AVR 8-bit Microcontrollers

ATtiny416 / ATtiny417 / ATtiny814 / ATtiny816 / ATtiny817

DATASHEET ADVANCE INFORMATION

Introduction

The Atmel® ATtiny416/417/814/816/817 is a series of microcontrollers using the 8-bit AVR® processor with hardware multiplier, running at up to 20MHz and with up to 8KB Flash, 512 bytes of SRAM and 128 bytes of EEPROM in a 14-, 20- and 24-pin package. The series uses the latest technologies from Atmel with a flexible and low power architecture including Event System and SleepWalking, accurate analog features and advanced peripherals. Capacitive touch interfaces with proximity sensing and driven shield are supported with the integrated QTouch® peripheral touch controller.

Features

- CPU
  - Atmel® AVR® 8-bit CPU
  - Running at up to 20MHz
  - Single cycle I/O access
  - Two-level interrupt controller
  - Two-cycle hardware multiplier
- Memories
  - 4/8KB In-system self-programmable Flash memory
  - 128B EEPROM
  - 256/512B SRAM
- System
  - Power-on Reset (POR)
  - Brown-out Detection (BOD)
  - Internal and external clock options:
    - 16/20MHz low power RC oscillator with:
      - ±3% accuracy over full temp and voltage range
      - ±1.5% drift over limited temp and full voltage range
    - 32.768kHz Ultra Low Power (ULP) internal RC oscillator with ±10% accuracy, ±2% calibration step size
    - 32.768kHz external crystal oscillator

        - External clock input
    - Single pin programming and debugging interface (UPDI)
    - Three sleep modes:
        - Idle with all peripherals running and mode for immediate wake up time
        - Standby
            - Configurable operation of selected peripherals
            - SleepWalking peripherals
        - Power Down with limited wake-up functionality
- Peripherals
    - One 16-bit Timer/Counter type A with dedicated period register, 3 compare channels (TCA)
    - One 16-bit Timer/Counter type B with input capture (TCB)
    - One 12-bit Timer/Counter type D optimized for control applications (TCD)
    - 16-bit Real Time Counter (RTC) running from external crystal or internal RC oscillator
    - One USART with fractional baud rate generator, autobaud, and start-of-frame detection
    - Master/slave Serial Peripheral Interface (SPI)
    - Master/slave I^2C with dual address match
        - Standard mode (Sm, 100kHz)
        - Fast mode (Fm, 400kHz)
        - Fast mode plus (Fm+, 1MHz)
    - Configurable Custom Logic (CCL) with two programmable Lookup Tables (LUT)
    - Analog Comparator (AC) with fast propagation delay
    - 10-bit 150ksps Analog to Digital Converter (ADC)
    - 8-bit Digital to Analog Converter (DAC)
    - Five selectable internal voltage references: 0.55V, 1.1V, 1.5V, 2.5V and 4.3V
    - Automated CRC memory scan
    - Window Watchdog Timer (WDT) with separate on-chip oscillator
    - Peripheral Touch Controller (PTC)[(1)]
        - Capacitive touch buttons, sliders and wheels
        - Wake-up on touch
        - Driven shield for improved moisture and noise handling performance
        - Six self-capacitance and nine mutual-capacitance channels
    - External interrupt on all general purpose pins
- I/O and Packages:
    - 12 to 22 programmable I/O lines
    - 14-pin SOIC150
    - 20-pin QFN 3x3 and SOIC300
    - 24-pin QFN 4x4
- Temperature Ranges: -40°C to 105°C within specification
    - Some device variants: -40°C to 125°C with reduced spec
- Speed Grades:
    - 0-5MHz @ 1.8V – 5.5V
    - 0-10MHz @ 2.7V – 5.5V
    - 0-20MHz @ 4.5V – 5.5V

Note:

1. Only available in devices with 8KB Flash.

Table of Contents

1. Configuration Summary

Table 1-1. Configuration Summary

	ATtiny417 / ATtiny817	ATtiny416 / ATtiny816	ATtiny814
Pins	24	20	14
SRAM	256/512B	256/512B	512B
Flash	4/8KB	4/8KB	8KB
EEPROM	128B	128B	128B
Max. frequency	20MHz	20MHz	20MHz
16-bit Timer/Counter type A (TCA)	1	1	1
16-bit Timer/Counter type B (TCB)	1	1	1
12-bit Timer/Counter type D (TCD)	1	1	1
Real Time Counter (RTC)	1	1	1
USART	1	1	1
SPI	1	1	1
TWI (I^2C)	1	1	1
ADC	1	1	1
ADC channels	12	12	10
DAC	1	1	1
AC	1	1	1
AC inputs	2p/2n	2p/2n	1p/1n
Peripheral Touch Controller (PTC)(1)	No / Yes(2)	No / Yes(2)	Yes(2)
PTC number of self-capacitance channels(1)	6	6	6
PTC number of mutual-capacitance channels(1)	9	9	9
Custom Logic	1	1	1
Window Watchdog	1	1	1
Event System channels	6	6	6
General purpose I/O	22	18	12
External interrupts	22	18	12
CRCSCAN	1	1	1

Note:

1. PTC is only available in devices with 8KB Flash (ATtiny817, ATtiny816 and ATtiny814).
2. The PTC serves as input to the ADC.

2. Ordering Information

2.1. ATtiny81x

Table 2-1. ATtiny817 Ordering Codes

Ordering Code[1]	Flash	Package Type	Leads	Power Supply	Operational Range	Carrier Type
ATtiny817-MNRES[2]	8KB	QFN 4x4	24	1.8V - 5.5V	Industrial (-40°C +105°C)	Tape & Reel
ATtiny817-MNR	8KB	QFN 4x4	24	1.8V - 5.5V	Industrial (-40°C +105°C)	Tape & Reel
ATtiny817-MFR	8KB	QFN 4x4	24	1.8V - 5.5V	Industrial (-40°C +125°C)	Tape & Reel

Table 2-2. ATtiny816 Ordering Codes

Ordering Code[1]	Flash	Package Type	Leads	Power Supply	Operational Range	Carrier Type
ATtiny816-MNRES[2]	8KB	QFN 3x3	20	1.8V - 5.5V	Industrial (-40°C +105°C)	Tape & Reel
ATtiny816-MNR	8KB	QFN 3x3	20	1.8V - 5.5V	Industrial (-40°C +105°C)	Tape & Reel
ATtiny816-MFR	8KB	QFN 3x3	20	1.8V - 5.5V	Industrial (-40°C +125°C)	Tape & Reel
ATtiny816-SNR	8KB	SOIC300	20	1.8V - 5.5V	Industrial (-40°C +105°C)	Tape & Reel
ATtiny816-SFR	8KB	SOIC300	20	1.8V - 5.5V	Industrial (-40°C +125°C)	Tape & Reel

Table 2-3. ATtiny814 Ordering Codes

Ordering Code[1]	Flash	Package Type	Leads	Power Supply	Operational Range	Carrier Type
ATtiny814-SSNRES[2]	8KB	SOIC150	14	1.8V - 5.5V	Industrial (-40°C +105°C)	Tape & Reel
ATtiny814-SSNR	8KB	SOIC150	14	1.8V - 5.5V	Industrial (-40°C +105°C)	Tape & Reel
ATtiny814-SSFR	8KB	SOIC150	14	1.8V - 5.5V	Industrial (-40°C +125°C)	Tape & Reel

Note:

1. Pb-free packaging complies to the European Directive for Restriction of Hazardous Substances (RoHS directive). Also Halide free and fully Green.
2. Engineering samples.

2.2. ATtiny41x

Table 2-4. ATtiny417 Ordering Codes

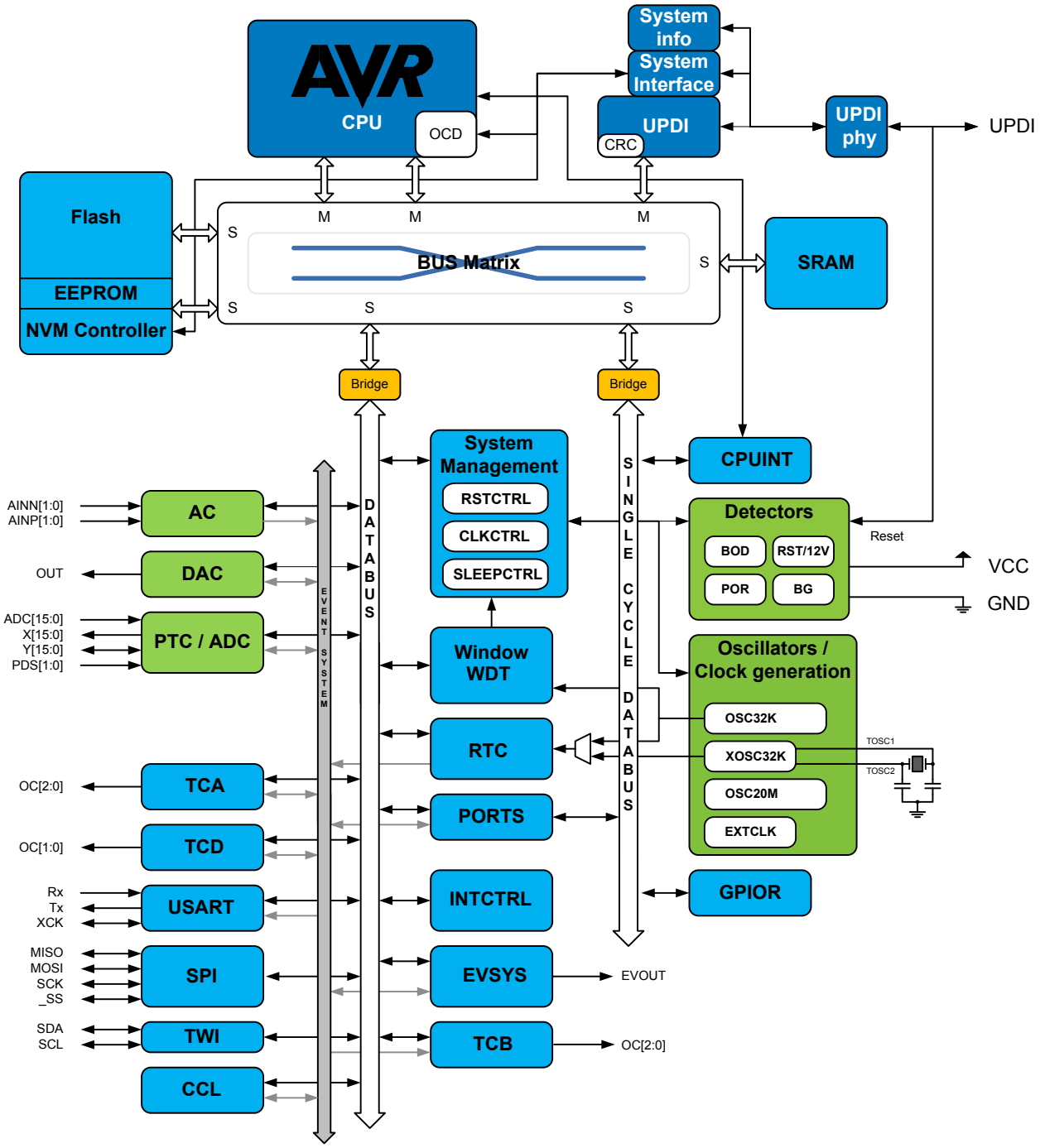
Ordering Code[1]	Flash	Package Type	Leads	Power Supply	Operational Range	Carrier Type
ATtiny417-MNR	4KB	QFN 4x4	24	1.8V - 5.5V	Industrial (-40°C +105°C)	Tape & Reel
ATtiny417-MFR	4KB	QFN 4x4	24	1.8V - 5.5V	Industrial (-40°C +125°C)	Tape & Reel

Table 2-5. ATtiny416 Ordering Codes

Ordering Code[1]	Flash	Package Type	Leads	Power Supply	Operational Range	Carrier Type
ATtiny416-MNR	4KB	QFN 3x3	20	1.8V - 5.5V	Industrial (-40°C +105°C)	Tape & Reel
ATtiny416-MFR	4KB	QFN 3x3	20	1.8V - 5.5V	Industrial (-40°C +125°C)	Tape & Reel
ATtiny416-SNR	4KB	SOIC300	20	1.8V - 5.5V	Industrial (-40°C +105°C)	Tape & Reel
ATtiny416-SFR	4KB	SOIC300	20	1.8V - 5.5V	Industrial (-40°C +125°C)	Tape & Reel

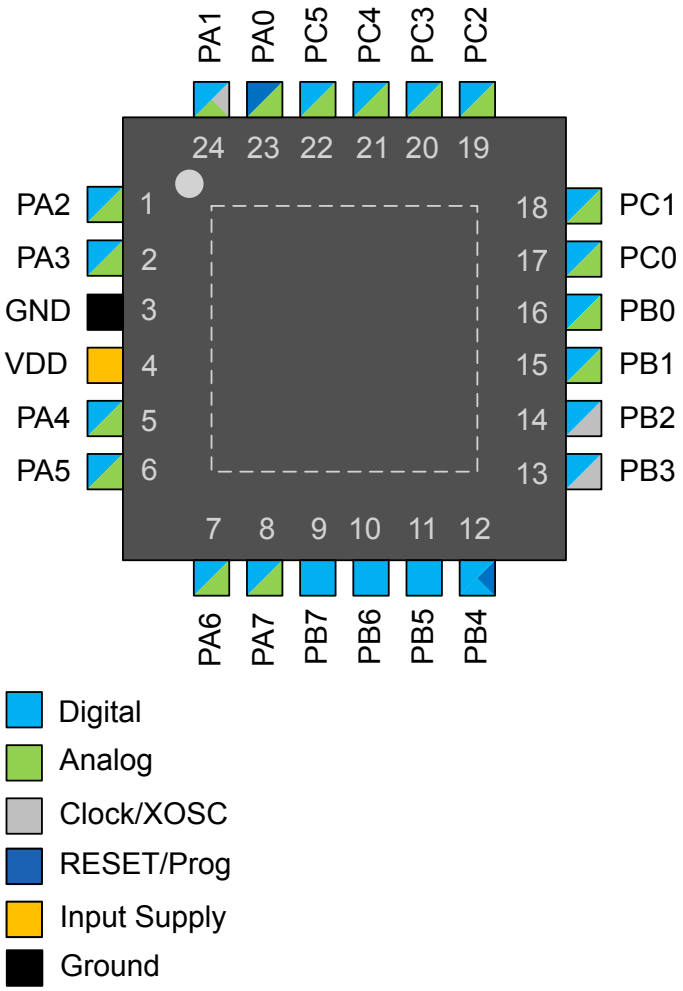
1. Pb-free packaging complies to the European Directive for Restriction of Hazardous Substances (RoHS directive). Also Halide free and fully Green.

3. Block Diagram

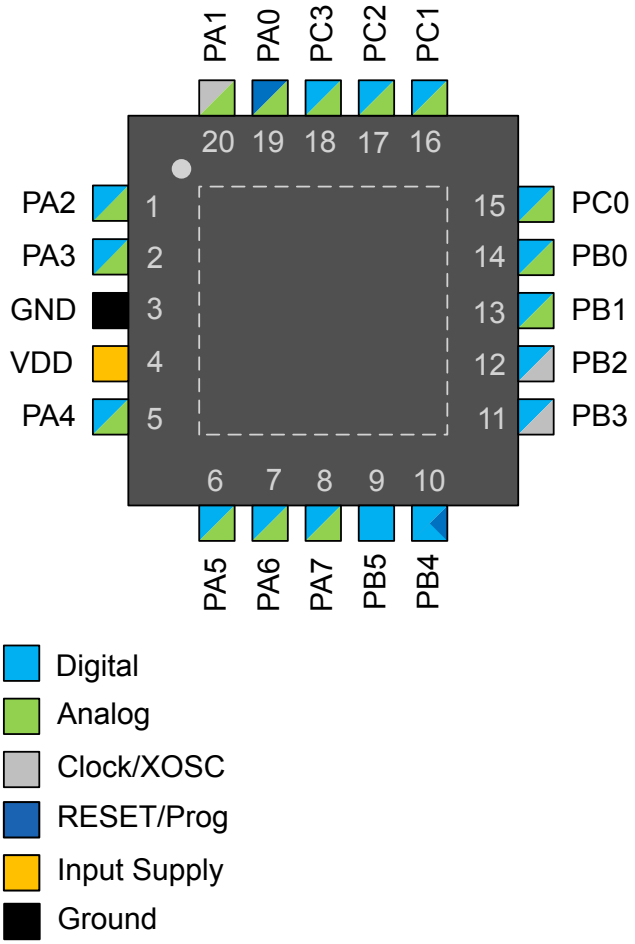


4. Pinout

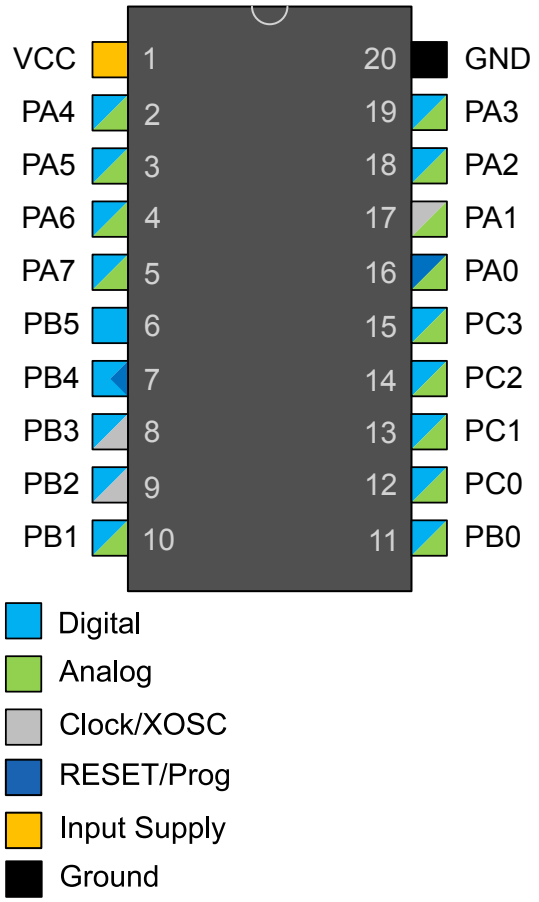
4.1. 24-pin QFN



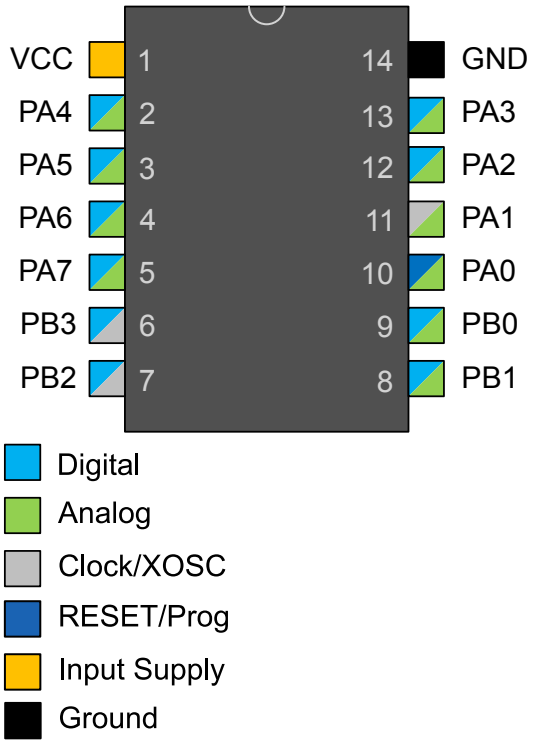
4.2. 20-pin QFN


PA1
PA0
PC3
PC2
PC1
20
19
18
17
16
PA2
1
PA3
2
GND
3
VDD
4
PA4
5
15
PC0
14
PB0
13
PB1
12
PB2
11
PB3
6
7
8
9
10
PA5
PA6
PA7
PB5
PB4
Digital
Analog
Clock/XOSC
RESET/Prog
Input Supply
Ground

4.3. 20-pin SOIC



4.4. 14-pin SOIC



5. I/O Multiplexing and Considerations

5.1. Multiplexed Signals

Table 5-1. PORT Function Multiplexing

QFN 24-pin	QFN 20-pin	SOIC 20-pin	SOIC 14-pin	Pin Name (1)	Other/ Special	EXTINTn (2)	ADC0	ADC1	PTC (3)	AC0	DAC	USART0	SPI0	TWI0	TCA0		TCD0	CCL
23	19	16	10	PA0	$\overline{\text{RESET}}$ UPDI	S	AIN0											LUT0-IN0
24	20	17	11	PA1		S	AIN1					*TXD*	MOSI	*SDA*				LUT0-IN1
1	1	18	12	PA2	EVOUT0	A	AIN2					*RxD*	MISO	*SCL*				LUT0-IN2
2	2	19	13	PA3	CLKI	S	AIN3					*XCK*	SCK		WO3			
3	3	20	14	GND														
4	4	1	1	VDD														
5	5	2	2	PA4		S	AIN4	AIN0	X0/Y0			*XDIR*	$\overline{\text{SS}}$				WOA	LUT0-OUT
6	6	3	3	PA5		S	AIN5	AIN1	X1/Y1	ACOUT					WO4	W0	WOB	
7	7	4	4	PA6		A	AIN6	AIN2	X2/Y2	N0	OUT				WO5			
8	8	5	5	PA7		S	ADC7	ADC3	X3/Y3	P0								LUT1-OUT
9				PB7		S		ADC4										
10				PB6		A		ADC5										
11	9	6		PB5	CLKOUT	S	ADC8			N1					*WO2*			
12	10	7		PB4		S	ADC9		DS1	P1					*WO1*			*LUT0-OUT*
13	11	8	6	PB3	TOSC1 EVOUT1	S						RxD			*WO0*			
14	12	9	7	PB2	TOSC2	A			DS0			TxD			WO2			
15	13	10	8	PB1		S	AIN10		X4/Y4			XCK		SDA	WO1			
16	14	11	9	PB0		S	AIN11		X5/Y5			XDIR		SCL	WO0			
17	15	12		PC0		S		ADC6					*SCK*			*W0*	WOC	
18	16	13		PC1		S		ADC7					*MISO*				WOD	*LUT1-OUT*
19	17	14		PC2	EVOUT2	A		ADC8					*MOSI*					
20	18	15		PC3		S		ADC9					*$\overline{\text{SS}}$*		*WO3*			LUT1-IN0
21				PC4		S		10							*WO4*			LUT1-IN1
22				PC5	*RESET*	S		11							*WO5*			LUT1-IN2

Note:

1. Pins PAn are configured by PORT instance A, pins PBn are configured by PORT B etc.
2. S: Detection = Synchronous and limited asynchronous.
   A: Detection = Synchronous and full asynchronous.
3. PTC is only available in devices with 8KB Flash (ATtiny817, ATtiny816, ATtiny814). Every PTC line can be configured as X-line or Y-line.
4. Default pin assignment of signals are in regular font. Signals on alternative pin locations are in *italic*.

6. Memories

6.1. Overview

The main memories are SRAM data memory, EEPROM data memory, and Flash program memory. In addition, the peripheral registers are located in the I/O memory space.

Table 6-1. Physical Properties of EEPROM

Property	ATtiny817/816/814	ATtiny417/416
Size	128B	128B
Page size	32B	32B
Number of pages	4	4
Start address	0x1400	0x1400

Table 6-2. Physical Properties of SRAM

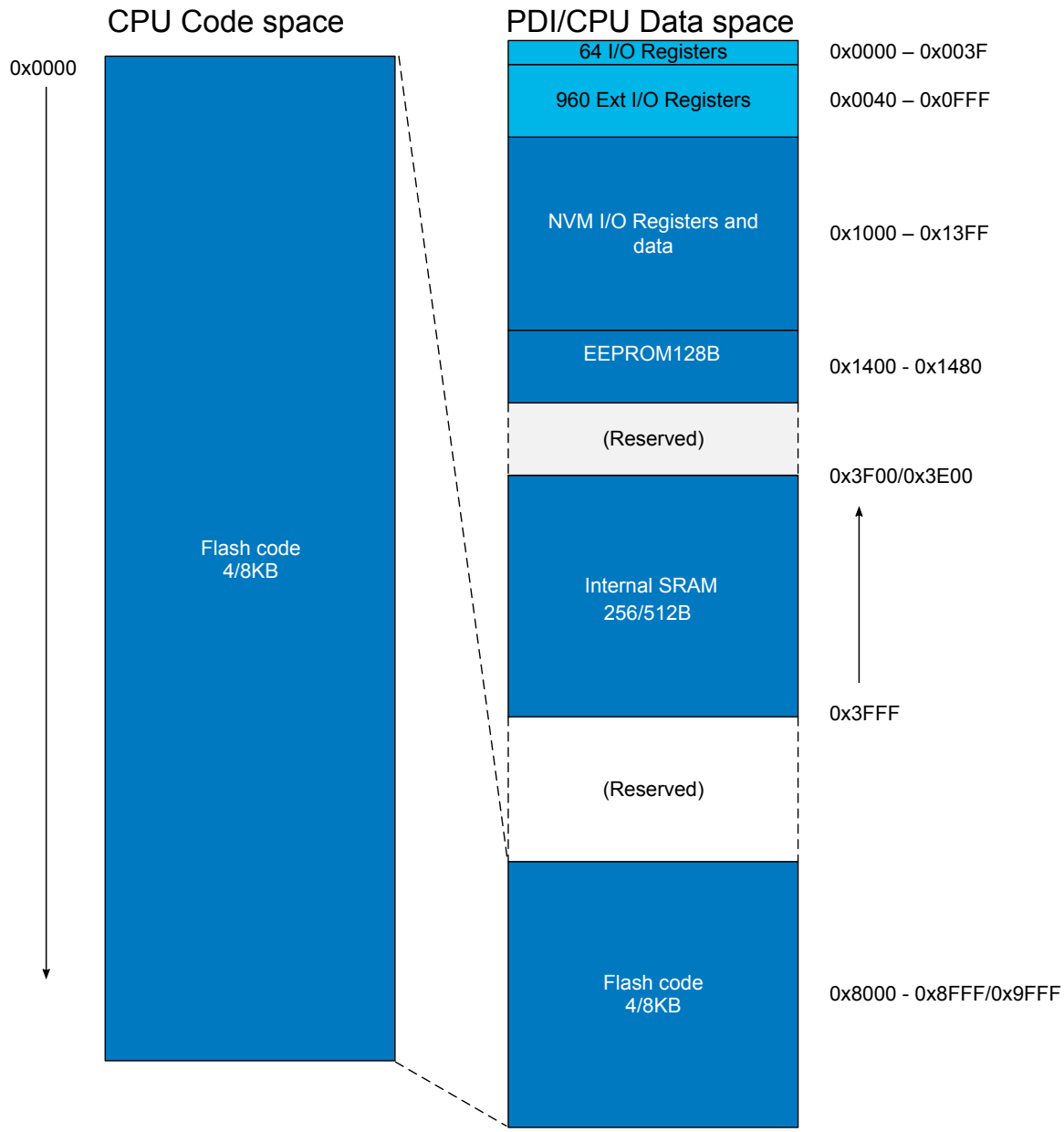
Property	ATtiny817/816/814	ATtiny417/416
Size	512B	256B
Start address	0x3E00	0x3F00

Table 6-3. Physical Properties of Flash Memory

Property	ATtiny817/816/814	ATtiny417/416
Size	8KB	4KB
Page size	64B	64B
Number of pages	128	64
Start address	0x8000	0x8000

6.2. Memory Map

Figure 6-1. Memory Map



6.3. In-System Reprogrammable Flash Program Memory

The ATtiny416/417/814/816/817 contains 4/8KB On-Chip In-System Reprogrammable Flash memory for program storage. Since all AVR instructions are 16 or 32 bits wide, the Flash is organized as 4K x 16. For write protection, the Flash Program memory space can be divided into three sections: Boot Loader section, Application code section and Application data section, with restricted access rights among them.

The program counter is 12/11 bits wide to address the whole program memory. The procedure for writing Flash memory is described in detail in the documentation of the Non-Volatile Memory Controller (NVMCTRL) peripheral.

The entire Flash memory is mapped in the memory space and is accessible with normal LD/ST instructions as well as the LPM instruction. For LD/ST instructions, the Flash is mapped from address 0x8000. For the LPM instruction, the Flash start address is 0x0000.

The ATtiny416/417/814/816/817 also has a CRC module that is a master on the bus. If the CRC is configured to run in the background it will read the Flash memory and can modify the program timing.

Related Links

Configuration Summary on page 11
NVMCTRL - Non Volatile Memory Controller on page 52
Flash Write Protection on page 57

6.4. SRAM Data Memory

The 512B SRAM is used for data storage and stack.

Related Links

AVR CPU on page 40
Stack and Stack Pointer on page 44

6.5. EEPROM Data Memory

The ATtiny416/417/814/816/817 contains 128 bytes of data EEPROM memory, see Memory Map. The EEPROM memory supports single byte read and write. The EEPROM is controlled by the Non-Volatile Memory Controller (NVMCTRL).

Preventing EEPROM Corruption

During periods of low V_{DD}, the EEPROM data can be corrupted because the supply voltage is too low for the CPU and the EEPROM to operate properly. These issues are the same as for board level systems using EEPROM, and the same design solutions should be applied.

An EEPROM data corruption can be caused by two situations when the voltage is too low: First, a regular write sequence to the EEPROM requires a minimum voltage to operate correctly. Also, the CPU itself can execute instructions incorrectly when the supply voltage is too low.

EEPROM data corruption can easily be avoided by these measures:

Keep the AVR $\overline{\text{RESET}}$ active (low) during periods of insufficient power supply voltage. This is done by enabling the internal Brown-Out Detector (BOD).

If the detection levels of the internal BOD does not match the required detection level, an external low-V_{DD}-reset protection circuit can be used. If a Reset occurs while a write operation is ongoing, the write operation will be completed, provided that the power supply voltage is sufficient.

Related Links

Memory Map on page 21
NVMCTRL - Non Volatile Memory Controller on page 52
BOD - Brownout Detector on page 155

6.6. User Row

In addition to the EEPROM, the ATtiny416/417/814/816/817 contains one extra page of EEPROM memory that can be used for firmware settings, the User Row (USERROW). This memory supports single byte read and write as the normal EEPROM. The CPU can write this memory as normal EEPROM and the UPDI can write it as a normal EEPROM memory if the part is unlocked. The User Row can also be written by the UPDI when the part is locked with a special key. USERROW is not affected by a chip erase.

Related Links

Memory Map on page 21
NVMCTRL - Non Volatile Memory Controller on page 52
UPDI - Unified Program Debug Interface on page 490

6.7. I/O Memory

All ATtiny416/417/814/816/817 I/Os and peripherals are located in the I/O space. All I/O locations can be accessed by the LD/LDS/LDD and ST/STS/STD instructions, transferring data between the 32 general purpose working registers and the I/O space.

I/O Registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions. Refer to the Instruction Set section for more details.

The I/O specific commands IN and OUT can access the I/O addresses 0x00 - 0x3F.

The I/O address range from 0x00 to 0x3F can be accessed in single cycle using IN and OUT instructions. For the Extended I/O space from 0x0040 - 0x0FFF, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.

Some of the Status Flags are cleared by writing a '1' to them. Note that on ATtiny416/417/814/816/817 devices, the CBI and SBI instructions will only operate on the specified bit, and can therefore be used on registers containing such Status Flags. The CBI and SBI instructions work with registers 0x00 - 0x1F only.

General Purpose I/O Registers

The ATtiny416/417/814/816/817 devices provide four General Purpose I/O Registers. These registers can be used for storing any information, and they are particularly useful for storing global variables and Status Flags.

Related Links

Memory Map on page 21
Instruction Set Summary on page 525

6.8. FUSES - Configuration and User Fuses

6.8.1. Signature Row Summary

Offset	Name	Bit Pos.								
0x00	SIGROW_DEVICEID0	7:0	DEVICEID[7:0]							
0x01	SIGROW_DEVICEID1	7:0	DEVICEID[7:0]							
0x02	SIGROW_DEVICEID2	7:0	DEVICEID[7:0]							
0x03	SIGROW_SERNUM0	7:0	SERNUM[7:0]							
0x04	SIGROW_SERNUM1	7:0	SERNUM[7:0]							
0x05	SIGROW_SERNUM2	7:0	SERNUM[7:0]							
0x06	SIGROW_SERNUM3	7:0	SERNUM[7:0]							
0x07	SIGROW_SERNUM4	7:0	SERNUM[7:0]							
0x08	SIGROW_SERNUM5	7:0	SERNUM[7:0]							
0x09	SIGROW_SERNUM6	7:0	SERNUM[7:0]							
0x0A	SIGROW_SERNUM7	7:0	SERNUM[7:0]							
0x0B	SIGROW_SERNUM8	7:0	SERNUM[7:0]							
0x0C	SIGROW_SERNUM9	7:0	SERNUM[7:0]							
0x0D ... 0x1F	Reserved									
0x20	SIGROW_TEMPSENSE0	7:0	TEMPSENSE7	TEMPSENSE6	TEMPSENSE5	TEMPSENSE4	TEMPSENSE3	TEMPSENSE2	TEMPSENSE1	TEMPSENSE0
0x21	SIGROW_TEMPSENSE1	7:0	TEMPSENSE7	TEMPSENSE6	TEMPSENSE5	TEMPSENSE4	TEMPSENSE3	TEMPSENSE2	TEMPSENSE1	TEMPSENSE0

6.8.2. Signature Row Description

6.8.2.1. Device ID n

Name: SIGROW_DEVICEIDn
Offset: 0x00 + n*0x01 [n=0..2]
Reset: [Device ID]
Property: -

Bit	7	6	5	4	3	2	1	0
	DEVICEID[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	-	-	-	-	-	-	-	-

Bits 7:0 – DEVICEID[7:0]: Byte n of the Device ID

6.8.2.2. Serial Number Byte n

Name: SIGROW_SERNUMn
Offset: 0x03 + n*0x01 [n=0..9]
Reset: [device serial number]
Property: -

Bit	7	6	5	4	3	2	1	0
	SERNUM[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	x	x	x	x	x	x	x	x

Bits 7:0 – SERNUM[7:0]: Serial Number n [n=0..9]
Each device has an individual serial number, representing a unique ID. This can be used to identify a specific device in the field. The serial number consists of ten bytes..

6.8.2.3. Temperature Sensor Calibration n

Name: SIGROW_TEMPSENSEn
Offset: 0x20 + n*0x01 [n=0..1]
Reset: [Temperature sensor calibration value]
Property: -

Bit	7	6	5	4	3	2	1	0
	TEMPSENSE7	TEMPSENSE6	TEMPSENSE5	TEMPSENSE4	TEMPSENSE3	TEMPSENSE2	TEMPSENSE1	TEMPSENSE0
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	-

Bits 7:0 – TEMPSENSEn: Temperature Sensor Calibration Byte n [n=0..1]
TBD.

6.8.3. Fuse Summary

Offset	Name	Bit Pos.								
0x00	FUSE_WDTCFG	7:0	WINDOW[3:0]				PERIOD[3:0]			
0x01	FUSE_BODCFG	7:0	LVL[2:0]			SAMPLFREQ	ACTIVE[1:0]		SLEEP[1:0]	
0x02	FUSE_OSCCFG	7:0	OSCLOCK						FREQSEL[1:0]	
0x03	Reserved									
0x04	TCD0CFG	7:0	CMPD	CMPC	CMPB	CMPA	CMPD	CMPC	CMPB	CMPA
0x05	FUSE_SYSCFG0	7:0	CRCBOOTDIS	CRCAPPDIS			RSTPINCFG[1:0]			EESAVE
0x06	FUSE_SYSCFG1	7:0						SUT[2:0]		
0x07	FUSE_APPEND	7:0	APPEND[7:0]							
0x08	FUSE_BOOTEND	7:0	BOOTEND[7:0]							
0x09	Reserved									
0x0A	FUSE_LOCKBIT	7:0	LOCKBIT[7:0]							

6.8.4. Fuse Description

6.8.4.1. Watchdog Configuration

Name: FUSE_WDTCFG
Offset: 0x00
Reset: -
Property: -

Bit	7	6	5	4	3	2	1	0
	WINDOW[3:0]				PERIOD[3:0]			
Access	R	R	R	R	R	R	R	R
Reset	-	-	-	-	-	-	-	-

Bits 7:4 – WINDOW[3:0]: Watchdog Window Timeout Period
This value is loaded into the Watchdog Control A register (WDT_CTRLA.WINDOW) at the end of the startup sequence.

Bits 3:0 – PERIOD[3:0]: Watchdog Timeout Period
This value is loaded into the Watchdog Control A register (WDT_CTRLA.PERIOD) at the end of the startup sequence.

6.8.4.2. BOD Configuration

Name: FUSE_BODCFG
Offset: 0x01
Reset: -
Property: -

Bit	7	6	5	4	3	2	1	0
	LVL[2:0]			SAMPLFREQ	ACTIVE[1:0]		SLEEP[1:0]	
Access	R	R	R	R	R	R	R	R
Reset	-	-	-	-	-	-	-	-

Bits 7:5 – LVL[2:0]: BOD Level
This value is loaded into the BOD Control B register (BOD_CTRLB.LVL) at the end of the startup sequence.

Bit 4 – SAMPLFREQ: BOD Sample Frequency
This value is loaded into the BOD Control A register (BOD_CTRLA.SAMPLEFREQ) at the end of the startup sequence.

Value	Description
0x0	Sample frequency is 1kHz
0x1	Sample frequency is 125Hz

Bits 3:2 – ACTIVE[1:0]: BOD Operation Mode in Active and Idle
This value is loaded into the BOD Control A register (BOD_CTRLA.ACTIVE) at the end of the startup sequence.

Value	Description
0x0	Disabled
0x1	Enabled
0x2	Sampled
0x3	Enabled with wake-up halted until BOD is ready

Bits 1:0 – SLEEP[1:0]: BOD Operation Mode in Sleep
This value is loaded into the BOD Control A register (BOD_CTRLA.SLEEP) at the end of the startup sequence.

Value	Description
0x0	Disabled
0x1	Enabled
0x2	Sampled
0x3	Reserved

6.8.4.3. Oscillator Configuration

Name: FUSE_OSCCFG
Offset: 0x02
Reset: -
Property: -

Bit	7	6	5	4	3	2	1	0
	OSCLOCK						FREQSEL[1:0]	
Access	R						R	R
Reset	-						-	-

Bit 7 – OSCLOCK: Oscillator Lock
This fuse bit is written to CLKCTRL_MCLKLOCK.LOCKEN at startup.

Value	Description
0	Calibration registers of the main oscillator are accessible
1	Calibration registers of the main oscillator are locked

Bits 1:0 – FREQSEL[1:0]: Frequency Select
These bits selects the operation frequency of the 16/20MHz internal oscillator (OSC20M), and determines the respective calibration value to be loaded to CLKCTRL_OSC20MCALIBA.CAL20M.

Value	Description
0x1	Run at 16MHz with corresponding factory calibration
0x2	Run at 20MHz with corresponding factory calibration
Other	Reserved

6.8.4.4. Timer Counter Type D Configuration

The bit values of this fuse register are written to the corresponding bits in the TCD.FAULTCTRL register of the TCD0.

Name: TCD0CFG
Offset: 0x04
Reset: -
Property: -

Bit	7	6	5	4	3	2	1	0
	CMPD	CMPC	CMPB	CMPA	CMPD	CMPC	CMPB	CMPA
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	-	0	0	0	-

Bits 4, 5, 6, 7 – CMPA, CMPB, CMPC, CMPD: Compare x Enable

Value	Description
0	Compare x output on Pin is disabled
1	Compare x output on Pin is enabled

Bits 0, 1, 2, 3 – CMPA, CMPB, CMPC, CMPD: Compare x

This bit selects the default state of Compare x after Reset, or when entering debug if FAULTDET is set.

Value	Description
0	Compare x default state is 0
1	Compare x default state is 1

6.8.4.5. System Configuration 0

Name: FUSE_SYSCFG0
Offset: 0x05
Reset: -
Property: -

Bit	7	6	5	4	3	2	1	0
	CRCBOOTDIS	CRCAPPDIS			RSTPINCFG[1:0]			EESAVE
Access	R	R			R	R		R
Reset	-	-			-	-		-

Bit 7 – CRCBOOTDIS: CRC of Boot Section in Reset Disable
See CRC description for more information about the functionality.

Value	Description
0	Boot section undergoing a CRC before Reset releases
1	No CRC of the boot section before Reset releases

Bit 6 – CRCAPPDIS: CRC of Application Code Section in Reset Disable
See CRC description for more information about the functionality.

Value	Description
0	Application code section undergoing a CRC before Reset releases
1	No CRC of the application code section before Reset releases

Bits 3:2 – RSTPINCFG[1:0]: Reset Pin Configuration
These bits select the Reset/UPDI pin configuration.

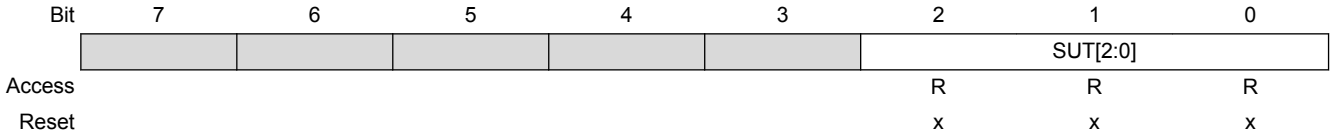
Value	Description
0x0	GPIO
0x1	UPDI
0x2	RESET
0x3	UPDI, with external Reset on alternate location

Bit 0 – EESAVE: EEPROM Save during chip erase
Note: If the device is locked the EEPROM is always erased by a chip erase, regardless of this bit.

Value	Description
0	EEPROM erased during chip erase
1	EEPROM not erased under chip erase

6.8.4.6. System Configuration 1

Name: FUSE_SYSCFG1
Offset: 0x06
Reset: -
Property: -



Bits 2:0 – SUT[2:0]: Start Up Time Setting
These bits selects the start-up time.

Value	Description
0x0	0ms
0x1	1ms
0x2	2ms
0x3	4ms
0x4	8ms
0x5	16ms
0x6	32ms
0x7	64ms
other	Reserved

6.8.4.7. Application Code End

Name: FUSE_APPEND
Offset: 0x07
Reset: -
Property: -

Bit	7	6	5	4	3	2	1	0
	APPEND[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	-

Bits 7:0 – APPEND[7:0]: Application Code Section End
These bits set the end of the application code section in blocks of 256 bytes. The end of the application code section should be set as BOOT size + application code size. A value of 0x00 defines the whole Flash as application code section.
Note: When both FUSE_APPEND and FUSE_BOOTEND are 0x00, the entire Flash is application code section.

6.8.4.8. Boot End

Name: FUSE_BOOTEND
Offset: 0x08
Reset: -
Property: -

Bit	7	6	5	4	3	2	1	0
	BOOTEND[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	-

Bits 7:0 – BOOTEND[7:0]: Boot Section End
These bits set the end of the boot section in blocks of 256 bytes. A value of 0x00 defines the whole Flash as BOOT section.
Note: When both FUSE_APPEND and FUSE_BOOTEND are 0x00, the entire Flash is application code section.

6.8.4.9. Lock Bits

Name: FUSE_LOCKBIT
Offset: 0x0A
Reset: -
Property: -

Bit	7	6	5	4	3	2	1	0
	LOCKBIT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	-

Bits 7:0 – LOCKBIT[7:0]: Lock Bits

Value	Description
0xC5	The device is open
other	The device is locked

7. Peripherals and Architecture

7.1. Peripheral Module Address Map

The address map show the base address for each peripheral. For complete register description and summary for each peripheral module, refer to the respective module chapters.

Table 7-1. Peripheral Module Address Map

Base Address	Name	Description
0x0000	VIRTA	Virtual Port A
0x0004	VIRTB	Virtual Port B
0x0008	VIRTC	Virtual Port C
0x001C	GPIO	General Purpose IO registers
0x0030	CPU	CPU
0x0040	RSTCTRL	Reset Controller
0x0050	SLPCTRL	Sleep Controller
0x0060	CLKCTRL	Clock Controller
0x0080	BOD	Brown-Out Detector
0x00A0	VREF	Voltage Reference
0x0100	WDT	Watchdog Timer
0x0110	CPUINT	Interrupt Controller
0x0120	CRCSCAN	Cyclic Redundancy Check Memory Scan
0x0140	RTC	Real Time Counter
0x0180	EVSYS	Event System
0x01C0	CCL	Configurable Custom Logic
0x0200	PORTMUX	Port Multiplexer
0x0400	PORTA	Port A Configuration
0x0420	PORTB	Port B Configuration
0x0440	PORTC	Port C Configuration
0x0600	ADC0	Analog to Digital Converter/Peripheral Touch Controller
0x0800	USART0	Universal Synchronous Asynchronous Receiver Transmitter
0x0810	TWI0	Two Wire Interface
0x0820	SPI0	Serial Peripheral Interface
0x0A00	TCA0	Timer/Counter Type A
0x0A40	TCB0	Timer/Counter Type B 0

Base Address	Name	Description
0x0A80	TCD0	Timer/Counter Type D
0x0F00	SYSCFG	System Configuration
0x1000	NVMCTRL	Non Volatile Memory Controller
0x1100	SIGROW	Signature Row
0x1280	FUSES	Device specific fuses
0x1300	USERROW	User Row

7.2. Interrupt Vector Mapping

Each of the 26 interrupt vectors is connected to one peripheral instance, as shown in the table below. A peripheral can have one or more interrupt sources, see the 'Interrupt' section in the 'Functional Description' of the respective peripheral for more details on the available interrupt sources.

When the interrupt condition occurs, an Interrupt Flag is set in the Interrupt Flags register of the peripheral (*peripheral*_INTFLAGS.*name*IF).

An interrupt is enabled or disabled by writing to the corresponding bit in the peripheral's Interrupt Control register (*peripheral*_INTCTRL.*name*IE).

An interrupt request is generated when the corresponding interrupt is enabled and the Interrupt Flag is set. The interrupt request remains active until the Interrupt Flag is cleared. See the peripheral's INTFLAGS register for details on how to clear Interrupt Flags.

Note: Interrupts must be enabled globally for interrupt requests to be generated.

Related Links

NVMCTRL - Non Volatile Memory Controller on page 52
PORT - I/O Pin Controller on page 132
RTC - Real Time Counter on page 302
SPI - Serial Peripheral Interface on page 363
USART - Universal Synchronous and Asynchronous Receiver and Transmitter on page 326
TWI - Two Wire Interface on page 378
CRCSCAN - Cyclic Redundancy Check Memory Scan on page 410
TCA - 16-bit Timer/Counter Type A on page 179
TCB - 16-bit Timer/Counter Type B on page 228
TCD - 12-bit Timer/Counter Type D on page 252
AC – Analog Comparator on page 437
ADC - Analog to Digital Converter on page 448

8. AVR CPU

8.1. Overview

All Atmel AVR devices use the 8-bit AVR CPU. The main function of the CPU is to execute the code and perform all calculations. The CPU is able to access memories, perform calculations, control peripherals, and execute instructions in the program memory. Interrupt handling is described in a separate section.

Related Links

Memories on page 20
NVMCTRL - Non Volatile Memory Controller on page 52
CPUINT - CPU Interrupt Controller on page 97

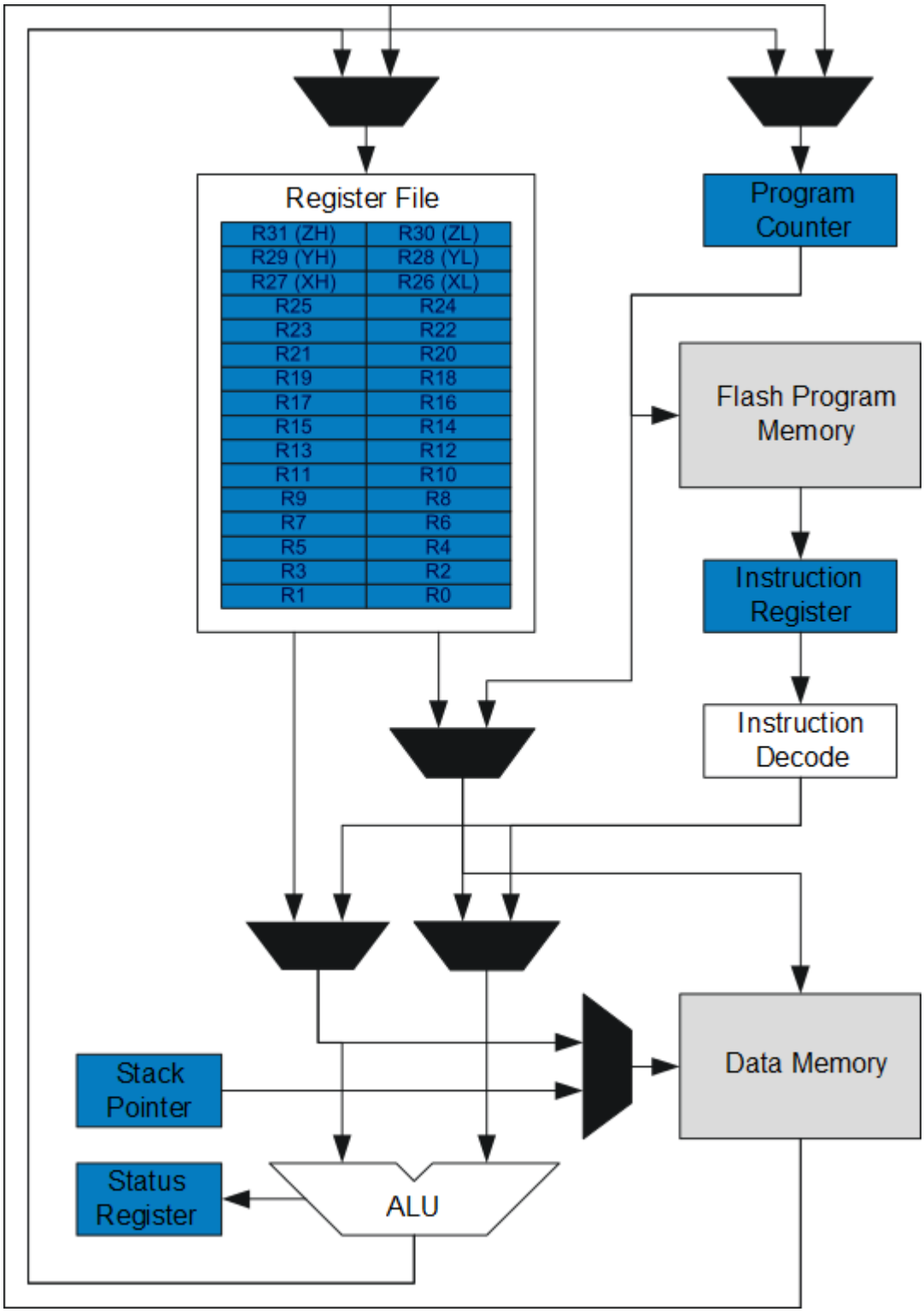
8.2. Features

- 8-bit, high-performance Atmel AVR RISC CPU
  - 135 instructions
  - Hardware multiplier
- 32x8-bit registers directly connected to the ALU
- Stack in RAM
- Stack pointer accessible in I/O memory space
- Direct addressing of up to 64KB of unified memory
  - Entire Flash accessible with all LD/ST instructions
- True 16-bit access to 16-bit I/O registers
- Efficient support for 8-, 16-, and 32-bit arithmetic
- Configuration Change Protection for system-critical features
- Native OCD support
  - 2 hardware breakpoints
  - Change of flow, interrupt and software breakpoints
  - Runtime readout of Stack Pointer register, program counter (PC), and Status register
  - Register file read- and writable in stopped mode

8.3. Architecture

In order to maximize performance and parallelism, the AVR CPU uses a Harvard architecture with separate buses for program and data. Instructions in the program memory are executed with single-level pipelining. While one instruction is being executed, the next instruction is pre-fetched from the program memory. This enables instructions to be executed on every clock cycle.

Figure 8-1. AVR CPU Architecture



The Arithmetic Logic Unit (ALU) supports arithmetic and logic operations between registers or between a constant and a register. Single-register operations can also be executed in the ALU. After an arithmetic operation, the status register is updated to reflect information about the result of the operation.

The ALU is directly connected to the fast-access register file. The 32x8-bit general purpose working registers all have single clock cycle access time allowing single-cycle arithmetic logic unit operation

between registers or between a register and an immediate. Six of the 32 registers can be used as three 16-bit address pointers for program and data space addressing, enabling efficient address calculations.

The program memory bus is connected to Flash, and the first program memory Flash address is 0x0000.

The data memory space is divided into I/O registers, SRAM, EEPROM and Flash.

All I/O status and control registers reside in the lowest 4KB addresses of the data memory. This is referred to as the I/O memory space. The lowest 64 addresses are accessed directly with single cycle IN/OUT instructions, or as the data space locations from 0x00 to 0x3F. These addresses can also be accessed using load (LD/LDS/LDD) and store (ST/STS/STD) instructions. The lowest 32 addresses can even be accessed with single cycle SBI/CBI instructions and SBIS/SBIC instructions. The rest is the extended I/O memory space, ranging from 0x0040 to 0x0FFF. I/O registers here must be accessed as data space locations using load and store instructions.

Data addresses 0x1000 to 0x1800 are reserved for memory mapping of fuses, the NVM controller and EEPROM. The addresses from 0x1800 to 0x7FFF are reserved for other memories, such as SRAM.

The Flash is mapped in the data space from 0x8000 and above. The Flash can be accessed with all load and store instructions by using addresses above 0x8000. The LPM instruction accesses the Flash as in the code space, where the Flash starts at address 0x0000.

For a summary of all AVR instructions, refer to the *Instruction Set Summary*. For details of all AVR instructions, refer to http://www.atmel.com/avr.

Related Links

Instruction Set Summary on page 525
NVMCTRL - Non Volatile Memory Controller on page 52
Memories on page 20

8.4. ALU - Arithmetic Logic Unit

The Arithmetic Logic Unit supports arithmetic and logic operations between registers, or between a constant and a register. Single-register operations can also be executed.

The ALU operates in direct connection with all 32 general purpose registers. Arithmetic operations between general purpose registers or between a register and an immediate are executed in a single clock cycle, and the result is stored in the register file. After an arithmetic or logic operation, the Status register (CPU_SREG) is updated to reflect information about the result of the operation.

ALU operations are divided into three main categories – arithmetic, logical, and bit functions. Both 8- and 16-bit arithmetic is supported, and the instruction set allows for efficient implementation of 32-bit arithmetic. The hardware multiplier supports signed and unsigned multiplication and fractional format.

8.4.1. Hardware Multiplier

The multiplier is capable of multiplying two 8-bit numbers into a 16-bit result. The hardware multiplier supports different variations of signed and unsigned integer and fractional numbers:

- Multiplication of unsigned integers
- Multiplication of signed integers
- Multiplication of a signed integer with an unsigned integer
- Multiplication of unsigned fractional numbers
- Multiplication of signed fractional numbers
- Multiplication of a signed fractional number with an unsigned one

A multiplication takes two CPU clock cycles.

8.5. Functional Description

8.5.1. Program Flow

After Reset, the CPU starts to execute instructions from the lowest address in the Flash program memory, 0x0000. The program counter (PC) addresses the next instruction to be fetched.

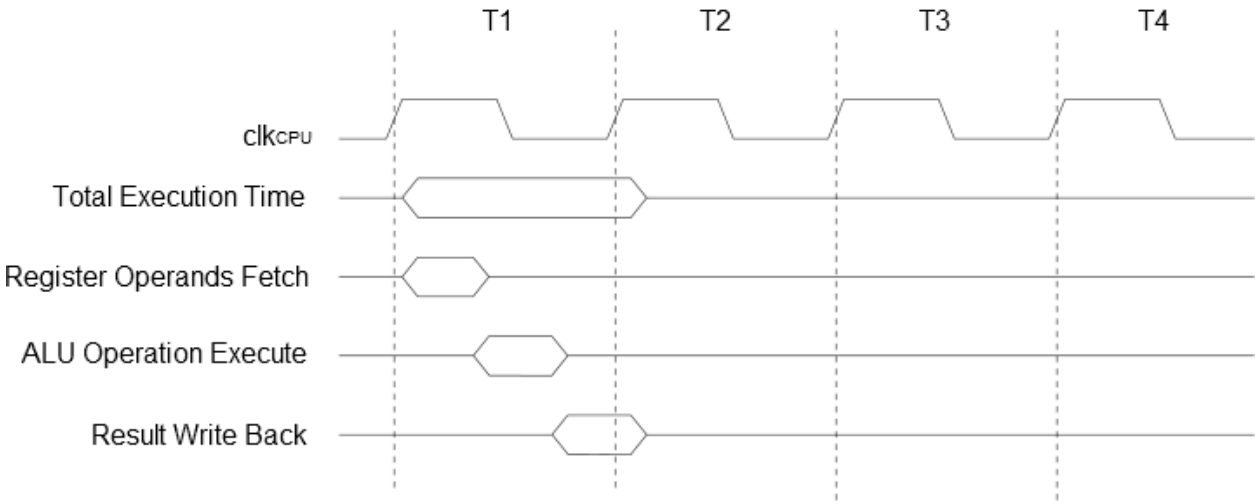
Program flow is provided by conditional and unconditional jump and call instructions, capable of addressing the whole address space directly. Most AVR instructions use a 16-bit word format, while a limited number use a 32-bit format.

During interrupts and subroutine calls, the return address PC is stored on the stack as a word pointer. The stack is allocated in the general data SRAM, and consequently the stack size is only limited by the total SRAM size and the usage of the SRAM. After Reset, the stack pointer (SP) points to the highest address in the internal SRAM. The SP is read/write accessible in the I/O memory space, enabling easy implementation of multiple stacks or stack areas. The data SRAM can easily be accessed through the five different addressing modes supported in the AVR CPU.

8.5.2. Instruction Execution Timing

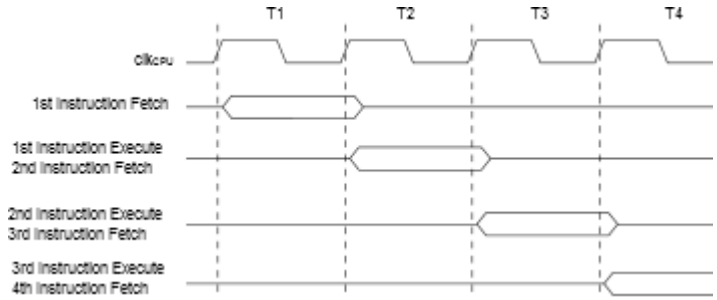
The AVR CPU is clocked by the CPU clock, CL_CPU. No internal clock division is used. The Figure below shows the parallel instruction fetches and instruction executions enabled by the Harvard architecture and the fast-access register file concept. This is the basic pipelining concept used to obtain up to 1MIPS/MHz performance with high efficiency.

Figure 8-2. The Parallel Instruction Fetches and Instruction Executions



The following Figure shows the internal timing concept for the register file. In a single clock cycle, an ALU operation using two register operands is executed, and the result is stored back to the destination register.

Figure 8-3. Single Cycle ALU Operation



8.5.3. Status Register

The Status register (CPU_SREG) contains information about the result of the most recently executed arithmetic or logic instruction. This information can be used for altering program flow in order to perform conditional operations.

Note: The Status register is updated after all ALU operations, as specified in the Instruction Set Summary.

This will in many cases remove the need for using the dedicated compare instructions, resulting in faster and more compact code.

The Status register is not automatically stored when entering an interrupt routine, and not restored when returning from an interrupt. This must be handled by software.

The Status register (CPU_SREG(is accessible in the I/O memory space.

Related Links

Instruction Set Summary on page 525

8.5.4. Stack and Stack Pointer

The Stack is used for storing return addresses after interrupts and subroutine calls. It can also be used for storing temporary data. The Stack Pointer (SP) always point to the top of the Stack. The SP consists of the Stack Pointer bits in the Stack Pointer register (CPU_SP.SP). CPU_SP is implemented as two 8-bit registers that are accessible in the I/O memory space.

Data are pushed and popped from the Stack using the PUSH and POP instructions. The Stack grows from a higher memory location to a lower memory location. This implies that pushing data onto the Stack decreases the SP, and popping data off the Stack increases the SP. The Stack Pointer is automatically loaded after Reset, and the initial value is the highest address of the internal SRAM. If the Stack is changed, it must be set to point above address 0x2000, and it must be defined before any subroutine calls are executed or before interrupts are enabled.

During interrupts or subroutine calls, the return address is automatically pushed on the Stack as a word pointer. The return address is two bytes and pushed to the Stack with the least significant byte first (at the higher address). The return address is popped off the Stack when returning from interrupts using the RETI instruction, and from subroutine calls using the RET instruction. The return address is saved on the Stack as a word pointer, which means a byte pointer return address of 0x0006 is saved on the Stack as 0x0003 (shifted one bit to the right), pointing to the fourth 16-bit instruction word in the program memory.

The SP is decremented by '1' when data are pushed on the Stack with the PUSH instruction, and incremented by '1' when data is popped off the Stack using the POP instruction.

To prevent corruption when updating the Stack pointer from software, a write to SPL will automatically disable interrupts for up to four instructions or until the next I/O memory write.

8.5.5. Register File

The register file consists of 32x 8-bit general purpose working registers with single clock cycle access time. The register file supports the following input/output schemes:

- One 8-bit output operand and one 8-bit result input
- Two 8-bit output operands and one 8-bit result input
- Two 8-bit output operands and one 16-bit result input
- One 16-bit output operand and one 16-bit result input

Six of the 32 registers can be used as three 16-bit address register pointers for data space addressing, enabling efficient address calculations. One of these address pointers can also be used as an address pointer for lookup tables in Flash program memory.

Figure 8-4. AVR CPU General Purpose Working Registers

7 0	Addr.	
R0	0x00	
R1	0x01	
R2	0x02	
...		
R13	0x0D	
R14	0x0E	
R15	0x0F	
R16	0x10	
R17	0x11	
...		
R26	0x1A	X-register Low Byte
R27	0x1B	X-register High Byte
R28	0x1C	Y-register Low Byte
R29	0x1D	Y-register High Byte
R30	0x1E	Z-register Low Byte
R31	0x1F	Z-register High Byte

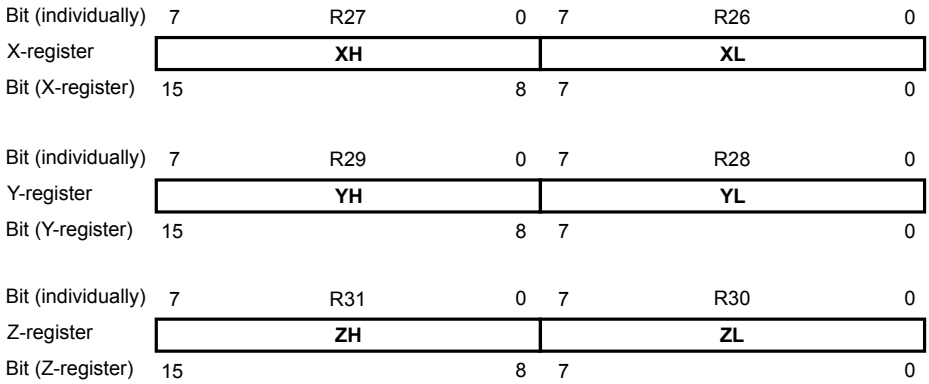
The register file is located in a separate address space, so the registers are not accessible as data memory.

8.5.5.1. The X-, Y-, and Z- Registers

Registers R26...R31 have added functions besides their general-purpose usage.

These registers can form 16-bit address pointers for addressing data memory. These three address registers are called the X-register, Y-register, and Z-register. The Z-register can also be used as an address pointer to read from and/or write to the Flash program memory, signature and user rows, fuses, and lock bits.

Figure 8-5. The X-, Y- and Z-registers



The lowest register address holds the least-significant byte (LSB), and the highest register address holds the most-significant byte (MSB). In the different addressing modes, these address registers function as fixed displacement, automatic increment, and automatic decrement.

Related Links

Instruction Set Summary on page 525

8.5.6. Accessing 16-bit Registers

The AVR data bus is 8 bits wide, and so accessing 16-bit registers requires atomic operations. These registers must be byte-accessed using two read or write operations. 16-bit registers are connected to the 8-bit bus and a temporary register using a 16-bit bus.

For a write operation, the low byte of the 16-bit register must be written before the high byte. The low byte is then written into the temporary register. When the high byte of the 16-bit register is written, the temporary register is copied into the low byte of the 16-bit register in the same clock cycle.

For a read operation, the low byte of the 16-bit register must be read before the high byte. When the low byte register is read by the CPU, the high byte of the 16-bit register is copied into the temporary register in the same clock cycle as the low byte is read. When the high byte is read, it is then read from the temporary register.

This ensures that the low and high bytes of 16-bit registers are always accessed simultaneously when reading or writing the register.

Interrupts can corrupt the timed sequence if an interrupt is triggered and accesses the same 16-bit register during an atomic 16-bit read/write operation. To prevent this, interrupts can be disabled when writing or reading 16-bit registers.

The temporary registers can also be read and written directly from user software.

8.5.7. CCP - Configuration Change Protection

System critical I/O register settings are protected from accidental modification. Flash self-programming (via store to NVM controller) is protected from accidental execution. This is handled globally by the configuration change protection (CCP) register. Changes to the protected I/O registers or bits, or execution of protected instructions, are only possible after the CPU writes a signature to the CCP register. The different signatures are listed in the description of the CCP register (CPU_CCP).

There are two modes of operation: one for protected I/O registers, and one for the protected self-programming.

Related Links

CCP on page 49

8.5.7.1. Sequence for Write Operation to Configuration Change Protected I/O Registers

In order to write to registers protected by CCP, these steps are required:

1. The application code writes the signature that enables change of protected I/O registers to the CCP bit field in the CPU_CCP register (CPU_CCP.CCP).
2. Within four instructions, the application code must write the appropriate data to the protected register.
   Note: Most protected registers also contain a write enable/change enable/lock bit. This bit must be written to '1' in the same operation as the data are written.

   The protected change is immediately disabled if the CPU performs write operations to the I/O register or data memory, if load or store accesses to Flash, NVMCTRL, EEPROM are conducted, or if the SLEEP instruction is executed.

8.5.7.2. Sequence for Execution of Self-Programming

In order to execute self-programming (the execution of writes to the NVM controller's command register), these steps are required:

1. The application code temporary enables self-programming by writing the SPM signature to the CCP register (CPU_CCP.CCP).
2. Within four instructions, the application code must execute the appropriate instruction. The protected change is immediately disabled if the CPU performs accesses to the Flash, NVMCTRL, or EEPROM, or if the SLEEP instruction is executed.

Once the correct signature is written by the CPU, interrupts will be ignored for the duration of the configuration change enable period. Any interrupt request (including non-maskable interrupts) during the CCP period will set the corresponding interrupt flag as normal, and the request is kept pending. After the CCP period is completed, any pending interrupts are executed according to their level and priority.

8.6. Register Summary

Offset	Name	Bit Pos.								
0x04	CCP	7:0	CCP[7:0]							
0x05 ... 0x0C	Reserved									
0x0D	SP	7:0	SP[7:0]							
0x0E		15:8	SP[15:8]							
0x0F	SREG	7:0	I	T	H	S	V	N	Z	C

8.7. Register Description

8.7.1. Configuration Change Protection

Name: CCP
Offset: 0x04
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	CCP[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – CCP[7:0]: Configuration Change Protection
Writing the correct signature to this bit field allows changing protected I/O registers or executing protected instructions within the next four CPU instructions.

All interrupts are ignored during these cycles. After these cycles, interrupts will automatically be handled again by the CPU, and any pending interrupts will be executed according to their level and priority.

When the protected I/O register signature is written, CCP[0] will read as '1' as long as the CCP feature is enabled.

When the protected self-programming signature is written, CCP[1] will read as '1' as long as the CCP feature is enabled.

CCP[7:2] will always read as zero.

Value	Name	Description
0x9D	SPM	Allow Self-Programming
0xD8	IOREG	Un-protect protected I/O registers

8.7.2. Stack Pointer

The CPU.SP holds the Stack Pointer (SP) that points to the top of the Stack. After Reset, the Stack Pointer points to the highest internal SRAM address.

Only the number of bits required to address the available data memory including external memory (up to 64KB) is implemented for each device. Unused bits will always read as zero.

The CPU.SPL and CPU.SPH register pair represents the 16-bit value, CPU.SP. The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + 0x01. For more details on reading and writing 16-bit registers, refer to Accessing 16-bit Registers.

Note: To prevent corruption when updating the Stack Pointer from software, a write to CPU.SPL will automatically disable interrupts for the next four instructions or until the next I/O memory write.

Name: SP
Offset: 0x0D
Reset: 0xxxxx
Property: -

Bit	15	14	13	12	11	10	9	8
	SP[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	x	x	x	x	x	x	x	x

Bit	7	6	5	4	3	2	1	0
	SP[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	x	x	x	x	x	x	x	x

Bits 15:8 – SP[15:8]: Stack Pointer high byte
These bits hold the MSB of the 16-bit register.

Bits 7:0 – SP[7:0]: Stack Pointer low byte
These bits hold the LSB of the 16-bit register.

8.7.3. Status Register

The Status register contains information about the result of the most recently executed arithmetic or logic instruction. For details about the bits in this register and how they are affected by the different instructions, see the *Instruction Set Summary*.

Name: SREG
Offset: 0x0F
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	I	T	H	S	V	N	Z	C
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 7 – I: Global Interrupt Enable
Writing a '1' to this bit enable interrupts on the device.

Writing a '0' to this bit disables interrupts on the device, independent of the individual interrupt enable settings of the peripherals.

This bit is not cleared by hardware after an interrupt has occurred.

This bit can be set and cleared by the application with the SEI and CLI instructions.

Changing the I flag through the I/O-register result in a one-cycle wait state on the access.

Bit 6 – T: Bit Copy Storage
The bit copy instructions bit load (`BLD`) and bit store (`BST`) use the T bit as source or destination for the operated bit.

A bit from a register in the register file can be copied into this bit by the `BST` instruction, and this bit can be copied into a bit in a register in the register file by the `BLD` instruction.

Bit 5 – H: Half Carry Flag
This bit indicates a half carry in some arithmetic operations. Half carry is useful in BCD arithmetic.

Bit 4 – S: Sign Bit, S = N ⊕ V
The sign bit (S) is always an exclusive or (*xor*) between the negative flag (N) and the two's complement overflow flag (V).

Bit 3 – V: Two's Complement Overflow Flag
The two's complement overflow flag (V) supports two's complement arithmetic.

Bit 2 – N: Negative Flag
The negative flag (N) indicates a negative result in an arithmetic or logic operation.

Bit 1 – Z: Zero Flag
The zero flag (Z) indicates a zero result in an arithmetic or logic operation.

Bit 0 – C: Carry Flag
The carry flag (C) indicates a carry in an arithmetic or logic operation.

9. NVMCTRL - Non Volatile Memory Controller

9.1. Overview

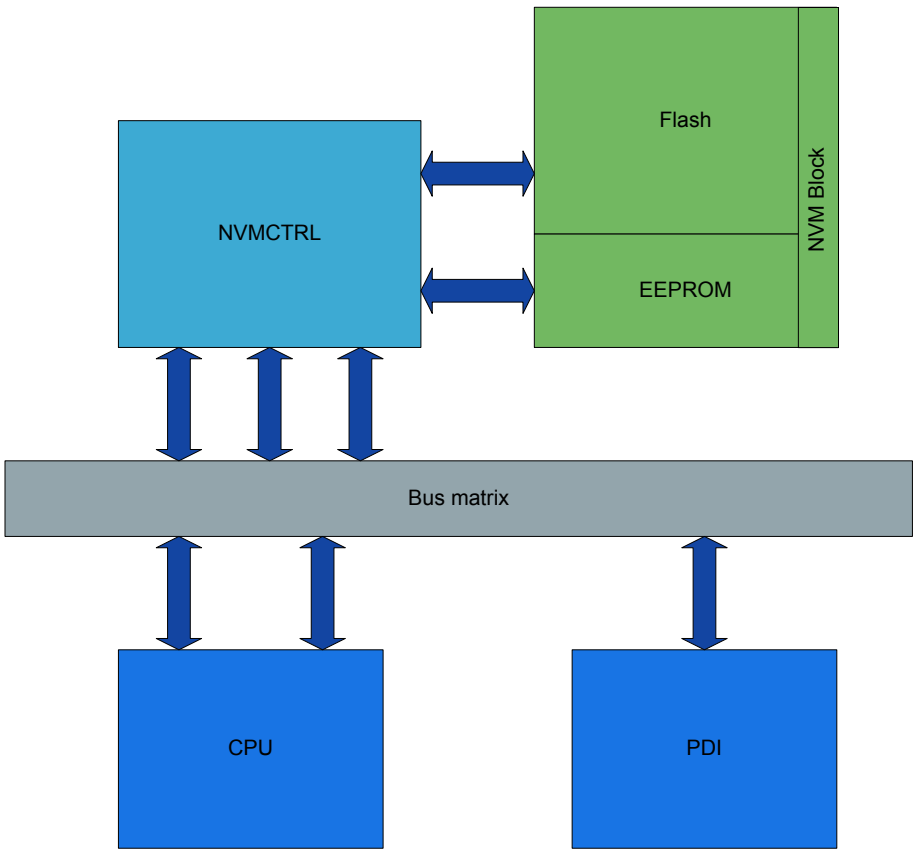
The NVM Controller (NVMCTRL) is the interface between the device, the Flash, and the EEPROM. The Flash and EEPROM are reprogrammable memory blocks that retain their values even with power off. The Flash is mainly used for program storage, but can also be used for data storage. The EEPROM is used for data storage and can be programmed while the CPU is running the program from the Flash.

9.2. Features

- Unified memory
- In-system programmable
- Self-programming and boot loader support
- Interface to Sleep Controller for power-down of Flash blocks in sleep modes
- Configurable sections for write protection:
  - Boot section for boot loader code or application code
  - Application code section for application code
  - Application data section for application code or data storage
- Signature row for factory-programmed data:
  - ID for each device type
  - Serial number for each device
  - Calibration bytes for factory calibrated peripherals
- User Row for application data:
  - 16 bytes in size
  - Can be read and written from software
  - Can be written from UPDI on locked device
  - Content is kept after chip erase

9.3. Block Diagram

Figure 9-1. NVMCTRL Block Diagram



9.4. Product Dependencies

In order to use this peripheral, other parts of the system must be configured correctly, as described below.

Table 9-1. NVMCTRL Product Dependencies

Dependency	Applicable	Peripheral
Clocks	Yes	CLKCTRL
I/O Lines and Connections	No	-
Interrupts	Yes	CPUINT
Events	No	-
Debug	Yes	UPDI

Related Links

Clocks on page 54
Interrupts on page 54
Debug Operation on page 54

9.4.1. Clocks

This peripheral always runs on the NVM clock (CLK_NVM). It will request this clock also in sleep modes if a write/erase is ongoing.

Note: On this device, CLK_CPU, CLK_NVM and CLK_PER share the prescaler setting, and hence, always have the same frequency.

Related Links

CLKCTRL - Clock Controller on page 68

9.4.2. I/O Lines and Connections

Not applicable.

9.4.3. Interrupts

Using the interrupts of this peripheral requires the Interrupt Controller to be configured first.

Related Links

CPUINT - CPU Interrupt Controller on page 97
SREG on page 51
Interrupts on page 136

9.4.4. Events

Not applicable.

9.4.5. Debug Operation

When run-time debugging, this peripheral will continue normal operation. Halting the CPU in debugging mode will halt normal operation of the peripheral.

If the peripheral is configured to require periodical service by the CPU through interrupts or similar, improper operation or data loss may result during halted debugging.

Related Links

UPDI - Unified Program Debug Interface on page 490

9.5. Functional Description

9.5.1. Flash/EEPROM Memory Organization

The Flash and EEPROM are divided into a set of pages. A page is the basic unit addressed when programming the Flash and EEPROM.

For the Flash it is only possible to write or erase a whole page at a time, while for the EEPROM it is possible to write and erase one byte at a time. A page consists of several words. The Flash can be divided into three sections of up to 256 pages for different security.

Related Links

Flash Write Protection on page 57

9.5.2. User Page

The User Page is an additional page, outside the regular Flash/EEPROM array. This page can be used to store various data, such as calibration data and serial numbers. This page is not erased by regular chip erase. The User Page is written as normal EEPROM, but there is a special mode to write this page with the UPDI on a locked device.

9.5.3. Read

Reading of the Flash and EEPROM is done by using the normal load instructions with the correct address as shown in the memory map. Reading of the arrays while a write or erase is already ongoing will result in a wait on the bus, and the instruction will hang until the ongoing operation is complete.

9.5.4. Write Operation

All write operations to Flash and EEPROM must go through an internal buffer called the page buffer. The page buffer contains the same number of bytes as an NVM page. The page buffer can be cleared with a command in the NVM control register, and it is automatically cleared after each `write` command. When the page buffer is cleared, all bits are set and all further writes to the page buffer will do an AND operation with the byte and the current contents of the page buffer.

When writing to an address in the Flash, EEPROM, or extra rows, the write goes to the correct address in the page buffer. The address in the page buffer will be set by the least significant bits in the address. After all writes are done to the page buffer, the content of the page buffer can be written to the array with a `write` command. The page buffer will be automatically cleared after each `write` command so it is ready for the next data.

The page buffer is shared between the EEPROM and Flash, so if the page buffer is written with one address in Flash and another in EEPROM, a following `write` command will only clear the page buffer and set the Write Error flag in the Status register (NVMCTRL_STATUS.WRERROR).

Procedure for Page Writes

- Fill the page buffer by writing to the corresponding page address in the memory-mapped data space.
- Write the page buffer to memory with the `erase-write` page command
  - Alternatively, issue an `erase` page command first, then fill the page buffer, and then use the `write` page command
- The Flash Busy flag (NVMCTRL_STATUS.FBUSY) or EEPROM Busy flag (NVMCTRL_STATUS.EEBUSY) will be '1' while the programming is in progress. All access to the array will be stalled.

9.5.4.1. Write Flash

All writes go through the page buffer. To write to the page buffer, write to the wanted Flash address in the memory map, and the data will go to the page buffer. The least significant bits of the address are used to select the correct byte in the page buffer. When the page buffer is filled with the correct data, the contents of the page buffer can be written to the Flash with the command bits in the control register. The page buffer will be automatically cleared after a `write` command. When writing Flash, the whole page in the Flash will be written/erased when the `write`/`erase` command is executed, not just the bytes written.

If data is written outside the current page, the address will wrap around, and the content of the page buffer will be an AND operation of the two writes to the same address.

9.5.4.2. EEPROM Write

All write operations to EEPROM must go through the page buffer. When writing to the EEPROM addresses the data will go to the page buffer. The least significant bits of the address will be used to select which byte in the page buffer to write to. When the wanted number of bytes are written, they can be written to the EEPROM with a `write` command. For the EEPROM, only the bytes changed in the page buffer will be changed in the array.

If writing to an address outside the wanted page, the content of the page buffer will be a logical AND between the two writes.

EEREADY Interrupt. The peripheral provides an EEPROM Ready interrupt (EERADY), with the corresponding flag in the Interrupt Flags register (NVMCTRL_INTFLAGS.EEREADY). This flag is set when the EEPROM is ready for new write/erase operations.

The interrupt is enabled by writing a '1' to the Interrupt Control register (NVMCTRL_INTCTRL.EEREADY). When the EEREADY interrupt is enabled, it will immediately request an interrupt, and it will continue to request interrupts continuously - even if no EEPROM writes are initiated.

To use the EEREADY interrupt feature, the EEPROM interrupt must be enabled immediately after an EEPROM write is initiated, and then disabled again.

9.5.5. Commands

Reading of the Flash/EEPROM and writing of the page buffer is handled with normal load/store instructions. Other operations, such as writing and erasing the memory arrays, are handled by commands in the NVM.

To execute a command in the NVM:

1. Write the NVM command unlock to the Configuration Change Protection register in the CPU (CPU_CCP.CCP).
2. Write the desired command to the Control A register (NVMCTRL_CTRLA.CMD) within the next 4 cycles.

9.5.5.1. Write Command

The `write` command of the Flash controller writes the content of the page buffer to the Flash or EEPROM. The address used for this operation is stored in the Address register (NVMCTRL_ADDR.ADDR). This register will be automatically updated with each page write. The higher bits of the address are used to select which page to write to. The lower bits of the address are used to decode the location inside the page.

If the write is to the Flash, the CPU will stop executing code as long as the Flash is busy with the write operation. If the write is to the EEPROM, the CPU can continue executing code while the operation is ongoing.

The page buffer will be automatically cleared after the operation.

If writing the Flash, the memory sections have to be set up correctly: prior to self-programming, the Flash has to be divided into two or three sections using with the FUSE_BOOTEND and FUSE_APPLEND fuses, since only code in the BOOT section can write to the Application code section, and code in the BOOT and Application sections can write to Application Data.

Related Links

Flash Write Protection on page 57

9.5.5.2. Erase Command

The `erase` command erases the current page. There must be one byte written in the page buffer so the `erase` command can take effect.

For erasing the Flash, first write to one address in the desired page, then execute the command. The whole page in the Flash will then be erased. The CPU will be halted while the erase is ongoing.

For the EEPROM, only the bytes written in the page buffer will be erased when the command is executed. To erase a specific byte, write to its corresponding address before executing the command. The CPU can continue running code while the operation is ongoing.

The page buffer will be automatically cleared after the operation is finished.

9.5.5.3. Erase-Write Operation

The `erase-write` command is a combination of the `erase` and `write` command, but without clearing the page buffer after the erase command: The `erase-write` operation first erases the selected page, then it writes the content of the page buffer to the same page.

When executed on the Flash, the CPU will be halted when the operations are ongoing. When executed on EEPROM, the CPU can continue executing code.

The page buffer will be automatically cleared after the operation.

9.5.5.4. Page Buffer Clear Command

The `page buffer clear` command clears the page buffer. The contents of the page buffer will be all-ones after the operation. The CPU will be halted when the operation executes (7 CPU cycles).

9.5.5.5. Chip Erase Command

The `chip erase` command erases the Flash and the EEPROM. The EEPROM is unaltered if the EEPROM Save During Chip Erase fuse (FUSE_SYSCFG0.EESAVE) is set. The memory will be all-ones after the operation.

9.5.5.6. EEPROM Erase Command

The `EEPROM erase` command erases the EEPROM. The EEPROM will be all-ones after the operation. The CPU will be halted while the EEPROM is being erased.

9.5.5.7. Fuse Write Command

The `fuse write` command writes the fuses. It can only be used by the UPDI, the CPU cannot start this command.

Follow this procedure to use this command:

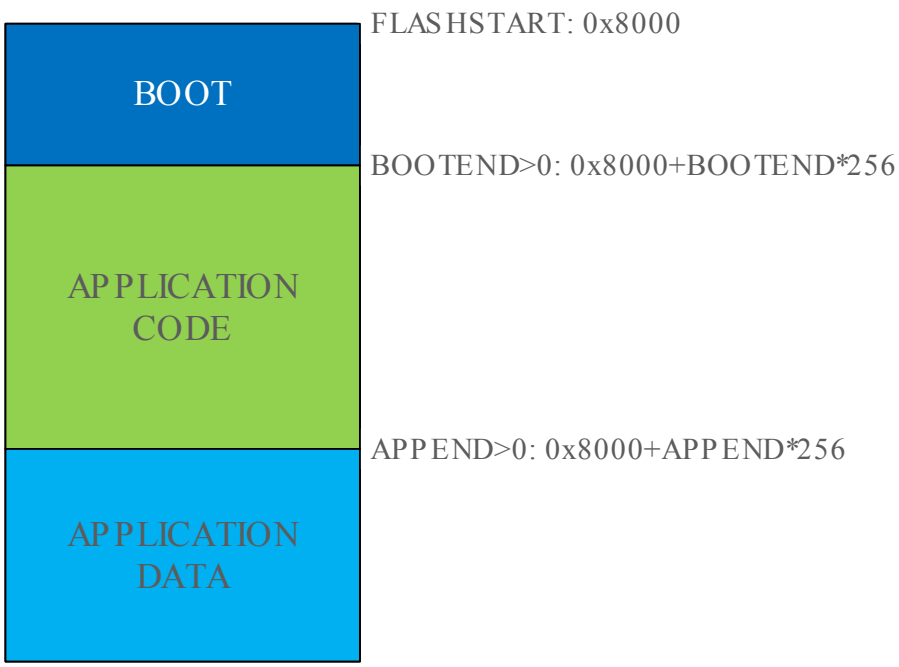
- Write the address of the fuse to the Address register (NVMCTRL_ADDR.ADDR)
- Write the data to be written to the fuse to the Data register (NVMCTRL_DATA.DATA)
- Execute the `fuse write` command.
- After the fuse is written, a Reset is required for the updated value to take effect.

9.5.6. Additional Features

9.5.6.1. Flash Write Protection

The Flash array can be divided into three different sections for write protection. The three different sections are BOOT, application code (APPCODE) and application data (APPDATA).

Figure 9-2. Flash Sections



Section Sizes

The sizes of these sections are set by the Boot Section End fuse (FUSE.BOOTEND.BOOTEND) and Application Code Section End fuse (FUSE_APPEND.APPEND).

The fuses select the section sizes in blocks of 256 bytes. As shown in Figure 9-2, the BOOT section stretches from the start of the Flash until BOOTEND. The APPCODE section runs from right after BOOTEND until APPEND. The remaining area is the APPDATA section. If one of the fuses is written to 0x00, the entire Flash becomes the according section. When both fuses are written to 0x00, the entire Flash becomes BOOT Section.

Inter-Section Write Protection

Between the three Flash sections, a directional write protection is implemented:

- Code in the BOOT section can write to APPCODE and APPDATA.
- Code in APPCODE can write to APPDATA
- Code in APPDATA cannot write to Flash or EEPROM.

Boot Section Lock and Application Code Section Write Protection

The two lock bits (NVMCTRL_CTRLB.APCWP and .BOOTLOCK) can be set to lock further updates of the respective APPCODE or BOOT section until the next Reset.

Note: The CPU can never write to the BOOT section. NVMCTRL_CTRLB.BOOTLOCK prevents reads and execution of code from the BOOT section.

9.5.7. Interrupts

Table 9-2. Available Interrupt Vectors and Sources

Offset	Name	Vector Description	Conditions
0x00	EEREADY	NVM	The EEPROM is ready for new write/erase operations.

When an interrupt condition occurs, the corresponding Interrupt Flag is set in the Interrupt Flags register of the peripheral (*peripheral*_INTFLAGS).

An interrupt source is enabled or disabled by writing to the corresponding bit in the peripheral's Interrupt Enable register (*peripheral*_INTEN).

An interrupt request is generated when the corresponding interrupt source is enabled and the Interrupt Flag is set. The interrupt request remains active until the Interrupt Flag is cleared. See the peripheral's INTFLAGS register for details on how to clear Interrupt Flags.

Note: Interrupt requests are only generated when Interrupts are enabled globally.

9.5.8. Sleep Mode Operation

If there is no ongoing write operation, the NVMCTRL will enter sleep mode when the system enters sleep mode.

If a write operation is ongoing when the system is issued to enter a sleep mode, the NVM block, the NVM Controller and the system clock will remain on until the write is finished. This is valid for all sleep modes, including Power Down sleep mode.

The EEPROM Ready interrupt will wake up the device only from Idle sleep mode.

9.5.9. Configuration Change Protection

This peripheral has registers that are under Configuration Change Protection (CCP). In order to write to these, a certain key must be written to the CPU_CCP register first, followed by a write access to the protected bits within four CPU instructions.

It is possible to try writing to these registers any time, but the values are not altered.

The following registers are under CCP:

Table 9-3. NVMCTRL - Registers under Configuration Change Protection

Register	Key
NVMCTRL_CTRLA	SPM

Related Links

Sequence for Execution of Self-Programming on page 47

9.6. Register Summary

Offset	Name	Bit Pos.								
0x00	CTRLA	7:0						CMD[2:0]		
0x01	CTRLB	7:0							BOOTLOCK	APCWP
0x02	STATUS	7:0						WRERROR	EEBUSY	FBUSY
0x03	INTCTRL	7:0								EEREADY
0x04	INTFLAGS	7:0								EEREADY
0x05	Reserved									
0x06	DATA	7:0	DATA[7:0]							
0x07		15:8	DATA[15:8]							
0x08	ADDR	7:0	ADDR[7:0]							
0x09		15:8	ADDR[15:8]							

9.7. Register Description

9.7.1. Control A

Name: CTRLA
Offset: 0x00
Reset: 0x00
Property: Configuration Change Protection

Bit	7	6	5	4	3	2	1	0
						CMD[2:0]		
Access						R/W	R/W	R/W
Reset						0	0	0

Bits 2:0 – CMD[2:0]: Command
Write this bit field to issue a command. The Configuration Change Protection key for self-programming (SPM) has to be written 5 instructions before this write.

Value	Name	Description
0x0	-	No command
0x1	WP	Write page buffer to memory (NVMCTRL.ADDR selects which memory)
0x2	ER	Erase page (NVMCTRL.ADDR selects which memory)
0x3	ERWP	Erase and write page (NVMCTRL.ADDR selects which memory)
0x4	PBC	Page buffer clear
0x5	CHER	Chip erase: erase Flash and EEPROM (unless EESAVE in FUSE.SYSCFG is '1')
0x6	EEER	EEPROM Erase
0x7	WFU	Write fuse (only accessible through UPDI)

9.7.2. Control B

Name: CTRLB
Offset: 0x01
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
							BOOTLOCK	APCWP
Access							R/W	R/W
Reset							0	0

Bit 1 – BOOTLOCK: Boot Section Lock
Writing a '1' to this bit locks the boot section from read and instruction fetch.

If this bit is '1', a read from the boot section will return '0'. A fetch from the boot section will also return 0 as instruction.

This bit can only be written from the boot section. It can only be cleared to '0' by a Reset.

This bit will only take effect when the boot section is left the first time after the bit is written.

Bit 0 – APCWP: Application Code Section Write Protection
Writing a '1' to this bit protects the application code section from further writes.

This bit can only be written to '1', it is cleared to '0' only by Reset.

9.7.3. Status

Name: STATUS
Offset: 0x02
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
						WRERROR	EEBUSY	FBUSY
Access						R/W	R/W	R/W
Reset						0	0	0

Bit 2 – WRERROR: Write Error
This bit will read '1' when a write error has happened. A write error could be writing to different addresses before doing a page write or writing to a protected area. This bit is valid for the last operation.

Bit 1 – EEBUSY: EEPROM Busy
This bit will read '1' when the EEPROM is busy with a command.

Bit 0 – FBUSY: Flash Busy
This bit will read '1' when the Flash is busy with a command.

9.7.4. Interrupt Control

Name: INTCTRL
Offset: 0x03
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
								EEREADY
Access								R/W
Reset								0

Bit 0 – EEREADY: EEPROM Ready Interrupt
Writing a '1' to this bit enables the interrupt which indicates that the EEPROM is ready for new write/erase operations.

9.7.5. Interrupt Flags

Name: INTFLAGS
Offset: 0x04
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
								EEREADY
Access								R/W
Reset								0

Bit 0 – EEREADY: EEREADY Interrupt Flag
Interrupt flag for the EEPROM interrupt. This bit is cleared by writing a '1' to it.

9.7.6. Data

The NCMCTRL.DATAL and NCMCTRL.DATAH register pair represents the 16-bit value, NCMCTRL.DATA. The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + 0x01. For more details on reading and writing 16-bit registers, refer to Accessing 16-bit Registers.

Name: DATA
Offset: 0x06
Reset: 0x00
Property: -

Bit	15	14	13	12	11	10	9	8
	DATA[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	DATA[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – DATA[15:0]: Data Register
Data register shared between read of Flash/EEPROM/NVM registers and fuse write.

9.7.7. Address

The NCMCTRL.ADDRL and NCMCTRL.ADDRH register pair represents the 16-bit value, NCMCTRL.ADDR. The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + 0x01. For more details on reading and writing 16-bit registers, refer to Accessing 16-bit Registers.

Name: ADDR
Offset: 0x08
Reset: 0x00
Property: -

Bit	15	14	13	12	11	10	9	8
	ADDR[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ADDR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:0 – ADDR[15:0]: Address

Address used for page write/erase and fuse write. The address is automatically updated with each page buffer write.

10. CLKCTRL - Clock Controller

10.1. Overview

The Clock Controller peripheral (CLKCTRL) controls and distributes the clock signals from the available oscillators. The CLKCTRL supports internal and external clock sources, such as 32kHz ULP or 32.768kHz crystal oscillators.

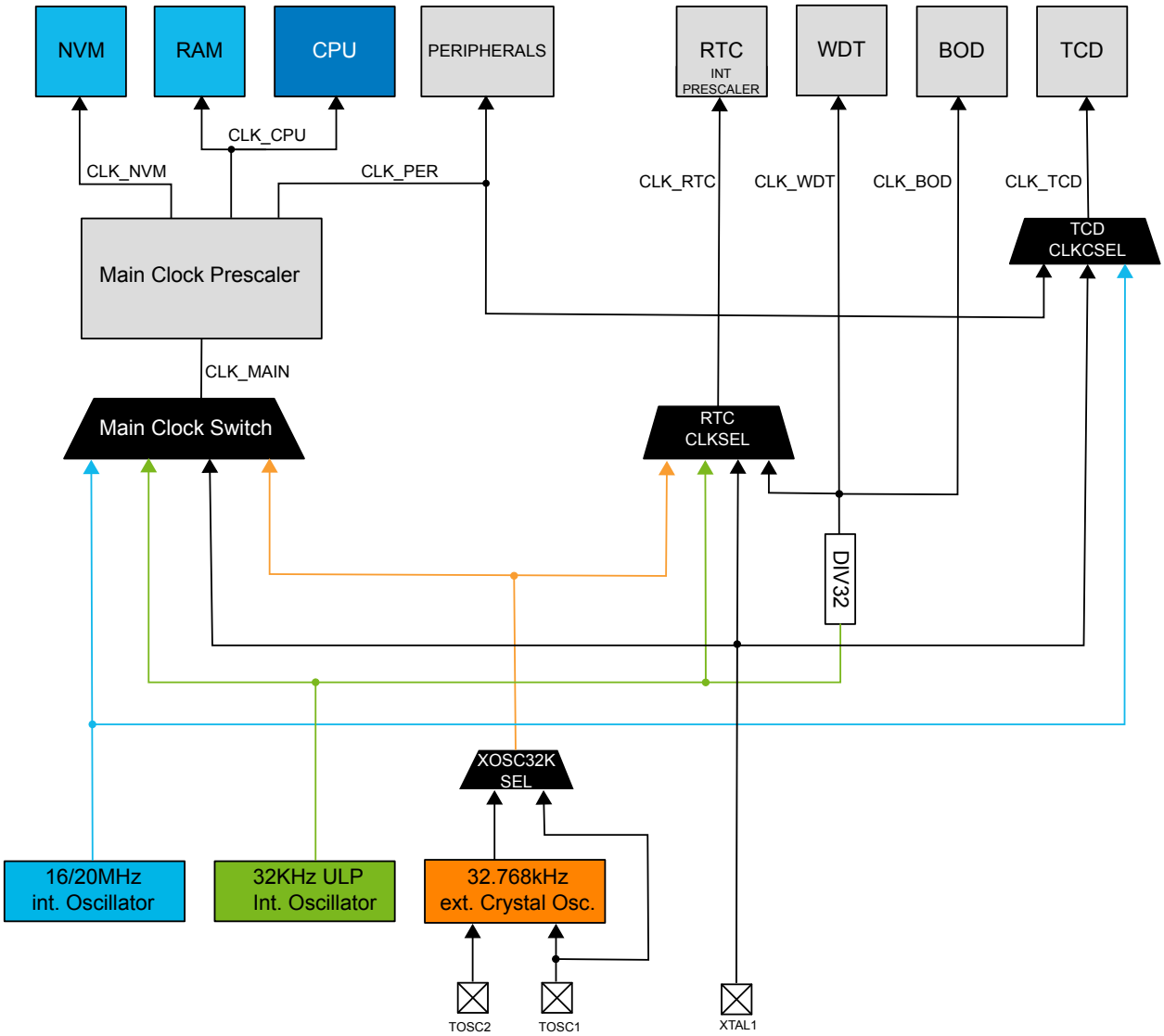
The Main Clock is used by the CPU, RAM, and all peripherals connected to the IO bus. The Main Clock source can be selected and prescaled. Other peripherals can share the same clock source as the Main Clock, but are running asynchronous to the Main Clock domain.

10.2. Features

- All clocks and clock sources are automatically enabled when requested by peripherals
- Internal oscillators:
  - 16/20MHz oscillator (OSC20M)
  - 32KHz Ultra Low Power oscillator (OSCULP32K)
- External clock options:
  - 32.768kHz crystal oscillator (XOSC32K) with external clock option
  - External clock
- Main clock features:
  - Safe run-time switching
  - Prescaler with 1x to 64x division in 12 different settings

10.3. Block Diagram

Figure 10-1. CLKCTRL Block Diagram



The clock system consists of the Main Clock and other asynchronous clocks:

- Main Clock
  This clock is used by the CPU, RAM, Flash, the IO bus and all peripherals connected to the IO bus. It will always be running in Active and Idle sleep mode, and can be running in Standby mode or Power down sleep mode if requested.

  The main clock CLK_MAIN is prescaled and distributed by the Clock Controller:
  - CLK_CPU is used by the CPU and SRAM
  - CLK_NVM is used by the NVMCTRL peripheral to access the non-volatile memory.
  - CLK_PER is used by all peripherals that are not listed under asynchronous clocks.
  - **Note:** On this device, CLK_CPU, CLK_NVM and CLK_PER share the prescaler setting, and hence, always have the same frequency.
- Asynchronous clocks: All these clocks are asynchronous to the Main Clock domain.

- CLK_RTC is used by the RTC/PIT. It will be requested when the RTC/PIT is enabled. The clock source for CLK_RTC should only be changed if the peripheral is disabled.
- CLK_WDT is used by the WDT. It will be requested when the WDT is enabled.
- CLK_BOD is used by the BOD. It will be requested when the BOD is enabled in Sampled Mode.
- CLK_TCD is used by the TCD. It will be requested when the TCD is enabled. The clock source can only be changed if the peripheral is disabled.

The available clock sources are two internal oscillators, one input for an external crystal oscillator that also can be used as a external clock, and a dedicated external clock input.

The clock source for the for the Main Clock domain is configured by writing to Clock Select bits the in the Main Clock Control A register (CLKCTRL_MCLKCTRLA.CKSEL). The asynchronous clock sources are configured by registers in the respective peripheral.

10.4. Signal Description

Signal	Type	Description
CLKOUT	Digital output	CLK_SYS output

Related Links

I/O Multiplexing and Considerations on page 19

10.5. Functional Description

10.5.1. Principle of Operation

The Clock Controller is based on an automatic clock request system, implemented in all peripherals on the device. The peripherals will automatically request the clocks needed. If different clock sources are available, the request is routed to the correct clock source.

10.5.2. Main Clock Selection and Prescaler

All calibrated internal oscillators can be used as the Main Clock source for CLK_MAIN. The Main Clock source is selectable from software, and can be changed during normal operation.

Built-in hardware protection prevents unsafe clock switching:

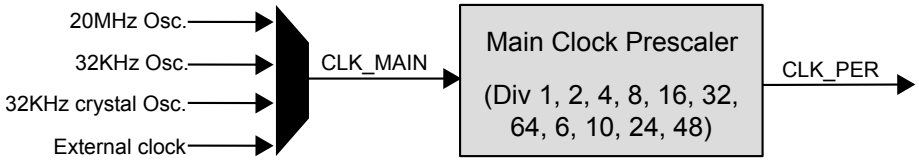
If an external clock source is selected, the switch will only take place if there are edges on the external clock source, indicating that it is stable. If no edges are detected, the clock source switching is not executed. If this happens, it is not possible to change to another clock source again without executing a Reset.

An ongoing clock source switch is indicated by the System Oscillator Changing flag in the Main Clock Status register (CLKCTRL_MCLKSTATUS.SOSC). The stability of the external clock sources is indicated by the respective status flags (CLKCTRL_MCLKSTATUS.EXTS and .XOSC32KS).

Caution: If an external clock source fails while used as CLK_MAIN source, only the WDT can provide a mechanism to switch back via Reset.

CLK_MAIN is fed into a prescaler block before it is used by the peripherals (CLK_PER) in the device. The prescaler has only one stage and can divide CLK_MAIN by a factor from 1 to 64.

Figure 10-2. Main Clock and Prescaler



The Main Clock and Prescaler configuration registers (CLKCTRL_MCLKCTRLA, CLKCTRL_MCLKCTRLB) are protected by the Configuration Change Protection Mechanism, employing a timed write procedure for changing these registers.

Note: On this device, CLK_CPU, CLK_NVM and CLK_PER share the prescaler setting, and hence, always have the same frequency.

10.5.2.1. Sleep Mode Operation

The Main Clock will always run in Active and Idle sleep mode. In Standby, the Main Clock will only run if any peripheral is requesting it and the Run Standby bit in the respective oscillator's Control A register is written to '1' (CLKCTRL_[oscillator]CTRLA.RUNSTDBY).

In Power Down sleep mode, the Main Clock will stop after all NVM operations are done.

10.5.2.2. Main Clock after Reset

After any Reset, CLK_MAIN is provided by the 16/20MHz oscillator (OSC20M), the Prescaler has a division factor 6. Since the actual frequency of the OSC20M is determined by the Frequency Select bits of the Oscillator Configuration fuse (FUSE_OSCCFG.FREQSEL), these frequencies are possible after Reset:

Table 10-1. Peripheral Clock Frequencies after Reset

CLK_MAIN as per FUSE_OSCCFG.FREQSEL	Resulting CLK_PER
16MHz	2.66MHz
20MHz	3.33MHz

10.5.3. Clock Sources

All internal clock sources are enabled automatically when they are requested by a peripheral. The crystal oscillator, based on an external crystal, must be enabled by writing a '1' to the Enable bit in the 32KHz Crystal Oscillator Control A register (CLKCTRL_XOSC32KCTRLA.ENABLE) before it can serve as clock source.

When a clock source is not used/requested it will turn off. It is possible to request an oscillator directly by writing a '1' to the Run Standby bit in the respective oscillator's Control A registers (CLKCTRL_[osc]CTRLA.RUNSTDBY). This will make the oscillator run all the time, except for Power Down sleep mode. When this bit is written to '1', the oscillator startup time is eliminated when requested by a clock.

To check if a clock source is running and stable, read out the oscillator's Status bit in the Main Clock Status register (CLKCTRL_MCLKSTATUS.[osc]S).

Related Links

FUSES - Configuration and User Fuses on page 23

CCP - Configuration Change Protection on page 46

10.5.3.1. Internal Oscillators

The internal oscillators do not require any external components to run. See the related links for accuracy and electrical characteristics.

Related Links

Electrical Characteristics on page 532

16/20MHz Oscillator (OSC20M)

This oscillator can operate at selected frequencies which are selected by the value of the Frequency Select bits in the Oscillator Configuration fuse (FUSE_OSCCFG.FREQSEL). The center frequencies are:

- 16MHz
- 20MHz

The OSC20M is calibrated at startup. It has two different calibration bit fields. The Calibration bit field in the Calibration A register (CLKCTRL_OSC20MCALIBA.CAL20M) allows to calibrate around the current center frequency. The Oscillator Temperature Coefficient Calibration bit field in the Calibration B register (CLKCTRL_OSC20MCALIBB.TEMPCAL20M) allows for adjusting the slope of the temperature drift compensation.

The oscillator calibration can be locked by the Oscillator Lock fuse (FUSE_OSCCFG.OSCLOCK). When this fuse is '1', it is not possible to change the calibration. The calibration is also locked if this oscillator is used as Main Clock source and the Lock Enable bit in the Control B register (CLKCTRL_OSC20MCALIBB.LOCKEN) is '1'.

The calibration bits are also protected by the Configuration Change Protection Mechanism, requiring a timed write procedure for changing the Main Clock and Prescaler settings.

The startup time of this oscillator is analog startup time plus 4 oscillator cycles.

Related Links

Electrical Characteristics on page 532
FUSES - Configuration and User Fuses on page 23
Configuration Change Protection on page 73

32KHz Oscillator (OSCULP32K)

The 32KHz oscillator is optimized for Ultra Low Power (ULP) operation. As compromise, the accuracy is not as high as the accuracy of an external crystal oscillator.

This oscillator also provides the 1KHz signal for the Real Time Counter (RTC), the Watchdog Timer (WDT), and the Brownout Detector (BOD).

Related Links

Electrical Characteristics on page 532
BOD - Brownout Detector on page 155
WDT - Watchdog Timer on page 170
RTC - Real Time Counter on page 302

10.5.3.2. External Clock Sources

The CLKI pin can be used as input for an external clock signal. The TOSC1 and TOSC2 pins are dedicated to driving a 32.768kHz crystal oscillator.

32.768kHz Crystal Oscillator (XOSC32K)

This oscillator supports two input options: Either a crystal is connected to the pins TOSC1 and TOSC2, or an external clock is connected to TOSC1. The input option must be configured by writing the Source Select bit in the XOSC32K Control A register (CLKCTRL_XOSC32KCTRLA.SEL).

The XOSC32K is enabled by writing a '1' to its Enable bit (CLKCTRL_XOSC32KCTRLA.ENABLE). When enabled, the configuration of the GPIO pins used by the XOSC32K is overridden as TOSC1, TOSC2 pins. The Enable bit needs to be set for the oscillator to start running when requested.

The startup time of a given crystal oscillator can be accommodated by writing to the Crystal Startup Time bits in the XOSC32K Control A register (CLKCTRL_XOSC32KCTRLA.CSUT).

When XOSC32K is configured to use an external clock on TOSC1, the startup time is fixed to 2 cycles.

Note: The XOSC32K does not have any calibration mechanism.

External Clock (EXTCLK)

The External Clock is using the input pin CLKI. This GPIO pin is automatically configured for EXTCLK if any peripheral is requesting this clock.

This clock source has a startup time of 2 cycles when first used.

10.5.4. Configuration Change Protection

This peripheral has registers that are under Configuration Change Protection (CCP). In order to write to these, a certain key must be written to the CPU_CCP register first, followed by a write access to the protected bits within four CPU instructions.

It is possible to try writing to these registers any time, but the values are not altered.

The following registers are under CCP:

Table 10-2. CLKCTRL - Registers under Configuration Change Protection

Register	Key
CLKCTRL_MCLKCTRLB	IOREG
CLKCTRL_MCLKLOCK	IOREG

Related Links

Sequence for Write Operation to Configuration Change Protected I/O Registers on page 46

10.6. Register Summary

Offset	Name	Bit Pos.								
0x00	MCLKCTRLA	7:0	CLKOUT						CKSEL[1:0]	
0x01	MCLKCTRLB	7:0				PDIV[3:0]				PEN
0x02	MCLKLOCK	7:0								LOCKEN
0x03	MCLKSTATUS	7:0	EXTS	XOSC32KS	OSC32KS	OSC20MS				SOSC
0x04 ... 0x0F	Reserved									
0x10	OSC20MCTRLA	7:0							RUNSTDBY	
0x11	OSC20MCALIBA	7:0			CAL20M[5:0]					
0x12	OSC20MCALIBB	7:0	LOCK				TEMPCAL20M[3:0]			
0x13 ... 0x17	Reserved									
0x18	OSC32KCTRLA	7:0							RUNSTDBY	
0x19 ... 0x1B	Reserved									
0x1C	XOSC32KCTRLA	7:0			CSUT[1:0]			SEL	RUNSTDBY	ENABLE

10.7. Register Description

10.7.1. Main Clock Control A

Name: MCLKCTRLA
Offset: 0x00
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	CLKOUT						CKSEL[1:0]	
Access	R/W	R	R	R	R	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 7 – CLKOUT: System Clock Out
When this bit is set the system clock is output to pin. This bit is protected by IO write.

Bits 1:0 – CKSEL[1:0]: Clock Select
This bit field selects the source for the Main Clock (CLK_MAIN).

Value	Name	Description
0x0	OSC20M	16/20MHz internal oscillator
0x1	OSCULP32K	32KHz internal Ultra Low Power oscillator
0x2	XOSC32K	32.768kHz external crystal oscillator
0x3	EXTCLK	External clock

10.7.2. Main Clock Control B

Name: MCLKCTRLB
Offset: 0x01
Reset: 0x11
Property: Configuration Change Protection

Bit	7	6	5	4	3	2	1	0
				PDIV[3:0]				PEN
Access	R	R	R	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 4:1 – PDIV[3:0]: Prescaler Division
If the Prescaler Enable (PEN) bit is written to '1', these bits define the division ratio of the Main Clock prescaler.

These bits can be written run-time to vary the clock frequency of the system to suit the application requirements.

The software must ensure a correct configuration of input frequency (CLK_MAIN) and Prescaler settings, such that the resulting frequency CLK_PER never exceeds the allowed maximum.

Value	Description
0x0	2
0x1	4
0x2	8
0x3	16
0x4	32
0x5	64
0x8	6
0x9	10
0xA	12
0xB	24
0xC	48
other	64

Bit 0 – PEN: Prescaler Enable
This bit must be written '1' to enable the prescaler. When enabled, the division ratio is selected by the PDIV bit field.

When this bit is written to '0', the Main Clock will pass through undivided (CLK_PER=CLK_MAIN), regardless of the value of PDIV.

