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	ARM® Cortex®-M4
Core Size	32-Bit Single-Core
Speed	120MHz
Connectivity	CANbus, EBI/EMI, Ethernet, IrDA, SD, SPI, UART/USART, USB
Peripherals	Brown-out Detect/Reset, DMA, POR, PWM, WDT
Number of I/O	117
Program Memory Size	512KB (512K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	128K x 8
Voltage - Supply (Vcc/Vdd)	1.62V ~ 3.6V
Data Converters	A/D 16x12b; D/A 2x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 105°C (TA)
Mounting Type	Surface Mount
Package / Case	144-LQFP
Supplier Device Package	144-LQFP (20x20)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atsam4e8ea-an

If the software does not configure any priorities, then all exceptions with a configurable priority have a priority of 0. For information about configuring exception priorities see “System Handler Priority Registers” , and “Interrupt Priority Registers” .

Note: Configurable priority values are in the range 0-15. This means that the Reset, Hard fault, and NMI exceptions, with fixed negative priority values, always have higher priority than any other exception.

For example, assigning a higher priority value to IRQ[0] and a lower priority value to IRQ[1] means that IRQ[1] has higher priority than IRQ[0]. If both IRQ[1] and IRQ[0] are asserted, IRQ[1] is processed before IRQ[0].

If multiple pending exceptions have the same priority, the pending exception with the lowest exception number takes precedence. For example, if both IRQ[0] and IRQ[1] are pending and have the same priority, then IRQ[0] is processed before IRQ[1].

When the processor is executing an exception handler, the exception handler is preempted if a higher priority exception occurs. If an exception occurs with the same priority as the exception being handled, the handler is not preempted, irrespective of the exception number. However, the status of the new interrupt changes to pending.

12.4.3.6 Interrupt Priority Grouping

To increase priority control in systems with interrupts, the NVIC supports priority grouping. This divides each interrupt priority register entry into two fields:

- An upper field that defines the *group priority*
- A lower field that defines a *subpriority* within the group.

Only the group priority determines preemption of interrupt exceptions. When the processor is executing an interrupt exception handler, another interrupt with the same group priority as the interrupt being handled does not preempt the handler.

If multiple pending interrupts have the same group priority, the subpriority field determines the order in which they are processed. If multiple pending interrupts have the same group priority and subpriority, the interrupt with the lowest IRQ number is processed first.

For information about splitting the interrupt priority fields into group priority and subpriority, see “Application Interrupt and Reset Control Register” .

12.4.3.7 Exception Entry and Return

Descriptions of exception handling use the following terms:

Preemption

When the processor is executing an exception handler, an exception can preempt the exception handler if its priority is higher than the priority of the exception being handled. See “Interrupt Priority Grouping” for more information about preemption by an interrupt.

When one exception preempts another, the exceptions are called nested exceptions. See “Exception Entry” more information.

Return

This occurs when the exception handler is completed, and:

- There is no pending exception with sufficient priority to be serviced
- The completed exception handler was not handling a late-arriving exception.

The processor pops the stack and restores the processor state to the state it had before the interrupt occurred. See “Exception Return” for more information.

Tail-chaining

This mechanism speeds up exception servicing. On completion of an exception handler, if there is a pending exception that meets the requirements for exception entry, the stack pop is skipped and control transfers to the new exception handler.

Late-arriving

12.6.6.12 SDIV and UDIV

Signed Divide and Unsigned Divide.

Syntax

```
SDIV{cond} {Rd,} Rn, Rm
UDIV{cond} {Rd,} Rn, Rm
```

where:

cond is an optional condition code, see “Conditional Execution” .

Rd is the destination register. If *Rd* is omitted, the destination register is *Rn*.

Rn is the register holding the value to be divided.

Rm is a register holding the divisor.

Operation

SDIV performs a signed integer division of the value in *Rn* by the value in *Rm*.

UDIV performs an unsigned integer division of the value in *Rn* by the value in *Rm*.

For both instructions, if the value in *Rn* is not divisible by the value in *Rm*, the result is rounded towards zero.

Restrictions

Do not use SP and do not use PC.

Condition Flags

These instructions do not change the flags.

Examples

```
SDIV  R0, R2, R4  ; Signed divide, R0 = R2/R4
UDIV  R8, R8, R1  ; Unsigned divide, R8 = R8/R1
```

12.6.11.22 VNEG

Floating-point Negate.

Syntax

```
VNEG{cond}.F32 Sd, Sm
```

where:

cond is an optional condition code, see “Conditional Execution” .

Sd is the destination floating-point value.

Sm is the operand floating-point value.

Operation

This instruction:

1. Negates a floating-point value.
2. Places the results in a second floating-point register.

The floating-point instruction inverts the sign bit.

Restrictions

There are no restrictions.

Condition Flags

These instructions do not change the flags.

12.6.11.23 VNMLA, VNMLS, VNMUL

Floating-point multiply with negation followed by add or subtract.

Syntax

```
VNMLA{cond}.F32 Sd, Sn, Sm
VNMLS{cond}.F32 Sd, Sn, Sm
VNMUL{cond}.F32 {Sd,} Sn, Sm
```

where:

cond is an optional condition code, see “Conditional Execution” .

Sd is the destination floating-point register.

Sn, Sm are the operand floating-point registers.

Operation

The VNMLA instruction:

1. Multiplies two floating-point register values.
2. Adds the negation of the floating-point value in the destination register to the negation of the product.
3. Writes the result back to the destination register.

The VNMLS instruction:

1. Multiplies two floating-point register values.
2. Adds the negation of the floating-point value in the destination register to the product.
3. Writes the result back to the destination register.

The VNMUL instruction:

1. Multiplies together two floating-point register values.
2. Writes the negation of the result to the destination register.

Restrictions

There are no restrictions.

