Features

- **ARM7TDMI® ARM® Thumb® Processor Core**
  - **High Performance 32-bit RISC**
  - **High-density 16-bit Instruction set (Thumb)**
  - **Leader in MIPS/Watt**
  - **Embedded ICE (In Circuit Emulation)**
- **16 Kbytes Internal SRAM**
- **Fully Programmable External Bus Interface (EBI)**
  - **Maximum External Address Space of 6 Mbytes, Up to Four Chip Select Lines**
- **8-level Priority, Vectored Interrupt Controller**
  - **Three External Interrupts Including One Fast Interrupt Line**
- **Ten-channel Peripheral Data Controller (PDC)**
- **57 Programmable I/O Lines**
- **Four 16-bit General Purpose Timers (GPT)**
  - **Three Configurable Modes: Counter, PWM, Capture**
  - **Four External Clock Inputs, Three Multi-purpose I/O Pins per Timer**
- **Four 16-bit Simple Timers (ST)**
- **Four Channel 16-bit Pulse Width Modulation (PWM)**
- **Four CAN Controllers 2.0A and 2.0B Full CAN**
  - **One with 32 Buffers, Three with 16 Buffers**
- **Two USARTs**
  - **Support for J1587 and LIN Protocols**
- **One Master/Slave SPI Interface**
  - **8 to 16-bit Programmable Data Length**
  - **Four External Serial Peripheral Chip Selects**
- **Two 8-channel 10-bit Analog to Digital Converters (ADC)**
- **Two 16-bit Capture Modules (CAPT)**
- **Programmable Watch Timer (WT)**
- **Programmable Watchdog (WD)**
- **Power Management Controller (PMC)**
  - **32 kHz Oscillator, Main Oscillator and PLL**
- **IEEE 1149.1 JTAG Boundary-scan on all Digital Pins**
- **Fully Static Operation: 0 Hz to 30 MHz at VDDCORE = 3.3V, 85°C**
- **3.0V to 5.5V Operating Voltage Range**
- **3.0V to 3.6V Core, Memory and Analog Voltage Range**
- **-40° to +85°C Operating Temperature Range**
- **Available in a 176-lead LQFP Package**

Description

The AT91SAM7A2 is based on the ARM7TDMI embedded processor. This processor has a high-performance 32-bit RISC architecture with a high-density 16-bit instruction set and very low power consumption.

In addition, a large number of internally banked registers result in very fast exception handling, making the device ideal for real-time control applications.

The AT91SAM7A2 has a direct connection to off-chip memory, including Flash, through the fully programmable External Bus Interface.

An 8-level priority vectored Interrupt Controller in conjunction with the Peripheral Data Controller significantly improves the real time performance of the device. The device is manufactured using high-density CMOS technology.

By combining the ARM7TDMI processor with an on-chip SRAM, and a wide range of peripheral functions, including USART, SPI, CAN Controllers, Timer Counter and Analog-to-Digital Converters, on a monolithic chip, the AT91SAM7A2 is a powerful device that provides a flexible, cost-effective solution to many compute-intensive embedded control applications in the automotive and industrial world.

AT91 ARM® Thumb®- based Microcontrollers

AT91SAM7A2

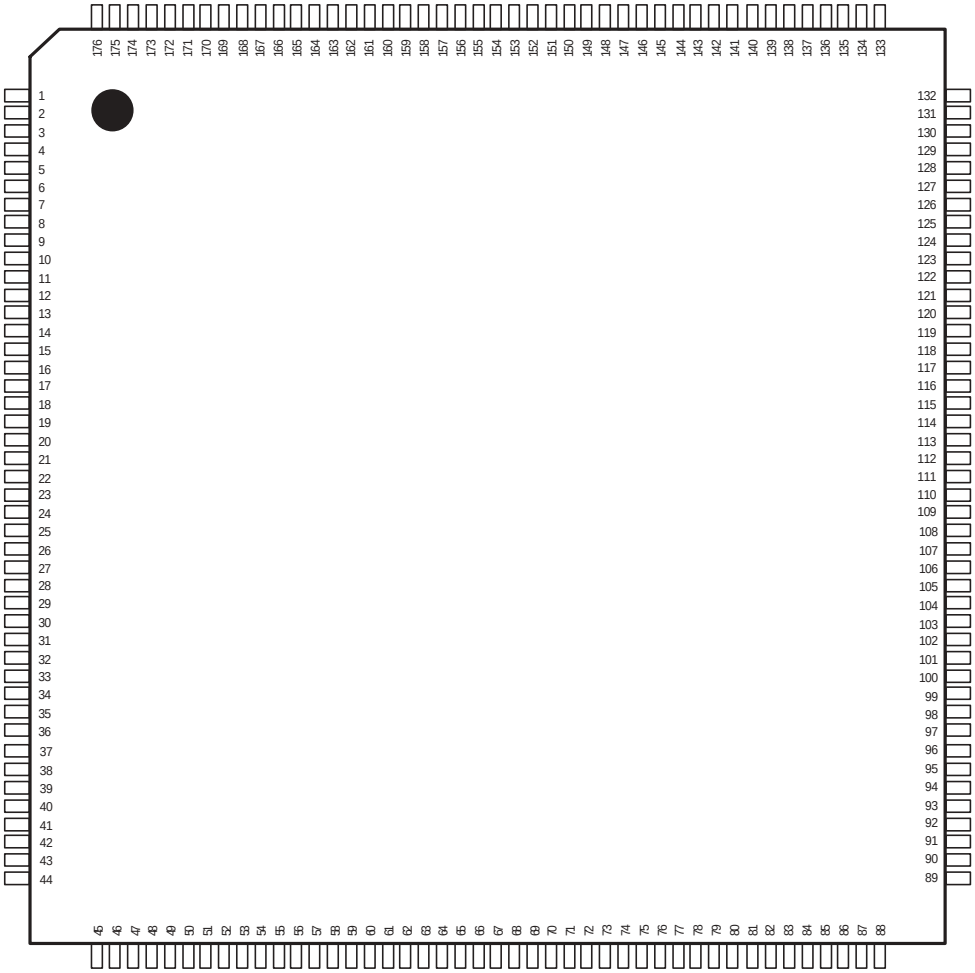
6021A–ATARM–07/04

Pin Configuration

Table 1. Pinout

Pin	Name	Pin	Name	Pin	Name	Pin	Name
1	VDDIO	45	GND	89	VDDIO	133	NOE/NRD
2	IRQ0	46	VDDIO	90	VDDANA	134	NCS0
3	IRQ1	47	UPIO5	91	VREFP0	135	ADD1
4	FIQ	48	UPIO6	92	ANA0IN0	136	D9
5	SCK0/MPIO	49	GND	93	ANA0IN1	137	D2
6	TXD0/MPIO	50	VDDIO	94	ANA0IN2	138	VDDCORE
7	RXD0/MPIO	51	UPIO7	95	ANA0IN3	139	D10
8	SCK1/MPIO	52	UPIO 8	96	ANA0IN4	140	D3
9	TXD1/MPIO	53	UPIO 9	97	ANA0IN5	141	D11
10	RXD1/MPIO	54	UPIO 10	98	ANA0IN6	142	D4
11	VDDCORE	55	UPIO 11	99	GND	143	D12
12	CANTX3	56	UPIO 12	100	VDDANA	144	D5
13	CANRX3	57	UPIO 13	101	ANA0IN7	145	D13
14	CAPT0	58	UPIO 14	102	VREFP1	146	D6
15	CAPT1	59	UPIO 15	103	ANA1IN0	147	D14
16	SPCK/MPIO	60	UPIO 16	104	ANA1IN1	148	D7
17	MISO/MPIO	61	UPIO 17	105	ANA1IN2	149	D15
18	MOSI/MPIO	62	UPIO 18	106	ANA1IN3	150	GND
19	NPCS0/MPIO	63	GND	107	ANA1IN4	151	ADD0/NLB
20	VDDIO	64	VDDIO	108	ANA1IN5	152	ADD17
21	GND	65	UPIO19	109	ANA1IN6	153	ADD16
22	NPCS1/MPIO	66	UPIO20	110	ANA1IN7	154	ADD15
23	NPCS2/MPIO	67	UPIO21	111	GND	155	ADD14
24	NPCS3/MPIO	68	UPIO22	112	VDDCORE	156	ADD13
25	T0TIOA0/MPIO	69	UPIO23	113	RTCKI	157	ADD12
26	T0TIOB0/MPIO	70	UPIO24	114	RTCKO	158	ADD11
27	T0TCLK0/MPIO	71	UPIO25	115	GND	159	ADD10
28	T0TIOA1/MPIO	72	UPIO26	116	VDDCORE	160	ADD9
29	T0TIOB1/MPIO	73	UPIO27	117	SCANEN	161	ADD20/CS3
30	T0TCLK1/MPIO	74	UPIO28	118	TEST	162	VDDCORE
31	T0TIOA2/MPIO	75	UPIO29	119	TMS	163	NWR0/NWE
32	T0TIOB2/MPIO	76	UPIO30/NWAIT	120	TDO	164	NCS2
33	VDDIO	77	UPIO31/CORECLK	121	TDI	165	NCS1
34	GND	78	CANTX0	122	TCK	166	ADD19
35	T0TCLK2/MPIO	79	CANRX0	123	GND	167	ADD18
36	T1TIOA0/MPIO	80	CANTX1	124	PLLRC	168	ADD8
37	T1TIOB0/MPIO	81	CANRX1	125	VDDCORE	169	ADD7
38	T1TCLK0/MPIO	82	CANTX2	126	MCKI	170	ADD6
39	NRESET	83	CANRX2	127	MCKO	171	ADD2
40	UPIO0	84	PWM0	128	GND	172	ADD3
41	UPIO1	85	PWM1	129	NWR1/NUB	173	ADD4
42	UPIO2	86	PWM2	130	D8	174	ADD5
43	UPIO3	87	PWM3	131	D1	175	GND
44	UPIO4	88	GND	132	D0	176	GND

Figure 1. Pin Configuration



Signal Description

Table 2. Signal Description

Module	Name	Function	Type	Active Level	Comments
EBI	ADD[19:1]	External address bus	O	(Z)[1]	The EBI is tri-stated when NRESET is at a logical low level. Internal pull-downs on data bus bits
	ADD0/NLB	External address line line/ Lower byte enable	O	L (Z)	
	ADD20/CS3	External address line/ Chip select	O	H (Z)	
	D[15:0]	External data bus	I/O	(Z)	
	NOE	Output enable	O	L (Z)	
	NWR0/NWE	Write enable	O	L (Z)	
	NCS[2:0]	Chip select lines	O	L (Z)	
	NWR1/NUB	Upper byte enable	O	L (Z)	
	NWAIT	External Wait	I	L	Disable at reset, multiplexed with UPIO30
	CORECLK	Core CLock	O		Disable at reset, multiplexed with UPIO31
GIC	IRQ[1:0]	External interrupt lines	I		
	FIQ	Fast interrupt line	I		
Power-on Reset	NRESET	Hardware reset input	I	L	Schmitt input with internal filter
Master Clock	MCKI	Master clock input	I		Connected to external crystal (4 to 6 Mhz)
	MCKO	Master clock output	O		
	PLLRC	PLL RC network input	I		
32.768 kHz clock	RTCKI	32.768 KHz clock input	I		Connected to external 32.768 Khz crystal
	RTCKO	32.768 KHz clock output	O		
PIO	UPIO[31:0]	General purpose I/O	I/O	(Z)	
USART0	SCK0/MPIO	USART0 clock line	I/O	(Z)	Multiplexed with general purpose I/O
	RXD0/MPIO	USART0 receive line	I/O	(Z)	Multiplexed with general purpose I/O
	TXD0/MPIO	USART0 transmit line	I/O	(Z)	Multiplexed with general purpose I/O
USART1	SCK1/MPIO	USART1 clock line	I/O	(Z)	Multiplexed with general purpose I/O
	RXD1/MPIO	USART1 receive line	I/O	(Z)	Multiplexed with general purpose I/O
	TXD1/MPIO	USART1 transmit line	I/O	(Z)	Multiplexed with general purpose I/O
Capture0	CAPT0	Capture input	I		
Capture1	CAPT1	Capture input	I		
PWM	PWM[3:0]	Pulse Width Modulation output	O	(L)	
Timer T0	T0TIOA[2:0]/MPIO	Capture/waveform I/O	I/O	(Z)	Multiplexed with a general purpose I/O
	T0TIOB[2:0]/MPIO	Trigger/waveform I/O	I/O	(Z)	Multiplexed with a general purpose I/O
	T0TIOCLK[2:0]/MPIO	External clock/trigger/input	I/O	(Z)	Multiplexed with a general purpose I/O

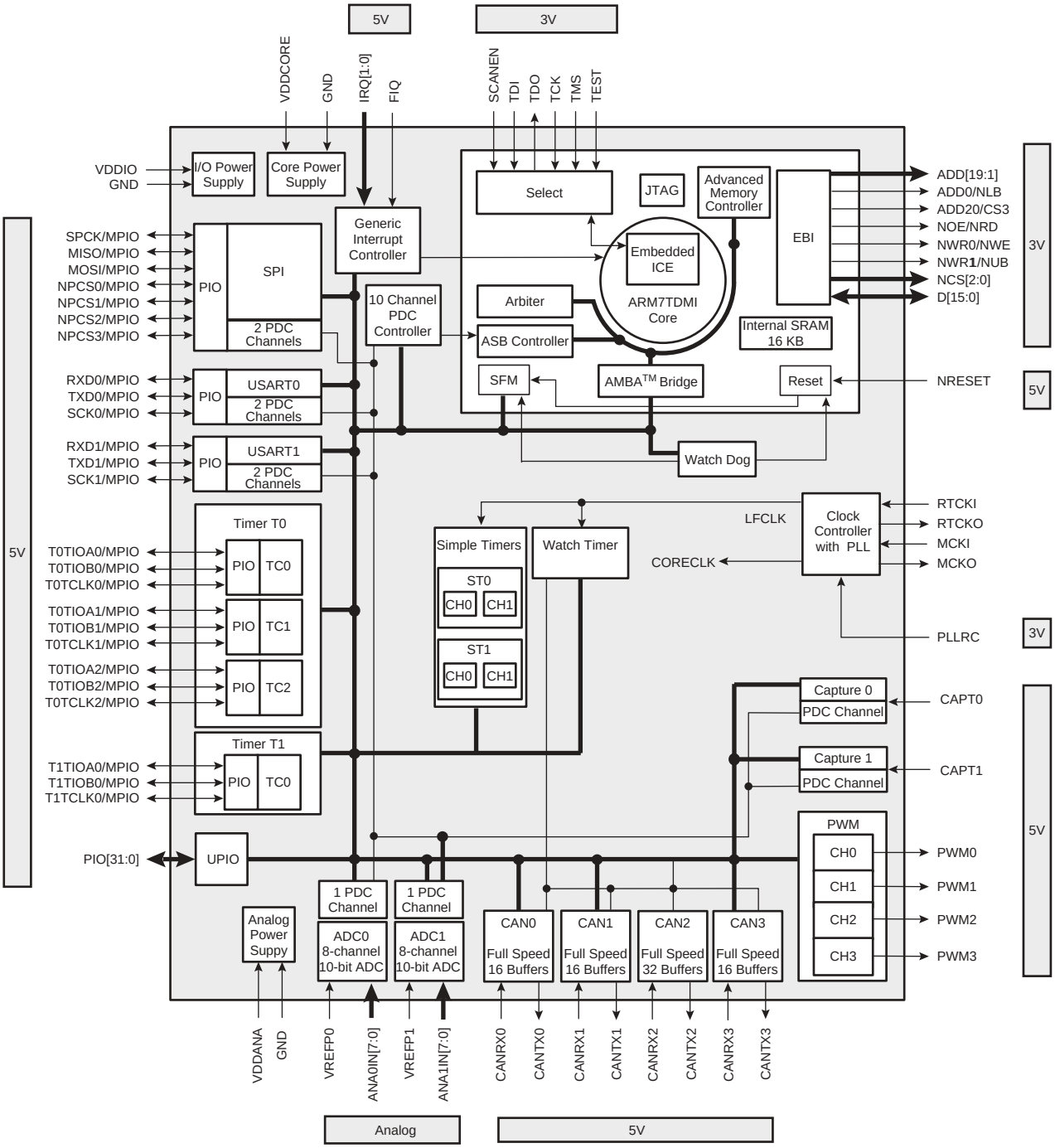
Table 2. Signal Description (Continued)

Module	Name	Function	Type	Active Level	Comments
Timer T1	T1TIOA/MPIO	Capture/waveform I/O	I/O	(Z)	Multiplexed with a general purpose I/O
	T1TIOB/MPIO	Trigger/waveform I/O	I/O	(Z)	Multiplexed with a general purpose I/O
	T0TIOCLK/MPIO	External clock/trigger/input	I/O	(Z)	Multiplexed with a general purpose I/O
ADC0	ANA0IN[7:0]	Analog input	I		
	VREFP0	Positive voltage reference	I		
ADC1	ANA1IN[7:0]	Analog input	I		
	VREFP1	Positive voltage reference	I		
SPI	SPCK/MPIO	SPI clock line	I/O	(Z)	Multiplexed with a general purpose I/O
	MISO/MPIO	SPI master in slave out	I/O	(Z)	Multiplexed with a general purpose I/O
	MOSI/MPIO	SPI master out slave in	I/O	(Z)	Multiplexed with a general purpose I/O
	NPCS[3:1]/MPIO	SPI chip select	I/O	(Z)	Multiplexed with a general purpose I/O
	NPCS0/NSS/MPIO	SPI chip select (slave input)	I/O	(Z)	Multiplexed with a general purpose I/O
CAN0	CANRX0	CAN0 receive line	I	L	
	CANTX0	CAN0 transmit line	O	L (H)	
CAN1	CANRX1	CAN1 receive line	I	L	
	CANTX1	CAN1 transmit line	O	L (H)	
CAN2	CANRX2	CAN2 receive line	I	L	
	CANTX2	CAN2 transmit line	O	L (H)	
CAN3	CANRX3	CAN3 receive line	I	L	
	CANTX3	CAN3 transmit line	O	L (H)	
JTAG	SCANEN	Scan enable	I	H	Internal pull-down (connected GND or leave unconnected)
	TDI	Test Data In	I		Schmitt trigger, internal pull-up
	TDO	Test Data Out	O		
	TMS	Test Mode Select	I		Schmitt trigger, internal pull-up
	TCK	Test Clock	I		Schmitt trigger, internal pull-up
	TEST	Factory Test	I	H	Internal pull-down (connected GND or leave unconnected)
Core Power Supply	VDDCORE	3.3V	-		
	GND	Ground	-		
Analog Power Supply	VDDANA	3.3V	-		
	GND	Analog Ground	-		
I/O power supply	VDDIO	3.3V to 5V	-		
	GND	Ground	-		

Note: 1. Values in brackets are the values at reset (H = High, L = Low, Z = High Impedance State).

Block Diagram

Figure 2. Block Diagram



Product Overview

Register Considerations

Enable/Disable/Status Registers

In order to reduce code size and subsequently increase speed when accessing internal peripherals, most of the registers have been split into three address locations:

- The first address location (Enable or Set Register) is used to set a bit to a logical 1.
- The second address location (Disable or Clear Register) is used to set a bit to a logical 0.
- The third address location (Status register or Mask Register) gives the current state of the bit.

To set a bit to a logical 1 in the Status or Mask Register, a write command in the Enable or Set Register must be performed with the corresponding bit at a logical 1.

To set a bit to a logical 0 in the Status or Mask Register, a write command in the Disable or Clear Register must be performed with the corresponding bit at a logical 1.

Example

Supposing that the US0_PSR register value is 0x00000000. To enable the RXD and SCK pins as PIOs in the USART0 block, 0x00050000 must be written in the US0_PER register. The value read in the US0_PSR register will be 0x00050000.

Now if the software wants to disable the RXD pin as a PIO (i.e. enable it for USART0 use), a write access to the US0_PDR register with the value 0x00040000 must be performed. The new value read in the US0_PSR register will be 0x00010000.

Key Access to Registers

Some bits in registers can be set to a value (0 or 1) only if the right key is written at the same time.

Example

The TESTEN bit in the SFM_TM register can be set to a logical 0 or 1 only if the KEY[15:0] bits are equal to 0xD64A.

To enable test mode, 0xD64A0002 must be written in the SFM_TM register.

To disable test mode, 0xD64A0000 must be written in the SFM_TM register.

Ghost Registers

The AT91SAM7A2 microcontroller integrates an ICE (In-Circuit Emulation) interface that is associated with a JTAG connection and a software debugger that provides powerful debug possibility.

Effectively,

- A running program can be stopped.
- Internal registers and internal/external memories can be monitored.
- Instructions can be added when the core is stopped.
- The program can be resumed.

However, some AT91SAM7A2 registers are "read-active", meaning that reading such registers can affect the state of other registers. This is usual and wanted register behavior. For example, in the ADC module, the bit EOC (End Of Conversion) is automatically cleared when the DR (numerical value of the input converted) is read. The aim is to cut off the amount of code needed in an application.

Meanwhile, when debugging software, users can monitor the value of a register, without modifying the state of another register. For this purpose, for each module, a ghost regis-

ter field has been implemented in the design. Users reading in this ghost field will not affect the value of any other register.

Ghost registers are not "read-active", and are mirrors of original registers. They are located in memory by inversing the 13th bit in the module base address. For example, base address of ADC module is 0xFFFC0000, so the ghost register's base address of ADC is 0xFFFC2000. By reading this ghost field, users do not disturb the behavior of the ADC module.

Ghost registers exist for all modules.

Read Active Registers The following table demonstrates the effects of the read active registers.

Table 3. Read-active Registers

Module	Read-active registers	Effect
GIC	GIC_IVR	Clears IRQ interrupt if present at the GIC.
	GIC_FVR	Clears FIQ interrupt if present at the GIC.
ADC	ADC_DR	Clears EOC bit in ADC_SR register if set.
USART	US_SR	CLears the following bits in the US_SR register if set. IDLE ENDRX ENDTX SCK TXD RXD
	US_RHR	Clears RXRDY bit in the US_SR register if set.
CAPT	CAPT_DR	Clears DATACAPT bit in the CAPT_SR register if set.
SPI	SPI_SR	Clears the following bits in the SPI_SR register if set. MODF SPIOVRE REND TEND SPCK MISO MOSI NPCS0 NPCS1 NPCS2 NPCS3
	SPI_RDR	Clears RDRF bit in the SPI_SR register if set.
PIO	UPIO_SR	Clears all bits in the UPIO_SR register if set.
GPT	GPT_SR	When in capture mode, it clears the following bits in the GPT_SR register if set. COVFS LOVRS CPCS LDRAS LDRBS ETRGS TIOBS TIOAS TCLKS
	GPT_SR	When in waveform mode, it clears the following bits in the GPT_SR register if set. COVFS CPAS CPBS CPCS ETRGS TIOBS TIOAS TCLKS

Power Consumption

The power consumption is described in the specific modes of the AT91SAM7A2.

Working Modes

The AT91SAM7A2 microcontroller provides different working modes as outlined inTable 4 below.

Table 4. Working Modes

Mode		Note
Low Power Mode	LPM	The master clock oscillator. The PLL and the internal divider are switched off. The real time oscillator is enabled. The low frequency clock is selected from the real time oscillator and used as system clock (i.e. 32.768 kHz used for GIC, WD, WT, ST and any peripheral needed for interrupt generation). CORECLK = RTCK, LFCLK = RTCK.
Slow Mode	SLM	The PLL is switched off. The system clock is the master clock (CORECLK = MCK) or the master clock divided by β (CORECLK = MCK/β, β is in the range of [2:256]).
Operational	OPE	Master oscillator and PLL are enabled. The system clock is the clock from the PLL, CORECLK = α x MCK (α is in the range of [x2:x20]).

Low Power Mode

Low power mode is defined as the state in which:

- Master clock oscillator and PLL are halted.
- Low frequency oscillator (32.768 kHz) used as internal system clock for core and all the peripherals (CORECLK = RTCK, LFCLK = RTCK)

$V_{VDDCORE}$ = 3.3 V, V_{VDDIO} = 5 V. No loads on outputs, ground level on all inputs, 25°C, fetch out of internal RAM in ARM mode.

Table 5. Low Power Mode Consumption.

Mode	Parameters	Typical	Max	Unit
LPM	All peripheral clocks disable, ARM clock enable		240	µA
LPM	All peripheral clocks disable, ARM clock disable		240	µA

Slow Power Mode

Slow mode is defined as the state in which:

- Master clock oscillator is enabled, divided by β (β is in the range of [2:256]) and used as the system clock (CORECLK = MCK or MCK/β).
- The low frequency clock can still be used as low frequency clock for peripherals (LFCLK = RTCK or MCK/β).

$V_{VDDCORE}$ = 3.3 V, V_{VDDIO} = 5 V. No loads on outputs, ground level on all inputs, 25°C, oscillator 4 MHz, β = 256.

Table 6. Slow Mode Consumption.

Mode	Parameters	Typical	Max	Unit
SLM	All peripheral clocks disable, ARM clock enable		1140	µA
SLM	All peripheral clocks disable, ARM clock disable		1140	µA

Operational Mode

Operational mode is defined as the state in which:

- Master clock oscillator and PLL are enabled, system clock is taken from the PLL output (CORECLK = α x MCK, where α is in the range of [2:20]).
- The low frequency clock can still be used as low frequency clock for peripherals (LFCLK = RTCK or MCK/β, β is in the range of [2:256]).

The total power dissipation of the AT91SAM7A2 embedded system, when in operational mode, is estimated to be 200 mW[1] maximum, at an operating voltage of 3.3 V, over the operating temperature range.

Table 7. Operational Mode Consumption.

Mode	Parameters	Typ	Unit
OPE	Core	900	µA/MHz
SLM	PLL (frequency independent)	1.5	mA

Module Consumption

Initial condition: 3.3 V, 25°C, All inputs grounded, low level and no load on all outputs, ARM clock enable.

Table 8. Operational Mode Consumption.

Symbol	Parameters	Typ	Unit
PDC	Peripheral Data controller	160	µA/MHz
UPIO	Unified Parallel Input Output	40	µA/MHz
USART	Universal Sync/Async Receiver Transceiver	110	µA/MHz
SPI	Serial Peripheral Interface	60	µA/MHz
GPT3CH	General Purpose Timer 3 Channels	150	µA/MHz
GPT1CH	General Purpose Timer 1 Channel	40	µA/MHz
ADC	Analog to Digital Converter	20	µA/MHz
CAN16	CAN 16 Channels	210	µA/MHz
CAN32	CAN 32 Channels	280	µA/MHz
ST	Simple Timer	40	µA/MHz
CAPT	Capture	20	µA/MHz
PWM4C	PWM 4 Channels	60	µA/MHz
ALL	All Modules	1650	µA/MHz

Reset

To properly reset the chip, users must maintain a reset of at least 1µs.

After a reset, the program starts executing after the PLL stabilization time (11.3 ms for an oscillator of 4 MHz).

Note: 1. ARM core and modules working at CORECLK frequency = 30 MHz (i.e. MCK = 6 MHz, PLL multiplier = 5).

Electrical Characteristics

Table 9. Pin Connection

Pin	Name	Pad	Pin	Name	Pad	Pin	Name	Pad	Pin	Name	Pad
1	VDDIO		45	GND		89	VDDIO		133	NOE/NRD	PC3T03
2	IRQ0	MC5D00	46	VDDIO		90	VDDANA		134	NCS0	PC3T01
3	IRQ1	MC5D00	47	UPIO5	MC5B03	91	VREFP0	ANAIN	135	ADD1	PC3T01
4	FIQ	MC5D00	48	UPIO6	MC5B03	92	ANA0IN0	AIMUX1	136	D9	PC3B01D
5	SCK0/MPIO	MC5B01	49	GND		93	ANA0IN1	AIMUX1	137	D2	PC3B01D
6	TXD0/MPIO	MC5B01	50	VDDIO		94	ANA0IN2	AIMUX1	138	VDDCORE	
7	RXD0/MPIO	MC5B01	51	UPIO7	MC5B03	95	ANA0IN3	AIMUX1	139	D10	PC3B01D
8	SCK1/MPIO	MC5B01	52	UPIO8	MC5B02	96	ANA0IN4	AIMUX1	140	D3	PC3B01D
9	TXD1/MPIO	MC5B01	53	UPIO9	MC5B02	97	ANA0IN5	AIMUX1	141	D11	PC3B01D
10	RXD1/MPIO	MC5B01	54	UPIO10	MC5B02	98	ANA0IN6	AIMUX1	142	D4	PC3B01D
11	VDDCORE		55	UPIO11	MC5B02	99	GND		143	D12	PC3B01D
12	CANTX3	MC5O01	56	UPIO12	MC5B02	100	VDDANA		144	D5	PC3B01D
13	CANRX3	MC5D00	57	UPIO13	MC5B02	101	ANA01N7	AIMUX1	145	D13	PC3B01D
14	CAPT0	MC5D00	58	UPIO14	MC5B02	102	VREFP1	ANAIN	146	D6	PC3B01D
15	CAPT1	MC5D00	59	UPIO15	MC5B02	103	ANA1IN0	AIMUX1	147	D14	PC3B01D
16	SPCK/MPIO	MC5B01	60	UPIO16	MC5B01	104	ANA1IN1	AIMUX1	148	D7	PC3B01D
17	MISO/MPIO	MC5B01	61	UPIO17	MC5B01	105	ANA1IN2	AIMUX1	149	D15	PC3B01D
18	MOSI/MPIO	MC5B01	62	UPIO18	MC5B01	106	ANA1IN3	AIMUX1	150	GND	
19	NPCS0/MPIO	MC5B01	63	GND		107	ANA1IN4	AIMUX1	151	ADD0/NLB	PC3T01
20	VDDIO		64	VDDIO		108	ANA1IN5	AIMUX1	152	ADD17	PC3T01
21	GND		65	UPIO19	MC5B01	109	ANA1IN6	AIMUX1	153	ADD16	PC3T01
22	NPCS1/MPIO	MC5B01	66	UPIO20	MC5B01	110	ANA1IN7	AIMUX1	154	ADD15	PC3T01
23	NPCS2/MPIO	MC5B01	67	UPIO21	MC5B01	111	GND		155	ADD14	PC3T01
24	NPCS3/MPIO	MC5B01	68	UPIO22	MC5B01	112	VDDCORE		156	ADD13	PC3T01
25	T0TIOA0/MPIO	MC5B01	69	UPIO23	MC5B01	113	RTCKI	32.768 kHz crystal oscillator pad	157	ADD12	PC3T01
26	T0TIOB0/MPIO	MC5B01	70	UPIO24	MC5B01	114	RTCKO	32.768 kHz crystal oscillator pad	158	ADD11	PC3T01
27	T0TCLK0/MPIO	MC5B01	71	UPIO25	MC5B01	115	GND		159	ADD10	PC3T01
28	T0TIOA1/MPIO	MC5B01	72	UPIO26	MC5B01	116	VDDCORE		160	ADD9	PC3T01
29	T0TIOB1/MPIO	MC5B01	73	UPIO27	MC5B01	117	SCANEN	PC3D01D	161	ADD20/CS3	PC3T01
30	T0TCLK1/MPIO	MC5B01	74	UPIO28	MC5B01	118	TEST	PC3D01D	162	VDDCORE	
31	T0TIOA2/MPIO	MC5B01	75	UPIO29	MC5B01	119	TMS	PC3D21U	163	NWR0/NWE	PC3B01
32	T0TIOB2/MPIO	MC5B01	76	UPIO30/NWAIT	MC5B01	120	TDO	PC3T03	164	NCS2	PC3T01
33	VDDIO		77	UPIO31/CORE CLK	MC5B01	121	TDI	PC3D21U	165	NCS1	PC3T01
34	GND		78	CANTX0	MC5O01	122	TCK	PC3D21U	166	ADD19	PC3T01
35	T0TCLK2/MPIO	MC5B01	79	CANRX0	MC5D00	123	GND		167	ADD18	PC3T01
36	T1TIOA0/MPIO	MC5B01	80	CANTX1	MC5O01	124	PLLRC	PLL080M1	168	ADD8	PC3T01
37	T1TIOB0/MPIO	MC5B01	81	CANRX1	MC5D00	125	VDDCORE		169	ADD7	PC3T01
38	T1TCLK0/MPIO	MC5B01	82	CANTX2	MC5O01	126	MCKI	OSC16M	170	ADD6	PC3T01
39	NRESET	MC5D20	83	CANRX2	MC5D00	127	MCKO	OSC16M	171	ADD2	PC3T01
40	UPIO0	MC5B04	84	PWM0	MC5O01	128	GND		172	ADD3	PC3T01
41	UPIO1	MC5B04	85	PWM1	MC5O01	129	NWR1/NUB	PC3B01	173	ADD4	PC3T01
42	UPIO2	MC5B04	86	PWM2	MC5O01	130	D8	PC3B01D	174	ADD5	PC3T01
43	UPIO3	MC5B04	87	PWM3	MC5O01	131	D1	PC3B01D	175	GND	
44	UPIO4	MC5B03	88	GND		132	D0	PC3B01D	176	GND	

Pad types are given in Table 10 below.

Table 10. Pad Types

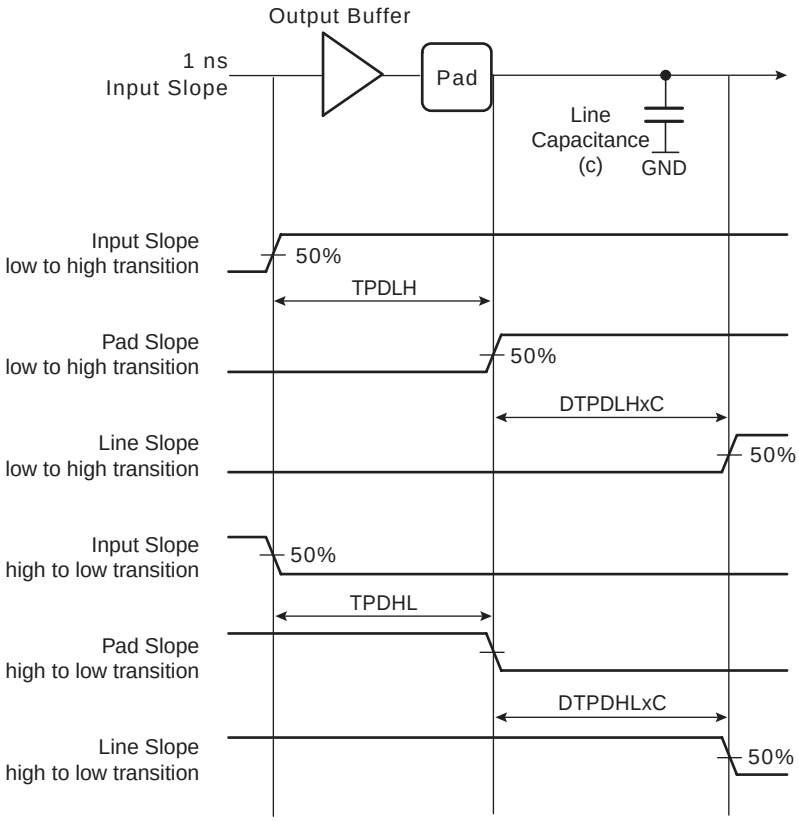
Pad	Type	DTPDHL(1)	DTPDLH(2)	TPDHL(3)	TPDLH(4)	Output Current
MC5B01	5 V CMOS bidirectional pad	0.144 ns/pF	0.131 ns/pF	2.327 ns	2.192 ns	2 mA AC 2 mA DC
MC5B02	5 V CMOS bidirectional pad	0.072 ns/pF	0.066 ns/pF	2.298 ns	2.179 ns	4 mA AC 4 mA DC
MC5B03	5 V CMOS bidirectional pad	0.036 ns/pF	0.033 ns/pF	2.727 ns	2.034 ns	8 mA AC 8 mA DC
MC5B04	5 V CMOS bidirectional pad	0.018 ns/pF	0.017 ns/pF	3.265 ns	2.449 ns	16 mA AC 16 mA DC
MC5O01	5 V CMOS output pad	0.144 ns/pF	0.131 ns/pF	2.310 ns	2.174 ns	2 mA AC 2 mA DC
MC5D00	5 V CMOS non-inverting input pad					
MC5D20	5 V CMOS schmitt non-inverting input pad					
PC3D01D	3 V CMOS non-inverting input pad with pull-down resistor					
PC3D01U	3 V CMOS non-inverting input pad with pull-up resistor					
PC3D21	3 V CMOS schmitt non-inverting input pad					
PC3D21U	3V CMOS schmitt non-inverting input pad with pull-up resistor					
PC3T01	3 V CMOS three state output pad	0.120 ns/pF	0.116 ns/pF	1.357 ns	1.011 ns	2 mA AC 0.3 mA DC
PC3T02	3 V CMOS three state output pad	0.060 ns/pF	0.058 ns/pF	1.002 ns	0.781 ns	4 mA AC 0.3 mA DC
PC3T03	3 V CMOS three state output pad	0.040 ns/pF	0.039 ns/pF	0.943 ns	0.800 ns	6 mA AC 0.3 mA DC
PC3B01D	3 V CMOS bidirectional pad with pull-down resistor	0.118 ns/pF	0.116 ns/pF	1.357 ns	1.040 ns	2 mA AC 0.3 mA DC
PC3B01	3 V CMOS non-inverting bidirectional pad	0.120 ns/pF	0.116 ns/pF	1.372 ns	1.033 ns	2 mA AC 0.3 mA DC
PC3B02	3 V CMOS non-inverting bidirectional pad	0.060 ns/pF	0.058 ns/pF	1.010 ns	0.789 ns	6 mA AC 0.3 mA DC
PC3B03	3 V CMOS non-inverting bidirectional pad	0.040 ns/pF	0.039 ns/pF	0.948 ns	0.808 ns	6 mA AC 0.3 mA DC
OSCK33	32.768 kHz crystal oscillator pad					
OSC16M	2-6 MHz crystal oscillator pad					
PLL080M1	20 MHz to 80 MHz single pad Phase-Locked Loop					
AIMUX1	Analog input pad					

Notes:
1. Differential (load-dependent) propagation delay, high-to-low or high impedance-to-low (V_{DD} = 3.3 V, Temp. = 25°C, Input Slope = 1 ns)
2. Differential (load-dependent) propagation delay, low-to-high or high impedance-to-high (V_{DD} = 3.3 V, Temp. = 25°C, Input Slope = 1 ns)
3. Propagation delay, high-to-low (V_{DD} = 3.3 V, Temp. = 25°C, Input Slope = 1 ns)
4. Propagation delay, low-to-high (V_{DD} = 3.3 V, Temp. = 25°C, Input Slope = 1 ns)

Propagation Time

The propagation delay time shown in Table 10, “Pad Types,” on page 13, is the time in nanoseconds from the 50% point of the input to the 50% point of the output.

Figure 3. Propagation Delay



Clocks

Overview

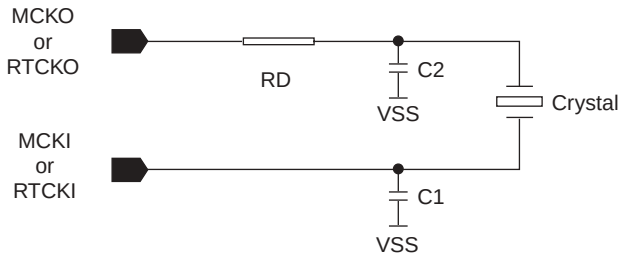
The AT91SAM7A2 microcontroller provides:

- 32.768 kHz oscillator
- 4 MHz to 6 MHz oscillator

Crystals

Crystals with 10 pF load capacitance can be directly connected to the oscillator pads. Nevertheless, it is recommended to implement the circuitry as described hereafter and in Figure 4 below.

Figure 4. Circuitry for 10 pF Load Capacitance



If the crystal recommended capacitor Cx is greater than 10 pF, then C1 and C2 must be added. Cx can be approximated to: Cx = (C1 x C2)/(C1 + C2).

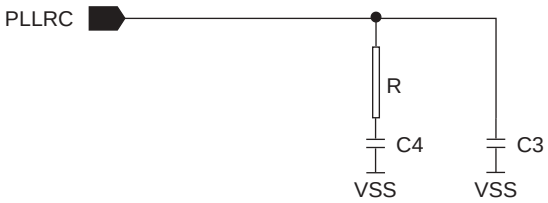
Table 11. Typical Crystal Series Resistor

Signal	RD	Conditions
MCKO	0 Ω	Crystal: CP12A-4MHz-S1-4085-1050 (NDK®)
RTCKO	10 kΩ	Crystal: MC-306 32.768K-A (EPSON®)

Phase Locked Loop

The AT91SAM7A2 microcontroller integrates a programmable PLL with a default ratio value of x5. The PLL requires an external RC network as described hereafter and in Figure 5 below.

Figure 5. External RC Network



The optimum response with a simple RC filter is obtained when:

Equation1: $$0.4 < \sqrt{\left(\frac{K_0 \times I_P}{n \times (C_3 + C_4)}\right)} \times \frac{R \times C_4}{2} < 1$$ with an optimum value of 0.707

Where:

- K0 is the PLL V_{CO} gain (typ 105.106 Hz/V, min 65.106 Hz/V, max 172.106 Hz/V)

- IP is the peak current delivered by the charge pump into the filter (typ 350 mA, min 50 mA, max 800 mA)
- n is the division ration of the divider (i.e. PLL multiplication factor)
- Stability can be improved with an additional capacitor C3. The value of C3 must be chosen so that:

Equation 2: $4 < \frac{C_4}{C_3} < 15$

Equation 3: $\sqrt{\left(\frac{K_0 \times I_p}{n \times (C_3 + C_4)}\right)} \leq \frac{\text{II} \times f_{CKR}}{5}$

Where:

- f_{CKR} is the PLL input frequency (i.e. MCK):

Phase jitter for the PLL is 200 ps Typically.

PLL Characteristics

Table 12. PLL Characteristics

Code	Parameter	Conditions	Min	Typ	Max	Unit
f_{CKR}	Input frequency		0.02		30	MHz
f_{CK}	Output frequency		20		30	MHz
Wlow	Duty cycle			50		%
j_{CK}	Jitter	With ratio 1:1		200		ps
n	Division ratio		1:1		1:1024	
K_0	V_{CO} gain		65	105	172	MHz/V
I_P	CHP current		50	350	800	mA

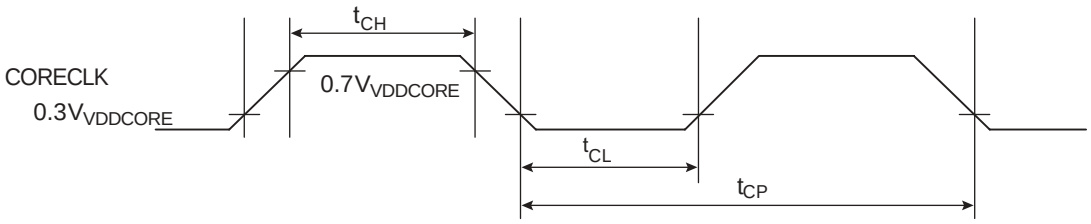
Clock Timings

Core Clock

Table 13. Core Clock Timings

Symbol	Parameter	Minimum	Maximum	Unit
$1/t_{CP}$	Oscillator frequency	32.768	30000	kHz
t_{CP}	Main clock period	33		ns
t_{CH}	Main clock high time	12		ns
t_{CL}	Main clock low time	12		ns

Figure 6. Core Clock Waveform

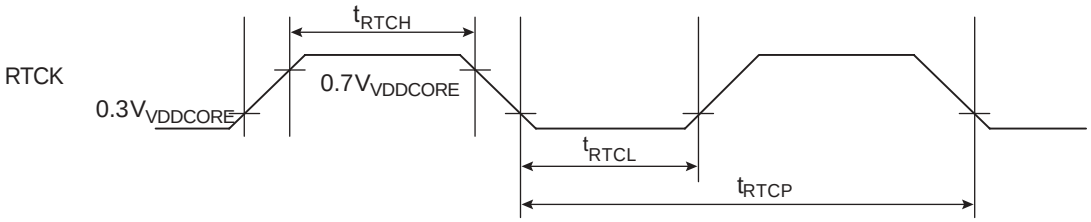


32.768 kHz Frequency Clock The 32.768 kHz clock is the clock generated by the real time clock oscillator. The real time clock (RTCK) characteristics are given below in Table 14.

Table 14. Low Frequency Clock Timings

Symbol	Parameter	Minimum	Typical	Maximum	Unit
$1/t_{RTCP}$	32.768kHz oscillator frequency		32.768		kHz

Figure 7. 32.768 kHz Clock Waveform



Internal Oscillator Characteristics

Core Clock Oscillator

Table 15. Core Clock Oscillator

Code	Parameter	Conditions	Min	Typ	Max	Unit
Du	Duty cycle	Crystal @ 4 MHz	40	50	60	%
Opf	Operating frequency		4		8	MHz
t_{SU}	Startup time	Crystal @ 4 MHz			10	ms
t_{SU}	Startup time	Crystal @ 8 MHz			5	ms
C1	Internal capacitance (MCKI/GND)			10		pF
C2	Internal capacitance (MCKO/GND)			10		pF
CL	Equivalent load capacitance (MCKI/MCKO)			5		pF
DL	Drive level				50	W
Rs	Equivalent Series Resistance	Fundamental @ 8 Mhz			100	
Rs	Equivalent Series Resistance	Fundamental @ 4 Mhz			50	
Cs	Shunt capacitance	Crystal			6	pF
CL	Load capacitance	Crystal @ 4 MHz		10		pF
Cm	Motional capacitance	Crystal @ 4 MHz			3	fF

Real Time Clock Oscillator

Table 16. Real Time Clock Oscillator

Code	Parameter	Conditions	Min	Typ	Max	Unit
Du	Duty cycle	@ 32.768 kHz	40	50	60	%
tsu	Startup time				1.5	s
C1	Internal capacitance (RTCKI/GND)			20		pF
C2	Internal capacitance (RTCKO/GND)			20		pF
CL	Equivalent load capacitance (RTCKI/RTCKO)			10		pF
DL	Drive level				1	W
Rs	Series resistance	Crystal			60	k
Cs	Shunt capacitance	Crystal	0.8		1.7	pF
	Load capacitance	Crystal @ 32.768 kHz		10		pF
Cm	Motional capacitance	Crystal @ 32.768 kHz	1		4	fF

Memory Map

When the AT91SAM7A2 microcontroller is reset, the ARM core is in reboot mode to access the external memory (usually ROM) on NCS0 at address 0x00000000. The internal RAM is located at address 0x00300000.

When the software executes the remap command (write 1 in RCB field in AMC_RCR register), the internal RAM is automatically located at address 0x00000000 and the external memory accessed on the NCS0 is located in the memory space from 0x40000000 to 0x7FFFFFFF, depending on the AMC_NCS0 register in the Advanced Memory Controller, then the chip is in remap mode.

Reboot Mode

The memory map in reboot mode is described below.

Table 17. Memory Map in Reboot Mode

Memory Space	Application	Abort
0xFFFFFFFF – 0xFFE00000	Peripheral devices	No
0xFFDFFFFF – 0x00400000	Reserved	Yes
0x003FFFFF – 0x00300000	Internal RAM 16 kbytes repeated 64 times	No
0x002FFFFF – 0x00200000	Reserved (Read as '0')	No
0x001FFFFF – 0x00100000	Reserved	Yes
0x000FFFFF – 0x00000000	External memory on NCS0	No

Remap Mode

The memory map in remap mode is described below.

Table 18. Memory Map in Remap Mode

Memory Space	Application	Abort
0xFFFFFFFF – 0xFFE00000	Peripheral devices	No
0xFFDFFFFF – 0x80000000	Reserved	Yes
0x7FFFFFFF – 0x40000000	External memories (up to 4) Memory values repeated within the page size programmed	Yes, outside of page defined in the AMC
0x3FFFFFFF – 0x00300000	Reserved	Yes
0x002FFFFF – 0x00100000	Reserved (Read as '0')	No
0x000FFFFF – 0x00000000	Internal RAM 16 kbytes repeated 64 times	No

External Memory

The AT91SAM7A2 external memories can be relocated in the address space from 0x40000000 to 0x7FFFFFFF. The configuration of the base address and the page size of each EBI chip select line (NCS[2:0], CS3) is done through the Advanced Memory Controller (AMC) registers. It is to be noted that the two most significant bits of the base address are fixed to 01b, allocating these memories between 0x40000000 to 0x7FFFFFFF.

The maximum external memory space is 6 Mbytes, while CS3/A20 is used as address line A20.

Table 19. External Memory Map

Memory Space	Size	Application
0x(01XXb)XXFFFFF – 0x(01XXb)XX00000	Up to 1 Mbytes	External memory on CS3
0x(01XXb)X1FFFFF – 0x(01XXb)XX00000	Up to 2 Mbytes	External memory on NCS2
0x(01XXb)X1FFFFF – 0x(01XXb)XX00000	Up to 2 Mbytes	External memory on NCS1
0x(01XXb)X1FFFFF – 0x(01XXb)XX00000	Up to 2 Mbytes	External memory on NCS0

Peripheral Memory

The peripheral memory map is described below.

Table 20. Peripheral Memory Map

Peripheral	Address	IRQ
AMC	0xFFE00000	-
SFM	0xFFF00000	-
Watchdog	0xFFFA0000	2
Watch Timer	0xFFFA4000	3
USART0	0xFFFA8000	4
USART1	0xFFFAC000	5
CAN3 (16 channels)	0xFFFB0000	6
SPI	0xFFFB4000	7
CAN1 (16 channels)	0xFFFB8000	8
CAN2 (32 channels)	0xFFFBC000	9
ADC0 (8 channels 10-bit)	0xFFFC0000	10
ADC1 (8 channels 10-bit)	0xFFFC4000	11
GPT0 (3 channels)	0xFFFC8000	12
		13
		14
GPT1 (1 channel)	0xFFFCC000	18
PWM	0xFFFD0000	19
CAN0 (16 channels)	0xFFFD4000	20
UPIO	0xFFFD8000	21
Capture CAPT0	0xFFFDC000	22
Capture CAPT1	0xFFFE0000	23
Simple Timer ST0	0xFFFE4000	24
Simple Timer ST1	0xFFFE8000	25
Clock Manager	0xFFFEC000	-
PMC	0xFFFF4000	-
PDC	0xFFFF8000	-
GIC	0xFFFFF000	-

Power Management Block

In order to reduce power consumption, the AT91SAM7A2 microcontroller provides a power management block in some peripherals used to switch on/off the peripheral clocks (peripheral and PIO block).

This function is independent of the Power Management Controller (peripheral) used to switch on/off the ARM7TDMI core and the PDC clocks.

Three registers are provided:

- PERIPHERAL_ECR (at peripheral offset 0x0050) enables the clock.
- PERIPHERAL_DCR (at peripheral offset 0x0054) disables the clock.
- PERIPHERAL_PMSR (at peripheral offset 0x0058) gives the status of the clock.

Two bits are provided in these registers:

- Bit 0 controls the PIO block of the peripheral.
- Bit 1 controls the peripheral function.

When the peripheral clock (and/or the PIO clock) is disabled, the clock is immediately stopped. When the clock is re-enabled, the peripheral controller (and/or the PIO controller) resumes action where it left off.

The interrupt registers are common to the peripheral controller and its PIO controller. The clock on the interrupt registers and its associated logic are stopped only if both the peripheral controller clock and the PIO controller clock are stopped.

Table 21. Power Management Blocks

Module	Power Management Block Present
AMC	No
SFM	No
Watchdog	No
Watch Timer	No
USART0	Yes
USART1	Yes
CAN3	Yes
SPI	Yes
ADC0	Yes
ADC1	Yes
GPT0 Ch0	Yes
GPT0 Ch1	Yes
GPT0 Ch2	Yes
GPT1 Ch0	Yes
PWM	Yes
CAN0	Yes
CAN1	Yes
CAN2	Yes
UPIO	Yes
CAPT0	Yes
CAPT1	Yes
Simple Timer ST0	Yes
Simple Timer ST1	Yes
CM	No
PMC	Yes
PDC	No
GIC	No

PIO Controller Block

To match different applications, the AT91SAM7A2 peripherals have their dedicated pins multiplexed with general purpose I/O pins (MPIO).

The following table lists the module sharing the dedicated pins with MPIOs.

Table 22. PIO Blocks

Module	PIO Block Present	Number of MPIOs	Name of PIO Lines
AMC	No	-	-
SFM	No	-	-
Watchdog	No	-	-
Watch Timer	No	-	-
USART0	Yes	3	TXD0, RXD0, SCK0
USART1	Yes	3	TXD1, RXD1, SCK1
CAN3	No	-	-
SPI	Yes	7	MISO, MOSI, SPCK, NPCS[3:0]
ADC0	No	-	-
ADC1	No	-	-
GPT0 TC0	Yes	3	TIOA0, TIOB0, TCLK0
GPT0 TC1	Yes	3	TIOA1, TIOB1, TCLK1
GPT0 TC2	Yes	3	TIOA2, TIOB2, TCLK2
PWM	No	-	-
CAN0	No	-	-
CAN1	No	-	-
CAN2	No	-	-
UPIO	Yes	32	UPIO[31:0]
CAPT0	No	-	-
CAPT1	No	-	-
Simple Timer ST0	No	-	-
Simple Timer ST1	No	-	-
CM	No	-	-
PMC	No	-	-
PDC	No	-	-
GIC	No	-	-
GPT1 TC0	Yes	3	TIOA, TIOB, TCLK

Each PIO block in the peripheral is controlled through the peripheral interface. The PIO block clock is enabled/disabled by the peripheral power management controller. See Table 20, “Peripheral Memory Map,” on page 21.

Multiplexed I/O Lines

All I/O lines are multiplexed with an I/O signal of the peripheral. After reset, the pin is controlled by the peripheral PIO controller. When a peripheral signal is not used in an application, the corresponding pin can be used as a parallel I/O.

Each parallel I/O line is bidirectional, whether, the peripheral defines the signal as input or output.

Figure 8 “Parallel I/O Block,” on page 25 shows the multiplexing of the peripheral signals with the PIO controller signal.

Each pin of the peripheral can be independently controlled using the Peripheral_PER (PIO Enable) and Peripheral_PDR (PIO Disable) registers.

The Peripheral_PSR (PIO Status) register indicates whether the pin is controlled by the peripheral or by the PIO controller block.

Output Selection

The user can select the direction of each individual I/O signal (input or output) using the Peripheral_OER (Output Enable) and Peripheral_ODR (Output Disable) registers. The output status of the I/O signal can be read in the Peripheral_OSR (Output Status) register. The direction defined has effect only if the pin is configured to be controlled by the PIO controller block.

I/O Levels

Each pin can be configured to be independently driven high or low. The level is defined in different ways, according to the following conditions.

If a pin is controlled by the PIO controller block and is defined as an output (see “Output Selection” above), the level is programmed using the Peripheral_SODR (Set Output Data) and Peripheral_CODR (Clear Output Data) registers. In this case, the programmed value can be read in the Peripheral_ODSR (Output Data Status) register.

If a pin is controlled by the PIO controller block and is not defined as an output, the level is determined by the external circuit. If a pin is not controlled by the PIO controller block, the state of the pin is defined by the Peripheral controller. In all cases, the level on the pin can be read in the Peripheral_PDSR (Pin Data Status) register.

Interrupts

Each PIO controller block also provides an internal interrupt signal shared with the peripheral interrupt.

Each PIO can be programmed to generate an interrupt when a level change occurs. This is controlled by the Peripheral_IER (Interrupt Enable) and Peripheral_IDR (Interrupt Disable) registers which enable/disable the I/O interrupt (and the peripheral interrupts) by setting/clearing the corresponding bit in the Peripheral_IMR.

When a change in level occurs, the corresponding bit in the Peripheral_SR (Interrupt Status) register is set whether the pin is used as a PIO or a peripheral signal and whether it is defined as input or output.

If the corresponding interrupt in Peripheral_IMR (Interrupt Mask) register is enabled, the PIO interrupt is asserted.

The PIO interrupt is cleared when:

- a write access is performed on the Peripheral_CISR register (with the corresponding bit set at a logical 1),

or

- a read access is performed in the Peripheral_SR register (if no Peripheral_CISR register is present in the peripheral).

User Interface

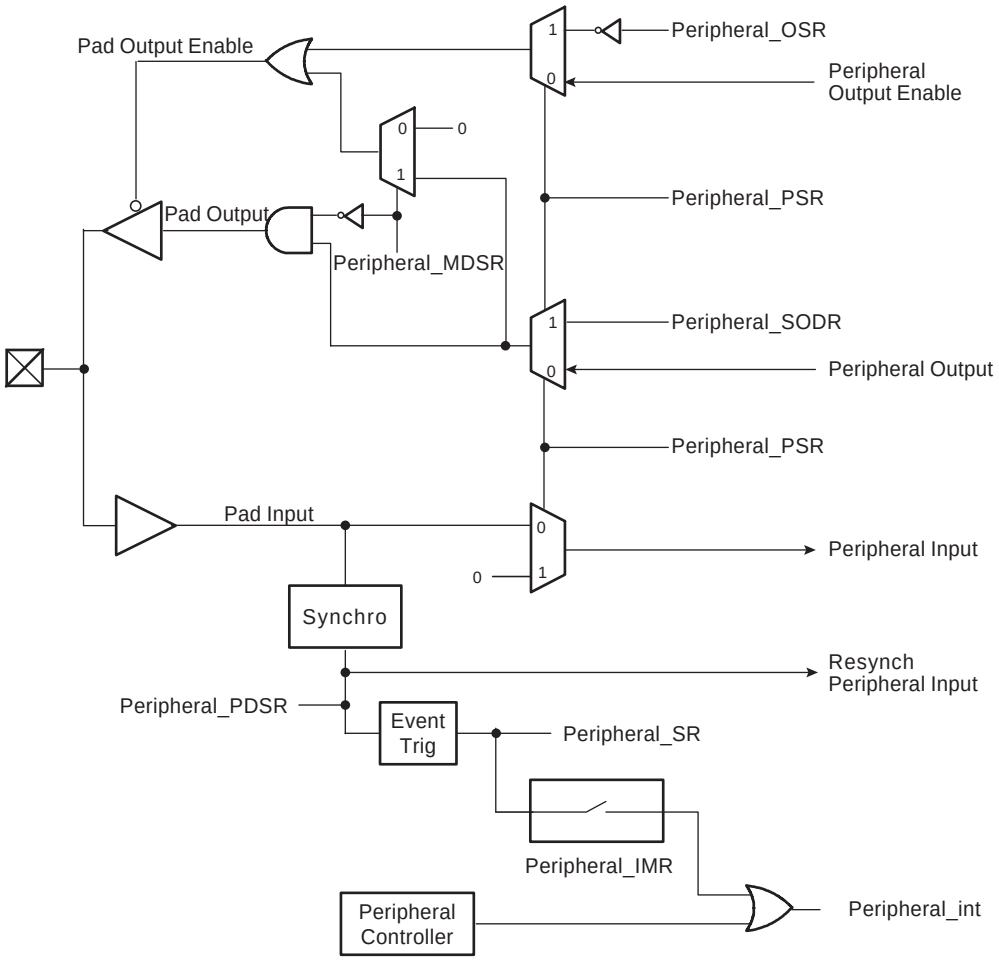
Each individual MPIO is associated with a bit position in the PIO controller user interface registers. Each of these registers is 32 bits wide. If a parallel I/O line is not defined, writing to the corresponding bit has no effect. Undefined bits read zero.

Multi-driver (Open Drain)

Each PIO can be programmed for a multi-driver option. This means that the PIO is configured as open drain (can only drive a low level) in order to support external drivers on the same pin. An external pull-up is necessary to guarantee a logic level (logical one) when the pin is not being driven. The Peripheral_MDER (Multi-Driver Enable) and Peripheral_MDDR (Multi-Driver Disable) registers control this option. The Multi-driver can be selected whether the I/O pin is controlled by the PIO controller or the peripheral controller. Peripheral_MDSR (Multi-driver Status) indicates which pins are configured to support external drivers.

MPIO Block Diagram

Figure 8. Parallel I/O Block



Advanced Memory Controller (AMC)

Overview

The AT91SAM7A2 microcontroller is provided with an Advanced Memory Controller (AMC) enabling the software to configure external and internal memory mapping (at boot level).

The external 16-bit data bus interface is called the External Bus Interface (EBI) and is the physical layer used to connect external devices to the AT91SAM7A2 microcontroller. Subsequently, the EBI generates the signals which control the access to the external memory or peripheral devices.

The EBI is fully programmable through the Advanced Memory Controller and can address up to 6 Mbytes. It has four chip selects and a 20-bit address bus.

The AT91SAM7A2 can only boot on a 16-bit external memory device connected to the NCS0 signal. All the other chip select lines can be configured to access 8 or 16-bit memory devices.

Boot on NCS0

Automatically, the AT91SAM7A2 boots on a 16-bit external memory device connected on NCS0.

At reset, access through NCS0 is configured as follows (in the AMC_CSR0 register).

- 8 wait states (WSE = 1, NWS = 7)
- 16-bit data bus width
- Base address is at 0x00000000
- Byte access type is configured as Byte Write Access, BAT = 0
- The number of data float time is 0.
- The EBI is configured in normal read protocol (DRP = 0 in AMC_MCR register).

The user can modify the chip select 0 configuration, programming the AMC_CSR0 with exact boot memory characteristics. The base address becomes effective after the remap command (set to a logical 1 the RCB in AMC_RCR), but the other parameters are changed immediately after the write access in the AMC_CSR0 register.

External Memory Mapping

The memory map associates the internal 32-bit address space with the external 20-bit address bus.

The memory map is defined by programming the base address and page size of the external memories.

If the physical memory device is smaller than the programmed page size, it wraps around and appears to be repeated within the page. The AMC correctly handles any valid access to the memory device within the page.

In the event of an access request to an address outside any programmed page, a data abort signal is generated. Two types of abort are possible: instruction prefetch abort and data abort. The corresponding exception vector addresses are respectively 0x0000000C and 0x00000010.

It is up to the system programmer to program the error handling routine to use in case of an abort (see the ARM7TDMI datasheet for further information).

The AT91SAM7A2 microcontroller must be wired so that the NCS0 accesses a non volatile 16-bit memory as shown in Figure 9 “EBI Connection for External 16-bit Memory Device, 16-bit Access Only,” on page 27 or Figure 10 “EBI Connection for 2x8-bit External Memory Devices, 16-bit Access Only,” on page 28.

External Memory Device Connection

Each chip select can operate with one of two different types of write access by setting the Byte Access Type bit.

1. Byte select access (BAT = 1): uses one write signal, one read signal, and two signals to select upper and lower memory bank (used for SRAM) in a 16-bit memory.

   Typically used with 16-bit memories, except when user want to connect 2x8-bit memories in parallel, in that case seen by the AMC this is a 16-bit memory.

2. Byte write access (BAT = 0): uses two write signals for selecting two different 8-bit memories and a read signal.

   Typically used with 2x8-bit memories, this mode is used at reset to boot on the memory connected on NCS0 (Chip Select 0).

Byte Select Access (BAT = 1)

This mode is selected by setting the BAT bit to 1 in AMC_CSRX registers and is typically used to connect the EBI with a 16-bit memory. All 2X8-bit memories can be connected in this mode.

Users can use the upper/lower bank selection signal to get either an 8-bit or a 16-bit access.

The signal NOE is used for reading and the signal NWE is used for writing. Signals NUB (upper bank selection) and NLB (lower bank selection) are used to have a 8-bit access.

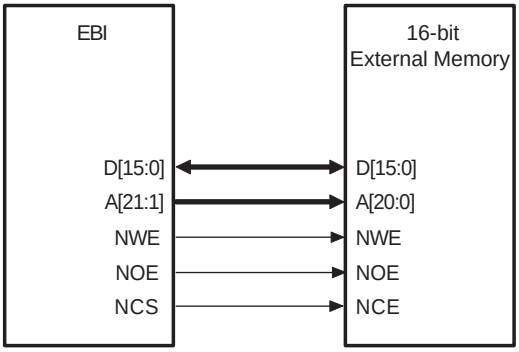
The following illustrations Figure 9, Figure 10 on page 28 and Figure 11 on page 28 show how to connect a typical 16-bit memory and 2x8-bit memories with 16-bit access and a 16-bit memory with 8-bit access.

16-bit Access Device Connection

A typical 16-bit memory (e.g. Flash) device connection with 16-bit access is listed below, NLB and NUB are not used.

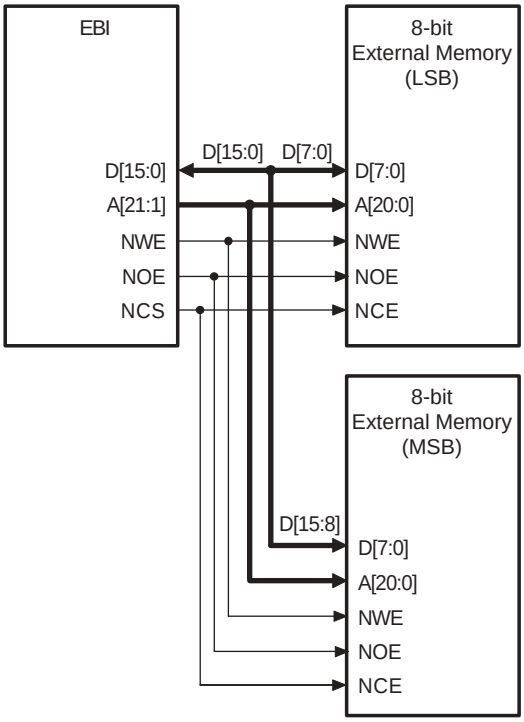
- The A0/NLB signal is not used
- The NWR1/NUB signal is not used
- The NWR0/NWE signal is used as NWE and enables half-word writes.
- The NRD/NOE signal is used as NOE and enables half-word and byte reads.

Figure 9. EBI Connection for External 16-bit Memory Device, 16-bit Access Only



In the same configuration as shown in Figure 9 above, it is possible to connect 2x8-bit memory devices with 16-bit access. The configuration shown in Figure 10 on page 28 demonstrates how to interface the EBI with 2x8-bit memories (for example 2x8-bit ROM memory) as a 16-bit memory page.

Figure 10. EBI Connection for 2x8-bit External Memory Devices, 16-bit Access Only



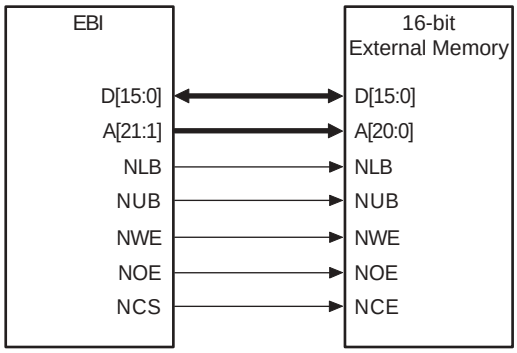
8-bit or 16-bit Access Device Connection

A typical 16-bit memory (e.g. 16-bit SRAM) device connection with 8- or 16-bit access is shown below.

This 16-bit memory allows upper/lower bank selection and NUB, NLB are used to achieve an 8-bit access.

- The A0/NLB signal is used as NLB and enables the lower byte for both read and write operations.
- The NWR1/NUB signal is used as NUB and enables the upper byte for both read and write operations.
- The NWR0/NWE signal is used as NWE and enables half-word or byte writes.
- The NRD/NOE signal is used as NOE and enables half-word and byte reads.

Figure 11. EBI Connection for External 16-bit Memory Devices, 8- or 16-bit Access



Byte Write Access (BAT = 0)

This mode is selected by setting the BAT bit to 0 in AMC_CSRX registers. This is the mode selected at reset.

In this mode users can interface the EBI with one or two 8-bit memories.

If the EBI is interfaced with two 8-bit memories, then users can choose to have either an 8- or 16-bit access.

The NRD signal is used for reading and two signals are used for writing, NWR0 for lower byte writes and NWR1 for upper byte writes.

Figure 12 shows how to connect one and two 8-bit memories with the EBI.

The example shown in Figure 13 demonstrates what happens when AT91SAM7A2 boots in this mode on a 16-bit memory (type Flash).

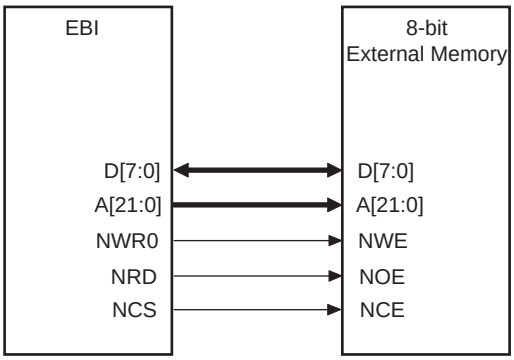
8-bit Access Device Connection

A typical 8-bit memory device connection with 8-bit access is shown here.

DBW[1:0] should be set for a 8-bit-data bus width and only NWR0 is used.

- The A0/NLB signal is used as A0.
- The NWR1/NUB signal is not used.
- The NWR0/NWE signal is used as NWR0 and enables lower byte writes.
- The NRD/NOE signal is used as NRD and enables half-word and byte reads.

Figure 12. EBI Connection for External 8-bit Memory Device, 8-bit Access Only

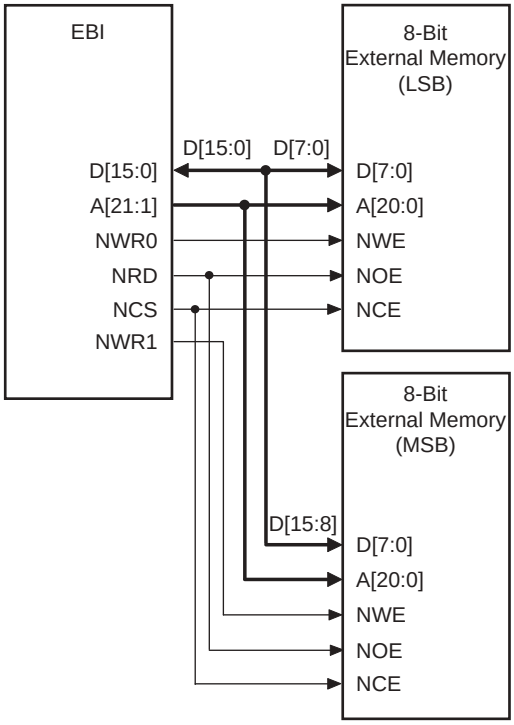


8-bit or 16-bit Access Device Connection

A typical 2x8-bit memory device connection with 8-bit or 16-bit access is shown below.

- The A0/NLB signal is not used.
- The NWR1/NUB signal is used as NWR1 and enables upper byte writes.
- The NWR0/NWE signal is used as NWR0 and enables lower byte writes.
- The NRD/NOE signal is used as NRD and enables half-word and byte reads.

Figure 13. EBI Connection for External 2x8-bit Memory Devices, 8-or 16-bit Access



16-bit Access Device Connection

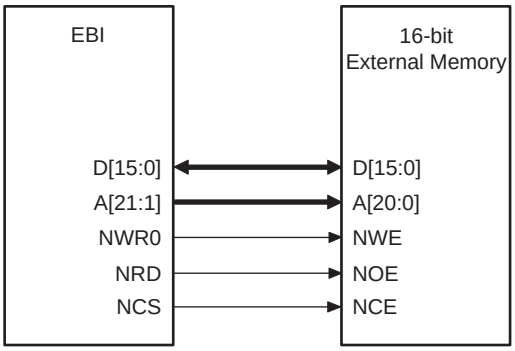
A typical 16-bit memory device connection with 16-bit access only is shown below.

This is typically the configuration of the memory after a reset or at power up when using a 16-bit flash memory on NCS0, in that case AT91SAM7A2 is in byte write access mode and boots on the 16-bit memory. NWR1 and NWR0 are used by the EBI but only NWR0 is used by the memory enabling a 16-bit access.

The correct mode to use with this configuration is byte select access and should be set in the boot.

