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	Obsolete
Core Processor	ARM7TDMI
Core Size	16/32-Bit
Speed	66MHz
Connectivity	I²C, MMC, SPI, SSC, UART/USART, USB
Peripherals	WDT
Number of I/O	63
Program Memory Size	256KB (256K x 8)
Program Memory Type	ROM
EEPROM Size	-
RAM Size	96K x 8
Voltage - Supply (Vcc/Vdd)	1.65V ~ 1.95V
Data Converters	-
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/at91rm3400-au-002

PIO Controller B Multiplexing

Table 3. Multiplexing PIO controller B

PIO Controller B			Application Usage	
I/O Line	Peripheral A	Peripheral B	Function	Comments
PB0	TF0	TIOB3		
PB1	TK0	TCLK3		
PB2	TD0	RTS2		
PB3	RD0	RTS3		
PB4	RK0	PCK0		
PB5	RF0	TIOA3		
PB6	TF1	TIOB4		
PB7	TK1	TCLK4		
PB8	TD1	NPCS1		
PB9	RD1	NPCS2		
PB10	RK1	PCK1		
PB11	RF1	TIOA4		
PB12	TF2	TIOB5		
PB13	TK2	TCLK5		
PB14	TD2	NPCS3		
PB15	RD2	PCK1		
PB16	RK2	PCK2		
PB17	RF2	TIOA5		
PB18	RTS3	MCCDB		
PB19	CTS3	MCDB0		
PB20	TXD3	DTR1		
PB21	RXD3			
PB22	SCK3	PCK3		
PB23	FIQ			
PB24	IRQ0	TD0		
PB25	IRQ1	TD1		
PB26	IRQ2	TD2		
PB27	IRQ3	DTXD		
PB28	IRQ4	MCDB1		
PB29	IRQ5	MCDB2		
PB30	IRQ6	MCDB3		

Table 8. ARM Instruction Mnemonic List

Mnemonic	Operation
MOV	Move
ADD	Add
SUB	Subtract
RSB	Reverse Subtract
CMP	Compare
TST	Test
AND	Logical AND
EOR	Logical Exclusive OR
MUL	Multiply
SMULL	Sign Long Multiply
SMLAL	Signed Long Multiply Accumulate
MSR	Move to Status Register
B	Branch
BX	Branch and Exchange
LDR	Load Word
LDRSH	Load Signed Halfword
LDRSB	Load Signed Byte
LDRH	Load Half Word
LDRB	Load Byte
LDRBT	Load Register Byte with Translation
LDRT	Load Register with Translation
LDM	Load Multiple
SWP	Swap Word
MCR	Move To Coprocessor
LDC	Load To Coprocessor

Mnemonic	Operation
CDP	Coprocessor Data Processing
MVN	Move Not
ADC	Add with Carry
SBC	Subtract with Carry
RSC	Reverse Subtract with Carry
CMN	Compare Negated
TEQ	Test Equivalence
BIC	Bit Clear
ORR	Logical (inclusive) OR
MLA	Multiply Accumulate
UMULL	Unsigned Long Multiply
UMLAL	Unsigned Long Multiply Accumulate
MRS	Move From Status Register
BL	Branch and Link
SWI	Software Interrupt
STR	Store Word
STRH	Store Half Word
STRB	Store Byte
STRBT	Store Register Byte with Translation
STRT	Store Register with Translation
STM	Store Multiple
SWPB	Swap Byte
MRC	Move From Coprocessor
STC	Store From Coprocessor

Thumb Instruction Set Overview

The Thumb instruction set is a re-encoded subset of the ARM instruction set.

The Thumb instruction set is divided into:

- Branch instructions
- Data processing instructions
- Load and Store instructions
- Load and Store Multiple instructions
- Exception-generating instruction

In Thumb mode, eight general-purpose registers, R0 to R7, are available that are the same physical registers as R0 to R7 when executing ARM instructions. Some Thumb instructions also access to the Program Counter (ARM Register 15), the Link Register (ARM Register 14)

Table 11. JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
117	PA11/SCK0/TCLK0	IN/OUT	INPUT
116			OUTPUT
115			CONTROL
114	PA12/CTS0/TCLK1	IN/OUT	INPUT
113			OUTPUT
112			CONTROL
111	PA13/RTS0/TCLK2	IN/OUT	INPUT
110			OUTPUT
109			CONTROL
108	PA14/RXD1	IN/OUT	INPUT
107			OUTPUT
106			CONTROL
105	PA15/TXD1	IN/OUT	INPUT
104			OUTPUT
103			CONTROL
102	PA16/RTS1/TIOA0	IN/OUT	INPUT
101			OUTPUT
100			CONTROL
99	PA17/CTS1/TIOB0	IN/OUT	INPUT
98			OUTPUT
97			CONTROL
96	PA18/DTR1/TIOA1	IN/OUT	INPUT
95			OUTPUT
94			CONTROL
93	PA19/DSR1/TIOB1	IN/OUT	INPUT
92			OUTPUT
91			CONTROL
90	PA20/DCD1/TIOA2	IN/OUT	INPUT
89			OUTPUT
88			CONTROL
87	PA21/RI1/TIOB2	IN/OUT	INPUT
86			OUTPUT
85			CONTROL

