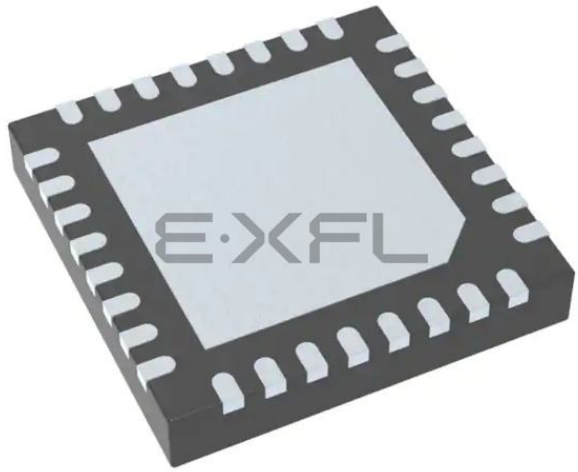


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

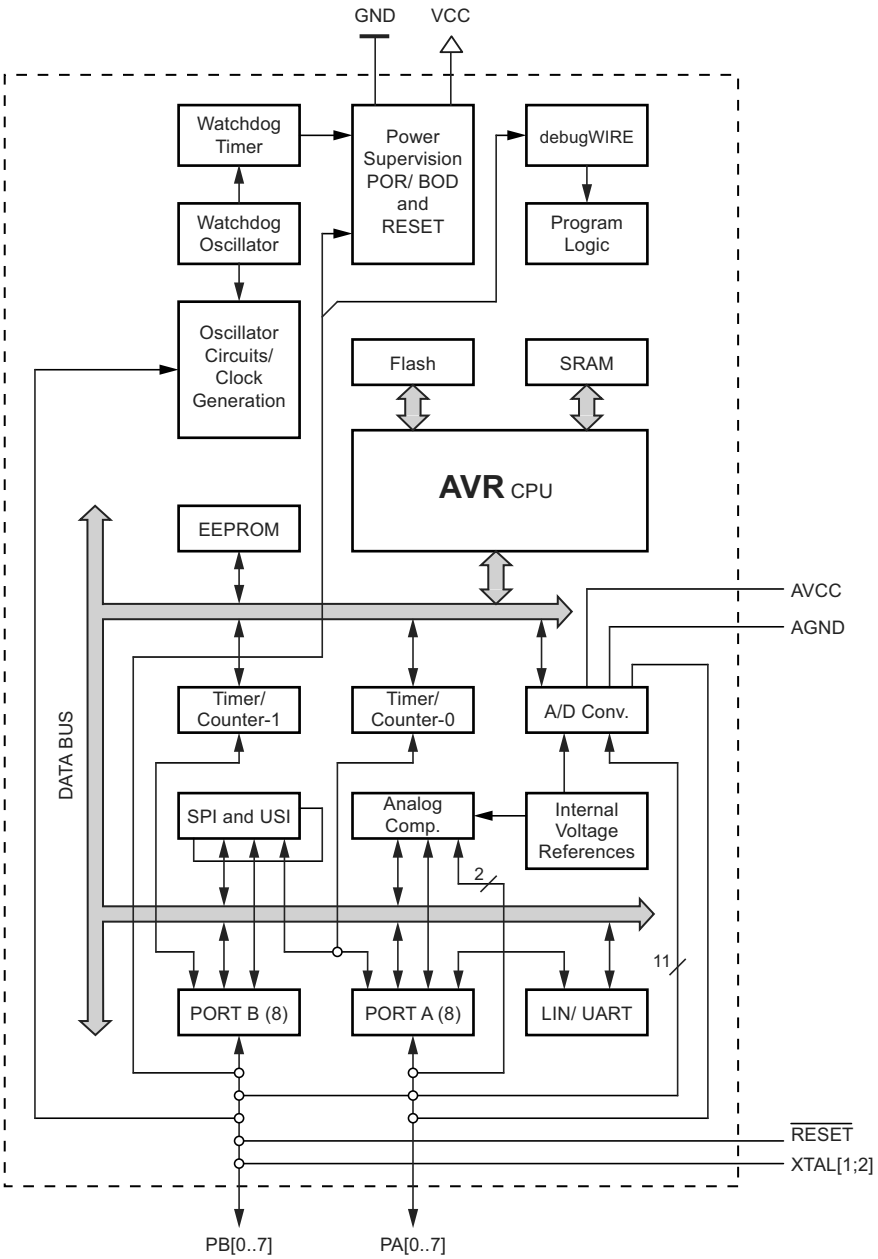
"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded"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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	AVR
Core Size	8-Bit
Speed	16MHz
Connectivity	I²C, LINbus, SPI, UART/USART, USI
Peripherals	Brown-out Detect/Reset, POR, PWM, Temp Sensor, WDT
Number of I/O	16
Program Memory Size	16KB (8K x 16)
Program Memory Type	FLASH
EEPROM Size	512 x 8
RAM Size	512 x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 11x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	32-VFQFN Exposed Pad
Supplier Device Package	32-QFN (5x5)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/attiny167-15mz