12.9.1.5 Application Interrupt and Reset Control Register

Name: SCB_AIRCR

Access: Read-write

Reset: 0x000000000

31	30	29	28	27	26	25	24
VECTKEYSTAT/VECTKEY							
23	22	21	20	19	18	17	16
VECTKEYSTAT/VECTKEY							
15	14	13	12	11	10	9	8
ENDIANNESS	–				PRIGROUP		
7	6	5	4	3	2	1	0
–					SYSRESETREQ	VECTCLRACTIVE	VECTRESET

The SCB_AIRCR register provides priority grouping control for the exception model, endian status for data accesses, and reset control of the system. To write to this register, write 0x5FA to the VECTKEY field, otherwise the processor ignores the write.

- **VECTKEYSTAT: Register Key**

Read:

Reads as 0xFA05.

- **VECTKEY: Register Key**

Write:

Writes 0x5FA to VECTKEY, otherwise the write is ignored.

- **ENDIANNESS: Data Endianness**

0: Little-endian.

1: Big-endian.

- **PRIGROUP: Interrupt Priority Grouping**

This field determines the split of group priority from subpriority. It shows the position of the binary point that splits the PRI_*n* fields in the Interrupt Priority Registers into separate *group priority* and *subpriority* fields. The table below shows how the PRIGROUP value controls this split:

	Interrupt Priority Level Value, PRI_*N*[7:0]			Number of	
PRIGROUP	**Binary Point**[1]	**Group Priority Bits**	**Subpriority Bits**	**Group Priorities**	**Subpriorities**
0b000	bxxxxxxx.y	[7:1]	None	128	2
0b001	bxxxxxx.yy	[7:2]	[4:0]	64	4
0b010	bxxxxx.yyy	[7:3]	[4:0]	32	8
0b011	bxxxx.yyyy	[7:4]	[4:0]	16	16
0b100	bxxx.yyyyy	[7:5]	[4:0]	8	32

12.11.2.1 MPU Type Register

Name: MPU_TYPE

Access: Read-write

Reset: 0x00000800

31	30	29	28	27	26	25	24
–							

23	22	21	20	19	18	17	16
IREGION							

15	14	13	12	11	10	9	8
DREGION							

7	6	5	4	3	2	1	0
–							SEPARATE

The MPU_TYPE register indicates whether the MPU is present, and if so, how many regions it supports.

- **IREGION: Instruction Region**

Indicates the number of supported MPU instruction regions.

Always contains 0x00. The MPU memory map is unified and is described by the DREGION field.

- **DREGION: Data Region**

Indicates the number of supported MPU data regions:

0x08 = Eight MPU regions.

- **SEPARATE: Separate Instruction**

Indicates support for unified or separate instruction and date memory maps:

0: Unified.

13.6.5.1 SW-DP and JTAG-DP Selection Mechanism

Debug port selection mechanism is done by sending specific **SWDIOTMS** sequence. The JTAG-DP is selected by default after reset.

- Switch from JTAG-DP to SW-DP. The sequence is:
  - Send more than 50 **SWCLKTCK** cycles with **SWDIOTMS** = 1
  - Send the 16-bit sequence on **SWDIOTMS** = 0111100111100111 (0x79E7 MSB first)
  - Send more than 50 **SWCLKTCK** cycles with **SWDIOTMS** = 1
- Switch from SWD to JTAG. The sequence is:
  - Send more than 50 **SWCLKTCK** cycles with **SWDIOTMS** = 1
  - Send the 16-bit sequence on **SWDIOTMS** = 0011110011100111 (0x3CE7 MSB first)
  - Send more than 50 **SWCLKTCK** cycles with **SWDIOTMS** = 1

13.6.6 FPB (Flash Patch Breakpoint)

The FPB:

- Implements hardware breakpoints
- Patches code and data from code space to system space.

The FPB unit contains:

- Two literal comparators for matching against literal loads from Code space, and remapping to a corresponding area in System space.
- Six instruction comparators for matching against instruction fetches from Code space and remapping to a corresponding area in System space.
- Alternatively, comparators can also be configured to generate a Breakpoint instruction to the processor core on a match.

13.6.7 DWT (Data Watchpoint and Trace)

The DWT contains four comparators which can be configured to generate the following:

- PC sampling packets at set intervals
- PC or Data watchpoint packets
- Watchpoint event to halt core

The DWT contains counters for the items that follow:

- Clock cycle (CYCCNT)
- Folded instructions
- Load Store Unit (LSU) operations
- Sleep Cycles
- CPI (all instruction cycles except for the first cycle)
- Interrupt overhead

13.6.8 ITM (Instrumentation Trace Macrocell)

The ITM is an application driven trace source that supports printf style debugging to trace Operating System (OS) and application events, and emits diagnostic system information. The ITM emits trace information as packets which can be generated by three different sources with several priority levels:

- **Software trace**: Software can write directly to ITM stimulus registers. This can be done thanks to the “printf” function. For more information, refer to Section 13.6.8.1 “How to Configure the ITM”.
- **Hardware trace**: The ITM emits packets generated by the DWT.
- **Time stamping**: Timestamps are emitted relative to packets. The ITM contains a 21-bit counter to generate the timestamp.

18.6.4 RTC Calendar Register

Name: RTC_CALR

Address: 0x400E186C

Access: Read-write

31	30	29	28	27	26	25	24
–	–	DATE					

23	22	21	20	19	18	17	16
DAY			MONTH				

15	14	13	12	11	10	9	8
YEAR							

7	6	5	4	3	2	1	0
–	CENT						

- **CENT: Current Century**

The range that can be set is 19 - 20 (gregorian) or 13-14 (persian) (BCD).

The lowest four bits encode the units. The higher bits encode the tens.

- **YEAR: Current Year**

The range that can be set is 00 - 99 (BCD).

The lowest four bits encode the units. The higher bits encode the tens.

- **MONTH: Current Month**

The range that can be set is 01 - 12 (BCD).

The lowest four bits encode the units. The higher bits encode the tens.

- **DAY: Current Day in Current Week**

The range that can be set is 1 - 7 (BCD).

The coding of the number (which number represents which day) is user-defined as it has no effect on the date counter.