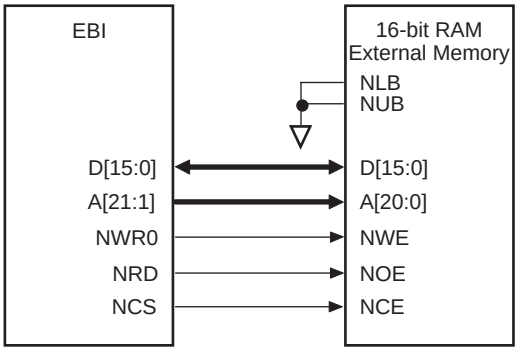
- The A0/NLB signal is not used.
- The NWR1/NUB signal is not used.
- The NWR0/NWE signal is used as NWR0 and enables half-word writes.
- The NRD/NOE signal is used as NRD and enables half-word and byte reads.

Figure 14. EBI Connection for External 16-bit Memory Devices, 16-bit Access Only



If users want to boot on a RAM memory for debug purposes, the RAM memory should be connected the same way as a flash memory (NUB and NLB of the RAM memory connected to the ground) to emulate a pure 16-bit Flash memory as shown in Figure 13.

Figure 15. EBI Connected to an External 16-bit RAM Memory Device, 16-bit Access Only and Used as Boot Memory for Debug Purposes



External Bus Interface Timings

Simple read and write access cycles are explained in detail where read access can be done through two modes as follows:

- The standard read protocol.
- The early read protocol, which increases the EBI performance for read access.

The EBI can automatically insert wait states during the external access cycles. These wait states are applied within the actual access cycle.

Data float wait states can also be inserted and applied in between cycles. The data float wait states depend strongly on the previous and next access contingent upon whether the state is a write or read cycle (early or standard) and if it is on the same chip select. These conditions are detailed in the pages that follow.

Read Access

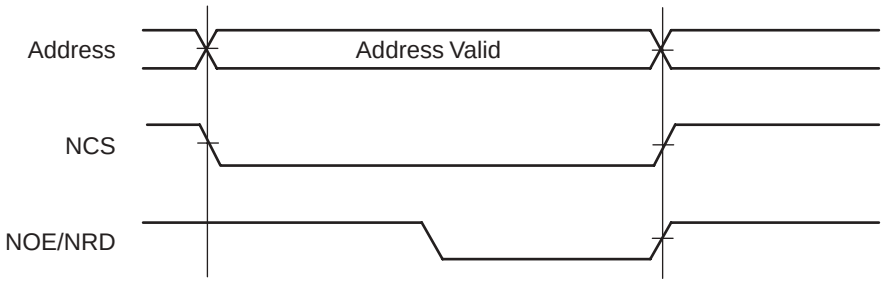
Standard Read Protocol

Standard read protocol (default read mode) implements a read cycle in which NRD/NOE is active during the second half of the read cycle.

The first half of the read cycle allows time to ensure completion of the previous access as well as address and NCS output before the read cycle begins.

During a standard read protocol external memory access, NCS is set low and ADDR is valid at the beginning of the access while NRD/NOE goes low only in the second half of the read cycle to avoid bus conflict.

Figure 16. Standard Read Address

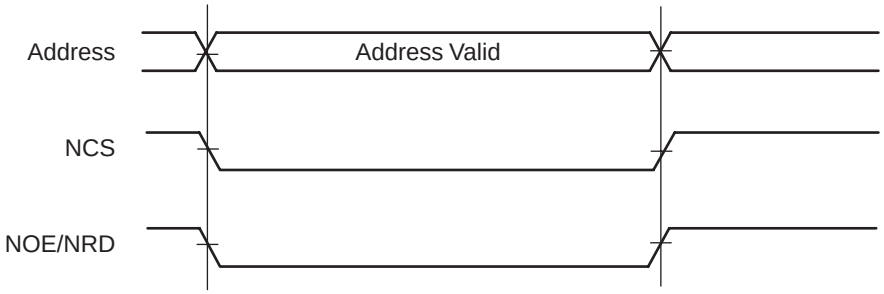


Early Read Protocol

Early read protocol provides more memory access time for a read access by asserting NRD at the beginning of the read cycle. This mode is selected by setting the DRP bit in the AMC_MCR register.

In the case of successive read cycles in the same memory, NRD remains active continuously. Since a read cycle normally limits the speed of operation of the external memory system, early read protocol can allow a faster timing of the EBI to be used. However, an extra data float wait state is required in some cases to avoid contentions on the external bus, this is explained in “Data Float Wait State” on page 34.

Figure 17. Early Read Address



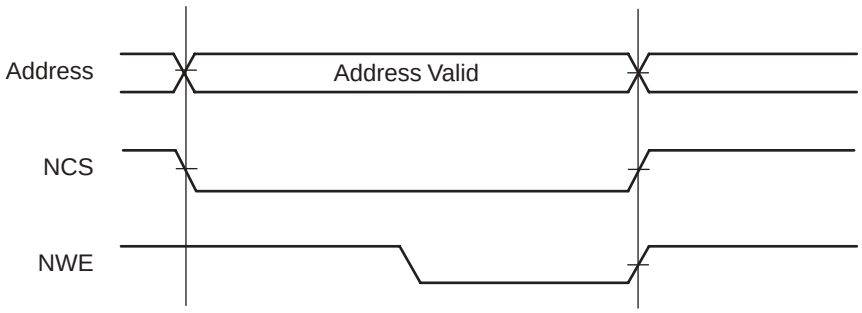
Write Access

In a write access cycle, NWE (or NWR0, NWR1) is active during the second half of the write cycle.

The first half of the write cycle allows time to ensure completion of the previous access as well as the address and NCS set up time before NEW (or NWR0, NWR1) is asserted.

During a write external memory access, NCS is set low and ADDR is valid at the beginning of the access while NWE (or NWR0, NWR1) goes low only in the second half of the write cycle to avoid bus conflict.

Figure 18. Write Access



NWE (or NWR0, NWR1) goes high at the end of the write cycle, this is not true if a wait state is asserted.

Wait State

Each chip select line can be programmed to insert one or more wait states during an external access. This is done by setting the WSE bit in the corresponding AMC_CSRx register. The number of cycles to insert is programmed in the NWS[2:0] field in the same register.

The correspondence between the number of standard wait states programmed and the number of cycles during which the NWE pulse is held low is as follows:

- 0 wait states 1/2 cycle
- 1 wait state 1 cycle

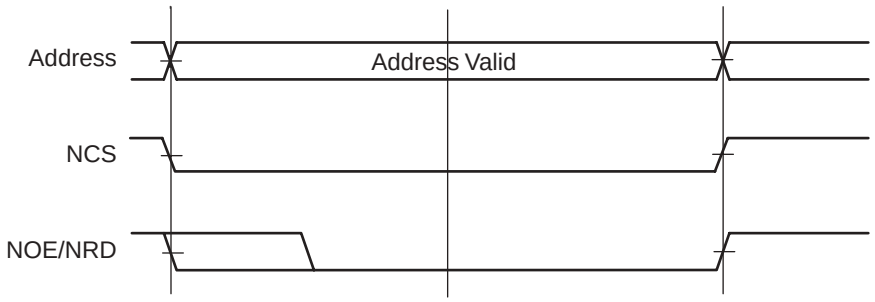
For each additional wait state programmed, an additional cycle is added.

Wait state with Read Cycle

The read cycle is delayed one cycle for each wait state programmed. In early mode, NOE/NRD goes low at the start of the read cycle while in standard mode, this signal goes low at the half of the first cycle.

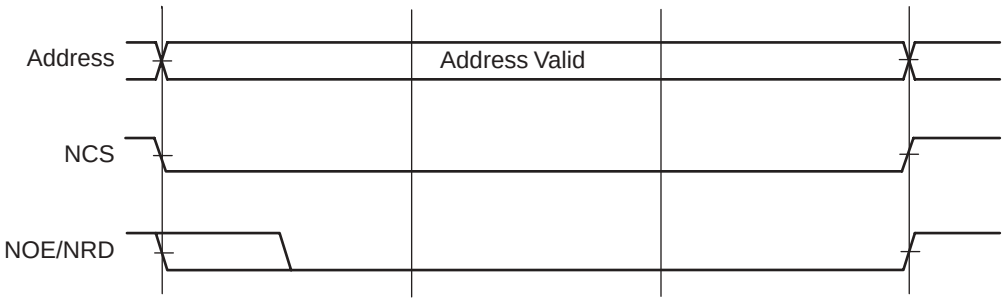
The following figure shows a read cycle with one wait state.

Figure 19. Read Cycle with One Wait State



The following figure shows a read cycle with two wait states.

Figure 20. Read Cycle with Two Wait States

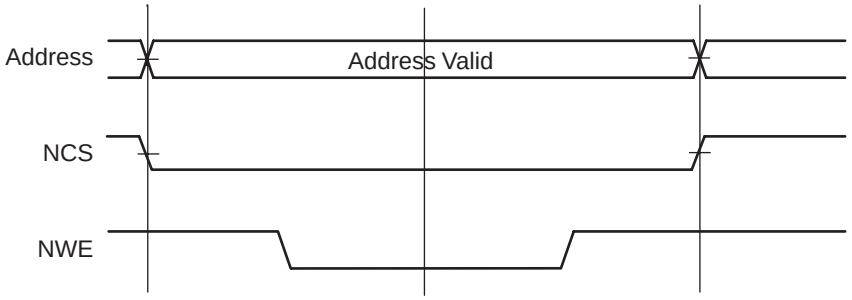


Wait State with Write Cycle

The write cycle is delayed one cycle for each wait state programmed. NWE (or NWR0, NWR1) goes high one half cycle before the end of the write cycle.

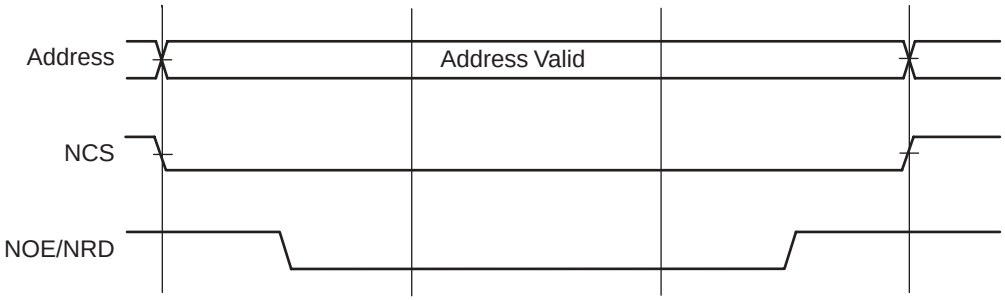
The following figure shows a write cycle with one wait state.

Figure 21. Write Cycle with One Wait State



The following figure shows a write cycle with two wait states.

Figure 22. Write Cycle with Two Wait States



Data Float Wait State

Data float wait states are added to avoid data bus conflict.

After a read access, the data float wait state gives more time for the external memory to release the data bus.

After a write access the data float wait state gives more time for the EBI to release the data bus.

The Data Float Output Time (t_{DF}) for each external memory device is programmed in the TDF field of the AMC_CSR register for the corresponding chip select. The value (0-7 clock cycles) indicates the number of data float wait states to be inserted.

Data float wait state are asserted in between accesses.

Data float wait state insertion depends strongly on the previous access and the next access contingent upon whether the state is a write or read cycle (early or standard) and if it is on the same chip select.

The following table describes the data float wait state applied.

Table 23. Data Float State Applied

		Number of Data Float Wait State Applied	
Previous Access	**Next Access**	**Early Read Mode**	**Standard Read Mode**
NCSx Read	NCSx Read	0	0
NCSx Read	NCSx Write	nTDF	NTDF
NCSx Write	NCSx Read	1	0
NCSx Write	NCSx Write	0	0
NCSx Read	NCSy Read	Max(1, nTDFx)	Max(1, nTDFx)
NCSx Read	NCSy Write	Max(1, nTDFx)	Max(1, nTDFx)
NCSx Write	NCSy Read	1	1
NCSx Write	NCSy Write	1	1

Table 24. Examples

Previous Access	Next Access	Early Read Mode				Standard Read Mode			
		TDFx = 0	TDFx = 1	TDFx = 2	TDFx = 3	TDFx = 0	TDFx = 1	TDFx = 2	TDFx = 3
NCSx Read	NCSx Read	0	0	0	0	0	0	0	0
NCSx Read	NCSx Write	0	1	2	3	0	1	2	3
NCSx Write	NCSx Read	1	1	1	1	0	0	0	0
NCSx Write	NCSx Write	0	0	0	0	0	0	0	0
NCSx Read	NCSy Read	1	1	2	3	1	1	2	3
NCSx Read	NCSy Write	1	1	2	3	1	1	2	3
NCSx Write	NCSy Read	1	1	1	1	1	1	1	1
NCSx Write	NCSy Write	1	1	1	1	1	1	1	1

The waveforms appearing below and on the following pages give an exhaustive description of how data float wait states apply.

Figure 23. Read and Write Access on Different Chip Select with NTDF = 0 or 1

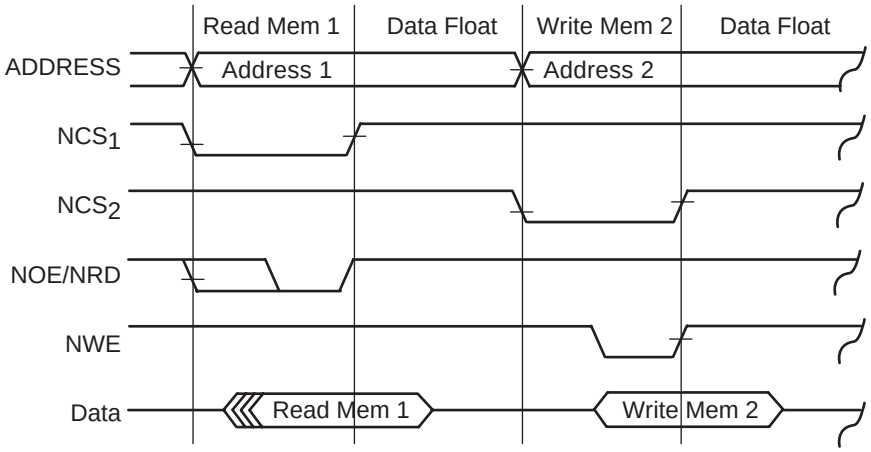


Figure 24. Read and Write Access on Different Chip Select with NTDF = 2

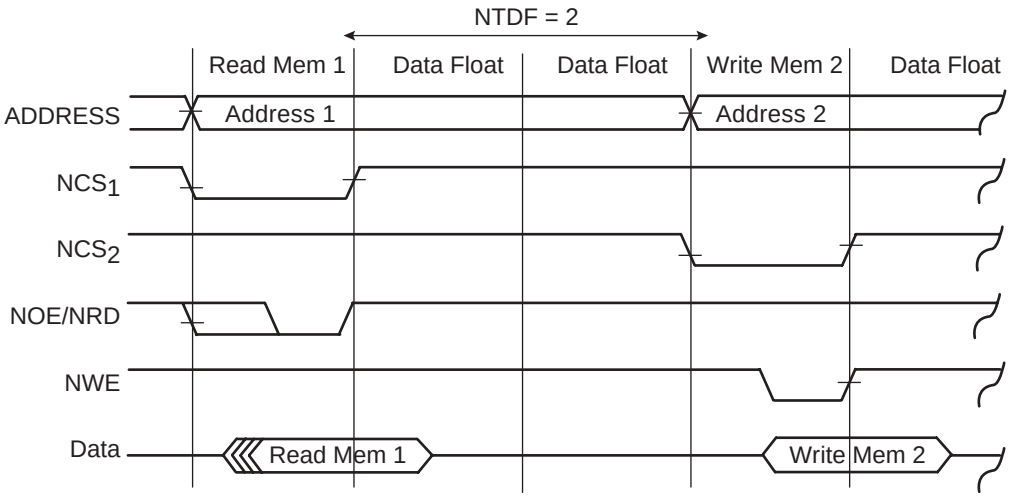


Figure 25. Standard Read and Write Access on the Same Chip Select with NTDF = 2

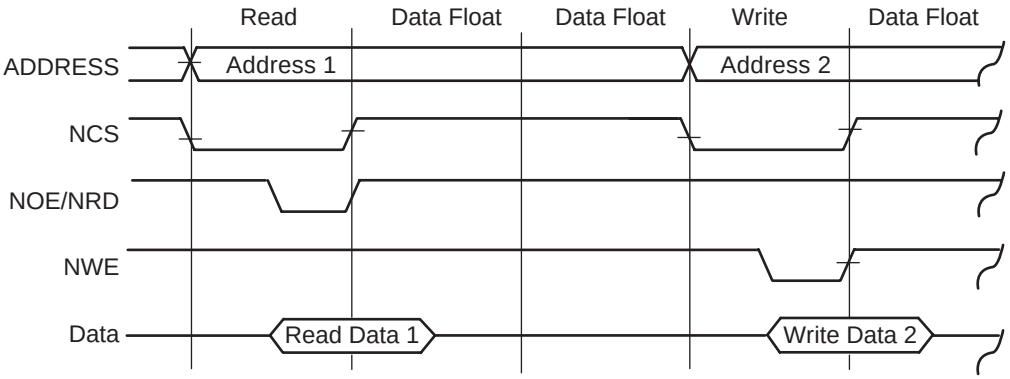


Figure 26. Sequential Early Read Access on the Same Chip Select with One Wait State

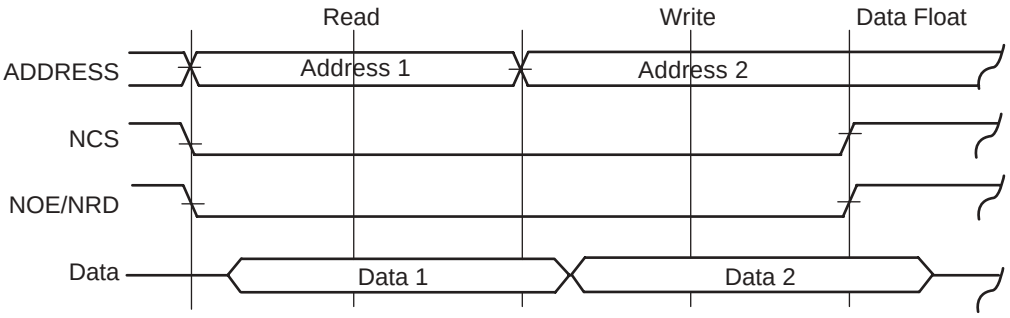


Figure 27. Sequential Early Read Access on the Same Chip Select with No Wait State

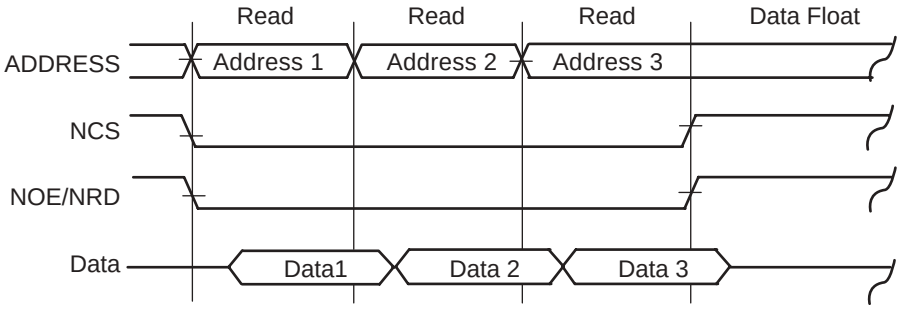


Figure 28. Sequential Read Access on Different Chip Select with NTDF = 2

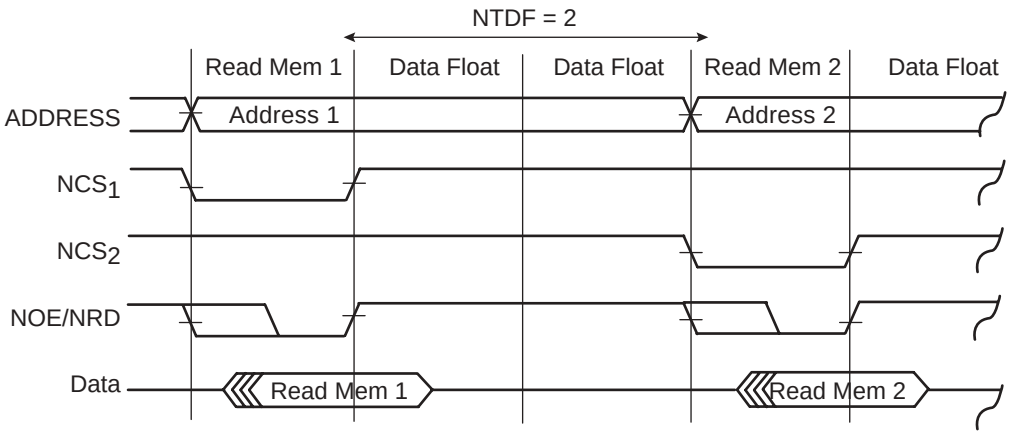


Figure 29. Sequential Read Access on Different Chip Select with NTDF = 0 or 1

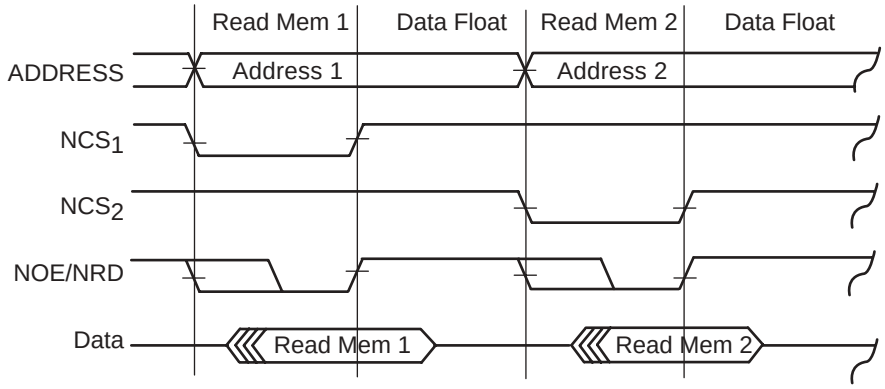


Figure 30. Sequential Standard Read Access on the Same Chip Select with One Wait State

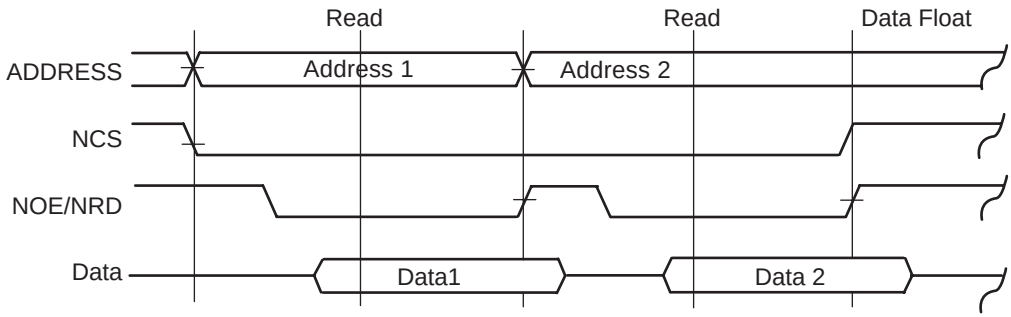


Figure 31. Sequential Standard Read Access on the Same Chip with No Wait State

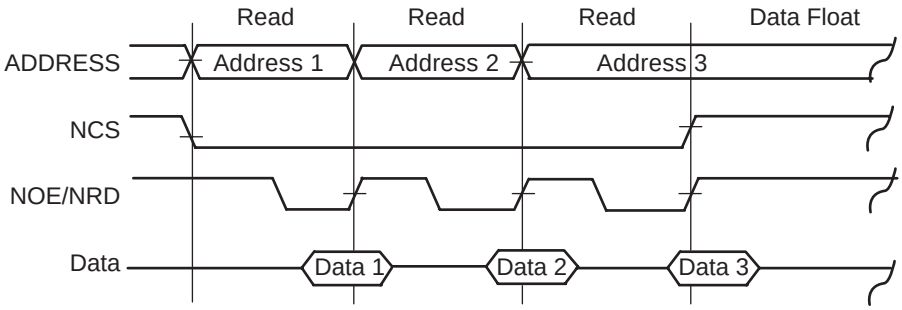


Figure 32. Sequential Write Access on the Same Chip Select with One Wait State

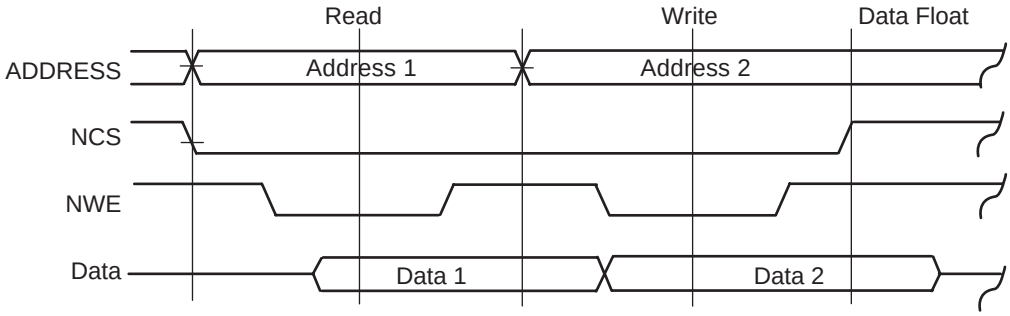


Figure 33. Sequential Write Access on the Same Chip Select with No Wait State

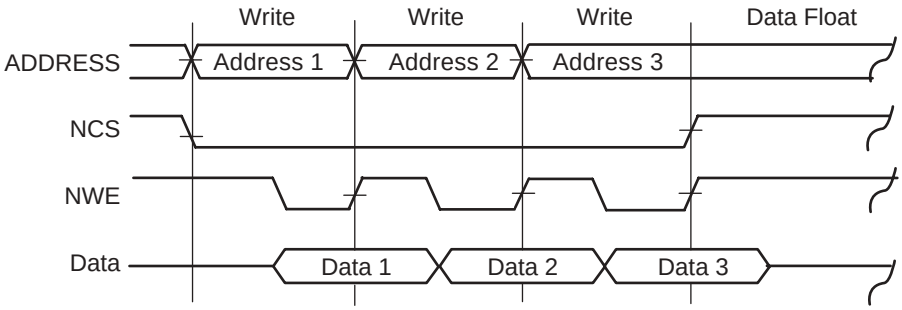


Figure 34. Sequential Write Access on Different Chip Select

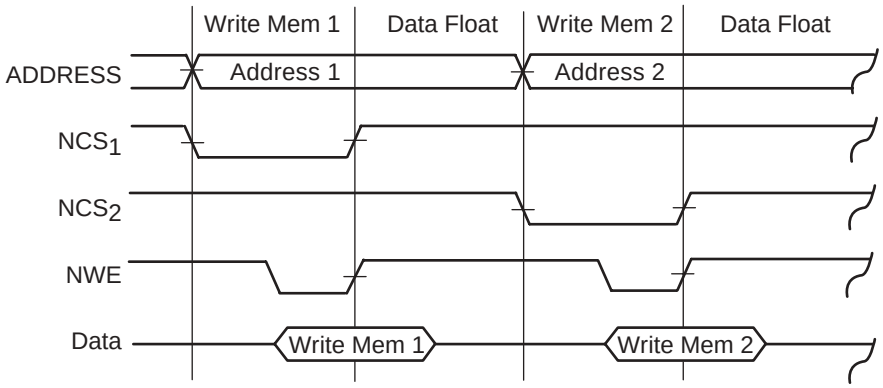


Figure 35. Write and Early Read on the Same Chip Select

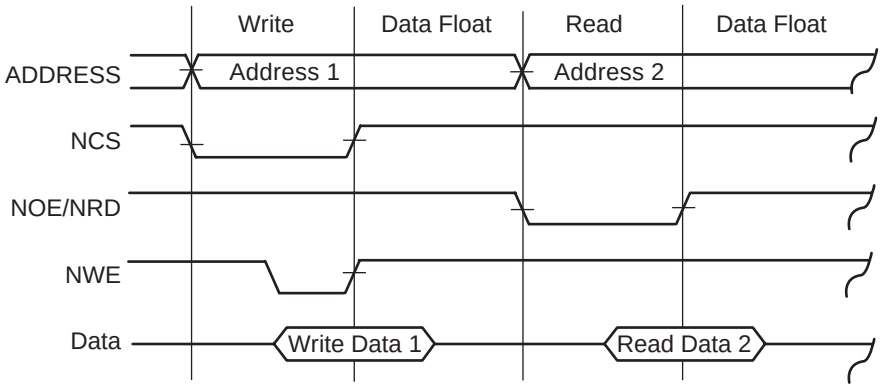


Figure 36. Write and Read on Different Chip Select

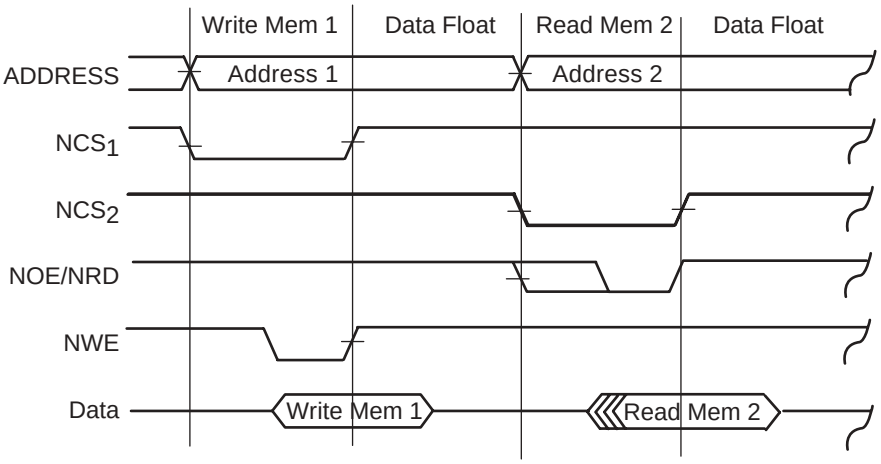
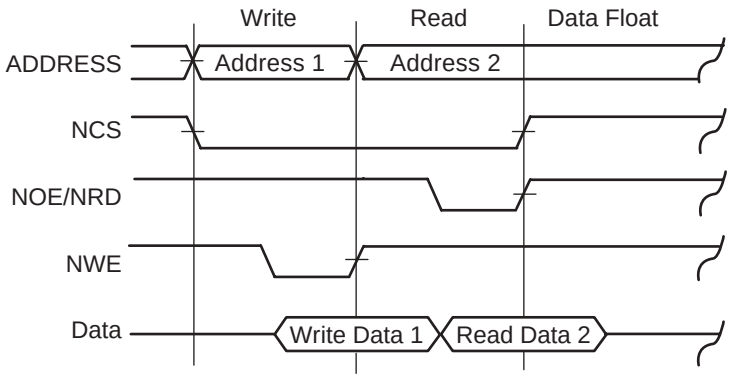


Figure 37. Write and Standard Read on the Same Chip Select



Timings

Table 25 below and Table 26 on page 41 show the minimum and maximum timings for external memory Read/Write cycles (valid over the recommended operating conditions) for a capacitive load of 15 pF and 30 pF. These timings are represented on the relevant waveform.

Table 25. Timings for Read Access

Read Access		Load = 15pf			Load = 30pf		
		MIN	TYP	MAX	MIN	TYP	MAX
$tr_{ADCRDV2}$	Address Change to Read Data Valid (Read/Write Memory, 0 Wait State, standard read)			t_{CYCLE} - 27 ns			t_{CYCLE} - 31 ns
$tr_{ADCRDV1}$	Address Change to Read Data Valid (All other cases)			(1 + NWS) * t_{CYCLE} - 15 ns			(1 + NWS) * t_{CYCLE} - 17 ns
tr_{CSLRDV}	Chip Select Low to Read Data Valid			(1 + NWS) * t_{CYCLE} - 14 ns			(1 + NWS) * t_{CYCLE} - 16 ns
$tr_{OELRDV1}$	Output Enable Low to Read Data Valid (standard read)			(0.5 + NWS) * t_{CYCLE} - 12 ns			(0.5 + NWS) * t_{CYCLE} - 13 ns
$tr_{OELRDV2}$	Output Enable Low to Read Data Valid (early read)			(1 + NWS) * t_{CYCLE} - 12 ns			(1 + NWS) * t_{CYCLE} - 13 ns
$tr_{BSLRDV2}$	Byte Select Low to Read Data Valid (Read/Write Memory, 0 Wait State, standard read)			(1 + NWS) * t_{CYCLE} - 26 ns			(1 + NWS) * t_{CYCLE} - 31 ns
$tr_{BSLRDV1}$	Byte Select Low to Read Data Valid (all other cases)			(1 + NWS) * t_{CYCLE} - 16 ns			(1 + NWS) * t_{CYCLE} - 18 ns
tr_{DH}	Data Hold-time from Address Change/NCS High/NOE High	0			0		
$tr_{DHZOEL1}$	Data Hi-Z to Output Enable Low (previous is a write cycle - standard read)	- 2 ns			- 3 ns		
$tr_{DHZOEL2}$	Data Hi-Z to Output Enable Low (previous is a write cycle - early read)	0.5 * t_{CYCLE} - 1 ns			0.5 * t_{CYCLE} - 3 ns		
tr_{DHZCSL}	Data Hi-Z to Chip Select Low (previous is a write cycle - standard)	0.5 * t_{CYCLE} + 1 ns			0.5 * t_{CYCLE} + 1 ns		

Table 26. Timings for Write Access

Write Access		Load = 15pf			Load = 30pf		
		MIN	TYPICAL	MAX	MIN	TYPICAL	MAX
tw_{ADSWL}	Address/NCS/NUB/NLB Setup-time to Write Low	0.5 * t_{CYCLE} - 1 ns			0.5 * t_{CYCLE} - 1 ns		
tw_{WPL1}	Write Pulse Low (one or more Wait States)	(1+ NWS) * t_{CYCLE} - 1 ns			(1+ NWS) * t_{CYCLE} - 1 ns		
tw_{WPL2}	Write Pulse low (0 Wait State)	0.5 * t_{CYCLE} - 1 ns			0.5 * t_{CYCLE} - 1 ns		
tw_{DSWH1}	Data Setup-time to Write High (one or more Wait States)	(1+ NWS) * t_{CYCLE} + 1 ns			(1+ NWS) * t_{CYCLE} + 1 ns		
tw_{DSWH2}	Data Setup time to Write High (0 Wait State)	0.5 * t_{CYCLE}			0.5 * t_{CYCLE}		
tw_{ADHWH}	Address/CS/NUB/NLB Hold-time from Write High	5 ns			6 ns		
tw_{OEHDD}	Output Enable High (previous is a read cycle) to Data Drive	0.5 * t_{CYCLE} - 3 ns			0.5 * t_{CYCLE} - 3 ns		
tw_{CSHDD}	Chip Select High (previous is a read cycle) to Data Drive	1.5 * t_{CYCLE} - 4 ns			1.5 * t_{CYCLE} - 7 ns		
tw_{DHWH1}	Data Hold-time from Write High (one or more Wait States)	t_{CYCLE} - 6 ns			t_{CYCLE} - 8 ns		
tw_{DHWH2}	Data Hold-time from Write High (0 Wait State)	0.5 * t_{CYCLE} - 6 ns			0.5 * t_{CYCLE} - 8 ns		

Figure 38. Read Access Waveform

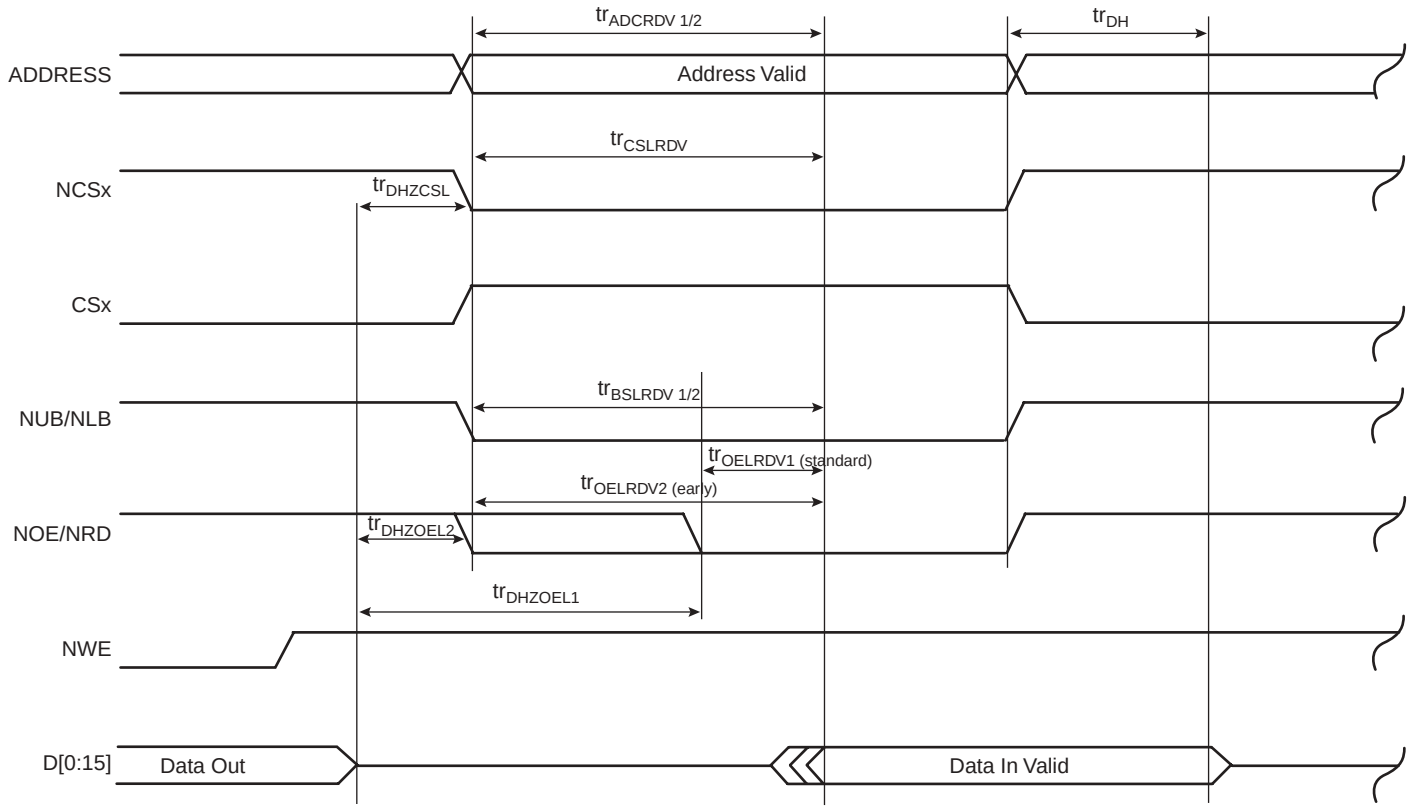
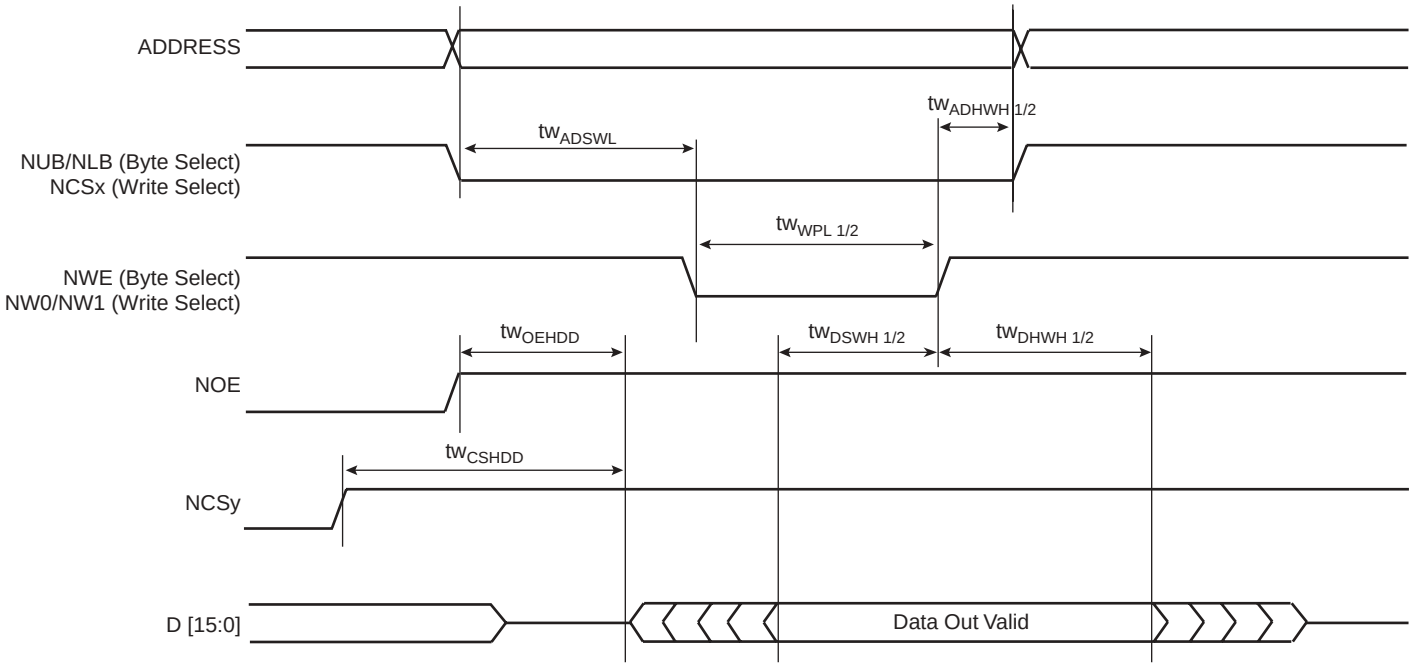


Figure 39. Write Access Waveform



Advanced Memory Controller (AMC) Memory Map

Base Address: 0xFFE0000

Table 27. AMC Memory Map [1] [2]

Offset	Register	Name	Access	Reset State
0x000	AMC Chip Select Register 0	AMC_CSR0	Read/Write	0x4000203D
0x004	AMC Chip Select Register 1	AMC_CSR1	Read/Write	0x48000000
0x008	AMC Chip Select Register 2	AMC_CSR2	Read/Write	0x50000000
0x00C – 0x018	Reserved	–	–	–
0x01C	AMC Chip Select Register 3	AMC_CSR3	Read/Write	0x78000000
0x020	AMC Remap Control Register	AMC_RCR	Read/Write	0x00000000
0x024	AMC Memory Control Register	AMC_MCR	Read/Write	0x00000000

Notes: 1. The software must set the AMC Registers for correct operation.
2. In register tables, individual bits are defined: W: Write, R: Read, -0: 0 after reset, -1: 1 after reset, -U: undefined after reset.

AMC Chip Select Register 0

Name: AMC_CSR0
Access: Read/Write
Base Address: 0x000

31	30	29	28	27	26	25	24
–	–	BA[9:4]					

23	22	21	20	19	18	17	16
BA[3:0]				–	–	–	–

15	14	13	12	11	10	9	8
–	–	CSEN	BAT	TDF[2:0]			PAGES1

7	6	5	4	3	2	1	0
PAGES0	–	WSE	NWS[2:0]			DBW[1:0]	

- **BA[9:0]: Base Address**

Bits [31:30] are set by hardware, so the base address can only be in the memory space 0x40000000-0x7FFFFFFF. The other bits contain the highest bits of the base address. If the page size is larger than 1Mbyte, the unused bits of the base address are ignored by the AMC decoder.

- **CSEN: Chip Select Enable**

0: Chip select is disabled.

1: Chip select is enabled.

- **BAT: Byte Access Type**

0: Byte write access type.

1: Byte select access type.

- **TDF[2:0]: Data Float Output Time**

These bits select the number of cycles added after a memory transfer.

TDF[2:0]			Cycles Added
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

- **PAGES[1:0]: Page Size**

These bits select the memory page size.

PAGES[1:0]		Page Size	Active Bits in Base Address
0	0	1 Mbytes	12 (31-20)
0	1	4 Mbytes	10 (31-22)
1	0	16 Mbytes	8 (31-24)
1	1	64 Mbytes	6 (31-26)

- **WSE: Wait State Enable**

0: Wait state generation is disabled. No wait state is inserted.

1: Wait state generation is enabled.

- **NWS[2:0]: Number of Wait States**

These bits select the number of wait states added. This field is only valid if the **WSE** bit is set.

NWS[2:0]			WS Added
0	0	0	1
0	0	1	2
0	1	0	3
0	1	1	4
1	0	0	5
1	0	1	6
1	1	0	7
1	1	1	8

- **DBW[1:0]: Data Bus Width**

Type of data bus selected.

DBW[1:0]		Data Bus Width
0	0	Reserved
0	1	16-bit Data Bus
1	0	8-bit Data Bus
1	1	Reserved

AMC Chip Select Register

Name: AMC_CSR1...AMC_CSR3
Access: Read/Write
Base Address: 0x004, 0x008, 0x01C

31	30	29	28	27	26	25	24
–	–	BA[9:4]					

23	22	21	20	19	18	17	16
BA[3:0]				–	–	–	–

15	14	13	12	11	10	9	8
–	–	CSEN	BAT	TDF[2:0]			PAGES1

7	6	5	4	3	2	1	0
PAGES0	–	WSE	NWS[2:0]			DBW[1:0]	

- **BA[9:0]: Base Address**

Bits [31:30] are set by hardware, so the base address can only be in the memory space 0x40000000-0x7FFFFFFF. The other bits contain the highest bits of the base address. If the page size is larger than 1Mbyte, the unused bits of the base address are ignored by the AMC decoder.

- **CSEN: Chip Select Enable**

0: Chip select is disabled.

1: Chip select is enabled.

- **BAT: Byte Access Type**

0: Byte write access type.

1: Byte select access type.

- **TDF[2:0]: Data Float Output Time**

These bits select the number of cycles added after a memory transfer.

TDF[2:0]			Cycles Added
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

- **PAGES[1:0]: Page Size**

These bits select the memory page size.

PAGES[1:0]		Page Size	Active Bits in Base Address
0	0	1 Mbytes	12 (31-20)
0	1	4 Mbytes	10 (31-22)
1	0	16 Mbytes	8 (31-24)
1	1	64 Mbytes	6 (31-26)

- **WSE: Wait State Enable**

0: Wait state generation is disabled. No wait state is inserted.

1: Wait state generation is enabled.

- **NWS[2:0]: Number of Wait States**

These bits select the number of wait states added. This field is only valid if the **WSE** bit is set.

NWS[2:0]		WS Added	
0	0	0	1
0	0	1	2
0	1	0	3
0	1	1	4
1	0	0	5
1	0	1	6
1	1	0	7
1	1	1	8

- **DBW[1:0]: Data Bus Width**

Type of data bus selected.

DBW[1:0]		Data Bus Width
0	0	Reserved
0	1	16-bit Data Bus
1	0	8-bit Data Bus
1	1	Reserved

AMC Remap Control Register

Name: AMC_RCR
Access: Read/Write
Base Address: 0x020

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
–	–	–	–	–	–	–	RCB

- **RCB: Remap Command Bit**

0: No effect

1: Cancel the remapping (performed at reset) of the two memory devices (internal RAM and external memory on NCS0).

This bit is read at a logical 0 during remapping and read at logical 1 when remapping has been canceled.

AMC Memory Control Register

Name: AMC_MCR
Access: Read/Write
Base Address: 0x024

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
–	–	–	DRP	–	ALE[2:0]		

- **DRP: Data Read Protocol**

0: standard read protocol for all external memory devices enabled.

1: early read protocol for all external memory devices enabled.

- **ALE[2:0]: Address Line Enable**

These bits indicate the number of valid chip select lines.

ALE2	ALE1	ALE0	Valid Address Bits	Maximum Addressable Space per Chip Select Line	Valid Chip Select
0	x	x	ADD[20:0]	2 Mbytes	NCS[2:0]
1	0	x	ADD[20:0]	2 Mbytes	NCS[2:0]
1	1	0	ADD[20:0]	2 Mbytes	NCS[2:0]
1	1	1	ADD[19:0]	1 Mbytes	NCS[2:0], CS3

Clock Manager (CM)

Overview

The AT91SAM7A2 microcontroller provides:

- 32.768 kHz Oscillator (real time clock oscillator)
- 2 MHz to 6 MHz Oscillator
- Programmable PLL (x2 to x20)
- Programmable Master Clock Divider

The clock management is done through the Clock Manager (CM). This allows the user to select between the different working modes; LPM, SLM and OPE.

At power up, the master clock oscillator and the real time clock oscillator are enabled. As the application can use (or not) the real time clock oscillator, the DIVCLK clock is used as the low frequency clock (LFCLK). This ensures that both the CORECLK and the LFCLK clock can be used at power up. The master clock (MCK) is multiplied by 10 through the PLL and divided by 2 giving a core clock frequency (CORECLK) equal to MCK x 5.

Figure 40. Clock Management

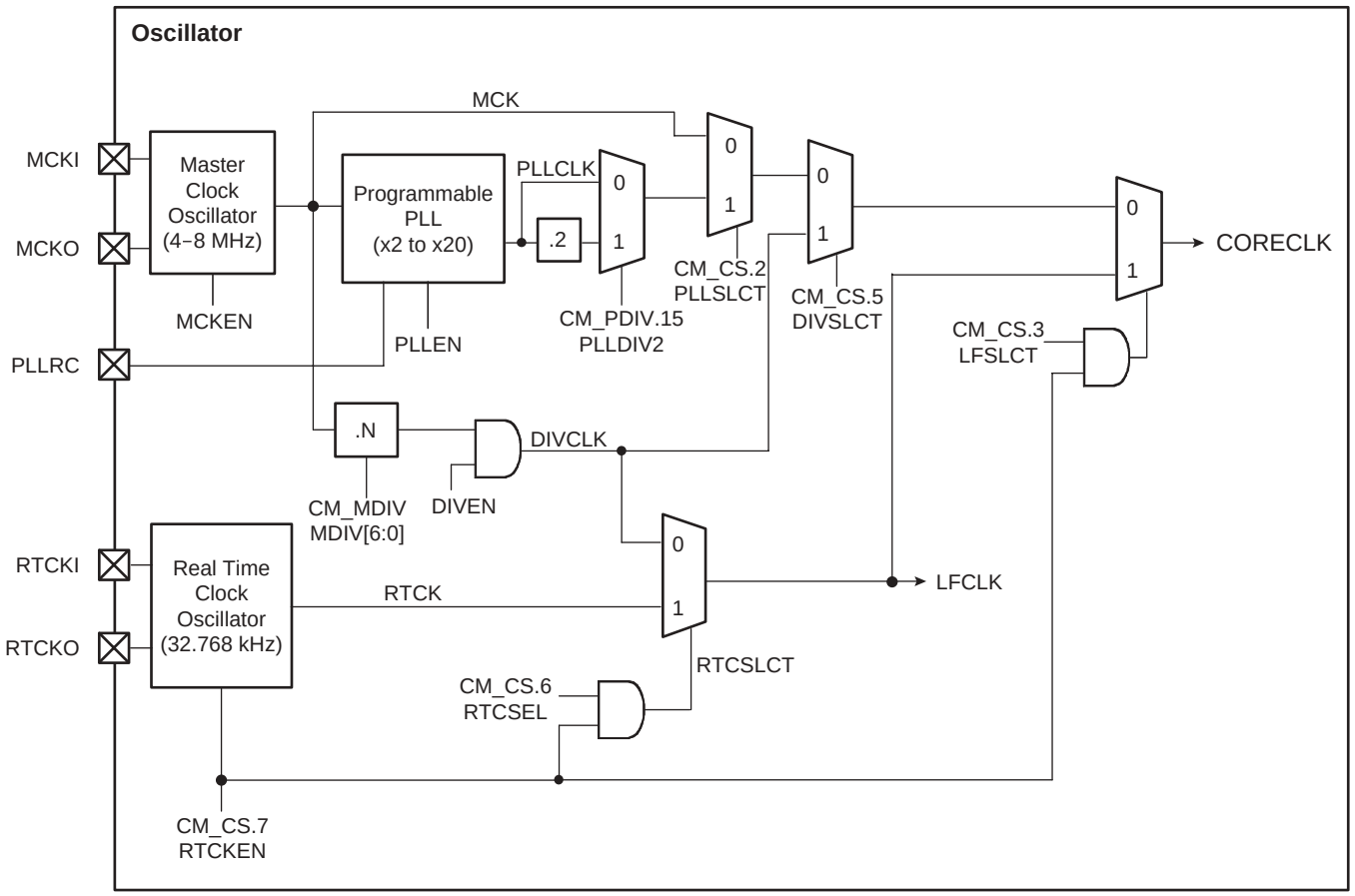


Table 28 below lists the different working modes according to value set in the CM_CS register. Line 14 show values at reset.

Table 28. Clock Selection

Mode Number	CM_CS.7 RTCKEN	CM_CS.6 RTCSEL	CM_CS.3 LFSLCT	CM_CS.5 DIVSLCT	CM_CS.2 PLLSLCT	CM_CS.1 PLLDIV2	CM_CS.8 MCKEN	CM_CS.9 PLLEN	CM_CS.13 DIVEN	CM_CS.14 RTCSLCT	CORECLK	LFCLK	Mode
1	0	X	X	1	X	X	1	0	1	0	DIVCLK	DIVCLK	SLM
2	0	X	X	0	0	X	1	0	1	0	MCK	DIVCLK	SLM
3	0	X	X	0	1	0	1	1	1	0	PLLCLK	DIVCLK	OPE
4	0	X	X	0	1	1	1	1	1	0	PLLCLK/2	DIVCLK	OPE
5	1	1	1	X	X	X	0	0	0	1	RTCK	RTCK	LPM
6	1	1	0	1	X	X	1	0	1	1	DIVCLK	RTCK	SLM
7	1	1	0	0	0	X	1	0	0	1	MCK	RTCK	SLM
8	1	1	0	0	1	0	1	1	0	1	PLLCLK	RTCK	OPE
9	1	1	0	0	1	1	1	1	0	1	PLLCLK/2	RTCK	OPE
10	1	0	1	X	X	X	1	0	1	0	DIVCLK	DIVCLK	SLM
11	1	0	0	1	X	X	1	0	1	0	DIVCLK	DIVCLK	SLM
12	1	0	0	0	0	X	1	0	1	0	MCK	DIVCLK	SLM
13	1	0	0	0	1	0	1	1	1	0	PLLCLK	DIVCLK	OPE
14	1	0	0	0	1	1	1	1	1	0	PLLCLK/2	DIVCLK	OPE

For each mode listed in Table 28 above, the corresponding equation is calculated for LFCLK and CORECLK as shown below in Table 29.

Table 29. Corresponding Equation for LFCLK and CORECLK

Mode Number	CORECLK	LFCLK
1	MCK/(2*(MDIV+1))	MCK/(2*(MDIV+1))
2	MCK	MCK/(2*(MDIV+1))
3	MCK*PLL	MCK/(2*(MDIV+1))
4	MCK*PLL/DIV2	MCK/(2*(MDIV+1))
5	RTCK	RTCK
6	MCK/(2*(MDIV+1))	RTCK
7	MCK	RTCK
8	MCK*PLL	RTCK
9	MCK*PLL/DIV2	RTCK
10	MCK/(2*(MDIV+1))	MCK/(2*(MDIV+1))
11	MCK/(2*(MDIV+1))	MCK/(2*(MDIV+1))
12	MCK	MCK/(2*(MDIV+1))
13	MCK*PLL	MCK/(2*(MDIV+1))
14	MCK*PLL/DIV2	MCK/(2*(MDIV+1))

Example

To switch from the default mode, OPE, to the LPM mode, do the following:

Configuration:

1. Enable the real time clock oscillator and the RTCSEL switch by writing the RTCKEN, RTCSEL bits and CLKEKEY field in the CM_CE register.
2. Then wait that the RTCSEL status flag is set in the CE_CS register. Once this bit is set, LFCLK clock is derived from the RCTK clock (see Figure 40 on page 50).
3. Then set the LFSLCT bit in the CM_CE register to obtain a core clock derived from RTCK.

To change the PLL multiplier from the default mode OPE, do the following:

1. Disable the PLL by writing the PLLSLCT bit in CM_CD. This this enables writing in the CM_PDIV and CM_PST registers. Now the core clock is equal to the master oscillator clock.
2. Then change the PLL multiplier and the divider by 2 in the CM_PDIV register. Users can also optimize the stabilization time of the PLL of CM_PST. This time is used in the next step when enabling the PLL. During that time the core clock will be cut.
3. Enable the PLL by writing the PLLSLCT bit in the CM_CE register. After this action, the core clock will be stopped during the time described by the PSTB field in the CM_PST register.

Clock Manager (CM) Memory Map

Base Address: 0xFFEC000

Table 30. Clock Manager Memory Map

Offset	Register	Name	Access	Reset State
0x000	CM Clock Enable	CM_CE	Write-only	–
0x004	CM Clock Disable	CM_CD	Write-only	–
0x008	CM Clock Status	CM_CS	Read-only	0x00002384
0x00C	CM PLL Stabilization Time	CM_PST	Read/Write	0x000000B0
0x010	CM PLL Divider	CM_PDIV	Read/Write	0x0000800A
0x014	CM Oscillator Stabilization Time	CM_OST	Read/Write	0x000000B0
0x018	CM Master Clock Divider	CM_MDIV	Read/Write	0x0000001F

CM Clock Enable Register

Name: CM_CE
Access: Write-only
Base Address: 0x000

31	30	29	28	27	26	25	24
CLKEYKEY[15:8]							
23	22	21	20	19	18	17	16
CLKEKEY[7:0]							
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
RTCKEN	RTCSEL	DIVSLCT	–	LFSLCT	PLLSLCT	–	–

- **CLKEKEY[15:0]: Key for Write Access into the CM_CE Register**

Any write in the CM_CD register bits will be effective only if CLKEKEY[15:0] is equal to 0x2305.

CM Clock Disable Register

Name: CM_CD
Access: Write-only
Base Address: 0x004

31	30	29	28	27	26	25	24
CLKEYKEY[15:8]							
23	22	21	20	19	18	17	16
CLKEKEY[7:0]							
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
RTCKEN	RTCSEL	DIVSLCT	–	LFSLCT	PLLSLCT	–	–

- **CLKDKEY[15:0]: Key for Write Access into the CM_CD Register**

Any write in the CM_CD register bits will be effective only if CLKDKEY[15:0] is equal to 0x1807.

CM Clock Status Register

Name: CM_CE
Access: Read-only
Base Address: 0x008

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

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

15	14	13	12	11	10	9	8
–	RTCSLCT	DIVEN	–	–	–	PLLEN	MCKEN

7	6	5	4	3	2	1	0
RTCKEN	RTCSEL	DIVSLCT	–	LFSLCT	PLLSLCT	–	–

- **PLLSLCT: PLL/Master Clock Selection**

0: Selects MCK clock (deselects PLLCLK or PLLCLK/2 clock).

1: Selects PLLCLK or PLLCLK/2 clock (deselects MCK clock).

- **LFSLCT: Low Frequency Clock Selection**

0: Allows selection of MCK, PLLCLK, PLLCK/2 or DIVCLK.

1: Selects low frequency clock LFCLK (also disables master clock oscillator and PLL).

- **DIVSLCT: Programmable Clock Selection**

0: Allows selection of MCK, PLLCK or PLLCLK/2 (also deselects the DIVCLK clock).

1: Selects DIVCLK i.e. MCK divided by MDIV[6:0] (also deselects the master clock or PLL clock).

- **RTCSEL: RTC frequency clock selection**

0: Selects the DIVCLK clock for low power clock (deselects the RTCK clock).

1: Selects the RTCK clock for low power clock (deselects the DIVCLK clock).

- **RTCKEN: Low Frequency Clock Oscillator**

0: The low frequency clock oscillator is disabled.

1: The low frequency clock oscillator is enabled.

- **MCKEN: Master Clock Oscillator Enable**

0: MCKEN signal is at a logical 0. The master clock oscillator is disabled and bypassed.

1: MCKEN signal is at a logical 1. The master clock oscillator is activated.

- **PLLEN: PLL Enable**

0: PLLEN signal is at a logical 0. PLL is deactivated.

1: PLLEN signal is at a logical 1. PLL is enabled.

- **DIVEN: Programmable Divider Enable**

0: DIVEN signal is at a logical 0. The programmable divider is disabled.

1: DIVEN signal is at a logical 1. The programmable divider is enabled.

- **RTCSLCT: Low Frequency Clock Selection**

0: RTCSLCT signal is at a logical 0. The DIVCLK is selected for LFCLK.

1: RTCSLCT signal is at a logical 1. The RTCK is selected for LFCLK.

CM PLL Stabilization Timer Register

Name: CM_PST
Access: Read/Write
Base Address: 0x00C

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

- **PSTB[9:0]: PLL Stabilization Time**

Number of clock cycles needed before PLL stabilization. This register gives the possibility to optimize the PLL stabilization time when changing the PLL multiplier. The PLL must be disabled before writing this register.

This stabilization time is used when the PLL is enabled, after a reset and at power up, during the stabilization time the clock is halted.

The default value is 0x000000B0 guarantying 176 clock cycles with MCK/256 clock (i.e. Tsetup = 176x(1/4MHz)*256 = 11.264 ms with MCK = 4.0 MHz). This default value includes the stabilization time for the oscillator and the PLL at start up and after a reset.

When the clock manager is configured in LPM mode and the program enables both the PLL and the master oscillator, PSTB should include the oscillator stabilization time and the PLL stabilization time. This is due to the fact that PSTB counter and OSTB counter decrement in parallel.

The PLL transient behavior before mathematical locking (phase error between the reference signal and derived signal less than ± 2 π) is complex and difficult to describe using simple mathematical expression. Thus, there is no general formula giving the set-up time for any step-response transient behavior that unlocks the loop. Nevertheless, this set-up time can be approximated by a simple loop filter capacitor charging time Tsetup in the worst case:

$$\text{Tsetup} \leq \alpha \cdot \left[\frac{C1 + C2}{I_P}\right] \cdot \frac{VDDPLL}{2}$$

where:

C1 and C2 are the loop filter capacitors,

I_p the charge pump current (see “PLL Characteristics” on page 16),

α is a margin factor, set to 3 or 4 as a minimum,

VDDPLL / 2 (approximately equal to 1.6V) is chosen because the PLL's VCO operates linearly.

This formula over estimates the required time, but gives an easy way to approximate this setup time.

- PLL stabilization time will be effective when PLL value is modified.
- During PLL stabilization time, CORECLOCK is stopped.

- **PSTKEY[15:0]: Key for Write Access into the CM_PST Register**

Any write in PSTB[9:0] will be effective only if PSTKEY[15:0] is equal to 0x59C1. These bits are always read at 0.

Note: Write accesses to this register are only valid if PLLEN is at logical 0 (i.e. PLL not enabled).

CM PLL Divider Register

Name: CM_PDIV
Access: Read/Write
Base Address: 0x010

31	30	29	28	27	26	25	24
PDIVKEY[15:8]							

23	22	21	20	19	18	17	16
PDIVKEY[7:0]							

15	14	13	12	11	10	9	8
PLLDIV2	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	PMUL[4:0]				

- **PMUL[4:0]: PLL Multiplier**

These bits select the PLL multiplier.

PMUL[4:0]	PLL multiplier
0	Remains in previous state
1	Remains in previous state
2	2
3	3
…	…
19	19
20	20
21 to 31	Remains in previous state

- **PLLDIV2: PLL Divider**

0: Selects PLLCLK clock (deselects PLLCLK/2)

1: Selects PLLCLK/2 clock (deselects PLLCLK)

- **PDIVKEY[15:0]: Key for Write Access into the CM_PDIV Register**

Any write in the PMUL[4:0] and PLLDIV2 bits will only be effective if the PDIVKEY[15:0] is equal to 0x762D. These bits are always read at 0.

The output frequency of the PLL is equal to: MCK x PMUL[4:0], where MCK is the PLL input clock.

Note: Write accesses to this register are only valid if PLLEN is at logical 0 (i.e. PLL disabled).

CM Oscillator Stabilization Timer Register

Name: CM_OST
Access: Read/Write
Base Address: 0x014

31	30	29	28	27	26	25	24
OSTKEY[15:8]							

23	22	21	20	19	18	17	16
OSTKEY[7:0]							

15	14	13	12	11	10	9	8
–	–	–	–	–	–	OSTB[9:8]	

7	6	5	4	3	2	1	0
OSTB[7:0]							

- **OSTB[9:0]: Oscillator Stabilization Time**

Number of clock cycles needed before master oscillator stabilization. This register provides optimization of the oscillator stabilization time during the stabilization time the clock is halted.

This stabilization time is used when changing from low power mode (Core clock derived from RTCK) to slow mode (PLL not used).

The required time for master oscillator stabilization is 4 ms maximum and is based on the MCK/256 clock.

The default value is 0x000000B0 guarantying 176 clock cycles with MCK/256 clock (i.e. Tsetup = 176x(1/4MHz)*256 = 11.264 ms with MCK = 4.0 MHz)

The oscillator stabilization time can be estimated at $32/F_{osc}$ ms with F_{osc} in MHz (i.e. Tsetup = 32/4 = 8 ms with MCK = 4.0 MHz).

- **OSTKEY[15:0]: Key for Write Access into the CM_OST Register**

Any write in the OSTB[9:0] bits will only be effective if the OSTKEY[15:0] bits are equal to 0xDB5A. These bits are always read at 0.

CM Master Clock Divider Register

Name: CM_MDIV
Access: Read/Write
Base Address: 0x00C

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

- **MDIV[6:0]: Master Clock Divider**

MDIV[6:0] is used to divide the MCK clock and generate the DIVCLK. Default value for MDIV[6:0] is 0x1F.

$$DIVCLK = \frac{MCK}{2 \times (MDIV[6:0] + 1)}$$

- **MDIVKEY[15:0]: Key for Write Access into the CM_MDIV Register**

Any write in the MDIV[6:0] bits will only be effective if the MDIVKEY[15:0] bits are equal to 0xACDC. These bits are always read at 0.

Special Function Mode (SFM)

Overview

The AT91SAM7A2 provides registers which implement the following special functions:

- Chip identification
- Reset status

Chip Identification

Chip identification is done via the Chip ID register (SFM_CIDR). This register gives information on the internal memories used (type and size) and the architecture of the device.

Reset status

The AT91SAM7A2 includes the Reset Status (SFM_RSR) register to give the last cause of reset (i.e. hardware reset or internal watchdog reset).

Special Function Mode (SFM) Memory Map

Base Address: 0xFFF0000

Table 31. SFM Memory Map

Offset	Register	Name	Access	Reset State
0x000	SFM Chip ID	SFM_CIDR	Read-only	0x80000500
Reserved	–	–	–	–
0x008	SFM Reset Status	SFM_RSR	Read-only	0x0000006C or 0x00000053

SFM Chip ID Register

Name: SFM_CIDR
Access: Read-only
Base Address: 0x000

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

23	22	21	20	19	18	17	16
				ARCH[3:0]			

15	14	13	12	11	10	9	8
NVPMT[3:0]				IRS[3:0]			

7	6	5	4	3	2	1	0
NVDMS[3:0]				NVPMS[3:0]			

- **NVPMS[3:0]: Non Volatile Program Memory Size**

0000_b: None.

Other: memory size = $2^{(14+NVPMS[3:0])}$ bytes.

- **NVDMS[3:0]: Non Volatile Data Memory Size**

0000_b: None.

Other: Reserved.

- **IRS[3:0]: Internal RAM Size**

Internal RAM size = $2^{(9+IRS[3:0])}$ bytes.

- **NVPMT[3:0]: Non Volatile Program Memory Type**

0000_b: ROM less.

0001_b: Mask ROM.

Other: reserved.

- **ARCH[3:0]: Core Architecture**

0000_b: ARM7TDMI.

Other: Reserved.

- **EXT: Extension Flag**

0: No extended chip ID.

1: Extended chip ID existing.

SFM Reset Status Register

Name: SFM_RSR
Access: Read-only
Base Address: 0x008

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
RESET[7:0]							

This register gives the last cause of reset.

- **RESET[7:0]: Cause of Reset**

0x6C: External reset on NRESET pin.

0x53: Internal watchdog reset.

Watchdog (WD)

Overview

The watchdog timer is used to prevent the system from locking-up (for example in infinite software loops). If the software does not write to the watchdog during the programmed time, then it can generate an interrupt (WDOVF) or an internal reset.

The watchdog timer has a programmable 16-bit down counter in WD_MR register.

The software can control the action to perform when the WD counter overflows (i.e. reaches 0):

- If the RSTEN bit is set in the WD_OMR register, an internal reset is generated.
- If the WDOVF bit is set in the WD_IMR register, an interrupt is generated on the Generic Interrupt Controller (GIC).

In this case, an “overflow” occurs when the watchdog down-counter reaches zero.

The low frequency clock from the clock manager supplies the watchdog counter (see Figure 41 “Watchdog Block Diagram,” on page 64).

It is possible to set a programmable pending window that provides users with the option to restart the watchdog counter only from within this window. This protection is set with the RSTALW bit, otherwise users can restart the watchdog counter at any time. When the pending windows is reached, the WDPEND bit is set followed by the PENDING bit.

The WDPDIVCLK is then divided by the WDPDIV[2:0] divider and provided to the down-counter input WDCLK.

All write accesses are protected by control access keys to help prevent corruption of the watchdog should an error condition occur.

To update the contents of the mode and control registers it is necessary to write the correct bit pattern to the control access key bits at the same time as the control bits are written (the same write access).

Note: Due to internal synchronization of the restart command (write restart key in the WD_CR register), no further restart commands can be taken in account during 2 WDCLK and ½ LFCLK periods.

Architecture

The WD contains a programmable length down-counter. The count length determines the timeout period, and is controlled by loading the PCV field of the WD_MR register. The time out period (in seconds) is:

$$\frac{(\text{PCV}[15:0]) + 1}{WDCLK_{\text{freq}}}$$

When the counter reaches the value programmed in the pending window PWL[15:0] of WD_PWR register, the watchdog can generate a watchdog pending interrupt. The pending interrupt occurs after:

$$\frac{(\text{PCV}[15:0]) - (\text{PWL}[15:0])}{\text{WDCLK}_{\text{freq}}}\text{seconds}$$

If the previous time is negative, the WD pending interrupt should not be used.

In order to prevent an internal reset (if the RSTEN bit is set in the WD_OMR register) or interrupt (if the WDOVF bit is set in the WD_IMR), the software must reset the counter before it reaches 0 by writing the correct key in the WD_CR register (0xC071). The time (in seconds) between the WD pending interrupt and the WD overflow is:

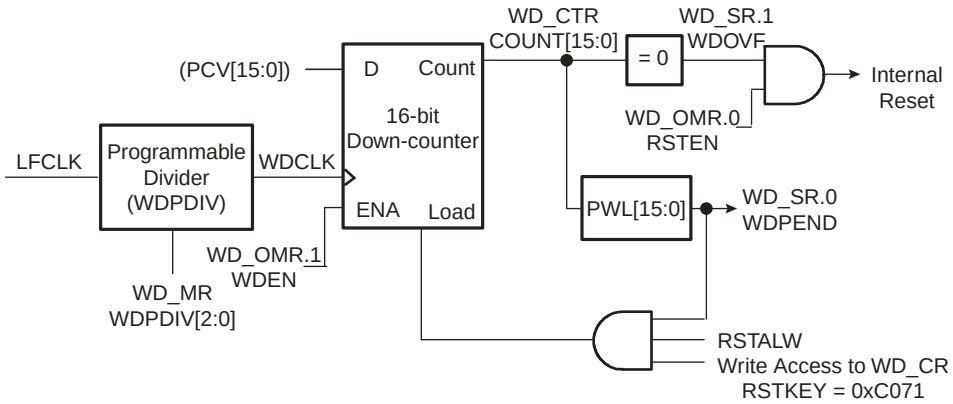
$$\frac{(PWL[15:0])}{WDCLK_{freq}}$$

When the counter reaches 0, it triggers the programmed action (internal reset or interrupt).

If no WD reset is programmed (i.e. RSTEN is at a logical 0) when the WD reaches 0, it is reset to the programmed value and continues to count, unless it is disabled. This enables it to be used to generate periodic interrupts.