Using the Service

The following steps show how to initialize and use the ***Xmodem Service*** in an application:

```
Variables definitions:
AT91S_RomBoot const *pAT91; // struct containing Openservice functions
AT91S_SBuffer   sXmBuffer; // Xmodem Buffer allocation
AT91S_SvcXmodem svcXmodem; // Xmodem service structure allocation
AT91S_Pipe      xmodemPipe;// xmodem pipe communication struct
AT91S_CtlTempo  ctlTempo; // Tempo struct
AT91PS_Buffer pXmBuffer; // Pointer on a buffer structure
AT91PS_SvcComm pSvcXmodem; // Pointer on a Media Structure
Initialisations
// Call Open methods:
pAT91 = AT91C_ROM_BOOT_ADDRESS;
// OpenCtlTempo on the system timer
pAT91->OpenCtlTempo(&ctlTempo, (void *) &(pAT91->SYSTIMER_DESC));
ctlTempo.CtlTempoStart((void *) &(pAT91->SYSTIMER_DESC));
// Xmodem buffer initialisation
pXmBuffer     = pAT91->OpenSBuffer(&sXmBuffer);
pSvcXmodem = pAT91->OpenSvcXmodem(&svcXmodem, AT91C_BASE_DBGU, &ctlTempo);
// Open communication pipe on the xmodem service
pAT91->OpenPipe(&xmodemPipe, pSvcXmodem, pXmBuffer);
// Init the DBGU peripheral
// Open PIO for DBGU
  AT91F_DBGU_CfgPIO();
  // Configure DBGU
  AT91F_US_Configure (
    (AT91PS_USART) AT91C_BASE_DBGU,     // DBGU base address
    MCK,             // Master Clock
    AT91C_US_ASYNC_MODE,        // mode Register to be programmed
    BAUDRATE ,              // baudrate to be programmed
    0);                   // timeguard to be programmed
  // Enable Transmitter
  AT91F_US_EnableTx((AT91PS_USART) AT91C_BASE_DBGU);
  // Enable Receiver
  AT91F_US_EnableRx((AT91PS_USART) AT91C_BASE_DBGU);
// Initialize the Interrupt for System Timer and DBGU (shared interrupt)
// Initialize the Interrupt Source 1 for SysTimer and DBGU
  AT91F_AIC_ConfigureIt(AT91C_BASE_AIC,
             AT91C_ID_SYS,
             AT91C_AIC_PRIOR_HIGHEST,
             AT91C_AIC_SRCTYPE_INT_LEVEL_SENSITIVE,
             AT91F_ASM_ST_DBGU_Handler);

  // Enable SysTimer and DBGU interrupt
  AT91F_AIC_EnableIt(AT91C_BASE_AIC, AT91C_ID_SYS);
xmodemPipe.Read(&xmodemPipe, (char *) BASE_LOAD_ADDRESS, MEMORY_SIZE,
XmodemProtocol, (void *) BASE_LOAD_ADDRESS);
```

Abort Status

There are three reasons for an abort to occur:

- access to an undefined address
- access to a protected area without the permitted state
- an access to a misaligned address.

When an abort occurs, a signal is sent back to all the masters, regardless of which one has generated the access. However, only the ARM7TDMI can take an abort signal into account, and only under the condition that it was generating an access. The Peripheral Data Controller does not handle the abort input signal. Note that the connection is not represented in Figure 21.

To facilitate debug or for fault analysis by an operating system, the Memory Controller integrates an Abort Status register set.

The full 32-bit wide abort address is saved in MC_AASR. Parameters of the access are saved in MC_ASR and include:

- the size of the request (field ABTSZ)
- the type of the access, whether it is a data read or write, or a code fetch (field ABTTYP)
- whether the access is due to accessing an undefined address (bit UNDADD), a misaligned address (bit MISADD) or a protection violation (bit MPU)
- the source of the access leading to the last abort (bits MST0 and MST1)
- whether or not an abort occurred for each master since the last read of the register (bit SVMST0 and SVMST1) unless this information is loaded in MST bits

In the case of a Data Abort from the processor, the address of the data access is stored. This is useful, as searching for which address generated the abort would require disassembling the instructions and full knowledge of the processor context.

