Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	AVR
Core Size	8-Bit
Speed	16MHz
Connectivity	I²C, IrDA, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	12
Program Memory Size	16KB (16K x 8)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 14x10b; D/A 3x8b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	14-SOIC (0.154", 3.90mm Width)
Supplier Device Package	14-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/attiny1614-ssfr

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 (1,2)	Other/ Special	ADC0	ADC1	PTC (3)	AC0	AC1	AC2	DAC0	USART0	SPI0	TWI0	TCA0	TCBn	TCD0	CCL
23	19	16	10	PA0	$\overline{\text{RESET}}$ $\overline{\text{T}}$ UPDI	AIN0													LUT0-IN0
24	20	17	11	PA1	BREAK	AIN1							TXD	MOSI	SDA				LUT0-IN1
1	1	18	12	PA2	EVOUT0	AIN2							RxD	MISO	SCL				LUT0-IN2
2	2	19	13	PA3	EXTCLK	AIN3							XCK	SCK		WO3	TCB1 WO		
3	3	20	14	GND															
4	4	1	1	VDD															
5	5	2	2	PA4		AIN4	AIN0	X0/Y0					XDIR	$\overline{\text{SS}}$		WO4		WOA	LUT0-OUT
6	6	3	3	PA5	VREFA	AIN5	AIN1	X1/Y1	OUT	AINN0						WO5	TCB0 WO	WOB	
7	7	4	4	PA6		AIN6	AIN2	X2/Y2	AINN0	AINP1	AINP0	OUT							
8	8	5	5	PA7		AIN7	AIN3	X3/Y3	AINP0	AINP0	AINN0								LUT1-OUT
9				PB7			AIN4			AINN1	AINP3								
10				PB6			AIN5		AINP3		AINN1								
11	9	6		PB5	CLKOUT	AIN8		X12/Y12	AINP1		AINP2					WO2			
12	10	7		PB4		AIN9		X13/Y13/	AINN1	AINP3						WO1			LUT0-OUT
13	11	8	6	PB3	TOSC1					OUT			RxD			WO0			
14	12	9	7	PB2	TOSC2, EVOUT1						OUT		TxD			WO2			
15	13	10	8	PB1		AIN10		X4/Y4	AINP2				XCK		SDA	WO1			
16	14	11	9	PB0		AIN11		X5/Y5		AINP2	AINP1		XDIR		SCL	WO0			
17	15	12		PC0			AIN6	X6/Y6						SCK			TCB0 WO	WOC	
18	16	13		PC1			AIN7	X7/Y7						MISO				WOD	LUT1-OUT
19	17	14		PC2	EVOUT2		AIN8	X9/Y8						MOSI					
20	18	15		PC3			AIN9	X9/Y9						$\overline{\text{SS}}$		WO3			LUT1-IN0
21				PC4	BREAK		AIN10	X10/Y10								WO4	TCB1 WO		LUT1-IN1
22				PC5			AIN11	X11/Y11								WO5			LUT1-IN2

13. CPUINT - CPU Interrupt Controller

Related Links

Interrupts

13.1 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

13.2 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 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 an interrupt request is acknowledged by the CPUINT, the program counter is set 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. CPUINT.STATUS is not saved automatically upon an interrupt request.

By default, all peripherals are priority level 0. It is possible to 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.

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

13.2.3.5 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

UPDI - Unified Program and Debug Interface

13.3 Functional Description

13.3.1 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 (IVSEL and CVT) in the Control A register (CPUINT.CTRLA).
   - Vector prioritizing by round-robin is enabled by writing a '1' to the Round-Robin Priority Enable bit (LVL0RR) in CPUINT.CTRLA.
   - Select the priority level 1 vector by writing its address to the Interrupt Vector (LVL1VEC) in the Level 1 Priority register (CPUINT.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 (I) in the CPU Status register (CPU.SREG).

13.3.2 Operation

13.3.2.1 Enabling, Disabling, and Resetting

Global enabling of interrupts is done by writing a '1' to the Global Interrupt Enable bit (I) in the CPU Status register (CPU.SREG). To disable interrupts globally, write a '0' to the I bit in CPU.SREG.

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. The respective INTFLAGS register descriptions provide information on how to clear specific flags.

13.3.2.2 Interrupt Vector Locations

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

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

16.4 Register Summary - PORT

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	INT[7:0]							
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]		

16.5 Register Description - Ports

16.5.1 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.