Block Diagram

Figure 41. Watchdog Block Diagram



Internal Reset Pulse Generation

If the RSTEN bit is set in the WD_OMR register, an internal reset pulse is generated when the overflow occurs. This pulse is 8 clock cycles long (CORECLK) and is not affected by hardware reset.

After a reset (hardware or WD), the clock selected by the watchdog is LFCLK/2.

Internal Interrupt Request

The watchdog can generate an internal interrupt request when an overflow occurs. The software can enable or disable this interrupt either in the WD module (WD_IMR register) or in the GIC registers.

Example

The LFCLK clock source from the clock manager is used to select the WDSCLK.

The following example uses LFCLK as a value of 32.768 kHz and a pending window of 0xFF.

Table 32. Watch Dog Example

WD Counter Start Value	WDPDIV[2:0]	WDCLK	Time to WD Pending	Time to WD Overflow
0x07FF	000_b	LFCLK/2	0.109 s	0.125 s
0x0FFF	001_b	LFCLK/4	0.468 s	0.500 s
0x17FF	010_b	LFCLK/8	1.437 s	1.500 s
0xAFFF	101_b	LFCLK/128	175 s	176 s
0xFFFF	111_b	LFCLK/1024	1040 s	2048 s

Example use of the Watchdog

Use of the Pending Window to generate an interrupt and reload the watchdog counter within the window only.

If the interrupt is not considered due to a bug, a reset occurs when the watchdog counter reaches 0.

Configuration

- Configuration of WD_MR: Choice of the clock to decrement counter, preload value from which the counter starts to decrement.
- Configuration of WD_ PWR: Upper limit of the window from which it generates an interrupt when reached and the bit which restarts the counter only within this window.
- Configuration of WD_IER: Enable Interrupt at the peripheral level when the window is reached (WDPEND bit) or when the counter overflow (WDOVF bit if watchdog reset is not enabled), GIC must be configured.
- Configuration of WD_OMR: Enable the watchdog (start decrementing the counter) and enable the watchdog reset in case of counter overflow.

Interrupt Handling

- IRQ Entry and call C function.
- Read WD_SR and verify the source of the interrupt.
- Clear the corresponding interrupt at peripheral level by writing in the WD_CSR.
- Interrupt treatment. If this is a pending window interrupt, restart the watchdog by writing in WD_CR.
- IRQ Exit.

Watchdog (WD) Memory Map

Base Address: 0xFFA0000

Table 33. Watchdog Memory Map

Offset	Register	Name	Access	Reset State
0x000 – 0x05C	Reserved	–	–	–
0x060	Control Register	WD_CR	Write-only	–
0x064	Mode Register	WD_MR	Read/Write	0x0007FF00
0x068	Overflow Mode Register	WD_OMR	Read/Write	0x00000000
0x06C	Clear Status Register	WD_CSR	Write-only	–
0x070	Status Register	WD_SR	Read-only	0x00000000
0x074	Interrupt Enable Register	WD_IER	Write-only	–
0x078	Interrupt Disable Register	WD_IDR	Write-only	–
0x07C	Interrupt Mask Register	WD_IMR	Read-only	0x00000000
0x080	Pending Window Register	WD_PWR	Read-only	0x00000000

WD Control Register

Name: WD_CR
Access: Write-only
Base Address: 0x060

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

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

15	14	13	12	11	10	9	8
RSTKEY[15:8]							

7	6	5	4	3	2	1	0
RSTKEY[7:0]							

- **RSTKEY[15:0]: Restart Key**

0xC071: Watchdog counter is restarted if its value is equal or less than the pending window length or if the pending window is disabled.

Other value: No effect.

Note: A restart command (write the restart key in WD_CR register) will not be effective if it occurs less than 2 WDCLK and ½ LFCLK periods after a previous start command.

WD Mode Register

Name: WD_MR
Access: Read/Write
Base Address: 0x064

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

- **WDPDIV[2:0]: WD Clock Divider**

WDPDIV[2:0]			WDCLK
0	0	0	LFCLK/2
0	0	1	LFCLK /4
0	1	0	LFCLK /8
0	1	1	LFCLK /16
1	0	0	LFCLK /32
1	0	1	LFCLK /128
1	1	0	LFCLK /256
1	1	1	LFCLK /1024

- **PCV[15:0]: Preload Counter Value**

Counter is preloaded when watchdog counter is restarted.

- **CKEY[7:0]: Clock Access Key**

Used only when writing in WD_MR. CKEY is read as 0.

Write access in WD_MR is allowed only if CKEY[7:0] = 0x37.

WD Overflow Mode Register

Name: WD_OMR
Access: Read/Write
Base Address: 0x068

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

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

15	14	13	12	11	10	9	8
OKEY[11:4]							

7	6	5	4	3	2	1	0
OKEY[3:0]				–	–	TSTEN	WDEN

- **WDEN: Watchdog Enable**

0: Watchdog is disabled.

1: Watchdog is enabled.

- **RSTEN: Reset Enable**

0: Generation of an internal reset by the Watchdog is disabled.

1: When overflow occurs, the Watchdog generates an internal reset.

- **OKEY[11:0]: Overflow Access Key**

Used only when writing WD_OMR. OKEY is read as 0.

0x234: Write access in WD_OMR is allowed.

Other value: Write access in WD_OMR is prohibited.

WD Clear Status Register

Name: WD_CSR
Access: Write-only
Base Address: 0x06C

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
–	–	–	–	–	–	WDOVF	WDPEND

- **WDPEND: Watchdog Pending Clear**

0: No effect.

1: Clear Watchdog pending interrupt.

- **WDOVF: Watchdog Overflow Clear**

0: No effect.

1: Clear Watchdog overflow interrupt.

WD Status Register

Name: WD_SR
Access: Read-only
Base Address: 0x070

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	RESTART	PENDING

7	6	5	4	3	2	1	0
–	–	–	–	–	–	WDOVF	WDPEND

- **WDPEND: Watchdog Pending**

0: No Watchdog pending.

1: A Watchdog pending has occurred.

- **WDOVF: Watchdog Overflow**

0: No Watchdog overflow.

1: A Watchdog overflow has occurred.

- **PENDING: Watchdog Pending Status**

0: Watchdog counter is over pending window length.

1: Watchdog counter is equal or less than pending window length.

- **RESTART: Watchdog Restart Status**

0: Watchdog available for new restart.

1: Watchdog restart executing.

WD Interrupt Enable Register

Name: WD_IER
Access: Write-only
Base Address: 0x074

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
–	–	–	–	–	–	WDOVF	WDPEND

WD Interrupt Disable Register

Name: WD_IDR
Access: Write-only
Base Address: 0x078

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
–	–	–	–	–	–	WDOVF	WDPEND

WD Interrupt Mask Register

Name: WD_IMR
Access: Read-only
Base Address: 0x07C

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
–	–	–	–	–	–	WDOVF	WDPEND

- **WDPEND: Watchdog Pending Interrupt Mask**

0: The WDPEND interrupt is disabled.

1: The WDPEND interrupt is enabled.

- **WDOVF: Watchdog Overflow Interrupt Mask**

0: The WDOVF interrupt is disabled.

1: The WDOVF interrupt is enabled.

WD Pending Window Register

Name: WD_PWR
Access: Read/Write
Base Address: 0x080

31	30	29	28	27	26	25	24
PWKEY[7:0]							

23	22	21	20	19	18	17	16
PWL[158]							

15	14	13	12	11	10	9	8
PWL[7:0]							

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	RSTALW

- **RSTALW: Restart Allowed**

0: restart allowed every time.

1: restart allowed only within Pending Window.

This bit does not disable the bit WDPEND in the interrupt register.

- **PWL[15:0]: Pending Window Length**

Length of the window.

- **PWKEY[7:0]: Pending Window Access Key**

Used only when writing in WD_PWR. PWKEY is read as 0.

Write access in WD_PWR is allowed only if PWKEY[7:0] = 0x91.

Watch Timer (WT)

Overview

The Watch Timer provides a seconds counter and an alarm function.

Seconds Counter

The seconds counter is a 32-bit counter that indicates the number of low frequency clock (LFCLK) pulses elapsed since the last time it was reset to zero.

The seconds counter is incremented every 30.518 µs (one period of 32 KHz clock).

The counter is reset to 0x00000000 when the counter reaches 0xA8C00000 (86400 seconds or 24 hours) or 0xFFFFFFFF (it is configurable on the Mode Register).

A write access can only be performed when the seconds counter is disabled because of an asynchronous interface (see “Asynchronous Interface” below).

Alarm

The alarm register has the same resolution as the seconds counter. This enables a 32-bit register to have sufficient range to cater for a 24 hour period.

An interrupt is generated at the end of the period at which the value in the seconds register equals the value in the alarm register.

A write access can only be performed when the alarm counter is disabled because of an asynchronous interface. An invalid data (i.e. value greater or equal to 0xA8C00000) will not be written into the alarm register in 24 hours mode.

Asynchronous Interface

As the seconds counter is an asynchronous counter (can use the RTCK clock), some precautions must be taken with it.

When enabling or disabling the alarm or seconds counter, software must wait for an enabled or disabled interrupt to be sure that the alarm or the counter is really enabled or disabled.

CAN Time Stamp

The 32-bit register forming the seconds counter is provided to the CAN module. After each transmission or reception of a CAN frame, the value of the current seconds counter will be automatically written in the corresponding CAN channel CAN_STPx register.

Example

An example use of the Watch Timer: Use of the Watch Timer to generate an interrupt after 24 hours. The low frequency clock should be 32 KHz.

Configuration

- Do a software reset of the watch timer to be in a known state by writing SWRST bit in WT_CR and wait about four LFCLK periods for the circuitry to be stabilized.
- Configuration of WT_ MR: Selects the 24 hour mode by writing the SECRST bit. The seconds counter will increment and reset when equal to the value of 0xA8BFFFFF.
- Configuration of WT_ ALARM: When the seconds counter is equal to this value, an interrupt can be generated. For 24 hours at 32 KHz, the Alarm value should be 0xA8BFFFFF.
- Configuration of WT_SECS: Starting value from which the seconds counter will start to increment. Should be left to 0.
- Configuration of WT_IER: The ALARM bit enables an interrupt at the peripheral level when the seconds counter is equal to the programmed value in WT_ALARM. Other interrupts can be activated to indicate when the seconds counter or alarm functionality are really enabled or disabled, as the watch timer is clocked on the LFCLK. GIC must be configured.

- Configuration of WT_CR: Starts the seconds counter and enables the alarm (ALARMEN and SECSEN bits). The counter will start to increment when the SECSEN bit is set in WT_SR. The same is true for the alarm functionality bit, ALARMEN. An interrupt can be programmed with these events.

Interrupt Handling

- IRQ Entry and call C function.
- Read WT_SR and verify the source of the interrupt.
- Clear the corresponding interrupt at the peripheral level by writing in WD_CSR.
- Interrupt treatment: The seconds counter will restart automatically, counting from 0 when it reaches 0xA8BFFFFF (WT_MR). If the user wants to set a lower value in WT_ALARM and desires to restart the seconds counter; the seconds counter must first be disabled, set to 0 and once again enabled.
- IRQ Exit.

Watch Timer (WT) Memory Map

Base Address: 0xFFA4000

Table 34. Watch Timer Memory Map

Offset	Register	Name	Access	Reset State
0x000 – 0x05C	Reserved	–	–	–
0x060	Control Register	WT_CR	Write-only	–
0x064	Mode Register	WT_MR	Read/Write	0x00000000
0x068	Reserved	–	–	–
0x06C	Clear Status Register	WT_CSR	Write-only	–
0x070	Status Register	WT_SR	Read-only	0x00000000
0x074	Interrupt Enable Register	WT_IER	Write-only	–
0x078	Interrupt Disable Register	WT_IDR	Write-only	–
0x07C	Interrupt Mask Register	WT_IMR	Read-only	0x00000000
0x080	Seconds Register	WT_SECR	Read/Write	0x00000000
0x084	Alarm Register	WT_ALR	Read/Write	0x00000000

WT Control Register

Name: WT_CR
Access: Write-only
Base Address: 0x060

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
–	–	–	ALARMDIS	ALARMEN	SECSDIS	SECSEN	SWRST

- **SWRST: WT Software Reset**

0: No effect.

1: Reset the WT.

A Software triggered hardware reset of the WT is performed. It resets all the registers. The software must wait for LFCLK to set up properly before using other registers.

- **SECSEN: WT Seconds Counter Enable**

0: No effect.

1: Enables the WT seconds counter.

- **SECSDIS: WT Seconds Counter Disable**

0: No effect.

1: Disables the WT seconds counter.

In case both SECSEN and SECSDIS are equal to one when the control register is written, the WT seconds counter will be disabled.

- **ALARMEN: WT Alarm Enable**

0: No effect.

1: Enables the WT alarm.

- **ALARMDIS: WT Alarm Disable**

0: No effect.

1: Disables the WT alarm.

In case both ALARMEN and ALARMDIS are equal to one when the control register is written, the WT alarm will be disabled

WT Mode Register

Name: WT_MR
Access: Read/Write
Base Address: 0x064

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
–	–	–	–	–	–	–	SECRST

- **SECRST: Second Reset**

0: The seconds counter is reset to 0x00000000 at the end of the period when it reaches 0xA8BFFFFF.

1: The seconds counter is reset to 0x00000000 at the end of the period when it reaches 0xFFFFFFFF.

WT Clear Status Register

Name: WT_CSR
Access: Write-only
Base Address: 0x06C

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
–	–	–	ALARMDIS	ALARMEN	SECSDIS	SECSEN	ALARM
							W

- **ALARM: Clear Alarm Interrupt**

0: No effect.

1: Clear ALARM interrupt.

- **SECSEN: Clear Seconds Counter Enabled**

0: No effect.

1: Clear the seconds counter enabled interrupt.

- **SECSDIS: Clear Seconds Counter Disabled**

0: No effect.

1: Clear the seconds counter disabled interrupt.

- **ALARMEN: Clear Alarm Enabled**

0: No effect.

1: Clear the alarm enabled interrupt.

- **ALARMDIS: Clear Alarm Disabled**

0: No effect.

1: Clear the alarm disabled interrupt.

WT Status Register

Name: WT_SR
Access: Read-only
Base Address: 0x070

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	ALARMENS	SECENS

7	6	5	4	3	2	1	0
–	–	WSEC	ALARMDIS	ALARMEN	SECSDIS	SECSEN	ALARM

- **ALARM: Alarm Interrupt**

0: No alarm occurred.

1: An alarm occurred since last clear of the status register.

- **SECSEN: Seconds Counter Enabled Interrupt**

0: No seconds counter enabled interrupt.

1: A seconds counter enabled interrupt occurred since last clear of the status register.

- **SECSDIS: Seconds Counter Disabled Interrupt**

0: No seconds counter disabled interrupt.

1: A seconds counter disabled interrupt occurred since last clear of the status register.

- **ALARMEN: Alarm Enabled Interrupt**

0: No alarm enabled interrupt.

1: An alarm enabled interrupt occurred since last clear of the status register.

- **ALARMDIS: Alarm Disabled Interrupt**

0: No alarm disabled interrupt.

1: An alarm disabled interrupt occurred since last clear of the status register.

- **WSEC: Write Second**

0: No effect.

1: A write is occurring on the seconds counter register.

- **SECSENS: Seconds Counter Enable Status**

0: Seconds counter is disabled.

1: Seconds counter is enabled.

- **ALARMENS: Alarm Enable Status**

0: Alarm is disabled.

1: Alarm is enabled.

WT Interrupt Enable Register

Name: WT_IER
Access: Write-only
Base Address: 0x074

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
–	–	–	ALARMDIS	ALARMEN	SECSDIS	SECSEN	ALARM

WT Interrupt Disable Register

Name: WT_IMR
Access: Write-only
Base Address: 0x078

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
–	–	–	ALARMDIS	ALARMEN	SECSDIS	SECSEN	ALARM

WT Interrupt Mask Register

Name: WT_IMR
Access: Read-only
Base Address: 0x07C

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
–	–	–	ALARMDIS	ALARMEN	SECSDIS	SECSEN	ALARM

- **ALARM: Alarm Interrupt Mask**

0: ALARM interrupt is disabled.

1: ALARM interrupt is enabled.

- **SECSEN: Seconds Counter Enabled Interrupt Mask**

0: SECSEN interrupt is disabled.

1: Seconds counter enabled interrupt is enabled.

- **SECSDIS: Seconds Counter Disabled Interrupt Mask**

0: SECSDIS interrupt is disabled.

1: SECSDIS interrupt is enabled.

- **ALARMEN: Alarm Enabled Interrupt Mask**

0: ALARMEN interrupt is disabled.

1: ALARMEN interrupt is enabled.

- **ALARMDIS: Alarm Disabled Interrupt Mask**

0: ALARMDIS interrupt is disabled.

1: ALARMDIS interrupt is enabled.

WT Seconds Register

Name: WT_SECR
Access: Read/Write
Base Address: 0x080

31	30	29	28	27	26	25	24
SECONDS[31:24]							

23	22	21	20	19	18	17	16
SECONDS[23:16]							

15	14	13	12	11	10	9	8
SECONDS[15:8]							

7	6	5	4	3	2	1	0
SECONDS[7:0]							

- **SECONDS[31:0]: Seconds Register**

Number of LFCLK clock cycle periods elapsed since last reset to zero.

This register can only be written when SECSENS = 0.

An invalid data (i.e. value greater or equal to 0xA8C00000) will not be written into the seconds register if in 24 hours mode.

WT Alarm Register

Name: WT_ALR
Access: Read/Write
Base Address: 0x084

31	30	29	28	27	26	25	24
ALARMREG[31:24]							

23	22	21	20	19	18	17	16
ALARMREG[23:16]							

15	14	13	12	11	10	9	8
ALARMREG[15:8]							

7	6	5	4	3	2	1	0
ALARMREG[7:0]							

- **ALARMREG[31:0]: Alarm Register**

An interrupt can be generated when the seconds register reaches this value.

This register can only be written when ALARMENS = 0.

An invalid data (i.e. value greater or equal to 0xA8C00000) will not be written into the alarm register if in 24 hours mode.

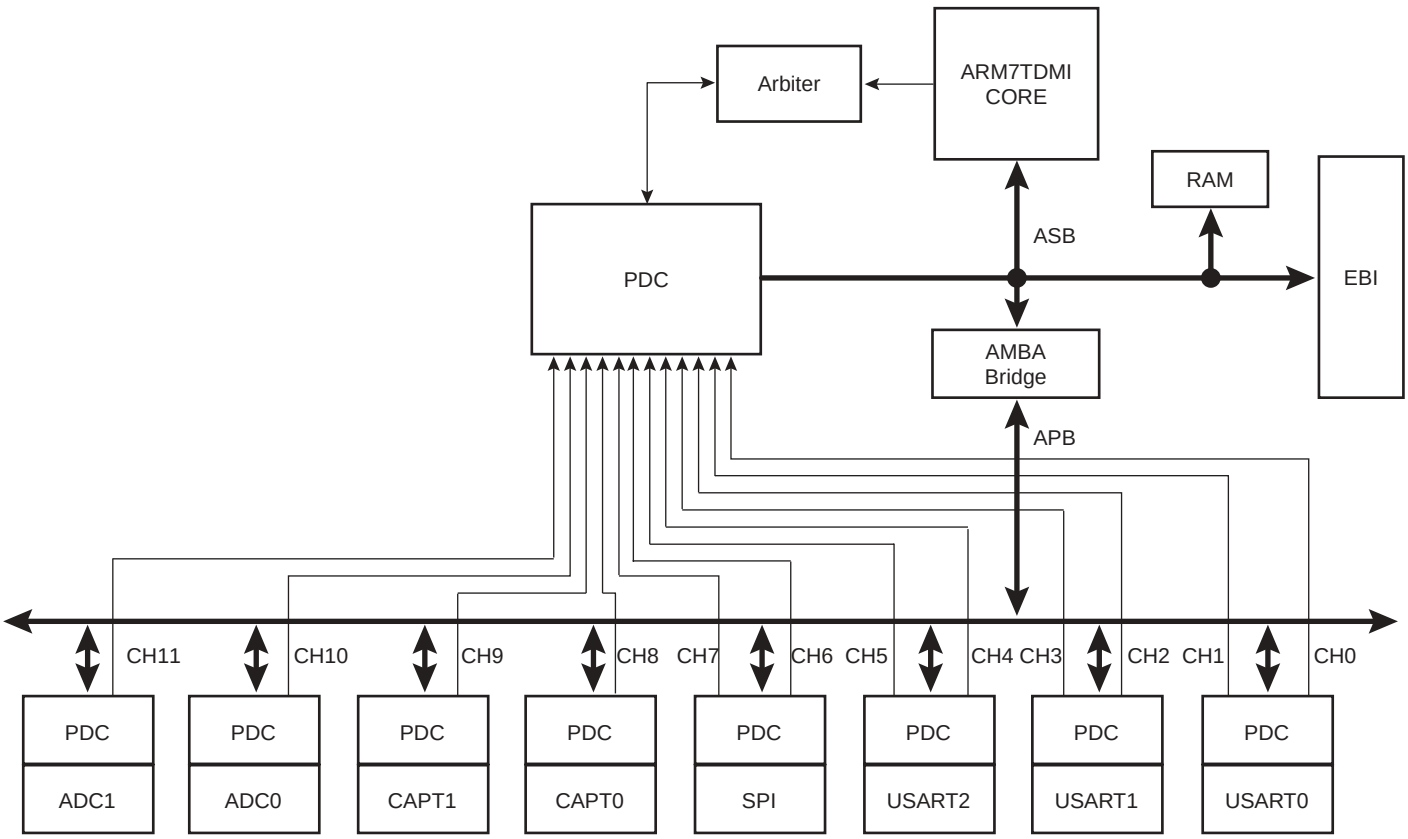
Peripheral Data Controller (PDC)

Overview

The Peripheral Data Controller (PDC) permits easy and quick transfers of large blocks of words from memory to peripheral or from peripheral to memory.

Block Diagram

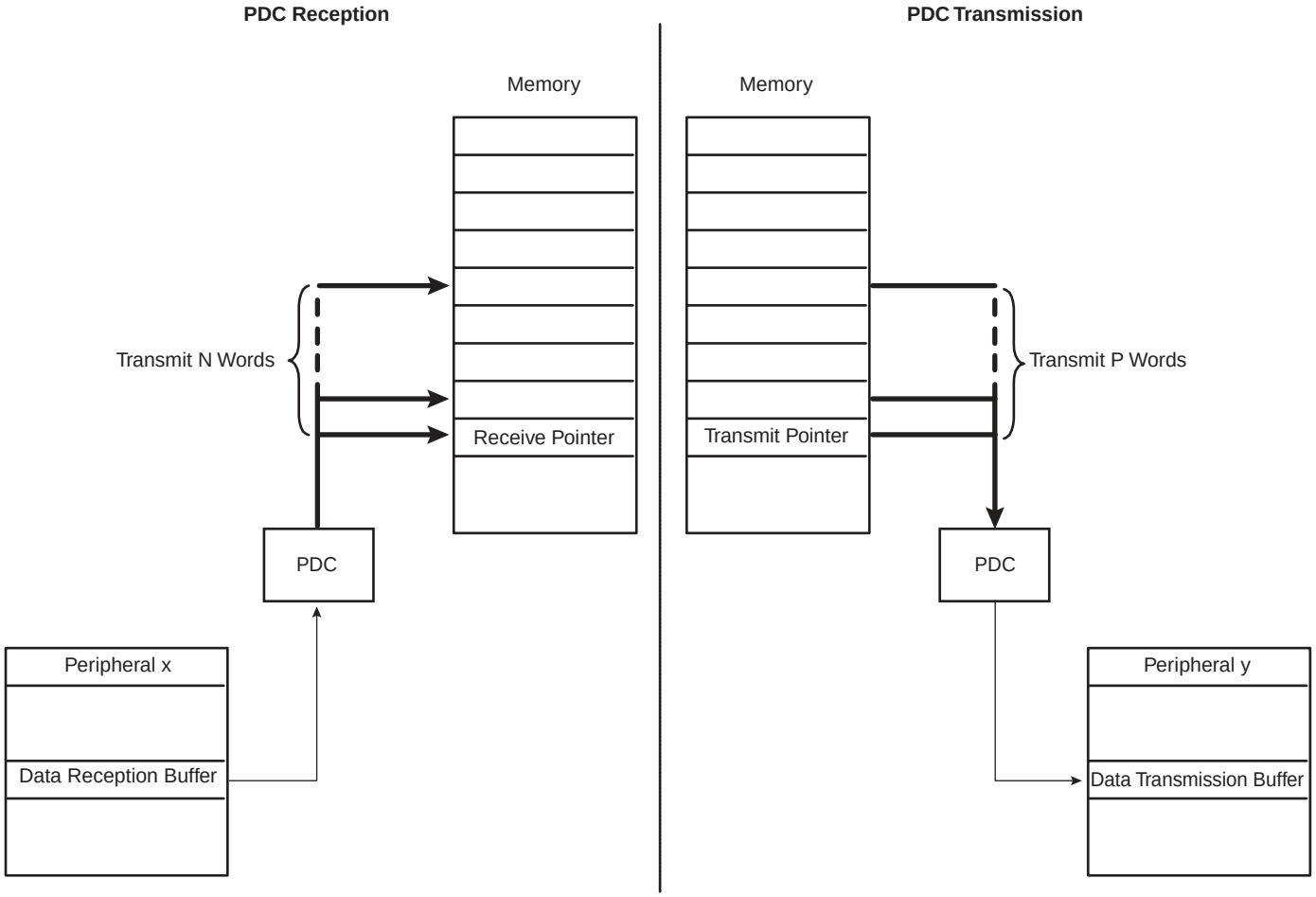
Figure 42. PDC Block Diagram



Each PDC channel, with a start address and a counter (number of words), can automatically put/get a block of transmitted words in/from a specific memory area. Each peripheral that transfers data can have a channel corresponding to a Peripheral Data Controller channel.

The peripherals associated to the PDC channels are listed in Table 35, “PDC Connection,” on page 86.

Figure 43. PDC Functional Diagram



The PDC has data transfer ports which connect to the ASB and the AMBA Bridge. It is programmed via the APB. The ASB bus request and grant signals are used to request bus access, and to detect when that access has been granted according to the standard AMBA bus arbitration scheme.

Data transfer to a peripheral is made directly via the APB (AMBA Bridge, to avoid tying up the ASB). A simple arbitration scheme is implemented between the ASB and PDC, to control APB access.

Before programming any PDC transfer, the PDC module must be enabled. This is done by setting the PDC bit to a logical 1 in the PMC_ECR register (see “Power Management Controller (PMC)” on page 245.

The PDC channel is programmed using PDC_CRx (Control Register x, x = 0 to 9), PDC_MPRx (Memory Pointer x) and PDC_TCRx (Transfer Counter x).

The status of the PDC transfer is given in the Status Register of the associated peripheral.

The pointer registers (PDC_MPRx) are used to store the address of the buffer.

The counter registers (PDC_TCRx) are used to store the size of these buffers (i.e. the number of data to be transferred).

When a transfer is performed, the counter is decremented and the pointer is incremented. When the counter reaches 0 (i.e. all the data have been sent/received to/from the module), the end status bit is set in the peripheral status register and can be programmed to generate an interrupt.

PDC Transfers

PDC transfers consist of byte (8-bit), half-word (16-bit) or word (32-bit) data transmitted from peripheral to memory or from memory to peripheral.

Transfers are triggered by the peripheral signals.

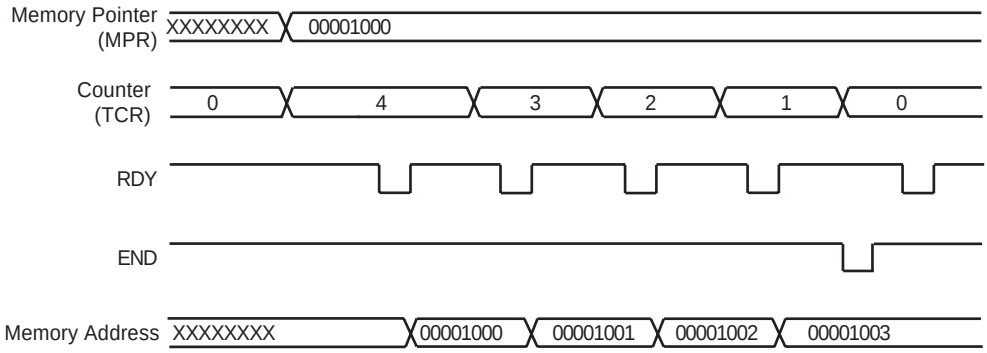
The PDC makes block transfers one byte, one half-word or one word at a time (programmed in the PDC_CRx register, with direction). Each transfer is triggered by a PDC request from the peripheral. The PDC releases the AMBA bus after each transfer. A new trigger is needed for each transfer.

Between transfers, the ASB memory pointer is incremented and the transfer counter is decremented.

Each block of data can be programmed to be up to 64 Kbytes.

Transfers stop when the transfer counter reaches zero, and the trigger is disabled.

Figure 44. Transfer Example (Byte)



Memory Pointers

There is one 32-bit memory address pointer for each channel (PDC_MPRx). Each memory pointer points to a location in the AT91SAM7A2 memory space (on chip RAM or external memory on the EBI).

The PDC_MPRx is automatically incremented by 1, 2 or 4 after each transfer, for byte, half-word or word transfers. The PDC_MPRx must be initialized before any transfers are started.

If PDC_MPRx is reprogrammed while the PDC is operating, the address of the transfers will be changed. The PDC will continue to perform transfers when triggered, from the newly programmed address.

Transfer Counter

There is one 16-bit Transfer Counter (PDC_TCRx) for each channel, which is used to count the size of the block of transfers. The PDC_TCRx is decremented after every data transfer. When the PDC_TCRx reaches zero the block transfer is complete, the PDC stops transferring data and disables the trigger.

It is not possible to trigger a block of transfers if the PDC_TCRx value is zero. When the PDC_TCRx is programmed to a non-zero value, transfers can be triggered by the peripheral for that particular channel.

The number of transfers required is programmed in the PDC_TCRx which is memory mapped as a 16-bit Read/Write register. The number of transfers remaining can be read in the PDC_TCRx register.

If the PDC_TCRx is reprogrammed while the PDC is operating, the number of transfers will be changed. The PDC will continue to count transfers when triggered, from the newly programmed value.

The end of transfer is signaled to the peripheral via the PDC_END signal. The PDC does not have any dedicated status registers.

PDC Configuration

For emulation purposes, each PDC channel can be software configured to be attached to a different peripheral.

In the AT91SAM7A2 microcontroller, each PDC channel is attached to a dedicated peripheral (with a fixed direction and fixed address). Software must configure each PDC channel so that accesses are correctly done by the PDC module.

Table 35. PDC Connection

Peripheral	PDC Channel	Transfer Direction	DIR Bit in PDC_CRx	Associated Peripheral Register	Associated Peripheral Address
USART0	RX: Ch0	Reception	0	USART0_RHR	0xFFFA8080
	TX: Ch1	Transmission	1	USART0_THR	0xFFFA8084
USART1	RX: Ch2	Reception	0	USART1_RHR	0xFFFAC080
	TX: Ch3	Transmission	1	USART1_THR	0xFFFAC084
ADC0 (8-channel 10-bit)	Ch4	Reception	0	ADC0_DR	0xFFFB0080
ADC1 (8-channel 10-bit)	Ch5	Reception	0	ADC1_DR	0xFFFB0084
SPI	RX: Ch6	Reception	0	SPI_RDR	0xFFFB4080
	TX: Ch7	Transmission	1	SPI_TDR	0xFFFB4084

The end of transmission or reception for each PDC channel transfer is indicated in the status register of the attached peripheral. The PDC_TCRx is decremented with the peripheral trigger when a word has been transferred either from memory to peripheral or from peripheral to memory.

Table 36. PDC Transfer Status

Peripheral	PDC Channel	Transfer Direction	Associated Peripheral Status Register	End of Transfer Bit in Status Register	Status Bit for PDC_TCRx Decrement (Trigger)
USART0	RX: Ch0	Reception	USART0_SR	ENDRX	RXRDY
	TX: Ch1	Transmission	USART0_SR	ENDTX	TXRDY
USART1	RX: Ch2	Reception	USART1_SR	ENDRX	RXRDY
	TX: Ch3	Transmission	USART1_SR	ENDTX	TXRDY
ADC0 (8-channel 10-bit)	Ch4	Reception	ADC0_SR	TEND	EOC
ADC1 (8-channel 10-bit)	Ch5	Reception	ADC1_SR	TEND	EOC

Table 36. PDC Transfer Status (Continued)

Peripheral	PDC Channel	Transfer Direction	Associated Peripheral Status Register	End of Transfer Bit in Status Register	Status Bit for PDC_TCRx Decrement (Trigger)
SPI	RX: Ch6	Reception	SPI_SR	REND	RDRF
	TX: Ch7	Transmission	SPI_SR	TEND	TDRE
Capture CAPT0	Ch8	Reception	CAPT0_SR	PDCEND	DATACAPT
Capture CAPT1	Ch9	Reception	CAPT1_SR	PDCEND	DATACAPT

Configuration Steps

- Enable PDC clock by writing the PDC bit in PMC_PMSR of the PMC peripheral
- Configuration of PDC_PRA: Address of targeted register TX or RX
- Configuration of PDC_CR: Flux direction and element size (8-bit, 16-bit, 32-bit)
- Configuration of PDC_MR: Address of a memory space to receive or transmit data
- Configuration of PDC_TC: Number of transmissions or receptions to do and start PDC

PDC Transfer Example

Transmission on SPI

Assuming the following:

- SPI bits per transfer = 10 on NPCS0 (i.e. BITS[3:0] = 0010b in SPI_CRS0)
- Number of 10-bit words to transfer: 15
- Address of buffer in internal RAM for 10-bit words to be transmitted 0x00000100 (first 10-bit word is at address 0x00000100, second 10-bit word is at address 0x00000102, …)
- SPI clock is enabled (SPI = 1 in SPI_PMSR)
- SPI is enabled (SPIENS = 1 in SPI_SR)

PDC channel 7 must be configured as follows:

- PDC_PRA7 = 0xFFFB4084 (i.e. address of the SPI_TDR register)
- PDC_CR7 = 0x00000003 (i.e. 10-bit words cater in 16-bit words so PDC transfer size is a half-word incrementing the address pointer by 2 after each transfer, transfers are done from memory to peripheral so DIR = 1)
- PDC_MPR7 = 0x00000100 (address of buffer)
- PDC_TCR7 = 0x0000000F (number of 10-bit words to transfer)

As soon as the software writes the number of bytes to transfer in the PDC_TCR7 register, the PDC starts transmitting the 15 10-bit words.

When all the 10-bit words have been transferred to the SPI_TDR register, the TEND bit in the SPI_SR register will be set to a logical 1 informing the software that the transfer is completed. The TEND bit in the SPI_SR register can also generate an interrupt if the corresponding bit is set in the SPI_IMR register.

Note: If a module is used in reception and transmission, the reception channel must be configured before the transmission channel.

Reception on SPI

Assuming the following:

- SPI bits per transfer = 8 on NPCS0 (i.e. BITS[3:0] = 0000b in SPI_CRS0)
- Number of 8-bit words to transfer: 69

- Address of buffer in external RAM for 8-bit words to be transmitted 0x48000000 (first 8-bit word is at address 0x48000000, second 8-bit word is at address 0x48000001, …)
- ·SPI clock is enabled (SPI = 1 in SPI_PMSR)
- SPI is enabled (SPIENS = 1 in SPI_SR)

PDC channel 6 must be configured as follows:

- PDC_PRA6 = 0xFFFB4084 (i.e. address of the SPI_TDR register)
- PDC_CR6 = 0x00000000 (i.e. 8-bit words cater in 8-bit words so PDC transfer size is a byte incrementing the address pointer by 1 after each transfer, transfers are done from peripheral to memory so DIR = 0)
- PDC_MPR6 = 0x48000000 (address of buffer in external RAM)
- PDC_TCR6 = 0x00000045 (number of 8-bit words to transfer)

As soon as the software writes the number of bytes to transfer in the PDC_TCR6 register, the PDC starts receiving the 69 8-bit words.

When all the 8-bit words have been received (i.e. all bytes have been written in external RAM), the REND bit in the SPI_SR register will be set to a logical 1 informing the software that the transfer is completed. The REND bit in the SPI_SR register can also generate an interrupt if the corresponding bit is set in the SPI_IMR register.

Peripheral Data Controller (PDC) Memory Map

Base Address: 0xFFFF8000

Table 37. PDC Memory Map

Offset	Register	Name	Access	Reset State
0x000 – 0x07C	Reserved	–	–	–
0x080	CH0 Peripheral Register Address	PDC_PRA0	Read/Write	0xFFE00000
0x084	CH0 Control Register	PDC_CR0	Read/Write	0x00000000
0x088	CH0 Memory Pointer	PDC_MPR0	Read/Write	0x00000000
0x08C	CH0 Transfer Counter	PDC_TCR0	Read/Write	0x00000000
0x090	CH1 Peripheral Register Address	PDC_PRA1	Read/Write	0xFFE00000
0x094	CH1 Control Register	PDC_CR1	Read/Write	0x00000000
0x098	CH1 Memory Pointer	PDC_MPR1	Read/Write	0x00000000
0x09C	CH1 Transfer Counter	PDC_TCR1	Read/Write	0x00000000
0x0A0	CH2 Peripheral Register Address	PDC_PRA2	Read/Write	0xFFE00000
0x0A4	CH2 Control Register	PDC_CR2	Read/Write	0x00000000
0x0A8	CH2 Memory Pointer	PDC_MPR2	Read/Write	0x00000000
0x0AC	CH2 Transfer Counter	PDC_TCR2	Read/Write	0x00000000
0x0B0	CH3 Peripheral Register Address	PDC_PRA3	Read/Write	0xFFE00000
0x0B4	CH3 Control Register	PDC_CR3	Read/Write	0x00000000
0x0B8	CH3 Memory Pointer	PDC_MPR3	Read/Write	0x00000000
0x0BC	CH3 Transfer Counter	PDC_TCR3	Read/Write	0x00000000
0x0C0	CH4 Peripheral Register Address	PDC_PRA4	Read/Write	0xFFE00000
0x0C4	CH4 Control Register	PDC_CR4	Read/Write	0x00000000
0x0C8	CH4 Memory Pointer	PDC_MPR4	Read/Write	0x00000000
0x0CC	CH4 Transfer Counter	PDC_TCR4	Read/Write	0x00000000
0x0D0	CH5 Peripheral Register Address	PDC_PRA5	Read/Write	0xFFE00000
0x0D4	CH5 Control Register	PDC_CR5	Read/Write	0x00000000
0x0D8	CH5 Memory Pointer	PDC_MPR5	Read/Write	0x00000000
0x0DC	CH5 Transfer Counter	PDC_TCR5	Read/Write	0x00000000
0x0E0	CH6 Peripheral Register Address	PDC_PRA6	Read/Write	0xFFE00000
0x0E4	CH6 Control Register	PDC_CR6	Read/Write	0x00000000
0x0E8	CH6 Memory Pointer	PDC_MPR6	Read/Write	0x00000000
0x0EC	CH6 Transfer Counter	PDC_TCR6	Read/Write	0x00000000
0x0F0	CH7 Peripheral Register Address	PDC_PRA7	Read/Write	0xFFE00000
0x0F4	CH7 Control Register	PDC_CR7	Read/Write	0x00000000
0x0F8	CH7 Memory Pointer	PDC_MPR7	Read/Write	0x00000000
0x0FC	CH7 Transfer Counter	PDC_TCR7	Read/Write	0x00000000
0x100	CH8 Peripheral Register Address	PDC_PRA8	Read/Write	0xFFE00000
0x104	CH8 Control Register	PDC_CR8	Read/Write	0x00000000
0x108	CH8 Memory Pointer	PDC_MPR8	Read/Write	0x00000000
0x10C	CH8 Transfer Counter	PDC_TCR8	Read/Write	0x00000000
0x110	CH9 Peripheral Register Address	PDC_PRA9	Read/Write	0xFFE00000
0x114	CH9 Control Register	PDC_CR9	Read/Write	0x00000000
0x118	CH9 Memory Pointer	PDC_MPR9	Read/Write	0x00000000
0x11C	CH9 Transfer Counter	PDC_TCR9	Read/Write	0x00000000

PDC CH0...CH9 Peripheral Register Address

Name: PDC_PRA0...PDCPRA9
Access: Read/Write
Base Address: 0xXX0

31	30	29	28	27	26	25	24
CHPRA[31:24]							

23	22	21	20	19	18	17	16
CHPRA[23:16]							

15	14	13	12	11	10	9	8
CHPRA[15:8]							

7	6	5	4	3	2	1	0
CHPRA[7:0]							

- **CHPRA[31:0] Peripheral Register Address**

CHPRA[31:0] must be loaded with the address of the target register (peripheral receive or transmit register).

PDC CH0...CH9 Control Register

Name: PDC_CR0...PDC_CR9
Access: Read/Write
Base Address: 0xXX4

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
–	–	–	–	–	SIZE[1:0]		DIR

- **DIR: Transfer direction**

0: Peripheral to memory.

1: Memory to peripheral.

- **SIZE[1:0]: Transfer size**

Defines the size of the transfer.

SIZE[1:0]		Transfer Size
0	0	Byte (8-bit)
0	1	Half-word (16-bit)
1	0	Word (32-bit)
1	1	Reserved

PDC CH0...CH9 Memory Pointer Register

Name: PDC_MPR0...PDC_MPR9
Access: Read/Write
Base Address: 0xXX8

31	30	29	28	27	26	25	24
CHPTR[31:24]							

23	22	21	20	19	18	17	16
CHPTR[23:16]							

15	14	13	12	11	10	9	8
CHPTR[15:8]							

7	6	5	4	3	2	1	0
CHPTR[7:0]							

- **CHPTR[31:0]: Channel Pointer**

CHPTR[31:0] must be loaded with the address of the target buffer (memory address).

PDC CH0...CH9 Transfer Register

Name: PDC_TCR0...PDC_TCR9
Access: Read/Write
Base Address: 0xXXC

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

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

15	14	13	12	11	10	9	8
CHCTR15:8]							

7	6	5	4	3	2	1	0
CHCTR[7:0]							

- **CHCTR[15:0]: Channel Counter**

CHCTR[15:0] must be loaded with the size of the receive buffer.

0: Stop Peripheral Data Transfer dedicated to the peripheral X.

1 to 65535: Start immediately Peripheral Data Transfer.

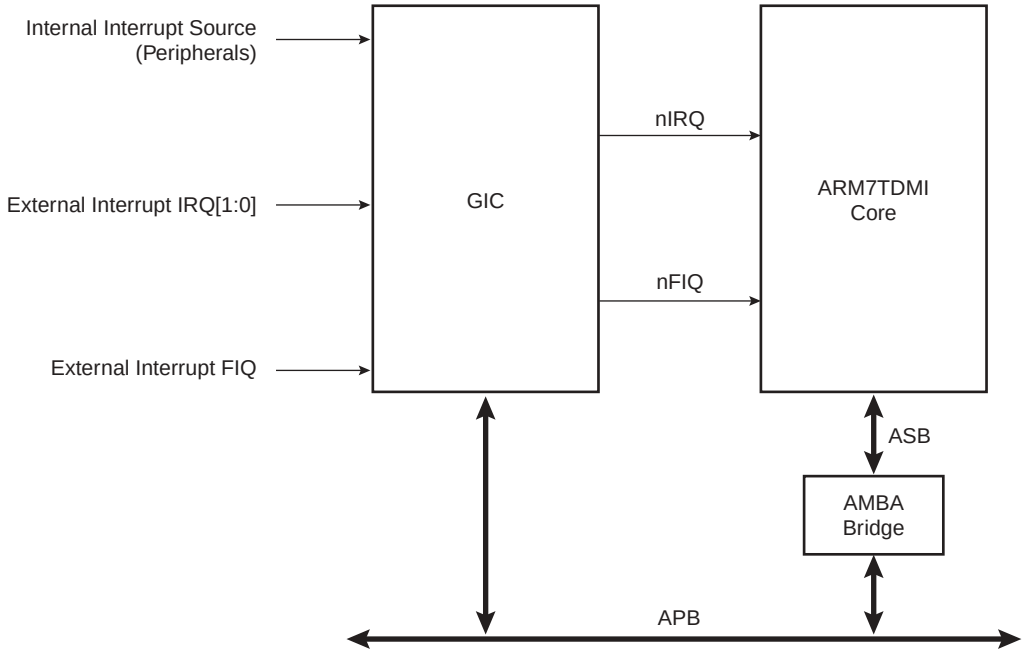
Generic Interrupt Controller (GIC)

Overview

The GIC is an 8-level priority, individually maskable, vectored interrupt controller. It can substantially reduce the software and real time overhead in handling internal and external interrupts.

The interrupt controller is connected to the nFIQ (fast interrupt request) and the nIRQ (standard interrupt request) inputs of the ARM7TDMI processor (see Figure 45 below). The processor's nFIQ line can only be asserted by the external fast interrupt request input, FIQ. The nIRQ line can be asserted by all other internal and external interrupt sources.

Figure 45. GIC Connection to Core



An 8-level priority encoder allows the user to define the priority between the different nIRQ interrupt sources. The internal interrupt sources are programmed to be level sensitive or edge triggered. The external interrupt sources can be programmed to be positive or negative edge triggered or high or low level sensitive.

Table 38. Interrupt Sources

Interrupt Source	Name	Description	GIC bit
0	FIQ	Fast interrupt	FIQ
1	SWIRQ0	Software interrupt 0	SWIRQ0
2	INT_0	Watch Dog	WD
3	INT_1	Watch Timer	WT
4	INT_2	USART0	USART0
5	INT_3	USART1	USART1
6	INT_4	CAN3	CAN3
7	INT_5	SPI	SPI
8	INT_6	CAN1	CAN1
9	INT_7	CAN2	CAN2
10	INT_8	ADC0	ADC0
11	INT_9	ADC1	ADC1
12	INT_10	General Purpose Timer 0 channel 0	GPT0CH0
13	INT_11	General Purpose Timer 0 channel 1	GPT0CH1
14	INT_12	General Purpose Timer 0 channel 2	GPT0CH2
15	SWIRQ0	Software interrupt 1	SWIRQ0
16	SWIRQ1	Software interrupt 2	SWIRQ1
17	SWIRQ2	Software interrupt 3	SWIRQ2
18	INT_13	General Purpose Timer 1 channel 0	GPT1CH0
19	INT_14	Pulse Width Modulation	PWM
20	INT_18	CAN0	CAN0
21	INT_19	UPIO	UPIO
22	INT_20	Capture 0	CAPT0
23	INT_21	Capture 1	CAPT1
24	INT_22	Simple Timer 0	ST0
25	INT_23	Simple Timer 1	ST1
26	SWIRQ4	Software interrupt 4	SWIRQ4
27	SWIRQ5	Software interrupt 5	SWIRQ5
28	EXT_0	External interrupt IRQ0	IRQ0
29	EXT_1	External interrupt IRQ1	IRQ1
30	SWIRQ6	Software interrupt 6	SWIRQ6
31	SWIRQ7	Software interrupt 7	SWIRQ7

Interrupt Handling

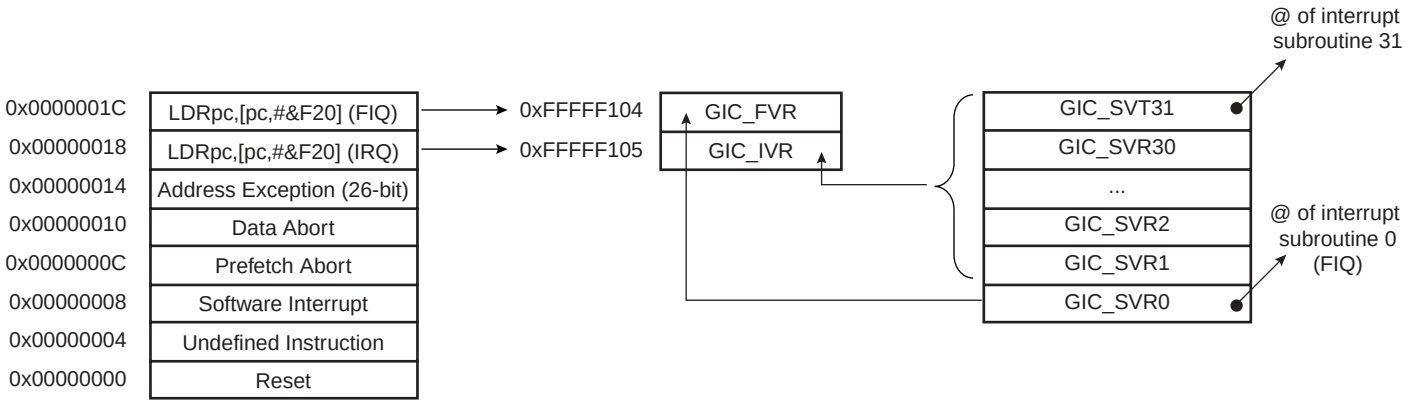
Hardware Interrupt Vectoring

Hardware interrupt vectoring reduces the number of instructions to reach the interrupt handler to only one. By storing the following instruction at address 0x00000018, the processor loads the program counter with the interrupt handler address stored in the GIC_IVR register. Execution is then vectored to the interrupt handler corresponding to the current interrupt (see Figure 46 below).

```
ldr PC,[PC,# -&F20]
```

The current interrupt is the interrupt with the highest priority when the Interrupt Vector Register (GIC_IVR) is read. The value read in the GIC_IVR corresponds to the address stored in the Source Vector Register (GIC_SVR) of the current interrupt. Each interrupt source has its corresponding GIC_SVR. In order to take advantage of the hardware interrupt vectoring, it is necessary to store the address of each interrupt handler in the corresponding GIC_SVR at system initialization.

Figure 46. GIC Automatic Vectoring



Priority Controller

The nIRQ line is controlled by an 8-level priority encoder. Each source has a programmable priority level of 7 to 0. Level 7 is the highest priority and level 0 the lowest.

When the GIC receives more than one unmasked interrupt at a time, the interrupt with the highest priority is serviced first. If both interrupts have equal priority, the interrupt with the lowest interrupt source number is serviced first (see Table 38, “Interrupt Sources,” on page 93).

The current priority level is defined as the priority level of the current interrupt at the time the GIC_IVR register is read (the interrupt that will be serviced).

If a higher priority unmasked interrupt occurs while an interrupt already exists, there are two possible outcomes depending on whether the GIC_IVR has been read:

- If the nIRQ line has been asserted but the GIC_IVR has not been read, then the processor will read the new higher priority interrupt handler address in the GIC_IVR register and the current interrupt level is updated.
- If the processor has already read the GIC_IVR, then the nIRQ line is reasserted. When the processor has authorized nested interrupts to occur and reads the GIC_IVR again, it reads the new higher priority interrupt handler address. At the same time the current priority value is pushed onto a first-in last-out stack and the current priority is updated to the higher priority.

When the end of Interrupt Command Register (GIC_EOICR) is written, the current interrupt level is updated with the last stored interrupt level from the stack (if any). Hence at the end of a higher priority interrupt, the GIC returns to the previous state corresponding to the preceding lower priority interrupt that had been interrupted.

Software Interrupt Handling

The interrupt handler must read the GIC_IVR as soon as possible. This de-asserts the nIRQ request to the processor and clears the interrupt in case it is programmed to be edge triggered. This permits the GIC to assert the nIRQ line again when a higher priority unmasked interrupt occurs.

At the end of the interrupt service routine, the end of Interrupt Command Register (GIC_EOICR) must be written. This allows pending interrupts to be serviced.

Interrupt Masking

Each interrupt source, including FIQ, can be enabled or disabled using the command registers GIC_IECR and GIC_IDCR. The interrupt mask can be read in the read only register GIC_IMR. A disabled interrupt does not affect the servicing of other interrupts.

Interrupt Clearing and Setting

All interrupt sources that are programmed to be edge triggered (including FIQ) can be individually set or cleared by respectively writing to the GIC_ISCR and GIC_ICCR registers. This function of the interrupt controller is available for auto-test or software debug purposes.

Configuration Steps

- Initialization of the interrupt at the peripheral level
- Configuration of GIC_SMR: Priority and detection mode
- Configuration of GIC_SVR: Address of the function associated to this interrupt
- Configuration of GIC_IER: Enable the corresponding peripheral

Fast Interrupt Request

The external FIQ line is the only source which can raise a fast interrupt request to the processor. Therefore it has no priority controller. It can be programmed to be positive or negative edge triggered or high or low level sensitive in the GIC_SMR0 register.

The fast interrupt handler address can be stored in the GIC_SVR0 register. The value written into this register is available by reading the GIC_FVR register when an FIQ interrupt is raised. By storing the following instruction at address 0x0000001C, the processor will load the program counter with the interrupt handler address stored in the GIC_FVR register.

```
ldr PC,[PC, #-&F20]
```

Alternatively the interrupt handler can be stored starting from address 0x0000001C as described in the ARM7TDMI datasheet.

Software Interrupt

Any interrupt source of the GIC can be a software interrupt. It must be programmed to be edge triggered in order to set or clear it by writing to the GIC_ISCR and GIC_ICCR. This is totally independent of the SWI instruction of the ARM7TDMI processor.

Spurious Interrupt

A spurious interrupt is a signal of very short duration on one of the interrupt input lines.

Standard Interrupt Sequence

For details on the registers mentioned in the steps below, refer to the ARM7TDMI Embedded Core Datasheet.

It is assumed that:

- The Generic Interrupt Controller has been programmed, GIC_SVR are loaded with corresponding interrupt service routine addresses and interrupts are enabled.
- The Instruction at address 0x18 (IRQ exception vector address) is:

  `ldr pc, [pc, #-&F20].`

When nIRQ is asserted, if the bit I of CPSR is 0, the sequence is:

1. The CPSR is stored in SPSR_irq, the current value of the Program Counter is loaded in the IRQ link register (r14_irq) and the Program Counter (r15) is loaded with 0x18. In the following cycle during fetch at address 0x1C, the ARM core adjusts r14_irq, incrementing it by 4.
2. The ARM core enters IRQ mode, if it is not already.
3. When the instruction loaded at address 0x18 is executed, the Program Counter is loaded with the value read in GIC_IVR. Reading the GIC_IVR has the following effects:
   - Sets the current interrupt to be the one pending with the highest priority. The current level is the priority level of the current interrupt.
   - De-asserts the nIRQ line on the processor (even if vectoring is not used, GIC_IVR must be read in order to de-assert nIRQ).
   - Automatically clears the interrupt, if it has been programmed to be edge triggered; pushes the current level on to the stack, returns the value written in the GIC_SVR corresponding to the current interrupt.
4. The previous step branches to the corresponding interrupt service routine. This should start by saving the Link Register (r14_irq) and the SPSR (SPSR_irq). Note that the Link Register must be decremented by 4 when it is saved, if it is to be restored directly into the Program Counter at the end of the interrupt. The instruction: `sub pc, lr, #4` may be used, for example.
5. Further interrupts can then be unmasked by clearing the I bit in CPSR, allowing reassertion of the nIRQ to be taken into account by the core. This can arise if an interrupt with a higher priority than the current one occurs.
6. The Interrupt Handler can then proceed as required, saving the registers which will be used and restoring them at the end. During this phase, an interrupt of higher priority than the current level will restart the sequence from step 1. Note that if the interrupt is programmed to be level sensitive, the source of the interrupt must be cleared during this phase.
7. The I bit in CPSR must be set in order to mask interrupts before exiting, to ensure that the interrupt is completed in an orderly manner.
8. The End Of Interrupt Command Register (GIC_EOICR) must be written in order to indicate to the GIC that the current interrupt is finished. This causes the current level to be popped from the stack, restoring the previous current level if one exists on the stack. If another interrupt is pending, with lower or equal priority than the old current level but with higher priority than the new current level, the nIRQ line is reasserted, but the interrupt sequence does not immediately start because the I bit is set in the core.
9. The SPSR (SPSR_irq) is restored. Finally, the saved value of the Link Register is restored directly into the PC. This has the effect of returning from the interrupt to whatever was being executed before, and of loading CPSR with the stored

SPSR, masking or unmasking the interrupts depending on the state saved in the SPSR (the previous state of the ARM core).

Note: The I bit in the SPSR is significant. If it is set, it indicates that the ARM core was just about to mask IRQ interrupts when the mask instruction was interrupted. Hence, when the SPSR is restored, the mask instruction is completed (IRQ is masked).

Fast Interrupt Sequence

For details on the registers mentioned in the steps below, refer to the ARM7TDMI Embedded Core Datasheet.

It is assumed that:

- The Generic Interrupt Controller has been programmed, GIC_SVR[0] is loaded with a fast interrupt service routine address and the fast interrupt is enabled.
- The Instruction at address 0x1C (FIQ exception vector address) is:

  ```
  ldr pc, [pc, #-&F20]
  ```

- Nested Fast Interrupts are not needed by the user.

When nFIQ is asserted, if the F bit of CPSR is 0, the sequence is:

1. CPSR is stored in SPSR_fiq, the current value of the Program Counter is loaded in the FIQ link register (r14_fiq) and the Program Counter (r15) is loaded with 0x1C. In the following cycle, during fetch at address 0x20, the ARM core adjusts r14_fiq, incrementing it by 4.
2. The ARM core enters FIQ mode.
3. When the instruction loaded at address 0x1C is executed, the Program Counter is loaded with the value read in GIC_FVR. Reading the GIC_FVR automatically clears the fast interrupt (source 0 connected to the FIQ line), if it has been programmed to be edge triggered. In this case only, it de-asserts the nFIQ line on the processor.
4. The previous step branches to the corresponding interrupt service routine. It is not necessary to save the Link Register (r14_fiq) and the SPSR (SPSR_fiq) if nested fast interrupts are not needed.
5. The Interrupt Handler can then proceed as required. It is not necessary to save registers r8 to r13 because FIQ mode has its own dedicated registers and the user registers r8 to r13 are banked. The other registers, r0 to r7, must be saved before being used, and restored at the end (before the next step). Note that if the fast interrupt is programmed to be level sensitive, the source of the interrupt must be cleared during this phase in order to de-assert the nFIQ line.
6. Finally, the Link Register (r14_fiq) is restored into the PC after decrementing it by 4 (with instruction `sub pc, lr, #4` for example). This has the effect of returning from the interrupt to whatever was being executed before, and of loading CPSR with the SPSR, masking or unmasking the fast interrupt depending on the state saved in the SPSR.

Note: The F bit in the SPSR is significant. If it is set, it indicates that the ARM core was just about to mask FIQ interrupts when the mask instruction was interrupted. Hence when the SPSR is restored, the interrupted instruction is completed (FIQ is masked).

Spurious Interrupt Sequence

A spurious interrupt is a signal of very short duration on one of the interrupt input lines. It is handled by the following sequence of actions.

1. When an interrupt is active, the GIC asserts the IRQ (or nFIQ) line and the ARM7TDMI enters IRQ (or FIQ) mode. At this moment, if the interrupt source disappears, the nIRQ (or nFIQ) line is de-asserted but the ARM7TDMI continues with the interrupt handler.
2. If the IRQ Vector Register (GIC_IVR) is read when the nIRQ is not asserted, the GIC_IVR is read with the contents of the Spurious Interrupt Vector Register.
3. If the FIQ Vector Register (GIC_FVR) is read when the nFIQ is not asserted, the GIC_FVR is read with the contents of the Spurious Interrupt Vector Register.
4. The Spurious Interrupt Routine must at least write into the GIC_EOICR to perform an end of interrupt command. Until the GIC_EOICR write is received by the interrupt controller, the nIRQ (or nFIQ) line is not reasserted.
5. This causes the ARM7TDMI to jump into the Spurious Interrupt Routine.
6. During a Spurious Interrupt Routine, the Interrupt Status Register GIC_ISR reads 0.

Generic Interrupt Controller (GIC) Memory Map

Base Address: 0xFFFFF000

Table 39. GIC Memory Map

Offset	Register	Name	Access	Reset State
0x000 – 0x07C	GIC Source Mode Register 0 – GIC Source Mode Register 31	GIC_SMR0 – GIC_SMR31	Read/Write	0x00000000
0x080 – 0x0FC	GIC Source Vector Register 0 – GIC Source Vector Register 31	GIC_SVR0 – GIC_SVR31	Read/Write	0x00000000
0x100	GIC IRQ Vector	GIC_IVR	Read-only	0x00000000
0x104	GIC FIQ Vector	GIC_FVR	Read-only	0x00000000
0x108	GIC Interrupt Status	GIC_ISR	Read-only	0x00000000
0x10C	GIC interrupt Pending	GIC_IPR	Read-only	0xXXXXXXXX
0x110	GIC Interrupt Mask	GIC_IMR	Read-only	0x00000000
0x114	GIC Core Interrupt Status	GIC_CISR	Read-only	0x00000000
0x118 – 0x11C	Reserved	–	–	–
0x120	GIC Interrupt Enable Command	GIC_IECR	Write-only	–
0x124	GIC Interrupt Disable Command	GIC_IDCR	Write-only	–
0x128	GIC Interrupt Clear Command	GIC_ICCR	Write-only	–
0x12C	GIC Interrupt Set Command	GIC_ISCR	Write-only	–
0x130	GIC End of Interrupt Command	GIC_EOICR	Write-only	–
0x134	GIC Spurious Vector	GIC_SPU	Read/Write	0x00000000

GIC Source Mode Register

Name: GIC_SMR0...GIC_SMR31
Access: Read/Write
Base Address: 0x000...0x07C

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
–	SRCTYP[1:0]		–	–	PRIOR[2:0]		

- **PRIOR[2:0]: Priority Level**

These bits program the priority level (from 0-lowest to 7-highest) of all the interrupt sources. The priority level is not used for the FIQ in the SMR0.

- **SRCTYP[1:0]: Interrupt Source Type**

SRCTYP[1:0](1)		Internal Sources	External Sources
0	0	High level sensitive	Low level sensitive
0	1	Positive edge triggered	Negative edge triggered
1	0	High level sensitive	High level sensitive
1	1	Positive edge triggered	Positive edge triggered

Note: 1. All the interrupts used by internal peripherals are considered as internal interrupts and subsequently the SRCTYP1 bit is always read at 0.

GIC Source Vector Register

Name: GIC_SVR0...GIC_SVR31
Access: Read/Write
Base Address: 0x080...0x0FC

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

- **VECT[31:0]: Interrupt Handler Address**

Address of the corresponding handler for each interrupt source.

GIC Interrupt Vector Register

Name: GIC_IVR
Access: Read-only
Base Address: 0x100

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

- **IRQV[31:0]: Interrupt Vector Address**

Address of the currently serviced interrupt vector (user programmed in the GIC_SVR register).

Note: GIC_IVR = 0x00000000 when there is no current interrupt.

Note: When debugging, to read the GIC_IVR register clears the IRQ interrupt if present at the GIC. To avoid this behavior, users should use ghost registers (see “Ghost Registers” on page 7).

GIC FIQ Vector Register

Name: GIC_FVR
Access: Read-only
Base Address: 0x104

31	30	29	28	27	26	25	24
FIQV[31:24]							

23	22	21	20	19	18	17	16
FIQV[23:16]							

15	14	13	12	11	10	9	8
FIQV[15:8]							

7	6	5	4	3	2	1	0
FIQV[7:0]							

- **FIQV[31:0]: FIQ Vector Address**

Address of the FIQ serviced interrupt (user programmed in the GIC_SVR0 register).

Note: When debugging, to read the GIC_FVR register clears the FRQ interrupt if present at the GIC. To avoid this behavior, users should use ghost registers (see “Ghost Registers” on page 7).

GIC Interrupt Status Register

Name: GIC_ISR
Access: Read-only
Base Address: 0x108

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[4:0]				

- **IRQID[4:0]: Current IRQ Identifier**

Current interrupt source number.

GIC Interrupt Pending Register

Name: GIC_IPR
Access: Read-only
Base Address: 0x10C

31	30	29	28	27	26	25	24
SWIRQ7	SWIRQ6	IRQ1	IRQ0	SWIRQ5	SWIRQ4	ST1	ST0
23	22	21	20	19	18	17	16
CAPT1	CAPT0	UPIO	CAN0	PWM	GPT1CH0	SWIRQ3	SWIRQ2
15	14	13	12	11	10	9	8
SWIRQ1	GPT0CH2	GPT0CH1	GPT0CH0	ADC1	ADC0	CAN2	CAN1
7	6	5	4	3	2	1	0
SPI	CAN3	USART1	USART0	WT	WD	SWIRQ0	FIQ

- **Interrupt Pending**

0: Corresponding interrupt is inactive.

1: Corresponding interrupt is pending.

GIC Interrupt Mask Register

Name: GIC_IMR
Access: Read-only
Base Address: 0x110

31	30	29	28	27	26	25	24
SWIRQ7	SWIRQ6	IRQ1	IRQ0	SWIRQ5	SWIRQ4	ST1	ST0
23	22	21	20	19	18	17	16
CAPT1	CAPT0	UPIO	CAN0	PWM	GPT1CH0	SWIRQ3	SWIRQ2
15	14	13	12	11	10	9	8
SWIRQ1	GPT0CH2	GPT0CH1	GPT0CH0	ADC1	ADC0	CAN2	CAN1
7	6	5	4	3	2	1	0
SPI	CAN3	USART1	USART0	WT	WD	SWIRQ0	FIQ

- **Interrupt Mask**

0: Corresponding interrupt is disabled.

1: Corresponding interrupt is enabled.

GIC Core Interrupt Status Register

Name: GIC_CISR
Access: Read-only
Base Address: 0x114

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
–	–	–	–	–	–	NIRQ	NFIQ

- **NIRQ: NIRQ Status**

0: nIRQ line is inactive.

1: nIRQ line is active.

- **NFIQ: NFIQ Status**

0: nFIQ line is inactive.

1: nFIQ line is active.

GIC Interrupt Enable Command Register

Name: GIC_IECR
Access: Write-only
Base Address: 0x120

31	30	29	28	27	26	25	24
SWIRQ7	SWIRQ6	IRQ1	IRQ0	SWIRQ5	SWIRQ4	ST1	ST0
23	22	21	20	19	18	17	16
CAPT1	CAPT0	UPIO	CAN0	PWM	GPT1CH0	SWIRQ3	SWIRQ2
15	14	13	12	11	10	9	8
SWIRQ1	GPT0CH2	GPT0CH1	GPT0CH0	ADC1	ADC0	CAN2	CAN1
7	6	5	4	3	2	1	0
SPI	CAN3	USART1	USART0	WT	WD	SWIRQ0	FIQ

GIC Interrupt Disable Command Register

Name: GIC_IDCR
Access: Write-only
Base Address: 0x124

31	30	29	28	27	26	25	24
SWIRQ7	SWIRQ6	IRQ1	IRQ0	SWIRQ5	SWIRQ4	ST1	ST0
23	22	21	20	19	18	17	16
CAPT1	CAPT0	UPIO	CAN0	PWM	GPT1CH0	SWIRQ3	SWIRQ2
15	14	13	12	11	10	9	8
SWIRQ1	GPT0CH2	GPT0CH1	GPT0CH0	ADC1	ADC0	CAN2	CAN1
7	6	5	4	3	2	1	0
SPI	CAN3	USART1	USART0	WT	WD	SWIRQ0	FIQ

GIC Interrupt Clear Command Register

Name: GIC_ICCR
Access: Write-only
Base Address: 0x128

31	30	29	28	27	26	25	24
SWIRQ7	SWIRQ6	IRQ1	IRQ0	SWIRQ5	SWIRQ4	ST1	ST0
23	22	21	20	19	18	17	16
CAPT1	CAPT0	UPIO	CAN0	PWM	GPT1CH0	SWIRQ3	SWIRQ2
15	14	13	12	11	10	9	8
SWIRQ1	GPT0CH2	GPT0CH1	GPT0CH0	ADC1	ADC0	CAN2	CAN1
7	6	5	4	3	2	1	0
SPI	CAN3	USART1	USART0	WT	WD	SWIRQ0	FIQ

- **Software Interrupt Clear**

0: No effect.

1: Clears corresponding software interrupt.

GIC Interrupt Set Command Register

Name: GIC_ICSR
Access: Write-only
Base Address: 0x12C

31	30	29	28	27	26	25	24
SWIRQ7	SWIRQ6	IRQ1	IRQ0	SWIRQ5	SWIRQ4	ST1	ST0
23	22	21	20	19	18	17	16
CAPT1	CAPT0	UPIO	CAN0	PWM	GPT1CH0	SWIRQ3	SWIRQ2
15	14	13	12	11	10	9	8
SWIRQ1	GPT0CH2	GPT0CH1	GPT0CH0	ADC1	ADC0	CAN2	CAN1
7	6	5	4	3	2	1	0
SPI	CAN3	USART1	USART0	WT	WD	SWIRQ0	FIQ

- **Software Interrupt Set**

0: No effect.

1: Clears corresponding software interrupt.

GIC End of Interrupt Command Register

Name: GIC_EOICR
Access: Write-only
Base Address: 0x130

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
–	–	–	–	–	–	–	–

A write access to this register (with any value) indicates that the interrupt treatment is completed.

GIC Spurious Vector Register

Name: GIC_SPU
Access: Read/Write
Base Address: 0x080...0x0FC

31	30	29	28	27	26	25	24
SPUVECT[31:24]							

23	22	21	20	19	18	17	16
SPUVECT[23:16]							

15	14	13	12	11	10	9	8
SPUVECT[15:8]							

7	6	5	4	3	2	1	0
SPUVECT[7:0]							

- **SPUVECT[31:0]: Spurious Interrupt Vector Handler Address**

Address of the spurious interrupt handler.

10-bit Analog to Digital Converter (ADC)

Overview

The AT91SAM7A2 includes two 10-bit ADCs (8 channels).

The 10-bit Analog to Digital Converter (ADC) can be configured in different modes as follows.

- Single input/one shot mode: One input selected and a single conversion.
- Single input/continuous mode: One input selected, the microprocessor gives the first start to the peripheral which is then completely independent. The PDC can be used to save the resulting data in memory.
  The conversion is stopped in two ways:
  1. By the microprocessor, by setting the STOP bit of the active control register.
  2. By the PDC: TEND can stop the conversion if the STOPEN bit of the mode register is active. This allows the user to order a conversion of a certain number of data without any software intervention.
- Multiple input/one shot mode: The user selects which analog inputs will be converted and specifies the names of the inputs that will be considered by the ADC and in which order they will be converted. This allows the user to make conversions of some of the eight inputs in the order of preference, in one shot. The PDC can be used to save each result.
- Multiple input/continuous mode: Several analog inputs are converted, the microprocessor first starts up the peripheral which then becomes completely independent.
  The conversion is stopped in two ways:
  – By the microprocessor, by setting the STOP bit of the active control register.
  – By the PDC: TEND can stop the conversion if the STOPEN bit of the mode register is active. This allows the user to order a conversion of a certain number of data without any software intervention.

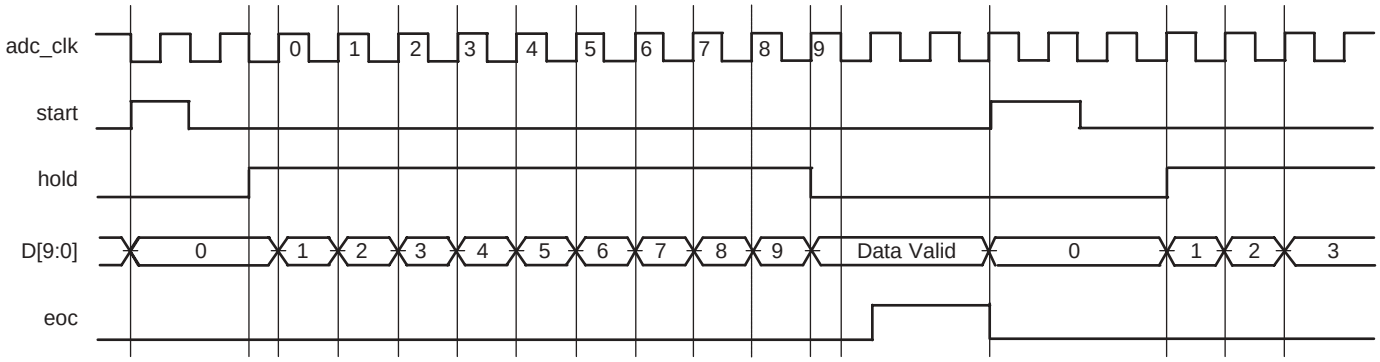
If the PDC is active, a flag is set when the transfer of all the data is finished.