1.5 Block Diagram

Figure 1-1. Block Diagram



1.7 Pin Description

1.7.1 Vcc

Supply voltage.

1.7.2 GND

Ground.

1.7.3 AVcc

Analog supply voltage.

1.7.4 AGND

Analog ground.

1.7.5 Port A (PA7..PA0)

Port A is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port A output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port A pins that are externally pulled low will source current if the pull-up resistors are activated. The Port A pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port A also serves the functions of various special features of the Atmel® ATtiny87/167 as listed on Section 9.3.3 “Alternate Functions of Port A” on page 73.

1.7.6 Port B (PB7..PB0)

Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port B also serves the functions of various special features of the ATtiny87/167 as listed on Section 9.3.4 “Alternate Functions of Port B” on page 78.

1.8 Resources

A comprehensive set of development tools, application notes and datasheets are available for download on http://www.atmel.com/avr.

1.9 About Code Examples

This documentation contains simple code examples that briefly show how to use various parts of the device. These code examples assume that the part specific header file is included before compilation. Be aware that not all C compiler vendors include bit definitions in the header files and interrupt handling in C is compiler dependent. Please confirm with the C compiler documentation for more details.

3.4 I/O Memory

The I/O space definition of the Atmel® ATtiny87/167 is shown in Section "" on page 243.

All ATtiny87/167 I/Os and peripherals are placed in the I/O space. All I/O locations may 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. When using the I/O specific commands IN and OUT, the I/O addresses 0x00 - 0x3F must be used. When addressing I/O registers as data space using LD and ST instructions, 0x20 must be added to these addresses. The Atmel ATtiny87/167 is a complex microcontroller with more peripheral units than can be supported within the 64 location reserved in opcode for the IN and OUT instructions. For the extended I/O space from 0x60 - 0xFF in SRAM, 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 logical one to them. Note that, unlike most other AVRs, 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 to 0x1F only.

The I/O and peripherals control registers are explained in later sections.

3.4.1 General Purpose I/O Registers

The Atmel ATtiny87/167 contains three 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.

The general purpose I/O registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI, CBI, SBIS, and SBIC instructions.

3.5 Register Description

3.5.1 EEARH and EEARL – EEPROM Address Register

Bit	7	6	5	4	3	2	1	0	
	-	-	-	-	-	-	-	**EEAR8**	EEARH
	EEAR7	**EEAR6**	**EEAR5**	**EEAR4**	**EEAR3**	**EEAR2**	**EEAR1**	**EEAR0**	EEARL
Bit	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R/W	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	X	
Initial Value	X	X	X	X	X	X	X	X	

- **Bit 7:1 – Reserved Bits**

These bits are reserved for future use and will always read as 0 in ATtiny87/167.