In the case of a Prefetch Abort, the address may have changed, as the prefetch abort is pipelined in the ARM processor. The ARM processor takes the prefetch abort into account only if the read instruction is executed and it is probable that several aborts have occurred during this time. Thus, in this case, it is preferable to use the content of the Abort Link register of the ARM processor.

Memory Protection Unit

The Memory Protection Unit allows definition of up to 16 memory spaces within the internal memories.

After reset, the Memory Protection Unit is disabled. Enabling it requires writing the Protection Unit Enable Register (MC_PUER) with the PUEB at 1.

Programmming of the 16 memory spaces is done in the registers MC_PUIA0 to MC_PUIA15.

The size of each of the memory spaces is programmable by a power of 2 between 1K bytes and 4M bytes. The base address is also programmable on a number of bits according to the size.

The Memory Protection Unit also allows the protection of the peripherals by programming the Protection Unit Peripheral Register (MC_PUP) with the field PROT at the appropriate value.

The peripheral address space and each internal memory area can be protected against write and non-privileged access of one of the masters. When one of the masters performs a forbidden access, an Abort is generated and the Abort Status traces what has happened.

PDC Receive Next Pointer Register

Register Name: PERIPH_RNPR

Access Type: Read/Write

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

- RXNPTR: Receive Next Pointer Address

RXNPTR is the address of the next buffer to fill with received data when the current buffer is full.

PDC Receive Next Counter Register

Register Name: PERIPH_RNCR

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
RXNCR							
7	6	5	4	3	2	1	0
RXNCR							

- RXNCR: Receive Next Counter Value

·RXNCR is the size of the next buffer to receive.

PDC Transmit Next Pointer Register

Register Name: PERIPH_TNPR

Access Type: Read/Write

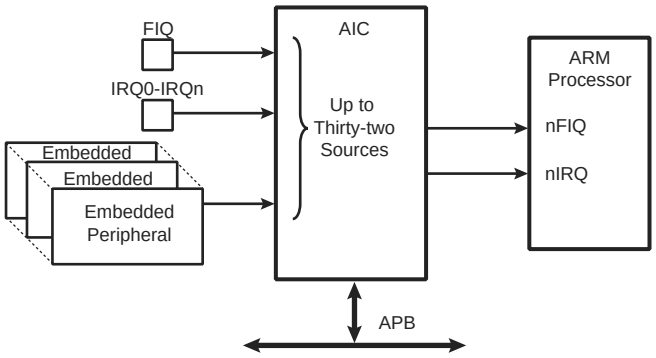
31	30	29	28	27	26	25	24
TXNPTR							
23	22	21	20	19	18	17	16
TXNPTR							
15	14	13	12	11	10	9	8
TXNPTR							
7	6	5	4	3	2	1	0
TXNPTR							

- TXNPTR: Transmit Next Pointer Address

TXNPTR is the address of the next buffer to transmit when the current buffer is empty.