10.7.3. Main Clock Lock

Name: MCLKLOCK
Offset: 0x02
Reset: 0x00
Property: Configuration Change Protection

Bit	7	6	5	4	3	2	1	0
								LOCKEN
Access	R	R	R	R	R	R	R	R/W
Reset	0	0	0	0	0	0	0	0

Bit 0 – LOCKEN: Lock Enable
Writing this bit to '1' will lock the CLKCTRL.MCLKCTRLA and CLKCTRL.MCLKCTRLB registers, and, if applicable, the calibration settings for the current Main Clock source from further software updates. Once locked, the CLKCTRL.MCLKLOCK registers cannot be accessed until the next hardware reset.

This provides a safe method for protecting the CLKCTRL.MCLKCTRLA and CLKCTRL.MCLKCTRLB registers and the calibration settings for the Main Clock source oscillator from unintentional modification by software runaway.

10.7.4. Main Clock Status

Name: MCLKSTATUS
Offset: 0x03
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	EXTS	XOSC32KS	OSC32KS	OSC20MS				SOSC
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit 7 – EXTS: External Clock Status

Value	Description
0	EXTCLK is not stable
1	EXTCLK is stable

Bit 6 – XOSC32KS: XOSC32K Status

Value	Description
0	XOSC32K is not stable
1	XOSC32K is stable

Bit 5 – OSC32KS: OSCULP32K Status

Value	Description
0	OSCULP32K is not stable
1	OSCULP32K is stable

Bit 4 – OSC20MS: OSC20M Status

Value	Description
0	OSC20M is not stable
1	OSC20M is stable

Bit 0 – SOSC: System Oscillator Changing

Value	Description
0	The clock source for CLK_MAIN will change as soon as the new source is stable
1	The clock source for CLK_MAIN is not undergoing a switch

10.7.5. 16/20MHz Oscillator Control A

Name: OSC20MCTRLA
Offset: 0x10
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
							RUNSTDBY	
Access	R	R	R	R	R	R	R/W	R
Reset	0	0	0	0	0	0	0	0

Bit 1 – RUNSTDBY: Run Standby
This bit enables the oscillator in Active, Idle and Standby sleep modes.

The oscillator output is not sent to other peripherals if not requested.

It takes two cycles to open the clock gate after a request, but the oscillator startup time will be removed.

This bit is Configuration Change Protected.

10.7.6. 16/20MHz Oscillator Calibration A

Name: OSC20MCALIBA
Offset: 0x11
Reset: 0x9F
Property: -

Bit	7	6	5	4	3	2	1	0
			CAL20M[5:0]					
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			x	x	x	x	x	x

Bits 5:0 – CAL20M[5:0]: Calibration
These bits change the frequency around the current center frequency of the OSC20M for fine tuning.

This bit is Configuration Change Protected.

10.7.7. 16/20MHz Oscillator Calibration B

Name: OSC20MCALIBB
Offset: 0x12
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	LOCK				TEMPCAL20M[3:0]			
Access	R	R	R		R/W	R/W	R/W	R/W
Reset	0	0	0		0	0	0	0

Bit 7 – LOCK: Oscillator Calibration Locked by Fuse
This register contains the LOCK fuse for the OSC20M oscillator. When set, the calibration settings in CLKCTRL.OSC20MCALIBA or CLKCTRL.OSC20MCALIBB cannot be changed.

Bits 3:0 – TEMPCAL20M[3:0]: Oscillator Temperature Coefficient Calibration
These bits tune the slope of the temperature compensation.

This bit field is Configuration Change Protected.

10.7.8. 32KHz Oscillator Control A

Name: OSC32KCTRLA
Offset: 0x18
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
							RUNSTDBY	
Access	R	R	R	R	R	R	R/W	R
Reset	0	0	0	0	0	0	0	0

Bit 1 – RUNSTDBY: Run Standby
This bit enables the oscillator in Active, Idle and Standby sleep modes.

The output of the OSCULP32K is not sent to other peripherals if not requested.

It takes two cycles to open the clock gate after a request, but the oscillator startup time will be removed.

This bit has Configuration Change Protection to prevent unintentional enabling of the oscillator.

10.7.9. 32.768kHz Crystal Oscillator Control A

Note: The SEL and CSUT bits cannot be changed as long as the ENABLE bit is set or the XOSC32K Stable bit (XOSC32KS) in CLKCTRL.MCLKSTATUS is high.
To change settings in a safe way: write a '0' to the ENABLE bit and wait until XOSC32KS is '0' before re-enabling the XOSC32K with new settings.

Name: XOSC32KCTRLA
Offset: 0x1C
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
			CSUT[1:0]			SEL	RUNSTDBY	ENABLE
Access	R	R	R/W	R/W	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 5:4 – CSUT[1:0]: Crystal Start-Up Time
These bits select the startup time for the XOSC32K. It is write protected when the oscillator is enabled (ENABLE=1).

If SEL=1, the startup time will not be applied.

Value	Name	Description
0x0	1K	1k cycles
0x1	16K	16k cycles
0x2	32K	32k cycles
0x3	64K	64k cycles

Bit 2 – SEL: Source Select
This bit select the external source type. It is write protected when the oscillator is enabled (ENABLE=1).

Value	Description
0	External crystal
1	External clock on TOSC1 pin

Bit 1 – RUNSTDBY: Run Standby
Writing this bit to '1' enables the oscillator in Active, Idle and Standby sleep modes. When this bit is '0', the oscillator is only enabled when requested.

The output of the OSCULP32K is not sent to other peripherals if not requested.

It takes two cycles to open the clock gate after a request, but the oscillator startup time will be removed.

This bit is I/O protected to prevent unintentional enabling of the oscillator.

Bit 0 – ENABLE: Enable
When this bit is written to '1', the current configuration of input pins is overridden. Also, the Source Select bit (SEL) and Crystal Start-Up Time (CSUT) are read-only.

Depending on the on RUNSTBY bit, the oscillator will be turned on all the time if the device is in Active, Idle or Standby sleep mode, or only be enabled when requested.

11. SLPCTRL - Sleep Controller

11.1. Overview

Sleep modes are used to shut down peripherals and clock domains in the device in order to save power. The Sleep Controller (SLPCTRL) controls and handles the transitions between active modes and sleep modes.

In Active mode, the CPU is executing application code. In addition to Active mode, there are these sleep modes: Idle, Standby, and Power Down.

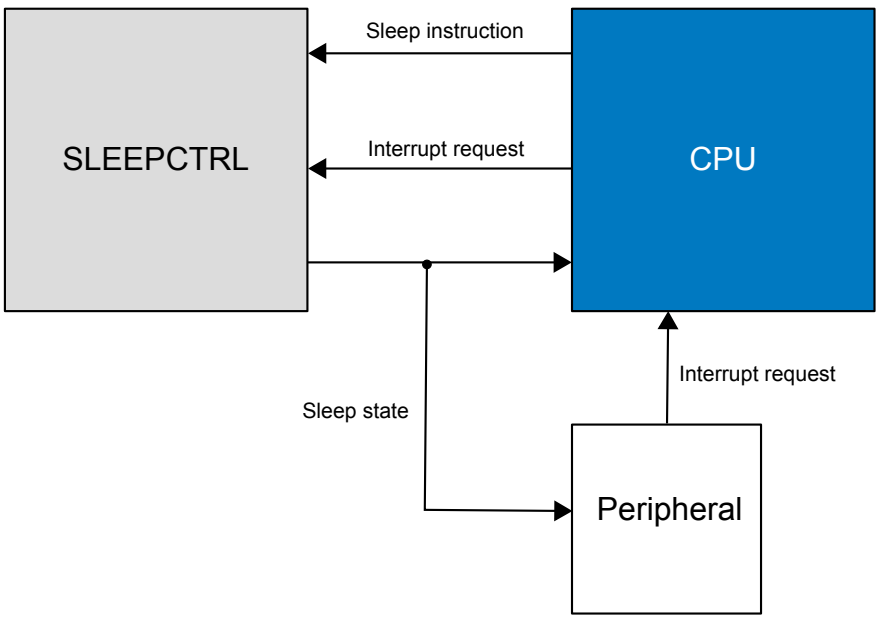
When the device enters a sleep mode, program execution is stopped, and depending on the entered sleep mode, different peripherals and clock domains are turned off.

11.2. Features

- Three sleep modes:
  - Idle
  - Standby
  - Power Down
- Configurable Standby sleep mode where peripherals can be configured as on or off.
- SleepWalking in Standby sleep mode, where the PTC can start running on an Event without waking up the device.

11.3. Block Diagram

Figure 11-1. Sleep Controller in System



11.4. Product Dependencies

In order to use this peripheral, other parts of the system must be configured correctly, as described below.

Table 11-1. SLEEPCTRL Product Dependencies

Dependency	Applicable	Peripheral
Clocks	Yes	CLKCTRL
I/O Lines and Connections	No	-
Interrupts	no	-
Events	No	-
Debug	Yes	UPDI

11.4.1. Clocks

This peripheral depends on the peripheral clock.

Related Links

CLKCTRL - Clock Controller on page 68
Product Dependencies on page 98

11.4.2. I/O Lines and Connections

Not applicable.

11.4.3. Interrupts

Not applicable.

11.4.4. Events

Not applicable.

11.4.5. Debug Operation

When run-time debugging, this peripheral will continue normal operation. The SLPCTRL is only affected by a break in debug operation: If the SLPCTRL is in a sleep mode when a break occurs, the device will wake up and the CLPCTRL will go to active mode, even if there are no pending interrupt requests.

If the peripheral is configured to require periodical service by the CPU through interrupts or similar, improper operation or data loss may result during halted debugging.

11.5. Functional Description

11.5.1. Principle of Operation

To enter a sleep mode, the SLPCTRL must be enabled and the desired sleep mode must be stated. The application code decides when to enter that sleep mode by using a dedicated sleep instruction (SLEEP).

To wake up the device from sleep, Interrupts are used. The available Interrupt wake-up sources are dependent on the configured sleep mode. When an interrupt occurs, the device will wake up and execute the interrupt service routine before continuing normal program execution from the first instruction after the sleep instruction. Any Reset will also take the device out of a sleep mode.

11.5.2. Sleep Modes

In addition to Active mode, there are three different sleep modes, with decreasing power consumption and decreasing functionality.

Idle The CPU stops executing code, no peripherals are disabled.

All interrupt sources can wake up the device.

Standby The user can configure peripherals to be enabled or not, using the respective RUNSTBY bit. This means that the power consumption is highly dependent on what functionality is enabled. The current consumption may vary between the Idle value and close to the Power Down consumption.

SleepWalking is available for the PTC module.

The wake-up sources are Pin interrupts, TWI address match, USART Start-of-Frame interrupt (USART is enabled to run in Standby), PTC window interrupt (PTC enabled to run in Standby), RTC interrupt (RTC enabled to run in Standby), and TCB interrupt.

Power Down Only the WDT is enabled.

The only wake-up sources are the pin change interrupt and TWI address match.

Table 11-2. Sleep Mode Overview

	Active clock domain						Oscillators			Wake-up Sources						
Sleep Mode	CPU clock	Peripheral clock	WDT clock	RTC clock	TCB clock	ADC/PTC clock										
	CLK_CPU	CLK_PER	CLK_WDT	CLK_RTC	CLK_PER	CLK_PER	Main Clock source	WDT oscillator	RTC clock source	INTn and pin change	TWI address match	USART start of frame	PTC window	RTC interrupt	TCB	All other interrupts
Idle		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Standby			X	X*	X*	X*	X*	X	X*	X	X	X*	X*	X*	X	
Power Down			X					X		X	X					

Note: X means active. X* indicates that the RUNSTBY bit of the corresponding peripheral must be set to enter active state.

Note: On this device, CLK_CPU, CLK_NVM and CLK_PER share the prescaler setting, and hence, always have the same frequency.

11.5.3. Basic Operation

11.5.3.1. Initialization

To put the device into a sleep mode, follow these steps:

- Configure and enable the interrupts that should wake up the device from sleep. Also enable global interrupts.

Warning: If there are no interrupts enabled when going to sleep, the device can not wake up again.

- Select the sleep mode to be entered and enable the Sleep Controller by writing to the Sleep Mode bits and the Enable bit in the Control A register (SLPCTRL_CTRLA.SMODE and .SEN).

The wake-up time for the device is dependent on the sleep mode and the Main Clock source: The startup time of the Main Clock source must be added to the wake-up time for sleep modes where the Main Clock source is not kept running.

The content of the register file, SRAM and registers are kept during sleep. If a Reset occurs during sleep, the device will reset, start up, and execute from the Reset vector.

11.5.3.2. Wake-Up Time

The normal wake-up time for the device is 6 Main Clock cycles (CLK_PER), plus the time it takes to start up the Main Clock source:

- In Idle sleep mode the Main Clock source is kept running so it will not be any extra wake-up time.
- In Standby sleep mode the Main Clock might be running, so it depends on the peripheral configuration.
- In Power Down sleep mode only the ULP 32KHz oscillator may be running, if it is used by the BOD or WDT. All other clock sources will be off.

The startup time for the different clock sources is described in the Clock Controller (CLKCTRL) section.

In addition to the normal wake-up time it is possible to make the device wait until the BOD is ready before executing code. This is done by writing 0x3 to the BOD Operation Mode in Sleep bits in the BOD Configuration fuse (FUSE_BODCFG.ACTIVE). If the BOD is ready before the normal wake-up time, the net wake-up time will be the same. If the BOD takes longer than the normal wake-up time, the wake-up time will be extended until the BOD is ready. This is ensures correct supply voltage whenever code is executed.

11.5.4. Configuration Change Protection

Not applicable.

11.6. Register Summary

Offset	Name	Bit Pos.								
0x00	CTRLA	7:0						SMODE[1:0]		SEN

11.7. Register Description

11.7.1. Control A

Name: CTRLA
Offset: 0x00
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
						SMODE[1:0]		SEN
Access	R	R	R	R	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 2:1 – SMODE[1:0]: Sleep Mode
Writing these bits selects the sleep mode entered when the Sleep Enable bit (SEN) is written to '1' and the SLEEP instruction is executed.

Value	Name	Description
0x0	IDLE	Idle sleep mode enabled
0x1	STANDBY	Standby sleep mode enabled
0x2	PDOWN	Power Down sleep mode enabled
other	-	Reserved

Bit 0 – SEN: Sleep Enable
This bit must be written to '1' before the SLEEP instruction is executed to make the MCU enter the selected sleep mode.

To avoid unintentional entering of sleep modes, it is recommended to write SEN=1 just before executing the SLEEP instruction, and to write SEN=0 immediately after waking up.

12. RSTCTRL - Reset Controller

12.1. Overview

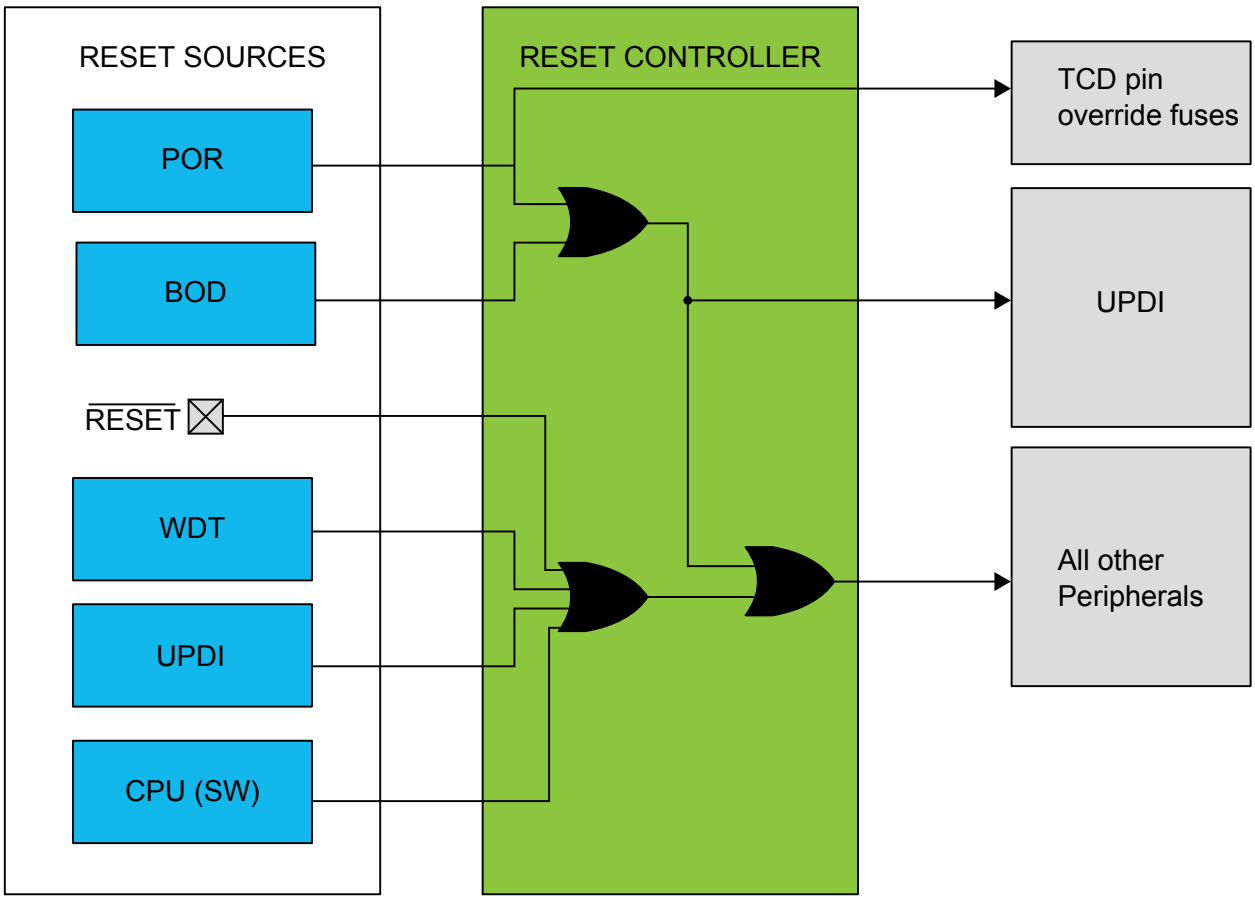
The Reset Controller (RSTCTRL) manages the Reset of the device. It issues a device Reset, sets the device to its initial state, and allows the Reset source to be identified by software.

12.2. Features

- Reset the device and set it to an initial state.
- Reset Flag register for identifying the Reset source by application code.
- Multiple Reset sources:
  - Power supply Reset sources: Brownout Detect (BOD), Power On Reset (POR)
  - User Reset sources: External Reset pin, Watchdog Reset, software Reset, UPDI reset

12.3. Block Diagram

Figure 12-1. Reset System Overview



Note: The UPDI and some registers containing fuse values, such as the TCD pin override fuses, are not reset by all Reset sources.

12.4. Signal Description

Signal	Description	Type
$\overline{\text{RESET}}$	External Reset (active low)	Digital input

12.5. Functional Description

12.5.1. Principle of Operation

The Reset controller combines the different Reset sources and routes the generated Reset on the device.

12.5.2. Basic Operation

12.5.2.1. Initialization

The Reset Controller (RSTCTRL) is always enabled, but the some of the Reset sources must be enabled (either by fuses or by software) before they can request a Reset.

After any Reset the Reset source that caused the Reset is found in the Reset Flag register (RSTCTRL_RSTFR).

After a Power-On Reset, only the POR flag is be set.

The flags are kept until the user clears them by writing a '1' to them.

After Reset from any Reset source, all registers that are loaded from fuses are reloaded.

12.5.2.2. Reset Sources

There are two kind of sources for Resets:

- Power supply Resets are caused by changes in the power supply voltage: Power-on Reset (POR) and Brownout Detector (BOD).
- User Resets are issued by the application, by debug or by pin changes (Software Reset, Watchdog Reset, UPDI Reset and external Reset pin $\overline{\text{RESET}}$).

Power-On Reset (POR)

The Power-on Reset detector is always enabled and is not configurable. It triggers a Reset when the supply voltage is too low for the device to function.

This Reset source will reset everything on the entire device.

Brownout Detector (BOD) Reset Source

The BOD has a similar functionality like the POR, but can be configured to check the supply voltage in different ranges than the POR. See the BOD documentation for details on how to configure it.

A BOD Reset request will reset the entire device except the BOD setting (given by fuses) and the TCD pin override settings.

Related Links

BOD - Brownout Detector on page 155

Software Reset

A software Reset is generated by the CPU by writing a '1' to the Software Reset Enable bit in the Software Reset register (RSTCTRL_SWRR.SWRE)

The software Reset will keep the device in Reset for the next cycle of CLK_PER and resets all peripherals except the UPDI.

External Reset

The External Reset is enabled by fuse (see fuse map). If both UPDI and External Reset are enabled, the External Reset will be moved to an alternative pin.

When enabled, the External Reset requests a Reset as long as the $\overline{\text{RESET}}$ pin is low. The device will stay in Reset until $\overline{\text{RESET}}$ is high again. The UPDI will not be affected by an external Reset.

Related Links

FUSES - Configuration and User Fuses on page 23

Watchdog Reset

The Watchdog Reset is generated by the Watchdog Timer (WDT). See the WDT documentation for details on configuration.

The WDT Reset request resets the entire device except the PDI.

Related Links

WDT - Watchdog Timer on page 170

Universal Program Debug Interface (UPDI) Reset

The UPDI Reset request resets the entitre device except the UPDI itself. See PDI documentation on how to generate a UPDI Reset request.

Related Links

UPDI - Unified Program Debug Interface on page 490

12.5.3. Sleep Mode Operation

The Reset Controller continues to operate in all active and sleep modes.

12.5.4. Configuration Change Protection

This peripheral has registers that are under Configuration Change Protection (CCP). In order to write to these, a certain key must be written to the CPU_CCP register first, followed by a write access to the protected bits within four CPU instructions.

It is possible to try writing to these registers any time, but the values are not altered.

The following registers are under CCP:

Table 12-1. RSTCTRL - Registers under Configuration Change Protection

Register	Key
RSTCTRL_SWRR	IOREG

Related Links

Sequence for Write Operation to Configuration Change Protected I/O Registers on page 46

12.6. Register Summary

Offset	Name	Bit Pos.								
0x00	RSTFR	7:0			PDIRF	SWRF	WDRF	EXTRF	BORF	PORF
0x01	SWRR	7:0								SWRE

12.7. Register Description

12.7.1. Reset Flag Register

All flags are cleared by writing a '1' to them. They are also cleared by a Power-on Reset, with the exception of the Power-On Reset Flag (PORF).

Name: RSTFR
Offset: 0x00
Reset: 0xXX
Property: -

Bit	7	6	5	4	3	2	1	0
			PDIRF	SWRF	WDRF	EXTRF	BORF	PORF
Access	R	R	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	x	x	x	x	x	x

Bit 5 – PDIRF: UPDI Reset Flag
This bit is set if a UPDI Reset occurs.

Bit 4 – SWRF: Software Reset Flag
This bit is set if a Software Reset occurs.

Bit 3 – WDRF: Watchdog Reset Flag
This bit is set if a Watchdog Reset occurs.

Bit 2 – EXTRF: External Reset Flag
This bit is set if an External reset occurs.

Bit 1 – BORF: Brownout Reset Flag
This bit is set if a Brownout Reset occurs.

Bit 0 – PORF: Power-On Reset Flag
This bit is set if a Power-on Reset occurs.

This flag is only cleared by writing a '1' it.

Note: After a POR, only the POR flag is set: all other flags are cleared. No other flag can be set before a full system boot is run after the POR.

12.7.2. Software Reset Register

Name: SWRR
Offset: 0x01
Reset: 0x00
Property: Configuration Change Protection

Bit	7	6	5	4	3	2	1	0
								SWRE
Access	R	R	R	R	R	R	R	R/W
Reset	0	0	0	0	0	0	0	0

Bit 0 – SWRE: Software Reset Enable
When this bit is written to '1', a software reset will occur.

Note that this bit will always read '0', because it is reset when a Reset is issued.

13. CPUINT - CPU Interrupt Controller

13.1. Overview

An interrupt request signals a change of state inside a peripheral, and can be used to alter program execution. Peripherals can have one or more interrupts, and all are individually enabled and configured.

When an interrupt is enabled and configured, it will generate an interrupt request when the interrupt condition occurs.

The CPU Interrupt Controller (CPUINT) handles and prioritizes interrupt requests. When an interrupt request is acknowledged by the CPUINT, the program counter is set to point to the interrupt vector, and the interrupt handler can be executed.

By default, all peripherals are priority level 0. User can set one single interrupt vector to the higher priority level 1. Interrupts are prioritized according to their priority level and their interrupt vector address. Priority level 1 interrupts will interrupt level 0 interrupt handlers. Among priority level 0 interrupts, the priority is determined from the interrupt vector address, where the lowest interrupt vector address has the highest interrupt priority.

Optionally, a round-robin scheduling scheme can be enabled for priority level 0 interrupts. This ensures that all interrupts are serviced within a certain amount of time.

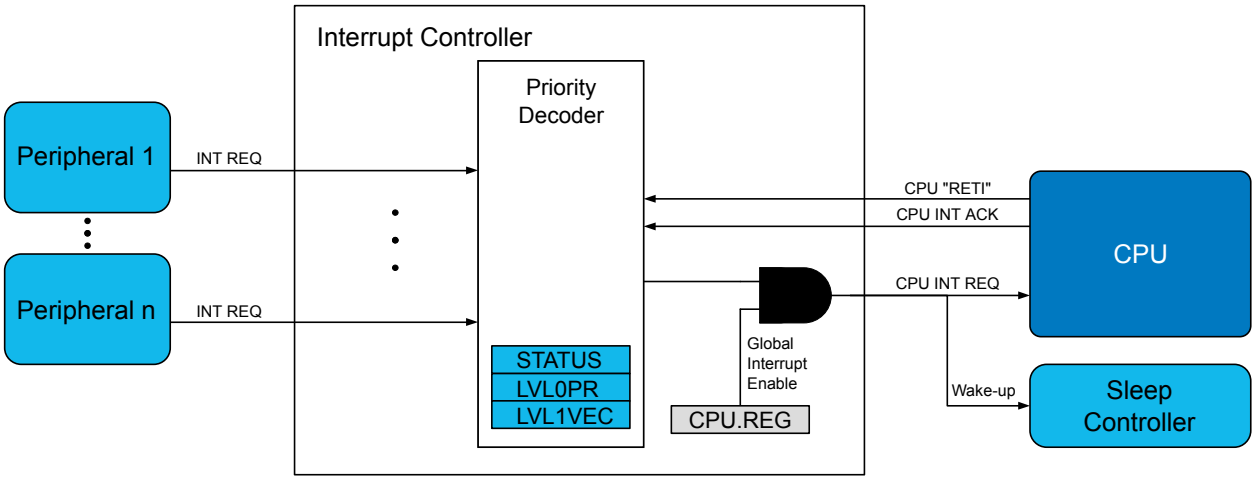
Non-maskable interrupts (NMI) are also supported, and can be used for system critical functions.

13.2. Features

- Short and predictable interrupt response time
- Separate interrupt configuration and vector address for each interrupt
- Interrupt prioritizing by level and vector address
- Two interrupt priority levels: 0 (normal) and 1 (high)
- Higher priority for one interrupt
- Optional round-robin priority scheme for priority level 0 interrupts
- Non-maskable interrupts (NMI) for critical functions
- Interrupt vectors optionally placed in the application section or the boot loader section
- Selectable compact vector table

13.3. Block Diagram

Figure 13-1. CPUINT Block Diagram



13.4. Signal Description

Not applicable.

13.5. Product Dependencies

In order to use this peripheral, other parts of the system must be configured correctly, as described below.

Table 13-1. CPUINT Product Dependencies

Dependency	Applicable	Peripheral
Clocks	Yes	CLKCTRL
I/O Lines and Connections	No	-
Interrupts	No	-
Events	No	-
Debug	Yes	UPDI

Related Links

Clocks on page 86

Debug Operation on page 99

13.5.1. Clocks

This peripheral depends on the peripheral clock.

Related Links

CLKCTRL - Clock Controller on page 68

Product Dependencies on page 98

13.5.2. I/O Lines and Connections

Not applicable.

13.5.3. Interrupts

Not applicable.

13.5.4. Events

Not applicable.

13.5.5. Debug Operation

When run-time debugging, this peripheral will continue normal operation. Halting the CPU in debugging mode will halt normal operation of the peripheral.

If the peripheral is configured to require periodical service by the CPU through interrupts or similar, improper operation or data loss may result during halted debugging.

Related Links

Product Dependencies on page 98
UPDI - Unified Program Debug Interface on page 490

13.6. Functional Description

13.6.1. Principle of Operation

Interrupt generation must be globally enabled by This is done by writing a '1' to the Global Interrupt Enable bit in the CPU Status register (CPU_SREG.I). This bit is not cleared when an interrupt is acknowledged.

When an interrupt is enabled and the interrupt condition occurs, the CPUINT will receive the interrupt request. Based on the interrupt's priority level and the priority level of any ongoing interrupts, the interrupt request is either acknowledged or kept pending until it has priority.

When the interrupt request is acknowledged, the program counter is updated to point to the interrupt vector. The interrupt vector is normally a jump to the interrupt handler (i.e., the software routine that handles the interrupt). After returning from the interrupt handler, program execution continues from where it was before the interrupt occurred. One instruction is always executed before any pending interrupt is served.

The CPUINT Status register (CPUINT_STATUS) contains state information that ensures that the CPUINT returns to the correct interrupt level when the RETI (interrupt return) instruction is executed at the end of an interrupt handler. Returning from an interrupt will return the CPUINT to the state it had before entering the interrupt. The status register (CPUINT_STATUS) is not saved automatically upon an interrupt request. The RET (subroutine return) instruction cannot be used when returning from the interrupt handler routine, as this will not return the CPUINT to its correct state.

13.6.2. Basic Operation

13.6.2.1. Initialization

An interrupt must be initialized in the following order:

1. Configure the CPUINT, if the default configuration is not adequate (optional):
   - Vector handling is configured by writing to the respective bits in the Control A register (CPUINT_CTRLA.IVSEL and .CVT)
   - Vector prioritizing by round-robin is enabled by writing a '1' to the Round-Robin Priority Enable bit (CPUINT_CTRLA.LVL0RR)
   - Select the priority level 1 vector by writing its address to the Interrupt Vector with Level 1 Priority register (CPUINT_LVL1VEC.LVL1VEC).

2. Configure the interrupt conditions within the peripheral, and enable the peripheral's interrupt.
3. Enable interrupts globally by writing a '1' to the Global Interrupt Enable bit in the CPU Status register (CPU_SREG.I).

13.6.2.2. Enabling, Disabling, and Resetting

Global enabling and disabling of interrupts is done by writing a '1' to the Global Interrupt Enable bit in the CPU Status register (CPU_SREG.I).

The desired interrupt lines must also be enabled in the respective peripheral, by writing to the peripheral's Interrupt Control register ([peripheral]_INTCTRL).

Interrupt flags are not automatically cleared after the interrupt is executed. See the respective _INTFLAGS register description on how to clear specific flags.

13.6.2.3. Interrupt Vector Locations

The table below shows Reset addresses and Interrupt vector placement, dependent on the value of Interrupt Vector Select bit in the Control A register (CPUINT_CTRLA.IVSEL).

If the program never enables an interrupt source, the Interrupt Vectors are not used, and regular program code can be placed at these locations.

Table 13-2. Reset and Interrupt Vector Placement

IVSEL	Reset address	Interrupt vectors start address
0	0x0000	Application start address + Interrupt vector offset address
1	0x0000	Interrupt vector offset address(1)

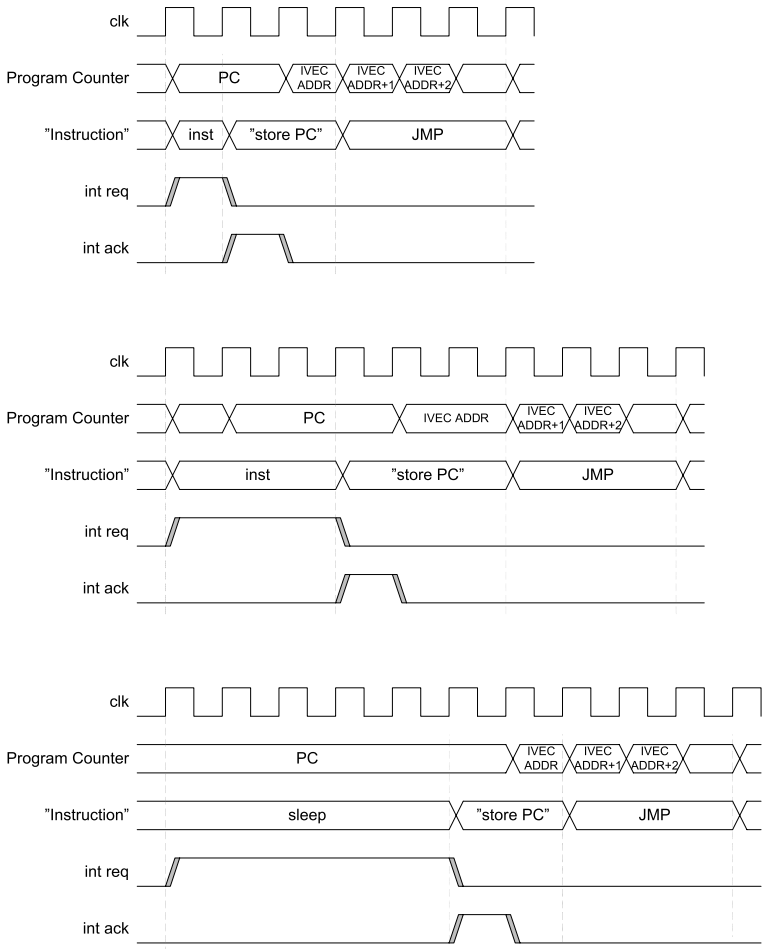
Note: 1. If boot section is defined, it will be placed before application section.

13.6.2.4. Interrupt Response Time

The minimum interrupt response time for all enabled interrupts is three CPU clock cycles: one cycle to finish the ongoing instruction, and two cycles to store the program counter to the stack. After the program counter is pushed on the stack, the program vector for the interrupt is executed.

The jump to the interrupt handler takes three clock cycles. If an interrupt occurs during execution of a multicycle instruction, this instruction is completed before the interrupt is served.

Figure 13-2. Interrupt Execution of a Multicycle Instruction



13.6.3. Interrupt Priority

13.6.3.1. NMI - Non-Maskable Interrupts

An NMI will be executed regardless of the setting of the CPU_SREG.I bit, and it will never change the CPU_SREG.I bit. No other interrupt can interrupt a NMI handler. If more than one NMI is requested at the same time, priority is static according to the interrupt vector address, where the lowest address has highest priority.

Which interrupts are non-maskable is device-dependent and not subject to configuration. Non-maskable interrupts must be enabled before they can be used. Refer to the Interrupt Line Mapping of the device for available NMI lines.

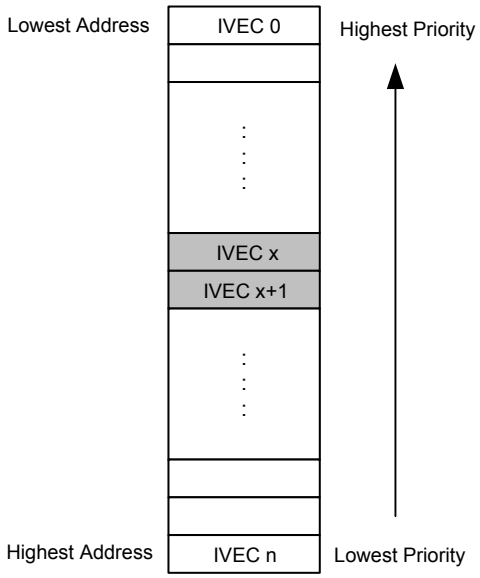
Related Links

Interrupt Vector Mapping on page 39

13.6.3.2. Static Priority

Interrupt vectors (IVEC) are located at fixed addresses. For static priority, the interrupt vector address decides the priority within normal interrupt level, where the lowest interrupt vector address has the highest priority. Refer to the Interrupt Line Mapping of the device for available interrupt lines and their base address offset.

Figure 13-3. Static Priority



Related Links

Interrupt Vector Mapping on page 39

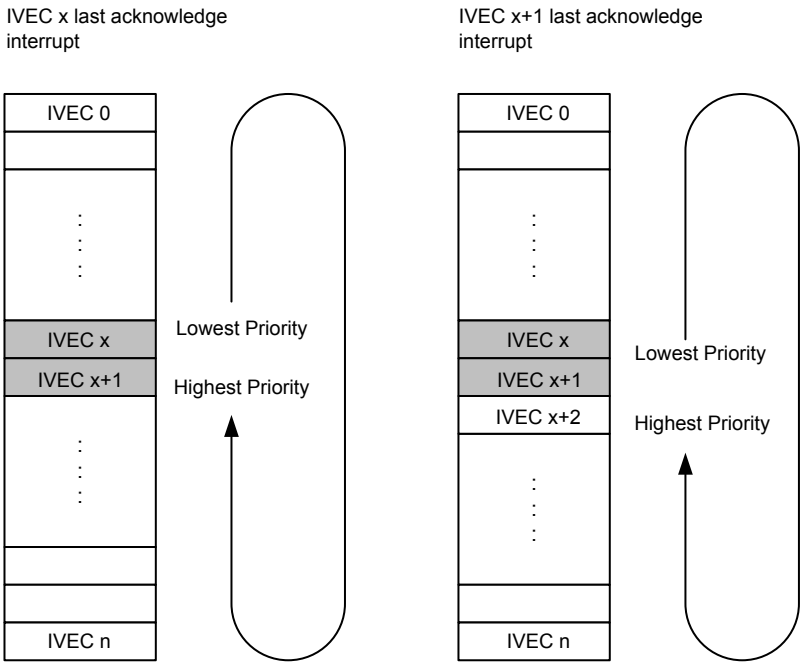
13.6.3.3. Round-Robin Scheduling

To avoid "starvation" for priority level 0 (LVL0) interrupt requests with static priority, i.e. some interrupts might never be served, the CPUINT offers round-robin scheduling for LVL0 interrupts.

Round-robin scheduling for LVL0 interrupt requests is enabled by writing a '1' to the Round-Robin Priority Enable bit in the Control A register (CPUINT_CTRLA.LVL0RR).

When round-robin scheduling is enabled, the interrupt vector address for the last acknowledged LVL0 interrupt will have the lowest priority the next time one or more LVL0 interrupts are requested.

Figure 13-4. Round-Robin Scheduling



13.6.4. Additional Features

13.6.4.1. Compact Vector Table

The Compact Vector Table (CVT) is a feature to allow for writing of compact code. When CVT is enabled by writing a '1' to the CVT bit in the Control A register (CPUINT_CTRLA.CVT), the vector table is three times the size of the interrupt vector.

When CVT is enabled, non-maskable interrupts (NMI), priority level 1 (LVL1) interrupt and priority level 0 (LVL0) interrupts each have one vector: NMI has vector address 1, LVL1 interrupt has vector address 2, and all the LVL0 interrupts requests share the vector address 3.

This feature is most suitable for applications using a small number of interrupt generators.

13.6.5. Events

Not applicable.

13.6.6. Sleep Mode Operation

Not applicable.

13.6.7. Configuration Change Protection

This peripheral has registers that are under Configuration Change Protection (CCP). In order to write to these, a certain key must be written to the CPU_CCP register first, followed by a write access to the protected bits within four CPU instructions.

It is possible to try writing to these registers any time, but the values are not altered.

The following registers are under CCP:

Table 13-3. INTCTRL - Registers under Configuration Change Protection

Register	Key
CPUINT_CTRLA.IVSEL	IOREG
CPUINT_CTRLA.CVT	IOREG

Related Links

Sequence for Write Operation to Configuration Change Protected I/O Registers on page 46

13.7. Register Summary

Offset	Name	Bit Pos.								
0x00	CTRLA	7:0		IVSEL	CVT					LVL0RR
0x01	STATUS	7:0	NMIEX						LVL1EX	LVL0EX
0x02	LVL0PRI	7:0	LVL0PRI[7:0]							
0x03	LVL1VEC	7:0	LVL1VEC[7:0]							

13.8. Register Description

13.8.1. Control A

Name: CTRLA
Offset: 0x00
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
		IVSEL	CVT					LVL0RR
Access		R/W	R/W					R/W
Reset		0	0					0

Bit 6 – IVSEL: Interrupt Vector Select
If boot section is defined, it will be placed before application section. The actual start address of the application section is determined by the BOOTEND Fuse.

This bit is protected by the Configuration Change Protection mechanism.

Value	Description
0	Interrupt vectors are placed at the start of the application section of the Flash.
1	Interrupt vectors are placed at the start of the boot section of the Flash.

Bit 5 – CVT: Compact Vector Table
This bit is protected by the Configuration Change Protection mechanism.

Value	Description
0	Compact Vector Table function is disabled
1	Compact Vector Table function is enabled

Bit 0 – LVL0RR: Round-Robin Priority Enable

Value	Description
0	Priority is fixed for priority level 0 interrupt requests: The lowest interrupt vector address has highest priority.
1	Round Robin priority scheme is enabled for low-level interrupts.

13.8.2. Status

Name: STATUS
Offset: 0x01
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	NMIEX						LVL1EX	LVL0EX
Access	R						R	R
Reset	0						0	0

Bit 7 – NMIEX: Non-Maskable Interrupt Executing
This flag is set if a non-maskable interrupt is executing. The flag is cleared when returning (RETI) from the interrupt handler.

Bit 1 – LVL1EX: Level 1 Interrupt Executing
This flag is set when a priority level 1 interrupt is executing, or when the interrupt handler has been interrupted by an NMI. The flag is cleared when returning (RETI) from the interrupt handler.

Bit 0 – LVL0EX: Level 0 Interrupt Executing
This flag is set when a priority level 0 interrupt is executing, or when the interrupt handler has been interrupted by a priority level 1 interrupt or an NMI. The flag is cleared when returning (RETI) from the interrupt handler.

13.8.3. Interrupt Priority Level 0

Name: LVL0PRI
Offset: 0x02
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	LVL0PRI[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – LVL0PRI[7:0]: Interrupt Priority Level 0
When Round Robin is enabled (LVL0RR bit in CPUINT.CTRLA is '1'), this bit field stores the vector of the last acknowledged priority level 0 (LVL0) interrupt. The stored vector will have the lowest priority next time one or more LVL0 interrupts are pending.

If Round Robin is disabled (LVL0RR in CPUINT.CTRLA is '0'), the vector address based priority scheme (lowest address has highest priority) is governing the priorities of LVL0 interrupt requests.

This bit field can be written to change the priority queue. If a system Reset is asserted, the lowest interrupt vector address will have highest priority within the LVL0.

13.8.4. Interrupt Vector with Priority Level 1

Name: LVL1VEC
Offset: 0x03
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	LVL1VEC[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – LVL1VEC[7:0]: Interrupt Vector with Priority Level 1
This bit field contains the address of the single vector with priority level 1 (LVL1).

If this bit field has the value 0x00, no vector has LVL1. Consequently, the LVL1 interrupt is disabled.

14. EVSYS - Event System

14.1. Overview

The Event System (EVSYS) enables direct peripheral-to-peripheral communication and signaling. It allows a change in one peripheral's state to trigger actions in other peripherals. It is designed to provide short and predictable response times between peripherals. It allows for autonomous peripheral control and interaction without the use of interrupts and CPU resources, and is thus a powerful tool for reducing the complexity, size, and execution time of application code. It also allows for synchronized timing of actions in several peripheral modules.

A change in a peripheral's state is referred to as an Event, and usually corresponds to the peripheral's interrupt conditions. Events can be directly passed to other peripherals using a dedicated routing network called the Event routing network. How Events are routed and used by the peripherals is configured in software.

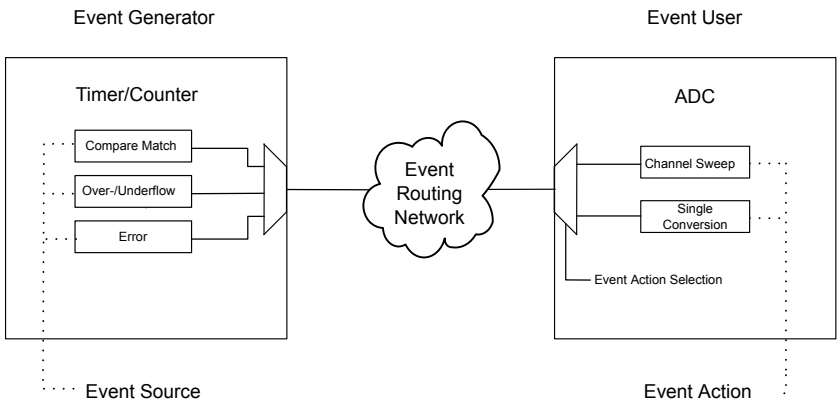
The Event System can directly connect analog and digital converters, analog comparators, I/O port pins, the real-time counter, timer/counters, and the configurable custom logic peripheral. Events can also be generated from software and the peripheral clock.

14.2. Features

- System for direct peripheral-to-peripheral communication and signaling
- Peripherals can directly send, receive, and react to peripheral events
- Short and guaranteed response time
- 4 asynchronous- and 2 synchronous- Event channels for up to 6 different and parallel signal paths and configurations
- Events can be sent and/or received by most peripherals, clock system, and software
- Works in active mode and standby sleep mode

14.3. Block Diagram

Figure 14-1. Example of Event Source, Generator, User, and Action



14.4. Signal Description

Not applicable.

14.5. Product Dependencies

In order to use this peripheral, other parts of the system must be configured correctly, as described below.

Table 14-1. EVSYS Product Dependencies

Dependency	Applicable	Peripheral
Clocks	Yes	CLKCTRL
I/O Lines and Connections	No	-
Interrupts	No	-
Events	No	-
Debug	Yes	UPDI

Related Links

Clocks on page 110

Debug Operation on page 110

14.5.1. Clocks

The EVSYS uses the peripheral clock for I/O registers and strobes. When set up correctly, the routing network can be used also in sleep modes without any clock. Strobe registers will not work in sleep modes where the peripheral clock is halted.

Related Links

CLKCTRL - Clock Controller on page 68

14.5.2. I/O Lines and Connections

Not applicable.

14.5.3. Interrupts

Not applicable.

14.5.4. Events

Not applicable.

14.5.5. Debug Operation

This peripheral is unaffected by entering debug mode.

Note: When the CPU is halted, the strobe registers are not cleared automatically.

Related Links

UPDI - Unified Program Debug Interface on page 490

14.6. Functional Description

14.6.1. Principle of Operation

The Event System consists of several channels which route the internal Events from peripherals (Event generators) to other internal peripherals or IO pins (Event users). Each Event generator can be selected as source for multiple channels, but a channel cannot be set to use multiple Event generators at the same time.

A channel path can be configured in asynchronous or synchronous mode of operation. The mode of operation must be selected based on the requirements of the application.

14.6.2. Basic Operation

14.6.2.1. Initialization

Before enabling Events routing within the system, the Event Users Multiplexer and Event Channels must be configured.

Related Links

User Multiplexer Setup on page 111
Event System Channel on page 111

14.6.2.2. User Multiplexer Setup

The user multiplexer defines the channel to be connected to which Event user. Each user multiplexer is dedicated to one Event user. A user multiplexer receives all Event channel outputs that are supported and can be configured to select one of these channels. A user which does not support asynchronous Events can not be configured to use any of the asynchronous Event channels. Users which support asynchronous Events also support synchronous Events. Default setup of all user multiplexers is off.

The user multiplexers are configured by writing to the respective registers:

- Event users supporting synchronous and asynchronous Events are configured by writing to the respective Asynchronous User Channel Input Selection n register (EVSYS_ASYNCUSERn.ASYNCUSER).
- The users of synchronous-only Events are configured by writing to the respective Synchronous User Channel Input Selection n register (EVSYS_SYNCUSERn.SYNCUSER).

14.6.2.3. Event System Channel

An Event channel can select one Event from a list of Event generators. The channels are divided into either asynchronous generators or synchronous generators.

The asynchronous Event channels are configured by writing to the respective Asynchronous Channel n Input Selection register (EVSYS_ASYNCCHn.ASYNCCH).

The synchronous Event channels are configured by writing to the respective Synchronous Channel n Input Selection register (EVSYS_SYNCCHn.SYNCCH).

14.6.2.4. Event Generators

Each Event channel can receive the Events from several Event generators. For details on Event generation, refer to the documentation of the corresponding peripheral.

The channel Event generator is selected by the respective channel registers (EVSYS_ASYNCCHn, EVSYS_SYNCCHn) By default, the channels are not connected to any Event generator.
Note: Some Event generators are available for several Event channels.

14.6.2.5. Software Event

A software Event can be initiated on a channel by writing a '1' to the respective Strobe bit in the appropriate Channel Strobe register:

- Software Events on asynchronous channel m are initiated by writing a '1' to the ASYNCSTROBEm bit in the Asynchronous Channel n Strobe register (EVSYS_ASYNCSTROBE.ASYNCSTROBEm).
- Software Events on synchronous channel n are initiated by writing a '1' to the SYNCSTROBEn bit in the Synchronous Channel n Strobe register (EVSYS_SYNCSTROBE.SYNCSTROBEn).

Then, the software Event is serviced like any Event generator; i.e., when the bit is written to '1', an Event will be generated on the respective channel.

14.6.3. Interrupts

Not applicable.

14.6.4. Sleep Mode Operation

When configured, the Event System will work in all sleep modes. One exception are software Events which require a system clock.

14.6.5. Synchronization

Asynchronous Events are synchronized and handled within the Event users supporting this. For modules that do not support asynchronous Events, only use synchronous Event channels can be used to set up their Event generator.

14.6.6. Configuration Change Protection

Not applicable.

Related Links

Sequence for Write Operation to Configuration Change Protected I/O Registers on page 46

14.7. Register Summary

Offset	Name	Bit Pos.									
0x00	ASYNCSTROBE	7:0	ASYNCSTROBE[7:0]								
0x01	SYNCSTROBE	7:0	SYNCSTROBE[7:0]								
0x02	ASYNCCH0	7:0	ASYNCCH[7:0]								
0x03	ASYNCCH1	7:0	ASYNCCH[7:0]								
0x04	ASYNCCH2	7:0	ASYNCCH[7:0]								
0x05	ASYNCCH3	7:0	ASYNCCH[7:0]								
0x06 ... 0x09	Reserved										
0x0A	SYNCCH0	7:0	SYNCCH[7:0]								
0x0B	SYNCCH1	7:0	SYNCCH[7:0]								
0x0C ... 0x11	Reserved										
0x12	ASYNCUSER0	7:0	ASYNCUSER[7:0]								
0x13	ASYNCUSER1	7:0	ASYNCUSER[7:0]								
0x14	ASYNCUSER2	7:0	ASYNCUSER[7:0]								
0x15	ASYNCUSER3	7:0	ASYNCUSER[7:0]								
0x16	ASYNCUSER4	7:0	ASYNCUSER[7:0]								
0x17	ASYNCUSER5	7:0	ASYNCUSER[7:0]								
0x18	ASYNCUSER6	7:0	ASYNCUSER[7:0]								
0x19	ASYNCUSER7	7:0	ASYNCUSER[7:0]								
0x1A	ASYNCUSER8	7:0	ASYNCUSER[7:0]								
0x1B	ASYNCUSER9	7:0	ASYNCUSER[7:0]								
0x1C	ASYNCUSER10	7:0	ASYNCUSER[7:0]								
0x1D ... 0x21	Reserved										
0x22	SYNCUSER0	7:0	SYNCUSER[7:0]								
0x23	SYNCUSER1	7:0	SYNCUSER[7:0]								

14.8. Register Description

14.8.1. Asynchronous Channel Strobe

The CPU can override the Event channel lines by writing to the respective bit in the strobe register I/O location. The CPU override lasts for one system clock cycle. The write to the strobe register will invert the current value on the Event channel for one system clock cycle. Set the channel to OFF state to guarantee a positive pulse.

Name: ASYNCSTROBE
Offset: 0x00
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	ASYNCSTROBE[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – ASYNCSTROBE[7:0]: Asynchronous Channel Strobe
If the strobe register location is written, each Event channel will be inverted for one system clock cycle, i.e. a single Event is generated.

14.8.2. Synchronous Channel Strobe

The CPU can override the Event channel lines by writing to the respective bit in the strobe register I/O location. The CPU override lasts for one system clock cycle. The write to strobe register will invert the current value on the Event channel for one system clock cycle. Set the channel to OFF state to guarantee a positive pulse.

Name: SYNCSTROBE
Offset: 0x01
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	SYNCSTROBE[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – SYNCSTROBE[7:0]: Synchronous Channel Strobe
If the strobe register location is written, each events channel will be inverted for one system clock cycle, i.e. a single event is generated.

14.8.3. Asynchronous Channel n Input Selection

Name: ASYNCCH0, ASYNCCH1, ASYNCCH2, ASYNCCH3
Offset: 0x02 + n*0x01 [n=0..3]
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	ASYNCCH[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – ASYNCCH[7:0]: Asynchronous Channel Input Selection

Table 14-2. ASYNCCH0

Value	Description
0x00	OFF
0x01	CCL_LUT0
0x02	CCL_LUT1
0x03	AC0_OUT
0x04	TCD0_CMPBCLR
0x05	TCD0_CMPASET
0x06	TCD0_CMPBSET
0x07	TCD0_PROGEV
0x08	RTC_OVF
0x09	RTC_CMP
0x0A	PORTA0
0x0B	PORTA1
0x0C	PORTA2
0x0D	PORTA3
0x0E	PORTA4
0x0F	PORTA5
0x10	PORTA6
0x11	PORTA7
0x12	UPDI
Other	Reserved

Table 14-3. ASYNCCH1

Value	Description
0x00	OFF
0x01	CCL_LUT0
0x02	CCL_LUT1
0x03	AC0_OUT
0x04	TCD0_CMPBCLR
0x05	TCD0_CMPASET
0x06	TCD0_CMPBSET
0x07	TCD0_PROGEV
0x08	RTC_OVF
0x09	RTC_CMP
0x0A	PORTB0
0x0B	PORTB1
0x0C	PORTB2
0x0D	PORTB3
0x0E	PORTB4
0x0F	PORTB5
0x10	PORTB6
0x11	PORTB7
Other	Reserved

Table 14-4. ASYNCCH2

Value	Description
0x00	OFF
0x01	CCL_LUT0
0x02	CCL_LUT1
0x03	AC0_OUT
0x04	TCD0_CMPBCLR
0x05	TCD0_CMPASET
0x06	TCD0_CMPBSET
0x07	TCD0_PROGEV
0x08	RTC_OVF
0x09	RTC_CMP

Value	Description
0x0A	PORTC0
0x0B	PORTC1
0x0C	PORTC2
0x0D	PORTC3
0x0E	PORTC4
0x0F	PORTC5
Other	Reserved

Table 14-5. ASYNCCH3

Value	Description
0x00	OFF
0x01	CCL_LUT0
0x02	CCL_LUT1
0x03	AC0_OUT
0x04	TCD0_CMPBCLR
0x05	TCD0_CMPASET
0x06	TCD0_CMPBSET
0x07	TCD0_PROGEV
0x08	RTC_OVF
0x09	RTC_CMP
0x0A	PIT_DIV8192
0x0B	PIT_DIV4096
0x0C	PIT_DIV2048
0x0D	PIT_DIV1024
0x0E	PIT_DIV512
0x0F	PIT_DIV256
0x10	PIT_DIV128
0x11	PIT_DIV64
Other	Reserved

14.8.4. Synchronous Channel n Input Selection

Name: SYNCCH0, SYNCCH1
Offset: 0x0A + n*0x01 [n=0..1]
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	SYNCCH[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – SYNCCH[7:0]: Synchronous Channel Input Selection

Table 14-6. SYNCCH0

Value	Description
0x00	OFF
0x01	TCB0
0x02	TCA0_OVF_LUNF
0x03	TCA0_HUNF
0x04	TCA0_CMP0
0x05	TCA0_CMP1
0x06	TCA0_CMP2
0x07	PORTC0
0x08	PORTC1
0x09	PORTC2
0x0A	PORTC3
0x0B	PORTC4
0x0C	PORTC5
0x0D	PORTA0
0x0E	PORTA1
0x0F	PORTA2
0x10	PORTA3
0x11	PORTA4
0x12	PORTA5
0x13	PORTA6
0x14	PORTA7
Other	Reserved

Table 14-7. SYNCCH1

Value	Description
0x00	OFF
0x01	TCB0
0x02	TCA0_OVF_LUNF
0x03	TCA0_HUNF
0x04	TCA0_CMP0
0x05	TCA0_CMP1
0x06	TCA0_CMP2
0x07	Reserved
0x08	PORTB0
0x09	PORTB1
0x0A	PORTB2
0x0B	PORTB3
0x0C	PORTB4
0x0D	PORTB5
0x0E	PORTB6
0x0F	PORTB7
Other	Reserved

14.8.5. Asynchronous User Channel n Input Selection

Name: ASYNCUSER0, ASYNCUSER1, ASYNCUSER2, ASYNCUSER3, ASYNCUSER4, ASYNCUSER5, ASYNCUSER6, ASYNCUSER7, ASYNCUSER8, ASYNCUSER9, ASYNCUSER10
Offset: 0x12 + n*0x01 [n=0..10]
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	ASYNCUSER[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – ASYNCUSER[7:0]: Asynchronous User Channel Selection

Table 14-8. ASYNCUSER0 - TCB0

Value	Description
0x0	OFF
0x1	SYNCCH0
0x2	SYNCCH1
0x3	ASYNCCH0
0x4	ASYNCCH1
0x5	ASYNCCH2
0x6	ASYNCCH3

Table 14-9. ASYNCUSER1 - ADC0

Value	Description
0x0	OFF
0x1	SYNCCH0
0x2	SYNCCH1
0x3	ASYNCCH0
0x4	ASYNCCH1
0x5	ASYNCCH2
0x6	ASYNCCH3

Table 14-10. ASYNCUSER2 - CCL LUT0 Event 0

Value	Description
0x0	OFF
0x1	SYNCCH0

Value	Description
0x2	SYNCCH1
0x3	ASYNCCH0
0x4	ASYNCCH1
0x5	ASYNCCH2
0x6	ASYNCCH3

Table 14-11. ASYNCUSER3 - CCL LUT1 Event 0

Value	Description
0x0	OFF
0x1	SYNCCH0
0x2	SYNCCH1
0x3	ASYNCCH0
0x4	ASYNCCH1
0x5	ASYNCCH2
0x6	ASYNCCH3

Table 14-12. ASYNCUSER4 - CCL LUT0 Event 1

Value	Description
0x0	OFF
0x1	SYNCCH0
0x2	SYNCCH1
0x3	ASYNCCH0
0x4	ASYNCCH1
0x5	ASYNCCH2
0x6	ASYNCCH3

Table 14-13. ASYNCUSER5 - CCL LUT1 Event 1

Value	Description
0x0	OFF
0x1	SYNCCH0
0x2	SYNCCH1
0x3	ASYNCCH0
0x4	ASYNCCH1

Value	Description
0x5	ASYNCCH2
0x6	ASYNCCH3

Table 14-14. ASYNCUSER6 - TCD0 Event 0

Value	Description
0x0	OFF
0x1	SYNCCH0
0x2	SYNCCH1
0x3	ASYNCCH0
0x4	ASYNCCH1
0x5	ASYNCCH2
0x6	ASYNCCH3

Table 14-15. ASYNCUSER7 - TCD0 Event 1

Value	Description
0x0	OFF
0x1	SYNCCH0
0x2	SYNCCH1
0x3	ASYNCCH0
0x4	ASYNCCH1
0x5	ASYNCCH2
0x6	ASYNCCH3

Table 14-16. ASYNCUSER8 - EVOUT0

Value	Description
0x0	OFF
0x1	SYNCCH0
0x2	SYNCCH1
0x3	ASYNCCH0
0x4	ASYNCCH1
0x5	ASYNCCH2
0x6	ASYNCCH3

Table 14-17. ASYNCUSER9 - EVOUT1

Value	Description
0x0	OFF
0x1	SYNCCH0
0x2	SYNCCH1
0x3	ASYNCCH0
0x4	ASYNCCH1
0x5	ASYNCCH2
0x6	ASYNCCH3

Table 14-18. ASYNCUSER10 - EVOUT2

Value	Description
0x0	OFF
0x1	SYNCCH0
0x2	SYNCCH1
0x3	ASYNCCH0
0x4	ASYNCCH1
0x5	ASYNCCH2
0x6	ASYNCCH3

14.8.6. Synchronous User Channel n Input Selection

Name: SYNCUSER0, SYNCUSER1
Offset: 0x22 + n*0x01 [n=0..1]
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	SYNCUSER[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – SYNCUSER[7:0]: Synchronous User Channel Selection

Table 14-19. SYNCUSER0 - TCA0

Value	Description
0x0	OFF
0x1	SYNCCH0
0x2	SYNCCH1

Table 14-20. SYNCUSER1 - USART0

Value	Description
0x0	OFF
0x1	SYNCCH0
0x2	SYNCCH1

15. PORTMUX - Port Multiplexer

15.1. Overview

The Port Multiplexer (PORTMUX) can either enable or disable functionality of pins, or change between default and alternative pin positions. This depends on the actual pin and property, and is described in detail in the register map of this peripheral.

15.2. Signal Description

Signal	Type	Description
EVOUT[2:0]	Digital output	Event output

Related Links

I/O Multiplexing and Considerations on page 19

15.3. Register Summary

Offset	Name	Bit Pos.								
0x00	CTRLA	7:0	EXTBRK		LUT1	LUT0		EVOUT2	EVOUT1	EVOUT0
0x01	CTRLB	7:0				TWI0		SPI0		USART0
0x02	CTRLC	7:0			TCA05	TCA04	TCA03	TCA02	TCA01	TCA00
0x03	CTRLD	7:0								TCB0

15.4. Register Description

15.4.1. Control A

Name: CTRLA
Offset: 0x00
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	EXTBRK		LUT1	LUT0		EVOUT2	EVOUT1	EVOUT0
Access	R/W		R/W	R/W		R/W	R/W	R/W
Reset	0		0	0		0	0	0

Bit 7 – EXTBRK: External Break Pin
Write this bit to '1' to select alternative output pin for external break.

Bit 5 – LUT1: CCL LUT 1 output
Write this bit to '1' to select alternative output pin for CCL LUT 1.

Bit 4 – LUT0: CCL LUT 0 output
Write this bit to '1' to select alternative output pin for CCL LUT 0.

Bit 2 – EVOUT2: Event Output 2
Write this bit to '1' to enable event output 2.

Bit 1 – EVOUT1: Event Output 1
Write this bit to '1' to enable event output 1.

Bit 0 – EVOUT0: Event Output 0
Write this bit to '1' to enable event output 0.

15.4.2. Control B

Name: CTRLB
Offset: 0x01
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
				TWI0		SPI0		USART0
Access				R/W		R/W		R/W
Reset				0		0		0

Bit 4 – TWI0: TWI 0 communication
Write this bit to '1' to select alternative communication pins for TWI 0.

Bit 2 – SPI0: SPI 0 communication
Write this bit to '1' to select alternative communication pins for SPI 0.

Bit 0 – USART0: USART 0 communication
Write this bit to '1' to select alternative communication pins for USART 0.

15.4.3. Control C

Name: CTRLC
Offset: 0x02
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
			TCA05	TCA04	TCA03	TCA02	TCA01	TCA00
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bit 5 – TCA05: TCA0 Waveform output 5
Write this bit to '1' to select alternative output pin for TCA0 waveform output 5 in split mode.
Note: Not applicable when TCA in normal mode.

Bit 4 – TCA04: TCA0 Waveform output 4
Write this bit to '1' to select alternative output pin for TCA0 waveform output 4 in split mode.
Note: Not applicable when TCA in normal mode.

Bit 3 – TCA03: TCA0 Waveform output 3
Write this bit to '1' to select alternative output pin for TCA0 waveform output 3 in split mode.
Note: Not applicable when TCA in normal mode.

.

Bit 2 – TCA02: TCA0 Waveform output 2
Write this bit to '1' to select alternative output pin for TCA0 waveform output 2.

In Split Mode, this bit controls output from compare channel C low byte.

Bit 1 – TCA01: TCA0 Waveform output 1
Write this bit to '1' to select alternative output pin for TCA0 waveform output 1.

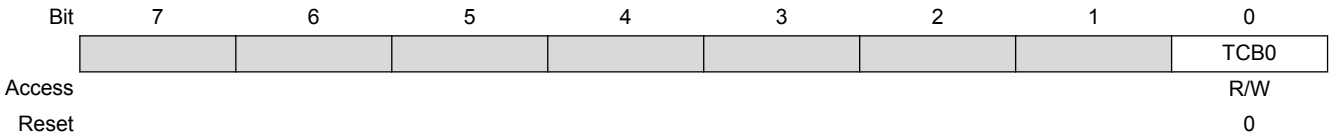
In Split Mode, this bit controls output from compare channel B low byte.

Bit 0 – TCA00: TCA0 Waveform output 0
Write this bit to '1' to select alternative output pin for TCA0 waveform output 0.

In Split Mode, this bit controls output from compare channel A low byte.

15.4.4. Control D

Name: CTRLD
Offset: 0x03
Reset: 0x00
Property: -



Bit	7	6	5	4	3	2	1	0
								TCB0
Access								R/W
Reset								0

Bit 0 – TCB0: TCB0 output
Write this bit to '1' to select alternative output pin for 16-bit timer/counter B 0.

16. PORT - I/O Pin Controller

16.1. Overview

This device has up to three instances of the I/O Pin Controller (PORT). Refer to the I/O Multiplexing table to see which pins are controlled by what instance. The offset of the PORT instances and of the corresponding Virtual PORT instances is listed in the Peripherals and Architecture section.

The PORT controls the I/O pins of the device. One port consists of up to eight port pins: pin 0 to 7. Each port pin can be configured as input or output with configurable driver and pull settings. They also implement synchronous and asynchronous input sensing with interrupts and events for selectable pin change conditions. Asynchronous pin-change sensing means that a pin change can wake the device from all sleep modes, including the modes where no clocks are running.

All functions are individual and configurable per pin. The pins have hardware read-modify-write (RMW) functionality for safe and correct change of drive value and/or pull resistor configuration. The direction of one port pin can be changed without unintentionally changing the direction of any other pin.

The PORT pin configuration also controls input and output selection of other device functions.
Note: Some peripherals may override the configuration set by the PORT controller.

Related Links

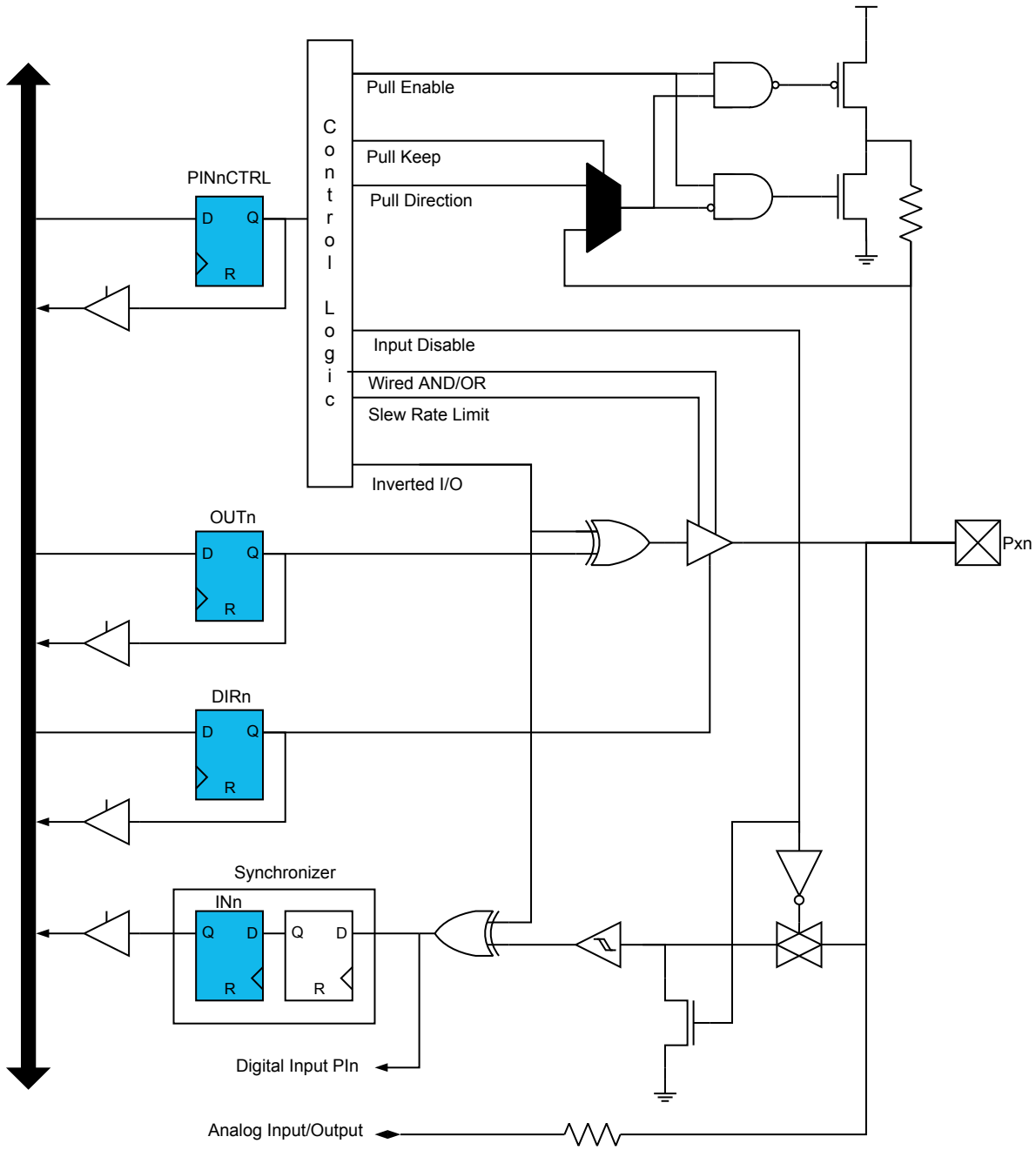
I/O Multiplexing and Considerations on page 19
Peripherals and Architecture on page 38

16.2. Features

- General purpose input and output pins with individual configuration
- Output driver with configurable inverted I/O and pullup.
- Input with synchronous and/or asynchronous sensing with interrupts and Events:
  - Sense both edges
  - Sense rising edges
  - Sense falling edges
  - Sense low level
- Asynchronous pin change sensing that can wake the device from all sleep modes
- Efficient and safe access to PORT pins
  - Hardware read-modify-write through dedicated toggle/clear/set registers
  - Mapping of often-used PORT registers into bit-accessible I/O memory space (Virtual Ports)

16.3. Block Diagram

Figure 16-1. PORT Block Diagram



16.4. Signal Description

Signal	Type	Description
EXTINT	Digital input	External interrupt - available on all I/O pins

Related Links

I/O Multiplexing and Considerations on page 19

16.5. Product Dependencies

In order to use this peripheral, other parts of the system must be configured correctly, as described below.

Table 16-1. PORT Product Dependencies

Dependency	Applicable	Peripheral
Clocks	Yes	CLKCTRL
I/O Lines and Connections	No	-
Interrupts	Yes	CPUINT
Events	No	-
Debug	No	-

Related Links

Clocks on page 86
Interrupts on page 54

16.5.1. Clocks

This peripheral depends on the peripheral clock.

Related Links

CLKCTRL - Clock Controller on page 68
Product Dependencies on page 98

16.5.2. I/O Lines and Connections

Not applicable.

16.5.3. Interrupts

Using the interrupts of this peripheral requires the Interrupt Controller to be configured first.

Related Links

CPUINT - CPU Interrupt Controller on page 97
SREG on page 51
Interrupts on page 136

16.5.4. Events

Not applicable.

16.5.5. Debug Operation

This peripheral is unaffected by entering debug mode.

16.6. Functional Description

16.6.1. Principle of Operation

The I/O pins of the device are controlled by PORT peripheral registers. Each of the port pins has a corresponding bit in the Data Direction (DIR) and Data Output Value (OUT) registers to enable that pin as an output and to device the output state.

The direction of each pin in a pin group is configured by the DIR register. If a bit in DIR is set to '1', the corresponding pin is configured as an output pin. If a bit in DIR is set to '0', the corresponding pin is configured as an input pin.

When the direction is set as output, the corresponding bit in the OUT register will set the level of the pin. If bit n in OUT is written to '1', pin n is driven HIGH. If bit n in OUT is written to '0', pin n is driven LOW. Pin configuration can be set by writing to the Pin n Control registers (PORT_PINnCTRL) with n=0..7 representing the bit position.

The Data Input Value (IN) is set as the input value of a PORT pin with resynchronization to the Main Clock. To reduce power consumption, these input synchronizers are clocked only when the value of the Input Sense Configuration bit field (PORT_PINnCTRL.ISC) is not INPUT_DISABLE. The value of the pin can always be read, whether the pin is configured as input or output.

16.6.2. Basic Operation

16.6.2.1. Initialization

After Reset, all standard function device I/O pads are connected to the PORT with outputs tri-stated and input buffers enabled, even if there is no clock running.

Note: For best power consumption, leave unused pins in their default configuration after Reset.

Specific pins, such as those used for connecting a debugger, may be configured differently, as required by their special function.

16.6.2.2. Basic Functions

Each I/O pin Pxn can be controlled by the registers in PORT x. Each pin group x has its own set of PORT registers, the base address of the register set for pin n is at the byte address $\mathrm{PORT} + 0\mathrm{x}10 + n$. The index within that register set is n.

To use pin number n as an output, write bit n of the DIR register to '1'. This can also be done by writing bit n in the DIRSET register to '1' - this will avoid disturbing the configuration of other pins in that group. The n^{th} bit in the OUT register must be written to the desired output value.

Similarly, writing an OUTSET bit to '1' will set the corresponding bit in the OUT register to '1'. Writing a bit in OUTCLR to '1' will set that bit in OUT to zero. Writing a bit in OUTTGL or IN to '1' will toggle that bit in OUT.

To use pin n as an input, bit n in the DIR register must be written to '0'. This can also be done by writing bit n in the DIRCLR register to '1' - this will avoid disturbing the configuration of other pins in that group. The input value can be read from bit n in register IN as long as the ISC is not set to INPUT DISABLE.

In a similar way as OUTTGL can be used to toggle the value on an output DIRTGL can be used to toggle a bit in DIR.

16.6.3. Additional Features

16.6.3.1. Virtual Ports

The Virtual Port registers map the most frequently used regular Port registers into the bit-accessible I/O space. Writing to the Virtual Port registers has the same effect as writing to the regular registers, but allows for memory-specific instructions, such as bit-manipulation instructions, which are not valid for the extended I/O memory space where the regular Port registers reside.

Table 16-2. Virtual Port Mapping

Regular Port Register	Mapped to Virtual Port Register
PORT_DIR	VPORT_DIR
PORT_OUT	VPORT_OUT
PORT_IN	VPORT_IN
PORT_INTFLAG	VPORT_INTFLAG

Related Links

I/O Multiplexing and Considerations on page 19
Peripherals and Architecture on page 38

16.6.4. Pin Configuration

The Pin n Configuration register (PORT_PINnCTRL) is used to configure inverted I/O, pullup, and input sensing of a pin.

All input and output on the respective pin n can be inverted by writing a '1' to the Inverted I/O Enable bit (PORT_PINnCTRL.INVEN).
Note: A small number of peripherals will override this setting.

Toggling the .INVEN bit causes an edge on the pin, which can be detected by all peripherals using this pin, and is seen by interrupts or Events if enabled.

Pullup of pin n is enabled by writing a '1' to the Pullup Enable bit (PORT_PINnCTRL.PULLUPEN).

Changes of the signal on a pin can trigger an interrupt. The exact conditions are defined by writing to the Input/Sense bit field (PORT_PINnCTRL.ISC).

When setting or changing interrupt settings, take these points into account:

- If an .INVEN bit is toggled in the same cycle as the interrupt setting, the edge caused by the inversion toggling may not cause an interrupt request.
- If an input is disabled while synchronizing an interrupt, that interrupt may be requested on re-enabling the input, even if it is re-enabled with a different interrupt setting.
- If the interrupt setting is changed while synchronizing an interrupt, that interrupt may be not accepted.

16.6.5. Interrupts

Table 16-3. Available Interrupt Vectors and Sources

Offset	Name	Vector Description	Conditions
0x00	PORTx	PORT A, B, C interrupt	PORT_INTFLAGS.INTn is raised as configured in the Pin n Control register (PORT_PINnCTRL.ISC).

Each PORT pin n can be configured as interrupt source. Each interrupt can be individually enabled or disabled by writing to .ISC.

When an interrupt condition occurs, the corresponding Interrupt Flag is set in the Interrupt Flags register of the peripheral (*peripheral*_INTFLAGS).

An interrupt request is generated when the corresponding interrupt is enabled and the Interrupt Flag is set. The interrupt request remains active until the Interrupt Flag is cleared. See the peripheral's INTFLAGS register for details on how to clear Interrupt Flags.

Note: Interrupt requests are only generated when Interrupts are enabled globally.

Asynchronous Sensing Pin Properties

Table 16-4. Behavior Comparison of Fully/Partly Asynchronous Sense Pin

	Pin Behavior With/Without Full Asynchronous Sense Support	
Property	**Without**	**With**
Minimum pulse width to trigger interrupt	at least one system clock cycle	less than a system clock cycle
Waking the device from sleep modes	Only from sleep modes with stopped Main Clock by using low-priority interrupts or pin change interrupts	From any sleep mode using any interrupt setting
Interrupt "dead time"	no new interrupt for three cycles after the previous	no such limitation
Minimum Wakeup pulse length	the value on pad should be kept until the system clock has restarted	No such limitation
Writing a '1' to a cleared flag in the same cycle as hardware tries to set the flag	the interrupt is lost	interrupt flag will be set one cycle delayed

Related Links

AVR CPU on page 40
SREG on page 51

16.6.6. Sleep Mode Operation

With the exception of interrupts and input synchronization, all pin configurations are independent of sleep mode. Peripherals connected to the Ports can be affected by sleep modes, refer to the respective peripherals' documentation.

The PORT peripheral will always use the Main Clock. Input synchronization will halt when this clock stops.

16.6.7. Synchronization

Not applicable.

16.6.8. Configuration Change Protection

Not applicable.

16.7. Register Summary - Ports

Offset	Name	Bit Pos.								
0x00	DIR	7:0	DIR[7:0]							
0x01	DIRSET	7:0	DIRSET[7:0]							
0x02	DIRCLR	7:0	DIRCLR[7:0]							
0x03	DIRTGL	7:0	DIRTGL[7:0]							
0x04	OUT	7:0	OUT[7:0]							
0x05	OUTSET	7:0	OUTSET[7:0]							
0x06	OUTCLR	7:0	OUTCLR[7:0]							
0x07	OUTTGL	7:0	OUTTGL[7:0]							
0x08	IN	7:0	IN[7:0]							
0x09	INTFLAGS	7:0	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0
0x0A ... 0x0F	Reserved									
0x10	PINCTRL0	7:0	INVEN				PULLUPEN	ISC[2:0]		
0x11	PINCTRL1	7:0	INVEN				PULLUPEN	ISC[2:0]		
0x12	PINCTRL2	7:0	INVEN				PULLUPEN	ISC[2:0]		
0x13	PINCTRL3	7:0	INVEN				PULLUPEN	ISC[2:0]		
0x14	PINCTRL4	7:0	INVEN				PULLUPEN	ISC[2:0]		
0x15	PINCTRL5	7:0	INVEN				PULLUPEN	ISC[2:0]		
0x16	PINCTRL6	7:0	INVEN				PULLUPEN	ISC[2:0]		
0x17	PINCTRL7	7:0	INVEN				PULLUPEN	ISC[2:0]		

16.8. Register Description - Ports

16.8.1. Data Direction

Name: DIR
Offset: 0x00
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	DIR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – DIR[7:0]: Data Direction
This bit field selects the data direction for the individual pins n of the Port.
Writing a '1' to PORT.DIR[n] configures and enables pin n as output pin.

Writing a '0' to PORT.DIR[n] configures pin n as input pin. It can be configured by writing to the ISC bit in PORT.PINnCTRL.

16.8.2. Data Direction Set

Name: DIRSET
Offset: 0x01
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	DIRSET[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – DIRSET[7:0]: Data Direction Set
This bit field can be used instead of a read-modify-write to set individual pins as output. Writing a '1' to DIRSET[n] will set the corresponding PORT.DIR[n] bit.

Reading this bit field will always return the value of PORT.DIR.

16.8.3. Data Direction Clear

Name: DIRCLR
Offset: 0x02
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	DIRCLR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – DIRCLR[7:0]: Data Direction Clear
This register can be used instead of a read-modify-write to configure individual pins as input. Writing a '1' to DIRCLR[n] will clear the corresponding bit in PORT.DIR.

Reading this bit field will always return the value of PORT.DIR.

16.8.4. Data Direction Toggle

Name: DIRTGL
Offset: 0x03
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	DIRTGL[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – DIRTGL[7:0]: Data Direction Toggle
This bit field can be used instead of a read-modify-write to toggle the direction of individual pins.

Writing a '1' to DIRTGL[n] will toggle the corresponding bit in PORT.DIR.

Reading this bit field will always return the value of PORT.DIR.

16.8.5. Output Value

Name: OUT
Offset: 0x04
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	OUT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – OUT[7:0]: Output Value
This bit field defines the data output value for the individual pins n of the port.
If OUT[n] is written to '1', pin n is driven high.

If OUT[n] is written to '0', pin n is driven low.

In order to have any effect, the pin direction must be configured

16.8.6. Output Value Set

Name: OUTSET
Offset: 0x05
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	OUTSET[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – OUTSET[7:0]: Output Value Set
This bit field can be used instead of a read-modify-write to set the output value of individual pins to '1'. Writing a '1' to OUTSET[n] will set the corresponding bit in PORT.OUT.

Reading this bit field will always return the value of PORT.OUT.

16.8.7. Output Value Clear

Name: OUTCLR
Offset: 0x06
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	OUTCLR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – OUTCLR[7:0]: Output Value Clear
This register can be used instead of a read-modify-write to clear the output value of individual pins to '0'. Writing a '1' to OUTSET[n] will clear the corresponding bit in PORT.OUT.

Reading this bit field will always return the value of PORT.OUT.

16.8.8. Output Value Toggle

Name: OUTTGL
Offset: 0x07
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	OUTTGL[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – OUTTGL[7:0]: Output Value Toggle
This register can be used instead of a read-modify-write to toggle the output value of individual pins. Writing a '1' to OUTSET[n] will toggle the corresponding bit in PORT.OUT.

Reading this bit field will always return the value of PORT.OUT.

16.8.9. Input Value

Name: IN
Offset: 0x08
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	IN[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – IN[7:0]: Input Value
This register shows the value present on the pins if the digital input driver is enabled. IN[n] shows the value of pin n of the Port. The input is not sampled and cannot be read if the digital input buffers are disabled.

16.8.10. Interrupt Flags

Name: INTFLAGS
Offset: 0x09
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6, 7 – INTn: Interrupt Pin n Flag
The INTn Flag is set when a pin change/state matches the pin's input sense configuration. Writing a '1' to a flag's bit location will clear the flag.

For enabling and executing the interrupt, refer to the interrupt level description.

16.8.11. Pin n Control

Name: PINCTRL0, PINCTRL1, PINCTRL2, PINCTRL3, PINCTRL4, PINCTRL5, PINCTRL6, PINCTRL7
Offset: 0x10 + n*0x01 [n=0..7]
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	INVEN				PULLUPEN	ISC[2:0]		
Access	R/W				R/W	R/W	R/W	R/W
Reset	0				0	0	0	0

Bit 7 – INVEN: Inverted I/O Enable

Value	Description
0	I/O on pin n not inverted
1	I/O on pin n inverted

Bit 3 – PULLUPEN: Pullup Enable

Value	Description
0	Pullup disabled for pin n
1	Pullup enabled for pin n

Bits 2:0 – ISC[2:0]: Input/Sense Configuration
These bits configure the input and sense configuration of pin n. The sense configuration decides how the pin can trigger port interrupts. If the input buffer is disabled, the input cannot be read in the IN register.

Value	Name	Description
0x0	INTDISABLE	Interrupt disabled but input buffer enabled
0x1	BOTHEDGES	Sense both edges
0x2	RISING	Sense rising edge
0x3	FALLING	Sense falling edge
0x4	INPUT_DISABLE	Digital input buffer disabled
0x5	LEVEL	Sense low level
other	-	Reserved

16.9. Register Summary - Virtual Ports

Offset	Name	Bit Pos.								
0x00	DIR	7:0	DIR[7:0]							
0x01	OUT	7:0	OUT[7:0]							
0x02	IN	7:0	IN[7:0]							
0x03	INTFLAGS	7:0	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0

16.10. Register Description - Virtual Ports

16.10.1. Data Direction

Name: DIR
Offset: 0x00
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	DIR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – DIR[7:0]: Data Direction
This bit field selects the data direction for the individual pins in the Port. When a Port is mapped as virtual, accessing this bit field is identical to accessing the actual PORT.DIR register for the Port.

16.10.2. Output Value

Name: OUT
Offset: 0x01
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	OUT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – OUT[7:0]: Output Value
This bit field selects the data output value for the individual pins in the Port. When a Port is mapped as virtual, accessing this bit field is identical to accessing the actual PORT.OUT register for the Port.

16.10.3. Input Value

Name: IN
Offset: 0x02
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	IN[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – IN[7:0]: Input Value
This bit field holds the value present on the pins if the digital input buffer is enabled. This bit field sets the data direction for the individual pins in the Port. When a Port is mapped as virtual, accessing this bit field is identical to accessing the actual PORT.IN register for the Port.

16.10.4. Interrupt Flag

Name: INTFLAGS
Offset: 0x03
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6, 7 – INT0, INT1, INT2, INT3, INT4, INT5, INT6, INT7: Interrupt Pin n Flag
The INT[n] flag is set when a pin change/state matches the pin's input sense configuration, and the pin is configured as source for port interrupt.
Writing a '1' to this flag's bit location will clear the flag.

For enabling and executing the interrupt, refer to the interrupt level description.

17. BOD - Brownout Detector

17.1. Overview

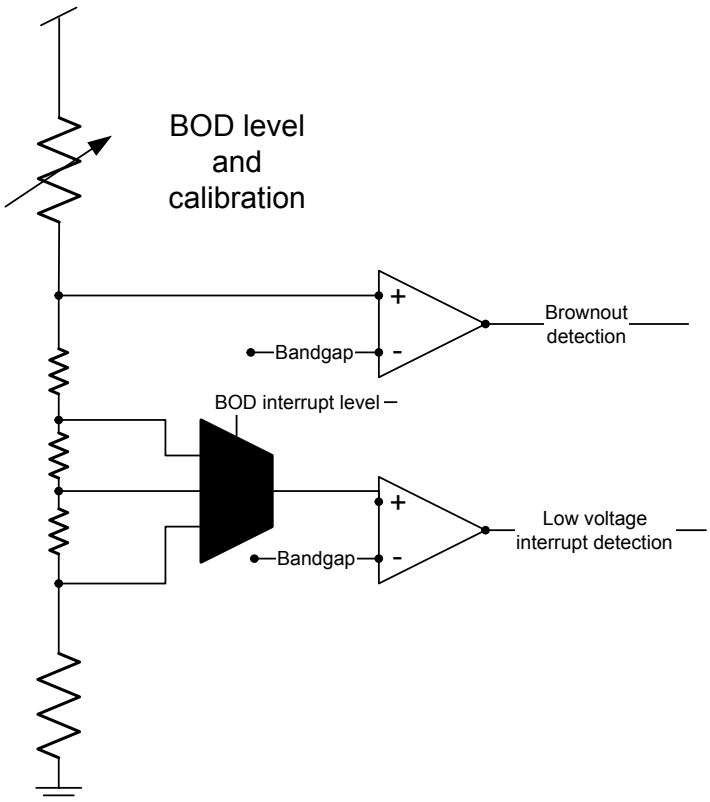
The Brownout Detector (BOD) monitors the power supply and compares the voltage with two programmable threshold levels. One threshold level defines when to generate a Reset, the other threshold defines when to generate an interrupt request.

17.2. Features

- Brownout detection
- Programmable BOD level
- Three modes:
  - Enabled
  - Sampled
  - Disabled
- Separate selection of mode for Active mode and sleep modes
- Voltage level monitor (VLM) with interrupt
- Programmable VLM level relative to the BOD level

17.3. Block Diagram

Figure 17-1. BOD Block Diagram



17.4. Product Dependencies

In order to use this peripheral, other parts of the system must be configured correctly, as described below.

Table 17-1. BOD Product Dependencies

Dependency	Applicable	Peripheral
Clocks	Yes	CLKCTRL
I/O Lines and Connections	No	-
Interrupts	Yes	CPUINT
Events	Yes	EVSYS
Debug	Yes	UPDI

Related Links

Clocks on page 156
Interrupts on page 54
Events on page 54
Debug Operation on page 156

17.4.1. Clocks

The BOD uses the 32KHz oscillator (OSCULP32K) as clock source for CLK_BOD.

17.4.2. I/O Lines and Connections

Not applicable.

17.4.3. Interrupts

Using the interrupts of this peripheral requires the Interrupt Controller to be configured first.

Related Links

CPUINT - CPU Interrupt Controller on page 97
SREG on page 51
Interrupts on page 136

17.4.4. Events

Not applicable.

17.4.5. Debug Operation

This peripheral is unaffected by entering debug mode.

Note: The VLM interrupt will not be executed if the CPU is halted in debug mode.

If the peripheral is configured to require periodical service by the CPU through interrupts or similar, improper operation or data loss may result during halted debugging.

17.5. Functional Description

17.5.1. Principle of Operation

The BOD monitors the power supply and compares the voltage with the programmable brownout threshold level. The BOD generates a Reset if the supply voltage crosses below the brownout threshold level.

The BOD also contains a voltage level monitor (VLM) that also monitors the power supply, but compares it to a threshold higher than the BOD threshold. The VLM can then generate an interrupt request as "early warning" when the supply voltage is dropping below the VLM threshold. The VLM threshold level is expressed in percent above the BOD threshold level.

The BOD is mainly controlled by fuses. The mode used in Standby sleep mode and Power Down sleep mode can be altered in normal program execution. The VLM part of the BOD is controlled by I/O registers as well.

In addition to Off mode, the BOD can also operate in Continuous mode (the BOD is on all the time) and Sampled mode (the BOD is periodically turned on briefly to check the supply voltage level).

17.5.2. Basic Operation

17.5.2.1. Initialization

The BOD is getting its default settings from fuses at device startup. The BOD level, operating mode in Active mode and Idle sleep mode are set by fuses and cannot be changed. The operation mode in Standby sleep mode and Power Down sleep mode are loaded from fuses, too, but can be changed in normal program execution.

The Voltage Level Monitor function can be enabled by writing a '1' to the VLMIE bit in the Interrupt Control register (BOD_INTCTRL.VLMIE).

Note: The VLM functionality will follow the BOD mode. If the BOD is turned off, the VLM will not enabled, even if the VLMINTEN is '1'. If the BOD is using sampled mode, the VLM will also be sampled.

The VLM threshold is defined by writing the VLM Level bits in the VLM Control A register (BOD_VLMCTRLA.VLMLVL).

The VLM interrupt is enabled by writing a '1' to the VLM Interrupt Enable bit in the VLM Interrupt Control register (BOD_INTCTRL.VLMIE). The VLM interrupt is configured by writing the VLM Configuration bits in the same register (BOD_INTCTRL.VLMCFG). An interrupt is requested either when the supply voltage crosses the VLM threshold from above, from below, or from any direction.
Note: If the BOD/VLM is enabled in sampled mode, only BOD_INTCTRL.VLMCFG=0x1 (crossing threshold from above) is valid.

17.5.3. Interrupts

Table 17-2. Available Interrupt Vectors and Sources

Offset	Name	Vector Description	Conditions
0x00	VLM	Voltage Level Monitor	Supply voltage crossing the VLM threshold as configured in the VLM Interrupt Control configuration (BOD_INTCTRL.VLMCFG)

When an interrupt condition occurs, the corresponding Interrupt Flag is set in the Interrupt Flags register of the peripheral (*peripheral*_INTFLAGS).

An interrupt source is enabled or disabled by writing to the corresponding bit in the peripheral's Interrupt Control register (*peripheral*_INTCTRL).

An interrupt request is generated when the corresponding interrupt source is enabled and the Interrupt Flag is set. The interrupt request remains active until the Interrupt Flag is cleared. See the peripheral's INTFLAGS register for details on how to clear Interrupt Flags.

Note: Interrupt requests are only generated when Interrupts are enabled globally.

Related Links

AVR CPU on page 40
SREG on page 51

17.5.4. Sleep Mode Operation

There are two separate fuses defining the BOD configuration in different sleep modes: One fuse defines the mode used in Active mode and Idle sleep mode (FUSE_BODCFG.ACTIVE), and is written to the Active bits in the Control A register (BOD_CTRLA.ACTIVE). The second fuse (FUSE_BODCFG.SLEEP) selects the mode used in Standby sleep mode and Power Down sleep mode, and is loaded into the Sleep bits in the Control A register (BOD_CTRLA.SLEEP).

The operating mode in Idle sleep mode (BOD_CTRLA.ACTIVE) cannot be altered by software. The operation mode in Standby sleep mode and Power Down sleep mode can be altered by writing to the Sleep bits in the Control A register (BOD_CTRLA.SLEEP).

When the device is going into Standby sleep mode or Power Down sleep mode, the BOD will change operation mode as defined by BOD_CTRLA.SLEEP. When the device is waking up from Standby or Power Down sleep mode, the BOD will operate in the mode defined by the Active bit field (BOD_CTRLA.ACTIVE).

17.5.5. Synchronization

When the BOD changes mode from Enabled to Sampled mode it needs to re-synchronize to the sampled clock. This is done automatically.

Note: When the sampling rate for the BOD is 1KHz and the BOD is switching from Enabled to Sampled mode, the first sample is delayed up to 1/2 cycle due to synchronization. This only happens on the first sample after changing to Sampled mode.

17.5.6. Configuration Change Protection

This peripheral has registers that are under Configuration Change Protection (CCP). In order to write to these, a certain key must be written to the CPU_CCP register first, followed by a write access to the protected bits within four CPU instructions.

It is possible to try writing to these registers any time, but the values are not altered.

The following registers are under CCP:

Table 17-3. RSTCTRL - Registers under Configuration Change Protection

Register	Key
BOD_CTRLA.SLEEP	IOREG

Related Links

Sequence for Write Operation to Configuration Change Protected I/O Registers on page 46

17.6. Register Summary

Offset	Name	Bit Pos.								
0x00	CTRLA	7:0				SAMPLFREQ	ACTIVE[1:0]		SLEEP[1:0]	
0x01	CTRLB	7:0						LVL[2:0]		
0x02 ... 0x07	Reserved									
0x08	CTRLA	7:0							VLMLVL[1:0]	
0x09	INTCTRL	7:0						VLMCFG[1:0]		VLMIE
0x0A	INTFLAGS	7:0								VLMIF
0x0B	STATUS	7:0								VLMS

17.7. Register Description

17.7.1. Control A

Name: CTRLA
Offset: 0x00
Reset: 0x05
Property: -

Bit	7	6	5	4	3	2	1	0
				SAMPLFREQ	ACTIVE[1:0]		SLEEP[1:0]	
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	1	0	1

Bit 4 – SAMPLFREQ: Sample Frequency
This bit selects the BOD sample frequency.
Note: The Reset value is loaded from the SAMPLEFREQ bit in FUSE.BODCFG.

Value	Description
0x0	Sample frequency is 1kHz
0x1	Sample frequency is 125Hz

Bits 3:2 – ACTIVE[1:0]: Active
These bits selects the BOD operation mode when the device is in active or idle mode.

Note: The Reset value is loaded from the ACTIVE bits in FUSE.BODCFG.

Value	Description
0x0	Disabled
0x1	Enabled
0x2	Sampled
0x3	Enabled with wake-up halted until BOD is ready

Bits 1:0 – SLEEP[1:0]: Sleep
These bits select the BOD operation mode when the device is in standby or power-down sleep mode.
Note: The Reset value is loaded from the SLEEP bits in FUSE.BODCFG.

Value	Description
0x0	Disabled
0x1	Enabled
0x2	Sampled
0x3	Reserved

17.7.2. Control B

Name: CTRLB
Offset: 0x01
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
						LVL[2:0]		
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 2:0 – LVL[2:0]: BOD Level
These bits selects the BOD threshold level. The value is loaded from the BOD Level bits (LVL) in the BOD Configuration fuse (FUSE.BODCFG).

Value	Description
0x0	1.8V
0x1	2.15V
0x2	2.51V
0x3	2.87V
0x4	3.23V
0x5	3.59V
0x6	3.94V
0x7	4.3V

17.7.3. VLM Control A

Name: CTRLA
Offset: 0x08
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
							VLMLVL[1:0]	
Access	R	R	R	R	R	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 1:0 – VLMLVL[1:0]: VLM Level
These bit select the Voltage Level Monitor threshold relative to the BOD threshold (BOD_CTRLB.LVL).

Value	Description
0x0	VLM threshold 5% above BOD threshold
0x1	VLM threshold 15% above BOD threshold
0x2	VLM threshold 25% above BOD threshold
other	Reserved

17.7.4. Interrupt Control

Name: INTCTRL
Offset: 0x09
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
						VLMCFG[1:0]		VLMIE
Access	R	R	R	R	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 2:1 – VLMCFG[1:0]: VLM Configuration
These bits select which incidents will trigger a VLM interrupt.

Value	Description
0x0	Voltage crosses VLM threshold from above
0x1	Voltage crosses VLM threshold from below
0x2	Either direction is triggering an interrupt request
Other	Reserved

Bit 0 – VLMIE: VLM Interrupt Enable
Writing a '1' to this bit enables the Voltage Level Monitor (VLM) interrupt.

17.7.5. VLM Interrupt Flags

Name: INTFLAGS
Offset: 0x0A
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
								VLMIF
Access	R	R	R	R	R	R	R	R/W
Reset	0	0	0	0	0	0	0	0

Bit 0 – VLMIF: VLM Interrupt Flag
This flag is set when a trigger from the VLM is given, as configured by the VLMCFG bit in the BOD.INTCTRL register. The flag remains set only as long as the BOD is enabled.

17.7.6. VLM Status

Name: STATUS
Offset: 0x0B
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
								VLMS
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit 0 – VLMS: VLM Status

Value	Description
0	The voltage is above the VLM threshold level
1	The voltage is below the VLM threshold level

18. VREF - Voltage Reference

18.1. Overview

The Voltage Reference peripheral (VREF) provides control registers for the voltage reference sources used by several peripherals.

A voltage reference source is enabled automatically when requested by a peripheral. They can also be enabled even if not requested.

18.2. Features

- Two programmable voltage reference sources
  - One for ADC peripheral
  - One for AC and DAC peripheral
- Each reference source supports five different voltages:
  - 0.55V
  - 1.1V
  - 1.5V
  - 2.5V
  - 4.34V

18.3. Functional Description

18.3.1. Principal of Operation

The user can select the reference voltages for the ADC by writing to the ADC Reference Select bit field in the Control A register (VREF_CTRLA.ADCREFSEL), and for both AC and DAC by writing to the DAC and AC Reference Select bit field (VREF_CTRLA.DACREFSEL).

The user can enable the reference voltage sources (and thus, override the automatic disabling of unused sources) by writing to the respective Force Enable bit in the Control B register (VREF_CTRLB.NVMREFEN, .ADCREFEN, .DACREFEN). This may be desirable to shorten startup times, on the cost of power consumption.

18.3.2. Basic Operation

18.3.2.1. Initialization

The default configuration will enable the respective source when the ADC, AC or DAC is requesting a reference voltage. The default reference voltages are 0.55V, but can be configured by writing to the respective Reference Select bit field in the Control A register (VREF_CTRLA.ADCREFSEL, .DACREFSEL).

18.4. Register Summary

Offset	Name	Bit Pos.								
0x00	CTRLA	7:0		ADCREFSEL[2:0]				DACREFSEL[2:0]		
0x01	CTRLB	7:0						NVMREFEN	ADCREFEN	DACREFEN

18.5. Register Description

18.5.1. Control A

Name: CTRLA
Offset: 0x00
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
		ADCREFSEL[2:0]				DACREFSEL[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		0	0	0		0	0	0

Bits 6:4 – ADCREFSEL[2:0]: ADC Reference Select
These bits select the reference voltage for the ADC.

Table 18-1. ADC Reference Voltage

ADCREFSEL	Reference Voltage
0x0	0.55 V
0x1	1.1 V
0x2	2.5 V
0x3	4. 34 V
0x4-0x7	1.5 V

Value	Description
0x0	0.55V
0x1	1.1V
0x2	2.5V
0x3	4.34V
other	1.5V

Bits 2:0 – DACREFSEL[2:0]: DAC and AC Reference Select
These bits select reference voltage for the DAC and AC.

Value	Description
0x0	0.55V
0x1	1.1V
0x2	2.5V
0x3	4.34V
other	1.5V

18.5.2. Control B

Name: CTRLB
Offset: 0x01
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
						NVMREFEN	ADCREFEN	DACREFEN
Access						R/W	R/W	R/W
Reset						0	0	0

Bit 2 – NVMREFEN: NVM Reference Force Enable
Writing a '1' to this bit forces the voltage reference for the NVM to be running, even if it not requested.

Writing a '0' to this bit allows automatic disabling of the reference source when not requested.

Bit 1 – ADCREFEN: ADC Reference Force Enable
Writing a '1' to this bit forces the voltage reference for the ADC to be running, even if it not requested.

Writing a '0' to this bit allows automatic disabling of the reference source when not requested.

Bit 0 – DACREFEN: DAC and AC Reference Force Enable
Writing a '1' to this bit forces the voltage reference for the DAC and AC to be running, even if it not requested.

Writing a '0' to this bit allows automatic disabling of the reference source when not requested.

19. WDT - Watchdog Timer

19.1. Overview

The Watchdog Timer (WDT) is a system function for monitoring correct program operation. It allows to recover from error situations such as runaway or deadlocked code. The WDT is a timer, configured to a predefined timeout period, and is constantly running when enabled. If the WDT is not Reset within the timeout period, it will issue a System Reset. The WDT is reset by executing the `WDR` (Watchdog Timer Reset) instruction from the application code.

A window mode allows to define a time slot or window inside the total timeout period during which WDT must be reset. If the WDT is reset outside this window, either too early or too late, a System Reset will be issued. Compared to the normal mode, the window mode can also catch situations where a code error causes constant `WDR` execution.

The WDT will run in active mode and all sleep modes, if enabled. It is asynchronous, i.e. running from a CPU-independent clock source. For this reason it will continue to operate and be able to issue a System Reset even if the main clock fails.

The Configuration Change Protection mechanism ensures that the WDT settings cannot be changed by accident. For increased safety, a configuration for locking the WDT settings is also available.

Related Links

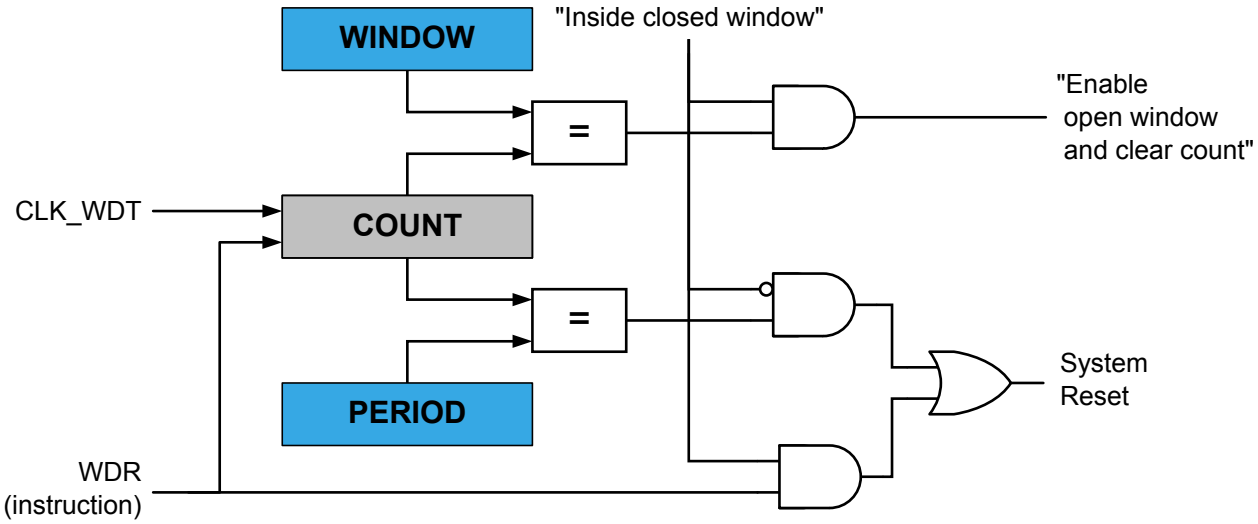
CCP - Configuration Change Protection on page 46

19.2. Features

- Issues a System Reset if the Watchdog Timer is not cleared before its timeout period
- Asynchronous operation from dedicated oscillator
- Using the 1KHz output of the 32KHz Ultra Low Power oscillator (OSCULP32K)
- 11 selectable timeout periods, from 8ms to 8s
- Two operation modes:
  - Normal Mode
  - Window Mode
- Configuration lock to prevent unwanted changes
- Closed period timer activation after first WDT instruction for easy setup

19.3. Block Diagram

Figure 19-1. WDT Block Diagram



19.4. Signal Description

Not applicable.

19.5. Product Dependencies

In order to use this peripheral, other parts of the system must be configured correctly, as described below.

Table 19-1. WDT Product Dependencies

Dependency	Applicable	Peripheral
Clocks	Yes	CLKCTRL
I/O Lines and Connections	No	-
Interrupts	No	-
Events	No	-
Debug	Yes	UPDI

Related Links

Clocks on page 171
Debug Operation on page 172

19.5.1. Clocks

A 1KHz oscillator clock (CLK_WDT_OSC) is sourced from the internal ultra-low-power oscillator, OSC32K. Due to the ultra-low-power design, the oscillator is not very accurate, and so the exact time-out period may vary from device to device. This variation must be kept in mind when designing software that uses the WDT to ensure that the time-out periods used are valid for all devices.

The counter clock CLK_WDT_OSC is asynchronous to the system clock. Due to this asynchronicity, writing to WDT control register will require synchronization between the clock domains.

19.5.2. I/O Lines and Connections

Not applicable.

19.5.3. Interrupts

Not applicable.

19.5.4. Events

Not applicable.

19.5.5. Debug Operation

When run-time debugging, this peripheral will continue normal operation. Halting the CPU in debugging mode will halt normal operation of the peripheral.

When halting the CPU in debug mode, the WDT counter is reset.

when starting the CPU again and the WDT was operating in window mode, the first closed window timeout period will be disabled.

19.6. Functional Description

19.6.1. Principle of Operation

The Watchdog Timer (WDT) is a system for monitoring correct program operation, making it possible to recover from error situations such as runaway code, by issuing a Reset. When enabled, the WDT is a constantly running timer that is configured to a predefined time-out period. Before the end of the time-out period, the WDT should be set back, or else, a system Reset is issued.

The WDT has two modes of operation, Normal mode and Window mode. The description for each of the basic modes is given below. The settings in the Control A register (CTRLA) determine the mode of operation.

19.6.2. Basic Operation

19.6.2.1. Initialization

- Enable the WDT by writing a non-zero value to the Period bits in the Control A register (WDT_CTRLA.PERIOD).
- Optional: Write a non-zero value to the Window bits (WDT_CTRLA.PERIOD) to enable window mode operation.

All bits in the Control A register and the Lock bit in the Status register (WDT_STATUS.LOCK) are write protected by the Configuration Change Protection mechanism.

If WDT is enabled from Boot code, the WDT_STATUS.LOCK bit is set automatically.

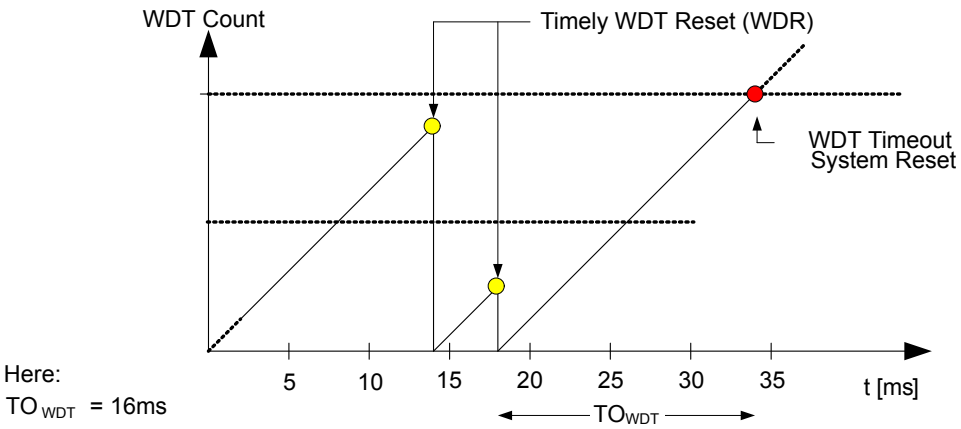
19.6.2.2. Normal Mode

In normal mode operation, a single timeout period is set for the WDT. If the WDT is not reset from the application code using the before the timeout occurs, the WDT will issue a System Reset.