The converter is composed of a 10-bit cascaded potentiometric digital to analog converter connected to the negative input of a Sample and Hold comparator. It is based on a string of 64 polysilicon resistors connected between reference inputs VREF. So, the analog input to be converted needs to be in the interval [GND:VREFP].

In the terminology, the Integral Non-Linearity (INL) is a measure of the maximum deviation from a straight line passing through the end-points of the transfer function. It does not include the full scale error and the zero error. It is calculated from the real value of the LSB which is calculated from the output range (output variation between the minimal value 0 and the maximal one).

The Differential Non-linearity (DNL) is the difference between the measured change and the ideal change between any two adjacent codes. A specified DNL of ±1 LSB max over the operating point temperature range ensures monotony. The DNL is relative to the real value of the LSB.

Figure 47. Signal Description



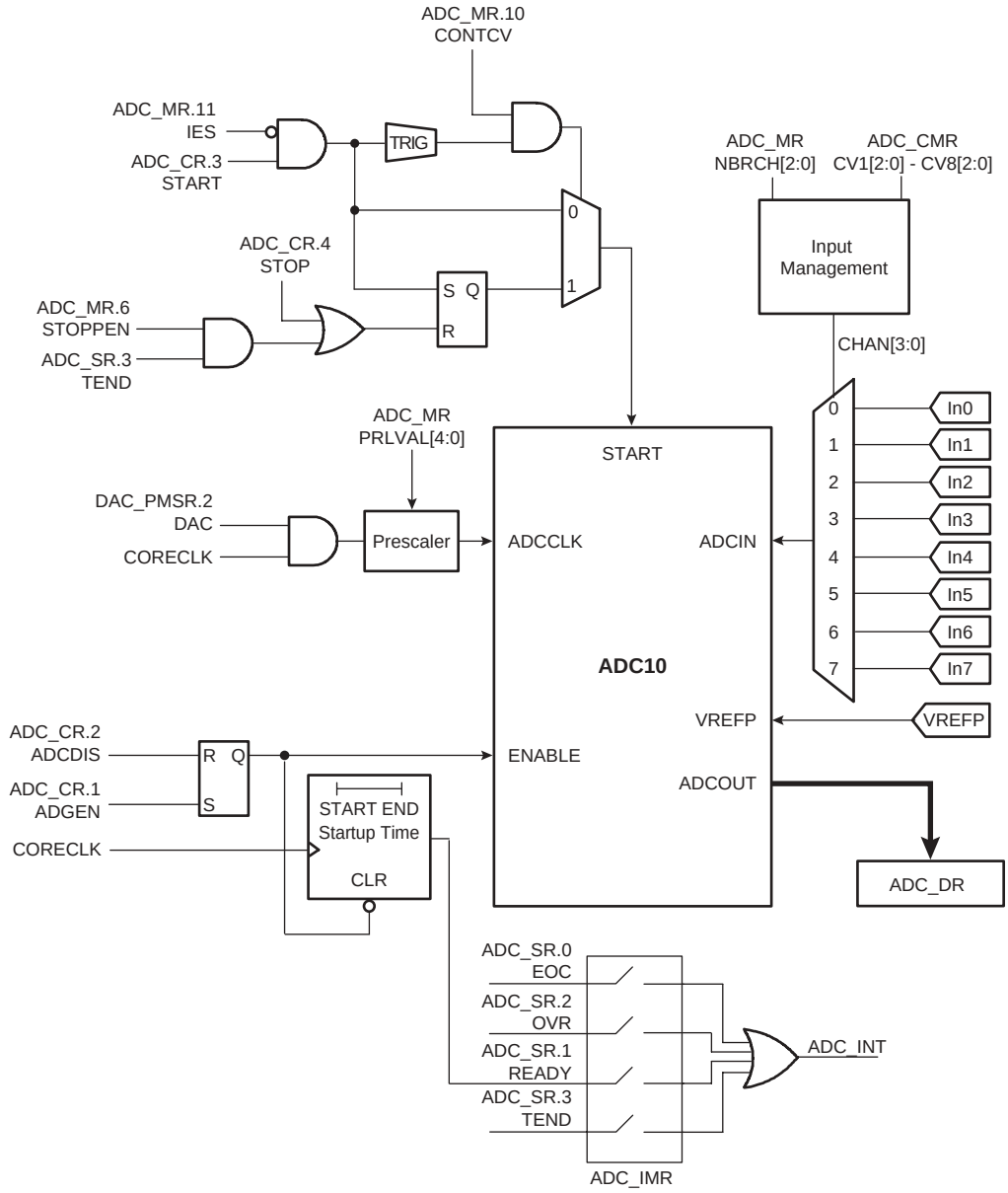
When start is high, the cell is reset and the internal clock is inhibited. The D[9:0] signal is at 512, so the internal ADC output is at (VREFP - GND)/2. The hold output is low, so the input voltage at the Sample and Hold stage of the comparator is sampled.

When the start goes low, the hold signal goes high with the next falling clock edge, and the input voltage is stored on the Sample and Hold capacitor. The comparator performs a comparison between the stored input voltage and the ADC output.

With the next rising clock signal, the comparator output becomes valid, this value being stored as D[9] in the internal register. The internal shift register now sets D[8] high, and a new comparison is performed. The next rising edge of clock stores the result in D[8]. After another 8 low-pulse of clock, all 10 bits are valid at output D[9:0]. The End of Conversion signal (EOC) is set high. The input is sampled again as the hold signal goes low and a new conversion can be started with a high-pulse of the start signal.

Block Diagram

Figure 48. ADC Block Diagram



Conversion Details

The conversion of a single analog value to 10-bit digital data requires 11 ADC clock cycles comprised of the following:

- One ADC clock cycle to sample the analog input signal.
- Ten ADC clock cycles to fix the ten resultant bits.

The clock input to the ADC has to be lower than 700 kHz. The user can choose a frequency by writing in the ADC mode register (ADC_MR) the preload value of the counter (PRLVAL[4:0]). The master clock is divided by this value, and the result is the ADC clock. The preload value is coded on 7 bits with the 2 LSBs always low to guarantee a duty cycle of ½. The user can divide master clock by 0 (so ADC clock = CORECLK), 4, 8, …, 48, ..., 64, ..., 124. This allows adapting the ADC clock as well as possible with a master clock comprised between 700 kHz and 30 MHz. For example, if CORECLK = 30 MHz with a preload value of 48 the ADC clock is 625 kHz.

A single conversion at the maximum clock rate permitted (i.e. 700 kHz) will occur in 15.7 ms.

The ADC starts the conversion by writing 1 in the START bit of the control register (ADC_CR).

Writing 1 to the START bit starts the conversion even if the analog structure begins the conversion when start goes low, the interface transmits the opposite of the start command to the analog part.

For a conversion, different input combinations can be selected. The 3-bit NBRCH[2:0] indicates how many inputs will be converted (the real number is the value of NBRCH incremented by one).

The result of the conversion is stored in the convert data register (ADC_DR). When the conversion is complete, the analog part activates the EOC bit in the ADC Status register (ADC_SR) and sends an EOC signal to the PDC which can take the result and write at a memory location. The EOC bit in ADC_SR (Status Register) is cleared when the ADC_DR (Convert Data Register) is read. If a new result arrives before the PDC or the CPU read the old data, the Overrun bit (OVR) is set active to specify to the microprocessor that data is lost. If the PDC is used to save the results and if the transfer of all the data is finished, the PDC sets the TEND bit to a logical 1.

The READY bit is set after an absolute time of 4 µs after an enable command, which corresponds to the initialization time of the analog part. This time is necessary to stabilize the analog structure and does not depend on the choice of the ADC clock or the names of analog inputs considered. The number of master clock periods necessary to wait during 4 µs is remembered in the STARTUPTIME bits of the mode register.

The user can make conversions in continuous mode. This status is indicated by the CONTCV bit of the ADC Mode Register (ADC_MR). In this case, the microprocessor gives the first start to the ADC and the peripheral does not stop the conversion until the STOP bit of the ADC Control Register is set, or when the TEND bit of the status register is set if STOPEN is active. However, the user should be vigilant, because after a stop command in continuous mode, the ADC finishes the ongoing conversion and this may appear to be an extra conversion. The digital interface between the analog part and the APB bus is in stand alone mode; this permits conversion without any help. This mode can be associated with multiple inputs as well as a single input. The different steps of the conversion are equivalent to those of a single conversion.

If the ADC is configured in continuous mode, a particular sequence should be observed at the end of a PDC transfer. When a set of PDC transfers have reached the end, the ADC runs an extra conversion. The CPU must clear the TEND flag before the end (EOC in ADC_SR) of the extra conversion. If the software can not ensure clearing the TEND flag before the EOC of the extra conversion, two solutions are available:

- When a set of PDC transfers is completed, before starting an additional set of ADC conversions associated with PDC transfers, software must reset the ADC (SWRST in ADC_CR).
- When a set of PDC transfers is completed, before starting an additional set of ADC conversions associated with PDC transfers, software should start a conversion, wait for the EOC flag in the ADC_SR register and read ADC_DR to clear the EOC flag.

Modes of Operation

The ADC can be active or shutdown; in the latter case it is in a power saving mode.

At any time the software can program the ADC to be disabled to save power. Setting the ADCDIS bit of the ADC Control register will put the ADC Analog circuitry into standby mode. To reduce the power consumption near to 0, the user can switch off the ADC

peripheral clock in the Disable Clock Register (ADC_DCR), thus also disabling the Digital part of the ADC.

When the ADC is re-enabled, a minimum of 4 µs is required before the analog circuitry is stabilized and ready for reliable usage. The user has to initialize the STARTUPTIME in the ADC_MR register value by indicating how many clock periods of the master clock are necessary to make 4 µs.

When the ADC is enabled for the first time (after standby mode but not after wait mode) the interface starts counting and then sets the READY bit in ADC_SR (Status Register) which allows a conversion. The ready flag also indicates if the ADC is converting data, or is waiting for a start, because this flag is low when ADC is disabled or converting and is high when ADC is waiting for a start.

Wait Mode

When the analog part of the ADC peripheral is in standby mode, but the digital part of the peripheral is active, the circuitry is in wait mode. To leave this status, the user can do one of the following.

- Disable the CPU clock. The peripheral is then in standby mode.
- Enable the analog circuitry by setting the ADCEN of the control register. The peripheral is active.

Standby Mode

When the analog part as well as the digital part are in standby mode, the ADC peripheral is really in standby mode. The digital part is in standby mode when the clock is disabled. The microcontroller disables the peripheral clock by writing in the Disable Clock register (ADC_DCR) The microcontroller disables the analog part by setting the ADCDIS bit of the control register.

Warnings

As the standby mode can be effective only after having been in wait mode, a specific order in the different actions has to be respected. If the user disables the clock of the interface before disabling the analog part of the peripheral, the peripheral is not in standby mode.

START Mode

The peripheral is commanded by an internal start (given by the microprocessor). In this mode, the analog part of the ADC peripheral is waiting for a start.

NBRCH[2:0]

These bits indicate how many inputs will be converted within a one shot or continuous mode.

CONT Mode

The CONTCV bit of the ADC mode register indicates if the peripheral is converting in a continuous mode (in this case the bit is high) or in a one shot mode. This bit is initialized to 0.

Conversion Sequence

The basic sequence of operation after a reset is detailed below.

1. Program the ADC Mode register: the interface has to know the STARTUPTIME and the PRLVAL before receiving a start command. Determine the names of analog inputs. Indicate if conversions are made in continuous mode (CONTCV=1).
2. Program the ADC Control register (ADC_CR) to enable the ADC.
3. Wait for READY bit in ADC_SR. When the flag is set, the ADC is ready to start conversion. An interrupt can be generated to detect when the ADC is ready to start a conversion if the corresponding bit is enabled in the ADC_IMR (Interrupt Mask Register). This flag can be high only after an enable command has been performed because it is this operation which starts the startup time of 4 µs. Every command (even start) before the ready flag is ignored.
4. Initiate conversion start by writing a start command in the ADC Control register.

If a single conversion is being done, the following applies:

5. The analog input voltage is sampled during 1 ADC clock cycle. The digital result is transferred after 10 ADC clock cycles.
6. Conversion completes after 11 ADC clock cycles from the start command. The digital 10-bit data is latched into ADC_DR (Convert Data Register), the EOC bit in ADC_SR (Status Register) is set.
7. The PDC or the CPU can then read the digital value in ADC_DR, which automatically clears the EOC bit in ADC_SR. Another result can be saved before the data has been read. In this case, the OVR bit is set. When the PDC has stored all the data required, the TEND bit is set.

When multiple conversions or continuous conversions are programmed, steps 5, 6 and 7, as stated above, are repeated.

Power Management

The ADC is provided with a power management block allowing optimization of power consumption. See “Power Management Block” on page 22.

Example of Use

Acquisition in a single shot on eight different inputs using the PDC, with an interrupt associated to the end of conversion, is performed according to the following steps.

Configuration Steps

- Enable the clock on the ADC peripheral by writing the ADC bit in ADC_ECR.
- Do a software reset of the ADC to be in a known state by writing the SWRST bit in ADC_CR.
- Configuration of ADC_ MR: Select the ADC clock to be less than 700kHz (PRLVAL = 9 for 30MHz), STOPEN bit remains at 0 as the device is not in continuous conversion mode. Analog circuitry stabilization time is programmed using the STARTUPTIME field (0x78 for 30 MHz) and must be more than 4 µs. The number of acquisitions is programmed in the NBRCH field; write 7 for eight conversions. The CONTCV bit selects the conversion mode and is left at 0.
- Enable the ADC by writing the ADCEN bit in ADC_CR.
- Configuration of ADC_ CMR: This register indicates which input will be used for the eight conversions.
- Choose the input order to be {5,7,6,2,0,1,4,3}, therefore ADC_MR has the following programmed value:

```
((5<<0)|(7<<4)|(6<<8)|(2<<12)|(0<<16)|(1<<20)|(4<<24)|(3<<28))
```

- Configuration of ADC _IER: Generate an interrupt at the end of the eight conversions. This is done by writing TEND. When the PDC completes the eight conversions, an interrupt will be generated. GIC must be configured.
- Configure PDC for ADC module and set it to eight conversions of 16 bits.
- If the status READY bit is set in ADC_SR, start conversions by writing the START bit in ADC_CR, otherwise users should wait the READY flag, in consequence of bit ADCEN in ADC_CR. An interrupt can be associated to this event.

Interrupt Handling

- IRQ Entry and call C function.
- Read ADC_SR and verify the source of the interrupt.
- Clear the corresponding interrupt at peripheral level by writing in the ADC_CSR.
- Interrupt treatment: read data conversion in the memory space programmed in the PDC. Result are between 0 to 1023 (0 to 3.3V with V_{REFP} = 3.3V).
- IRQ Exit.

10-bit Analog to Digital Converter (ADC) Memory Map

Base Address ADC0: 0xFFFC0000

Base Address ADC1: 0xFFFC4000

Table 40. ADC Memory Map

Offset	Register	Name	Access	Reset State
0x000 – 0x04C	Reserved	–	–	–
0x050	Enable Clock Register	ADC_ECR	Write-only	–
0x054	Disable Clock Register	ADC_DCR	Write-only	–
0x058	Power Management Status Register	ADC_PMSR	Read-only	0x00000000
0x05C	Reserved	–	–	–
0x060	Control Register	ADC_CR	Write-only	–
0x064	Mode Register	ADC_MR	Read/Write	0x00000000
0x068	Conversion Mode Register	ADC_CMR	Read/Write	0x00000000
0x06C	Clear Status Register	ADC_CSR	Write-only	–
0x070	Status Register	ADC_SR	Read-only	0x00000000
0x074	Interrupt Enable Register	ADC_IER	Write-only	–
0x078	Interrupt Disable Register	ADC_IDR	Write-only	–
0x07C	Interrupt Mask Register	ADC_IMR	Read-only	0x00000000
0x080	Convert Data Register	ADC_DR	Read-only	0x00000000

ADC Enable Clock Register

Name: ADC_ECR
Access: Write-only
Base Address: 0x050

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

ADC Disable Clock Register

Name: ADC_DCR
Access: Write-only
Base Address: 0x054

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
–	–	–	–	–	–	ADC	–

ADC Power Management Status Register

Name: ADC_PMSR
Access: Read-only
Base Address: 0x058

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
–	–	–	–	–	–	ADC	–

- **ADC: ADC Clock Status**

0: ADC clock disabled.

1: ADC clock enabled.

ADC Control Register

Name: ADC_CR
Access: Write-only
Base Address: 0x060

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
–	–	–	STOP	START	ADCDIS	ADCEN	SWRST

- **SWRST: ADC Software Reset**

0: No effect.

1: Reset of the ADC peripheral.

When a software reset is performed, all the registers of the peripheral are reset.

- **ADCEN: ADC Enable**

0: No effect.

1: ADC is enabled for conversion.

- **ADCDIS: ADC Disable**

0: No effect.

1: ADC is disabled (Standby Mode).

- **START: Start Conversion**

0: No analog to digital conversion to be started.

1: Begin analog to digital conversion, clears EOC bit.

Note: Before starting conversions, users should ensure that the ADC is ready for conversion (READY bit is set to logical one in ADC_SR).

- **STOP: Stop Conversion in Continuous Conversion**

0: No effect.

1: Stop the continuous conversion.

ADC Mode Register

Name: ADC_MR
Access: Read/Write
Base Address: 0x064

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

23	22	21	20	19	18	17	16
–	–	–	–	CONTCV	NBRCH[2:0]		

15	14	13	12	11	10	9	8
STARTUPTIME[7:0]							

7	6	5	4	3	2	1	0
–	STOPEN	–	PRVAL[4:0]				

- **PRLVAL[4:0]: Preload Value**

A division of the Master clock (MCLK) determines ADC_clk. The preload value is chosen by the user to adapt the Master clock to the ADC peripheral as well as possible. It is the start value of the down counter. The LSB is fixed to 0 because this value has to be even parity to guaranty a duty cycle of ½, so the user has only to initialize the 5 MSB.

ADC_clk = CORECLK/(4xPRLVAL[4:0]).

Note: The clock rate to the ADC must not exceed 700 kHz.

- **STOPEN: Stop Enable**

0: TEND cannot stop conversion when the device is in continuous mode.

1: TEND stops conversion when the device is in continuous conversion mode.

This bit is initialized to 0.

- **STARTUPTIME[7:0]: Startup Time**

This value indicates the number of master clock periods necessary to make 4 µs.

This time is the stabilization time of the analog circuitry.

For example, if CORECLK = 30 MHz, 120 periods of CORECLK are needed to obtain the absolute time of 4 µs. So, the STARTUPTIME value of the mode register is 120 (0x78 in hexadecimal code).

- **NBRCH[2:0]: Number of Conversions**

NBRCH[2:0]	Number of Conversions
000	1
001	2
010	3
011	4
100	5
101	6
110	7
111	8

Note: Even in one shot mode, ADC will run multiple conversions if NBRCH[2:0] is greater than 000.

- **CONTCV: Continuous Conversion**

0: one shot mode. ADC converts as much inputs as specified by the NBRCH[2:0] in the order specified in the ADC_CMR, and stops.

1: continuous mode. ADC converts as much inputs as specified by the NBRCH[2:0] in the order specified in the ADC_CMR, and repeats. This bit is initialized to 0.

DC Conversion Mode Register

Name: ADC_CMR
Access: Read/Write
Base Address: 0x068

31	30	29	28	27	26	25	24
–	CV8[2:0]			–	CV7[2:0]		

23	22	21	20	19	18	17	16
–	CV6[2:0]			–	CV5[2:0]		

15	14	13	12	11	10	9	8
–	CV4[2:0]			–	CV3[2:0]		

7	6	5	4	3	2	1	0
–	CV2[2:0]			–	CV1[2:0]		

- **CVx[2:0]: Input Selection**

x = {1, 2, 3, 4, 5, 6, 7, 8} is the conversion number.

CVx[2:0]	Input Selected
000	In0
001	In1
010	In2
011	In3
100	In4
101	In5
110	In6
111	In7

ADC Clear Status Register

Name: ADC_CSR
Access: Write-only
Base Address: 0x06C

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
–	–	–	–	TEND	OVR	–	–

- **OVR: Overrun Interrupt**

0: No effect.

1: Clear OVR interrupt.

- **TEND: End of PDC Transfer Interrupt**

0: No effect.

1: Clear TEND interrupt.

ADC Status Register

Name: ADC_CSR
Access: Read-only
Base Address: 0x070

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

- **EOC: End of Conversion**

0: Conversion not complete or inactive.

1: Conversion complete, data in ADC_DR is valid.

This bit is cleared when the ADC_DR is read.

- **READY: ADC Ready for Conversion**

0: ADC ignores start or stop command. It is not ready for conversion or is converting data.

1: ADC is ready to start a conversion.

To explain more completely “ready_flag”, define “working” as the event high when ADC is converting data and "analog_ready" the event low when the analog part is disabled or in the initializing phase.

analog_ready	working	ready_flag
0	0	0
0	1	0
1	0	1
1	1	0

- **OVR: Overrun**

0: No data has been transferred by ADC from the last ADC_DR read.

1: At least one data has been transferred by ADC since the last ADC_DR read.

- **TEND: End of Total Transfer of PDC**

0: Transfer of all data not complete.

1: PDC transfer complete.

This bit is set when the transfer of all the data by the PDC is complete. When we are in continuous mode, it stops the conversion only if the STOPEN bit of the mode register is active.

- **ADCENS: ADC Enable Status**

0: ADC is disabled.

1: ADC is enabled.

- **CTCVS: Continuous Mode Status**

0: One shot mode with help of microprocessor.

1: Continuous mode, the peripheral is stand-alone. This bit is initialized to 0 and changes when there is a change of mode. This bit never generates an interrupt.

ADC Interrupt Enable Register

Name: ADC_IER
Access: Write-only
Base Address: 0x074

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
–	–	–	–	TEND	OVR	READY	EOR

ADC Interrupt Disable Register

Name: ADC_IDR
Access: Write-only
Base Address: 0x078

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
–	–	–	–	TEND	OVR	READY	EOR

ADC Interrupt Mask Register

Name: ADC_IMR
Access: Read-only
Base Address: 0x07C

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
–	–	–	–	TEND	OVR	READY	EOR

- **EOC: End of Conversion**

0: EOC interrupt is disabled.

1: EOC interrupt is enabled.

- **READY: ADC Ready for Conversion**

0: READY interrupt is disabled.

1: READY interrupt is enabled.

- **OVR: Overrun**

0: OVR interrupt is disabled.

1: OVR interrupt is enabled.

- **TEND: End of PDC Transfer**

0: TEND interrupt is disabled.

1: TEND interrupt is enabled.

ADC Convert Data Register

Name: ADC_CDR
Access: Read-only
Base Address: 0x080

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	DATA[9:8]	

7	6	5	4	3	2	1	0
DATA[7:0]							

- **DATA[9:0]: Converted Data**

The result data from an analog to digital conversion is latched into this register at the end of a conversion and remains valid until a new conversion is completed.

When this register is read, the EOC bit in the ADC_SR register is cleared.

Note: When debugging, to avoid clearing the EOC bit, users should use ghost registers (see “Ghost Registers” on page 7).

Universal Synchronous/ Asynchronous Receiver/Transmitter (USART)

Overview

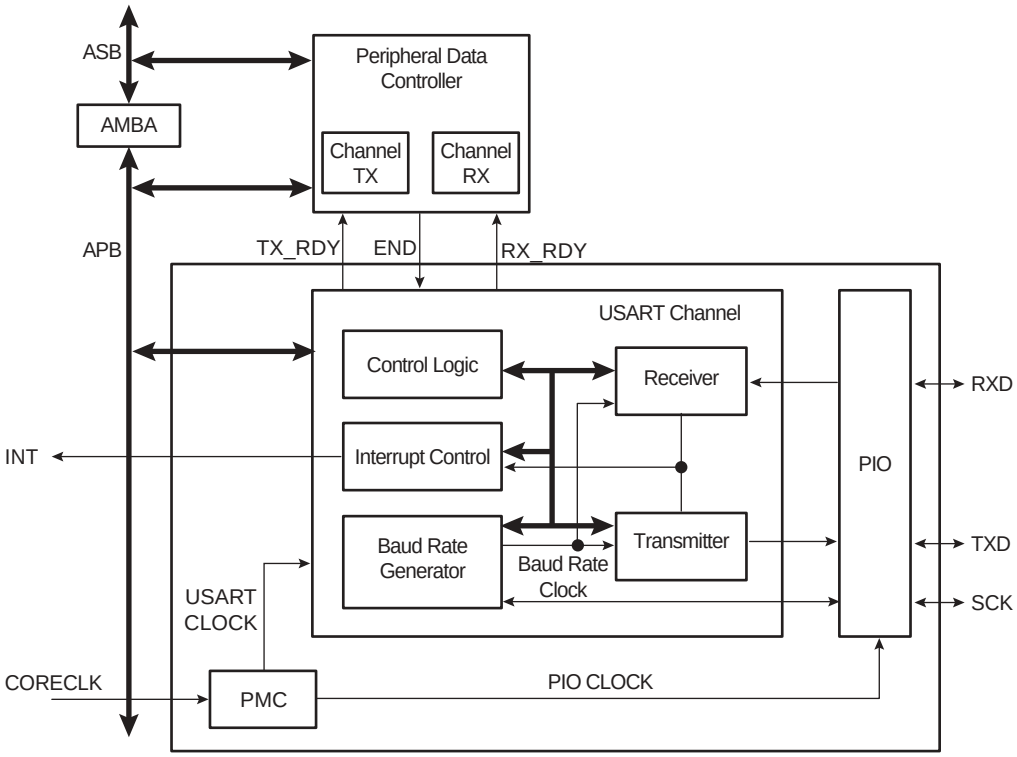
The AT91SAM7A2 includes two USARTs. Each transmitter and receiver module is connected to the Peripheral Data Controller.

The main features are:

- Programmable baud rate generator
- Parity, framing and overrun error detection
- Supports Hardware LIN protocol (specification 1.2)
- Idle flag for J1587 protocol
- Line break generation and detection
- Automatic echo, local loopback and remote loopback channel modes
- Multi-drop mode: address detection and generation
- Interrupt generation
- Two dedicated Peripheral Data Controller channels per USART
- 5 to 9-bit character length

Block Diagram

Figure 49. USART Block Diagram



Baud Rate Generator

The baud rate generator provides the bit period clock, named the baud rate clock, to both the receiver and the transmitter.

The baud rate generator can select between external and internal clock sources. The external clock source is SCK. The internal clock sources can be either CORECLK or CORECLK divided by 8 (CORECLK/8).

Note: In all cases, if an external clock is used, the duration of each of its levels must be longer than the CORECLK period. The external clock frequency must be less than 40% of CORECLK frequency.

When the USART is programmed to operate in asynchronous mode (SYNC = 0 in the Mode Register US_MR), the selected clock is divided by 16 times the value (CD) written in US_BRGR (Baud Rate Generator Register). If US_BRGR is set to 0, the baud rate clock is disabled.

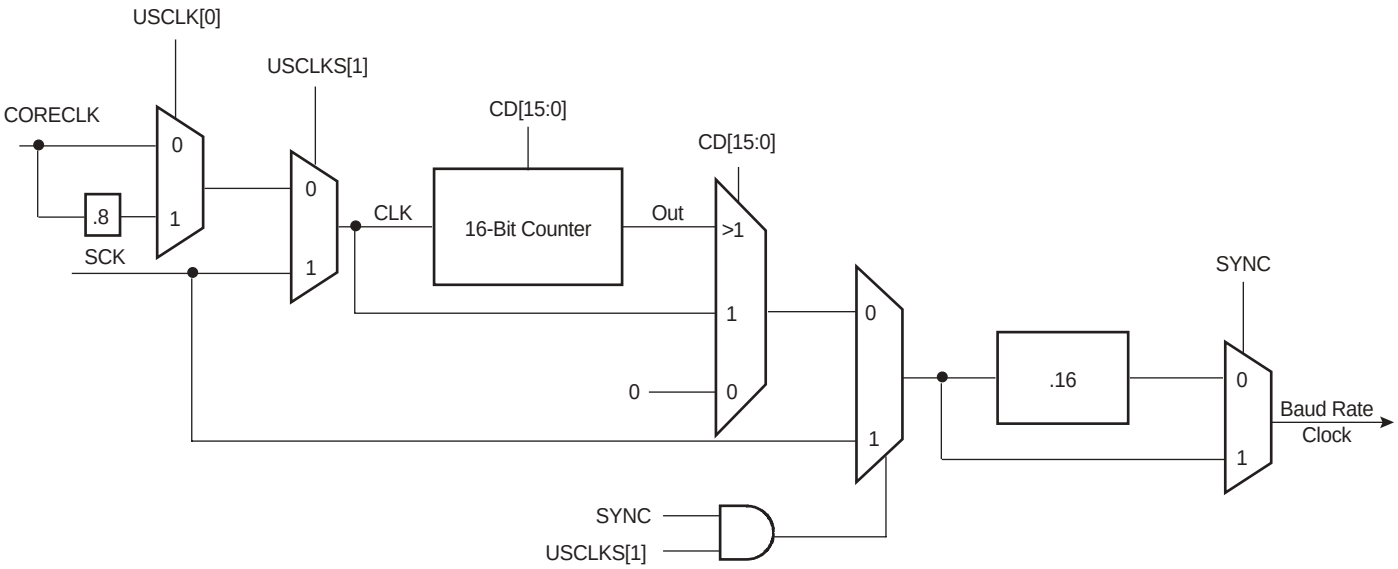
Baud rate = Selected Clock/16 x CD when the selected clock is either CORECLK, CORECLK/8 or SCK.

When the USART is programmed to operate in synchronous mode (SYNC = 1) and the selected clock is internal (USCLKS[1] = 0 in the Mode Register US_MR), the Baud Rate Clock is the internal selected clock divided by the value written in US_BRGR. If US_BRGR is set to 0, the Baud Rate Clock is disabled.

Baud Rate = Selected Clock/CD

In synchronous mode with external clock selected (USCLKS[1] = 1), the clock is provided directly by the signal on the SCK pin. No division is active. The value written in US_BRGR has no effect.

Figure 50. Baud Rate Generator



Receivers

The USART is configured with two receiver operating modes, one for asynchronous operations and the other for synchronous operations.

Asynchronous Receiver

The USART is configured for asynchronous operation when SYNC = 0 (bit 7 of US_MR). In asynchronous mode, the USART detects the start of a received character by sampling the RXD signal until it detects a valid start bit. A low level (space) on RXD is interpreted as a valid start bit if it is detected for more than 7 cycles of the sampling clock, which is 16 times the baud rate. Hence a space which is longer than 7/16 of the bit period is detected as a valid start bit. A space that is 7/16 of a bit period or shorter is ignored and the receiver continues to wait for a valid start bit.

When a valid start bit has been detected, the receiver samples RXD at the theoretical mid-point of each bit. It is assumed that each bit lasts 16 cycles of the sampling clock

(one bit period) so the sampling point is 8 cycles (0.5 bit periods) after the start of the bit. The first sampling point is therefore 24 cycles (1.5 bit periods) after the falling edge of the start bit was detected. Each subsequent bit is sampled 16 cycles (1 bit period) after the previous one.

Figure 51. Asynchronous Mode Start Bit Detection

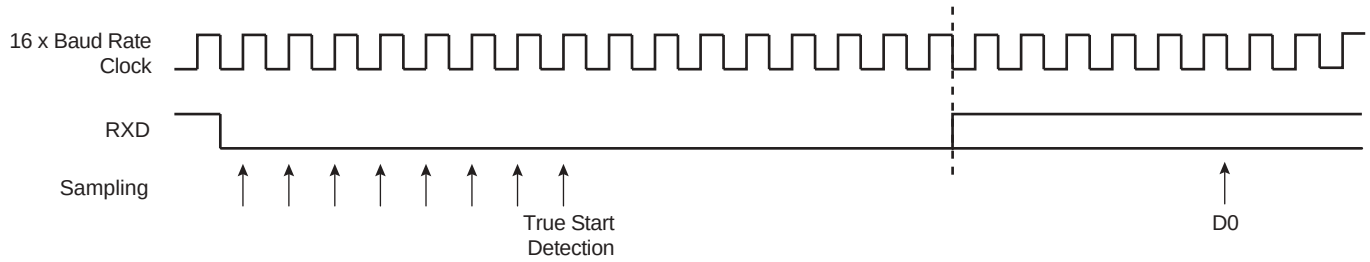
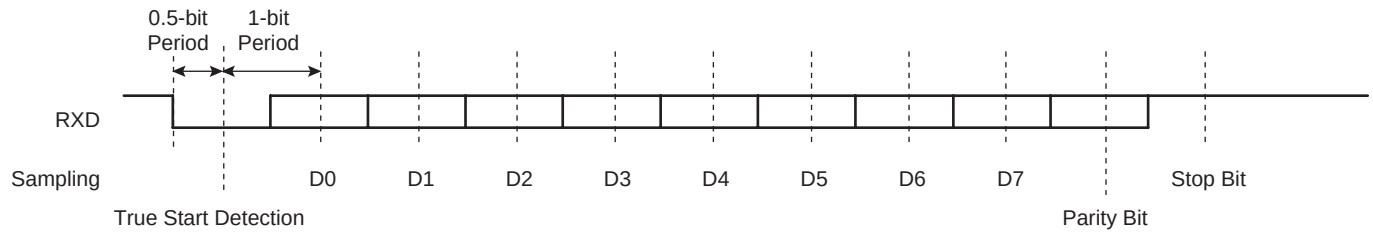


Figure 52. Asynchronous Mode Character Reception

Example: 8-bit parity enabled, 1 stop

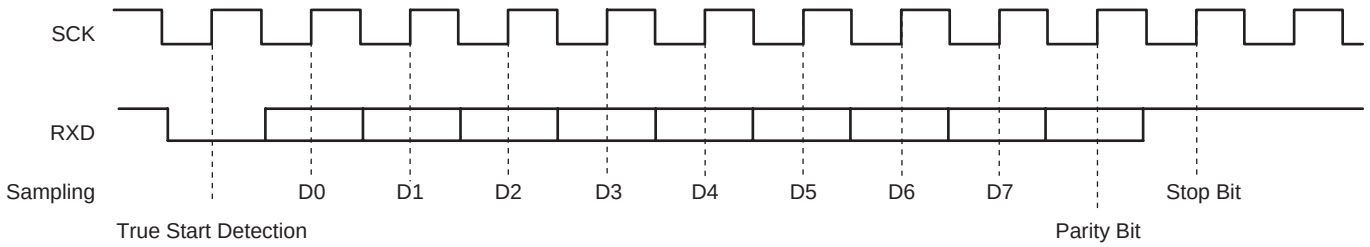


Synchronous Receiver

When configured for synchronous operation (SYNC = 1), the receiver samples the RXD signal on each rising edge of SCK. If a low level is detected, it is considered as a start. Data bits, parity bit and stop bit are sampled and the receiver waits for the next start bit. See the example below.

Figure 53. Synchronous Mode Character Reception

Example: 8-bit parity enabled, 1 stop



Receiver Ready

When a complete character is received, it is transferred to the US_RHR and the RXRDY status bit in US_SR is set. The RXRDY is set after the last stop bit.

If US_RHR has not been read since the last transfer, the USOVRE status bit in US_SR is set.

Parity Error

Each time a character is received, the receiver calculates the parity of the received data bits, in accordance with the PAR field in US_MR. It then compares the result with the received parity bit. If different, the parity error bit PARE in US_SR is set.

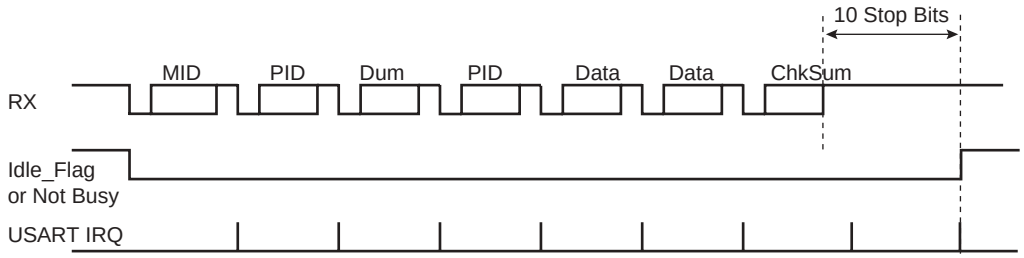
Framing Error

If a character is received with a stop bit at low level and with at least one data bit at high level, a framing error is generated. This sets FRAME in US_SR.

Idle Flag for J1587 Protocol Frame

The idle flag turns low when USART receives a start bit and turns high at the end of a J1587 protocol frame (after 10 stop bits). An interrupt can be generated on the rising edge of the idle flag.

Figure 54. Idle Flag



Time Out

The time out function allows an idle condition on the RXD line to be detected. The maximum delay for which the USART should wait for a new character to arrive while the RXD line is inactive (high level) is programmed in US_RTOR (Receiver Time Out Register). When this register is set to 0, no time out is detected.

Otherwise, the receiver waits for a first character and then initializes a counter which is decremented at each bit period and reloaded at each byte reception. When the counter reaches 0, the TIMEOUT bit in US_SR is set. The user starts (or restarts) the wait for a first character by setting the STTTO (start time out) bit in US_CR.

To start a time out, the following conditions must be met:

- US_RTOR must not be equal to 0.
- The time out must be started by setting STTTO to a logical 1 in the US_CR register.
- One character must be received.

Calculation of time out duration in asynchronous mode:

Duration = Value x 4 x Bit period.

Transmitter

The transmitter has the same behavior in both synchronous and asynchronous operating modes. Start bit, data bits, parity bit and stop bits are serially shifted, least significant bit first, on the falling edge of the serial clock.

The number of data bits is selected in the CHRL field in US_MR.

The parity bit is set according to the PAR field in US_MR.

The number of stop bits is selected in the NBSTOP field in US_MR.

When a character is written to US_THR (Transmit Holding Register), it is transferred to the Shift Register as soon as it is empty.

When the transfer occurs, the TXRDY bit in US_SR is set until a new character is written to US_THR. If the Transmit Shift Register and US_THR are both empty, the TXEMPTY bit in US_SR is set (after the last stop bit of the last transfer).

Time Guard

The Time Guard function allows the transmitter to insert an idle state on the TXD line between two characters. The duration of the idle state is programmed in US_TTGR (Transmitter Time Guard Register). When this register is set to zero, no time-guard is generated. Otherwise, the transmitter holds a high level on TXD after each transmitted byte during the number of bit periods programmed in US_TTGR.

Multi Drop Mode

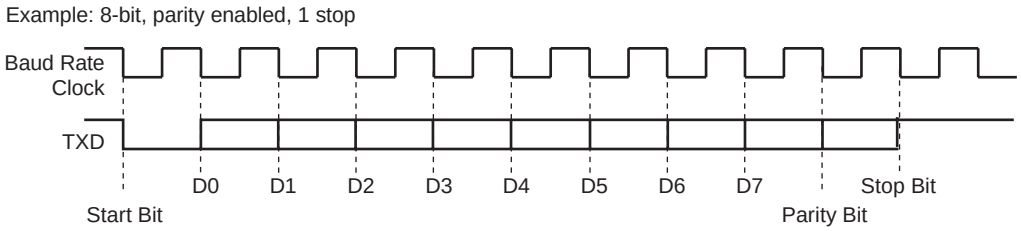
When the PAR field in US_MR equals $11X_b$, the USART is configured to run in multi drop mode. In this case, the parity error bit (PARE in US_SR) is set when data is detected with a parity bit set to identify an address byte. PARE is cleared with the Reset Status Bits Command (RSTSTA) in US_CR. If the parity bit is detected low, identifying a data byte, PARE is not set.

The transmitter sends an address byte (parity bit set) when a Send Address Command (SENDA) is written to US_CR.

In this case, the next byte written to US_THR will be transmitted as an address. After this any byte transmitted will have the parity bit cleared.

Idle state duration between two characters = Time Guard Value x Bit Period

Figure 55. Synchronous and Asynchronous Modes, Character Transmission



Break Condition

Transmit Break

The transmitter can generate a break condition on the TXD line when the STTBRK command is set in US_CR (Control Register). In this case, the characters present in US_THR and in the Transmit Shift Register are completed before the line is held low.

To remove this break condition on the TXD line, the STPBRK command in US_CR must be set. The USART generates a minimum break duration of one character length.

The TXD line then returns to high level (idle state) for at least 12 bit periods to ensure that the end of break is correctly detected. Then the transmitter resumes normal operation.

Receive Break

The break condition is detected by the receiver when all data, parity and stop bits are low (1 character length set to low). At the moment of the low stop bit detection, the receiver asserts the RXBRK bit in US_SR.

End of receive break is detected by a high level for at least 2/16 of the bit period in asynchronous operating mode or at least one sample in synchronous operating mode. RXBRK is also set after end of break has been detected.

Interrupt Generation

Each status bit in US_SR has a corresponding bit in US_IER (Interrupt Enable Register) and US_IDR (Interrupt Disable Register) which controls the generation of interrupts by asserting the USART interrupt line connected to the Generic Interrupt Controller. US_IMR (Interrupt Mask Register) indicates the status of the corresponding bits.

When a bit is set in US_SR and the same bit is set in US_IMR, the interrupt line is asserted.

Channel Modes

The USART can be programmed to operate in three different test modes, using the CHMODE field in US_MR.

Automatic Echo Mode provides bit by bit retransmission. When a bit is received on the RXD line, it is sent to the TXD line. Programming the transmitter has no effect

Local Loopback Mode enables the transmitted characters to be received. TXD and RXD pins are not used and the output of the transmitter is internally connected to the input of the receiver. The RXD pin level has no effect and the TXD pin is held high, as in idle state.

Remote Loopback Mode directly connects the RXD pin to the TXD pin. The Transmitter and the Receiver are disabled and have no effect. This mode provides bit by bit retransmission.

LIN Protocol

This section describes the LIN protocol supported by the USART.

If the LIN is set in the Mode Register, the USART works as a "Master Control Unit" (as it is defined in the LIN Protocol Specification).

It generates the "HEADER FRAME" automatically if the STHEADER bit is set on the Controller Register and the end of header is signaled through the ENDHEADER bit on the Status Register. The header content concerning the "IDENTIFIER" is defined by the Identifier Register (on Read/Write access). The parity is automatically calculated. In the header part, the SYNC_BREAK LENGTH is configurable through the Sync Break Length Register (the SYNC_BREAK can be set from 8 Tbits up to 31Tbits.

Because bit transfer in LIN Protocol depends on the bit rate and because of the large difference between Slave and Master bit rates in LIN specifications, the Tbit represents bit time based on the Master time base. Hence, 1 Tbit = the Master bit rate.

To send a "RESPONSE" (see the LIN Protocol Specification), the STRESP bit is set in the Controller Register, then a part or the totality of Data Field Write 0 Register content and the Data Field Write1 Register content is sent (depending on the value of ID5 and ID4). The CHECKSUM FIELD is also automatically calculated and sent. It is also possible to fill the message response to the THR register. Then if the software wants to use the PDC, it has to wait for the END HEADER interrupt and to start the PDC.

The "MESSAGE RESPONSE" content which has been received by the USART is available on the Data Field Read Register content and the Data Field Read 1 Register even if the USART has filed the "MESSAGE RESPONSE".

The USART is able to detect and to signal the end of "LIN MESSAGE" through the ENDMESS bit in the Status Register. Moreover the USART detects the Bit-Error, the Identifier-Parity-Error, Not-Responding-Error, the Checksum-Error and the Wakeup.

It is not possible to write in the LIN Identifier Register (US_LIR) and in the Data Field Write Register (US_DFWR) during a Message Transfer.

It is also advised not to write on the Transmitter Holding Register (US_THR) during the Header.

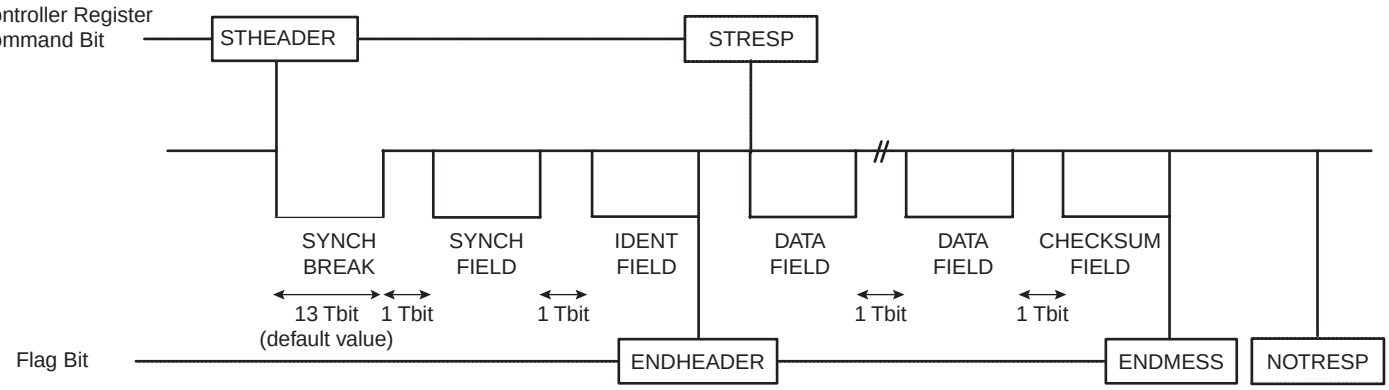
The interbyte space is configurable through the TIME GUARD register, the default interbyte space is one Tbit.

Line Configuration in LIN Mode

To work in LIN mode, the USART has to be set in normal mode (see“USART Mode Register” on page 146). Moreover, when using USART's LIN function, the module must have its transmitter and its receiver enabled.

Message Characteristics

Figure 56. Message Characteristics



In Figure 56 above, unless the master has to file the “Response”, no bits are to be set on the Control Register to wait for the slave’s data field.

If the “Response” is not correctly ended, a time out flag rises 91, 119 or 175 Tbits (depending if Ndata = 2, 4 or 8 bytes) after the HEADER’s beginning.

Smart Card Protocol (ISO7816-3)

The USARTs are ISO7816-3 compliant and allow character repetition or error signaling on parity errors.

Following below is a description of the functions available if the SMCARDPT bit is set on the US_MR register.

Character Transmission to Smart Card

To perform character transmission to a smart card, the transmitter and the receiver have to be enabled.

If the SMCARDPT bit is set in the US_MR register, the USART is able to detect that the smart card has not correctly received the last transmitted byte (parity error signaled by the card).

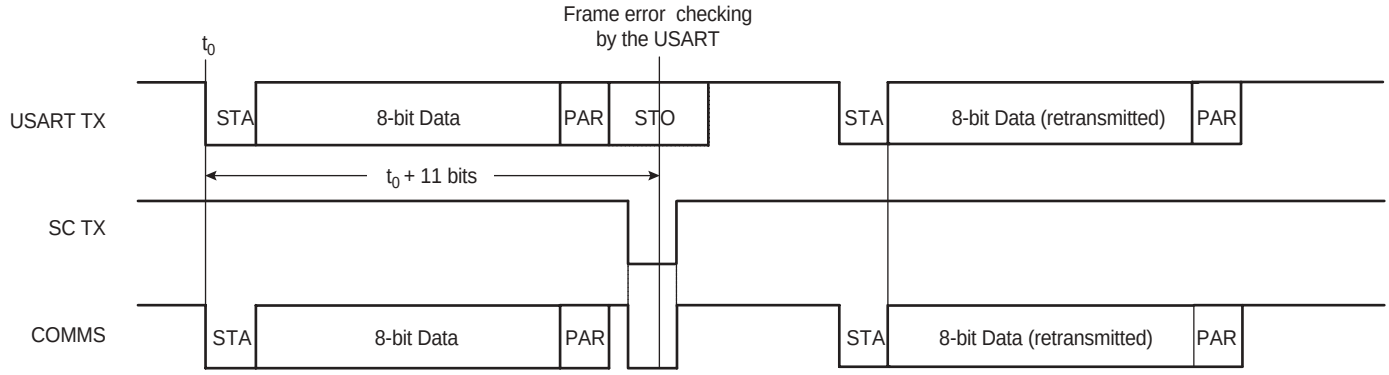
When the card generates the error signal, the last transmitted byte is retransmitted again as many times as determined in the SCREPEAT[1:0] bits of the US_MR register until the error signal is no longer generated by the smart card.

When the error signal is detected by the USART, the PARE and FRAME error flags are set in the US_SR register.

The error signal is checked by the USART at t0 + 11 bits where t_0 is the falling edge of the start bit (i.e. between the two stop bits).

In the example shown Figure 57 on page 132, a parity error is detected by the smart card. The smart card generates the error signal on the COMMS line. The error signal is detected by the USART which retransmits the last character.

Figure 57. Smart Card Transmission Error



If a PDC transfer is used to send a data byte to the card, the PDC counter will not be decremented and the PDC memory pointer will not be incremented until the card has received a correct byte or the maximum repetition time has been reached.

Character Reception From Smart Card

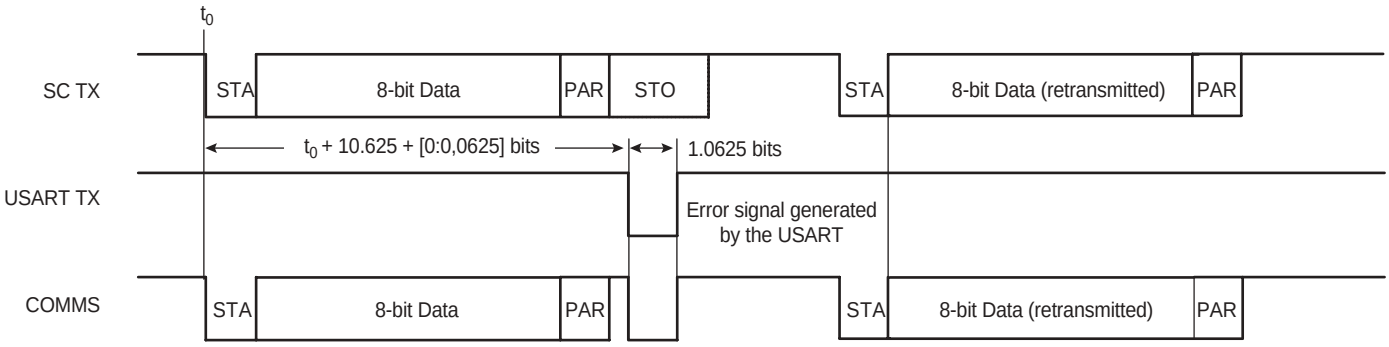
To obtain character reception from a smart card, the receiver has to be enabled and the transmitter has to be disabled.

If the SMCARDPT bit is set in the US_MR register, the USART is able to generate the error signal (see ISO7816-3 protocol) when the last byte received has a parity error.

When a parity error is detected by the USART, the USART transmission line is driven low for 1.0625 bit period starting at t_0 + 10.625 + [0:0.0625] (i.e. during the two stop bits) to indicate to the smart card that a bad reception has occurred on the USART. With T = 0 protocol type smart cards, the smart card must resend the last character.

In that case, the USART signals only a parity error by setting the PARE bit in the US_SR register.

Figure 58. Error Signaling on Reception



If a PDC transfer is used to receive a data byte from the card, the PDC counter will be decremented and the PDC memory pointer will be incremented even when a parity error is detected. The user must reconfigure the PDC memory pointer (decrement by 1) and counter (increment by 1) in order to receive all the remaining bytes.

Line Configuration in Smart Card Mode

To work in Smart Card mode, the USART has to be set in normal mode (see the "USART Mode Register" on page 146).

PIO Controller

Each USART has three programmable I/O lines. These lines are multiplexed with signals (TXD, RXD, SCK) of the USART to optimize the use of the available package pins. These lines are controlled by the USART PIO controller.

Power Management

The USART is provided with a power management block allowing optimization of power consumption (see "Power Management Block" on page 22).

Example

Example USART usage with LIN functionality:

Send a Header Frame and send the Message Frame with an identification of 0x30 with a message of 2x16 bit at 19600 bps using interrupt.

Configuration

- Enable the clock on USART and PIO peripheral by writing the USART bit in US_ECR.
- Do a software reset of the USART peripheral to be in a known state by writing the SWRST bit in US_CR.
- Disable PIO pins RXD, TXD, SCK in US_PDR.
- Configuration of US_BRGR: For 19600 bps field CD = Core clock/(16*19600).
- Enable Transmitter and Receiver bits, RXEN and TXEN, in US_CR.
- Configuration of US_MR: The standard configuration is preferred.
  - No parity, 8 bits, 1 stop bit, normal mode, LIN supported, USART clock is Coreclk
- Configuration of US_SBLR: Configure the syncbreak to 0x13.
- Enter the data to send in DFWR0 and DFWR1.
- Configuration of US_IER: Generate an interrupt at the end of the header and message transmission. This is done by writing the ENDHEADER, ENDMESS. The user can associate interrupts to errors; such as BIT error, parity error, checksum error, etc, (GIC must be configured).
- Configure the identifier by writing 0x30 in US_LIR.
- Enable the LIN header transmission by writing the STHEADER bit in US_CR, this will send the header. Before going to the next step, the user should be informed by the interrupt that the header has been sent.
- Enable the LIN response transmission by writing the STRESP bit in US_CR, this will send the message. Then an interrupt is generated at the end of the transmission.

Interrupt Handling

- IRQ Entry and call C function.
- Read US_SR and verify the source of the interrupt. This register is read and cleared for certain status fields. Care should be taken to keep status information so as to be able to proceed in all cases. Fields that are present in US_CSR are not read and cleared and should be cleared in the next step.
- Clear the corresponding interrupt at peripheral level by writing in the ST_CSR.
- Interrupt treatment: Inform background software that header or message is transmitted.
- IRQ Exit.

USART Memory Map

Base Address USART0: 0xFFFA8000

Base Address USART1: 0xFFFAC000

Table 41. USART Memory Map

Offset	Register	Name	Access	Reset State
0x000	PIO Enable Register	US_PER	Write-only	–
0x004	PIO Disable Register	US_PDR	Write-only	–
0x008	PIO Status Register	US_PSR	Read-only	0x00070000
0x00C	Reserved	–	–	–
0x010	Output Enable Register	US_OER	Write-only	–
0x014	Output Disable Register	US_ODR	Write-only	–
0x018	Output Status Register	US_OSR	Read-only	0x00000000
0x01C – 0x02C	Reserved	–	–	–
0x030	Set Output Data Register	US_SODR	Write-only	–
0x034	Clear Output Data Register	US_CODR	Write-only	–
0x038	Output Data Status Register	US_ODSR	Read-only	0x00000000
0x03C	Pin Data Status Register	US_PDSR	Read-only	0x000X0000
0x040	Multi-Driver Enable Register	US_MDER	Write-only	–
0x044	Multi-Driver Disable Register	US_MDDR	Write-only	–
0x048	Multi-Driver Status Register	US_MDSR	Read-only	0x00000000
0x04C	Reserved	–	–	–
0x050	Enable Clock Register	US_ECR	Write-only	–
0x054	Disable Clock Register	US_DCR	Write-only	–
0x058	Power Management Status Register	US_PMSR	Read-only	0x00000000
0x05C	Reserved	–	–	–
0x060	Control Register	US_CR	Write-only	0x00000000
0x064	Mode Register	US_MR	Read/Write	0x00000000
0x068	Reserved	–	–	–
0x06C	Clear Status Register	US_CSR	Write-only	0x00000000
0x070	Status Register	US_SR	Read-only	0x00000800
0x074	Interrupt Enable Register	US_IER	Write-only	–
0x078	Interrupt Disable Register	US_IDR	Write-only	–
0x07C	Interrupt Mask Register	US_IMR	Read-only	0x00000000
0x080	Receiver Holding Register	US_RHR	Read-only	0x00000000
0x084	Transmitter Holding Register	US_THR	Write-only	–
0x088	Baud Rate Generator Register	US_BRGR	Read/Write	0x00000000
0x08C	Receiver Time-out Register	US_RTOR	Read/Write	0x00000000
0x090	Transmitter Time-guard Register	US_TTGR	Read/Write	0x00000000
0x094	LIN Identifier Register	US_LIR	Read/Write	0x00000000
0x098	Data Field Write0 Register	US_DFWR0	Read/Write	0x00000000
0x09C	Data Field Write 1 Register	US_DFWR1	Read/Write	0x00000000
0x0A0	Data Field Read 0 Register	US_DFRR0	Read-only	0x00000000
0x0A4	Data Field Read 1 Register	US_DFRR1	Read-only	0x00000000
0x0A8	Sync Break Length Register	US_SBLR	Read/Write	0x0000000D

USART PIO Enable Register

Name: US_PER
Access: Write-only
Base Address: 0x000

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

USART PIO Disable Register

Name: US_PDR
Access: Write-only
Base Address: 0x004

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

USART PIO Status Register

Name: US_PSR
Access: Read-only
Base Address: 0x008

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

23	22	21	20	19	18	17	16
–	–	–	–	–	RXD	TXD	SCK

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **SCK: SCK Pin**

0: PIO is inactive on the SCK pin.

1: PIO is active on the SCK pin.

- **TXD: TXD Pin**

0: PIO is inactive on the TXD pin.

1: PIO is active on the TXD pin.

- **RXD: RXD Pin**

0: PIO is inactive on the RXD pin.

1: PIO is active on the RXD pin.

USART PIO Output Enable Register

Name: US_OER
Access: Write-only
Base Address: 0x010

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

USART PIO Output Disable Register

Name: US_ODR
Access: Write-only
Base Address: 0x014

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

USART PIO Output Status Register

Name: US_OSR
Access: Read-only
Base Address: 0x018

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

23	22	21	20	19	18	17	16
–	–	–	–	–	RXD	TXD	SCK

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **SCK: SCK Pin**

0: PIO is an input on the SCK pin.

1: PIO is an output on the SCK pin.

- **TXD: TXD Pin**

0: PIO is an input on the TXD pin.

1: PIO is an output on the TXD pin.

- **RXD: RXD Pin**

0: PIO is an input on the RXD pin.

1: PIO is an output on the RXD pin.

USART PIO Set Output Data Register

Name: US_SODR
Access: Write-only
Base Address: 0x030

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

23	22	21	20	19	18	17	16
–	–	–	–	–	RXD	TXD	SCK

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

USART PIO Clear Output Data Register

Name: US_CODR
Access: Write-only
Base Address: 0x034

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

USART PIO Output Data Status Register

Name: US_ODSR
Access: Read-only
Base Address: 0x038

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

- **SCK: SCK Pin**

0: The output data for the SCK pin is programmed to 0.

1: The output data for the SCK pin is programmed to 1.

- **TXD: TXD Pin**

0: The output data for the TXD pin is programmed to 0.

1: The output data for the TXD pin is programmed to 1.

- **RXD: RXD Pin**

0: The output data for the RXD pin is programmed to 0.

1: The output data for the RXD pin is programmed to 1.

USART PIO Pin Data Status Register

Name: US_PDSR
Access: Read-only
Base Address: 0x03C

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

23	22	21	20	19	18	17	16
–	–	–	–	–	RXD	TXD	SCK

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **SCK: SCK Pin**

0: The output data for the SCK pin is programmed to 0.

1: The output data for the SCK pin is programmed to 1.

- **TXD: TXD Pin**

0: The output data for the TXD pin is programmed to 0.

1: The output data for the TXD pin is programmed to 1.

- **RXD: RXD Pin**

0: The output data for the RXD pin is programmed to 0.

1: The output data for the RXD pin is programmed to 1.

USART PIO Multi Drive Enable Register

Name: US_MDER
Access: Write-only
Base Address: 0x040

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

23	22	21	20	19	18	17	16
–	–	–	–	–	RXD	TXD	SCK

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

USART PIO Multi Drive Disable Register

Name: US_MDDR
Access: Write-only
Base Address: 0x044

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

USART PIO Multi Drive Status Register

Name: US_MDSR
Access: Read-only
Base Address: 0x048

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

- **SCK: SCK Pin**

0: The SCK pin is not configured as an open drain.

1: The SCK pin is configured as an open drain.

- **TXD: TXD Pin**

0: The TXD pin is not configured as an open drain.

1: The TXD pin is configured as an open drain.

- **RXD: RXD Pin**

0: The RXD pin is not configured as an open drain.

1: The RXD pin is configured as an open drain.

USART Enable Clock Register

Name: US_ECR
Access: Write-only
Base Address: 0x050

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
–	–	–	–	–	–	USART	PIO

USART Disable Clock Register

Name: US_DCR
Access: Write-only
Base Address: 0x054

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
–	–	–	–	–	–	USART	PIO

USART Power Management Status Register

Name: US_PMSR
Access: Read-only
Base Address: 0x058

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
–	–	–	–	–	–	USART	PIO

- **PIO: PIO Clock Status**

0: PIO clock disabled.

1: PIO clock enabled.

- **USART: USART Clock Status**

0: USART clock disabled.

1: USART clock enabled.

Note: The US_PMSR register is not reset by software reset.

USART Control Register

Name: US_CR
Access: Write-only
Base Address: 0x060

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

23	22	21	20	19	18	17	16
–	–	–	–	–	–	STRESP	STHEADER

15	14	13	12	11	10	9	8
–	–	–	SENDA	STTO	STPBRK	STTBRK	RSTSTA

7	6	5	4	3	2	1	0
TXDIS	TXEN	RXDIS	RXEN	RSTTX	RSTRX	–	SWRST

- **SWRST: Software Reset**

0: No effect.

1: Reset the USART.

A software triggered hardware reset of the USART is performed. It resets all the registers, including PIO registers (except US_PMSR).

- **RSTRX: Reset Receiver**

0: No effect.

1: The receiver logic is reset.

- **RSTTX: Reset Transmitter**

0: No effect.

1: The transmitter logic is reset.

- **RXEN: Receiver Enable**

0: No effect.

1: The receiver is enabled if RXDIS is 0.

- **RXDIS: Receiver Disable**

0: No effect.

1: The receiver is disabled.

- **TXEN: Transmitter Enable**

0: No effect.

1: The transmitter is enabled if TXDIS is 0.

- **TXDIS: Transmitter Disable**

0: No effect.

1: The transmitter is disabled.

- **RSTSTA: Reset Status Bit**

0: No effect.

1: Resets the status bits PARE, FRAME, USOVRE and RXBRK in US_SR.

- **STTBRK: Start Break**

0: No effect.

1: If break is not being transmitted, start transmission of a break after the characters present in US_THR and the Transmit Shift Register have been transmitted.

- **STPBRK: Stop Break**

0: No effect.

1: If a break is being transmitted, stop transmission of the break after a minimum of one character length and transmit a high level during 12 bit periods.

- **STTTO: Start Time-out**

0: No effect.

1: Start waiting for a character before clocking the time-out counter.

- **SENDA: Send Address**

0: No effect.

1: In Multi-drop Mode only, the next character written to the US_THR is sent with the address bit set.

- **STHEADER: Start Header**

0: No effect.

1: If the LIN bit is set on the Mode Register, send the LIN 's Header Frame.

- **STRESP: Start Response**

0: No effect.

1: Send a part or the Data Field 0 Register content and Data Field 1 Register content.

USART Mode Register

Name: US_MR
Access: Read/Write
Base Address: 0x064

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

23	22	21	20	19	18	17	16
–	–	–	–	–	CLKO	MODE9	SMCARDPT

15	14	13	12	11	10	9	8
CHMODE[1:0]		NBSTOP[1:0]		PAR[2:0]			SYNC

7	6	5	4	3	2	1	0
CHRL[1:0]		USCLKS[1:0]		SENDTIME[1:0]		–	LIN

- **LIN: Local Interconnect Network Mode**

0: USART does not support LIN protocol.

1: USART supports LIN protocol.

- **SENDTIME [1:0]: Send Time**

Indicates the maximum number of times that the USART has to send data in case of a bad reception of the smart card linked to this same USART.

SENDTIME[1:0]		Number of Time
0	0	0
0	1	1
1	0	2
1	1	3

- **·USCLKS[1:0]: Clock Selection (Baud Rate Generator Input Clock)**

USCLKS[1:0]		Selected Clock
0	0	CORECLK
0	1	CORECLK/8
1	X	External (SCK)

- **CHRL[1:0]: Character Length**

Start, stop and parity bits are added to the character length.

CHRL[1:0]		Character Length
0	0	5 bits
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[2:0]: Parity Type**

PAR[2:0]			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	Multi-drop mode

- **NBSTOP[1:0]: Number of Stop Bits**

The interpretation of the number of stop bits depends on SYNC.

NBSTOP[1:0]		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[1:0]: Channel Mode**

CHMODE[1:0]		Mode Description
0	0	Normal Mode: The USART Channel operates as a Rx/Tx USART
0	1	Automatic Echo: Receiver Data Input is connected to TXD pin
1	0	Local Loopback: Transmitter Output Signal is connected to Receiver Input Signal
1	1	Remote Loopback: RXD pin is internally connected to TXD pin

- **SMCARDPT: Smart Card Protocol**

0: The USART is not in smart card protocol.

1: The USART is in smart card protocol.

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

0: CHRL defines character length.

1: 9-Bit character length.

- **CLKO: Clock Output Select**

0: The USART does not drive the SCK pin.

1: The USART drives the SCK pin if CLKS[1] is 0.

USART Clear Status Register

Name: US_CSR
Access: Write-only
Base Address: 0x06C

31	30	29	28	27	26	25	24
–	WAKEUP	CHECKSUM	IPERROR	BITERROR	NOTRESP	ENDMESS	ENDHEADER
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **ENDHEADER: End of Header**

0: No effect.

1: Clear ENDHEADER interrupt.

- **ENDMESS: End of Message**

0: No effect.

1: Clear ENDMESS interrupt.

- **NOTRESP: Not Responding**

0: No effect.

1: Clear NOTRESP interrupt.

- **BITERROR: Bit Error**

0: No effect.

1: Clear BITERROR interrupt.

- **IPERROR: Identity Parity Error**

0: No effect.

1: Clear IPERROR interrupt.

- **CHECKSUM: Check Sum**

0: No effect.

1: Clear CHECKSUM interrupt.

- **WAKEUP: Wake Up**

0: No effect.

1: Clear WAKEUP interrupt.

USART Status Register

Name: US_SR
Access: Read-only
Base Address: 0x070

31	30	29	28	27	26	25	24
–	WAKEUP	CHECKSUM	IPERROR	BITERROR	NOTRESP	ENDMESS	ENDHEADER
23	22	21	20	19	18	17	16
–	–	–	–	–	RXD	TXD	SCK
15	14	13	12	11	10	9	8
–	–	–	–	IDLEFLAG	IDLE	TXEMPTY	TIMEOUT
7	6	5	4	3	2	1	0
PARE	FRAME	USOVRE	ENDTX	ENDRX	RXBRK	TXRDY	RXRDY

Note: This register is a "read-active" register, which means that reading it can affect the state of some bits. When reading US_SR register, the following bits are cleared if set: IDLE, ENDRX, ENDTX, SCK, TXD and RXD. When debugging, to avoid this behavior, users should use ghost registers (see “Ghost Registers” on page 7).

- **RXRDY: Receiver Ready**

0: No complete character has been received since the last read of the US_RHR or the receiver is disabled.

1: At least one complete character has been received and the US_RHR has not yet been read.

- **TXRDY: Transmitter Ready**

0: A character is in US_THR waiting to be transferred to the Transmit Shift Register or the transmitter is disabled.

1: There is no character in US_THR.

Equal to zero when the USART is disabled or at reset. Transmitter Enable command (in US_CR) sets this bit to one.

- **RXBRK: Break Received/End**

0: No Break Received and no End of Break has been detected since the last "Reset Status Bits" command in the Control Register.

1: Break Received or End of Break has been detected since the last "Reset Status Bits" command in the Control Register.

- **ENDRX: End of PDC Receiver Transfer**

0: The End of Transfer signal from the Peripheral Data Controller channel dedicated to the receiver is inactive.

1: The End of Transfer signal from the Peripheral Data Controller channel dedicated to the receiver is active.

- **ENDTX: End of PDC Transmitter Transfer**

0: The End of Transfer signal from the Peripheral Data Controller channel dedicated to the transmitter is inactive.

1: The End of Transfer signal from the Peripheral Data Controller channel dedicated to the transmitter is active.

- **USOVRE: Overrun Error**

0: No byte has been transferred from the Receive Shift Register to the US_RHR when RxRDY was asserted since the last "Reset Status Bits" command.

1: At least one byte has been transferred from the Receive Shift Register to the US_RHR when RxRDY was asserted since the last "Reset Status Bits" command.

- **FRAME: Framing Error**

0: No stop bit has been detected low since the last "Reset Status Bits" command.

1: At least one stop bit has been detected low since the last "Reset Status Bits" command.

- **PARE: Parity Error**

0: No parity bit has been detected false (or a parity bit high in multi-drop mode) since the last "Reset Status Bits" command.

1: At least one parity bit has been detected false (or a parity bit high in multi-drop mode) since the last "Reset Status Bits" command.

- **TIMEOUT: Receiver Time Out**

0: There has not been a time-out since the last "Start Time-out" command or the Time-out Register is 0.

1: There has been a time-out since the last "Start Time-out" command.

- **TXEMPTY: Transmitter Empty**

0: There are characters in either US_THR or the Transmit Shift Register.

1: There are no characters in either US_THR or the Transmit Shift Register.

Equal to zero when the USART is disabled or at reset. Transmitter Enable command (in US_CR) sets this bit to one.

- **IDLE: Idle Interrupt**

0: No end of J1587 protocol frame since last read of the US_SR register.

1: An end of J1587 protocol frame occurred since last read of the US_SR register.

- **IDLEFLAG: Idle Flag**

0: A frame is being received by the USART.

1: No frame is being received by the USART.

This bit indicates a frame transmission in J1587 protocol. It's turned low when a reception starts and turned high when a reception is followed by at least 10 stop bits (10 bits at high level).

- **SCK, TXD, RXD: PIO Interrupt Status**

These bits indicate for each pin when an input logic value change has been detected (rising or falling edge). This is valid whether the PIO is selected for the pin or not.

These bits are reset to zero following a read and at reset.

0: No input change has been detected on the corresponding pin since the register was last read.

1: At least one input change has been detected on the corresponding pin since the register was last read.

- **ENDHEADER: End of Header**

0: No end of Header has occurred on a LIN Frame.

1: An end of Header has occurred on a LIN Frame.

- **ENDMESS: End of Message**

0: No end of Message occurred on a LIN Frame.

1: An end of Message has occurred on a LIN Frame.

- **NOTRESP: Not Responding**

0: No Slave-not-responding-error has been detected LIN Frame.

1: A Slave-not-responding-error has been detected LIN Frame.

- **BITERROR: Bit Error**

0: No Bit-error has been detected on a LIN Frame.

1: A Bit-error has been detected on a LIN Frame.

- **IPERROR: Identity Parity Error**

0: No Identity-parity-Error has been detected on a LIN Frame.

1: An Identity-parity-error has been detected on a LIN Frame.

- **CHECKSUM: Check Sum**

0: No Checksum-error has been detected LIN Frame.

1: A Checksum-error has been detected LIN Frame.

- **WAKEUP: Wake Up**

0: No Wakeup has been detected.

1: A Wakeup has been detected.