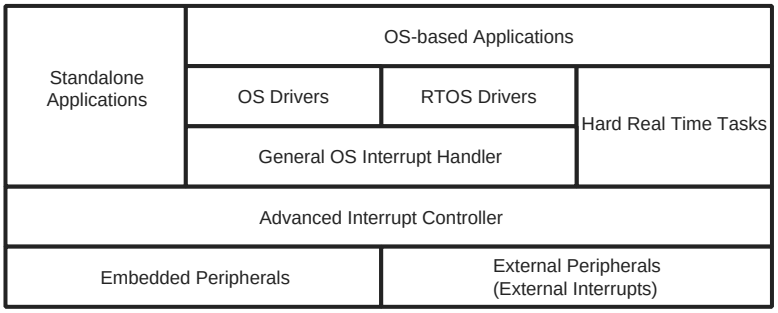
Block Diagram

Figure 25. Block Diagram



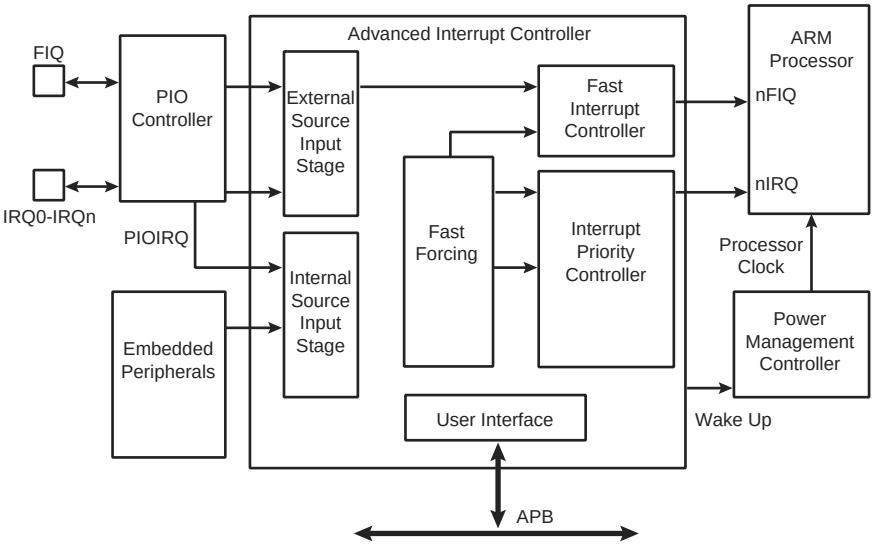
Application Block Diagram

Figure 26. Description of the Application Block



AIC Detailed Block Diagram

Figure 27. AIC Detailed Block Diagram



prevents using the USB ports. Selecting the PLLB Clock saves the power consumption of the PLLA by running the processor and the peripheral at 48 MHz required by the USB ports. Selecting the PLLA Clock runs the processor and the peripherals at their maximum speed while running the USB ports at 48 MHz.

The Master Clock Controller is made up of a clock selector and a prescaler, as shown in Figure 42. It also contains an optional Master Clock divider in products integrating an ARM9 processor. This allows the processor clock to be faster than the Master Clock.

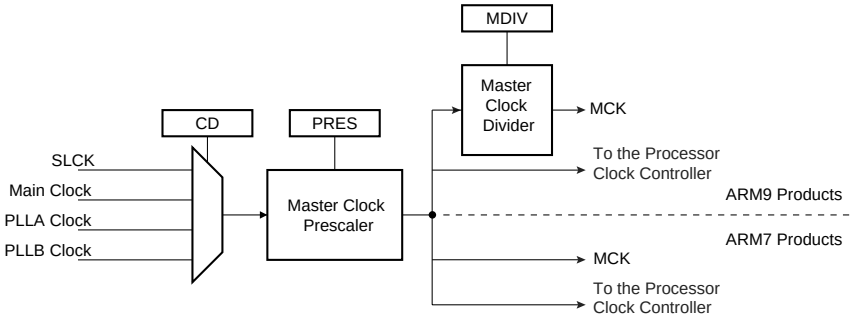
The Master Clock selection is made by writing the CSS field (Clock Source Selection) in PMC_MCKR (Master Clock Register). The prescaler supports the division by a power of 2 of the selected clock between 1 and 64. The PRES field in PMC_MCKR programs the prescaler.

When the Master Clock divider is implemented, it can be programmed between 1 and 4 through the MDIV field in PMC_MCKR.

Each time PMC_MCKR is written to define a new Master Clock, the MCKRDY bit is cleared in PMC_SR. It reads 0 until the Master Clock is established. Then, the MCKRDY bit is set and can trigger an interrupt to the processor. This feature is useful when switching from a high-speed clock to a lower one to inform the software when the change is actually done.

Note: A new value to be written in PMC_MCKR must not be the same as the current value in PMC_MCKR.

Figure 42. Master Clock Controller



Processor Clock Controller

The PMC features a Processor Clock Controller that implements the Idle Mode. The Processor Clock can be enabled and disabled by writing the System Clock Enable (PMC_SCER) and System Clock Disable Registers (PMC_SCDR). The status of this clock (at least for debug purpose) can be read in the System Clock Status Register (PMC_SCSR).

Processor Clock Source

The clock provided to the processor is determined by the Master Clock controller. On ARM7-based systems, the Processor Clock source is directly the Master Clock.

On ARM9-based systems, the Processor Clock source might be 2, 3 or 4 times the Master Clock. This ratio value is determined by programming the field MDIV of the Master Clock Register (PMC_MCKR).

Idle Mode

The Processor Clock is enabled after a reset and is automatically re-enabled by any enabled interrupt. The Idle Mode is achieved by disabling the Processor Clock, which is automatically re-enabled by any enabled fast or normal interrupt, or by the reset of the product.

Debug Unit (DBGU)

Overview

The Debug Unit provides a single entry point from the processor for access to all the debug capabilities of Atmel's ARM-based systems.

The Debug Unit features a two-pin UART that can be used for several debug and trace purposes and offers an ideal medium for in-situ programming solutions and debug monitor communications. Moreover, the association with two peripheral data controller channels permits packet handling for these tasks with processor time reduced to a minimum.