There are 11 possible WDT timeout periods (TOWDT), selectable from 8ms to 8s by writing to the Period bit field in the Control A register (WDT_CTRLA.PERIOD).

The WDT can be reset by the `WDR` instruction any time during the timeout period. A new WDT timeout period will be started each time the WDT is reset by `WDR`.

Figure 19-2. Normal Mode Operation



19.6.2.3. Window Mode

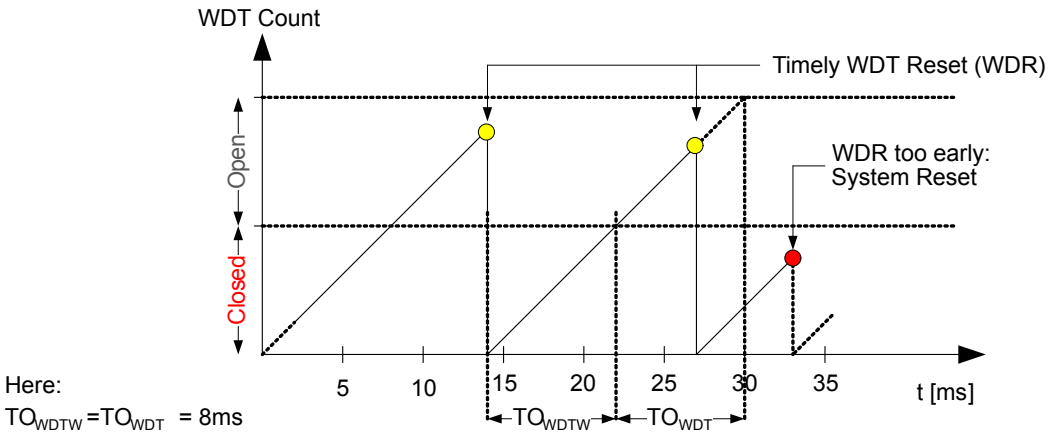
In window mode operation, the WDT uses two different timeout periods, a "closed" window timeout period (TOWDTW) and the normal timeout period (TOWDT):

The closed window timeout period defines a duration from 8ms to 8s where the WDT cannot be reset. If the WDT is reset during this period, the WDT will issue a System Reset.

The normal WDT timeout period, which is also 8ms to 8s, defines the duration of the "open" period during which the WDT can (and should) be reset. The open period will always follow the closed period, so the total duration of the timeout period is the sum of the closed window and the open window timeout periods.

After device startup or in debug break mode, the first closed period is activated after the first `WDR` instruction.

Figure 19-3. Window Mode Operation



19.6.3. Additional Features

19.6.3.1. Configuration Protection and Lock

The WDT provides two security mechanisms to avoid unintentional changes to the WDT settings:

The first mechanism is the Configuration Change Protection mechanism, employing a timed write procedure for changing the WDT control registers.

The second mechanism locks the configuration by writing a '1' to the Lock bit in the Status register (WDT_STATUS.LOCK). When this bit is '1', the Control A register (WDT_CTRLA) cannot be changed. Consequently, the WDT cannot be disabled from software.

WDT_STATUS.LOCK can only be written to '1'. It can only be cleared in debug mode.

If the WDT is enabled in startup Boot code, WDT_STATUS.LOCK is automatically set.

Related Links

CCP - Configuration Change Protection on page 46

19.6.4. Events

Not applicable.

19.6.5. Interrupts

Not applicable.

19.6.6. Sleep Mode Operation

The WDT will continue to operate in any sleep mode where the source clock is active.

19.6.7. Synchronization

Due to asynchronicity between the main clock domain and the peripheral clock domain, the Control A register (WDT_CTRLA) is synchronized when written. The Synchronization Busy flag in the Status register (WDT_STATUS.SYNCBUSY) indicates if there is an ongoing synchronization.

Writing to the WDT_CTRLA register while WDT_STATUS.SYNCBUSY=1 is not allowed.

The following registers are synchronized when written:

- Period bits in Control A register (WDT_CTRLA.PERIOD)
- Window Period bits in Control A register (WDT_CTRLA.WINDOW)

19.6.8. Configuration Change Protection

This peripheral has registers that are under Configuration Change Protection (CCP). In order to write to these, a certain key must be written to the CPU_CCP register first, followed by a write access to the protected bits within four CPU instructions.

It is possible to try writing to these registers any time, but the values are not altered.

The following registers are under CCP:

Table 19-2. WDT - Registers under Configuration Change Protection

Register	Key
WDT_CTRLA	IOREG
WDT_STATUS.LOCK	IOREG

List of bits/registers protected by CCP.

- Period bits in Control A register (CTRLA.PERIOD)
- Window Period bits in Control A register (CTRLA.WINDOW)
- Lock bit in Status register (STATUS.LOCK)

Related Links

CCP - Configuration Change Protection on page 46

19.7. Register Summary

Offset	Name	Bit Pos.								
0x00	CTRLA	7:0	WINDOW[3:0]				PERIOD[3:0]			
0x01	STATUS	7:0	LOCK							SYNCBUSY

19.8. Register Description

19.8.1. Control A

Name: CTRLA
Offset: 0x00
Reset: From FUSE.WDTCFG
Property: Configuration Change Protection

Bit	7	6	5	4	3	2	1	0
	WINDOW[3:0]				PERIOD[3:0]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	x	x	x	x	x	x	x	x

Bits 7:4 – WINDOW[3:0]: Window
Writing a non-zero value to these bits enables the window mode, and selects the according duration of the closed period.

The bits are optionally lock protected:

- If LOCK bit in WDT.STATUS is '1', all bits are change protected (Access = R)
- If LOCK bit in WDT.STATUS is '0', all bits can be changed (Access = R/W)

Value	Name	Description
0x0	OFF	-
0x1	8CLK	0.008s
0x2	16CLK	0.016s
0x3	32CLK	0.032s
0x4	64CLK	0.064s
0x5	128CLK	0.128s
0x6	256CLK	0.256s
0x7	512CLK	0.512s
0x8	1KCLK	1.0s
0x9	2KCLK	2.0s
0xA	4KCLK	4.1s
other	8KCLK	8.2s

Bits 3:0 – PERIOD[3:0]: Period
Writing a non-zero value to this bit enables the WDT, and selects the according timeout period in normal mode. In window mode, these bits select the duration of the open window.

The bits are optionally lock protected:

- If LOCK in WDT.STATUS is '1', all bits are change protected (Access = R)
- If LOCK in WDT.STATUS is '0', all bits can be changed (Access = R/W)

Value	Name	Description
0x0	OFF	-
0x1	8CLK	0.008s
0x2	16CLK	0.016s
0x3	32CLK	0.032s
0x4	64CLK	0.064s
0x5	128CLK	0.128s
0x6	256CLK	0.256s
0x7	512CLK	0.512s
0x8	1KCLK	1.0s
0x9	2KCLK	2.0s
0xA	4KCLK	4.1s
other	8KCLK	8.2s

19.8.2. Status

Name: STATUS
Offset: 0x01
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	LOCK							SYNCBUSY
Access	R/W							R
Reset	0							0

Bit 7 – LOCK: Lock
Writing this bit to '1' write protects the WDT.CTRLA register.

It is only possible to write this bit to '1'. This bit can only be cleared in debug.

If the PERIOD bits in WDT.CTRLA are different from zero after boot code, Lock will automatically be set.

This bit is under Configuration Change Protection (CCP).

Bit 0 – SYNCBUSY: Synchronization Busy
This bit is set after writing to the WDT.CTRLA register while the data is being synchronized from the system clock domain to the WDT clock domain.

This bit is cleared by system after the synchronization is finished.

20. TCA - 16-bit Timer/Counter Type A

20.1. Overview

The flexible 16-bit PWM Timer/Counter type A (TCA) provides accurate program execution timing, frequency and waveform generation, and command execution.

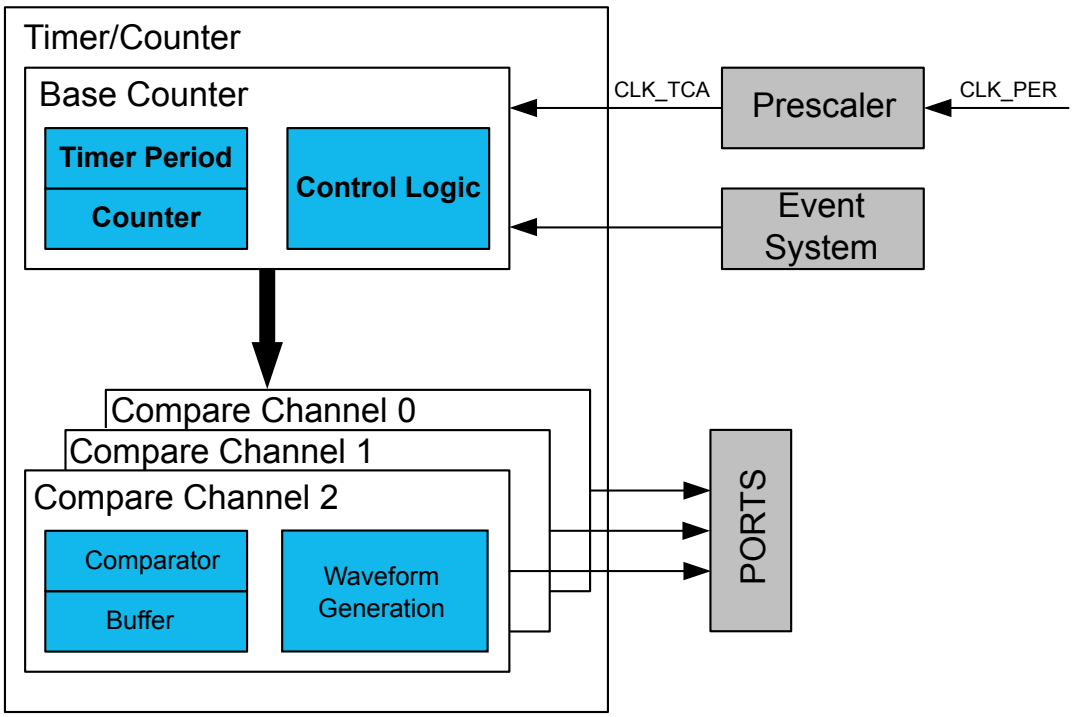
A TCA consists of a base counter and a set of compare channels. The base counter can be used to count clock cycles or events, or let events control how it counts clock cycles. It has direction control and period setting that can be used for timing. The compare channels can be used together with the base counter to do compare match control, frequency generation, and pulse width waveform modulation.

A timer/counter can be clocked and timed from the peripheral clock with optional prescaling or from the event system. The event system can also be used for direction control or to synchronize operations.

The timer/Counter has a split mode feature that splits it into two 8-bit timer/counters with three compare channels each.

A block diagram of the 16-bit timer/counter with closely related peripheral modules (in grey) is shown below.

Figure 20-1. 16-bit Timer/counter and Closely Related Peripherals



20.2. Features

- 16-bit timer/counter
- Three compare channels
- Double buffered timer period setting
- Double buffered compare channels
- Waveform generation:

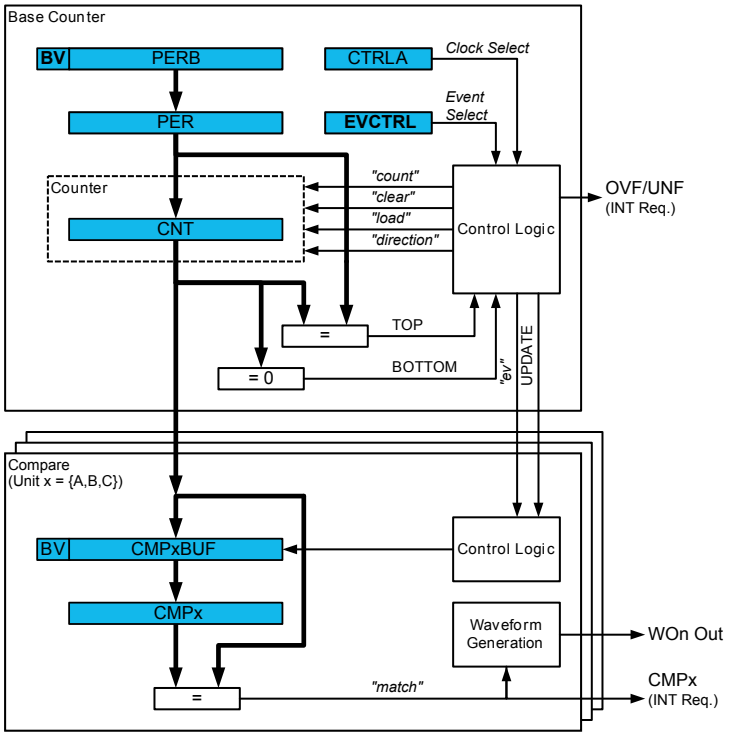
- Frequency generation
- Single-slope PWM (pulse width modulation)
- Dual-slope PWM

- Count on event
- Timer overflow interrupts/events
- One compare match per compare channel
- Two 8-bit timer/counters in Split Mode

20.3. Block Diagram

The below figure shows a detailed block diagram of the timer/counter.

Figure 20-2. Timer/Counter Block Diagram



The counter register (CNT), period registers with buffer (PER and PERB), and compare registers with buffers (COMPx and COMPBx) are 16-bit registers. All buffer register have a buffer valid (BV) flag that indicates when the buffer contains a new value.

During normal operation, the counter value is continuously compared to zero and the period (PER) value to determine whether the counter has reached TOP or BOTTOM.

The counter value is also compared to the COMPx registers. These comparisons can be used to generate interrupt requests. The waveform generator modes use these comparisons to set the waveform period or pulse width.

A prescaled peripheral clock and events from the event system can be used to control the counter.

20.4. Signal Description

Signal	Description	Type
WO[2:0]	Digital output	Waveform output
WO[5:3]	Digital output	Waveform output - Split Mode only

20.5. Product Dependencies

In order to use this peripheral, other parts of the system must be configured correctly, as described below.

Table 20-1. TCA Product Dependencies

Dependency	Applicable	Peripheral
Clocks	Yes	CLKCTRL
I/O Lines and Connections	No	-
Interrupts	Yes	CPUINT
Events	Yes	EVSYS
Debug	Yes	UPDI

Related Links

Clocks on page 181
Interrupts on page 54
Events on page 181
Debug Operation on page 182

20.5.1. Clocks

This peripheral uses the system clock CLK_PER, and has its own prescaler.

Related Links

CLKCTRL - Clock Controller on page 68

20.5.2. I/O Lines and Connections

Not applicable.

20.5.3. Interrupts

Using the interrupts of this peripheral requires the Interrupt Controller to be configured first.

Related Links

CPUINT - CPU Interrupt Controller on page 97
SREG on page 51
Interrupts on page 136

20.5.4. Events

The events of this peripheral are connected to the Event System.

Related Links

EVSYS - Event System on page 109

20.5.5. Debug Operation

When run-time debugging, this peripheral will continue normal operation. Halting the CPU in debugging mode will halt normal operation of the peripheral.

This peripheral can be forced to operate with halted CPU by writing a '1' to the Debug Run bit in the Debug Control register of the peripheral (*peripheral*_DBGCTRL.DBGRUN).

Related Links

UPDI - Unified Program Debug Interface on page 490

20.6. Functional Description

20.6.1. Principle of Operation

Depending on the mode of operation, the counter is cleared, reloaded, incremented, or decremented at each timer/counter clock or event input.

By default, the TCA is a 16-bit timer/counter. It can be split to work as two 8-bit timer/counters.

20.6.2. Basic Operation

20.6.2.1. Definitions

The following definitions are used throughout the documentation:

Table 20-2. Timer/Counter Definitions

Name	Description
BOTTOM	The counter reaches BOTTOM when it becomes zero.
MAX	The counter reaches MAXimum when it becomes all ones.
TOP	The counter reaches TOP when it becomes equal to the highest value in the count sequence. Depending on the waveform generator mode, the TOP value can be equal to the Period register a Compare channel n register setting.
UPDATE	The timer/counter signals an update when it reaches BOTTOM or TOP, depending on the waveform generator mode.
CNT	Counter register value
CAPT	Capture/compare register value

In general, the term “timer” is used when the timer/counter clock control is handled by an internal source, and the term “counter” is used when the clock control is handled externally (e.g. counting external Events).

20.6.2.2. Initialization

To start using the timer/counter in a basic mode, follow these steps:

- Write a TOP value to the Period register (TCA_PER.PER)
- Enable the peripheral by writing a '1' to the Enable bit in the Control A register (TCA_CTRLA.ENABLE).
  The counter will start counting clock ticks according to the prescaler setting in the Clock Select bit field (TCA_CTRLA.CLKSEL)
- Optional: By writing a '1' to the Enable Count on Event Input bit in the Event Control register (TCA_EVCTRL.CNTEI), Event inputs are counted instead of clock ticks.

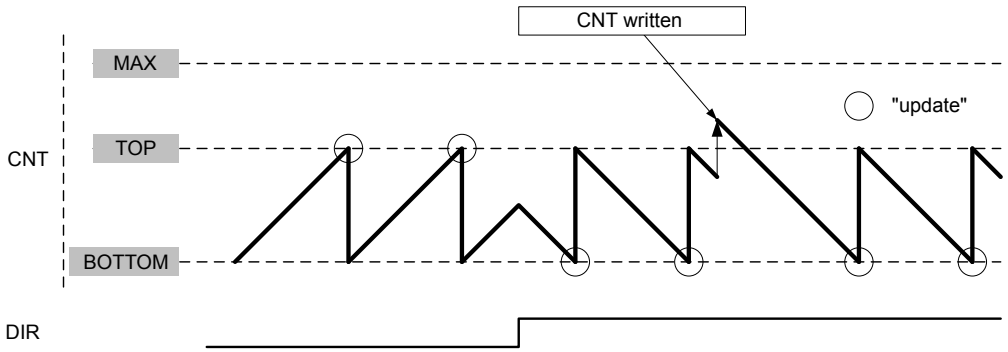
- The counter value can be read from the Counter bit field in the Counter register (TCA_CNT.CNT).

20.6.2.3. Normal Operation

In normal operation, the counter is counting clock ticks in the direction selected by the Direction bit in the Control E register (TCA_CTRLE.DIR), until it reaches TOP or BOTTOM. The clock ticks are from the peripheral clock CLK_PER, optionally prescaled, depending on the Clock Select bit field in the Control A register (TCA_CTRLA.CLKSEL).

When up-counting and TOP is reached, the counter will wrap to zero at the next clock tick. When down-counting, the counter is reloaded with the Period register value (TCA_PER.PER) when BOTTOM is reached.

Figure 20-3. Normal Operation



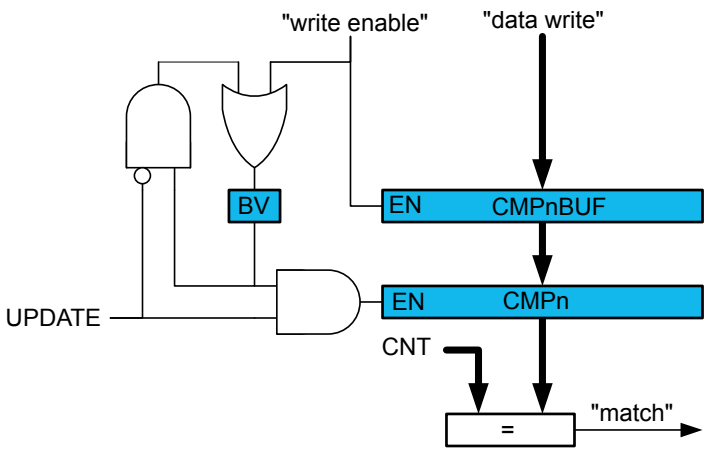
It is possible to change the Counter value in the Counter register (TCA_CNT.CNT) when the counter is running. The write access to TCA_CNT.CNT has higher priority than count, clear, or reload, and will be immediate. The direction of the counter can also be changed during normal operation by writing to TCA_CTRLE.DIR.

20.6.2.4. Double Buffering

The Period register value (TCA_PER.PER) and the Compare n register values (TCA_CMPn.CMP) are all double buffered (TCA_PERBUF.PERBUF and TCA_CMPnBUF.CMPBUF).

Each buffer register has a Buffer Valid (BV) flag in the Control F register (TCA_CTRLF.PERBV and .CMPnBV, respectively), which indicates that the buffer register contains a valid, i.e. new, value that can be copied into the corresponding Period or Compare register. When the Period register and Compare n registers are used for a Compare operation, the BV flag is set when data is written to the buffer register, and cleared on an UPDATE condition. This is shown for a Compare register in below.

Figure 20-4. Period and Compare Double Buffering



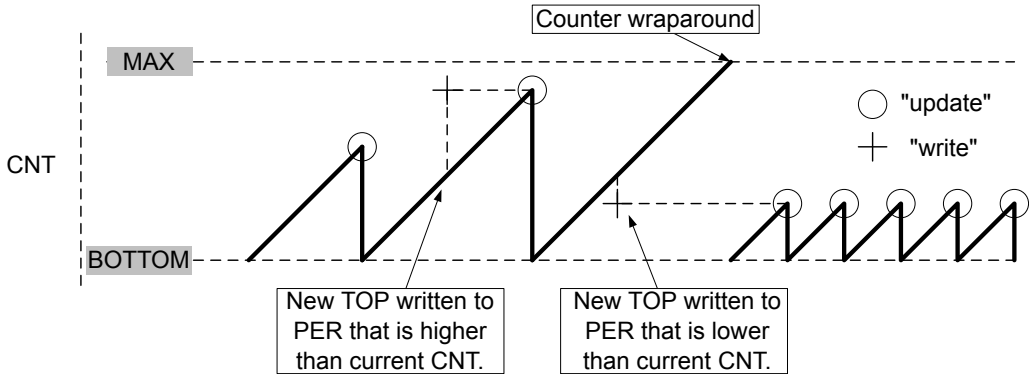
Both the TCA_CMPn and TCA_CMPnBUF registers are available as I/O registers. This allows initialization and bypassing of the buffer register and the double buffering function.

20.6.2.5. Changing the Period

The Counter period is changed by writing a new TOP value to the Period register (TCA_PER.PER).

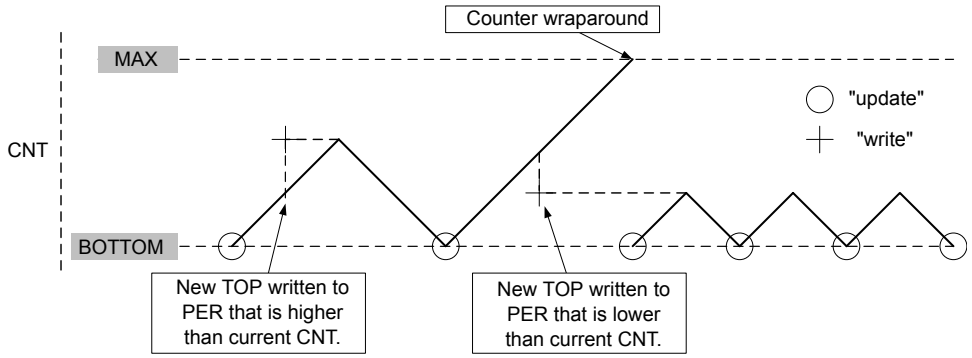
No Buffering: If double buffering is not used, any period update is immediate.

Figure 20-5. Changing the Period without Buffering



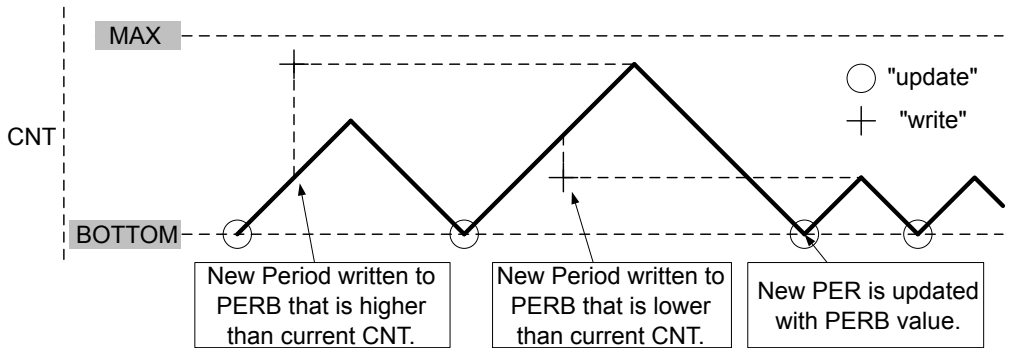
A counter wraparound can occur in any mode of operation when up-counting without buffering. This due to the fact that the registers TCA_CNT and TCA_PER are continuously compared: if a new TOP value is written to TCA_PER that is lower than current TCA_CNT, the counter will wrap first before a compare match happened.

Figure 20-6. Unbuffered Dual-slope Operation



With Buffering: When double buffering is used, the buffer can be written at any time and still maintain correct operation. The TCA_PER is always updated on the UPDATE condition, as shown for dual-slope operation in the figure below. This prevents wraparound and the generation of odd waveforms.

Figure 20-7. Changing the Period Using Buffering



20.6.2.6. Compare Channel

Each Compare Channel n continuously compares the counter value (TCA_CNT) with the Compare n register (TCA_CMPn.CMP). If TCA_CNT.CNT equals TCA_CMPn.CMP, the comparator n signals a match. The match will set the Compare Channel's interrupt flag at the next timer clock cycle, and the optional interrupt is generated.

The Compare n Buffer register (TCA_CMPnBUF.CMPBUF) provides double buffer capability equivalent to that for the period buffer. The double buffering synchronizes the update of the TCA_CMPn register with the buffer value to either the TOP or BOTTOM of the counting sequence, according to the UPDATE condition. The synchronization prevents the occurrence of odd-length, non-symmetrical pulses for glitch-free output.

Waveform Generation

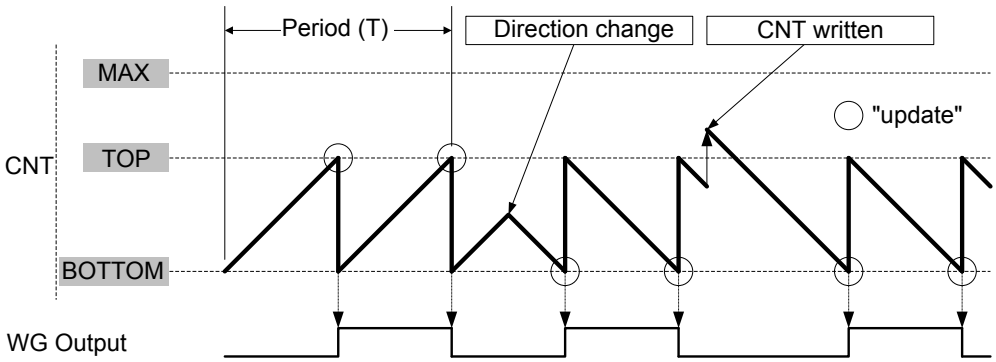
The compare channels can be used for waveform generation on the corresponding port pins. To make the waveform visible on the connected port pin, the following requirements must be fulfilled:

1. A waveform generation mode must be selected by writing TCA_CTRLB.WGMODE
2. The TCA is counting clock tick, not Events (TCA_EVCTRL.CNTEI=0)
3. The compare channels used must be enabled (TCA_CTRLB.CMPnEN=1). This will override the corresponding PORT pin output register. An alternative pin can be selected by writing to the respective TCA Waveform Output n bit in the Control C register of the Port Multiplexer (PORTMUX_CTRLC.TCA0n).
4. The direction for the associated Port pin n must be configured as output (PORT_DIR.DIR[n]=1).
5. Optional: Enable inverted waveform output for the associated Port pin n (PORT_PINn.INVEN=1).

Frequency (FRQ) Waveform Generation

For frequency generation, the period time (T) is controlled by a TCA_CMPn register instead of the Period register (TCA_PER). The waveform generation output WG is toggled on each compare match between the TCA_CNT and TCA_CMPn registers.

Figure 20-8. Frequency Waveform Generation



The waveform frequency (f_{FRQ}) is defined by the following equation:

$$f_{FRQ} = \frac{f_{CLK\_PER}}{2N(CMPn+1)}$$

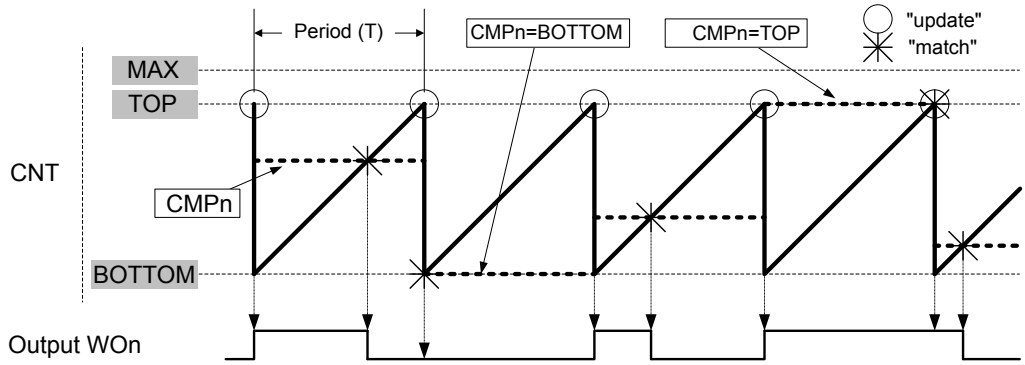
where *N* represents the prescaler divider used (TCA_CTRLA.CLKSEL), and f_{CLK_PER} is the system clock for the peripherals.

The maximum frequency of the waveform generated is half of the peripheral clock frequency ($f_{CLK\_PER}/2$) when TCA_CMPn.CMP is written to zero (0x0000) and no prescaling is used (*N*=1, TCA_CTRLA.CLKSEL=0x0).

Single-Slope PWM Generation

For single-slope Pulse Width Modulation (PWM) generation, the period (T) is controlled by TCA_PER.PER, while the values of TCA_CMPn.CMP control the duty cycle of the WG output. The figure below shows how the counter counts from BOTTOM to TOP and then restarts from BOTTOM. The waveform generator (WG) output is set on the compare match between the TCA_CNT and TCA_CMPn registers, and cleared at TOP.

Figure 20-9. Single-Slope Pulse Width Modulation



The TCA_PER register defines the PWM resolution. The minimum resolution is 2 bits (TCA_PER.PER=0x0003), and the maximum resolution is 16 bits (TCA_PER.PER=MAX).

The following equation calculate the exact resolution for single-slope PWM (R_{PWM_SS}):

$$R_{PWM\_SS} = \frac{\log(PER+1)}{\log(2)}$$

The single-slope PWM frequency (f_{PWM_SS}) depends on the period setting (TCA_PER), the system's peripheral clock frequency f_{CLK_PER}, the TCA prescaler (TCA_CTRLA.CLKSEL). It is calculated by the following equation:

$$f_{PWM\_SS} = \frac{f_{CLK\_PER}}{N(PER+1)}$$

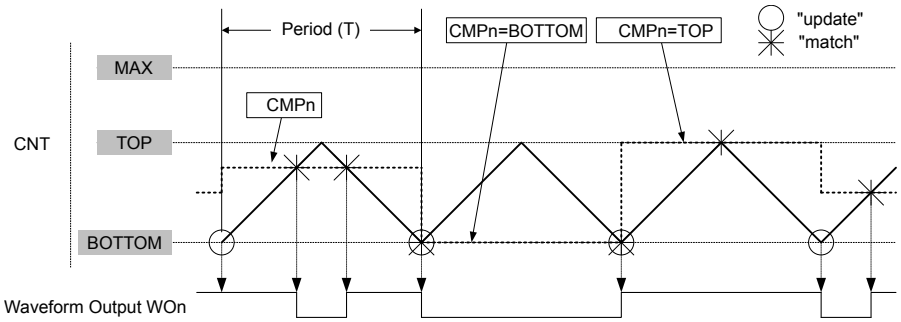
where *N* represents the prescaler divider used.

Dual-slope PWM

For dual-slope PWM generation, the period (T) is controlled by TCA_PER.PER, while the values of TCA_CMPn.CMP control the duty cycle of the WG output.

The figure below shows how for dual-slope PWM the counter counts repeatedly from BOTTOM to TOP and then from TOP to BOTTOM. The waveform generator output is set on BOTTOM, cleared on compare match when up-counting, and set on compare match when down-counting.

Figure 20-10. Dual-slope Pulse Width Modulation



Using dual-slope PWM results in a lower maximum operation frequency compared to the single-slope PWM operation.

The period register (TCA_PER) defines the PWM resolution. The minimum resolution is 2 bits (TCA_PER.PER=0x0003), and the maximum resolution is 16 bits (TCA_PER.PER=MAX).

The following equation calculate the exact resolution for dual-slope PWM (R_{PWM_DS}):

$$R_{\mathrm{PWM\_DS}} = \frac{\log(\mathrm{PER}+1)}{\log(2)}$$

The PWM frequency depends on the period setting (TCA_PER), the peripheral clock frequency (f_{CLK_PER}), and the prescaler divider used (TCA_CTRLA.CLKSEL). It is calculated by the following equation:

$$f_{\mathrm{PWM\_DS}} = \frac{f_{\mathrm{CLK\_PER}}}{2N \cdot \mathrm{PER}}$$

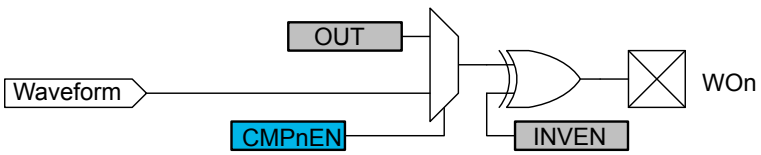
N represents the prescaler divider used.

Port Override for Waveform Generation

To make the waveform generation available on the Port pins, the corresponding Port pin direction must be set as output (PORT_DIR.DIR[n]=1). The TCA will override the port pin values when the compare channel is enabled (TCA_CTRLB.CMPnEN=1) and a waveform generation mode is selected.

The figure below shows the port override for TCA. The timer/counter compare channel will override the port pin output value (OUT) on the corresponding port pin. Enabling inverted I/O on the port pin (PORT_PINn.INVEN=1) inverts the corresponding WG output.

Figure 20-11. Port Override for Timer/Counter Type A



20.6.2.7. Timer/Counter Commands

A set of commands can be issued by software to immediately change the state of the peripheral. These commands give direct control of the UPDATE, RESTART, and RESET signals. A command is issued by writing the respective value to the Command bit field in the Control E register (TCA_CTRLESET.CMD).

An `UPDATE` command has the same effect as when an update condition occurs, except that the `UPDATE` command is not affected by the state of the Lock Update bit in the Control E register (TCA_CTRLE.LUPD).

The software can force a restart of the current waveform period by issuing a `RESTART` command. In this case the counter, direction, and all compare outputs are set to zero.

A `RESET` command will set all timer/counter registers to their initial values. A `RESET` can be issued only when the timer/counter is not running (TCA_CTRLA.ENABLE=0).

20.6.2.8. Split Mode - Two 8-bit Timer/Counters

Split Mode Overview

To double the number of timers and PWM channels in the TCA at a minimal cost, a Split Mode is provided. In this Split Mode, the 16-bit timer/counter acts as two separate 8-bit timers, which each have three compare channels for PWM generation. To eliminate the need for buffer registers, the split mode

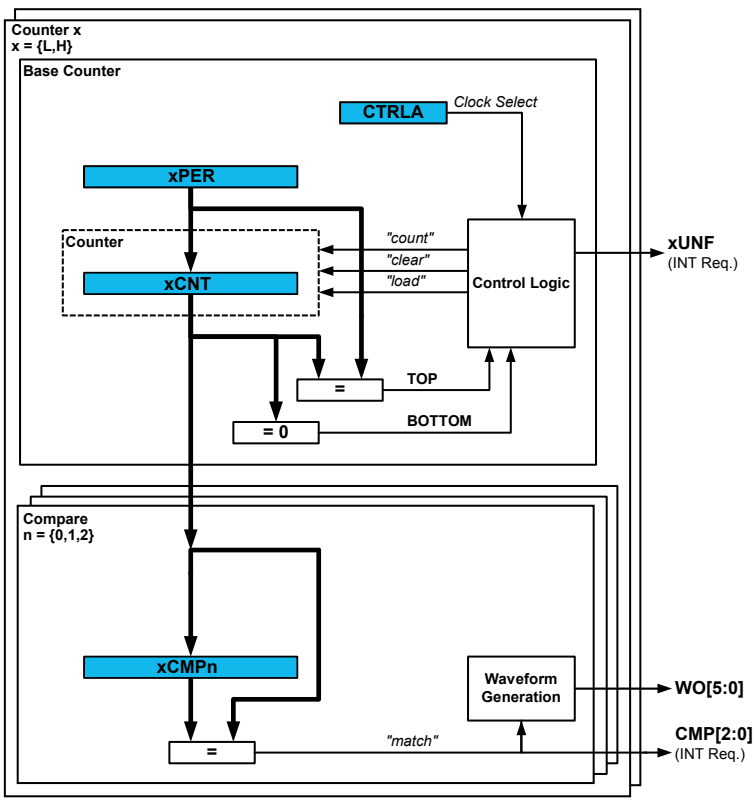
will only work with single slope down count. Split Mode does not support Event action controlled operation.

Split Mode Differences to Normal Mode

- Count
  - Down-count only
  - Timer/Counter counter high and low byte are independent (TCA_LCNT, TCA_HCNT)
- Waveform generation
  - Single Slope PWM only (TCA_CTRLB.WGMODE=SINGLESLOPE)
- Interrupt
  - No change for low byte Timer/Counter (TCA_LCNT)
  - Underflow interrupt for high byte Timer/Counter (TCA_HCNT)
  - No compare interrupt or flag for Compare n registers (TCA_HCMPn)
- Event actions: Not compatible
- Buffer registers and Buffer Valid flags: Unused
- Register access: Byte access to all registers.
- Temp register: Unused, 16-bit register of the Normal Mode are accessed as 8-bit 'TCA_H' and 'TCA_L', respectively.

Block Diagram

Figure 20-12. Timer/Counter Block Diagram Split Mode



Split Mode Initialization

Note: When shifting between Normal Mode and Split Mode, the functionality of some registers and bits change, but their values do not. For this reason, disabling the peripheral (TCA_CTRLA.ENABLE=0) and

doing a Hard Reset (TCA_CTRLESET.CMD=RESET) is recommended when changing the mode to avoid unexpected behavior.

To start using the timer/counter in basic Split Mode after a Hard Reset, follow these steps:

- Enable Split Mode by writing a '1' to the Split Mode Enable bit in the Control D register (TCA_CTRLD.SPLITM).
- Write a TOP value to the Period registers (TCA_PER.PERn)
- Enable the peripheral by writing a '1' to the Enable bit in the Control A register (TCA_CTRLA.ENABLE).
  The counter will start counting clock ticks according to the prescaler setting in the Clock Select bit field (TCA_CTRLA.CLKSEL)
- The counter values can be read from the Counter bit field in the Counter registers (TCA_CNTn.CNT).

Activating Split Mode results in changes to the functionality of some registers and register bits. The modifications are described in a separate register map. See Register Summary - Split Mode.

20.6.3. Events

The peripheral can take the following actions on an input Event:

- The counter counts positive edges of the Event signal.
- The counter counts both positive and negative edges of the Event signal.
- The counter counts prescaled clock cycles as long as the Event signal is high.
- The event signal controls the direction of counting.

The specific action is selected by writing to the Event Action bits in the Event Control register (TCA_EVCTRL.EVACT). Events as input are enabled by writing a '1' to the Enable Count on Event Input bit (TCA_EVCTRL.CNTEI).

Event controlled operation is not available in Split Mode.

20.6.4. Interrupts

Table 20-3. Available Interrupt Vectors and Sources in Normal Mode

Offset	Name	Vector Description	Conditions
0x00	OVF	Overflow and Compare match interrupt	The counter has reached its top value and wrapped to zero.
0x04	CMP0	Compare or capture channel 0 interrupt	Match between the counter value and the Compare 0 register.
0x06	CMP1	Compare or capture channel 1 interrupt	Match between the counter value and the Compare 1 register.
0x08	CMP2	Compare or capture channel 2 interrupt	Match between the counter value and the Compare 2 register.

Table 20-4. Available Interrupt Vectors and Sources in Split Mode

Offset	Name	Vector Description	Conditions
0x00	LUNF	Low-byte Underflow interrupt	Low-byte timer reaches BOTTOM.
0x02	HUNF	High-byte Underflow interrupt	High-byte timer reaches BOTTOM.

Offset	Name	Vector Description	Conditions
0x04	LCMP0	Compare or capture channel 0 interrupt	Match between the counter value and the low-byte of Compare 0 register.
0x06	LCMP1	Compare or capture channel 1 interrupt	Match between the counter value and the low-byte of Compare 1 register.
0x08	LCMP2	Compare or capture channel 2 interrupt	Match between the counter value and the low-byte of the Compare 2 register.

When an interrupt condition occurs, the corresponding Interrupt Flag is set in the Interrupt Flags register of the peripheral (*peripheral*_INTFLAGS).

An interrupt source is enabled or disabled by writing to the corresponding bit in the peripheral's Interrupt Control register (*peripheral*_INTCTRL).

An interrupt request is generated when the corresponding interrupt source is enabled and the Interrupt Flag is set. The interrupt request remains active until the Interrupt Flag is cleared. See the peripheral's INTFLAGS register for details on how to clear Interrupt Flags.

Note: Interrupt requests are only generated when Interrupts are enabled globally.

Related Links

AVR CPU on page 40
SREG on page 51

20.6.5. Sleep Mode Operation

The timer/counter will halt operation in all sleep modes.

20.6.6. Configuration Change Protection

Not applicable.

20.7. Register Summary - Normal Mode (CTRLD.SPLITM=0)

Offset	Name	Bit Pos.								
0x00	CTRLA	7:0					CLKSEL[2:0]			ENABLE
0x01	CTRLB	7:0		CMP2EN	CMP1EN	CMP0EN	ALUPD	WGMODE[2:0]		
0x02	CTRLC	7:0						CMP0	CMP1	CMP0
0x03	CTRLD	7:0								SPLITM
0x04	CTRLECLR	7:0					CMD[1:0]		LUPD	DIR
0x05	CTRLESET	7:0					CMD[1:0]		LUPD	DIR
0x06	CTRLFCLR	7:0					CMP2BV	CMP1BV	CMP0BV	PERBV
0x07	CTRLFSET	7:0					CMP2BV	CMP1BV	CMP0BV	PERBV
0x08	Reserved									
0x09	EVCTRL	7:0					EVACT[2:0]			CNTEI
0x0A	INTCTRL	7:0		CMP2	CMP1	CMP0				OVF
0x0B	INTFLAGS	7:0		CMP2	CMP1	CMP0				OVF
0x0C ... 0x0D	Reserved									
0x0E	DBGCTRL	7:0								DBGRUN
0x0F	TEMP	7:0	TEMP[7:0]							
0x10 ... 0x1F	Reserved									
0x20	CNT	7:0	CNT[7:0]							
0x21		15:8	CNT[15:8]							
0x22 ... 0x25	Reserved									
0x26	PER	7:0	PER[7:0]							
0x27		15:8	PER[15:8]							
0x28	CMP0	7:0	CMP[7:0]							
0x29		15:8	CMP[15:8]							
0x2A	CMP1	7:0	CMP[7:0]							
0x2B		15:8	CMP[15:8]							
0x2C	CMP2	7:0	CMP[7:0]							
0x2D		15:8	CMP[15:8]							
0x2E ... 0x35	Reserved									
0x36	PERBUF	7:0	PERBUF[7:0]							
0x37		15:8	PERBUF[15:8]							
0x38	CMPBUF0	7:0	CMPBUF[7:0]							
0x39		15:8	CMPBUF[15:8]							
0x3A	CMPBUF1	7:0	CMPBUF[7:0]							
0x3B		15:8	CMPBUF[15:8]							
0x3C	CMPBUF2	7:0	CMPBUF[7:0]							
0x3D		15:8	CMPBUF[15:8]							

20.8. Register Description - Normal Mode

20.8.1. Control A

Name: CTRLA
Offset: 0x00
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
					CLKSEL[2:0]			ENABLE
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bits 3:1 – CLKSEL[2:0]: Clock Select
These bits select the clock frequency for the timer/counter.

Value	Name	Description
0x0	DIV1	$f_{TCA} = f_{CLK\_PER}/1$
0x1	DIV2	$f_{TCA} = f_{CLK\_PER}/2$
0x2	DIV4	$f_{TCA} = f_{CLK\_PER}/4$
0x3	DIV8	$f_{TCA} = f_{CLK\_PER}/8$
0x4	DIV16	$f_{TCA} = f_{CLK\_PER}/16$
0x5	DIV64	$f_{TCA} = f_{CLK\_PER}/64$
0x6	DIV256	$f_{TCA} = f_{CLK\_PER}/256$
0x7	DIV1024	$f_{TCA} = f_{CLK\_PER}/1024$

Bit 0 – ENABLE: Enable

Value	Description
0	The peripheral is disabled
1	The peripheral is enabled

20.8.2. Control B - Normal Mode

Name: CTRLB
Offset: 0x01
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
		CMP2EN	CMP1EN	CMP0EN	ALUPD	WGMODE[2:0]		
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bit 6 – CMP2EN: Capture 2 Enable
See CMP0EN.

Bit 5 – CMP1EN: Compare 1 Enable
See CMP0EN.

Bit 4 – CMP0EN: Compare 0 Enable
Setting the CMPnEN bits in the FRQ or PWM waveform generation mode of operation will override the port output register for the corresponding TCn output pin.

Bit 3 – ALUPD: Auto Lock Update
The Auto Lock Update feature controls the Lock Update (LUPD) bit in the CTRLE register. When ALUPD is set to 1, LUPD will be set to 1 until the Buffer Valid (CMPnBV) bit of all enabled compare channels is 1. This event will clear LUPD, and it will remain cleared until the next UPDATE condition, where the buffer values will be transferred to the CMPn registers and LUPD will be set to 1 again. This makes sure that CMPnBUF register values are not transferred to the CMPn registers until all enabled compare buffers are written.

Bits 2:0 – WGMODE[2:0]: Waveform Generation Mode
These bits select the waveform generation mode, and control the counting sequence of the counter, TOP value, UPDATE condition, interrupt condition, and type of waveform that is generated.

No waveform generation is performed in the normal mode of operation. For all other modes, the result from the waveform generator will only be directed to the port pins if the corresponding CMPnEN bit has been set to enable this. The port pin direction must be set as output and output must be enabled in portmux.

Table 20-5. Timer Waveform Generation Mode

WGMODE[2:0]	Group Configuration	Mode of Operation	Top	Update	OVF
000	NORMAL	Normal	PER	TOP	TOP
001	FRQ	Frequency	CMP0	TOP	TOP
010		Reserved	-	-	-
011	SINGLESLOPE	Single-slope PWM	PER	BOTTOM	BOTTOM
100		Reserved	-	-	-
101	DSTOP	Dual-slope PWM	PER	BOTTOM	TOP

WGMODE[2:0]	Group Configuration	Mode of Operation	Top	Update	OVF
110	DSBOTH	Dual-slope PWM	PER	BOTTOM	TOP and BOTTOM
111	DSBOTTOM	Dual-slope PWM	PER	BOTTOM	BOTTOM

20.8.3. Control C - Normal Mode

Name: CTRLC
Offset: 0x02
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
						CMP0	CMP1	CMP0
Access						R/W	R/W	R/W
Reset						0	0	0

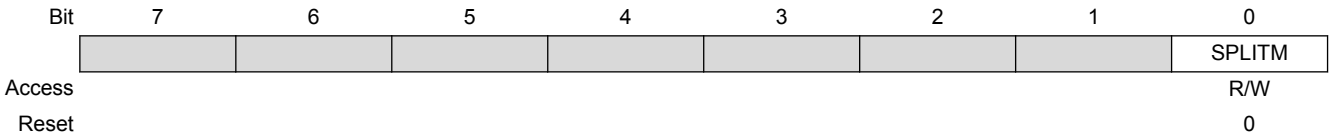
Bit 2 – CMP0: Compare Output Value 2
See CMP0

Bit 1 – CMP1: Compare Output Value 1
See CMP0

Bit 0 – CMP0: Compare Output Value 0
The CMPn bits allow direct access to the waveform generator's output compare value when the timer/counter is not enabled. This is used to set or clear the WG output value when the timer/counter is not running.

20.8.4. Control D

Name: CTRLD
Offset: 0x03
Reset: 0x00
Property: -



Bit 0 – SPLITM: Enable Split Mode
This bit set the timer/counter in split mode operation. It will then work as two 8-bit timer/counters. Note that the register map will change compared to normal 16-bit mode.

20.8.5. Control Register E Clear - Normal Mode

The individual status bit can be cleared by writing a one to its bit location. This allows each bit to be cleared without use of a read-modify-write operation on a single register.
Each Status bit can be read out either by reading TCA.CTRLESET or TCA.CTRLECLR.

Name: CTRLECLR
Offset: 0x04
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
					CMD[1:0]		LUPD	DIR
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bits 3:2 – CMD[1:0]: Command
These bits can be used for software control of update, restart, and reset of the timer/counter. The command bits are always read as zero.

Table 20-6. Command Selections

CMD[1:0]	Group Configuration	Command Action
00	NONE	None
01	UPDATE	Force update
10	RESTART	Force restart
11	RESET	Force hard reset (ignored if T/C is enabled)

Bit 1 – LUPD: Lock Update
When this bit is set, no update of the buffered registers is performed, even though an UPDATE condition has occurred. Locking the update ensures that all buffers, are valid before an update is performed.

Bit 0 – DIR: Counter Direction
When zero, this bit indicates that the counter is counting up (incrementing). A one indicates that the counter is in the down-counting (decrementing) state.

Normally this bit is controlled in hardware by the waveform generation mode or by event actions, but this bit can also be changed from software.

20.8.6. Control Register E Set - Normal Mode

The individual status bit can be set by writing a '1' to its bit location. This allows each bit to be set without use of a read-modify-write operation on a single register.
Each Status bit can be read out either by reading TCA.CTRLESET or TCA.CTRLECLR.

Name: CTRLESET
Offset: 0x05
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
					CMD[1:0]		LUPD	DIR
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bits 3:2 – CMD[1:0]: Command
These bits can be used for software control of update, restart, and reset of the timer/counter. The command bits are always read as zero.

Table 20-7. Command Selections

CMD[1:0]	Group Configuration	Command Action
00	NONE	None
01	UPDATE	Force update
10	RESTART	Force restart
11	RESET	Force hard reset (ignored if T/C is enabled)

Bit 1 – LUPD: Lock Update
When this bit is set, no update of the buffered registers is performed, even though an UPDATE condition has occurred. Locking the update ensures that all buffers, are valid before an update is performed.

Bit 0 – DIR: Counter Direction
When zero, this bit indicates that the counter is counting up (incrementing). A one indicates that the counter is in the down-counting (decrementing) state.

Normally this bit is controlled in hardware by the waveform generation mode or by event actions, but this bit can also be changed from software.

20.8.7. Control Register F Clear

The individual status bit can be cleared by writing a one to its bit location. This allows each bit to be cleared without use of a read-modify-write operation on a single register.

Name: CTRLFCLR
Offset: 0x06
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
					CMP2BV	CMP1BV	CMP0BV	PERBV
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit 3 – CMP2BV: Compare 2 Buffer Valid
See CMP0BV.

Bit 2 – CMP1BV: Compare 1 Buffer Valid
See CMP0BV.

Bit 1 – CMP0BV: Compare 0 Buffer Valid
The CMPnBV bits are set when a new value is written to the corresponding CMPnBUF register. These bits are automatically cleared on an UPDATE condition.

Bit 0 – PERBV: Period Buffer Valid
This bit is set when a new value is written to the PERB register. This bit is automatically cleared on an UPDATE condition.

20.8.8. Control Register F Set

The individual status bit can be set by writing a one to its bit location. This allows each bit to be set without use of a read-modify-write operation on a single register.

Name: CTRLFSET
Offset: 0x07
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
					CMP2BV	CMP1BV	CMP0BV	PERBV
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit 3 – CMP2BV: Compare 2 Buffer Valid
See CMP0BV.

Bit 2 – CMP1BV: Compare 1 Buffer Valid
See CMP0BV.

Bit 1 – CMP0BV: Compare 0 Buffer Valid
The CMPnBV bits are set when a new value is written to the corresponding CMPnBUF register. These bits are automatically cleared on an UPDATE condition.

Bit 0 – PERBV: Period Buffer Valid
This bit is set when a new value is written to the PERB register. This bit is automatically cleared on an UPDATE condition.

20.8.9. Event Control

Name: EVCTRL
Offset: 0x09
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
					EVACT[2:0]			CNTEI
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bits 3:1 – EVACT[2:0]: Event Action
These bits define on what type of event action the counter will increment or decrement.

Table 20-8. Timer Event Action Selection

EVACT[2:0]	Group Configuration	Description
00	EVACT_POSEDGE	Count on postive edge event
01	EVACT_ANYEDGE	Count on any edge event
10	EVACT_HIGHLVL	Count on high level event
11	EVACT_UPDOWN	Externally controlled up/down count

Bit 0 – CNTEI: Enable count on event input
This bit enables count on event input according to setting of EVACT[1:0].

20.8.10. Interrupt Control Register - Normal Mode

Name: INTCTRL
Offset: 0x0A
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
		CMP2	CMP1	CMP0				OVF
Access		R/W	R/W	R/W				R/W
Reset		0	0	0				0

Bit 6 – CMP2: Compare Channel 2 Interrupt Enable

Bit 5 – CMP1: Compare Channel 1 Interrupt Enable

Bit 4 – CMP0: Compare Channel 0 Interrupt Enable

Bit 0 – OVF: Timer Overflow/Underflow Interrupt Enable

20.8.11. Interrupt Flag Register - Normal Mode

The individual status bit can be set by writing a '1'e to its bit location. This allows each bit to be set without use of a read-modify-write operation on a single register.

Name: INTFLAGS
Offset: 0x0B
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
		CMP2	CMP1	CMP0				OVF
Access		R/W	R/W	R/W				R/W
Reset		0	0	0				0

Bit 6 – CMP2: Compare Channel 2 Interrupt Flag
See CMP0 flag description.

Bit 5 – CMP1: Compare Channel 1 Interrupt Flag
See CMP0 flag description.

Bit 4 – CMP0: Compare Channel 0 Interrupt Flag
The compare interrupt flag (CMPn) is set on a compare match on the corresponding compare channel.

For all modes of operation, the CMPn flag will be set when a compare match occurs between the count register (CNT) and the corresponding compare register (CMPn). The CMPn flag will not be cleared automatically and has to be cleared by software. This is done by writing a one to its bit location.

Bit 0 – OVF: Overflow/Underflow Interrupt Flag
This flag is set either on a TOP (overflow) or BOTTOM (underflow) condition, depending on the WGMODE setting. OVF is not automatically cleared and needs to be cleared by software. This is done by writing a one to its bit location.

20.8.12. Debug Control Register

Name: DBGCTRL
Offset: 0x0E
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
								DBGRUN
Access								R/W
Reset								0

Bit 0 – DBGRUN: Run in Debug
When this bit is set, the module will not be stopped when device is set in break for debugging.

20.8.13. Temporary bits for 16-bit Access

The Temporary register is used by the CPU for single-cycle, 16-bit access to the 16-bit registers of this peripheral. It can also be read and written by software. See also Accessing 16-bit Registers. There is one common Temporary register for all the 16-bit registers of this peripheral.

Name: TEMP
Offset: 0x0F
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	TEMP[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – TEMP[7:0]: Temporary Bits for 16-bit Access

20.8.14. Counter Register - Normal Mode

The TCA.CNTL and TCA.CNTH register pair represents the 16-bit value, TCA.CNT. The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + 0x01. For more details on reading and writing 16-bit registers, refer to Accessing 16-bit Registers.

CPU and PDI write access has priority over internal updates of the register.

Name: CNT
Offset: 0x20
Reset: 0x00
Property: -

Bit	15	14	13	12	11	10	9	8
	CNT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:8 – CNT[15:8]: Counter high byte
These bits hold the MSB of the 16-bit counter register.

Bits 7:0 – CNT[7:0]: Counter low byte
These bits hold the LSB of the 16-bit counter register.

20.8.15. Period Register - Normal Mode

TCA.PER contains the 16-bit TOP value in the timer/counter.

The TCA.PERL and TCA.PERH register pair represents the 16-bit value, TCA.PER. The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + 0x01. For more details on reading and writing 16-bit registers, refer to Accessing 16-bit Registers.

Name: PER
Offset: 0x26
Reset: 0xFFFF
Property: -

Bit	15	14	13	12	11	10	9	8
	PER[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bit	7	6	5	4	3	2	1	0
	PER[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bits 15:8 – PER[15:8]: Periodic high byte
These bits hold the MSB of the 16-bit period register.

Bits 7:0 – PER[7:0]: Periodic low byte
These bits hold the LSB of the 16-bit period register.

20.8.16. Compare n Register - Normal Mode

This register is continuously compared to the counter value. Normally, the outputs from the comparators are then used for generating waveforms.

TCA.CMPn registers are updated with the buffer value from their corresponding CMPnBUF register when an UPDATE condition occurs.

The TCA.CMPnL and TCA.CMPnH register pair represents the 16-bit value, TCA.CMPn. The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + 0x01. For more details on reading and writing 16-bit registers, refer to Accessing 16-bit Registers.

Name: CMP0, CMP1, CMP2
Offset: 0x28 + n*0x02 [n=0..2]
Reset: 0x00
Property: -

Bit	15	14	13	12	11	10	9	8
	CMP[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CMP[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:8 – CMP[15:8]: Compare high byte
These bits hold the MSB of the 16-bit compare register.

Bits 7:0 – CMP[7:0]: Compare low byte
These bits hold the LSB of the 16-bit compare register.

20.8.17. Period Buffer Register

This register serves as the buffer for the period register (TCA.PER). Accessing this register using the CPU or PDI will affect the PERBV flag.

The TCA.PERBUFL and TCA.PERBUFH register pair represents the 16-bit value, TCA.PERBUF. The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + 0x01. For more details on reading and writing 16-bit registers, refer to Accessing 16-bit Registers.

Name: PERBUF
Offset: 0x36
Reset: 0x00
Property: -

Bit	15	14	13	12	11	10	9	8
	PERBUF[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bit	7	6	5	4	3	2	1	0
	PERBUF[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bits 15:8 – PERBUF[15:8]: Period Buffer high byte
These bits hold the MSB of the 16-bit period buffer register.

Bits 7:0 – PERBUF[7:0]: Period Buffer low byte
These bits hold the LSB of the 16-bit period buffer register.

20.8.18. Compare n Buffer Register

This register serves as the buffer for the associated compare registers (TCA.CMPn). Accessing any of these registers using the CPU or PDI will affect the corresponding CMPnBV status bit.

The TCA.CMPnBUFL and TCA.CMPnBUFH register pair represents the 16-bit value, TCA.CMPnBUF. The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + 0x01. For more details on reading and writing 16-bit registers, refer to Accessing 16-bit Registers.

Name: CMPBUF0, CMPBUF1, CMPBUF2
Offset: 0x38 + n*0x02 [n=0..2]
Reset: 0x00
Property: -

Bit	15	14	13	12	11	10	9	8
	CMPBUF[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CMPBUF[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:8 – CMPBUF[15:8]: Compare high byte
These bits hold the MSB of the 16-bit compare buffer register.

Bits 7:0 – CMPBUF[7:0]: Compare low byte
These bits hold the LSB of the 16-bit compare buffer register.

20.9. Register Summary - Split Mode (CTRLD.SPLITM=1)

Offset	Name	Bit Pos.								
0x00	CTRLA	7:0					CLKSEL[2:0]			ENABLE
0x01	CTRLB	7:0		HCMP2EN	HCMP1EN	HCMP0EN		LCMP2EN	LCMP1EN	LCMP0EN
0x02	CTRLC	7:0		HCMP2	HCMP1	HCMP0		LCMP2	LCMP1	LCMP0
0x03	CTRLD	7:0								SPLITM
0x04	CTRLECLR	7:0					CMD[1:0]			
0x05	CTRLESET	7:0					CMD[1:0]			
0x06 ... 0x09	Reserved									
0x0A	INTCTRL	7:0		LCMP2	LCMP1	LCMP0			HUNF	LUNF
0x0B	INTFLAGS	7:0		LCMP2	LCMP1	LCMP0			HUNF	LUNF
0x0C ... 0x0D	Reserved									
0x0E	DBGCTRL	7:0								DBGRUN
0x0F ... 0x1F	Reserved									
0x20	LCNT	7:0	LCNT[7:0]							
0x21	HCNT	7:0	HCNT[7:0]							
0x22 ... 0x25	Reserved									
0x26	LPER	7:0	LPER[7:0]							
0x27	HPER	7:0	HPER[7:0]							
0x28	LCMP0	7:0	LCMP[7:0]							
0x29	HCMP0	7:0	HCMP[7:0]							
0x2A	LCMP1	7:0	LCMP[7:0]							
0x2B	HCMP1	7:0	HCMP[7:0]							
0x2C	LCMP2	7:0	LCMP[7:0]							
0x2D	HCMP2	7:0	HCMP[7:0]							

20.10. Register Description - Split Mode

20.10.1. Control A

Name: CTRLA
Offset: 0x00
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
					CLKSEL[2:0]			ENABLE
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bits 3:1 – CLKSEL[2:0]: Clock Select
These bits select the clock frequency for the timer/counter.

Value	Name	Description
0x0	DIV1	$f_{TCA} = f_{CLK\_PER}/1$
0x1	DIV2	$f_{TCA} = f_{CLK\_PER}/2$
0x2	DIV4	$f_{TCA} = f_{CLK\_PER}/4$
0x3	DIV8	$f_{TCA} = f_{CLK\_PER}/8$
0x4	DIV16	$f_{TCA} = f_{CLK\_PER}/16$
0x5	DIV64	$f_{TCA} = f_{CLK\_PER}/64$
0x6	DIV256	$f_{TCA} = f_{CLK\_PER}/256$
0x7	DIV1024	$f_{TCA} = f_{CLK\_PER}/1024$

Bit 0 – ENABLE: Enable

Value	Description
0	The peripheral is disabled
1	The peripheral is enabled

20.10.2. Control B - Split Mode

Name: CTRLB
Offset: 0x01
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
		HCMP2EN	HCMP1EN	HCMP0EN		LCMP2EN	LCMP1EN	LCMP0EN
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		0	0	0		0	0	0

Bit 6 – HCMP2EN: High-byte Compare 2 Enable
See LCMP0EN.

Bit 5 – HCMP1EN: High-byte Compare 1 Enable
See LCMP0EN.

Bit 4 – HCMP0EN: High-byte Compare 0 Enable
See LCMP0EN.

Bit 2 – LCMP2EN: Low-byte Compare 2 Enable
See LCMP0EN.

Bit 1 – LCMP1EN: Low-byte Compare 1 Enable
See LCMP0EN.

Bit 0 – LCMP0EN: Low-byte Compare 0 Enable
Setting the LCMPnEN/HCMPnEN bits in the FRQ or PWM waveform generation mode of operation will override the port output register for the corresponding WOn pin.

20.10.3. Control C - Split Mode

Name: CTRLC
Offset: 0x02
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
		HCMP2	HCMP1	HCMP0		LCMP2	LCMP1	LCMP0
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		0	0	0		0	0	0

Bit 6 – HCMP2: High-byte Compare 2 Output Value
See LCMP0A

Bit 5 – HCMP1: High-byte Compare 1 Output Value
See LCMP0

Bit 4 – HCMP0: High-byte Compare 0 Output Value
See LCMP0

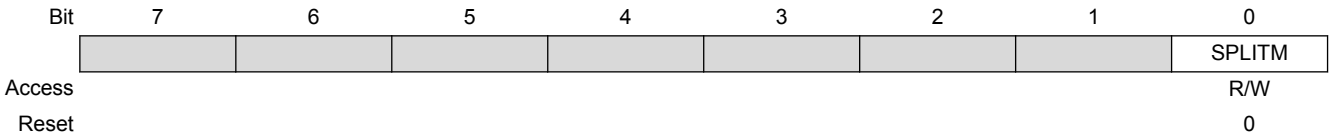
Bit 2 – LCMP2: Low-byte Compare 2 Output Value
See LCMP0

Bit 1 – LCMP1: Low-byte Compare 1 Output Value
See LCMP0

Bit 0 – LCMP0: Low-byte Compare 0 Output Value
The LCMPn/HCMPn bits allow direct access to the waveform generator's output compare value when the timer/counter is not enabled. This is used to set or clear the WOn output value when the timer/counter is not running.

20.10.4. Control D

Name: CTRLD
Offset: 0x03
Reset: 0x00
Property: -



Bit 0 – SPLITM: Enable Split Mode
This bit set the timer/counter in split mode operation. It will then work as two 8-bit timer/counters. Note that the register map will change compared to normal 16-bit mode.

20.10.5. Control Register E Clear - Split Mode

The individual status bit can be cleared by writing a '1' to its bit location. This allows each bit to be cleared without use of a read-modify-write operation on a single register.
Each Status bit can be read out either by reading TCA.CTRLESET or TCA.CTRLECLR.

Name: CTRLECLR
Offset: 0x04
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
					CMD[1:0]			
Access					R/W	R/W		
Reset					0	0		

Bits 3:2 – CMD[1:0]: Command
These bits can be used for software control of update, restart, and reset of the timer/counter. The command bits are always read as zero.

Table 20-9. Command Selections

CMD[1:0]	Group Configuration	Command Action
00	NONE	None
01	UPDATE	Force update
10	RESTART	Force restart
11	RESET	Force hard reset (ignored if T/C is enabled)

20.10.6. Control Register E Set - Split Mode

The individual status bit can be set by writing a '1' to its bit location. This allows each bit to be set without use of a read-modify-write operation on a single register.
Each Status bit can be read out either by reading TCA.CTRLESET or TCA.CTRLECLR.

Name: CTRLESET
Offset: 0x05
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
					CMD[1:0]			
Access					R/W	R/W		
Reset					0	0		

Bits 3:2 – CMD[1:0]: Command
These bits can be used for software control of update, restart, and reset of the timer/counter. The command bits are always read as zero.

Table 20-10. Command Selections

CMD[1:0]	Group Configuration	Command Action
00	NONE	None
01	UPDATE	Force update
10	RESTART	Force restart
11	RESET	Force hard reset (ignored if T/C is enabled)

20.10.7. Interrupt Control Register - Split Mode

Name: INTCTRL
Offset: 0x0A
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
		LCMP2	LCMP1	LCMP0			HUNF	LUNF
Access		R/W	R/W	R/W			R/W	R/W
Reset		0	0	0			0	0

Bit 6 – LCMP2: Low-byte Compare Channel 0 Interrupt Enable

Bit 5 – LCMP1: Low-byte Compare Channel 1 Interrupt Enable

Bit 4 – LCMP0: Low-byte Compare Channel 0 Interrupt Enable

Bit 1 – HUNF: High-byte Underflow Interrupt Enable

Bit 0 – LUNF: Low-byte Underflow Interrupt Enable

20.10.8. Interrupt Flag Register - Split Mode

The individual status bit can be set by writing a one to its bit location. This allows each bit to be set without use of a read-modify-write operation on a single register.

Name: INTFLAGS
Offset: 0x0B
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
		LCMP2	LCMP1	LCMP0			HUNF	LUNF
Access		R/W	R/W	R/W			R/W	R/W
Reset		0	0	0			0	0

Bit 6 – LCMP2: Low-byte Compare Channel 0 Interrupt Flag
See LCMP0 flag description.

Bit 5 – LCMP1: Low-byte Compare Channel 0 Interrupt Flag
See LCMP0 flag description.

Bit 4 – LCMP0: Low-byte Compare Channel 0 Interrupt Flag
The compare interrupt flag (LCMPn) is set on a compare match on the corresponding compare channel.

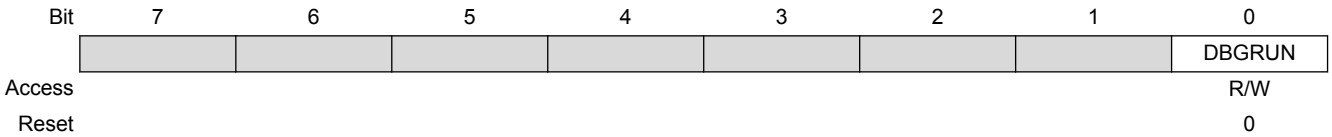
For all modes of operation, the LCMPn flag will be set when a compare match occurs between the Low-byte count register (LCNT) and the corresponding compare register (LCMPn). The LCMPn flag will not be cleared automatically and has to be cleared by software. This is done by writing a one to its bit location.

Bit 1 – HUNF: High-byte Underflow Interrupt Flag
This flag is set on a high-byte timer BOTTOM (underflow) condition. HUNF is not automatically cleared and needs to be cleared by software. This is done by writing a one to its bit location.

Bit 0 – LUNF: Low-byte Underflow Interrupt Flag
This flag is set on a low-byte timer BOTTOM (underflow) condition. LUNF is not automatically cleared and needs to be cleared by software. This is done by writing a one to its bit location.

20.10.9. Debug Control Register

Name: DBGCTRL
Offset: 0x0E
Reset: 0x00
Property: -



Bit 0 – DBGRUN: Run in Debug
When this bit is set, the module will not be stopped when device is set in break for debugging.

20.10.10. Low-byte Timer Counter Register - Split Mode

TCA.LCNT contains the counter value in low-byte timer. CPU and PDI write access has priority over count, clear, or reload of the counter.

Name: LCNT
Offset: 0x20
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	LCNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – LCNT[7:0]: Counter value for low-byte timer
These bits define the counter value of the low-byte timer.

20.10.11. High-byte Timer Counter Register - Split Mode

TCA.HCNT contains the counter value in high-byte timer. CPU and PDI write access has priority over count, clear, or reload of the counter.

Name: HCNT
Offset: 0x21
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	HCNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – HCNT[7:0]: Counter value for high-byte timer
These bits define the counter value in high-byte timer.

20.10.12. Low-byte Timer Period Register - Split Mode

The TCA.LPER register contains the TOP value of low-byte timer.

Name: LPER
Offset: 0x26
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	LPER[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bits 7:0 – LPER[7:0]: Period value low-byte timer
These bits hold the TOP value of low-byte timer.

20.10.13. High-byte Period Register - Split Mode

The TCA.HPER register contains the TOP value of high-byte timer.

Name: HPER
Offset: 0x27
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	HPER[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bits 7:0 – HPER[7:0]: Period value high-byte timer

These bits hold the TOP value of high-byte timer.

20.10.14. Compare Register n for low-byte Timer - Split Mode

The TCA.LCMPn register represents the compare value of compare channel n for low-byte Timer. This register is continuously compared to the counter value of low-byte timer, LCNT. Normally, the outputs from the comparators are then used for generating waveforms.

Name: LCMPn
Offset: 0x28 + n*0x02 [n=0..2]
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	LCMP[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – LCMP[7:0]: Compare value of channel n
These bits hold the compare value of channel n that is compared to LCNT.

20.10.15. High-byte Compare Register n - Split Mode

The TCA.HCMPn register represents the compare value of compare channel n for high-byte Timer. This register is continuously compared to the counter value of high-byte timer, HCNT. Normally, the outputs from the comparators are then used for generating waveforms.

Name: HCMP0, HCMP1, HCMP2
Offset: 0x29 + n*0x02 [n=0..2]
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	HCMP[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – HCMP[7:0]: Compare value of channel n

These bits hold the compare value of channel n that is compared to HCNT.

21. TCB - 16-bit Timer/Counter Type B

21.1. Overview

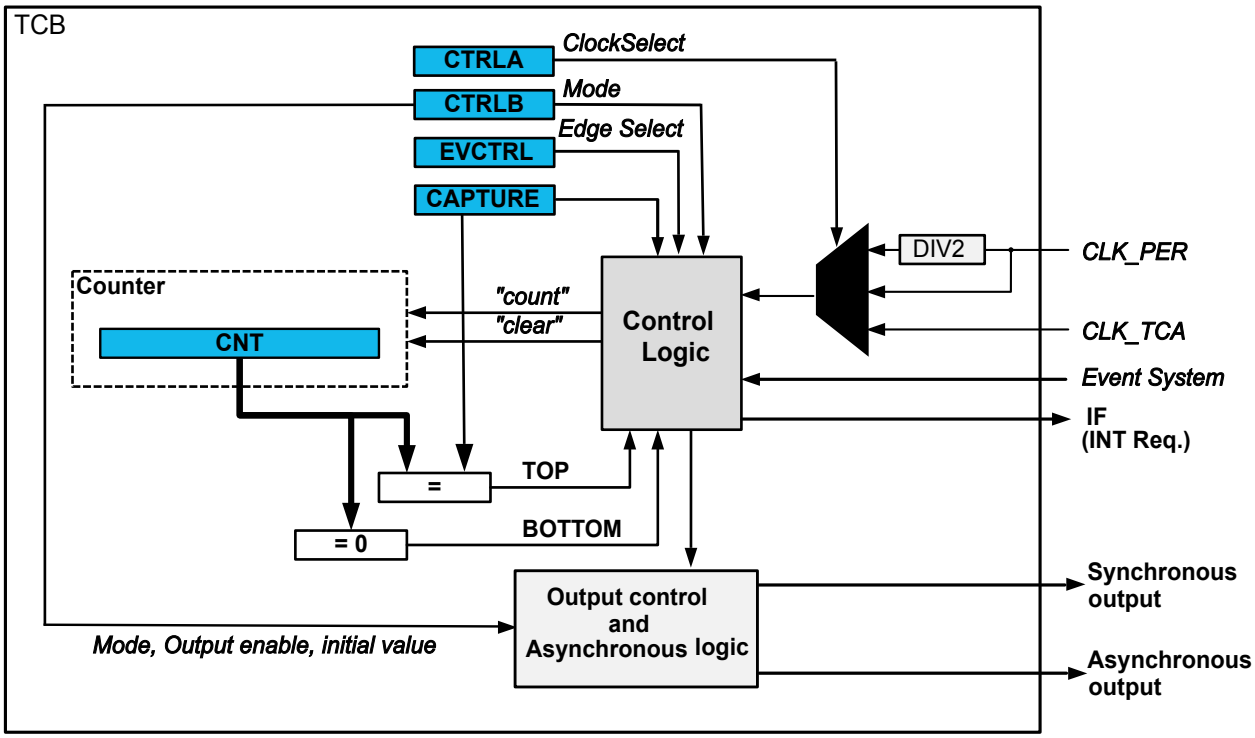
The capabilities of the 16-bit Timer/Counter type B (TCB) include frequency and waveform generation, and input capture with time and frequency measurement of digital signals. It consists of a base counter and control logic which can be set in one of eight different modes, each mode providing unique functionality. The base counter is clocked by the peripheral clock with optional prescaling.

21.2. Features

- 16-bit counter operation modes:
  - Input capture
    - Input capture with noise canceling
    - Frequency capture
    - Pulse width capture
    - Frequency and pulse width capture
  - Periodic interrupt
  - Timeout checking
  - 8-bit Pulse Width Modulation (PWM)
  - Single shot
    - Pull pin high while the counter is counting
- Optional: operation synchronous with TCA

21.3. Block Diagram

Figure 21-1. Timer/Counter Type B Block Diagram



21.4. Signal Description

Signal	Description	Type
W0	Digital asynchronous output	Waveform Output

Related Links

I/O Multiplexing and Considerations on page 19

21.5. Product Dependencies

In order to use this peripheral, other parts of the system must be configured correctly, as described below.

Table 21-1. TCB Product Dependencies

Dependency	Applicable	Peripheral
Clocks	Yes	CLKCTRL
I/O Lines and Connections	No	-
Interrupts	Yes	CPUINT
Events	Yes	EVSYS
Debug	Yes	UPDI

Related Links

Clocks on page 230
Interrupts on page 54
Events on page 181
Debug Operation on page 230

21.5.1. Clocks

This peripheral uses the system's peripheral clock CLK_PER. The peripheral has its own local prescaler, or can be configured to run off the prescaled clock signal of the Timer Counter type A (TCA).

Related Links

CLKCTRL - Clock Controller on page 68

21.5.2. I/O Lines and Connections

Not applicable.

21.5.3. Interrupts

Using the interrupts of this peripheral requires the Interrupt Controller to be configured first.

Related Links

CPUINT - CPU Interrupt Controller on page 97
SREG on page 51
Interrupts on page 136

21.5.4. Events

The events of this peripheral are connected to the Event System.

Related Links

EVSYS - Event System on page 109

21.5.5. Debug Operation

When the CPU is halted in debug mode, this peripheral will halt normal operation. This peripheral can be forced to continue operation during debugging.

Related Links

UPDI - Unified Program Debug Interface on page 490

21.6. Functional Description

21.6.1. Principle of Operation

Depending on the mode of operation, the counter/timer is cleared, reloaded, incremented or decremented at each clock tick or Event input.

21.6.2. Basic Operation

21.6.2.1. Definitions

The following definitions are used throughout the documentation:

Table 21-2. Timer/Counter Definitions

Name	Description
BOTTOM	The counter reaches BOTTOM when it becomes zero.
MAX	The counter reaches MAXimum when it becomes all ones.
TOP	The counter reaches TOP when it becomes equal to the highest value in the count sequence. Depending on the waveform generator mode, the TOP value can be equal to the Period register a Compare channel n register setting.
UPDATE	The timer/counter signals an update when it reaches BOTTOM or TOP, depending on the waveform generator mode.
CNT	Counter register value
CAPT	Capture/compare register value

In general, the term “timer” is used when the timer/counter clock control is handled by an internal source, and the term “counter” is used when the clock control is handled externally (e.g. counting external Events).

21.6.2.2. Initialization

By default the TCB is in Periodic Interrupt mode. Follow these steps to start using it:

- Write a TOP value to the Compare/Capture register (TCB_CC.CC).
- Enable the counter by writing a '1' to the Enable bit in the Control A register (TCB_CTRLA.ENABLE).
  The counter will start counting clock ticks according to the prescaler setting in the Clock Select bit field (CTRLA.CLKSEL).
- The counter value can be read from the Count register (TCB_CNT.CNT). The peripheral will generate an interrupt when the CNT value reaches TOP.

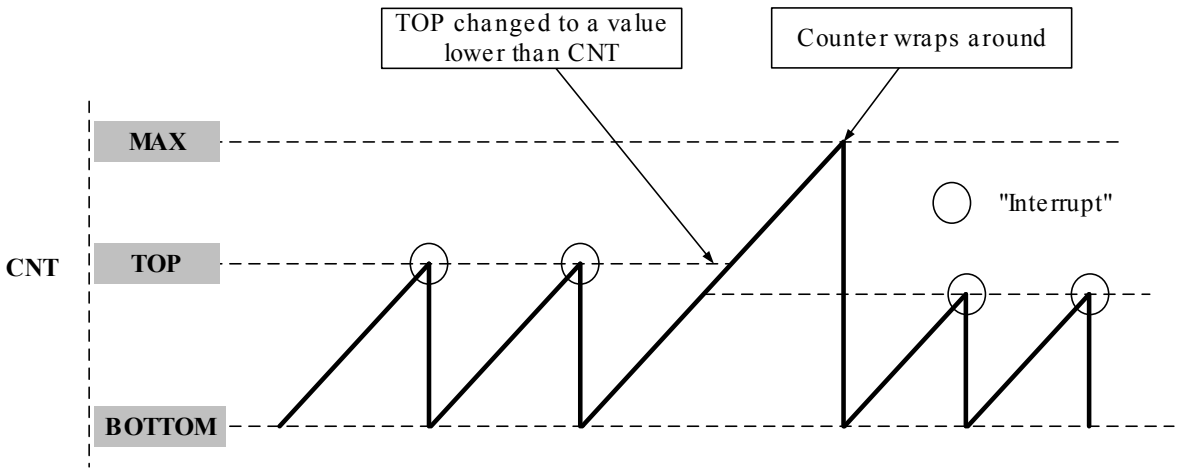
21.6.2.3. Modes

The timer can be put in eight different modes which are explained below.

Periodic Interrupt Mode

In the periodic interrupt mode the counter counts to the capture value and restarts from zero. Interrupt is generated when counter is equal to TOP. If TOP is updated to a value lower than count, the counter will continue till MAX and wraps around without generating interrupt.

Figure 21-2. Periodic Interrupt Mode



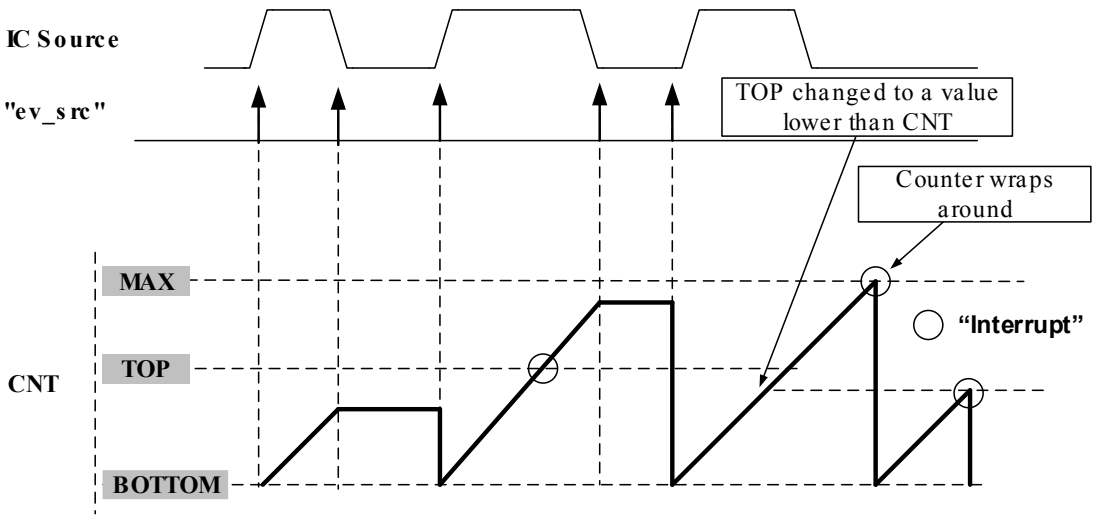
Timeout Check Mode

In timeout check mode, the counter counts till MAX and wraps around. On the first edge the counter will start and on the second it will stop. If count register reaches TOP before the second edge, an interrupt will be generated.

Going out of freeze and clearing flag:

1. Count register is initialized and started on the first edge and stopped on the second edge.
2. Reading count, capture and writing run bit will have no effect.

Figure 21-3. Timeout Check Mode



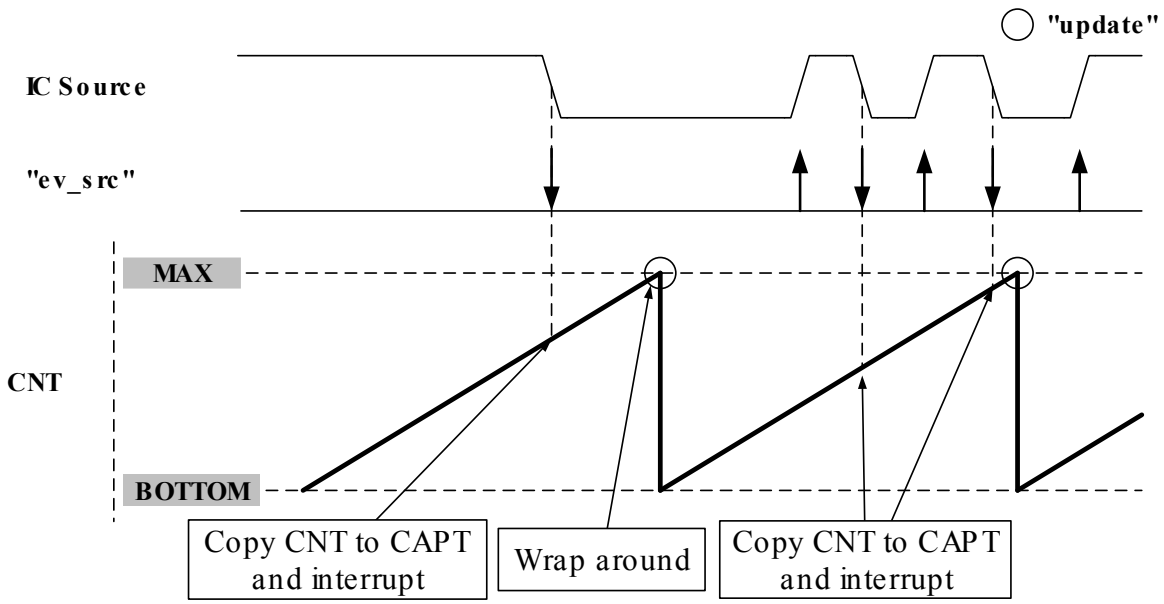
Input Capture on Event

The counter will count from BOTTOM to MAX continuously and when an event occurs the counter value will be transferred to the CC register and interrupt is generated. The module has two asynchronous edge detectors and the user can set up either posedge, negedge.

The figure below shows the input capture unit configured to capture on negedge of capture source. Interrupt is cleared when higher byte of capture register is read in this mode.

It is recommended to write zero to CNT register while entering this mode from any other mode.

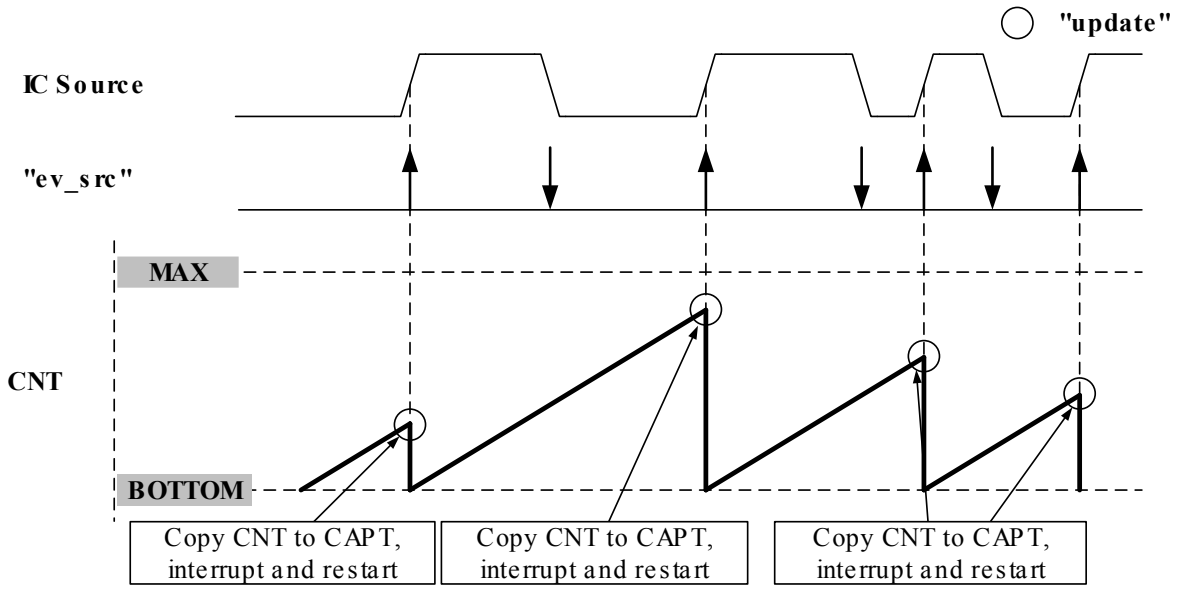
Figure 21-4. Input Capture on Event



Input Capture Frequency Measurement

In the input capture frequency measurement mode the timer will do a capture and a restart on each positive or negative edge. Interrupt is cleared when high byte of compare/capture register is read in this mode.

Figure 21-5. Input Capture Frequency Measurement

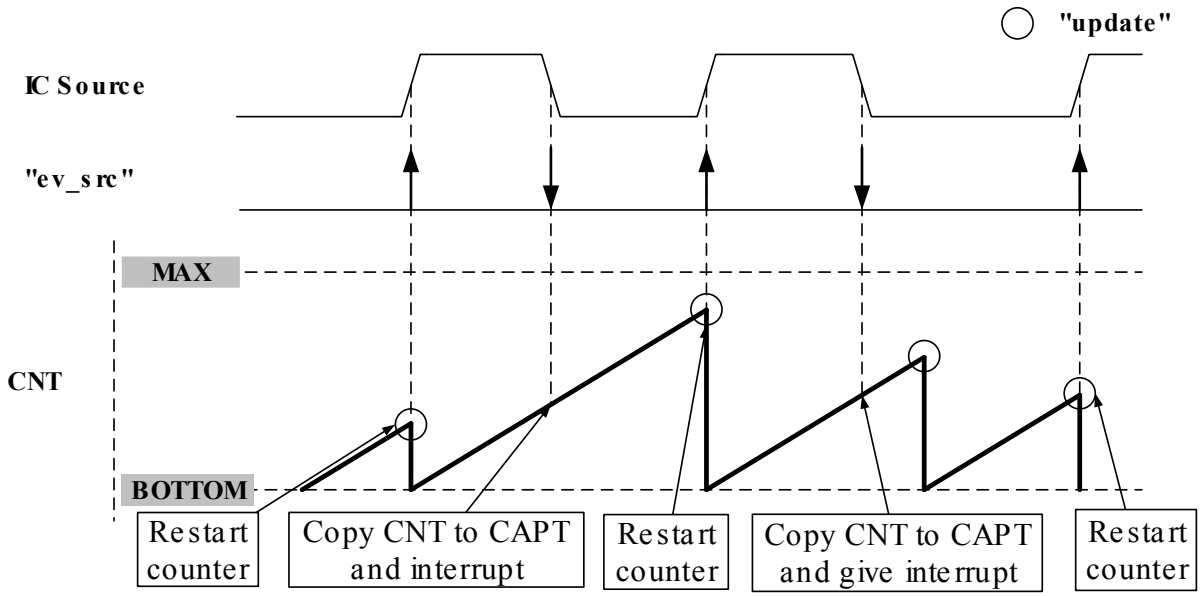


Input Capture Pulse Width Measurement

The input capture pulse width measurement will restart the counter on a positive edge and do a capture on a negative edge. This will require it to switch between the asynchronous edge detectors and require a

two cycle synchronization time. Interrupt is cleared when higher byte of capture register is read in this mode.

Figure 21-6. Input Capture Pulse Width Measurement

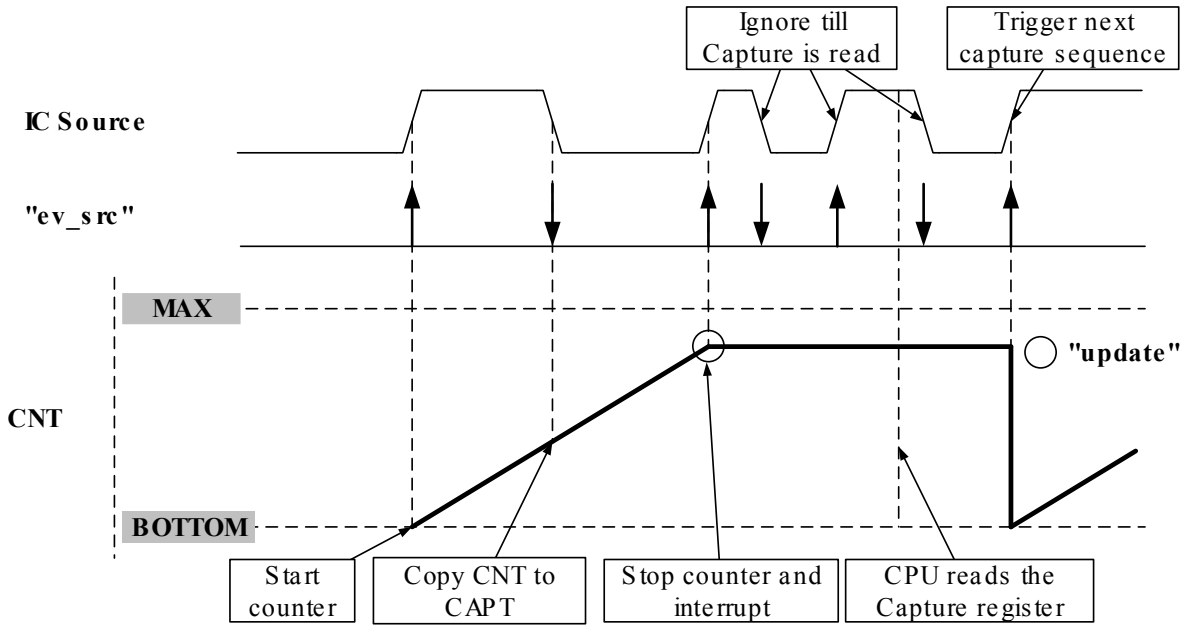


Input Capture Frequency and Pulse Width Measurement

The input capture frequency and pulse width measurement will start the counter on the positive edge of a capture, do a capture on the negative edge and stop the counter on the next positive edge.

As can be seen in the figure, on the second positive edge of event, counter freezes. At any stage in this mode, reading capture will clear the flag. If it is during the time when counter is counting, flag is not set but this will cause no problem. In the freeze stage if capture is read, flag will be cleared but count register will sustain its value. A positive edge will make it start from 0, so it is a must to read the counter register first and then capture register, and an event should not occur.

Figure 21-7. Input Capture Frequency and Pulse Width Measurement

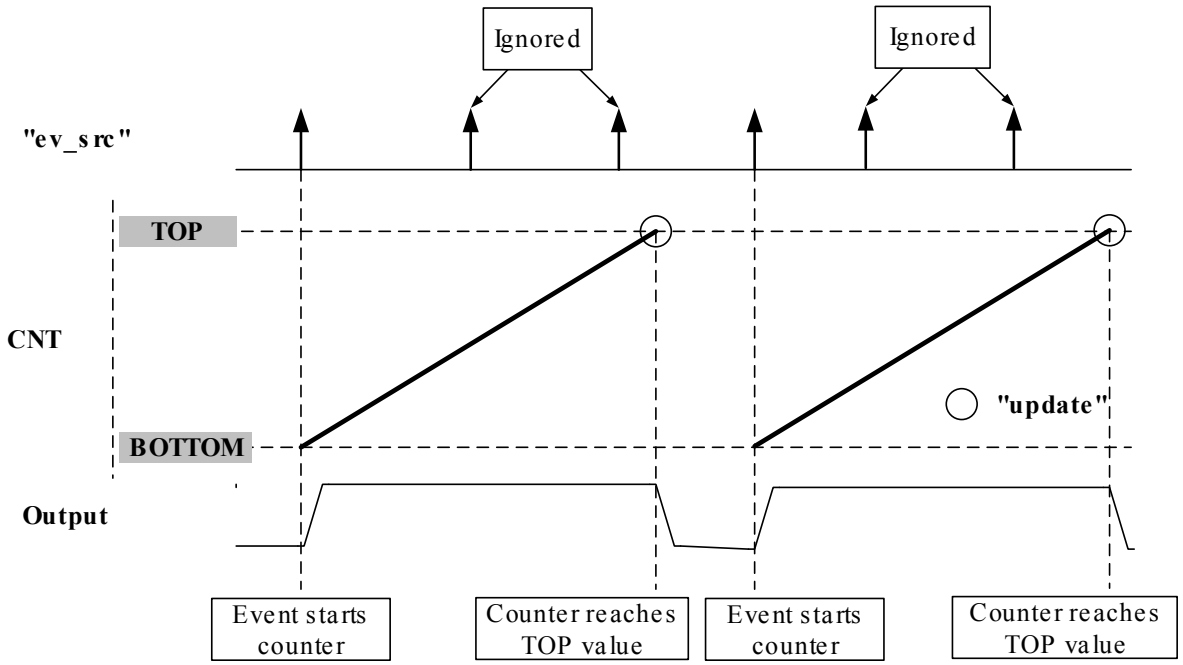


Single-Shot Mode

This mode can be used to generate a constant length pulse from the time an event occurs. When the counter is not counting the output pin is set to 0. When the event trigger is received the output will go high and the counter will start from zero and count up to the CCMP register value. The RUN bit in the Status register can be read to see if the counter is counting or not. When the counter register reaches the CCMP register value, counter will stop and the output pin will go low for at least one pre-scaller cycle. If a new event arrives during this time, that event will be ignored. The following figure shows an example waveform. There is a two clock cycle delay from when the event is received until the output is set high. If an immediate action is wanted the ASYNC bit in TCB_CTRLB need to be set. When EDGE bit of EVCTRL register is set, any edge will can trigger the start of counter otherwise only positive edge will trigger the start.

After the module is enabled, counter starts with or without any event which causes the output to go high. Writing TOP in the counter register will avoid this to happen. Similar behavior is seen if EDGE bit of TCB_EVCTRL register is set while the module is enabled, and the work around is the same. It is not recommended to change configuration while the module is enable.

Figure 21-8. Single-Shot Mode



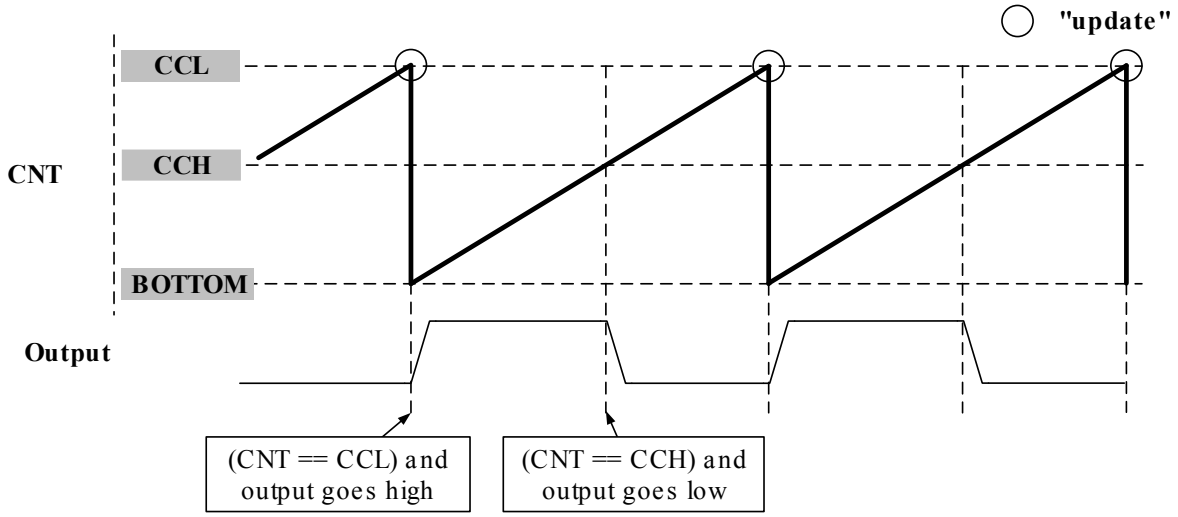
8-bit PWM Mode

This timer can also be set in an 8-bit PWM mode where each of the register pairs in the 16-bit compare/capture register (TCB_CCH and TCB_CCL) are used as individual compare registers. The counter will continuously count from zero to CCL and the output will be set at BOTTOM and cleared when the counter reaches CCH. When module is enabled and in PWM mode, changing the value of compare/capture register will change the output but the transition is not guaranteed to be successful. It is recommended to disable the module, write compare/capture register to {CCH, CCL}, then write 0x0000 to count register and then enable the module. CCH is the number of cycles for which output will stay high, CCL+1 is the period of the output pulse.

Output of the module for different value of capture register are explained below.

- CCL = 0 ⇒ Output = 0
- CCL = 0xFF
  - CCH = 0 ⇒ Output = 0
  - 0 < CCH ≤ 0xFF ⇒ Output = 1 for CCH cycles, low for the rest of the period
- For 0 < CCL < 0xFF
  - CCH = 0 ⇒ Output = 0
  - If 0 < CCH ≤ CCL ⇒ Output = 1 for CCH cycles, low for the rest
  - CCH = CCL + 1 ⇒ Output = 1

Figure 21-9. 8-bit PWM Mode



21.6.2.4. Synchronous Output

TCB_CTRLB.CCINIT, TCB_CTRLB.CCEN and TCB_CTRLB.CNTMODE are handling the behavior of the synchronous output signal which is shown in the table below.

Table 21-3. Synchronous Output

CCEN	MODE	Output sync
0	Any mode	0
1	Single shot mode	Output high when counter starts and output low when counter stops
1	8-bit PWM mode	PWM mode output
1	Modes except single shot and PWM	PININIT

21.6.2.5. Asynchronous Output

TCB_CTRLB.CCIT, TCB_CTRLB.CCEN, TCB_CTRLB.ASYNC and TCB_CTRLB.CNTMODE are handling the behavior of the asynchronous output signal which is show in the table below.

Table 21-4. Asynchronous Output

CCEN	ASYNC	MODE	Output async
0	0	Any mode	0
1	0	Single-shot mode	Output high when counter starts and output low when counter stops
1	1	Single-shot mode	Output high when event arrives and output low when counter stops

CCEN	ASYNC	MODE	Output async
1	1	8-bit PWM mode	PWM mode output
1	1	Modes except single-shot and PWM	PININIT

Changing modes while the peripheral is enabled can produce unpredictable output and it is not recommended. There is a possibility that the interrupt flag is set during the configuration of the timer and it is recommended to clear it after configuring the peripheral.

21.6.2.6. Noise Canceller

The noise canceller improves noise immunity by using a simple digital filtering scheme. The noise canceller input is monitored over four samples, and all four must be equal to change the output that is used by the edge detector.

When enabled, the noise canceller introduces an additional four system clock cycles of delay from a change applied to the input to the update of the input compare register. The noise canceller uses the system clock and is therefore not affected by the prescaler.

21.6.3. Additional Features

21.6.3.1. Synchronized with TCA

The TCB can also be configured to use the clock of the Timer/Counter type A (TCA). This is configured by writing 0x2 to the Clock Select bit field in the Control A register (TCB_CTRLA.CLKSEL).

The TCB will use the prescaled clock signal used by TCA for counting, as configured by TCA_CTRLA.CLKSEL.

When the Synchronize Update bit in the Control A register (TCB_CTRLA.SYNCUPD) is written to '1', the TCB counter will restart when the TCA counter restarts.

21.6.4. Events

TCB can generate an single strobe output event when TCB interrupt condition occurs, and this Event is sent to the event system.

There is one input Event line to this peripheral which is used when the Capture Event Input Enable bit in the Event Control register (TCB_EVCTRL.CAPTEI) is written to '1', and different actions are taken depending on which mode the peripheral is configured. The actions are explained in the description of the Event Edge bit field (TCB_EVCTRL.EDGE).

Related Links

EVCTRL on page 243
EVSYS - Event System on page 109

21.6.5. Interrupts

Table 21-5. Available Interrupt Vectors and Sources

Offset	Name	Vector Description	Conditions
0x00	CAPT	TCB interrupt	Depending on operating mode. See description of TCB_INTFLAG.CAPT

When an interrupt condition occurs, the corresponding Interrupt Flag is set in the Interrupt Flags register of the peripheral (*peripheral*_INTFLAGS).

An interrupt source is enabled or disabled by writing to the corresponding bit in the peripheral's Interrupt Control register (*peripheral*_INTCTRL).

An interrupt request is generated when the corresponding interrupt source is enabled and the Interrupt Flag is set. The interrupt request remains active until the Interrupt Flag is cleared. See the peripheral's INTFLAGS register for details on how to clear Interrupt Flags.

Note: Interrupt requests are only generated when Interrupts are enabled globally.

Related Links

CPUINT - CPU Interrupt Controller on page 97

INTFLAGS on page 246

21.6.6. Sleep Mode Operation

The TCB will halt operation in Power Down sleep mode and Standby sleep mode. However, if the Run in Standby bit in the Control A register (TCB_CTRLA.RUNSTDBY) is written to '1', the TCB will continue operation in Standby sleep mode.

21.6.7. Synchronization

Not applicable.

21.6.8. Configuration Change Protection

Not applicable.

21.7. Register Summary

Offset	Name	Bit Pos.								
0x00	CTRLA	7:0		RUNSTDBY		SYNCUPD		CLKSEL[1:0]		ENABLE
0x01	CTRLB	7:0		ASYNC	CCINIT	CCEN		CNTMODE[2:0]		
0x02 ... 0x03	Reserved									
0x04	EVCTRL	7:0		FILTER		EDGE				CAPTEI
0x05	INTCTRL	7:0								CAPT
0x06	INTFLAGS	7:0								CAPT
0x07	STATUS	7:0								RUN
0x08	DBGCTRL	7:0								DBGRUN
0x09	TEMP	7:0	TEMP[7:0]							
0x0A	CNT	7:0	CNT[7:0]							
0x0B		15:8	CNT[15:8]							
0x0C	CCMP	7:0	CCMP[7:0]							
0x0D		15:8	CCMP[15:8]							

21.8. Register Description

21.8.1. Control A

Name: CTRLA
Offset: 0x00
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
		RUNSTDBY		SYNCUPD		CLKSEL[1:0]		ENABLE
Access		R/W		R/W		R/W	R/W	R/W
Reset		0		0		0	0	0

Bit 6 – RUNSTDBY: Run Standby
Writing a '1' to this bit will enable the peripheral to run in Standby sleep mode.

Bit 4 – SYNCUPD: Synchronize Update
When this bit is written to '1', at restart of the TCB, the counter will restart, too.

Bits 2:1 – CLKSEL[1:0]: Clock Select
Writing these bits selects the clock source for this peripheral.

Value	Description
0x0	CLK_PER
0x1	CLK_PER / 2
0x2	Use clock from TCA
0x3	Reserved

Bit 0 – ENABLE: Enable
Writing this bit to '1' enables the Timer/Counter type B peripheral.

21.8.2. Control B

Name: CTRLB
Offset: 0x01
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
		ASYNC	CCINIT	CCEN		CNTMODE[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		0	0	0		0	0	0

Bit 6 – ASYNC: Asynchronous Enable
Writing this bit to '1' will change the behavior of the asynchronous output in single-shot mode.

Value	Description
0	The output will go HIGH at when the counter actually starts
1	The output will go HIGH at when an Event arrives

Bit 5 – CCINIT: Compare Pin Initial Value
This bit is used to set the initial state of the pin when a pin output is used.

Value	Description
0	Initial pin state is LOW
1	Initial pin state is HIGH

Bit 4 – CCEN: Compare/Capture Output Enable

Value	Description
0	Compare/Capture Output is zero
1	Compare/Capture Output has a valid value

Bits 2:0 – CNTMODE[2:0]: Timer Mode
Writing these bits selects the timer mode.

Value	Description
0x0	Periodic interrupt mode
0x1	Periodic timeouts mode
0x2	Input capture event mode
0x3	Input capture frequency measurement mode
0x4	Input capture pulse-width measurement mode
0x5	Input capture frequency and pulse-width measurement mode
0x6	Single shot mode
0x7	8-bit PWM mode

21.8.3. Event Control

Name: EVCTRL
Offset: 0x04
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
		FILTER		EDGE				CAPTEI
Access		R/W		R/W				R/W
Reset		0		0				0

Bit 6 – FILTER: Input Capture Noise Cancellation Filter
Writing this bit to '1' enables the input capture noise cancellation unit.

Bit 4 – EDGE: Event Edge
This bit is used to select the Event edge. The effect of this bit is dependent on the selected Count Mode (CNTMODE) in TCB.CTRLB.

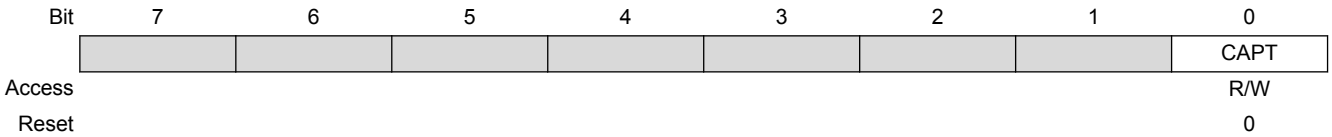
Count Mode	EDGE	Positive Edge	Negative Edge
Periodic interrupt mode	0	NA	NA
	1	NA	NA
Periodic timeout mode	0	Start counter	Stop counter
	1	Stop counter	Start counter
Input capture on event	0	Capture = count	NA
	1	NA	Capture = count
Input capture frequency	0	Capture = count, initialize, interrupt	NA
	1	NA	Capture = count, initialize, interrupt
Input capture frequency	0	Initialize	Capture = count, interrupt
	1	Capture = count, interrupt	Initialize
Input capture frequency and pulse	0	On 1st Positive: initialize On following Negative: capture 2nd Positive: stop, interrupt	
	1	On 1st Negative: initialize On following Positive: capture 2nd Negative: stop, interrupt	
Single shot mode	0	Start counter	NA
	1	Start counter	Start counter

Count Mode	EDGE	Positive Edge	Negative Edge
8-bit PWM mode	0	NA	NA
	1	NA	NA

Bit 0 – CAPTEI: Capture Event Input Enable
Writing this bit to '1' enables the event interrupt.

21.8.4. Interrupt Control

Name: INTCTRL
Offset: 0x05
Reset: 0x00
Property: -



Bit 0 – CAPT: Capture Interrupt Enable
Writing this bit to '1' enables the Capture interrupt.

21.8.5. Interrupt Flags

Name: INTFLAGS
Offset: 0x06
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
								CAPT
Access								R/W
Reset								0

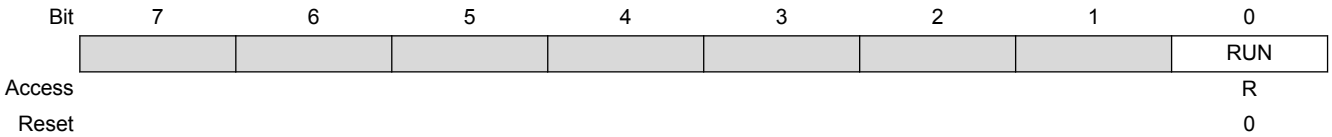
Bit 0 – CAPT: Interrupt Flag
This bit is set when an interrupt occurs. The interrupt conditions are dependent on the Counter mode (CNTMODE) in TCB.CTRLB.