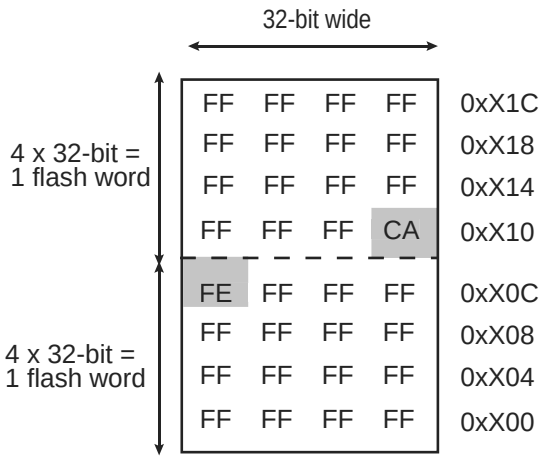
- **DATE: Current Day in Current Month**

The range that can be set is 01 - 31 (BCD).

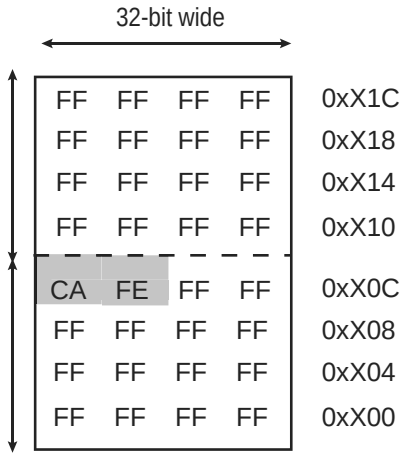
The lowest four bits encode the units. The higher bits encode the tens.

All non-significant bits read zero.

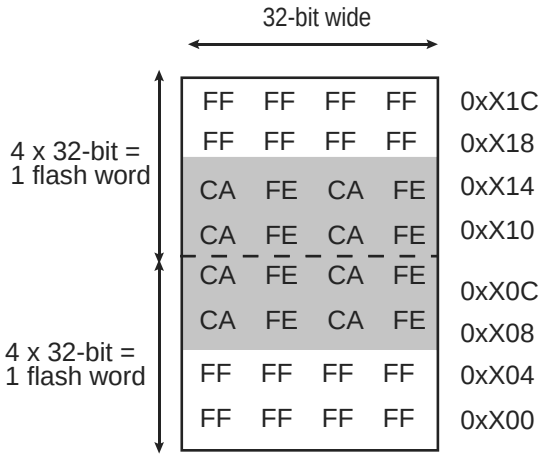
Figure 22-8. Optimized Partial Programming



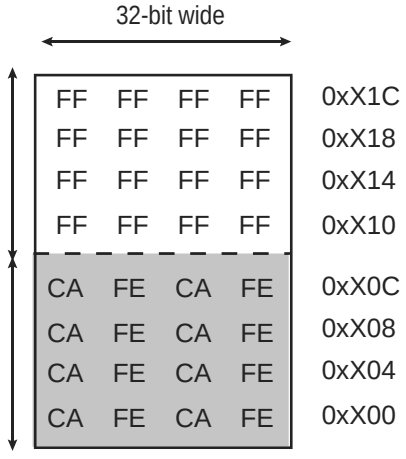
Case 1: 2 x 32-bit modified, across 128-bit boundaries
User programs WP, Flash Controller sends WP
=> Whole page programmed



Case 2: 2 x 32-bit modified, within 1 flash word
User programs WP, Flash Controller sends Write Word
=> Only 1 Word programmed => programming period reduced



Case 3: 4 x 32-bit modified, across 128-bit boundaries
User programs WP, Flash Controller sends WP
=> Whole page programmed



Case 2: 4 x 32-bit modified, within 1 flash word
User programs WP, Flash Controller sends Write Word
=> Only 1 Word programmed => programming period reduced

22.4.3.4 Erase Commands

Erase commands are allowed only on unlocked regions. Depending on the Flash memory, several commands can be used to erase the Flash:

- Erase all memory (EA): all memory is erased. The processor must not fetch code from the Flash memory.
- Erase pages (EPA): 4, 8, 16 or 32 pages are erased in the memory plane. The first page to be erased is specified in the FARG[15:2] field of the MC_FCR register. The first page number must be modulo 4, 8,16 or 32 according to the number of pages to erase at the same time. The processor must not fetch code from the Flash memory.
- Erase Sector (ES): A full memory sector is erased. Sector size depends on the Flash memory. FARG must be set with a page number that is in the sector to be erased. The processor must not fetch code from the Flash memory.

The erase sequence is:

- Erase starts as soon as one of the erase commands and the FARG field are written in the Flash Command Register.

32.9.21 CAN Message Control Register

Name: CAN_MCRx [x=0..7]

Address: 0x4001021C (0)[0], 0x4001023C (0)[1], 0x4001025C (0)[2], 0x4001027C (0)[3], 0x4001029C (0)[4], 0x400102BC (0)[5], 0x400102DC (0)[6], 0x400102FC (0)[7], 0x4001421C (1)[0], 0x4001423C (1)[1], 0x4001425C (1)[2], 0x4001427C (1)[3], 0x4001429C (1)[4], 0x400142BC (1)[5], 0x400142DC (1)[6], 0x400142FC (1)[7]

Access: Write-only

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

23	22	21	20	19	18	17	16
MTCR	MACR	–	MRTR	MDLC			

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **MDLC: Mailbox Data Length Code**

Mailbox Object Type	Description
Receive	No action.
Receive with overwrite	No action.
Transmit	Length of the mailbox message.
Consumer	No action.
Producer	Length of the mailbox message to be sent after the remote frame reception.