The Debug Unit also makes the Debug Communication Channel (DCC) signals provided by the In-circuit Emulator of the ARM processor visible to the software. These signals indicate the status of the DCC read and write registers and generate an interrupt to the ARM processor, making possible the handling of the DCC under interrupt control.

Chip Identifier registers permit recognition of the device and its revision. These registers inform as to the sizes and types of the on-chip memories, as well as the set of embedded peripherals.

Finally, the Debug Unit features a Force NTRST capability that enables the software to decide whether to prevent access to the system via the In-circuit Emulator. This permits protection of the code, stored in ROM.

Important features of the Debug Unit are:

- System Peripheral to Facilitate Debug of Atmel's ARM-based Systems
- Composed of Four Functions
  - Two-pin UART
  - Debug Communication Channel (DCC) Support
  - Chip ID Registers
  - ICE Access Prevention
- Two-pin UART
  - Implemented Features are 100% Compatible with the Standard Atmel USART
  - 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
  - Interrupt Generation
  - Support for Two PDC Channels with Connection to Receiver and Transmitter
- Debug Communication Channel Support
  - Offers Visibility of COMMRX and COMMTX Signals from the ARM Processor
  - Interrupt Generation
- Chip ID Registers
  - Identification of the Device Revision, Sizes of the Embedded Memories, Set of Peripherals
- ICE Access Prevention
  - Enables Software to Prevent System Access Through the ARM Processor's ICE
  - Prevention is Made by Asserting the NTRST Line of the ARM Processor's ICE

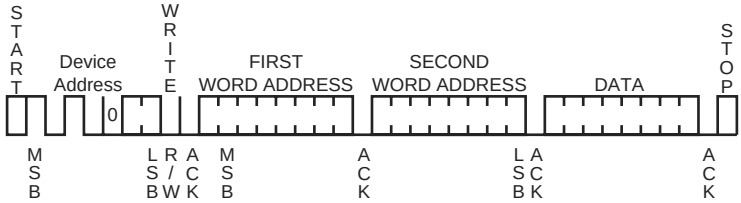
Mode Fault Detection

A mode fault is detected when the SPI is programmed in Master Mode and a low level is driven by an external master on the NPCS0/NSS signal.

When a mode fault is detected, the MODF bit in the SPI_SR is set until the SPI_SR is read and the SPI is disabled until re-enabled by bit SPIEN in the SPI_CR (Control Register).

By default, Mode Fault Detection is enabled. It is disabled by setting the MODFDIS bit in the SPI Mode Register.

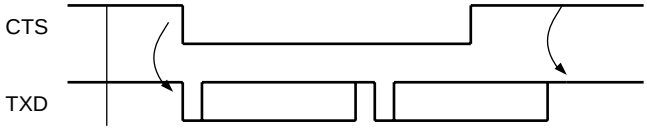
Figure 87. Internal Address Usage



Read/Write Flowcharts

The following flowcharts shown in Figure 88 on page 256 and in Figure 89 on page 257 give examples for read and write operations in Master Mode. A polling or interrupt method can be used to check the status bits. The interrupt method requires that the interrupt enable register (TWI_IER) be configured first.

Figure 107. Transmitter Behavior when Operating with Hardware Handshaking



ISO7816 Mode

The USART features an ISO7816-compatible operating mode. This mode permits interfacing with smart cards and Security Access Modules (SAM) communicating through an ISO7816 link. Both T = 0 and T = 1 protocols defined by the ISO7816 specification are supported.

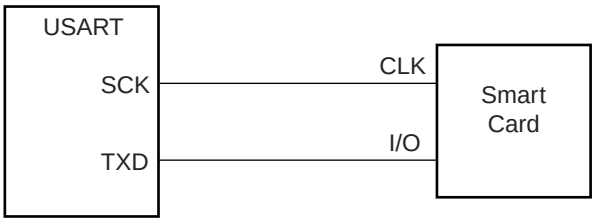
Setting the USART in ISO7816 mode is performed by writing the USART_MODE field in the Mode Register (US_MR) to the value 0x4 for protocol T = 0 and to the value 0x5 for protocol T = 1.

ISO7816 Mode overview

The ISO7816 is a half duplex communication on only one bidirectional line. The baud rate is determined by a division of the clock provided to the remote device (see “Baud Rate Generator” on page 271).

The USART connects to a smart card. as shown in Figure 108. The TXD line becomes bidirectional and the Baud Rate Generator feeds the ISO7816 clock on the SCK pin. As the TXD pin becomes bidirectional, its output remains driven by the output of the transmitter but only when the transmitter is active while its input is directed to the input of the receiver. The USART is considered as the master of the communication as it generates the clock.