16.5.2 Data Direction Set

Offset: 0x01
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
			DAC2REFEN	ADC1REFEN	DAC1REFEN		ADC0REFEN	DAC0REFEN
Access			R/W	R/W	R/W		R/W	R/W
Reset			0	0	0		0	0

Bit 5 – DAC2REFEN: DAC2 and AC2 Reference Force Enable
Writing a '1' to this bit forces the voltage reference for the DAC2 and AC2 to be running, even if it not requested.

Writing a '0' to this bit allows to automatic enable/disable of the reference source when not requested.

Bit 4 – ADC1REFEN: ADC1 Reference Force Enable
Writing a '1' to this bit forces the voltage reference for the ADC1 to be running, even if it not requested.

Writing a '0' to this bit allows to automatic enable/disable of the reference source when not requested.

Bit 3 – DAC1REFEN: DAC1 and AC1 Reference Force Enable
Writing a '1' to this bit forces the voltage reference for the DAC1 and AC1 to be running, even if it not requested.

Writing a '0' to this bit allows to automatic enable/disable of the reference source when not requested.

Bit 1 – ADC0REFEN: ADC0 Reference Force Enable
Writing a '1' to this bit forces the voltage reference for the ADC0 to be running, even if it not requested.

Writing a '0' to this bit allows to automatic enable/disable of the reference source by the peripheral.

Bit 0 – DAC0REFEN: DAC0 and AC0 Reference Force Enable
Writing a '1' to this bit forces the voltage reference for the DAC0 and AC0 to be running, even if it not requested.

Writing a '0' to this bit allows to automatic enable/disable of the reference source by the peripheral.

18.5.3 Control C

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

Bit	7	6	5	4	3	2	1	0
		ADC1REFSEL[2:0]				DAC1REFSEL[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 – ADC1REFSEL[2:0]: ADC1 Reference Select
These bits select the reference voltage for the ADC1.

20.5.8 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.

20.5.9 Event Control

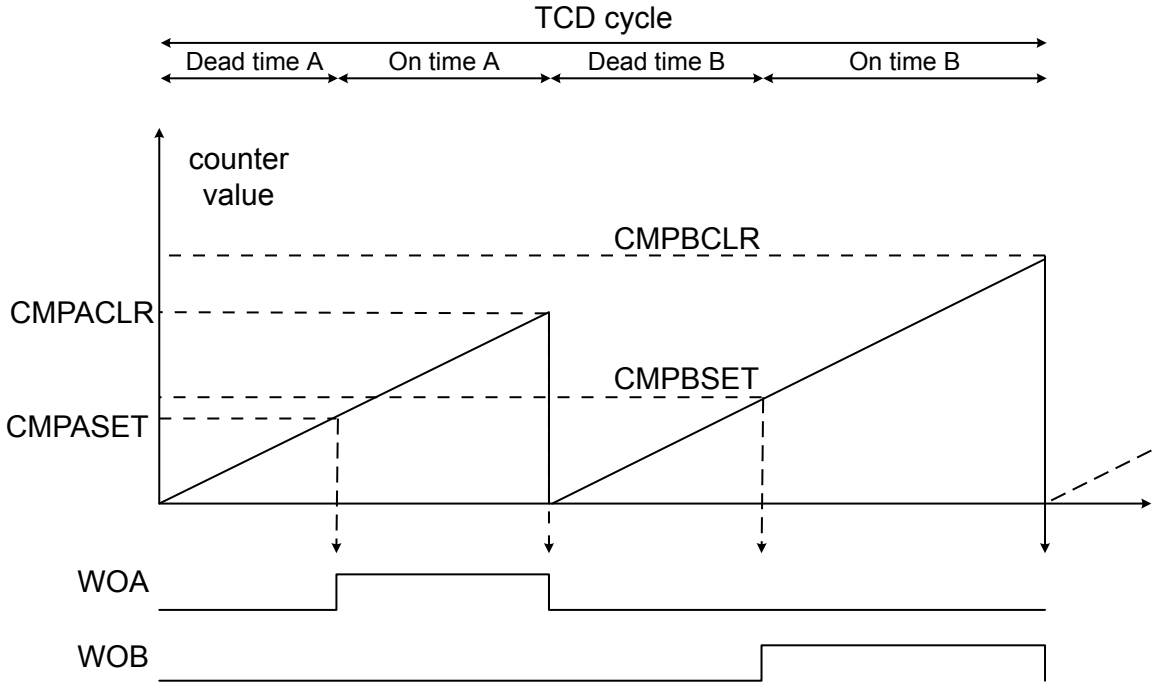
Name: EVCTRL
Offset: 0x09
Reset: 0x00
Property: -

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

Bits 2:1 – EVACT[1:0]: Event Action
These bits define on what type of event action the counter will increment or decrement.