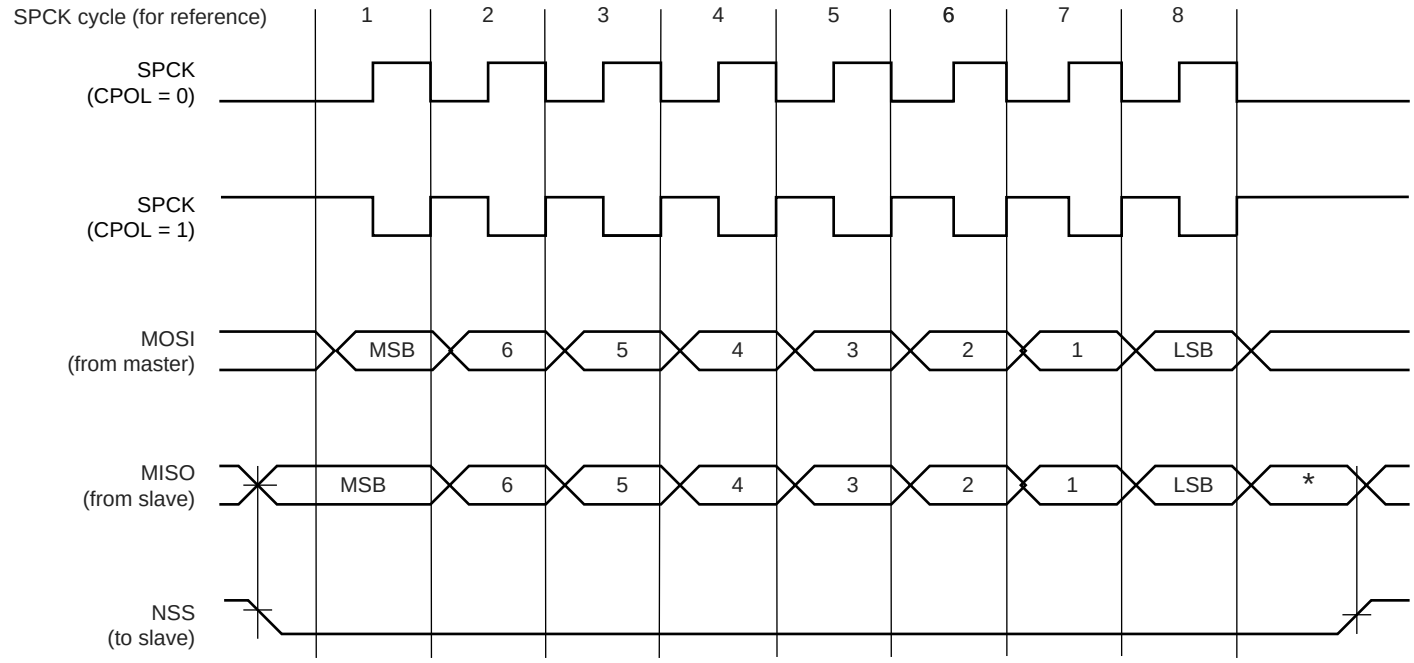
- **MRTR: Mailbox Remote Transmission Request**

Mailbox Object Type	Description
Receive	No action
Receive with overwrite	No action
Transmit	Set the RTR bit in the sent frame
Consumer	No action, the RTR bit in the sent frame is set automatically
Producer	No action

Consumer situations can be handled automatically by setting the mailbox object type in Consumer. This requires only one mailbox.

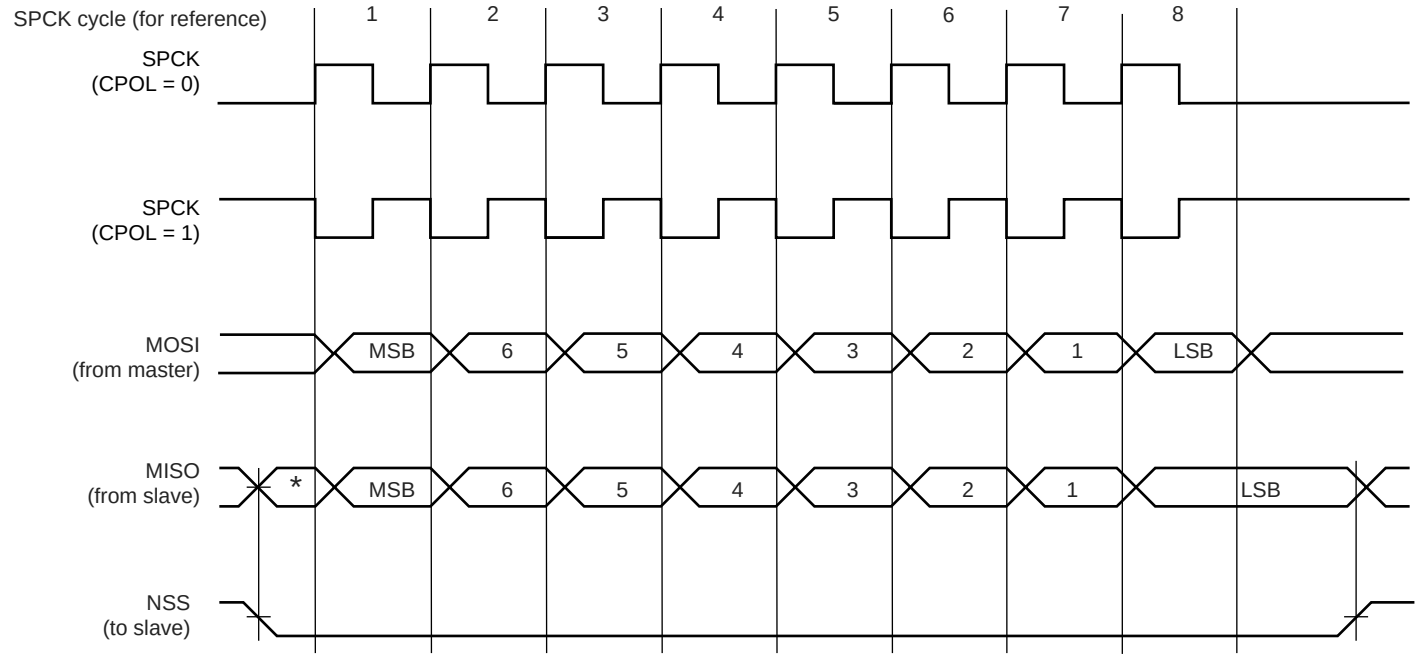
It can also be handled using two mailboxes, one in reception, the other in transmission. The MRTR and the MTCR bits must be set in the same time.