Figure 108. Connection of a Smart Card to the USART



When operating in ISO7816, either in T = 0 or T = 1 modes, the character format is fixed. The configuration is 8 data bits, even parity and 1 or 2 stop bits, regardless of the values programmed in the CHRL, MODE9, PAR and CHMODE fields. MSBF can be used to transmit LSB or MSB first.

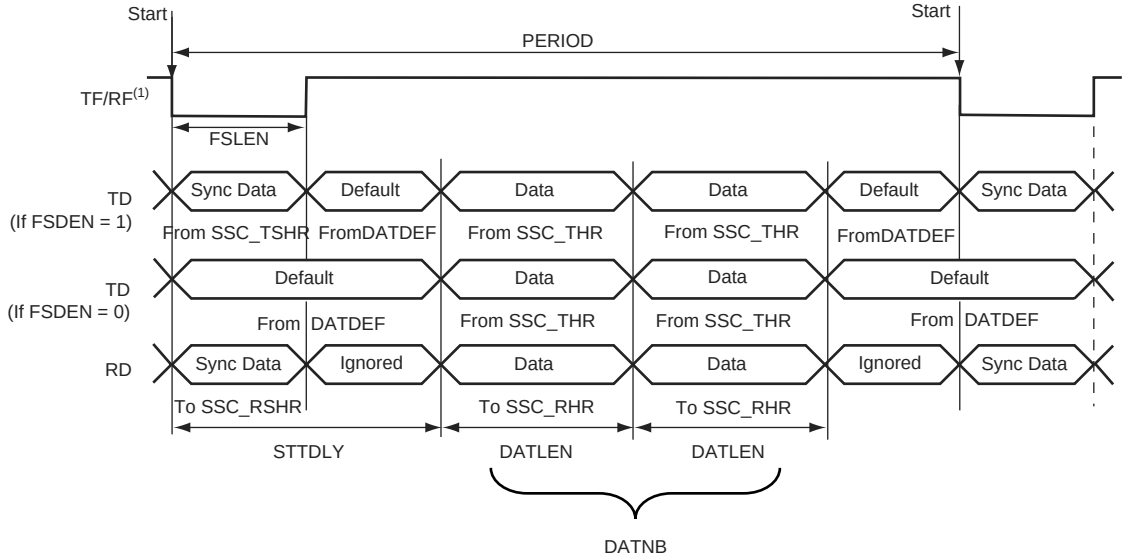
The USART cannot operate concurrently in both receiver and transmitter modes as the communication is unidirectional at a time. It has to be configured according to the required mode by enabling or disabling either the receiver or the transmitter as desired. Enabling both the receiver and the transmitter at the same time in ISO7816 mode may lead to unpredictable results.

The ISO7816 specification defines an inverse transmission format. Data bits of the character must be transmitted on the I/O line at their negative value. The USART does not support this format and the user has to perform an exclusive OR on the data before writing it in the Transmit Holding Register (US_THR) or after reading it in the Receive Holding Register (US_RHR).

Table 57. Data Frame Registers

Transmitter	Receiver	Field	Length	Comment
SSC_TFMR	SSC_RFMR	DATLEN	Up to 32	Size of word
SSC_TFMR	SSC_RFMR	DATNB	Up to 16	Number Word transmitter in frame
SSC_TFMR	SSC_RFMR	MSBF		1 most significant bit in first
SSC_TFMR	SSC_RFMR	FSLEN	Up to 16	Size of Synchro data register
SSC_TFMR		DATDEF	0 or 1	Data default value ended
SSC_TFMR		FSDEN		Enable send SSC_TSHR
SSC_TCMR	SSC_RCMR	PERIOD	up to 512	Frame size
SSC_TCMR	SSC_RCMR	STTDLY	up to 255	Size of transmit start delay

Figure 131. Transmit and Receive Frame Format in Edge/Pulse Start Modes



Note: 1. Input on falling edge on TF/RF example.

SSC Transmit Clock Mode Register

Name: SSC_TCMR

Access Type: Read/Write

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

- CKS: Transmit Clock Selection

CKS	Selected Transmit Clock
0x0	Divided Clock
0x1	RK Clock signal
0x2	TK Pin
0x3	Reserved

- CKO: Transmit Clock Output Mode Selection

CKO	Transmit Clock Output Mode	TK pin
0x0	None	Input-only
0x1	Continuous Transmit Clock	Output
0x2-0x7	Reserved	

- CKI: Transmit Clock Inversion

0: The data and the Frame Sync signal are shifted out on Transmit Clock falling edge.

1: The data and the Frame Sync signal are shifted out on Transmit Clock rising edge.

CKI affects only the Transmit Clock and not the output clock signal.