Value	Name	Description
0x0	EVACT_POSEDGE	Count on positive edge event
0x1	EVACT_ANYEDGE	Count on any edge event
0x2	EVACT_HIGHLVL	Count on prescaled clock while event line is 1.
0x3	EVACT_UPDOWN	Count on prescaled clock. Event controls count direction. Up-count when event line is 0, down-count when event line is 1.

Figure 22-5. 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.

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 given by

$$T_{\mathrm{TCD\_cycle}} = \frac{(\mathrm{CMPASET} + 1 + \mathrm{CMPACLR} + 1 + \mathrm{CMPBSET} + 1 + \mathrm{CMPBCLR} + 1)}{f_{\mathrm{CLK\_TCD\_CNT}}}$$

23.4.2 Operation - PIT

23.4.2.1 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.

23.5 Events

The RTC, when enabled, will 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, when enabled, will 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.

The Event users are configured by the Event System (EVSYS).

Related Links

EVSYS - Event System

23.6 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 enable 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.

Related Links

24.5.3 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: R/W

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]: Transmit Data Register

24.5.4 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 USART.TXDATAL except if CHSIZE in USART.CTRLC is set to 9BIT Low byte first where USART.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 in USART.CTRLC.

24.5.5 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	R/W	R	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 to '1' 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 '1' 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).

This flag is automatically cleared when the transmit complete interrupt vector is executed. The flag can also be cleared by writing a '1' 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 set to '1' when the transmit buffer is empty, and is '0' 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 '0' when writing the STATUS register.

DREIF is cleared to '0' 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 '1' to its bit location. This flag is not used in master SPI mode operation.

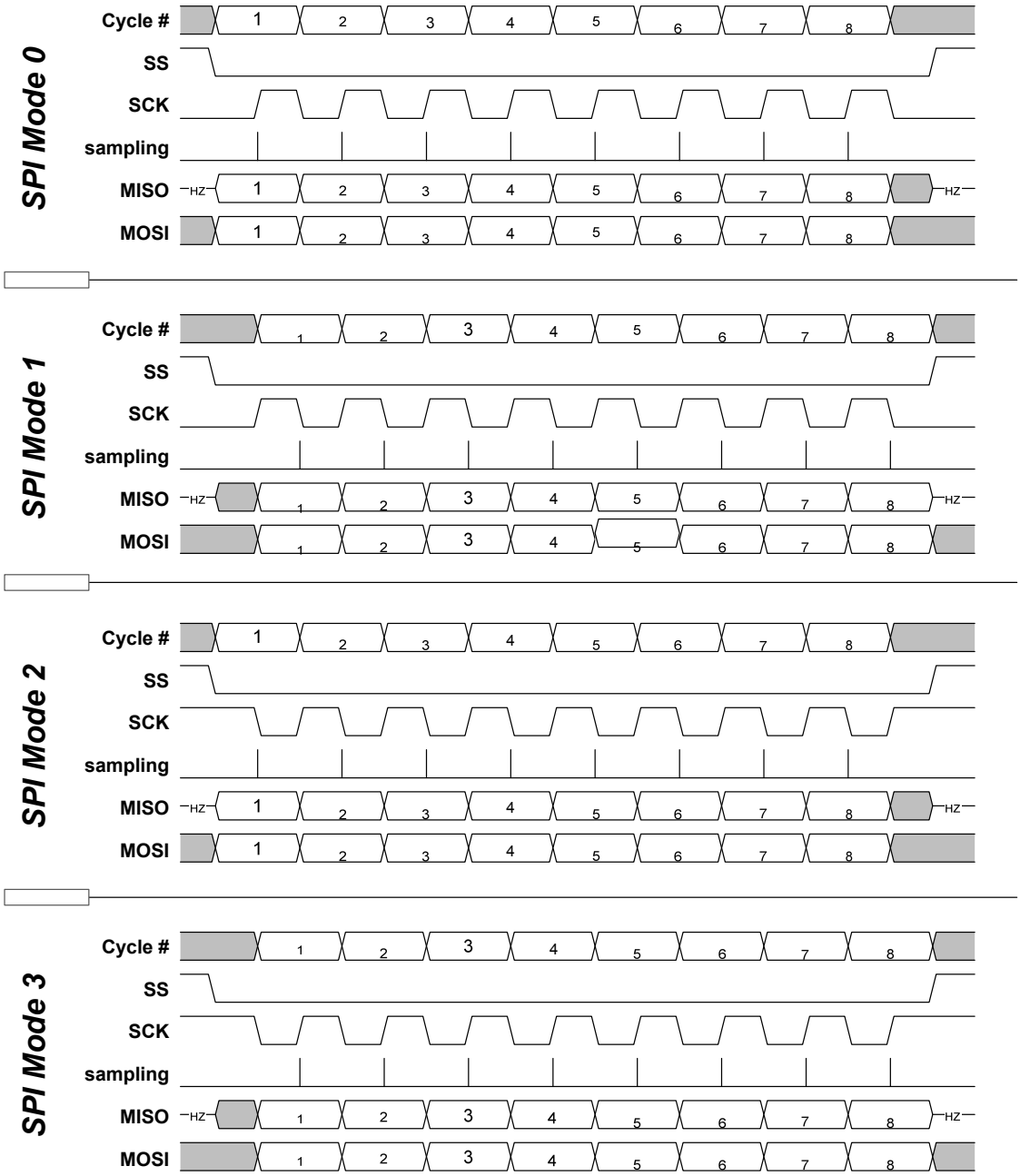
Bit 3 – ISFIF: Inconsistent Sync Field Interrupt Flag
This bit is set when the auto-baud is enabled and the sync field bit time are too fast or too slow to give a valid baud setting. It will also be set when USART is set to LINAUTO mode and the SYNC character differ from data value 0x55.