Figure 34-4. SPI Transfer Format (NCPHA = 1, 8 bits per transfer)



* Not defined, but normally MSB of previous character received.

Figure 34-5. SPI Transfer Format (NCPHA = 0, 8 bits per transfer)



* Not defined but normally LSB of previous character transmitted.

34.7.3 Master Mode Operations

When configured in Master Mode, the SPI operates on the clock generated by the internal programmable baud rate generator. It fully controls the data transfers to and from the slave(s) connected to the SPI bus. The SPI drives the chip select line to the slave and the serial clock signal (SPCK).

36. Universal Asynchronous Receiver Transmitter (UART)

36.1 Description

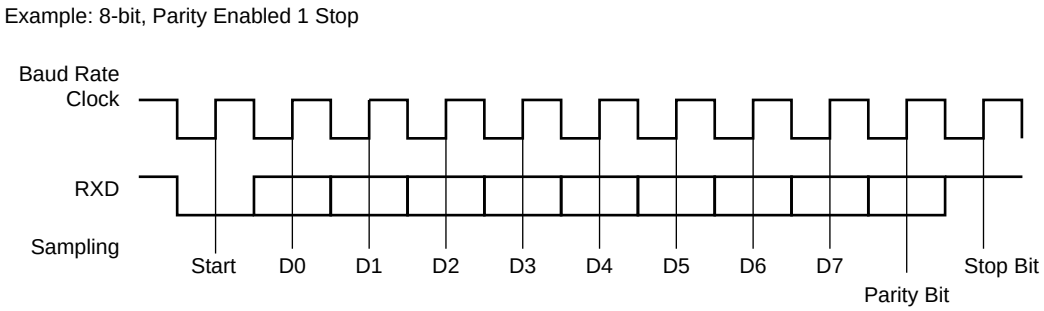
The Universal Asynchronous Receiver Transmitter (UART) features a two-pin UART that can be used for communication and trace purposes and offers an ideal medium for in-situ programming solutions.

Moreover, the association with peripheral DMA controller (PDC) permits packet handling for these tasks with processor time reduced to a minimum.

36.2 Embedded Characteristics

- Two-pin UART
  - Independent receiver and transmitter with a common programmable Baud Rate Generator
  - Even, Odd, Mark or Space Parity Generation
  - Parity, Framing and Overrun Error Detection
  - Automatic Echo, Local Loopback and Remote Loopback Channel Modes
  - Support for two PDC channels with connection to receiver and transmitter

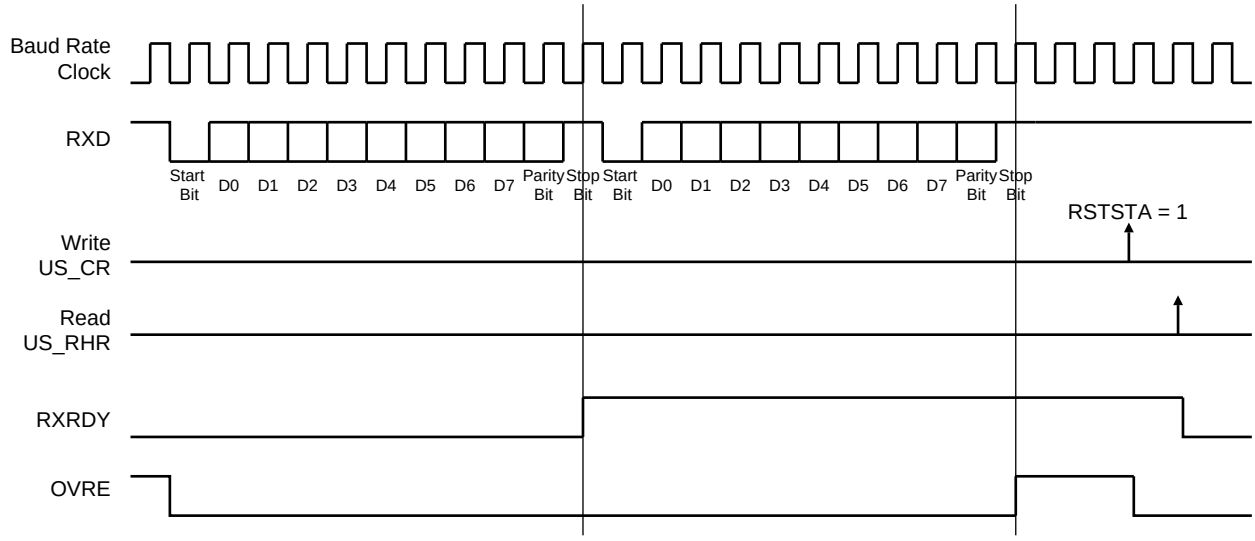
Figure 37-20. Synchronous Mode Character Reception



37.7.3.7 Receiver Operations

When a character reception is completed, it is transferred to the Receive Holding Register (US_RHR) and the RXRDY bit in the Status Register (US_CSR) rises. If a character is completed while the RXRDY is set, the OVRE (Overrun Error) bit is set. The last character is transferred into US_RHR and overwrites the previous one. The OVRE bit is cleared by writing the Control Register (US_CR) with the RSTSTA (Reset Status) bit to 1.

Figure 37-21. Receiver Status



37.7.3.8 Parity

The USART supports five parity modes selected by programming the PAR field in the Mode Register (US_MR). The PAR field also enables the Multidrop mode, see “Multidrop Mode” on page 841. Even and odd parity bit generation and error detection are supported.