USART Interrupt Enable Register

Name: US_IER
Access: Write-only
Base Address: 0x074

31	30	29	28	27	26	25	24
–	WAKEUP	CHECKSUM	IPERROR	BITERROR	NOTRESP	ENDMESS	ENDHEADER
23	22	21	20	19	18	17	16
–	–	–	–	–	RXD	TXD	SCK
15	14	13	12	11	10	9	8
–	–	–	–	–	IDLE	TXEMPTY	TIMEOUT
7	6	5	4	3	2	1	0
PARE	FRAME	USOVRE	ENDTX	ENDRX	RXBRK	TXRDY	RXRDY

USART Interrupt Disable Register

Name: US_IDR
Access: Write-only
Base Address: 0x078

31	30	29	28	27	26	25	24
–	WAKEUP	CHECKSUM	IPERROR	BITERROR	NOTRESP	ENDMESS	ENDHEADER
23	22	21	20	19	18	17	16
–	–	–	–	–	RXD	TXD	SCK
15	14	13	12	11	10	9	8
–	–	–	–	–	IDLE	TXEMPTY	TIMEOUT
7	6	5	4	3	2	1	0
PARE	FRAME	USOVRE	ENDTX	ENDRX	RXBRK	TXRDY	RXRDY

USART Interrupt Mask Register

Name: US_IMR
Access: Read-only
Base Address: 0x07C

31	30	29	28	27	26	25	24
–	WAKEUP	CHECKSUM	IPERROR	BITERROR	NOTRESP	ENDMESS	ENDHEADER
23	22	21	20	19	18	17	16
–	–	–	–	–	RXD	TXD	SCK
15	14	13	12	11	10	9	8
–	–	–	–	IDLEFLAG	IDLE	TXEMPTY	TIMEOUT
7	6	5	4	3	2	1	0
PARE	FRAME	USOVRE	ENDTX	ENDRX	RXBRK	TXRDY	RXRDY

- **RXRDY: Mask RXRDY Interrupt**

0: RXRDY Interrupt is Disabled.

1: RXRDY Interrupt is Enabled.

- **TXRDY: Mask TXRDY Interrupt**

0: TXRDY Interrupt is Disabled.

1: TXRDY Interrupt is Enabled.

- **RXBRK: Mask Receiver Break Interrupt**

0: RXBRK Interrupt is Disabled.

1: RXBRK Interrupt is Enabled.

- **ENDRX: Mask End of PDC Receive Transfer Interrupt**

0: ENDRX Interrupt is Disabled.

1: ENDRX Interrupt is Enabled.

- **ENDTX: Mask End of PDC Transmit Transfer Interrupt**

0: ENDTX Interrupt is Disabled.

1: ENDTX Interrupt is Enabled.

- **USOVRE: Mask Overrun Error Interrupt**

0: USOVRE Interrupt is Disabled.

1: USOVRE Interrupt is Enabled.

- **FRAME: Mask Framing Error Interrupt**

0: FRAME Interrupt is Disabled.

1: FRAME Interrupt is Enabled.

- **PARE: Mask Parity Error Interrupt**

0: PARE Interrupt is Disabled.

1: PARE Interrupt is Enabled.

- **TIMEOUT: Mask Time Out Interrupt**

0: TIMEOUT Interrupt is Disabled.

1: TIMEOUT Interrupt is Enabled.

- **TXEMPTY: Mask TXEMPTY Interrupt**

0: TXEMPTY Interrupt is Disabled.

1: TXEMPTY Interrupt is Enabled.

- **IDLE: Mask IDLE Interrupt**

0: IDLE Interrupt is Disabled.

1: IDLE Interrupt is Enabled.

- **SCK, TXD, RXD: PIO Interrupt Mask**

These bits show which pins have interrupts enabled. They are updated by writing to US_IER or US_IDR.

0: Interrupt is not enabled on the corresponding input pin.

1: Interrupts is enabled on the corresponding input pin.

- **ENDHEADER: Enable ENDHEADER Interrupt**

0: No effect.

1: Enables ENDHEADER interrupt.

- **ENDMESS: Enable ENDMESS Interrupt**

0: No effect.

1: Enables ENDMESS Interrupt.

- **NOTRESP: Enable NOTRESP Interrupt**

0: No effect.

1: Enables NOTRESP Interrupt.

- **BITERROR: Enable BITERRROR Interrupt**

0: No effect.

1: Enables BITERROR Interrupt.

- **IPERROR: Enable IPERROR Interrupt**

0: No effect.

1: Enables IPERROR Interrupt.

- **CHECKSUM: Enable CHECKSUM Interrupt**

0: No effect.

1: Enables CHECKSUM Interrupt.

- **WAKEUP: Enable WAKEUP Interrupt**

0: No effect.

1: Enables WAKEUP Interrupt.

USART Receiver Holding Register

Name: US_RHR
Access: Read-only
Base Address: 0x080

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	RXCHR[8]

7	6	5	4	3	2	1	0
RXCHR[7:0]							

- **RXCHR[8:0]: Received Character**

Last character received if RXRDY is set. When the number of data bits is less than 9 bits, the bits are right aligned.

Notes: 1. When reading this register, RXRDY bit is clear in the US_RHR.
2. When debugging, to avoid clearing RXRDY bit, users should use ghost registers (see “Ghost Registers” on page 7).

USART Transmit Holding Register

Name: US_THR
Access: Write-only
Base Address: 0x084

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	TXCHR[8]

7	6	5	4	3	2	1	0
TXCHR[7:0]							

- **TXCHR[8:0]: Character to be Transmitted**

Next character to be transmitted after the current character if TXRDY is not set. When the number of data bits is less than 9 bits, the bits are right aligned.

USART Baud Rate Generator Register

Name: US_BRGR
Access: Read/Write
Base Address: 0x088

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

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

15	14	13	12	11	10	9	8
CD[15:8]							

7	6	5	4	3	2	1	0
CD[7:0]							

- **CD[15:0]: Clock divisor**

This register has no effect if synchronous mode is selected with an external clock. (1) (2)

CD[15:0]	Action
0	Disables clock
1	Clock divider bypassed
2 to 65535	Baud Rate (Asynchronous Mode) = Selected clock/(16 x CD) Baud Rate (Synchronous Mode) = Selected clock/CD

Notes: 1. In synchronous mode, the value programmed must be even to ensure a 50:50 mark/space ratio.
2. CD = 1 must not be used when internal clock (CORECLK) is selected (i.e USCLKS[1:0] = 00b).

USART Receiver Time Out Register

Name: US_RTOR
Access: Read/Write
Base Address: 0x08C

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

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

15	14	13	12	11	10	9	8
TO[15:8]							

7	6	5	4	3	2	1	0
TO[7:0]							

- **TO[15:0]: Time Out Value**

O[7:0]	Action
0	Disables the RX Time-out function.
1-65535	The Time-out counter is loaded with TO[15:0] when the Start Time-out Command is given or when each new data character is received (after reception has started).

In asynchronous mode:

Time out duration = TO[15:0] x Bit period.

In synchronous mode:

Time out duration = TO[15:0] x 4 x Bit period.

When the receiver is disabled by setting the RXDIS bit in the US_CR register, the time out is stopped. If the receiver is re-enabled by setting the RXEN bit in the US_CR register, the time out restarts where it left off (i.e. it is not reset).

USART Transmit Time Guard Register

Name: US_TTGR
Access: Read/Write
Base Address: 0x090

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
TO[7:0]							

- **TG[7:0]: Time Guard Value**

TG[7:0]	Action
0	Disables the TX Time Guard function.
1-255	TXD is inactive high after the transmission of each character for the Time Guard duration.

Time Guard duration = TG[7:0] x Bit period.

USART LIN Identifier Register

Name: US_LIR
Access: Read/Write
Base Address: 0x094

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
–	–	IDENTIFIER[5:0]					

- **IDENTIFIER [5:0]: LIN 's IDENTIFIER**

Indicates the LIN 's Identifier Content to be transmitted on the next "HEADER MESSAGE".

USART Data Field Write 0 Register

Name: US_DFWR0
Access: Read/Write
Base Address: 0x098

31	30	29	28	27	26	25	24
DATA3[7:0]							
23	22	21	20	19	18	17	16
DATA2[7:0]							
15	14	13	12	11	10	9	8
DATA1[7:0]							
7	6	5	4	3	2	1	0
DATA0[7:0]							

- **DATAx [7:0]: LIN 's BYTE FIELD to be Transmitted**

Data to be transmitted on the "RESPONSE 's LIN MESSAGE" when STRESP is set on the Controller Register.

USART Data Field Write 1 Register

Name: US_DFWR1
Access: Read/Write
Base Address: 0x09C

31	30	29	28	27	26	25	24
DATA7[7:0]							
23	22	21	20	19	18	17	16
DATA6[7:0]							
15	14	13	12	11	10	9	8
DATA5[7:0]							
7	6	5	4	3	2	1	0
DATA4[7:0]							

- **DATAx [7:0]: LIN 's BYTE FIELD to be Transmitted**

Data to be transmitted on the "RESPONSE 's LIN MESSAGE" when STRESP is set on the Controller Register.

USART Data Field Read 0 Register

Name: US_DFRR0
Access: Read/Write
Base Address: 0x0A0

31	30	29	28	27	26	25	24
DATA3[7:0]							
23	22	21	20	19	18	17	16
DATA2[7:0]							
15	14	13	12	11	10	9	8
DATA1[7:0]							
7	6	5	4	3	2	1	0
DATA0[7:0]							

- **DATAx [7:0]: LIN 's BYTE FIELD to be Received**

Data read on the latest "RESPONSE 's LIN MESSAGE".

USART Data Field Read 1 Register

Name: US_DFRR1
Access: Read/Write
Base Address: 0x0A4

31	30	29	28	27	26	25	24
DATA7[7:0]							
23	22	21	20	19	18	17	16
DATA6[7:0]							
15	14	13	12	11	10	9	8
DATA5[7:0]							
7	6	5	4	3	2	1	0
DATA4[7:0]							

- **DATAx [7:0]: LIN 's BYTE FIELD to be Received**

Data read on the latest "RESPONSE 's LIN MESSAGE".

USART Sync Break Length Register

Name: US_SBLR
Access: Read/Write
Base Address: 0x0A8

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
–	–	–	SYNCH_BRK				

W: Write R: Read -0: 0 after reset -1: 1 after reset -U: undefined after reset

- **SYNC_BRK [4:0]: Sync Break Length**

Configures sync break field length.

If SYNC_BRK is less than 8, it remains in previous state.

Capture (CAPT)

Overview

The AT91SAM7A2 provides a Capture module that serves as a frame analyzer. It stores the duration period (high and low level) in the CAPTURE Data Register (CAPT_DR); these durations are described as a number of counter cycles (CAPTCLK). The Capture peripheral provides data transfer with the PDC.

Capture can detect all frame variations (with an error of one CAPTCLK period maximum) if the input signal has a frequency less than CAPTCLK/2.

It is possible to choose among three modes of measurement:

- Duration between both edges (positive and negative).
- Duration between positive edges.
- Duration between negative edges.

It is important to check that the frame to be watched by the capture module is not too fast, otherwise the PDC can monopolize the ASB bus as the PDC, in this case, will often have something to read on the Capture Data Register (CAPT_DR).

If an overrun occurs, it is possible to choose to overwrite the data stored in the Data Register (CAPT_DR) or to stop the data acquisition through the mode register (CAPT_MR).

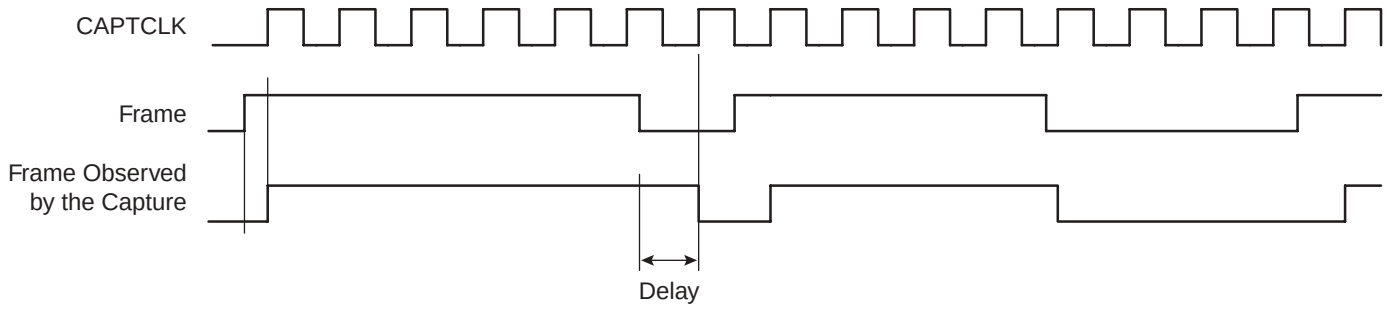
After a read of the data register, the DATACAPT bit (in the CAPT_SR register) is automatically cleared (i.e. set to a logical 0).

It is recommended to disable the capture module after every modification of the CAPT_MR register, otherwise the first measurement can be false.

When Capture is disabled, the capture counter is reset.

Example of Use

Figure 59. Capture Pin Resynchronization



Example Use of CAPT

Capture of one period of a signal with a core clock of 30 MHz using the PDC and the associated interrupt. The signal frequency should be below 7.5 MHz and above 228 Hz. The period measurement will be between two positive edges.

Configuration

- Enable the clock on CAPT peripheral by writing the CAP bit in CAPT_ECR.
- Do a software reset of the capture peripheral to be in a known state by writing the SWRST bit in CAPT_CR.
- Configuration of CAPT_ MR: Select the CAPT clock to be 15 MHz with the PRESCALAR field set to 0, OVERMODE is ignored here as one capture is done and

ONESHOT is set. The measurement is described by MEASMODE and the value 0x2 is chosen for this field (between two positive edges).

- Enable the CAPT by writing the CAPEN bit in CAPT_CR.
- Configuration of CAPT_IER: An interrupt should be generated at the end of the capture. This is done by writing PDCEND. When the PDC finishes the capture, an interrupt will be generated. GIC must be configured.
- Configure PDC for CAPT module and set it to 1 capture of 16 bits.
- If the status bit CAPENS is set in CAPT_SR, start the capture by writing the STARTCAPT bit in CAPT_CR, otherwise users should wait the CAPENS flag (consequence of the CAPEN bit in CAPT _CR).

Interrupt Handling

- IRQ Entry and call C function.
- Read CAPT _SR and verify the source of the interrupt.
- Clear the corresponding interrupt at peripheral level by writing in CAPT _CSR.
- Interrupt treatment: Read the duration between the two positive edges in the received memory space programmed in the PDC. The duration is expressed by the number of capture clocks (duration/15 MHz).
- IRQ Exit.

Capture Limits

To prevent the capture from missing frame edges it is important to check that:

Frame frequency = CAPTCLK/2

Moreover, the capture detects each frame edge with a delay equal to the CAPTCLK period.

The CAPTCLK frequency can take a value between 15 MHz and 916 Hz (if 30 MHz is the CORECLK frequency).

Table 42. Capture Delay and Error (Example)

CAPTCLK Frequency	Fattest Observable Frame	Delay Max for Edge Detection	Error Max in Level Duration Value
F	F/2 if F< CORECLK frequency F/4 if F= CORECLK frequency	1/F	2/F
30 MHz	7.5 MHz	33 ns	66 ns
916 Hz	458 Hz	1.1 ms	2.2 ms

Power Management

Capture is provided with a power management block allowing optimization of power consumption. See “Power Management Block” on page 22.

Capture (CAPT) Memory Map

Base Address CAPT0: 0xFFFDC000

Base Address CAPT1: 0xFFFE0000

Table 43. CAPT Memory Map

Offset	Register	Name	Access	Reset State
0x000 – 0x04C	Reserved	–	–	–
0x050	Enable Clock Register	CAP_ECR	Write-only	–
0x054	Disable Clock Register	CAP_DCR	Write-only	–
0x058	Power Management Status Register	CAP_PMSR	Read-only	0x00000000
0x05C	Reserved	–	–	–
0x060	Control Register	CAP_CR	Write-only	–
0x064	Mode Register	CAP_MR	Read/Write	0x00000000
0x068	Reserved	–	–	–
0x06C	Clear Status Register	CAP_CSR	Write-only	–
0x070	Status Register	CAP_SR	Read-only	0x00000000
0x074	Interrupt Enable Register	CAP_IER	Write-only	–
0x078	Interrupt Disable Register	CAP_IDR	Write-only	–
0x07C	Interrupt Mask Register	CAP_IMR	Read-only	0x00000000
0x080	Data Register	CAP_DR	Read-only	0x00000000

CAPTURE Enable Clock Register

Name: CAPT_ECR
Access: Write-only
Base Address: 0x050

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
–	–	–	–	–	–	CAP	–

CAPTURE Disable Clock Register

Name: CAPT_DCR
Access: Write-only
Base Address: 0x054

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
–	–	–	–	–	–	CAP	–

CAPTURE Power Management Status Register

Name: CAPT_PMSR
Access: Read-only
Base Address: 0x058

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
–	–	–	–	–	–	CAP	–

- **CAP: CAPTURE Clock**

0: Capture clock disabled.

1: Capture clock enabled.

Note: The CAPT_PMSR register is not reset by a software reset.

CAPTURE Control Register

Name: CAPT_CR
Access: Write-only
Base Address: 0x060

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

- **SWRST: Capture Software Reset**

0: No effect.

1: Resets the Capture.

A software reset of the Capture is performed. It resets all the registers (except CAPT_PMSR).

- **CAPEN: Capture Enable**

0: No effect.

1: Enables the Capture.

- **CAPDIS: Capture Disable**

0: No effect.

1: Disables the Capture.

If both CAPEN and CAPDIS are equal to one when the control register is written, the CAPTURE will be disabled.

- **STARTCAP: Start Capture**

0: No effect.

1: The Capture starts a new Capture.

CAPTURE Mode Register

Name: CAPT_MR
Access: Read/Write
Base Address: 0x064

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
ONESHOT	OVERMODE	MEASMODE[1:0]		PRESCLAR[3:0]			

- **PRESCALR[3:0]: Counter CLock Prescalar**

CAPTCLK Frequency = $2^{PRESCALAR+1}$

MEASMODE[1:0]: Measurement Mode

MEASMODE[1:0]		Measure
0	X	Measure between each edge (positive and negative).
1	0	Measure between positive edges.
0	1	Measure between negative edges.

- **OVERMODE: Overrun Mode**

0: In case of an overrun the capture stops writing on the Data Register. If DATACAPT bit is enabled and the CAPTURE module receives new data, the data register will not be refreshed.

1: In case of an overrun the capture does not stop writing on the Data Register.

- **ONESHOT: One Shot**

0: The capture still captures a frame variation.

1: The module captures a frame variation and stops. To ask for another capture, the STARTCAPT bit has to be set at 1 in CAPT_CR.

CAPTURE Clear Status Register

Name: CAPT_CSR
Access: Write-only
Base Address: 0x06C

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
–	–	–	–	–	CAPDIS	CAPEN	SWRST

- **Clear PDCEND Interrupt**

0: No effect.

1: Clears the PDCEND interrupt.

- **OVERRUN: Clear Overrun Interrupt**

0: No effect.

1: Clears the OVERRUN interrupt.

- **OVERFLOW: Clear Overflow Interrupt**

0: No effect.

1: Clears the OVERFLOW interrupt.

CAPTURE Status Register

Name: CAPT_SR
Access: Read-only
Base Address: 0x070

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	CAPENS

7	6	5	4	3	2	1	0
–	–	–	–	DATACAPT	OVERFLOW	OVERRUN	PDCEND

- **PDCEND: PDC end**

0: No effect.

1: The PDC has finished the data transfer.

- **OVERRUN: Overrun**

0: No effect.

1: An overrun has occurred.

Overrun indicates a valid data was not read when an overwrite occurred.

- **OVERFLOW: Overflow**

0: No effect.

1: An overflow has occurred.

Overflow indicates that the counter of the duration has been saturated.

- **DATACAPT: Data Captured**

0: No effect.

1: Data in CAP_DR has to be read.

This bit is cleared by reading the CAP_DR register.

- **CAPENS: Capture Enable Status**

0: Capture is disabled.

1: Capture is enabled.

CAPTURE Interrupt Enable Register

Name: CAPT_IER
Access: Write-only
Base Address: 0x074

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
–	–	–	–	DATACAPT	OVERFLOW	OVERRUN	PDCEND

CAPTURE Interrupt Disable Register

Name: CAPT_IDR
Access: Write-only
Base Address: 0x078

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
–	–	–	–	DATACAPT	OVERFLOW	OVERRUN	PDCEND

CAPTURE Interrupt Mask Register

Name: CAPT_IMR
Access: Write-only
Base Address: 0x07C

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
–	–	–	–	DATACAPT	OVERFLOW	OVERRUN	PDCEND

- **PDCEND: PDC End Interrupt Mask**

0: PDCEND Interrupt is disabled.

1: PDCEND Interrupt is enabled.

- **OVERRUN: Overrun Interrupt Mask**

0: OVERRUN Interrupt is disabled.

1: OVERRUN interrupt is enabled.

- **OVERFLOW: Overflow Interrupt Mask**

0: OVERFLOW Interrupt is disabled.

1: OVERFLOW interrupt is enabled.

- **DATACAPT: Data Capture Interrupt Mask**

0: DATACAPT interrupt is disabled.

1: DATACAPT interrupt is enabled.

CAPTURE Data Register

Name: CAPT_DR(1)(2)
Access: Read/Write
Base Address: 0x064

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

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

15	14	13	12	11	10	9	8
LEVEL	DURATION[14:8]						

7	6	5	4	3	2	1	0
DURATION[7:0]							

Notes: 1. When reading this register, the DATACAPT bit is clear in CAPT_SR
2. When debugging, to avoid clearing the DATACAPT bit, users should use ghost registers (see “Ghost Registers” on page 7).

- **DURATION[14:0]: Capture Duration**

Numbers of CAPTCLK cycles during which the output is at the level indicated by the LEVEL bit, or during a frame period, depending of MEASMODE in CAP_MR.

- **LEVEL: Level measured**

0: the duration concerns a low level.

1: the duration concerns a high level.

Note that if the MEASMODE[1:0] (CAPT_MR) equals 1X, the LEVEL bit is used as a duration bit.

Simple Timer (ST)

Overview

The AT91SAM7A2 microcontroller includes two 16-bit Simple Timers (ST0 and ST1) with two channels per simple timer.

Each simple timer channel provides basic functions for timing calculation including two cascaded dividers and a 16-bit counter.

The prescalar defines the clock frequency of the channel counter.

For each channel it is possible to select the divider clock between the core clock (CORECLK) and the low frequency clock (LFCLK).

The 16-bit counter starts down-counting when a value different from zero is loaded. An interrupt is generated when the counter reaches 0x0000.

When a value is loaded in the LOAD field of the STx_CTz register and the channel is started, the counter starts down-counting at channel clock frequency until the counter reaches zero. The delay between the load and the interrupt is the counter value multiplied by the clock period.

The counter value is the value loaded in the counter. The clock period is the period of the channel clock (divided by the prescalar).

The precision is one clock period (from 0 to 1 channel clock period lost).

Note: When enabling or disabling the Simple Timer, software must wait for enabled or disabled interrupt (or status) to be sure that the counter is really enabled or disabled (it is asynchronously clocked).

It is not possible to change the Channel x Counter Register content when the Simple Timer is enabled.

If a channel is disabled before it reaches the end of the down-counting, the counter value is held. If no write has occurred on the Counter Register before it is re-enabled, the down-counting restarts from the latest counter value.

When the core clock (CORECLK) is selected on the Prescalar Register, the clock is divided twice to obtain the counter clock: firstly by a divider driven by the SYSCAL bits of the Prescalar Register and finally by a divider driven by the prescalar bits of the Prescalar Register.

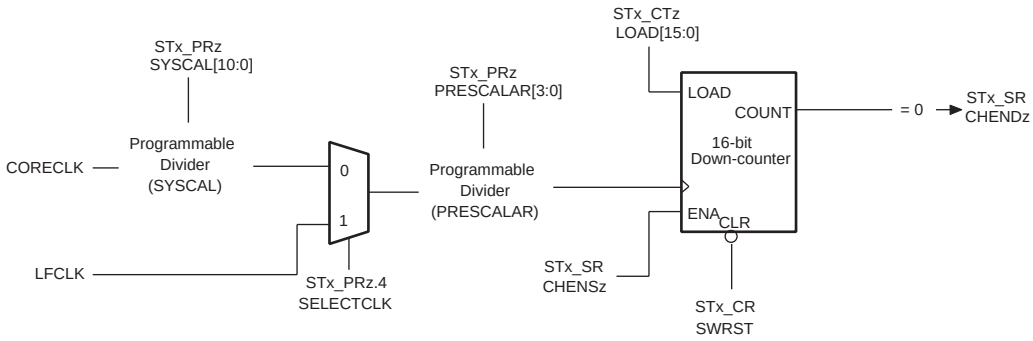
However, if the low frequency clock (LFCLK) is selected, the clock is divided just once to obtain the counter clock by a divider driven by the prescalar bits of the Prescalar Register.

The simple timer also integrates an automatic reload function if the AUTOREL bit is set on the corresponding STx_PRz register. When the counter reaches 0x0000, it is automatically reloaded with the value written in the LOAD[15:0] bits of the STx_CTz registers (x for ST0 or ST1 and z for channel 0 or channel 1).

The current value of the counter can be read in the corresponding ST_CCVx register.

Block Diagram

Figure 60. Simple Timer Block Diagram



Peripheral Structure

Interrupt Description

For each channel there are three types of interrupt:

- A CHDISz interrupt which indicates that the channel has been disabled.
- A CHLOADz interrupt which indicates that the channel has started to down count and loaded the data.
- A CHENDz interrupt which indicates that the channel has reach the end of the down-counting. This interrupt rises 3 CORECLK cycles after the down counter reaches the value 0x0000.

where z = is the channel number.

Power Management

Each Simple Timer (ST0 and ST1) is provided with a power management block allowing optimization of power consumption (see “Power Management Block” on page 22). The PMC controls the clock inputs and the clock divider blocks.

When the timer is stopped, the clock is immediately stopped. When the clock is re-enabled, the timer resumes action where it left off.

Example of Use

This section gives an example of the use of the Simple Timer, explaining how to generate a tick (usually used in RTOS) of 10 ms with auto-reload and an interrupt.

Core clock is 30 MHz.

Configuration

- Enable the clock on the Simple Timer (ST) by writing bit ST in ST _ECR.
- Do a software reset of the peripheral so that it is in a known state by writing SWRST bit in ST_CR. Wait about four LFCLK period for the circuitry to become stable.
- Configuration of ST_PRX: A tick should occur every 10 ms, thus giving a frequency of 100 Hz; e.g., the counter frequency is 15 kHz and is decremented by 150. To produce a counter frequency of 15 kHz, calculate

$$\text{Divider clock} = \frac{\text{CORECLK}}{2\ \text{SYSCAL}} + 1$$

with SYSCAL = 0 and SELCETCLK = 0, divider clock = 15 MHz. Then with a PRESCALAR of 10,

$$\text{Counter clock} = \frac{\text{dividerclock}}{2^{PRESCALAR}}$$

or about 14648 Hz. Users can adjust using SYSCAL and PRESCALAR to match the

nearest wanted value. To enable the auto-reload functionality, the bit AUTOREL must be set.

- Configuration of ST_CTX: As shown previously, to obtain a 10 ms tick timer, the counter load value must be 150. This is the value to be written in the LOAD field. An interrupt can be generated when this value is really loaded. Users can verify that CHLDX bit in ST_SR is set before going to the next step.
- Configuration of ST_IER: Generation occurs when the counter reaches 0. This is done by writing the CHENDX bit. GIC must be configured.
- Start the simple timer by writing the CHENX bit in ST_CR. Users can verify that CHENSX bit in ST_SR is set.

Interrupt Handling

- IRQ Entry and call C function.
- Read ST_SR and verify the source of the interrupt.
- Clear the corresponding interrupt at peripheral level by writing in the ST_CSR.
- Interrupt treatment.
- IRQ Exit.

Simple Timer (ST) Memory Map

Base Address ST0: 0xFFFE4000

Base Address ST1: 0xFFFE8000

Table 44. ST Memory Map

Offset	Register	Name	Access	Reset State
0x000 - 0x04C	Reserved	–	–	–
0x050	Enable Clock Register	ST_ECR	Write-only	–
0x054	Disable Clock Register	ST_DCR	Write-only	–
0x058	Power Management Status Register	ST_PMSR	Read-only	0x00000000
0x05C	Reserved	–	–	–
0x060	Control Register	ST_CR	Write-only	–
0x064	Reserved	–	–	–
0x068	Reserved	–	–	–
0x06C	Clear Status Register	ST_CSR	Write-only	–
0x070	Status Register	ST_SR	Read-only	0x00000000
0x074	Interrupt Enable Register	ST_IER	Write-only	–
0x078	Interrupt Disable Register	ST_IDR	Write-only	–
0x07C	Interrupt Mask Register	ST_IMR	Read-only	0x00000000
0x080	Channel 0 Prescalar	ST_PR0	Read/Write	0x00000000
0x084	Channel 0 Counter	ST_CT0	Read/Write	0x00000000
0x088	Channel 1 Prescalar	ST_PR1	Read/Write	0x00000000
0x08C	Channel 1 Counter	ST_CT1	Read/Write	0x00000000
0x200	Current Counter Value 0	ST_CCV0	Read	0x00000000
0x204	Current Counter Value 1	ST_CCV1	Read	0x00000000

ST Enable Clock Register

Name: ST_ECR
Access: Write-only
Base Address: 0x050

ST Disable Clock Register

Name: ST_DCR
Access: Write-only
Base Address: 0x054

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
–	–	–	–	–	–	ST	–

- **ST: Simple Timer Clock Status**

0: Simple Timer clock disabled.

1: Simple Timer clock enabled.

ST Power Management Status Register

Name: ST_PMSR
Access: Read-only
Base Address: 0x058

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
–	–	–	–	–	–	ST	–

- **ST: Simple Timer Clock Status**

0: Simple Timer clock disabled.

1: Simple Timer clock enabled.

Note: The ST_PMSR register is not reset by a software reset.

ST Control Register

Name: ST_CR
Access: Write-only
Base Address: 0x060

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
–	–	–	CHDIS1	CHEN1	CHDIS0	CHEN0	SWRST

- **SWRST: ST Software Reset**

0: No effect.

1: Resets the Simple Timer.

A software reset of the ST is performed. It resets all the registers.

- **CHENX: Simple Timer Channel Enable**

0: No effect.

1: Enables the Simple Timer Channel X.

- **CHDISX: Simple Timer Channel Disable**

0: No effect.

1: Disables the Simple Timer Channel X.

If both CHENX and CHDISX are equal to one when the control register is written, the Simple Timer Channel X will be disabled.

ST Clear Status Register

Name: ST_CSR
Access: Write-only
Base Address: 0x06C

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
–	–	–	CHDIS1	CHEN1	CHDIS0	CHEN0	SWRST

- **CHENDX: Clear Channel End Interrupt**

0: No effect.

1: Clears CHENDX interrupt.

- **CHDISX: Clear Channel Disable Interrupt**

0: No effect.

1: Clears CHDISX interrupt.

- **CHLDX: Clear Channel Load Interrupt**

0: No effect.

1: Clears CHLDX interrupt.

ST Status Register

Name: ST_SR
Access: Read-only
Base Address: 0x070

31	30	29	28	27	26	25	24
–	–	–	–	–	–	CHENS1	CHENS0

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	CHLD1	CHDIS1	CHEND1	CHLD0	CHDIS0	CHEND0

- **CHENDX: Channel End Status**

0: No end of counting on Channel X.

1: End of counting on Channel X.

- **CHDISX: Channel Disable Status**

0: No effect.

1: Channel X divider has been reset.

CHDISX indicates that the divider module has been reset (a delay due to asynchronous clock is introduced).

- **CHLDX: Channel Load Status**

0: No effect.

1: Channel X down-counter is loaded.

Note: CHLDX indicates also that the counter (divider) has been enabled (a delay due to asynchronous clock is introduced).

- **CHENSX: Channel Enable Status**

0: Channel X is disabled.

1: Channel X is enabled.

ST Interrupt Enable Register

Name: ST_IER
Access: Write-only
Base Address: 0x074

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
–	–	CHLD1	CHDIS1	CHEND1	CHLD0	CHDIS0	CHEND0

ST Interrupt Disable Register

Name: ST_IDR
Access: Write-only
Base Address: 0x078

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
–	–	CHLD1	CHDIS1	CHEND1	CHLD0	CHDIS0	CHEND0

ST Interrupt Mask Register

Name: ST_IMR
Access: Read-only
Base Address: 0x07C

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
–	–	CHLD1	CHDIS1	CHEND1	CHLD0	CHDIS0	CHEND0

- **CHENDX: Channel End Interrupt Mask**

0: Channel X end interrupt is disabled.

1: Channel X end interrupt is enabled.

- **CHDISX: Channel Disable Interrupt Mask**

0: Channel X disable interrupt is disabled.

1: Channel X disable interrupt is enabled.

- **CHLDX: Channel Load Interrupt Mask**

0: Channel X load interrupt is disabled.

1: Channel X load interrupt is enabled.

ST Channel 0 Prescalar Register

Name: ST_PR0
Access: Read/Write
Base Address: 0x080

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

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

15	14	13	12	11	10	9	8
SYSCAL							

7	6	5	4	3	2	1	0
–	–	AUTOREL	SELECTCLK	PRESCALAR			

- **PRESCALAR: Channel 0 Prescalar**

  Channel 0 counter_clock_frequency = divider_clock_0 / $2^{PRESCALAR}$

- **SELECTCLK: Select Clock**

0: The divider_clock_0 is generated by divider driven by the SYSCAL bits.

1: The divider_clock_0 is connected to the low frequency clock.

- **AUTOREL: Auto Reload**

0: The counter is not automatically reloaded with the COUNTER[15:0] value when it reaches 0x0000.

1: The counter is automatically reloaded with the COUNTER[15:0] value when it reaches 0x0000.

- **SYSCAL: System Clock Prescalar Value**

This prescalar is used to divide the CORECLK when the system clock is selected (SELECTCLK = 0).

divider_clock_0 = CORECLK / (2 × (SYSCAL + 1))

A write in this register can only be done if CHENS0 = 0 (i.e., channel 0 is disabled).

ST Channel 0 Counter Register

Name: ST_CT0
Access: Read/Write
Base Address: 0x084

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

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

15	14	13	12	11	10	9	8
LOAD							

7	6	5	4	3	2	1	0
LOAD							

- **LOAD: Channel 0 Preload Value**

Gives the instruction to the timer; i.e., from which counter value the Simple Timer has to decrement.

ST Channel 1 Prescalar Register

Name: ST_PR1
Access: Read/Write
Base Address: 0x088

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

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

15	14	13	12	11	10	9	8
SYSCAL							

7	6	5	4	3	2	1	0
–	–	AUTOREL	SELECTCLK	PRESCALAR			

- **PRESCALAR: Channel 1 Prescalar**

  Channel 1 counter_clock_frequency = divider_clock_0 / $2^{PRESCALAR}$

- **SELECTCLK: Select Clock**

0: The divider_clock_0 is generated by divider driven by the SYSCAL bits.

1: The divider_clock_0 is connected to the low frequency clock.

- **AUTOREL: Auto Reload**

0: The counter is not automatically reloaded with the COUNTER[15:0] value when it reaches 0x0000.

1: The counter is automatically reloaded with the COUNTER[15:0] value when it reaches 0x0000.

- **SYSCAL: System Clock Prescalar Value**

This prescalar is used to divide the CORECLK when the system clock is selected (SELECTCLK = 0).

divider_clock_0 = CORECLK / (2 × (SYSCAL + 1))

A write in this register can only be done if CHENS0 = 0 (i.e. channel 0 is disabled).

ST Channel 1 Counter Register

Name: ST_CT1
Access: Read/Write
Base Address: 0x08C

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

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

15	14	13	12	11	10	9	8
LOAD							

7	6	5	4	3	2	1	0
LOAD							

- **LOAD: Channel 1 Preload Value**

Gives the instruction to the timer; i.e., from which counter value the Simple Timer has to decrement.

ST Current Counter Value 0 Register

Name: ST_CCV0
Access: Read-only
Base Address: 0x0200

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

23	22	21	20	19	18	17	16
–	–	–	–	–	–		

15	14	13	12	11	10	9	8
COUNT							

7	6	5	4	3	2	1	0
COUNT							

- **COUNT[15:0]: Current Counter Value 0 Register**

This register gives the current value of the Channel 0 down counter.

ST Current Counter Value 1 Register

Name: ST_CCV1
Access: Read-only
Base Address: 0x0204

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

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

15	14	13	12	11	10	9	8
COUNT							

7	6	5	4	3	2	1	0
COUNT							

- **COUNT[15:0]: Current Counter Value 1 Register**

This register gives the current value of the Channel 1 down counter.

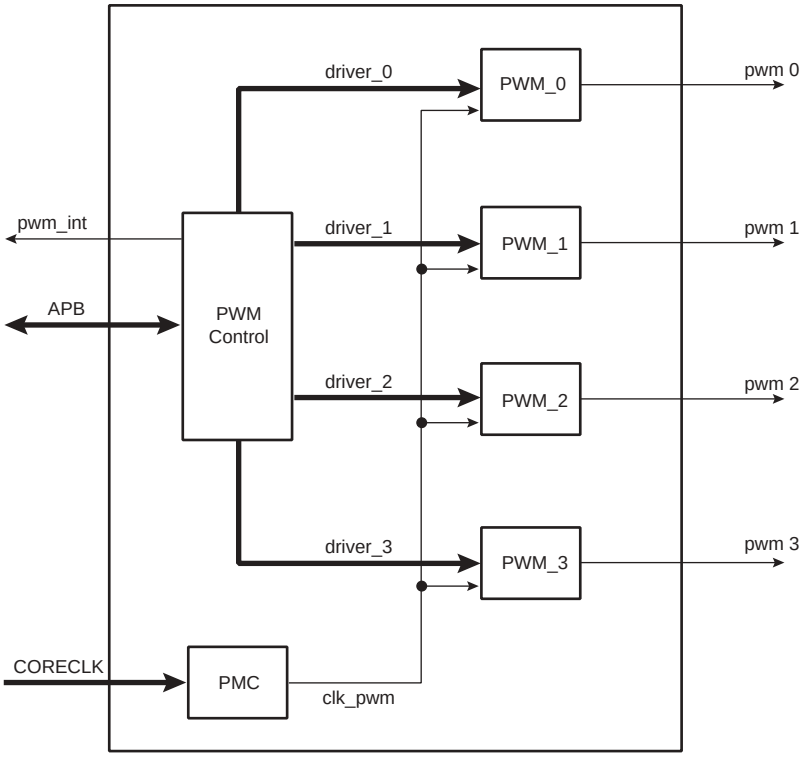
Pulse Width Modulator (PWM)

Overview

The AT91SAM7A2 includes a four-channel PWM that generates pulses. The width and the frequency of the pulses can be controlled independently on each channel, thus making it possible to generate four different waveforms at the same time.

Block Diagram

Figure 61. PWM Block Diagram



Pin Description

There are four output pins: PWM_0, PWM_1, PWM_2, PWM_3. They provide four digital signals that can be used to control, for example, a voltage level.

PWM Parameters

There are three parameters for each PWM: counter frequency, delay and pulse width.

A fourth parameter allows users to change polarity of the pulse and of the delay levels. By default, the pulse is at a logical 1 level and the delay is at a logical 0 level.

Counter Frequency

The PWM module is a counter. It counts delay width cycles, setting the PWMX output inactive, then it counts pulse width cycles, setting the PWMX output active. This operation is repeated until the PWM channel is stopped.

The user can control the frequency of this counter with a prescalar.

Delay Width

Delay width is the number of counter cycles during which the output is inactive. PWM starts with a delay when enabled.

Pulse Width

Pulse width is the number of counter cycles during which the output is active.

PWM Output Level When Disabled

When the PWM module goes from enable to disable state, its outputs are set to the delay level defined in the mode register by the PLX bits.

If PLX is set to 0, when the PWM is disabled, the output will drive a logical 1.

If PLX is set to 1, when the PWM is disabled, the output will drive a logical 0.

Power Management

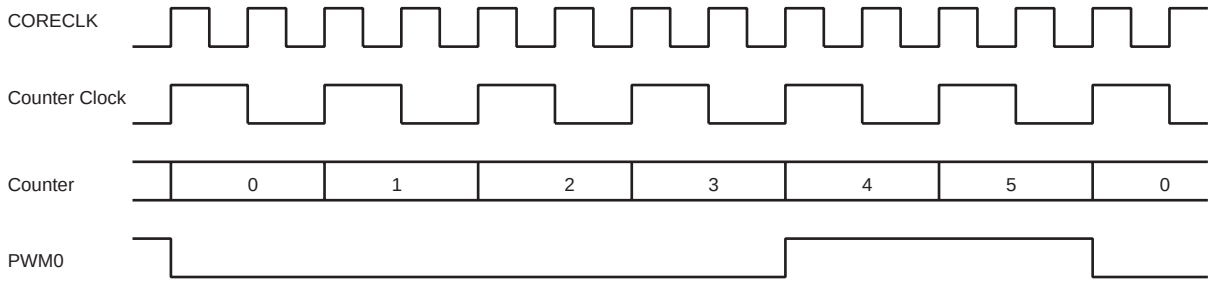
The PWM is provided with a power management block allowing optimization of power consumption (see “Power Management Block” on page 22).

Example of Use

This section gives an example of use of the PWM, explaining how to generate a pulse with a frequency of 5 KHz with a duty cycle of 50%, core clock = 30 MHz.

If the counter frequency is set to CORECLK/2, delay width is equal to four cycles and pulse width to two cycles. See Figure 62.

Figure 62. Waveform Diagram



Configuration

- Enable the clock on PWM peripheral by writing bit PWM in PWM_ECR.
- Do a software reset of the PWM peripheral to be in a known state by writing bit SWRST in PWM _CR.
- Configuration of PWM_ MR: Use a PWM clock of 15 MHz with a PRESCAL of 0 and the pulse at a logical state 1. Users can choose one of the four PWMs available.
- Configuration of PWM_ DLYX: This register indicates the number of PWM clocks needed to form the delay (level 0), e.g. 1500, that should result in half a period of 0.1 ms.
- Configuration of PWM_ PULX: This register indicates the number of PWM clocks needed to form the pulse (level 1), e.g. 1500, that should result in half a period of 0.1ms.
- Start the selected PWM by writing the bit PWMENX in PWM_CR. This should supply a pulse of 5 kHz on the selected PWM with a duty cycle of 50%.

Pulse Width Modulator (PWM) Memory Map

Base Address:0xFFFD0000

Table 45. PWM Memory Map

Offset	Register	Name	Access	Reset State
0x000 - 0x04C	Reserved	–	–	–
0x050	Enable Clock Register	PWM_ECR	Write-only	–
0x054	Disable Clock Register	PWM_DCR	Write-only	–
0x058	Power Management Status Register	PWM_PMSR	Read-only	0x00000000
0x05C	Reserved	–	–	–
0x060	Control Register	PWM_CR	Write-only	–
0x064	Mode Register	PWM_MR	Read/Write	0x10101010
0x068	Reserved	–	–	–
0x06C	Clear Status Register	PWM_CSR	Write-only	–
0x070	Status Register	PWM_SR	Read-only	0x00000000
0x074	Interrupt Enable Register	PWM_IER	Write-only	–
0x078	Interrupt Disable Register	PWM_IDR	Write-only	–
0x07C	Interrupt Mask Register	PWM_IMR	Read-only	0x00000000
0x080	Delay Register 0	PWM_DLY_0	Read/Write	0x00000000
0x084	Pulse Register 0	PWM_PUL_0	Read/Write	0x00000000
0x088	Delay Register 1	PWM_DLY_1	Read/Write	0x00000000
0x08C	Pulse Register 1	PWM_PUL_1	Read/Write	0x00000000
0x090	Delay Register 2	PWM_DLY_2	Read/Write	0x00000000
0x094	Pulse Register 2	PWM_PUL_2	Read/Write	0x00000000
0x098	Delay Register 3	PWM_DLY_3	Read/Write	0x00000000
0x09C	Pulse Register 3	PWM_PUL_3	Read/Write	0x00000000

PWM Enable Clock Register

Name: PWM_ECR
Access: Write-only
Base Address: 0x050

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
–	–	–	–	–	–	PWM	–

- **PWM: PWM Clock**

0: PWM clock disabled.

1: PWM clock enabled.

The PWM_PMSR register is not reset by software reset.

PWM Disable Clock Register

Name: PWM_DCR
Access: Write-only
Base Address: 0x054

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
–	–	–	–	–	–	PWM	–

- **PWM: PWM Clock**

0: PWM clock disabled.

1: PWM clock enabled.

Note: The PWM_PMSR register is not reset by software reset.

PWM Power Management Status Register

Name: PWM_PMSR
Access: Read-only
Base Address: 0x058

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
–	–	–	–	–	–	PWM	–

- **PWM: PWM Clock**

0: PWM clock disabled.

1: PWM clock enabled.

Note: The PWM_PMSR register is not reset by software reset.

PWM Control Register

Name: PWM_CR
Access: Write-only
Base Address: 0x060

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	PWMDIS3

7	6	5	4	3	2	1	0
PWMEN3	PWMDIS2	PWMEN2	PWMDIS1	PWMEN1	PWMDIS0	PWMEN0	SWRST

- **SWRST: PWM Software Reset**

0: No effect

1: Resets the PWM.

A software-triggered hardware reset of the PWM is performed. It resets all the registers except the PWM_PMSR register.

- **PWMENX: PWM Enable Channel Number X**

0: No effect.

1: Enables the PWM.

- **PWMDISX: PWM Disable Channel Number X**

0: No effect.

1: Disables the PWM.

If both PWMENX and PWMDISX are equal to one, the corresponding PWM channel is disabled.

PWM Mode Register

Name: PWM_MR
Access: Read/Write
Base Address: 0x064

31	30	29	28	27	26	25	24
–	–	–	PL3	PRESCAL3			

23	22	21	20	19	18	17	16
–	–	–	PL2	PRESCAL2			

15	14	13	12	11	10	9	8
–	–	–	PL1	PRESCAL1			

7	6	5	4	3	2	1	0
			PL0	PRESCAL0			

- **PRESCALX[3:0]: Counter Clock Prescalar for PWM Channel X**

  Counter Clock Frequency = CORECLK$/2^{PRESCALAR1}$

Note: "X" indicates that the bit concerns the PWMX.

- **PLX: Pulse Level for PWM Channel X**

0: The pulses are at logic level 0. During the delay, the output is at logic level 1.

1: The pulses are at logic level 1. During the delay, the output is at logic level 0.

Note: When pulse level change, a pulse start or a pulse end is detected, and the corresponding bit is set in the status register (generating an interrupt if enabled).

PWM Clear Status Register

Name: PWM_CSR
Access: Write-only
Base Address: 0x06C

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
PEND3	PSTA3	PEND2	PSTA2	PEND1	PSTA1	PEND0	PSTA0

- **PSTAX: Pulse Start Interrupt**

0: No effect.

1: Clears the PSTA interrupt.

- **PENDX: Pulse End Interrupt**

0: No effect.

1: Clears the PEND interrupt.

Note: "X" indicates that the bit concerns the PWMX.

PWM Status Register

Name: PWM_SR
Access: Read-only
Base Address: 0x06C

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

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

15	14	13	12	11	10	9	8
–	–	–	–	PWMENS3	PWMENS2	PWMENS1	PWMENS0

7	6	5	4	3	2	1	0
PEND3	PSTA3	PEND2	PSTA2	PEND1	PSTA1	PEND0	PSTA0

- **PSTAX: Pulse Start**

0: No pulse started since the last read of PWM_SR.

1: A pulse started since the last read of PWM_SR.

- **PENDX: Pulse End**

0: No pulse ended since the last read of PWM_SR.

1: A pulse ended since the last read of PWM_SR.

- **PWMENSX: PWM Enable Status of Channel X**

0: PWM is disabled.

1: PWM is enabled.

Note: "X" indicates that the bit concerns the PWMX.

PWM Interrupt Enable Register

Name: PWM_IER
Access: Write-only
Base Address: 0x074

PWM Interrupt Disable Register

Name: PWM_IDR
Access: Write-only
Base Address: 0x078

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
PEND3	PSTA3	PEND2	PSTA2	PEND1	PSTA1	PEND0	PSTA0

- **PSTAX: Pulse Start Interrupt**

0: Pulse Start Interrupt is disabled.

1: Pulse Start Interrupt is enabled.

- **PENDX: Pulse End Interrupt**

0: Pulse End Interrupt is disabled.

1: Pulse End Interrupt is enabled.

Note: "X" indicates that the bit concerns the PWMX.

PWM Interrupt Mask Register

Name: PWM_IMR
Access: Read-only
Base Address: 0x07C

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
PEND3	PSTA3	PEND2	PSTA2	PEND1	PSTA1	PEND0	PSTA0

- **PSTAX: Pulse Start Interrupt**

0: Pulse Start Interrupt is disabled.

1: Pulse Start Interrupt is enabled.

- **PENDX: Pulse End Interrupt**

0: Pulse End Interrupt is disabled.

1: Pulse End Interrupt is enabled.

Note: "X" indicates that the bit concerns the PWMX.

PWM Delay Register x [x = 0..3]

Name: PWM_DLY_x
Access: Read/Write
Address: 0x080...0x098

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

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

15	14	13	12	11	10	9	8
DELAY							

7	6	5	4	3	2	1	0
DELAY							

- **DELAY[15:0]: PWM Delay on Channel X**

Number of counter cycles during which the output is inactive.

Note: "X" indicates that the bit concerns the PWMX.

Change of this value is immediately taken into account. This may cause a bad delay on the current pulse.

PWM Pulse Register x [x = 0..3]

Name: PWM_PUL_x
Access: Read/Write
Address: 0x084...0x9C

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

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

15	14	13	12	11	10	9	8
PULSE							

7	6	5	4	3	2	1	0
PULSE							

- **PULSE[15:0]: Pulse Width on Channel X**

Number of counter cycles during which the output is active.

Note: "X" indicates that the bit concerns the PWMX.

Change of this value is immediately taken into account. This may cause a bad delay on the current pulse.

Serial Peripheral Interface (SPI)

Overview

The AT91SAM7A2 includes a Serial Peripheral Interface (SPI) that provides communication with external devices in Master or Slave Mode.

Seven pins are associated with the SPI interface. When not needed for the SPI function, each of these pins can be configured as a PIO, using PIO Controller functions.

After a hardware reset, the SPI pins are not enabled by default. The user must configure the PIO Controller to enable the corresponding pins to their PIO function.

To configure a SPI pin as open-drain to support external drivers, the software can set the corresponding bits in the multi-driver registers. The NPCS0 pin can function as a peripheral chip select output or slave select input, or as a single PIO.

Master Mode

In Master Mode, the SPI controls data transfers to and from the slave(s) connected to the SPI bus. The SPI drives the chip select(s) to the slave(s) and the serial clock (SPCK). After enabling the SPI, a data transfer begins when the ARM core writes to the Transmit Data Register (SPI_TDR).

Transmit and Receive buffers maintain the data flow at a constant rate with a reduced requirement for high-priority interrupt servicing. As long as new data is available in the Transmit Data Register (SPI_TDR), the SPI continues to transfer data. If the Receive Data Register (SPI_RDR) has not been read before new data is received, the overrun error (SPIOVRE) flag is set.

The delay between the activation of the chip select and the start of the data transfer (DLYBS) as well as the delay between each data transfer (DLYBCT) can be programmed for each of the four external chip selects. All data transfer characteristics including the two timing values are programmed in registers SPI_CSR0 to SPI_CSR3 (Chip Select Registers).

In Master Mode, the peripheral selection can be defined in two different ways:

- Fixed peripheral select: SPI exchanges data with only one peripheral
- Variable peripheral select: Data can be exchanged with more than one peripheral

Note: The SPI master should never be configured as described below:
- SPI master clock equals CORECLK by setting the bit DIV32 to a logical 0 in the SPI mode register (SPI_MR).
- Pin NPCS0/NSS configured as an open drain by setting bit NPCS0 to a logical 1 in the SPI Multi-Driver Enable Register (SPI_MDER).
These two options set in the same configuration may cause a mode fault due to the frequency of the CORECLK used to sample the state of the NPCS0/NSS signal after each transmission.

Fixed Peripheral Select

This mode is used for transferring memory blocks without the extra overhead in the Transmit Data Register to determine the peripheral.

Fixed peripheral select is activated by setting bit PS to a logical 0 in the SPI Mode Register (SPI_MR).

The peripheral is defined by the PCS field in SPI_MR.

This option is only available when the SPI is programmed in Master Mode.

Variable Peripheral Select

Variable peripheral select is activated by setting bit PS to a logical 1 in the SPI Mode Register (SPI_MR). The PCS field in Transmit Data Register (SPI_TDR) is used to

select the destination peripheral. The data transfer characteristics are changed when the selected peripheral changes, according to the associated chip select register.

The PCS field in the SPI_MR has no effect.

This option is only available when the SPI is programmed in Master Mode.

Chip Selects

The Chip Select lines are driven by the SPI only if it is programmed in Master Mode. These lines are used to select the destination peripheral.

The PCSDEC field in SPI_MR (Mode Register) is used to select one to four peripherals or up to 16 peripherals:

- PCSDEC = 0, each NPCSx acts as a chip select line so up to four devices can be selected.
- PCSDEC = 1, the NPCS[3:0] signals act as an address bus selecting up to 16 peripherals.

If variable peripheral select is active (PS = 1 in SPI_MR), the chip select signals are defined for each transfer in the PCS field in SPI_TDR. Chip select signals can thus be defined independently for each transfer.

If fixed peripheral select is active (PS = 0 in SPI_MR), chip select signals are defined for all transfers by the field PCS in SPI_MR. If a transfer with a new peripheral is necessary, the software must wait until the current transfer is completed, then change the value of PCS in SPI_MR before writing new data in SPI_TDR.

The value on the NPCS pins at the end of each transfer can be read in the Receive Data Register (SPI_RDR). By default, all NPCS signals are high (equal to one) before and after each transfer.

Mode Fault Detection

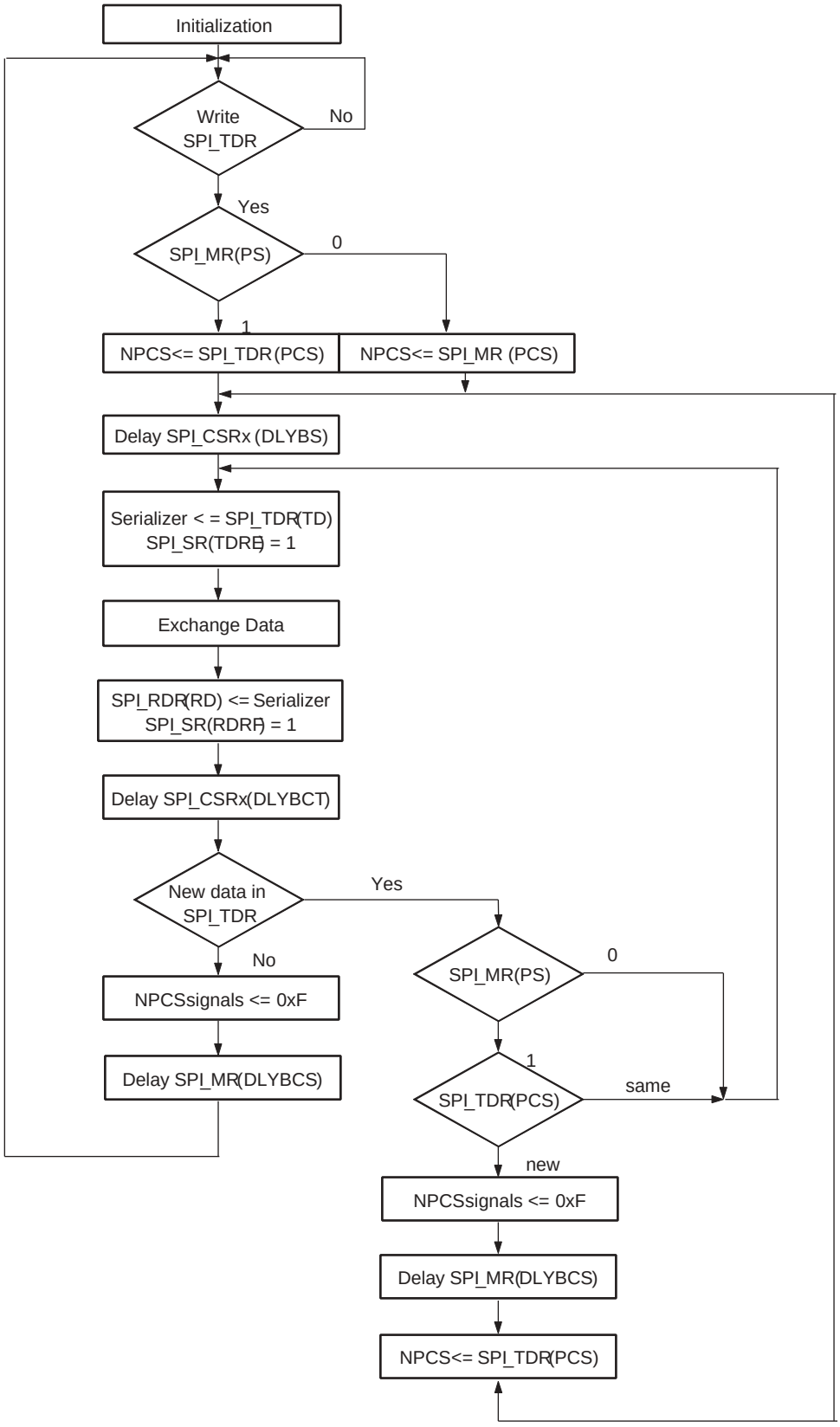
A mode fault is detected when the SPI is programmed in Master Mode and a low level is driven by an external master on the NPCS0 signal.

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).

A mode fault is not detected when the master NPCS0 signal is configured in PIO and driven low.

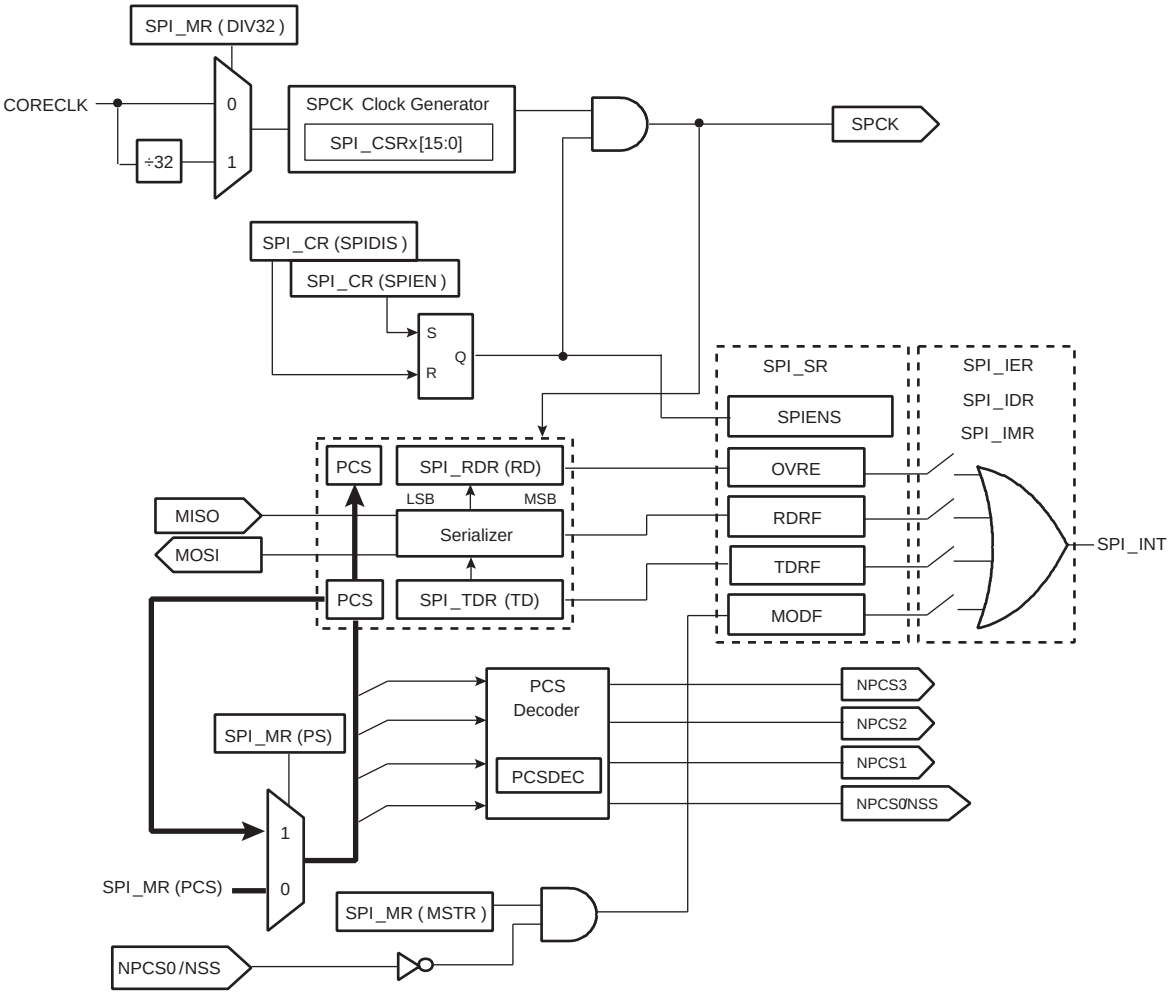
Functional Flow Diagram in Master Mode

Figure 63. SPI Flow Diagram in Master Mode



SPI in Master Mode

Figure 64. SPI in Master Mode

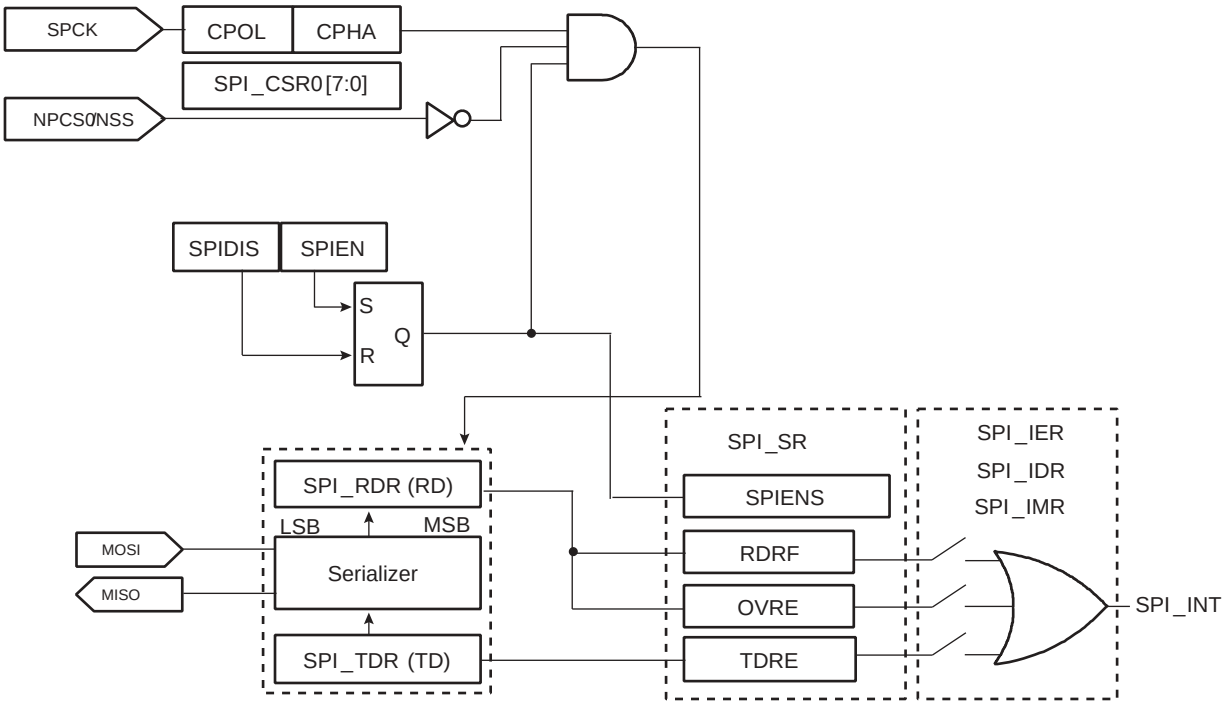


Slave Mode

In Slave Mode, the SPI waits for NSS to go active low before receiving the serial clock from an external master. In Slave Mode, CPOL, NCPHA and BITS fields of SPI_CSR0 are used to define the transfer characteristics. The other fields in SPI_CSR0 and the other Chip Select Registers are not used in Slave Mode.

Note: The SPI in Slave Mode can not be used with the PDC for data transmission or reception.

Figure 65. SPI in Slave Mode



PIO Controller

The SPI has 7 programmable I/O lines. These I/O lines are multiplexed with signals (MISO, MOSI, SPCK, NPCS[3:0]) of the SPI to optimize the use of available package pins. These lines are controlled by the SPI PIO controller.

Power Management

The SPI is provided with a power management block allowing optimization of power consumption (see “Power Management Block” on page 22).

Serial Peripheral Interface (SPI) Memory Map

Base Address: 0xFFFB4000

Table 46. SPI Memory Map

Offset	Register	Name	Access	Reset State
0x000	PIO Enable Register	SPI_PER	Write-only	–
0x004	PIO Disable Register	SPI_PDR	Write-only	–
0x008	PIO Status Register	SPI_PSR	Read-only	0x007F0000
0x00C	Reserved	–	–	–
0x010	Output Enable Register	SPI_OER	Write-only	–
0x014	Output Disable Register	SPI_ODR	Write-only	–
0x018	Output Status Register	SPI_OSR	Read-only	0x00000000
0x01C - 0x02C	Reserved	–	–	–
0x030	Set Output Data Register	SPI_SODR	Write-only	–
0x034	Clear Output Data Register	SPI_CODR	Write-only	–
0x038	Output Data Status Register	SPI_ODSR	Read-only	0x00000000
0x03C	Pin Data Status Register	SPI_PDSR	Read-only	0x00XX0000
0x040	Multi-Driver Enable Register	SPI_MDER	Write-only	–
0x044	Multi-Driver Disable Register	SPI_MDDR	Write-only	–
0x048	Multi-Driver Status Register	SPI_MDSR	Read-only	0x00000000
0x04C	Reserved	–	–	–
0x050	Enable Clock Register	SPI_ECR	Write-only	–
0x054	Disable Clock Register	SPI_DCR	Write-only	–
0x058	Power Management Status Register	SPI_PMSR	Read-only	0x00000000
0x05C	Reserved	–	–	–
0x060	Control Register	SPI_CR	Write-only	0x00000000
0x064	Mode Register	SPI_MR	Read/Write	0x00000000
0x068 - 0x06C	Reserved	–	–	–
0x070	Status Register	SPI_SR	Read-only	0x00000000
0x074	Interrupt Enable Register	SPI_IER	Write-only	–
0x078	Interrupt Disable Register	SPI_IDR	Write-only	–
0x07C	Interrupt Mask Register	SPI_IMR	Read-only	0x00000000
0x080	Receive Data Register	SPI_RDR	Read-only	–
0x084	Transmit Data Register	SPI_TDR	Write-only	0x00000000
0x088 - 0x08C	Reserved	–	–	–
0x090	Chip Select Register 0	SPI_CSR0	Read/Write	0x00000000
0x094	Chip Select Register 1	SPI_CSR1	Read/Write	0x00000000
0x098	Chip Select Register 2	SPI_CSR2	Read/Write	0x00000000
0x09C	Chip Select Register 3	SPI_CSR3	Read/Write	0x00000000

SPI PIO Enable Register

Name: SPI_PER
Access: Write-only
Base Address: 0x000

SPI PIO Disable Register

Name: SPI_PDR
Access: Write-only
Base Address: 0x004

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	NPCS3	NPCS2	NPCS1	NPCS0	MOSI	MISO	SPCK
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **SPCK: SPCK Pin**

0: PIO is inactive on the pin SPCK.

1: PIO is active on the pin SPCK.

- **MISO: MISO Pin**

0: PIO is inactive on the pin MISO.

1: PIO is active on the pin MISO.

- **MOSI: MOSI Pin**

0: PIO is inactive on the pin MOSI.

1: PIO is active on the pin MOSI.

- **NPCS0: NPCS0 Pin**

0: PIO is inactive on the pin NPCS0/NSS.

1: PIO is active on the pin NPCS0/NSS.

- **NPCSx[x = 1..3]: NPCSx Pin**

0: PIO is inactive on the pin NPCSx.

1: PIO is active on the pin NPCSx.

SPI PIO Status Register

Name: SPI_PSR
Access: Read-only
Base Address: 0x008

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

23	22	21	20	19	18	17	16
–	NPCS3	NPCS2	NPCS1	NPCS0	MOSI	MISO	SPCK

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **SPCK: SPCK Pin**

0: PIO is inactive on the pin SPCK.

1: PIO is active on the pin SPCK.

- **MISO: MISO Pin**

0: PIO is inactive on the pin MISO.

1: PIO is active on the pin MISO.

- **MOSI: MOSI Pin**

0: PIO is inactive on the pin MOSI.

1: PIO is active on the pin MOSI.

- **NPCS0: NPCS0 Pin**

0: PIO is inactive on the pin NPCS0/NSS.

1: PIO is active on the pin NPCS0/NSS.

- **NPCSx[x = 1..3]: NPCSx Pin**

0: PIO is inactive on the pin NPCSx.

1: PIO is active on the pin NPCSx.

SPI PIO Output Enable Register

Name: SPI_OER
Access: Write-only
Base Address: 0x010

SPI PIO Output Disable Register

Name: SPI_ODR
Access: Write-only
Base Address: 0x014

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

23	22	21	20	19	18	17	16
–	NPCS3	NPCS2	NPCS1	NPCS0	MOSI	MISO	SPCK

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **SPCK: SPCK Pin**

0: PIO is input on the pin SPCK.

1: PIO is output on the pin SPCK.

- **MISO: MISO Pin**

0: PIO is input on the pin MISO.

1: PIO is output on the pin MISO.

- **MOSI: MOSI Pin**

0: PIO is input on the pin MOSI.

1: PIO is output on the pin MOSI.

- **NPCS0: NPCS0 Pin**

0: PIO is input on the pin NPCS0/NSS.

1: PIO is output on the pin NPCS0/NSS.

- **NPCSx[x = 1..3]: NPCSx Pin**

0: PIO is input on the pin NPCSx.

1: PIO is output on the pin NPCSx.

SPI PIO Output Status Register

Name: SPI_OSR
Access: Read-only
Base Address: 0x018

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

23	22	21	20	19	18	17	16
–	NPCS3	NPCS2	NPCS1	NPCS0	MOSI	MISO	SPCK

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **SPCK: SPCK Pin**

0: PIO is input on the pin SPCK.

1: PIO is output on the pin SPCK.

- **MISO: MISO Pin**

0: PIO is input on the pin MISO.

1: PIO is output on the pin MISO.

- **MOSI: MOSI Pin**

0: PIO is input on the pin MOSI.

1: PIO is output on the pin MOSI.

- **NPCS0: NPCS0 Pin**

0: PIO is input on the pin NPCS0/NSS.

1: PIO is output on the pin NPCS0/NSS.

- **NPCSx [x = 1..3]: NPCSx Pin**

0: PIO is input on the pin NPCSx.

1: PIO is output on the pin NPCSx.

SPI PIO Set Output Data Register

Name: SPI_SODR
Access: Write-only
Base Address: 0x030

SPI PIO Clear Output Data Register

Name: SPI_CODR
Access: Write-only
Base Address: 0x034

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

23	22	21	20	19	18	17	16
–	NPCS3	NPCS2	NPCS1	NPCS0	MOSI	MISO	SPCK

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **SPCK: SPCK Pin**

0: The output data for the pin SPCK is programmed to 0.

1: The output data for the pin SPCK is programmed to 1.

- **MISO: MISO Pin**

0: The output data for the pin MISO is programmed to 0.

1: The output data for the pin MISO is programmed to 1.

- **MOSI: MOSI Pin**

0: The output data for the pin MOSI is programmed to 0.

1: The output data for the pin MOSI is programmed to 1.

- **NPCS0: NPCS0 Pin**

0: The output data for the pin NPCS0/NSS is programmed to 0.

1: The output data for the pin NPCS0/NSS is programmed to 1.

- **NPCSx [x = 1..3]: NPCSx Pin**

0: The output data for the pin NPCSx is programmed to 0.