Writing a '1' to this bit will clear the flag and bring the USART back to idle state.

Bit 1 – BDF: Break Detected Flag
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 unused.

This bit is cleared by writing a '1' to it.

Figure 25-2. SPI Data Transfer Modes



25.3.3 Interrupts

Table 25-3. 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

26. TWI - Two Wire Interface

26.1 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

26.2 Overview

The Two-Wire Interface (TWI) peripheral 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 peripheral 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

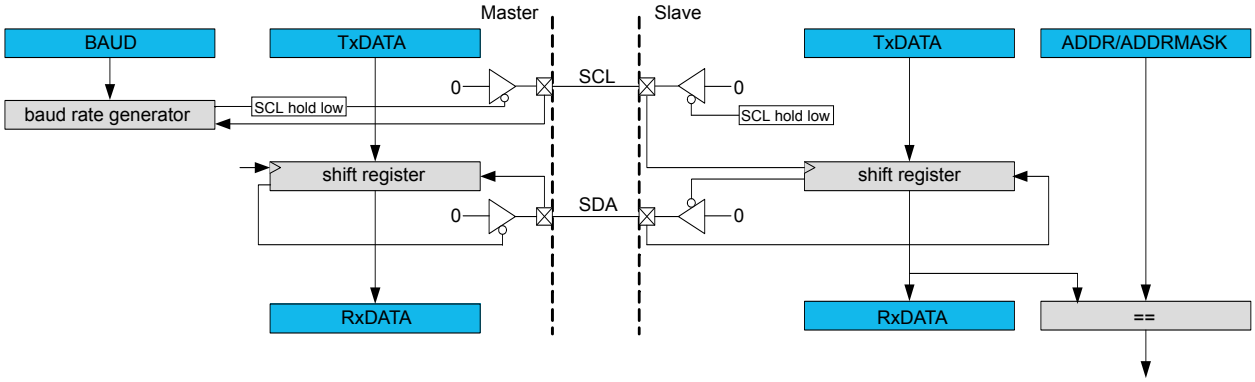
register or as a register for address range masking. The slave continues to operate in all sleep modes, including power-down mode. This enables the slave to wake up the device from all sleep modes on TWI address match. It is possible to disable the address matching to let this be handled in software instead.

The TWI peripheral 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.

This device provides one instance of the TWI peripheral, TWI0.