If even parity is selected, the parity generator of the transmitter drives the parity bit to 0 if a number of 1s in the character data bit is even, and to 1 if the number of 1s is odd. Accordingly, the receiver parity checker counts the number of received 1s and reports a parity error if the sampled parity bit does not correspond. If odd parity is selected, the parity generator of the transmitter drives the parity bit to 1 if a number of 1s in the character data bit is even, and to 0 if the number of 1s is odd. Accordingly, the receiver parity checker counts the number of received 1s and reports a parity error if the sampled parity bit does not correspond. If the mark parity is used, the parity generator of the transmitter drives the parity bit to 1 for all characters. The receiver parity checker reports an error if the parity bit is sampled to 0. If the space parity is used, the parity generator of the transmitter drives the parity bit to 0 for all characters. The receiver parity checker reports an error if the parity bit is sampled to 1. If parity is disabled, the transmitter does not generate any parity bit and the receiver does not report any parity error.

A time base must be defined on channel 2 by writing the TC_RC2 period register. Channel 2 must be configured in waveform mode (WAVE bit field set) in TC_CMR2 register. WAVSEL bit field must be defined with 0x10 to clear the counter by comparison and matching with TC_RC value. ACPC field must be defined at 0x11 to toggle TIOA output.

This time base is automatically fed back to TIOA of channel 0 when QDEN and SPEEDEN are set.

Channel 0 must be configured in capture mode (WAVE = 0 in TC_CMR0). ABETRG bit field of TC_CMR0 must be configured at 1 to get TIOA as a trigger for this channel.

EDGTRG can be set to 0x01, to clear the counter on a rising edge of the TIOA signal and LDRA field must be set accordingly to 0x01, to load TC_RA0 at the same time as the counter is cleared (LDRB must be set to 0x01). As a consequence, at the end of each time base period the differentiation required for the speed calculation is performed.

The process must be started by configuring the TC_CR register with CLKEN and SWTRG.

The speed can be read on TC_RA0 register in TC_CMR0.

Channel 1 can still be used to count the number of revolutions of the motor.

38.6.17 2-bit Gray Up/Down Counter for Stepper Motor

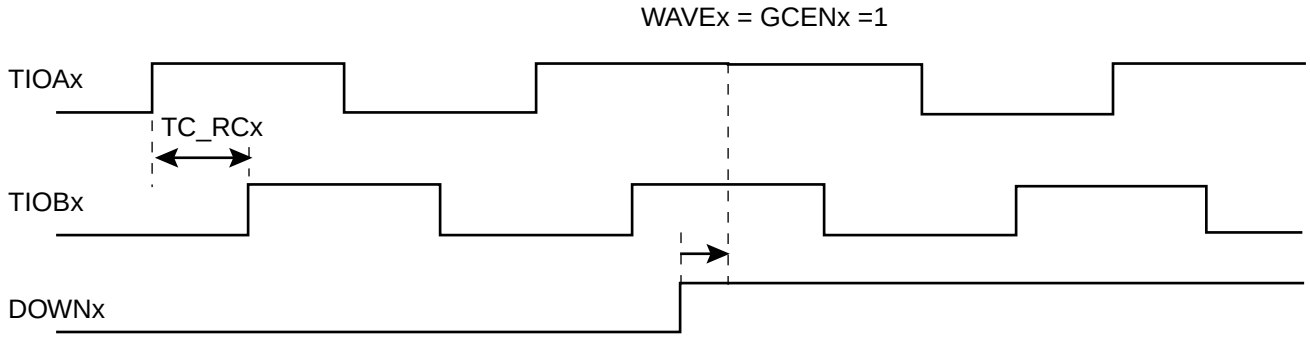
Each channel can be independently configured to generate a 2-bit gray count waveform on corresponding TIOA,TIOB outputs by means of GCEN bit in TC_SMMRx registers.

Up or Down count can be defined by writing bit DOWN in TC_SMMRx registers.

It is mandatory to configure the channel in WAVE mode in TC_CMR register.

The period of the counters can be programmed on TC_RCx registers.

Figure 38-22. 2-bit Gray Up/Down Counter.



38.6.18 Write Protection System

In order to bring security to the Timer Counter, a write protection system has been implemented.

The write protection mode prevent the write of TC_BMR, TC_FMR, TC_CMRx, TC_SMMRx, TC_RAx, TC_RBx, TC_RCx registers. When this mode is enabled and one of the protected registers write, the register write request canceled.

Due to the nature of the write protection feature, enabling and disabling the write protection mode requires the use of a security code. Thus when enabling or disabling the write protection mode the WPKEY field of the TC_WPMR register must be filled with the “TIM” ASCII code (corresponding to 0x54494D) otherwise the register write will be canceled.

38.6.19 Fault Mode

At anytime, the TC_RCx registers can be used to perform a comparison on the respective current channel counter value (TC_CVx) with the value of TC_RCx register.

The CPCSx flags can be set accordingly and an interrupt can be generated.

This interrupt is processed but requires an unpredictable amount of time to be achieve the required action.

39.8.2 Data Transfer Operation

The High Speed MultiMedia Card allows several read/write operations (single block, multiple blocks, stream, etc.). These kinds of transfer can be selected setting the Transfer Type (TRTYP) field in the HSMCI Command Register (HSMCI_CMDR).

These operations can be done using the features of the Peripheral DMA Controller (PDC). If the PDCMODE bit is set in HSMCI_MR, then all reads and writes use the PDC facilities.

In all cases, the block length (BLKLEN field) must be defined either in the Mode Register HSMCI_MR, or in the Block Register HSMCI_BLKR. This field determines the size of the data block.