1: The output data for the pin NPCSx is programmed to 1.

SPI PIO Output Data Status Register

Name: SPI_ODSR
Access: Read-only
Base Address: 0x038

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

23	22	21	20	19	18	17	16
–	NPCS3	NPCS2	NPCS1	NPCS0	MOSI	MISO	SPCK

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **SPCK: SPCK Pin**

0: The output data for the pin SPCK is programmed to 0.

1: The output data for the pin SPCK is programmed to 1.

- **MISO: MISO Pin**

0: The output data for the pin MISO is programmed to 0.

1: The output data for the pin MISO is programmed to 1.

- **MOSI: MOSI Pin**

0: The output data for the pin MOSI is programmed to 0.

1: The output data for the pin MOSI is programmed to 1.

- **NPCS0: NPCS0 Pin**

0: The output data for the pin NPCS0/NSS is programmed to 0.

1: The output data for the pin NPCS0/NSS is programmed to 1.

- **NPCSx [x = 1..3]: NPCSx Pin**

0: The output data for the pin NPCSx is programmed to 0.

1: The output data for the pin NPCSx is programmed to 1.

SPI PIO Pin Data Status Register

Name: SPI_PDSR
Access: Read-only
Base Address: 0x03C

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

23	22	21	20	19	18	17	16
–	NPCS3	NPCS2	NPCS1	NPCS0	MOSI	MISO	SPCK

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **SPCK: SPCK Pin**

0: The pin SPCK is at logic 0.

1: The pin SPCK is at logic 1.

- **MISO: MISO Pin**

0: The pin MISO is at logic 0.

1: The pin MISO is at logic 1.

- **MOSI: MOSI Pin**

0: The pin MOSI is at logic 0.

1: The pin MOSI is at logic 1.

- **NPCS0: NPCS0 Pin**

0: The pin NPCS0/NSS is at logic 0.

1: The pin NPCS0/NSS is at logic 1.

- **NPCSx [x = 1..3]: NPCSx Pin**

0: The pin NPCSx is at logic 0.

1: The pin NPCSx is at logic 1.

SPI PIO Multi-driver Enable Register

Name: SPI_MDER
Access: Write-only
Base Address: 0x040

SPI PIO Multi-driver Disable Register

Name: SPI_MDDR
Access: Write-only
Base Address: 0x044

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

23	22	21	20	19	18	17	16
–	NPCS3	NPCS2	NPCS1	NPCS0	MOSI	MISO	SPCK

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **SPCK: SPCK Pin**

0: The pin SPCK is not configured as an open drain.

1: The pin SPCK is configured as an open drain.

- **MISO: MISO Pin**

0: The pin MISO is not configured as an open drain.

1: The pin MISO is configured as an open drain.

- **MOSI: MOSI Pin**

0: The pin MOSI is not configured as an open drain.

1: The pin MOSI is configured as an open drain.

- **NPCS0: NPCS0 Pin**

0: The pin NPCS0/NSS is not configured as an open drain.

1: The pin NPCS0/NSS is configured as an open drain.

- **NPCSx [x = 1..3]: NPCSx Pin**

0: The pin NPCSx is not configured as an open drain.

1: The pin NPCSx is configured as an open drain.

SPI PIO Multi-driver Status Register

Name: SPI_MDSR
Access: Read-only
Base Address: 0x048

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

23	22	21	20	19	18	17	16
–	NPCS3	NPCS2	NPCS1	NPCS0	MOSI	MISO	SPCK

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **SPCK: SPCK Pin**

0: The pin SPCK is not configured as an open drain.

1: The pin SPCK is configured as an open drain.

- **MISO: MISO Pin**

0: The pin MISO is not configured as an open drain.

1: The pin MISO is configured as an open drain.

- **MOSI: MOSI Pin**

0: The pin MOSI is not configured as an open drain.

1: The pin MOSI is configured as an open drain.

- **NPCS0: NPCS0 Pin**

0: The pin NPCS0/NSS is not configured as an open drain.

1: The pin NPCS0/NSS is configured as an open drain.

- **NPCSx [x = 1..3]: NPCSx Pin**

0: The pin NPCSx is not configured as an open drain.

1: The pin NPCSx is configured as an open drain.

SPI Enable Clock Register

Name: SPI_ECR
Access: Write-only
Base Address: 0x050

SPI Disable Clock Register

Name: SPI_DCR
Access: Write-only
Base Address: 0x054

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
–	–	–	–	–	–	SPI	PIO

- **PIO: PIO Clock Status**

0: PIO clock disabled.

1: PIO clock enabled.

- **SPI: SPI Clock Status**

0: SPI clock disabled.

1: SPI clock enabled.

SPI Power Management Status Register

Name: SPI_PMSR
Access: Read-only
Base Address: 0x058

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
–	–	–	–	–	–	SPI	PIO

- **PIO: PIO Clock Status**

0: PIO clock disabled.

1: PIO clock enabled.

- **SPI: SPI Clock Status**

0: SPI clock disabled.

1: SPI clock enabled.

Note: The SPI_PMSR register is not reset by software reset.

SPI Control Register

Name: SPI_CR
Access: Write-only
Base Address: 0x060

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
–	–	–	–	–	SPIDIS	SPIEN	SWRST

- **SWRST: SPI Software Reset**

0: No effect.

1: Resets the SPI.

A software-triggered hardware reset of the SPI is performed. It reset all the registers (except SPI_PMSR).

- **SPIEN: SPI Enable**

0: No effect.

1: Enables the SPI.

This enables the SPI to transfer and receive data.

- **SPIDIS: SPI Disable**

0: No effect.

1: Disables the SPI.

All pins will be set to input mode and no data is received or transmitted.

In case a transfer is in progress, the transfer is finished before the SPI is disabled.

In case both SPIEN and SPIDIS are equal to one when the control register is written, the SPI is disabled.

SPI Mode Register

Name: SPI_MR
Access: Read/Write
Base Address: 0x064

31	30	29	28	27	26	25	24
DLYBCS							
23	22	21	20	19	18	17	16
–	–	–	–	PCS			
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
LLB	–	–	–	DIV32	PCSDEC	PS	MSTR

- **MSTR: Master/Slave Mode**

0: SPI is in Slave Mode. The SPI in Slave Mode can not be used with the PDC for data transmission or reception.

1: SPI is in Master Mode.

MSTR configures the SPI for either master or Slave Mode operation.

- **PS: Peripheral Select**

0: Fix peripheral select.

1: Variable peripheral select.

The peripheral mode is only used in Master Mode. In case of fix peripheral select, the selected peripheral is defined in the mode register. For variable peripheral select, it is defined in the transmit data register.

- **PCSDEC: Chip Select Decode**

0: The chip selects are directly connected to a peripheral device.

1: The four chip select lines are connected to a 4-to-16 decoder.

In case PCSDEC is one, up to 16 Chip Select signals can be generated with the 4 lines using an external 4 to 16 decoder.

The Chip Select Registers define the characteristics of the 16 chip selects according to the following rules:

- SPI_CSR0 defines peripheral chip select signals 0 to 3.
- SPI_CSR1 defines peripheral chip select signals 4 to 7.
- SPI_CSR2 defines peripheral chip select signals 8 to 11.
- SPI_CSR3 defines peripheral chip select signals 12 to 15.

- **DIV32: Clock Selection**

0: SPI Master Clock equals CORECLK.

1: SPI Master Clock equals CORECLK/32.

- **LLB: Local Loopback**

0: Local loopback path disabled.

1: Local loopback path enabled.

LLB controls the local loopback on the data serializer for testing.

- **PCS[3:0]: Peripheral Chip Select**

This field is only used if single peripheral select is active (PS = 0).

If PCSDEC = 0:

PCS[3:0] = xxx0 NPCS[3:0] = 1110
PCS[3:0] = xx01 NPCS[3:0] = 1101
PCS[3:0] = x011 NPCS[3:0] = 1011
PCS[3:0] = 0111 NPCS[3:0] = 0111
PCS[3:0] = 1111 forbidden (no peripheral is selected)

where x = don't care.

If PCSDEC = 1:

NPCS[3:0] output signals = PCS[3:0]

- **DLYBCS[7:0]: Delay Between Chip Selects**

This field defines the delay from one NPCS inactive to the activation of another NPCS. The DLYBCS[7:0] time will guarantee non-overlapping chip selects and resolves bus contentions in case of peripherals with long data float times.

When DLYBCS[7:0] is less than 6, six SPI Master Clock periods are inserted by default.

Else, the following equation determines the delay:

$$NPCS\_to\_SPCK\_Delay = DLYBCS[7:0] \times SPI\_Master\_Clock\_Period$$

SPI Status Register

Name: SPI_SR
Access: Read-only
Base Address: 0x070

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

23	22	21	20	19	18	17	16
–	NPCS3	NPCS2	NPCS1	NPCS0	MOSI	MISO	SPCK

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	SPIENS

7	6	5	4	3	2	1	0
–	–	TEND	REND	SPIOVRE	MODF	TDRE	RDRF

Note: This register is a "read-active" register, which means that reading it can affect the state of some bits. When reading SPI_SR register, following bits are cleared if set: MODF, SPIOVRE, REND, TEND, SPCK, MISO, MOSI, NPCS0, NPCS1, NPCS2 and NPCS3. When debugging, to avoid this behavior, users should use ghost registers (see “Ghost Registers” on page 7).

- **RDRF: Receive Data Register Full**

0: No data has been received since the last read of SPI_RDR.

1: A data has been received and the receive data has been transferred from the serializer in SPI_RDR since the last read of SPI_RDR.

- **TDRE: Transmit Data Register Empty**

0: A data has been written in SPI_TDR and not yet transferred to the serializer.

1: The last data written in the Transmit Data Register has been transferred to the serializer.

TDRE equals zero when the SPI is disabled or at reset. The SPI enable command sets this bit to one.

- **MODF: Mode Fault Error**

0: No Mode Fault has been detected since the last read of SPI_SR.

1: A Mode Fault occurred since the last read of SPI_SR.

- **SPIOVRE: Overrun Error**

0: No overrun has been detected since the last read of SPI_SR.

1: An overrun has occurred since the last read of SPI_SR.

An overrun occurs when SPI_RDR is loaded at least twice from the serializer since the last read of the SPI_RDR.

- **REND: Reception End**

0: No end of PDC reception has been detected since last read of SPI_SR.

1: At least one end of PDC reception occurs since last read of SPI_SR.

- **TEND: Transfer End**

0: No end of PDC transfer has been detected since last read of SPI_SR.

1: At least one end of PDC transfer occurs since last read of SPI_SR.

- **SPIENS: SPI Enable**

0: SPI is disabled.

1: SPI is enabled.

- **SPCK, MISO, MOSI, NPCS0, NPCS1, NPCS2, NPCS3: PIO Interrupt**

These bits indicate for each pin when an input logic value change has been detected (rising or falling edge). This is valid whether the PIO is selected for the pin or not.

These bits are reset to zero following a read and at reset.

0: No input change has been detected on the corresponding pin since the register was last read.

1: At least one input change has been detected on the corresponding pin since the register was last read.

SPI Interrupt Enable Register

Name: SPI_IER
Access: Write-only
Base Address: 0x074

SPI Interrupt Disable Register

Name: SPI_IDR

Access: Write-only

Base Address: 0x078

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	NPCS3	NPCS2	NPCS1	NPCS0	MOSI	MISO	SPCK
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	TEND	REND	SPIOVRE	MODF	TDRE	RDRF

- **RDRF: Receive Data Register Full Interrupt Mask**

0: Receive Data Register Full Interrupt is disabled.

1: Receive Data Register Full Interrupt is enabled.

- **TDRE: Transmit Data Register Empty Interrupt Mask**

0: Transmit Data Register Empty Interrupt is disabled.

1: Transmit Data Register Empty Interrupt is enabled.

- **MODF: Mode Fault Error Interrupt Mask**

0: Mode Fault Interrupt is disabled.

1: Mode Fault Interrupt is enabled.

Only used in Master Mode.

- **SPIOVRE: Overrun Error Interrupt Mask**

0: Overrun Error Interrupt is disabled.

1: Overrun Error Interrupt is enabled.

- **REND: PDC Reception End Interrupt Mask**

0: Reception End Interrupt is disabled.

1: Reception End Interrupt is enabled.

- **TEND: PDC Transfer End Interrupt Mask**

0: Transfer End Interrupt is disabled.

1: Transfer End Interrupt is enabled.

- **SPCK, MISO, MOSI, NPCS0, NPCS1, NPCS2, NPCS3: PIO Interrupt Mask**

These bits show which pins have interrupts enabled. They are updated when interrupts are enabled or disabled by writing to SPI_IER or SPI_IDR.

0: Interrupt is not enabled on the corresponding input pin.

1: Interrupt is enabled on the corresponding input pin.

SPI Interrupt Mask Register

Name: SPI_IMR
Access: Read-only
Base Address: 0x07C

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

23	22	21	20	19	18	17	16
–	NPCS3	NPCS2	NPCS1	NPCS0	MOSI	MISO	SPCK

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	TEND	REND	SPIOVRE	MODF	TDRE	RDRF

- **RDRF: Receive Data Register Full Interrupt Mask**

0: Receive Data Register Full Interrupt is disabled.

1: Receive Data Register Full Interrupt is enabled.

- **TDRE: Transmit Data Register Empty Interrupt Mask**

0: Transmit Data Register Empty Interrupt is disabled.

1: Transmit Data Register Empty Interrupt is enabled.

- **MODF: Mode Fault Error Interrupt Mask**

0: Mode Fault Interrupt is disabled.

1: Mode Fault Interrupt is enabled.

Only used in Master Mode.

- **SPIOVRE: Overrun Error Interrupt Mask**

0: Overrun Error Interrupt is disabled.

1: Overrun Error Interrupt is enabled.

- **REND: PDC Reception End Interrupt Mask**

0: Reception End Interrupt is disabled.

1: Reception End Interrupt is enabled.

- **TEND: PDC Transfer End Interrupt Mask**

0: Transfer End Interrupt is disabled.

1: Transfer End Interrupt is enabled.SPCK, MISO, MOSI, NPCS0, NPCS1, NPCS2, NPCS3: PIO Interrupt Mask

These bits show which pins have interrupts enabled. They are updated when interrupts are enabled or disabled by writing to SPI_IER or SPI_IDR.

0: Interrupt is not enabled on the corresponding input pin.

1: Interrupts is enabled on the corresponding input pin.

SPI Receive Data Register

Name: SPI_RDR
Access: Read-only
Base Address: 0x080

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

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

15	14	13	12	11	10	9	8
RD							

7	6	5	4	3	2	1	0
RD							

Note: When reading this register, RDRF bit is cleared in the SPI_RDR.

Note: When debugging, to avoid clearing RDRF bit, users should use ghost registers (see “Ghost Registers” on page 7).

- **RD[15:0]: Receive Data**

Data received by the SPI interface is stored in this register. Data stored is right-justified. Unused bits are read at zero.

- **PCS[3:0]: Peripheral Chip Select Status**

In Master Mode only, these bits indicate the value on the NPCS pins at the end of a transfer.

SPI Transmit Data Register

Name: SPI_TDR
Access: Write-only
Base Address: 0x084

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

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

15	14	13	12	11	10	9	8
TD							

7	6	5	4	3	2	1	0
TD							

- **TD[15:0]: Transmit Data**

Data that is to be transmitted by the SPI is stored in this register. Information to be transmitted must be written to the transmit data register in a right-justified format.

- **PCS[3:0]: Peripheral Chip Select**

This field is only used if multiple peripheral select is active (PS = 1).

If PCSDEC = 0:

PCS[3:0] = xxx0 NPCS[3:0] = 1110
PCS[3:0] = xx01 NPCS[3:0] = 1101
PCS[3:0] = x011 NPCS[3:0] = 1011
PCS[3:0] = 0111 NPCS[3:0] = 0111
PCS[3:0] = 1111 forbidden (no peripheral is selected)

where x = don't care.

If PCSDEC = 1:

NPCS[3:0] output signals = PCS[3:0].

SPI Chip Select Register 0..3

Name: SPI_CSR0...SPI_CSR3
Access: Read/Write
Base Address: 0x090...0x09C

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

- **CPOL: Clock Polarity**

0: The inactive state value of SPCK is logic level zero.

1: The inactive state value of SPCK is logic level one.

CPOL is used to determine the inactive state value of the serial clock (SPCK). It is used with NCPHA to produce a desired clock/data relationship between master and the slave devices.

- **NCPHA: Clock Phase**

0: Data is changed on the leading edge of SPCK and captured on the following edge of SPCK.

1: Data is captured on the leading edge of SPCK and changed on the following edge of SPCK.

NCPHA determines which edge of SPCK causes data to change and which edge causes data to be captured.

NCPHA is used with CPOL to produce a desired clock/data relationship between master and slave devices.

- **BITS[3:0]: Bits Per Transfer**

The BITS field determines the number of data bits transferred. Reserved values default to 8 bits.

BITS[3:0]	Bits Per Transfer
0000	8
0001	9
0010	10
0011	11
0100	12
0101	13
0110	14

BITS[3:0]	Bits Per Transfer
0111	15
1000	16
1001	Reserved
1010	Reserved
1011	Reserved
11XX	Reserved

- **SCBR[7:0]: Serial Clock Baud Rate**

The SPI interface uses a modulus counter to derive SPCK baud rate from the SPI Master Clock, selected between CORECLK and CORECLK/32. Baud rate is selected by writing a value from 2 to 255 into SCBR[7:0]. The following equation determines the SPCK baud rate:

$$SPCK\_Baud\_Rate = SPI\_Master\_Clock\_Frequency / (2 \times SCBR[7:0])$$

Note: Giving SCBR[7:0] a value of zero or one disables the baud rate generator. SPCK is disabled and assumes its inactive state value. No serial transfers occur. At reset, baud rate is disabled.

- **DLYBS[7:0]: Delay Before SPCK**

This field defines the length of delay from NPCS valid to the first valid SPCK transition.

When DLYBS[7:0] equals zero, the NPCS valid to SPCK transition is one-half SPCK clock period.

Else, the following equation determines the delay:

$$NPCS\_to\_SPCK\_Delay = DLYBS[7:0\} \times SPI\_Master\_Clock\_Period$$

- **DLYBCT[7:0]: Delay Between Consecutive Transfers**

This field determines the delay between two consecutive transfers with the same peripheral without removing the chip select or the length of delay after transfer before chip select is deselected.

When DLYBCT[7:0] equals zero, the delay is equal to four SPI Master Clock periods.

Else, the following equation determines the delay:

$$Delay\_After\_Transfer = 32 \times DLYBCT[7:0] \times SPI\_Master\_Clock\_Period$$

Timing Diagrams

Figure 66. SPI in Master Mode, Phase = 0

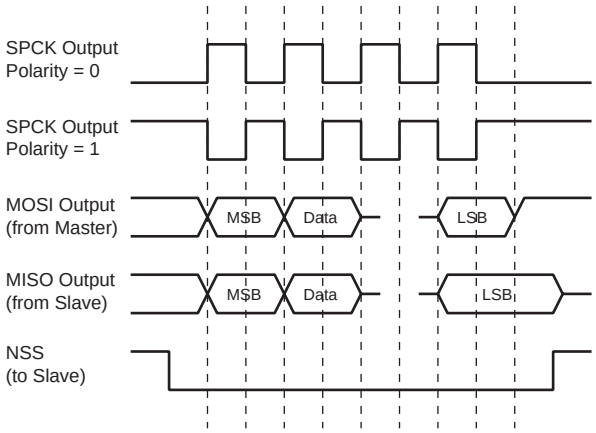


Figure 67. SPI in Master Mode, Phase = 1

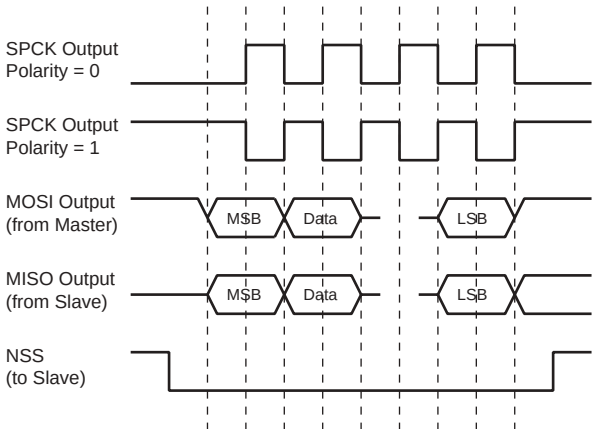


Figure 68. DLYBCS, DLYBS and DLYBCT

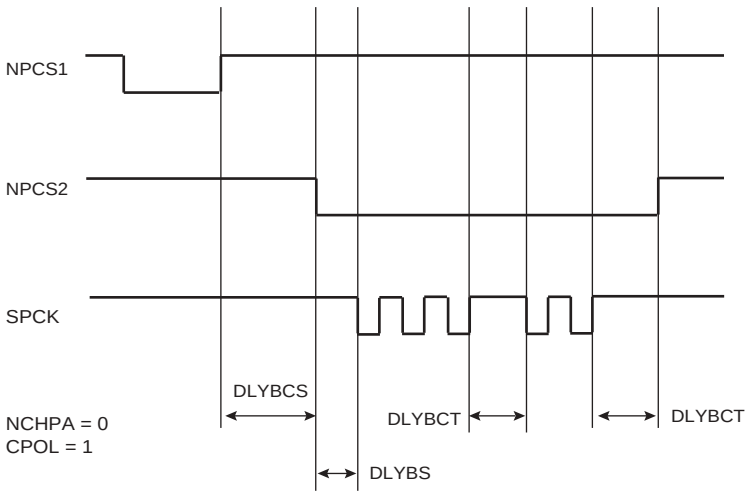


Figure 69. SPI Timings

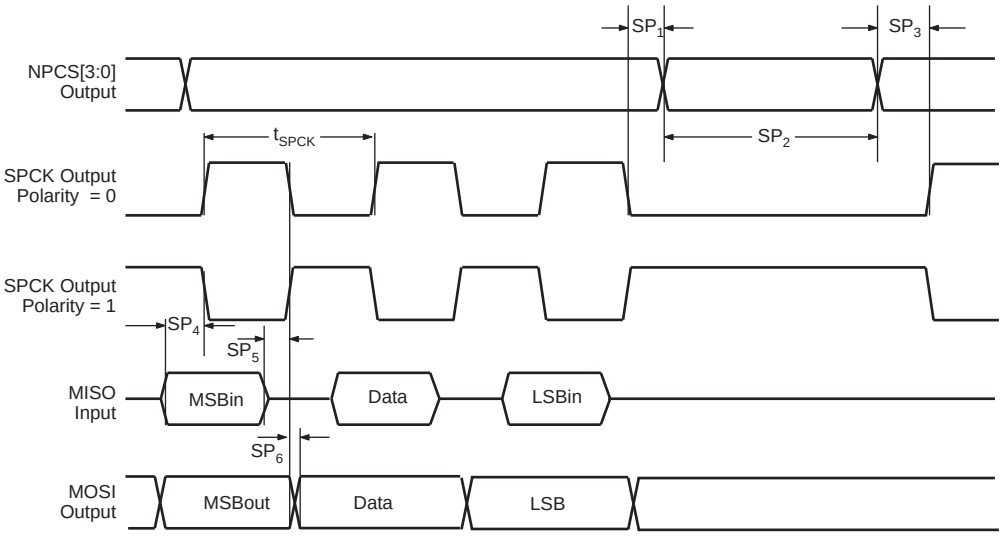


Table 47. SPI Timing Parameters

Symbol	Parameter	Minimum	Maximum	Unit
t_{SPCK}	SPI Operating Period	4 x (t_{CP})	16320 x (t_{CP})	ns
f_{SPCK}	SPI Operating Frequency	1/16320 x (t_{CP})	1/4 x (t_{CP})	GHz
SP_1	Delay Before NPCS[3:0]	4 x (t_{CP})	261120 x (t_{CP})	ns
SP_2	Delay Between Chip Select	6 x (t_{CP})	8160 x (t_{CP})	ns
SP_3	Delay Before SPCK	2 x (t_{CP})	8160 x (t_{CP})	ns
SP_4	MISO/SPCK Setup Time		18	ns
SP_5	MOSI/SPCK Hold Time	0		ns
SP_6	MOSI valid after SPCK edge		7	ns

Unified Parallel I/O Controller (UPIO)

Overview

The AT91SAM7A2 microcontroller has 32 unified programmable I/O lines. These lines are controlled by a UPIO module and are not multiplexed with other module pins. The UPIO controller also provides an internal interrupt signal to the Generic Interrupt Controller.

Output Selection

The user can enable each individual I/O signal as an output with the UPIO_OER (Output Enable) register or as an input with the UPIO_ODR (Output Disable) register. The output status of the I/O signals can be read in the register UPIO_OSR (Output Status).

I/O Levels

Each pin can be configured to be driven high or low. The level is defined in two different ways, according to the following conditions:

If a pin is defined as output, the level is programmed using the registers UPIO_SODR (Set Output Data) and UPIO_CODR (Clear Output Data). In this case, the programmed value can be read in UPIO_ODSR (Output Data Status).

If a pin is not defined as output, the level is determined by the external circuit.

In all cases, the level on the pin can be read in the register UPIO_PDSR (Pin Data Status).

Interrupts

Each parallel I/O can be programmed to generate an interrupt when a level change occurs. This is controlled by the UPIO_IER (Interrupt Enable) and UPIO_IDR (Interrupt Disable) registers which enable/disable the I/O interrupt by setting/clearing the corresponding bit in the UPIO_IMR.

When a change in level occurs, the corresponding bit in the UPIO_SR (Interrupt Status) is set whether it is defined as input or output. If the corresponding interrupt in UPIO_IMR (Interrupt Mask) is enabled, the UPIO interrupt is asserted.

When the UPIO_SR is read, the register is automatically cleared.

User Interface

Each individual I/O is associated with a bit position in the Parallel I/O user interface registers. If a parallel I/O line is not defined, writing to the corresponding bits has no effect. Undefined bits are read at zero.

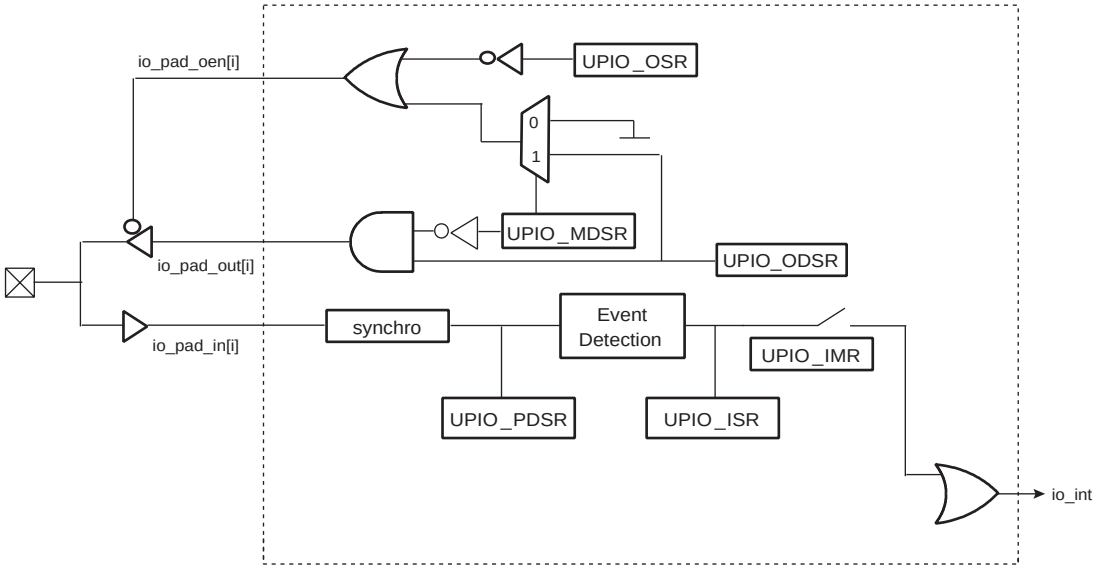
Multi-Driver (Open Drain)

Each I/O can be programmed for multi-driver option. This means that the I/O is configured as open drain (can only drive a low level) in order to support external drivers on the same pin. An external pull-up is necessary to guarantee a logic level of one when the pin is not being driven.

Registers UPIO_MDER (Multi-Driver Enable) and UPIO_MDDR (Multi-Driver Disable) control this option. Multi-driver can be selected whether the I/O pin is controlled by the UPIO Controller. UPIO_MDSR (Multi-Driver Status) indicates which pins are configured to support external drivers.

Block Diagram

Figure 70. UPIO Block Diagram



Special Multiplexed PIO

Two dedicated PIO pins, CORECLK and NWAIT, are multiplexed with other signals. CORECLK is multiplexed with PIO[31] and is selected by bit 31 in the PIO_MR register. NWAIT is multiplexed with PIO[30] and is selected by bit 30 in the PIO_MR register.

Figure 71. CORECLK Multiplexing

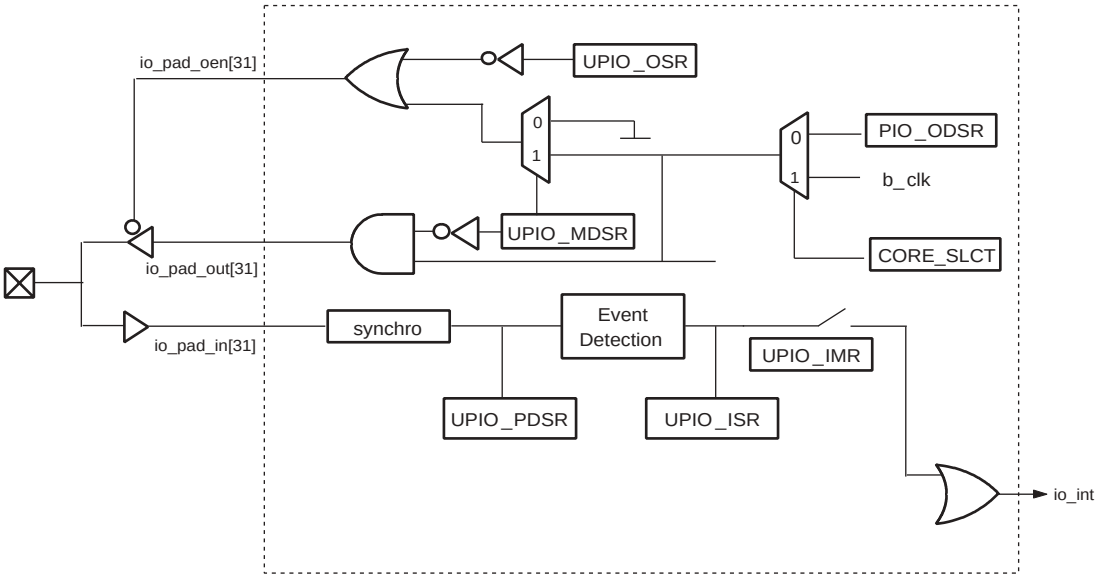
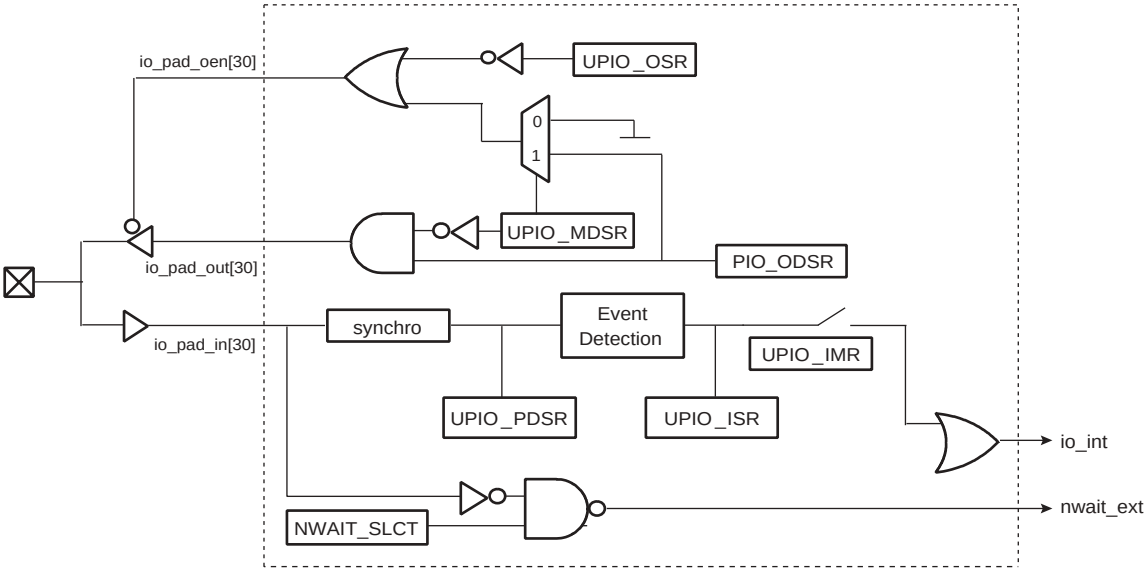


Figure 72. NWAIT Multiplexing



Power Management

The UPIO is provided with a power management block allowing optimization of power consumption (see “Power Management Block” on page 22).

Unified Parallel I/O Controller (UPIO) Memory Map

Base Address: 0xFFFD8000

Table 48. UPIO Memory Map

Offset	Register	Name	Access	Reset State
0x000 - 0x00C	Reserved	–	–	–
0x010	Output Enable Register	UPIO_OER	Write-only	–
0x014	Output Disable Register	UPIO_ODR	Write-only	–
0x018	Output Status Register	UPIO_OSR	Read-only	0x00000000
0x01C - 0x02C	Reserved	–	–	–
0x030	Set Output Data Register	UPIO_SODR	Write-only	–
0x034	Clear Output Data Register	UPIO_CODR	Write-only	–
0x038	Output Data Status Register	UPIO_ODSR	Read-only	0x00000000
0x03C	Pin Data Status Register	UPIO_PDSR	Read-only	0xXXXXXXXX
0x040	Multi-Driver Enable Register	UPIO_MDER	Write-only	–
0x044	Multi-Driver Disable Register	UPIO_MDDR	Write-only	–
0x048	Multi-Driver Status Register	UPIO_MDSR	Read-only	0x00000000
0x04C	Reserved	–	–	–
0x050	Enable Clock Register	UPIO_ECR	Write-only	–
0x054	Disable Clock Register	UPIO_DCR	Write-only	–
0x058	Power Management Status Register	UPIO_PMSR	Read-only	0x00000000
0x05C	Reserved	–	–	–
0x060	Control Register	UPIO_CR	Write-only	–
0x064	Mode Register	UPIO_MR	Read/Write	0x00000000
0x068 - 0x06C	Reserved	–	–	–
0x070	Status Register	UPIO_SR	Read-only	0x00000000
0x074	Interrupt Enable Register	UPIO_IER	Write-only	–
0x078	Interrupt Disable Register	UPIO_IDR	Write-only	–
0x07C	Interrupt Mask Register	UPIO_IMR	Read-only	0x00000000

UPIO Output Enable Register

Name: UPIO_OER
Access: Write-only
Base Address: 0x010

UPIO Output Disable Register

Name: UPIO_ODR
Access: Write-only
Base Address: 0x014

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

- **Px: Px Pin**

0: The corresponding PIO is input on this line.

1: The corresponding PIO is output on this line.

UPIO Output Status Register

Name: UPIO_OSR
Access: Read-only
Base Address: 0x018

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

- **Px: Px Pin**

0: The corresponding PIO is input on this line.

1: The corresponding PIO is output on this line.

UPIO Set Output Data Register

Name: UPIO_SODR
Access: Write-only
Base Address: 0x030

UPIO Clear Output Data Register

Name: UPIO_CODR
Access: Write-only
Base Address: 0x034

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

- **Px: Px Pin**

0: The output data for the corresponding pin is programmed to 0.

1: The output data for the corresponding pin is programmed to 1.

UPIO Output Data Status Register

Name: UPIO_ODSR
Access: Read-only
Base Address: 0x038

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

- **Px: Px Pin**

0: The output data for the corresponding pin is programmed to 0.

1: The output data for the corresponding pin is programmed to 1.

UPIO Pin Data Status Register

Name: UPIO_PDSR
Access: Read-only
Base Address: 0x030

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

- **Px: Px Pin**

0: The corresponding pin is at logic 0.

1: The corresponding pin is at logic 1.

UPIO Multi-driver Enable Register

Name: UPIO_MDER
Access: Write-only
Base Address: 0x040

UPIO Multi-driver Disable Register

Name: UPIO_MDDR
Access: Write-only
Base Address: 0x044

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

- **Px: Px Pin**

0: PIO is not configured as an open drain.

1: PIO is configured as an open drain.

UPIO Multi-driver Status Register

Name: UPIO_MDSR
Access: Read-only
Base Address: 0x048

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

- **Px: Px Pin**

0: PIO is not configured as an open drain.

1: PIO is configured as an open drain.

UPIO Enable Clock Register

Name: UPIO_ECR
Access: Write-only
Base Address: 0x050

UPIO Disable Clock Register

Name: UPIO_DCR
Access: Write-only

Base Address: 0x054

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
–	–	–	–	–	–	–	PIO

- **PIO: PIO Clock Status**

0: PIO clock disabled.

1: PIO clock enabled.

UPIO Power Management Status Register

Name: UPIO_PMSR
Access: Read-only
Base Address: 0x058

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
–	–	–	–	–	–	–	PIO

- **PIO: PIO Clock Status**

0: PIO clock disabled.

1: PIO clock enabled.

UPIO Control Register

Name: UPIO_CR
Access: Write-only
Base Address: 0x060

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
–	–	–	–	–	–	–	SWRST

- **SWRST: PIO Software Reset**

0: No effect.

1: Resets the PIO.

A software-triggered hardware reset of the PIO is performed. It resets all the registers (except the UPIO_PMSR).

UPIO Mode Register

Name: UPIO_MR
Access: Read/Write
Base Address: 0x064

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

- **CLK_SLCT: Core Clock Select**

0: Normal function PIO select.

1: Core Clock select in output mode is connected to PIO[31].

- **NWAIT_SLCT: NWait External Select**

0: Normal function PIO select.

1: PIO[30] is connected to nWAIT external to the EBI module.

UPIO Status Register

Name: UPIO_SR
Access: Read-only
Base Address: 0x070

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

Note: This register is a “read-active” register, thus reading it can affect the state of some bits. When read UPIO_SR register, all bits set to logical 1 are cleared. When debugging, to avoid this behavior, users should use ghost registers (see “Ghost Registers” on page 7).

- **Px: PIO Interrupt Status**

These bits indicate for each pin when an input logic value change has been detected (rising or falling edge).

These bits are reset to zero following a read and at reset.

0: No input change has been detected on the corresponding pin since the register was last read.

1: At least one input change has been detected on the corresponding pin since the register was last read.

UPIO Interrupt Enable Register

Name: UPIO_IER
Access: Write-only
Base Address: 0x074

UPIO Interrupt Disable Register

Name: UPIO_IDR
Access: Write-only
Base Address: 0x078

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

- **Px: PIO Interrupt Mask**

0: Interrupt is not enabled on the corresponding input pin.

1: Interrupt is enabled on the corresponding input pin.

UPIO Interrupt Mask Register

Name: UPIO_IMR
Access: Read-only
Base Address: 0x07C

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

- **Px: PIO Interrupt Mask**

0: Interrupt is not enabled on the corresponding input pin.

1: Interrupt is enabled on the corresponding input pin.

Power Management Controller (PMC)

Overview

The AT91SAM7A2 Power Management Controller allows optimization of power consumption. The PMC controls the clock inputs of the ARM core and of the PDC module.

When the ARM core clock is disabled, the current instruction is finished before the clock is stopped. The clock can be re-enabled by any interrupt or by any hardware reset.

The PDC clock cannot be stopped during PDC transfers (if any TCR register is different from 0). The request is stored but the Power Management Controller will wait until the end of the transfers to stop the PDC clock (the Power Management Controller needs an authorization from the PDC before it stops the PDC clock).

Power Management Controller (PMC) Memory Map

Base Address: 0xFFFF4000

Table 49. PMC Memory Map

Offset	Register	Name	Access	Reset State
0x000 - 0x04C	Reserved	–	–	–
0x050	Enable Clock Register	PMC_ECR	Write-only	–
0x054	Disable Clock Register	PMC_DCR	Write-only	–
0x058	Power Management Status Register	PMC_PMSR	Read-only	0x00000001

PMC Enable Clock Register

Name: PMC_ECR
Access: Write-only
Base Address: 0x050

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
–	–	–	–	–	–	PDC	–

- **PDC: PDC Clock**

0: PDC clock is disabled, or user instructed PDC clock to be disabled during PDC transfers. The PDC clock is disabled when all TCR registers reach 0 (all transfers are finished).

1: PDC clock is enabled.

PMC Disable Clock Register

Name: PMC_DCR
Access: Write-only
Base Address: 0x054

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
–	–	–	–	–	–	PDC	ARM

- **ARM: ARM Clock**

0: ARM core clock is disabled.

1: ARM core clock is enabled.

- **PDC: PDC Clock**

0: PDC clock is disabled, or user instructed PDC clock to be disabled during PDC transfers. The PDC clock is disabled when all TCR registers reach 0 (all transfers are finished).

1: PDC clock is enabled.

PMC Power Management Status Register

Name: PMC_PMSR
Access: Read-only
Base Address: 0x058

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
–	–	–	–	–	–	PDC	ARM

- **ARM: ARM Clock**

0: ARM core clock is disabled.

1: ARM core clock is enabled.

- **PDC: PDC Clock**

0: PDC clock is disabled, or user instructed PDC clock to be disabled during PDC transfers. The PDC clock is disabled when all TCR registers reach 0 (all transfers are finished).

1: PDC clock is enabled.

Controller Area Network (CAN)

Overview

The Controller Area Network (CAN) is a serial communications protocol that supports distributed real-time control with a very high level of security.

Possible applications range from high-speed networks to low-cost multiplex wiring, e.g., in automotive electronics, engine control units, sensors, anti-skid systems, etc. may be connected using CAN with bit rates up to 1 Mbit/s. At the same time, it is a cost-effective application for vehicle body electronics, such as lamp clusters, electric windows, etc. in replacement of the wiring harness otherwise required.

This section describes how to achieve compatibility between any two CAN implementations. Compatibility, however, has different aspects regarding, e.g., electrical features and the interpretation of data to be transferred.

To achieve design transparency and implementation flexibility, the CAN has been subdivided into different layers according to the ISO/OSI Reference Model:

- Data Link Layer
  - Logical Link Control (LLC) sublayer
  - Medium Access Control (MAC) sublayer
- Physical Layer

Note that in previous versions of the CAN specification, services and functions of the LLC and MAC sublayers of the Data Link Layer were described in layers denoted as 'object layer' and 'transfer layer'.

The tasks of the LLC sublayer include

- determining which messages received by the LLC sublayer are to be accepted
- providing services for data transfer and for remote data request
- providing the means for recovery management and overload notifications

There is much flexibility in defining object handling.

The tasks of the MAC sublayer concern primarily the transfer protocol, i.e., controlling framing, performing arbitration, error checking, error and fault confinement. Within the MAC sublayer, it is determined whether the bus is free to start signaling a new transmission or whether a reception is just starting. Also, some general features of bit timing are regarded as a task of the MAC sublayer. One of the constraints of the MAC sublayer is that it is not possible to make modifications.

The task of the physical layer is the actual transfer of bits between the different nodes with respect to electrical properties. Within a network, the physical layer has to be the same for all nodes. There are, however, many possible implementations of a physical layer.

In the following, the MAC sublayer and a small part of the LLC sublayer of the Data Link Layer are defined and the consequences of the CAN protocol on the surrounding layers are described.

Basic Concepts

CAN Properties

The CAN has the following properties:

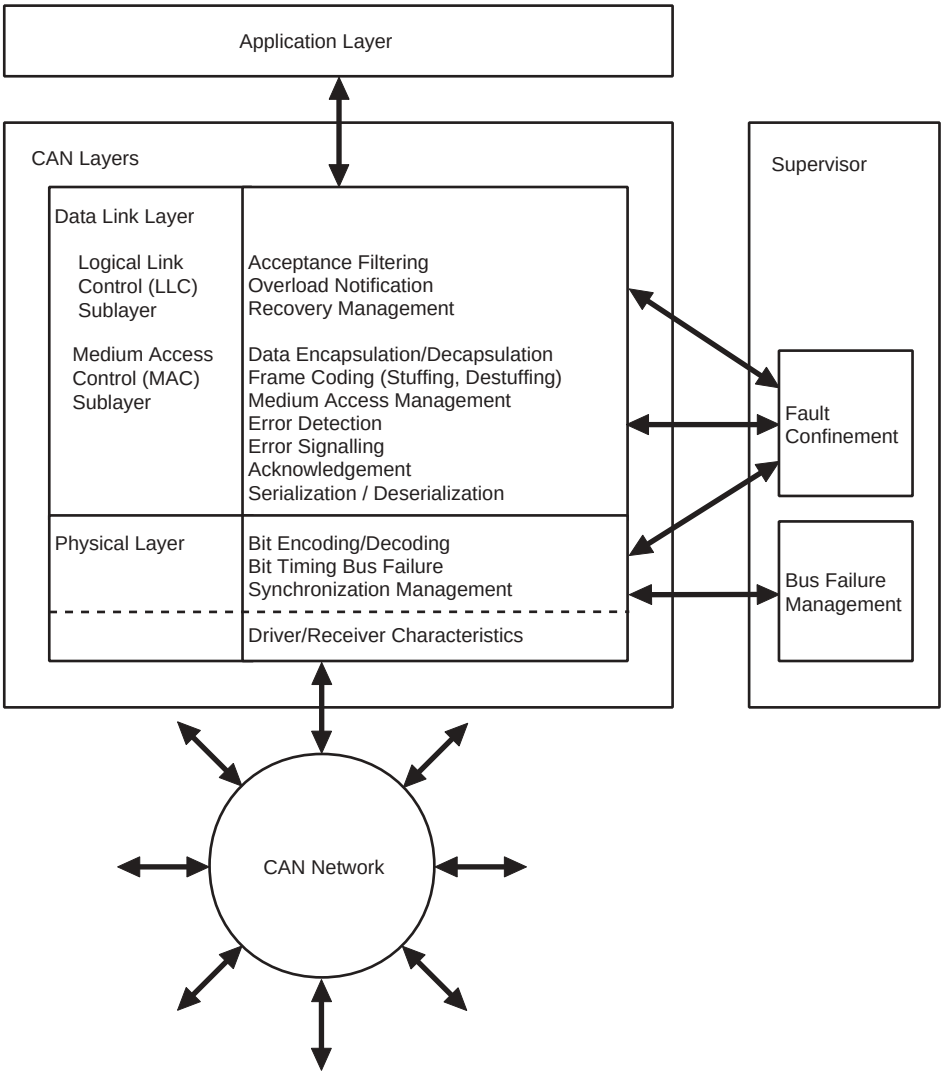
- Prioritization of messages
- Guarantee of latency times
- Configuration flexibility
- Multicast reception with time synchronization
- System wide data consistency
- Multi-master functions
- Error detection and error signaling
- Automatic retransmission of corrupted messages as soon as the bus is idle again
- Distinction between temporary errors and permanent failures of nodes and autonomous switching off of defect nodes.

Layered Structure of a CAN Node

The layered architecture of the CAN is compliant with the ISO/OSI reference model:

- The LLC sublayer is concerned with message filtering, overload notification and recovery management.
- The MAC sublayer represents the kernel of the CAN protocol. It presents messages received from the LLC sublayer and accepts messages to be transmitted to the LLC sublayer. The MAC sublayer is responsible for Message Framing, Arbitration, Acknowledgement, Error Detection and Signalling. The MAC sublayer is supervised by a management entity called Fault Confinement which is a self-checking mechanism for distinguishing short disturbances from permanent failures.
- The physical layer defines how signals are actually transmitted and deals with the description of bit timing, bit encoding, and synchronization. Within this description, the physical layer is not defined, as it will vary according to the requirements of individual applications (for example, transmission medium and signal level implementations).

Figure 73. Layered Structure of a CAN Node



Messages

Information on the bus is sent in fixed format messages of different but limited length (see “Message Transfer” on page 253). When the bus is free, any connected unit may start to transmit a new message.

Information Routing

In CAN systems, a CAN node does not make use of any information about the system configuration (e.g., node addresses). This has several important consequences:

- System flexibility: Nodes can be added to the CAN network without requiring any change in the software or hardware of any node and application layer.
- Message routing: The content of a message is named by an identifier. The identifier does not indicate the destination of the message, but describes the meaning of the data, so that all nodes in the network are able to decide, by message filtering, whether the data is to be acted upon by them or not.
- Multicast: As a consequence of the message filtering, any number of nodes can receive and simultaneously act upon the same message.

- Data consistency: Within a CAN network, it is guaranteed that a message is simultaneously accepted either by all nodes or by no node. Thus, data consistency of a system is achieved by the concepts of multicast and by error handling.

Bit Rate

The speed of the CAN may be different in different systems. However, in a given system the bit-rate is uniform and fixed.

Priorities

The identifier defines a static message priority during bus access.

Remote Data Request

By sending a remote frame, a node requiring data may request another node to send the corresponding data frame. The data frame and the corresponding remote frame are named by the same identifier.

Multi-master Function

When the bus is free, any unit may start transmitting a message. The unit with the message of highest priority to be transmitted gains bus access.

Arbitration

Whenever the bus is free, any unit may start transmitting a message. If two or more units start transmitting messages at the same time, the bus access conflict is resolved by bit-wise arbitration using the identifier. The mechanism of arbitration guarantees that neither information nor time is lost. If a data frame and a remote frame with the same identifier are initiated at the same time, the data frame prevails over the remote frame. During arbitration every transmitter compares the level of the bit transmitted with the level that is monitored on the bus. If these levels are equal the unit may continue to send. When a recessive level is sent and a dominant level is monitored (see “Bus Values” on page 252), the unit has lost arbitration and must withdraw without sending one more bit.

Data Transfer Security

In order to achieve the utmost security of data transfer, powerful measures for error detection, signalling and self-checking are implemented in every CAN node.

Error Detection

For detecting errors, the following methods are used:

- Monitoring (transmitters compare the bit levels to be transmitted with the bit levels detected on bus)
- Cyclic Redundancy Check (CRC)
- Bit stuffing
- Message frame check

Performance of Error Detection

The error detection mechanisms have the following properties:

- All global errors are detected by means of monitoring.
- All local errors at transmitters are detected by means of monitoring.
- Up to 5 randomly distributed errors in a message are detected by means of CRC.
- Burst errors of length less than 15 in a message are detected by means of CRC.
- Errors of any odd number in a message are detected by means of CRC.

Total residual error probability for undetected corrupted messages is less than

$$\text{Message Error Rate} \times 4.7 \times 10^{11}$$

Error Signaling and Recovery Time

Corrupted messages are flagged by any node detecting an error. Such messages are aborted and will be retransmitted automatically. The recovery time from detecting an error until the start of the next message is at most 31 bit times if there is no further error.

Fault Confinement

CAN nodes are able to distinguish short disturbances from permanent failures. Defective nodes are switched off.

Connections

The CAN serial communication link is a bus to which a number of units may be connected. This number has no theoretical limit. In practice, the total number of units is limited by delay times and/or electrical loads on the bus line.

Single Channel

The bus consists of a single bidirectional channel that carries bits. From this data, resynchronization information can be derived. The way in which this channel is implemented is not fixed in this document, e.g., single wire (plus ground), two differential wires, optical fibers, etc.

Bus Values

The bus can have one of two complementary logical values: dominant or recessive. During simultaneous transmission of dominant and recessive bits, the resulting bus value will be dominant. For example, in case of a wired-AND implementation of the bus, the dominant level is represented by a logical 0 and the recessive level by a logical 1. Physical states (e.g., electrical voltage, light) that represent the logical levels are not given in this document.

Acknowledgement

All receivers check the consistency of the message being received and acknowledge a consistent message and flag an inconsistent message.

Channel Overview

The CAN module has a set of buffers, also called channels or mailboxes. Table 50 presents the number of channels of each CAN module present in the AT91SAM7A2 microcontroller.

Table 50. Number of Channels

Module	Number of Channels
CAN0	16 channels
CAN1	16 channels
CAN2	32 channels
CAN3	16 channels

Each mailbox is assigned an identifier and can be set to transmit or receive.

It is possible to reconfigure mailboxes dynamically.

When the CAN module receives a message, it checks the mailboxes in order to see if there is a receive mailbox with the same identifier as the message. If such a mailbox is found, the message is stored in it. If several mailboxes are configured with the same identifier, only the smaller channel store the message. If no mailbox is found, the message is discarded.

When the CAN module has to transmit a message, the message length, data and identifier are written to a mailbox set in transmission. If several messages in different mailboxes are waiting to be transmitted, the mailbox sending the highest priority message (smaller identifier) sends its message first.

If a remote frame is received, the CAN module checks the remote identifier against the mailbox identifier. If a match is found, and if the mailbox is set to automatically reply to the remote frame, the CAN module automatically sends a message with the identifier and data contained in that mailbox.

Message Transfer

Definition of Transmitter/Receiver

A node originating a message is called the transmitter of that message. The node stays transmitter until the bus is idle or the node loses arbitration.

A node is called receiver of a message if it is not the transmitter of that message and the bus is not idle.

Frame Formats

There are two different formats that differ in the length of the identifier field:

Table 51. Identifier Length within Standard and Extended Frames

Frame Format	Identifier Length
Standard frame	11 bits
Extended frame	29 bits

Frame Types

Message transfer is manifested and controlled by four different frame types:

- A data frame carries data from a transmitter to the receivers.
- A remote frame is transmitted by a bus unit to request the transmission of the data frame with the same identifier.
- An error frame is transmitted by any unit on detecting a bus error.
- An overload frame is used to provide for an extra delay between the preceding and the succeeding data or remote frame.

Data frames and remote frames can be used both in standard frame format and extended frame format. They are separated from preceding frames by an interframe space.

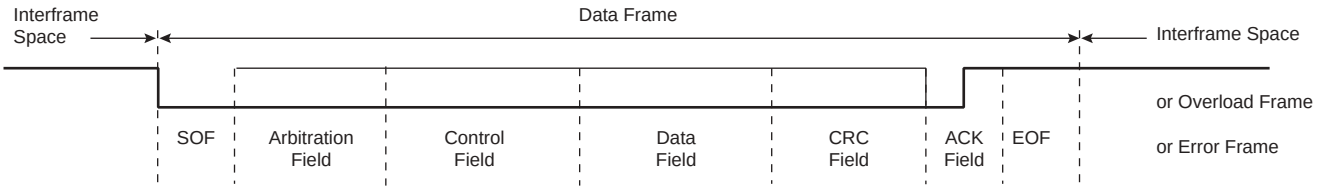
Data Frame

A data frame is composed of seven different bit fields:

- Start Of Frame (SOF)
- Arbitration field
- Control field
- Data field
- CRC field
- ACK field
- End Of Frame (EOF)

The data field can be of length zero.

Figure 74. Data Frame



Start Of Frame (Standard and Extended Format)

The Start of Frame (SOF) marks the beginning of data frames and remote frames. It consists of a single dominant bit.

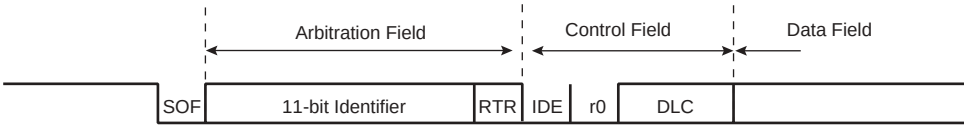
A node is only allowed to start transmission when the bus is idle. All nodes have to synchronize to the leading edge caused by start of frame of the node starting transmission first.

Arbitration Field

The format of the arbitration field is different for standard format and extended format frames.

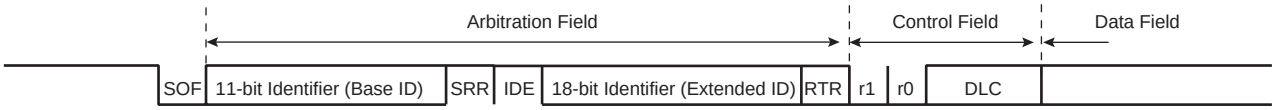
In standard format, the arbitration field consists of the 11-bit identifier and the RTR bit. The identifier bits are denoted ID[10:0].

Figure 75. Standard Format



In extended format, the arbitration field consists of the 29-bit identifier, the SRR bit, the IDE bit, and the RTR bit. The identifier bits are denoted ID[28:0]. The base identifier ID[10:0] are sent first, and extended identifier ID[28:11] later.

Figure 76. Extended Format



In order to distinguish between standard format and extended format, the reserved bit r1 in previous CAN specifications version 1.0-1.2 now is denoted as IDE bit.

- Identifier - Standard Format

The identifier's length is 11 bits and corresponds to the base ID in extended format. These bits are transmitted in the order from ID10 to ID0. The least significant bit is ID0. The 7 most significant bits ID[10:4] must not be all recessive.

- Identifier - Extended Format

In contrast to the standard format, the extended format consists of 29 bits. The format comprises two sections:

  – Base ID: the base ID consists of 11 bits. It is transmitted in the order from ID10 to ID0. It is equivalent to format of the Standard Identifier. The base ID defines the Extended Frame's base priority.
  – Extended ID: the Extended ID consists of 18 bits. It is transmitted in the order of ID28 to ID11.

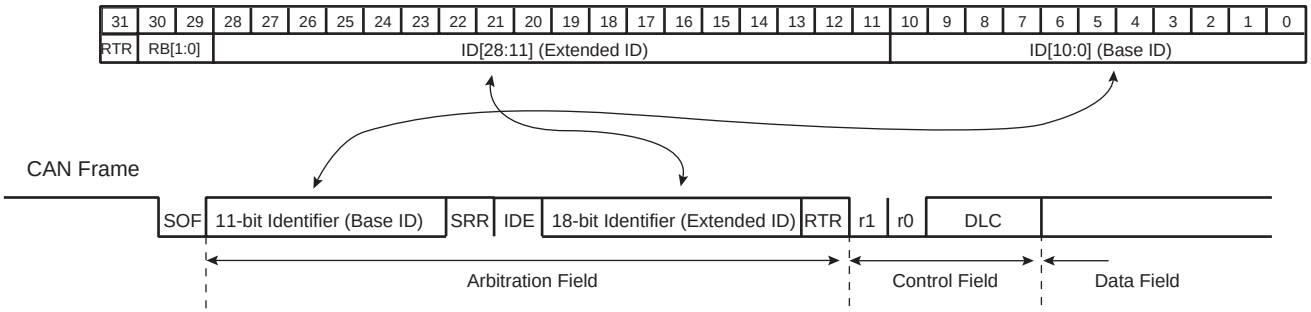
In extended format, the identifier has to be written to the CAN_IRx register in a specific format:

CAN_IRx = ((id & 0x1FFC0000) >> 18) | ((id & 0x3FFFF) << 11)

where (id & 0x1FFC0000) is the 11-bit base ID and (id & 0x3FFFF) is the 18-bit extended ID.

Figure 77 shows how the identifier written in the CAN_IRx register is transmitted to the CAN network.

Figure 77. Identifier Format from CAN_IRx Register to CAN Frame



- RTR Bit (Standard and Extended Formats)

Table 52 shows the RTR bit logical value depending on frame type.

Table 52. RTR Bit Logical Value Depending on Frame Type

Frame Type	RTR Bit Logical Value
Data frame	Dominant
Remote frame	Recessive

In an extended frame, the base ID is transmitted first, followed by the IDE bit and the SRR bit. The extended ID is transmitted after the SRR bit.

- SRR Bit (Extended Format)

The SRR bit (Substitute Remote Request) is a recessive bit. It is transmitted in Extended Frames at the position of the RTR bit in standard frames and so substitutes the RTR bit in the standard frame.

Therefore, collisions of a standard frame and an extended frame, the base ID (see Extended Identifier below) of which is the same as the standard frame's identifier, are resolved in such a way that the standard frame prevails the extended frame.

- IDE Bit (Extended Format)

Table 53 shows the IDE bit logical value and membership according to frame format type.

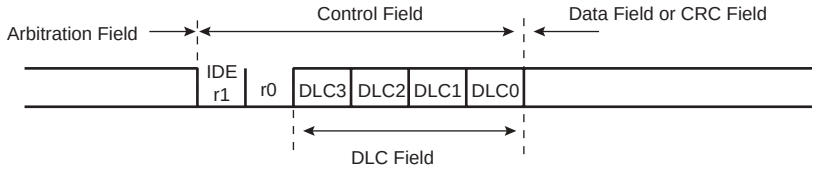
Table 53. IDE Bit Logical Value and Membership Depending on Format Type

Frame Format Type	IDE Bit Belongs to	IDE Bit Logical Value
Standard frame format	Control field	Dominant
Extended frame format	Arbitration field	Recessive

Control Field (Standard and Extended Format)

The control field consists of six bits. The format of the control field is different for standard format and extended format. Frames in standard format include the data length code, the IDE bit, which is transmitted dominant, and the reserved bit r0. Frames in extended format include the data length code and two reserved bits, r1 and r0. The reserved bits have to be sent dominant, but receivers accept dominant and recessive bits in all combinations.

Figure 78. Control Field



- Data Length Code (Standard and Extended Format)

The number of bytes in the data field is indicated by the data length code. The DLC field is four bits wide and is transmitted within the control field. Coding of the number of data bytes by the DLC field is described in Table 54.

Table 54. Data Length Code (D = dominant, R = recessive)

Number of Data (Bytes)	Data Length Code			
	DLC3	DLC2	DLC1	DLC0
0	D	D	D	D
1	D	D	D	R
2	D	D	R	D
3	D	D	R	R
4	D	R	D	D
5	D	R	D	R
6	D	R	R	D
7	D	R	R	R
8	R	D	D	D

Data Frame (Standard and Extended Format)

The admissible numbers of data bytes is 0 to 8. Other values may not be used.

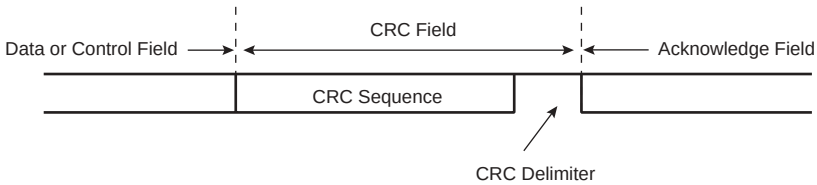
Data Field (Standard and Extended Format)

The data field consists of the data to be transferred within a data frame. It can contain from 0 to 8 bytes, which each contain 8 bits which are transferred MSB first.

CRC Field (Standard and Extended Format)

The CRC (Cyclic Redundancy Check) field contains the CRC sequence followed by a CRC delimiter.

Figure 79. CRC Field



- CRC Sequence (Standard and Extended Format)

The frame check sequence is derived from a cyclic redundancy code best suited for frames with bit counts less than 127 bits (BCH code).

In order to carry out the CRC calculation, the polynomial to be divided is defined as the polynomial, the coefficients of which are given by the de-stuffed bit stream consisting of start of frame, arbitration field, control field, data field (if present) and, for the 15 lowest coefficients, by 0. This polynomial is divided (the coefficients are calculated modulo 2) by the generator-polynomial:

X15 + X14 + X10 + X8 + X7 + X4 + X3 + 1

The remainder of this polynomial division is the CRC sequence transmitted over the bus. In order to implement this function, a 15-bit shift register CRC_RG(14:0) can be used. If NXTBIT denotes the next bit of the bit stream, given by the de-stuffed bit sequence from start of frame until the end of the data field, the CRC sequence is calculated as follows:

```
CRC_RG = 0// initialize shift register
REPEAT
CRCNXT = NXTBIT EXOR CRC_RG[14]
CRC_RG[14:1] = CRC_RG[13:0]// shift left by
CRC_RG[0] = 0// 1 position
IF CRCNXT THEN
CRC_RG[14:0] = CRC_RG[14:0] EXOR 0x4599
ENDIF
UNTIL (CRC sequence starts or there is an error condition)
```

After the transmission/reception of the last bit of the data field, CRC_RG contains the CRC sequence.

- CRC Delimiter (Standard and Extended Format)

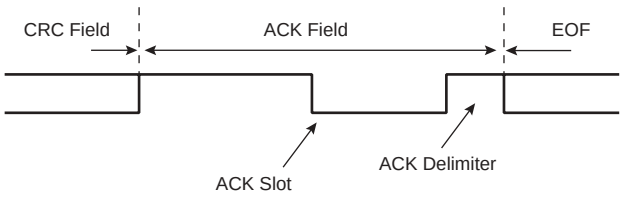
The CRC sequence is followed by the CRC delimiter which consists of a single recessive bit.

ACK Field (Standard and Extended Format)

The ACK (acknowledgement) field is two bits long and contains the ACK slot and the ACK delimiter. In the ACK field, the transmitting node sends two recessive bits.

A receiver that has received a valid message correctly reports this to the transmitter by sending a dominant bit during the ACK slot (it sends ACK).

Figure 80. ACK Field



- ACK Slot

All stations having received the matching CRC sequence report this within the ACK slot by overwriting the recessive bit of the transmitter by a dominant bit.

- ACK Delimiter

The ACK delimiter is the second bit of the ACK field and has to be a recessive bit. As a consequence, the ACK slot is surrounded by two recessive bits (CRC delimiter, ACK delimiter).

End Of Frame (Standard and Extended Format)

Each data frame and remote frame is delimited by seven recessive bits. These bits form the end of frame sequence (EOF).

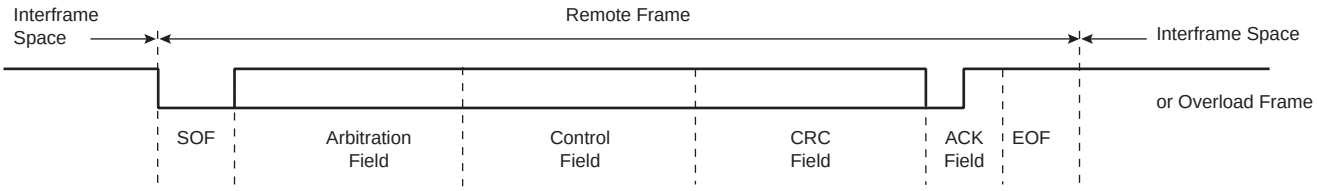
Remote Frame

A station acting as a receiver for certain data can initiate the transmission of the respective data by its source node by sending a remote frame. A remote frame exists both in standard format and in extended format. In both cases, it is composed of six different bit fields:

- Start Of Frame (SOF)
- Arbitration field
- Control field
- CRC field
- ACK field
- End Of Frame

Contrary to data frames, the RTR bit of remote frames is recessive. There is no data field, independent of the values of the data length code which may be signed any value within the admissible range from 0 to 8. The value is the data length code of the corresponding data frame.

Figure 81. Remote Frame



The polarity of the RTR bit (CAN_IRX register) indicates whether a transmitted frame is a data frame (RTR bit dominant) or a remote frame (RTR bit recessive).

If a mailbox is set to reply automatically to the remote frame (RPLYV bit set to logical 1 in CAN_CRx register), the CAN module automatically sends a message with the identifier and data contained in that mailbox after receiving the remote frame.

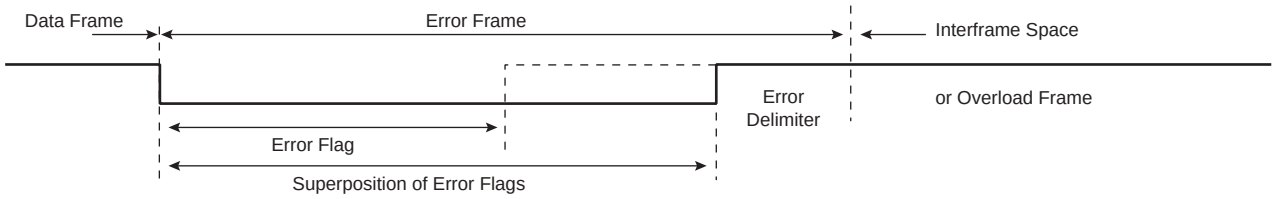
To allow a channel to receive a remote frame, the RTR bit in CAN_IRx register has to be recessive (i.e., at logical 1), or masked (MRTR bit of CAN_MSKx register set to a logical 1).

Error Frame

The error frame consists of two different fields. The first field is given by the superposition of error flags contributed from different stations. The second field is the error delimiter.

In order to terminate an error frame correctly, an error passive node may need the bus to be bus idle for at least three bit times if there is a local error at an error passive receiver. Therefore, the bus should not be loaded to 100%.

Figure 82. Error Frame



Error Flag

There are two forms of error flag: an active error flag and a passive error flag.

- The active error flag consists of six consecutive dominant bits.
- The passive error flag consists of six consecutive recessive bits unless it is overwritten by dominant bits from other nodes.

An error active node detecting an error condition signals this by transmission of an active error flag. The error flag's form violates the law of bit stuffing (see “Coding” on page 262) applied to all fields from start of frame to CRC delimiter or destroys the fixed form ACK field or end of frame field. As a consequence, all other nodes detect an error condition and start transmission of an error flag. Thus the sequence of dominant bits that can actually be monitored on the bus results from a superposition of different error flags transmitted by individual nodes. The total length of this sequence varies between a minimum of six and a maximum of twelve bits.

An error passive node detecting an error condition tries to signal this by transmission of a passive error flag. The error passive node waits for six consecutive bits of equal polarity, beginning at the start of the passive error flag. The passive error flag is complete when these six equal bits have been detected.

Error Delimiter

The error delimiter consists of eight recessive bits.

After transmission of an error flag, each node sends recessive bits and monitors the bus until it detects a recessive bit. Afterwards, it starts transmitting seven more recessive bits.

Overload Frame

The overload frame (this emission can be enabled/disabled in CAN_CR register) contains the two bit fields, overload flag and overload delimiter. There are two kinds of overload conditions that lead to the transmission of an overload flag:

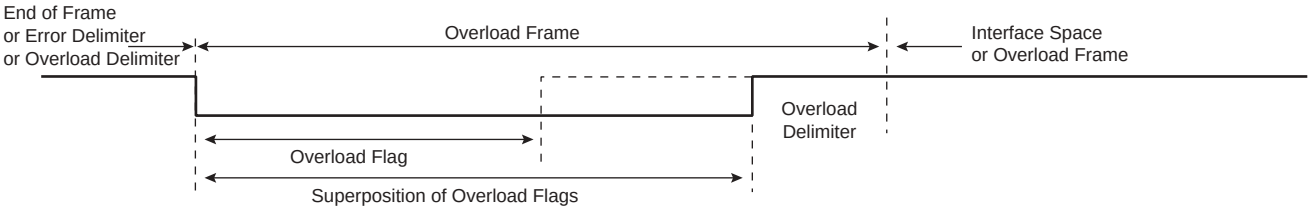
1. The internal conditions of a receiver, requiring a delay of the next data frame or remote frame.
2. Detection of a dominant bit at the first and second bit of intermission.

If a CAN node samples a dominant bit at the eighth bit (the last one) of an error delimiter or overload delimiter, it starts transmitting an overload frame (not an error frame). The error counters are incremented.

The start of an overload frame due to the first overload condition is only allowed to be started at the first bit time of an expected intermission, whereas overload frames due to the second overload condition and condition 3 start one bit after detecting the dominant bit.

At most two overload frames may be generated to delay the next data or remote frame.

Figure 83. Overload Frame



Overload Flag

The overload flag consists of six dominant bits. The overall form corresponds to that of the active error flag.

The overload flag's form destroys the fixed form of the intermission field. As a consequence, all other nodes also detect an overload condition and start transmission of an overload flag. If there is a dominant bit detected during the 3rd bit of intermission, then it interpret this bit as start of frame.

Note: Controllers based on the CAN Specification version 1.0 and 1.1 have another interpretation of the 3rd bit of intermission: if a dominant bit was detected locally at some node, the other nodes do not interpret the overload flag correctly, but interpret the first of these six dominant bits as start of frame; the sixth dominant bit violates the rule of bit stuffing causing an error condition.

Overload Delimiter

The overload delimiter consists of eight recessive bits.