26.2.1 Block Diagram

Figure 26-1. TWI Block Diagram



26.2.2 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

26.2.3 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
Events	No	-
Debug	Yes	UPDI

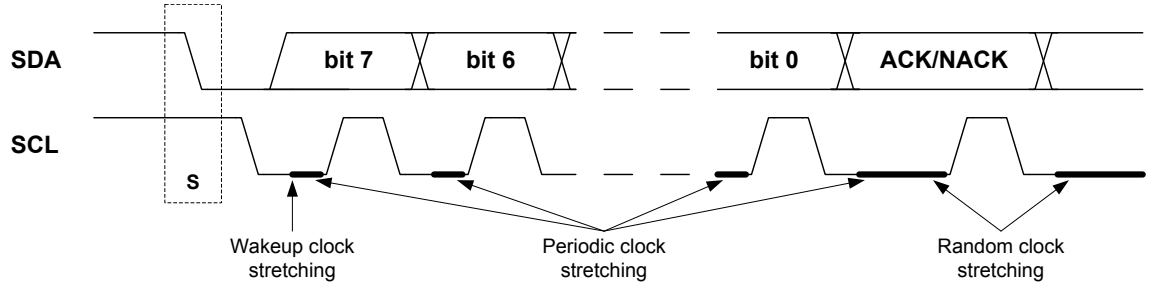
Related Links

Clocks

Debug Operation

I/O Lines and Connections

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.

26.3.2.7 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

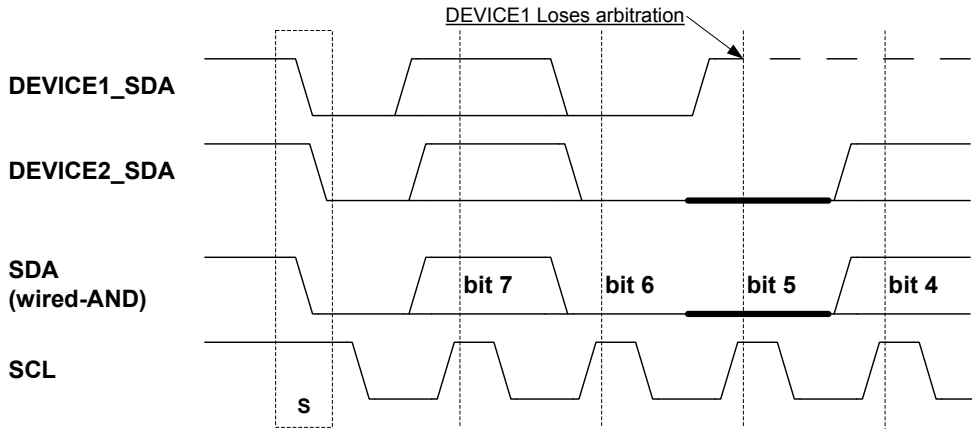


Figure 26-10 shows an example where two TWI masters are contending for bus ownership. Both devices are able to issue a START condition, but DEVICE1 loses arbitration when attempting to transmit a high level (bit 5) while DEVICE2 is transmitting a low level.

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.

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.

26.5.9 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

28.3.2 Operation

28.3.2.1 Enabling, Disabling and Resetting

The CCL is enabled by writing a '1' to the ENABLE bit in the Control register (CCL.CTRLA). The CCL is disabled by writing a '0' to that ENABLE bit.

Each LUT is enabled by writing a '1' to the LUT Enable bit (ENABLE) in the LUT n Control A register (CCL.LUTnCTRLA). Each LUT is disabled by writing a '0' to the ENABLE bit in CCL.LUTnCTRLA.

28.3.2.2 Lookup Table Logic

The lookup table in each LUT unit can generate a combinational logic output as a function of up to three inputs IN[2:0]. Unused inputs can be masked (tied low). The truth table for the combinational logic expression is defined by the bits in the CCL.TRUTHn registers. Each combination of the input bits (IN[2:0]) corresponds to one bit in the TRUTHn register, as shown in the table:

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]

28.3.2.3 Truth Table Inputs Selection

Input Overview

The inputs can be individually:

- Masked
- Driven by peripherals:
  - Analog comparator output (AC)
  - Timer/Counters waveform outputs (TC)
- Driven by internal events from Event System
- Driven by other CCL sub-modules

The Input Selection for each input y of LUT n is configured by writing the Input y Source Selection bit in the LUT n Control x=[B,C] registers

- INSEL0 in CCL.LUTnCTRLB
- INSEL1 in CCL.LUTnCTRLB
- INSEL2 in CCL.LUTnCTRLC.

Internal Feedback Inputs (FEEDBACK)

When selected (INSELy=FEEDBACK in CCL.LUTnCTRLx), the Sequential (SEQ) output is used as input for the corresponding LUT.