Consequent to MMC Specification 3.1, two types of multiple block read (or write) transactions are defined (the host can use either one at any time):

- Open-ended/Infinite Multiple block read (or write):

  The number of blocks for the read (or write) multiple block operation is not defined. The card will continuously transfer (or program) data blocks until a stop transmission command is received.
- Multiple block read (or write) with pre-defined block count (since version 3.1 and higher):

  The card will transfer (or program) the requested number of data blocks and terminate the transaction. The stop command is not required at the end of this type of multiple block read (or write), unless terminated with an error. In order to start a multiple block read (or write) with pre-defined block count, the host must correctly program the HSMCI Block Register (HSMCI_BLKR). Otherwise the card will start an open-ended multiple block read. The BCNT field of the Block Register defines the number of blocks to transfer (from 1 to 65535 blocks). Programming the value 0 in the BCNT field corresponds to an infinite block transfer.

39.8.3 Read Operation

The following flowchart shows how to read a single block with or without use of PDC facilities. In this example (see Figure 39-8), a polling method is used to wait for the end of read. Similarly, the user can configure the Interrupt Enable Register (HSMCI_IER) to trigger an interrupt at the end of read.

- **RCRCE: Response CRC Error**

0 = No error.

1 = A CRC7 error has been detected in the response. Cleared when writing in the HSMCI_CMDR.

- **RENDE: Response End Bit Error**

0 = No error.

1 = The end bit of the response has not been detected. Cleared when writing in the HSMCI_CMDR.

- **RTOE: Response Time-out Error**

0 = No error.

1 = The response time-out set by MAXLAT in the HSMCI_CMDR has been exceeded. Cleared when writing in the HSMCI_CMDR.

- **DCRCE: Data CRC Error**

0 = No error.

1 = A CRC16 error has been detected in the last data block. Cleared by reading in the HSMCI_SR register.

- **DTOE: Data Time-out Error**

0 = No error.

1 = The data time-out set by DTOCYC and DTOMUL in HSMCI_DTOR has been exceeded. Cleared by reading in the HSMCI_SR register.

- **CSTOE: Completion Signal Time-out Error**

0 = No error.

1 = The completion signal time-out set by CSTOCYC and CSTOMUL in HSMCI_CSTOR has been exceeded. Cleared by reading in the HSMCI_SR register. Cleared by reading in the HSMCI_SR register.

- **FIFOEMPTY: FIFO empty flag**

0 = FIFO contains at least one byte.

1 = FIFO is empty.

- **XFRDONE: Transfer Done flag**

0 = A transfer is in progress.

1 = Command Register is ready to operate and the data bus is in the idle state.

- **ACKRCV: Boot Operation Acknowledge Received**

0 = No Boot acknowledge received since the last read of the status register.

1 = A Boot acknowledge signal has been received. Cleared by reading the HSMCI_SR register.

- **ACKRCVE: Boot Operation Acknowledge Error**

0 = No error

1 = Corrupted Boot Acknowledge signal received.

- **OVRE: Overrun**

0 = No error.

1 = At least one 8-bit received data has been lost (not read). Cleared when sending a new data transfer command.

40.7.29 PWM Spread Spectrum Register

Name: PWM_SSPR

Address: 0x400000A0

Access: Read-write

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

23	22	21	20	19	18	17	16
SPRD							

15	14	13	12	11	10	9	8
SPRD							

7	6	5	4	3	2	1	0
SPRD							

This register can only be written if the bits WPSWS3 and WPHWS3 are cleared in “PWM Write Protect Status Register” on page 1070.

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

- **SPRD: Spread Spectrum Limit Value**

The spread spectrum limit value defines the range for the spread spectrum counter. It is introduced in order to achieve constant varying PWM period for the output waveform.

- **SPRDM: Spread Spectrum Counter Mode**

0 = TRIANGULAR mode. The spread spectrum counter starts to count from -SPRD when the channel 0 is enabled and counts upwards at each PWM period. When it reach +SPRD, it restarts to count from -SPRD again.

1 = RANDOM mode. The spread spectrum counter is loaded with a new random value at each PWM period. This random value is uniformly distributed and is between -SPRD and +SPRD.

42.7.2 ACC Mode Register

Name: ACC_MR

Address: 0x400BC004

Access: 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
–	FE	SELFS	INV	–	EDGETYP		ACEN
7	6	5	4	3	2	1	0
–	SELPLUS			–	SELMINUS		

This register can only be written if the WPEN bit is cleared in the ACC Write Protect Mode Register.

- **SELMINUS: SELection for MINUS comparator input**

0..7 = selects the input to apply on analog comparator SELMINUS comparison input.

Value	Name	Description
0	TS	Select TS
1	ADVREF	Select ADVREF
2	DAC0	Select DAC0
3	DAC1	Select DAC1
4	AD0	Select AD0
5	AD1	Select AD1
6	AD2	Select AD2
7	AD3	Select AD3

- **SELPLUS: SELection for PLUS comparator input**

0..7 = selects the input to apply on analog comparator SELPLUS comparison input.

Value	Name	Description
0	AD0	Select AD0
1	AD1	Select AD1
2	AD2	Select AD2
3	AD3	Select AD3
4	AD4	Select AD4
5	AD5	Select AD5
6	AD6	Select AD6
7	AD7	Select AD7

- **ACEN: Analog Comparator ENable**

0 (DIS) = Analog Comparator Disabled.

1 (EN) = Analog Comparator Enabled.

44.7.1 Network Control Register

Name: GMAC_NCR

Address: 0x40034000

Access: Read-write

31	30	29	28	27	26	25	24
–							

23	22	21	20	19	18	17	16
					FNP	TXPBPF	ENPBPR

15	14	13	12	11	10	9	8
SRTSM	RDS	–	TXZQPF	TXPF	THALT	TSTART	BP

7	6	5	4	3	2	1	0
WESTAT	INCSTAT	CLRSTAT	MPE	TXEN	RXEN	LBL	–

- **LBL: Loop Back Local**

Connects GTX to GRX, GTXEN to GRXDV and forces full duplex mode. GRXCK and GTXCK may malfunction as the GMAC is switched into and out of internal loop back. It is important that receive and transmit circuits have already been disabled when making the switch into and out of internal loop back.

- **RXEN: Receive Enable**

When set, RXEN enables the GMAC to receive data. When reset frame reception stops immediately and the receive pipeline will be cleared. The Receive Queue Pointer Register is unaffected.

