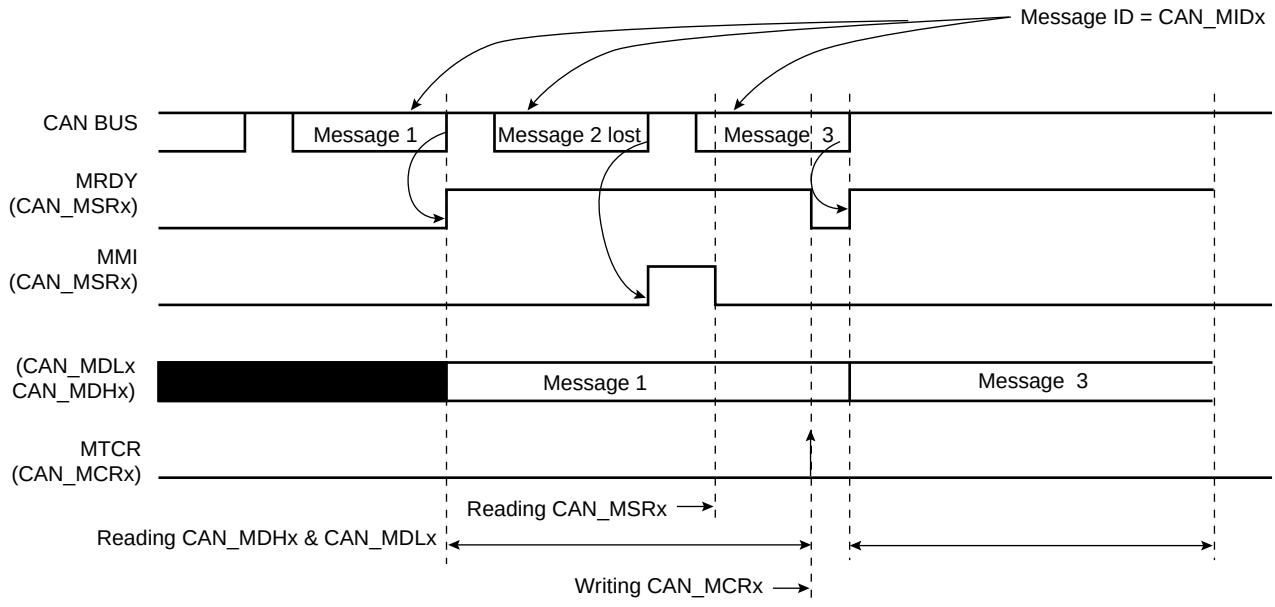
A mailbox is in Receive Mode once the MOT field in the CAN_MMRx register has been configured. Message ID and Message Acceptance Mask must be set before the Receive Mode is enabled.

After Receive Mode is enabled, the MRDY flag in the CAN_MSR register is automatically cleared until the first message is received. When the first message has been accepted by the mailbox, the MRDY flag is set. An interrupt is pending for the mailbox while the MRDY flag is set. This interrupt can be masked depending on the mailbox flag in the CAN_IMR global register.

Message data are stored in the mailbox data register until the software application notifies that data processing has ended. This is done by asking for a new transfer command, setting the MTCR flag in the CAN_MCRx register. This automatically clears the MRDY signal.

The MMI flag in the CAN_MSRx register notifies the software that a message has been lost by the mailbox. This flag is set when messages are received while MRDY is set in the CAN_MSRx register. This flag is cleared by reading the CAN_MSRs register. A receive mailbox prevents from overwriting the first message by new ones while MRDY flag is set in the CAN_MSRx register. See Figure 36-11.

Figure 36-11. Receive Mailbox



Note: In the case of ARM architecture, CAN_MSRx, CAN_MDLx, CAN_MDHx can be read using an optimized ldm assembler instruction.

36.7.3.3 Receive with Overwrite Mailbox

A mailbox is in Receive with Overwrite Mode once the MOT field in the CAN_MMRx register has been configured. Message ID and Message Acceptance masks must be set before Receive Mode is enabled.

36.8.6 CAN Baudrate Register

Name: CAN_BR

Access Type: Read-write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	SMP

23	22	21	20	19	18	17	16
–	BRP						

15	14	13	12	11	10	9	8
–	–	SJW		–	PROPAG		

7	6	5	4	3	2	1	0
–	PHASE1			–	PHASE2		

Any modification on one of the fields of the CANBR register must be done while CAN module is disabled.

To compute the different Bit Timings, please refer to the Section 36.6.4.1 “CAN Bit Timing Configuration” on page 488.

- **PHASE2: Phase 2 segment**

This phase is used to compensate the edge phase error.

$$t_{PHS2} = t_{CSC} \times (PHASE2 + 1)$$

Warning: PHASE2 value must be different from 0.