- Enable the AC by writing a '1' to the ENABLE bit in AC.CTRLA.

During the start-up time after enabling the AC, the output of the AC may be invalid.

The start-up time of the AC by itself is at most 2.5µs. If a internal referance is used, the referance start-up time is normally longer than the AC startup time. The VREF startup time is at most 60 µs.

29.3.2 Operation

29.3.2.1 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 (HYSMODE) in the Control A register (AC.CTRLA).

29.3.2.2 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 (MUXPOS and MUXNEG) in the MUX Control A register (AC.MUXTRLA).

Pin Inputs

The following Analog input pins on the port can be selected as input to the analog comparator

- AINN0
- AINN1
- AINP0
- AINP1
- AINP2
- AINP3

Internal Inputs

Two internal inputs are available for the analog comparator:

- Output from the DAC
- DAC and AC voltage reference

Related Links

VREF - Voltage Reference

29.3.2.3 Low Power Mode

For power-conscious applications, the AC provides a Low Power with reduced power consumption and increased propagation delay.

29.3.3 Events

The AC will generate the following Event automatically when the AC is enabled:

- Comparator output Event

Mnemonic	Operands	Description		Op		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

Table 34-2. Branch Instructions

Mnemonic	Operands	Description		Op		Flags	#Clocks
RJMP	k	Relative Jump	PC	←	PC + k + 1	None	2
IJMP		Indirect Jump to (Z)	PC(15:0) PC(21:16)	← ←	Z 0	None	2
JMP	k	Jump	PC	←	k	None	3
RCALL	k	Relative Call Subroutine	PC	←	PC + k + 1	None	2 / 3
ICALL		Indirect Call to (Z)	PC(15:0) PC(21:16)	← ←	Z 0	None	2 / 3
CALL	k	Call Subroutine	PC	←	k	None	3 / 4
RET		Subroutine Return	PC	←	STACK	None	4 / 5
RETI		Interrupt Return	PC	←	STACK	I	4 / 5
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
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

Figure 38-54. ATtiny1614/1616/1617 REFSEL = External reference, DNL vs. V_{REF} (V_{DD}=5.0V, f_{ADC}=115ksps)

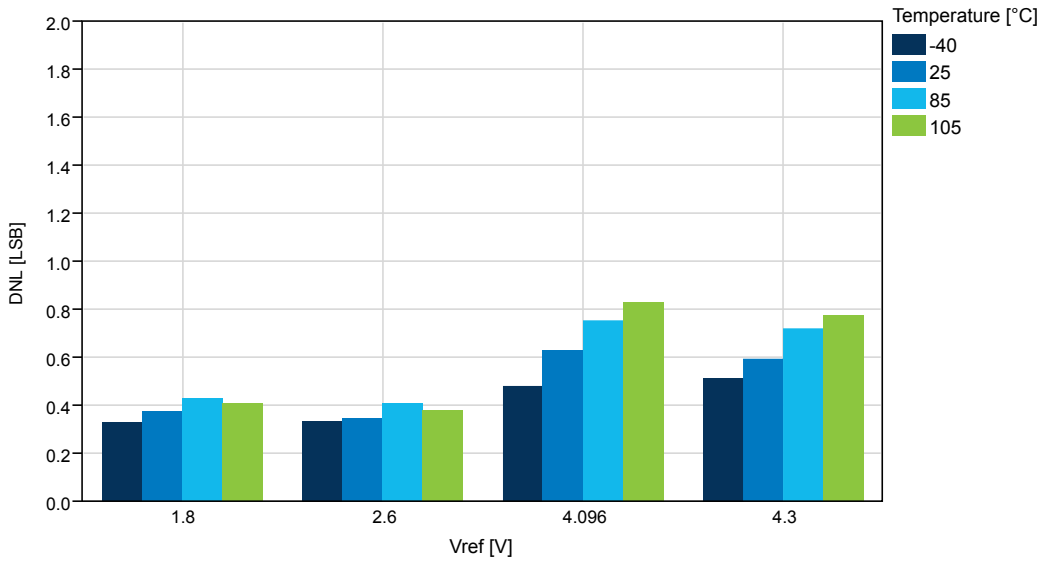


Figure 38-55. ATtiny1614/1616/1617 REFSEL = External reference, Gain vs. V_{DD} (f_{ADC}=115ksps, T=25°C)