This bit is cleared by writing a '1' to it, when the interrupt is executed, or when the Capture register is read in capture mode.

Counter Mode	Interrupt Set Condition
Periodic interrupt mode	Set when the counter reaches TOP
Periodic timeout mode	Set when the counter reaches TOP
Input capture on event	Set when an event occurs and the capture register is loaded, Flag clears when capture is read
Input capture frequency measurement	Set on edge when the capture register is loaded and count initialized, Flag clears when capture is read
Input capture pulse width measurement	Set on a edge when the capture register is loaded, previous edge initialized the count, Flag clears when capture is read
Input capture frequency and pulse width measurement	Set on second (positive or negative) edge when the counter is stopped, Flag clears when capture is read
8-bit PWM mode	Set when the counter reaches CCH
Single shot mode	Set when counter reaches TOP

21.8.6. Status

Name: STATUS
Offset: 0x07
Reset: 0x00
Property: -

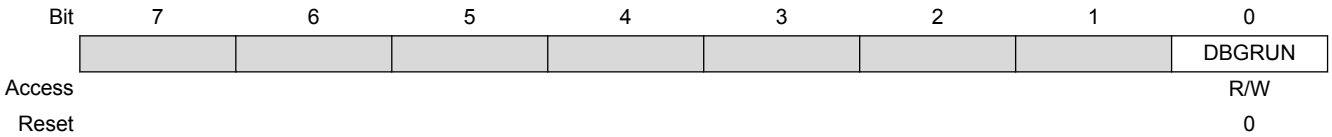


Bit 0 – RUN: Run
When the counter is running, this bit is set to '1'. When the counter is stopped, this bit is cleared '0'.

The bit it is read only bit and cannot be set by UPDI.

21.8.7. Debug Control

Name: DBGCTRL
Offset: 0x08
Reset: 0x00
Property: -



Bit 0 – DBGRUN: Debug Run
Writing the bit to '0' causes the peripheral to stop and ignore Events in Debug mode.

Writing this bit to '1' enables the peripheral to continue running in Debug mode, and may generate interrupts that could interrupt the program flow when single stepping.

21.8.8. Temporary Value

The Temporary register is used by the CPU for single-cycle, 16-bit access to the 16-bit registers of this peripheral. It can also be read and written by software. See also Accessing 16-bit Registers. There is one common Temporary register for all the 16-bit registers of this peripheral.

Name: TEMP
Offset: 0x09
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	TEMP[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – TEMP[7:0]: Temporary Value

21.8.9. Count

The TCB.CNTL and TCB.CNTH register pair represents the 16-bit value TCB.CNT. The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + 0x01. For more details on reading and writing 16-bit registers, refer to Accessing 16-bit Registers.

CPU and PDI write access has priority over internal updates of the register.

Name: CNT
Offset: 0x0A
Reset: 0x00
Property: -

Bit	15	14	13	12	11	10	9	8
	CNT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:8 – CNT[15:8]: Count Value high
These bits hold the MSB of the 16-bit counter register.

Bits 7:0 – CNT[7:0]: Count Value low
These bits hold the LSB of the 16-bit counter register.

21.8.10. Capture

The TCB.CCMPL and TCB.CCMPH register pair represents the 16-bit value TCB.CCMP. The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + 0x01. For more details on reading and writing 16-bit registers, refer to Accessing 16-bit Registers.

This register has different functions depending on the mode of operation:

- For capture operation, these registers contains the captured value of the counter at the time the capture occurs
- In periodic interrupt/timeout and single shot mode this register acts as the TOP value.
- In 8-bit PWM mode, TCB.CCMPL and TCB.CCMPH act as two independent compare registers.

Name: CCMP
Offset: 0x0C
Reset: 0x00
Property: -

Bit	15	14	13	12	11	10	9	8
	CCMP[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CCMP[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:8 – CCMP[15:8]: Compare/Capture Value high byte
These bits hold the MSB of the 16-bit compare, capture and top value

Bits 7:0 – CCMP[7:0]: Compare/Capture Value low byte
These bits hold the LSB of the 16-bit compare, capture and top value

22. TCD - 12-bit Timer/Counter Type D

22.1. Overview

The Timer/Counter type D (TCD) is a high performance waveform controller that consists of an asynchronous counter, a prescaler, compare logic, capture logic, and control logic. The purpose of the TCD is to control power application like LED, motor control, H-bridge and power converters.

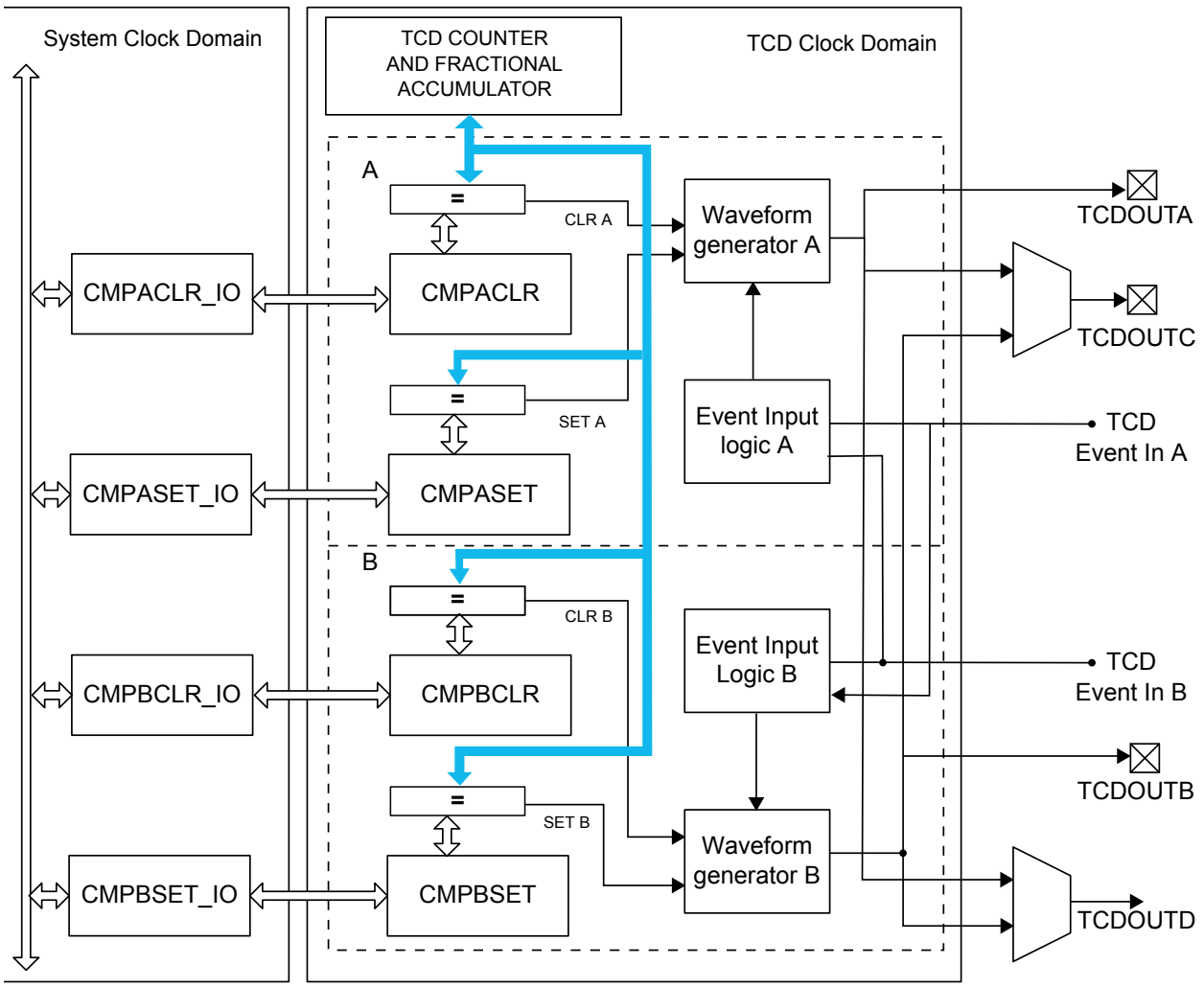
The TCD counter can select what clock source it should run on. The counter and the compare logic is used for waveform generation on two separate outputs and can generate outputs such as pulse-width modulation. Each output has a corresponding input event that can be used to capture counter values or trigger fault exceptions. The different input fault options will enable fault protection for safe and deterministic handling, disabling and/or shut down of external drivers.

22.2. Features

- 12-bit timer/counter
- Frequency up to 32MHz
  - Programmable prescaler
- Double buffered compare registers
- Waveform generation
  - Frequency generation
  - Single slope pulse width modulation
  - Dual slope pulse width modulation
  - Dead time control
  - 4 bit PWM dithering for enhanced waveform resolution
- Two separate input capture, double buffered
- Connection to event system
  - Programmable filter
- Conditional waveform on external events
  - Fault handling
  - Input blanking
  - Overload protection function
  - Fast emergency stop by hardware
- Supports both half bridge and full bridge output

22.3. Block Diagram

Figure 22-1. Timer/Counter Block Diagram



22.4. Signal Description

Signal	Description	Type
TCDOUTA	TCD waveform output A	Digital output
TCDOUTB	TCD waveform output B	Digital output
TCDOUTC	TCD waveform output C	Digital output
TCDOUTD	TCD waveform output D	Digital output

22.5. Product Dependencies

In order to use this peripheral, other parts of the system must be configured correctly, as described below.

Table 22-1. TCD Product Dependencies

Dependency	Applicable	Peripheral
Clocks	Yes	CLKCTRL
I/O Lines and Connections	No	-
Interrupts	Yes	CPUINT
Events	Yes	EVSYS
Debug	Yes	UPDI

Related Links

Clocks on page 254

Interrupts on page 54

Events on page 181

Debug Operation on page 254

22.5.1. Clocks

The TCD can be connected to the 20MHz oscillator directly, to an external clock, or to the System clock. This is configured by the Clock Select bit field in the Control A register (TCD_CTRLA.CLKSEL).

Related Links

CLKCTRL - Clock Controller on page 68

22.5.2. I/O Lines and Connections

Not applicable.

22.5.3. Interrupts

Using the interrupts of this peripheral requires the Interrupt Controller to be configured first.

Related Links

CPUINT - CPU Interrupt Controller on page 97

SREG on page 51

Interrupts on page 136

22.5.4. Events

The events of this peripheral are connected to the Event System.

Related Links

EVSYS - Event System on page 109

22.5.5. Debug Operation

When the CPU is halted in debug mode, this peripheral will continue normal operation. If the peripheral is configured to require periodical service by the CPU through interrupts or similar, improper operation or data loss may result during debugging. This peripheral can be forced to halt operation during debugging.

When run-time debugging, this peripheral will continue normal operation. Halting the CPU in debugging mode will halt normal operation of the peripheral.

This peripheral can be forced to operate with halted CPU by writing a '1' to the Debug Run bit in the Debug Control register of the peripheral (*peripheral*_DBGCTRL.DBGRUN).

When the Fault Detection bit (TCD_DBGCTRL.FAULTDET) is written to '1' and the CPU is halted in debug mode, an event/fault is created on both input event channels. These events/faults will last as long as the break, and can serve as safeguard in Debug mode, e.g. by forcing external components off.

If the peripheral is configured to require periodical service by the CPU through interrupts or similar, improper operation or data loss may result during halted debugging.

Related Links

UPDI - Unified Program Debug Interface on page 490

22.6. Functional Description

22.6.1. Principle of Operation

The TCD contains an asynchronous counter with compare logic that can generate two independent outputs called TCDOUTA and TCDOUTB. The outputs are generated based on what wave generation mode the TCD is using. The compare values are defined in the four compare registers.

In addition to the counter and compare registers, input Events can be used as input to the TCD. The input Events can be used to capture the value of the asynchronous counter, or the inputs can be used as fault trigger that alter the outputs and/or counter state.

A TCD cycle is defined as the time it takes to go through the different output states for the two outputs TCDOUTA and TCDOUTB. The TCD cycle is split into these states:

- Dead time TCDOUTA (DTA)
- On time TCDOUTA (OTA)
- Dead time TCDOUTB (DTB)
- On time TCDOUTB (OTA)

In a standard setup all states are present in the order above. They are non-overlapping. It is possible to set up the TCD to not go through all output states, or to have overlapping states, see sections on wave generation. The compare values Compare A Set (CMPASET), Compare A Clear (CMPACLR), Compare B Set (CMPBSET) and Compare B Clear (CMPBCLR) are used to switch between the different output states.

Note: The parameters CMPxSET, CMPxCLR are stored in the respective registers (TCD_CMPxSET, TCD_CMPxCLR), which consist of both a low and a high byte.

The TCD counter has four different ways to go through a TCD cycle. The different ways are called Wave Generation Modes. They are controlled by the Wave Generation Mode bits in the Control A register (TCD_CTRLA.WGMODE). The wave generation modes are:

- One Ramp mode
- Two Ramp mode
- Four Ramp mode
- Dual Slope mode

The names tell how the TCD counter is counting during one TCD cycle. Depending on the wave generation mode, different compare values are used in different orders.

The TCD core is asynchronous to the system clock. It is possible to select between 3 different clock sources that can be prescaled for the TCD core. The TCD core clock is used to clock the TCD counter, the outputs (TCDOUTx), and Events generated by the TCD.

The TCD input Events must be set up in the Event System. The two inputs are called Input Event A and Input Event B, and functionality is connected to the two outputs. The input Events can be configured as follows:

Blanking on Input Event A	The input blanking functionality is removing the input Events for a programmable time in a selectable part of the TCD cycle.
Digital Filtering on Events	The digital filter that removes short Events (i.e., Events lasting less than 4 TCD clock cycles).
Selecting Event Level/Edge	Either falling edge/low level or rising edge/high level of the Event input are accepted.
Asynchronous Output Control	This configuration allows the Event to override the output instantly when the Event occurs used for non-recoverable faults.

The TCD can use the input Events in ten different input modes, selected separately for the two input Events. The input mode defines how the input Event will affect the outputs, and where in the TCD cycle the counter should go when an Event occurs.

When the TCD is running, it generates output Events that can be used by the Event System. The output Events are generated based on when in the TCD cycle the different compare values match. The output Events can also be delayed by a programmable time.

The TCD can also generate Interrupts, based on where in the TCD cycle it is.

Related Links

EVSYS - Event System on page 109
Wave Generation Modes on page 258

22.6.2. Basic Operation

22.6.2.1. Initialization

To initialize the TCD:

1. Configure the static registers to the desired functionality.
2. Write desired initial values to the double-buffered registers.
3. Ensure that the Enable Ready bit in the Status register (TCD_STATUS.ENRDY) is set to '1'.
4. Enable the TCD by writing a '1' to the Enable bit in the Control A register (TCD_CTRLA.ENABLE).

It is possible to disable the TCD in two different ways:

1. By writing a '0' to TCD_CTRLA.ENABLE. This disables the TCD instantly when synchronized to the TCD core domain.
2. By writing a '1' to the Disable at End of Cycle Strobe bit in the Control E register (TCD_CTRLE.DISEOC). This disables the TCD at the end of the TCD cycle.

Related Links

Register Synchronization Categories on page 256

22.6.2.2. Register Synchronization Categories

Most of the IO registers needs to be synchronized to the asynchronous TCD core clock domain. This is done in different ways for different register categories:

- Command and Enable Control registers
- Doubled-buffered registers
- Static registers

- Normal IO and STATUS registers

See the Table below for categorized registers.

Command and Enable Registers
Because of synchronization between the clock domains it is only possible to change the Enable bits while the Enable Ready bit in the Status register (TCD_STATUS.ENRDY) is '1'.

The Control E register commands (TCD_CTRLE) are automatically synchronized to the TCD core domain when the TCD is enabled and as long as there not a synchronization ongoing already. Check in the Status register if the Command Ready bit is '1' (TCD_STATUS.CCMDRDY) to ensure that it is possible to write a new command. TCD_CTRLE is a strobe register that will clear itself when the command is done.

The Control E register commands are:

- Synchronize at end of TCD cycle: synchronizes all doubled buffered registers to TCD clock domain at the end of the TCD cycle
- Synchronize: synchronized all doubled buffered registers to TCD clock domain when the command is synchronized to TCD clock domain
- Restart: Restarts the TCD counter
- Software Capture A: Capture TCD counter value to TCD_CAPTUREA
- Software Capture B: Capture TCD counter value to TCD_CAPTUREB

Double-Buffered Registers
The doubled-buffered registers can be updated in normal IO writes while the TCD is enabled and no synchronization between the two clock domains is ongoing. Check that the TCD_STATUS.CMDRDY bit is '1' to ensure that it is possible to update the doubled buffered IO registers. The values will be synchronized to the TCD core domain when a synchronization command is sent or when the TCD is enabled.

Static Registers
The static registers are kept static whenever the TCD is enabled. That means that these registers must be configured before enabling the TCD. It is not possible to write to these registers as long as the TCD is enabled. To see if the TCD is enabled, check if TCDCTRLA.ENABLE is reading '1'.

Normal IO and Status Registers
The read-only registers inform about synchronization status and values synchronized from the core domain. The reset of these registers and normal IO registers are not constrained by any synchronization between the domains.

Table 22-2. Categorization of Registers

Enable and Command registers	Doubled-buffered registers	Static registers	Read-only registers	Normal IO registers
TCD_CTRLA (ENABLE bit)	TCD_DLYCTRL	TCD_CTRLA (All bits Except ENABLE bit)	TCDSTATUS	TCD_INTCTRL
TCD_CTRLE	TCD_DLYCNT	TCD_CTRLB	TCD_CAPTUREAL/H	TCD_INTFLAGS
	TCD_DITCTRL	TCD_CTRLC	TCD_CAPTUREBL/H	
	TCD_DITVAL	TCD_CTRLD		

Enable and Command registers	Doubled-buffered registers	Static registers	Read-only registers	Normal IO registers
	TCD_DBGCTRL	TCD_EVCTRLA		
	TCD_CMPASETL/H	TCD_EVCTRLB		
	TCD_CMPACLRL/H	TCD_INPUTA		
	TCD_CMPBSETL/H	TCD_INPUTB		
	TCD_CMPBCLRL/H	TCD_FAULTCTRL		

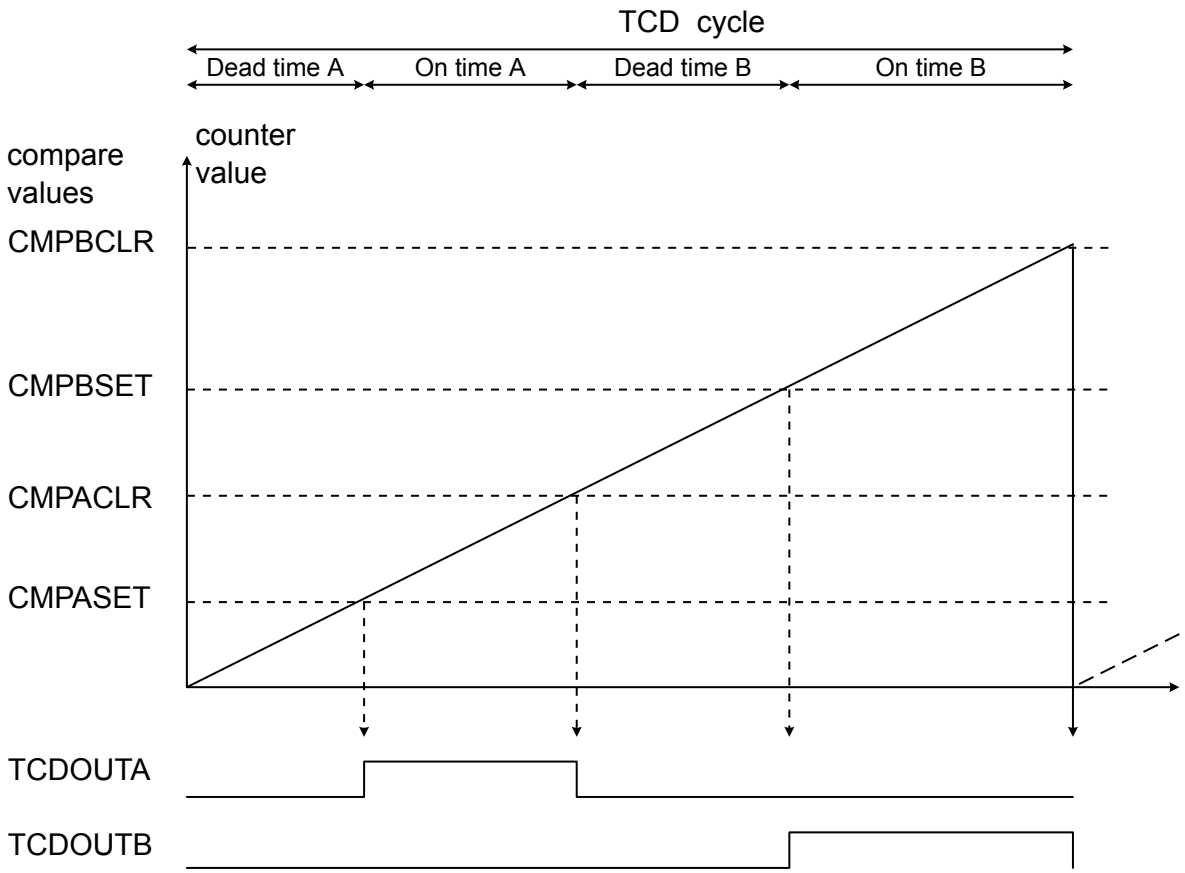
22.6.2.3. Wave Generation Modes

The TCD provides four different wave generation modes. The wave generation modes determine how the TCD counter is counting during a TCD cycle, and when the compare values are matching.

One Ramp Mode

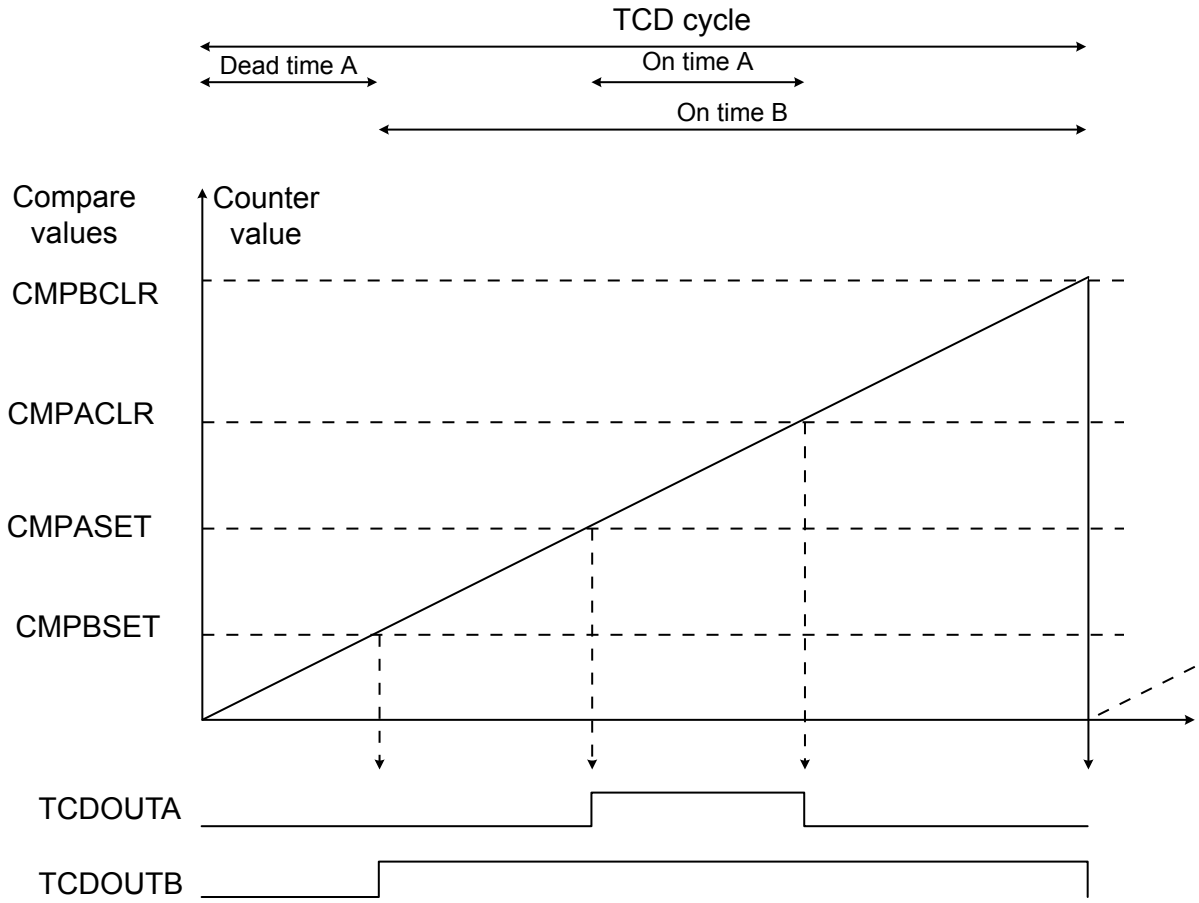
In One Ramp Mode, TCD counter counts up until it reaches the CMPBCLR value. Then the TCD cycle is done and the counter restarts from 0x000, beginning a new TCD cycle. The TCD cycle period will therefore be CMPBCLR value times TCD clock period.

Figure 22-2. One Ramp Mode



In the figure above, CMPASET < CMPACLR < CMPBSET < CMPBCLR. This is required in One Ramp Mode to avoid overlapping outputs. The figure below is an example where CMPBSET < CMPASET < CMPACLR < CMPBCLR, resulting in an overlap of the outputs.

Figure 22-3. One Ramp Mode with CMPBSET < CMPASET



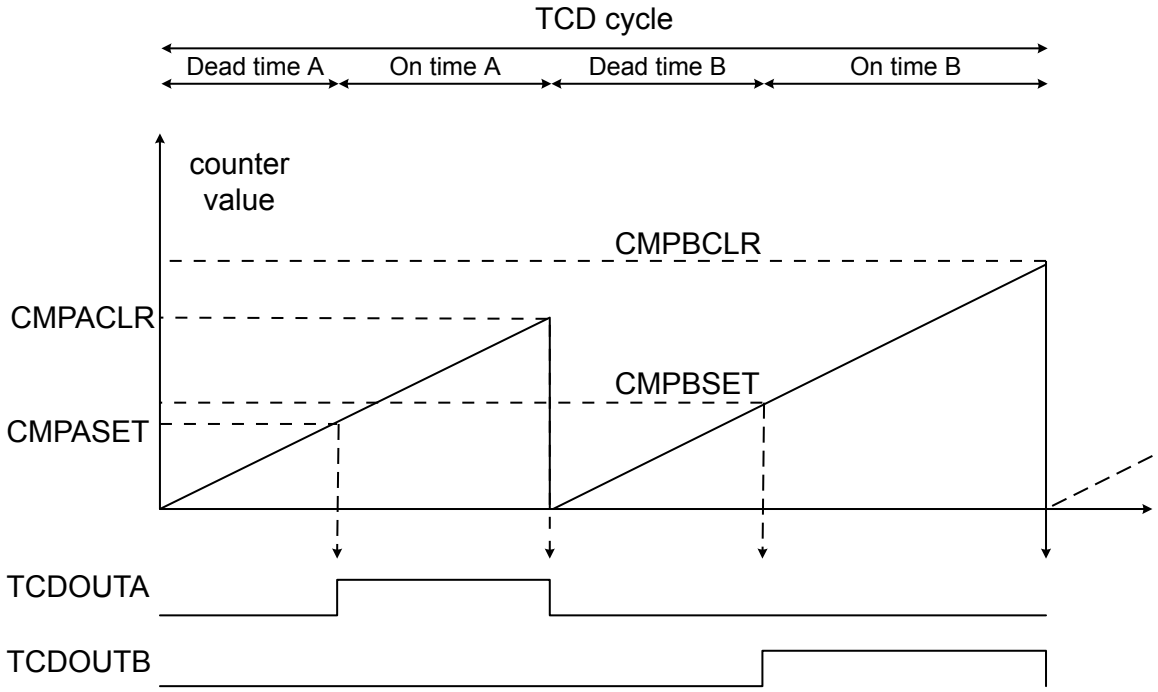
Note: If any of the other compare values are bigger than CMPBCLR it will never be triggered when running in One ramp mode. And if The CMPACLR is smaller than the CMPASET value, the clear value will not have any effect.

Two Ramp Mode

In Two Ramp Mode the TCD counter counts up until it reaches the CMPACLR value, then it resets and counts up until it reaches the CMPBCLR value. Then, the TCD cycle is done and the counter restarts from 0x000, beginning a new TCD cycle. The TCD cycle period is:

$(\mathrm{CMPACLR} + \mathrm{CMPBCLR}) \times \mathrm{TC\ clock\ period}$.

Figure 22-4. Two Ramp Mode



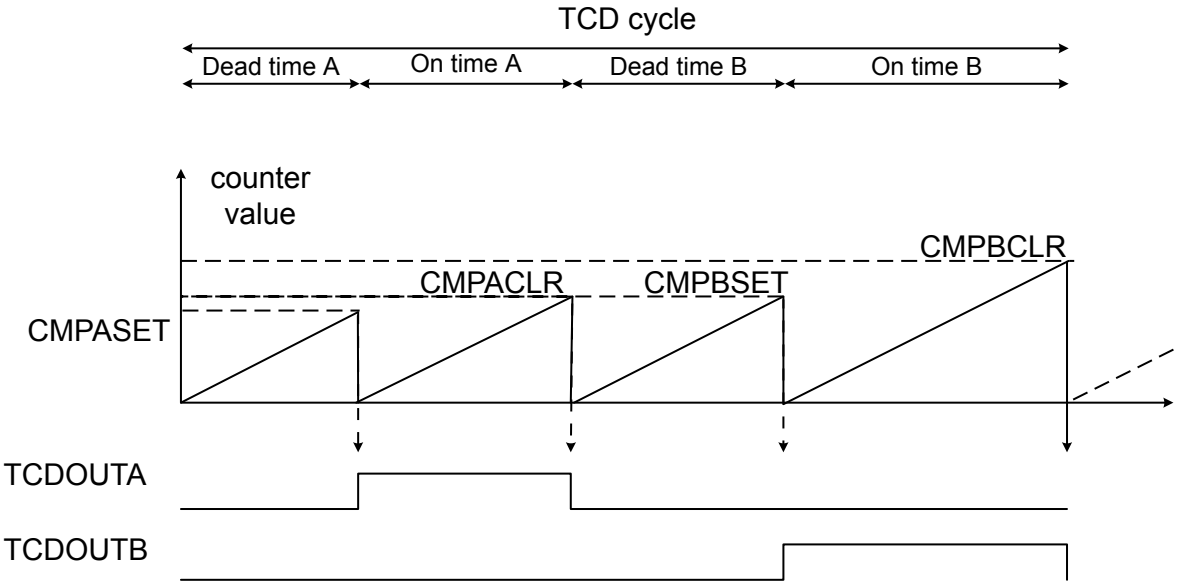
In the figure above, CMPASET < CMPACLR and CMPBSET < CMPBCLR. This causes the outputs to go high. There are no restrictions on the CMPASET/CLR compared to the CMPBSET/CLR values.
Note: In two ramp mode it is not possible to get overlapping outputs.

Four Ramp Mode
In Four Ramp Mode the TCD cycle is following this pattern:

1. A TCD cycle begins with the TCD counter counting up from zero until it reaches the CMPASET value, and resets to zero.
2. The Counter counts up from zero until it reaches the CMPACLR value, and resets to zero.
3. The Counter counts up from zero until it reaches the CMPBSET value, and resets to zero.
4. The Counter counts up from zero until it reaches the CMPBCLR value, and ends the TCD cycle by resetting to zero.

The TCD cycle period is .

Figure 22-5. Four Ramp Mode



There are no restrictions on the compare values compared to each others.

Note: In Four Ramp Mode it is not possible to get overlapping outputs.

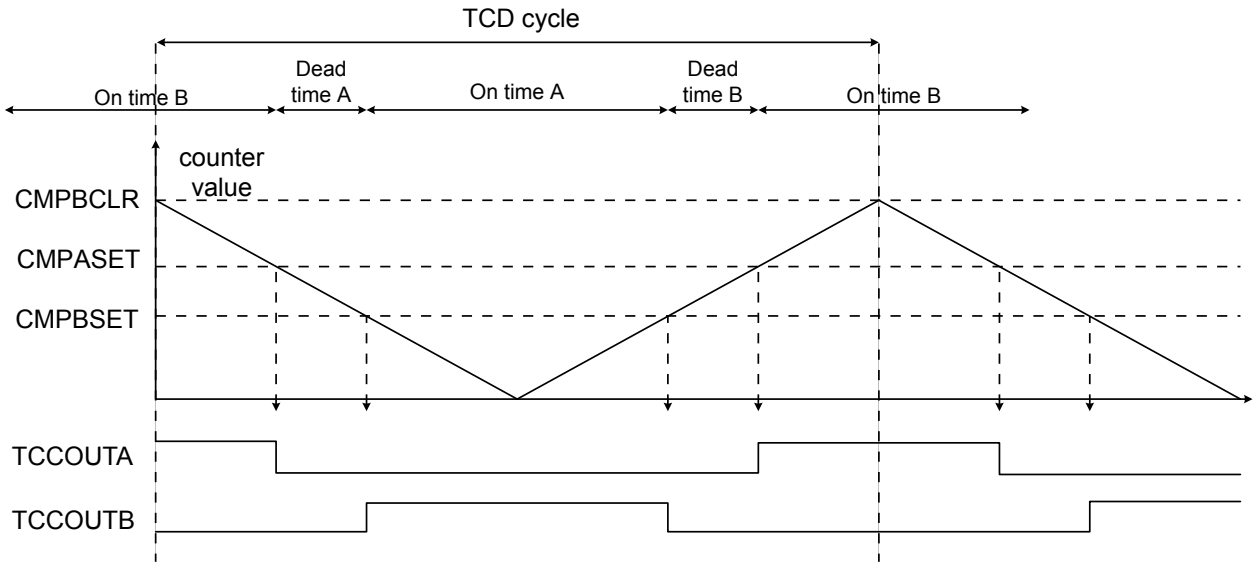
Dual Slope Mode

In Dual Slope mode, a TCD cycle consists of the TCD counter counting down from CMPBCLR value to zero, and up again to the CMPBCLR value. This gives a TCD cycle period of $2 \times \text{CMPBCLR} \times \text{TC clock period}$.

The TCDOUTA output is set when the TCD counter counts up and matches the CMPASET value. TCDOUTA is cleared when the TCD counter counts down and matches the CMPASET value.

The TCDOUTB output is set when the TCD counter counts down and matches the CMPBSET value. TCDOUTB is cleared when the TCD counter counts up and matches the CMPBSET value.

Figure 22-6. Dual Slope Mode

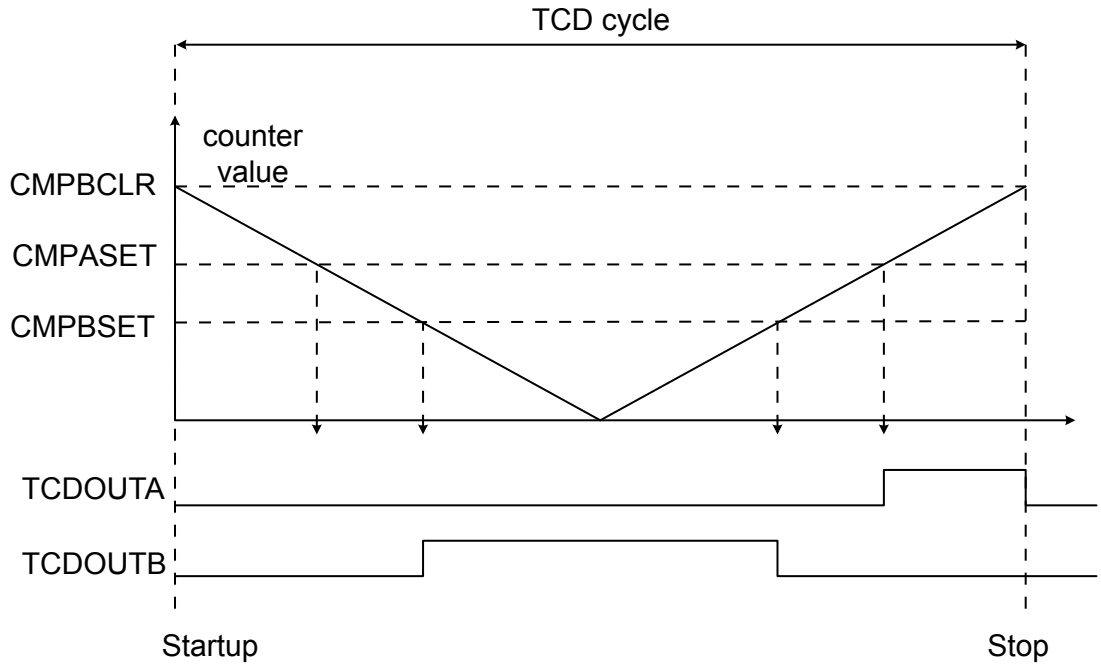


The outputs will be overlapping if CMPBSET > CMPASET.

Note:

- CMPACLR is not used in Dual Slope Mode. Writing a value to CMPACLR has no effect.
- When starting the TCD in Dual Slope Mode, the TCD counter starts at the CMPBCLR value and counts down. The TCDOUTA will not be set before the end of the first TCD cycle.

Figure 22-7. Dual Slope Mode Starting and Stopping



22.6.2.4. TCD Inputs

The TCD has two inputs that are connected to the Event System, Input A and Input B. Each input has functionality that are connected to corresponding output (TCDOUTA and TCDOUTB). That functionality is controlled by the Event Control x registers (TCD_EVCTRLA and TCD_EVCTRLB) and the Input Control x registers (TCD_INPUTACTRL and TCD_INPUTBCTRL).

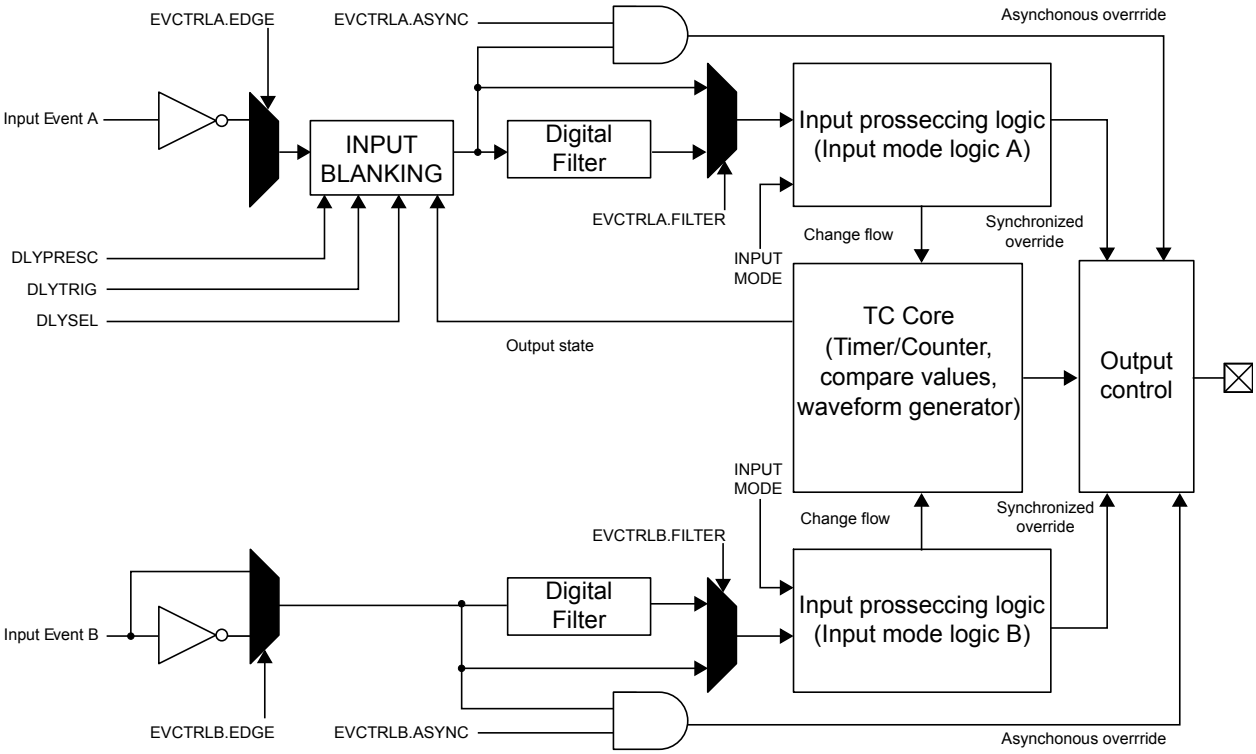
To enable the input Events, write a '1' to the Trigger Event Input Enable bit in the Event Control register (TCD_EVCTRLx.TRIGEI). The inputs will be used as a fault detect and/or capture trigger. To enable capture trigger, write a '1' to the Action bit in Event Control register (TCD_EVCTRLx.ACTION).

There are ten different input modes for the fault detection. The two inputs have the same functionality, but for input blanking, which is only supported by input A. Input blanking is configured by the Delay Control and Delay Value registers (TCD_DLYCTRL and TCD_DLYVAL).

The inputs are connected to the Event System. The connections between the Event source and the TCD input must be configured in the Event System.

An overview of the input system is shown below.

Figure 22-8. TCD Input Overview



There is a delay of 2-3 clock cycles on the TCD synchronizer clock between receiving the input Event and processing it and overriding the outputs. If using the asynchronous Event detection, the outputs will override instantly outside the input processing.

Input Blanking

Input blanking denotes removing the input Events for a programmable time in a selectable part of the TCD cycle. Input blanking can be used to mask out "false" input Events that are triggered right after changes on the outputs.

To enable input blanking, write 0x1 to the Delay Select bit field in the Delay Control register (TCD_DLYCTRL.DLYSEL). The trigger source is selected by the Delay Trigger bit field (TCD_DLYCTRL.DLYTRIG).

Input blanking uses the TCD delay clock: after a trigger, a counter is counting up until the Delay Value (TCD_DLYVAL.DLYVAL) is reached before input blanking is turned off. The TCD delay clock is a prescaled version of the TCD synchronization clock. The division factor is set by the Delay Prescaler bit field in the Delay Control register (TCD_DLYCTRL.DLYPRESC). The input blanking will last for

t_{blank} = (TCD clock period) x (DLYPRESC division factor) x DLYVAL .

Note: Input blanking is using the same logic as the programmable output Event. For this reason it is not possible to use both at the same time.

Digital Filter

The digital filter for Event input x is enabled by writing a '1' to the Filter bit in the Event Control x register (TCD_EVCTRLx.FILTER). When the digital filter is enabled, all changes to input lasting less than 4 TCD counter clock cycles will be filtered. When the filter is used, 4 TCD counter clock cycles are added to the delay between when the interrupt is received until it will affect the input processing logic.

Asynchronous Event Detection

To enable asynchronous Event detection on an input Event, write a '1' to the Asynchronous Event Control bit in the Event Control register (TCD_EVCTRLx.ASYNC).

The input Event will override the output, no matter how short the input Event is. If the input Event is very short it can be missed when synchronized to the TCD synchronization clock. The output will then only be overridden for the input Event time, and this might not be the intended functionality. It is therefore recommended that the input Event lasts for minimum 3 TCD synchronization clock cycles to ensure spike free output.

Note: It is not possible to use both asynchronous Event detection and digital filtering at the same time.

Input Modes

The user can select between 10 input modes. The selection is done by writing the Input Mode bit field in the Input x register (TCD_NPUTx.INPUTMODE).

INPUTMODE	Description
0x0	TCD Input has no action on TCD output
0x1	Stop signal, Jump to opposite dead time and wait
0x2	Stop signal, Execute opposite dead time and wait
0x3	Stop signal, Execute opposite dead while fault active
0x4	Deactivate outputs without charging time
0x5	Stop signal, and Insert dead time
0x6	Stop signal, Jump to next dead time and wait
0x7	Halt TCD and wait for software action
0x8	Edge retrigger TCD
0x9	Fixed frequency edge retrigger
0xA	Fixed frequency edge retrigger and deactivate output Reserved
other	Reserved

Not all input modes works in all wave generation modes. Below is a table that shows what wave generation modes the different input modes are valid in.

INPUTMODE	One Ramp mode	Two ramp mode	Four ramp mode	Dual slope mode
0x1	Valid	Valid	Valid	Do not use
0x2	Do not use	Valid	Valid	Do not use
0x3	Do not use	Valid	Valid	Do not use
0x4	Valid	Valid	Valid	Valid
0x5	Do not use	Valid	Valid	Do not use
0x6	Do not use	Valid	Valid	Do not use
0x7	Valid	Valid	Valid	Valid
0x8	Valid	Valid	Valid	Do not use

INPUTMODE	One Ramp mode	Two ramp mode	Four ramp mode	Dual slope mode
0x9	Valid	Valid	Valid	Do not use
0xA	Valid	Valid	Valid	Do not use

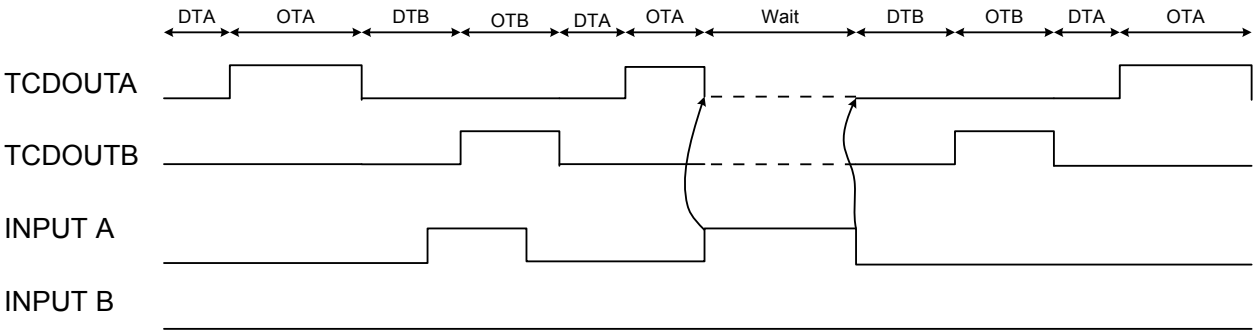
In the following sections the different input modes are presented in detail.

Input Mode 1: Stop Signal, Jump to Opposite Dead-Time and Wait

An input Event in Input mode 1 will stop the output signal, jump to the opposite dead-time, and wait until the input event goes low before the TCD counter continues.

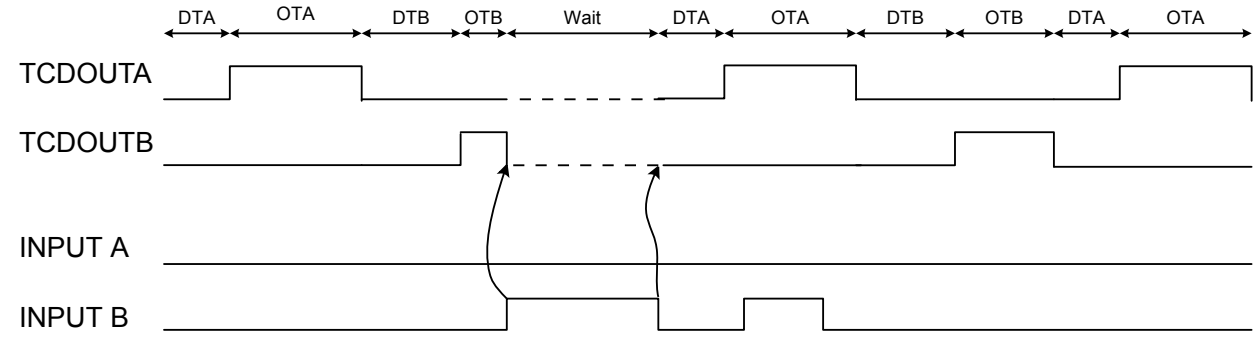
If Input mode 1 is used on input A, an Event will only have an effect if the TCD is in Dead time A or On-time A, and it will only effect the output TCDOUTA. When the Event is done, the TCD counter starts at Dead time B

Figure 22-9. Input Mode 1 on Input A



If Input mode 1 is used on input B, an Event will only have an effect if the TCD is in Dead time B or On-time B, and it will only effect the output TCDOUTB. When the event is done, the TCD counter starts at Dead time A.

Figure 22-10. Input Mode 1 on Input B

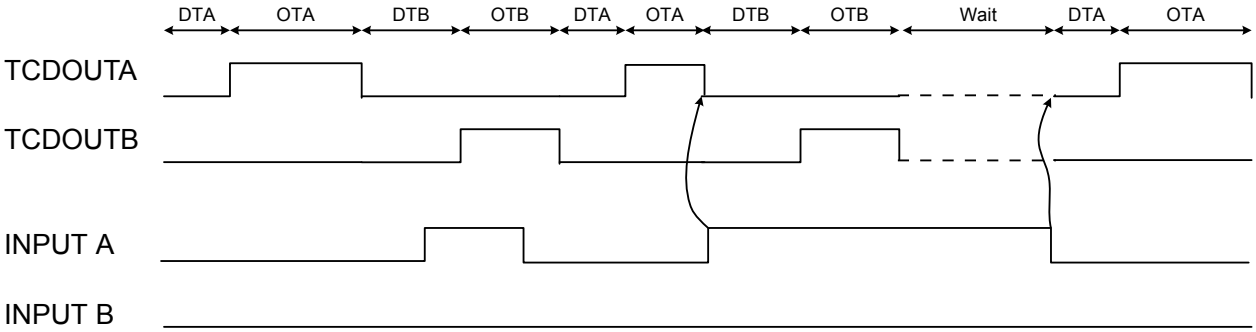


Input Mode 2: Stop Signal, Execute Opposite Dead and on Time and Wait

An input Event in Input mode 2 will stop the output signal, execute to the opposite dead-time and on-time, then wait until the input Event goes low before the TCD counter continues. If the input is done before the opposite dead-time and on-time have finished, there will be no waiting, but the opposite dead-time and on-time will continue.

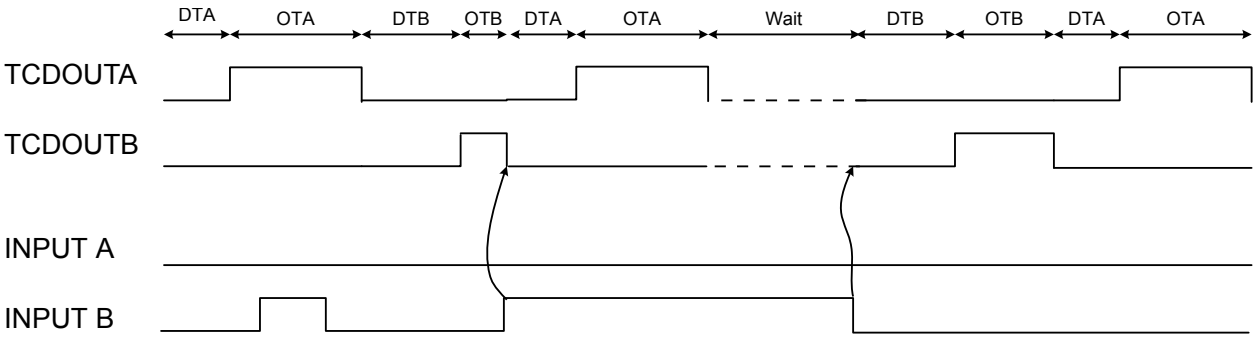
If Input mode 2 is used on input A, an Event will only have an effect if the TCD is in Dead time A or On-time A, and it will only effect the outputTCDOUTA.

Figure 22-11. Input Mode 2 on Input A



If Input mode 2 is used on input B, an Event will only have an effect if the TCD is in Dead time B or On-time B, and it will only effect the outputTCDOUTB.

Figure 22-12. Input Mode 2 on Input B

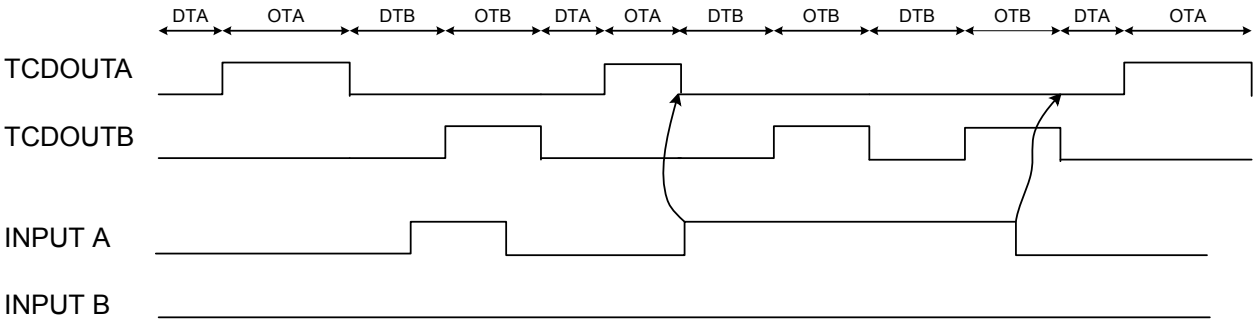


Input Mode 3: Stop Signal, Execute Opposite Dead Time and On Time while Fault is Active

An input Event in Input mode 3 will stop the output signal and start executing the opposite dead-time and on-time, as long as the fault/input is active. When the input is released, the ongoing dead time and/or on-time will finish and then, normal flow will start.

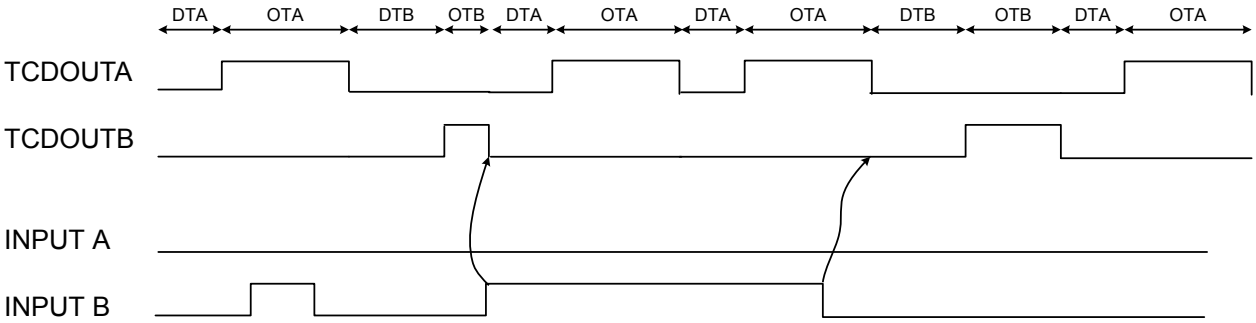
If Input mode 3 is used on input A, an Event will only have an effect if the TCD is in Dead time A or On-time A.

Figure 22-13. Input Mode 3 on Input A



If Input mode 3 is used on input B, an Event will only have an effect if the TCD is in Dead time B or On-time B.

Figure 22-14. Input Mode 3 on Input B

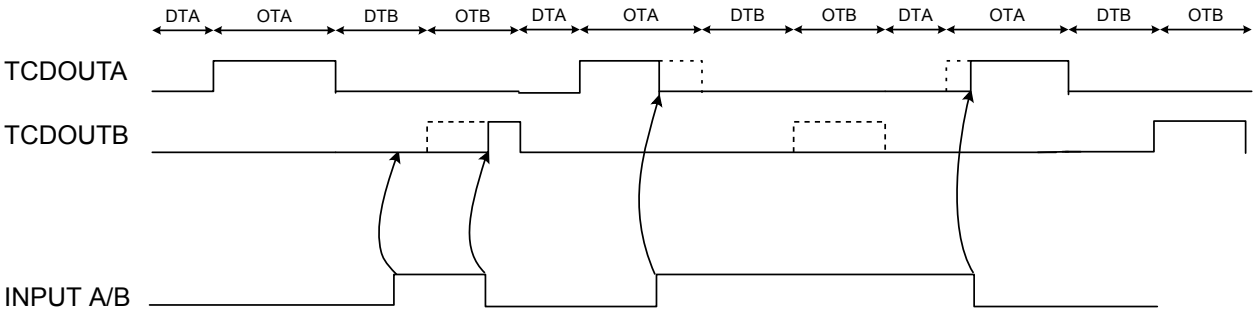


Input Mode 4: Deactivate Outputs without Changing Timing

When input mode 4 is used, both input A and input B will give the same functionality.

An input Event will deactivate the outputs as long as the Event is active. TheTCD counter will not be affected by Events in this input mode.

Figure 22-15. Input Mode 4

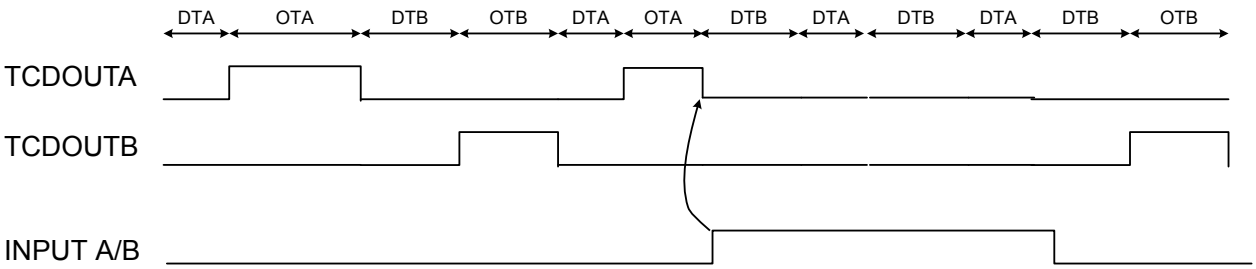


Input Mode 5: Stop Signal, Insert Dead Time

When input mode 5 is used, both input A and input B give the same functionality:

The input Event stops the outputs and starts on the opposite dead time if it occurs during an on-time. If the Event occurs during a dead time, it will continue until the next on-time should start, but instead it will jump to the next dead time. As long as the input Event is active, alternating dead times will occur. When the input Event stops, the ongoing dead time will finish and the next on-time will continue in the normal flow.

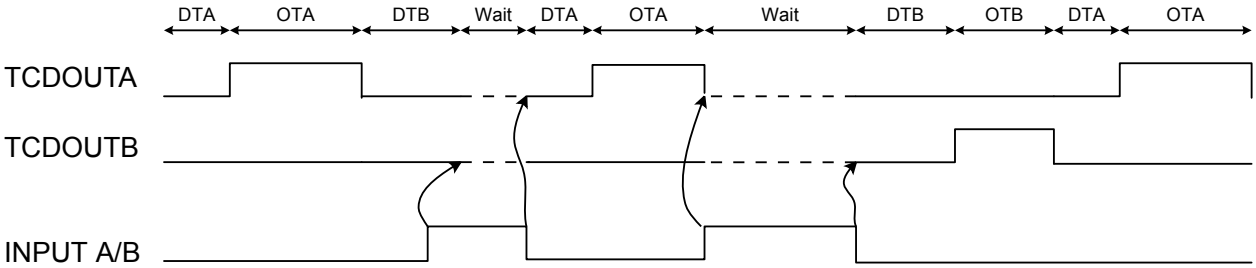
Figure 22-16. Input Mode 5



Input Mode 6: Stop Signal, Jump to Next Dead Time and Wait

When input mode 6 is used both input A and input B will give the same functionality. The input Event stops the outputs and jumps to the opposite dead time if it occurs during a on time. If the Event occurs during a dead time, it will continue until the next on-time should start, but instead, it will jump to the next dead time. As long as the input Event is active, theTCD counter will wait. When the input Events stops, the next dead time will start and normal flow will continue.

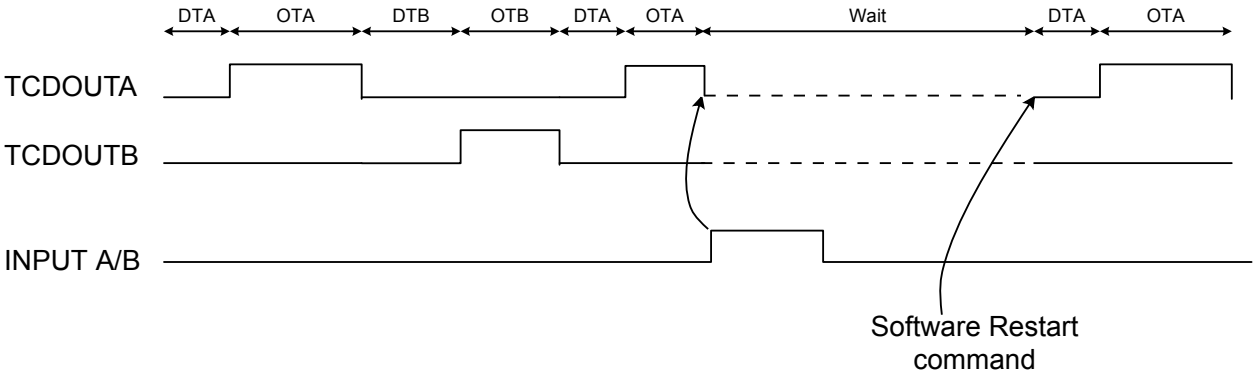
Figure 22-17. Input Mode 6



Input Mode 7: Stop Signals andTCD Counter Wait for Software Restart

When input mode 6 is used, both input A and input B will give the same functionality. The input Events stops the outputs andTCD counter. It will be stopped as until a Restart command is given. If the input Event still is high when the Restart command is given, it will just stop again. When theTCD counter restarts, it will always start on Dead time A.

Figure 22-18. Input Mode 7

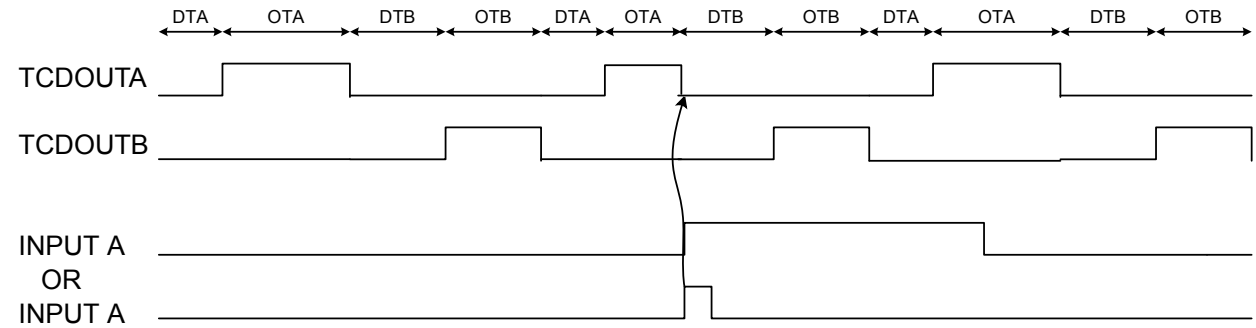


Input Mode 8: Input Edge RetriggerTCD

In Input mode 8, a positive edge on the input Event while the corresponding output is on will cause the output to stop and theTCD counter jump to the opposite Dead time.

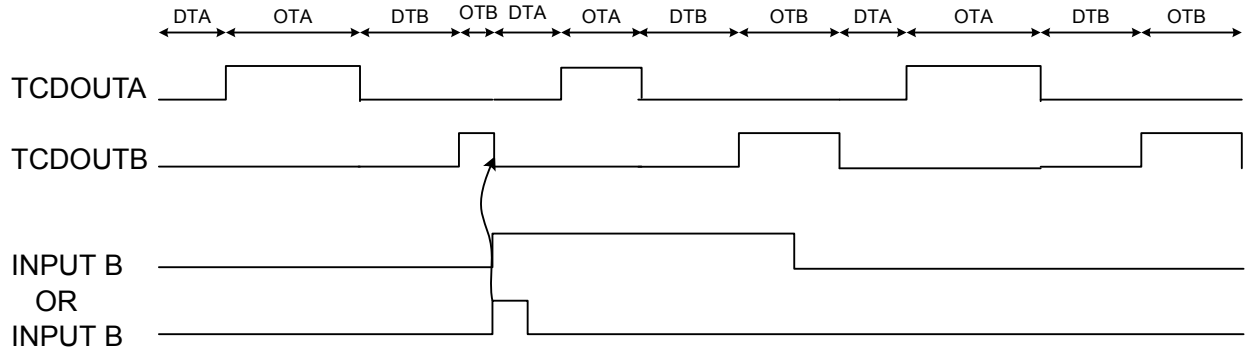
If Input mode 8 is used on input A and an positive input event occurs while in On time A, the TCD counter jumps to Dead time B.

Figure 22-19. Input Mode 8 on Input A



If Input mode 8 is used on input B and an positive input event occurs while in On time B, the TCD counter jumps to Dead time A.

Figure 22-20. Input Mode 8 on Input B

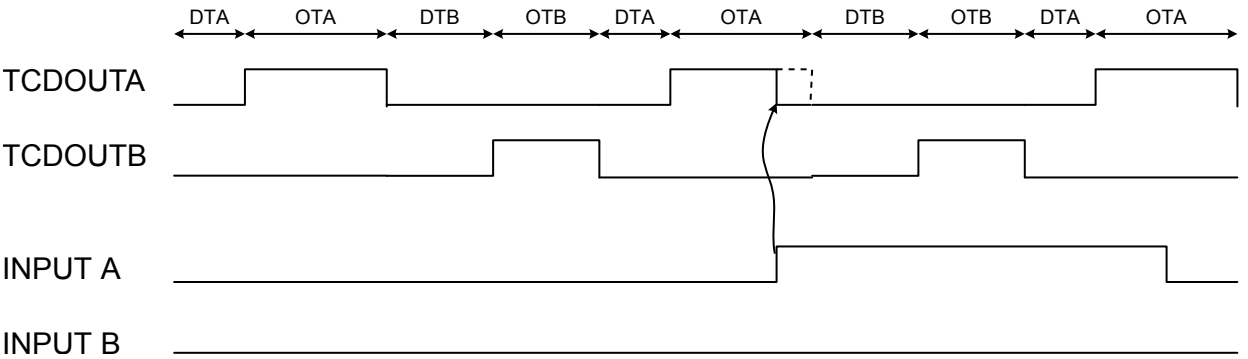


Input Mode 9: Fix Frequency Edge Retrigger

In Input mode 9 a positive edge on the input Event while the corresponding output is on will cause the output to stop during the rest of the on-time. The TCD counter will not be affected by the Event, only the output.

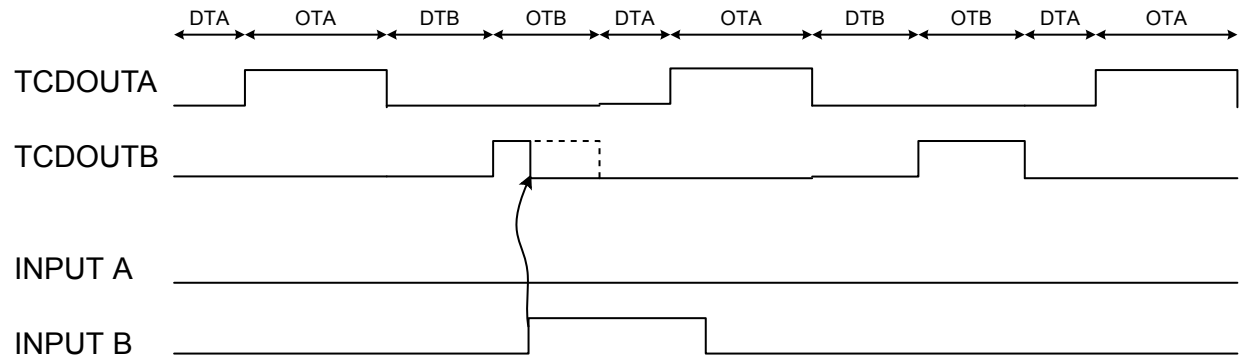
If Input mode 9 is used on input A and an positive input Event occurs while in On time A, the output will be off for the rest of the on-time.

Figure 22-21. Input Mode 9 on Input A



If Input mode 9 is used on input B and an positive input event occurs while in On time B, the output will be off for the rest of the on-time.

Figure 22-22. Input Mode 9 on Input B

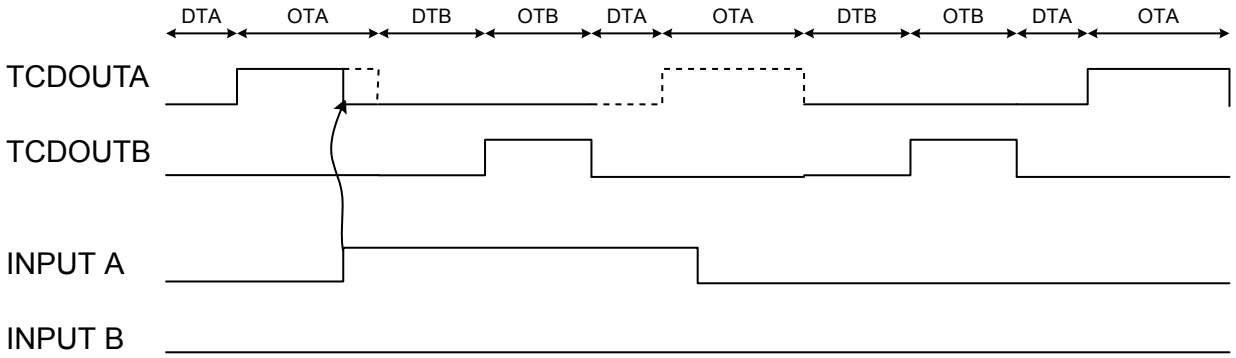


Input Mode 10: Fixed Frequency, Input Deactivate Output

In Input mode 10 the input Event will cause the corresponding output to stop as long as the input is active. If the input goes low while there should have been an on-time on the corresponding output, the output will be deactivated for the rest of the on-time, too. The TCD counter is not affected by the Event, only the output.

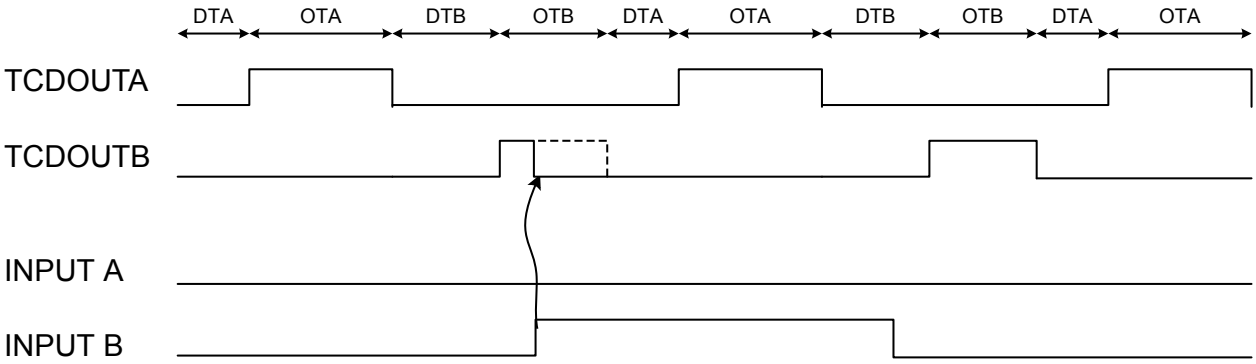
If Input mode 10 is used on input A and an input event occurs, the output A will be off as long as the event lasts. If released during an on-time, it will be off for the rest of the on-time, too.

Figure 22-23. Input Mode 10 on Input A



If Input mode 10 is used on input B and an input Event occurs, the output B will be off as long as the Event lasts. If released during an on-time, it will be off for the rest of the on-time, too.

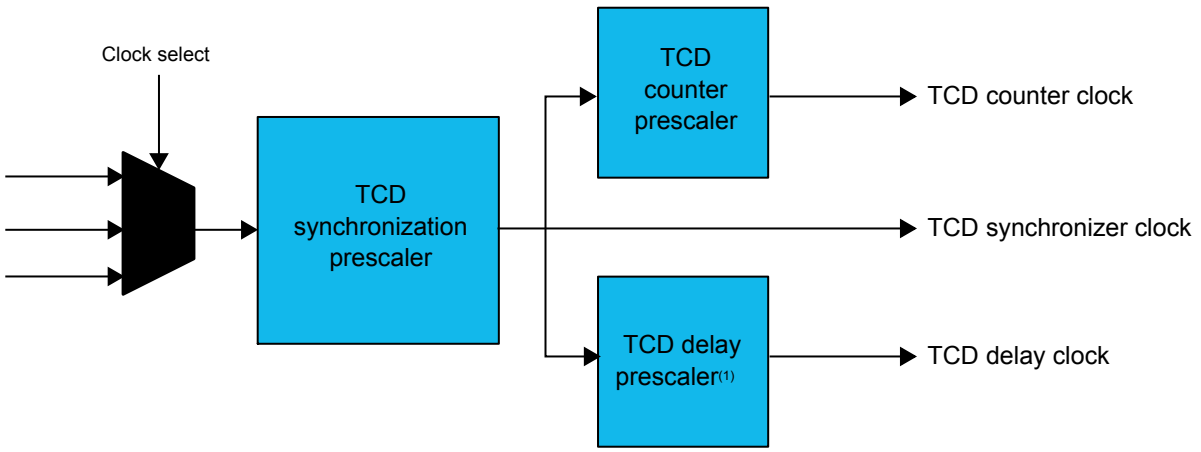
Figure 22-24. Input Mode 10 on Input B



22.6.2.5. Clock Selection and Prescalers

The TCD can select between 3 different clock sources that can be prescaled. The TCD has three different prescalers with separate controls as shown below.

Figure 22-25. Clock Selection and Prescalers Overview



1. Used by input blanking/delay event out.

The reason to have both a TCD synchronization prescaler and a TCD counter prescaler is to give the user the possibility to prescale the TCD counter clock without delaying the synchronization between the TCD core domain and the IO domain. The total prescaling on the TCD counter is:

(TCD synchronization prescaler division factor) x (TCD counter prescaler division factor)

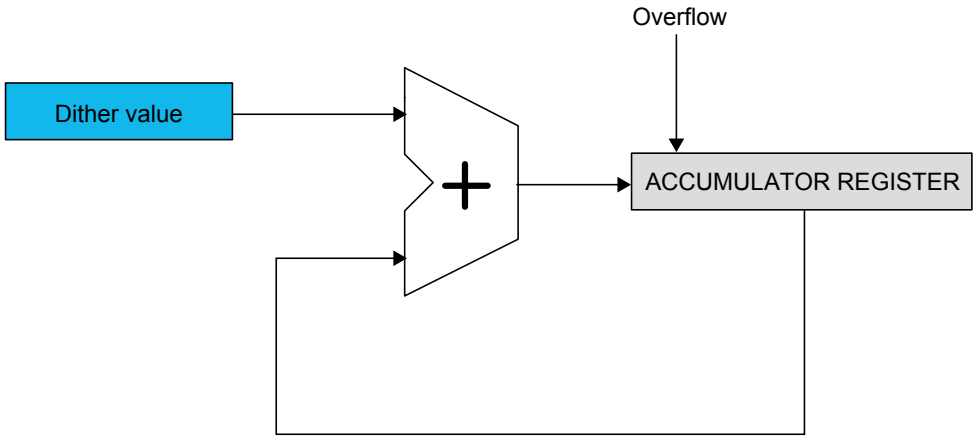
The TCD delay prescaler is used to prescale the clock used for the input blanking/ delayed event output functionality. Having separate prescalers allows the user to choose between range and accuracy for these functions independent of the TCD counter functionality.

22.6.2.6. TCD Dithering

The TCD precision is limited by the clock source accuracy: In the long term, a slightly wrong output frequency will accumulate and result in a large error. To counteract this, the TCD uses dithering, i.e. an additional TCD clock cycle is added to the TCD cycle when the accumulated error is above 1 TCD clock cycle.

The dither functionality is accumulating the fractional error of the TCD counter clock for each TCD cycle. When the fractional error overflows, an additional cycle is added to user selected part of the TCD cycle.

Figure 22-26. Dither Logic



The user can select where in the TCD cycle the dither will be added by writing to the Dither Selection bits in the Dither Control register (TCD_DITCTRL.DITHERSEL):

- On-time B
- On-time A and B
- Dead-time B
- Dead-time A and B

How much the dithering will affect the TCD cycle time depends on what Wave Generation Mode is used, see table below.

Note: Dithering is not supported in Dual Slope Mode.

Table 22-3. Mode-Dependent Dithering Additions to TCD Cycle

WAVEGEN	TCD_DITCTRL.DITHERSEL	Additional TCD clock cycles to TCD cycle
One Ramp Mode	On-time B	1
	On-time A and B	1
	Dead-time B	0
	Dead-time A and B	0

WAVEGEN	TCD_DITCTRL.DITHERSEL	Additional TCD clock cycles to TCD cycle
Two Ramp Mode	On-time B	1
	On-time A and B	2
	Dead-time B	0
	Dead-time A and B	0
Four Ramp Mode	On-time B	1
	On-time A and B	2
	Dead-time B	1
	Dead-time A and B	2
Dual Slope Mode	On-time B	0 (not supported)
	On-time A and B	0 (not supported)
	Dead-time B	0 (not supported)
	Dead-time A and B	0 (not supported)

The differences in the number of TCD clock cycles added to the TCD cycle is caused by the different number of compare values used by the TCD cycle. For example in One Ramp Mode, only CMPBCLR affects the TCD cycle time.
Note: For DITHERSEL configurations where no extra cycles are added to the TCD cycles, compensation is reached by shortening the following output state.

> in One Ramp Mode with DITHERSEL selecting Dead-time B, the dead-time B will be increased by one cycle when dither overflow occurs. This reduces On-time B by one cycle.

22.6.2.7. TCD Counter Capture

Because the TCD counter is asynchronous to the system clock it is not possible to read out the counter value directly. It is possible to capture the TCD counter value, synchronized to the IO clock domain in two different ways.

- Capture value on input Events
- Software capture

The capture logic contains two separate capture blocks, CAPTUREA and CAPTUREB, that can capture and synchronize the TCD counter value to the IO clock domain. CAPTUREA/B can be triggered by input Event A/B or by software.

The capture values can be read by reading first TCD_CAPTUREXL and then TCD_CAPTUREXH registers.

Captures Triggered by Input Events

To enable capture on input Event, write a '1' to the Action bit in the respective Event Control x register (TCD_EVCTRL.ACTION) when configuring an Event input.

When a capture has occurred, the TRIGA/B flag is raised in the Interrupt Flags register (TCD_INTFLAGS.TRIGx). The according TRIGA/B interrupt is executed if enabled by writing a '1' to the respective Trigger Interrupt x Enable bit in the Interrupt Control register (TCD_INTCTRL.TRIGx). By

polling TCD_INTFLAGS.TRIGx, the user knows that a CAPTUREx value is available, and can read out the value by reading first the TCD_CAPTUREXL and then TCD_CAPTURxH registers.

Capture Triggered by Software

Software can capture the TCD value by writing a '1' to respective Software Capture A/B Strobe bit in the Control E register (TCD_CTRLE.SCAPTUREx). When this command is executed and the Command Ready bit in the Status register (TCD_STATUS.CMDRDY) reads '1' again, the CAPTUREA/B value is available. It can now be read by reading first the TCD_CAPTUREXL and then the TCD_CAPTURxH registers.

Operating the Capture

The capture synchronization keeps the captured value until it is read:

- Reading out the High byte register of the CAPTUREx value (TCD_CAPTUREXH.CAPTURE) tells the synchronizer that synchronization is done, so a new CAPTUREx value can be captured. Consequently, the CAPTUREx value will not be available anymore on the IO clock domain, so reading the low byte value first ((TCD_CAPTUREXL.CAPTURE)) is mandatory.
- If the TCD_CAPTUREXH.CAPTURE of the CAPTUREx value is not read before a new trigger for the capture occurs (input or software), the newest capture request will be discarded, because the synchronization logic is busy.

Related Links

Initialization on page 256

22.6.2.8. Output Control

The outputs are configured by writing to the Fault Control register (TCD_FAULTCTRL). TCD_FAULTCTRL is only reset to zero after a POR reset, giving the user more control over the outputs. After any Reset, TCD_FAULTCTRL will get its values automatically from the TCD Fuse (FUSE_TCDCFG).

The Compare x Enable bits (TCD_FAULTCTRL.CMPxEN) enable the different outputs. The CMPx bits (TCD_FAULTCTRL.CMPxEN) set the value the registers should have after Reset or when a fault is triggered.

The TCD itself generates two different outputs, TCDOUTA and TCDOUTB. The two additional outputs TCDOUTC and TCDOUTD can be configured by software to be connected to either TCDOUTA or TCDOUTB by writing the Compare C/D Output Select bits in the Control C register (TCD_CTRLC.CMPCSEL and .CMPDSEL).

The user can also override the outputs based on the TCD counter state by writing a '1' to the Compare Output Value Override bit in the Control C register (TCD_CTRLC.CMPOVR). The user can then select the output values in the different dead- and on times by writing to the Compare x Value bit fields in the Control D register (TCD_CTRLD.CMPAVAL and .CMPBVAL).

Note: When used in One Ramp mode, TCD_OUTA will only use the setup for Dead Time A (DTA) and On Time A (OTA) to set the output. TCD_OUTB will only use Dead-Time B (DTB) and On Time B (OTB) values to set the output. This is due to possible overlap between the different on times and dead times in One Ramp mode.

Note: When using the override feature together with faults detection (input modes) the CMPA (and CMPC/D if TCDOUTC/D equals TCDOUTA) bit in TCD_FAULTCTRL should be equal to TCD_CTRLD.CMPAVAL[0] and [2]. if not there will be glitches on the outputs. The same applies to TCD_FAULTCTRL.CMPB (and CMPC/D if TCDOUTC/D equals TCDOUTB) bit, which should be equal to TCD_CTRLD.CMPBVAL[0] and [2].

Related Links

TCD0CFG on page 32

22.6.3. Events

The TCD can generate the following output events:

- TCD counter matches CMPBCLR
- TCD counter matches CMPASET
- TCD counter matches CMPBSET
- Programmable TCD output event. The user can select trigger, and all the different compare matches. In addition it is possible to delay the output Event from 0 to 256 TCD delay cycles.

The TCD have the possibility to receive these input Events:

- Input A
- Input B

Related Links

TCD Inputs on page 262

Asynchronous Event Detection on page 264

EVSYS - Event System on page 109

22.6.3.1. Programmable output events

Note: Programmable output event uses the same logic as the Input blanking for trigger selection and delay. Therefore it is not possible to configure the functionalities independantly. If the input blanking functionality is used the output event cannnot be delayed and the trigger used for input blanking will also be the one used for the output event.

The programmable ouput events are controlled by the DLYCTRL and DLYVAL registers. It is possible to delay the output event by 0 to 256 TCC delay clock cycles if the DLYTRIG bits in DLYCTRL is set to 0x2. The delayed output event functionality uses the TCC delay clock and counts until the DLYVAL value is reached before the trigger is sent out as a event. The TCC delay clock is a prescaled version of the TCC synchronization clock and the division factor is set by the DLYPRESC bits in the DLYCTRL register. The output event will be delayed by TCC clock period x DLYPRESC division factor x DLYVAL.

22.6.4. Interrupts

Table 22-4. Available Interrupt Vectors and Sources

Offset	Name	Vector Description	Conditions
0x00	OVF	Overflow interrupt	The TCD is done with one TCD cycle.
0x02	TRIG	Trigger interrupt	• TRIGA: Counter is entering On-Time A • TRIGB: Counter is entering On-Time B

When an interrupt condition occurs, the corresponding Interrupt Flag is set in the Interrupt Flags register of the peripheral (*peripheral*_INTFLAGS).

An interrupt source is enabled or disabled by writing to the corresponding bit in the peripheral's Interrupt Control register (*peripheral*_INTCTRL).

An interrupt request is generated when the corresponding interrupt source is enabled and the Interrupt Flag is set. The interrupt request remains active until the Interrupt Flag is cleared. See the peripheral's INTFLAGS register for details on how to clear Interrupt Flags.

Note: Interrupt requests are only generated when Interrupts are enabled globally.

When several interrupt request conditions are supported by an interrupt vector, the interrupt requests are ORed together into one combined interrupt request to the Interrupt Controller. The user must read the peripheral's INTFLAGS register to determine which of the interrupt conditions are present.

Related Links

CPUINT - CPU Interrupt Controller on page 97

SREG on page 51

22.6.5. Sleep Mode Operation

The TCD will operate as normal in Idle sleep, but in Stand By and Power Down, the TCD will stop when entering sleep.

22.6.6. Synchronization

The TCC has two different clock domains and needs to synchronize the communication between the domains. See Initialization section for detail on how the synchronization of values from the IO clock domain to the TCC clock domain is done. See the Capture section for details on how the synchronization of values from the TCC clock domain to the IO clock domain is done.

Related Links

Initialization on page 256

TCD Counter Capture on page 272

22.6.7. Configuration Change Protection

This peripheral has registers that are under Configuration Change Protection (CCP). In order to write to these, a certain key must be written to the CPU_CCP register first, followed by a write access to the protected bits within four CPU instructions.

It is possible to try writing to these registers any time, but the values are not altered.

The following registers are under CCP:

Table 22-5. TCD - Registers under Configuration Change Protection

Register	Key
TCD_FAULTCTRL	IOREG

22.7. Register Summary

Offset	Name	Bit Pos.								
0x00	CTRLA	7:0		CLKSEL[1:0]		CNTPRES[1:0]		SYNCPRES[1:0]		ENABLE
0x01	CTRLB	7:0							WGMODE[1:0]	
0x02	CTRLC	7:0	CMPDSEL	CMPCSEL			FIFTY		AUPDATE	CMPOVR
0x03	CTRLD	7:0	CMPBVAL[3:0]				CMPAVAL[3:0]			
0x04	CTRLE	7:0	DISEOC			SCAPTUREB	SCAPTUREA	RESTART	SYNC	SYNCEOC
0x05 ... 0x07	Reserved									
0x08	EVCTRLA	7:0	CFG[1:0]			EDGE		ACTION		TRIGEI
0x09	EVCTRLB	7:0	CFG[1:0]			EDGE		ACTION		TRIGEI
0x0A ... 0x0B	Reserved									
0x0C	INTCTRL	7:0					TRIGB	TRIGA		OVF
0x0D	INTFLAGS	7:0					TRIGB	TRIGA		OVF
0x0E	STATUS	7:0	PWMACTB	PWMACTA					CMDRDY	ENRDY
0x0F	Reserved									
0x10	INPUTCTRLA	7:0					INPUTMODE[3:0]			
0x11	INPUTCTRLB	7:0					INPUTMODE[3:0]			
0x12	FAULTCTRL	7:0	CMPDEN	CMPCEN	CMPBEN	CMPAEN	CMPD	CMPC	CMPB	CMPA
0x13	Reserved									
0x14	DLYCTRL	7:0			DLYPRESC[1:0]		DLYTRIG[1:0]		DLYSEL[1:0]	
0x15	DLYVAL	7:0	DLYVAL[7:0]							
0x16 ... 0x17	Reserved									
0x18	DITCTRL	7:0							DITHERSEL[1:0]	
0x19	DITVAL	7:0					DITHER[3:0]			
0x1A ... 0x1D	Reserved									
0x1E	DBGCTRL	7:0						FAULTDET		DBGRUN
0x1F ... 0x21	Reserved									
0x22	CAPTURELA	7:0	CAPTURE[7:0]							
0x23	CAPTUREHA	7:0					CAPTURE[11:8]			
0x24	CAPTURELB	7:0	CAPTURE[7:0]							
0x25	CAPTUREHB	7:0					CAPTURE[11:8]			
0x26 ... 0x27	Reserved									
0x28	CMPSETLA	7:0	CMPSET[7:0]							
0x29	CMPSETHA	7:0					CMPSET[11:8]			
0x2A	CMPCLRLA	7:0	CMPCLR[7:0]							

Offset	Name	Bit Pos.								
0x2B	CMPCLRHA	7:0					CMPCLR[11:8]			
0x2C	CMPSETLB	7:0	CMPSET[7:0]							
0x2D	CMPSETHB	7:0					CMPSET[11:8]			
0x2E	CMPCLRLB	7:0	CMPCLR[7:0]							
0x2F	CMPCLRHB	7:0					CMPCLR[11:8]			

22.8. Register Description

22.8.1. Control A

Name: CTRLA
Offset: 0x00
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
		CLKSEL[1:0]		CNTPRES[1:0]		SYNCPRES[1:0]		ENABLE
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bits 6:5 – CLKSEL[1:0]: Clock Select
The clock select bits set the clock source of the TCD clock. This bit can only be changed when the TCD is not enabled.

CLKSEL[1:0]	Clock Source
0x0	20MHz internal oscillator
0x1	Reserved
0x2	External clock
0x3	System clock

Bits 4:3 – CNTPRES[1:0]: Counter Prescaler
The counter prescaler bits set the division factor of the TCD counter clock. This bit can only be changed when the TCD is not enabled.

CNTPRES[1:0]	Division Factor
0x0	1
0x1	4
0x2	32
0x3	Reserved

Bits 2:1 – SYNCPRES[1:0]: Synchronization Prescaler
The synchronization prescaler bits set the division factor of the TCD clock. This bit can only be changed when the TCD is not enabled.

SYNCPRES[1:0]	Division factor
0x0	1
0x1	2
0x2	4
0x3	Reserved

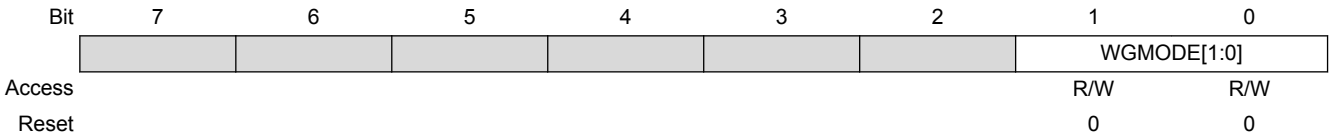
Bit 0 – ENABLE: Enable
When this bit is set the TCD is enabled and is running.

When written to, it will automatically be synchronized to the TCD clock domain.

This bit can be changed as long as synchronization of this bit is not ongoing, see Enable Ready bit in Status register (STATUS.ENRDY).

22.8.2. Control B

Name: CTRLB
Offset: 0x01
Reset: 0x00
Property: -



Bits 1:0 – WGMODE[1:0]: Waveform Generation Mode
These bits select the waveform generation

Value	Name	Description
0x0	ONERAMP	One ramp mode
0x1	TWORAMP	Two ramp mode
0x2	FOURRAMP	Four ramp mode
0x3	DS	Dual-slope mode

22.8.3. Control C

Name: CTRLC
Offset: 0x02
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	CMPDSEL	CMPCSEL			FIFTY		AUPDATE	CMPOVR
Access	R/W	R/W			R/W		R/W	R/W
Reset	0	0			0		0	0

Bit 7 – CMPDSEL: Compare D Output Select

Value	Name	Description
0	PWMA	Waveform A
1	PWMB	Waveform B

Bit 6 – CMPCSEL: Compare C Output Select

Value	Name	Description
0	PWMA	Waveform A
1	PWMB	Waveform B

Bit 3 – FIFTY: Fifty Percent Waveform
If the two waveforms have identical characteristics, this bit can be written to '1'. This will cause any values written to register CMPBSET/CLR also to be written to the register CMPASET/CLR.

Bit 1 – AUPDATE: Automatically Update
If this bit is set a synchronization at the end of the TCD cycle is automatically requested after the Compare B Clear High register (CMPBCLRH) is written.

If the Fifty Percent Waveform is enabled by setting the FIFTY bit in this register, writing the Compare A Clear High register will also request a synchronization at the end of the TCD cycle if the AUPDATE bit is set.

Bit 0 – CMPOVR: Compare Output Value Override
When this bit is written, default values of the Waveform Outputs A and B are overridden by the values written in the Compare x Value in active state bit fields in the Control D register (CTRLD.CMPnxVAL). See the Control D register description for more details.

22.8.4. Control D

Name: CTRLD
Offset: 0x03
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	CMPBVAL[3:0]				CMPAVAL[3:0]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 0:3, 4:7 – CMPAxVAL, CMPBxVAL: Compare x Value (in active state)
These bits sets the active state for the different ramps for compare x.

These settings are only valid if the Compare Output Value Override bit in the Control C register (CTRLC.CMPOVR) is written to '1'.

CMPxVAL	A_off	A_on	B_off	B_on
PWMA	CMPAVAL[0]	CMPAVAL[1]	CMPAVAL[2]	CMPAVAL[3]
PWMB	CMPBVAL[0]	CMPBVAL[1]	CMPBVAL[2]	CMPBVAL[3]

Note: In One Ramp mode, PWMA will only use A_off and A_on values and PWMB will only use B_off and B_on values. This is due to possible overlap between the values A_off, A_on, B_off and B_on.

22.8.5. Control E

Name: CTRLE
Offset: 0x04
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	DISEOC			SCAPTUREB	SCAPTUREA	RESTART	SYNC	SYNCEOC
Access	R/W			R/W	R/W	R/W	R/W	R/W
Reset	0			0	0	0	0	0

Bit 7 – DISEOC: Disable at End of TCD Cycle Strobe
When this bit is written, the TCD will automatically disable at the end of the TCD cycle.

When this bit is written to '1', the ENRDY in TCD.STATUS will keep low until the TCD is disabled.

Writing to this bit only has effect if there is no ongoing synchronization of Enable. See also ENRDY bit in TCD.STATUS.

Bit 4 – SCAPTUREB: Software Capture B Strobe
When this bit is written to '1', a software capture to Capture register B (TCD.CAPTUREBL/H) is done as soon as the strobe is synchronized to the TCD domain.

Writing to this bit only has effect if there is no ongoing synchronization of a command. See also CMDRDY bit in TCD.STATUS.

Bit 3 – SCAPTUREA: Software Capture A Strobe
When this bit is written to '1', a software capture to Capture register A (TCD.CAPTUREAL/H) is done as soon as the strobe is synchronized to the TCD domain.

Writing to this bit only has effect if there is no ongoing synchronization of a command. See also CMDRDY bit in TCD.STATUS.

Bit 2 – RESTART: Restart Strobe
When this bit is written a Restart of the TCD counter is executed as soon as this bit is synchronized to the TCD domain.

Writing to this bit only has effect if there is no ongoing synchronization of a command. See also CMDRDY bit in TCD.STATUS.

Bit 1 – SYNC: Synchronize Strobe
When this bit is written to '1' the doubled buffered registers will be loaded to the TCD domain as soon as this bit is synchronized to the TCD domain.

Writing to this bit only has effect if there is no ongoing synchronization of a command. See also CMDRDY bit in TCD.STATUS.

Bit 0 – SYNCEOC: Synchronize End of PSC Cycle Strobe
When this bit is written to '1' the doubled buffered registers will be loaded to the TCD domain at the end of the next TCD cycle.

Writing to this bit only has effect if there is no ongoing synchronization of a command. See also CMDRDY bit in TCD.STATUS.

22.8.6. Event Control x

Name: EVCTRLA, EVCTRLB
Offset: 0x08 + n*0x01 [n=0..1]
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	CFG[1:0]			EDGE		ACTION		TRIGEI
Access	R/W	R/W		R/W		R/W		R/W
Reset	0	0		0		0		0

Bit 7 – ASYNC: Asynchronous Event Control
Writing this bit to '1' will cause

Bits 7:6 – CFG[1:0]: Event Configuration
When the Input Capture Noise canceler is activated (FILTERON), the Event input is filtered. The filter function requires four successive equal valued samples of the Retrigger pin for changing its output. The Input Capture is therefore delayed by four clock cycles when the noise canceler is enabled.

When the Asynchronous Event is enabled (ASYNCON), the Event input will qualify the output directly.

Value	Name	Description
0x1	FILTERON	Input Capture Noise Cancellation Filter enabled.
0x2	ASYNCON	Asynchronous Event output qualification enabled.
other	-	Neither Filter nor Asynchronous Event is enabled.

Bit 6 – FILTER: Filter Enable

Bit 4 – EDGE: Edge Selection
This bit is used to select the active edge or level for the event input.

Value	Name	Description
0	FALL_LOW	The falling edge or low level of the Event input generates Retrigger or Fault action.
1	RISE_HIGH	The rising edge or high level of the Event input generates Retrigger or Fault action.

Bit 2 – ACTION: Event Action
This bit enables Capture on Event input. By default, the input will trigger a Fault, depending on the Input x register input mode (TCD.INPUTx). It is also possible to trigger a Capture on the Event input.

Value	Name	Description
0	FAULT	Event triggers a Fault.
1	CAPTURE	Event triggers a Fault and Capture.

Bit 0 – TRIGEI: Trigger Event Input Enable
Writing this bit to '1' enables Event as trigger for input A.

22.8.7. Interrupt Control

Name: INTCTRL
Offset: 0x0C
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
					TRIGB	TRIGA		OVF
Access					R/W	R/W		R/W
Reset					0	0		0

Bit 0 – OVF: Counter Overflow
Writing this bit to '1' enables executing an interrupt at Restart of the sequence or Overflow of the counter.

Bits 2, 3 – TRIGA, TRIGB: Trigger x Interrupt Enable
Writing this bit to '1' enables executing an interrupt when trigger input x is received.

22.8.8. Interrupt Flags

Name: INTFLAGS
Offset: 0x0D
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
					TRIGB	TRIGA		OVF
Access					R/W	R/W		R/W
Reset					0	0		0

Bit 0 – OVF: Overflow Interrupt Flag
This bit is cleared by writing a '1' to it.

Bits 2, 3 – TRIGA, TRIGB: Trigger x Interrupt Flag
This bit is cleared by writing a '1' to it.

22.8.9. Status

Name: STATUS
Offset: 0x0E
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	PWMACTB	PWMACTA					CMDRDY	ENRDY
Access	R/W	R/W					R	R
Reset	0	0					0	0

Bit 1 – CMDRDY: Command Ready
This status bit tells when a command is synced to the PSC domain and the system is ready to receive new commands.

The following clears the CMDRDY bit:

1. TCD.CTRLE SYNCEOC strobe
2. TCD.CTRLE SYNC strobe
3. TCD.CTRLE RESTART strobe
4. TCD.CTRLE SCAPTUREA Capture A strobe
5. TCD.CTRLE SCAPTUREB Capture B strobe
6. TCD.CTRLC AUPDATE written to '1' and writing to TCD.CMPBCLRH register

Bit 0 – ENRDY: Enable Ready
This status bit tells when the ENABLE value in TCD.CTRLA is synced to the TCD domain, and is ready to be written to again.

The following clears the ENRDY bit:

1. Writing to the ENABLE bit in (TCD.CTRLA
2. TCD.CTRLE DISEOC strobe
3. Going to break in an On-Chip Debugging (OCD) session with and the Debug Run bit (DBGCTRL) in TCD.DBGCTRL is not '1'

Bits 6, 7 – PWMACTA, PWMACTB: PWM Activity on x
This bit is set by hardware each time the output TCDOUT1 toggles from 0 to 1 or from 1 to 0.

This status bit must be cleared by software by writing a '1' to it before new PWM activity can be detected.

22.8.10. Input x Control

Name: INPUTCTRLA, INPUTCTRLB
Offset: 0x10 + n*0x01 [n=0..1]
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
					INPUTMODE[3:0]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bits 3:0 – INPUTMODE[3:0]: Input Mode

INPUTMODE[3:0]	Group Configuration	Trigger(1)	Fault On/Active	Fault Release/ Inactive
0x00	NONE	-	No action	No action
0x01	-	A and B	End current on-time and wait	Start with dead-time for other compare
0x02	-	A and B	End current on-time. Execute other and wait	Start with dead-time for current compare
0x03	-	A and B	End current on-time. Execute other compare cycle	Re-enable current compare cycle
0x04	-	A or B	Deactivate outputs	
0x05	-	A or B	Execute dead-time only	
0x06	-	A or B	End on-time and wait	Start with dead-time for other compare
0x07	-	A or B	End on-time and wait for software action	Start with dead-time for current compare
0x08	-	A or B	End current on-time and continue with other off-time	
0x09	-	A and B	Block current on-time and continue sequence	
0x0A	-	A and B	Deactive on-time until end of sequence while trigger is active	
other	-	-	-	-

1. "A and B" means that A can only trigger fault during A-period, and B can only trigger fault during B-period. "A or B" means that either A or B can trigger fault at any time.

22.8.11. Fault Control

Name: FAULTCTRL
Offset: 0x12
Reset: 0x00
Property: Configuration Change Protection

Bit	7	6	5	4	3	2	1	0
	CMPDEN	CMPCEN	CMPBEN	CMPAEN	CMPD	CMPC	CMPB	CMPA
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 4, 5, 6, 7 – CMPAEN, CMPBEN, CMPCEN, CMPDEN: Compare x Enable
These bits enable Compare as output on pin. The settings are not reset on any Reset except POR. Any other Reset will reload the values from the TCD Configuration fuse (FUSE.TCDCFG).

Bits 0, 1, 2, 3 – CMPA, CMPB, CMPC, CMPD: Compare Value x
These bits set the default state from Reset, or when a input Event triggers a Fault causing changes to the output. The settings are not reset on any Reset except POR. Any other Reset will reload the values from the TCD Configuration fuse (FUSE.TCDCFG).

22.8.12. Delay Control

Name: DLYCTRL
Offset: 0x14
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
			DLYPRESC[1:0]		DLYTRIG[1:0]		DLYSEL[1:0]	
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bits 5:4 – DLYPRESC[1:0]: Delay Prescaler
These bits control the prescaler settings for the blanking or Output event delay.

DLYPRESC[1:0]	Prescaler Division Factor
0x0	1
0x1	2
0x2	4
0x3	8

Bits 3:2 – DLYTRIG[1:0]: Delay Trigger
These bits control what should trigger the blanking or output event delay

DLYTRIG[1:0]	Group Configuration	Description
0x0	CMPASET	
0x1	CMPACLR	
0x2	CMPBSET	
0x3	CMPBCLR (End of Cycle)	

Bits 1:0 – DLYSEL[1:0]: Delay Select
These bits control what function should be used by the delay trigger the blanking or output event delay.

DLYSEL[1:0]	Description
0x0	Delay functionality not used
0x1	Input blanking enabled
0x2	Event delay enabled
0x3	Reserved

22.8.13. Delay Value

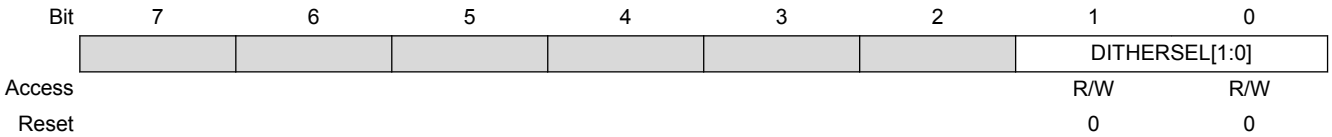
Name: DLYVAL
Offset: 0x15
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	DLYVAL[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – DLYVAL[7:0]: Delay Value
These bits configure the blanking/ output event delay time or event output synchronization delay in number of prescaled TCD cycles.

22.8.14. Dither Control

Name: DITCTRL
Offset: 0x18
Reset: 0x00
Property: -



Bits 1:0 – DITHERSEL[1:0]: Dither Select
These bits select which Compare register is using the dither function. See also TCD Dithering.

Value	Name	Description
0x0	ONTIMEB	On-time ramp B
0x1	ONTIMEAB	On-time ramp A and B
0x2	DEADTIMEB	Dead-time ramp B
0x3	DEADTIMEAB	Dead-time ramp A and B

22.8.15. Dither Value

Name: DITVAL
Offset: 0x19
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
					DITHER[3:0]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bits 3:0 – DITHER[3:0]: Dither Value
These bits configure the fractional adjustment of the on-time or off-time according to Dither Selection bits (DITHERSEL) in the Dither Control register (TCD.DITCTRL). The DITHER value is added to a 4-bit accumulator at the end of each TCD cycle. When the accumulator overflows the frequency adjustment will occur.

The DITHER bits are doubled buffered so the new value is copied in at an update condition.

22.8.16. Debug Control

Name: DBGCTRL
Offset: 0x1E
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
						FAULTDET		DBGRUN
Access						R/W		R/W
Reset						0		0

Bit 2 – FAULTDET: Fault Detection
This bit defines how the peripheral behaves when stopped in Debug Mode.

Value	Name	Description
0	NONE	No fault is generated if PSC is stopped in debug mode.
1	FAULT	A fault is generated and both trigger flags are set if PSC is halted in debug mode.

Bit 0 – DBGRUN: Debug Run
When written to '1', the peripheral will continue operating in debug mode.

22.8.17. Capture x Low byte

The TCD.CAPTUREXH and TCD.CAPTUREXL register pair represents the 12-bit value CAPTUREx. For capture operation, these registers constitute the second buffer level and access point for the CPU. The CAPTUREXL/H registers are updated with the buffer value when an UPDATE condition occurs. CAPTURE A register contains the value from the TCD counter when a Trigger A or a software capture A occurs. CAPTURE B register contain the value from the TCD counter when Trigger B or software capture B occurs.

Note: To ensure expected behavior, read TCD.CAPTUREXL prior to TCD.CAPTUREXH.

Name: CAPTURELA, CAPTURELB
Offset: 0x22 + n*0x02 [n=0..1]
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	CAPTURE[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – CAPTURE[7:0]: Capture Low Byte
These bits hold the LSB of the 12-bit capture register.

22.8.18. Capture x High byte

The TCD.CAPTUREXH and TCD.CAPTUREXL register pair represents the 12-bit value CAPTUREx.

Note: To ensure expected behavior, read TCD.CAPTUREXL prior to TCD.CAPTUREXH.

Name: CAPTUREHA, CAPTUREHB
Offset: 0x23 + n*0x02 [n=0..1]
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
					CAPTURE[11:8]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bits 3:0 – CAPTURE[11:8]: Capture High Byte

These bits hold the MSB of the 12-bit capture register.

22.8.19. Compare x Set Low byte

The TCD.CMPxSETH and TCD.CMPxSETL register pair represents the 12-bit value CMPxSET. For compare operation, these registers are continuously compared to the counter value. Normally, the outputs form the comparators are then used for generating waveforms.

Name: CMPSETLA, CMPSETLB
Offset: 0x28 + n*0x04 [n=0..1]
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	CMPSET[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – CMPSET[7:0]: Compare x Set low byte
These bits hold the LSB of the 12-bit compare register.

22.8.20. Compare x Set High byte

The TCD.CMPxSETH and TCD.CMPxSETL register pair represents the 12-bit value CMPxSET.

Name: CMPSETHA, CMPSETHB
Offset: 0x29 + n*0x04 [n=0..1]
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
					CMPSET[11:8]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bits 3:0 – CMPSET[11:8]: Compare x Set high byte

These bits hold the MSB of the 12-bit compare register

22.8.21. Compare x Clear Low byte

The TCD.CMPxCLRH and TCD.CMPxCLRL register pair represents the 12-bit value CMPxCLR. For compare operation, these registers are continuously compared to the counter value. Normally, the outputs form the comparators are then used for generating waveforms.

Name: CMPCLRLA, CMPCLRLB
Offset: 0x2A + n*0x04 [n=0..1]
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	CMPCLR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – CMPCLR[7:0]: Compare x Clear Low Byte
These bits hold the LSB of the 12-bit compare register.

22.8.22. Compare x Clear High byte

The TCD.CMPxCLRH and TCD.CMPxCLRL register pair represents the 12-bit value CMPxCLR. For compare operation, these registers are continuously compared to the counter value. Normally, the outputs form the comparators are then used for generating waveforms.

Name: CMPCLRHA, CMPCLRHB
Offset: 0x2B + n*0x04 [n=0..1]
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
					CMPCLR[11:8]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bits 3:0 – CMPCLR[11:8]: Compare x Clear high byte
These bits hold the MSB of the 12-bit compare register

23. RTC - Real Time Counter

23.1. Overview

The RTC peripheral offers two timing functions: the Real-Time Counter (RTC) and a Periodic Interrupt Timer (PIT).

The RTC peripheral typically runs continuously, including in low-power sleep modes, to keep track of time. It can wake up the device from sleep modes and/or interrupt the device at regular intervals.

The reference clock is typically the 32.768kHz output from a high-accuracy crystal. The RTC can also be clocked from an external clock signal, the 32KHz internal Ultra-Low Power oscillator (OSCULP32K), or the OSCULP32K divided by 32.

The RTC peripheral includes a 15-bit programmable prescaler that can scale down the reference clock before it reaches the counter. A wide range of resolutions and time-out periods can be configured for the RTC. With a 32.768kHz clock source, the maximum resolution is 30.5µs, and time-out periods can range up to 2 seconds. With a resolution of 1s, the maximum time-out period is more than 18 hours (65536 seconds). The RTC can give a compare interrupt and/or Event when the counter equals the compare register value, and an overflow interrupt and/or Event when it equals the period register value.

The RTC peripheral also provides a Periodic Interrupt Timer (PIT). Using the same clock source as the RTC function, the PIT can request an interrupt or trigger an output Event on every *n*-th clock period. *n* can be selected from {4, 8, 16,... 32768}.