The overload delimiter is of the same form as the error delimiter. After transmission of an overload flag, the node monitors the bus until it detects a transition from a dominant to a recessive bit. At this time, every bus node has finished sending its overload flag and all nodes start transmission of seven more recessive bits simultaneously.

Interframe Spacing

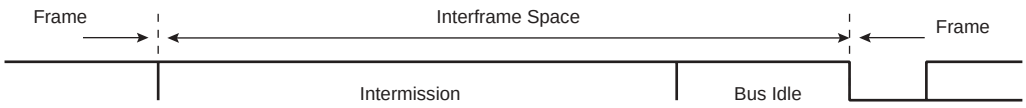
Data frames and remote frames are separated from preceding frames whatever type they are (data frame, remote frame, error frame, overload frame) by a bit field called interframe space. In contrast, overload frames and error frames are not preceded by an interframe space and multiple overload frames are not separated by an interframe space.

Interframe Space

The interframe space contains the bit field intermission and bus idle and, for error passive nodes that have been transmitter of the previous message, suspend transmission.

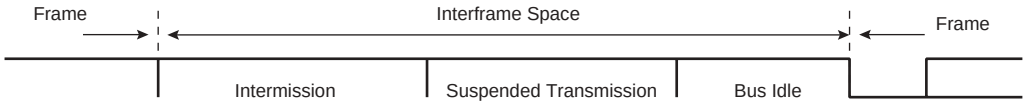
For nodes that are not error passive or have been receiver of the previous message, see Figure 84.

Figure 84. Interframe Space for Receiver



For error passive nodes which have been transmitter of the previous message, see Figure 85.

Figure 85. Interframe Space for Transmitter



Intermission

Intermission consists of three recessive bits.

During intermission, the only action to be taken is signalling an overload condition. No node is allowed to actively start a transmission of a data frame or remote frame.

Note: If a CAN node has a message waiting for transmission and it samples a dominant bit at the third bit of intermission, it will interpret this as a start of frame bit, and, with the next bit, start transmitting its message with the first bit of its identifier without first transmitting a start of frame bit and without becoming receiver.

Bus Idle

The period of bus idle may be of arbitrary length. The bus is recognized to be free and any node having something to transmit can access the bus. A message, which is pending for transmission during the transmission of another message, is started in the first bit following intermission.

The detection of a dominant bit on the bus is interpreted as start of frame.

Suspend Transmission

After an error passive node has transmitted a message, it sends eight recessive bits following intermission, before starting to transmit a further message or recognizing the bus to be idle. If a transmission (caused by another node) starts in the meantime, the node becomes receiver of this message.

Conformance with Frame Formats

The Standard Format is equivalent to the Data/Remote Frame Format as described in the CAN Specification 1.2. In contrast, the Extended Format is a new feature of the CAN protocol. In order to allow the design of relatively simple controllers, the implementation of the Extended Format to its full extent is not required (e.g., send messages or accept data from messages in Extended Format), whereas the Standard Format must be supported without restriction.

New controllers are considered to be in conformance with this CAN Specification if they have at least the following properties with respect to the frame formats defined in “Definition of Transmitter/Receiver” on page 253 and “Frame Types” on page 253.

- Every new controller supports the standard format.
- Every new controller can receive messages of the extended format. This requires that extended frames are not destroyed just because of their format. However, it is not required that the extended format must be supported by new controllers.

Message Filtering

Message filtering is based upon the whole identifier. Mask registers that allow any identifier bit to be set “don't care” for message filtering may be used to select groups of identifiers to be mapped into the attached received buffers.

If mask registers are implemented, every bit of the mask registers must be programmable, i.e., they can be enabled or disabled for message filtering. The length of the mask register can comprise the whole identifier or only part of it.

Message Validation

The point in time at which a message is taken to be valid is different for the transmitter and the receiver of the message.

- Transmitter

The message is valid for the transmitter if there is no error until the end of end of frame. If a message is corrupted, retransmission will follow automatically and according to prioritization. In order to be able to compete for bus access with other messages, retransmission has to start as soon as the bus is idle.

- Receivers

The message is valid for the receivers if there is no error until the first-to-last bit of end of frame. The value of the last bit of EOF is treated as “don't care”, a dominant value does not lead to a FORM error (see “Error Detection” on page 262).

Coding

In bit stream coding, the frame segments start of frame, arbitration field, control field, data field and CRC sequence are coded by bit stuffing. Whenever a transmitter detects five consecutive bits of identical value in the bit stream to be transmitted, it automatically inserts a complementary bit in the currently transmitted bit stream.

The remaining bit fields of the data frame or remote frame (CRC delimiter, ACK field and end of frame) are of fixed form and not stuffed. The error frame and the overload frame are of fixed form as well and not coded via bit stuffing.

The bit stream in a message is coded according to the Non-Return-to-Zero (NRZ) method. This means that during the total bit time the generated bit level is either dominant or recessive.

Error Handling

Error Detection

There are 5 different error types (not mutually exclusive):

- Bit error (BUS bit in CAN_SRX): A unit that is sending a bit on the bus also monitors the bus. A bit error has to be detected when the bit value that is monitored is different from the bit value that is sent. An exception is the sending of a recessive bit during the stuffed bit stream of the arbitration field or during the ACK slot. In this case, no bit error occurs when a dominant bit is monitored. A transmitter sending a passive error flag and detecting a dominant bit does not interpret this as a bit error.
- Stuff error (STUFF bit in CAN_SRX): A stuff error is detected at the bit time of the 6th consecutive equal bit level in a message field that should be coded by the method of bit stuffing.
- CRC error (CRC bit in CAN_SRX): The CRC sequence consists of the result of the CRC calculation by the transmitter. The receivers calculate the CRC in the same way as the transmitter. A CRC error has to be detected if the calculated result is not the same as that received in the CRC sequence.
- Form error (FRAME bit in CAN_SRX): A form error has to be detected when a fixed-form bit field contains one or more illegal bits. (Note that for a receiver a dominant bit during the last bit of EOF is not treated as a form error).
- Acknowledgement error (ACK bit in CAN_SRX): An acknowledgement error is detected by a transmitter whenever it does not monitor a dominant bit during ACK slot.

Error Signalling

A node detecting an error condition signals this by transmitting an error flag. For an error active node, it is an active error flag; for an error passive node, it is a passive error flag.

Whenever a bit error, a stuff error, a form error or an acknowledgement error is detected by any node, transmission of an error flag is started at the respective node at the next bit.

Whenever a CRC error is detected, transmission of an error flag starts at the bit following the ACK delimiter, unless an error flag for another error condition has already been started.

Fault Confinement

Regarding fault confinement, a unit may be in one of three states:

- error active
- error passive
- bus off

An error active unit can normally take part in bus communication and sends an active error flag when an error has been detected.

An error passive unit must not send an active error flag. It takes part in bus communication, but when an error has been detected, only a passive error flag is sent. After a transmission, an error passive unit will wait before initiating a further transmission (see “Suspend Transmission” on page 261).

A bus off unit is not allowed to have any influence on the bus. (e.g., output drivers switched off).

For fault confinement, two counts are implemented in every bus unit:

- transmit error count
- receive error count

These counts are modified according to the following rules (note that more than one rule may apply during a given message transfer):

- When a receiver detects an error, the receive error count will be increased by 1, except when the detected error was a bit error during the sending of an active error flag or an overload flag.
- When a receiver detects a dominant bit as the first bit after sending an error flag, the receive error count will be increased by 8.
- When a transmitter sends an error flag, the transmit error count is increased by 8 except:
  - if the transmitter is error passive and detects an acknowledgement error because of not detecting a dominant ACK. It does not detect a dominant bit while sending its passive error flag.
  - if the transmitter sends an error flag because a stuff error occurred during arbitration, and should never been recessive, and has been sent as recessive but monitored as dominant

In case of one of these exceptions, the transmit error count is not changed.

- If a transmitter detects a bit error while sending an active error flag or an overload flag, the transmit error count is increased by 8.
- If a receiver detects a bit error while sending an active error flag or an overload flag, the receive error count is increased by 8.
- Any node tolerates up to 7 consecutive dominant bits after sending an active error flag, passive error flag or overload flag. After detecting the 14th consecutive dominant bit (in case of an active error flag or an overload flag) or after detecting the 8th consecutive dominant bit following a passive error flag, and after each sequence

of additional eight consecutive dominant bits, every transmitter increases its transmit error count by 8 and every receiver increases its receive error count by 8.

- After the successful transmission of a message (getting ACK and no error until end of frame is finished), the transmit error count is decreased by 1 unless it was already 0.
- After the successful reception of a message (reception without error up to the ACK slot and the successful sending of the ACK bit), the receive error count is decreased by 1, if it was between 1 and 127. If the receive error count was 0, it stays 0, and if it was greater than 127, then it will be set to a value between 119 and 127.
- A node is error passive when the transmit error count equals or exceeds 128, or when the receive error count equals or exceeds 128. An error condition letting a node become error passive causes the node to send an active error flag.
- A node is bus off when the transmit error count is greater than or equal to 256.
- An error passive node becomes error active again when both the transmit error count and the receive error count are less than or equal to 127.
- An node that is bus off is permitted to become error active (no longer bus off) with its error counters both set to 0 after 128 occurrences of 11 consecutive recessive bits have been monitored on the bus.

Note: An error count value greater than 96 indicates a heavily disturbed bus. It may be useful to provide means to test for this condition.

Note: Start-up/Wake-up: If during system start-up only one node is online, and if this node transmits a message, it gets no acknowledgement, detects an error and repeats the message. It can become error passive but not bus off due to this reason.

Oscillator Tolerance

A maximum oscillator tolerance of 1.58% is given and therefore a ceramic resonator can be used at a bus speed of up to 125 Kbits/s as a rule of thumb. For a more precise evaluation, refer to "Impact of Bit Representation on Transport Capacity and Clock Accuracy in Serial Data Streams" by S. Dais and M. Chapman, SAE Technical Paper Series 890532, Multiplexing in Automobile SP-773, March 1989.

For the full bus speed range of the CAN protocol, a quartz oscillator is required.

The chip of the CAN network with the highest requirement for its oscillator accuracy determines the oscillator accuracy which is required from all the other nodes.

Note: CAN controllers compliant with this CAN Specification and controllers compliant with the previous versions 1.0 and 1.1, used in one and the same network, must all be equipped with a quartz oscillator. Thus ceramic resonators can only be used in a network with all the nodes of the network according to the CAN Protocol Specification versions 1.2 or later.

Bit Timing Requirements

Nominal Bit Rate

The nominal bit rate is the number of bits per second transmitted in the absence of resynchronization by an ideal transmitter.

Nominal Bit Time

The nominal bit time of the network is uniform throughout the network and is given by:

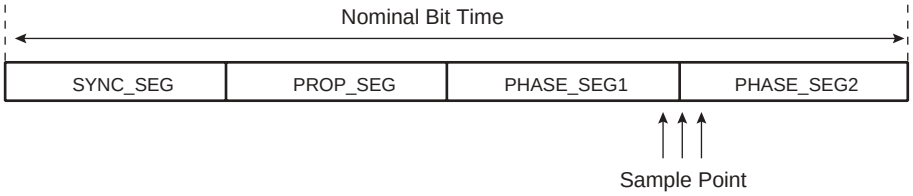
$$\text{Nominal Bit Time} = 1 / \text{Nominal Bit Rate}$$

The nominal bit time can be thought of as being divided into separate non-overlapping time segments. These segments are:

- synchronization segment (SYNC_SEG)
- propagation time segment (PROP_SEG)

- phase buffer segment 1 (PHASE_SEG1)
- phase buffer segment 2 (PHASE_SEG2)

Figure 86. Partition of the Bit Time



The duration of each segment is set in the mode register and is expressed as a multiple of time quantum (t_{CAN}).

Time Quantum

Each segment (SYNC_SEG, PROP_SEG, PHASE_SEG1 and PHASE_SEG2) is an integer multiple of a unit of time called a Time Quantum (t_{CAN}). The duration of a Time Quantum is equal to the period of the CAN system clock (t_{CAN}), which is derived from the microcontroller system clock (CORECLK) by way of a programmable prescalar, called the Baud Rate Prescalar (in the CAN_MR register).

The formula relating t_{CAN}, CORECLK and BD[5:0] is the following:

$$t_{CAN} = \frac{(BD[5:0] + 1)}{CORECLK}$$

Synchronization Segment

This part of the bit time is used to synchronize the various nodes on the bus. An edge is expected to lie within this segment.

The duration of the synchronization segment, SYNC_SEG, is not programmable and is fixed at one time quantum.

Propagation Segment

This part of the bit time is used to compensate for the physical delay times within the network. It is twice the sum of the signal's propagation time on the bus line, the input comparator delay, and the output driver delay.

The duration of the propagation segment, PROP_SEG, may be between 1 and 8 time quanta. It is programmable using the PROP bits in the CAN_MR, and is equal to:

$$t_{PRS} = t_{CAN} \times (PROP[2:0] + 1)$$

PHASE_SEG1, PHASE_SEG2

The phase buffer segments are used to compensate for edge phase errors. The segments can be lengthened or shortened by resynchronization.

The duration of the segment PHASE_SEG1 may be between 1 and 8 time quanta.

The duration of segment PHASE_SEG2 is the maximum of PHASE_SEG1 and the information processing time (See “Information Processing Time (IPT)” on page 266).

PHASE_SEG1 and PHASE_SEG2 are programmable using the PHSEG1[2:0] and PHSEG1[2:0] bits in the MR, respectively. They respect the following equations:

$$t_{PHS1} = t_{CAN} \times (PHSEG1[2:0] + 1)$$

$$t_{PHS2} = t_{CAN} \times (PHSEG2[2:0] + 1)$$

Sample Point

The sample point is the point of time at which the bus level is read and interpreted as the value of that respective bit. Its location is at the end of PHASE_SEG1. Two methods can be used: the incoming stream is sampled once at a sample point, or sampled 3 times with a period of half a CAN clock period, centered at one sample point.

Information Processing Time (IPT)

The information processing time is the time segment starting with the sample point reserved for calculation of the subsequent bit level.

In other words, the time after the sample point that is needed to calculate the next bit to be sent (e.g., data bit, CRC bit, stuff bit, error flag, or idle) is called the Information Processing Time (IPT).

The CAN module has zero delay IPT.

Length of Time Segments

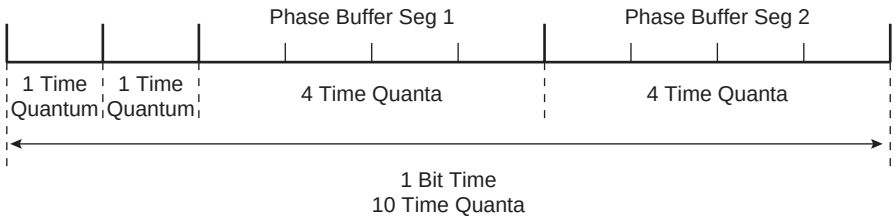
- SYNC_SEG is 1 time quantum long.
- PROP_SEG is programmable to be between 1 and 8 time quanta long.
- PHASE_SEG1 is programmable to be between 1 and 8 time quanta long.
- PHASE_SEG2 is the maximum of PHASE_SEG1 and the information processing time.
- The information processing time is 0 time quanta long.

The total number of time quanta in a bit time is programmable at least from 8 to 25.

Note: It is often intended that control units do not make use of different oscillators for the local CPU and its communication device. Therefore, the oscillator frequency of a CAN device tends to be that of the local CPU and is determined by the requirements of the control unit. In order to derive the desired bit rate, programmability of the bit timing is necessary. In case of CAN implementations that are designed for use without a local CPU, the bit timing cannot be programmable. On the other hand, these devices allow selection of an external oscillator in such a way that the device is adjusted to the appropriate bit rate so that the programmability is dispensable for such components.

The position of the sample point, however, should be selected in common for all nodes. Therefore, the bit timing of CAN devices without local CPU should be compatible with the definition of the bit time in Figure 87.

Figure 87. Example of Nominal Bit Time



Hard Synchronization

After a hard synchronization, the internal bit time is restarted with SYNC_SEG. Thus hard synchronization forces the edge that caused the hard synchronization to lie within the synchronization segment of the restarted bit time.

Resynchronization Jump Width

As a result of resynchronization, PHASE_SEG1 may be lengthened or PHASE_SEG2 may be shortened. The amount of lengthening or shortening of the phase buffer segments has an upper bound given by the resynchronization jump width. The resynchronization jump width is programmable between 1 and min(4, PHASE_SEG1).

Clocking information may be derived from transitions from one bit value to the other. The property that only a fixed maximum number of successive bits have the same value provides the possibility of resynchronizing a bus unit to the bit stream during a frame. The maximum length between two transitions that can be used for resynchronization is 29 bit times.

Phase Error of an Edge

The phase error of an edge is given by the position of the edge relative to SYNC_SEG, measured in time quanta. The sign of phase error is defined as follows:

- e = 0 if the edge lies within SYNC_SEG.
- e > 0 if the edge lies before the SAMPLE POINT.
- e < 0 if the edge lies after the SAMPLE POINT of the previous bit.

Resynchronization

The effect of a resynchronization is the same as that of a hard synchronization when the magnitude of the phase error of the edge causing the resynchronization is less than or equal to the programmed value of the resynchronization jump width. When the magnitude of the phase error is larger than the resynchronization jump width,

- and if the phase error is positive, then PHASE_SEG1 is lengthened by an amount equal to the resynchronization jump width.
- and if the phase error is negative, then PHASE_SEG2 is shortened by an amount equal to the resynchronization jump width.

Synchronization Rules

Hard synchronization and resynchronization are the two forms of synchronization. They obey the following rules:

1. Only one synchronization within one bit time is allowed.
2. An edge is used for synchronization only if the value detected at the previous sample point (previous read bus value) differs from the bus value immediately after the edge.
3. Hard synchronization is performed whenever there is a recessive-to-dominant edge during bus idle.
4. All other recessive-to-dominant edges fulfilling rules 1 and 2 are used for resynchronization with the exception that a node transmitting a dominant bit does not perform a resynchronization as a result of a recessive-to-dominant edge with a positive phase error if only recessive-to-dominant edges are used for resynchronization.

Reception Mode

In reception, channels can be configured in two different modes:

- Normal mode (OVERWRITE bit reset to 0 in CAN_CRX): the channel is disabled after a successful reception.
- Overwrite mode (OVERWRITE bit set to 1 in CAN_CRX): the channel is still enabled after a successful reception.

Time Stamp

A 32-bit stamp dates all messages sent/received (depending on the producer/consumer bit).

The 32-bit register forming the second counter in the WT module is provided to the CAN module. After each transmission or reception of a CAN frame, the value of the current second counter is automatically written in the corresponding CAN channel CAN_STPx register.

Power Management

The CAN is provided with a power management block allowing optimization of power consumption (see “Power Management Block” on page 22).

Example of Use

This section gives an example of use of the CAN, explaining how to transmit a data frame of 5 bytes with an extended identifier on channel 0, and how to receive a data frame of 5 bytes with an extended identifier on channel 1.

The interrupt is configured in transmission and in reception. Core clock is 30 MHz.

Configuration

- Enable the clock on CAN peripheral by writing bit CAN in CAN_ECR.
- Do a software reset of the CAN peripheral to be in a known state by writing bit SWRST in CAN_CR. Software should wait 1 cycle*nbchannel*8 for channel register initialization. Users can also use the associated interrupt flag ENDINIT.
- Configuration of CAN_MR: Choose a Time quantum to be t_{CAN} = 2/CORECLOCK, a propagation segment value to be $t_{PRS} = t_{CAN} * 8$, a synchronization jump width to be $t_{SWJ} = t_{CAN}*3$, and the phase segments 1 and 2 to be $t_{PHS1} = t_{PHS2} = t_{CAN}*3$.
- Enable the CAN by writing bit CANEN in CAN_CR. An interrupt can be associated to know when the CAN is really enabled.
- Configuration of CAN_IERX: In order to generate an interrupt at the end of the transmission, set the bit TXOK in CAN_IER0 and in reception RXOK in CAN_IER1; set the channel source 0 and 1 in CAN_SIER. GIC must be configured.
- Fill the extended identifier to send in CAN_IR0.
- Fill the extended identifier to receive in CAN_IR1 and the mask on the identifier CAN_MSK1 if users want to receive other identifiers.
- Fill in the four first data in CAN_DRA0 and the fifth in CAN_DRB0 to send.
- Set the length of byte (5) to receive field DLC, bit IDE for the extended identifier, and CHANEN to enable this channel in CAN_CR1. An interrupt is generated if this CAN receives a frame with the identifier programmed.
- Set the length and then the transmit command by setting bit PCB, bit IDE, DLC filled with 5, and CHANEN to start the transmission in CAN_CR0. An interrupt is generated after this transmission.

Interrupt Handling

- IRQ Entry and call C function.
- Read CAN_SR and verify the source of the interrupt. If bit ISS is set, a channel generates an interrupt.
- Interrupt handling: If ISS is set in CAN_SR, then we should look at the corresponding CAN_SRx and treat the corresponding interrupt generated by the channel, then clear the interrupt flag by writing to the CAN_CSRx.
- IRQ Exit

Controller Area Network (CAN) Memory Map

Base Address CAN0: 0xFFFD4000 (16 Channels)

Base Address CAN1: 0xFFFB8000 (16 Channels)

Base Address CAN2: 0xFFFBC000 (32 Channels)

Base Address CAN3: 0xFFFB0000 (16 Channels)

Table 55. CAN Memory Map

Offset	Register	Name	Access	Reset State
0x000 - 0x04C	Reserved	–	–	–
0x050	Enable Clock Register	CAN_ECR	Write -only	–
0x054	Disable Clock Register	CAN_DCR	Write-only	–
0x058	Power Management Status Register	CAN_PMSR	Read-only	0x00000000
0x05C	Reserved	–	–	–
0x060	Control Register	CAN_CR	Write-only	–
0x064	Mode Register	CAN_MR	Read/Write	0x00000000
0x068	Reserved	–	–	–
0x06C	Clear Status Register	CAN_CSR	Write-only	–
0x070	Status Register	CAN_SR	Read-only	0x00000002
0x074	Interrupt Enable Register	CAN_IER	Write-only	–
0x078	Interrupt Disable Register	CAN_IDR	Write-only	–
0x07C	Interrupt Mask Register	CAN_IMR	Read-only	0x00000000
0x080	Clear Interrupt Source Status Register	CAN_CISR	Write-only	–
0x084	Interrupt Source Status Register	CAN_ISSR	Read-only	0x00000000
0x088	Source Interrupt Enable Register	CAN_SIER	Write-only	–
0x08C	Source Interrupt Disable Register	CAN_SIDR	Write-only	–
0x090	Source Interrupt Mask Register	CAN_SIMR	Read-only	0x00000000
0x094 - 0x0FC	Reserved	–	–	–
0x100	Channel 0 Data Register A	CAN_DRA0	Read/Write	0x00000000
0x104	Channel 0 Data Register B	CAN_DRB0	Read/Write	0x00000000
0x108	Channel 0 Mask Register	CAN_MSK0	Read/Write	0x00000000
0x10C	Channel 0 Identifier Register	CAN_IR0	Read/Write	0x00000000
0x110	Channel 0 Control Register	CAN_CR0	Read/Write	0x00000000
0x114	Channel 0 Stamp Register	CAN_STP0	Read-only	0x00000000
0x118	Channel 0 Clear Status Register	CAN_CSR0	Write-only	–
0x11C	Channel 0 Status Register	CAN_SR0	Read-only	0x00000000
0x120	Channel 0 Interrupt Enable Register	CAN_IER0	Write-only	–
0x124	Channel 0 Interrupt Disable Register	CAN_IDR0	Write-only	–
0x128	Channel 0 Interrupt Mask Register	CAN_IMR0	Read-only	0x00000000

Table 55. CAN Memory Map (Continued)

Offset	Register	Name	Access	Reset State
0x12C - 0x13C	Reserved	–	–	–
0x140	Channel 1 Data Register A	CAN_DRA1	Read/Write	0x00000000
0x144	Channel 1 Data Register B	CAN_DRB1	Read/Write	0x00000000
0x148	Channel 1 Mask Register	CAN_MSK1	Read/Write	0x00000000
–	–	–	–	–
0x8E0	Channel 31 Interrupt Enable Register	CAN_IER31	Read/Write	0x00000000
0x8E4	Channel 31 Interrupt Disable Register	CAN_IDR31	Read/Write	0x00000000
0x8E8	Channel 31 Interrupt Mask Register	CAN_IMR31	Read/Write	0x00000000

CAN Enable Clock Register

Name: CAN_ECR
Access: Write-only
Base Address: 0x050

CAN Disable Clock Register

Name: CAN_DCR
Access: Write-only
Base Address: 0x054

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
–	–	–	–	–	–	CAN	–

- **CAN: CAN Clock Status**

0: CAN clock disabled.

1: CAN clock enabled.

Note: The CAN_PMSR register is not reset by software reset.

CAN Power Management Status Register

Name: CAN_PMSR
Access: Read-only
Base Address: 0x058

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
–	–	–	–	–	–	CAN	–

- **CAN: CAN Clock Status**

0: CAN clock disabled.

1: CAN clock enabled.

Note: The CAN_PMSR register is not reset by software reset.

CAN Control Register

Name: CAN_CR
Access: Write-only
Base Address: 0x060

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
–	OVDIS	OVEN	ABDIS	ABEN	CANDIS	CANEN	SWRST

- **SWRST: CAN Software Reset**

0: No effect.

1: Resets the CAN.

A software reset triggered hardware reset of the CAN is performed. It resets all the registers (except the CAN_PMSR).

- **CANEN: CAN Enable**

0: No effect.

1: Enables the CAN.

This enables the CAN to transfer and receive data.

A CAN bus synchronization delay is observed between the moment the CAN is enabled and the moment the CAN becomes an active node of the bus, ready to receive or transmit CAN frames. This synchronization delay is specified as part of the CAN standard and is equal to eleven nominal bit times. An additional internal delay between 0 and 1 bit times could be observed due to the re-synchronization of the enable command on the CAN bit-time clock.

- **CANDIS: CAN Disable**

0: No effect.

1: Disables the CAN.

No data is received or transmitted.

In case a transfer is in progress, the transfer is finished before the CAN is disabled.

In case the CPU disables CAN while a transmit channel is enabled and ready to transmit, a spontaneous frame could be generated from that channel when re-enabling the CAN. To avoid spontaneous frame, CPU should disable transmit channels, and then wait for 'channel scan' before disabling the CAN module. The channel scan delay equals 3 x 32 = 192 system clock periods for CAN2, and 3 x 16 = 96 system clock periods for CAN0, CAN1 and CAN3. Waiting for the channel scan ensures that no transmission is memorized by the CAN state machine. Another solution consists in resetting the CAN module (bit SWRST in CAN_CR register) before re-enabling CAN.

In case both CANEN and CANDIS are equal to one when the control register is written, the CAN will be disabled.

- **ABEN: Abort Request Activate**

0: No effect.

1: Activates the CAN abort request.

When set to 1, it masks CHANEN bits of the control register channels. In this mode, the CAN module still acknowledges messages on the bus, and its error counters (TEC and REC) continue to count.

- **ABDIS: Abort Request Deactivate**

0: No effect.

1: Deactivates the CAN abort request.

In case both ABEN and ABDIS are equal to one when the control register is written, the CAN abort request is deactivated.

- **OVEN: Overload Request Activate**

0: No effect.

1: Activates the CAN overload request.

- **OVDIS: Overload Request Deactivate**

0: No effect.

1: Deactivates the CAN overload request.

In case both OVEN and OVDIS are equal to one when the control register is written, the CAN overload request is deactivated.

CAN Mode Register

Name: CAN_MR
Access: Read/Write
Base Address: 0x064

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

23	22	21	20	19	18	17	16
–	PHSEG2			–	PHSEG1		

15	14	13	12	11	10	9	8
–	SMP	SJW		–	PROP		

7	6	5	4	3	2	1	0
–	–	BD					

- **BD[5:0]: Time Quantum Period**

These bits are used to determine the time quantum t_{CAN}:

$$t_{CAN} = \frac{(BD[5:0] + 1)}{CORECLK}$$

- **PROP[2:0]: Propagation Segment Value**

This data reflects the physical delay within the network, including the signal propagation time and the chip internal delay.

$$t_{PRS} = t_{CAN} \times (PROP[2:0] + 1)$$

- **SJW[1:0]: Synchronization Jump Width**

The CAN controller re-synchronizes on each edge of the transmission. The SJW value defines the maximum number of CAN clock cycles a bit period may be shortened or lengthened.

$$t_{SWJ} = t_{CAN} \times (SWJ[1:0] + 1)$$

- **SMP: Sampling Mode**

0: The incoming stream is sampled once at sample point.

1: The incoming stream is sampled 3 times with a period of half a CAN clock period, centered at sample point.

- **PHSEG1[2:0]: Phase Segment 1 Value**

This data is used to compensate edge phase errors. This segment can be shortened or lengthened by SJW.

$$t_{PHS1} = t_{CAN} \times (PHSEG1[2:0] + 1)$$

- **PHSEG2[2:0]: Phase Segment 2 Value**

This data is used to compensate edge phase errors. This segment can be shortened or lengthened by SJW.

$$t_{PHS2} = t_{CAN} \times (PHSEG2[2:0] + 1)$$

CAN Clear Status Register

Name: CAN_CSR
Access: Write-only
Base Address: 0x06C

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
–	–	–	–	–	ENDINIT	–	–

- **ENDINIT: Clear End of CAN Initialization**

0: No effect.

1: Clears end of CAN initialization interrupt.

CAN Status Register

Name: CAN_SR
Access: Read-only
Base Address: 0x070

31	30	29	28	27	26	25	24
TEC							
23	22	21	20	19	18	17	16
REC							
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
ISS	OVRQ	ABRQ	BUSOFF	ERPAS	ENDINIT	CANINIT	CANENA

- **CANENA: CAN Enabled**

0: CAN is disabled.

1: CAN is enabled.

No interrupt is generated on CAN enable or disable.

- **CANINIT: CAN Initialized**

0: CAN is initialized.

1: CAN is in initialization phase.

During initialization phase, channel registers cannot be written. This bit does not generate an interrupt. The reset value is equal to 1, but after a short delay (8 times the number of Channel x CORECLOCK periods) corresponding to the initialization phase, this bit goes to 0.

- **ENDINIT: End of CAN Initialization**

0: No end of CAN initialization.

1: End of CAN initialization phase.

- **ERPAS: Error Passive**

0: No transition in error passive mode.

1: CAN enters in error passive mode.

- **BUSOFF: Bus Off**

0: No transition in bus off mode.

1: CAN enters in bus off mode.

- **ABRQ: CAN Abort Request**

0: No CAN abort requested.

1: All the enabled channels are disabled. If this bit is set during communication, the transmission will end.

No interrupt is generated on CAN abort request activation or deactivation.

- **OVRQ: Overload Frame Request**

0: No overload frame requested.

1: A channel, programmed as a reception channel, follows its data of remote frame by an overload frame.

The purpose of the overload frame is to introduce a delay between received frames.

No interrupt is generated on CAN overload frame request activation or deactivation.

- **ISS: Interrupt Source Status**

0: No interrupt in any channel.

1: At least one interrupt occurred in a channel (read CAN_ISSR for more information).

- **REC[7:0]: Reception Error Counter**

Value of the reception error counter.

- **TEC[7:0]: Transmit Error Counter**

Value of the transmit error counter.

CAN Interrupt Enable Register

Name: CAN_IER
Access: Write-only
Base Address: 0x074

CAN Interrupt Disable Register

Name: CAN_IDR
Access: Write-only
Base Address: 0x078

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
–	–	–	BUSOFF	ERPAS	ENDINIT	–	–

- **ENDINIT: End of CAN Initialization Mask**

0: End of CAN initialization interrupt is disabled.

1: End of CAN initialization interrupt is enabled.

- **ERPAS: Error Passive Mask**

0: Error passive interrupt is disabled.

1: Error passive interrupt is enabled.

- **BUSOFF: Bus Off Mask**

0: Bus off interrupt is disabled.

1: Bus off interrupt is enabled.

CAN Interrupt Mask Register

Name: CAN_IMR
Access: Read-only
Base Address: 0x07C

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
–	–	–	BUSOFF	ERPAS	ENDINIT	–	–

- **ENDINIT: End of CAN Initialization Mask**

0: End of CAN initialization interrupt is disabled.

1: End of CAN initialization interrupt is enabled.

- **ERPAS: Error Passive Mask**

0: Error passive interrupt is disabled.

1: Error passive interrupt is enabled.

- **BUSOFF: Bus Off Mask**

0: Bus off interrupt is disabled.

1: Bus off interrupt is enabled.

CAN Clear Interrupt Source Status Register

Name: CAN_CISR
Access: Write-only
Base Address: 0x080

31	30	29	28	27	26	25	24
CH31	CH30	CH29	CH28	CH27	CH26	CH25	CH24
23	22	21	20	19	18	17	16
CH23	CH22	CH21	CH20	CH19	CH18	CH17	CH16
15	14	13	12	11	10	9	8
CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8
7	6	5	4	3	2	1	0
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

- **CHX: Channel X Interrupt Clear**

0: No effect.

1: Clears interrupt for Channel X.

CAN Interrupt Source Status Register

Name: CAN_ISSR
Access: Write-only
Base Address: 0x084

31	30	29	28	27	26	25	24
CH31	CH30	CH29	CH28	CH27	CH26	CH25	CH24
23	22	21	20	19	18	17	16
CH23	CH22	CH21	CH20	CH19	CH18	CH17	CH16
15	14	13	12	11	10	9	8
CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8
7	6	5	4	3	2	1	0
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

- **CHX: Channel X Interrupt**

0: No interrupt occurred on Channel X.

1: An interrupt occurred on Channel X.

CAN Source Interrupt Enable Register

Name: CAN_SIER
Access: Write-only
Base Address: 0x088

CAN Source Interrupt Disable Register

Name: CAN_SIDR
Access: Write-only
Base Address: 0x08C

31	30	29	28	27	26	25	24
CH31	CH30	CH29	CH28	CH27	CH26	CH25	CH24
23	22	21	20	19	18	17	16
CH23	CH22	CH21	CH20	CH19	CH18	CH17	CH16
15	14	13	12	11	10	9	8
CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8
7	6	5	4	3	2	1	0
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

- **CHX: Channel X Interrupt Mask**

0: Channel X interrupt is disabled.

1: Channel X interrupt is enabled.

CAN Source Interrupt Mask Register

Name: CAN_SIMR
Access: Read-only
Base Address: 0x090

31	30	29	28	27	26	25	24
CH31	CH30	CH29	CH28	CH27	CH26	CH25	CH24
23	22	21	20	19	18	17	16
CH23	CH22	CH21	CH20	CH19	CH18	CH17	CH16
15	14	13	12	11	10	9	8
CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8
7	6	5	4	3	2	1	0
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

- **CHX: Channel X Interrupt Mask**

0: Channel X interrupt is disabled.

1: Channel X interrupt is enabled.

CAN Channel Data Register A

Name: CAN_DRA0...CAN_DRA31
Access: Read/Write
Base Address: 0xXX0

31	30	29	28	27	26	25	24
DATA3							
23	22	21	20	19	18	17	16
DATA2							
15	14	13	12	11	10	9	8
DATA1							
7	6	5	4	3	2	1	0
DATA0							

- **DATAy [y = 3..0]: Data y of Channel X**

Data number y of Channel x.

CAN Channel Data Register B

Name: CAN_DRB0...CAN_DRB31
Access: Read/Write
Base Address: 0xXX4

31	30	29	28	27	26	25	24
DATA7							
23	22	21	20	19	18	17	16
DATA6							
15	14	13	12	11	10	9	8
DATA5							
7	6	5	4	3	2	1	0
DATA4							

- **DATAy [y = 7..4]: Data y of Channel X**

Data number y of Channel x.

CAN Channel Mask Register

Name: CAN_MSK0...CAN_MSK31
Access: Read/Write
Base Address: 0xXX8

31	30	29	28	27	26	25	24
MRTR	MRB		MASK				

23	22	21	20	19	18	17	16
MASK							

15	14	13	12	11	10	9	8
MASK[15:8]							

7	6	5	4	3	2	1	0
MASK[7:0]							

- **MASK[28:0]: Identifier Mask of Channel X**

29-bit mask for identifier.

If CAN operates with 11-bit identifier (standard frame), the upper bits MASK[28:11] are not used.

If CAN operates with 29-bit identifier (extended frame), the lower bits MASK[10:0] are used against the first received identifier bits and the upper bits MASK[28:11] are used with the last received identifier bits.

- **MRB[1:0]: Reserved Mask Bits**

Masks the reserved bits. MRB[1] is only used in extended format.

- **MRTR: Remote Transmission Request Mask**

Masks the RTR bit.

CAN Channel Identifier Register

Name: CAN_IR0...CAN_IR31
Access: Read/Write
Base Address: 0xXXC

31	30	29	28	27	26	25	24
RTR	RB		ID				

23	22	21	20	19	18	17	16
ID							

15	14	13	12	11	10	9	8
ID							

7	6	5	4	3	2	1	0
ID							

- **ID[28:0]: Identifier of Channel X**

29-bit value for identifier.

For channels configured in standard frame mode (IDE bit reset to '0' in CAN_CRx), upper bits ID[28:11] are not used. In case of a transmission, only the base identifier bits ID[10:0] are sent from ID10 to ID0. In case of a reception, for acceptance filtering, only the base identifier ID[10:0] (together with MASK[10:0] bits from CAN_MSKx register) are checked against the received identifier.

For channels configured in extended frame mode (IDE bit set to '1' in CAN_CRx), upper bits ID[28:11] are used for the extended identifier and lower bits ID[10:0] are used for the base identifier. In case of a transmission, the base identifier ID[10:0] are sent first from ID10 to ID0, and extended identifier ID[28:11] are sent later from ID28 to ID11. And in case of a reception, for acceptance filtering, base identifier ID[10:0] (together with MASK[10:0] bits from CAN_MSKx register) are checked against first received identifier bits, and the extended identifier ID[28:11] (together with MASK[28:11] bits from CAN_MSKx register) are checked against last received identifier bits.

Frame Format	Identifier Length	Base ID	Extended ID
Standard frame	11 bits	ID[10:0]	N/A
Extended frame	29 bits	ID[10:0]	ID[28:11]

- **RB[1:0]: Reserved Bits**

These bits are sent to the control field. RB[1] generates the IDE bit in the standard frame format and the r1 bit in the extended frame format, respectively. RB[0] generates the r0 bit in the standard and extended frame formats. In the CAN 2.0A specification, RB[1] and RB[0] must be set dominant (i.e., to logical 0).

- **RTR: Remote Transmission Request**

In data frames, the RTR bit has to be dominant. Within a remote frame, the RTR bit has to be recessive.

CAN Channel Control Register

Name: CAN_CR0...CAN_CR31
Access: Read/Write
Base Address: 0xXX0

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	OVERWRITE

7	6	5	4	3	2	1	0
CHANEN	PCB	RPLYV	IDE	DLC[3:0]			

- **DLC[3:0]: Data Length Code**

This is the number of bytes in the data field of a message (from 0 to 8). This value is updated whenever a frame is received (data or remote). If the incoming DLC differs from the expected one, a warning is issued in the status register of the channel (CAN_SRX).

- **IDE: Extended Identifier Flag**

0: Identifier is 11 bits long (CAN rev 2.0A).

1: Identifier is 29 bits long (CAN rev 2.0B).

- **RPLYV: Automatic Reply**

0: No effect.

1: Channel x makes an automatic reply after receiving a remote frame.

- **PCB: Channel Producer**

0: Channel x is consumer.

1: Channel x is producer.

Bit PCB resets to 0 after a transmission.

- **CHANEN: Channel Enable**

0: Channel x disabled.

1: Channel x enabled.

Note: When a CAN channel configured in transmission is enabled, the frame is transmitted after a short delay. This delay results from the 'channel scan'. The CAN state machine continually scans all channels, looks for transmit channels, and memorizes the channel with the highest priority ID.

Scan delay of CAN channels depends on the channel state: a channel enabled and configured in transmission is scanned in three system clock periods, otherwise it takes two system clock periods. Scanning all the channels takes at maximum:

- For a 16-channel CAN: (1 + 3*16) * system clock periods
- For a 32-channel CAN: (1 + 3*32) * system clock periods.

As the channel scan is asynchronous with the moment the CPU enables the transmit channel, the result is that the delay is unsettled. It takes a maximum of twice the delay to scan all channels. Moreover, as the scan of the channels is also asynchronous with the bit time, between 0 and 1 bit times should be added to this delay.

- **OVERWRITE: Channel Overwrite Mode**

0: Channel x in normal mode.

1: Channel x in overwrite mode.

In overwrite mode, the channel is not disabled after each reception. New frames overwrite the previous frame.

CAN Channel Stamp Register

Name: CAN_STP0...CAN_STP31
Access: Read-only
Base Address: 0xXX4

31	30	29	28	27	26	25	24
STAMP[31:24]							

23	22	21	20	19	18	17	16
STAMP[23:16]							

15	14	13	12	11	10	9	8
STAMP[15:8]							

7	6	5	4	3	2	1	0
STAMP[7:0]							

- **STAMP[31:0]: Stamp Value**

These 32 bits stamp the date at which the message linked with Channel x has been emitted/received (depending on the producer/consumer bit). The value is copied from the WT second counter register.

CAN Channel Clear Status Register

Name: CAN_CSR0...CAN_CSR31
Access: Write-only
Base Address: 0xXX8

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

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

15	14	13	12	11	10	9	8
–	–	–	–	OVRUN	FILLED	DLCW	–

7	6	5	4	3	2	1	0
RFRAME	TXOK	RXOK	BUS	STUFF	CRC	FRAME	ACK

- **ACK: Acknowledge Error Clear**

0: No effect.

1: Clears acknowledge error interrupt.

- **FRAME: Frame Error Clear**

0: No effect.

1: Clears frame error interrupt.

- **CRC: CRC Error Clear**

0: No effect.

1: Clears CRC error interrupt.

- **STUFF: Stuffing Error Clear**

0: No effect.

1: Clears stuffing error interrupt.

- **BUS: Bus Error Clear**

0: No effect.

1: Clear bus error interrupt.

- **RXOK: Reception Completed Clear**

0: No effect.

1: Clears reception completed interrupt.

- **TXOK: Transmission Completed Clear**

0: No effect.

1: Clears transmission completed interrupt.

- **RFRAME: Remote Frame Clear**

0: No effect.

1: Clears remote frame interrupt.

- **DLCW: DLC Warning Clear**

0: No effect.

1: Clears DLC warning.

- **FILLED: Filled Flag Clear**

0: No effect.

1: Clears the FILLED flag.

- **OVRUN: Overrun Flag Clear**

0: No effect.

1: Clears the OVERRUN flag.

CAN Channel Status Register

Name: CAN_SR0...CAN_SR31
Access: Read-only
Base Address: 0xXXC

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

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

15	14	13	12	11	10	9	8
–	–	–	–	OVRUN	FILLED	DLCW	–

7	6	5	4	3	2	1	0
RFRAME	TXOK	RXOK	BUS	STUFF	CRC	FRAME	ACK

- **ACK: Acknowledge Error**

0: No acknowledge error during last transmission.

1: An acknowledge error occurred during the last transmission. There was not a dominant bit during the ACK slot.

- **FRAME: Frame Error**

0: No frame error during last communication.

1: A frame error occurred during last communication. A fixed form bit contained one or more illegal bits.

- **CRC: CRC Error**

0: No CRC error during last reception.

1: A CRC error has been detected during the last reception. Received CRC sequence is not equal to the calculated one.

- **STUFF: Stuffing Error**

0: No stuffing error during the last communication.

1: A stuffing error occurred during the last communication. At least 6 consecutive equal bits have been detected.

- **BUS: Bus Error**

0: No bus error during last transmission.

1: A bus error occurred during last transmission. The transmitter sent a dominant bit but a recessive bit was detected on the network.

- **RXOK: Reception Completed**

0: No new reception completed.

1: A reception was completed without any error.

- **TXOK: Transmission Completed**

0: No new transmission completed.

1: A transmission was completed without any error.

- **RFRAME: Remote Frame**

0: No remote frame received or sent since last clear of RFRAME bit.

1: A remote frame has been received or sent since last clear of RFRAME bit.

- **DLCW: DLC Warning**

0: No DLC warning.

1: DLC warning. Last message accepted with a different DLC than programmed in the CAN channel control register (CAN_CRx).

This bit does not generate an interrupt.

- **FILLED: Reception Buffer Filled**

0: No frame has been received since last clear of FILLED bit.

1: The buffers CAN_DRA and CAN_DRB not read since last frame has been received.

This flag can be enabled only when OVERWRITE mode is selected.

- **OVRUN: Overrun**

0: No frame has been received while FILLED flag is set to logical 1.

1: A frame has been received while FILLED flag is set to logical 1.

This flag can be raised only when OVERWRITE mode is selected.

This bit does not generate an interrupt.

CAN Channel Interrupt Enable Register

Name: CAN_IER0...CAN_IER31
Access: Write-only
Base Address: 0xXX0

CAN Channel Interrupt Disable Register

Name: CAN_IDR0...CAN_IDR31
Access: Write-only
Base Address: 0xXX4

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
RFRAME	TXOK	RXOK	BUS	STUFF	CRC	FRAME	ACK

- **ACK: Acknowledge Error Mask**

0: Acknowledge error interrupt is disabled.

1: Acknowledge error interrupt is enabled.

- **FRAME: Frame Error Mask**

0: Frame error interrupt is disabled.

1: Frame error interrupt is enabled.

- **CRC: CRC Error Mask**

0: CRC error interrupt is disabled.

1: CRC error interrupt is enabled.

- **STUFF: Stuffing Error Mask**

0: Stuffing error interrupt is disabled.

1: Stuffing error interrupt is enabled.

- **BUS: Bus Error Mask**

0: Bus error interrupt is disabled.

1: Bus error interrupt is enabled.

- **RXOK: Reception Completed Mask**

0: Completed reception interrupt is disabled.

1: Completed reception interrupt is enabled.

General-purpose Timer (GPT)

Overview

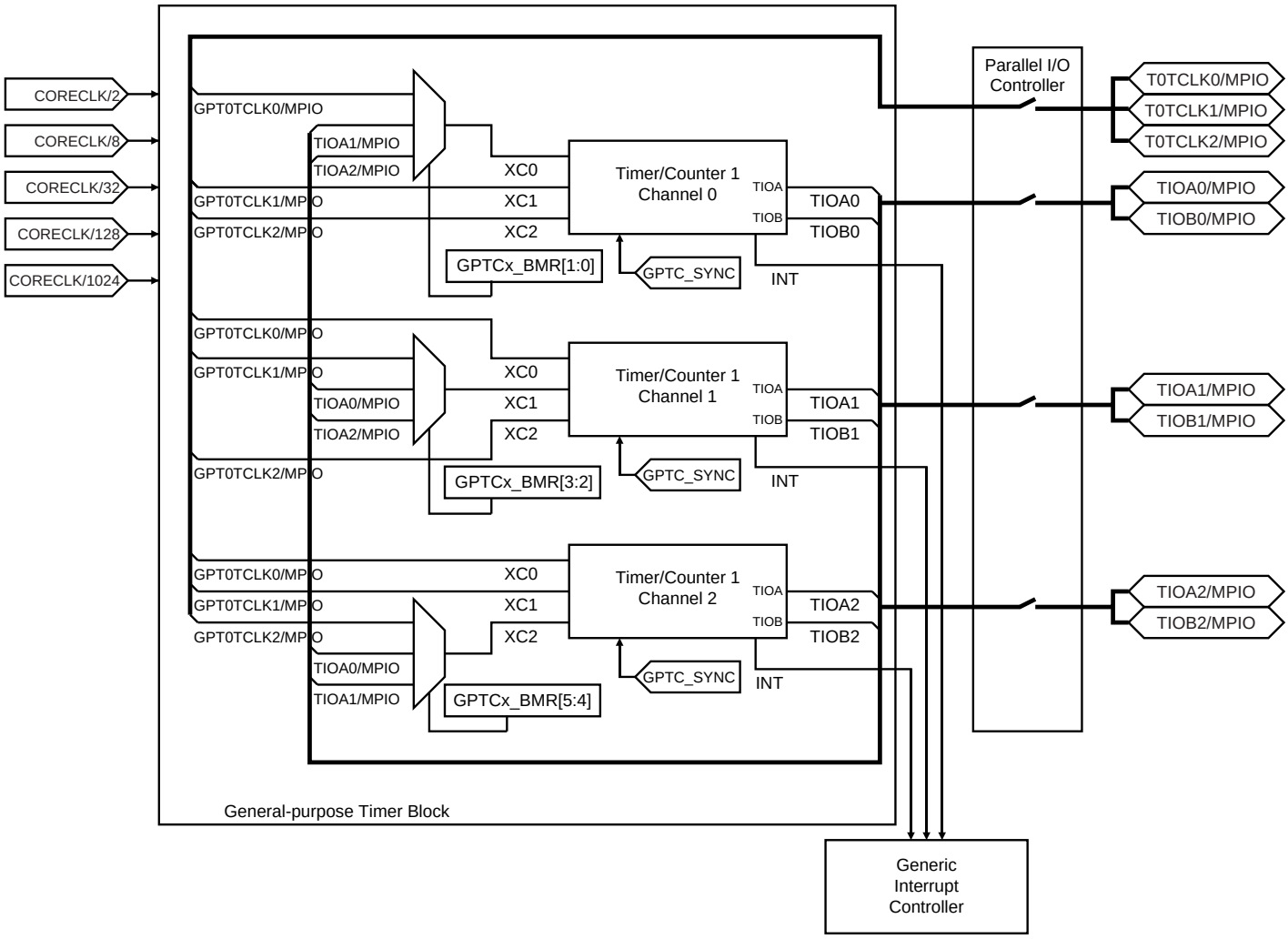
The AT91SAM7A2 has four independent general-purpose timer blocks. Three timers are grouped in the GPT0 module and can be cascaded; the fourth timer is the GPT1 module.

Each timer has a 16-bit Timer/counter channel, a Power Management Controller and a Parallel I/O Controller. Each channel can be independently programmed using the two operating modes (capture mode or waveform mode) to perform a range of functions including frequency measurement, event counting, interval measurement, pulse generation, delay timing, pulse width modulation and interrupt generation.

After a hardware reset, the timer pins are set as general-purpose I/O (configured as input), the interrupts are disabled in the interrupt controller and the Timer Controller Clock and the PIO Controller Clock are disabled.

Block Diagram

Figure 88. General-purpose Timer Three-channel Block Diagram



Each channel, shown in its configuration in Figure 88, has the following components:

- one 16-bit counter
- one 16-bit compare register (RC)
- two 16-bit capture/compare registers, RA and RB
- one multiplexer allowing the selection of eight clocks: five internal and three external (common to all channels). It is possible to combine two clocks to generate a burst clock. Each external clock can be used as external trigger source.
- an internal interrupt signal that can be programmed to generate processor interrupts via the Generic Interrupt Controller (GIC module). They are generated when one of the following events occurs:
  - Counter overflow
  - Load Register (A or B)
  - Equal Compare Register (A or B or C)
  - External edge detection
  - Overrun
- one software trigger
- one software reset that resets the channel and its associated registers (except Power Management registers)
- three parallel I/O pins that can be dedicated for multiple counter functions (Operation mode dependent) or in PIO mode.
  - TIOA, in capture mode, is an event input to load register A or register B or an input trigger. In waveform mode, it is an output to generate a waveform.
  - TIOB, in capture mode, is an input trigger. In waveform mode, it is either an output to generate a waveform (dual waveform mode) or an input trigger (single waveform mode).
  - TCLK, in capture mode and in waveform mode, is an external clock input.
- the synchronization bit that allows the synchronization of the three channels by generating a software trigger simultaneously on the three channels.
- the reset bit of the Block Control Register that resets the three channels simultaneously.

It is possible to chain two or three channels to increase the counter capacity (see “Timer Controller Block Programming” on page 348).

Pin Description

Table 56 shows the internal pin configurations in the different operating modes.

Table 56. Pin Definition

Pin (x = channel)	Capture Mode	Single Waveform Mode	Dual Waveform Mode
TIOAx	Input	Output	Output
	Capture or Trigger	Waveform	Waveform
TIOBx	Input	Input	Output
	Trigger	Trigger	Waveform
TCLKx	Input	Input	Input
	External Clock	External Clock	External Clock

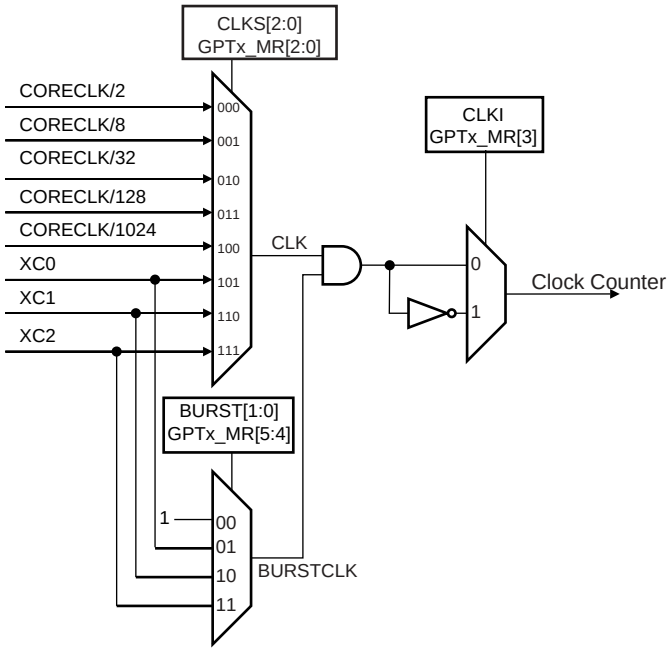
Clock Sources

The timer controller can use one of eight different clocks. The CLKS[2:0] bits of the mode registers GPTx_MR determine whether the counter is clocked by one of the five

internal clock sources generated in the prescalar block and derived from CORECLK or one of the three external clock sources (TCLKx).

Figure 89 shows the clock selection block.

Figure 89. Clock Selection Block



The counter clock sources can be any of the following:

- External event input XCx
- One of 5 internal clocks (CORECLK/2, CORECLK/8, CORECLK/32, CORECLK/128, CORECLK/1024)
- A burst clock (see “Burst Clock” on page 294).

The maximal count duration when an internal clock is used is determined by the internal clock MCK and the prescale number.

Maximal Count Duration (seconds) = 2^{16}/CLK where CLK is in Hz.

Counter Resolution = 1/CLK

External Clock

When an external clock source is used, the 16-bit counter can be programmed as a 16-bit event counter. An external transition (rising or falling following the state of the bit CLKI of the Mode register) increments the counter.

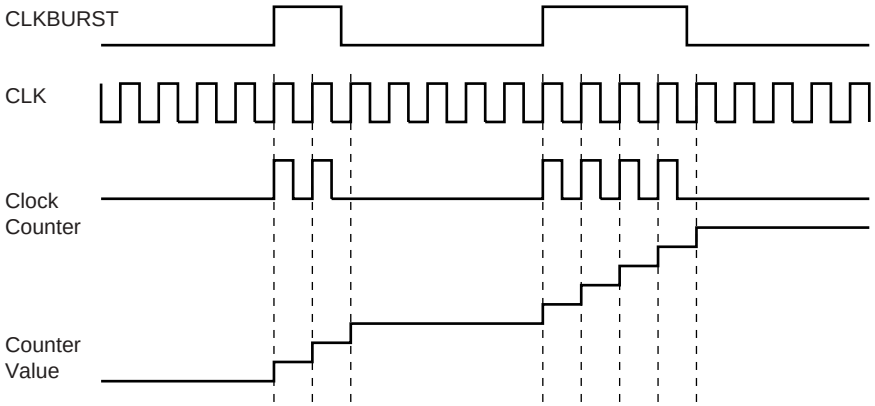
If an external clock is used, make sure that each of its pulse has a duration strictly higher than the CORECLK period.

Burst Clock

If the field BURST of the mode register selects an external clock, this clock is combined with CLK through a logical AND.

Thus the considered timer channel is clocked by CLK only when CLKBURST is high as shown in Figure 90.

Figure 90. Burst Clock



16-bit Counter

The 16-bit counter is a free-running counter clocked by eight sources: 5 internal and 3 external.

The program can access the counter value in real-time in read-only access with the counter value register GPT_CV.

When counter reset occurs, the counter is loaded with 0x0000 and begins its count if its clock is enabled (the clock is disabled or enabled by CLKDIS and CLKEN of the control register GPT_CR).

When the maximal value is reached (0xFFFF), the counter rolls over to a count of 0x0000, sets an overflow flag (the bit COVFS of the status register GPT_SR), may generate an interrupt (enabled by the bit COVFS of the interrupt enable register GPT_IER and disabled by the bit COVFS of the interrupt disable register GPT_IDR), and continues to count up.

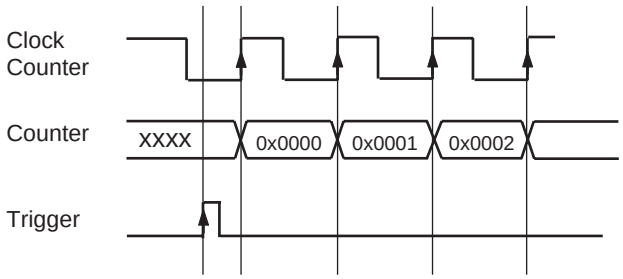
Counter Reset

During counting, the counter can be reset to 0x0000 following:

- a software Trigger
- an external Trigger
- an equality on Compare C
- the synchronous bit TCSYNC of the block control register GPT_BCR.

Each time it is reset, the counter passes to 0x0000 at the next valid counter clock edge. See Figure 91.

Figure 91. Counter Reset Diagram



16-bit Registers

Each channel contains three 16-bit registers.

The mode determines whether the capture/compare registers are used as capture registers or compare registers.

In capture mode, registers A and B are capture registers and can be loaded by TIOAx edges.

In waveform mode, registers A and B are compare registers.

Register C is always a compare register. It can generate a counter reset when the programmed value is reached.

RA, RB and RC compare registers can generate a waveform when their counters reach the programmed values.

In an application, the required compare register values must be calculated using the following equation:

$$\text{CompareValue} = (t \times \text{CLK}) - 1$$

where:

t = desired timer compare period (in seconds)

CLK = counter clock (in Hertz)

Example

To determine the value needed in a compare register to obtain an equality with the counter after 0.1 second with CORECLK = 30 MHz:

1. Determine the minimal prescale value by dividing CORECLK by the maximal counter value 0xFFFF (65,535) to know the divisor factor:

$$\frac{30,000,000}{65,535} = 457.77$$

The value of the divider greater than or equal to 457.77 is DIV_{min} = 1024.

CLK must therefore be at least CORECLK/1024 (29.3 kHz) in order to obtain an equal condition after 0.1 seconds.

2. Calculate the register value with this clock using the following equation:

$$\text{Compare Value} = 0.1 \times \frac{\text{CORECLK}}{1024} - 1 = 0.1 \times \frac{30,000,000}{1024} - 1 = 2928.69$$

Rounding value to 2929 (0x0B71) generates a 0.01% error.

External Edge Detection

The timer contains many external edge detection options.

The operating mode determines their number:

- Capture mode
  - TIOAx as load register and external trigger
  - TIOBx as external trigger
- Waveform mode: Dual waveform mode
  - XC0, XC1 or XC2 as external trigger
- Waveform mode: Single waveform mode
  - TIOBx as external trigger

For each edge detection, it is possible to choose a rising edge, a falling edge or both.

Whereas when a reset is caused, it occurs at the next valid counter clock edge.

If an external trigger is used, each of its pulses must have a duration strictly greater than the CORECLK period.

Interrupts

Each timer contains a total of eight timer interrupts and three PIO interrupts. They can be enabled or disabled from the GPT_IER and GPT_IDR. The programming mode determines which interrupts are available in Table 57.

Table 57. Available Interrupts

Interrupt Name	Capture Mode	Waveform Mode
Counter Overflow Interrupt COVFS	X	X
Load Overrun Interrupt LOVRS	X	
Compare Register A Interrupt CPAS		X
Compare Register B Interrupt CPBS		X
Compare Register C Interrupt CPCS	X	X
Load Capture Register A Interrupt LDRAS	X	
Load Capture Register B Interrupt LDRBS	X	
External Trigger Interrupt ETRGS	X	X
TCLK/MPIO Interrupt TCLKS	X	X
TIOA/MPIO Interrupt TIOAS	X	X
TIOB/MPIO Interrupt TIOBS	X	X

PIO Controller

Each timer channel has three programmable I/O lines. These I/O lines are multiplexed with signals (TIOA, TIOB, TCLK) of the timer channel to optimize the use of available package pins. These lines are controlled by the timer channel PIO controller.

Power Management

Each timer channel (GPT0, GPT1 and GPT2) is provided with a power management block allowing optimization of power consumption (see “Power Management Block” on page 22).

Example Of Use

This section gives an example of use of the General-purpose Timer, explaining how to generate a tick (usually used in RTOS) of 10 ms then in the interrupt. The timer is restarted. Core clock is 30MHz

Configuration

- Enable the clock on GPT peripheral by writing bit TC in GPT_ECR.
- Do a software reset of the GPT peripheral to be in a known state by writing bit SWRST in GPT_CR.
- Configuration of GPT_MR: Choose a clock divider of 1024, Wave mode selected, CPCTRG is selected when counter is equal to RC register, then it causes a trigger on the counter (restart to 0), CPCSTOP allows the counter to stop when equal to RC.
- Configuration of GPT_RC: This sets the period to 10 ms.
  Period = CORECLK/(GPT CLOCK divider*tick frequency) giving 30 MHz/(1024*100).
- Configuration of GPT_IER: To generate an interrupt when the counter is equal to the RC register value, write bit CPCS in GPT_IER. GIC must be configured.

- Start the timer by writing bit CLKEN and SWTRG in GPT_CR. After 10 ms, the interrupt is generated.

Interrupt Handling

- IRQ Entry and call C function.
- Read GPT_SR and verify the source of the interrupt. This register is read and clear, care should be taken to maintain status information to be able to proceed in all cases.
- Interrupt handling: Restart the timer by writing bit CLKEN and SWTRG in GPT_CR. After 10 ms, another interrupt is generated.
- IRQ Exit.

General-purpose Timer (GPT) Memory Map

Base Address GPT0 Channel 0: 0xFFFC8000

Base Address GPT0 Channel 1: 0xFFFC8100

Base Address GPT0 Channel 2: 0xFFFC8200

Base Address GPT1: 0xFFFCC000

Table 58. GPT Memory Map and Control Registers

Offset	Register	Name	Access	Reset State
0x00	PIO Enable Register	GPT_PER	Write-only	–
0x04	PIO Disable Register	GPT_PDR	Write-only	–
0x08	PIO Status Register	GPT_PSR	Read-only	0x00070000
0x0C	Reserved	–	–	–
0x10	PIO Output Enable Register	GPT_OER	Write-only	–
0x14	PIO Output Disable Register	GPT_ODR	Write-only	–
0x18	PIO Output Status Register	GPT_OSR	Read-only	0x00000000
0x1C - 0x2C	Reserved	–	–	–
0x30	PIO Set Output Data Register	GPT_SODR	Write-only	–
0x34	PIO Clear Output Data Register	GPT_CODR	Write-only	–
0x38	PIO Output Data Status Register	GPT_ODSR	Read-only	0x00000000
0x3C	PIO Pin Data Status Register	GPT_PDSR	Read-only	0x000X0000
0x40	PIO Multi-Driver Enable Register	GPT_MDER	Write-only	–
0x44	PIO Multi-Driver Disable Register	GPT_MDDR	Write-only	–
0x48	PIO Multi-Driver Status Register	GPT_MDSR	Read-only	0x00000000
0x4C	Reserved	–	–	–
0x50	Enable Clock Register	GPT_ECR	Write-only	–
0x54	Disable Clock Register	GPT_DCR	Write-only	–
0x58	Power Management Status Register	GPT_PMSR	Read-only	0x00000000
0x5C	Reserved	–	–	–
0x60	Control Register	GPT_CR	Write-only	–
0x64	Mode Register	GPT_MR	Read/Write	0x00000000
0x68	Reserved	–	–	–
0x6C	Reserved	–	–	–
0x70	Status Register	GPT_SR	Read-only	0x00000X00
0x74	Interrupt Enable Register	GPT_IER	Write-only	–
0x78	Interrupt Disable Register	GPT_IDR	Write-only	–
0x7C	Interrupt Mask Register	GPT_IMR	Read-only	0x00000000
0x80	Counter Value	GPT_CV	Read-only	0x00000000
0x84	Capture - Compare Register A	GPT_RA	Read/Write	0x00000000
0x88	Capture - Compare Register B	GPT_RB	Read/Write	0x00000000
0x8C	Compare Register C	GPT_RC	Read/Write	0x00000000
Control and Test Registers				
0x0300	Block Control Register	GPT_BCR	Write-only	–
0x0304	Block Mode Register	GPT_BMR	Read/Write	0x00000000

GPT PIO Enable Register

Name: GPT_PER
Access: Write-only
Base Address: 0x00

GPT PIO Disable Register

Name: GPT_PDR
Access: Write-only
Base Address: 0x04

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

23	22	21	20	19	18	17	16
–	–	–	–	–	TCLK	TIOA	TIOB

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **TIOB: TIOB Pin**

0: PIO is inactive on the TIOBx pin (Timer Controller is active).

1: PIO is active on the TIOBx pin (Timer Controller is inactive).

- **TIOA: TIOA Pin**

0: PIO is inactive on the TIOAx pin (Timer Controller is active).

1: PIO is active on the TIOAx pin (Timer Controller is inactive).

- **TCLK: TCLK Pin**

0: PIO is inactive on the TCLKx pin (Timer Controller is active).

1: PIO is active on the TCLKx pin (Timer Controller is inactive).

Note: x: channel number

GPT PIO Status Register

Name: GPT_PSR
Access: Read-only
Base Address: 0x08

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

23	22	21	20	19	18	17	16
–	–	–	–	–	TCLK	TIOA	TIOB

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **TIOB: TIOB Pin**

0: PIO is inactive on the TIOBx pin (Timer Controller is active).

1: PIO is active on the TIOBx pin (Timer Controller is inactive).

- **TIOA: TIOA Pin**

0: PIO is inactive on the TIOAx pin (Timer Controller is active).

1: PIO is active on the TIOAx pin (Timer Controller is inactive).

- **TCLK: TCLK Pin**

0: PIO is inactive on the TCLKx pin (Timer Controller is active).

1: PIO is active on the TCLKx pin (Timer Controller is inactive).

Note: x: channel number

GPT PIO Output Enable Register

Name: GPT_OER
Access: Write-only
Base Address: 0x10

GPT PIO Output Disable Register

Name: GPT_ODR
Access: Write-only
Base Address: 0x14

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

23	22	21	20	19	18	17	16
–	–	–	–	–	TCLK	TIOA	TIOB

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **TIOB: TIOB Pin**

0: The TIOBx PIO pin is input.

1: The TIOBx PIO pin is output

- **TIOA: TIOA Pin**

0: The TIOAx PIO pin is input.

1: The TIOAx PIO pin is output

- **TCLK: TCLK Pin**

0: The TCLKx PIO pin is input.

1: The TCLKx PIO pin is output.

Note: x: channel number

GPT PIO Output Status Register

Name: GPT_OSR
Access: Read-only
Base Address: 0x18

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

23	22	21	20	19	18	17	16
–	–	–	–	–	TCLK	TIOA	TIOB

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **TIOB: TIOB Pin**

0: The TIOBx PIO pin is input.

1: The TIOBx PIO pin is output

- **TIOA: TIOA Pin**

0: The TIOAx PIO pin is input.

1: The TIOAx PIO pin is output

- **TCLK: TCLK Pin**

0: The TCLKx PIO pin is input.

1: The TCLKx PIO pin is output.

Note: x: channel number

GPT PIO Set Output Data Register

Name: GPT_SODR
Access: Write-only
Base Address: 0x30

GPT PIO Clear Output Data Register

Name: GPT_CODR
Access: Write-only
Base Address: 0x34

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

23	22	21	20	19	18	17	16
–	–	–	–	–	TCLK	TIOA	TIOB

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **TIOB: TIOB Pin**

0: The output data for the TIOBx is programmed to 0.

1: The output data for the TIOBx is programmed to 1.

- **TIOA: TIOA Pin**

0: The output data for the TIOAx is programmed to 0.

1: The output data for the TIOAx is programmed to 1.

- **TCLK: TCLK Pin**

0: The output data for the TCLKx is programmed to 0.

1: The output data for the TCLKx is programmed to 1.

Note: x: channel number

GPT PIO Output Data Status Register

Name: GPT_ODSR
Access: Read-only
Base Address: 0x38

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

23	22	21	20	19	18	17	16
–	–	–	–	–	TCLK	TIOA	TIOB

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **TIOB: TIOB Pin**

0: The output data for the TIOBx is programmed to 0.

1: The output data for the TIOBx is programmed to 1.

- **TIOA: TIOA Pin**

0: The output data for the TIOAx is programmed to 0.

1: The output data for the TIOAx is programmed to 1.

- **TCLK: TCLK Pin**

0: The output data for the TCLKx is programmed to 0.

1: The output data for the TCLKx is programmed to 1.

Note: x: channel number

GPT PIO Pin Data Status Register

Name: GPT_PDSR
Access: Read-only
Base Address: 0x3C

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

23	22	21	20	19	18	17	16
–	–	–	–	–	TCLK	TIOA	TIOB

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **TIOB: TIOB Pin**

0: The pin TIOBx is at logic 0.

1: The pin TIOBx is at logic 1.

- **TIOA: TIOA Pin**

0: The pin TIOAx is at logic 0.

1: The pin TIOAx is at logic 1.

- **TCLK: TCLK Pin**

0: The pin TCLKx is at logic 0.

1: The pin TCLKx is at logic 1.

Note: x: channel number

GPT PIO Multi-driver Enable Register

Name: GPT_MDER
Access: Write-only
Base Address: 0x40

GPT PIO Multi-driver Disable Register

Name: GPT_MDDR
Access: Write-only
Base Address: 0x44

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

23	22	21	20	19	18	17	16
–	–	–	–	–	TCLK	TIOA	TIOB

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **TIOB: TIOB Pin**

0: TIOBx pin is not configured as an open drain.

1: TIOBx pin is configured as an open drain.

- **TIOA: TIOA Pin**

0: TIOAx pin is not configured as an open drain.

1: TIOAx pin is configured as an open drain.

- **TCLK: TCLK Pin**

0: TCLKx pin is not configured as an open drain.

1: TCLKx pin is configured as an open drain.

Note: x: channel number

GPT PIO Multi-driver Status Register

Name: GPT_MDSR
Access: Read-only
Base Address: 0x48

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

23	22	21	20	19	18	17	16
–	–	–	–	–	TCLK	TIOA	TIOB

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **TIOB: TIOB Pin**

0: TIOBx pin is not configured as an open drain.

1: TIOBx pin is configured as an open drain.

- **TIOA: TIOA Pin**

0: TIOAx pin is not configured as an open drain.

1: TIOAx pin is configured as an open drain.

- **TCLK: TCLK Pin**

0: TCLKx pin is not configured as an open drain.

1: TCLKx pin is configured as an open drain.

Note: x: channel number

GPT Enable Clock Register

Name: GPT_ECR
Access: Write-only
Base Address: 0x50

GPT Disable Clock Register

Name: GPT_DCR
Access: Write-only
Base Address: 0x54

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
–	–	–	–	–	–	TC	PIO

- **PIO: PIO Clock**

1: PIO controller clock is enabled.

0: PIO controller clock is disabled.

- **TC: General-purpose Timer Clock**

1: General-purpose Timer channel and clock divider clock enabled.

0: General-purpose Timer channel and clock divider clock disabled.

GPT Power Management Status Register

Name: GPT_PMSR
Access: Read-only
Base Address: 0x58

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
–	–	–	–	–	–	TC	PIO

- **PIO: PIO Clock**

1: PIO controller clock is enabled.

0: PIO controller clock is disabled.

- **TC: General-purpose Timer Clock**

1: General-purpose Timer channel and clock divider clock enabled.