- START: Transmit Start Selection

START	Transmit Start
0x0	Continuous, as soon as a word is written in the SSC_THR Register (if Transmit is enabled) and immediately after the end of transfer of the previous data.
0x1	Receive Start
0x2	Detection of a low level on TF signal
0x3	Detection of a high level on TF signal
0x4	Detection of a falling edge on TF signal
0x5	Detection of a rising edge on TF signal
0x6	Detection of any level change on TF signal
0x7	Detection of any edge on TF signal
0x8-0xF	Reserved

WAVSEL = 00

When WAVSEL = 00, the value of TC_CV is incremented from 0 to 0xFFFF. Once 0xFFFF has been reached, the value of TC_CV is reset. Incrementation of TC_CV starts again and the cycle continues. See Figure 143.

An external event trigger or a software trigger can reset the value of TC_CV. It is important to note that the trigger may occur at any time. See Figure 144.

RC Compare cannot be programmed to generate a trigger in this configuration. At the same time, RC Compare can stop the counter clock (CPCSTOP = 1 in TC_CMR) and/or disable the counter clock (CPCDIS = 1 in TC_CMR).

Figure 143. WAVSEL= 00 without trigger

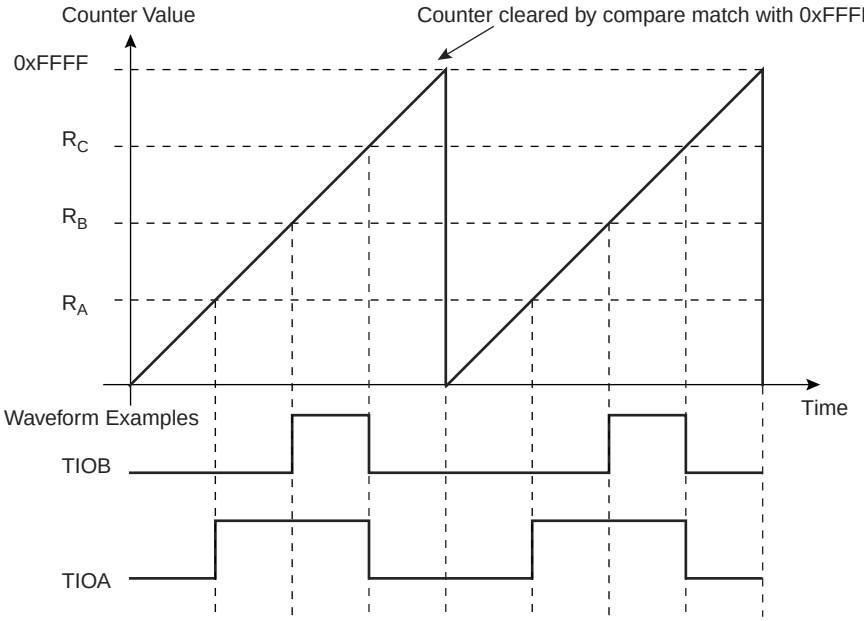
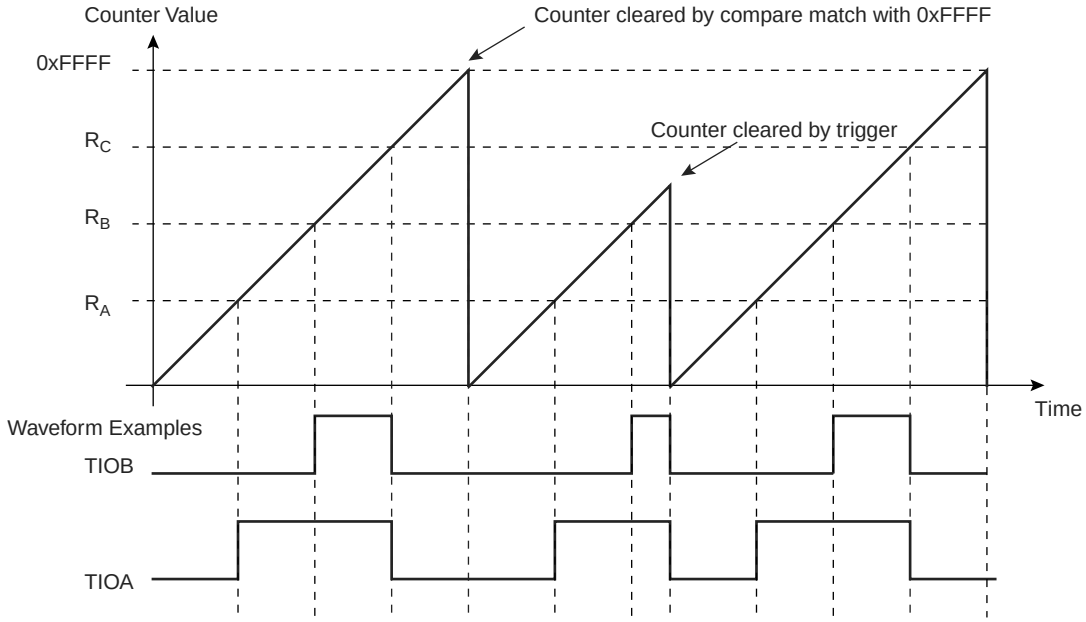