The PIT functionality can be enabled independent of the RTC functionality.

Related Links

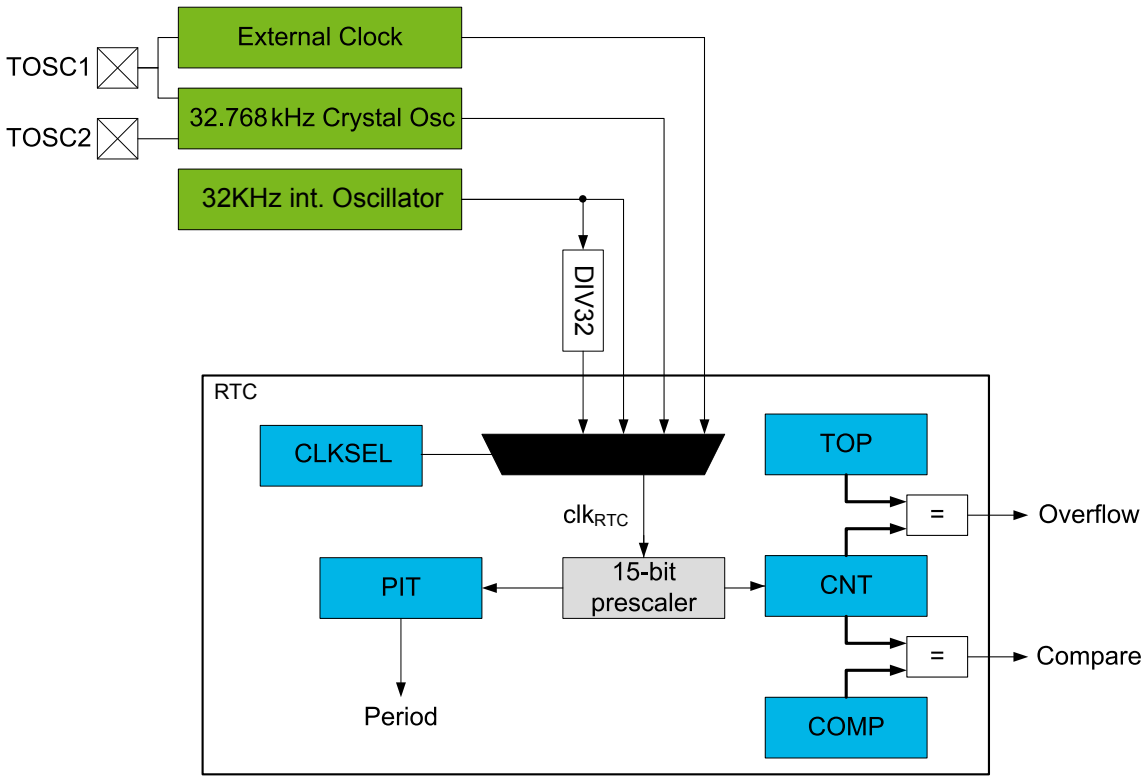
RTC Functional Description on page 304
PIT Functional Description on page 305

23.2. Features

- 16-bit resolution
- Selectable clock source
  - 32.768kHz external crystal (XOSC32K)
  - External clock
  - 32KHz internal ULP oscillator (OSCULP32K)
  - OSCULP32K divided by 32
- Programmable 15-bit clock prescaling
- One compare register
- One period register
- Clear timer on period overflow
- Optional interrupt/Event on overflow and compare match
- Periodic interrupt and Event

23.3. Block Diagram

Figure 23-1. Block Diagram



23.4. Signal Description

Not applicable.

23.5. Product Dependencies

In order to use this peripheral, other parts of the system must be configured correctly, as described below.

Table 23-1. RTC Product Dependencies

Dependency	Applicable	Peripheral
Clocks	Yes	CLKCTRL
I/O Lines and Connections	Yes	PORT
Interrupts	Yes	CPUINT
Events	Yes	EVSYS
Debug	Yes	UPDI

Related Links

Clocks on page 304
I/O Lines and Connections on page 304
Interrupts on page 54

Events on page 181
Debug Operation on page 304

23.5.1. Clocks

System clock (CLK_PER) is required to be at least four times faster than RTC clock (CLK_RTC) for reading counter value.

Related Links
CLKCTRL - Clock Controller on page 68

23.5.2. I/O Lines and Connections

A 32.768kHz crystal can be connected to the TOSC1 or TOSC2 pins, along with any required load capacitors.

An external clock can be used on the TOSC1 pin.

Related Links
CLKCTRL - Clock Controller on page 68
Electrical Characteristics on page 532

23.5.3. Interrupts

Using the interrupts of this peripheral requires the Interrupt Controller to be configured first.

Related Links
CPUINT - CPU Interrupt Controller on page 97
SREG on page 51
Interrupts on page 136

23.5.4. Events

The events of this peripheral are connected to the Event System.

Related Links
EVSYS - Event System on page 109

23.5.5. Debug Operation

When run-time debugging, this peripheral will continue normal operation. Halting the CPU in debugging mode will halt normal operation of the peripheral.

This peripheral can be forced to operate with halted CPU by writing a '1' to the Debug Run bit in the Debug Control register of the peripheral (*peripheral*_DBGCTRL.DBGRUN).

Related Links
UPDI - Unified Program Debug Interface on page 490

23.6. RTC Functional Description

The RTC peripheral offers two timing functions: the Real-Time Counter (RTC) and a Periodic Interrupt Timer (PIT). This subsection describes the RTC.

Related Links
PIT Functional Description on page 305

23.6.1. Principle of Operation

The RTC peripheral offers two timing functions: the Real-Time Counter (RTC) and a Periodic Interrupt Timer (PIT).

RTC The RTC is counting (prescaled) clock cycles in a Counter register, and compares the content of the Counter register to a Period register and a Compare register.

The RTC can generate both interrupts and Events. The RTC will give a compare interrupt and/or Event at the first count after the counter value equals the Compare register value.

The RTC will give an overflow interrupt request and/or Event at the first count after the counter value equals the Period register value. The overflow will also reset the Counter value to zero.

PIT Using the same clock source as the RTC function (CLK_RTC), the PIT can request an interrupt or trigger an output Event on every n-th clock period of CLK_RTC. n can be selected from {4, 8, 16,... 32768}.

23.6.2. Basic Operation - RTC

23.6.2.1. Initialization

To operate the RTC, the source clock for the RTC counter must be configured before enabling the RTC peripheral, and the desired actions (interrupt requests, output Events) must be configured:

- Configure the RTC clock CLK_RTC:
  1.1. In the Clock Controller (CLKCTRL), configure the desired oscillator to operate as required.
  1.2. In the Clock Control register, write the Clock Select bits accordingly (RTC_CLKCTRL.CLKSEL).
  1.3. Configure the RTC-internal prescaler by writing the Prescaler bit field in the Control A register (RTC_CTRLA.PRESCALER).

  Note: The RTC clock configuration is used by both RTC and PIT functionality.
- Configure the actions on Compare match and Overflow:
  2.1. Enable the desired Interrupts by writing to the respective Interrupt Enable bits in the Interrupt Control register (RT_INTCTRL.CMP, .OVF).
  2.2. When an interrupt is enabled, the according output Event is enabled, too.
- Optional: configure the Periodic Interrupt Timer (PIT)
- Enable the RTC by writing a '1' to the RTC Enable bit in the Control A register (RTC_CTRLA.RTCEN).

Related Links

CLKCTRL - Clock Controller on page 68
PIT Functional Description on page 305

23.6.2.2. Enabling, Disabling, and Resetting

The RTC is enabled by setting the Enable bit in the Control A register (CTRLA.ENABLE=1). The RTC is disabled by writing CTRLA.ENABLE=0.

23.6.3. PIT Functional Description

The RTC peripheral offers two timing functions: the Real-Time Counter (RTC) and a Periodic Interrupt Timer (PIT). This subsection describes the PIT.

Related Links

PIT Functional Description on page 305

23.6.3.1. Principle of Operation

The PIT can generate both interrupt and events. One interrupt at given periodic intervall, and several events selectable in the event system.

23.6.3.2. Basic Operation - PIT

Initialization

To operate the PIT, follow these steps:

1. Configure the RTC clock CLK_RTC as described in Initialization.
2. Select the period for the interrupt by writing the Period bit field in the PIT Control A register (RTC_PITCTRLA.PERIOD).
3. Enable the interrupt by writing a '1' to the Periodic Interrupt bit in the PIT Interrupt Control register (RTC_PITINTCTRL.PI).
4. Optional: configure the RTC function as described in Initialization.
5. Enable the PIT by writing a '1' to the PIT Enable bit in the PIT Control A register (RTC_PITCTRLA.PITENABLE).

Enabling, Disabling, and Resetting

The PIT is enabled by setting the Enable bit in the PIT Control A register (RTC_PITCTRLA.PITEN=1).
The PIT is disabled by writing RTC_PITCTRLA.PITEN=0.

Note: The PIT will be enabled even the RTC Enable bit in the Control A register (RTC_CTRLA.RTCEN) is written to '0'.

23.6.4. Events

The RTC can generate the following output Events:

- Overflow (OVF): Generated when the counter has reached its top value and wrapped to zero.
- Compare (CMP): Indicates a match between the counter value and the compare register.

The PIT can generate the following output Events:

- Event 0: 8192 RTC clock periods interval.
- Event 1: 4096 RTC clock periods interval.
- Event 2: 2048 RTC clock periods interval.
- Event 3: 1024 RTC clock periods interval.
- Event 4: 512 RTC clock periods interval.
- Event 5: 256 RTC clock periods interval.
- Event 6: 128 RTC clock periods interval.
- Event 7: 64 RTC clock periods interval.

Related Links

EVSYS - Event System on page 109

23.6.5. Interrupts

Table 23-2. Available Interrupt Vectors and Sources

Offset	Name	Vector Description	Conditions
0x00	RTC	Real-time counter overflow and compare match interrupt	• Overflow (OVF): The counter has reached its top value and wrapped to zero. • Compare (CMP): Match between the counter value and the compare register.
0x02	PIT	Periodic Interrupt Timer interrupt	A time period has passed, as configured in RTC_PITCTRLA.PERIOD.

When an interrupt condition occurs, the corresponding Interrupt Flag is set in the Interrupt Flags register of the peripheral (*peripheral*_INTFLAGS).

An interrupt source is enabled or disabled by writing to the corresponding bit in the peripheral's Interrupt Control register (*peripheral*_INTCTRL).

An interrupt request is generated when the corresponding interrupt source is enabled and the Interrupt Flag is set. The interrupt request remains active until the Interrupt Flag is cleared. See the peripheral's INTFLAGS register for details on how to clear Interrupt Flags.

Note: Interrupt requests are only generated when Interrupts are enabled globally.

When several interrupt request conditions are supported by an interrupt vector, the interrupt requests are ORed together into one combined interrupt request to the Interrupt Controller. The user must read the peripheral's INTFLAGS register to determine which of the interrupt conditions are present.

Related Links

CPUINT - CPU Interrupt Controller on page 97
INTCTRL on page 313
PITINTCTRL on page 323

23.6.6. Sleep Mode Operation

The RTC will continue to operate in any sleep mode where the source clock is active. It will also run in Standby sleep mode, if the Run in Standby bit in the Control A register is written to '1' (RTC_CTRLA.RUNSTDBY).

The PIT will continue to operate in any sleep mode.

Related Links

CTRLA on page 310

23.6.7. Synchronization

Both the RTC and the PIT are asynchronous, operating from a different clock source (CLK_RTC) independently of the main clock (CLK_PER). For control and count register updates, it will take a number of RTC clock and/or peripheral clock cycles before an updated register value is available in a register or until a configuration change has effect on the RTC or PIT, respectively.

for some RTC registers, a synchronization Busy flag is available in the Status register (RTC_STATUS.CMPBSY, .PERBSY, .CNTBSY, .CTRLABSY).

for the RTC_PITCTRLA register, a synchronization Busy flag is available in the PIT Status register (RTC_PITSTATUS.SYNCBUSY).

Related Links

CLKCTRL - Clock Controller on page 68

23.6.8. Configuration Change Protection

Not applicable.

23.7. Register Summary

Offset	Name	Bit Pos.								
0x00	CTRLA	7:0	RUNSTDBY	PRESCALER[3:0]				CORREN		RTCEN
0x01	STATUS	7:0					CMPBSY	PERBSY	CNTBSY	CTRLABSY
0x02	INTCTRL	7:0							CMP	OVF
0x03	INTFLAGS	7:0							CMP	OVF
0x04	TEMP	7:0	TEMP[7:0]							
0x05	DBGCTRL	7:0								DBGRUN
0x06	Reserved									
0x07	CLKCTRL	7:0							CLKSEL[1:0]	
0x08	CNT	7:0	CNT[7:0]							
0x09		15:8	CNT[15:8]							
0x0A	PER	7:0	PER[7:0]							
0x0B		15:8	PER[15:8]							
0x0C	CMP	7:0	CMP[7:0]							
0x0D		15:8	CMP[15:8]							
0x0E ... 0x0F	Reserved									
0x10	PITCTRLA	7:0		PERIOD[3:0]						PITEN
0x11	PITSTATUS	7:0								SYNCBUSY
0x12	PITINTCTRL	7:0								PI
0x13	PITINTFLAGS	7:0								PI
0x14	Reserved									
0x15	PITDBGCTRL	7:0								DBGRUN

23.8. Register Description

23.8.1. Control A

Name: CTRLA
Offset: 0x00
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	RUNSTDBY	PRESCALER[3:0]				CORREN		RTCEN
Access	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
Reset	0	0	0	0	0	0	0	0

Bit 7 – RUNSTDBY: Run in Standby

Value	Description
0	RTC disabled in Standby sleep mode
1	RTC enabled in Standby sleep mode

Bits 6:3 – PRESCALER[3:0]: Prescaler
These bits define the prescaling of the CLK_RTC clock signal.

Value	Name	Description
0x0	DIV1	RTC clock / 1 (no prescaling)
0x1	DIV2	RTC clock / 2
0x2	DIV4	RTC clock / 4
0x3	DIV8	RTC clock / 8
0x4	DIV16	RTC clock / 16
0x5	DIV32	RTC clock / 32
0x6	DIV64	RTC clock / 64
0x7	DIV128	RTC clock / 128
0x8	DIV256	RTC clock / 256
0x9	DIV512	RTC clock / 512
0xA	DIV1024	RTC clock / 1024
0xB	DIV2048	RTC clock / 2048
0xC	DIV4096	RTC clock / 4096
0xD	DIV8192	RTC clock / 8192
0xE	DIV16384	RTC clock / 16384
0xF	DIV32768	RTC clock / 32768

Bit 2 – CORREN: Correction Enable

Value	Description
0	Correction is disabled
1	Correction is enabled

Bit 0 – RTCEN: RTC Enable

Value	Description
0	RTC disabled
1	RTC enabled

23.8.2. Status

Name: STATUS
Offset: 0x01
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
					CMPBSY	PERBSY	CNTBSY	CTRLABSY
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit 3 – CMPBSY: Comp Busy
This bit is indicating whether the RTC is busy synchronizing the Compare register (RTC.CMP) in RTC clock domain.

Bit 2 – PERBSY: Top Busy
This bit is indicating whether the RTC is busy synchronizing the Period register (RTC.PER) in RTC clock domain.

Bit 1 – CNTBSY: CNT Busy
This bit is indicating whether the RTC is busy synchronizing the Count register (RTC.CNT) in RTC clock domain.

Bit 0 – CTRLABSY: CTRLA Busy
This bit is indicating whether the RTC is busy synchronizing the Control A register (RTC.CTRLA) in RTC clock domain.

23.8.3. Interrupt Control

Name: INTCTRL
Offset: 0x02
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
							CMP	OVF
Access	R	R	R	R	R	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 1 – CMP: Compare Match Interrupt Enable
Enable interrupt on compare match, i.e. when the Counter value (RTC.CNT) matches the Compare value (RTC.CMP).

Bit 0 – OVF: Overflow Interrupt Enable
Enable interrupt on counter overflow, i.e. when the Counter value (RTC.CNT) matched the Period value (RTC.PER) and wraps around to zero.

23.8.4. Interrupt Flag

Name: INTFLAGS
Offset: 0x03
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
							CMP	OVF
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit 1 – CMP: Compare Match Interrupt Flag
This flag is set when the Counter value (RTC.CNT) matches the Compare value (RTC.CMP).

Writing a '1' to this bit clears the flag.

Bit 0 – OVF: Overflow Interrupt Flag
This flag is set when the Counter value (RTC.CNT) has reached the Period value (RTC.PER) and wrapped to zero.

Writing a '1' to this bit clears the flag.

23.8.5. Temporary

The Temporary register is used by the CPU for single-cycle, 16-bit access to the 16-bit registers of this peripheral. It can also be read and written by software. See also Accessing 16-bit Registers. There is one common Temporary register for all the 16-bit registers of this peripheral.

Name: TEMP
Offset: 0x4
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	TEMP[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – TEMP[7:0]: Temporary

23.8.6. Debug Control

Name: DBGCTRL
Offset: 0x05
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
								DBGRUN
Access	R	R	R	R	R	R	R	R/W
Reset	0	0	0	0	0	0	0	0

Bit 0 – DBGRUN: Debug Run
Writing this bit to '1' will enable the RTC counter to run in Debug mode.

23.8.7. Clock Control

Name: CLKCTRL
Offset: 0x07
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
							CLKSEL[1:0]	
Access	R	R	R	R	R	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 1:0 – CLKSEL[1:0]: Clock Select
Writing these bits selects the source for the RTC clock (CLK_RTC).

Note: When configuring the RTC to use either XOSC32K or the external clock on TOSC1, the Source Select bit (SEL) in the XOSC32K Control A register of the Clock Controller (CLKCTRL_XOSC32KCTRLA) must be configured accordingly.

Value	Description
0x0	32KHz from OSCULP32K
0x1	1KHz from OSCULP32K
0x2	32.768kHz from XOSC32K
0x3	External clock from TOSC1 pin

23.8.8. Count

The RTC.CNTL and RTC.CNTH register pair represents the 16-bit value, RTC.CNT. The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + 0x01. For more details on reading and writing 16-bit registers, refer to Accessing 16-bit Registers.

Name: CNT
Offset: 0x08
Reset: 0x00
Property: -

Bit	15	14	13	12	11	10	9	8
	CNT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CNT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:8 – CNT[15:8]: Counter high byte
These bits hold the MSB of the 16-bit counter register.

System clock synchronized version of RTC counter register. When this register is written, the actual RTC counter register will be updated. Read out the CNTBSY flag in RTC.STATUS to see when the synchronization is complete.

Bits 7:0 – CNT[7:0]: Counter low byte
These bits hold the LSB of the 16-bit counter register.

System clock synchronized version of RTC counter register. When this register is written, the actual RTC counter register will be updated. Read out the CNTBSY flag in RTC.STATUS to see when the synchronization is complete.

23.8.9. Top

The RTC.PERL and RTC.PERH register pair represents the 16-bit value, RTC.PER. The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + 0x01. For more details on reading and writing 16-bit registers, refer to Accessing 16-bit Registers.

Name: PER
Offset: 0x0A
Reset: 0xFF
Property: -

Bit	15	14	13	12	11	10	9	8
	PER[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bit	7	6	5	4	3	2	1	0
	PER[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bits 15:8 – PER[15:8]: Period high byte
These bits hold the MSB of the 16-bit period register.

System clock synchronized version of RTC period register. When the counter register reaches this value, the Overflow flag (OVF) in RTC.INTFLAGS is set and the counter register is reset. The register can be written from software. Read out the Period Busy flag (PERBSY) in RTC.STATUS) to see when the synchronization is complete.

Bits 7:0 – PER[7:0]: Period low byte
These bits hold the LSB of the 16-bit period register.

System clock synchronized version of RTC period register. When the counter register reaches this value, the Overflow flag (OVF) in RTC.INTFLAGS is set and the counter register is reset. The register can be written from software. Read out the Period Busy flag (PERBSY) in RTC.STATUS) to see when the synchronization is complete.

23.8.10. Compare

The RTC.CMPL and RTC.CMPH register pair represents the 16-bit value, RTC.CMP. The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + 0x01. For more details on reading and writing 16-bit registers, refer to Accessing 16-bit Registers.

Name: CMP
Offset: 0x0C
Reset: 0x00
Property: -

Bit	15	14	13	12	11	10	9	8
	CMP[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CMP[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:8 – CMP[15:8]: Compare high byte
These bits hold the MSB of the 16-bit compare register.

System clock synchronized version of RTC compare register. When the counter register reaches this value, the Compare Match flag (CMP) in RTC.INTFLAGS is set. The register can be written from software. Read out the Compare Busy flag (CMPBSY) in RTC.STATUS to see when the synchronization is complete.

Bits 7:0 – CMP[7:0]: Compare low byte
These bits hold the LSB of the 16-bit compare register.

System clock synchronized version of RTC compare register. When the counter register reaches this value, the Compare Match flag (CMP) in RTC.INTFLAGS is set. The register can be written from software. Read out the Compare Busy flag (CMPBSY) in RTC.STATUS to see when the synchronization is complete.

23.8.11. Periodic Interrupt Timer Control A

Name: PITCTRLA
Offset: 0x10
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
		PERIOD[3:0]						PITEN
Access	R	R/W	R/W	R/W	R/W	R	R	R/W
Reset	0	0	0	0	0	0	0	0

Bits 6:3 – PERIOD[3:0]: Period
Writing this bit field selects the number of RTC clock cycles between each interrupt.

Value	Name	Description
0x0	OFF	No interrupt
0x1	CYC4	4 cycles
0x2	CYC8	8 cycles
0x3	CYC16	16 cycles
0x4	CYC32	32 cycles
0x5	CYC64	64 cycles
0x6	CYC128	128 cycles
0x7	CYC256	256 cycles
0x8	CYC512	512 cycles
0x9	CYC1024	1024 cycles
0xA	CYC2048	2048 cycles
0xB	CYC4096	4096 cycles
0xC	CYC8192	8192 cycles
0xD	CYC16384	16384 cycles
0xE	CYC32768	32768 cycles
0xF	-	Reserved

Bit 0 – PITEN: Periodic Interrupt Timer Enable
Writing a '1' to this bit enables the Periodic Interrupt Timer.

23.8.12. Periodic Interrupt Timer Status

Name: PITSTATUS
Offset: 0x11
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
								SYNCBUSY
Access		R	R	R	R	R	R	R
Reset		0	0	0	0	0	0	0

Bit 0 – SYNCBUSY: Synchronization Busy
This bit indicates whether the RTC is busy synchronizing RTC.PITCTRLA settings in the RTC clock domain.

23.8.13. PIT Interrupt Control

Name: PITINTCTRL
Offset: 0x12
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
								PI
Access		R	R	R	R	R	R	R/W
Reset		0	0	0	0	0	0	0

Bit 0 – PI: Periodic interrupt

Value	Description
0	The periodic interrupt is disabled
1	The periodic interrupt is enabled

23.8.14. PIT Interrupt Flag

Name: PITINTFLAGS
Offset: 0x13
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
								PI
Access		R	R	R	R	R	R	R
Reset		0	0	0	0	0	0	0

Bit 0 – PI: Periodic interrupt Flag
This flag is set when a periodic interrupt is issued.

Writing a '1' clears the flag.

23.8.15. Periodic Interrupt Timer Debug Control

Name: PITDBGCTRL
Offset: 0x15
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
								DBGRUN
Access	R	R	R	R	R	R	R	R/W
Reset	0	0	0	0	0	0	0	0

Bit 0 – DBGRUN: Debug Run
Writing this bit to '1' will enable the PIT to run in Debug mode.

24. USART - Universal Synchronous and Asynchronous Receiver and Transmitter

24.1. Overview

The Universal Synchronous and Asynchronous serial Receiver and Transmitter (USART) peripheral is a fast and flexible serial communication module. The USART supports full-duplex communication, asynchronous and synchronous operation and one-wire configurations. The USART can be set in SPI master mode and used for SPI communication.

Communication is frame based, and the frame format can be customized to support a wide range of standards. The USART is buffered in both directions, enabling continued data transmission without any delay between frames. Separate interrupts for receive and transmit complete enable fully interrupt driven communication. Frame error and buffer overflow are detected in hardware and indicated with separate status flags. Even or odd parity generation and parity check can also be enabled.

The main functional blocks are the clock generator, the transmitter, and the receiver:

The clock generator includes a fractional baud rate generator that is able to generate a wide range of USART baud rates from any system clock frequencies. This removes the need to use an external crystal oscillator with a specific frequency to achieve a required baud rate. It also supports external clock input in synchronous slave operation.

The transmitter consists of a single write buffer (DATA), a Shift Register and a parity generator. The write buffer allows continuous data transmission without any delay between frames.

The receiver consists of a two-level receive buffer (DATA) and a Shift Register. Data and clock recovery units ensure robust synchronization and noise filtering during asynchronous data reception. It includes frame error, buffer overflow, and parity error detection.

When the USART is set in one-wire mode, the transmitter and the receiver share the same RxD I/O pin.

When the USART is set in master SPI mode, all USART-specific logic is disabled, leaving the transmit and receive buffers, Shift registers, and baud rate generator enabled. Pin control and interrupt generation are identical in both modes. The registers are used in both modes, but their functionality differs for some control settings.

An IRCOM module can be enabled for one USART to support IrDA 1.4 physical compliant pulse modulation and demodulation for baud rates up to 115.2kbps.

The USART can be linked to the Configurable Custom Logic unit (CCL). When used with the CCL, the TxD/RxD data can be encoded/decoded before the signal is fed into the USART receiver or after the signal is output from transmitter when the USART is connected to CCL LUT outputs.

Related Links

CCL – Configurable Custom Logic on page 419

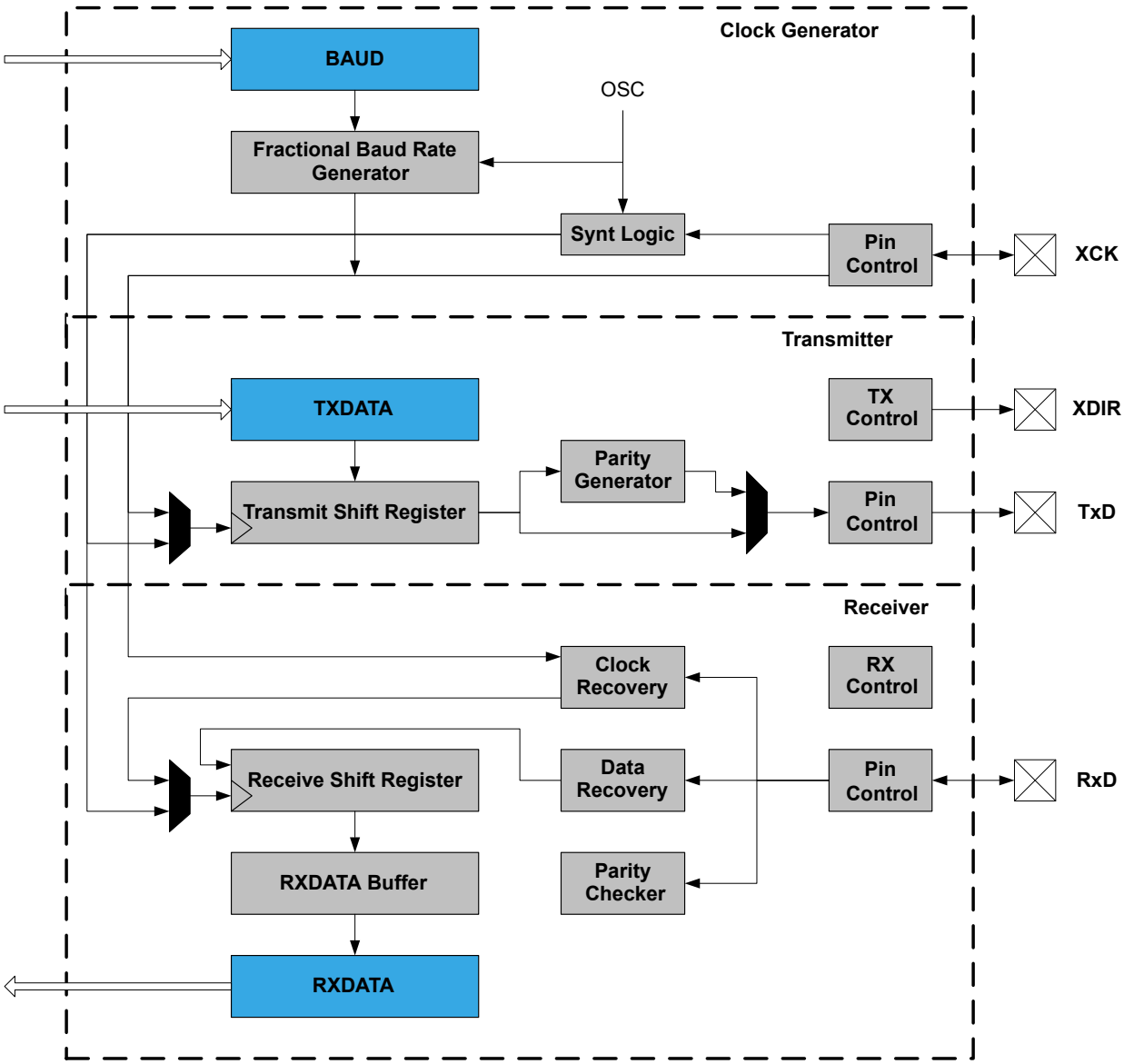
24.2. Features

- Full-duplex or one-wire half-duplex operation
- Asynchronous or synchronous operation
  - Synchronous clock rates up to 1/2 of the device clock frequency
  - Asynchronous clock rates up to 1/8 of the device clock frequency

- Supports serial frames with:
  - 5, 6, 7, 8, or 9 data bits
  - Optionally even and odd parity bits
  - 1 or 2 stop bits
- Fractional baud rate generator
  - Can generate desired baud rate from any system clock frequency
  - No need for external oscillator with certain frequencies
- Built-in error detection and correction schemes
  - Odd or even parity generation and parity check
  - Data overrun and framing error detection
  - Noise filtering includes false start bit detection and digital low-pass filter
- Separate interrupts for
  - Transmit complete
  - Transmit Data Register empty
  - Receive complete
- Multiprocessor communication mode
  - Addressing scheme to address a specific devices on a multi-device bus
  - Enable unaddressed devices to automatically ignore all frames
- Start Frame detection in UART mode
- Master SPI mode
  - Double buffered operation
  - Configurable data order
  - Operation up to 1/2 of the peripheral clock frequency
- IRCOM module for IrDA compliant pulse modulation/demodulation
- LIN slave support
  - Auto-baud and Break character detection
- RS-485 support

24.3. Block Diagram

Figure 24-1. USART Block Diagram



24.4. Signal Description

Signal	Type	Description
RxD	Input/output	Receiving line
TxD	output	Transmitting line
XCK	Input/output	Clock for synchronous operation
XDIR	Output	Transmit Enable for RS485

Related Links

I/O Multiplexing and Considerations on page 19

24.5. Product Dependencies

In order to use this peripheral, other parts of the system must be configured correctly, as described below.

Table 24-1. NVMCTRL Product Dependencies

Dependency	Applicable	Peripheral
Clocks	Yes	CLKCTRL
I/O Lines and Connections	Yes	PORT
Interrupts	Yes	CPUINT
Events	Yes	EVSYS
Debug	Yes	UPDI

Related Links

Clocks on page 86
I/O Lines and Connections on page 329
Interrupts on page 54
Events on page 181
Debug Operation on page 330

24.5.1. Clocks

This peripheral depends on the peripheral clock.

Related Links

CLKCTRL - Clock Controller on page 68
Product Dependencies on page 98

24.5.2. I/O Lines and Connections

Using the I/O lines of the peripheral requires configuration the I/O pins.

Related Links

PORT - I/O Pin Controller on page 132

24.5.3. Interrupts

Using the interrupts of this peripheral requires the Interrupt Controller to be configured first.

Related Links

CPUINT - CPU Interrupt Controller on page 97
SREG on page 51
Interrupts on page 136

24.5.4. Events

The events of this peripheral are connected to the Event System.

Related Links

EVSYS - Event System on page 109

24.5.5. Debug Operation

When the CPU is halted in debug mode, this peripheral will continue normal operation. If the peripheral is configured to require periodical service by the CPU through interrupts or similar, improper operation or data loss may result during debugging. This peripheral can be forced to halt operation during debugging.

Related Links

UPDI - Unified Program Debug Interface on page 490

24.6. Functional Description

24.6.1. Principle of Operation

The USART uses three communication lines for data transfer:

- RxD for receiving
- TxD for transmitting
- XCK for the transmission clock in synchronous operation

USART data transfer is frame based, where a serial frame consists of:

- 1 start bit
- 5, 6, 7, 8, or 9 data bits
- MSB or LSB first
- Parity bit: Even, odd, or none
- 1 or 2 stop bits

A frame starts with the start bit, followed by one character of data bits. If enabled, the parity bit is inserted between the data bits and the first stop bit. One frame can be directly followed by a new frame, or the communication line can return to the idle (high) state.

24.6.2. Basic Operation

24.6.2.1. Initialization

For setting the USART in full-duplex mode, the following initialization sequence is recommended:

1. Set the TxD pin value high, and optionally set the XCK pin low.
2. Set the TxD and optionally the XCK pin as output.
3. Set the baud rate and frame format.
4. Set the mode of operation (enables XCK pin output in synchronous mode).
5. Enable the transmitter or the receiver, depending on the usage.

For interrupt-driven USART operation, global interrupts should be disabled during the initialization.

Before doing a re-initialization with a changed baud rate or frame format, be sure that there are no ongoing transmissions while the registers are changed.

For setting the USART in one-wire mode, the following initialization sequence is recommended:

1. Set the TxD/RxD pin value high, and optionally set the XCK pin low.
2. Optionally, set the TxD/RxD input pin as Wired-AND or Wired-OR.
3. Set the TxD/RxD and optionally the XCK pin as output.
4. Set the baud rate and frame format.
5. Set the mode of operation (enables XCK pin output in synchronous mode).

6. Enable the transmitter or the receiver, depending on the usage.

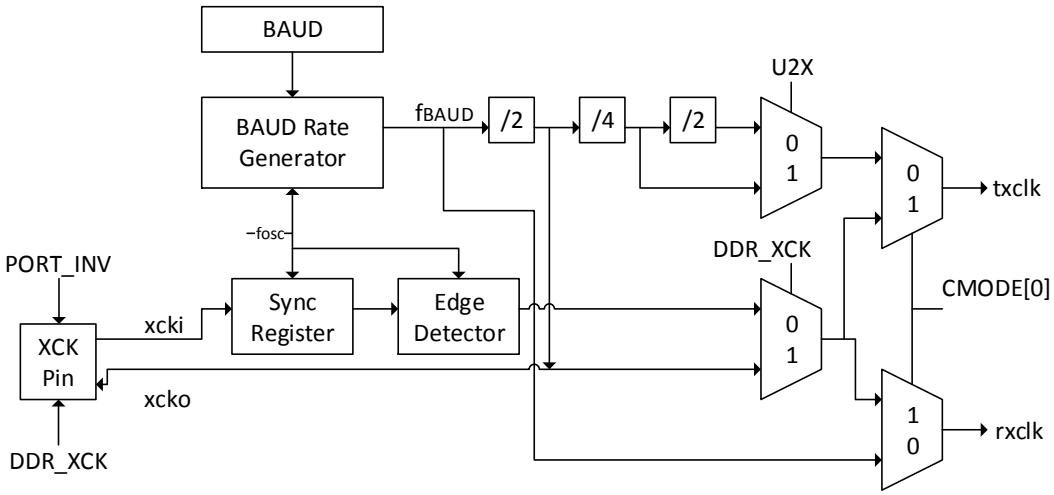
For interrupt-driven USART operation, global interrupts should be disabled during the initialization.

Before doing a re-initialization with a changed baud rate or frame format, be sure that there are no ongoing transmissions while the registers are changed.

24.6.2.2. Clock Generation

The clock used for baud rate generation and for shifting and sampling data bits is generated internally by the fractional baud rate generator or externally from the transfer clock (XCK) pin. Five modes of clock generation are supported: normal and double-speed asynchronous mode, master and slave synchronous mode, and master SPI mode.

Figure 24-2. Clock Generation Logic Block Diagram



Internal Clock Generation - The Fractional Baud Rate Generator

The baud rate generator is used for internal clock generation for asynchronous modes, synchronous master mode, and master SPI mode operation. The output frequency generated (f_{BAUD}) is determined by the Baud register value (USART_BAUD.BAUD) and the baud reference frequency (f_{REF}). The following table contains equations for calculating the baud rate (in bits per second) and for calculating the BAUD value for each mode of operation. It also shows the maximum baud rate versus peripheral clock frequency. For asynchronous operation, the BAUD register value is 16 bits (64 to 65535). The 10 MSBs (BAUD[15:6]) hold the integer part, while the 6 LSBs (BAUD[5:0]) hold the fractional part. In Synchronous mode, only the integer part of the BAUD register determine the baud rate.

Table 24-2. Equations for Calculating Baud Rate Register Setting

Operating Mode	Conditions	Baud Rate (Bits Per Seconds)	USART_BAUD.BAUD Register Value Calculation
Asynchronous	$f_{BAUD} \leq \frac{f_{REF}}{S}$	$f_{BAUD} = \frac{64 \times f_{REF}}{S \times BAUD}$	$BAUD = \frac{64 \times f_{REF}}{S \times f_{BAUD}}$
Synchronous	$f_{BAUD} \leq \frac{f_{REF}}{2}$	$f_{BAUD} = \frac{f_{REF}}{2 \times BAUD[15:6]}$	$BAUD[15:6] = \frac{f_{REF}}{2 \times f_{BAUD}}$

S is the number of samples per bit. Can be 16 or 8.

External Clock

External clock (XCK) is used in synchronous slave mode operation. The XCK clock input is sampled on the peripheral clock frequency (f_{PER}), and the maximum XCK clock frequency (f_{XCK}) is limited by the following:

$$f_{XCK} < \frac{f_{PER}}{4}$$

For each high and low period, XCK clock cycles must be sampled twice by the peripheral clock. If the XCK clock has jitter, or if the high/low period duty cycle is not 50/50, the maximum XCK clock speed must be reduced accordingly.

Double Speed Operation

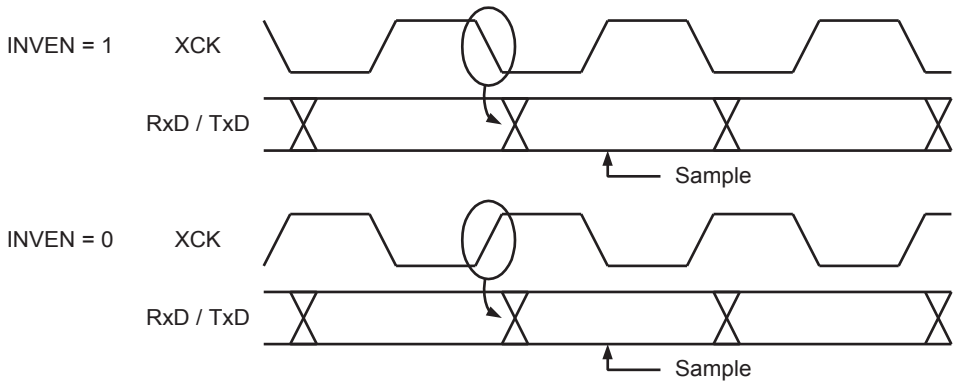
Double speed operation allows for higher baud rates under asynchronous operation with lower peripheral clock frequencies. This operation mode is enabled by writing the RXMODE bit in the Control B register (USART_CTRLB.RXMODE) to CLK2X.

When enabled, the baud rate for a given asynchronous baud rate setting shown in Table 24-2 will be doubled. In this mode, the receiver will use half the number of samples (reduced from 16 to 8) for data sampling and clock recovery. Due to the reduced sampling, a more accurate baud rate setting and peripheral clock are required. See Asynchronous Data Reception for more details.

Synchronous Clock Operation

When synchronous mode is used, the XCK pin controls whether the transmission clock is input (slave mode) or output (master mode). The corresponding port pin must be set to output for master mode or to input for slave mode. The normal port operation of the XCK pin will be overridden. The dependency between the clock edges and data sampling or data change is the same. Data input (on RxD) is sampled at the XCK clock edge which is opposite the edge where data output (TxD) is changed.

Figure 24-3. Synchronous Mode XCK Timing



The I/O pin can be inverted by writing a '1' to the Inverted I/O Enable bit in the Pin n Control register of the Port peripheral (PORT_PINnCTRL.INVEN). Using the inverted I/O setting for the corresponding XCK port pin, the XCK clock edges used for data sampling and data change can be selected. If inverted I/O is disabled (INVEN=0), data will be changed at the rising XCK clock edge and sampled at the falling XCK clock edge. If inverted I/O is enabled (INVEN=1), data will be changed at the falling XCK clock edge and sampled at the rising XCK clock edge.

Master SPI Mode Clock Generation

For master SPI mode operation, only internal clock generation is supported. This is identical to the USART synchronous master mode, and the baud rate or BAUD setting is calculated using the same equations (see Table 24-2).

There are four combinations of the SPI clock (SCK) phase and polarity with respect to the serial data, and these are determined by the Clock Phase bit in the Control C register (USART_CTRLC.UCPHA) and the

Inverted I/O Enable bit in the Pin n Control register of the Port peripheral (PORT_PINnCTRL.INVEN). The data transfer timing diagrams are shown below.

Data bits are shifted out and latched in on opposite edges of the XCK signal, ensuring sufficient time for data signals to stabilize.

Note: Check if these bits still are available

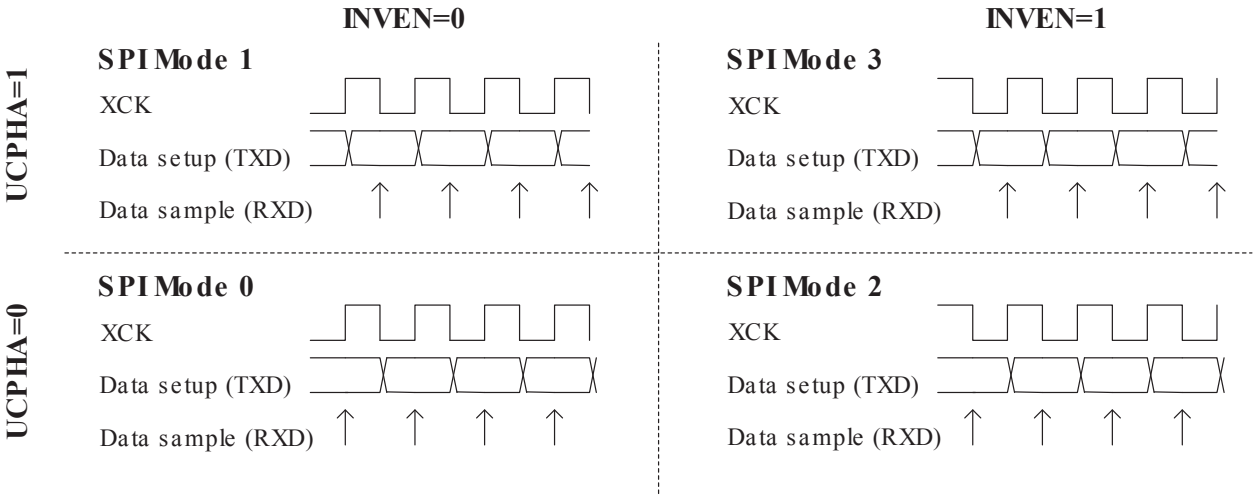
settings are summarized in the following Table. Changing the setting of any of these bits during transmission will corrupt both the receiver and transmitter.

Table 24-3. PORT_PINnCTRL.INVEN and USART_CTRLC.UCPHA Functionality

SPI mode	INVEN	UCPHA	Leading edge	Trailing edge
0	0	0	Rising, sample	Falling, setup
1	0	1	Rising, setup	Falling, sample
2	1	0	Falling, sample	Rising, setup
3	1	1	Falling, setup	Rising, sample

The leading edge is the first clock edge of a clock cycle. The trailing edge is the last clock edge of a clock cycle.

Figure 24-4. UCPHA and INVEN Data Transfer Timing Diagrams



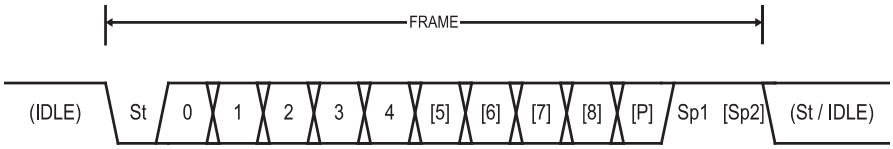
24.6.2.3. Frame Formats

Data transfer is frame based, where a serial frame consists of one character of data bits with synchronization bits (start and stop bits) and an optional parity bit for error checking. Note that this does not apply to master SPI operation (See SPI Frame Formats). The USART accepts all combinations of the following as valid frame formats:

- 1 start bit
- 5, 6, 7, 8 or 9 data bits
- no, even, or odd parity bit
- 1 or 2 stop bits

A frame starts with the start bit, followed by all the data bits (least-significant bit first and most significant bit last). If enabled, the parity bit is inserted after the data bits, before the first stop bit. One frame can be directly followed by a start bit and a new frame, or the communication line can return to the idle (high) state. Figure 24-5 illustrates the possible combinations of frame formats. Bits inside brackets are optional.

Figure 24-5. Frame Formats



St Start bit, always low.

(n) Data bits (0 to 8 in standard mode, variable when controlled by PEC).

P Parity bit, may be odd or even.

Sp Stop bit, always high.

IDLE No transfer on the communication line (RxD or TxD). The IDLE state is always high.

Parity

Even or odd parity can be selected for error checking by writing the Parity Mode bits in the Control C register (USART_CTRLC.PMODE). If even parity is selected, the parity bit is set to '1' if the number of logical one data bits is odd (making the total number of logical ones even). If odd parity is selected, the parity bit is set to '1' if the number of logical one data bits is even (making the total number of ones odd).

When enabled, the parity checker calculates the parity of the data bits in incoming frames and compares the result with the parity bit of the corresponding frame. If a parity error is detected, the parity error flag is set.

When variable data length mode is enabled, the parity checker is not supported.

SPI Frame Formats

The serial frame in SPI mode is defined to be one character of eight data bits. The USART in master SPI mode has two valid frame formats:

- 8-bit data, msb first
- 8-bit data, lsb first

The data order is selected by writing to the Data Order bit in the Control C register (USART_CTRLC.UDORD).

After a complete frame is transmitted, a new frame can directly follow it, or the communication line can return to the idle (high) state.

24.6.2.4. Data Transmission - USART Transmitter

When the transmitter has been enabled, the normal Port operation of the TxD pin is overridden by the USART and given the function as the transmitter's serial output. The direction of the pin n must be configured as output by writing the Direction register for the corresponding port (PORT_DIR.DIR[n]). If the USART is configured for one-wire operation, the USART will automatically override the RxD/TxD pin to output, when the transmitter is enabled.

Related Links

PORTMUX - Port Multiplexer on page 126
PORT - I/O Pin Controller on page 132

Sending Frames

A data transmission is initiated by loading the Transmit buffer (USART_TXDATA.DATA) with the data to be sent. The data in the transmit buffer are moved to the Shift Register when the Shift Register is empty and ready to send a new frame. The Shift Register is loaded if it is in idle state (no ongoing transmission) or immediately after the last stop bit of the previous frame is transmitted. When the Shift Register is loaded with data, it will transfer one complete frame.

When the data frame length is controlled by the peripheral counter in the XCL unit, the XCL and event controlled mode in USART have to be initiated before enabling the transmitter. In variable data length mode, the minimum frame length is one bit, the maximum is 256 bits. The maximum size must be chosen according to the oscillator accuracy.

A transmission is initiated by loading the TXDATA with the data to be sent. When the first data is loaded in the Shift Register, the USART restarts the peripheral counter. Each bit shift will decrement the peripheral counter. A compare match is provided by the XCL when the internal counter value reaches BOTTOM (zero). While the compare match is not received, the USART continues to shift out the data bits. If the compare match occurs before completing an 8-bit data shift, the USART changes its state to stop bits. If the Shift Register is empty before the compare match is received, then new data is automatically loaded in the Shift Register and transmission continues. If there is no more data to transmit and the compare match is not received, the transmission is aborted and Data Register Empty Flag (USART_STATUS.DREIF) is generated. The USART returns to IDLE state and stops any event generation for peripheral counter. The user can then calculate the number of bits already sent over the line.

When the entire frame in the Shift Register has been shifted out and there are no new data present in the transmit buffer, the Transmit Complete Interrupt Flag (USART_STATUS.TXCIF) is set and the optional interrupt is generated

TXDATA can only be written when the Data Register Empty Flag (DREIF) is set, indicating that the register is empty and ready for new data.

When using frames with fewer than eight bits, the most-significant bits written to TXDATA are ignored. If 9-bit characters are used, USART_TXDATAH.DATA[8] has to be written before USART_TXDATAL.DATA[7:0].

Disabling the Transmitter

A disabling of the transmitter will not become effective until ongoing and pending transmissions are completed; i.e., when the Transmit Shift Register and Transmit Buffer Register do not contain data to be transmitted, and optionally, while the compare match is not received from peripheral counter. In this case, it is possible to write 0x00 to the peripheral Counter Register to generate the compare match. When the transmitter is disabled, it will no longer override the TxDn pin, and the pin direction is set as input automatically by hardware, even if it was configured as output by the user.

24.6.2.5. Data Reception - USART Receiver

When the receiver is enabled, the RxD pin functions as the receiver's serial input. The direction of the pin n must be set as input in the Direction register of the Port (PORT_DIR.DIR[n]=0), which is the default pin setting.

Receiving Frames

The receiver starts data reception when it detects a valid start bit. Each bit that follows the start bit will be sampled at the baud rate or XCK clock, and shifted into the Receive Shift Register until the first stop bit of a frame is received. A second stop bit will be ignored by the receiver. When the first stop bit is received and a complete serial frame is present in the Receive Shift Register, the contents of the Shift Register will be moved into the receive buffer. The receive complete interrupt flag (USART_STATUS.RXCIF) is set, and the optional interrupt is generated.

When the data frame length is controlled by the peripheral counter in the CCL unit, the CCL and event controlled mode in USART have to be initiated before enabling the receiver. When a start bit is detected, the USART sends the restart command to the peripheral counter.

Each bit shift will decrement the peripheral counter. A compare match is provided by the CCL when the internal counter value reaches BOTTOM (zero). While the compare match is not received, the USART continues to shift in the data bits. If the compare match occurs before completing an 8-bit data shifts, the

USART changes its state to stop bits. After each 8-bit data reception, data receive flag (DRIF) and optionally an interrupt, is generated. If the data buffer overflow condition is generated, the reception is aborted and buffer overflow flag is set. No more counter commands are generated for the peripheral counter while a new start bit condition is not detected. In such error condition, it is highly recommended to disable the receiver part, unless any falling edge will be considered as a valid start bit and the peripheral counter can automatically restart its operation. Data receive interrupt flag and receive complete interrupt flag share the same interrupt line and interrupt settings.

The receiver buffer can be read by reading RXDATA, comprising of USART_RXDATAL.DATA[7:0] and USART_RXDATAH.DATA[8]. RXDATA should not be read unless the Receive Complete Interrupt Flag (USART_STATUS.RXCIF) is set. When using frames with fewer than eight bits, the unused most-significant bits are read as zero. If 9-bit characters are used, the ninth bit (USART_RXDATAH.DATA[8]) must be read before the low byte (USART_RXDATAL.DATA[7.0]).

Receiver Error Flags

The USART receiver has three error flags in the Receiver Data Register High Byte register (USART_RXDATAH):

- Frame Error (FERR)
- Buffer overflow (BUFOVF)
- Parity error (PERR)

The error flags are located in the receive FIFO buffer together with their corresponding frame. Due to the buffering of the error flags, the USART_RXDATAH must be read before the USART_RXDATAL, since reading the USART_RXDATAL changes the FIFO buffer.

Parity Checker

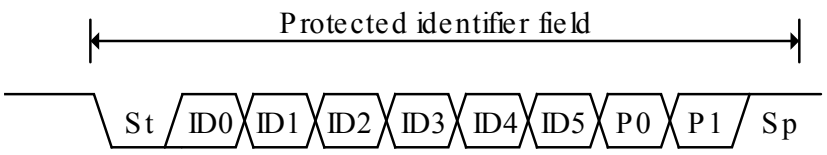
When enabled, the parity checker calculates the parity of the data bits in incoming frames and compares the result with the parity bit of the corresponding frame. If a parity error is detected, the Parity Error flag (USART_RXDATAH.PERR) is set.

When variable data length mode is enabled, the parity checker is not supported.

If USART LIN mode is enabled (USART_CTRLB.RXMODE), a parity check is only performed on the protected identifier field. An parity error is detected if one of the equations below is not true which sets USART_RXDATAH.PERR.

$$P0 = ID0 \oplus ID1 \oplus ID2 \oplus ID4$$
$$P1 = \neg(ID1 \oplus ID3 \oplus ID4 \oplus ID5)$$

Figure 24-6. Protected identifier field and mapping of identifier and parity bits



Disabling the Receiver

A disabling of the receiver will be immediate. The receiver buffer will be flushed, and data from ongoing receptions will be lost.

Flushing the Receive Buffer

If the receive buffer has to be flushed during normal operation, read the DATA location (USART_RXDATAL and USART_RXDATAH registers) until the Receive Complete Interrupt Flag (USART_RXDATAH.RXCIF) is cleared.

Asynchronous Data Reception

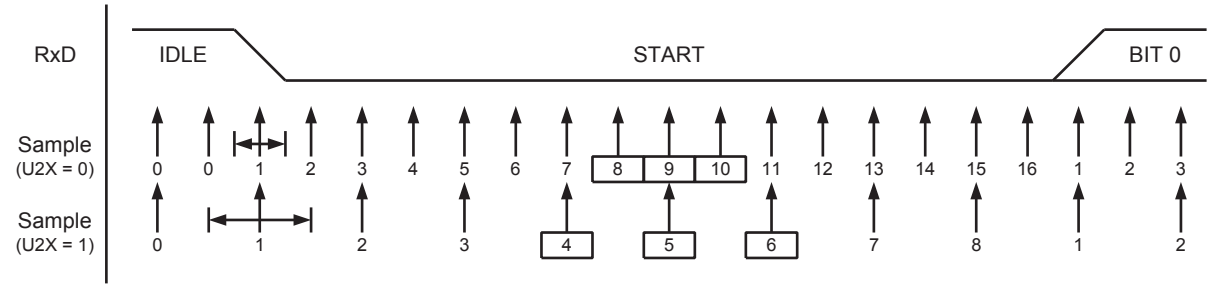
The USART includes a clock recovery and a data recovery unit for handling asynchronous data reception.

The clock recovery unit is used for synchronizing the incoming asynchronous serial frames at the RxD pin to the internally generated baud rate clock. It samples and low-pass filters each incoming bit, thereby improving the noise immunity of the receiver. The asynchronous reception operational range depends on the accuracy of the internal baud rate clock, the rate of the incoming frames, and the frame size in number of bits.

Asynchronous Clock Recovery

The clock recovery unit synchronizes the internal clock to the incoming serial frames. Figure 24-7 illustrates the sampling process for the start bit of an incoming frame. The sample rate is 16 times the baud rate for normal mode, and eight times the baud rate for double speed mode. The horizontal arrows illustrate the synchronization variation due to the sampling process. Note the larger time variation when using the double speed mode of operation. Samples denoted as zero are samples done when the RxD line is idle; i.e., when there is no communication activity.

Figure 24-7. Start Bit Sampling

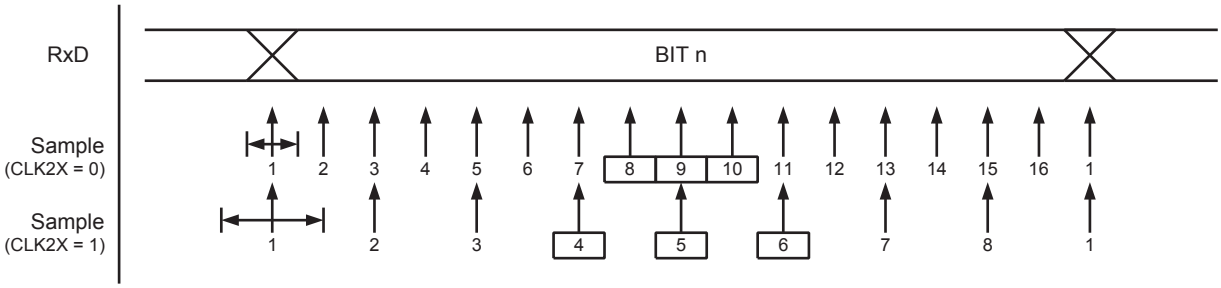


When the clock recovery logic detects a high (idle) to low (start) transition on the RxD line, the start bit detection sequence is initiated. Sample 1 denotes the first zero-sample, as shown in the figure. The clock recovery logic then uses samples 8, 9, and 10 for normal mode and samples 4, 5, and 6 for double speed mode to decide if a valid start bit is received. If two or three samples have a low level, the start bit is accepted. The clock recovery unit is synchronized, and the data recovery can begin. If two or three samples have a high level, the start bit is rejected as a noise spike, and the receiver looks for the next high-to-low transition. The process is repeated for each start bit.

Asynchronous Data Recovery

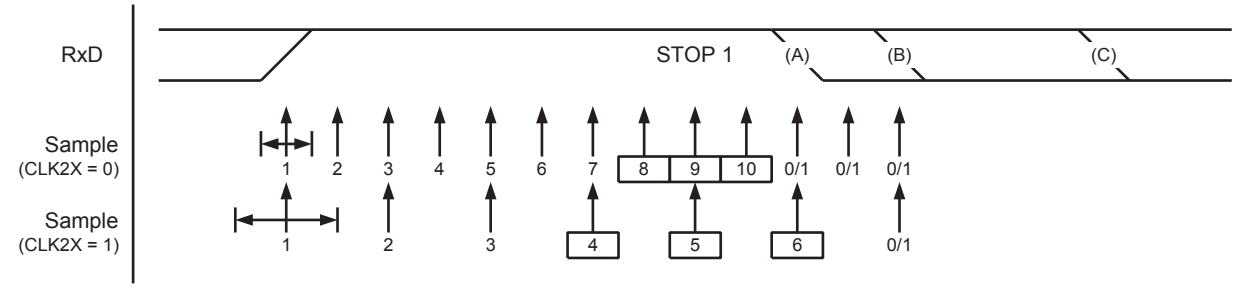
The data recovery unit uses sixteen samples in normal mode and eight samples in double speed mode for each bit. The following Figure shows the sampling process of data and parity bits.

Figure 24-8. Sampling of Data and Parity Bits



As for start bit detection, an identical majority voting technique is used on the three center samples for deciding of the logic level of the received bit. The process is repeated for each bit until a complete frame is received. It includes the first stop bit, but excludes additional ones. If the sampled stop bit is a '0' value, the Frame Error (USART_RXDATAH.FERR) flag will be set. The next Figure shows the sampling of the stop bit in relation to the earliest possible beginning of the next frame's start bit.

Figure 24-9. Stop Bit and Next Start Bit Sampling



A new high-to-low transition indicating the start bit of a new frame can come right after the last of the bits used for majority voting. For normal speed mode, the first low level sample can be at the point marked (A) in Stop Bit Sampling and Next Start Bit Sampling. For double speed mode, the first low level must be delayed to point (B). Point (C) marks a stop bit of full length at nominal baud rate. The early start bit detection influences the operational range of the receiver.

Asynchronous Operational Range

The operational range of the receiver is dependent on the mismatch between the received bit rate and the internally generated baud rate. If an external transmitter is sending using bit rates that are too fast or too slow, or if the internally generated baud rate of the receiver does not match the external source's base frequency, the receiver will not be able to synchronize the frames to the start bit.

The following equations can be used to calculate the ratio of the incoming data rate and internal receiver baud rate.

$R_{SLOW} = \frac{16(D+1)}{16(D+1)+6}$	$R_{FAST} = \frac{16(D+2)}{16(D+1)+8}$

D Sum of character size and parity size (D = 5 to 10 bit).

R_{slow} Is the ratio of the slowest incoming data rate that can be accepted in relation to the receiver baud rate.

R_{fast} Is the ratio of the fastest incoming data rate that can be accepted in relation to the receiver baud rate.

Table 24-4 and Table 24-5 list the maximum receiver baud rate error that can be tolerated. Normal Speed mode has higher toleration of baud rate variations.

Table 24-4. Recommended Maximum Receiver Baud Rate Error for Normal Speed Mode (CLK2X = 0)

D #(Data + Parity bit)	R_{slow} [%]	R_{fast} [%]	Maximum total error [%]	Receiver max. receiver error [%]
5	93.20	106.67	+6.67/-6.80	±3.0
6	94.12	105.79	+5.79/-5.88	±2.5
7	94.81	105.11	+5.11/-5.19	±2.0
8	95.36	104.58	+4.58/-4.54	±2.0
9	95.81	104.14	+4.14/-4.19	±1.5
10	96.17	103.78	+3.78/-3.83	±1.5

Table 24-5. Recommended Maximum Receiver Baud Rate Error for Double Speed Mode (CLK2X = 1)

D #(Data + Parity bit)	R_{slow} [%]	R_{fast} [%]	Maximum total error [%]	Receiver max. receiver error [%]
5	94.12	105.66	+5.66/-5.88	±2.5
6	94.92	104.92	+4.92/-5.08	±2.0
7	95.52	104.35	+4.35/-4.48	±1.5
8	96.00	103.90	+3.90/-4.00	±1.5
9	96.39	103.53	+3.53/-3.61	±1.5
10	96.70	103.23	+3.23/-3.30	±1.0

The recommendations of the maximum receiver baud rate error were made under the assumption that the Receiver and Transmitter equally divide the maximum total error.

24.6.3. Additional Features

24.6.3.1. USART in Master SPI mode

Using the USART in master SPI mode requires the transmitter to be enabled. The receiver can optionally be enabled to serve as the serial input. The XCK pin will be used as the transfer clock.

As for the USART, a data transfer is initiated by writing to the DATA Register. This is the case for both sending and receiving data, since the transmitter controls the transfer clock. The data written to DATA are moved from the transmit buffer to the Shift Register when the Shift Register is ready to send a new frame.

The transmitter and receiver interrupt flags and corresponding USART interrupts used in master SPI mode are identical in function to their use in normal USART operation. The receiver error status flags are not in use and are always read as zero.

Disabling of the USART transmitter or receiver in master SPI mode is identical to their disabling in normal USART operation.

USART SPI vs. SPI

The USART in master SPI mode is fully compatible with the standalone SPI module in that:

- Timing diagrams are the same
- UCPHA bit functionality is identical to that of the SPI CPHA bit
- UDORD bit functionality is identical to that of the SPI DORD bit

When the USART is set in master SPI mode, configuration and use are in some cases different from those of the standalone SPI module. In addition, the following difference exists:

- The USART in master SPI mode does not include the SPI (Write Collision) feature

The USART in master SPI mode does not include the SPI double speed mode feature, but this can be achieved by configuring the baud rate generator accordingly:

- Interrupt timing is not compatible
- Pin control differs due to the master-only operation of the USART in SPI master mode

A comparison of the USART in master SPI mode and the SPI pins is shown in Table 24-6.

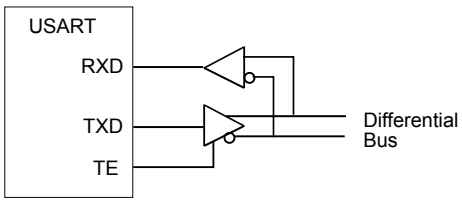
Table 24-6. Prescaler Options

USART	SPI	Comment
TxD	MOSI	Master out only
RxD	MISO	Master in only
XCK	SCK	Functionally identical
N/A	SS	Not supported by USART in master SPI mode

24.6.3.2. RS485 Mode of Operation

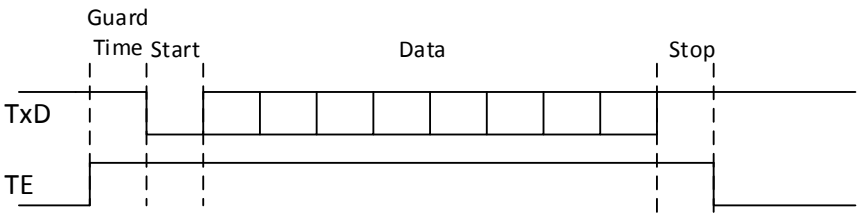
The RS485 feature enables control of either an external line driver (CTRLA.RS485=1) as shown in the figure below, or control of the transmitter driving the TxD pin (CTRLA.RS485=2). While operating in RS485 mode, the transmit enable pin (TE) is driven high when the transmitter is active.

Figure 24-10. RS485 Bus Connection



The TE pin go high one baud clock cycle in advance of data being shifted out to allow some guard time to enable the external line driver. The TE pin will remain high for the complete frame including stop bit(s).

Figure 24-11. TE Drive Timing



24.6.3.3. Start Frame Detection

The start frame detection is supported in UART mode only and takes place only if the system is in deeper sleep modes. The UART start frame detector can wake up the system from power save, standby or extended standby sleep modes when a start bit is detected. In power save mode, the internal 8MHz oscillator in low power mode must be used as clock source, but in standby or extended standby modes it can operate with any clock source.

When a high-to-low transition is detected on RxDn, the oscillator is powered up and the UART clock is enabled. After start-up, the rest of the data frame can be received, provided that the baud rate is slow enough in relation to the oscillator start-up time. Start-up time of the oscillators varies with supply voltage and temperature. For details on oscillator start-up time characteristics, refer to device datasheet.

If a false start bit is detected and if the system has not been waken-up by another source, the oscillator will be automatically powered-off and the UART waits for the next transition.

The UART start frame detection works in asynchronous mode only. It is enabled by writing the Start Frame Detection bit (SFDEN) in “CTRLB – Control Register B” on page 298”. If the start bit is detected, the UART Start Interrupt Flag (RXSIF) bit is set.

In active and idle sleep modes, the asynchronous detection is automatically disabled. In power down sleep mode, the asynchronous detector is enabled, but the internal oscillator is never powered. In such case, it is highly recommended to disable the start detector before going in power down sleep mode.

The UART receive complete flag and UART start interrupt flag share the same interrupt line, but each has its dedicated interrupt settings. The Table 21-5 shows the USART start frame detection modes, depending of interrupt setting.

Table 24-7. USART Start Frame Detection Modes

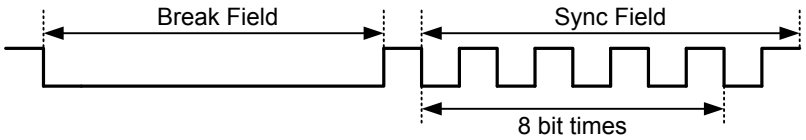
SFDEN	RXSIF interrupt	RXCIF interrupt	Comment
0	x	x	Standard mode
1	Disabled	Disabled	Only the oscillator is powered during the frame reception. If the interrupts are disabled and buffer overflow is ignored, all incoming frames will be lost
1 (1)	Disabled	Enabled	System/all clocks waked-up on Receive Complete interrupt
1 (1)	Enabled	x	System/all clocks waked-up on UART Start Detection

Note: 1. The SLEEP instruction will not shut down the oscillator if on going communication.

24.6.3.4. Break Character Detection and Auto-baud

When USART receive mode is set to LINAUTO mode (RXMODE), it follows the LIN format. All LIN Frames start with a Break Field followed by a Sync Field. The USART uses a break detection threshold of greater than 11 nominal bit times at the configured baud rate. At any time, if more than 11 consecutive dominant bits are detected on the bus, the USART detects a Break Field. When a Break Field has been detected, the USART expects the Sync Field character to be 0x55. This field is used to update the actual baud rate in order to stay synchronized. If the received Sync character is not 0x55, then the Inconsistent Sync Field error flag (STATUS.ISFIF) is set and the baud rate is unchanged.

Figure 24-12. LIN Break and Sync Fields



After a break field is detected and the start bit of the Sync Field is detected, a counter is started. The counter is then incremented for the next 8 bit times of the Sync Field. At the end of these 8 bit times, the counter is stopped. At this moment, the 10 most significant bits of the counter (value divided by 64) gives the new clock divider and the 6 least significant bits of this value (the remainder) gives the new fractional part. When the Sync Field has been received and all bits found valid, the clock divider and the fractional part are updated in the Baud Rate Generator register (BAUD). After the Break and Sync Fields, n characters of data can be received.

When the USART receive mode is set to GENAUTO mode, a generic auto-baud mode is enabled. In this mode there no check of Sync character to equal 0x55. After detection of a Break Field the USART expects the next character to be a Sync field, counting 8 low and high bit times. If the measured Sync field result in a valid BAUD value (0x0064-0xffff), the BAUD register is updated. Setting the Wait for Break bit (USART_STATUS.WFB) before receiving the next Break character, the next negative plus positive edge of RxD line is detected as a Break. This makes it possible to set an arbitrary new Baud Rate without knowing the current Baud Rate.

24.6.3.5. One-wire Mode

In this mode the TxD pin is connected to the RxD pin internally. If the receiver is enabled when transmitting it will receive what the transmitter is sending. This can be used to check that no one else is trying to transmit since received data will not be the same as the transmitted data.

24.6.3.6. Multiprocessor Communication Mode

The multiprocessor communication mode effectively reduces the number of incoming frames that have to be handled by the receiver in a system with multiple microcontrollers communicating via the same serial bus. In this mode, a dedicated bit in the frames is used to indicate whether the frame is an address or data frame type.

If the receiver is set up to receive frames that contain five to eight data bits, the first stop bit is used to indicate the frame type. If the receiver is set up for frames with nine data bits, the ninth bit is used. When the frame type bit is one, the frame contains an address. When the frame type bit is zero, the frame is a data frame. If 5-bit to 8-bit character frames are used, the transmitter must be set to use two stop bits, since the first stop bit is used for indicating the frame type.

If a particular slave MCU has been addressed, it will receive the following data frames as usual, while the other slave MCUs will ignore the frames until another address frame is received.

Using Multiprocessor Communication Mode

The following procedure should be used to exchange data in multiprocessor communication mode (MPCM):

1. All slave MCUs are in multiprocessor communication mode.
2. The master MCU sends an address frame, and all slaves receive and read this frame.
3. Each slave MCU determines if it has been selected.
4. The addressed MCU will disable MPCM and receive all data frames. The other slave MCUs will ignore the data frames.
5. When the addressed MCU has received the last data frame, it must enable MPCM again and wait for a new address frame from the master.

The process then repeats from step 2.

Using any of the 5- to 8-bit character frame formats is impractical, as the receiver must change between using n and n+1 character frame formats. This makes full-duplex operation difficult, since the transmitter and receiver must use the same character size setting.

24.6.3.7. IRCOM Mode of Operation

IRCOM mode can be enabled to use the IRCOM module with the USART. This enables IrDA 1.4 compliant modulation and demodulation for baud rates up to 115.2kbps. When IRCOM mode is enabled, double speed mode cannot be used for the USART. For devices with more than one USART, IRCOM mode can be enabled for only one USART at a time, which is not linked with the CCL peripheral. For details, refer to IRCOM – IR Communication Module.

24.6.4. Events

The USART can accept the following input Events:

- IREI - IrDA Event Input

The Event is enabled by writing a '1' to the IrDA Event Input bit in the Event Control register (USART_EVCTRL.IREI)

Related Links

EVSYS - Event System on page 109

EVCTRL on page 360

24.6.5. Interrupts

Table 24-8. Available Interrupt Vectors and Sources

Offset	Name	Vector Description	Conditions
0x00	RXC	Receive Complete interrupt	There are unread data in the receive buffer.
0x02	DRE	Data Register Empty interrupt	The transmit buffer is empty/ready to receive new data
0x04	TXC	Transmit Complete interrupt	The entire frame in the Transmit Shift Register has been shifted out and there are no new data in the transmit buffer.

When an interrupt condition occurs, the corresponding Interrupt Flag is set in the Status register (USART_STATUS).

An interrupt source is enabled or disabled by writing to the corresponding bit in the Control A register (USART_CTRLA).

An interrupt request is generated when the corresponding interrupt source is enabled and the Interrupt Flag is set. The interrupt request remains active until the Interrupt Flag is cleared. See the USART_STATUS register for details on how to clear Interrupt Flags.

Note: Interrupt requests are only generated when Interrupts are enabled globally.

Related Links

CPUINT - CPU Interrupt Controller on page 97
STATUS on page 349
CTRLA on page 351

24.6.6. Configuration Change Protection

Not applicable.

24.7. Register Summary

Offset	Name	Bit Pos.								
0x00	RXDATAL	7:0	DATA[7:0]							
0x01	RXDATAH	7:0	RXCIF	BUFOVF				FERR	PERR	DATA[8]
0x02	TXDATAL	7:0	DATA[7:0]							
0x03	TXDATAH	7:0								DATA[8]
0x04	STATUS	7:0	RXCIF	TXCIF	DREIF	RXSIF	ISFIF		BDF	WFB
0x05	CTRLA	7:0	RXCIE	TXCIE	DREIE	RXSIE	LBME	ABEIE	RS485[1:0]	
0x06	CTRLB	7:0	RXEN	TXEN		SFDEN	ODME	RXMODE[1:0]		MPCM
0x07	CTRLC	7:0	CMODE[1:0]		PMODE[1:0]		SBMODE	CHSIZE[2:0]		
0x08	BAUD	7:0	BAUD[7:0]							
0x09		15:8	BAUD[15:8]							
0x0A	Reserved									
0x0B	DBGCTRL	7:0								DBGRUN
0x0C	EVCTRL	7:0								IREI
0x0D	TXPLCTRL	7:0	TXPL[7:0]							
0x0E	RXPLCTRL	7:0		RXPL[6:0]						

24.8. Register Description

24.8.1. Receiver Data Register Low Byte

Reading the RXDATAL Register location will return the contents of the Receive Data Buffer Register (RXB).

The receive buffer consists of a two level FIFO. The FIFO and the corresponding flags in the high byte of RXDATA will change state whenever the receive buffer is accessed (read). If USART_CTRLC.CHSIZE is set to 9BIT Low byte first, read RXDATAL before RXDATAH, otherwise always read RXDATAH before RXDATAL in order to get the correct flags.

Name: RXDATAL
Offset: 0x00
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	DATA[7:0]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – DATA[7:0]: Receiver Data Register

24.8.2. Receiver Data Register High Byte

Reading the RXDATAH Register location will return the contents of the ninth DATA bit plus status bits. The receive buffer consists of a two level FIFO. The FIFO and the corresponding flags in the high byte of RXDATAH will change state whenever the receive buffer is accessed (read). If USART_CTRLC.CHSIZE is set to 9BIT Low byte first, read RXDATAL before RXDATAH, otherwise always read RXDATAH before RXDATAL in order to get the correct flags.

Name: RXDATAH
Offset: 0x01
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	RXCIF	BUFOVF				FERR	PERR	DATA[8]
Access	R	R				R	R	R
Reset	0	0				0	0	0

Bit 7 – RXCIF: USART Receive Complete Interrupt Flag
This flag is set when there are unread data in the receive buffer and cleared when the receive buffer is empty (i.e., does not contain any unread data). When the Receiver is disabled, the receive buffer will be flushed and consequently the RXCIF will become zero.

Bit 6 – BUFOVF: Buffer Overflow
The BUFOVF flag indicates data loss due to a receiver buffer full condition. This flag is set if a Buffer Overflow condition is detected. A Buffer Overflow occurs when the receive buffer is full (two characters), it is a new character waiting in the Receive Shift Register, and a new start bit is detected. This flag is valid until the receive buffer (RXDATAL) is read.
This flag is not used in Master SPI mode of operation.

Bit 2 – FERR: Frame Error
The FERR flag indicates the state of the first stop bit of the next readable frame stored in the receive buffer. The bit is set if the received character had a Frame Error, i.e. when the first stop bit was zero, and cleared when the stop bit of the received data is '1'. This bit is valid until the receive buffer (RXDATAL) is read. The FERR is not affected by the SBMODE bit in USART.CTRLC since the Receiver ignores all, except for the first stop bit.
This flag is not used in Master SPI mode of operation.

Bit 1 – PERR: Parity Error
If parity checking is enabled and the next character in the receive buffer has a Parity Error this flag is set. If Parity Check is not enabled the PERR will always be read as zero. This bit is valid until the receive buffer (RXDATAL) is read. For details on parity calculation refer to Parity . If USART is set to LINAUTO mode, this bit will be a Parity Check of the Protected identifier field and will be valid when DATA[8] in USART.RXDATAH reads low.
This flag is not used in Master SPI mode of operation.

Bit 0 – DATA[8]: Receiver Data Register
When USART receiver is set to LINAUTO mode, this bit indicates if the received data is within the response space of a LIN frame. If the received data is the Protected identifier field, this bit will be read as zero. Otherwise the bit will be read as one. For receiver mode other than LINAUTO mode, DATA[8] holds the ninth data bit in the received character when operating with serial frames with nine data bits.

24.8.3. Transmit Data Register Low Byte

The Transmit Data Buffer Register (TXB) will be the destination for data written to the USART.TXDATAL Register location.
For 5-, 6-, or 7-bit characters the upper unused bits will be ignored by the Transmitter and set to zero by the Receiver.

The transmit buffer can only be written when the DREIF Flag in the USART.STATUS Register is set. Data written to DATA when the DREIF Flag is not set, will be ignored by the USART Transmitter. When data is written to the transmit buffer, and the Transmitter is enabled, the Transmitter will load the data into the Transmit Shift Register when the Shift Register is empty. The data is then transmitted on the TxD pin.

Name: TXDATAL
Offset: 0x02
Reset: 0x00
Property: W

Bit	7	6	5	4	3	2	1	0
	DATA[7:0]							
Access	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – DATA[7:0]: Transmit Data Register

24.8.4. Transmit Data Register High Byte

USART.TXDATAH holds the ninth data bit in the character to be transmitted when operating with serial frames with nine data bits. When used this bit must be written before writing to TXDATAL except if USART_CTRLC.CHSIZE is set to 9BIT Low byte first where TXDATAL should be written first.
This bit is unused in Master SPI mode of operation.

Name: TXDATAH
Offset: 0x03
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
								DATA[8]
Access								W
Reset								0

Bit 0 – DATA[8]: Transmit Data Register
This bit is used when CHSIZE=9bit.

24.8.5. USART Status Register

Name: STATUS
Offset: 0x04
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	RXCIF	TXCIF	DREIF	RXSIF	ISFIF		BDF	WFB
Access	R/W	R/W	R/W	R/W	R/W		R/W	R/W
Reset	0	0	0	0	0		0	0

Bit 7 – RXCIF: USART Receive Complete Interrupt Flag
This flag is set when there are unread data in the receive buffer and cleared when the receive buffer is empty (i.e., does not contain any unread data). When the Receiver is disabled, the receive buffer will be flushed and consequently the RXCIF will become zero.
When interrupt-driven data reception is used, the receive complete interrupt routine must read the received data from RXDATA in order to clear the RXCIF. If not, a new interrupt will occur directly after the return from the current interrupt. This flag can also be cleared by writing a one to its bit location.

Bit 6 – TXCIF: USART Transmit Complete Interrupt Flag
This flag is set when the entire frame in the Transmit Shift Register has been shifted out and there are no new data in the transmit buffer (TXDATA). The TXCIF is automatically cleared when the transmit complete interrupt vector is executed. The flag can also be cleared by writing a one to its bit location.

Bit 5 – DREIF: USART Data Register Empty Flag
The DREIF indicates if the transmit buffer (TXDATA) is ready to receive new data. The flag is one when the transmit buffer is empty, and zero when the transmit buffer contains data to be transmitted that has not yet been moved into the Shift Register. DREIF is set after a reset to indicate that the Transmitter is ready. Always write this bit to zero when writing the STATUS register.
DREIF is cleared by writing TXDATAL. When interrupt-driven data transmission is used, the Data Register Empty interrupt routine must either write new data to TXDATA in order to clear DREIF or disable the Data Register Empty interrupt. If not, a new interrupt will occur directly after the return from the current interrupt.

Bit 4 – RXSIF: USART Receive Start Interrupt Flag
The RXSIF flag indicates a valid start condition on RxD line. The flag is set when the system is in standby modes and a high (IDLE) to low (START) valid transition is detected on the RxD line. If the start detection is not enabled, the RXSIF will always be read as zero. This flag can only be cleared by writing a one to its bit location. This flag is not used in master SPI mode operation.

Bit 3 – ISFIF: Inconsistent Sync Field Interrupt Flag
This bit is set when the auto-baud is enabled and the sync field bit time are too fast or too slow to give a valid baud setting. It will also be set when USART is set to LINAUTO mode and the SYNC character differ from data value 0x55. Writing a zero to this bit has no effect. Writing a one to this bit will clear the flag and bring the USART back to idle state.

Bit 1 – BDF: Break Detected Flag
This bit is cleared by writing a one to the bit. This bit is intended for USART configured to LINAUTO receive mode, see CTRLB. The break detector has a fixed threshold of 11 bits low for a BREAK to be

detected. The BDF bit is set after a valid BREAK and SYNC character is detected. The bit is automatically cleared when next data is received. The bit will behave identically when USART is set to GENAUTO mode. In NORMAL or CLK2X receive mode, the BDF bit is set when a BREAK is detected.

Bit 0 – WFB: Wait For Break
Setting this bit will make the next low and high transition on RxD line to be registered as a break character. This may be used to wait for a BREAK character of arbitrary width. Combined with USART set to GENAUTO mode, this allow the user to set any BAUD rate trough BREAK and SYNC as long as it falls within valid range of the USART.BAUD register. This bit will always read zero.

24.8.6. Control A

Name: CTRLA
Offset: 0x05
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	RXCIE	TXCIE	DREIE	RXSIE	LBME	ABEIE	RS485[1:0]	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 7 – RXCIE: Receive Complete Interrupt Enable
The bit enables the Receive Complete Interrupt. The enabled interrupt will be triggered when RXCIF in the USART.STATUS register is set.

Bit 6 – TXCIE: Transmit Complete Interrupt Enable
This bit enables the Transmit Complete Interrupt. The enabled interrupt will be triggered when the TXCIF in the USART.STATUS register is set.

Bit 5 – DREIE: Data Register Empty Interrupt Enable
This bit enables the Data Register Empty Interrupt. The enabled interrupt will be triggered when the DREIF in the USART.STATUS register is set.

Bit 4 – RXSIE: Receiver Start Frame Interrupt Enable
Writing a one to this bit enables the Start Frame Detector to generate an interrupt when a start of frame condition is detected

Bit 3 – LBME: Loop-back Mode Enable
Setting this bit will enable an internal connection between TxD and RxD pin.

Bit 2 – ABEIE: Auto-baud Error Interrupt Enable
Setting this bit enables the auto-baud error interrupt. The enabled interrupt will trigger for conditions where ISFIF flag is set.

Bits 1:0 – RS485[1:0]: RS485 Mode
These bits enables the RS485 mode according to table #BITFIELD_LGT_SDL_FS/TABLE_Y2W_P4B_G5.

Value	Name	Description
0x0	OFF	Disabled.
0x1	EXT	Enables RS485 mode with control of an external line driver through a dedicated Transmit Enable (TE) pin.
0x2	INT	Enables RS485 mode with control of the internal USART transmitter.
0x3	-	Reserved.

24.8.7. Control B

Name: CTRLB
Offset: 0x06
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	RXEN	TXEN		SFDEN	ODME	RXMODE[1:0]		MPCM
Access	R/W	R/W		R/W	R/W	R/W	R/W	R/W
Reset	0	0		0	0	0	0	0

Bit 7 – RXEN: Receiver Enable
Writing this bit to '1' enables the USART Receiver. The Receiver will override normal port operation for the RxD pin when enabled. Disabling the Receiver will flush the receive buffer invalidating the FERR, BUFOVF, and PERR flags. In GENAUTO and LINAUTO mode, disabling the receiver will reset the auto-baud detection logic.

Bit 6 – TXEN: Transmitter Enable
Writing this bit to '1' enables the USART Transmitter. The Transmitter will override normal port operation for the TxD pin when enabled. Disabling the Transmitter (writing TXEN to zero) will not become effective until ongoing and pending transmissions are completed, i.e., when the Transmit Shift Register and Transmit Buffer Register do not contain data to be transmitted. When disabled, the Transmitter will no longer override the TxD port.
sdfg

Bit 4 – SFDEN: Start Frame Detection Enable
Writing this bit to '1' enables the USART Start Frame Detection mode. The start frame detector is able to wake up the system from idle or standby sleep modes when a high (IDLE) to low (START) transition is detected on the RxD line.

Bit 3 – ODME: Open Drain Mode Enable
Writing this bit to '1' will make the TxD pin to have open-drain functionality. A pull-up resistor is needed to prevent the line from floating when a logic one is output to TxD pin.

Bits 2:1 – RXMODE[1:0]: Receiver Mode
In CLK2X mode, the divisor of the baud rate divider will be reduced from 16 to 8 effectively doubling the transfer rate for asynchronous communication modes. For synchronous operation the CLK2X mode has no effect and RXMODE should always be written to zero. RXMODE must be zero when the USART Communication Mode is configured to IRCOM. Setting RXMODE to GENAUTO enables generic auto-baud where the SYNC character is valid when eight low and high bits has been registered. In this mode any SYNC character that gives a valid BAUD rate will be accepted. In LINAUTO mode the SYNC character is constrained and found valid if each bits falls within 16 +/- 3 baud samples of the internal baud rate and match data value 0x55. Both GENAUTO and LINAUTO mode is only supported for USART operated in asynchronous slave mode.

Value	Name	Description
0x0	NORMAL	Normal USART Mode, Standard Transmission Speed
0x1	CLK2X	Normal USART Mode, Double Transmission Speed

Value	Name	Description
0x2	GENAUTO	Generic Auto-baud Mode
0x3	LINAUTO	LIN Constrained Auto-baud Mode

Bit 0 – MPCM: Multi-Processor Communication Mode
Writing a '1' to this bit enables the Multi-Processor Communication mode: the USART Receiver ignores all the incoming frames that do not contain address information. The Transmitter is unaffected by the MPCM setting. For more detailed information see Multiprocessor Communication Mode.

24.8.8. Control C

When the USART Communication Mode bits in this register are selecting Master SPI mode (CMODE written to MSPI), some bit fields are altered. See CTRLC for the correct description.

Name: CTRLC
Offset: 0x07
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	CMODE[1:0]		PMODE[1:0]		SBMODE	CHSIZE[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	1	1

Bits 7:6 – CMODE[1:0]: USART Communication Mode
Writing these bits selects the communication mode of the USART.
Writing a 0x3 to these bits alters the available bit fields in this register, see GUID-8196F8CB-3128-45C1-9F8B-01A19B6C5C52.

Value	Name	Description
0x0	ASYNCHRONOUS	Asynchronous USART
0x1	SYNCHRONOUS	Synchronous USART
0x2	IRCOM	Infrared Communication
0x3	MSPI	Master SPI

Bits 5:4 – PMODE[1:0]: Parity Mode
Writing these bits enables and selects the type of parity generation.

When enabled, the Transmitter will automatically generate and send the parity of the transmitted data bits within each frame. The Receiver will generate a parity value for the incoming data, compare it to the PMODE setting, and set the Parity Error flag (PERR) in the Status register (USART.STATUS) if a mismatch is detected.

In Master SPI mode (CMODE=0x3), these bits have no effect.

Value	Name	Description
0x0	DISABLED	Disabled
0x1	-	Reserved
0x2	EVEN	Enabled, Even Parity
0x3	ODD	Enabled, Odd Parity

Bit 3 – SBMODE: Stop Bit Mode
Writing this bit selects the number of stop bits to be inserted by the Transmitter.
The Receiver ignores this setting.

In Master SPI mode (CMODE written to MSPI), this bit has no effect.

Value	Description
0	1 stop bit
1	2 stop bits

Bits 2:0 – CHSIZE[2:0]: Character Size
Writing these bits select the number of data bits in a frame. The Receiver and Transmitter use the same setting. For 9BIT character size, the order of which byte to read or write first, low or high byte of RXDATA or TXDATA is selectable.

Value	Name	Description
0x0	5BIT	5-bit
0x1	6BIT	6-bit
0x2	7BIT	7-bit
0x3	8BIT	8-bit
0x4	-	Reserved
0x5	-	Reserved
0x6	9BIT	9-bit (Low byte first)
0x7	9BIT	9-bit (High byte first)

24.8.9. Control C

This register description is valid when the USART Communication Mode bits in this register are selecting Master SPI mode (CMODE written to MSPI). For other CMODE values, see CTRLC for the correct description.

See USART in Master SPI mode for full description of the Master SPI Mode operation.

Name: CTRLC
Offset: 0x07
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	CMODE[1:0]		PMODE[1:0]		SBMODE	UDORD	UCPHA	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Reset	0	0	0	0	0	0	0	

Bits 7:6 – CMODE[1:0]: USART Communication Mode
Writing these bits selects the communication mode of the USART.
Writing a value different than 0x3 to these bits alters the available bit fields in this register, see CTRLC.

Value	Name	Description
0x0	ASYNCHRONOUS	Asynchronous USART
0x1	SYNCHRONOUS	Synchronous USART
0x2	IRCOM	Infrared Communication
0x3	MSPI	Master SPI.

Bits 5:4 – PMODE[1:0]: Parity Mode
Writing these bits enables and selects the type of parity generation.

When enabled, the Transmitter will automatically generate and send the parity of the transmitted data bits within each frame. The Receiver will generate a parity value for the incoming data, compare it to the PMODE setting, and set the Parity Error flag (PERR) in the Status register (USART.STATUS) if a mismatch is detected.

In Master SPI mode (CMODE=0x3), these bits have no effect.

Value	Name	Description
0x0	DISABLED	Disabled
0x1	-	Reserved
0x2	EVEN	Enabled, Even Parity
0x3	ODD	Enabled, Odd Parity

Bit 3 – SBMODE: Stop Bit Mode
Writing this bit selects the number of stop bits to be inserted by the Transmitter.
The Receiver ignores this setting.

In Master SPI mode (CMODE=0x3), this bit has no effect.

Value	Description
0	1 stop bit
1	2 stop bits

Bit 2 – UDORD: Data Order
Writing this bit selects the frame format.

The Receiver and Transmitter use the same setting. Changing the setting of UDORD will corrupt all ongoing communication for both receiver and transmitter.

Value	Description
0	MSB of the data word is transmitted first
1	LSB of the data word is transmitted first

Bit 1 – UCPHA: Clock Phase
The UCPHA bit setting determine if data is sampled on the leading (first) edge or tailing (last) edge of XCKn. Refer to the Master SPI Mode Clock Generation for details.

24.8.10. Baud Register

The USART.BAUDL and USART.BAUDH register pair represents the 16-bit value, USART.BAUD. The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + 0x01. For more details on reading and writing 16-bit registers, refer to Accessing 16-bit Registers.

Ongoing transmissions of the Transmitter and Receiver will be corrupted if the baud rate is changed. Writing this register will trigger an immediate update of the baud rate prescaler. For more information of how to set the baud rate, see Internal Clock Generation - The Fractional Baud Rate Generator

Name: BAUD
Offset: 0x08
Reset: 0x00
Property: -

Bit	15	14	13	12	11	10	9	8
	BAUD[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	BAUD[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:8 – BAUD[15:8]: USART Baud Rate high byte
These bits hold the MSB of the 16-bit Baud register.

Bits 7:0 – BAUD[7:0]: USART Baud Rate low byte
These bits hold the LSB of the 16-bit Baud register.

24.8.11. Debug Control Register

Name: DBGCTRL
Offset: 0x0B
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
								DBGRUN
Access								R/W
Reset								0

Bit 0 – DBGRUN: Debug Run
When this bit is written to '1', XOCD_BREAK_REQ has no effect on the module.

24.8.12. IrDA Control Register

Name: EVCTRL
Offset: 0x0C
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
								IREI
Access								R/W
Reset								0

Bit 0 – IREI: IrDA Event Input Enable
This bit enables the event source for the IRCOM Receiver. If event input is selected for the IRCOM Receiver, the input from the USART's RX pin is automatically disabled.

24.8.13. IRCOM Transmitter Pulse Length Control Register

Name: TXPLCTRL
Offset: 0x0D
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	TXPL[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – TXPL[7:0]: Transmitter Pulse Length
The 8-bit value sets the pulse modulation scheme for the transmitter. Setting this register will have no effect if IRCOM mode is not selected by a USART. By leaving this register value to zero, 3/16 of baud rate period pulse modulation is used. Setting this value from 1 to 254 will give a fixed pulse length coding. The 8-bit value sets the number of system clock periods for the pulse. The start of the pulse will be synchronized with the rising edge of the baud rate clock. Setting the value to 255 (0xFF) will disable pulse coding, letting the RX and TX signals pass through the IRCOM Module unaltered. This enables other features through the IRCOM Module, such as half-duplex USART, Loop-back testing and USART RX input from an Event Channel.
Note: TXPL must be configured before USART transmitter is enabled (TXEN).

24.8.14. IRCOM Receiver Pulse Length Control Register

Name: RXPLCTRL
Offset: 0x0E
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
		RXPL[6:0]						
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bits 6:0 – RXPL[6:0]: Receiver Pulse Length
The 8-bit value sets the filter coefficient for the IRCOM transceiver. Setting this register will have no effect if IRCOM mode is not selected by a USART.
By leaving this register value to zero, filtering is disabled. Setting this value between 1 and 255 will enable filtering, where x+1 equal samples is required for the pulse to be accepted.

Note: RXPL must be configured before USART receiver is enabled (RXEN).

25. SPI - Serial Peripheral Interface

25.1. Overview

The Serial Peripheral Interface (SPI) is a high-speed synchronous data transfer interface using three or four pins. It allows fast communication between an AVR device and peripheral devices or between several microcontrollers. The SPI supports full-duplex communication.

A device connected to the bus must act as a master or slave. The master initiates and controls all data transactions. The interconnection between master and slave devices with SPI is shown in the block diagram. The system consists of two shift registers and a master clock generator. The SPI master initiates the communication cycle by pulling the slave select ($\overline{SS}$) signal low for the desired slave. Master and slave prepare the data to be sent in their respective shift registers, and the master generates the required clock pulses on the SCK line to interchange data. Data are always shifted from master to slave on the master output, slave input (MOSI) line, and from slave to master on the master input, slave output (MISO) line. After each data packet, the master can synchronize the slave by pulling the SS line high.

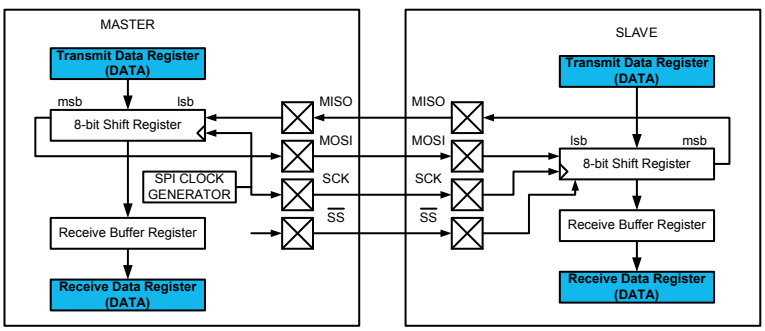
In SPI slave mode, the control logic will sample the incoming signal on the SCK pin. To ensure correct sampling of this clock signal, the minimum low and high periods must each be longer than two CPU clock cycles.

25.2. Features

- Full-duplex, Three-wire Synchronous Data Transfer
- Master or Slave Operation
- LSB First or MSB First Data Transfer
- Seven Programmable Bit Rates
- End of Transmission Interrupt Flag
- Write Collision Flag Protection
- Wake-up from Idle Mode
- Double Speed (CK/2) Master SPI Mode

25.3. Block Diagram

Figure 25-1. SPI Block Diagram



25.4. Signal Description

Signal	Description	Type
MOSI	Master Out Slave In	Master mode: User defined, Slave mode: Input
MISO	Master In Slave Out	Master mode: Input, Slave mode: User defined
SCK	Slave clock (generated by master)	Master mode: User defined, Slave mode: Input
$\overline{SS}$	Slave select (generated by master)	Master mode: User defined, Slave mode: Input

Related Links

I/O Multiplexing and Considerations on page 19

25.5. Product Dependencies

In order to use this peripheral, other parts of the system must be configured correctly, as described below.

Table 25-1. SPI Product Dependencies

Dependency	Applicable	Peripheral
Clocks	Yes	CLKCTRL
I/O Lines and Connections	Yes	PORT
Interrupts	Yes	CPUINT
Events	No	-
Debug	Yes	UPDI

Related Links

Clocks on page 86
I/O Lines and Connections on page 329
Interrupts on page 54
Debug Operation on page 365

25.5.1. Clocks

This peripheral depends on the peripheral clock.

Related Links

CLKCTRL - Clock Controller on page 68
Product Dependencies on page 98

25.5.2. I/O Lines and Connections

Using the I/O lines of the peripheral requires configuration the I/O pins.

Related Links

PORT - I/O Pin Controller on page 132

25.5.3. Interrupts

Using the interrupts of this peripheral requires the Interrupt Controller to be configured first.

Related Links

CPUINT - CPU Interrupt Controller on page 97
SREG on page 51
Interrupts on page 136

25.5.4. Events

Not applicable.

25.5.5. Debug Operation

When run-time debugging, this peripheral will continue normal operation. Halting the CPU in debugging mode will halt normal operation of the peripheral.

This peripheral can be forced to operate with halted CPU by writing a '1' to the Debug Run bit in the Debug Control register of the peripheral (*peripheral*_DBGCTRL.DBGRUN).

If the peripheral is configured to require periodical service by the CPU through interrupts or similar, improper operation or data loss may result during halted debugging.

Related Links

UPDI - Unified Program Debug Interface on page 490

25.6. Functional Description

25.6.1. Principle of Operation

The SPI is an interface is a high-speed synchronous data transfer interface, supporting full duplex communication.

The SPI peripheral can be configured as either Master, controlling all data transactions by pulling a Slave's Select ($\overline{SS}$) signal low, or Slave, controlled by an SPI Master.

Data are shifted from Master to Slave on the Master Output, Slave Input (MOSI) line, and from Slave to Master on the Master Input, Slave Output (MISO) line.

Master/Slave synchronization is achieved by $\overline{SS}$.

25.6.2. Basic Operation

25.6.2.1. Initialization

Initialize the SPI to a basic functional state by following these steps:

1. Configure the $\overline{SS}$ pin in the Port peripheral.
2. Select SPI Master / Slave operation by writing the Master/Slave Select bit in the Control A register (SPI_CTRLA.MASTER).
3. In Master mode, select the clock speed by writing the Prescaler bits and the Clock Double bit in the Control A register (SPI_CTRLA.CLK2X and .PRESC).
4. Optional: Select the data transfer mode by writing to the Mode bits in the Control B register (SPI_CTRLB.MODE).
5. Optional: Write the Data Order bit in Control A (SPI_CTRLA.DORD).
6. If Hardware $\overline{SS}$ control is required in Master mode, write '1' to the Slave Select Disable bit in the Control B register (SPI_CTRLB.SSD).
7. Enable the SPI by writing the Enable bit in the Control A register (SPI_CTRLA.ENABLE=1).

Related Links

I/O Multiplexing and Considerations on page 19

PORT - I/O Pin Controller on page 132

25.6.2.2. Master Mode Operation

In master mode, the SPI interface has no automatic control of the $\overline{SS}$ line. If the $\overline{SS}$ pin is used, it must be configured as output and controlled by user software. If the bus consists of several SPI slaves and/or masters, a SPI master can use general purpose I/O pins to control the $\overline{SS}$ line to each of the slaves on the bus.

Writing a byte to the DATA register starts the SPI clock generator and the hardware shifts the eight bits into the selected slave. After shifting one byte and when there are no pending data, the Data Register Empty Interrupt flag (SPI_INTFLAGS.DREIF) is set, the SPI clock generator stops, and the Transfer Complete interrupt flag (SPI_INTFLAGS.TXCIF) is set.

If there are pending data, SPI_INTFLAGS.DREIF is cleared; the master will continue to shift the next bytes. After each byte is shifted out, the new data is copied to the shift register and the DREIF flag is set. Only when a shift is completed and there are no more pending data, the SPI_INTFLAGS.TXCIF flag is set. An end-of-transfer can also be signaled by pulling the $\overline{SS}$ line high. The last incoming byte will be kept in the shift register.

If the $\overline{SS}$ pin is not used it can be disabled by writing the Slave Select Disable bit in the Control B register (SPI_CTRLB.SSD). If not disabled and configured as input, the pin must be held high to ensure master operation.

If the $\overline{SS}$ pin is set as input and is being driven low, the SPI peripheral will interpret this as another master trying to take control of the bus. To avoid bus contention, the master will take the following action:

1. The master enters slave mode.
2. The Slave Select Interrupt Fflag (SPI_INTFLAGS.SSIF) is set.

25.6.2.3. Slave Mode

In slave mode, the SPI peripheral will remain sleeping with the MISO line tri-stated as long as the $\overline{SS}$ pin is driven high. In this state, software may update the contents of the DATA register, but the data will not be shifted out by incoming clock pulses on the SCK pin until the $\overline{SS}$ pin is driven low. If $\overline{SS}$ is driven low, the slave will start to shift out data on the first SCK clock pulse. When one byte has been completely shifted, the SPI Interrupt flag (SPI_INTFLAGS.IF) is set. The slave may continue placing new data to be sent into the SDI_DATA register before reading the incoming data. The last incoming byte will be kept in the buffer register.

When $\overline{SS}$ is driven high, the SPI logic is halted, and the SPI slave will not receive any new data. Any partially received packet in the shift register will be dropped.

As the $\overline{SS}$ pin is used to signal the start and end of a transfer, it is also useful for doing packet/byte synchronization, keeping the slave bit counter synchronous with the master clock generator.

To ensure that write collision never can happen, the SPI peripheral can be configured in buffered mode by writing a '1' to the Buffer Mode Enable bit in the Control B register (SPI_CTRLB.BUFEN). Then, data is copied from the Transmit Register to the Shift Register only at receive complete. This means that, after data is written to the Transmit Buffer, one SPI transfer must be completed before the data is copied into the shift register.

25.6.2.4. Buffer Modes

There are three buffer modes:

- Unbuffered mode:
  The default mode is unbuffered in the transmit direction and single buffered in the receive direction. This means that bytes to be transmitted cannot be written to the SPI_DATA register before the entire shift cycle is completed. When receiving data, a received character must be read from the

SPI_DATA register before the next character has been completely shifted in. Otherwise, the first byte will be lost.

- Buffered mode with dummy transfer:
  The SPI peripheral is single buffered in the transmit direction and double buffered in the receive direction. A byte written to the transmit register will be copied to the shift register when a full character has been received. When receiving data, a received character must be read from the SPI_DATA register before the third character has been completely shifted in to avoid losing data.
- Buffered mode without dummy transfer:
  The SPI peripheral is single buffered in the transmit direction and double buffered in the receive direction. A byte written to the transmit register will be copied to the shift register when the SPI is enabled and $\overline{SS}$ is high.

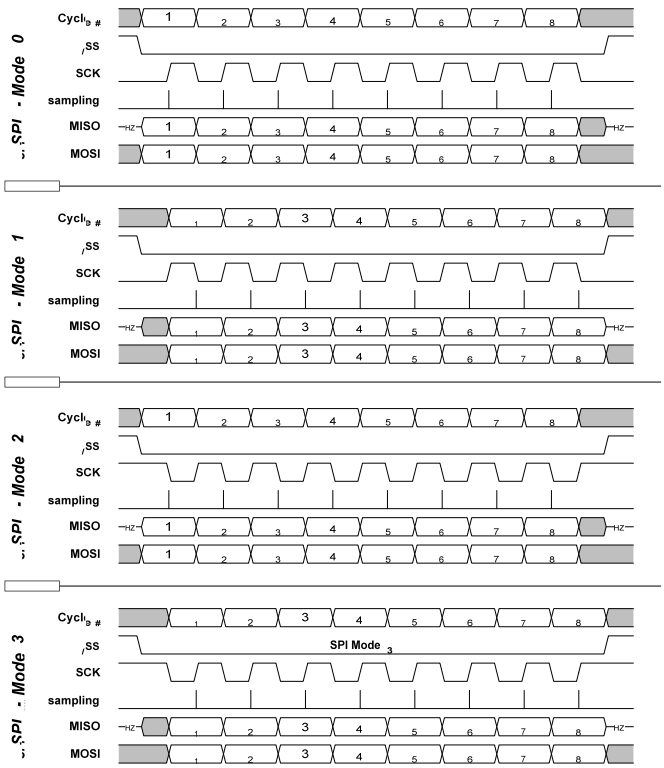
25.6.2.5. Data Modes

There are four combinations of SCK phase and polarity with respect to serial data. The desired combination is selected by writing to the Mode bits in the Control B register (SPI_CTRLB.MODE).

The SPI data transfer formats are shown below. Data bits are shifted out and latched in on opposite edges of the SCK signal, ensuring sufficient time for data signals to stabilize.

The leading edge is the first clock edge of a clock cycle. The trailing edge is the last clock edge of a clock cycle.

Figure 25-2. SPI Data Transfer Modes



25.6.3. Interrupts

Table 25-2. Available Interrupt Vectors and Sources

Offset	Name	Vector Description	Conditions
0x00	SPI	SPI interrupt	• SSI: Slave Select Trigger Interrupt • DRE: Data Register Empty Interrupt • TXC: Transfer Complete Interrupt • RXC: Receive Complete Interrupt

When an interrupt condition occurs, the corresponding Interrupt Flag is set in the Status register of the peripheral (USART_STATUS).

An interrupt source is enabled or disabled by writing to the corresponding bit in the peripheral's Interrupt Control register (USART_CTRLA).

An interrupt request is generated when the corresponding interrupt source is enabled and the Interrupt Flag is set. The interrupt request remains active until the Interrupt Flag is cleared. See the USART_STATUS register for details on how to clear Interrupt Flags.

Note: Interrupt requests are only generated when Interrupts are enabled globally.

Related Links

SREG on page 51
CPUINT - CPU Interrupt Controller on page 97

25.6.4. Sleep Mode Operation

The SPI will continue working in Idle sleep mode. When entering any deeper sleep mode, any ongoing transaction will be stopped.

Related Links

SLPCTRL - Sleep Controller on page 85

25.6.5. Configuration Change Protection

Not applicable.

25.7. Register Summary

Offset	Name	Bit Pos.								
0x00	CTRLA	7:0		DORD	MASTER	CLK2X		PRESC[1:0]		ENABLE
0x01	CTRLB	7:0	BUFEN	BUFWR				SSD	MODE[1:0]	
0x02	INTCTRL	7:0	RXCIE	TXCIE	DREIE	SSIE				IE
0x03	INTFLAGS	7:0	RXCIF/IF	TXCIF/ WRCOL	DREIF	SSIF				BUFOVF
0x04	DATA	7:0	DATA[7:0]							

25.8. Register Description

25.8.1. Control A

Name: CTRLA
Offset: 0x00
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
		DORD	MASTER	CLK2X		PRESC[1:0]		ENABLE
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		0	0	0		0	0	0

Bit 6 – DORD: Data Order

Value	Description
0	The MSB of the data word is transmitted first.
1	The LSB of the data word is transmitted first.

Bit 5 – MASTER: Master/Slave Select
If $\overline{SS}$ is configured as input and driven low while this bit is '1', this bit is cleared, and the IF flag in SPI.INTFLAGS is set. The user has to write MASTER=1 again to re-enable SPI Master mode.

Value	Description
0	SPI Slave mode selected
1	SPI Master mode selected

Bit 4 – CLK2X: Clock Double
When this bit is written to '1' the SPI speed (SCK frequency, after internal prescaler) is doubled in Master mode.

Value	Description
0	SPI speed (SCK frequency) is not doubled.
1	SPI speed (SCK frequency) is doubled in Master mode

Bits 2:1 – PRESC[1:0]: Prescaler
This bit field controls the SPI clock rate configured in master mode. These bits have no effect in slave mode. The relationship between SCK and the peripheral clock frequency (CLK_PER) is shown below.

Note: The output of the SPI prescaler can be doubled by writing the CLK2X bit to '1'.

Value	Name	Description
0x0	DIV4	CLK_PER/4
0x1	DIV16	CLK_PER/16
0x2	DIV64	CLK_PER/64
0x3	DIV128	CLK_PER/128

Bit 0 – ENABLE: SPI Enable

Value	Description
0	SPI is disabled.
1	SPI is enabled.

25.8.2. Control B

Name: CTRLB
Offset: 0x01
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	BUFEN	BUFWR				SSD	MODE[1:0]	
Access	R/W	R/W				R/W	R/W	R/W
Reset	0	0				0	0	0

Bit 7 – BUFEN: Buffer Mode Enable
Writing this bit to '1' enables Buffer Mode, meaning two buffers in reception, one buffer in transmission, and separated interrupt flags for transmission and reception.

Bit 6 – BUFWR: Buffer Mode Wait for Receive
When writing this bit to '0' the first data transferred will be a dummy sample.

Value	Description
0	One SPI transfer must be completed before the data is copied into the shift register.
1	When writing to the data register when the SPI is enabled and $\overline{SS}$ is high, the first write will go directly to the shift register.

Bit 2 – SSD: Slave Select Disable
Setting this bit will disable the Slave Select line when operating as SPI Master.

Value	Description
0	Enable the Slave Select line when operating as SPI Master.
1	Disable the Slave Select line when operating as SPI Master.