- **TXEN: Transmit Enable**

When set, TXEN enables the GMAC transmitter to send data. When reset transmission will stop immediately, the transmit pipeline and control registers will be cleared and the Transmit Queue Pointer Register will reset to point to the start of the transmit descriptor list.

- **MPE: Management Port Enable**

Set to one to enable the management port. When zero, forces MDIO to high impedance state and MDC low.

- **CLRSTAT: Clear Statistics Registers**

This bit is write only. Writing a one clears the statistics registers.

- **INCSTAT: Increment Statistics Registers**

This bit is write only. Writing a one increments all the Statistics Registers by one for test purposes.

- **WESTAT: Write Enable for Statistics Registers**

Setting this bit to one makes the Statistics Registers writable for functional test purposes.

- **BP: Back pressure**

If set in 10M or 100M half duplex mode, forces collisions on all received frames.

- **TSTART: Start Transmission**

Writing one to this bit starts transmission.

- **THALT: Transmit Halt**

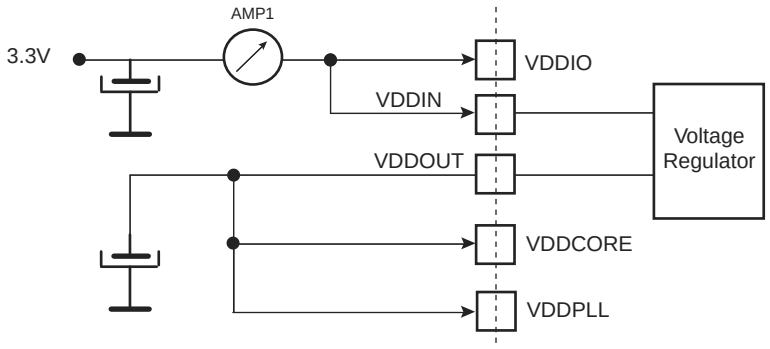
46.3 Power Consumption

- Power consumption of the device according to the different Low Power Mode Capabilities (Backup, Wait, Sleep) and Active Mode.
- Power consumption on power supply in different modes: Backup, Wait, Sleep and Active.
- Power consumption by peripheral: calculated as the difference in current measurement after having enabled then disabled the corresponding clock.
- All power consumption values are based on characterization. Note that these values are not covered by test limits in production.

46.3.1 Backup Mode Current Consumption

The backup mode configuration and measurements are defined as follow.

Figure 46-4. Measurement Setup



46.3.1.1 Configuration A: Embedded Slow Clock RC Oscillator Enabled

- Supply Monitor on VDDIO is disabled
- RTT used
- BOD disabled
- One WKUPx enabled
- Current measurement on AMP1 (see Figure 46-4)

46.3.1.2 Configuration B: 32768 kHz Crystal Oscillator Enabled

- Supply Monitor on VDDIO is disabled
- RTT used
- BOD disabled
- One WKUPx enabled
- Current measurement on AMP1 (see Figure 46-4)

Table 46-9. Power Consumption for Backup Mode Configuration A and B

BACKUP Total Consumption	Typical value @25 degrees		Maximum Value @85 degrees	Maximum Value @105 degrees	Unit
Conditions	(AMP1) Configuration A	(AMP1) Configuration B	(AMP1) Configuration A	(AMP1) Configuration A	Unit
VDDIO = 3.6V	2.0	1.9	7.2	15.4	µA
VDDIO = 3.3V	1.7	1.6	6.9	12.0	
VDDIO = 3.0V	1.5	1.5	6.2	11.2	
VDDIO = 2.5V	1.3	1.2	5.5	9.8	
VDDIO = 1.8V	1	0.9	4.6	8.3	

Table 46-10. Sleep Mode Current Consumption versus Master Clock (MCK) Variation with PLLA

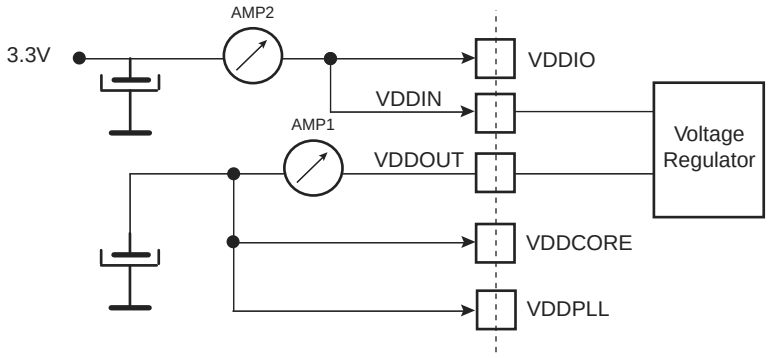
Core Clock/MCK (MHz)	VDDCORE Consumption (AMP1)	Total Consumption (AMP2)	Unit
120	9.8	11.4	mA
100	8.2	9.5	mA
84	7.1	9.4	mA
64	5.5	7.2	mA
48	4.2	5.5	mA
32	3.0	4.7	mA
24	2.3	3.5	mA

Table 46-11. Sleep Mode Current Consumption versus Master Clock (MCK) Variation with FAST RC

Core Clock/MCK (MHz)	VDDCORE Consumption (AMP1)	Total Consumption (AMP2)	Unit
12	1.11	1.14	mA
8	0.77	0.8	mA
4	0.45	0.48	mA
2	0.3	0.33	mA
1	0.22	0.25	mA
0.5	0.18	0.21	mA
0.25	0.16	0.19	mA

46.3.2.2 Wait Mode

Figure 46-7. Measurement Setup for Wait Mode



- VDDIO = VDDIN = 3.6V
- Core Clock and Master Clock stopped
- Current measurement as shown in the above figure
- All peripheral clocks deactivated
- BOD disabled
- RTT enabled

Table 46-12 gives current consumption in typical conditions.

Note: 1. Value from characterization, not tested in production.