- **Bits 8:0 – EEAR8:0: EEPROM Address**

The EEPROM address registers – EEARH and EEARL – specifies the high EEPROM address in the EEPROM space (see "E2 size" in Table 3-1 on page 16). The EEPROM data bytes are addressed linearly between 0 and "E2 size". The initial value of EEAR is undefined. A proper value must be written before the EEPROM may be accessed.

Note: For information only - ATtiny47: EEAR8 exists as register bit but it is not used for addressing.

The CKSEL0 fuse together with the SUT1..0 Fuses or CSEL0 together with CSUT1..0 field select the start-up times as shown in Table 4-7.

Table 4-7. Start-up Times for the Crystal Oscillator Clock Selection

CKSEL0[1] CSEL0[2]	SUT1..0[1] CSUT1..0[2]	Start-up Time from Power-down/save	Additional Delay from Reset (Vcc = 5.0V)	Recommended Usage
0	00	258CK[3]	14CK + 4.1ms	Ceramic resonator, fast rising power
0	01	258CK[3]	14CK + 65ms	Ceramic resonator, slowly rising power
0	10[5]	1K(1024)CK[4]	14CK	Ceramic resonator, BOD enabled
0	11	1K(1024)CK[4]	14CK + 4.1ms	Ceramic resonator, fast rising power
1	00	1K(1024)CK[4]	14CK + 65ms	Ceramic resonator, slowly rising power
1	01[5]	16K(16384)CK	14CK	Crystal Oscillator, BOD enabled
1	10	16K(16384)CK	14CK + 4.1ms	Crystal Oscillator, fast rising power
1	11	16K(16384)CK	14CK + 65ms	Crystal Oscillator, slowly rising power

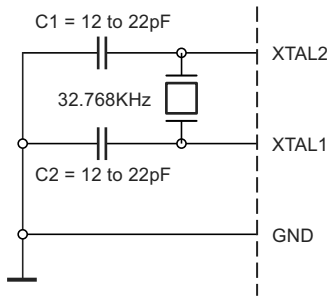
Notes:
1. Flash fuse bits.
2. CLKSELR register bits.
3. These options should only be used when not operating close to the maximum frequency of the device, and only if frequency stability at start-up is not important for the application. These options are not suitable for crystals.
4. These options are intended for use with ceramic resonators and will ensure frequency stability at start-up. They can also be used with crystals when not operating close to the maximum frequency of the device, and if frequency stability at start-up is not important for the application.
5. This setting is only available if RSTDISBL fuse is not set.

4.2.5 Low-frequency Crystal Oscillator

To use a 32.768kHz watch crystal as the clock source for the device, the low-frequency crystal oscillator must be selected by setting CKSEL fuses or CSEL field as shown in Table 4-1 on page 26. The crystal should be connected as shown in Figure 4-3. Refer to the 32.768 kHz crystal oscillator application note for details on oscillator operation and how to choose appropriate values for C1 and C2.

The 32.768kHz watch crystal oscillator can be used by the asynchronous timer if the (high-frequency) crystal oscillator is not running or if the external clock is not enabled (Section 4.3.3 “Enable/Disable Clock Source” on page 33). The asynchronous timer is then able to start itself this low-frequency crystal oscillator.

Figure 4-3. Low-frequency Crystal Oscillator Connections



12 to 22pF capacitors may be necessary if parasitic impedance (pads, wires and PCB) is very low.

When this oscillator is selected, start-up times are determined by the SUT fuses or by CSUT field as shown in Table 4-8.

Table 4-8. Start-up Times for the Low Frequency Crystal Oscillator Clock Selection

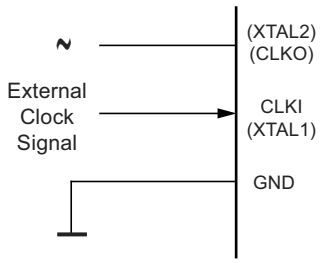
SUT1..0(1) CSUT1..0(2)	Start-up Time from Power-down/save	Additional Delay from Reset (Vcc = 5.0V)	Recommended usage
00	1K(1024)CK(3)	4.1ms	Fast rising power or BOD enabled
01	1K(1024)CK(3)	65ms	Slowly rising power
10	32K(32768)CK	65ms	Stable frequency at start-up
11	Reserved		

Notes: 1. Flash fuse bits.
2. CLKSELR register bits.
3. These options should only be used if frequency stability at start-up is not important for the application.