Bits 1:0 – MODE[1:0]: Mode
These bits select the transfer mode. The four combinations of SCK phase and polarity with respect to the serial data are shown in the table below. These bits decide whether the first edge of a clock cycle (leading edge) is rising or falling, and whether data setup and sample occur on the leading or trailing edge. When the leading edge is rising, the SCK signal is low when idle, and when the leading edge is falling, the SCK signal is high when idle.

Value	Name	Description
0x0	0	Leading edge: Rising, sample Trailing edge: Falling, setup
0x1	1	Leading edge: Rising, setup Trailing edge: Falling, sample
0x2	2	Leading edge: Falling, sample Trailing edge: Rising, setup

Value	Name	Description
0x3	3	Leading edge: Falling, setup Trailing edge: Rising, sample

25.8.3. Interrupt Control

Name: INTCTRL
Offset: 0x02
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	RXCIE	TXCIE	DREIE	SSIE				IE
Access	R/W	R/W	R/W	R/W				R/W
Reset	0	0	0	0				0

Bit 7 – RXCIE: Receive Complete Interrupt Enable
In buffer mode this bit enables the receive complete interrupt. The enabled interrupt will be triggered when the RXCIF flag in the INTFLAG register is set. In non-buffer mode this bit does not have any effect.

Bit 6 – TXCIE: Transfer Complete Interrupt Enable
In buffer mode this bit enables the transfer complete interrupt. The enabled interrupt will be triggered when the TXCIF flag in the INTFLAG register is set. In non-buffer mode this bit does not have any effect.

Bit 5 – DREIE: Data Register Empty Interrupt Enable
In buffer mode this bit enables the data register empty interrupt. The enabled interrupt will be triggered when the DREIF flag in the INTFLAG register is set. In non-buffer mode this bit does not have any effect.

Bit 4 – SSIE: Slave Select Trigger Interrupt Enable
In buffer mode this bit enables the Slave Select interrupt. The enabled interrupt will be triggered when the SSIF flag in the INTFLAG register is set. In non-buffer mode this bit does not have any effect.

Bit 0 – IE: Interrupt Enable
This bit enables the SPI interrupt when the SPI is not in buffer mode. The enabled interrupt will be triggered when the corresponding flag is set in the INTFLAG register.

25.8.4. Interrupt Flags

Name: INTFLAGS
Offset: 0x03
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	RXCIF/IF	TXCIF/WRCOL	DREIF	SSIF				BUFOVF
Access	R/W	R/W	R/W	R/W				R/W
Reset	0	0	0	0				0

Bit 7 – RXCIF/IF: Receive Complete Interrupt Flag/Interrupt Flag
RXCIF: In buffer mode this flag is set when there is unread data in the receive buffer and cleared when the receive buffer is empty (i.e., does not contain any unread data). In non-buffer mode this bit does not have any effect.

When interrupt-driven data reception is used, the receive complete interrupt routine must read the received data from DATA in order to clear RXCIF. If not, a new interrupt will occur directly after the return from the current interrupt. This flag can also be cleared by writing a one to its bit location.

IF: This flag is set when a serial transfer is complete and one byte is completely shifted in/out of the DATA register. If $\overline{SS}$ is configured as input and is driven low when the SPI is in master mode, this will also set this flag. IF is cleared by hardware when executing the corresponding interrupt vector. Alternatively, the IF flag can be cleared by first reading the SPI.INTFLAGS register when IF is set, and then accessing the DATA register.

Bit 6 – TXCIF/WRCOL: Transfer Complete Interrupt Flag/Write Collision Flag
TXCIF: In buffer mode this flag is set when all the data in the transmit shift register has been shifted out and there are no new data in the transmit buffer (DATA). The flag is cleared by writing a one to its bit location. In non-buffer mode this bit does not have any effect.

WRCOL: The WRCOL flag is set if the DATA register is written during a data transfer. This flag is cleared by first reading the SPI.INTFLAGS register when WRCOL is set, and then accessing the DATA register.

Bit 5 – DREIF: Data Register Empty Interrupt Flag
In buffer mode this flag indicates whether the transmit buffer (DATA) is ready to receive new data. The flag is one when the transmit buffer is empty and zero when the transmit buffer contains data to be transmitted that has not yet been moved into the shift register. DREIF is set after a reset to indicate that the transmitter is ready. In non-buffer mode this bit does not have any effect.

DREIF is cleared by writing DATA. When interrupt-driven data transmission is used, the data register empty interrupt routine must either write new data to DATA in order to clear DREIF or disable the data register empty interrupt. If not, a new interrupt will occur directly after the return from the current interrupt.

Bit 4 – SSIF: Slave Select Trigger Interrupt Flag
In buffer mode this flag indicates that the SPI has been in master mode and the SS line has been pulled low externally so the SPI is now working in slave mode. The flag will only be set if the Slave Select Disable (SSD) is not enabled. The flag is cleared by writing a one to its bit location. In non-buffer mode this bit does not have any effect.

Bit 0 – BUFOVF: Buffer Overflow

This flag indicates data loss due to a receiver buffer full condition. This flag is set if a buffer overflow condition is detected. A buffer overflow occurs when the receive buffer is full (two characters) and the third byte has been received. If there is no transmit data the buffer overflow will not be set before the start of a new serial transfer. This flag is valid until the receive buffer (DATA) is read. Always write this bit location to zero when writing the SPI.INTFLAGS register.

25.8.5. Data

Name: DATA
Offset: 0x04
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	DATA[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – DATA[7:0]: SPI Data
The DATA register is used for sending and receiving data. Writing to the register initiates the data transmission, and the byte written to the register will be shifted out on the SPI output line.

Reading the register causes the first byte in the buffer FIFO to be read. Additionally received bytes will then be shifted in the FIFO.

26. TWI - Two Wire Interface

26.1. Overview

The Two-Wire Interface (TWI) is a bidirectional, two-wire communication interface. It is I^2C and System Management Bus (SMBus) compatible. The only external hardware needed to implement the bus is one pull-up resistor on each bus line.

Any device connected to the bus must act as a master or a slave. The master initiates a data transaction by addressing a slave on the bus and telling whether it wants to transmit or receive data. One bus can have many slaves and one or several masters that can take control of the bus. An arbitration process handles priority if more than one master tries to transmit data at the same time. Mechanisms for resolving bus contention are inherent in the protocol.

The TWI module supports master and slave functionality. The master and slave functionality are separated from each other, and can be enabled and configured separately. The master module supports multi-master bus operation and arbitration. It contains the baud rate generator. All 100kHz, 400kHz, and 1MHz bus frequencies are supported. Quick command and smart mode can be enabled to auto-trigger operations and reduce software complexity.

The slave module implements 7-bit address match and general address call recognition in hardware. 10-bit addressing is also supported. A dedicated address mask register can act as a second address match register or as a register for address range masking. The slave continues to operate in all sleep modes, including power-down mode. This enables the slave to wake up the device from all sleep modes on TWI address match. It is possible to disable the address matching to let this be handled in software instead.

The TWI module will detect START and STOP conditions, bus collisions, and bus errors. Arbitration lost, errors, collision, and clock hold on the bus are also detected and indicated in separate status flags available in both master and slave modes.

It is possible to disable the TWI drivers in the device, and enable a four-wire digital interface for connecting to an external TWI bus driver. This can be used for applications where the device operates from a different V_{DD} voltage than used by the TWI bus. It is also possible to enable the bridge mode. In this case, the slave I/O pins are selected from an alternative port, enabling independent and simultaneous master and slave operation.

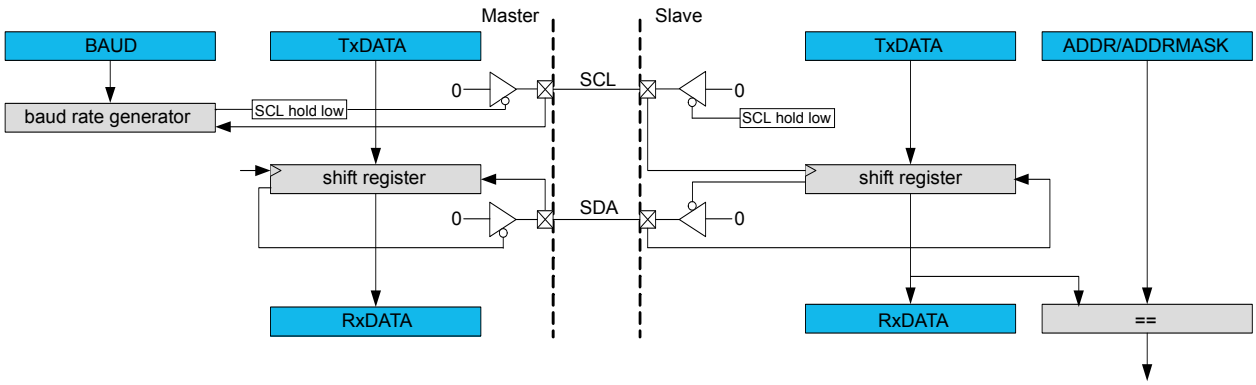
26.2. Features

- Bidirectional, two-wire communication interface
  - Philips I^2C compatible
  - System Management Bus (SMBus) compatible
- Bus master and slave operation supported
  - Slave operation
  - Single bus master operation
  - Bus master in multi-master bus environment
  - Multi-master arbitration
- Flexible slave address match functions
  - 7-bit and general call address recognition in hardware
  - 10-bit addressing supported
  - Address mask register for dual address match or address range masking

  - Optional software address recognition for unlimited number of addresses
- Slave can operate in all sleep modes, including power-down
- Slave address match can wake device from all sleep modes
- Up to 1MHz bus frequency support
- Slew-rate limited output drivers
- Input filter for bus noise and spike suppression
- Support arbitration between start/repeated start and data bit (SMBus)
- Slave arbitration allows support for address resolve protocol (ARP) (SMBus)
- Supports SMBus Layer 1 timeouts
- Configurable timeout values
- Independent timeout counters in master and slave (Bridge mode support)

26.3. Block Diagram

Figure 26-1. TWI Block Diagram



26.4. Signal Description

Signal	Description	Type
SCL	Serial clock line	Digital I/O
SDA	Serial data line	Digital I/O

Related Links

I/O Multiplexing and Considerations on page 19

26.5. Product Dependencies

In order to use this peripheral, other parts of the system must be configured correctly, as described below.

Table 26-1. TWI Product Dependencies

Dependency	Applicable	Peripheral
Clocks	Yes	CLKCTRL
I/O Lines and Connections	Yes	PORT
Interrupts	Yes	CPUINT

Dependency	Applicable	Peripheral
Events	No	-
Debug	Yes	UPDI

Related Links

Clocks on page 380

I/O Lines and Connections on page 329

Interrupts on page 54

Debug Operation on page 380

26.5.1. Clocks

This peripheral requires the system clock (CLK_PER). The relationship between CLK_PER and the TWI bus clock (SCL) is explained in the TWI_MBAUD register.

Related Links

CLKCTRL - Clock Controller on page 68

MBAUD on page 400

26.5.2. I/O Lines and Connections

Using the I/O lines of the peripheral requires configuration the I/O pins.

Related Links

PORT - I/O Pin Controller on page 132

26.5.3. Interrupts

Using the interrupts of this peripheral requires the Interrupt Controller to be configured first.

Related Links

CPUINT - CPU Interrupt Controller on page 97

SREG on page 51

Interrupts on page 136

26.5.4. Events

Not applicable.

26.5.5. Debug Operation

When run-time debugging, this peripheral will continue normal operation. Halting the CPU in debugging mode will halt normal operation of the peripheral.

This peripheral can be forced to operate with halted CPU by writing a '1' to the Debug Run bit in the Debug Control register of the peripheral (*peripheral*_DBGCTRL.DBGRUN).

Note: When the CPU is halted in debug mode an DBGRUN=1, that reading/writing the DATA register will neither trigger a bus operation nor cause transmit and clear flags.

If the peripheral is configured to require periodical service by the CPU through interrupts or similar, improper operation or data loss may result during halted debugging.

Related Links

UPDI - Unified Program Debug Interface on page 490

26.6. Functional Description

26.6.1. Principle of Operation

The Two-Wire Interface (TWI) uses two communication lines for data transfer. It can either act as master or slave. The master initiates a data transaction by addressing a slave on the bus and telling whether it wants to transmit or receive data.

The following sections explain general TWI bus concepts. See Basic Operation for the TWI implementation on this device.

26.6.1.1. General TWI Bus Concepts

The TWI provides a simple, bidirectional, two-wire communication bus consisting of a serial clock line (SCL) and a serial data line (SDA). The two lines are open-collector lines (wired-AND), and pull-up resistors (Rp) are the only external components needed to drive the bus. The pull-up resistors provide a high level on the lines when none of the connected devices are driving the bus.

The TWI bus is a simple and efficient method of interconnecting multiple devices on a serial bus. A device connected to the bus can be a master or slave, where the master controls the bus and all communication.

Figure 26-2 illustrates the TWI bus topology.

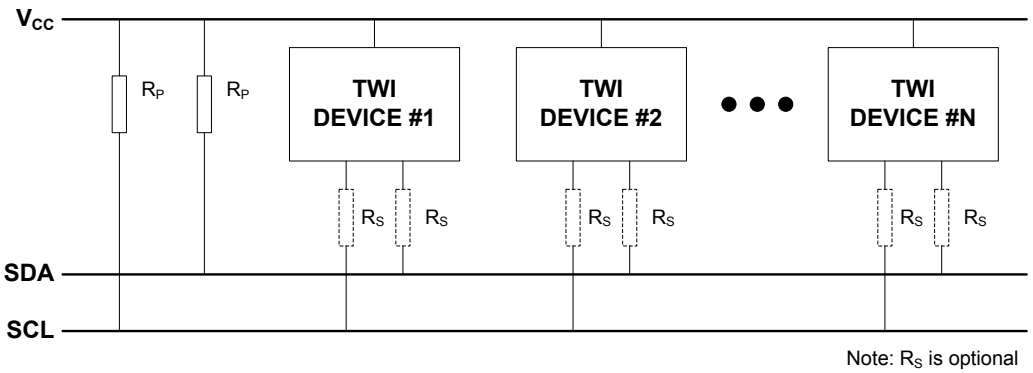


Figure 26-2. TWI Bus Topology

An unique address is assigned to all slave devices connected to the bus, and the master will use this to address a slave and initiate a data transaction.

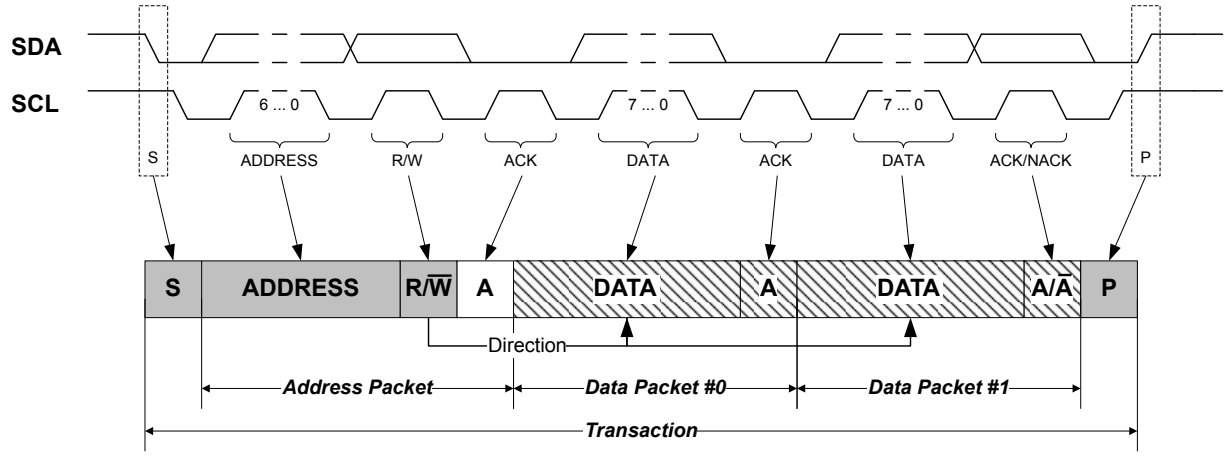
Several masters can be connected to the same bus, called a multi-master environment. An arbitration mechanism is provided for resolving bus ownership among masters, since only one master device may own the bus at any given time.

A device can contain both master and slave logic, and can emulate multiple slave devices by responding to more than one address.

A master indicates the start of a transaction by issuing a START condition (S) on the bus. An address packet with a slave address (ADDRESS) and an indication whether the master wishes to read or write data (R/$\overline{W}$) are then sent. After all data packets (DATA) are transferred, the master issues a STOP condition (P) on the bus to end the transaction. The receiver must acknowledge (A) or not-acknowledge ($\overline{A}$) each byte received.

Figure 26-3 shows a TWI transaction.

Figure 26-3. Basic TWI Transaction Diagram Topology for a 7-bit Address Bus

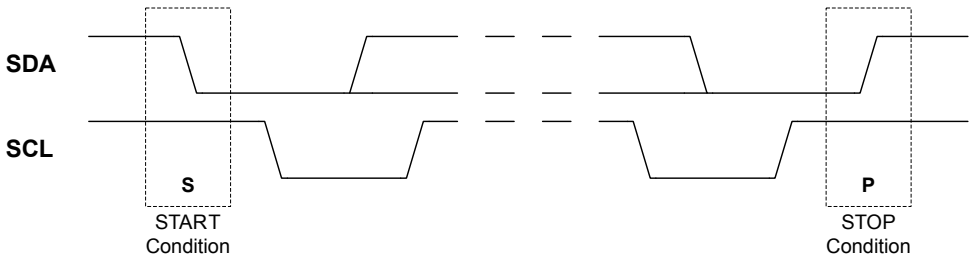


The master provides the clock signal for the transaction, but a device connected to the bus is allowed to stretch the low-level period of the clock to decrease the clock speed.

START and STOP Conditions

Two unique bus conditions are used for marking the beginning (START) and end (STOP) of a transaction. The master issues a START condition (S) by indicating a high-to-low transition on the SDA line while the SCL line is kept high. The master completes the transaction by issuing a STOP condition (P), indicated by a low-to-high transition on the SDA line while SCL line is kept high.

Figure 26-4. START and STOP Conditions

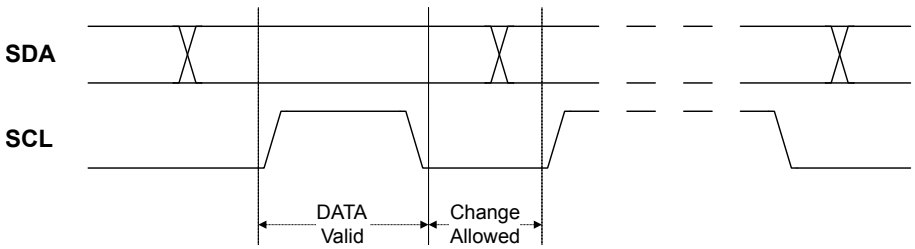


Multiple START conditions can be issued during a single transaction. A START condition that is not directly following a STOP condition is called a repeated START condition (Sr).

Bit Transfer

As illustrated by Figure 26-5, a bit transferred on the SDA line must be stable for the entire high period of the SCL line. Consequently the SDA value can only be changed during the low period of the clock. This is ensured in hardware by the TWI module.

Figure 26-5. Data Validity



Combining bit transfers results in the formation of address and data packets. These packets consist of eight data bits (one byte) with the most-significant bit transferred first, plus a single-bit not-acknowledge (NACK) or acknowledge (ACK) response. The addressed device signals ACK by pulling the SCL line low during the ninth clock cycle, and signals NACK by leaving the line SCL high.

Address Packet

After the START condition, a 7-bit address followed by a read/write (R/$\overline{W}$) bit is sent. This is always transmitted by the master. A slave recognizing its address will ACK the address by pulling the data line low for the next SCL cycle, while all other slaves should keep the TWI lines released and wait for the next START and address. The address, R/$\overline{W}$ bit, and acknowledge bit combined is the address packet. Only one address packet for each START condition is allowed, also when 10-bit addressing is used.

The R/$\overline{W}$ bit specifies the direction of the transaction. If the R/$\overline{W}$ bit is low, it indicates a master write transaction, and the master will transmit its data after the slave has acknowledged its address. If the R/$\overline{W}$ bit is high, it indicates a master read transaction, and the slave will transmit its data after acknowledging its address.

Data Packet

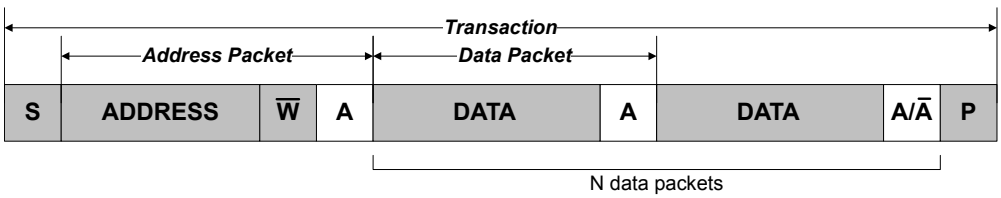
An address packet is followed by one or more data packets. All data packets are nine bits long, consisting of one data byte and an acknowledge bit. The direction bit in the previous address packet determines the direction in which the data are transferred.

Transaction

A transaction is the complete transfer from a START to a STOP condition, including any repeated START conditions in between. The TWI standard defines three fundamental transaction modes: Master write, master read, and a combined transaction.

Figure 26-6 illustrates the master write transaction. The master initiates the transaction by issuing a START condition (S) followed by an address packet with the direction bit set to zero (ADDRESS+$\overline{W}$).

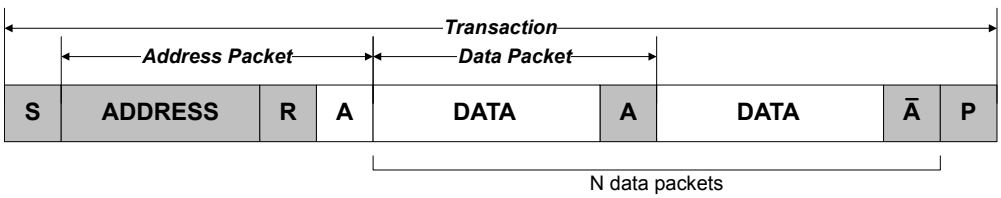
Figure 26-6. Master Write Transaction



Assuming the slave acknowledges the address, the master can start transmitting data (DATA) and the slave will ACK or NACK (A/$\overline{A}$) each byte. If no data packets are to be transmitted, the master terminates the transaction by issuing a STOP condition (P) directly after the address packet. There are no limitations to the number of data packets that can be transferred. If the slave signals a NACK to the data, the master must assume that the slave cannot receive any more data and terminate the transaction.

Figure 26-7 illustrates the master read transaction. The master initiates the transaction by issuing a START condition followed by an address packet with the direction bit set to one (ADDRESS+R). The addressed slave must acknowledge the address for the master to be allowed to continue the transaction.

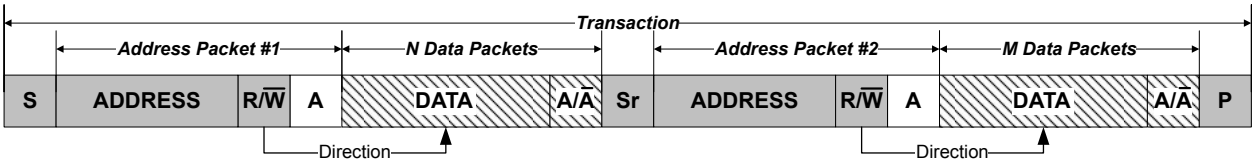
Figure 26-7. Master Read Transaction



Assuming the slave acknowledges the address, the master can start receiving data from the slave. There are no limitations to the number of data packets that can be transferred. The slave transmits the data while the master signals ACK or NACK after each data byte. The master terminates the transfer with a NACK before issuing a STOP condition.

Figure 26-8 illustrates a combined transaction. A combined transaction consists of several read and write transactions separated by repeated START conditions (Sr).

Figure 26-8. Combined Transaction

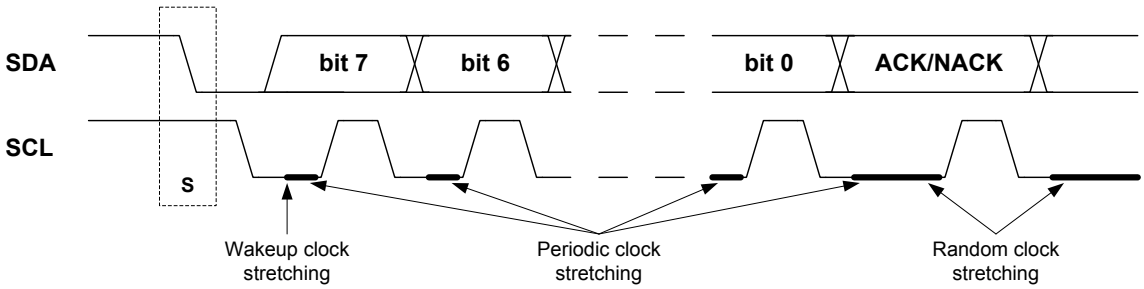


Clock and Clock Stretching

All devices connected to the bus are allowed to stretch the low period of the clock to slow down the overall clock frequency or to insert wait states while processing data. A device that needs to stretch the clock can do this by holding/forcing the SCL line low after it detects a low level on the line.

Three types of clock stretching can be defined, as shown in Figure 26-9.

Figure 26-9. Clock Stretching (1)



Note: Clock stretching is not supported by all I^2C slaves and masters.

If a slave device is in sleep mode and a START condition is detected, the clock stretching normally works during the wake-up period. For AVR devices, the clock stretching will be either directly before or after the ACK/NACK bit, as AVR devices do not need to wake up for transactions that are not addressed to it.

A slave device can slow down the bus frequency by stretching the clock periodically on a bit level. This allows the slave to run at a lower system clock frequency. However, the overall performance of the bus will be reduced accordingly. Both the master and slave device can randomly stretch the clock on a byte level basis before and after the ACK/NACK bit. This provides time to process incoming or prepare outgoing data, or perform other time-critical tasks.

In the case where the slave is stretching the clock, the master will be forced into a wait state until the slave is ready, and vice versa.

Arbitration

A master can start a bus transaction only if it has detected that the bus is idle. As the TWI bus is a multi-master bus, it is possible that two devices may initiate a transaction at the same time. This results in multiple masters owning the bus simultaneously. This is solved using an arbitration scheme where the master loses control of the bus if it is not able to transmit a high level on the SDA line. The masters who lose arbitration must then wait until the bus becomes idle (i.e., wait for a STOP condition) before attempting to reacquire bus ownership. Slave devices are not involved in the arbitration procedure.

Figure 26-10. TWI Arbitration

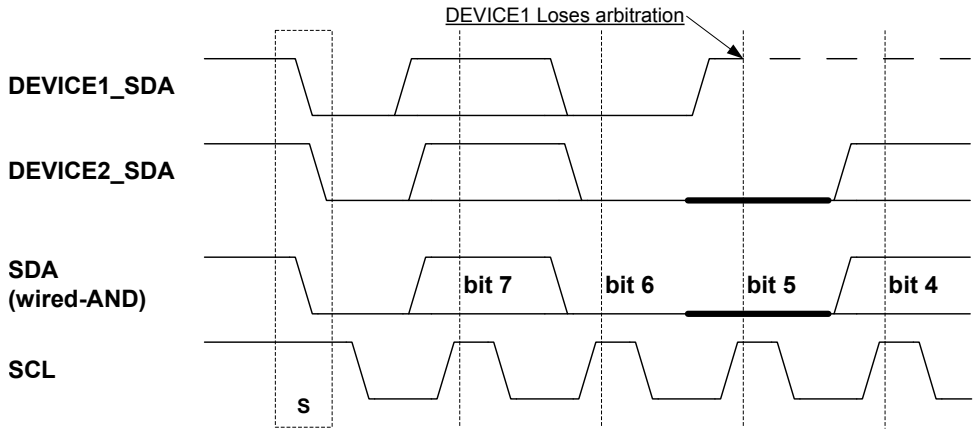


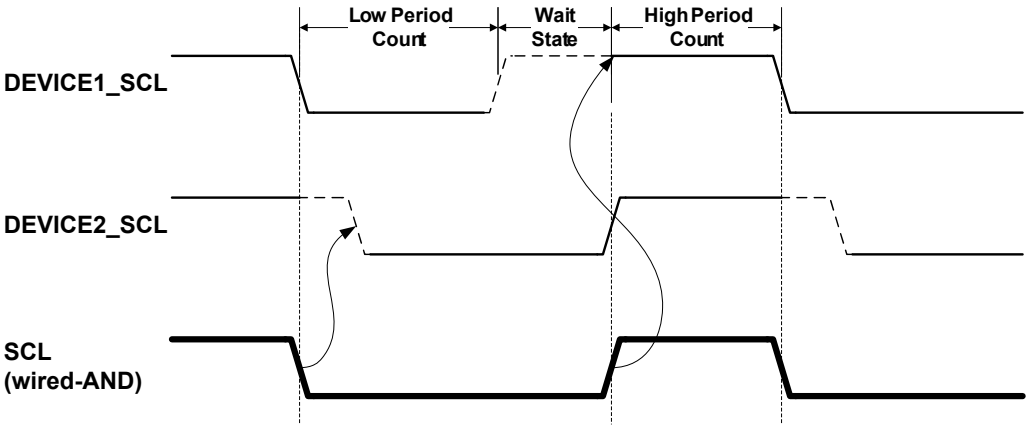
Figure 26-10 shows an example where two TWI masters are contending for bus ownership. Both devices are able to issue a START condition, but DEVICE1 loses arbitration when attempting to transmit a high level (bit 5) while DEVICE2 is transmitting a low level.

Arbitration between a repeated START condition and a data bit, a STOP condition and a data bit, or a repeated START condition and a STOP condition are not allowed and will require special handling by software.

Synchronization

A clock synchronization algorithm is necessary for solving situations where more than one master is trying to control the SCL line at the same time. The algorithm is based on the same principles used for the clock stretching previously described. Figure 26-11shows an example where two masters are competing for control over the bus clock. The SCL line is the wired-AND result of the two masters clock outputs.

Figure 26-11. Clock Synchronization



A high-to-low transition on the SCL line will force the line low for all masters on the bus, and they will start timing their low clock period. The timing length of the low clock period can vary among the masters. When a master (DEVICE1 in this case) has completed its low period, it releases the SCL line. However, the SCL line will not go high until all masters have released it. Consequently, the SCL line will be held low by the device with the longest low period (DEVICE2). Devices with shorter low periods must insert a wait state until the clock is released. All masters start their high period when the SCL line is released by all devices and has gone high. The device which first completes its high period (DEVICE1) forces the clock line low, and the procedure is then repeated. The result is that the device with the shortest clock period determines the high period, while the low period of the clock is determined by the device with the longest clock period.

26.6.1.2. TWI Bus State Logic

The bus state logic continuously monitors the activity on the TWI bus lines when the master is enabled. It continues to operate in all sleep modes, including power-down.

The bus state logic includes START and STOP condition detectors, collision detection, inactive bus timeout detection, and a bit counter. These are used to determine the bus state. Software can get the current bus state by reading the bus state bits in the master status register. The bus state can be unknown, idle, busy, or owner, and is determined according to the state diagram shown in Figure 26-12. The values of the bus state bits according to state are shown in binary in the figure.

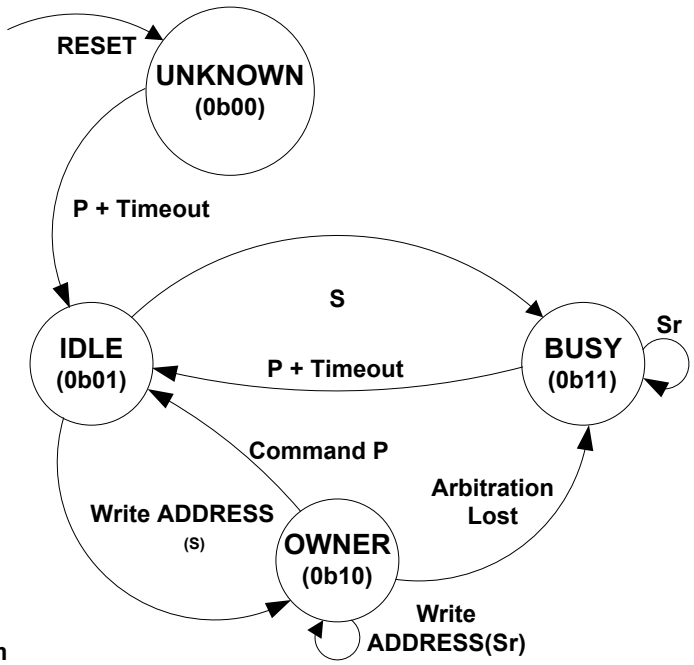


Figure 26-12. Bus State, State Diagram

After a system reset and/or TWI master enable, the bus state is unknown. The bus state machine can be forced to enter idle by writing to the bus state bits accordingly. If no state is set by application software, the bus state will become idle when the first STOP condition is detected. If the master inactive bus timeout is enabled, the bus state will change to idle on the occurrence of a timeout. After a known bus state is established, only a system reset or disabling of the TWI master will set the state to unknown.

When the bus is idle, it is ready for a new transaction. If a START condition generated externally is detected, the bus becomes busy until a STOP condition is detected. The STOP condition will change the bus state to idle. If the master inactive bus timeout is enabled, the bus state will change from busy to idle on the occurrence of a timeout.

If a START condition is generated internally while in idle state, the owner state is entered. If the complete transaction was performed without interference, i.e., no collisions are detected, the master will issue a STOP condition and the bus state will change back to idle. If a collision is detected, the arbitration is assumed lost and the bus state becomes busy until a STOP condition is detected. A repeated START condition will only change the bus state if arbitration is lost during the issuing of the repeated START. Arbitration during repeated START can be lost only if the arbitration has been ongoing since the first START condition. This happens if two masters send the exact same ADDRESS+DATA before one of the masters issues a repeated START (Sr).

26.6.2. Basic Operation

26.6.2.1. Electrical Characteristics

The TWI module in AVR devices follows the electrical specifications and timing of I^2C bus and SMBus. These specifications are not 100% compliant, and so to ensure correct behavior, the inactive bus timeout period should be set in TWI master mode. Refer to TWI Master Operation for more details.

26.6.2.2. Initialization

To start the TWI as Master, write a '1' to the Enable bit in the Master Control A register (TWI_MCTRLA.ENABLE), followed by writing the slave address to the Master Address (TWI_MADDR) register. The TWI_MADDR register also has a R/$\overline{W}$ bit which indicates whether the Master is transmitting or receiving. The Master DATA register (TWI_MDATA) is written in case master is transmitting data.

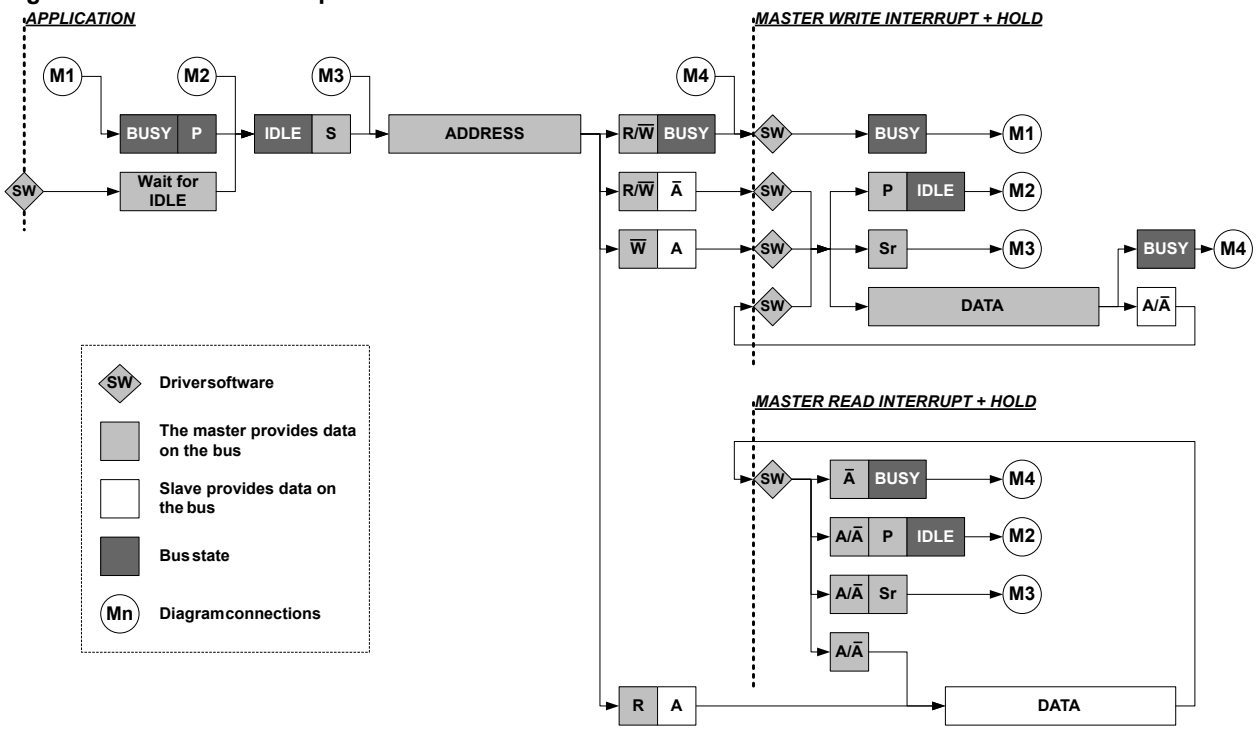
To enable the TWI as Slave, write a '1' to the Enable bit in the Slave Control A register (TWI_SCTRLA.ENABLE). The TWI peripheral will wait to receive a byte addressed to it.

26.6.2.3. TWI Master Operation

The TWI master is byte-oriented, with an optional interrupt after each byte. There are separate interrupts for master write and master read. Interrupt flags can also be used for polled operation. There are dedicated status flags for indicating ACK/NACK received, bus error, arbitration lost, clock hold, and bus state.

When an interrupt flag is set, the SCL line is forced low. This will give the master time to respond or handle any data, and will in most cases require software interaction. Figure 26-13 shows the TWI master operation. The diamond shaped symbols (SW) indicate where software interaction is required. Clearing the interrupt flags releases the SCL line.

Figure 26-13. TWI Master Operation



The number of interrupts generated is kept to a minimum by automatic handling of most conditions. Quick command and smart mode can be enabled to auto-trigger operations and reduce software complexity.

Transmitting Address Packets

After issuing a START condition, the master starts performing a bus transaction when the master address register is written with the 7-bit slave address and direction bit. If the bus is busy, the TWI master will wait until the bus becomes idle before issuing the START condition.

Depending on arbitration and the R/$\overline{W}$ direction bit, one of four distinct cases (M1 to M4) arises following the address packet. The different cases must be handled in software.

Case M1: Arbitration Lost or Bus Error during Address Packet

If arbitration is lost during the sending of the address packet, the Master Write Interrupt Flag (TWI_MSTATUS.WIF) and Arbitration Lost Flag (TWI_MSTATUS.ARBLOST) are both set. Serial data output to the SDA line is disabled, and the SCL line is released. The master is no longer allowed to perform any operation on the bus until the bus state has changed back to idle.

A bus error will behave in the same way as an arbitration lost condition, but the Bus Error Flag (TWI_MSTATUS.BUSERR) is set in addition to the write interrupt and arbitration lost flags.

Case M2: Address Packet Transmit Complete - Address not Acknowledged by Slave

If no slave device responds to the address, the Master Write Interrupt Flag (TWI_MSTATUS.WIF) and the Master Received Acknowledge Flag (TWI_MSTATUS.RXACK) are set. The clock hold is active at this point, preventing further activity on the bus.

Case M3: Address Packet Transmit Complete - Direction Bit Cleared

If the master receives an ACK from the slave, the Master Write Interrupt Flag (TWI_MSTATUS.WIF) is set and the Master Received Acknowledge Flag (TWI_MSTATUS.RXACK) is cleared. The clock hold is active at this point, preventing further activity on the bus.

Case M4: Address Packet Transmit Complete - Direction Bit Set

If the master receives an ACK from the slave, the master proceeds to receive the next byte of data from the slave. When the first data byte is received, the Master Read Interrupt Flag (TWI_MSTATUS.RIF) is set and the Master Received Acknowledge Flag (TWI_MSTATUS.RXACK) is cleared. The clock hold is active at this point, preventing further activity on the bus.

Transmitting Data Packets

The slave will know when an address packet with R/$\overline{W}$ direction bit set has been successfully received. It can then start sending data by writing to the slave data register. When a data packet transmission is completed, the data interrupt flag is set. If the master indicates NACK, the slave must stop transmitting data and expect a STOP or repeated START condition.

Receiving Data Packets

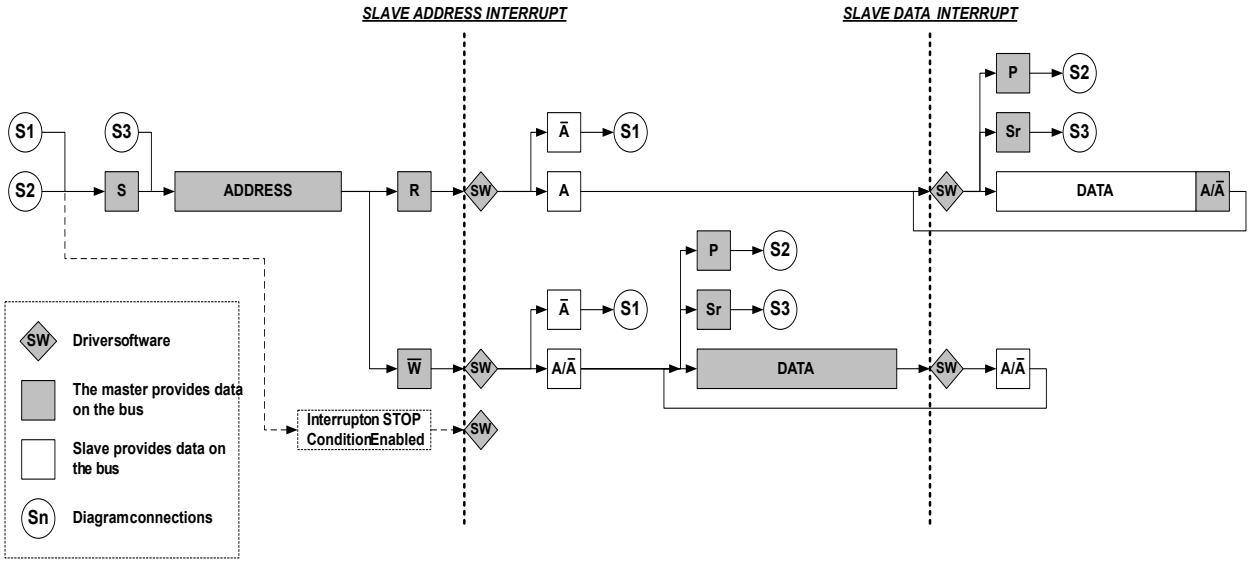
The slave will know when an address packet with R/$\overline{W}$ direction bit cleared has been successfully received. After acknowledging this, the slave must be ready to receive data. When a data packet is received, the data interrupt flag is set and the slave must indicate ACK or NACK. After indicating a NACK, the slave must expect a STOP or repeated START condition.

26.6.2.4. TWI Slave Operation

The TWI slave is byte-oriented with optional interrupts after each byte. There are separate slave data and address/stop interrupts. Interrupt flags can also be used for polled operation. There are dedicated status flags for indicating ACK/NACK received, clock hold, collision, bus error, and read/write direction.

When an interrupt flag is set, the SCL line is forced low. This will give the slave time to respond or handle data, and will in most cases require software interaction. Figure 26-14. shows the TWI slave operation. The diamond shapes symbols (SW) indicate where software interaction is required.

Figure 26-14. TWI Slave Operation



The number of interrupts generated is kept to a minimum by automatic handling of most conditions. Quick command can be enabled to auto-trigger operations and reduce software complexity.

Promiscuous mode can be enabled to allow the slave to respond to all received addresses.

Receiving Address Packets

When the TWI slave is properly configured, it will wait for a START condition to be detected. When this happens, the successive address byte will be received and checked by the address match logic, and the slave will ACK a correct address and store the address in the DATA register. If the received address is not a match, the slave will not acknowledge and store address, and will wait for a new START condition.

The slave address/stop interrupt flag is set when a START condition succeeded by a valid address byte is detected. A general call address will also set the interrupt flag.

A START condition immediately followed by a STOP condition is an illegal operation, and the bus error flag is set.

The R/$\overline{W}$ direction flag reflects the direction bit received with the address. This can be read by software to determine the type of operation currently in progress.

Depending on the R/$\overline{W}$ direction bit and bus condition, one of four distinct cases (S1 to S4) arises following the address packet. The different cases must be handled in software.

Case S1: Address Packet Accepted - Direction Bit Set

If the R/$\overline{W}$ direction flag is set, this indicates a master read operation. The SCL line is forced low by the slave, stretching the bus clock. If ACK is sent by the slave, the slave hardware will set the data interrupt flag indicating data is needed for transmit. Data, repeated START, or STOP can be received after this. If NACK is sent by the slave, the slave will wait for a new START condition and address match.

Case S2: Address Packet Accepted - Direction Bit Cleared

If the R/$\overline{W}$ direction flag is cleared, this indicates a master write operation. The SCL line is forced low, stretching the bus clock. If ACK is sent by the slave, the slave will wait for data to be received. Data, repeated START, or STOP can be received after this. If NACK is sent, the slave will wait for a new START condition and address match.

Case S3: Collision

If the slave is not able to send a high level or NACK, the collision flag is set, and it will disable the data and acknowledge output from the slave logic. The clock hold is released. A START or repeated START condition will be accepted.

Case S4: STOP Condition Received

When the STOP condition is received, the slave address/stop flag will be set, indicating that a STOP condition, and not an address match, occurred.

Receiving Data Packets

The slave will know when an address packet with R/$\overline{W}$ direction bit cleared has been successfully received. After acknowledging this, the slave must be ready to receive data. When a data packet is received, the data interrupt flag is set and the slave must indicate ACK or NACK. After indicating a NACK, the slave must expect a STOP or repeated START condition.

Transmitting Data Packets

The slave will know when an address packet with R/$\overline{W}$ direction bit set has been successfully received. It can then start sending data by writing to the slave data register. When a data packet transmission is completed, the data interrupt flag is set. If the master indicates NACK, the slave must stop transmitting data and expect a STOP or repeated START condition.

26.6.3. Events

Not applicable.

26.6.4. Interrupts

Table 26-2. Available Interrupt Vectors and Sources

Offset	Name	Vector Description	Conditions
0x00	Slave	TWI Slave interrupt	• DIF: Data Interrupt Flag (SSTATUS.DIF) set • APIF: Address or Stop Interrupt Flag (SSTATUS.APIF) set
0x02	Master	TWI Master interrupt	• RIF: Read Interrupt Flag (MSTATUS.RIF) set • WIF: Write Interrupt Flag (MSTATUS.WIF) set

When an interrupt condition occurs, the corresponding Interrupt Flag is set in the Master or Slave Status register (TWI_MSTATUS/TWI_SSTATUS).

When several interrupt request conditions are supported by an interrupt vector, the interrupt requests are ORed together into one combined interrupt request to the Interrupt Controller. The user must read the peripheral's INTFLAGS register to determine which of the interrupt conditions are present.

Related Links

CPUINT - CPU Interrupt Controller on page 97

SREG on page 51

26.6.5. Sleep Mode Operation

The bus state logic and Slave continue to operate in all sleep modes, including Power Down sleep mode. If a Slave device is in sleep mode and a START condition is detected, clock stretching is active during the wake-up period until the system clock is available. Master will stop operation in all sleep modes.

26.6.6. Synchronization

Not applicable.

26.6.7. Configuration Change Protection

Not applicable.

26.7. Register Summary

Offset	Name	Bit Pos.								
0x00	CTRLA	7:0				SDASETUP	SDAHOLD[1:0]		FMPEN	
0x01	Reserved									
0x02	DBGCTRL	7:0								DBGRUN
0x03	MCTRLA	7:0	RIEN	WIEN		QCEN	TIMEOUT[1:0]		SMEN	ENABLE
0x04	MCTRLB	7:0					FLUSH	ACKACT	CMD[1:0]	
0x05	MSTATUS	7:0	RIF	WIF	CLKHOLD	RXACK	ARBLOST	BUSERR	BUSSTATE[1:0]	
0x06	MBAUD	7:0	BAUD[7:0]							
0x07	MADDR	7:0	ADDR[7:0]							
0x08	MDATA	7:0	DATA[7:0]							
0x09	SCTRLA	7:0	DIEN	APIEN	PIEN			PMEN	SMEN	ENABLE
0x0A	SCTRLB	7:0						ACKACT	CMD[1:0]	
0x0B	SSTATUS	7:0	DIF	APIF	CLKHOLD	RXACK	COLL	BUSERR	DIR	AP
0x0C	SADDR	7:0	ADDR[7:0]							
0x0D	SDATA	7:0	DATA[7:0]							
0x0E	SADDRMASK	7:0	ADDRMASK[6:0]							ADDREN

26.8. Register Description

26.8.1. Control A

Name: CTRLA
Offset: 0x00
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
				SDASETUP	SDAHOLD[1:0]		FMPEN	
Access				R/W	R/W	R/W	R/W	
Reset				0	0	0	0	

Bit 4 – SDASETUP: SDA Setup Time
By default, there are 4 clock cycles of setup time on SDA out signal while reading from slave part of the TWI module. Writing this bit to '1' will change the setup time to 8 clocks.

Value	Name	Description
0	4CYC	SDA setup time is 4 clock cycles
1	8CYC	SDA setup time is 8 clock cycle

Bits 3:2 – SDAHOLD[1:0]: SDA Hold Time
Writing these bits selects the SDA hold time.

Table 26-3. SDA Hold Time

SDAHOLD[1:0]	Nominal Hold Time	Hold Time Range across All Corners (ns)	Description
0x0	OFF	0	Hold time off.
0x1	50ns	36 - 131	Backward compatible setting.
0x2	300ns	180 - 630	Meets SMBus specification under typical conditions.
0x3	500ns	300 - 1050	Meets SMBus specification across all corners.

Bit 1 – FMPEN: FM Plus Enable
Writing these bits selects the 1MHz bus speed for the TWI in default configuration or for TWI Master in bridge configuration.

Value	Description
0	Fm+ disabled
1	Fm+ enabled

26.8.2. Debug Control

Name: DBGCTRL
Offset: 0x02
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
								DBGRUN
Access								R/W
Reset								0

Bit 0 – DBGRUN: Debug Run

Value	Description
0	XOCD_BREAK_REQ is respected
1	XOCD_BREAK_REQ is has no effect

26.8.3. Master Control A

Name: MCTRLA
Offset: 0x03
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	RIEN	WIEN		QCEN	TIMEOUT[1:0]		SMEN	ENABLE
Access	R/W	R/W		R/W	R/W	R/W	R/W	R/W
Reset	0	0		0	0	0	0	0

Bit 7 – RIEN: Read Interrupt Enable
Writing this bit to '1' enables interrupt on the Master Read Interrupt Flag (RIF) in the Master Status register (TWI.MSTATUS). A TWI Master read interrupt would be generated only if this bit, the RIF, and the Global Interrupt Flag (I) in CPU.SREG are all '1'.

Bit 6 – WIEN: Write Interrupt Enable
Writing this bit to '1' enables interrupt on the Master Write Interrupt Flag (WIF) in the Master Status register (TWI.MSTATUS). A TWI Master write interrupt will be generated only if this bit, the WIF, and the Global Interrupt Flag (I) in CPU.SREG are all '1'.

Bit 4 – QCEN: Quick Command Enable
Writing this bit to '1' enables Quick Command. When Quick Command is enabled, the corresponding interrupt flag is set immediately after the slave acknowledges the address. At this point the software can either issue a Stop command or a repeated Start by writing either the Command bits (CMD) in the Master Control B register (TWI.MCTRLB) or the Master Address register (TWI.MADDR).

Bits 3:2 – TIMEOUT[1:0]: Inactive Bus Timeout

Value	Name	Description
0x0	DISABLED	Bus timeout disabled. I^2C.
0x1	50US	50µs - SMBus (assume baud is set to 100kHz)
0x2	100US	100µs (assume baud is set to 100kHz)
0x3	200US	200µs (assume baud is set to 100kHz)

Bit 1 – SMEN: Smart Mode Enable
Writing this bit to '1' enables the Master smart mode. When smart mode is enabled acknowledge action is sent immediate after reading the Master Data (TWI.MDATA) register.

Bit 0 – ENABLE: Enable TWI Master
Writing this bit to '1' enables the TWI as Master.

26.8.4. Master Control B

Name: MCTRLB
Offset: 0x04
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
					FLUSH	ACKACT	CMD[1:0]	
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit 3 – FLUSH: Flush
Writing a '1' to this bit generates a strobe for one clock cycle disabling and then enabling the master.

Writing '0' has no effect.

The purpose is to clear the internal state of master: For TWI master to transmit successfully, it is recommended to write the Master Address register (TWI.MADDR) first and then the Master Data register (TWI.MDATA).

The peripheral will transmit invalid data if TWI.MDATA is written before TWI.MADDR. To avoid this invalid transmission, write '1' to this bit to clear both registers.

Bit 2 – ACKACT: Acknowledge Action
This bit defines the master's behavior under certain conditions defined by the bus protocol state and software interaction. The acknowledge action is performed when DATA is read, or when an execute command is written to the CMD bits.

The ACKACT bit is not a flag or strobe, but an ordinary read/write accessible register bit. The default ACKACT for master read interrupt is "Send ACK" (0). For master write, the code will know that no acknowledge should be sent since it is itself sending data.

Value	Description
0	Send ACK
1	Send NACK

Bits 1:0 – CMD[1:0]: Command
Note: The master command bits are strobes. These bits are always read as zero.

Writing to these bits triggers a master operation as defined by the table below.

Table 26-4. Command Settings

CMD[1:0]	DIR	Description
0x0	X	NOACT
0x1	X	REPSTART - Execute Acknowledge Action succeeded by repeated Start.
0x2	0	RECVTRANS - Execute Acknowledge Action succeeded by a byte read operation.
	1	Execute Acknowledge Action (no action) succeeded by a byte send operation.(1)
0x3	X	STOP - Execute Acknowledge Action succeeded by issuing a STOP condition.

1. For a master being a sender, it will normally wait for new data written to the Master Data Register (TWI.MDATA).

Note: The acknowledge action bits and command bits can be written at the same time.

26.8.5. Master Status

Normal TWI operation dictates that this register is regarded purely as a read-only register. Clearing any of the status flags is done indirectly by accessing the Master transmits address (TWI.MADDR), Master Data register (TWI.MDATA), or the Command bits (CMD) in the Master Control B register (TWI.MCTRLB).

Name: MSTATUS
Offset: 0x05
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	RIF	WIF	CLKHOLD	RXACK	ARBLOST	BUSERR	BUSSTATE[1:0]	
Access	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 7 – RIF: Read Interrupt Flag
This bit is set to '1' when the master byte read operation is successfully completed, i.e. no arbitration lost or bus error occurred during the operation. The read operation is triggered by software reading DATA or writing to ADDR registers with bit '0' set. A slave device must have responded with an ACK to the address and direction byte transmitted by the master for this flag to be set.

Writing a '1' to this bit will clear the RIF. However, normal use of the TWI does not require the flag to be cleared by this method.

Clearing the RIF bit will follow the same software interaction as the CLKHOLD flag.

The RIF flag can generate a master read interrupt (see description of the RIEN control bit in the TWI.MCTRLA register).

Bit 6 – WIF: Write Interrupt Flag
This bit is set when a master transmit address or byte write is completed, regardless of the occurrence of a bus error or an arbitration lost condition.

Writing a '1' to this bit will clear the WIF. However, normal use of the TWI does not require the flag to be cleared by this method.

Clearing the WIF bit will follow the same software interaction as the CLKHOLD flag.

The WIF flag can generate a master write interrupt (see description of the WIEN control bit in the TWI.MCTRLA register).

Bit 5 – CLKHOLD: Clock Hold
If read as '1', this bit indicates that the master is currently holding the TWI clock (SCL) low, stretching the TWI clock period.

Writing a '1' to this bit will clear the CLKHOLD flag. However, normal use of the TWI does not require the CLKHOLD flag to be cleared by this method, since the flag is automatically cleared when accessing several other TWI registers. The CLKHOLD flag can be cleared by:

1. Writing a '1' to it.
2. Writing to the TWI.MADDR register.
3. Writing to the TWI.MDATA register.
4. Reading the TWI.DATA register while the ACKACT control bits in TWI.MCTRLB are set to either send ACK or NACK.

5. Writing a valid command to the TWI.MCTRLB register.

Bit 4 – RXACK: Received Acknowledge
This bit is read-only and contains to the most recently received acknowledge bit from slave.

Bit 3 – ARBLOST: Arbitration Lost
If read as '1' this bit indicates that the master has lost arbitration while transmitting a high data or NACK bit, or while issuing a start or repeated start condition (S/Sr) on the bus.

Writing a '1' to it will clear the ARBLOST flag. However, normal use of the TWI does not require the flag to be cleared by this method. However, as for the CLKHOLD flag, clearing the ARBLOST flag is not required during normal use of the TWI.

Clearing the ARBLOST bit will follow the same software interaction as the CLKHOLD flag.

Given the condition where the bus ownership is lost to another master, the software must either abort operation or resend the data packet. Either way, the next required software interaction is in both cases to write to the TWI.MADDR register. A write access to the TWI.MADDR register will then clear the ARBLOST flag.

Bit 2 – BUSERR: Bus Error
The BUSERR flag indicates that an illegal bus condition has occurred. An illegal bus condition is detected if a protocol violating start (S), repeated start (Sr), or stop (P) is detected on the TWI bus lines. A start condition directly followed by a stop condition is one example of protocol violation.

Writing a '1' to this bit will clear the BUSERR. However, normal use of the TWI does not require the BUSERR to be cleared by this method.

A robust TWI driver software design will treat the bus error flag similarly to the ARBLOST flag, assuming the bus ownership is lost when the bus error flag is set. As for the ARBLOST flag, the next software operation of writing the TWI.MADDR register will consequently clear the BUSERR flag. For bus error to be detected, the bus state logic must be enabled and the system frequency must be 4x the SCL frequency.

Bits 1:0 – BUSSTATE[1:0]: Bus State
These bits indicate the current TWI bus state as defined in the table below. After a System Reset or re-enabling, the TWI master bus state will be unknown. The change of bus state is dependent on bus activity.

Writing 0x1 to the BUSSTATE bits forces the bus state logic into its 'idle' state. However, the bus state logic cannot be forced into any other state. When the master is disabled, the bus-state is 'unknown'.

Value	Name	Description
0x0	UNKNOWN	Unknown bus state
0x1	IDLE	Bus is idle
0x0	OWNER	This USART controls the bus
0x0	BUSY	The bus is busy

26.8.6. Master Baud Rate

Name: MBAUD
Offset: 0x06
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	BAUD[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – BAUD[7:0]: Baud Rate
This bit field defines the relation between the system clock frequency (f_{CLK_PER}) and the TWI bus clock (SCL) frequency following this equation:

$$f_{TWI} = \frac{f_{CLK\_PER}}{2 \cdot (5 + BAUD)}$$

or, rearranged:

$$BAUD = \frac{f_{CLK\_PER}}{2 \cdot f_{TWI}} - 5$$

The BAUD must be set to a value that results in a TWI bus clock frequency (f_{TWI}) equal or less than 100kHz/400kHz, dependent on the standard used by the application.

This bit field should be written while the master is disabled (ENABLE bit in TWI.MCTRLA is '0').

26.8.7. Master Address

Name: MADDR
Offset: 0x07
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	ADDR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – ADDR[7:0]: Address
When this bit field is written, a START condition and slave address protocol sequence is initiated dependent on the bus state.

If the bus state is unknown the Master Write Interrupt Flag (WIF) and bus error flag (BUSERR) in the Master Status register (TWI.MSTATUS) are set and the operation is terminated.

If the bus is busy the master awaits further operation until the bus becomes idle. When the bus is or becomes idle, the master generates a START condition on the bus, copies the ADDR value into the data shift register (TWI.MDATA) and performs a byte transmit operation by sending the contents of the data register onto the bus. The master then receives the response i.e. the acknowledge bit from the slave. After completing the operation the bus clock (SCL) is forced and held low only if arbitration was not lost. The CLKHOLD bit in the Master Setup register (TWI.MSETUP) is set accordingly. Completing the operation sets the WIF in the Master Status register (TWI.MSTATUS).

If the bus is already owned, a repeated start (Sr) sequence is performed. In two ways the repeated start (Sr) sequence deviates from the start sequence. Firstly, since the bus is already owned by the master, no wait for idle bus state is necessary. Secondly, if the previous transaction was a read, the acknowledge action is sent before the repeated start bus condition is issued on the bus.

The master receives one data byte from the slave before the master sets the Master Read Interrupt Flag (RIF) in the Master Status register (TWI.MSTATUS). All TWI Master flags are cleared automatically when this bit field is written. This includes bus error, arbitration lost, and both master interrupt flags.

This register can be read at any time without interfering with ongoing bus activity, since a read access does not trigger the master logic to perform any bus protocol related operations.

Note: The master control logic uses bit 0 of the TWI.MADDR register as the bus protocol's read/write flag (R/$\overline{W}$).

26.8.8. Master DATA

Name: MDATA
Offset: 0x08
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	DATA[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – DATA[7:0]: Data
The bit field gives direct access to the masters physical shift register which is used both to shift data out onto the bus (write) and to shift in data received from the bus (read).

The direct access implies that the data register cannot be accessed during byte transmissions. Build-in logic prevents any write access to this register during the shift operations. Reading valid data or writing data to be transmitted can only be successfully done when the bus clock (SCL) is held low by the master, i.e. when the CLKHOLD bit in the Master Status register (TWI.MSTATUS) is set. However, it is not necessary to check the CLKHOLD bit in software before accessing this register if the software keeps track of the present protocol state by using interrupts or observing the interrupt flags.

Accessing this register assumes that the master clock hold is active, auto-triggers bus operations dependent of the state of the acknowledge action command bit (ACKACT) in TWI.MSTATUS and type of register access (read or write).

A write access to this register will, independent of ACKACT in TWI.MSTATUS, command the master to perform a byte transmit operation on the bus directly followed by receiving the acknowledge bit from the slave. When the acknowledge bit is received, the Master Write Interrupt Flag (WIF) in TWI.MSTATUS is set regardless of any bus errors or arbitration. If operating in a multi-master environment, the interrupt handler or application software must check the Arbitration Lost Status Flag (ARBLOST) in TWI.MSTATUS before continuing from this point. If the arbitration was lost, the application software must decide to either abort or to resend the packet by rewriting this register. The entire operation is performed (i.e. all bits are clocked), regardless of winning or losing arbitration before the write interrupt flag is set. When arbitration is lost, only '1's are transmitted for the remainder of the operation, followed by a write interrupt with ARBLOST flag set.

Both TWI master interrupt flags are cleared automatically when this register is written. However, the Master Arbitration Lost and Bus Error flags are left unchanged.

Reading this register triggers a bus operation, dependent on the setting of the acknowledge action command bit (ACKACT) in TWI.MSTATUS. Normally the ACKACT bit is preset to either ACK or NACK before the register read operation. If ACK or NACK action is selected, the transmission of the acknowledge bit precedes the release of the clock hold. The clock is released for one byte, allowing the slave to put one byte of data on the bus. The Master Read Interrupt flag RIF in TWI.MSTATUS is then set if the procedure was successfully executed. However, if arbitration was lost when sending NACK, or a bus error occurred during the time of operation, the Master Write Interrupt flag (WIF) is set instead. Observe that the two master interrupt flags are mutual exclusive, i.e. both flags will not be set simultaneously.

Both TWI master interrupt flags are cleared automatically if this register is read while ACKACT is set to either ACK or NACK. However, arbitration lost and bus error flags are left unchanged.

26.8.9. Slave Control A

Name: SCTRLA
Offset: 0x09
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	DIEN	APIEN	PIEN			PMEN	SMEN	ENABLE
Access	R/W	R/W	R/W			R/W	R/W	R/W
Reset	0	0	0			0	0	0

Bit 7 – DIEN: Data Interrupt Enable
Writing this bit to '1' enables interrupt on the Slave Data Interrupt Flag (DIF) in the Slave Status register (TWI.SSTATUS). A TWI slave data interrupt will be generated only if this bit, the DIF, and the Global Interrupt Flag (I) in CPU.SREG are all '1'.

Bit 6 – APIEN: Address or Stop Interrupt Enable
Writing this bit to '1' enables interrupt on the Slave Address or Stop Interrupt Flag (APIF) in the Slave Status register (TWI.SSTATUS). A TWI slave address or stop interrupt will be generated only if the this bit, APIF, PIEN in this register, and the Global Interrupt Flag (I) in CPU.SREG are all '1'.

Note: The slave stop interrupt shares the interrupt vector with slave address interrupt. The AP bit determines which caused the interrupt.

Bit 5 – PIEN: Stop Interrupt Enable
Writing this bit to '1' enables APIF to be set when a STOP condition occurs. To use this feature the system frequency must be 4x the SCL frequency.

Bit 2 – PMEN: Address Recognition Mode
If this bit is written to '1', the slave address match logic responds to all received addresses. If set to zero, the address match logic uses the slave address register (TWI.SADDR) to determine which address to recognize as the slaves own address.

Bit 1 – SMEN: Smart Mode Enable
Writing this bit to '1' enables the slave smart mode. When smart mode is enabled, issuing a command with CMD or reading/writing DATA resets the interrupt and operation continues. If smart mode is disabled, the slave always waits for a CMD command before continuing.

Bit 0 – ENABLE: Enable TWI Slave
Writing this bit to '1' enables the TWI slave.

26.8.10. Slave Control B

Name: SCTRLB
Offset: 0x0A
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
						ACKACT	CMD[1:0]	
Access						R/W	R/W	R/W
Reset						0	0	0

Bit 2 – ACKACT: Acknowledge Action
This bit defines the slave's behavior under certain conditions defined by the bus protocol state and software interaction. The table below lists the acknowledge procedure performed by the slave if action is initiated by software. The acknowledge action is performed when TWI.SDATA is read or written, or when an execute command is written to the CMD bits in this register.

The ACKACT bit is not a flag or strobe, but an ordinary read/write accessible register bit.

Value	Name	Description
0	ACK	Send ACK
1	NACK	Send NACK

Bits 1:0 – CMD[1:0]: Command
Unlike the acknowledge action bits, the slave command bits are strobes. These bits always read as zero. Writing to these bits trigger a slave operation as defined in the table below.

Table 26-5. Command Settings

CMD[1:0]	DIR	Description
0x0 - NOACT	X	No action
0x1	X	Reserved
0x2 - COMPTRANS	Used to complete a transaction.	
	0	Execute Acknowledge Action succeeded by waiting for any START (S/Sr) condition.
	1	Wait for any START (S/Sr) condition.
0x3 - RESPONSE	Used in response to an address interrupt (APIF).	
	0	Execute Acknowledge Action succeeded by reception of next byte.
	1	Execute Acknowledge Action succeeded by slave data interrupt.
	Used in response to a data interrupt (DIF).	
	0	Execute Acknowledge Action succeeded by reception of next byte.
	1	Execute a byte read operation followed by Acknowledge Action.

Note that the acknowledge action bits and command bits can be written at the same time.

26.8.11. Slave Status

Normal TWI operation dictates that the slave status register should be regarded purely as a read-only register. Clearing any of the status flags will indirectly be done when accessing the slave data (TWI.SDATA) register or the CMD bits in the Slave Control B register (TWI.SCTRLB).

Name: SSTATUS
Offset: 0x0B
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	DIF	APIF	CLKHOLD	RXACK	COLL	BUSERR	DIR	AP
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit 7 – DIF: Data Interrupt Flag
This flag is set when a slave byte transmit or byte receive operation is successfully completed without any bus error. The flag can be set with an unsuccessful transaction in case of collision detection (see description of the COLL status bit). Writing a '1' to its bit location will clear the DIF. However, normal use of the TWI does not require the DIF flag to be cleared by using this method, since the flag is automatically cleared when:

1. Writing to the slave DATA register.
2. Reading the slave DATA register.
3. Writing a valid command to the CTRLB register.

The DIF flag can be used to generate a slave data interrupt (see description of the DIEN control bit in TWI.CTRLA).

Bit 6 – APIF: Address or Stop Interrupt Flag
This flag is set when the slave address match logic detects that a valid address has been received or by a stop condition. Writing a '1' to its bit location will clear the APIF. However, normal use of the TWI does not require the flag to be cleared by this method since the flag is cleared using same software interactions as described for the DIF flag.

The APIF flag can be used to generate a slave address or stop interrupt (see description of the AIEN control bit in TWI.CTRLA). Take special note of that the slave stop interrupt shares the interrupt vector with slave address interrupt.

Bit 5 – CLKHOLD: Clock Hold
If read as '1', the slave clock hold flag indicates that the slave is currently holding the TWI clock (SCL) low, stretching the TWI clock period. This is a read only bit that is set when an address or data interrupt is set. Resetting the corresponding interrupt will indirectly reset this flag.

Bit 4 – RXACK: Received Acknowledge
This bit is read only and contains to the most recently received acknowledge bit from the master.

Bit 3 – COLL: Collision
If read as one, the transmit collision flag indicates that the slave has not been able to transmit a high data or NACK bit. If a slave transmit collision is detected, the slave will commence its operation as normal, except no low values will be shifted out onto the SDA line (i.e., when the COLL flag is set it disables the

data and acknowledge output from the slave logic). DIF flag will be set at the end as a result of internal completion of unsuccessful transaction. Similarly when collision occurs because slave is not been able to transmit NACK bit, it means address match already happened and APIF flag is set as a result. APIF/DIF flags can only generate interrupt whose handlers can be used to check for the collision. Writing a '1' to its bit location will clear the COLL flag. However, the flag is automatically cleared if any START condition (S/Sr) is detected.

This flag is intended for systems where address resolution protocol (ARP) is employed. However, a detected collision in non-ARP situations indicates that there has been a protocol violation and should be treated as a bus error.

Bit 2 – BUSERR: Bus Error
The BUSERR flag indicates that an illegal bus condition has occurred. An illegal bus condition is detected if a protocol violating start (S), repeated start (Sr), or stop (P) is detected on the TWI bus lines. A start condition directly followed by a stop condition is one example of protocol violation. Writing a one to its bit location will clear the BUSERR flag. However, normal use of the TWI does not require the BUSERR to be cleared by this method. A robust TWI driver software design will assume that the entire packet of data has been corrupted and restart by waiting for a new START condition (S). For bus errors to be detected, the master must be enabled (ENABLE bit in TWI.MCTRLA is '1'), and the system clock frequency must be at least four times the SCL frequency.

Bit 1 – DIR: Read/Write Direction
This bit is read only and indicates the current bus direction state. The DIR bit reflects the direction bit value from the last address packet received from a master TWI device. If this bit is read as '1', a master read operation is in progress. Consequently a zero indicates that a master write operation is in progress.

Bit 0 – AP: Address or Stop
When the TWI slave address or stop interrupt flag (APIF) is set, this bit determines whether the interrupt is due to address detection or a stop condition.

Value	Name	Description
0	STOP	A stop condition generated the interrupt on APIF.
1	ADR	Address detection generated the interrupt on APIF.

26.8.12. Slave Address

Name: SADDR
Offset: 0x0C
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	ADDR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – ADDR[7:0]: Address
The slave address register in combination with the slave address mask register (TWI.SADDRMASK) is used by the slave address match logic to determine if a master TWI device has addressed the TWI slave. The slave address interrupt flag (APIF) is set to one if the received address is recognized. The slave address match logic supports recognition of 7- and 10-bits addresses, and general call address.

When using 7-bit or 10-bit address recognition mode, the upper 7-bits of the address register (ADDR[7:1]) represents the slave address and the least significant bit (ADDR[0]) is used for general call address recognition. Setting the ADDR[0] bit in this case enables the general call address recognition logic. The TWI slave address match logic only support recognition of the first byte of a 10-bit address i.e. by setting ADDRA[7:1] = “0b11110aa” where “aa” represents bit 9 and 8 or the slave address. The second 10-bit address byte must be handled by software.

26.8.13. Slave Data

Name: SDATA
Offset: 0x0D
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	DATA[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – DATA[7:0]: Data
The slave data register I/O location (DATA) provides direct access to the slaves physical shift register, which is used both to shift data out onto the bus (transmit) and to shift in data received from the bus (receive). The direct access implies that the data register cannot be accessed during byte transmissions. Build-in logic prevents any write accesses to the data register during the shift operations. Reading valid data or writing data to be transmitted can only be successfully done when the bus clock (SCL) is held low by the slave, i.e. when the slave CLKHOLD bit is set. However, it is not necessary to check the CLKHOLD bit in software before accessing the slave DATA register if the software keeps track of the present protocol state by using interrupts or observing the interrupt flags. Accessing the slave DATA register, assumed that clock hold is active, auto-trigger bus operations dependent of the state of the slave acknowledge action command bits (ACKACT) and type of register access (read or write).

8

26.8.14. Slave Address Mask

Name: SADDRMASK
Offset: 0x0E
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	ADDRMASK[6:0]							ADDREN
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:1 – ADDRMASK[6:0]: Address Mask
The ADDRMASK register acts as a second address match register, or an address mask register depending on the ADDREN setting.

If ADDREN is written to '0', ADDRMASK can be loaded with a 7-bit Slave Address mask. Each of the bits in ADDRMASK register can mask (disable) the corresponding address bits in the TWI slave Address Register (ADDR). If the mask bit is written to '1' then the address match logic ignores the compare between the incoming address bit and the corresponding bit in slave ADDR register. In other words, masked bits will always match.

If ADDREN is written to '1', the slave ADDRMASK can be loaded with a second slave address in addition to the ADDR register. In this mode, the slave will match on 2 unique addresses, one in ADDR and the other in ADDRMASK.

Bit 0 – ADDREN: Address Mask Enable
If this bit is written to '1', the slave address match logic responds to the 2 unique addresses in slave ADDR and ADDRMASK.

If this bit is '0', the ADDRMASK register acts as a mask to the slave ADDR register.

27. CRCSCAN - Cyclic Redundancy Check Memory Scan

27.1. Overview

A Cyclic Redundancy Check (CRC) takes a data stream of bytes as input and generates a checksum. The CRC peripheral can be used to detect errors in program memory.

Typically, an n-bit CRC, applied to a data block of arbitrary length, will detect any single error burst not longer than n bits (i.e. any single alteration that spans no more than n bits of the data), and will detect a fraction 1-2-n of all longer error bursts.

The CRC-generator supports CRC-16 (CRC-CCITT).

Polynomial:

- CRC-16: $x^{16} + x^{12} + x^5 + 1$

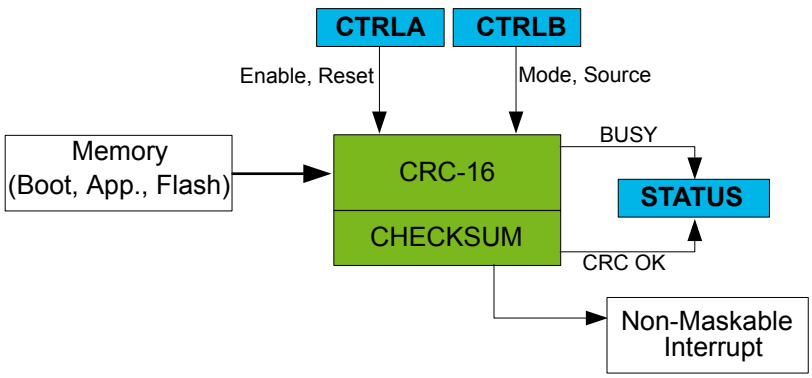
The initial value of the checksum register is 0xFFFF.

27.2. Features

- CRC-16 (CRC-CCITT)
- Can check full Flash, application code and/or boot section
- Single/continuous background or priority check selectable
- Selectable NMI trigger on failure
- User configurable check during startup
- Paused in all CPU sleep modes

27.3. Block Diagram

Figure 27-1. Cyclic Redundancy Check Block Diagram



27.4. Product Dependencies

In order to use this peripheral, other parts of the system must be configured correctly, as described below.

Table 27-1. CRCSCAN Product Dependencies

Dependency	Applicable	Peripheral
Clocks	Yes	CLKCTRL
I/O Lines and Connections	No	-
Interrupts	Yes	CPUINT
Events	No	-
Debug	Yes	UPDI

Related Links

Clocks on page 86
Interrupts on page 54

27.4.1. Clocks

This peripheral depends on the peripheral clock.

Related Links

CLKCTRL - Clock Controller on page 68
Product Dependencies on page 98

27.4.2. I/O Lines and Connections

Not applicable.

27.4.3. Interrupts

Using the interrupts of this peripheral requires the Interrupt Controller to be configured first.

Related Links

CPUINT - CPU Interrupt Controller on page 97
SREG on page 51
Interrupts on page 136

27.4.4. Events

Not applicable.

27.4.5. Debug Operation

Whenever the debugger accesses the device, for instance reading or writing a peripheral or memory location, the CRCSCAN peripheral will be disabled.

If the CRCSCAN is busy when the debugger accesses the device, the CRCSCAN will restart the ongoing operation when the debugger accesses an internal register or when the debugger disconnects.

The Busy bit in the Status register (CRCSCAN_STATUS.BUSY) bit will read '1' if the CRCSCAN was busy when the debugger caused it to disable, but it will not actively check any section as long as the debugger keeps it disabled. There are synchronized CRC status bits in the debugger's internal register space, which can be read by the debugger without disabling the CRCSCAN. Reading the debugger's internal CRC status bits will make sure that the CRCSCAN is enabled.

It is possible to write the STATUS register directly from the debugger:

- CRCSCAN_STATUS.BUSY:
  - Writing the BUSY bit to '0' will stop the ongoing CRC operation (so that the CRCSCAN does not restart its operation when the debugger allows it).

  - Writing the BUSY bit to '1' will make the CRC start a single check with the settings in the Control B register (CRCSCAN_CTRLB), but not until the debugger allows it.

  As long as CRCSCAN_STATUS.BUSY is '1', CRCSCAN_CRCTRLB and the non-maskable interrupt (NMI) bit in the Control A register (CRCSCAN_CTRLA.NMIEN) cannot be altered.
- CRCSCAN_STATUS.OK:
  - Writing the OK bit to '0' can trigger a non-maskable interrupt (NMI) if CRCSCAN_CTRLA.NMIEN is '1'. If an NMI has been triggered, no writes to the CRCSCAN are allowed.
  - Writing the OK bit to '1' will make the OK bit read as '1' when CRCSCAN_STATUS.BUSY is '0'.

Writes to CRCSCAN_CTRLA and CRCSCAN_CTRLB from the debugger are treated in the same way as writes from the CPU.

Related Links

UPDI - Unified Program Debug Interface on page 490
CTRLA on page 415
CTRLB on page 416

27.5. Functional Description

27.5.1. Principle of Operation

The CRCSCAN peripheral takes data input from the NVM (either entire Flash, only Boot section, or both application code and Boot section) and generates a checksum based on the data. The last location in the section to check has to contain the correct pre-calculated 16-bit checksum for comparison. If the checksum calculated by the CRCSCAN and the pre-calculated checksums match, a status bit in the CRCSCAN is set. If they do not match, the status register will indicate that it failed. The user can choose to let the CRCSCAN generate an NMI if the checksums do not match.

27.5.2. Basic Operation

The CRC can run in these modes:

- Continuous mode in the background: the CRC restarts checking from the beginning of the selected section. The CPU can continue executing code and the CRC fetches data when it can.
- Single mode in the background: A single CRC is executed. The CPU can continue executing code and the CRC fetches data when it can.
- Priority mode: the CRC peripheral has priority access to the Flash and will stall the CPU until completed.

27.5.2.1. Initialization

To enable a CRC in application (or via the debugger), write the Control B register (CRCSCAN_CTRLB) to select the desired mode and source settings, then enable the CRCSCAN by writing a '1' to the Enable bit in the Control A register (CRCSCAN_CTRLA.ENABLE).

The CRCSCAN can be enabled during startup to ensure the Flash is OK before letting the CPU execute code. If the CRCSCAN fails during startup, the CPU is not allowed to start. The full source settings are available during startup. See the fuse description in the device datasheet for more information.

If the CRCSCAN was enabled during startup, the CRC Control A,B registers will reflect this after startup: CRCSCAN_CTRLA.ENABLE will be '1', CRCSCAN_CTRLB.MODE will be BACKGROUND, and CRCSCAN_CTRLB.SRC will reflect which section(s) were checked.

27.5.2.2. Checksum

The pre-calculated checksum must be present in the last location of the section to be checked. If the BOOT section should be checked, the 16-bit checksum must be saved in the last two bytes of the BOOT section, and similarly for APPLICATION and entire Flash. Table 27-2 shows explicitly how the checksum should be stored for the different sections. Also see CTRLB for how to configure which section to check and the device fuse description for how to configure the BOOTEND and APPEND fuses.

Table 27-2. How to place the pre-calculated 16-bit checksum in Flash

Section to Check	CHECKSUM[15:8]	CHECKSUM[7:0]
BOOT	FUSE_BOOTEND*256-2	FUSE_BOOTEND*256-1
BOOT and APPLICATION	FUSE_APPEND*256-2	FUSE_APPEND*256-1
Full Flash	FLASHEND-1	FLASHEND

27.5.3. Interrupts

Table 27-3. Available Interrupt Vectors and Sources

Offset	Name	Vector Description	Conditions
0x00	NMI	Non-Maskable Interrupt	Generated on CRC failure

When the interrupt condition occurs, the OK flag in the Status register (CRCSCAN_STATUS.OK) is cleared to '0'.

An interrupt is enabled by writing a '1' to the respective Enable bit in the Control A register (CRCSCAN_CTRLA.NMIEN), but can only be disabled with a system Reset. An NMI is generated when the CRCSCAN_STATUS.OK flag is cleared and the CRCSCAN_CTRLA.NMIEN bit is '1'. The NMI request remains active until a system Reset, and can not be disabled.

Note: A non-maskable interrupt can be triggered even if interrupts are not globally enabled.

27.5.4. Sleep Mode Operation

In all CPU sleep modes, the CRCSCAN peripheral is halted, and will resume operation when the CPU wakes up.

It is possible to enter sleep mode after setting up the CRCSCAN to start a PRIORITY check (see CTRLB for more information), but before the actual check is started. If the CPU is able enter sleep mode before the check starts and a Priority check was configured, the check will start and complete (halting the CPU) immediately after waking up, and before entering any interrupt handler.

27.5.5. Configuration Change Protection

Not applicable.

27.6. Register Summary

Offset	Name	Bit Pos.								
0x00	CTRLA	7:0	RESET						NMIEN	ENABLE
0x01	CTRLB	7:0			MODE[1:0]				SRC[1:0]	
0x02	STATUS	7:0							OK	BUSY

27.7. Register Description

27.7.1. Control A

If an NMI has been triggered, this register is not writable.

Name: CTRLA
Offset: 0x00
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	RESET						NMIEN	ENABLE
Access	R/W						R/W	R/W
Reset	0						0	0

Bit 7 – RESET: Reset CRCSCAN
Writing this bit to '1' resets the CRCSCAN peripheral: The CRCSCAN Control registers and Status register (CTRLA, CTRLB, STATUS) will be cleared one clock cycle after the RESET bit was written to '1'.

If NMIEN is '0', this bit is writable when the CRCSCAN is busy (BUSY bit in CRCSCAN.STATUS is '1') and will take effect immediately.

If NMIEN is '1', this bit is only writable when the CRCSCAN is not busy (BUSY bit in CRCSCAN.STATUS is '0').

The RESET bit is a strobe bit.

Bit 1 – NMIEN: Enable NMI Trigger
When this bit is written to '1', any CRC failure will trigger an NMI.

This can only be cleared by a system Reset - it is not cleared by a write to the RESET bit.

This bit can only be written to '1' when the CRCSCAN is not busy (BUSY bit in CRCSCAN.STATUS is '0').

Bit 0 – ENABLE: Enable CRCSCAN
Writing this bit to '1' enables the CRCSCAN peripheral with the current settings. It will stay '1' even after a CRC check has completed, but writing it to 1 again will start a new check.

Writing the bit to '0' will disable the CRCSCAN after the ongoing check is completed (after reaching the end of the section it is set up to check). This is the preferred way to stop a continuous background check. A failure in the ongoing check will still be detected and can cause an NMI if the NMIEN bit is '1'.

The CRCSCAN can be enabled during startup to verify Flash sections before letting the CPU start (see device datasheet fuse description). If the CRCSCAN is enabled during startup, the ENABLE bit will read as '1' after startup.

To see whether the CRCSCAN peripheral is busy with an ongoing check, poll the Busy bit (BUSY) in the Status register (CRCSCAN.STATUS).

27.7.2. Control B

The CTRLB register contains the mode and source settings for the CRC. It is not writable when the CRC is busy or when an NMI has been triggered.

Name: CTRLB
Offset: 0x01
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
			MODE[1:0]				SRC[1:0]	
Access			R/W	R/W			R/W	R/W
Reset			0	0			0	0

Bits 5:4 – MODE[1:0]: CRC Flash Access Mode

The MODE bit field selects the priority of the CRC module in the system when accessing Flash, either in the background or completely stalling the CPU until the CRC is completed. It is also possible to select the CONTINUOUS mode, which will make the CRC module restart at the beginning of the selected section after finishing a section. The CONTINUOUS mode will stop if a failure occurs. To otherwise stop a CONTINUOUS mode, write the ENABLE or RESET bit in the CRCSCAN.CTRLA register.
The CRC can be enabled during startup to verify Flash sections before letting the CPU start (see device datasheet fuse description). If the CRC is enabled during startup, the MODE bit field will read out as BACKGROUND after startup.

Table 27-4. Modes of CRC Flash Access

MODE[1:0]	Group Configuration	Description
0x0	PRIORITY	The CRC module runs a single check with priority to Flash. The CPU is halted until the CRC completes.
0x1	Reserved	This mode is reserved and should not be selected.
0x2	BACKGROUND	The CRC module runs a single check with lowest priority to Flash.
0x3	CONTINUOUS	The CRC module runs continuous checks in the background. After completing a successful check it restarts.

Bits 1:0 – SRC[1:0]: CRC Source

The SRC bit field selects which section of the Flash the CRC module should check. To set up section sizes, please refer to the fuse description.
The CRC can be enabled during startup to verify Flash sections before letting the CPU start (see device datasheet fuse description). If the CRC is enabled during startup, the SRC bit field will read out as FLASH, APPLICATION or BOOT after startup (depending on the configuration).

Table 27-5. CRC Sources

SRC[1:0]	Group Configuration	Description
0x0	FLASH	The CRC is performed on the entire Flash (boot, application code and application data sections).
0x1	APPLICATION	The CRC is performed on the boot and application code sections of Flash.

SRC[1:0]	Group Configuration	Description
0x2	BOOT	The CRC is performed on the boot section of Flash.
0x3	Reserved	This source configuration is reserved and should not be selected.

27.7.3. Status

The status register contains the busy and OK information. It is not writable, only readable.

Name: STATUS
Offset: 0x02
Reset: 0x02
Property: -

Bit	7	6	5	4	3	2	1	0
							OK	BUSY
Access							R	R
Reset							1	0

Bit 1 – OK: CRC OK

When this bit is read as 1, the previous CRC completed successfully. The bit is set to '1' from Reset, but is cleared to '0' when enabling. As long as the CRC module is busy, it will be read '0'. When running continuously, the CRC status must be assumed OK until it fails or is stopped by the user.

Bit 0 – BUSY: CRC Busy

When this bit is read as 1, the CRC module is busy. As long as the module is busy, the access to the control registers are limited. See CTRLA and CTRLB for more information.

28. CCL – Configurable Custom Logic

28.1. Overview

The Configurable Custom Logic (CCL) is a programmable logic peripheral which can be connected to the device pins, to events, or to other internal peripherals. This allows the user to eliminate logic gates for simple glue logic functions on the PCB.

Each Lookup Table (LUT) consists of three inputs, a truth table, and as options synchronizer, filter and edge detector. Each LUT can generate an output as a user programmable logic expression with three inputs. Inputs can be individually masked.

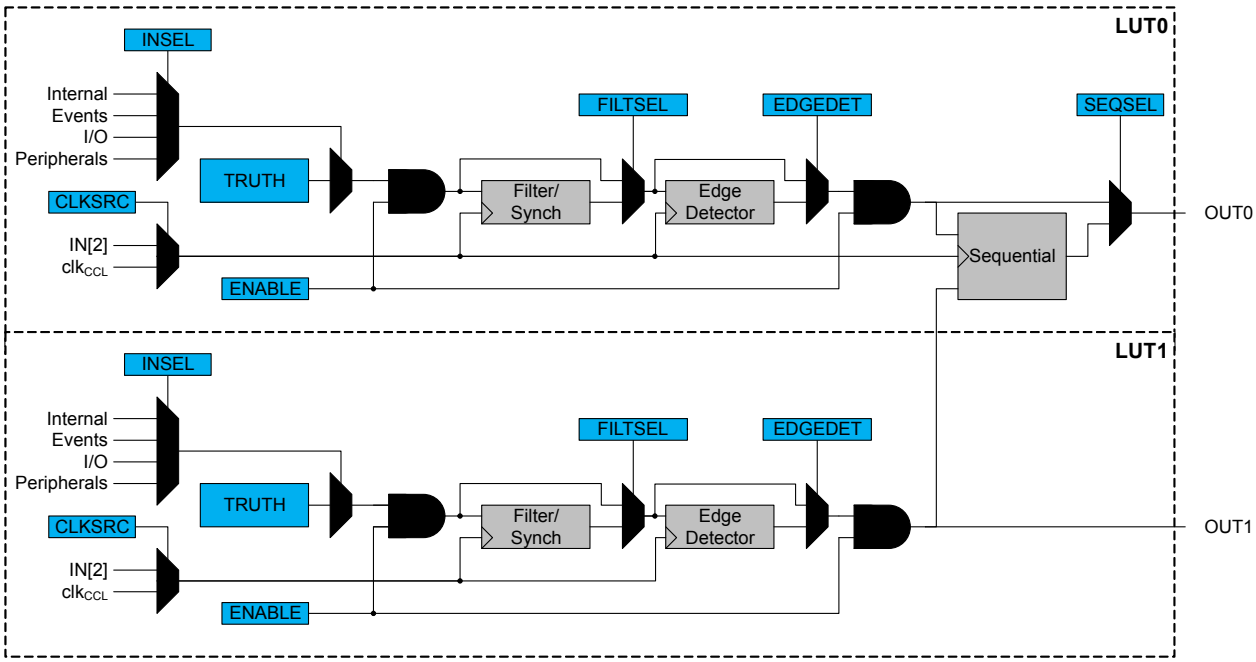
The output can be combinatorially generated from the inputs, and can be filtered to remove spikes. An optional sequential module can be enabled. The inputs of the sequential module are individually controlled by two independent, adjacent LUT (LUT0/LUT1) outputs, enabling complex waveform generation.

28.2. Features

- Glue logic for general purpose PCB design
- Up to two Programmable LookUp Tables LUT[1:0]
- Combinatorial Logic Functions: Any logic expression which is a function of up to three inputs.
- Sequential Logic Functions:
  Gated D Flip-Flop, JK Flip-Flop, gated D Latch, RS Latch
- Flexible LookUp Table Inputs Selection:
  - I/Os
  - Events
  - Internal Peripherals
  - Subsequent LUT Output
  - Analog Comparator
  - Timers
  - Power Stage Controller
  - USART
  - SPI
- Output can be connected to IO pins or Event System
- Optional synchronizer, filter, or edge detector available on each LUT output

28.3. Block Diagram

Figure 28-1. Configurable Custom Logic



28.4. Signal Description

Pin Name	Type	Description
LUTn-OUT	Digital output	Output from lookup table
LUTn-IN[2:0]	Digital input	Input to lookup table

Refer to *I/O Multiplexing and Considerations* for details on the pin mapping for this peripheral. One signal can be mapped on several pins.

28.5. Product Dependencies

In order to use this peripheral, other parts of the system must be configured correctly, as described below.

Table 28-1. CCL Product Dependencies

Dependency	Applicable	Peripheral
Clocks	Yes	CLKCTRL
I/O Lines and Connections	Yes	PORT
Interrupts	Yes	CPUINT
Events	No	EVSYS
Debug	Yes	UPDI

28.5.1. Clocks

The CCL is using the peripheral clock of the device (CLK_PER).

Optionally, one of the peripheral inputs that are available on input selection line 2, can be used to clock the CCL. This is configured by writing a '1' to the Clock Source Selection bit in the LUT Control n A register (CCL_LUTCTRLnA.CLKSRC). The sequential block is clocked by the same clock as that of its even LUT in the LUT pair (SEQn.clk = LUT2n.clk). It is advised to disable the peripheral by writing a '0' to the Enable bit in Control A register (CCL_CTRLA.ENABLE) before configuring CCL_LUTCTRLnA.CLKSRC.

28.5.2. I/O Lines

Using the CCL I/O lines requires the I/O pins to be configured. Refer to *PORT - I/O Pin Controller* for details.

Related Links

PORT - I/O Pin Controller on page 132

28.5.3. Interrupts

Not applicable.

28.5.4. Events

The events are connected to the Event System. Refer to *EVSYS – Event System* for details on how to configure the Event System.

Related Links

EVSYS - Event System on page 109

28.5.5. Debug Operation

When the CPU is halted in debug mode the CCL continues normal operation. If the CCL is configured in a way that requires it to be periodically serviced by the CPU through interrupts or similar, improper operation or data loss may result during debugging.

28.6. Functional Description

28.6.1. Principle of Operation

Configurable Custom Logic (CCL) is a programmable logic block that can use the device port pins, internal peripherals, and the internal Event System as both input and output channels. The CCL can serve as glue logic between the device and external devices. This increases the reliability of the PCB by reducing its complexity, and enables more powerful functions.

28.6.2. Basic Operation

28.6.2.1. Initialization

The following bits are enable-protected, meaning that they can only be written when the corresponding even LUT is disabled (CCL_LUTCTRL0A.ENABLE=0):

- Sequential Selection in Sequential Control 0 register (CCL_SEQCTRL0.SEQSEL)

The following registers are enable-protected, meaning that they can only be written when the corresponding LUT is disabled (CCL_LUTCTRL0A.ENABLE=0):

- LUT Control x register, except ENABLE bit (CCL_LUTCTRLxn)

Enable-protected bits in the CCL_LUTCTRLxn registers can be written at the same time as CC_LUTCTRLxA.ENABLE is written to '1', but not at the same time as CCL_LUTCTRLxA.ENABLE is written to '0'.

Enable-protection is denoted by the Enable-Protected property in the register description.

28.6.2.2. Enabling, Disabling and Resetting

The CCL is enabled by writing a '1' to the Enable bit in the Control register (CCL_CTRLA.ENABLE). The CCL is disabled by writing a '0' to CCL_CTRLA.ENABLE.

Each LUT is enabled by writing a '1' to the LUT Enable bit in the LUT Control x A register (CCL_LUTCTRLxA.ENABLE). Each LUT is disabled by writing a '0' to CCL_LUTCTRLxA.ENABLE.

28.6.2.3. Lookup Table Logic

The lookup table in each LUT unit can generate any logic expression OUT as a function of three inputs (IN[2:0]), as shown below. One or more inputs can be masked. The truth table for the expression is defined by the TRUTH bits in the Truth n registers (CCL_TRUTHn.TRUTH).

Figure 28-2. Truth Table Output Value Selection

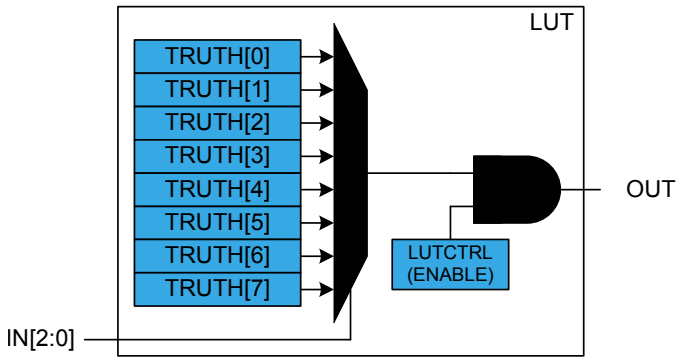


Table 28-2. Truth Table of LUT

IN[2]	IN[1]	IN[0]	OUT
0	0	0	TRUTH[0]
0	0	1	TRUTH[1]
0	1	0	TRUTH[2]
0	1	1	TRUTH[3]
1	0	0	TRUTH[4]
1	0	1	TRUTH[5]
1	1	0	TRUTH[6]
1	1	1	TRUTH[7]

28.6.2.4. Truth Table Inputs Selection

Input Overview

The inputs can be individually:

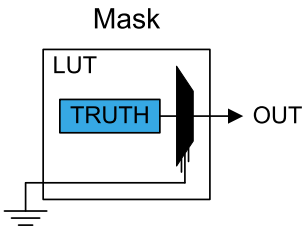
- Masked
- Driven by peripherals:
  - Analog comparator output (AC)
  - Timer/Counters waveform outputs (TC)
  - Power Stage Controller outputs
  - Serial Communication output transmit interface (SERCOM)

- Driven by internal events from Event System
- Driven by other CCL sub-modules

The Input Selection for each input y of LUT x is configured by writing the Input y Source Selection bit in the LUT x Control n=[B,C] registers (CCL_LUTCTRLxB.INSEL0, CCL_LUTCTRLxB.INSEL1, and CCL_LUTCTRLxC.INSEL2).

Masked Inputs (MASK)
When a LUT input is masked (CCL_LUTCTRLxn.INSELy=MASK), the corresponding TRUTH input (IN) is internally tied to zero, as shown in this figure:

Figure 28-3. Masked Input Selection



Internal Feedback Inputs (FEEDBACK)
When selected (CCL_LUTCTRLxn.INSELy=FEEDBACK), the Sequential (SEQ) output is used as input for the corresponding LUT.

The output from an internal sequential sub-module can be used as input source for the LUT, see figure below for an example for LUT0 and LUT1. The sequential selection for each LUT follows the formula:

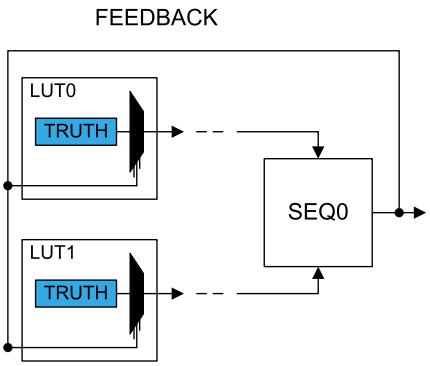
$\mathrm{IN}[2\mathrm{N}][i] = \mathrm{SEQ}[N]$

$\mathrm{IN}[2\mathrm{N}+1][i] = \mathrm{SEQ}[N]$

With *N* representing the sequencer number and *i*=0,1,2 representing the LUT input index.

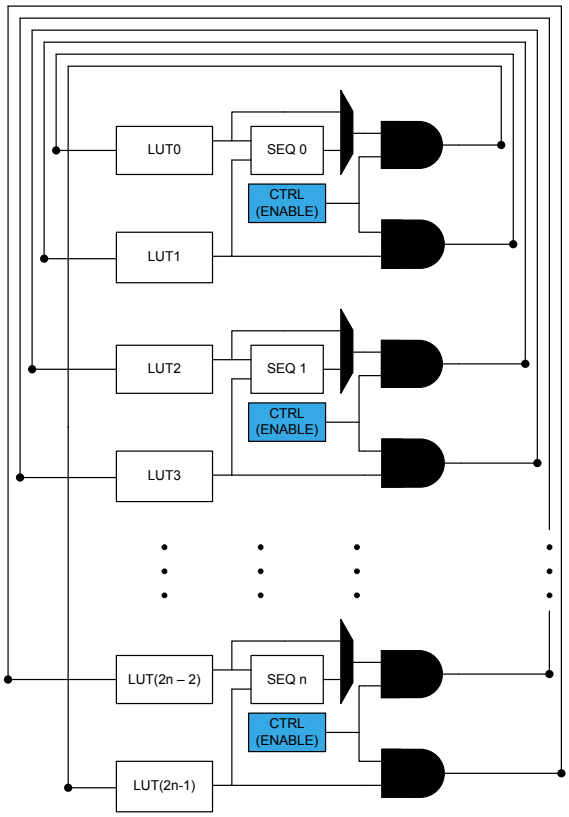
For details, refer to Sequential Logic.

Figure 28-4. Feedback Input Selection



Linked LUT (LINK)
When selected (CCL_LUTCTRLxn.INSELy=LINK), the subsequent LUT output is used as the LUT input (e.g., LUT2 is the input for LUT1), as shown in this figure:

Figure 28-5. Linked LUT Input Selection



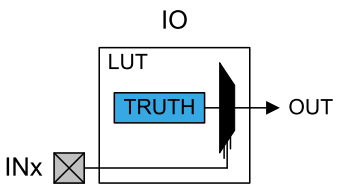
Internal Events Inputs Selection(EVENT)

Asynchronous events from the Event System can be used as input selection. For each LUT, two event input lines are available and can be selected on each LUT input. Before enabling the event selection by writing CCL_LUTCTRLxn.INSELy=EVENT, the Event System must be configured first.

I/O Pin Inputs (IO)

When the IO pin is selected as LUT input (CC_LUTCTRLxn.INSELy=IO), the corresponding LUT input will be connected to the pin, as shown in the figure below.

Figure 28-6. I/O Pin Input Selection



Peripherals

The different peripherals on the three input lines of each LUT are selected by writing to the respective LUT n Input x bit fields in the LUT n Control B and C registers (CCL_LUTnCTRLB.INSEL0, CCL_LUTnCTRLB.INSEL1, CCL_LUTnCTRLC.INSEL2).

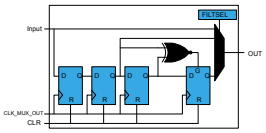
28.6.2.5. Filter

By default, the LUT output is a combinatorial function of the LUT inputs. This may cause some short glitches when the inputs change value. These glitches can be removed by clocking through filters, if demanded by application needs.

The Filter Selection bits in the LUT Control registers (CCL_LUTCTRLxA.FILTSEL) define the synchronizer or digital filter options. When a filter is enabled, the OUT output will be delayed by two to five CLK cycles (system clock or custom clock, based on CCL_LUTCTRLxA.CLKSRC settings). One clock cycle after the corresponding LUT is disabled, all internal filter logic is cleared.

Note: Events used as LUT input will also be filtered, if the filter is enabled.

Figure 28-7. Filter



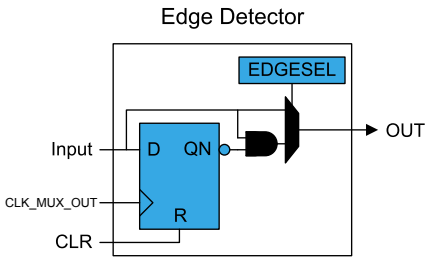
28.6.2.6. Edge Detector

The edge detector can be used to generate a pulse when detecting a rising edge on its input. To detect a falling edge, the TRUTH table should be programmed to provide the opposite levels.

The edge detector is enabled by writing '1' to the Edge Selection bit in the LUT x Control A register (CC_LUTCTRLxA.EDGEDET). In order to avoid unpredictable behavior, a valid filter option must be enabled as well.

Edge detection is disabled by writing a '0' to CC_LUTCTRLxA.EDGEDET. After disabling a LUT, the corresponding internal Edge Detector logic is cleared one clock cycle later.

Figure 28-8. Edge Detector



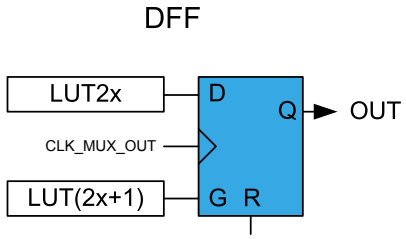
28.6.2.7. Sequential Logic

Each LUT pair can be connected to internal sequential logic: D flip flop, JK flip flop, gated D-latch or RS-latch can be selected by writing the Sequential Selection bits in Sequential Control 0 register (CCL_SEQCTRL0.SEQSEL). Before using sequential logic, the clock source (CCL_LUTCTRLxA.CLKSRC) and the LUT filters and edge detectors can be configured optionally by writing to the CCL_LUTCTRLxA register. The sequential logic is clocked by the peripheral clock if CCL_LUTCTRLxA.CLKSRC=0 or by a peripheral input if .CLKSRC=1.

Gated D Flip-Flop (DFF)

When the DFF is selected, the D-input is driven by the even LUT output (LUT0), and the G-input is driven by the odd LUT output (LUT2x+1).

Figure 28-9. D Flip Flop



When the even LUT is disabled (CCL_LUTCTRL0.ENABLE=0), the flip-flop is asynchronously cleared. The reset command (R) is kept enabled for one clock cycle. In all other cases, the flip-flop output (OUT) is refreshed on rising edge of the clock, as shown in the table below.

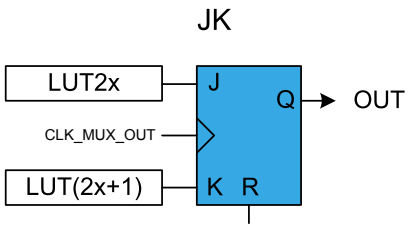
Table 28-3. DFF Characteristics

R	G	D	OUT
1	X	X	Clear
0	1	1	Set
		0	Clear
	0	X	Hold state (no change)

JK Flip-Flop (JK)

When this configuration is selected, the J-input is driven by the even LUT output (LUT0), and the K-input is driven by the odd LUT output (LUT1).

Figure 28-10. JK Flip Flop



When the even LUT is disabled (CCL_LUTCTRL0.ENABLE=0), the flip-flop is asynchronously cleared. The reset command (R) is kept enabled for one clock cycle. In all other cases, the flip-flop output (OUT) is refreshed on rising edge of the clock as showed in the table below.

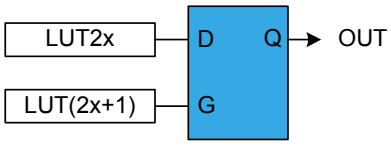
Table 28-4. JK Characteristics

R	J	K	OUT
1	X	X	Clear
0	0	0	Hold state (no change)
0	0	1	Clear
0	1	0	Set
0	1	1	Toggle

Gated D-Latch (DLATCH)

When the DLATCH is selected, the D-input is driven by the even LUT output (LUT2x), and the G-input is driven by the odd LUT output (LUT2x+1).

Figure 28-11. D-Latch



When the even LUT is disabled (CCL_LUTCTRL0.ENABLE=0), the latch output will be cleared. The G-input is forced enabled for one more clock cycle, and the D-input to zero. In all other cases, the latch output (OUT) is refreshed as shown in the table below.

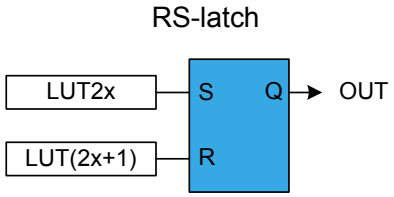
Table 28-5. D-Latch Characteristics

G	D	OUT
0	X	Hold state (no change)
1	0	Clear
1	1	Set

RS Latch (RS)

When this configuration is selected, the S-input is driven by the even LUT output (LUT0), and the R-input is driven by the odd LUT output (LUT1).

Figure 28-12. RS-Latch



When the even LUT is disabled (CCL_LUTCTRL0.ENABLE=0), the latch output will be cleared. The R-input is forced enabled for one more clock cycle and S-input to zero. In all other cases, the latch output (OUT) is refreshed as shown in the table below.

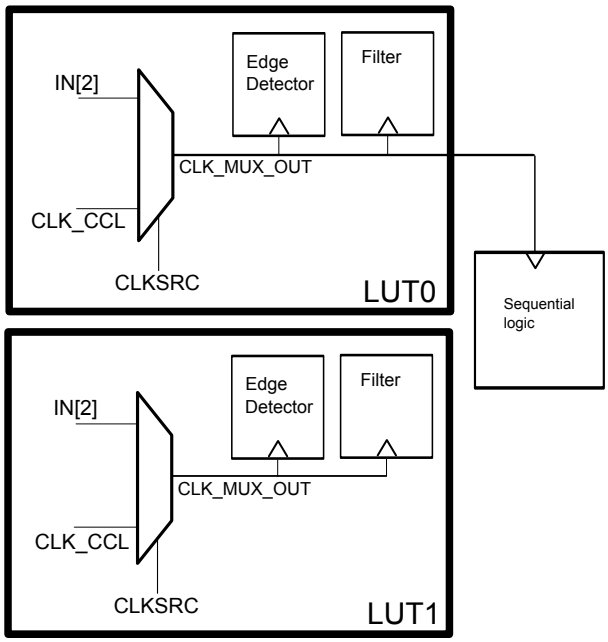
Table 28-6. RS-latch Characteristics

S	R	OUT
0	0	Hold state (no change)
0	1	Clear
1	0	Set
1	1	Forbidden state

28.6.2.8. Clock Source Settings

The Filter, Edge Detector and Sequential logic are clocked on the system clock (CLK_CCL) by default. It is also possible to employ one of the peripheral outputs available on the LUT input selection line 2 to clock, CLK_MUX_OUT, these blocks. This can be achieved by writing the CCL_LUTCTRLx.CLKSRC bit to '1'.