- **PHASE1: Phase 1 segment**

This phase is used to compensate for edge phase error.

$$t_{PHS1} = t_{CSC} \times (PHASE1 + 1)$$

- **PROPAG: Programming time segment**

This part of the bit time is used to compensate for the physical delay times within the network.

$$t_{PRS} = t_{CSC} \times (PROPAG + 1)$$

- **SJW: Re-synchronization jump width**

To compensate for phase shifts between clock oscillators of different controllers on bus. The controller must re-synchronize on any relevant signal edge of the current transmission. The synchronization jump width defines the maximum of clock cycles a bit period may be shortened or lengthened by re-synchronization.

$$t_{SJW} = t_{CSC} \times (SJW + 1)$$

- **BRP: Baudrate Prescaler.**

This field allows user to program the period of the CAN system clock to determine the individual bit timing.

$$t_{CSC} = (BRP + 1)/MCK$$

The BRP field must be within the range [1, 0x7F], i.e., BRP = 0 is not authorized.

- **SMP: Sampling Mode**

0 = The incoming bit stream is sampled once at sample point.

1 = The incoming bit stream is sampled three times with a period of a MCK clock period, centered on sample point.

SMP Sampling Mode is automatically disabled if BRP = 0.

36.8.12 CAN Message Mode Register

Name: CAN_MMRx

Access Type: Read-write

31	30	29	28	27	26	25	24
–	–	–	–	–	MOT		

23	22	21	20	19	18	17	16
–	–	–	–	PRIOR			

15	14	13	12	11	10	9	8
MTIMEMARK15	MTIMEMARK14	MTIMEMARK13	MTIMEMARK12	MTIMEMARK11	MTIMEMARK10	MTIMEMARK9	MTIMEMARK8

7	6	5	4	3	2	1	0
MTIMEMARK7	MTIMEMARK6	MTIMEMARK5	MTIMEMARK4	MTIMEMARK3	MTIMEMARK2	MTIMEMARK1	MTIMEMARK0

- **MTIMEMARK: Mailbox Timemark**

This field is active in Time Triggered Mode. Transmit operations are allowed when the internal timer counter reaches the Mailbox Timemark. See “Transmitting within a Time Window” on page 504.

In Timestamp Mode, MTIMEMARK is set to 0.

- **PRIOR: Mailbox Priority**

This field has no effect in receive and receive with overwrite modes. In these modes, the mailbox with the lowest number is serviced first.

When several mailboxes try to transmit a message at the same time, the mailbox with the highest priority is serviced first. If several mailboxes have the same priority, the mailbox with the lowest number is serviced first (i.e., MBx0 is serviced before MBx 15 if they have the same priority).

- **MOT: Mailbox Object Type**

This field allows the user to define the type of the mailbox. All mailboxes are independently configurable. Five different types are possible for each mailbox:

MOT			Mailbox Object Type
0	0	0	Mailbox is disabled. This prevents receiving or transmitting any messages with this mailbox.
0	0	1	Reception Mailbox. Mailbox is configured for reception. If a message is received while the mailbox data register is full, it is discarded.
0	1	0	Reception mailbox with overwrite. Mailbox is configured for reception. If a message is received while the mailbox is full, it overwrites the previous message.
0	1	1	Transmit mailbox. Mailbox is configured for transmission.
1	0	0	Consumer Mailbox. Mailbox is configured in reception but behaves as a Transmit Mailbox, i.e., it sends a remote frame and waits for an answer.
1	0	1	Producer Mailbox. Mailbox is configured in transmission but also behaves like a reception mailbox, i.e., it waits to receive a Remote Frame before sending its contents.
1	1	X	Reserved

Table 37-2. Transmit Buffer Descriptor Entry (Continued)

Bit	Function
31	Used. Needs to be zero for the EMAC to read data from the transmit buffer. The EMAC sets this to one for the first buffer of a frame once it has been successfully transmitted. Software has to clear this bit before the buffer can be used again. Note: This bit is only set for the first buffer in a frame unlike receive where all buffers have the Used bit set once used.
30	Wrap. Marks last descriptor in transmit buffer descriptor list.
29	Retry limit exceeded, transmit error detected
28	Transmit underrun, occurs either when hresp is not OK (bus error) or the transmit data could not be fetched in time or when buffers are exhausted in mid frame.
27	Buffers exhausted in mid frame
26:17	Reserved
16	No CRC. When set, no CRC is appended to the current frame. This bit only needs to be set for the last buffer of a frame.
15	Last buffer. When set, this bit indicates the last buffer in the current frame has been reached.
14:11	Reserved
10:0	Length of buffer

37.3.2 Transmit Block

This block transmits frames in accordance with the Ethernet IEEE 802.3 CSMA/CD protocol. Frame assembly starts by adding preamble and the start frame delimiter. Data is taken from the transmit FIFO a word at a time. Data is transmitted least significant nibble first. If necessary, padding is added to increase the frame length to 60 bytes. CRC is calculated as a 32-bit polynomial. This is inverted and appended to the end of the frame, taking the frame length to a minimum of 64 bytes. If the No CRC bit is set in the second word of the last buffer descriptor of a transmit frame, neither pad nor CRC are appended.