Figure 144. WAVSEL= 00 with trigger



Timer Counter (TC) User Interface

Table 61. Timer Counter Global Memory Map

Offset	Channel/Register	Name	Access	Reset Value
0x00	TC Channel 0	See Table 62 See Table 62 See Table 62		
0x40	TC Channel 1			
0x80	TC Channel 2			
0xC0	TC Block Control Register	TC_BCR	Write-only	–
0xC4	TC Block Mode Register	TC_BMR	Read/Write	0

TC_BCR (Block Control Register) and TC_BMR (Block Mode Register) control the whole TC block. TC channels are controlled by the registers listed in Table 62. The offset of each of the channel registers in Table 62 is in relation to the offset of the corresponding channel as mentioned in Table 62.

Table 62. Timer Counter Channel Memory Map

Offset	Register	Name	Access	Reset Value
0x00	Channel Control Register	TC_CCR	Write-only	–
0x04	Channel Mode Register	TC_CMR	Read/Write	0
0x08	Reserved			–
0x0C	Reserved			–
0x10	Counter Value	TC_CV	Read-only	0
0x14	Register A	TC_RA	Read/Write(1)	0
0x18	Register B	TC_RB	Read/Write(1)	0
0x1C	Register C	TC_RC	Read/Write	0
0x20	Status Register	TC_SR	Read-only	0
0x24	Interrupt Enable Register	TC_IER	Write-only	–
0x28	Interrupt Disable Register	TC_IDR	Write-only	–
0x2C	Interrupt Mask Register	TC_IMR	Read-only	0

Notes: 1. Read only if WAVE = 0

0 = RA Load has not occurred since the last read of the Status Register or WAVE = 1.

1 = RA Load has occurred since the last read of the Status Register, if WAVE = 0.

- LDRBS: RB Loading Status

0 = RB Load has not occurred since the last read of the Status Register or WAVE = 1.

1 = RB Load has occurred since the last read of the Status Register, if WAVE = 0.

- ETRGS: External Trigger Status

0 = External trigger has not occurred since the last read of the Status Register.

1 = External trigger has occurred since the last read of the Status Register.

- CLKSTA: Clock Enabling Status

0 = Clock is disabled.

1 = Clock is enabled.

- MTIOA: TIOA Mirror

0 = TIOA is low. If WAVE = 0, this means that TIOA pin is low. If WAVE = 1, this means that TIOA is driven low.

1 = TIOA is high. If WAVE = 0, this means that TIOA pin is high. If WAVE = 1, this means that TIOA is driven high.

- MTIOB: TIOB Mirror

0 = TIOB is low. If WAVE = 0, this means that TIOB pin is low. If WAVE = 1, this means that TIOB is driven low.

1 = TIOB is high. If WAVE = 0, this means that TIOB pin is high. If WAVE = 1, this means that TIOB is driven high.

TC Interrupt Enable Register

Register Name: TC_IER

Access Type: Write-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
ETRGS	LDRBS	LDRAS	CPCS	CPBS	CPAS	LOVRS	COVFS

- COVFS: Counter Overflow

0 = No effect.

1 = Enables the Counter Overflow Interrupt.

- LOVRS: Load Overrun

0 = No effect.

1 = Enables the Load Overrun Interrupt.

- CPAS: RA Compare

0 = No effect.

1 = Enables the RA Compare Interrupt.

- CPBS: RB Compare

0 = No effect.

1 = Enables the RB Compare Interrupt.

- CPCS: RC Compare

0 = No effect.

1 = Enables the RC Compare Interrupt.

- LDRAS: RA Loading

0 = No effect.

1 = Enables the RA Load Interrupt.

- LDRBS: RB Loading

0 = No effect.

1 = Enables the RB Load Interrupt.

- ETRGS: External Trigger

0 = No effect.

1 = Enables the External Trigger Interrupt.

MCI SD Receive Data Register

Name: MCI_RDR

Access Type: Read-only

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

- DATA: Data to Read

MCI SD Transmit Data Register

Name: MCI_TDR

Access Type: Write-only

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

- DATA: Data to Write