Figure 28-13. Clock Source Settings



When .CLKSRC is '1', the peripheral custom clock, CLK_MUX_OUT, clocks its corresponding Filter and Edge Detector. The Sequential logic is clocked by the same clock as that of its even LUT in the pair. The input selection line 2 will be treated as MASK to the truth table when .CLKSRC is '1'.

It is recommended to disable the CCL peripheral before configuring this setting to avoid undetermined outputs from the peripheral.

28.6.3. Events

The CCL can generate the following output events:

- LUTOUTx: Lookup Table Output Value

The CCL can take the following actions on an input event:

- INx: The event is used as input for the TRUTH table.

28.6.4. Sleep Mode Operation

Writing the Run In Standby bit in the Control A register (CCL_CTRLA.RUNSTDBY) to '1' will allow the system clock to be enabled in Standby sleep mode.

If CCL_CTRLA.RUNSTDBY=0, the system clock will be disabled. If the Filter, Edge Detector or Sequential logic are enabled, the LUT output will be forced to zero in Standby sleep mode. In all other cases, the TRUTH table decoder will continue operation and the LUT output will be refreshed accordingly.

If CCL_LUTCTRLxA.CLKSRC=1, then the peripheral on input selection line 2 will always clock the Filter, Edge Detector and Sequential logic. The availability of this custom clock in sleep modes will depend on the sleep settings of the peripheral employed.

28.6.5. Configuration Change Protection

Not applicable.

28.7. Register Summary

Offset	Name	Bit Pos.								
0x00	CTRLA	7:0		RUNSTDBY						ENABLE
0x01	SEQCTRL0	7:0					SEQSEL[3:0]			
0x02 ... 0x04	Reserved									
0x05	LUTCTRLA0	7:0	EDGEDET	CLKSRC	FILTSEL[1:0]		OUTEN			ENABLE
0x06	LUTCTRLB0	7:0	INSEL1[3:0]				INSEL0[3:0]			
0x07	LUTCTRLC0	7:0					INSEL2[3:0]			
0x08	TRUTH0	7:0	TRUTH[7:0]							
0x09	LUTCTRLA1	7:0	EDGEDET	CLKSRC	FILTSEL[1:0]		OUTEN			ENABLE
0x0A	LUTCTRLB1	7:0	INSEL1[3:0]				INSEL0[3:0]			
0x0B	LUTCTRLC1	7:0					INSEL2[3:0]			
0x0C	TRUTH1	7:0	TRUTH[7:0]							

28.8. Register Description

28.8.1. Control A

Name: CTRLA
Offset: 0x00
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
		RUNSTDBY						ENABLE
Access		R/W						R/W
Reset		0						0

Bit 6 – RUNSTDBY: Run in Standby
This bit indicates if the system clock is kept running in Standby sleep mode. The setting is ignored for configurations where the generic clock is not required. For details refer to Sleep Mode Operation.

Value	Description
0	System clock is not required in standby sleep mode.
1	System clock is required in standby sleep mode.

Bit 0 – ENABLE: Enable

Value	Description
0	The peripheral is disabled.
1	The peripheral is enabled.

28.8.2. Sequential Control 0

Name: SEQCTRL0
Offset: 0x01
Reset: 0x00
Property: Enable-Protected

Bit	7	6	5	4	3	2	1	0
					SEQSEL[3:0]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bits 3:0 – SEQSEL[3:0]: Sequential Selection
These bits select the sequential configuration.

Value	Name	Description
0x0	DISABLE	Sequential logic is disabled
0x1	DFF	D flip flop
0x2	JK	JK flip flop
0x3	LATCH	D latch
0x4	RS	RS latch
Other	-	Reserved

28.8.3. LUT n Control A

Name: LUTCTRLA0, LUTCTRLA1
Offset: 0x05 + n*0x04 [n=0..1]
Reset: 0x00
Property: Enable-Protected

Bit	7	6	5	4	3	2	1	0
	EDGEDET	CLKSRC	FILTSEL[1:0]		OUTEN			ENABLE
Access	R/W	R/W	R/W	R/W	R/W			R/W
Reset	0	0	0	0	0			0

Bit 7 – EDGEDET: Edge Detection

Value	Description
0	Edge detector is disabled.
1	Edge detector is enabled.

Bit 6 – CLKSRC: Clock Source Selection
When this bit is set, any input peripheral present on input line 2 can be used to clock its corresponding LUT. The clock employed by the even LUT in the LUT pair will also clock the sequential block. (SEQx.clk = LUT(2x).clk) When this bit is clear, by default, system clock is employed to clock the CCL module(both LUT and sequential block).

Bits 5:4 – FILTSEL[1:0]: Filter Selection
These bits select the LUT output filter options:

Filter Selection

Value	Name	Description
0x0	DISABLE	Filter disabled
0x1	SYNCH	Synchronizer enabled
0x2	FILTER	Filter enabled
0x3	-	Reserved

Bit 3 – OUTEN: Output Enable
Enabling this bit qualifies the CCL outputs going to the pads. This bit is also employed as override enable to the ports.

Bit 0 – ENABLE: LUT Enable

Value	Description
0	The LUT is disabled.
1	The LUT is enabled.

28.8.4. LUT n Control B

Note:

- SPI connections to the CCL work only in master SPI mode
- USART connections to the CCL work only in asynchronous/synchronous USART master mode.

Name: LUTCTRLB0, LUTCTRLB1
Offset: 0x06 + n*0x04 [n=0..1]
Reset: 0x00
Property: Enable-Protected

Bit	7	6	5	4	3	2	1	0
	INSEL1[3:0]				INSEL0[3:0]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:4 – INSEL1[3:0]: LUT n Input 1 Source Selection
These bits select the source for input 1 of LUT n:

Value	Name	Description
0x0	MASK	Masked input
0x1	FEEDBACK	Feedback input
0x2	LINK	Linked other LUT as input source
0x3	EVENT0/EVENT1	Event input source 0 (for LUT0) / 1 (for LUT1)
0x4	EVENT2/EVENT3	Event input source 2/3 LUT0/LUT1
0x5	IO	I/O pin LUTn-IN1 input source
0x6	AC0	ACOUT input source
0x7	TCB0	TCB W1 input source
0x8	TCA0	TCA W0 input source
0x9	TCD0	TCDOUTB input source
0xA	USART0	USART TXD input source
0xB	SPI0	SPI MOSI input source

Bits 3:0 – INSEL0[3:0]: LUT n Input 0 Source Selection
These bits select the source for input 0 of LUT n:

Value	Name	Description
0x0	MASK	Masked input
0x1	FEEDBACK	Feedback input
0x2	LINK	Linked other LUT as input source
0x3	EVENT0/EVENT1	Event input source 0 (for LUT0) / 1 (for LUT1)
0x4	EVENT2/EVENT3	Event input source 2/3 for LUT0/LUT1

Value	Name	Description
0x5	IO	I/O pin LUTn-IN0 input source
0x6	AC0	ACOUT input source
0x7	TCB0	TCB W0 input source
0x8	TCA0	TCA W0 input source
0x9	TCD0	TCDOUTA input source
0xA	USART0	USART XCK input source
0xB	SPI0	SPI SCK input source
Other	-	Reserved

28.8.5. LUT n Control C

Name: LUTCTRLC0, LUTCTRLC1
Offset: 0x07 + n*0x04 [n=0..1]
Reset: 0x00
Property: Enable-Protected

Bit	7	6	5	4	3	2	1	0
					INSEL2[3:0]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bits 3:0 – INSEL2[3:0]: LUT n Input 2 Source Selection
These bits select the source for input 2 of LUT n:

Value	Name	Description
0x0	MASK	Masked input
0x1	FEEDBACK	Feedback input
0x2	LINK	Linked other LUT as input source
0x3	EVENT0	Event input source 0
0x4	EVENT1	Event input source 1
0x5	IO	I/O pin LUTn-IN2 input source
0x6	AC0	ACOUT input source
0x7	TCB0	TCB W2 input source
0x8	TCA0	TCA W0 input source
0x9	TCD0	TCDOUTC input source
other	-	Reserved

28.8.6. TRUTHn

Name: TRUTH0, TRUTH1
Offset: 0x08 + n*0x04 [n=0..1]
Reset: 0x00
Property: Enable-Protected

Bit	7	6	5	4	3	2	1	0
	TRUTH[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – TRUTH[7:0]: Truth Table
These bits define the value of truth logic as a function of inputs IN[2:0].

29. AC – Analog Comparator

29.1. Overview

The Analog Comparator (AC) compares the voltage levels on two inputs and gives a digital output based on this comparison. The AC can be configured to generate interrupt requests and/or Events upon several different combinations of input change.

The dynamic behavior of the AC can be adjusted by a hysteresis feature. The hysteresis can be customized to achieve optimal operation for each application.

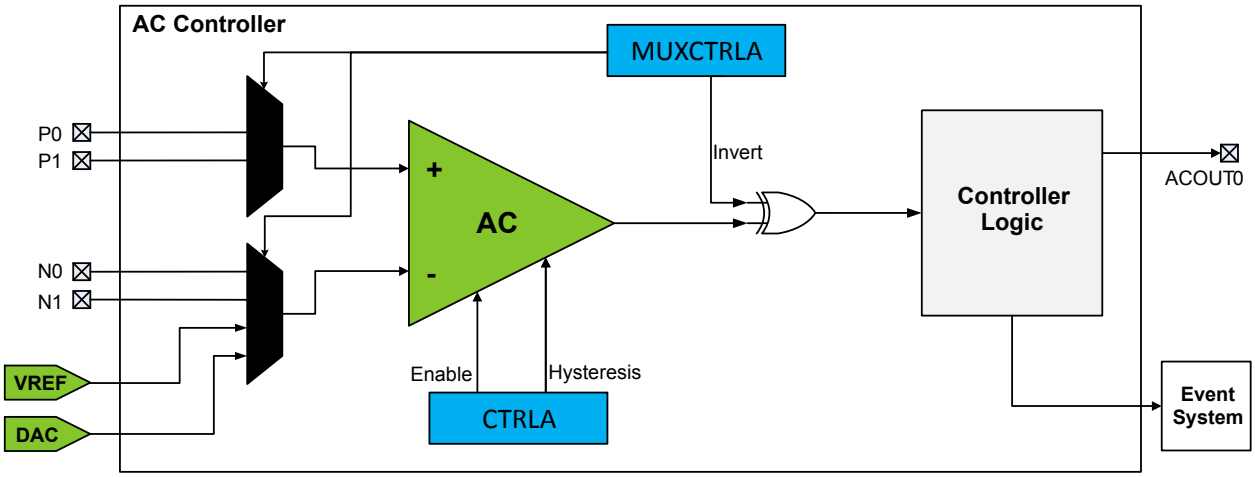
The input selection includes analog port pins, internal references and DAC output. The analog comparator output state can also be output on a pin for use by external devices.

29.2. Features

- 50ns response time for supply voltage above 2.7V
- Zero-cross detection
- Selectable hysteresis:
  - None
  - 10mV
  - 25mV
  - 50mV
- Analog comparator output available on pin
- Invert comparator output
- Flexible input selection:
  - 2 Positive pins
  - 2 Negative pins
  - Output from the DAC
  - Internal reference voltage
- Interrupt generation on:
  - Rising edge
  - Falling edge
  - Toggle
- Event generation on:
  - Comparator output

29.3. Block Diagram

Figure 29-1. Analog Comparator



29.4. Signal Description

Signal	Description	Type
N0	Negative Input 0	Analog
N1	Negative Input 1	Analog
P0	Postive Input 0	Analog
P1	Positive Input 1	Analog
ACOUT	Comparator Output for AC	Digital

Related Links

I/O Multiplexing and Considerations on page 19

29.5. Product Dependencies

In order to use this peripheral, other parts of the system must be configured correctly, as described below.

Table 29-1. AC Product Dependencies

Dependency	Applicable	Peripheral
Clocks	Yes	CLKCTRL
I/O Lines and Connections	Yes	PORT
Interrupts	Yes	CPUINT
Events	Yes	EVSYS
Debug	Yes	UPDI

Related Links

Clocks on page 86

I/O Lines and Connections on page 439
Interrupts on page 54
Events on page 181
Debug Operation on page 439

29.5.1. Clocks

This peripheral depends on the peripheral clock.

Related Links
CLKCTRL - Clock Controller on page 68
Product Dependencies on page 98

29.5.2. I/O Lines and Connections

I/O pins N0-N1, P0-P1 are all analog inputs to the AC.

For correct operation, the pins must be configured in the Port and Port Multiplexing peripherals.

Note: It is recommended to disable the GPIO when using the AC.

Related Links
I/O Multiplexing and Considerations on page 19

29.5.3. Interrupts

Using the interrupts of this peripheral requires the Interrupt Controller to be configured first.

Related Links
CPUINT - CPU Interrupt Controller on page 97
SREG on page 51
Interrupts on page 136

29.5.4. Events

The events of this peripheral are connected to the Event System.

Related Links
EVSYS - Event System on page 109

29.5.5. Debug Operation

This peripheral is unaffected by entering debug mode.

If the peripheral is configured to require periodical service by the CPU through interrupts or similar, improper operation or data loss may result during halted debugging.

Related Links
UPDI - Unified Program Debug Interface on page 490

29.6. Functional Description

29.6.1. Principle of Operation

The AC has one positive input and one negative input. The positive input source is one of a selection of analog input pins. The negative input is chosen either from analog input pins or from internal inputs, such as an internal voltage reference.

The digital output from the comparator is '1' when the difference between the positive and the negative input voltage is positive, and '0' otherwise.

29.6.2. Basic Operation

29.6.2.1. Initialization

For basic operation, follow these steps:

- Configure the desired input pins in the PORT peripheral.
- Select the positive and negative input sources by writing the Positive and Negative Input MUX Selection bit fields in the Mux Control A register (AC_MUXCTRLA.MUXPOS and .MUXNEG, respectively).
- Optional: Enable the output to pin by writing a '1' to the Output Pad Enable bit in the Control A register (AC_CTRLA.OUTEN).
- Enable the AC by writing a '1' to the Enable bit in Control A (AC_CTRLA.ENABLE).

29.6.2.2. Input Hysteresis

Applying an input hysteresis helps preventing constant toggling of the output when the noise-afflicted input signals are close to each other.

The input hysteresis can either be disabled or have one of three levels. The hysteresis is configured by writing to the Hysteresis Mode Select bit field in the Control A register (AC_CTRLA.HYSMODE).

29.6.2.3. Input Sources

The AC has one positive and one negative input. The inputs can be pins and internal sources, such as a voltage reference.

Each input is selected by writing to the Positive and Negative Input MUX Selection bit field in the MUX Control A register (AC_MUXTRLA.MUXPOS and .MUXNEG).

Pin Inputs

The following Analog input pins on the port can be selected as input to the analog comparator

- N0
- N1
- P0
- P1

Internal Inputs

Two internal inputs are available for the analog comparator:

- Output from the DAC
- Bandgap reference voltage

29.6.2.4. Low Speed Mode

For power-conscious applications, the AC provides a Low Speed Mode. When this mode is enabled, the current through the comparator is reduced, resulting in a reduced power consumption.

This mode is enabled by writing a '1' to the Low Speed Mode bit in the Control A register (AC_CTRLA.LSMODE).

Note: Enabling the Low Speed Mode will increase the latency of the AC.

29.6.3. Events

The AC can generate the following output Events:

- Comparator

The output Events are enabled if the AC is enabled. There are no dedicated enables for output Events.

29.6.4. Interrupts

Table 29-2. Available Interrupt Vectors and Sources

Offset	Name	Vector Description	Conditions
0x00	COMP0	Analog comparator interrupt	AC output is toggling as configured in AC_CTRLA.INTMODE

When an interrupt condition occurs, the corresponding Interrupt Flag is set in the Status register of the peripheral (AC_STATUS).

An interrupt source is enabled or disabled by writing to the corresponding bit in the peripheral's Interrupt Control register (AC_INTCTRL).

An interrupt request is generated when the corresponding interrupt source is enabled and the Interrupt Flag is set. The interrupt request remains active until the Interrupt Flag is cleared. See the AC_STATUS register for details on how to clear Interrupt Flags.

Note: Interrupt requests are only generated when Interrupts are enabled globally.

Related Links

CPUINT - CPU Interrupt Controller on page 97
SREG on page 51

29.6.5. Sleep Mode Operation

The AC will continue to operate in all sleep modes in which CLK_PER is available.

In Standby sleep mode, the AC and the output to pad are disabled by default. If the Run in Standby Sleep Mode bit in the Control A register (AC_CTRLA.RUNSTDBY) is written to '1', the AC will continue to operate, but the Status register will not be updated, and no Interrupts are generated. The AC output will be available to the Event System and at pad output, if enabled.

In Power Down sleep mode, the AC and the output to pad are disabled.

29.6.6. Configuration Change Protection

Not applicable.

29.7. Register Summary

Offset	Name	Bit Pos.								
0x00	CTRLA	7:0	RUNSTDBY	OUTEN	INTMODE[1:0]		LSMODE	HYSMODE[1:0]		ENABLE
0x01	Reserved									
0x02	MUXCTRLA	7:0	INVERT				MUXPOS		MUXNEG[1:0]	
0x03 ... 0x05	Reserved									
0x06	INTCTRL	7:0								CMP
0x07	STATUS	7:0				STATE				COMP

29.8. Register Description

29.8.1. Control A

Name: CTRLA
Offset: 0x00
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	RUNSTDBY	OUTEN	INTMODE[1:0]		LSMODE	HYSMODE[1:0]		ENABLE
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 7 – RUNSTDBY: Run in Standby Mode
Writing a '1' to this bit allows the AC to continue operation in Standby sleep mode. Since the clock is stopped, interrupts and status flags are not updated.

Value	Description
0	In Standby sleep mode, the peripheral is halted
1	In Standby sleep mode, the peripheral continues operation

Bit 6 – OUTEN: Analog Comparator Output Pad Enable
Writing this bit to '1' makes ACOUT0 available on the pin.

Bits 5:4 – INTMODE[1:0]: Interrupt Modes
Writing to these bits selects at what edge(s) of the AC output an interrupt request is triggered.

Value	Name	Description
0x0	BOTHEDGE	both negative and positive edge
0x1	-	Reserved
0x2	NEGEDGE	Negative edge
0x3	POSEDGE	Positive edge

Bit 3 – LSMODE: Low Speed Mode
Writing a '1' to this bit reduces the current through the comparator. This reduces power consumption, but increases the reaction time of the AC.

Value	Description
0	Low Speed Mode disabled
1	Low Speed Mode enabled

Bits 2:1 – HYSMODE[1:0]: Hysteresis Mode Select
Writing these bits selects the hysteresis mode for the AC input.

Value	Name	Description
0x0	OFF	OFF
0x1	10	±10mV

Value	Name	Description
0x2	25	±25mV
0x3	50	±50mV

Bit 0 – ENABLE: Enable AC
Writing this bit to '1' enables the AC.

29.8.2. Mux Control A

Name: MUXCTRLA
Offset: 0x02
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	INVERT				MUXPOS		MUXNEG[1:0]	
Access	R/W				R/W		R/W	R/W
Reset	0				0		0	0

Bit 7 – INVERT: Invert AC Output
Writing a '1' to this bit enables inversion of the output of the AC. This effectively inverts the input to all the peripherals connected to the signal, and also affects the internal status signals.

Bit 3 – MUXPOS: Positive Input MUX Selection
Writing to this bit field selects the input signal to the positive input of the AC.

Value	Name	Description
0	PIN0	Positive Pin 0
1	PIN1	Positive Pin 1

Bits 1:0 – MUXNEG[1:0]: Negative Input MUX Selection
Writing to this bit field selects the input signal to the negative input of the AC.

Value	Name	Description
0x0	PIN0	Negative Pin 0
0x1	PIN1	Negative Pin 1
0x2	VREF	Voltage Reference
0x3	DAC	DAC output

29.8.3. Interrupt Control

Name: INTCTRL
Offset: 0x06
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
								CMP
Access								R/W
Reset								0

Bit 0 – CMP: Analog Comparator Interrupt Enable
Writing this bit to '1' enables Analog Comparator Interrupt.

29.8.4. Status

Name: STATUS
Offset: 0x07
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
				STATE				COMP
Access				R				R/W
Reset				0				0

Bit 4 – STATE: Analog Comparator State
This shows current status of the ACOUT signal from AC. This will have a synchronizer delay to get updated in the I/O register (3 cycles).

Bit 0 – COMP: Analog Comparator Interrupt Flag
This is the interrupt flag for AC. Writing a '1' to this bit will clear the Interrupt flag.

30. ADC - Analog to Digital Converter

30.1. Overview

The Analog-to-Digital Converter (ADC) peripheral features a 10-bit successive approximation ADC, and is capable of a sampling rate of up to 150ksps. The ADC is connected to a 12-channel Analog Multiplexer which allows twelve single-ended voltage inputs. The single-ended voltage inputs refer to 0V (GND).

The ADC measurements can either be started by application software or an incoming Event from another peripheral. Using the Event System allows for predictable timing without software intervention.

The ADC has a compare function for accurate monitoring of user defined thresholds with minimum software intervention required.

The ADC contains a sample-and-hold circuit which ensures that the input voltage to the ADC is held at a constant level during measurement.

Internal reference voltages of nominally 1.1V or VDD are provided on-chip.

The minimum value represents GND and the maximum value represents the reference voltage.

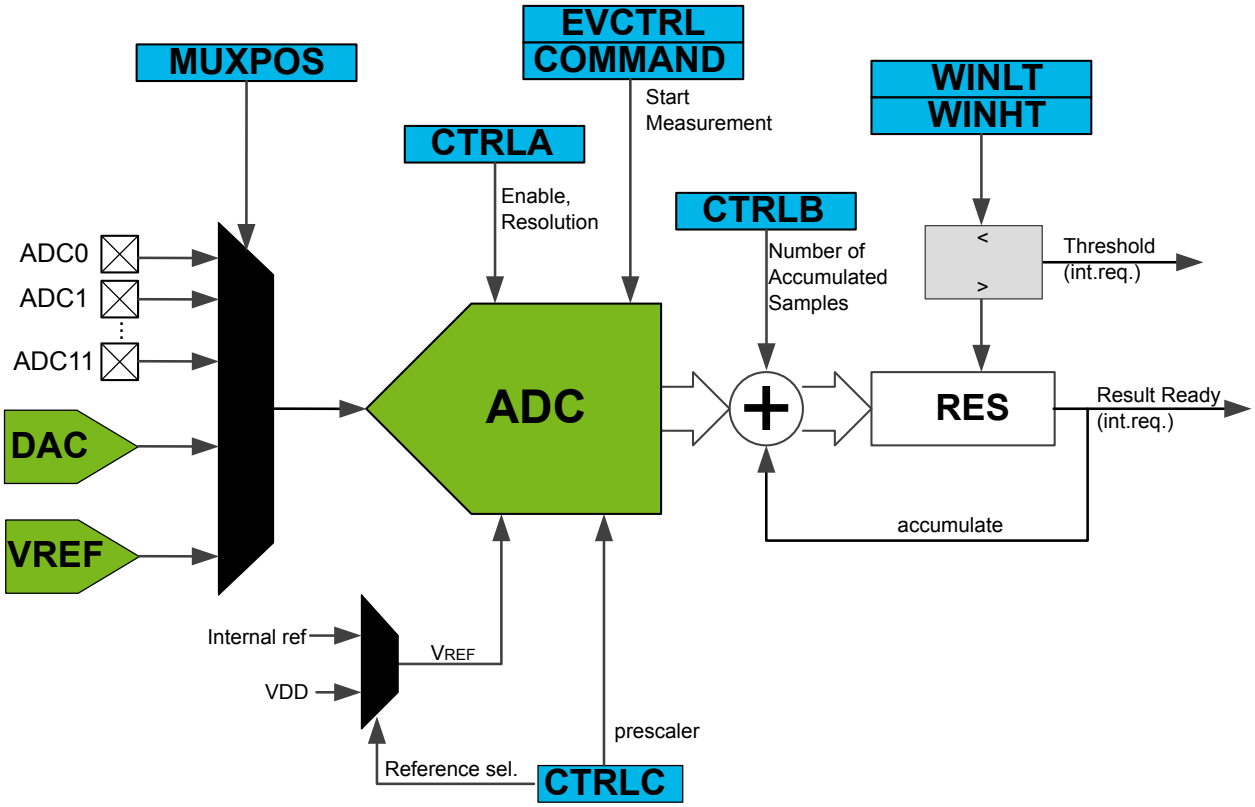
This device has two instances of the ADC, ADC0 and ADC1.

30.2. Features

- 10-bit resolution
- 0.5 LSB integral non-linearity
- ±2 LSB absolute accuracy
- 6.5 - 260µs conversion time
- Up to 150ksps
- Up to twelve multiplexed single-ended input channels
- Temperature sensor input channel
- 0V to V_{DD} ADC input voltage range
- Multiple internal ADC reference voltage between V_{DD} and 0.55V
- Free running or single measurement mode
- Interrupt on ADC measurement complete
- Optional Event triggered measurement
- Optional interrupt on measurement results
- Compare function for accurate monitoring or user defined thresholds
- Automatic accumulation up to 64 samples

30.3. Block Diagram

Figure 30-1. Block Diagram



The analog input channel is selected by writing to the MUX bits in the MUXPOS register (ADC_MUXPOS.MUXPOS). Any of the ADC input pins, as well as GND and an internal voltage reference (programmable) can be selected as single ended input to the ADC. The ADC is enabled by writing a '1' to the ADC Enable bit in the Control A register (ADC_CTRLA.ENABLE). Voltage reference and input channel selections will not go into effect until the ADC is enabled. The ADC does not consume power when ADC_CTRLA.ENABLE is '0'.

The ADC generates a 10-bit result which is presented in the Result Registers, (ADC_RESH,ADC_RESL). The result is presented right adjusted.

30.4. Signal Description

Pin Name	Type	Description
ADC[11:0]	Analog input	analog input to be converted

Related Links

Configuration Summary on page 11
I/O Multiplexing and Considerations on page 19

30.5. Product Dependencies

In order to use this peripheral, other parts of the system must be configured correctly, as described below.

Table 30-1. ADC Product Dependencies

Dependency	Applicable	Peripheral
Clocks	Yes	CLKCTRL
I/O Lines and Connections	Yes	PORT
Interrupts	Yes	CPUINT
Events	Yes	EVSYS
Debug	Yes	UPDI

30.5.1. Clocks

The ADC uses the peripheral clock CLK_PER, and has an internal prescaler to generate the ADC clock source CLK_ADC.

Related Links

CLKCTRL - Clock Controller on page 68

Prescaling and Conversion Timing on page 452

30.5.2. I/O Lines and Connections

The I/O-pins (AINx) are configured by the PORT - I/O Pin Controller.

Related Links

PORT - I/O Pin Controller on page 132

30.5.3. Interrupts

Using the interrupts of this peripheral requires the Interrupt Controller to be configured first.

Related Links

CPUINT - CPU Interrupt Controller on page 97

SREG on page 51

Interrupts on page 136

30.5.4. Events

The events of this peripheral are connected to the Event System.

Related Links

EVSYS - Event System on page 109

30.5.5. Debug Operation

When run-time debugging, this peripheral will continue normal operation. Halting the CPU in debugging mode will halt normal operation of the peripheral.

This peripheral can be forced to operate with halted CPU by writing a '1' to the Debug Run bit in the Debug Control register of the peripheral (*peripheral*_DBGCTRL.DBGRUN).

30.6. Functional Description

30.6.1. Principal of Operation

The ADC provides results with 10-bit resolution.

The input values can be either internal (e.g., a voltage reference) or external (connected I/O pins).

30.6.2. Basic Operation

30.6.2.1. Initialization

The following steps are recommended in order to initialize ADC operation:

1. Select the resolution by writing to the Resolution Selection bit in the Control A register (ADC_CTRLA.RESSEL).
2. Optional: Enable the Free Running mode by writing a '1' to the Free Running bit in the Control A register (ADC_CTRLA.FREERUN).
3. Select the number of conversions to be accumulated per measurement by writing the Sample Accumulation Number Select bit field in the Control B register (ADC_CTRLB.SAMPLNUM).
4. Select a voltage reference by writing to the Reference Selection bit in the Control C register (ADC_CTRLC.REFSEL). Default is the internal reference of the device.
5. Select the sampling rate by writing to the Prescaler bit field in the Control C register (ADC_CTRLC.PRESC).
6. Select an input by writing to the MUXPOS bit field in the MUXPOS register (ADC_MUXPOS.MUXPOS).
7. Optional: Enable Start Event input by writing a '1' to the Start Event Input bit in the Event Control register (ADC_EVCTRL.STARTEI). Configure the Event System accordingly.
8. Enable the ADC by writing a '1' to the Enable bit (ADC_CTRLA.ENABLE).

Following these steps will initialize the ADC for basic measurements, which can be triggered by an Event (if configured) or by writing a '1' to the Start Conversion bit in the Command register (ADC_COMMAND.STCONV).

30.6.2.2. Starting a Measurement

Once the input channels are selected by writing to the MUXPOS register (MUXPOS.MUXPOS), a measurement is triggered by writing a '1' to the ADC Start Conversion bit in the Command register (COMMAND.STCONV). This bit stays '1' as long as the measurement is in progress. It is cleared by hardware when the measurement is completed.

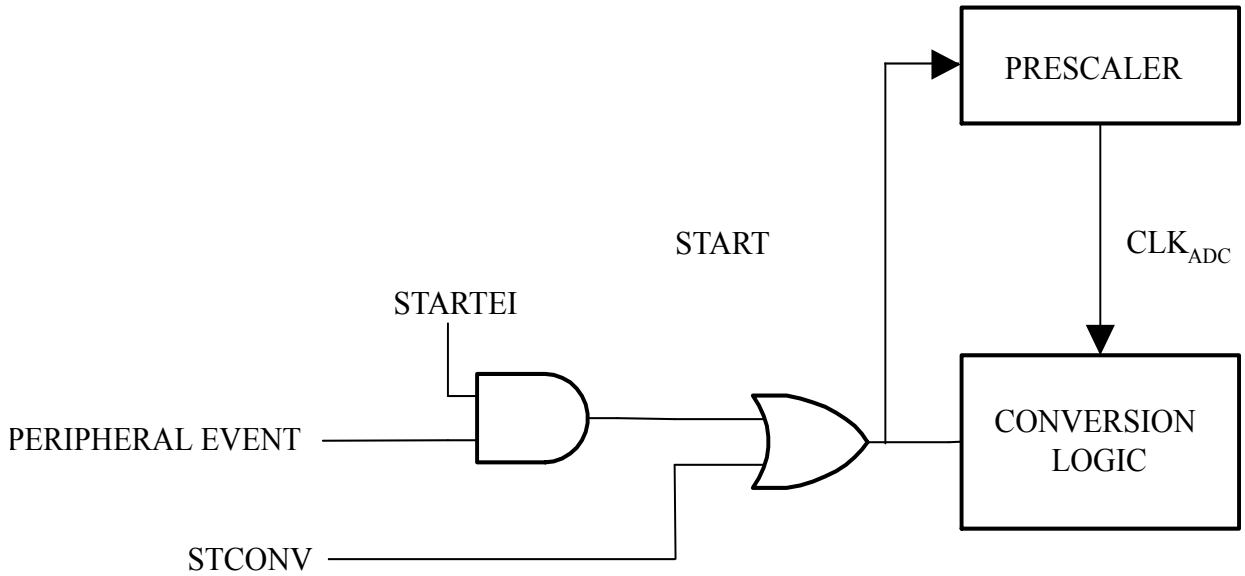
If a different data channel is selected while a measurement is in progress, the ADC will finish the current conversion before performing the channel change.

Depending on the Accumulator setting, the conversion result is from a single sensing operation, or from a sequence of accumulated samples. Once the triggered operation is finished, the Result Ready flag in the Interrupt Flag register (INTFLAG.RESRDY) is set. This flag can be used to set an Interrupt request using the Result Ready Interrupt Enable bit in the Interrupt Control register (INTCTRL.RESRDY).

Alternatively, a conversion can be triggered automatically by an event: this is enabled by writing a '1' to the Start Event Input bit in the Event Control register (EVCTRL.STARTEI). Any peripheral can trigger an ADC conversion. This provides a method to start conversions at predictable intervals.

The event trigger input is edge sensitive: when an event occurs, COMMAND.STCONV is set to '1' to show the status of the event input. STCONV will be automatically cleared on the occurrence of the INTFLAG.RESRDY interrupt flag.
Note: The INTFLAG.RESRDY interrupt flag will be set even if the specific interrupt is disabled. A conversion can thus be triggered without causing an interrupt request.

Figure 30-2. ADC Event Trigger Logic



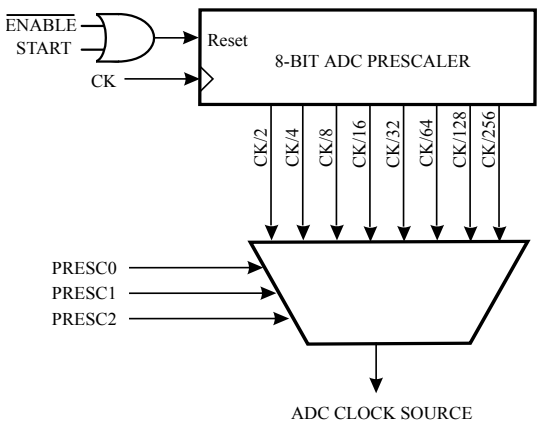
In Free Running mode, the first conversion is started by writing COMMAND.STCONV to '1'. A new conversion cycle is then started immediately after the previous conversion cycle has completed. This is signaled by INTFLAGS.RESRDY.

If the Event Input is not enabled, single conversions can be started by writing COMMAND.STCONV to '1'. STCONV can also be used to determine if a conversion is in progress. The STCONV bit will be read as '1' during a conversion, independently once the conversion is started.

During event input, COMMAND.STCONV is set to '1', and is de-asserted in INTFLAGS.RESRDY.

30.6.2.3. Prescaling and Conversion Timing

Figure 30-3. ADC Prescaler



By default, the successive approximation circuitry requires an input clock frequency between 50kHz and 2MHz to get maximum resolution. If a lower resolution than 10 bits is selected, the input clock frequency to the ADC can be higher than 2MHz to get a higher sample rate.

The ADC module contains a prescaler, which generates an acceptable ADC clock (CLK_ADC) frequency from any CPU frequency (CLK_PER) above 100kHz. The prescaling is selected by writing to the Prescaler bits in the Control C register (ADC_CTRLC.PRESC). The prescaler starts counting from the moment the ADC is switched on by writing a '1' to the Enable bit in the Control A register (ADC_CTRLA.ENABLE). The prescaler keeps running for as long as the ENABLE bit is '1', and holds value zero when ENABLE is '0'.

When initiating a conversion by writing a '1' to the Start Conversion bit in the Command register (ADC_COMMAND.STCONV), the conversion starts at the following rising edge of the CLK_ADC clock cycle.

A normal conversion takes 13 CLK_ADC cycles. This value is increased for increased sample lengths. Also, startup and changing channels will cause additional delays.

The sampling time is configured by the Sample Length bit field in the Sample Enable register (ADC_SAMPLEN.SAMPLEN). This controls the ADC sampling time in number of CLK_ADC cycles, depending of the prescaler value. This allows adaptation to high-impedance sources without relaxing conversion speed.

The actual sample-and-hold takes place 2 CLK_ADC cycles after the start of a normal conversion with SAMPLEN value zero. Increasing the SAMPLEN value will increase the number of CLK_ADC cycles, see SAMPLEN register description. When a conversion is complete, the result is written to the Result registers (ADC_RESH, ADC_RESL), and the Result Ready interrupt flag is set (ADC_INTFLAG.RESRDY). The interrupt flag will be cleared when the result is read from the Result registers, and it can be cleared by writing a '1' to ADC_INTFLAG.RESRDY. The software may then write ADC_COMMAND.STCONV to '1' again, and a new conversion will be initiated on the first rising ADC clock edge.

When a start Trigger occurs (from Event or software) the prescaler will be reset. This assures a fixed delay from the trigger event to the start of conversion. In this mode, the sample-and-hold takes place two ADC clock cycles after the rising edge on the trigger source signal.

In Free Running mode, a new conversion will be started immediately after the conversion completes, while STCONV remains '1'.

Figure 30-4. ADC Timing Diagram - Single Conversion

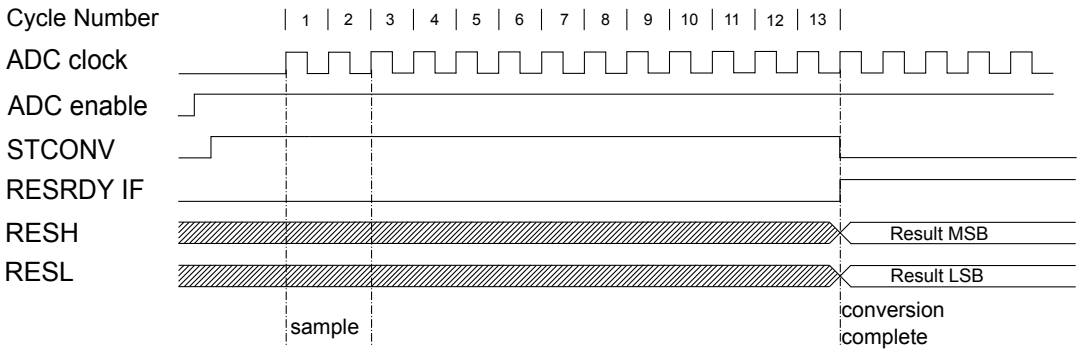


Figure 30-5. ADC Timing Diagram - Event-Triggered Conversion

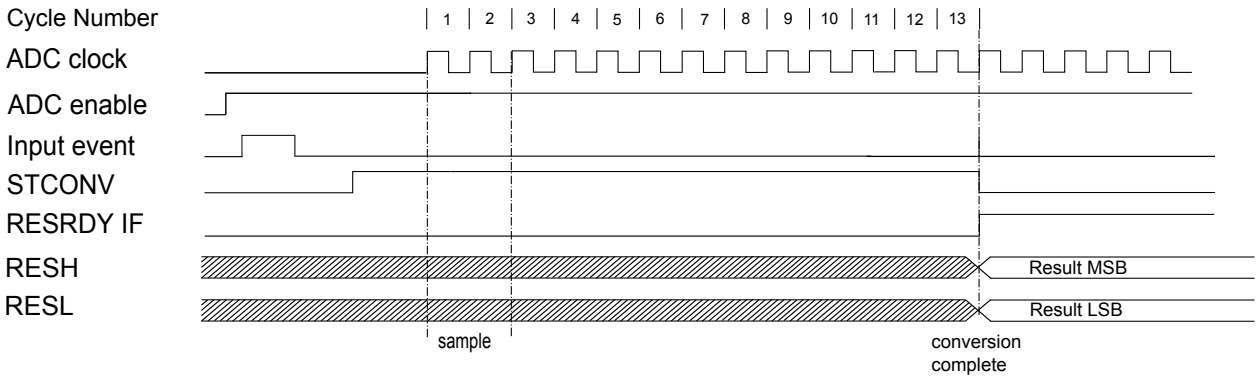
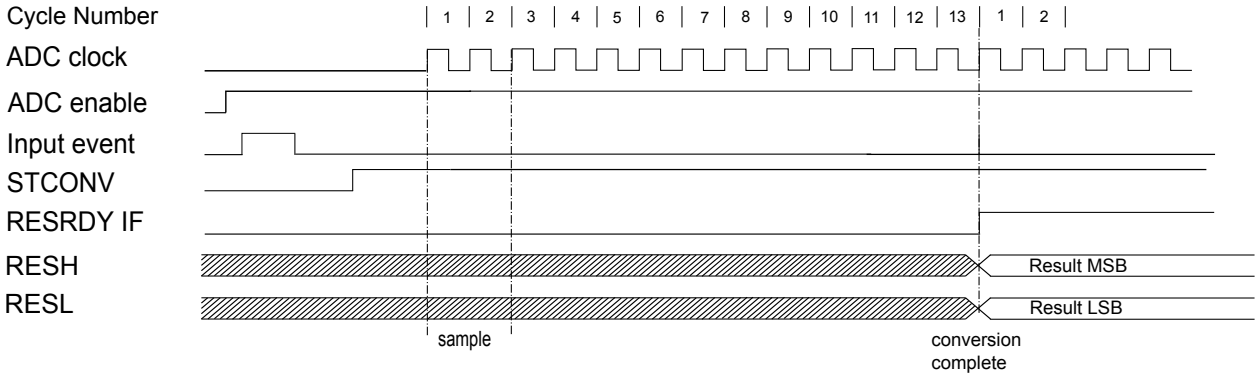


Figure 30-6. ADC Timing Diagram - Free Running Conversion



30.6.2.4. Changing Channel or Reference Selection

The MUXn and REFS1:0 bits in the MUXPOS and CTRLB, Register are single buffered through a temporary register to which the CPU has random access. This ensures that the channels and reference selection only takes place at a safe point during the conversion. The channel and reference selection is continuously updated until a conversion is started.

Once the conversion starts, the channel and reference selection is locked to ensure a sufficient sampling time for the ADC. Continuous updating resumes in the last ADC clock cycle before the conversion completes (RESRDY in INTFLAGS is set). Note that the conversion starts on the following rising ADC clock edge after STCONV is written.

ADC Input Channels

When changing channel selections, the user should observe the following guidelines to ensure that the correct channel is selected:

In Single Conversion mode, always select the channel before starting the conversion. The channel selection may be changed one ADC clock cycle after writing one to STCONV. However, the simplest method is to wait for the conversion to complete before changing the channel selection.

In Free Running mode, always select the channel before starting the first conversion. The channel selection may be changed one ADC clock cycle after writing one to STCONV. However, the simplest method is to wait for the first conversion to complete, and then change the channel selection. Since the next conversion has already started automatically, the next result will reflect the previous channel selection. Subsequent conversions will reflect the new channel selection.p

ADC Voltage Reference

The reference voltage for the ADC (V_{REF}) indicates the conversion range for the ADC. Channels that exceed VREF will result in codes close to 0x3FF. V_{REF} can be selected as either V_{DD} or internal reference. V_{DD} is connected to the ADC through a passive switch. The internal reference is generated from the internal bandgap reference (V_{BG}) through an internal amplifier.

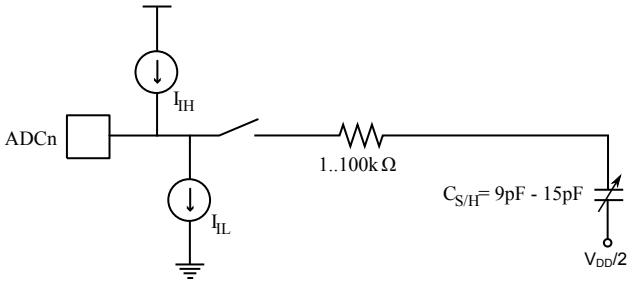
Analog Input Circuitry

The analog input circuitry is illustrated in the Figure below. An analog source applied to ADCn is subjected to the pin capacitance and input leakage of that pin, regardless of whether that channel is selected as input for the ADC. When the channel is selected, the source must drive the S/H capacitor through the series resistance (combined resistance in the input path).

The ADC is optimized for analog signals with an output impedance of approximately 10kΩ or less. If such a source is used, the sampling time will be negligible. If a source with higher impedance is used, the sampling time will depend on how long time the source needs to charge the S/H capacitor, with can vary widely. The user is recommended to only use low impedance sources with slowly varying signals, since this minimizes the required charge transfer to the S/H capacitor.

Signal components higher than the Nyquist frequency ($f_{ADC}/2$) should not be present for either kind of channels, to avoid distortion from unpredictable signal convolution. We advise to remove high frequency components with a low-pass filter before applying a signal as input to the ADC.

Figure 30-7. Analog Input Schematic



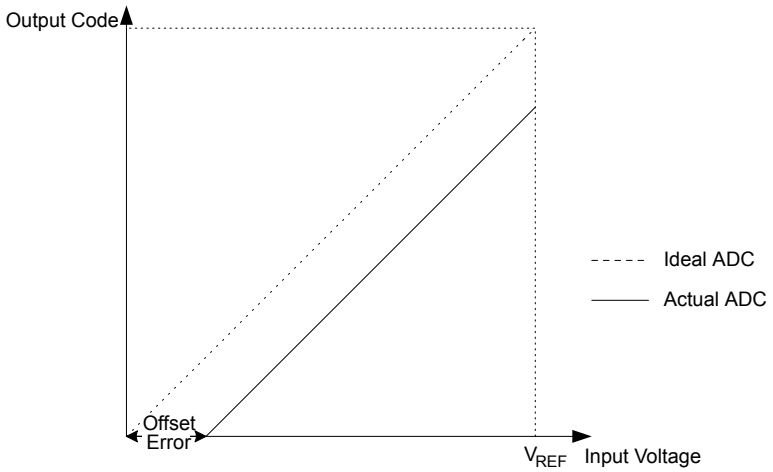
ADC Accuracy Definitions

An n-bit single-ended ADC converts a voltage linearly between GND and VREFin 2n steps (LSBs). The lowest code is read as 0, and the highest code is read as 2n-1. Several parameters describe the deviation from the ideal behavior:

Offset

The deviation of the first transition (0x000 to 0x001) compared to the ideal transition (at 0.5 LSB). Ideal value: 0 LSB.

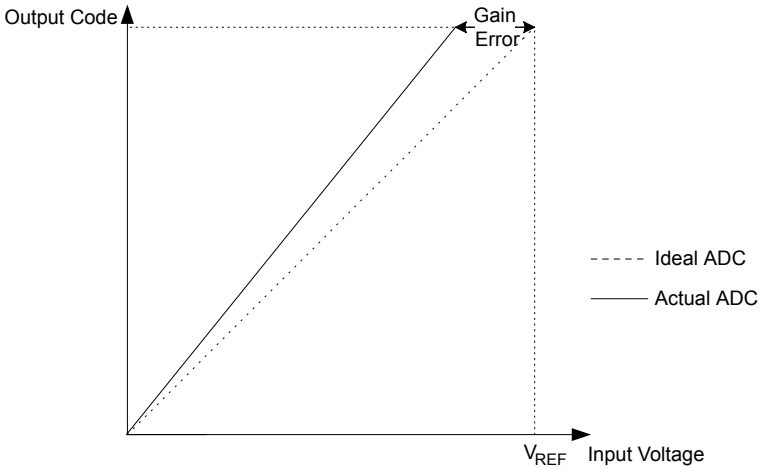
Figure 30-8. Offset Error



Gain Error

After adjusting for offset, the gain error is found as the deviation of the last transition (0x3FE to 0x3FF) compared to the ideal transition (at 1.5 LSB below maximum). Ideal value: 0 LSB.

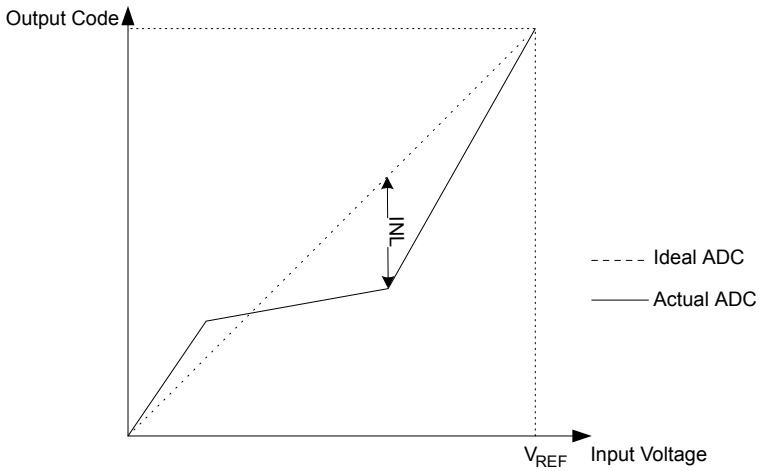
Figure 30-9. Gain Error



Integral Non-Linearity (INL)

After adjusting for offset and gain error, the INL is the maximum deviation of an actual transition compared to an ideal transition for any code. Ideal value: 0 LSB.

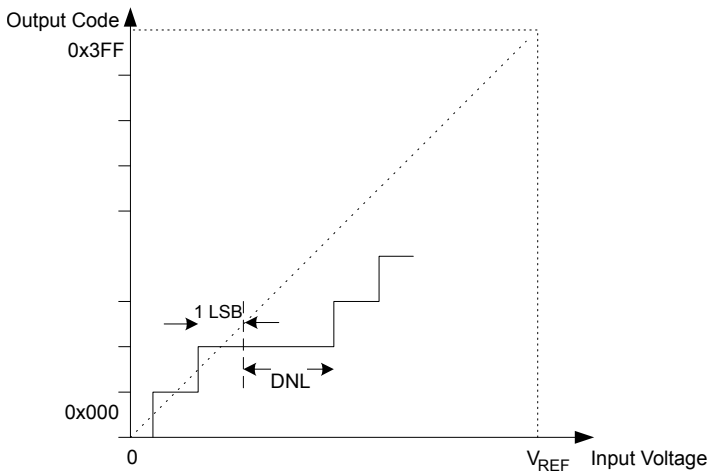
Figure 30-10. Integral Non-Linearity



Differential Non-Linearity (DNL)

The maximum deviation of the actual code width (the interval between two adjacent transitions) from the ideal code width (1 LSB). Ideal value: 0 LSB.

Figure 30-11. Differential Non-Linearity



Quantization Error	Due to the quantization of the input voltage into a finite number of codes, a range of input voltages (1 LSB wide) will code to the same value. Always ±0.5 LSB.
Absolute Accuracy	The maximum deviation of an actual (unadjusted) transition compared to an ideal transition for any code. This is the compound effect of offset, gain error, differential error, non-linearity, and quantization error. Ideal value: ±0.5 LSB.

30.6.2.5. ADC Conversion Result

After the conversion is complete (RESRDY is high), the conversion result RES can be found in the ADC Result Registers (RESL, RESH). The result is

$$\mathrm{RES} = \frac{1024 \times V_{\mathrm{IN}}}{V_{\mathrm{REF}}}$$

Where V_{IN} is the voltage on the selected input pin and V_{REF} the selected voltage reference (see register description for ADC_CTRLC.REFSEL and ADC_MUXPOS.MUXPOS). 0x000 represents analog ground, and 0x3FF represents the selected reference voltage minus one LSB.

30.6.2.6. Temperature Measurement

The temperature measurement is based on an on-chip temperature sensor that is coupled to channel ADC8. Selecting the ADC8 channel by writing the MUXPOS bits in the MUXPOS register (ADC_MUXPOS.MUXPOS) to 0x8 enables the temperature sensor. The voltage reference must be the internal 1.1V voltage reference. When the temperature sensor is enabled, the ADC acquires the temperature sensor voltage.

The measured voltage has a linear relationship to the temperature. Due to process variations, the temperature sensor output voltage varies between individual devices.

In order to achieve more accurate results, the temperature measurement can be calibrated in the application software. The software calibration requires that a calibration value is measured and stored in a register or EEPROM for each chip, e.g. as a part of the production test.

The software calibration is done utilizing the formula:

T = { [(ADCH << 8) | ADCL] - TOS} / k

Where ADCn are the high byte and low byte of the ADC data register value, k is a fixed coefficient and TOS is the temperature sensor offset value determined and stored into EEPROM as a part of the production test.

30.6.2.7. Window Comparator Mode

The ADC can raise a flag (ADC_INTFLAG.WCOMP) and request an interrupt (WCOMP) when the result of a conversion is above and/or below certain thresholds. The available modes are:

- The result is under a threshold
- The result is over a threshold
- The result is inside a window (above a lower threshold, but below the upper one)
- The result is outside a window (either under the lower or above the upper threshold)

The thresholds are defined by writing to the Window Comparator Threshold registers (ADC_WINLT and ADC_WINHT, respectively). Writing to the Window Comparator Mode bit field in the Control E register (ADC_CTRLE.WINCM) selects the conditions when the flag is raised and/or the interrupt is requested.

Assuming the ADC is already configured to run, follow these steps to use the Window Comparator Mode:

1. Set the required threshold(s) by writing to ADC_WINLT and/or ADC_WINHT.
2. Optional: enable the interrupt request by writing a '1' to the Window Comparator Interrupt Enable bit in the Interrupt Control register (ADC_INTCTRL.WCOMP)

3. Enable the Window Comparator and select a mode by writing a non-zero value to ADC_CTRLE.WINCM.

Note: When accumulating multiple samples, the comparison between the result and the threshold will happen after the last sample was acquired. Consequently, the flag is only raised once, after taking the last sample of the accumulation.

30.6.3. Events

An ADC conversion can be triggered automatically by an event input if ADC_EVCTRL.STARTEI is set to one.

30.6.4. Interrupts

Table 30-2. Available Interrupt Vectors and Sources

Offset	Name	Vector Description	Conditions
0x00	RESRDY	Result Ready interrupt	The TCD is done with one TCD cycle.
0x02	WCOMP	Window Comparator interrupt	As defined by ADC_CTRLE.WINCM

When an interrupt condition occurs, the corresponding Interrupt Flag is set in the Interrupt Flags register of the peripheral (*peripheral*_INTFLAGS).

An interrupt source is enabled or disabled by writing to the corresponding bit in the peripheral's Interrupt Control register (*peripheral*_INTCTRL).

An interrupt request is generated when the corresponding interrupt source is enabled and the Interrupt Flag is set. The interrupt request remains active until the Interrupt Flag is cleared. See the peripheral's INTFLAGS register for details on how to clear Interrupt Flags.

Note: Interrupt requests are only generated when Interrupts are enabled globally.

30.6.5. Sleep Mode Operation

In Standby sleep mode, the ADC is disabled by default. It may stay operational in Standby sleep mode when the Run in Standby bit in the Control A register (ADC_CTRLA.RUNSTDBY) is written to '1'.

When entering Standby with RUNSTDBY=1, any ongoing conversion is finished.

In Standby, no single shot conversion triggered by software is possible, since the CPU is stopped. All other trigger modes are available, including free-running mode. The peripheral clock is requested if needed, and is turned off after the measurement is completed. When an input Event trigger occurs, the positive edge will be detected, the Start Conversion bit in the Command register (ADC_COMMAND.STCONV) will be set to '1', and the measurement will start. When the measurement is completed, the Result Ready Flag (ADC_INTFLAGS.RESRDY) is set and STCONV is cleared to '0'. This will allow monitoring the status of the input Event.

Note: The reference source and supply infrastructure need time to stabilize when activated in Standby sleep mode. Configure a delay for the start of the measurement by writing a non-zero value to the Channel Change Delay in the Control D register (ADC_CTRLD.CHDLY).

In Power Down sleep mode, no conversions are possible; any ongoing conversion is aborted and the content of the Result register (ADC_RES) is invalid when entering Power Down.

Related Links

SLPCTRL - Sleep Controller on page 85

30.6.6. Synchronization

Not applicable.

30.6.7. Configuration Change Protection

Not applicable.

30.7. Register Summary

Offset	Name	Bit Pos.								
0x00	CTRLA	7:0	RUNSTBY					RESSEL	FREERUN	ENABLE
0x01	CTRLB	7:0						SAMPLNUM[2:0]		
0x02	CTRLC	7:0		SAMPCAP	REFSEL[1:0]			PRESC[2:0]		
0x03	CTRLD	7:0	CHDLY[2:0]			ASDV	SAMPDLY[3:0]			
0x04	CTRLE	7:0						WINCM[2:0]		
0x05	SAMPLEN	7:0				SAMPLEN[4:0]				
0x06	MUXPOS	7:0				MUXPOS[4:0]				
0x07	Reserved									
0x08	COMMAND	7:0								STCONV
0x09	EVCTRL	7:0								STARTEI
0x0A	INTCTRL	7:0							WCOMP	RESRDY
0x0B	INTFLAGS	7:0							WCOMP	RESRDY
0x0C	DBGRUN	7:0								DBGRUN
0x0D	TEMP	7:0	TEMP[7:0]							
0x0E ... 0x0F	Reserved									
0x10	RES	7:0	RES[7:0]							
0x11		15:8	RES[15:8]							
0x12	WINLT	7:0	WINLT[7:0]							
0x13		15:8	WINLT[15:8]							
0x14	WINHT	7:0	WINHT[7:0]							
0x15		15:8	WINHT[15:8]							
0x16	CALIB	7:0							COMPCURR	DUTYCYC

30.8. Register Description

30.8.1. Control A

Name: CTRLA
Offset: 0x00
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	RUNSTBY					RESSEL	FREERUN	ENABLE
Access	R/W	R	R	R	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 7 – RUNSTBY: Run in Standby
This bit determines whether the ADC needs to run when the chip is in standby sleep mode.

Bit 2 – RESSEL: Resolution Selection
This bit selects the ADC resolution.

Value	Description
0	Full 10-bit resolution. The 10-bit ADC results are accumulated or stored to the ADC Result registers (RESH, RESL).
1	8-bit resolution. The conversion results are truncated to 8 bits (MSBs) before they are accumulated or stored to the ADC Data register. The two least significant bits are discarded.

Bit 1 – FREERUN: Free Running
Writing a '1' to this bit will enable Free Running mode for the data acquisition. The first conversion is started by writing COMMAND.STCONV bit high. In Free running mode, a new conversion cycle is started immediately as soon as the previous conversion cycle has completed. This is signaled by INTFLAGS.RESRDY.

Bit 0 – ENABLE: ADC Enable

Value	Description
0	ADC is disabled.
1	ADC is enabled.

30.8.2. Control B

Name: CTRLB
Offset: 0x01
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
						SAMPLNUM[2:0]		
Access	R	R	R	R	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 2:0 – SAMPLNUM[2:0]: Sample Accumulation Number Select
These bits select how many consecutive ADC results are accumulated automatically. When the accumulation is selected, consecutive ADC results are accumulated into the ADC Result register (RESL and RESH) in one complete operation.

Note: The digital accumulation feature is available also in plain ADC modes.

Table 30-3. ADC Result Accumulation Numbers

SAMPLNUM[2:0]	Number of Accumulated Samples
0x0	1 (only one ADC conversion)
0x1	2
0x2	4
0x3	8
0x4	16
0x5	32
0x6	64
0x7	Reserved

30.8.3. Control C

Name: CTRLC
Offset: 0x02
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
		SAMPCAP	REFSEL[1:0]			PRESC[2:0]		
Access	R	R/W	R/W	R/W	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 6 – SAMPCAP: Sample Capacitance Selection
Writing this bit to '1' will increase the size of sampling capacitance, which is recommended for high sampling rates.

For low sampling rates and reference values below 1V, it is recommended to write this bit to '0'.

Bits 5:4 – REFSEL[1:0]: Reference Selection
These bits select the voltage reference for the ADC.
Note: Changing these bits will not go in effect until an ongoing conversion is complete (INTFLAGS.RESRDY is set).

Table 30-4. ADC Voltage Reference Selections

REFSEL[1:0]	Description
0x0	Internal reference
0x1	V_{DD}
0x2	Reserved
0x3	Reserved

Bits 2:0 – PRESC[2:0]: Prescaler
These bits define the integer prescaling ratio between peripheral clock (CLK_PER) and the ADC clock (CLK_ADC).

Table 30-5. ADC Prescaling Values

PRESC[2:0]	ADC Clock Division Ratio
0x0	2
0x1	4
0x2	8
0x3	16
0x4	32
0x5	64
0x6	128
0x7	256

30.8.4. Control D

Name: CTRLD
Offset: 0x03
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	CHDLY[2:0]			ASDV	SAMPDLY[3:0]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:5 – CHDLY[2:0]: Channel Change Delay Selection
These bits define the delay when changing input channels. The delay allows the analog circuits to settle on new channel. The delay is application dependent, and therefore this option allows the user to select a suitable delay. The delay is expressed as a number of CLK_ADC cycles.

The total Channel Change Delay is the sum of the programmable Channel Change Delay and the intrinsic Pipeline Delay.

Table 30-6. Channel Change Delay Selection

CHDLY[2:0]	Programmable Channel Change Delay [CLK_ADC cycles]	Pipeline Delay [CLK_ADC cycles]	Total Channel Change Delay [CLK_ADC cycles]
0x0	0	0	0
0x1	16	11	27
0x2	32	11	43
0x3	64	11	75
0x4	128	11	139
0x5	256	11	267
Other	Reserved	-	-

Bit 4 – ASDV: Automatic Sampling Delay Variation
Writing this bit to '1' enables automatic sampling delay variation between ADC conversions. The purpose of varying sampling instant is to randomize the sampling instant and thus avoid standing frequency components in frequency spectrum. The value of the SAMPDLY bits is incremented automatically by one after each sampling. Thus, the additional delay before next sample is varied automatically.

When the Automatic Sampling Delay Variation is enabled and the SAMPDLY value reaches 0x7, it wraps around to 0x0.

Value	Description
0	The Automatic Sampling Delay Variation is disabled.
1	The Automatic Sampling Delay Variation is enabled.

Bits 3:0 – SAMPDLY[3:0]: Sampling Delay Selection

These bits define the delay between consecutive ADC samples. The programmable Sampling Delay allows choosing (modifying) the sampling frequency that suits best in an application where other periodic noise sources may otherwise disturb the sampling. The SAMPDLY field can be also modified automatically from sampling cycle to another, by setting the ASDV bit.

Value	Description
0x0	0
0x1	1
0x2	2
0x3	3
0x4	4
0x5	5
0x6	6
0x7	7
0x8	8
0x9	9
0xA	10
0xB	11
0xC	12
0xD	13
0xE	14
0xF	15

30.8.5. Control E

Name: CTRLE
Offset: 0x4
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
						WINCM[2:0]		
Access	R	R	R	R	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 2:0 – WINCM[2:0]: Window Comparator Mode
These bits enable and define the Window Comparator mode. RESULT is the 16-bit wide Accumulator output. WINLT and WINHT are 16-bit lower threshold value and 16-bit higher threshold value, respectively.

Table 30-7. Sampling Delay Selection

WINCM[2:0]	Window Comparator Mode
0x0	No Window Comparison (default)
0x1	Mode 1: RESULT < WINLT
0x2	Mode 2: RESULT > WINHT
0x3	Mode 3: WINLT<RESULT<WINHT
0x4	Mode 4: !(WINLT<RESULT<WINHT)
0x5 - 0x7	Reserved

30.8.6. Sample Enable

Name: SAMPLEN
Offset: 0x5
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
				SAMPLEN[4:0]				
Access	R	R	R	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 4:0 – SAMPLEN[4:0]: Sample Length
These bits control the ADC sampling time in number of CLK_ADC cycles, depending of the prescaler value, thus controlling the ADC input impedance.

The total conversion time is determined by the number of Sample/Hold cycles (selected by this bit field), the pipeline delay (11 cycles), and the frequency of CLK_ADC.

SAMPLEN[4:0]	Sample/Hold cycles	Total Conversion Time [CLK_ADC cycles]
0x00	2	2+11 = 13
0x01	3	3+11 = 14
...	...	...
0x1F	33	33+11 =44

30.8.7. MUXPOS

Name: MUXPOS
Offset: 0x06
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
				MUXPOS[4:0]				
Access	R	R	R	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 4:0 – MUXPOS[4:0]: MUXPOS
The value of these bits selects which analog inputs are connected to the ADC. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (INTFLAGS.RESRDY is set).

Table 30-8. Analog Input Channel Selection

MUXPOS[4:0]	Single-Ended Input
0x00	AIN0
0x01	AIN1
0x02	AIN2
0x03	AIN3
0x04	AIN4
0x05	AIN5
0x06	AIN6
0x07	AIN7
0x08	AIN8
0x09	AIN9
0x0A	AIN10
0x0B	AIN11
0x0C - 1B	Reserved
0x1C	DAC0
0x1D	Internal reference (from reference controller)
0x1E	Temperature sensor
0x1F	0V (GND)

30.8.8. Command

Name: COMMAND
Offset: 0x08
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
								STCONV
Access	R	R	R	R	R	R	R	R/W
Reset	0	0	0	0	0	0	0	0

Bit 0 – STCONV: Start Conversion
Writing a '1' to this bit in single-shot mode starts one measurement. In free running mode, the first measurement is started. The measurement duration depends on the selected operating mode (single-shot or free running mode) and the accumulation of the ADC results.

STCONV will read as '1' as long as a measurement is in progress. When the measurement is complete, this bit returns to zero.

Writing '0' to this bit has no effect.

30.8.9. Event Control

Name: EVCTRL
Offset: 0x09
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
								STARTEI
Access	R	R	R	R	R	R	R	R/W
Reset	0	0	0	0	0	0	0	0

Bit 0 – STARTEI: Start Event Input
This is enable bit for event input. When this bit is written to '1', an event input from any peripheral can trigger an ADC conversion.

30.8.10. Interrupt Control

Name: INTCTRL
Offset: 0x0A
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
							WCOMP	RESRDY
Access	R	R	R	R	R	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 1 – WCOMP: Window Comparator Interrupt Enable
Writing a '1' to this register will enable the Window Comparator interrupt.

Writing a '0' to this register will disable the Window Comparator interrupt.

Bit 0 – RESRDY: Result Ready Interrupt Enable
Writing a '1' to this bit will enable the Result Ready (End of Conversion) interrupt.

Writing a '0' to this bit will disable the Result Ready (End of Conversion) interrupt.

30.8.11. Interrupt Flags

Name: INTFLAGS
Offset: 0x0B
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
							WCOMP	RESRDY
Access	R	R	R	R	R	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 1 – WCOMP: Window Comparator Interrupt Flag
This flag reflects the status of the digital Window Comparator, monitoring the ADC/Accumulator RESULT register. The Flag is set when the Window Comparator detects a trigger condition, defined in the Control E (CTRLE) register. The flag is set when the conversion is complete. The flag has to be cleared by software.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit clears the flag.

Note: There is no hardware clear on this interrupt flag. Only software can clear this interrupt.

Bit 0 – RESRDY: Result Ready Interrupt Flag
This bit will reflect the current status of the ADC conversion condition. The flag is set, when the ADC conversion is complete. If the ADC is enabled, the data registers and window comparator status flag are also updated.

Writing a '0' to this bit has no effect.

Writing a '1' to this bit clears the flag.

Note: There is no hardware clear on this interrupt flag. Only software can clear this interrupt.

30.8.12. Debug Run

Name: DBGRUN
Offset: 0x0C
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
								DBGRUN
Access	R	R	R	R	R	R	R	R/W
Reset	0	0	0	0	0	0	0	0

Bit 0 – DBGRUN: Debug Run
When written to '1', the peripheral will continue operating in debug mode.

30.8.13. Temporary

The Temporary register is used by the CPU for single-cycle, 16-bit access to the 16-bit registers of this peripheral. It can also be read and written by software. See also Accessing 16-bit Registers. There is one common Temporary register for all the 16-bit registers of this peripheral.

Name: TEMP
Offset: 0x0D
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	TEMP[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – TEMP[7:0]: Temporary
Temporary register for read/write operations in 16-bit registers.

30.8.14. Result

The RESL and RESH register pair represents the 16-bit value, RES. The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + 0x01. For more details on reading and writing 16-bit registers, refer to Accessing 16-bit Registers.

If the analog input of the ADC is over-driven, the 10 bit ADC saturates at full scale value 0x3FF. Likewise, if the ADC is under-driven, the ADC saturates at zero value 0x000. As the ADC cannot produce larger than 0x3FF values, the digital accumulator does not overflow beyond 0xFFFF, even after maximum allowed 64 accumulations.

Name: RES
Offset: 0x10
Reset: 0x00
Property: -

Bit	15	14	13	12	11	10	9	8
	RES[15:8]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	RES[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 15:8 – RES[15:8]: Result high byte
These bits constitute the MSB of RES register, where the msb is RES[15]. The ADC itself has 10-bit output, ADC[9:0], where the msb is ADC[9]. The data format in ADC and Digital Accumulation is 1's complement, where 0x0000 represents the zero and 0xFFFF represent the largest number (full scale).

Bits 7:0 – RES[7:0]: Result low byte
These bits constitute the LSB of ADC/Accumulator Result, (RES) register. The data format in ADC and Digital Accumulation is 1's complement, where 0x0000 represents the zero and 0xFFFF represent the largest number (full scale).

30.8.15. Window Comparator Low Threshold

This register is the 16-bit Low Threshold for the digital comparator monitoring the RES register. The ADC itself has 10-bit output, RES[9:0], where the msb is RES[9]. The data format in ADC and Digital Accumulation is one's complement, where 0x0000 represents the zero and 0xFFFF represent the largest number (full scale).

The WINLTH and WINLTL register pair represent the 16-bit value, WINLT. The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + 0x01. For more details on reading and writing 16-bit registers, refer to Accessing 16-bit Registers.

Note: When accumulating samples, the window comparator thresholds are applied on the accumulated value, not per sample.

Name: WINLT
Offset: 0x12
Reset: 0x00
Property: -

Bit	15	14	13	12	11	10	9	8
	WINLT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	WINLT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:8 – WINLT[15:8]: Window Comparator Low Threshold high byte
These bits hold the MSB of the 16-bit register.

Bits 7:0 – WINLT[7:0]: Window Comparator Low Threshold low byte
These bits hold the LSB of the 16-bit register.

30.8.16. Window Comparator High Threshold

This register is the 16-bit High Threshold for the digital comparator monitoring the RES register. The ADC itself has 10-bit output, RES[9:0], where the msb is RES[9]. The data format in ADC and Digital Accumulation is one's complement, where 0x0000 represents the zero and 0xFFFF represent the largest number (full scale).

The WINHTH and WINHTL register pair represent the 16-bit value, WINHT. The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + 0x01. For more details on reading and writing 16-bit registers, refer to Accessing 16-bit Registers.

Name: WINHT
Offset: 0x14
Reset: 0x00
Property: -

Bit	15	14	13	12	11	10	9	8
	WINHT[15:8]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	WINHT[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 15:8 – WINHT[15:8]: Window Comparator High Threshold high byte
These bits hold the MSB of the 16-bit register.

Bits 7:0 – WINHT[7:0]: Window Comparator High Threshold low byte
These bits hold the LSB of the 16-bit register.

30.8.17. Calibration

Name: CALIB
Offset: 0x16
Reset: 0x01
Property: -

Bit	7	6	5	4	3	2	1	0
							COMPCURR	DUTYCYC
Access	R	R	R	R	R		R/W	R/W
Reset	0	0	0	0	0		0	1

Bit 1 – COMPCURR: Comparator Current
Writing a '1' to this bit increases the comparator gain for higher accuracy.

Value	Description
0	Low preamp gain for high speed (CLK_ADC ≥2MHz)
1	High preamp gain for high accuracy (ADC VREF ≤0.55V)

Bit 0 – DUTYCYC: Duty Cycle
This bit determines the duty cycle of the ADC clock.

Note: Not supported for ADC running at minimum PRESC setting.

Value	Description
0	50% Duty Cycle
1	25% Duty Cycle (high 25% and low 75%)

31. DAC - Digital to Analog Converter

31.1. Overview

The Digital-to-Analog Converter (DAC) converts a digital value to a voltage.

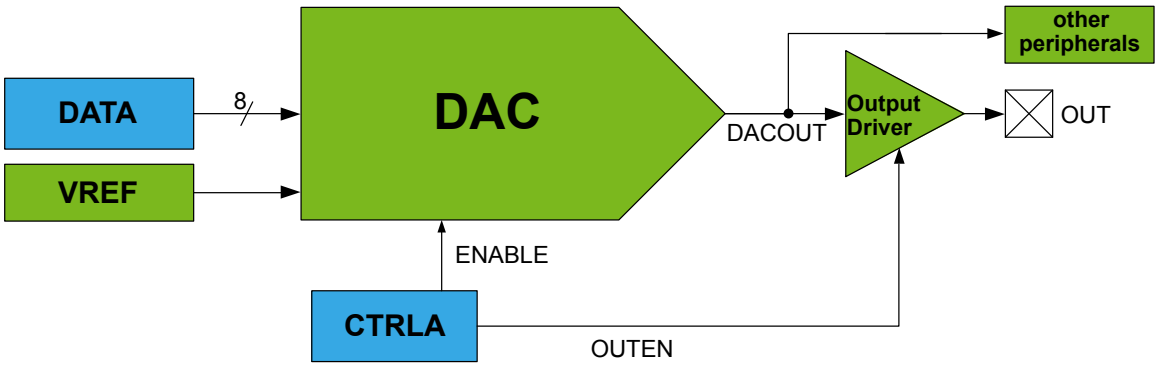
The DAC features an 8-bit Resistor String type DAC, capable of converting 350,000 samples per second (350ksps). The DAC uses the internal Voltage Reference (VREF) as upper limit for conversion. The DAC has one continuous time output with high drive capabilities, which is able to drive 5kΩ or 50pF load. The DAC conversion can be started from the application by writing the data conversion registers.

31.2. Features

- 8-bit resolution
- Up to 350ksps conversion rate
- High drive capabilities
- Possibility to use as input to Analog Comparator (AC) or ADC

31.3. Block Diagram

Figure 31-1. DAC Block Diagram



31.4. Signal Description

Signal	Description	Type
DACOUT	DAC output	Analog

Related Links

I/O Multiplexing and Considerations on page 19

31.5. Product Dependencies

In order to use this peripheral, other parts of the system must be configured correctly, as described below.

Table 31-1. DAC Product Dependencies

Dependency	Applicable	Peripheral
Clocks	Yes	CLKCTRL
I/O Lines and Connections	Yes	PORT
Interrupts	No	-
Events	No	-
Debug	Yes	UPDI

Related Links

Clocks on page 86

I/O Lines and Connections on page 480

Debug Operation on page 480

31.5.1. Clocks

This peripheral depends on the peripheral clock.

Related Links

CLKCTRL - Clock Controller on page 68

Product Dependencies on page 98

31.5.2. I/O Lines and Connections

Using the I/O lines of the peripheral requires configuration the I/O pins.

Table 31-2. I/O Lines

Instance	Signal	I/O Line	Peripheral Function
DAC	DACOUT	PA06	A

The DAC has one analog output pin (DACOUT) that must be configured before it can be used.

The DAC is also internally connected to the AC and to the ADC. To use this internal DACOUT as input, they must be configured in their respective registers.

Related Links

PORT - I/O Pin Controller on page 132

AC – Analog Comparator on page 437

ADC - Analog to Digital Converter on page 448

31.5.3. Events

Not applicable.

31.5.4. Interrupts

Not applicable.

31.5.5. Debug Operation

This peripheral is unaffected by entering debug mode.

If the peripheral is configured to require periodical service by the CPU through interrupts or similar, improper operation or data loss may result during halted debugging.

Related Links

PORT - I/O Pin Controller on page 132

31.6. Functional Description

31.6.1. Principle of Operation

The DAC converts the digital value written to the Data register (DAC_DATA) to analog voltage values. The conversion range is between GND and the selected reference voltage.

31.6.2. Basic Operation

31.6.2.1. Initialization

To operate the DAC, the following steps are required:

- Select the DAC reference voltage by writing the DAC and AC Reference Selection bits in the Control A register of the Voltage Reference (VREF_CTRLA.DACREFSEL). The conversion range is between GND and the selected reference voltage.
- Configure the further usage of the DAC output:
  - Configure an internal peripheral (e.g. AC, ADC) to use the DAC output. See the according peripheral's documentation.
  - Enable the output to a pin by writing a '1' to the Output Enable bit in the Control A register (DAC_CTRLA.OUTEN). This requires configuration of the Port peripheral.
- Write an initial digital value to the Data register (DAC_DATA.DATA).
- Enable the DAC by writing a '1' to the Enable bit in the Control A register (DAC_CTRLA.ENABLE).

Related Links

VREF - Voltage Reference on page 166

AC – Analog Comparator on page 437

ADC - Analog to Digital Converter on page 448

31.6.2.2. Enabling, Disabling and Resetting

The DAC is enabled by writing a '1' to the Enable bit in the Control A register (DAC_CTRLA.ENABLE), and disabled by writing a '0' to this bit.

The DACOUT output to a pin is enabled by writing the Output Enable bit in the CTRLA register (DAC_CTRLA.OUTEN).

31.6.2.3. Starting a Conversion

When the DAC is enabled (DAC_CTRLA.ENABLE=1), a conversion starts as soon as the Data register (DAC_DATA.DATA) is written.

When the DAC is disabled (DAC_CTRLA.ENABLE=0), writing DAC_DATA.DATA does not trigger a conversion. Instead, the conversion starts on writing a '1' to DAC_CTRLA.ENABLE.

31.6.2.4. DAC As Source For Internal Peripherals

The analog output of the DAC is internally connected to the AC and to the ADC. The DAC output is available when the peripheral is enabled (DAC_CTRLA.ENABLE=1). For internal usage of the DAC output signal it is not required to enable the output buffer, i.e. DAC_CTRLA.OUTEN=0 is acceptable.

Related Links

AC – Analog Comparator on page 437

ADC - Analog to Digital Converter on page 448

31.6.3. Sleep Mode Operation

If the Run in Standby bit in the Control A register (DAC_CTRLA.RUNSTDBY) is written to '1' and CLK_PER is available, the DAC will continue to operation in Standby sleep mode. If the .RUNSTDBY bit is zero, the DAC will stop the conversion in Standby sleep mode.

If conversion is stopped in Standby sleep mode, the DAC and the output buffer are disabled to reduce power consumption. When the device is exiting Standby sleep mode, the DAC and the output buffer (if configured by DAC_CTRLA.OUTEN=1) are enabled again. Therefore, a certain startup time is required before a new conversion is started.

In Power Down sleep mode, the DAC and Output Buffer are disabled to reduce power consumption.

31.6.4. Configuration Change Protection

Not applicable.

31.7. Register Summary

Offset	Name	Bit Pos.								
0x00	CTRLA	7:0	RUNSTDBY	OUTEN						ENABLE
0x01	DATA	7:0	DATA[7:0]							

31.8. Register Description

31.8.1. Control A

Name: CTRLA
Offset: 0x00
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	RUNSTDBY	OUTEN						ENABLE
Access	R/W	R/W						R/W
Reset	0	0						0

Bit 7 – RUNSTDBY: Run in Standby Mode
If this bit is written to '1', the DAC will not automatically disable Analog IP and Output buffer when the device is entering Standby sleep mode.

Bit 6 – OUTEN: Output Buffer Enable
Writing a '1' to this bit enables the Output Buffer and sends the DACOUT signal to a pin.

Bit 0 – ENABLE: DAC Enable
Writing a '1' to this bit enables the DAC.

31.8.2. DATA

Name: DATA
Offset: 0x01
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	DATA[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – DATA[7:0]: Data
These bits contains the digital data which will be converted to an analog voltage.

32. PTC - Peripheral Touch Controller

32.1. Overview

The Peripheral Touch Controller (PTC) acquires signals in order to detect touch on capacitive sensors. The external capacitive touch sensor is typically formed on a PCB, and the sensor electrodes are connected to the analog front end of the PTC through the I/O pins in the device. The PTC supports both self- and mutual-capacitance sensors.

In mutual-capacitance mode, sensing is done using capacitive touch matrices in various X-Y configurations, including indium tin oxide (ITO) sensor grids. The PTC requires one pin per X-line and one pin per Y-line.

In self-capacitance mode, the PTC requires only one pin (Y-line) for each touch sensor.

The number of available pins and the assignment of X- and Y-lines is depending on both package type and device configuration. Refer to the Configuration Summary and I/O Multiplexing table for details.

Related Links

Configuration Summary on page 11
I/O Multiplexing and Considerations on page 19

32.2. Features

- Low-power, high-sensitivity, environmentally robust capacitive touch buttons, sliders, wheels and proximity sensing
- Supports wake-up on touch from power-save sleep mode
- Supports mutual capacitance and self-capacitance sensing
  - Mix-and-match mutual-and self-capacitance sensors
- One pin per electrode – no external components
- Load compensating charge sensing
  - Parasitic capacitance compensation and adjustable gain for superior sensitivity
- Zero drift over the temperature and V_{DD} range
  - Auto calibration and re-calibration of sensors
- Single-shot and free-running charge measurement
- Hardware noise filtering and noise signal de-synchronization for high conducted immunity
- Driven shield for better noise immunity and moisture tolerance
- Selectable channel change delay allows choosing the settling time on a new channel, as required
- Acquisition-start triggered by command or through auto-triggering feature
- Low CPU utilization through interrupt on acquisition-complete
- Using ADC peripheral for signal conversion and acquisition
- Supported by the Atmel® QTouch® Composer development tools. See also Atmel|Start and Atmel Studio documentation.

Related Links

Configuration Summary on page 11
I/O Multiplexing and Considerations on page 19

32.3. Block Diagram

Figure 32-1. PTC Block Diagram Mutual-Capacitance

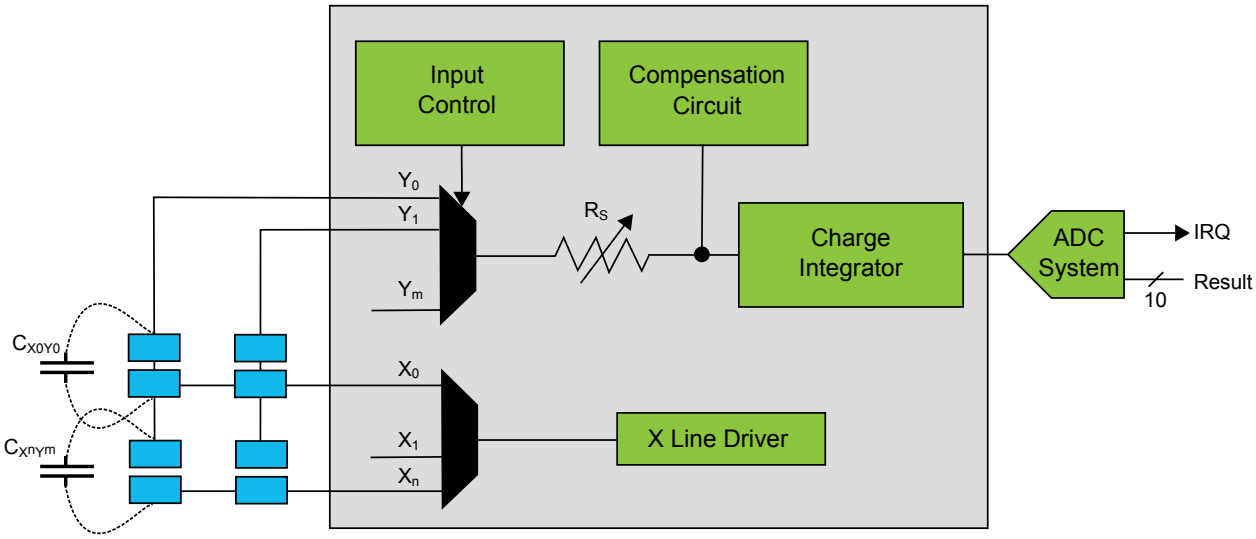
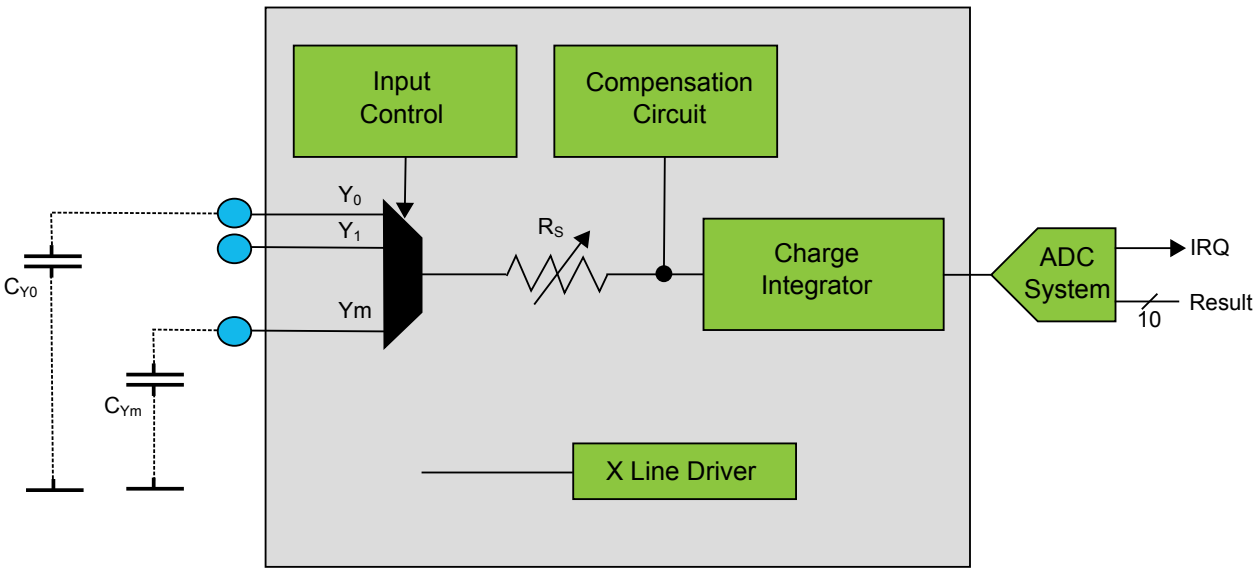


Figure 32-2. PTC Block Diagram Self-Capacitance



32.4. Signal Description

Table 32-1. Signal Description for PTC

Name	Type	Description
Y[m:0]	Analog	Y-line (Input/Output)
X[n:0]	Digital	X-line (Output)
DS[1:0]	Digital	Driven Shield

Note: The number of X and Y lines are device dependent. Refer to *Configuration Summary* for details.

Refer to *I/O Multiplexing and Considerations* for details on the pin mapping for this peripheral. One signal can be mapped on several pins.

Related Links

Configuration Summary on page 11
I/O Multiplexing and Considerations on page 19

32.5. Product Dependencies

In order to use this Peripheral, configure the other components of the system as described in the following sections.

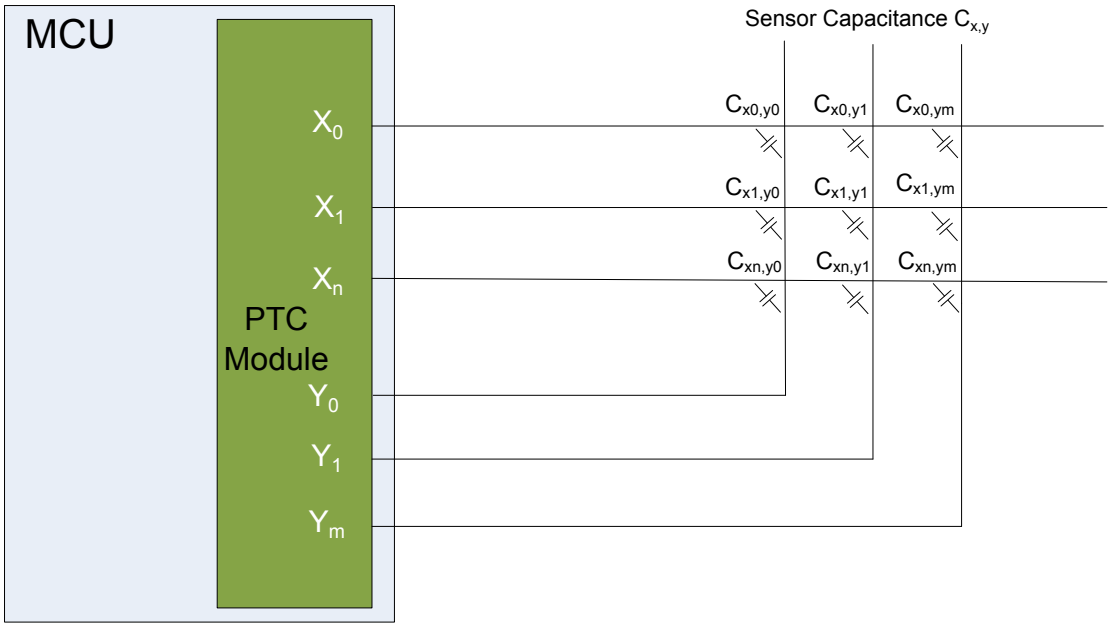
32.5.1. I/O Lines

The I/O lines used for analog X-lines and Y-lines must be connected to external capacitive touch sensor electrodes. External components are not required for normal operation. However, to improve the EMC performance, a series resistor of 1kΩ or more can be used on X-lines and Y-lines.

32.5.1.1. Mutual-capacitance Sensor Arrangement

A mutual-capacitance sensor is formed between two I/O lines - an X electrode for transmitting and Y electrode for receiving. The mutual capacitance between the X and Y electrode is measured by the Peripheral Touch Controller.

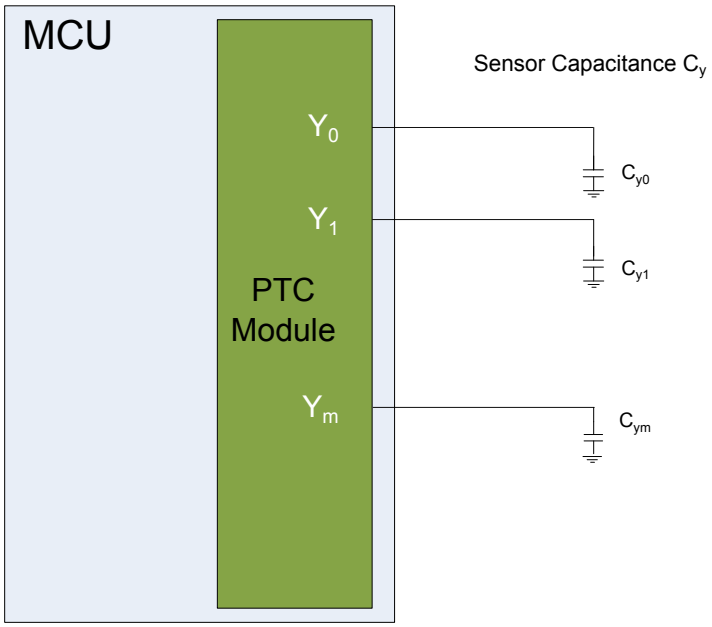
Figure 32-3. Mutual Capacitance Sensor Arrangement



32.5.1.2. Self-capacitance Sensor Arrangement

The self-capacitance sensor is connected to a single pin on the Peripheral Touch Controller through the Y electrode for receiving the signal. The sense electrode capacitance is measured by the Peripheral Touch Controller.

Figure 32-4. Self-capacitance Sensor Arrangement



For more information about designing the touch sensor, refer to Buttons, Sliders and Wheels Touch Sensor Design Guide on http://www.atmel.com.

32.5.2. Clocks

The PTC is clocked by the CLK_PER clock. See the Related Links for details on configuring CLK_PER..

Related Links

CLKCTRL - Clock Controller on page 68

32.5.3. Analog-Digital Converter (ADC)

The PTC is using the ADC of ATtiny416/417/814/816/817 for signal conversion and acquisition. The ADC must be enabled and configured appropriatly in order to allow correct behavior of the PTC.

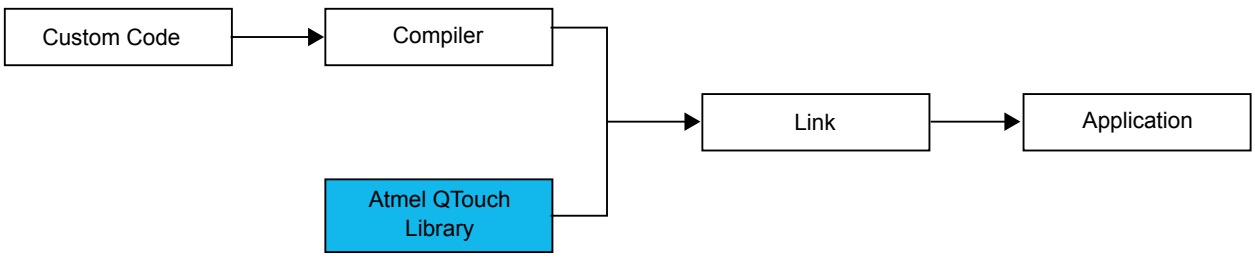
Related Links

ADC - Analog to Digital Converter on page 448

32.6. Functional Description

In order to access the PTC, the user must use the QTouch Composer tool to configure and link the QTouch Library firmware with the application code. QTouch Library can be used to implement buttons, sliders, wheels and proximity sensor in a variety of combinations on a single interface.

Figure 32-5. QTouch Library Usage



For more information about QTouch Library, refer to the Atmel QTouch Library Peripheral Touch Controller User Guide.

33. UPDI - Unified Program Debug Interface

Related Links

Debug Operation on page 99

33.1. Overview

The Unified Program and Debug Interface (UPDI) is an Atmel proprietary interface for external programming and on-chip debugging of a device.

The UPDI supports programming of nonvolatile memory (NVM) space; FLASH, EEPROM, fuses, lockbits and the user row. In addition the UPDI can access the entire I/O and data space of the device. See the NVM Controller documentation for programming via the NVM controller and executing NVM controller commands.

Programming and debugging is done through the UPDI Physical interface (UPDI PHY), which is by default a 1-wire UART based half-duplex interface using the Reset pin for data reception and transmission. Clocking is done by an internal 32MHz oscillator. Enabling of the 1-wire interface, by disabling the reset functionality, is either done by 12V programming or by setting a Fuse. Optionally, a 2-wire interface using a GPIO as a separate TX-pin can be enabled, with the PDI pin still used for the RX-part of the communication. The UPDI Access layer grants access to the system bus matrix, with memory mapped access to system blocks such as Memories, NVM, and peripherals.

The Asynchronous System Interface (ASI) is providing direct interface access to On-Chip Debugging (OCD), NVM and System Management features. This gives the debugger direct access to system information, without accessing though the bus. The ASI is also responsible for setting up and executing the system KEYs.

Related Links

NVMCTRL - Non Volatile Memory Controller on page 52
Enabling of KEY Protected Interfaces on page 505

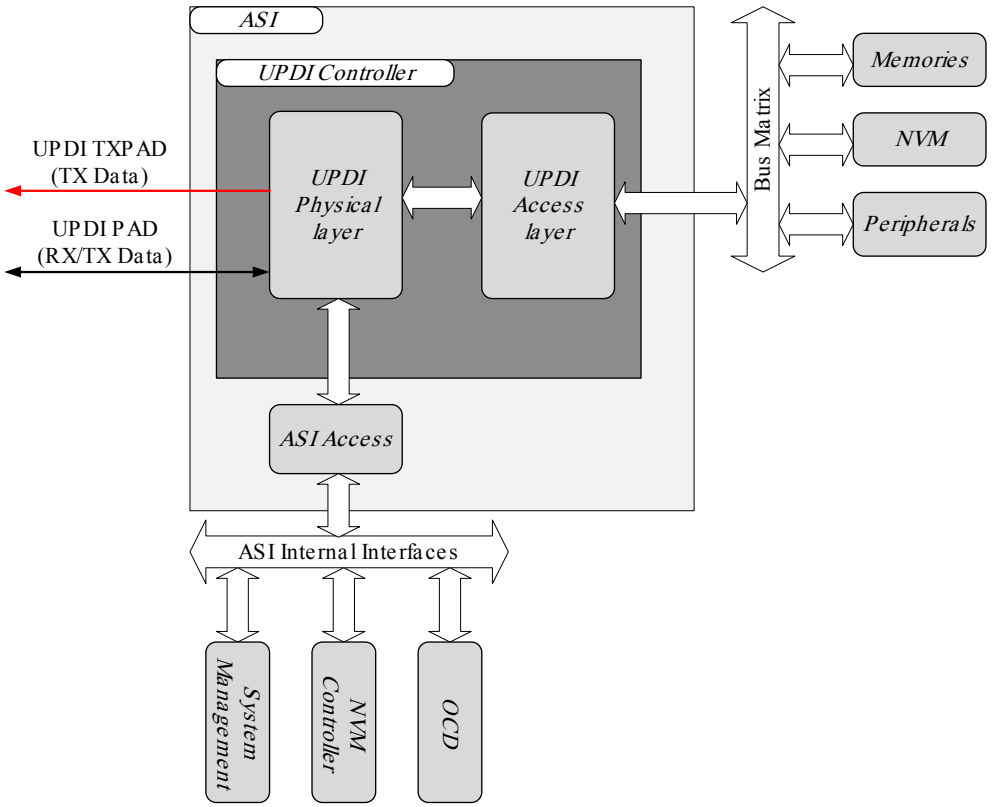
33.2. Features

- Programming
  - External programming through PDI 1-wire (1W) interface
    - Enable programming by 12V or fuse
    - Uses the $\overline{\text{RESET}}$ pin of the device for programming
    - No GPIO pins occupied during operation
    - Asynchronous Half-Duplex UART protocol towards the emulator
    - Optional programming with 2-wire asynchronous mode, using one additional GPIO pin
- Debugging
  - Memory mapped access to device address space (NVM, RAM, I/O)
  - Run-time readout of program counter (PC), Stack Pointer (SP) and CPU Status register (CPU_SREG) for code profiling
  - Non-intrusive run-time chip monitoring without accessing system registers
    - Monitor CRC status and sleep status
  - Support for power-aware Debugging
- Programming and Debug Interface (PDI)

–
– Default 1-wire interface, with optional enable of 2-wire asynchronous interface for programming and debug
– Reset pin used as RX/TX-pin in 1-wire mode. GPIO pin used as TX-pin in 2-wire mode.
– Built in error detection with error signature readout
– Frequency Measurement of internal oscillators using theEvent System

33.3. Block Diagram

Figure 33-1. UPDI Block Diagram



33.4. Product Dependencies

In order to use this peripheral, other parts of the system must be configured correctly, as described below.

Table 33-1. UPDI Product Dependencies

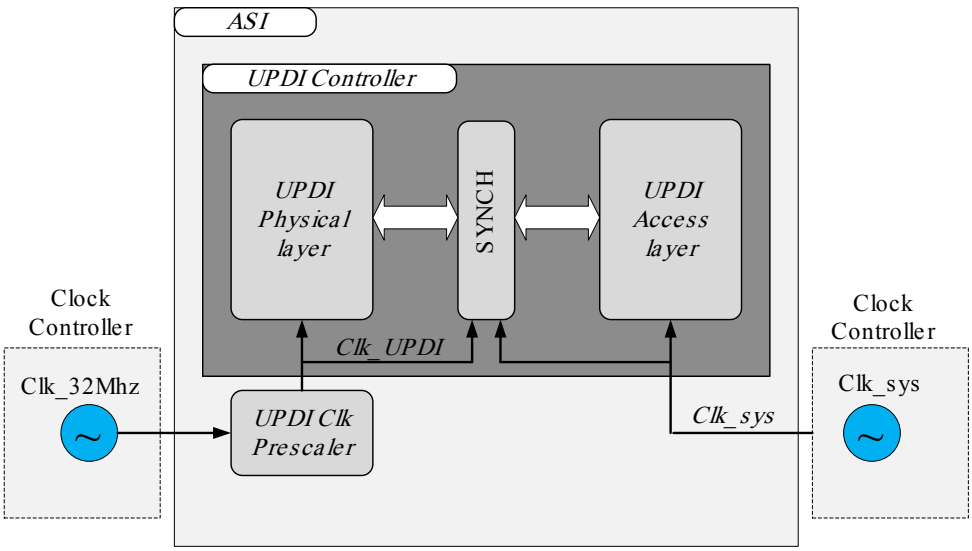
Dependency	Applicable	Peripheral
Clocks	Yes	CLKCTRL
I/O Lines and Connections	Yes	PORT
Interrupts	No	-
Events	Yes	EVSYS
Debug	Yes	UPDI

Related Links

33.4.1. Clocks

The UPDI and ASI operate on two separate clock domains. The UPDI Physical layer clock is derived from an internal 32MHz oscillator, and the UPDI access layer clock is the same as the system clock. There is a synchronization boundary between the UPDI physical layer and access layer, which ensures correct operation for different clock frequencies on each domain. The UPDI clock is prescaled in the ASI, and the default startup frequency is 4MHz (DIV8 prescaling) after enabling the UPDI. The UPDI prescaling is changed by writing the UPDICLKDIV bits in ASI_CTRLA register.

Figure 33-2. UPDI clock domains



33.4.2. I/O Lines and Connections

To operate the UPDI, the RST Pin must be set to UPDI Mode. This is not done through the PORT I/O Pin Controller as for regular I/O Pins, but through setting the correct FUSE_SYSCFG1.RSTPINCFG register as described in UPDI Enable with RESET Pin Override, or by following the UPDI 12V enable sequence from UPDI Enable by 12V Programming. Pull enable, input enable and output enable settings are automatically controlled by the UPDI when active.

33.4.3. Interrupts

Not applicable.

33.4.4. Events

The events of this peripheral are connected to the Event System.

Related Links

EVSYS - Event System on page 109

33.4.5. Power Management

The UPDI physical layer continues to operate in any sleep mode and is always accessible for a connected debugger, but read/write access to the system bus is restricted in sleep modes where the CPU clock is switched off. The UPDI can be enabled at any time, independent of the system sleep state. See Sleep Mode Operation for details on UPDI operation duing sleep modes.

33.4.6. Debug Operation

When run-time debugging, this peripheral will continue normal operation. Halting the CPU in debugging mode will halt normal operation of the peripheral.

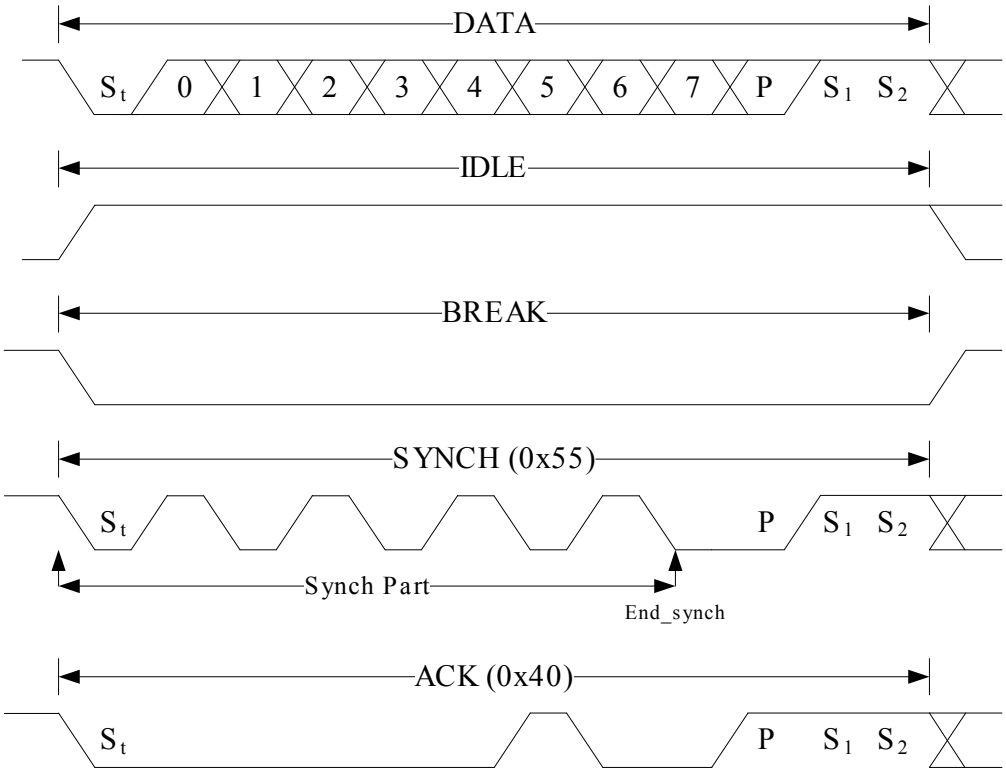
If the peripheral is configured to require periodical service by the CPU through interrupts or similar, improper operation or data loss may result during halted debugging.

33.5. Functional Description

33.5.1. Principle of Operation

Communication through the UPDI is based on standard UART communication, using a fixed frame format, and automatic baud rate detection for clock and data recovery. In addition to the data frame, there are several control frames which are important to the communication. The supported frame formats are presented in Figure 33-3.

Figure 33-3. Supported UPDI Frame formats



Data Frame Consist of 1 start bit (always low), 8 data bits, 1 parity bit (even parity) and two stop bits (always high). If the start bit, parity bit or stop bits have an incorrect value, an error will be detected and signalized by the UPDI. The parity bit check in the UPDI can be disabled by writing the UPDI_CTRLA.PARD bit, in which case the parity generation from the debugger can be ignored.

IDLE Frame Special Frame which consist of 12 high bits. This is the same as keeping the transmission line in an IDLE state.

BREAK The BREAK frame is used to reset the UPDI back to its default state, and is typically used for error recovery.

SYNCH The SYNCH frame (0x55) is used by the baud rate generator to set the baud rate for the coming transmission. A SYNCH character is always expected by the UPDI in front of every new instruction, and after a successful BREAK has been transmitted.

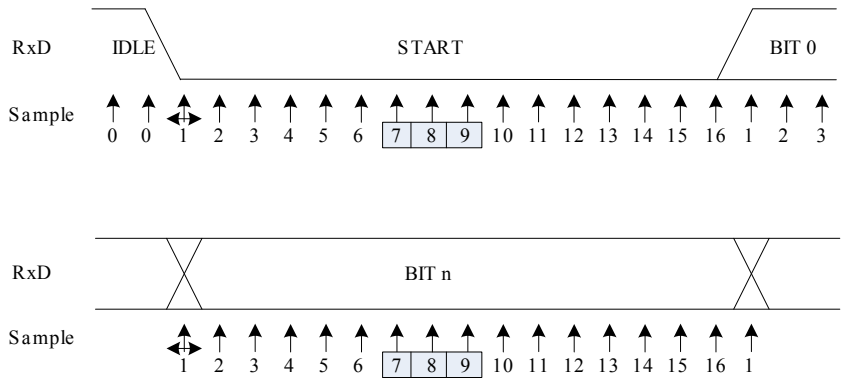
ACK The Acknowledge (ACK) character is transmitted from the UPDI whenever a ST or STS instruction has successfully crossed the synchronization border, and have gained bus access. When an ACK is received by the debugger, the next transmission can start.

33.5.1.1. UPDI UART

All transmission and reception of serial data on the UPDI is achieved using the UPDI frames presented in Figure 33-3. Communication is initiated from the master (debugger) side, and every transmission must start with a SYNCH character upon which the UPDI can recover the transmission baud rate, and store this setting for the coming data. The baud rate set by the SYNCH character will be used for both reception and transmission for the instruction byte received after the SYNCH. See UPDI Instruction Set for details on when the next SYNCH character is expected in the instruction stream.

There is no writable baud rate register in the UPDI, so the baud rate sampled from the SYNCH character is used for data recovery by sampling the start bit, and do a majority voting on the middle samples. This process is repeated for all bits in the frame, including the parity bit and two stop bits. The baud generator uses 16 samples, and the majority voting is done on sample 7,8 and 9.

Figure 33-4. UPDI UART Start Bit and Data/Parity/Stop bit sampling



The transmission Baud Rate must be set up in relation to the currently used UPDI clock prescaler, which can be adjusted in the UPDI_ASI_CTRLA.UPDICLKDIV. See Table 33-2 for recommended maximum and minimum baud rate settings.

Table 33-2. Recommended UART Baud Rate based on UPDICLKDIV setting

UPDICLKDIV[1:0]	MAX Recommended Baud Rate	MIN Recommended Baud Rate
0x0 (32MHz)	1.8Mbps	0.600kbps
0x1 (16MHz)	0.9Mbps	0.300kbps
0x2 (8MHz)	450kbps	0.150kbps
0x3 (4MHz) - Default	225kbps	0.075kbps

The UPDI baud rate generator utilizes fractional baud counting to minimize the transmission error. With the fixed frame format used by the UPDI, the maximum and recommended receiver transmission error limits can be seen in the following table:

Table 33-3. Receiver Baud Rate Error

Data + parity bits	R_{slow}	R_{fast}	Max Total Error (%)	Recommended max RX error (%)
9	96.39	104.76	+4.76 / -3.61	+1.5 / -1.5

33.5.1.2. BREAK Character

The BREAK character is used to reset the internal state of the UPDI to the default setting. This is useful if the UPDI enters an error state due to communication error, or when synchronization between the debugger and the UPDI is lost.

A single BREAK character is enough to reset the UPDI, but in some special cases where the BREAK character is sent when the UPDI has not yet entered the error state, a double BREAK character might be needed. A double BREAK is guaranteed to reset the UPDI from any state. Note that when sending a double BREAK it is required to have at least one stop bit between the BREAK characters.