In full-duplex mode, frames are transmitted immediately. Back-to-back frames are transmitted at least 96 bit times apart to guarantee the interframe gap.

In half-duplex mode, the transmitter checks carrier sense. If asserted, it waits for it to de-assert and then starts transmission after the interframe gap of 96 bit times. If the collision signal is asserted during transmission, the transmitter transmits a jam sequence of 32 bits taken from the data register and then retry transmission after the back off time has elapsed.

The back-off time is based on an XOR of the 10 least significant bits of the data coming from the transmit FIFO and a 10-bit pseudo random number generator. The number of bits used depends on the number of collisions seen. After the first collision, 1 bit is used, after the second 2, and so on up to 10. Above 10, all 10 bits are used. An error is indicated and no further attempts are made if 16 attempts cause collisions.

If transmit DMA underruns, bad CRC is automatically appended using the same mechanism as jam insertion and the tx_er signal is asserted. For a properly configured system, this should never happen.

If the back pressure bit is set in the network control register in half duplex mode, the transmit block transmits 64 bits of data, which can consist of 16 nibbles of 1011 or in bit-rate mode 64 1s, whenever it sees an incoming frame to force a collision. This provides a way of implementing flow control in half-duplex mode.

37.5.26.9 Late Collisions Register

Register Name: EMAC_LCOL

Access Type: Read-write

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
LCOL							

- **LCOL: Late Collisions**

An 8-bit register counting the number of frames that experience a collision after the slot time (512 bits) has expired. A late collision is counted twice; i.e., both as a collision and a late collision.

37.5.26.10Excessive Collisions Register

Register Name: EMAC_EXCOL

Access Type: Read-write

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
EXCOL							

- **EXCOL: Excessive Collisions**

An 8-bit register counting the number of frames that failed to be transmitted because they experienced 16 collisions.

37.5.26.17Receive Jabbers Register

Register Name: EMAC_RJA

Access Type: Read-write

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
RJB							

- **RJB: Receive Jabbers**

An 8-bit register counting the number of frames received exceeding 1518 bytes (1536 if bit 8 set in network configuration register) in length and have either a CRC error, an alignment error or a receive symbol error.

37.5.26.18Undersize Frames Register

Register Name: EMAC_USF

Access Type: Read-write

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
USF							

- **USF: Undersize Frames**

An 8-bit register counting the number of frames received less than 64 bytes in length but do not have either a CRC error, an alignment error or a receive symbol error.

- EOC[x] already active,
- DRDY already active,
- GOVRE inactive,
- previous data stored in LCDR being neither data from channel "y", nor data from channel "x".

GOVRE should be set but is not.

Problem Fix/Workaround

None

41.3.1.7 ADC: GOVRE Bit is not Set when Disabling a Channel

When disabling channel "y" at the same instant as an end of conversion on channel "x", EOC[x] and DRDY being already active, GOVRE does not rise.

Note: OVRE[x] rises as expected.

Problem Fix/Workaround

None

41.3.1.8 ADC: OVRE Flag Behavior

When the OVRE flag (on channel i) has been set but the related EOC status (of channel i) has been cleared (by a read of CDRi or LCDR), reading the Status register at the same instant as an end of conversion (causing the set of EOC status on channel i), does not lead to a reset of the OVRE flag (on channel i) as expected.

Problem Fix/Workaround:

None

41.3.1.9 ADC: EOC Set although Channel Disabled

If a channel is disabled while a conversion is running and if a read of CDR is performed at the same time as an end of conversion of any channel occurs, the EOC of the channel with the conversion running may rise (whereas it has been disabled).

Problem Fix/Workaround

Do not take into account the EOC of a disabled channel

41.3.1.10 ADC: Spurious Clear of EOC Flag

If "x" and "y" are two successively converted channels and "z" is yet another enabled channel ("z" being neither "x" nor "y"), reading CDR on channel "z" at the same instant as an end of conversion on channel "y" automatically clears EOC[x] instead of EOC[z].

Problem Fix/Workaround

None.

41.3.1.11 ADC: Sleep Mode

If Sleep mode is activated while there is no activity (no conversion is being performed), it will take effect only after a conversion occurs.

Problem Fix/Workaround

To activate sleep mode as soon as possible, it is recommended to write successively, ADC Mode Register (SLEEP) then ADC Control Register (START bit field); to start an analog-to-digital conversion, in order put ADC into sleep mode at the end of this conversion.