0: General-purpose Timer channel and clock divider clock disabled.

General-purpose Timer in Capture Mode

Description

The capture (wave measurement) mode is entered by setting WAVE (bit [15] in the Mode Register) to 0.

It is the default operating mode after a hardware reset. It forces TIOAx and TIOBx pins as input pins.

The capture mode provides the possibility to determine the duration between two events. An event may either be an external input signal on TIOAx or TIOBx or an internal event (software trigger or equality between the counter and a predefined compare value). An external event (rising or falling edge) on TIOAx can result in capture register A being loaded, capture register B being loaded or a trigger effect (reset and start the counter).

A predefined compare value (16-bit) can be set in the compare register C.

When the capture register B is loaded, it can disable the counter clock and/or stop the counter.

The user may choose an internal clock source (CORECLK/2, CORECLK/8, CORECLK/32, CORECLK/128 or CORECLK/1024) or an external clock (TCLK0, TCLK1 or TCLK2).

A burst mode is available. It generates a burst clock. For more details, refer to “Clock Sources” on page 293.

Six interrupts can be produced:

- External trigger detected
- RA loaded
- RB loaded
- Counter overflow (when the counter passes from 0xFFFF to 0x0000)
- Overrun (when RA or RB is reloaded before the old value is read)
- Compare RC (the counter reaches the value stored in register C)

Finally, the synchronize register can be used to cause a software trigger for reset and start the counter at the next valid counter clock edge on all channels at the same time.

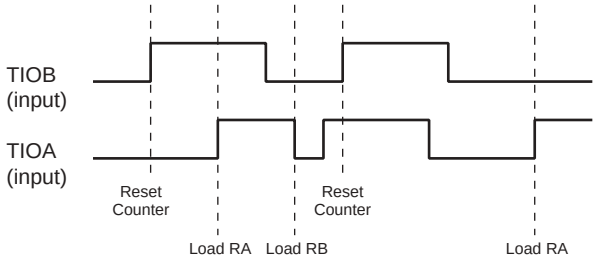
Figure 92 to Figure 95 show different applications using the capture mode.

For more details, refer to application notes.

Measure TIOA Pulse and Phase between TIOB and TIOA

A TIOBx rising edge resets and starts the counter. A rising TIOAx edge loads RA and a falling TIOAx edge loads RB. Once RB is loaded, a trigger restarts a capture cycle. RA contains the phase between TIOBx and TIOAx. (RB - RA) is the duration of the TIOAx pulse.

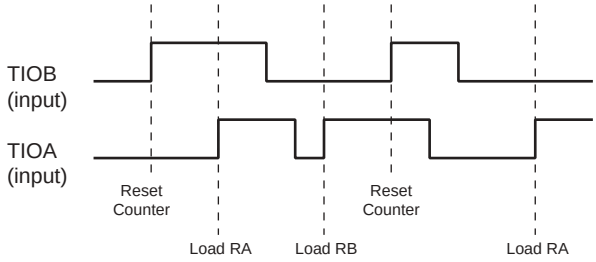
Figure 92. TIOA Pulse



Measure Duration between Two Successive Rising TIOA Edges

A TIOBx rising edge resets and starts the counter. The first rising TIOAx edge after the reset loads RA and a second loads RB. RA contains the phase between TIOBx and TIOAx. (RB-RA) is the period of the TIOAx pulse.

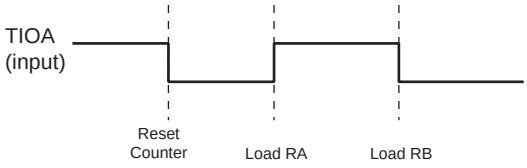
Figure 93. TIOA Edges



Measure the Duration of a TIOA Pulse or Period (TIOB Not Used)

A TIOAx falling edge resets, starts the counter and loads RB if RA is already loaded. A TIOAx rising edge loads RA. RA contains the duration of a TIOAx pulse (low level). RB contains the duration of the TIOAx period.

Figure 94. TIOA Pulse or Period



Event Counter on an External Clock TCLK

The counter is incremented with each TCLKx rising edge. This application can be generated in waveform mode. The counter value contains the number of detected TCLKx rising edges.

Figure 95. Event Counter on an External Clock TCLK

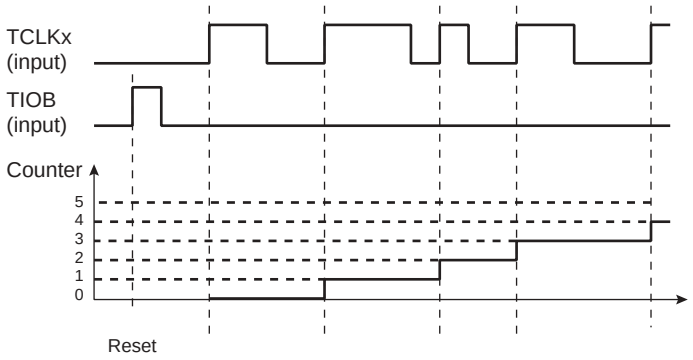
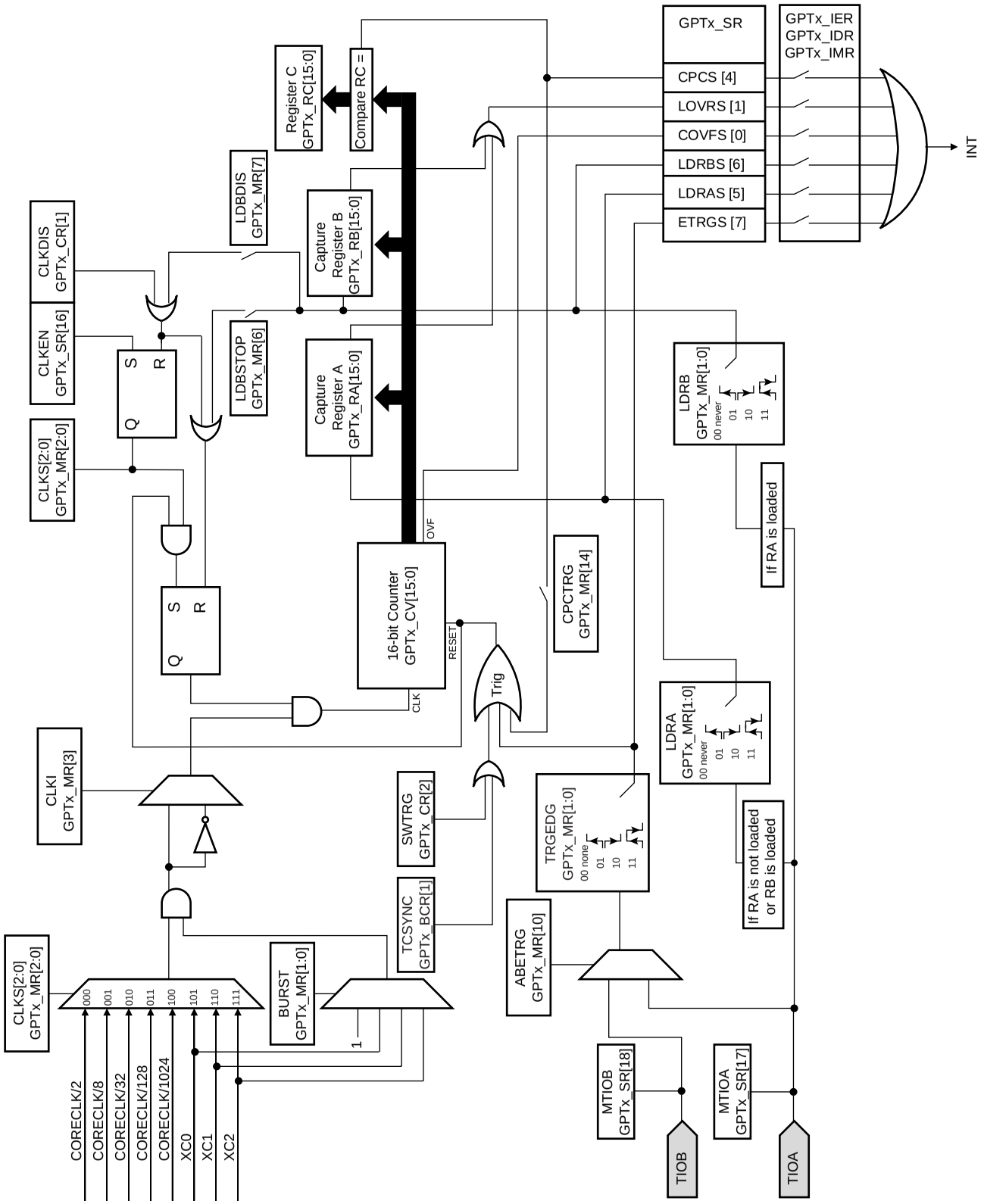


Figure 96. General-purpose Timer in Capture Mode



GPT Control Register in Capture Mode

Name: GPT_CR
Access: Write-only
Base Address: 0x60

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
–	–	–	–	SWTRG	CLKDIS	CLKEN	SWRST

- **SWRST: Software Reset**

0: No effect.

1: Generates a software reset.

A software triggered hardware reset of the channel is performed. It resets all the registers, including PIO and PMC registers (except GPTX_PMSR).

- **CLKEN: Counter Clock Enable**

0: No effect.

1: Enables counter clock if CLKDIS = 0.

- **CLKDIS: Counter Clock Disable**

0: No effect.

1: Disables counter clock.

- **SWTRG: Software Trigger**

0: No effect.

1: Generates a software trigger.

This bit generates a software trigger for resetting and starting the counter at the next valid counter clock edge when the counter clock is enabled.

Note: x: channel number

GPT Mode Register in Capture Mode

Name: GPT_MR
Access: Read/Write
Base Address: 0x64

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

23	22	21	20	19	18	17	16
–	–	–	–	LDRB[1:0]		LDRA[1:0]	

15	14	13	12	11	10	9	8
WAVE = 0	CPCTRG	–	–	–	ABETRG	ETRGEDG[1:0]	

7	6	5	4	3	2	1	0
LDBIS	LDBSTOP	BURST[1:0]		CLKI	CLKS[2:0]		

- **CLKS[2:0]: Clock Select**

CLKS[2:0]			Counter Clock Source
0	0	0	CORECLK/2
0	0	1	CORECLK/8
0	1	0	CORECLK/32
0	1	1	CORECLK/128
1	0	0	CORECLK/1024
1	0	1	XC0
1	1	0	XC1
1	1	1	XC2

MCKx consists of five external clocks.

XCx consists of three external clocks.

For more details, see “Clock Sources” on page 293.

- **CLKI: Clock Inverter**

0: Normal clock (The counter is incremented on a rising edge)

1: Inverted clock (The counter is incremented on a falling edge)

- **BURST[1:0]: Burst**

This signal is combined with the selected clock through a logical AND.

BURST[1:0]		Burst Signal Selected
0	0	None
0	1	XC0
1	0	XC1
1	1	XC2

For more details, see “Clock Sources” on page 293.

- **LDBSTOP Load RB Stops Counter**

0: The counter is not stopped when RB is loaded.

1: The counter is stopped when RB is loaded.

If the counter is stopped, it can restart (to 0x0000) just with a trigger condition.

If a TIOAx edge both induces a trigger condition and loads capture register B which in turn stops the counter, the trigger has no effect.

- **LDBDIS: Load RB Disables Clock**

0: The counter clock is not disabled when RB is loaded.

1: The counter clock is disabled and the counter stopped when RB is loaded.

If the counter clock is disabled, it can be enabled only by asserting CLKEN, bit [0] of the control register.

- **ETRGEDG[1:0]: External Trigger Edge**

The external trigger source is either TIOAx or TIOBx following ABETRG, bit [10] of the mode register.

ETRGEDG[1:0]		Edge
0	0	None
0	1	Rising edge
1	0	Falling edge
1	1	Each edge

When an external trigger is generated, three events occur:

- It resets and starts the counter.
- The ETRGS flag is set in the status register.
- If enabled, ETRGS interrupt is generated.

- **ABETRG: TIOA or TIOB as External Trigger**

0: Select TIOBx as external trigger

1: Select TIOAx as external trigger

Note: The counter can start only if the clock is enabled.

- **CPCTRG: Compare RC Trigger**

0: An equal condition on RC does not cause a trigger.

1: An equal condition on RC causes a trigger.

Note: The counter can start only if the clock is enabled.

- **WAVE: Waveform**

0: Capture mode.

1: Waveform mode.

Note: The capture mode is the default mode after hardware reset.

- **LDRA[1:0]: Load RA**

These two bits activate one of four possible TIOAx edge conditions to load RA.

Note: The application must ensure that the event that loads RA occurs after the next counter clock edge following the configuration of LDRA (the counter is reset on the next counter clock edge following the configuration of LDRA).

- **LDRB[1:0]: Load RB**

These two bits activate one of four possible TIOAx edge conditions to load RB.

LDRx		Edge
0	0	None
0	1	Rising edge
1	0	Falling edge
1	1	Each edge

GPT Status Register in Capture Mode

Name: GPT_SR
Access: Read-only
Base Address: 0x70

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

23	22	21	20	19	18	17	16
–	–	–	–	–	TCLKS	TIOAS	TIOBS

15	14	13	12	11	10	9	8
–	–	–	–	–	MTIOB	MTIOA	CLKSTA

7	6	5	4	3	2	1	0
ETRGS	LDRBS	LDRAS	CPCS	–	–	LOVRS	COVFS

Note: This register is a "read-active" register, which means that reading it can affect the state of some bits. When reading GPT_SR register, following bits are cleared if set: COVFS, LOVRS, CPCS, LDRAS, LDRBS, ETRGS, TIOBS, TIOAS and TCLKS. When debugging, to avoid this behavior, users should use ghost registers (see “Ghost Registers” on page 7).

- **COVFS: Counter Overflow Status**

This bit is set when a counter overflow is detected. An overflow occurs when the counter reaches its maximal value 0xFFFF (216 - 1) and passes to 0x0000.

0: No overflow detected.

1: Overflow detected since last read of GPTX_SR.

- **LOVRS: Load Overrun Status**

This bit is set when an overrun is detected. An overrun occurs when the capture registers A or B are reloaded before being read.

0: No overrun detected.

1: An overrun detected since last read of GPTX_SR.

- **CPCS: Compare Register C Status**

This bit is set when the counter reaches the register C value.

0: Compare C condition has not occurred since last read of GPTX_SR.

1: Compare C condition has occurred since last read of GPTX_SR.

- **LDRAS: Load Register A Status**

0: Register A not loaded.

1: Register A loaded since last read of GPTX_SR.

- **LDRBS: Load Register B Status**

0: Register B not loaded.

1: Register B loaded since last read of GPTX_SR.

- **ETRGS: External Trigger Status**

This bit is set when an external trigger is detected. An external trigger occurs with a valid edge (the edge polarity is set by ETRGEDG[1:0] of the mode register) on the valid trigger pin (set by ABETRG of the mode register).

0: External trigger not detected.

1: External trigger detected since last read of GPTX_SR.

- **CLKSTA: Clock Status**

0: Clock disabled.

1: Clock enabled.

- **MTIOA: TIOA Mirror**

This bit reflects the TIOAx pin value.

As TIOAx is an input after a hardware reset, its reset value is undefined.

- **MTIOB: TIOB Mirror**

This bit reflects the TIOBx pin value.

As TIOBx is an input after a hardware reset, its reset value is undefined.

- **TIOBS: TIOB Status**

0: At least one input change has been detected on the pin TIOBx since the register was last read.

1: No input change has been detected on the TIOBx pin since the register was last read.

- **TIOAS: TIOA Status**

0: At least one input change has been detected on the TIOAx pin since the register was last read.

1: No input change has been detected on the TIOAx pin since the register was last read.

- **TCLKS: TCLK Status**

0: At least one input change has been detected on the TCLKx pin since the register was last read.

1: No input change has been detected on the TCLKx pin since the register was last read.

Note: X: channel number

GPT Interrupt Enable Register in Capture Mode

Name: GPT_IER
Access: Write-only
Base Address: 0x74

GPT Interrupt Disable Register in Capture Mode

Name: GPT_IDR
Access: Write-only
Base Address: 0x78

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

23	22	21	20	19	18	17	16
–	–	–	–	–	TCLKS	TIOAS	TIOBS

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
ETRGS	LDRBS	LDRAS	CPCS	–	–	LOVRS	COVFS

- **COVFS: Counter Overflow Status**

0: COVFS interrupt is disabled.

1: COVFS interrupt is enabled.

- **LOVRS: Load Overrun Status**

0: LOVRS interrupt is disabled.

1: LOVRS interrupt is enabled.

- **CPCS: Compare Register C Status**

0: CPCS interrupt is disabled.

1: CPCS interrupt is enabled.

- **LDRAS: Load Register A Status**

0: LDRAS interrupt is disabled.

1: LDRAS interrupt is enabled.

- **LDRBS: Load Register B Status**

0: LDRBS interrupt is disabled.

1: LDRBS interrupt is enabled.

- **ETRGS: External Trigger Status**

0: ETRGS interrupt is disabled.

1: ETRGS interrupt is enabled.

- **TIOBS: TIOB Status**

0: TIOBS interrupt is disabled.

1: TIOBS interrupt is enabled.

- **TIOAS: TIOA Status**

0: TIOAS interrupt is disabled.

1: TIOAS interrupt is enabled.

- **TCLKS: TCLK Status**

0: TCLKS interrupt is disabled.

1: TCLKS interrupt is enabled.

GPT Interrupt Mask Register in Capture Mode

Name: GPT_IMR
Access: Read-only
Base Address: 0x7C

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	TCLKS	TIOAS	TIOBS
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
ETRGS	LDRBS	LDRAS	CPCS	–	–	LOVRS	COVFS

- **COVFS: Counter Overflow Status**

0: COVFS interrupt is disabled.

1: COVFS interrupt is enabled.

- **LOVRS: Load Overrun Status**

0: LOVRS interrupt is disabled.

1: LOVRS interrupt is enabled.

- **CPCS: Compare Register C Status**

0: CPCS interrupt is disabled.

1: CPCS interrupt is enabled.

- **LDRAS: Load Register A Status**

0: LDRAS interrupt is disabled.

1: LDRAS interrupt is enabled.

- **LDRBS: Load Register B Status**

0: LDRBS interrupt is disabled.

1: LDRBS interrupt is enabled.

- **ETRGS: External Trigger Status**

0: ETRGS interrupt is disabled.

1: ETRGS interrupt is enabled.

- **TIOBS: TIOB Status**

0: TIOBS interrupt is disabled.

1: TIOBS interrupt is enabled.

- **TIOAS: TIOA Status**

0: TIOAS interrupt is disabled.

1: TIOAS interrupt is enabled.

- **TCLKS: TCLK Status**

0: TCLKS interrupt is disabled.

1: TCLKS interrupt is enabled.

GPT Counter Value in Capture Mode

Name: GPT_CV
Access: Read-only
Base Address: 0x80

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

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

15	14	13	12	11	10	9	8
CV[15:8]							

7	6	5	4	3	2	1	0
CV[7:0]							

- **CV[15:0]: Counter Value**

These 16 bits contain the counter value in real time.

The maximal counter value is 0xFFFF = 65535.

When a trigger occurs, the counter will be reset to 0x0000 at the next valid counter clock edge.

GPT Register A in Capture Mode

Name: GPT_RA
Access: Read-only
Base Address: 0x84

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

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

15	14	13	12	11	10	9	8
RA[15:8]							

7	6	5	4	3	2	1	0
RA[7:0]							

- **RA[15:0]: Register A Value**

This register is loaded with the current counter value when a valid edge occurs on TIOAx pin.

This valid edge is defined by LDRA[1:0] of the mode register.

When this register is loaded, two events occur:

– The LDRAS flag is set in the status register.
– If enabled, LDRAS interrupt is generated.

This register can not be loaded if the counter is stopped or the counter clock disabled.

GPT Register B in Capture Mode

Name: GPT_RA
Access: Read-only
Base Address: 0x88

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

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

15	14	13	12	11	10	9	8
RB[15:8]							

7	6	5	4	3	2	1	0
RB[7:0]							

- **RB[15:0]: Register B Value**

This register is loaded with the current counter value when a valid edge occurs on TIOBx pin.

This valid edge is defined by LDRB[1:0] of the mode register.

When this register is loaded, four events occur:

- The LDRBS flag is set in the status register.
- If enabled, LDRBS interrupt is generated.
- The counter clock can be disabled according to LDBDIS (bit [7] of the mode register).
- The counter can be stopped according to LDBSTOP (bit [6] of the mode register).

This register cannot be loaded if the counter is stopped or the counter clock disabled.

GPT Register C in Capture Mode

Name: GPT_RC
Access: Read/Write
Base Address: 0x8C

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

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

15	14	13	12	11	10	9	8
RC[15:8]							

7	6	5	4	3	2	1	0
RC[7:0]							

- **RC[15:0]: Register C Value**

When the counter reaches this value, three events occur:

– The CPCS flag is set in the status register.
– If enabled, CPCS interrupt is generated.
– If CPCTRG (bit [14] of the mode register) is high, the counter is reset and restarts at 0x0000.

General-purpose Timer in Waveform Mode

Description

The waveform mode is entered by setting the bit WAVE in the GPTX_MR to 1. It forces TIOAx as an output pin. TIOBx can be used either as an output (dual waveform mode) or as an input (single waveform mode).

The waveform mode provides the possibility to generate either symmetrical or variable duty-cycle waveforms.

The TIOA pin is controlled (set, cleared or toggled) by four events:

- a software trigger
- an external event edge (rising, falling edge or both)
- an equality between the counter and compare register A value
- an equality between the counter and compare register C value

As an output pin, the TIOB pin is controlled (set, cleared or toggled) by four events:

- a software trigger
- an external event edge (rising, falling edge or both)
- an equality between the counter and compare register B value
- an equality between the counter and compare register C value

When TIOB is used as an external trigger source, the compare register B is not used.

When an equal condition on compare register C is detected, one of three events can occur:

- The counter can reset and start at the next valid counter clock edge.
- The counter can be stopped.
- The counter can be stopped and the counter clock disabled.

If this condition restarts the counter, the user can generate a continuous wave with a period proportional to compare register C +1 value. If it does not restart the counter, the user can generate a continuous waveform with a period proportional to 0xFFFF (maximal counter value: 2^{16} - 1).

The user may choose an internal clock source (CORECLK/2, CORECLK/8, CORECLK/32, CORECLK/128 or CORECLK/1024) or external clock (TCLK0, TCLK1 or TCLK2). A burst mode is available. It generates a burst clock. For more details, refer to “Clock Sources” on page 293.

Five interrupts can be produced:

- External trigger detected
- Counter overflow (when the counter passes from 0xFFFF to 0x0000)
- Compare RA (the counter reaches the value stored in register A)
- Compare RB (the counter reaches the value stored in register B)
- Compare RC (the counter reaches the value stored in register C)

Finally, the synchronize register can be used to cause a software trigger for reset and start the counter at the next valid counter clock edge on all channels at the same time.

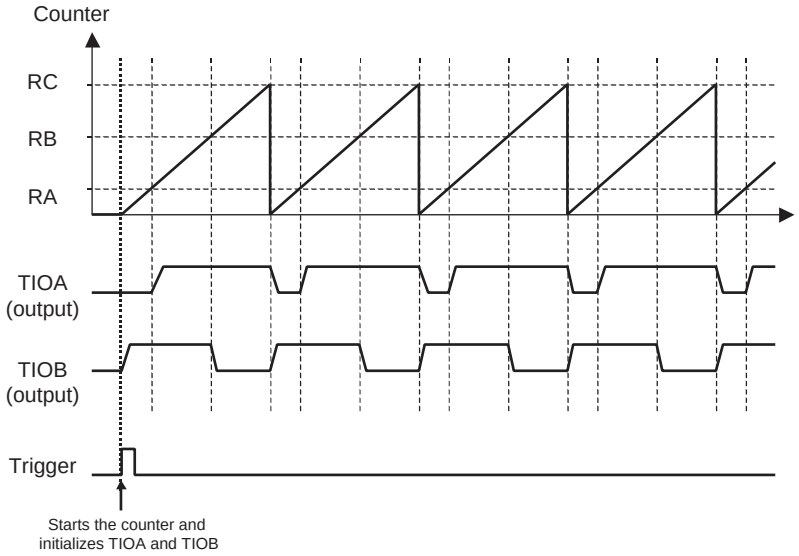
Figure 97 on page 329 to Figure 101 on page 331 show different applications possible with the waveform mode.

Dual Pulse Width Modulation (PWM) Generation

TIOAx is toggled by RA and RC, TIOBx by RB and RC.

RC contains frequency of both signals. RA determines the TIOAx duty cycle and RB the TIOBx duty cycle.

Figure 97. Dual Pulse Width Modulation



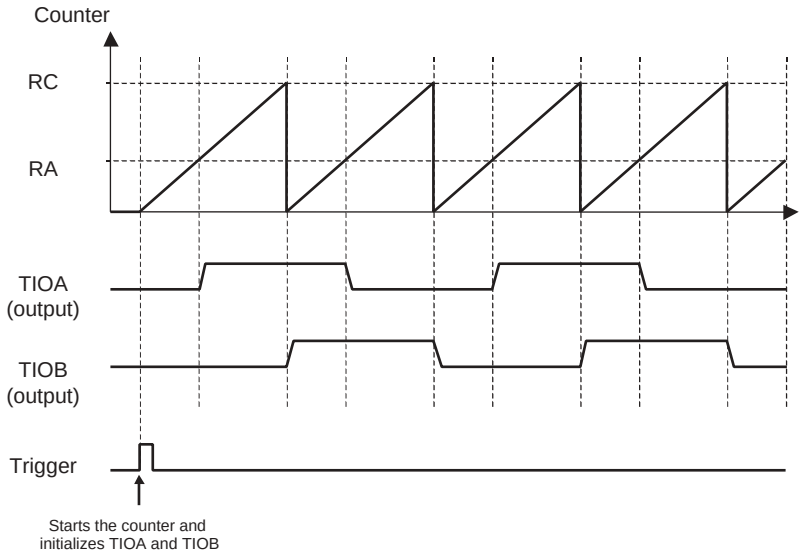
Generation of Two Identical Out-of-phase Square-wave Signals

TIOAx is toggled each time the counter reaches RA value, TIOBx each time the counter reaches RC value.

A trigger (external or software) starts the counter and initializes TIOAx and TIOBx.

RC contains the frequency of both signals. RA contains the delay between the signals.

Figure 98. Two Square Signals



Pulse Generation

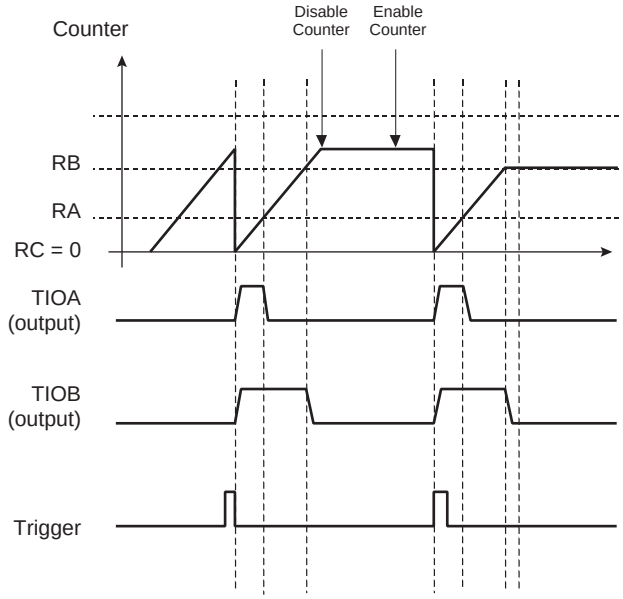
This application generates only one pulse on TIOAx and on TIOBx between two triggers. The pulses start with a trigger. The duration is given by the value of RA for TIOAx pulse and the value of RB for TIOBx pulse. After each new trigger, another pulse is generated. When a trigger occurs, the counter is reset at the next valid counter clock edge when the counter clock is enabled.

If we want to start the pulse exactly when the counter is reset, RC must equal 0. Thus, it is the comparison with RC = 0 that starts the pulse, and not the trigger.

In this case, the user must disable the counter clock when a compare RB is detected to stop the counter.

To accept a new trigger, the user must then re-enable the counter clock.

Figure 99. Pulse Generation

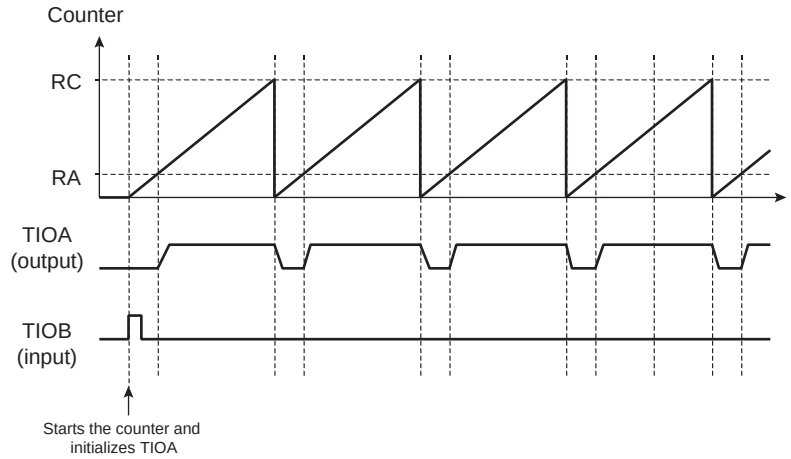


Trigger on TIOB Input Pin

In each of the previous examples, TIOBx is used as an output. It is possible to use it as a trigger input and generate only TIOAx output signal.

The following application is the same as “Dual Pulse Width Modulation (PWM) Generation” on page 329, where TIOBx is not used as an output but as an input.

Figure 100. Single Waveform with Trigger on TIOB

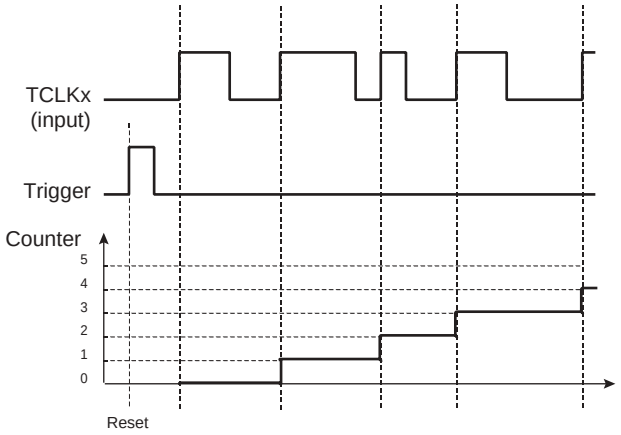


Event Counter on an External Clock (TCLK)

The counter is incremented with each TCLKx rising edge.

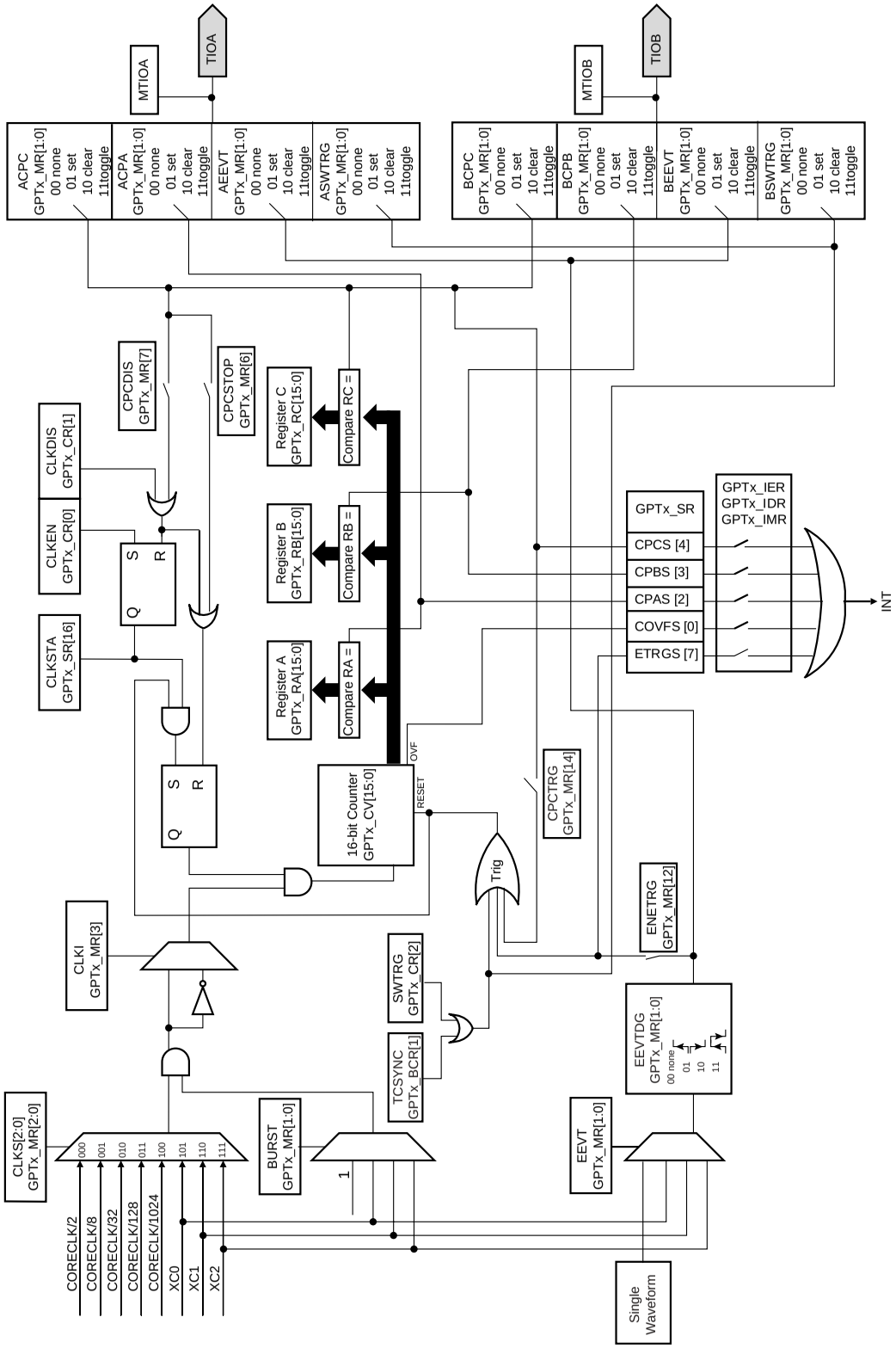
This application can be generated in capture mode.

Figure 101. Event Counter



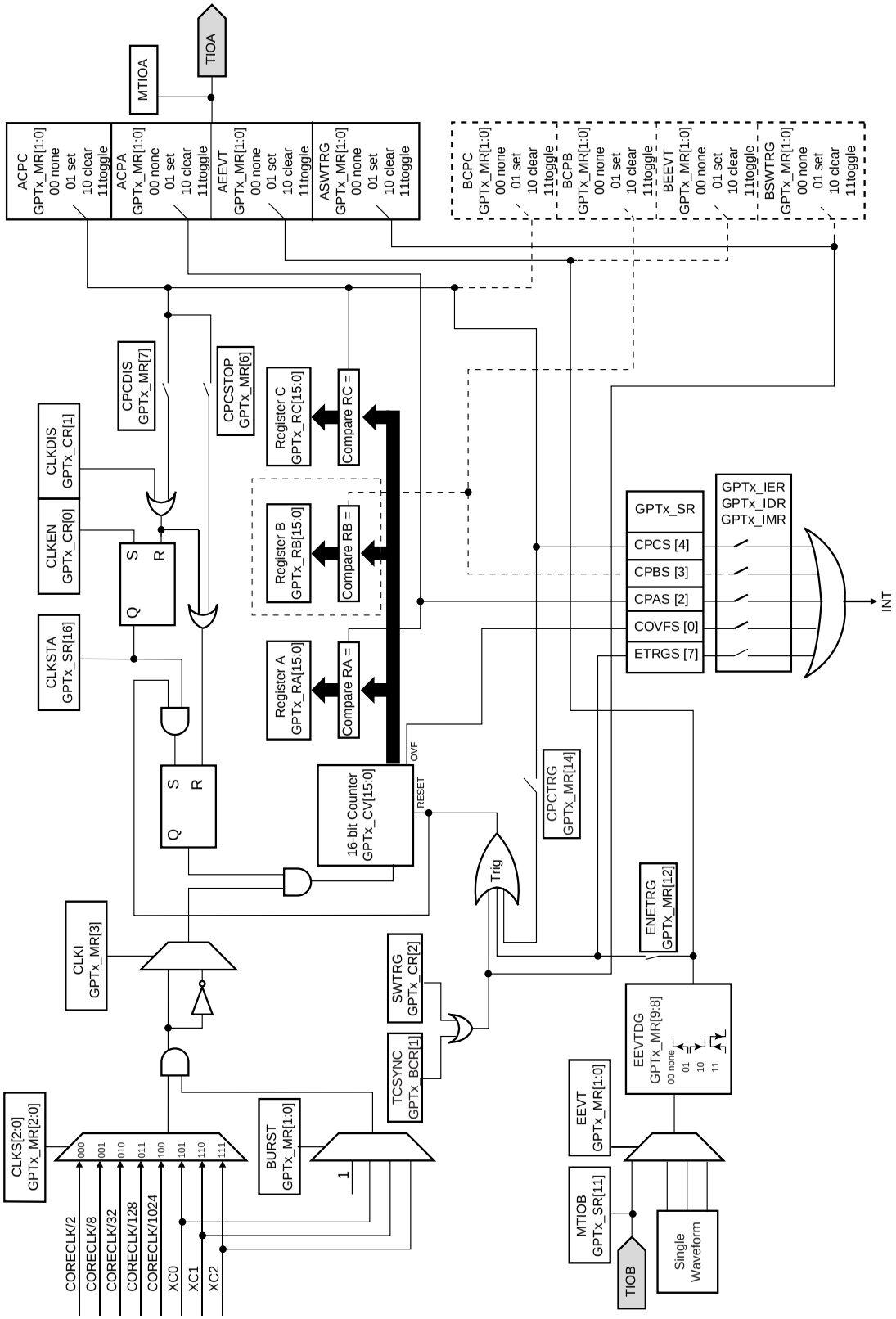
Dual Waveform Mode

Figure 102. Dual Waveform Mode



Single Waveform Mode

Figure 103. Single Waveform Mode (The dotted blocks are not used in this case.)



GPT Control Register in Waveform Mode

Name: GPT_CR
Access: Write-only
Base Address: 0x60

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
–	–	–	–	SWTRG	CLKDIS	CLKEN	SWRST

- **SWRST: Software Reset**

0: No effect.

1: Generates a software reset.

A software triggered hardware reset of the channel is performed. It reset all the registers, including PIO and PMC registers (except GPTX_PMSR).

- **CLKEN: Counter Clock Enable**

0: No effect.

1: Enables counter clock if CLKDIS = 0.

- **CLKDIS: Counter Clock Disable**

0: No effect.

1: Disables counter clock.

- **SWTRG: Software Trigger**

0: No effect.

1: Generates a software trigger.

This bit generates a software trigger for resetting and starting the counter at the next valid counter clock edge when the counter clock is enabled.

GPT Mode Register in Waveform Mode

Name: GPT_MR
Access: Read/Write
Base Address: 0x64

31	30	29	28	27	26	25	24
BSWTRG[1:0]		BEEVT[1:0]		BCPC[1:0]		BCPB[1:0]	
23	22	21	20	19	18	17	16
ASWTRG[1:0]		AEEVT[1:0]		ACPC[1:0]		ACPA[1:0]	
15	14	13	12	11	10	9	8
WAVE = 1	CPCTRG	–	ENETRG	EEVT[1:0]		EEVTEDG[1:0]	
7	6	5	4	3	2	1	0
CPCDIS	CPCSTOP	BURST[1:0]		CLKI	CLKS[2:0]		

- **CLKS[2:0]: Clock Select**

CLKS[2:0]			Counter Clock Source
0	0	0	CORECLK/2
0	0	1	CORECLK/8
0	1	0	CORECLK/32
0	1	1	CORECLK/128
1	0	0	CORECLK/1024
1	0	1	XC0
1	1	0	XC1
1	1	1	XC2

For more details, see “Clock Sources” on page 293.

- **CLKI: Clock Inverter**

0: Normal clock (The counter is incremented on a rising edge)

1: Inverted clock (The counter is incremented on a falling edge)

- **BURST[1:0]: Burst**

This signal is combined with the selected clock through a logical AND.

BURST[1:0]		Burst signal selected
0	0	None
0	1	XC0
1	0	XC1
1	1	XC2

For more details, see “Clock Sources” on page 293.

- **CPCSTOP: Compare RC Stops the Counter**

0: The counter is not stopped when an equal condition on RC is detected.

1: The counter is stopped when an equal condition on RC is detected.

If the counter is stopped, it can restart (to 0x0000) just with a trigger condition.

If an equal condition on RC induces both trigger condition and stop counter, the trigger will have no effect.

- **CPCDIS: Compare RC Disables Clock**

0: The counter clock is not disabled when an equal condition on RC is detected.

1: The counter clock is disabled and the counter stopped when an equal condition on RC is detected.

If the counter clock is disabled, it can be enabled only by asserting CLKEN, bit [1] of the control register.

- **EEVTEDG[1:0]: External Event Edge**

These two bits activate one of four possible external event modes. The external event source is selected by EEVT[1:0] of the mode register.

EEVTEDG[1:0]		Edge
0	0	None
0	1	Rising edge
1	0	Falling edge
1	1	Each edge

When an external event is generated, five events occur:

- The ETRGS flag is set in the status register.
- If enabled, ETRGS interrupt is generated.
- It can reset and start the counter at the next valid counter clock edge if ENETRG (bit [12] of the mode register) is high.
- TIOAx pin can be set, clear, toggle or unchanged following AEEVT[1:0] of the mode register.
- TIOBx pin can be set, clear, toggle or unchanged following BEEVT[1:0] of the mode register.

- **EEVT[1:0]: External Event**

These bits select an external event source among four pins:

EEVT[1:0]		External Trigger
0	0	TIOBx
0	1	XC0
1	0	XC1
1	1	XC2

If TIOBx is selected, the mode is in single waveform mode (see Figure 103 on page 333). TIOAx is used as an output and TIOBx as an input. The following bits are disabled:

- BSWTRG[1:0] of the mode register
- BEEVT[1:0] of the mode register
- BCPC[1:0] of the mode register
- BCPB[1:0] of the mode register
- Compare register B

If an external clock is selected, the mode is in dual waveform mode. TIOAx and TIOBx are used as outputs.

- **ENETRG: Enable External Trigger**

This bit determines whether an external event can be used as a trigger to reset and start the counter at the next valid counter clock edge. The external event source is selected by EEVT[1:0] of the mode register.

0: External event does not reset and start the counter. Selected external trigger can only be used to control TIOAx and TIOBx.

1: External trigger resets and starts the counter.

Note: The counter can start only if the clock is enabled.

- **CPCTRG: Compare RC Trigger**

This bit determines whether an equal condition on the Compare C register can cause a trigger (reset and start the counter at the next valid counter clock edge).

0: An equal condition on RC does not cause a trigger.

1: An equal condition on RC causes a trigger.

Note: The counter can start only if the clock is enabled.

- **WAVE: Waveform**

0: Capture mode.

1: Waveform mode.

- **ACPA: TIOA Compare A**

These two bits determine the effect on the TIOAx output pin caused by an equal comparison between the counter and the Compare Register A value. See the bit description table in “ASWTRG: TIOA Software Trigger” where xxx = CPA.

Note: If several events that control TIOAx output (set, clear or toggle) arrive at the same time, only one has an action according to the following priority order:
1. ASWTRG (highest priority)
2. AEEVT
3. ACPC
4. ACPA

- **ACPC: TIOA Compare C**

These two bits determine the effect on the TIOAx output pin caused by an equal comparison between the counter and the Compare register C value. See the bit description table in “ASWTRG: TIOA Software Trigger” where xxx = CPC.

- **AEEVT: TIOA External Event**

These two bits determine the effect on the TIOAx output pin caused by an external event. The external event source is selected by EEVT[1:0] of the mode register. See the bit description table in “ASWTRG: TIOA Software Trigger” where xxx = EEVT.

- **ASWTRG: TIOA Software Trigger**

These two bits determine the effect on the TIOAx output pin caused by a software trigger.

See the following table where xxx = SWTRG.

Axxx		Waveform Pin
0	0	None
0	1	Set
1	0	Clear
1	1	Toggle

- **BCPB: TIOB Compare B**

These two bits determine the effect on the TIOBx output pin caused by an equal comparison between the counter and the Compare Register B value. These bits are active only if TIOBx is not an input (see “EEVT[1:0]: External Event” of the “GPT Mode Register in Waveform Mode” on page 335). See the bit description table in “BSWTRG: TIOB Software Trigger” where xxx = CPB.

Note: If several events that control TIOBx output (set, clear or toggle) arrive at the same time, only one has an action according to the following priority:
1. BSWTRG (highest priority)
2. BEEVT
3. BCPC
4. BCPB

- **BCPC: TIOB Compare C**

These two bits determine the effect on the TIOBx output pin caused by an equal comparison between the counter and the Compare register C value. These bits are active only if TIOBx is not an input (see “EEVT[1:0]: External Event” of the “GPT Mode Register in Waveform Mode” on page 335). See the bit description table in “BSWTRG: TIOB Software Trigger” where xxx = CPC.

- **BEEVT: TIOB External Event**

These two bits determine the effect on the TIOBx output pin caused by an external event. The external event source is selected by EEVT[1:0] of the mode register. These bits are active only if TIOBx is not an input (see “EEVT[1:0]: External Event” of the “GPT Mode Register in Waveform Mode” on page 335). See the bit description table in “BSWTRG: TIOB Software Trigger” where xxx = EEVT.

- **BSWTRG: TIOB Software Trigger**

These two bits determine the effect on the TIOBx output pin caused by a software trigger. These bits are active only if TIOBx is not an input (see “EEVT[1:0]: External Event” of the “GPT Mode Register in Waveform Mode” on page 335). See following table with xxx = SWTRG:

Bxxx		Waveform Pin
0	0	None
0	1	Set
1	0	Clear
1	1	Toggle

GPT Status Register in Waveform Mode

Name: GPT_SR
Access: Read-only
Base Address: 0x70

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	TCLKS	TIOAS	TIOBS
15	14	13	12	11	10	9	8
–	–	–	–	–	MTIOB	MTIOA	CLKSTA
7	6	5	4	3	2	1	0
ETRGS	–	–	CPCS	CPBS	CPAS	–	COVFS

Note: This register is a "read-active" register; thus, reading it can affect the state of some bits. When reading GPT_SR register, following bits are cleared if set: COVFS, CPAS, CPBS, CPCS, ETRGS, TIOBS, TIOAS and TCLKS. When debugging, to avoid this behavior, users should use ghost registers (see “Ghost Registers” on page 7).

- **COVFS: Counter Overflow Status**

This bit is set when a counter overflow is detected. An overflow occurs when the counter reaches its maximal value 0xFFFF (2^{16}-1) and passes to 0x0000.

0: No overflow detected

1: Overflow detected since last read of GPTX_SR

- **CPAS: Compare Register A Status**

This bit is set when the counter reaches the register A value.

0: Compare A condition has not occurred since last read of GPTX_SR

1: Compare A condition has occurred since last read of GPTX_SR

- **CPBS: Compare Register B Status**

This bit is set when the counter reaches the register B value.

0: Compare B condition has not occurred since last read of GPTX_SR

1: Compare B condition has occurred since last read of GPTX_SR

- **CPCS: Compare Register C Status**

This bit is set when the counter reaches the register C value.

0: Compare C condition has not occurred since last read of GPTX_SR

1: Compare C condition has occurred since last read of GPTX_SR

- **ETRGS: External Trigger Status**

This bit is set when an external trigger is detected. An external trigger occurs with a valid edge (the edge polarity is set by EEVTEDG[1:0] of the mode register) on the valid trigger pin (set by EEVT[1:0] of the mode register if ENETRG, bit 12 of the Mode Register, is high).

0: External trigger not detected

1: External trigger detected since last read of GPTX_SR

- **CLKSTA: Clock Status**

0: Clock disabled

1: Clock enabled

- **MTIOA: TIOA Mirror**

This bit reflects the TIOAx pin value.

Its reset value is undefined because the operating mode after hardware reset is capture mode.

- **MTIOB: TIOB Mirror**

This bit reflects the TIOBx pin value.

Its reset value is undefined because the operating mode after hardware reset is capture mode.

- **TIOBS: TIOB Status**

0: At least one input change has been detected on the pin TIOBx since the register was last read.

1: No input change has been detected on the TIOBx pin since the register was last read.

- **TIOAS: TIOA Status**

0: At least one input change has been detected on the TIOAx pin since the register was last read.

1: No input change has been detected on the TIOAx pin since the register was last read.

- **TCLKS: TCLK Status**

0: At least one input change has been detected on the TCLKx pin since the register was last read.

1: No input change has been detected on the TCLKx pin since the register was last read.

Note: X: Channel number

GPT Interrupt Enable Register in Waveform Mode

Name: GPT_IER
Access: Write-only
Base Address: 0x74

GPT Interrupt Disable Register in Waveform Mode

Name: GPT_IDR
Access: Write-only
Base Address: 0x78

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	TCLKS	TIOAS	TIOBS
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
ETRGS	–	–	CPCS	CPBS	CPAS	–	COVFS

- **COVFS: Counter Overflow Status**

0: COVFS interrupt is disabled.

1: COVFS interrupt is enabled.

- **CPAS: Compare Register A Status**

0: CPAS interrupt is disabled.

1: CPAS interrupt is enabled.

- **CPBS: Compare Register B Status**

0: CPBS interrupt is disabled.

1: CPBS interrupt is enabled.

- **CPCS: Compare Register C Status**

0: CPCS interrupt is disabled.

1: CPCS interrupt is enabled.

- **ETRGS: External Trigger Status**

0: ETRGS interrupt is disabled.

1: ETRGS interrupt is enabled.

- **TIOBS: TIOB Status**

0: TIOBS interrupt is disabled.

1: TIOBS interrupt is enabled.

- **TIOAS: TIOA Status**

0: TIOAS interrupt is disabled.

1: TIOAS interrupt is enabled.

- **TCLKS: TCLK Status**

0: TCLKS interrupt is disabled.

1: TCLKS interrupt is enabled.

GPT Interrupt Mask Register in Waveform Mode

Name: GPT_IMR
Access: Read-only
Base Address: 0x7C

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	TCLKS	TIOAS	TIOBS
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
ETRGS	–	–	CPCS	CPBS	CPAS	–	COVFS

- **COVFS: Counter Overflow Status**

0: COVFS interrupt is disabled.

1: COVFS interrupt is enabled.

- **CPAS: Compare Register A Status**

0: CPAS interrupt is disabled.

1: CPAS interrupt is enabled.

- **CPBS: Compare Register B Status**

0: CPBS interrupt is disabled.

1: CPBS interrupt is enabled.

- **CPCS: Compare Register C Status**

0: CPCS interrupt is disabled.

1: CPCS interrupt is enabled.

- **ETRGS: External Trigger Status**

0: ETRGS interrupt is disabled.

1: ETRGS interrupt is enabled.

- **TIOBS: TIOB Status**

0: TIOBS interrupt is disabled.

1: TIOBS interrupt is enabled.

- **TIOAS: TIOA Status**

0: TIOAS interrupt is disabled.

1: TIOAS interrupt is enabled.

- **TCLKS: TCLK Status**

0: TCLKS interrupt is disabled.

1: TCLKS interrupt is enabled.

GPT Counter Value in Waveform Mode

Name: GPT_CV
Access: Read-only
Base Address: 0x80

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

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

15	14	13	12	11	10	9	8
CV[15:8]							

7	6	5	4	3	2	1	0
CV[7:0]							

- **CV[15:0]: Counter Value**

These 16 bits contain the counter value in real time.

The maximal counter value is 0xFFFF 2^{16} - 1 = 65535.

When a trigger occurs, the counter will be reset to 0x0000 at the next valid counter clock edge.

GPT Register A in Waveform Mode

Name: GPT_RA
Access: Read/Write
Base Address: 0x84

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

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

15	14	13	12	11	10	9	8
RA[15:8]							

7	6	5	4	3	2	1	0
RA[7:0]							

- **RA[15:0]: Register A Value**

When the counter reaches this value, three events occur:

– The CPAS flag is set in the status register.
– If enabled, CPAS interrupt is generated.
– TIOAx pin can be set, clear, toggle or unchanged following bits ACPA[1:0] of the mode register.

GPT Register B in Waveform Mode

Name: GPT_RB

Access: Read/Write

Base Address: 0x88

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

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

15	14	13	12	11	10	9	8
RB[15:8]							

7	6	5	4	3	2	1	0
RB[7:0]							

- **RB[15:0]: Register B Value**

When the counter reaches this value, three events occur:

– The CPBS flag is set in the status register.

– If enabled, CPBS interrupt is generated.

– TIOBx pin can be set, clear, toggle or unchanged following bits BCPB[1:0] of the mode register.

These bits are active only if TIOB is not an input (see “EEVT[1:0]: External Event” of the “GPT Mode Register in Waveform Mode” on page 335).

GPT Register C in Waveform Mode

Name: GPT_RC
Access: Read/Write
Base Address: 0x8C

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

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

15	14	13	12	11	10	9	8
RC[15:8]							

7	6	5	4	3	2	1	0
RC[7:0]							

- **RC[15:0]: Register C Value**

When the counter reaches this value, seven events can occur:

- The CPCS flag is set in the status register.
- If enabled, CPCS interrupt is generated.
- If bit CPCTRG (bit [14] of the GPTX_MR) is high, the counter is reset and restarts at 0x0000 at the next valid counter clock edge.
- The counter clock can be disabled according to CPCDIS (bit [7] of the mode register).
- The counter can be stopped according to CPCSTOP (bit [6] of the mode register).
- TIOAx pin can be set, clear, toggle or unchanged following bits ACPC[1:0] of the mode register.
- TIOBx pin can be set, clear, toggle or unchanged following bits BCPC[1:0] of the mode register.

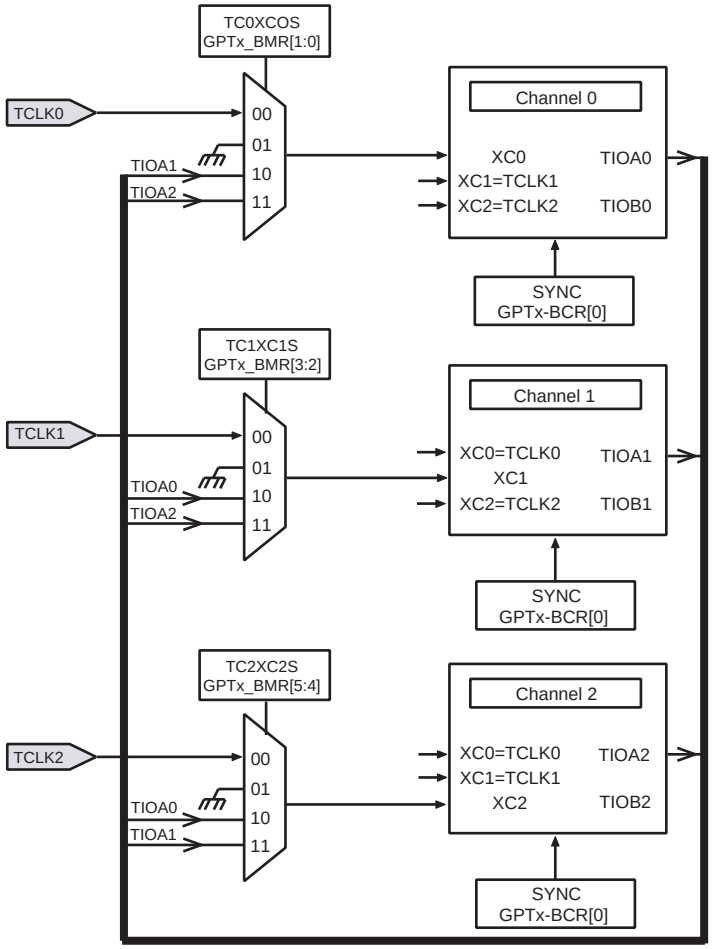
Timer Controller Block Programming

This module controls the entire timer controller with its three channels.

It has two functions:

- The block control register provides the means to synchronize the three timer channels. It can generate a software trigger on all three channels at exactly the same time.
- The block mode register provides the means to daisy chain two or three channels. Thus the user can improve counter capacity.

Figure 104. Timer Controller Block Programming Diagram



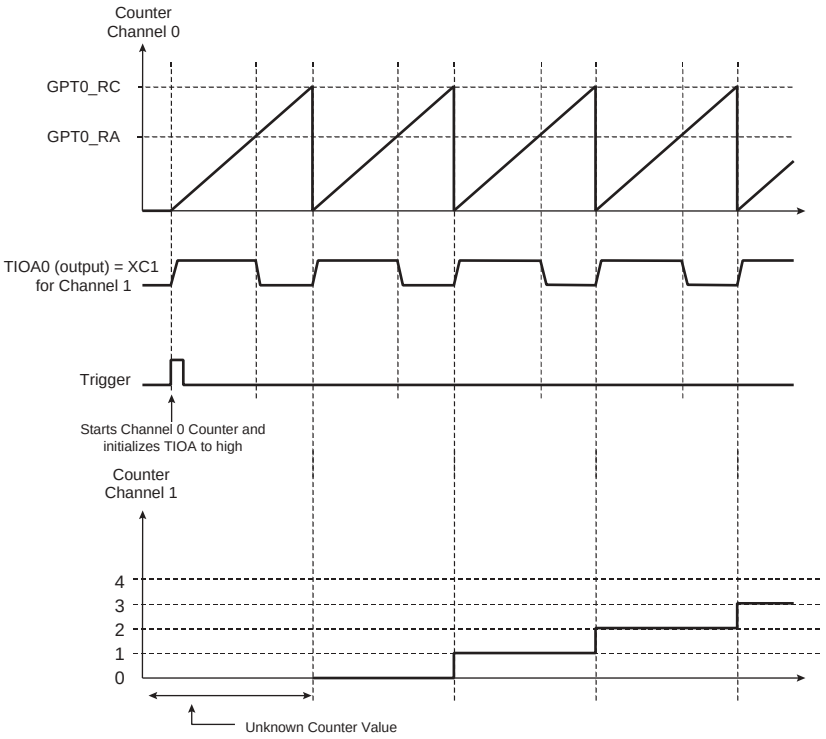
External Clock Generation for Channel 1 Using Channel 0

With GPT0_RA and GPT0_RC, Channel 0 generates a pulse width modulation output (PWM) in waveform mode that is used as an external clock by Channel 1.

Note that if RC is not used, this generates a 32-bit counter.

Each time the counter 0 passes 0xFFFF (overflow condition), this increments the counter by 1.

Figure 105. Timer Controller Block Programming Application



GPT Block Control Register

Name: GPT_BCR
Access: Write-only
Base Address: 0x00

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
–	–	–	–	–	–	TCSYNC	SWRST

- **SWRST: Software Reset**

This bit generates a software reset on the three timer channels simultaneously.

0: No effect.

1: Generates a software reset.

- **TCSYNC: Synchronization Bit**

This bit generates a software trigger on the three channels of the general-purpose timer simultaneously.

A software trigger resets and starts the counter at the next valid counter clock edge.

0: No effect.

1: Resets and starts all three timer channel counters simultaneously.

GPT Block Mode Register

Name: GPT_BMR
Access: Read/Write
Base Address: 0x04

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
–	–	–	–	–	–		

- **TC0XC0S[1:0]: TCLK0 XC0 Selection**

These bits select the external clock XC0 source for the channel 0.

TC0XC0S[1:0]		Selected Signal
0	0	TCLK0
0	1	none
1	0	TIOA1
1	1	TIOA2

- **TC1XC1S[1:0]: TCLK1 XC1 Selection**

These bits select the external clock XC1 source for the channel 1.

TC1XC1S[1:0]		Selected Signal
0	0	TCLK1
0	1	none
1	0	TIOA0
1	1	TIOA2

- **TC2XC2S[1:0]: TCLK2 XC2 Selection**

These bits select the external clock XC2 source for the channel 2.

TC2XC2S[1:0]		Selected Signal
0	0	TCLK2
0	1	none
1	0	TIOA0
1	1	TIOA1

Electrical Characteristics

Applicable over recommended operating temperature and voltage range.

3V Core and I/O Characteristics

Table 59. 3V Power Supply

Symbol	Parameter	Min	Typ	Max	Units
$V_{VDDCORE}$	Supply voltage	3.0	3.3	3.6	V
V_{IH}	High-level input voltage	0.7 x $V_{VDDCORE}$		$V_{VDDCORE}$ + 0.3	V
V_{IL}	Low-level input voltage	-0.3		0.3 x V_{DDCORE}	V
V_{IHST}	High-level input voltage (Schmitt trigger)	1.625		1.825	V
V_{ILST}	Low-level input voltage (Schmitt trigger)	1.075		1.225	V
V_{HYS}	Hysteresis input voltage (Schmitt trigger)	0.400		0.750	V
V_{OH}	High-level output voltage (drive = 0.3 mA)	$V_{VDDCORE}$ - 0.1			V
V_{OL}	Low-level output voltage (drive = 0.3 mA)			V_{SS} + 0.1	V
I_{LEAK}	Input leakage current		90		nA
O_{LEAK}	Output leakage current		100		nA
FANIN	Pad capacitance		6		pF
I_{PD}	Internal pull-down current	99		429	mA
I_{PU}	Internal pull-up current	130		352	mA

Note: 1. In CMOS, the behavior of a cell is independent of its load, as loads are purely capacitive.

5V I/O Characteristics

Table 60. 5V Power Supply

Symbol	Parameter	Min	Typ	Max	Units
V_{VDDIO}	Supply voltage	$V_{VDDCORE}$		5.5	V
V_{IH}	High-level input voltage	2.0		V_{VDDIO} + 0.3	V
V_{IL}	Low-level input voltage	-0.3		0.8	V
V_{IHST}	High-level input voltage (Schmitt trigger)	1.675		1.725	V
V_{ILST}	Low-level input voltage (Schmitt trigger)	1.025		1.125	V
V_{HYS}	Hysteresis input voltage (Schmitt trigger)	0.550		0.700	V
V_{OH}	High-level output voltage (drive = pin output current)	V_{VDDIO} - 0.4			V
V_{OL}	Low-level output voltage (drive = pin output current)			0.4	V
I_{LEAK}	Input leakage current		90		nA
O_{LEAK}	Output leakage current		100		nA
FANIN	Pad capacitance		6		pF

Note: 1. In CMOS, the behavior of a cell is independent of its load, as loads are purely capacitive.

Analog Characteristics

Table 61. Analog Power Supply

Symbol	Parameter	Min	Typ	Max	Units
V_{VDDANA}	Supply voltage	3.0		3.6	V
V_{ana}	Analog input voltage	V_{SS}		V_{VDDANA}	V
V_{REFP}	Positive analog voltage reference	2.4		V_{VDDANA}	V
I_{LEAK}	Input leakage current	-100		+100	nA
FANIN	Pad capacitance	4		8	pF

ADC Characteristics

Table 62. ADC Characteristics (Initial conditions: V_{VDDANA} = 3.3V +/-10%, V_{REFP} = V_{VDDANA} T = 25°C)

Parameter	Condition	Min	Typ	Max	Unit
Resolution			10		bit
Differential Non-Linearity	250 kHz			±1	lsb
	500 kHz			±2	
	700 kHz			±4	
Integral Non-Linearity	250 kHz			±2.5	lsb
	500 kHz			±3	
	700 kHz			±5	
Zero error (offset)				±2	lsb
Full scale error				±4	lsb
VREF input resistance	@ 25°C	12	18	24	kΩ
Conversion time	11 ADC_clk	15.7			µs
ADC Clock frequency	@ 50% duty cycle		250		kHz
Sampling frequency				63.6	kHz
Startup time			2.0	4.0	µs
Input capacitance (including sample and hold)	Pad selected		107		pF
DC power dissipation		0.7	10	13	mW
Standby power dissipation				100	µW
Operating supply voltage		3.0	3.3	3.6	V

Packaging Information

Thermal Data

The heat transfer between the top surface of the die and the surrounding ambient air can be characterized by the following equation:

$$T_J - T_A = P \times \theta_{JA}$$

where:

P = Device operating power (in W)

T_J = Temperature of a junction on the device (in °C)

T_A = Temperature of the surrounding ambient air (in °C)

θ_{JA} = Package thermal resistance i.e. between junction and ambient air (in °C/W)

The package thermal resistance θ_{JA} for the 176-pin LQFP package is 21°C/W.

Package Drawing

Figure 106. Mechanical Package Drawing of 176-pin LQFP

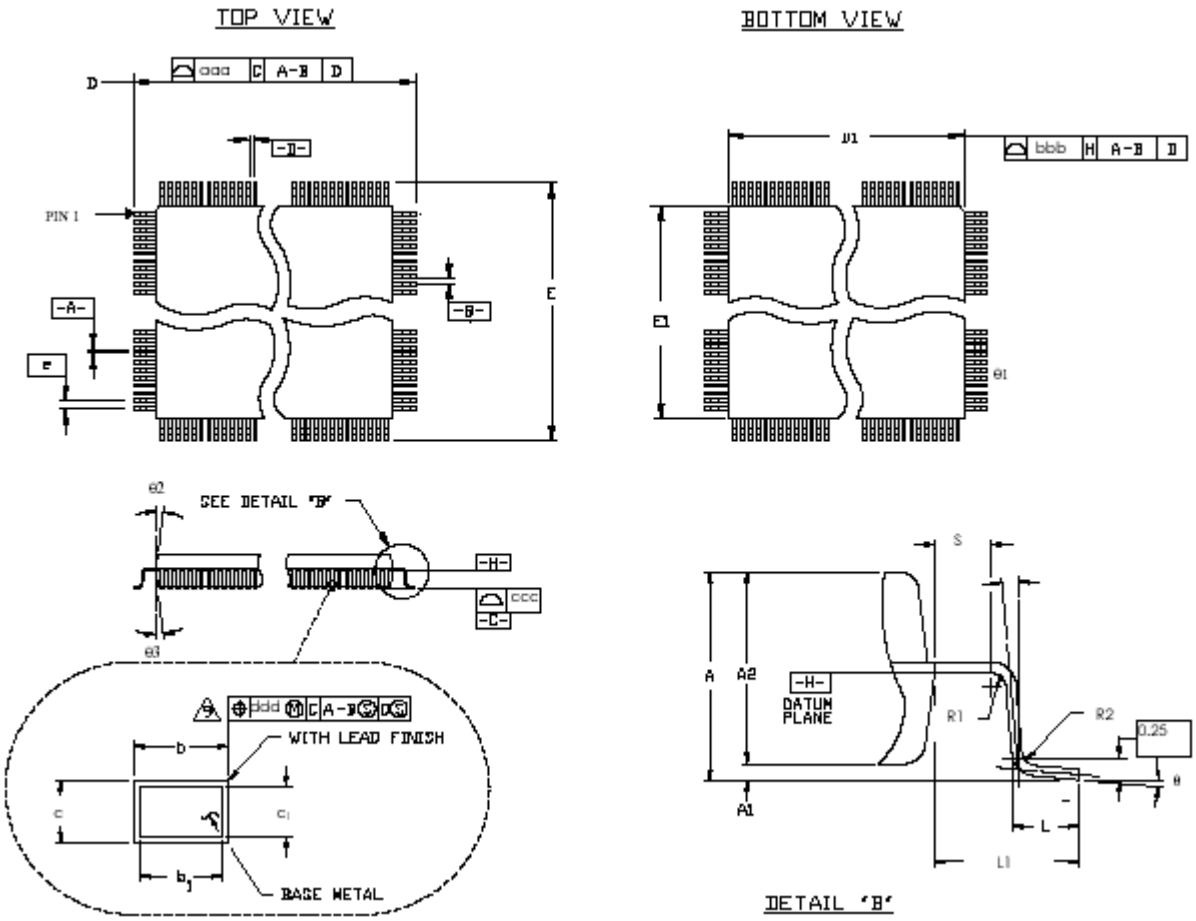


Table 63. Package Dimensions in mm

Symbol	Min	Nom	Max
c	0.09		0.20
c1	0.09		0.16
L	0.45	0.6	0.75
L1	1.00 REF		
R2	0.08		0.2
R1	0.08		
S	0.2		
q	0°	3.5°	7°
θ1	0°		
θ2	11°	12°	13°
θ3	11°	12°	13°
A			1.6
A1	0.05		0.15
A2	1.35	1.4	1.45
Tolerances of Form and Position			
aaa		0.2	
bbb		0.2	

Table 64. Lead Count Dimensions (mm)

Pin Count	D/E BSC	D1/E1 BSC	b			b1			e BSC	ccc	ddd
			Min	Nom	Max	Min	Nom	Max			
176	26.0	24.0	0.17	0.20	0.27	0.17	0.20	0.23	0.50	0.10	0.08

Table 65. Device and 176-lead LQFP Package Maximum Weight

1900	mg

Soldering Profile

Table 66 below gives the recommended soldering profile from J-STD-20.

Table 66. Soldering Profile

	Convection or IR/Convection	VPR
Average Ramp-up Rate (183°C to Peak)	3°C/sec. max	10°C/sec.
Preheat Temperature 125°C ±25°C	120 sec. max	
Temperature Maintained Above 183°C	60 sec. to 150 sec.	
Time within 5°C of Actual Peak Temperature	10 sec. to 20 sec.	60 sec.
Peak Temperature Range	220 +5/-0°C or 235 +5/-0°C	215 to 219°C or 235 +5/-0°C
Ramp-down Rate	6°C/sec.	10°C/sec.
Time 25°C to Peak Temperature	6 min. max	

Small packages may be subject to higher temperatures if they are reflowed in boards with larger components. In this case, small packages may have to withstand temperatures of up to 235°C, not 220°C (IR reflow).

Recommended package reflow conditions depend on package thickness and volume. See Table 67 below.

Table 67. Recommended Package Reflow Conditions (1)(2)(3)

Parameter	Temperature
Convection	235 +5/-0°C
VPR	215 to 219°C
IR/Convection	235 +5/-0°C

Notes:
1. The packages are qualified by Atmel by using IR reflow conditions, not convection or VPR.
2. By default, the package level 1 is qualified at 220°C (unless 235°C is stipulated).
3. The body temperature is the most important parameter but other profile parameters such as total exposure time to hot temperature or heating rate may also influence component reliability.

A maximum of three reflow passes is allowed per component.

Environmental Specifications

Operational Temperature

The operational temperature range of the AT91SAM7A2 embedded system is -40°C to +85°C.

Storage Temperature

The storage temperature range of the AT91SAM7A2 embedded system chip is -50°C to +150°C.

Table of Contents

Atmel Corporation

2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

Regional Headquarters

Europe
Atmel Sarl
Route des Arsenaux 41
Case Postale 80
CH-1705 Fribourg
Switzerland
Tel: (41) 26-426-5555
Fax: (41) 26-426-5500

Asia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Atmel Operations

Memory
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

Microcontrollers
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

La Chantrerie
BP 70602
44306 Nantes Cedex 3, France
Tel: (33) 2-40-18-18-18
Fax: (33) 2-40-18-19-60

ASIC/ASSP/Smart Cards
Zone Industrielle
13106 Rousset Cedex, France
Tel: (33) 4-42-53-60-00
Fax: (33) 4-42-53-60-01

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Scottish Enterprise Technology Park
Maxwell Building
East Kilbride G75 0QR, Scotland
Tel: (44) 1355-803-000
Fax: (44) 1355-242-743

RF/Automotive
Theresienstrasse 2
Postfach 3535
74025 Heilbronn, Germany
Tel: (49) 71-31-67-0
Fax: (49) 71-31-67-2340

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Biometrics/Imaging/Hi-Rel MPU/
High Speed Converters/RF Datacom
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
Tel: (33) 4-76-58-30-00
Fax: (33) 4-76-58-34-80

Literature Requests
www.atmel.com/literature

6021A–ATARM–07/04 0M