No SYNCH character is required before the BREAK, because the BREAK is used to reset the UPDI from any state. This means that the UPDI will sample the BREAK based on the last stored baud rate setting, derived from the last received valid SYNCH character. If the communication error was due to incorrect sampling of the SYNCH character, the baud rate is unknown to the connected debugger. For this reason the BREAK character should be transmitted at the slowest recommended baud rate setting for any particular UPDI oscillator setting according to Table 33-4:

Table 33-4. Recommended BREAK Character Duration

UPDICLKDIV[1:0]	Recommended BREAK Character Duration
0x0 (32MHz)	3.075ms
0x1 (16MHz)	6.15ms
0x2 (8MHz)	12.30ms
0x3 (MHz) - Default	24.60ms

33.5.2. Basic Operation

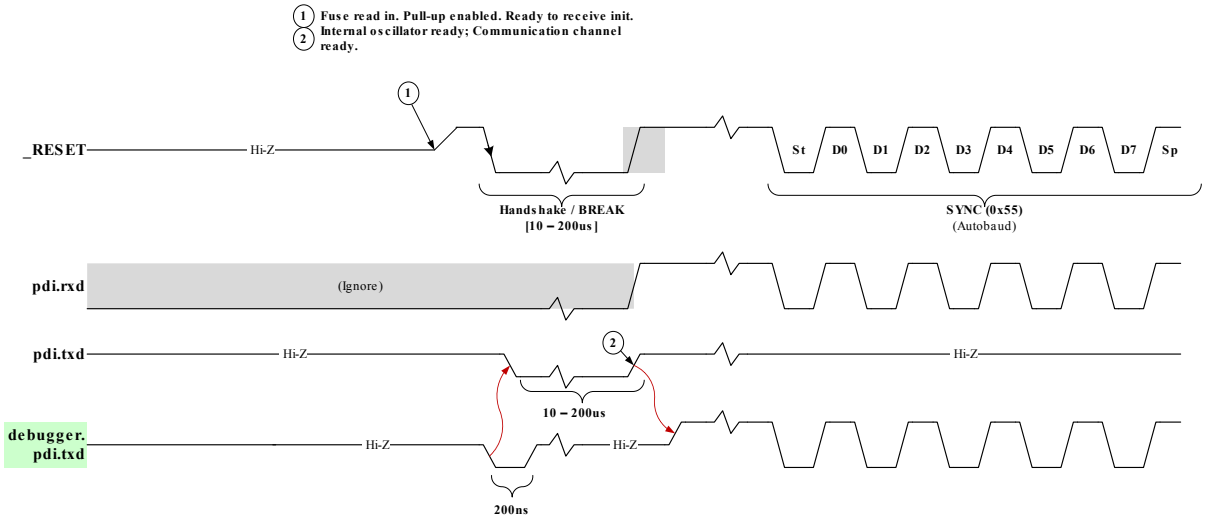
The UPDI must be enabled before the UART communication can start:

- UPDI Enable by 12V Programming
- UPDI Enable with RESET Pin Override

33.5.2.1. UPDI Enable with RESET Pin Override

When the Reset Pin Configuration bits in the System Configuration 0 fuse (FUSE_SYSCFG0.RSTPINCFG) are 0x1, the RESET Pin will be overridden by the UPDI Pin during the system boot sequence: the UPDI will take control of the Pin and configure it as input with pull-up. When the pull-up is detected by a connected debugger, the initial handshake sequence to enable the UPDI, as depicted below, is started.

Figure 33-5. UPDI Enable sequence with UPDI PAD Enabled by Fuse



When the pull-up is detected, the Emulator initiates the enable sequence by driving the line low for a minimum of 200ns. The duration can be longer, but the data line should be released from the debugger before the UPDI startup sequence is done.

The negative edge is detected by the UPDI, which requests the peripheral clock. The UPDI will continue to drive the line low until the clock is stable and ready for the UPDI to use. The duration of this will vary, depending on the status of the oscillator and when the UPDI is enabled relative to the boot sequence. A startup time between 10us and 200us can be expected. After this duration, the data line will be released by the UPDI, and pulled-high.

When the Debugger detects that the line is high, the initial SYNCH character (0x55) must be sent to properly enable the UPDI for communication. If the start bit of the SYNCH character is not sent within 13ms, the UPDI will disable itself, and the startup sequence must be repeated. This time is based on counted cycles on the 4MHz UPDI clock, which is default when enabling the UPDI. The disable is performed to avoid the UPDI being enabled unintentionally.

After successful SYNCH character transmission, the first instruction frame can be transmitted.

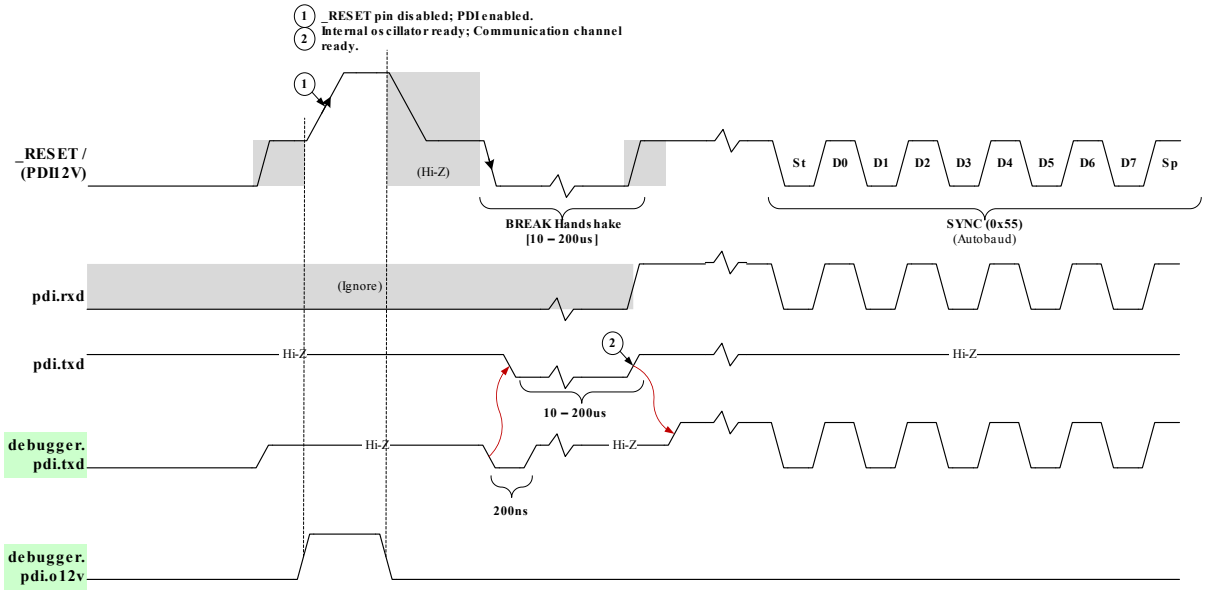
33.5.2.2. UPDI Enable by 12V Programming

GPIO or Reset functionality on the RESET Pin can be overridden by the UPDI by using 12V programming. By applying a 12V pulse to the RESET Pin, the pin functionality is switched to UPDI, independent of FUSE_SYSCFG0.RSTPINCFG. It is recommended to always reset the device before starting the 12V enable sequence.

After startup, the Power-on Reset (POR) must be released before the 12V pulse can be applied. The duration of the pulse is recommended in the range from 100us to 1ms, before tri-stating. When applying the rising edge of the 12V pulse, the UPDI will be reset. After tri-stating, the UPDI will remain in Reset until the RESET Pin is driven low by the debugger. This will release the UPDI Reset, and start the initial handshake sequence, as for UPDI Enable with RESET Pin Override. The rest of the enable sequence follows the same pattern as if UPDI is enabled with RESET Pin Override.

The following figure shows the 12V enable sequence. The `ireset` signal is the UPDI internal reset.

Figure 33-6. UPDI Enable Sequence by 12V Programming



When enabled by 12V, only a POR will disable the UPDI configuration on the RESET Pin, and restore the default setting. If issuing a UPDI disable command through the UPDI_CTRLB.UPDIDIS bit, the UPDI will be reset and the clock request will be canceled, but the RESET Pin will remain in UPDI configuration.

Output Enable Timer Protection for GPIO Configuration

When the Reset pin Configuration bits in the System Configration 0 fuse (FUSE_SYSCFG0.RSTPINCFG) are 0x0, the RESET Pin configured as GPIO. To avoid the potential conflict between the GPIO actively driving the output and a 12V UPDI enable sequence initiation, a timer protection is disabling the output enable for approximately 10ms after each System Reset.

Note: It is always recommended to issue a System Reset before entering the 12V programming sequence.

33.5.2.3. UPDI Disable

Any programming or debug session should be terminated by writing the UPDI_CTRLB.UPDIDIS bit. Writing this bit will reset the UPDI including any decoded KEYs, and disable the oscillator request for the module. If the disable operation is not performed, the UPDI will stay enabled and request its oscillator, causing increased power consumption for the application.

During the enable sequence the UPDI can disable itself in case of a faulty enable sequence. There are two cases which will cause an automatic disable.

- A SYNCH character is not sent within 16.4ms after the initial enable pulse described in UPDI Enable with RESET Pin Override.
- The first SYNCH character after an initiated enable is too short or too long to register as a valid SYNCH character. See Table 33-2 for recommended baud rate operating ranges.

33.5.2.4. UPDI Communication Error Handling

The UPDI contains a comprehensive error detection system, to be able to provide information to the debugger during a potential communication loss. The error detection consist of detecting physical transmission errors like start bit error, parity error, contention error and frame error, to more high level errors like access timeout error. See UPDI_STATUSB.PESIG for an overview of the available error signatures.

Whenever the UPDI detects an error, it will immediately transfer to an internal error state to avoid unwanted system communication. In the error state the UPDI will ignore all incoming data requests,

except if a BREAK character is transmitted. The following procedure should always be applied when recovering from an error condition.

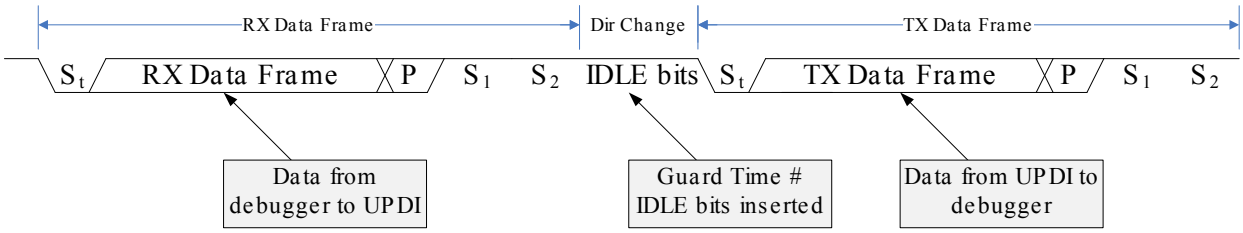
- Send a BREAK character. See BREAK Character for recommended BREAK character handling.
- Send a SYNCH character at the desired baud rate for the next data transfer. Note that upon receiving a BREAK, the UPDI oscillator setting in UPDI_ASI_CTRLA is reset to the slowest default setting of 4MHz prescaling. This affects the baud rate range of the UPDI according to Table 33-2 .
- Do an LDCS to UPDI_STATUSB register to read the PESIG field. PESIG will give information about the occurred error, and the error signature will be cleared when read.
- The UPDI is now recovered from the error state, and ready to receive the next SYNCH character and instruction.

33.5.2.5. Direction Change

In order to ensure correct timing for half-duplex UART operation, the UPDI has a build in Guard Time mechanism to relax the timing when changing direction from RX mode to TX mode. The Guard Time is a number of IDLE bits inserted before the next start bit is transmitted. The number of IDLE bits can be configured through UPDI_CTRLA.GTVAL. The duration of each IDLE bit is given by the baud rate used by the current transmission.

Note: It is not recommended to use GTVAL setting 0x7, with no additional IDLE bits. This can cause the UPDI to not respond correctly for some instructions.

Figure 33-7. UPDI Direction Change by inserting IDLE bits

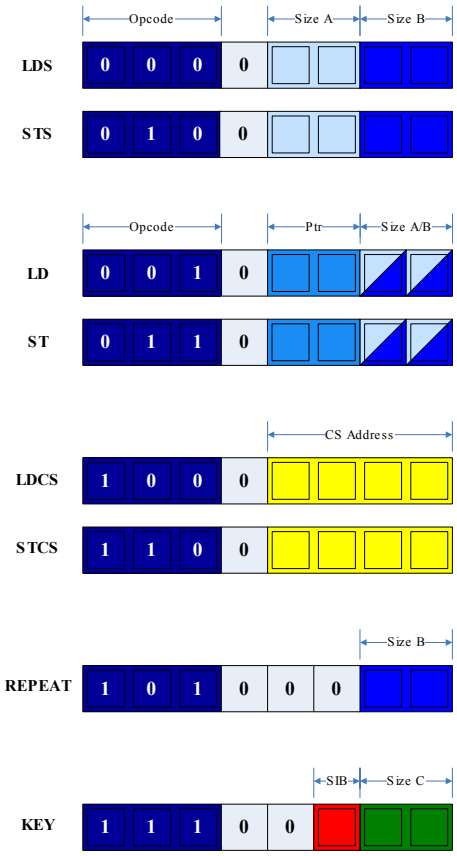


The UPDI Guard Time is the minimum IDLE time that the connected debugger will experience when waiting for data from the UPDI. Because of the asynchronous interface to the system, as presented in Clocks, the ratio between the UPDI baud rate and the system clock will affect the synchronization time, and how long it takes before the UPDI can transmit data. In the cases where the synchronization delay is shorter than the current Guard Time setting, the Guard Time will be given by GTVAL directly.

33.5.3. UPDI Instruction Set

Communication through the UPDI is based on a small instruction set. The instructions are used to access the internal UPDI and ASI Control and Status (CS) space, as well as the memory mapped system space. All instructions are byte instructions, and must be preceded by a SYNCH character to determine the baud rate for the communication. See UPDI UART for information about setting the baud rate for the transmission. The following Figure gives an overview of the UPDI instruction set.

Figure 33-8. UPDI Instruction Set Overview



OPCODE	
0 0 0	LDS
0 0 1	LD
0 1 0	STS
0 1 1	ST
1 0 0	LDCS (LDS Control/Status)
1 0 1	REPEAT
1 1 0	STCS (STS Control/Status)
1 1 1	KEY

Size A - Address size	
0 0	Byte
0 1	Word (2 Bytes)
1 0	Reserved
1 1	Reserved

Ptr - Pointer access	
0 0	*(ptr)
0 1	*(ptr++)
1 0	ptr
1 1	Reserved

Size B - Data size	
0 0	Byte
0 1	Word (2 Bytes)
1 0	Reserved
1 1	Reserved

CS Address (CS - Control/Status reg.)	
0 0 0 0	Reg 0
0 0 0 1	Reg 1
0 0 1 0	Reg 2
0 0 1 1	Reg 3
0 1 0 0	Reg 4 (ASI CS space)
.....	
1 1 1 1	Reg 15 (ASI CS space)

Size C - Key size	
0 0	64 bits (8 Bytes)
0 1	128 bits (16 Bytes)
1 0	Reserved
1 1	Reserved

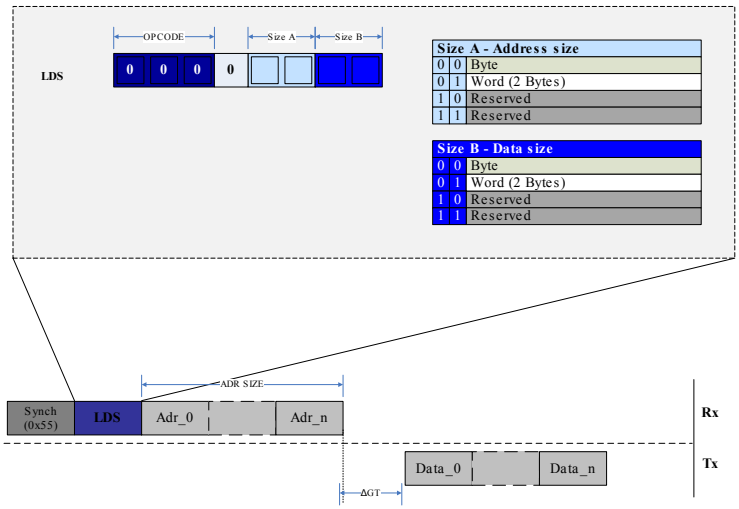
SIB – System Information Block sel.	
0	Receive KEY
1	Send SIB

33.5.3.1. LDS - Load Data from Data Space Using Direct Addressing

The LDS instruction is used to load data form the bus matrix and into the serial shift register for serial read out. The LDS instruction is based on direct addressing, and the address must be given as an operand to the instruction for the data transfer to start. Maximum supported size for address and data is 16 bit. LDS supports repeated memory access when combined with the REPEAT instruction.

The following figure shows the operands of the LDS instruction and a typical transmission example.

Figure 33-9. LDS Instruction Operation

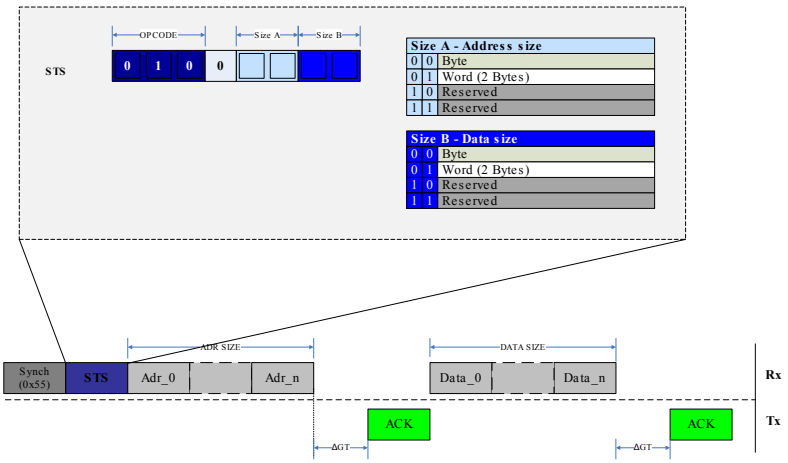


33.5.3.2. STS - Store Data to Data Space Using Direct Addressing

The STS instruction is used to store data that is shifted serially into the PHY layer to the bus matrix address space. The STS instruction is based on direct addressing, where the address is the first set of operands, and data is the second set. The size of the address and data operands are given by the size fields presented in the Figure below. Maximum size for both address and data is 16 bit.

STS supports repeated memory access when combined with the REPEAT instruction.

Figure 33-10. STS Instruction Operation



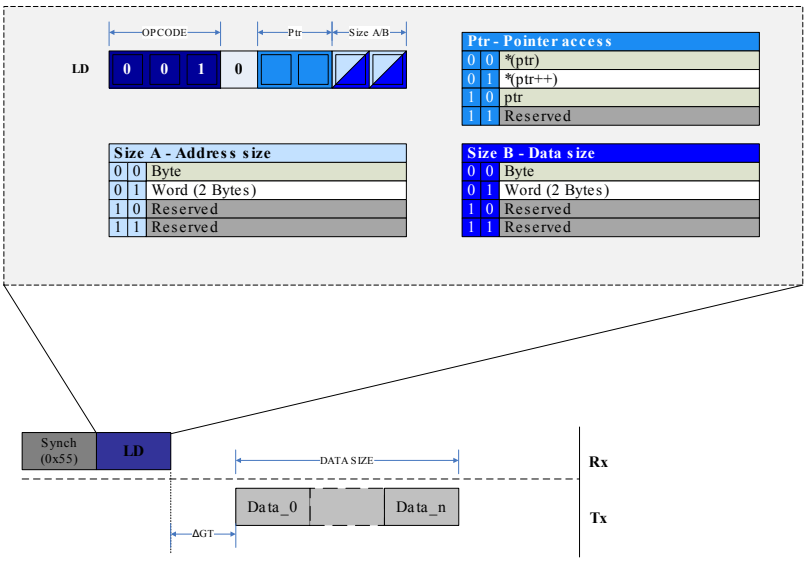
The transfer protocol for an STS instruction is depicted in the Figure as well, following this sequence:

1. The address is sent.
2. An Acknowledge (ACK) is sent back from the UPDI if the transfer was successful.
3. The data can is sent.
4. A new ACK is received after the data is successfully transferred.

33.5.3.3. LD - Load Data from Data Space Using Indirect Addressing

The LD instruction is used to load data form the bus matrix and into the serial shift register for serial read out. The LD instruction is based on indirect addressing, which means that the address pointer in the UPDI needs to be written prior to bus matrix access. Automatic pointer post increment operation is supported, and is useful when the LD instruction is used with REPEAT. It is also possible to do a LD of the UPDI pointer register. Maximum supported size for address and data load is 16 bit.

Figure 33-11. LD Instruction Operation

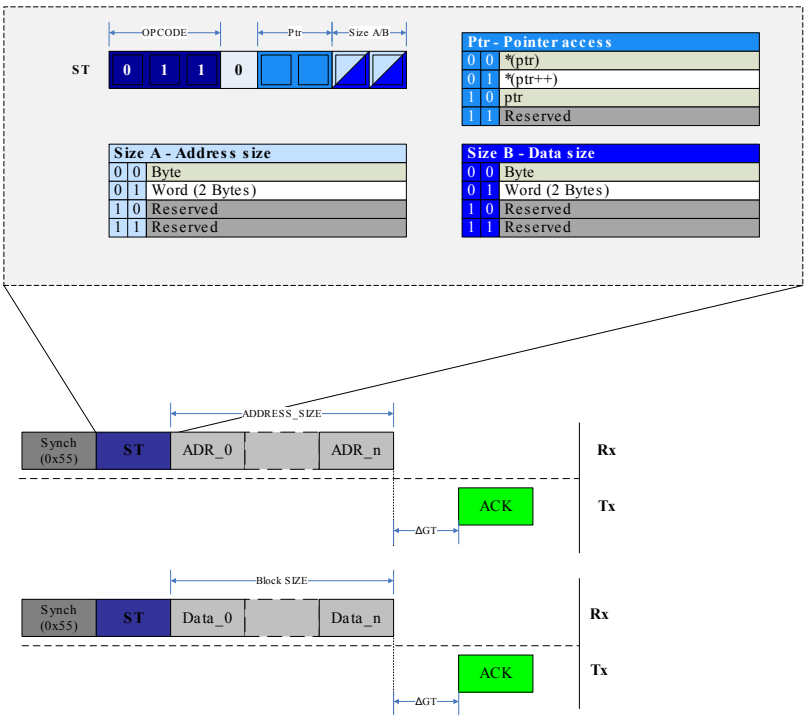


The Figure above shows an example of a typical LD sequence, where data is received after the Guard Time period. Loading data from the UPDI pointer register follows the same transmission protocol.

33.5.3.4. ST - Store Data from Data Space Using Indirect Addressing

The ST instruction is used to store data that is shifted serially into the PHY layer to the bus matrix address space. The ST instruction is based on indirect addressing, which means that the address pointer in the UPDI needs to be written prior to bus matrix access. Automatic pointer post increment operation is supported, and is useful when the ST instruction is used with REPEAT. ST is also used to store the UPDI address pointer into the pointer register. Maximum supported size for address and data load is 16 bit.

Figure 33-12. ST Instruction Operation

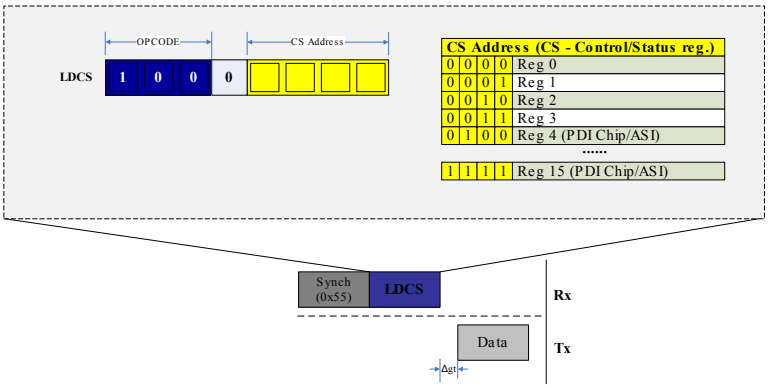


The Figure above gives an example of ST to the UPDI pointer register, and store of regular data. Note that in both cases an Acknowledge (ACK) is sent back by the UPDI if the store was successful.

33.5.3.5. LCDS - Load Data from Control and Status Register Space

The LCDS instruction is used to load data from the UPDI and ASI CS-space. LCDS is based on direct addressing, where the address is part of the instruction opcode. The total address space for LCDS is 16 byte, and can only access the internal UPDI register space. This instruction only supports byte access, and data size is not configurable.

Figure 33-13. LCDS Instruction Operation



The Figure above shows a typical example of LCDS data transmission. A data byte from the LCDS space is transmitted from the UPDI after the Guard-Time is completed.

33.5.3.6. STCS (Store Data to Control and Status register space)

The STCS instruction is used to store data to the UPDI and ASI CS-space. STCS is based on direct addressing, where the address is part of the instruction opcode. The total address space for STCS is 16 byte, and can only access the internal UPDI register space. This instruction only supports byte access, and data size is not configurable.

Figure 33-14. STCS instruction operation

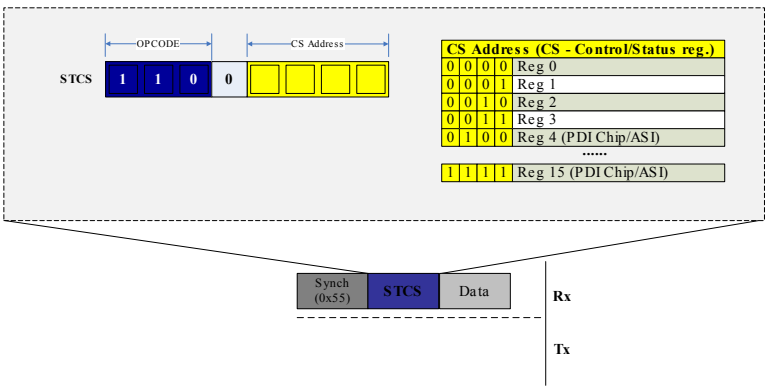


Figure 33-14 shows the data frame transmitted after the SYNCH and instruction frames. Note that there is no response generated from the STCS instruction, as is the case for ST and STS.

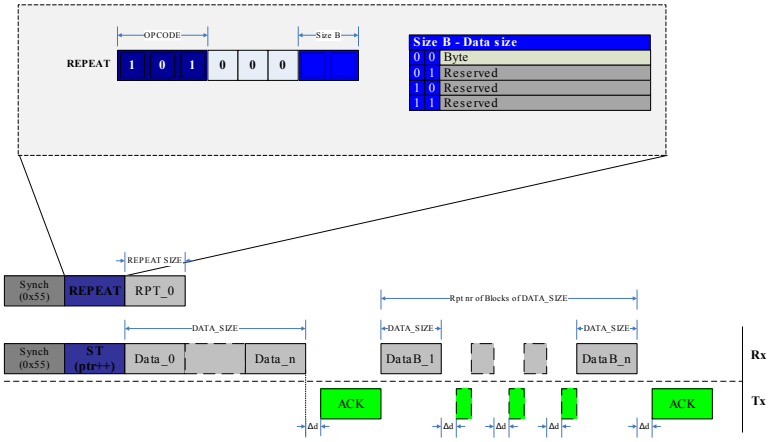
33.5.3.7. REPEAT - Set Instruction Repeat Counter

The REPEAT instruction is used to store the repeat count value into the UPDI repeat counter register. When instructions are used with REPEAT, protocol overhead for SYNCH- and Instruction Frame can be omitted on all instructions except the first instruction after the REPEAT is issued.

The DATA_SIZE opcode field refers to the size of the repeat value. Only byte size (up to 255 repeats) is supported. The instruction that is loaded directly after the REPEAT instruction will be repeated RPT_0

times. The instruction will be issued a total of RPT_0 + 1 times. An ongoing repeat can only be aborted by sending a BREAK character.

Figure 33-15. REPEAT Instruction Operation

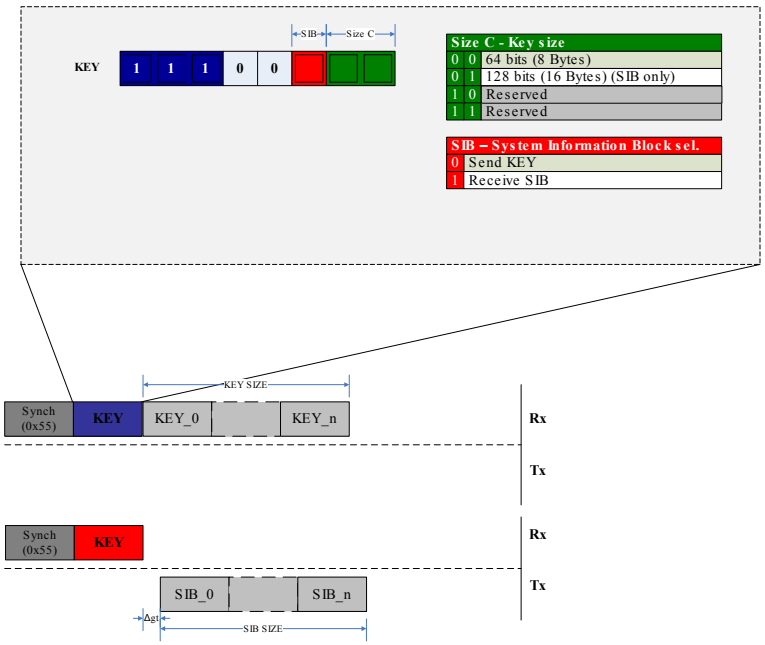


The Figure above gives an example of repeat operation with a ST instruction using pointer post increment operation. After the REPEAT instruction is sent with RPT_0 = ***n***, the first ST instruction is issued with SYNCH- and Instruction frame, while the next ***n*** ST-instructions are executed by only sending in data bytes according to the ST operand DATA_SIZE, and maintaining the Acknowledge (ACK) handshake protocol.

33.5.3.8. KEY - Set Activation KEY

The KEY instruction is used for communicating KEY bytes to the UPDI, opening up for executing protected features on the device. See Table 33-5for an overview over functions that are activated by KEYs. For the KEY instruction, only 64bit KEY size is supported. If the System Information Block (SIB) field of the KEY instruction is set, the KEY instruction returns the SIB instead of expecting incoming KEY bytes. Maximum supported size for SIB is 256bits.

Figure 33-16. KEY Instruction Operation



The Figure above shows the transmission of a KEY, and the reception of a SIB. In both cases, the SIZE_C field in the opcode determines the number of frames to be sent or received. Note that there is no response after sending a KEY to the UPDI. When requesting the SIB, data will be transmitted from the UPDI according to the current Guard Time setting.

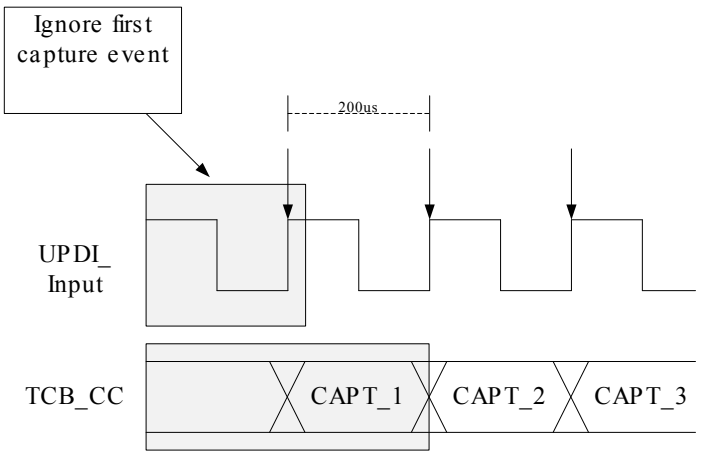
33.5.4. Additional Features

33.5.4.1. System Clock Measurement with UPDI

It is possible to use the UPDI to get a accurate measurement of the system clock frequency, by using the UPDI event connected to TCB with Input Capture capabilities. The next steps gives a recommended setup flow for this feature.

- Set up TCB_CTRLB.CNTMODE with setting 0x3, Input Capture frequency measurement mode. See for details.
- Set TCB_EVCTRL.CAPTEI to enable Event Interrupt. Keep TCB_EVCTRL.EDGE = 0.
- Configure the Event System as described in Events
- For the SYNCH character used to generate the UPDI events, it is recommended to use a slow baud frequency in the range of 10 - 50kHz to get a more accurate measurement on the value captured by the timer between each UPDI event. One particular thing to note, is that if the capture is set up to trigger an interrupt, the first captured value should be ignored. It is the second captured value based on the input event that should be used for the measurement. See Figure 33-17 for an example, using 10kHz UPDI SYNCH character pulses, giving a capture window of 200us for the Timer.
- It is possible to read out the captured value directly after the SYNCH character, by reading the TCB_CC register, or the value can be written to memory by the CPU once the capture is done.

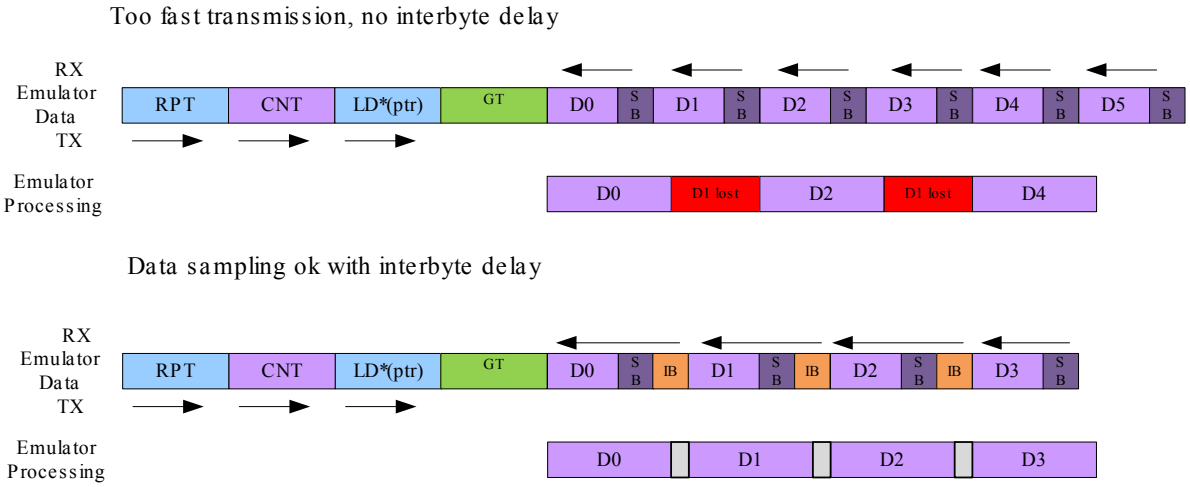
Figure 33-17. UPDI System Clock Measurement Events



33.5.4.2. Interbyte Delay

When loading data with the UPDI, or reading out the System Information Block, the output data will normally come out with two IDLE bits between each transmitted byte for a multibyte transfer. Depending on the application on the receiver side, data might be coming out too fast when there is no extra IDLE bits between each byte. By enabling the UPDI_CTRLB.IBDLY feature, two extra stop bits will be inserted between each byte to relax the sampling time for the debugger. Interbyte delay works in the same way as guard time, by inserting extra IDLE bits, but only a fixed number of IDLE bits and only for multibyte transfers. Note that the first transmitted byte after a direction change will be subject to the regular Guard Time before it is transmitted, and the interbyte delay is not added to this time.

Figure 33-18. Interbyte Delay exemple with LD and RPT



In Figure 33-18, GT denotes the Guard Time insertion, SB are stop bits and IB is the inserted interbyte delay. The rest of the frames are data and instructions.

33.5.5. System Information Block

The System Information Block (SIB) can be read out at any time by setting the SIB bit in the KEY instruction from KEY - Set Activation KEY. The SIB provides a compact form of providing information for the debugger which is vital in identifying and setting up the proper communication channel with the part. The output of the SIB should be interpreted as ASCII symbols. KEY size field should be set to 16byte when reading out the complete SIB, and 8 byte size can be used to read out only the Device_ID. See Figure 33-19 for SIB format description, and which data that is available at different readout sizes.

Figure 33-19. System Information Block format

16	8	[Byte][Bits]	Field Name
		[6:0] [55:0]	Device_ID
		[7][7:0]	Reserved
		[10:8][23:0]	NVM_VERSION
		[13:11][23:0]	OCD_VERSION
		[14][7:0]	RESERVED
		[15][7:0]	DBG_OSC_FREQ

33.5.6. Enabling of KEY Protected Interfaces

Access to some internal interfaces and features are protected by the UPDI KEY mechanism. To activate a KEY, the correct KEY data must be transmitted by using the `KEY` instruction as described in KEY instruction. Table 33-5 describes the available KEYs, and the condition required when doing operation with the KEY active. There is no requirement when shifting in the KEY, but you would for instance normally run a Chiperase before enabling the OCD KEY, to unlock the device for debug. But if the OCD KEY is shifted in first, it will not be reset by shifting in the Chiperase KEY afterwards.

Table 33-5. KEY Activation Overview

KEY name	Description	Requirements for operation	Reset
Chiperase	Start NVM Chiperase. Clear Lockbits	None	UPDI Disable / UPDI Reset
OCD	Activate On Chip Debug features.	Lockbits cleared	UPDI Disable / UPDI Reset
NVMPROG	Activate NVM Programming	Lockbits Cleared. ASI_SYS_STATUS.NVMPROG set.	Programming Done / UPDI Reset
USERROW-Write	Program User Row on Locked part	Lockbits Set. ASI_SYS_STATUS.UROWPROG set	Write to KEY status bit / UPDI Reset
2-Wire Mode	Enable 2-Wire mode with separate TX-pin.	UPDI enabled in 1W mode.	UPDI Disable / UPDI Reset

Table 33-6 gives an overview of the available KEY signatures that must be shifted in to activate the interfaces.

Table 33-6. KEY Activation Signatures

KEY name	KEY signature (LSB written first)	Size
Chiperase	0x4E564D4572617365	64bit
OCD	0x4F43442020202020	64bit
NVMPROG	0x4E564D50726F6720	64bit
USERROW-Write	0x4E564D5573267465	64bit
2-Wire Mode	0x5044493A41325720	64bit

For some KEY protected interfaces, special enable sequences must be followed. See the NVM documentation for details.

33.5.6.1. Chip Erase

The following steps should be followed to issue a Chip Erase.

1. Enter the CHIPERASE KEY by using the `KEY` instruction. See Table 33-6 for the CHIPERASE signature.
2. **Optional:** Read the Chip Erase bit in the AS Key Status register (UPDI_ASI_KEY_STATUS.CHIPERASE) to see that the KEY is successfully activated.
3. Write the Reset signature into the ASI_RESET_REQ register. This will issue a System Reset.
4. Write 0x00 to the ASI Reset Request register (UPDI_ASI_RESET_REQ.RSTREQ) to clear the System Reset.
5. Read the Lock Status bit in the ASI System Status register (UPDI_ASI_SYS_STATUS.LOCKSTATUS).
6. Chip Erase is done when UPDI_ASI_SYS_STATUS.LOCKSTATUS == 0. If UPDI_ASI_SYS_STATUS.LOCKSTATUS == 1, go to point 5 again.

After a successful Chip Erase, the Lockbits will be cleared, and the UPDI will have full access to the system. Until Lockbits are cleared, the UPDI cannot access the system bus, and only CS-space operations can be performed.

Caution: During Chip Erase, the BOD is forced ON (BOD_CTRLA.ACTIVE=0x1), and uses the BOD Level from the BOD Configuration fuse (BOD_CTRLB.LVL=FUSE_BODCFG.LVL). If the supply voltage V_{DD} is below that threshold level, the device is unserviceable until VDD is increased adequately.

33.5.6.2. NVM Programming

If the device is unlocked, it is possible to write directly to the NVM Controller by the UPDI - this will lead to unpredictable code execution if the CPU is active during the NVM programming. To avoid this, the following NVM Programming sequence should be executed.

1. Follow the Chiperase procedure as described in Chip Erase. If the part is already unlocked, this point can be skipped.
2. Enter the NVMPROG KEY by using the `KEY` instruction. See Table 33-6 for the NVMPROG signature.
3. **Optional:** Read the KEY_STATUS.NVMPROG field to see that the KEY has been activated.
4. Write the Reset signature into the ASI_RESET_REQ register. This will issue a System Reset.
5. Write 0x00 to the Reset signature in ASI_RESET_REQ register to clear the System Reset.
6. Read ASI_SYS_STATUS.NVMPROG.
7. NVM Programming can start when ASI_SYS_STATUS.NVMPROG == 1. If ASI_SYS_STATUS.NVMPROG == 0, go to point 6 again.
8. Write data to NVM through the UPDI.
9. Write the Reset signature into the ASI_RESET_REQ register. This will issue a System Reset.
10. Write 0x00 to the Reset signature in ASI_RESET_REQ register to clear the System Reset.
11. Programming is complete.

33.5.6.3. User Row Programming

The User Row Programming feature allows the user to program new values to the User Row (USERROW) on a locked device, even with access to the system bus blocked for the UPDI. To program with this functionality enabled, the following sequence should be followed.

- 1. Enter the USERROW-Write KEY located in Table 33-6 by using the `KEY` instruction. See Table 33-6 for the UROWWRITE signature.
- 2. **Optional:** Read the KEY_STATUS.UROWWRITE field to see that the KEY has been activated.
- 3. Write the Reset signature into the ASI_RESET_REQ register. This will issue a System Reset.
- 4. Write 0x00 to the Reset signature in ASI_RESET_REQ register to clear the System Reset.
- 5. Read ASI_SYS_STATUS.UROWPROG bit
- 6. User Row Programming can start when ASI_SYS_STATUS.UROWPROG == 1. If ASI_SYS_STATUS.UROWPROG == 0, go to point 5 again.
- 7. The writable area has a size of one EEPROM page, byte, and it is only possible to write User Row data to the first 32 byte addresses of the RAM. Addressing outside this memory range will result in a non executed write. The data will map 1:1 with the User Row space, when the data is copied into the User Row upon completion of the Programming sequence.
- 8. When all User Row data has been written to the RAM, write the ASI_SYS_CTRLA.UROWWRITEFINAL bit.

- 9. Read ASI_SYS_STATUS.UROWPROG bit
- 10. The User Row Programming is completed when ASI_SYS_STATUS.UROWPROG == 1. If ASI_SYS_STATUS.UROWPROG == 0, go to point 9 again.
- 11. Write the Reset signature into the ASI_RESET_REQ register. This will issue a System Reset.
- 12. Write 0x00 to the Reset signature in ASI_RESET_REQ register to clear the System Reset.
- 13: User Row Programming is complete.

Note: It is not possible to read back data from the SRAM in this mode. Only writes to the first 32 bytes of the SRAM is allowed.

33.5.6.4. Activate OCD Features

To open up for debug features like single stepping and breakpoints, the OCD KEY must be correctly decoded. The following steps must be followed to activate OCD features.

- Enter the OCD KEY by using the *KEY* instruction. See Table 33-6 for the OCD signature.
- **Optional** Read out the OCD bit in the ASI KEY Status register (UPDI_ASI_KEY_STATUS.OCD) to see that the OCD KEY is successfully activated.

After the OCD KEY is successfully decoded, the UPDI have full access to OCD features.
Note: The device must be unlocked by a successful Chiperase before any debug operations can be done, even with the OCD KEY decoded.

33.5.6.5. Enabling of 2-Wire Mode

In applications where it is desirable to have separate RX and TX Pins, the UPDI 2-Wire option can be used. This is still half-duplex operation, with exactly the same communication protocol as in 1-Wire, and the same speed requirements. The following steps must be followed to enable 2-Wire mode.

- Enter the 2-Wire Mode KEY by using the *KEY* instruction. See Table 33-6 for the 2-Wire Mode signature.

After the last byte of the KEY is transmitted, 2-Wire Mode is enabled, and ready to receive the next instruction.

33.5.7. Events

The UPDI is connected to the Event System (EVSYS) as described in ASYNCCH0, ASYNCCH1, ASYNCCH2, ASYNCCH3.

The UPDI can generate the following output events:

- SYNCH Character Positive Edge Event

This Event is set on the UPDI clock for each detected positive edge in the SYNCH character, and it is not possible to disable this event from the UPDI. The recommended application for this Event is system clock frequency measurement through the UPDI. Section System Clock Measurement with UPDI provides the details on how to setup the system for this operation.

Related Links

EVSYS - Event System on page 109

33.5.8. Sleep Mode Operation

The UPDI physical layer runs independently of all sleep modes, and the UPDI is always accessible for a connected debugger independent of the device sleep mode. If the system enters a sleep mode that turns off the CPU clock, the UPDI will not be able to access the system bus and read memories and peripherals. Note that the UPDI physical layer clock is unaffected by the sleep mode settings, as long as the UPDI is enabled. By reading the UPDI_ASI_SYS_STATUS.INSLEEP bit it is possible to monitor if the system domain is in sleep mode. UPDI_ASI_SYS_STATUS.INSLEEP is set if the system is in IDLE sleep mode or deeper.

It is possible to prevent the system clock from stopping when going into sleep mode, by setting UPDI_ASI_SYS_CTRL.CLKREQ bit. If this bit is set the system sleep mode state is emulated, and it is possible for the UPDI to access they system bus and read out peripheral registers even in the deepest sleep modes.

Note: UPDI_ASI_SYS_CTRL.CLKREQ is set by default, which means that default operation is keeping the system clock on during sleep modes.

33.6. Register Summary

Offset	Name	Bit Pos.								
0x00	STATUSA	7:0	UPDIREV[3:0]							
0x01	STATUSB	7:0						PESIG[2:0]		
0x02	CTRLA	7:0	IBDLY	TBEN	PARD	DTD	RSD	GTVAL[2:0]		
0x03	CTRLB	7:0				NACKDIS	CCDETDIS	UPDIDIS	PDRM[1:0]	
0x04	ASI_OCD_CTRLA	7:0	OCD_SOR_DIS						OCD_RUN	OCD_STOP
0x05	ASI_OCD_STATUS	7:0				OCDMV				OCD_STOPPED
0x06	Reserved									
0x07	ASI_KEY_STATUS	7:0			UROWWRITE	NVMPROG	CHIPERASE		OCD	
0x08	ASI_RESET_REQ	7:0	RSTREQ[7:0]							
0x09	ASI_CTRLA	7:0							UPDICLKDIV[1:0]	
0x0A	ASI_SYS_CTRLA	7:0							UROWWRITE_FINAL	CLKREQ
0x0B	ASI_SYS_STATUS	7:0			RSTSYS	INSLEEP	NVMPROG	UROWPROG	BOOTDONE	LOCKSTATUS
0x0C	ASI_CRC_STATUS	7:0						CRC_STATUS[2:0]		
0x0D	ASI_OCD_MESSAGE	7:0	OCDM[7:0]							

33.7. Register Description

These registers are only readable through the PDI with special instructions, and are NOT readable through the CPU.

Registers at offset addresses 0x00-0x03 are the PDI Physical configuration registers

Registers at offset addresses 0x04-0xD are the ASI (PDI chip) level registers.

33.7.1. Status A

Name: STATUSA
Offset: 0x00
Reset: 0x10
Property: -

Bit	7	6	5	4	3	2	1	0
	UPDIREV[3:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	1	0	0	0	0

Bits 7:4 – UPDIREV[3:0]: UPDI Revision
These bits are read-only and contain the revision of the current UPDI implementation.

33.7.2. Status B

Name: STATUSB
Offset: 0x01
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
						PESIG[2:0]		
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 2:0 – PESIG[2:0]: PDI Error Signature
These bits describe the UPDI Error Signature created on any occurred Error. The Errors can occur during a transmission or if a system error occurs while the UPDI is in the IDLE state. These bits are set by the UPDI upon any occurred error condition, and cleared on read from the emulator.

Table 33-7. Valid Error Signatures

PESIG[2:0]	Error Type	Error Description
0x0	No error	No error detected (Default)
0x1	Parity error	Wrong sampling of the parity bit
0x2	Frame error	Wrong Sampling of frame stop bits
0x3	Access Layer Timeout Error	UPDI can get no data or response from the Access layer. Examples of error cases are system domain in sleep or system domain reset.
0x4	Clock recovery error	Wrong sampling of frame start bit
0x5	-	Reserved
0x6	Reserved	Reserved
0x7	Contention error	Signalize Driving Contention on the PDI_DATA line

33.7.3. Control A

Name: CTRLA
Offset: 0x02
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	IBDLY	TBEN	PARD	DTD	RSD	GTVAL[2:0]		
Access	R	R	R	R	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 7 – IBDLY: Inter-Byte Delay Enable
Writing a '1' to this bit enables a fixed inter-byte delay between each data byte transmitted from the UPDI when doing multi-byte LD(S). The fixed length is two IDLE characters. Note that before the first transmitted byte will use the regular GT delay used for direction change.

Bit 6 – TBEN: Transmit BREAK Enable
When this bit is written to '1', the PDI can send a BREAK character to the Emulator when the CPU reaches a stop condition (SW Break, HW Break). This bit only has effect if running in Power Debug or Live Debug Run on Disable modes, set by the PDRM bits in PDICRH.

Bit 5 – PARD: Parity Disable
Writing this bit to '1' will disable parity detection in the UPDI by just ignoring the Parity bit. This feature is recommended only during testing.

Bit 4 – DTD: Disable Timeout Detection
Setting this bit disables the timeout detection on the PHY layer, which requests a response from the ACC layer within a specified time (65536 UPDI clock cycles).

Bit 3 – RSD: Response Signature Disable
Writing a '1' to this bit will disable any response signatures generated by the UPDI. This is to reduce the protocol overhead to a minimum when writing large blocks of data to the NVM space. Note that it is not recommended to disable this feature unless the UPDI clock and the system clock are the same, and only during device testing.

Bits 2:0 – GTVAL[2:0]: Guard Time Value
This bit field selects the Guard Time Value that will be used by the UPDI when the transmission mode switches from RX to TX. The Guard time is equal to the Baud Rate used in 1W or 2W modes

Table 33-8. UPDI Guard Time Settings

GTVAL[2:0]	UPDI Guard Time Cycles
0x0	128 (default value)
0x1	64
0x2	32
0x3	16
0x4	8

GTVAL[2:0]	UPDI Guard Time Cycles
0x5	4
0x6	2
0x7	GT off (no extra Idle bits inserted)

33.7.4. Control B

Name: CTRLB
Offset: 0x03
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
				NACKDIS	CCDETDIS	UPDIDIS	PDRM[1:0]	
Access	R	R	R	R	R	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 4 – NACKDIS: Disable NACK Response
Writing this bit to '1' disables the NACK signature sent by the UPDI if a System Reset is issued during an ongoing LD(S) and ST(S) operation.

Bit 3 – CCDETDIS: Collision and Contention Detection Disable
If this bit is written to '0', contention detection is enabled for 1W mode. This means that the PDI can detect a collision in an ongoing 1W transmission. Contention detection is permanently disabled for 2W mode, as it has a separate TX-pin.

Bit 2 – UPDIDIS: UPDI Disable
Writing a '1' to this bit disables the UPDI PHY interface. The clock request from the UPDI is lowered, and the UPDI is reset. All UPDI PHY configurations and KEYs will be reset when the UPDI is disabled.

Bits 1:0 – PDRM[1:0]: PDI Disable on Run Mode
These bits determine which mode the UPDI should enter when enabled the next time. Writing these bits are required for successfully entering PowerDebug and LiveDebug Modes. For regular disable of the UPDI, the PowerDebug option is recommended.

Table 33-9. PDRM Selections

PDRM[2:0]	PDI Disable on Run Modes
0x0	No-disable operation (default)
0x1	PowerDebug
0x2	LiveDebug
0x3	Reserved

33.7.5. ASI On-Chip Debug Control A

Name: ASI_OCD_CTRLA
Offset: 0x04
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	OCD_SOR_DIS						OCD_RUN	OCD_STOP
Access	R/W	R	R	R	R	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 7 – OCD_SOR_DIS: OCD Stop on Reset Disable
If this bit is written to '1', the “Stop on Reset” feature is disabled. “Stop on Reset” is used when the OCD is enabled and a System Reset occurs. The OCD will then halt the CPU on the first instruction (after running system boot). The cause of the OCD stopping can be read out through the OCD status register in the OCD domain.

Bit 1 – OCD_RUN: On-Chip Debug Run
When this bit is written to '1', an OCD Run request is sent to the CPU. By reading the ASI_OCD_STATUS register it is possible to know when the CPU enters the RUN state (OCD_STOPPED is low). This bit is also used during Single Step, when the OCD system is set up accordingly.

Bit 0 – OCD_STOP: On-Chip Debug Stop
When this bit is written to '1' an OCD Stop request is sent to the CPU. By reading the ASI_OCD_STATUS register it is possible to know when the CPU reached the STOP state.

Note: For this field to have any effect, Lockbits must be cleared and the system cannot be in active boot.

33.7.6. ASI On-Chip Debug Status

Name: ASI_OCD_STATUS
Offset: 0x05
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
				OCDMV				OCD_STOPPED
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit 4 – OCDMV: On-Chip Debug Message Valid
If this bit is written to '1', the message contained in the ASI_OCD_MESSAGE register is valid for the UPDI to read out. This bit is cleared when the UPDI reads the ASI_OCD_MESSAGAE register.

Bit 0 – OCD_STOPPED: On-Chip Debug Stopped
When this bit is written to '1', the CPU is in the stopped state. If this bit is cleared the CPU is cleared, the CPU is in the RUN state.

33.7.7. ASI Key Status

Name: ASI_KEY_STATUS
Offset: 0x07
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
			UROWWRITE	NVMPROG	CHIPERASE		OCD	
Access			R	R	R		R	
Reset			0	0	0		0	

Bit 5 – UROWWRITE: User Row Write Key Status
This bit is set to '1' if the UROWWRITE KEY is active. Otherwise this bit reads as zero.

Bit 4 – NVMPROG: NVM Programming
This bit is set to '1' if the NVMPROG KEY is active. This bit is automatically reset after the programming sequence is done. Otherwise this bit reads as zero.

Bit 3 – CHIPERASE: Chip Erase
This bit is set to '1' if the CHIPERASE KEY is active. This bit will automatically be reset when the chip erase sequence is completed. Otherwise this bit reads as zero.

Bit 1 – OCD: On-Chip Debug Key Status
This bit is set to '1' if the OCD KEY is successfully decoded and active. Otherwise this bit reads as zero.

33.7.8. ASI Reset Request

Name: ASI_RESET_REQ
Offset: 0x08
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	RSTREQ[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – RSTREQ[7:0]: Reset Request
A Reset is signalized to the System when writing the Reset signature 0x59h to this address.

Writing any other signature to this register will clear the Reset.

Note:

1. When reading this register, reading bit RSTREQ[0] will tell if the UPDI is holding an active Reset on the system. If this bit is '1', the UPDI has an active Reset request to the system. All other bits will read as '0'.
2. The UPDI will not be reset when issuing a System Reset from this register.

33.7.9. ASI Control A

Name: ASI_CTRLA
Offset: 0x09
Reset: 0x02
Property: -

Bit	7	6	5	4	3	2	1	0
							UPDICLKDIV[1:0]	
Access							R/W	R/W
Reset							1	1

Bits 1:0 – UPDICLKDIV[1:0]: UPDI Clock Divider Select
Writing these bits selects the prescaler setting for the UPDI internal oscillator (32MHz). Default setting at Reset is 4MHz. Any other prescaler settings is only recommended when the BOD is at the highest level. At any other level the 8MHz prescaler should be used.

Value	Description
0x0	No prescaling (32MHz)
0x1	Divide by 2 prescaling (16MHz)
0x2	Divide by 4 prescaling (8MHz)
0x3	Divide by 8 prescaling (4MHz) (Default Setting)

33.7.10. ASI System Control A

Name: ASI_SYS_CTRLA
Offset: 0x0A
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
							UROWWRITE_FINAL	CLKREQ
Access	R	R	R	R	R	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 1 – UROWWRITE_FINAL : User Row Programming Done
This bit should be set by the UPDI when the user row programming is done. The CPU will continue to progress in the BOOT code when it detects that this bit is written.

Note:

1. If this bit is written before the User Row code is written to RAM by the UPDI, the CPU will progress without the written data.
2. This bit is only writable if the Userrow-write KEY is successfully decoded.

Bit 0 – CLKREQ: Request System Clock
If this bit is written to '1', the ASI is requesting the system clock, independent of system sleep modes. This makes it possible for the UPDI to access the ACC layer, also if the system is in sleep mode.

Writing a zero to this bit will lower the clock request.

This bit will be reset when the UPDI is disabled.

Note: This bit is set by default when the UPDI is enabled in any mode (Fuse, 12V).

33.7.11. ASI System Status

Name: ASI_SYS_STATUS
Offset: 0x0B
Reset: 0x01
Property: -

Bit	7	6	5	4	3	2	1	0
			RSTSYS	INSLEEP	NVMPROG	UROWPROG	BOOTDONE	LOCKSTATUS
Access			R	R	R	R	R	R
Reset			0	0	0	0	0	1

Bit 5 – RSTSYS: System Reset Active
If this bit is set, there is an active Reset on the system domain. If this bit is cleared, the system is not in Reset.

This bit is cleared on read.

Note: A reset held from the ASI_RESET_REQ register will also affect this bit.

Bit 4 – INSLEEP: System Domain in Sleep
If this bit is set, the system domain is in IDLE or deeper sleep mode. If this bit is cleared, the system is not in sleep.

Bit 3 – NVMPROG: Start NVM Programming
If this bit is set, the CPU has signalized from the BOOT code that the NVM Programming can start from the UPDI.

When the UPDI is done, it must reset the system through the UPDI Reset Register.

Bit 2 – UROWPROG : Start User Row Programming
If this bit is set, the CPU has signalized from the BOOT code that User Row Programming can start from the UPDI.

When the UPDI is done, it must write the UROWWRITE_FINAL bit in ASI_SYS_CTRLA.

Bit 1 – BOOTDONE: Boot Sequence Done
This bit is set when the CPU is done with the BOOT sequence. The UPDI will not have regular access to the ACC layer until this bit is set.

Bit 0 – LOCKSTATUS: NVM Lock Status
If this bit is set, the device is locked. If a Chiperase is done, and the Lockbits are cleared, this bit will read as zero.

33.7.12. ASI CRC Status

Name: ASI_CRC_STATUS
Offset: 0x0C
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
						CRC_STATUS[2:0]		
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 2:0 – CRC_STATUS[2:0]: CRC Execution Status
These bits signalize the status of the CRC conversion. The bits are one-hot encoded.

Value	Description
0x0	Not enabled
0x1	CRC enabled, busy
0x2	CRC enabled, done with OK signature
0x3	CRC enabled, done with FAILED signature

33.7.13. ASI On-Chip Debug Message

Name: ASI_OCD_MESSAGE
Offset: 0x0D
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	OCDM[7:0]							
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – OCDM[7:0]: On-Chip Debug Message
This register is used for byte-wise communication between the UPDI and CPU. It is only readable by the UPDI. The UPDI can poll the OCDMV bit in the ASI_OCD_STATUS register to know when the CPU has written a new message.

34. Instruction Set Summary

Mnemonics	Operands	Description	Operation			Flags	#Clocks
Arithmetic and Logic Instructions							
ADD	Rd, Rr	Add without Carry	Rd	←	Rd + Rr	Z,C,N,V,S,H	1
ADC	Rd, Rr	Add with Carry	Rd	←	Rd + Rr + C	Z,C,N,V,S,H	1
ADIW	Rd, K	Add Immediate to Word	Rd	←	Rd + 1:Rd + K	Z,C,N,V,S	2
SUB	Rd, Rr	Subtract without Carry	Rd	←	Rd - Rr	Z,C,N,V,S,H	1
SUBI	Rd, K	Subtract Immediate	Rd	←	Rd - K	Z,C,N,V,S,H	1
SBC	Rd, Rr	Subtract with Carry	Rd	←	Rd - Rr - C	Z,C,N,V,S,H	1
SBCI	Rd, K	Subtract Immediate with Carry	Rd	←	Rd - K - C	Z,C,N,V,S,H	1
SBIW	Rd, K	Subtract Immediate from Word	Rd + 1:Rd	←	Rd + 1:Rd - K	Z,C,N,V,S	2
AND	Rd, Rr	Logical AND	Rd	←	Rd • Rr	Z,N,V,S	1
ANDI	Rd, K	Logical AND with Immediate	Rd	←	Rd • K	Z,N,V,S	1
OR	Rd, Rr	Logical OR	Rd	←	Rd v Rr	Z,N,V,S	1
ORI	Rd, K	Logical OR with Immediate	Rd	←	Rd v K	Z,N,V,S	1
EOR	Rd, Rr	Exclusive OR	Rd	←	Rd ⊕ Rr	Z,N,V,S	1
COM	Rd	One’s Complement	Rd	←	$FF - Rd	Z,C,N,V,S	1
NEG	Rd	Two’s Complement	Rd	←	$00 - Rd	Z,C,N,V,S,H	1
SBR	Rd,K	Set Bit(s) in Register	Rd	←	Rd v K	Z,N,V,S	1
CBR	Rd,K	Clear Bit(s) in Register	Rd	←	Rd • ($FFh - K)	Z,N,V,S	1
INC	Rd	Increment	Rd	←	Rd + 1	Z,N,V,S	1
DEC	Rd	Decrement	Rd	←	Rd - 1	Z,N,V,S	1
TST	Rd	Test for Zero or Minus	Rd	←	Rd • Rd	Z,N,V,S	1
CLR	Rd	Clear Register	Rd	←	Rd ⊕ Rd	Z,N,V,S	1
SER	Rd	Set Register	Rd	←	$FF	None	1
MUL	Rd,Rr	Multiply Unsigned	R1:R0	←	Rd x Rr (UU)	Z,C	2
MULS	Rd,Rr	Multiply Signed	R1:R0	←	Rd x Rr (SS)	Z,C	2
MULSU	Rd,Rr	Multiply Signed with Unsigned	R1:R0	←	Rd x Rr (SU)	Z,C	2
FMUL	Rd,Rr	Fractional Multiply Unsigned	R1:R0	←	Rd x Rr<<1 (UU)	Z,C	2

Mnemonics	Operands	Description	Operation			Flags	#Clocks
FMULS	Rd,Rr	Fractional Multiply Signed	R1:R0	←	Rd x Rr<<1 (SS)	Z,C	2
FMULSU	Rd,Rr	Fractional Multiply Signed with Unsigned	R1:R0	←	Rd x Rr<<1 (SU)	Z,C	2
Branch instructions							
RJMP	k	Relative Jump	PC	←	PC + k + 1	None	2
IJMP		Indirect Jump to (Z)	PC	← ←	Z	None	2
EIJMP		Extended Indirect Jump to (Z)	PC	← ←	Z	None	2[3]
JMP	k	Jump	PC	←	k	None	3
RCALL	k	Relative Call Subroutine	PC	←	PC + k + 1	None	2
ICALL		Indirect Call to (Z)	PC	← ←	Z	None	2
EICALL		Extended Indirect Call to (Z)	PC	← ←	Z	None	2[3]
CALL	k	call Subroutine	PC	←	k	None	3
RET		Subroutine Return	PC	←	STACK	None	4
RETI		Interrupt Return	PC	←	STACK	I	4
CPSE	Rd,Rr	Compare, Skip if Equal	if (Rd = Rr) PC	←	PC + 2 or 3	None	1 / 2 / 3
CP	Rd,Rr	Compare	Rd - Rr			Z,C,N,V,S,H	1
CPC	Rd,Rr	Compare with Carry	Rd - Rr - C			Z,C,N,V,S,H	1
CPI	Rd,K	Compare with Immediate	Rd - K			Z,C,N,V,S,H	1
SBRC	Rr, b	Skip if Bit in Register Cleared	if (Rr(b) = 0) PC	←	PC + 2 or 3	None	1 / 2 / 3
SBRS	Rr, b	Skip if Bit in Register Set	if (Rr(b) = 1) PC	←	PC + 2 or 3	None	1 / 2 / 3
SBIC	A, b	Skip if Bit in I/O Register Cleared	if (I/O(A,b) = 0) PC	←	PC + 2 or 3	None	1 / 2 / 3
SBIS	A, b	Skip if Bit in I/O Register Set	If (I/O(A,b) =1) PC	←	PC + 2 or 3	None	1 / 2 / 3
BRBS	s, k	Branch if Status Flag Set	if (SREG(s) = 1) then PC	←	PC + k + 1	None	1 / 2
BRBC	s, k	Branch if Status Flag Cleared	if (SREG(s) = 0) then PC	←	PC + k + 1	None	1 / 2

Mnemonics	Operands	Description	Operation			Flags	#Clocks
BREQ	k	Branch if Equal	if (Z = 1) then PC	←	PC + k + 1	None	1 / 2
BRNE	k	Branch if Not Equal	if (Z = 0) then PC	←	PC + k + 1	None	1 / 2
BRCS	k	Branch if Carry Set	if (C = 1) then PC	←	PC + k + 1	None	1 / 2
BRCC	k	Branch if Carry Cleared	if (C = 0) then PC	←	PC + k + 1	None	1 / 2
BRSH	k	Branch if Same or Higher	if (C = 0) then PC	←	PC + k + 1	None	1 / 2
BRLO	k	Branch if Lower	if (C = 1) then PC	←	PC + k + 1	None	1 / 2
BRMI	k	Branch if Minus	if (N = 1) then PC	←	PC + k + 1	None	1 / 2
BRPL	k	Branch if Plus	if (N = 0) then PC	←	PC + k + 1	None	1 / 2
BRGE	k	Branch if Greater or Equal, Signed	if (N ⊕ V= 0) then PC	←	PC + k + 1	None	1 / 2
BRLT	k	Branch if Less Than, Signed	if (N ⊕ V= 1) then PC	←	PC + k + 1	None	1 / 2
BRHS	k	Branch if Half Carry Flag Set	if (H = 1) then PC	←	PC + k + 1	None	1 / 2
BRHC	k	Branch if Half Carry Flag Cleared	if (H = 0) then PC	←	PC + k + 1	None	1 / 2
BRTS	k	Branch if T Flag Set	if (T = 1) then PC	←	PC + k + 1	None	1 / 2
BRTC	k	Branch if T Flag Cleared	if (T = 0) then PC	←	PC + k + 1	None	1 / 2
BRVS	k	Branch if Overflow Flag is Set	if (V = 1) then PC	←	PC + k + 1	None	1 / 2
BRVC	k	Branch if Overflow Flag is Cleared	if (V = 0) then PC	←	PC + k + 1	None	1 / 2
BRIE	k	Branch if Interrupt Enabled	if (I = 1) then PC	←	PC + k + 1	None	1 / 2
BRID	k	Branch if Interrupt Disabled	if (I = 0) then PC	←	PC + k + 1	None	1 / 2
Data transfer instructions							
MOV	Rd, Rr	Copy Register	Rd	←	Rr	None	1
MOVW	Rd, Rr	Copy Register Pair	Rd+1:Rd	←	Rr+1:Rr	None	1
LDI	Rd, K	Load Immediate	Rd	←	K	None	1

Mnemonics	Operands	Description	Operation			Flags	#Clocks
LDS	Rd, k	Load Direct from data space	Rd	←	(k)	None	3[1]
LD	Rd, X	Load Indirect	Rd	←	(X)	None	2[1]
LD	Rd, X+	Load Indirect and Post-Increment	Rd X	← ←	(X) X + 1	None	2[1]
LD	Rd, -X	Load Indirect and Pre-Decrement	X ← X - 1, Rd ← (X)	← ←	X - 1 (X)	None	2[1]
LD	Rd, Y	Load Indirect	Rd ← (Y)	←	(Y)	None	2[1]
LD	Rd, Y+	Load Indirect and Post-Increment	Rd Y	← ←	(Y) Y + 1	None	2[1]
LD	Rd, -Y	Load Indirect and Pre-Decrement	Y Rd	← ←	Y - 1 (Y)	None	2[1]
LDD	Rd, Y+q	Load Indirect with Displacement	Rd	←	(Y + q)	None	2[1]
LD	Rd, Z	Load Indirect	Rd	←	(Z)	None	2[1]
LD	Rd, Z+	Load Indirect and Post-Increment	Rd Z	← ←	(Z), Z+1	None	2[1]
LD	Rd, -Z	Load Indirect and Pre-Decrement	Z Rd	← ←	Z - 1, (Z)	None	2[1]
LDD	Rd, Z+q	Load Indirect with Displacement	Rd	←	(Z + q)	None	2[1]
STS	k, Rr	Store Direct to Data Space	(k)	←	Rd	None	2[1][2]
ST	X, Rr	Store Indirect	(X)	←	Rr	None	1[1][2]
ST	X+, Rr	Store Indirect and Post-Increment	(X) X	← ←	Rr, X + 1	None	1[1][2]
ST	-X, Rr	Store Indirect and Pre-Decrement	X (X)	← ←	X - 1, Rr	None	1[1][2]
ST	Y, Rr	Store Indirect	(Y)	←	Rr	None	1[1][2]
ST	Y+, Rr	Store Indirect and Post-Increment	(Y) Y	← ←	Rr, Y + 1	None	1[1][2]
ST	-Y, Rr	Store Indirect and Pre-Decrement	Y (Y)	← ←	Y - 1, Rr	None	1[1][2]
STD	Y+q, Rr	Store Indirect with Displacement	(Y + q)	←	Rr	None	1[1][2]
ST	Z, Rr	Store Indirect	(Z)	←	Rr	None	1[1][2]

Mnemonics	Operands	Description	Operation			Flags	#Clocks
ST	Z+, Rr	Store Indirect and Post-Increment	(Z) Z	← ←	Rr Z + 1	None	1[1][2]
ST	-Z, Rr	Store Indirect and Pre-Decrement	Z	←	Z - 1	None	1[1][2]
STD	Z+q,Rr	Store Indirect with Displacement	(Z + q)	←	Rr	None	1[1][2]
LPM		Load Program Memory	R0	←	(Z)	None	2[1]
LPM	Rd, Z	Load Program Memory	Rd	←	(Z)	None	2[1]
LPM	Rd, Z+	Load Program Memory and Post-Increment	Rd Z	← ←	(Z), Z + 1	None	2[1]
ELPM		Extended Load Program Memory	R0	←	(Z)	None	2[1][3]
ELPM	Rd, Z	Extended Load Program Memory	Rd	←	(Z)	None	2[1][3]
ELPM	Rd, Z+	Extended Load Program Memory and Post-Increment	Rd Z	← ←	(Z), Z + 1	None	2[1][3]
IN	Rd, A	In From I/O Location	Rd	←	I/O(A)	None	1
OUT	A, Rr	Out To I/O Location	I/O(A)	←	Rr	None	1
PUSH	Rr	Push Register on Stack	STACK	←	Rr	None	1
POP	Rd	Pop Register from Stack	Rd	←	STACK	None	2
Bit and bit-test instructions							
LSL	Rd	Logical Shift Left	Rd(n+1) Rd(0) C	← ← ←	Rd(n), 0, Rd(7)	Z,C,N,V,H	1
LSR	Rd	Logical Shift Right	Rd(n) Rd(7) C	← ← ←	Rd(n+1), 0, Rd(0)	Z,C,N,V	1
ROL	Rd	Rotate Left Through Carry	Rd(0) Rd(n+1) C	← ← ←	C, Rd(n), Rd(7)	Z,C,N,V,H	1
ROR	Rd	Rotate Right Through Carry	Rd(7) Rd(n) C	← ← ←	C, Rd(n+1), Rd(0)	Z,C,N,V	1
ASR	Rd	Arithmetic Shift Right	Rd(n)	←	Rd(n+1), n=0..6	Z,C,N,V	1

Mnemonics	Operands	Description	Operation			Flags	#Clocks
SWAP	Rd	Swap Nibbles	Rd(3..0)	↔	Rd(7..4)	None	1
BSET	s	Flag Set	SREG(s)	←	1	SREG(s)	1
BCLR	s	Flag Clear	SREG(s)	←	0	SREG(s)	1
SBI	A, b	Set Bit in I/O Register	I/O(A, b)	←	1	None	1
CBI	A, b	Clear Bit in I/O Register	I/O(A, b)	←	0	None	1
BST	Rr, b	Bit Store from Register to T	T	←	Rr(b)	T	1
BLD	Rd, b	Bit load from T to Register	Rd(b)	←	T	None	1
SEC		Set Carry	C	←	1	C	1
CLC		Clear Carry	C	←	0	C	1
SEN		Set Negative Flag	N	←	1	N	1
CLN		Clear Negative Flag	N	←	0	N	1
SEZ		Set Zero Flag	Z	←	1	Z	1
CLZ		Clear Zero Flag	Z	←	0	Z	1
SEI		Global Interrupt Enable	I	←	1	I	1
CLI		Global Interrupt Disable	I	←	0	I	1
SES		Set Signed Test Flag	S	←	1	S	1
CLS		Clear Signed Test Flag	S	←	0	S	1
SEV		Set Two's Complement Overflow	V	←	1	V	1
CLV		Clear Two's Complement Overflow	V	←	0	V	1
SET		Set T in SREG	T	←	1	T	1
CLT		Clear T in SREG	T	←	0	T	1
SEH		Set Half Carry Flag in SREG	H	←	1	H	1
CLH		Clear Half Carry Flag in SREG	H	←	0	H	1
MCU control instructions							
BREAK		Break	(See specific descr. for BREAK)			None	1
NOP		No Operation				None	1
SLEEP		Sleep	(see specific descr. for Sleep)			None	1
WDR		Watchdog Reset	(see specific descr. for WDR)			None	1

1. Cycle times for data memory accesses assume internal RAM access, and are not valid for accesses through the NVM controller. A minimum of one extra cycle must be added when accessing memory through the NVM controller (such as Flash and EEPROM), but depending on simultaneous accesses by other masters or the NVM controller state, there may be more than one extra cycle.

2. One extra cycle must be added when accessing lower (64 bytes of) I/O space.

3. Instruction behaviour identical to instruction variant without E prefix.

35. Electrical Characteristics

35.1. Disclaimer

All typical values are measured at T = 25°C unless otherwise specified. All minimum and maximum values are valid across operating temperature and voltage unless otherwise specified.

35.2. Absolute Maximum Ratings

Stresses beyond those listed in this section may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Table 35-1. Absolute Maximum Ratings

Symbol	Description	Conditions	Min.	Max.	Units
V_{DD}	Power Supply Voltage		-0.5	6	V
I_{VDD}	Current into a V_{DD} pin	T=-40..+85°C	-	200	mA
		T=-+85..+125°C	-	100	mA
I_{GND}	Current out of a GND pin	T=-40..+85°C	-	200	mA
		T=-+85..+125°C	-	100	mA
V_{rst}	RESET pin voltage with respect to GND		-0.5	13	V
V_{PIN}	Pin voltage with respect to GND		-0.5	V_{DD}+0.5V	V
I_{PIN}	I/O pin sink/source current		-40	40	mA
I_c	I/O pin injection current except RESET pin		-20	20	mA
$T_{storage}$	Storage temperature		-65	150	°C

Caution: This device is sensitive to electrostatic discharges (ESD). Improper handling may lead to permanent performance degradation or malfunctioning.
Handle the device following best practice ESD protection rules: Be aware that the human body can accumulate charges large enough to impair functionality or destroy the device.

35.3. General Operating Ratings

The device must operate within the ratings listed in this section in order for all other electrical characteristics and typical characteristics of the device to be valid.

Table 35-2. General Operating Conditions

Symbol	Description	Condition	Min.	Max.	Units
V_{DD}	Operating Supply Voltage		1.8(2)	5.5	V
V_{RAM}	Power Supply Voltage to guarantee SRAM retention		0.9	5.5	V
T	Operating Temperature range	Standard temperature range(1)	-40	105	°C
		Extended temperature range(1)	-40	125	

Note:

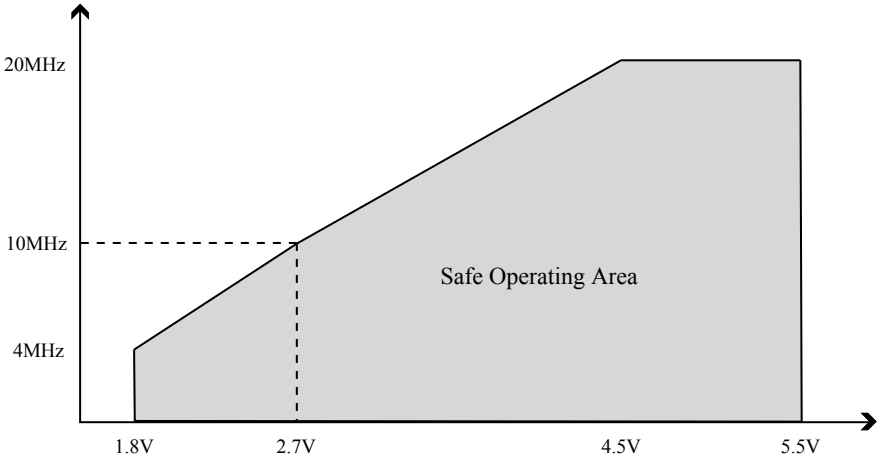
1. Please refer to the device ordering code for the device temperature range.
2. Operation guaranteed down to 1.8V or VBOD with BODLEVEL=1.8V, whichever is lower.

Symbol	Description	Condition	min	max	Unit
CLK_{SYS}	Operating system clock frequency	V_{DD}=1.8..5.5V T=-40..105°C(1)	0	5	MHz
		V_{DD}=2.7..5.5V T=-40..105°C(2)	0	10	
		V_{DD}=4.5..5.5V T=-40..105°C(3)	0	20	
		V_{DD}=2.7..5.5V T=-40..125°C(2)	0	8	
		V_{DD}=4.5..5.5V T=-40..125C(3)	0	16	

Note:

1. Operation guaranteed down to BOD triggering level V_{BOD} with BOD_CTRLB.LVL=1.8V.
2. Operation guaranteed down to BOD triggering level V_{BOD} with BOD_CTRLB.LVL=2.87V.
3. Operation guaranteed down to BOD triggering level V_{BOD} with BOD_CTRLB.LVL=4.3V.

The maximum CPU clock frequency depends on V_{DD}. As shown in the following figure, the Maximum Frequency vs. V_{DD} is linear between $1.8V<V_{DD}<2.7V$ and $2.7V<V_{DD}<4.5V$



35.4. Voltage Protection

Table 35-3. Power Supply Characteristics

Symbol	Description	Condition	Min	Typ	Max	Unit
SRON	Power-on Slope Rate		0.01	-	100	V/ms

Table 35-4. Power On Reset (POR) Characteristics

Symbol	Description	Condition	Min	Typ	Max	Unit
V_{POR}	POR threshold voltage on VDD falling	VDD falls/rises at 0.5V/ms or slower	0.80	-	1.60	V
	POR threshold voltage on VDD rising		1.40	-	1.70	

Table 35-5. Brownout Detection (BOD) Characteristics

Symbol	Description	Condition	Min	Typ	Max	Unit
V_{BOD}	BOD triggering level (falling/ rising)	BODLEVEL7 (4.30V)	4.1	-	4.5	V
		BODLEVEL2 (2.50V)	2.4	-	2.9	
		BODLEVEL0 (1.80V)	1.7	-	2.0	
V_{INT}	Interrupt level 0	Percentage above to the selected BOD level	-	5	-	%
	Interrupt level 1		-	15	-	
	Interrupt level 2		-	25	-	
V_{HYS}	Hysteresis	BODLEVEL7 (4.3V)	-	TBD	-	mV
		BODLEVEL2 (2.5V)	-	TBD	-	
		BODLEVEL0 (1.8V)	-	TBD	-	

Symbol	Description	Condition	Min	Typ	Max	Unit
T_{BOD}	Detection time(1)	Continuous	-	7	-	µs
		Sampled, 1kHz	-	1	-	ms
		Sampled, 125 Hz	-	8	-	
T_{start}	Startup time	Time from enable to ready	-	40	-	µs

Note: 1. T_{fall} 1V/µS, 100mV below BOD level.

36. Typical Characteristics

36.1. Power Consumption

ACTIVE Supply Current

Figure 36-1. ACTIVE SUPPLY CURRENT vs. FREQUENCY

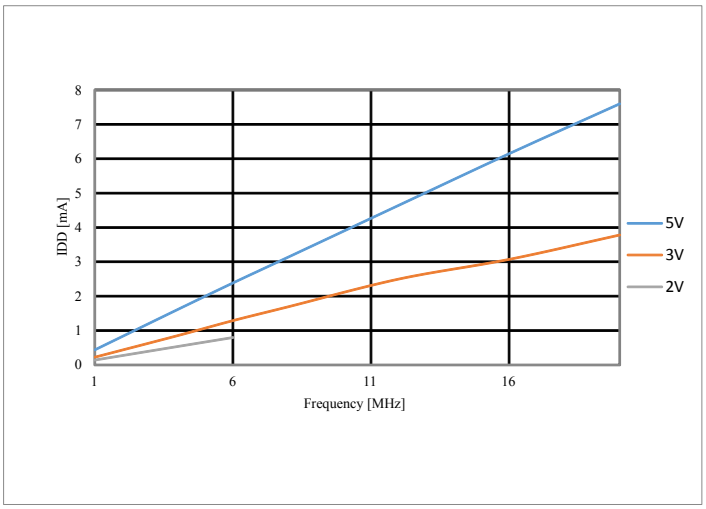


Figure 36-2. ACTIVE SUPPLY CURRENT vs. VDD INTERNAL RC OSCILLATOR, 20MHz

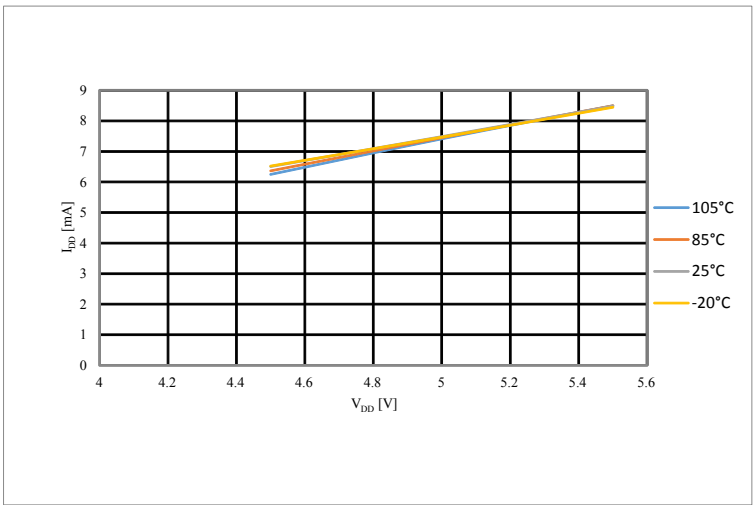


Figure 36-3. ACTIVE SUPPLY CURRENT vs. VDD INTERNAL RC OSCILLATOR, 10MHz

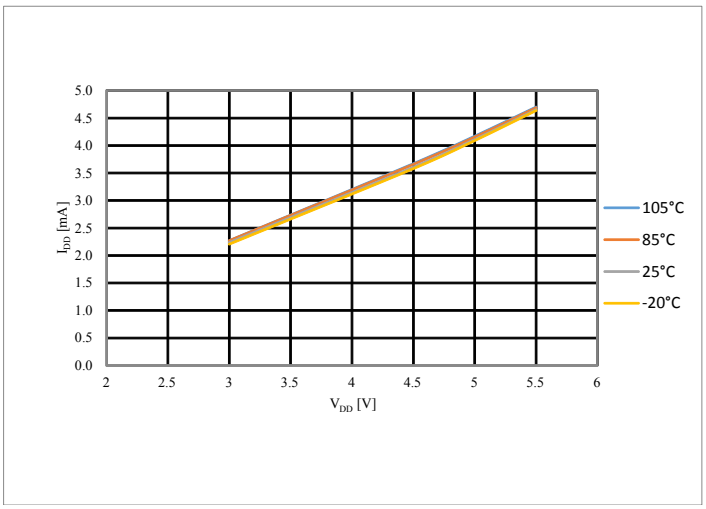


Figure 36-4. ACTIVE SUPPLY CURRENT vs. VDD INTERNAL RC OSCILLATOR, 5MHz

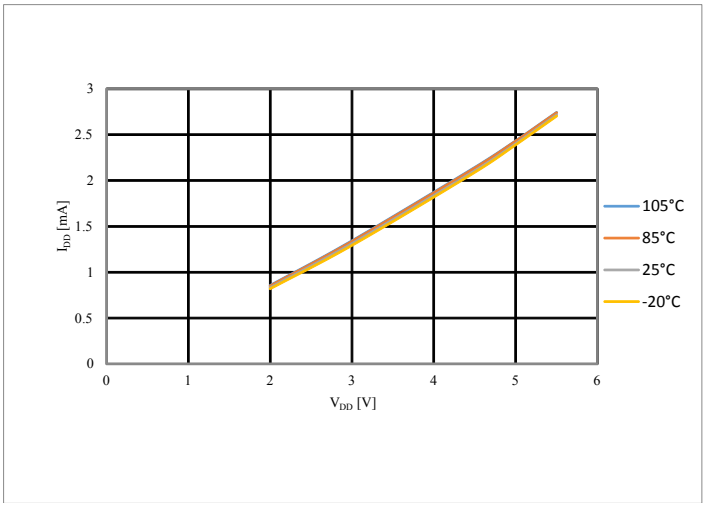
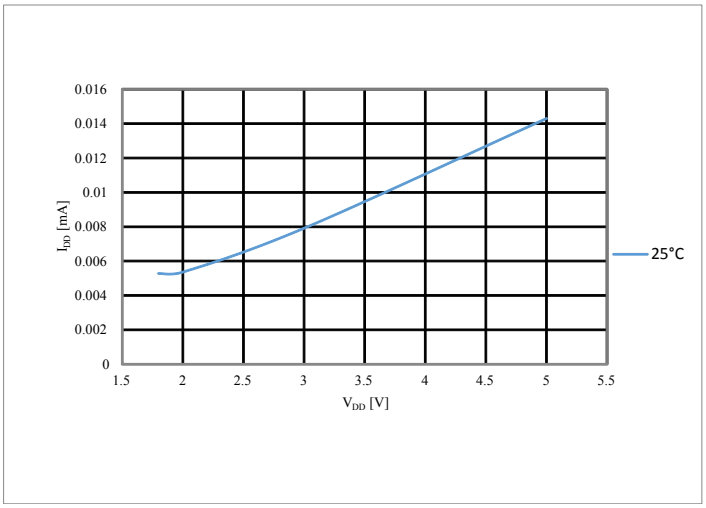


Figure 36-5. ACTIVE SUPPLY CURRENT vs. VDD INTERNAL RC OSCILLATOR, 32kHz



IDLE Supply Current

Figure 36-6. IDLE SUPPLY CURRENT vs. FREQUENCY

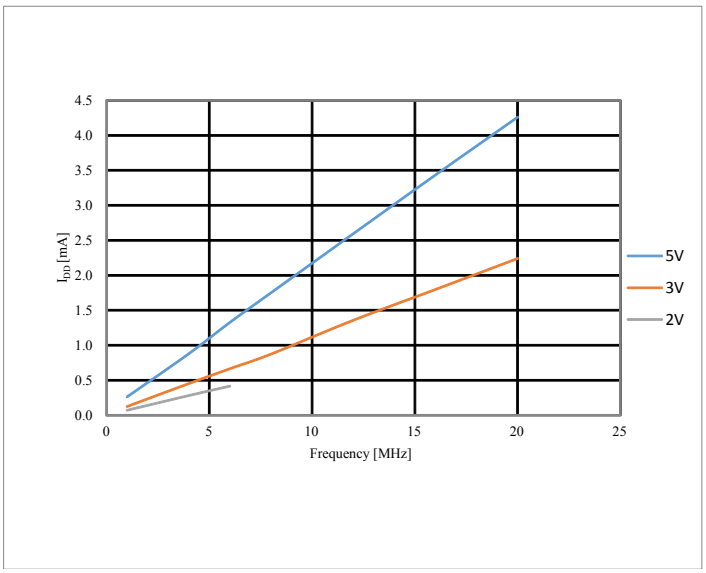


Figure 36-7. IDLE SUPPLY CURRENT vs. VDD INTERNAL RC OSCILLATOR, 20MHz

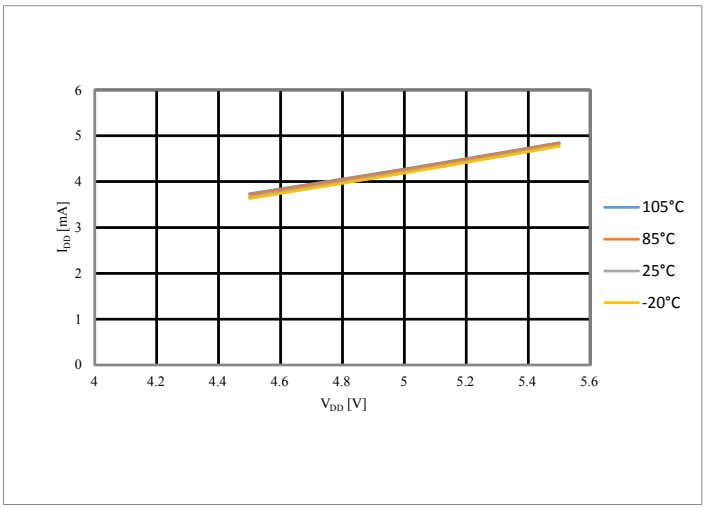
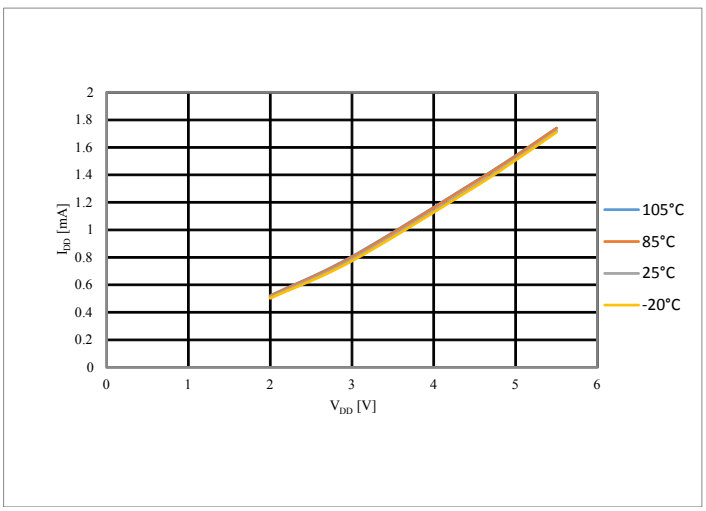
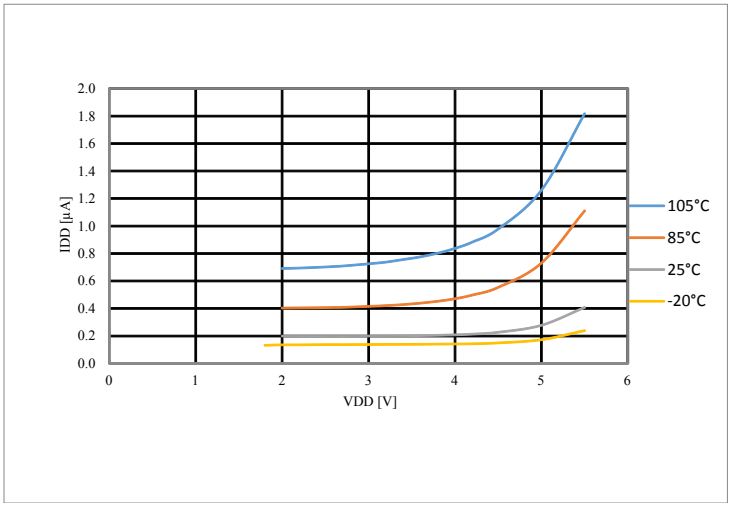


Figure 36-8. IDLE SUPPLY CURRENT vs. VDD INTERNAL RC OSCILLATOR, 5MHz



POWER-DOWN Supply Current

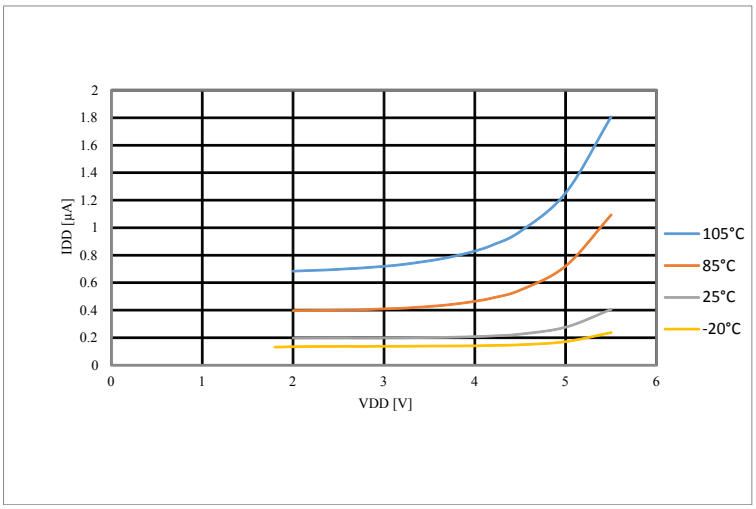
Figure 36-9. POWER-DOWN SUPPLY CURRENT vs. VDD



Note: Watchdog Timer (WDT) disabled.

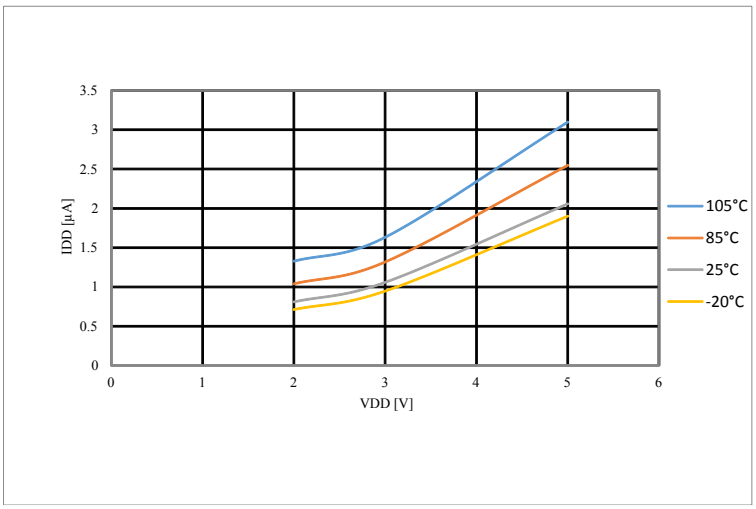
STANDBY Supply Current

Figure 36-10. STANDBY SUPPLY CURRENT vs. VDD



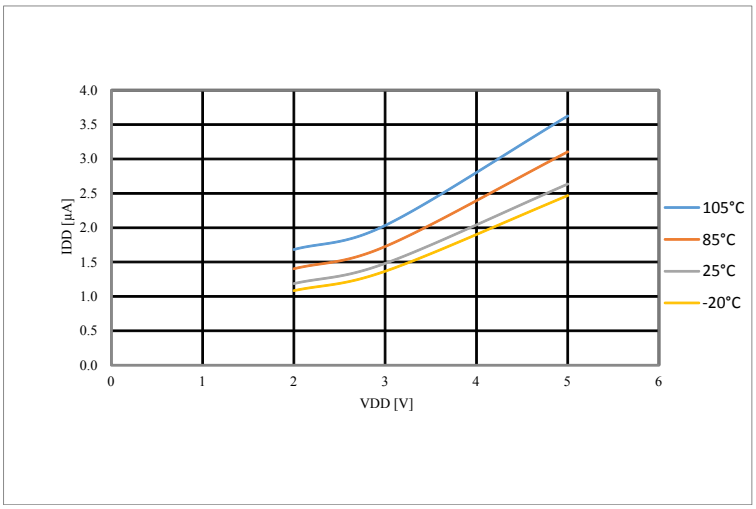
Note: All functions off.

Figure 36-11. STANDBY SUPPLY CURRENT vs. VDD



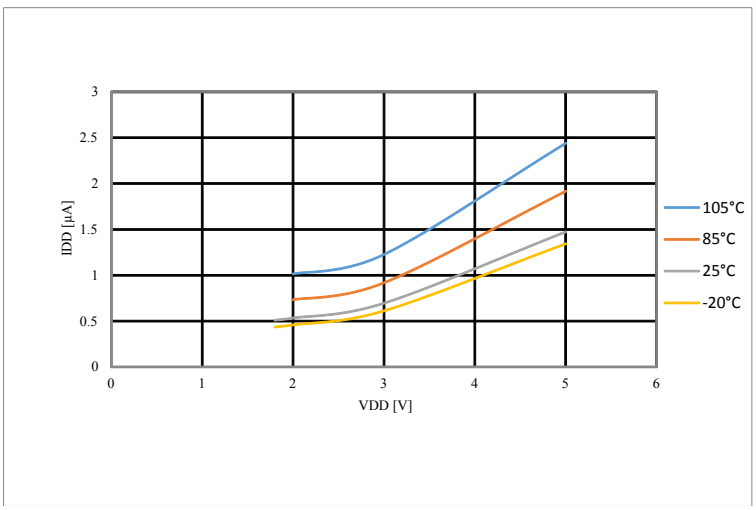
Note: BOD sampled at 125Hz.

Figure 36-12. STANDBY SUPPLY CURRENT vs. VDD



Note: BOD sampled at 1kHz

Figure 36-13. STANDBY SUPPLY CURRENT vs. VDD



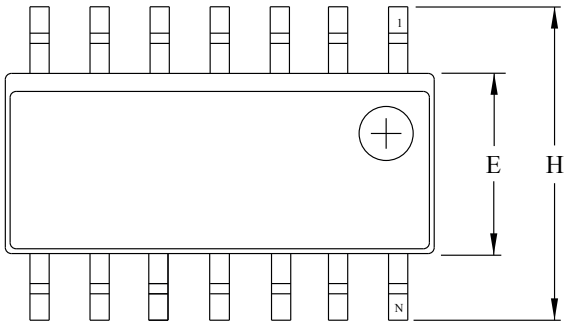
Note: RTC on OSC32K. BOD off.

37. Package Drawings

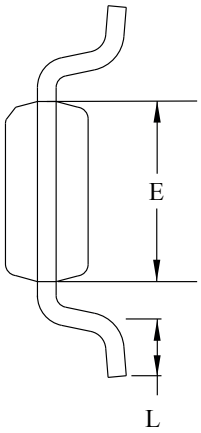
37.1. WARNING

Warning: Note that the package types are final, but the dimensions might slightly change.

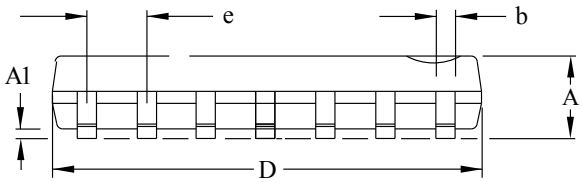
37.2. 14-pin SOIC150



Top View



End View



Side View

COMMON DIMENSIONS
(Unit of Measure = mm/inches)

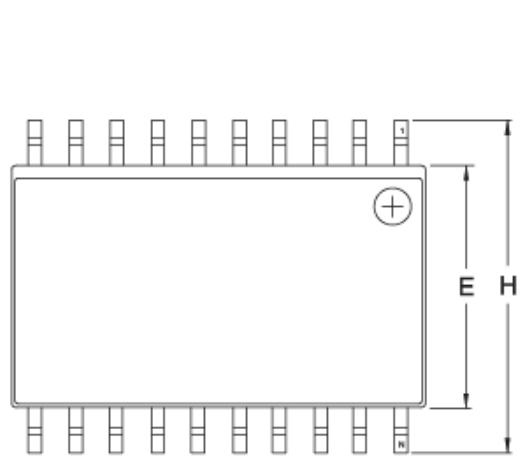
SYMBOL	MIN	NOM	MAX	NOTE
A	1.35/0.0532	–	1.75/0.0688	
A1	0.1/.0040	–	0.25/0.0098	
b	0.33/0.0130	–	0.5/0.0200	5
D	8.55/0.3367	–	8.74/0.34442	
E	3.8/0.1497	–	3.99/0.15743	
H	5.8/0.2284	–	6.19/0.2440	
L	0.41/0.0160	–	1.27/0.05004	
e	1.27/0.050 BSC			

Notes:
1. This drawing is for general information only; refer to JEDEC Drawing MS-012, Variation AB for additional information.
2. Dimension D does not include mold Flash, protrusions or gate burrs. Mold Flash, protrusion and gate burrs shall not exceed 0.15 mm (0.006") per side.
3. Dimension E does not include inter-lead Flash or protrusion. Inter-lead flash and protrusions shall not exceed 0.25 mm (0.010") per side.
4. L is the length of the terminal for soldering to a substrate.
5. The lead width B, as measured 0.36 mm (0.014") or greater above the seating plane, shall not exceed a maximum value of 0.61 mm (0.024") per side.

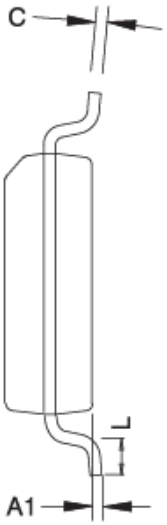
2/5/02

Package Drawing Contact: packagedrawings@atmel.com	TITLE **14S1**, 14-lead, 0.150" Wide Body, Plastic Gull Wing Small Outline Package (SOIC)	DRAWING NO. 14S1	REV. A

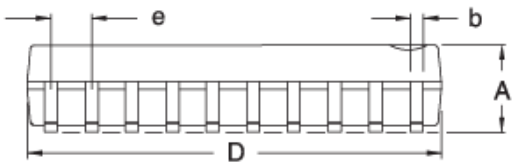
37.3. 20-pin SOIC300



Top View



End View



Side View

COMMON DIMENSIONS
(Unit of Measure – mm)

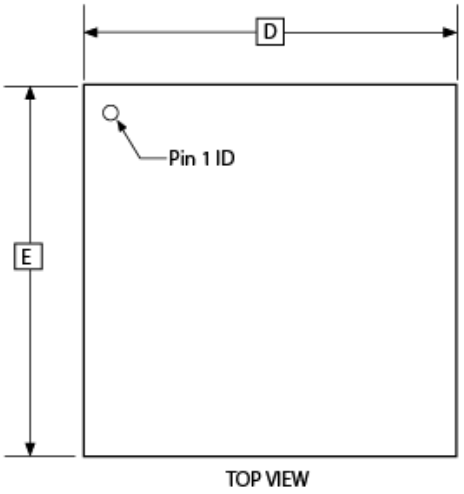
SYMBOL	MIN	NOM	MAX	NOTE
A	2.35		2.65	
A1	0.10		0.30	
b	0.33		0.51	4
C	0.23		0.32	
D	12.60		13.00	1
E	7.40		7.60	2
H	10.00		10.65	
L	0.40		1.27	3
e	1.27 BSC			

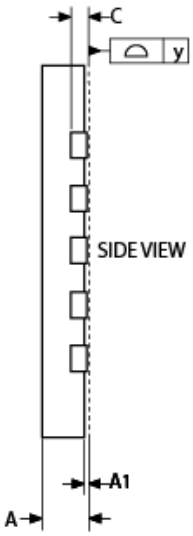
Notes.
1. This drawing is for general information only; refer to JEDEC Drawing MS-013, Variation AC for additional information.
2. Dimension 'D' does not include mold Flash, protrusions or gate burrs. Mold Flash, protrusions and gate burrs shall not exceed 0.15 mm (0.006') per side.
3. Dimension 'E' does not include inter-lead Flash or protrusion. Inter-lead Flash and protrusions shall not exceed 0.25 mm (0.010') per side.
4. 'L' is the length of the terminal for soldering to a substrate.
5. The lead width 'b', as measured 0.36 mm (0.014') or greater above the seating plane, shall not exceed a maximum value of 0.61 mm (0.024') per side.

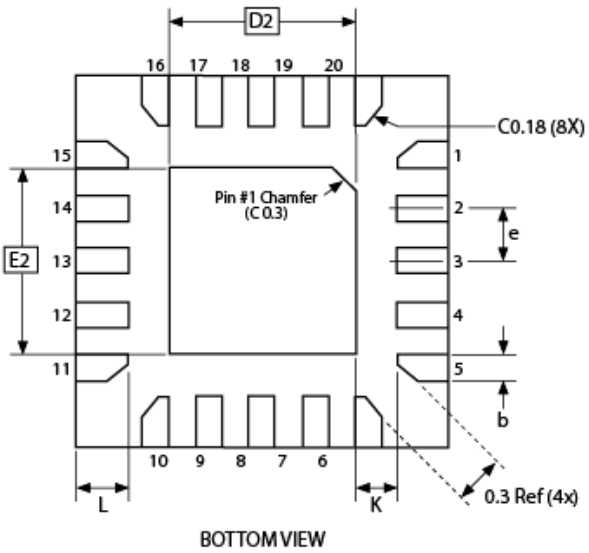
11/6/06

	TITLE	DRAWING NO.	REV.
Atmel 2325 Orchard Parkway San Jose, CA 95131	**20S2**, 20-lead, 0.300' Wide Body, Plastic Gull Wing Small Outline Package (SOIC)	20S2	B

37.4. 20-pin VQFN







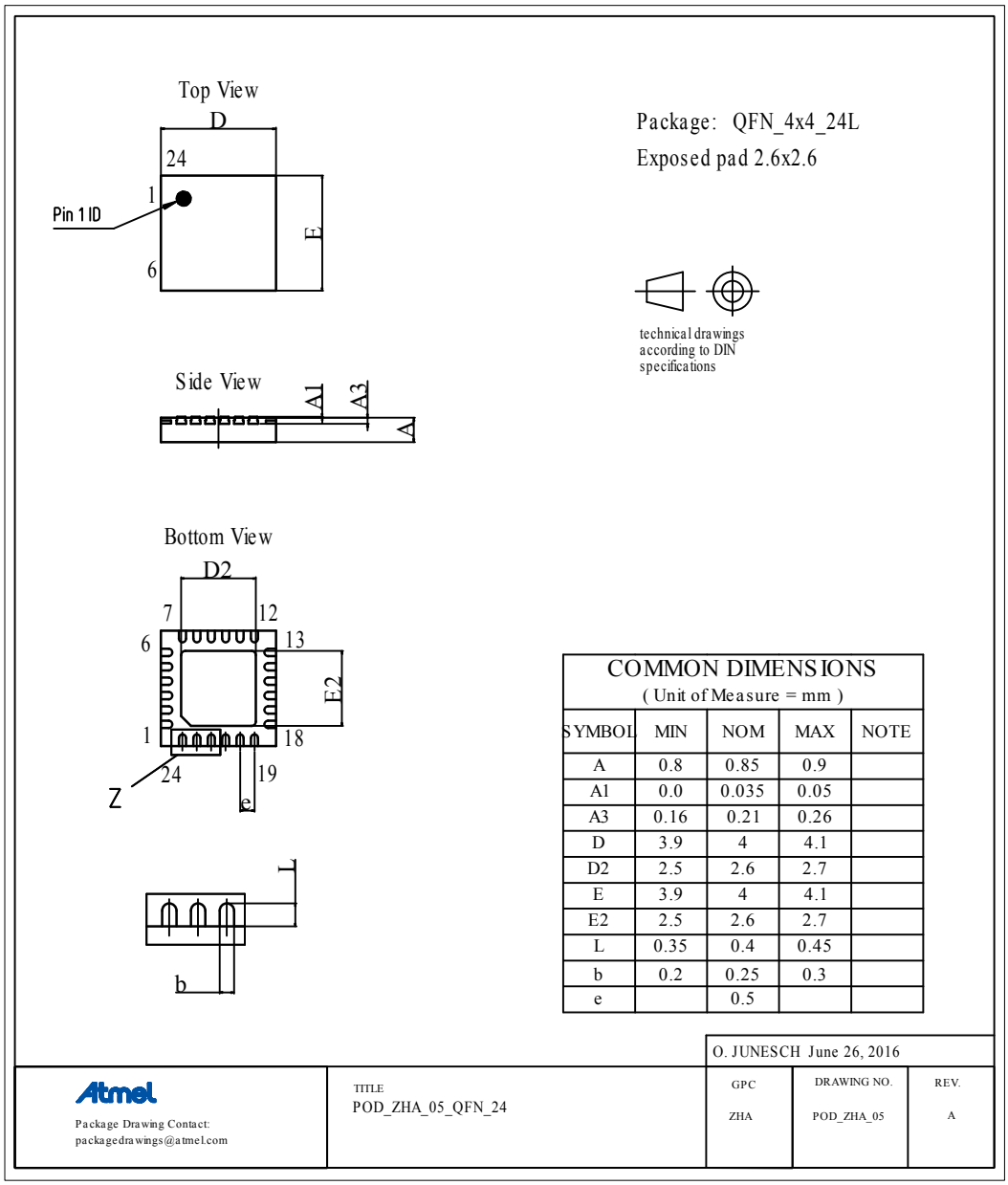
COMMON DIMENSIONS
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	0.75	0.80	0.85	
A1	0.00	0.02	0.05	
b	0.17	0.22	0.27	
C	0.152			
D	2.90	3.00	3.10	
D2	1.40	1.55	1.70	
E 2.90	3.00	3.10		
E2	1.40	1.55	1.70	
e	–	0.45	–	
L	0.35	0.40	0.45	
K 0.20	–	–		
y	0.00	–	0.08	

10/24/08

Atmel Package Drawing Contact: packagedrawings@atmel.com	TITLE	GPC	DRAWING NO.	REV.
	20M2, 20-pad, 3 x 3 x 0.85 mm Body, Lead Pitch 0.45 mm, 1.55 x 1.55 mm Exposed Pad, Thermally Enhanced Plastic Very Thin Quad Flat No Lead Package (VQFN)	ZFC	20M2	B

37.5. 24-pin QFN



COMMON DIMENSIONS (Unit of Measure = mm)				
SYMBOL	MIN	NOM	MAX	NOTE
A	0.8	0.85	0.9	
A1	0.0	0.035	0.05	
A3	0.16	0.21	0.26	
D	3.9	4	4.1	
D2	2.5	2.6	2.7	
E	3.9	4	4.1	
E2	2.5	2.6	2.7	
L	0.35	0.4	0.45	
b	0.2	0.25	0.3	
e		0.5		

O. JUNESCH June 26, 2016

Atmel
Package Drawing Contact:
packagedrawings@atmel.com

TITLE
POD_ZHA_05_QFN_24

GPC	DRAWING NO.	REV.
ZHA	POD_ZHA_05	A

38. Datasheet Revision History

Note: The datasheet revision is independent of the die revision and the device variant (last letter of the ordering number).

38.1. Rev.A - 06/2016

Initial release.

Atmel | Enabling Unlimited Possibilities®

Atmel Corporation 1600 Technology Drive, San Jose, CA 95110 USA **T:** (+1)(408) 441.0311 **F:** (+1)(408) 436.4200 | **www.atmel.com**