4.2.6 External Clock

To drive the device from this external clock source, CLKI should be driven as shown in Figure 4-4. To run the device on an external clock, the CKSEL Fuses or CSEL field must be programmed as shown in Table 4-1 on page 26.

Figure 4-4. External Clock Drive Configuration



4.4 System Clock Prescaler

4.4.1 Features

The Atmel® ATtiny87/167 system clock can be divided by setting the clock prescaler register – CLKPR. This feature can be used to decrease power consumption when the requirement for processing power is low. This can be used with all clock source options, and it will affect the clock frequency of the CPU and all synchronous peripherals. $clk_{I/O}$, clk_{ADC}, clk_{CPU}, and clk_{FLASH} are divided by a factor as shown in Table 4-10 on page 39.

4.4.2 Switching Time

When switching between prescaler settings, the system clock prescaler ensures that no glitches occur in the clock system and that no intermediate frequency is higher than neither the clock frequency corresponding to the previous setting, nor the clock frequency corresponding to the new setting.

The ripple counter that implements the prescaler runs at the frequency of the undivided clock, which may be faster than the CPU's clock frequency. Hence, it is not possible to determine the state of the prescaler – even if it were readable, and the exact time it takes to switch from one clock division to another cannot be exactly predicted.

From the time the CLKPS values are written, it takes between T1 + T2 and T1 + 2*T2 before the new clock frequency is active. In this interval, 2 active clock edges are produced. Here, T1 is the previous clock period, and T2 is the period corresponding to the new prescaler setting.

4.5 Register Description

4.5.1 OSCCAL – Oscillator Calibration Register

Bit	7	6	5	4	3	2	1	0	
	CAL7	CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0	OSCCAL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	Device Specific Calibration Value								

- **Bits 7:0 – CAL7:0: Oscillator Calibration Value**

The oscillator calibration register is used to trim the calibrated internal RC oscillator to remove process variations from the oscillator frequency. The factory-calibrated value is automatically written to this register during chip reset, giving an oscillator frequency of 8.0MHz at 25°C. The application software can write this register to change the oscillator frequency. The oscillator can be calibrated to any frequency in the range 7.3 - 8.1MHz within ±2% accuracy. Calibration outside that range is not guaranteed.

Note that this oscillator is used to time EEPROM and flash write accesses, and these write times will be affected accordingly. If the EEPROM or flash are written, do not calibrate to more than 8.8MHz. Otherwise, the EEPROM or flash write may fail.

The CAL7 bit determines the range of operation for the oscillator. Setting this bit to 0 gives the lowest frequency range, setting this bit to 1 gives the highest frequency range. The two frequency ranges are overlapping, in other words a setting of OSCCAL = 0x7F gives a higher frequency than OSCCAL = 0x80.

The CAL6..0 bits are used to tune the frequency within the selected range. A setting of 0x00 gives the lowest frequency in that range, and a setting of 0x7F gives the highest frequency in the range. Incrementing CAL6..0 by 1 will give a frequency increment of less than 2% in the frequency range 7.3 - 8.1MHz.

4.5.2 CLKPR – Clock Prescaler Register

Bit	7	6	5	4	3	2	1	0	
	CLKPCE	**–**	**–**	**–**	**CLKPS3**	**CLKPS2**	**CLKPS1**	**CLKPS0**	**CLKPR**
Read/Write	R/W	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	See Bit Description				

4.5.4 CLKSELR - Clock Selection Register

Bit	7	6	5	4	3	2	1	0	
	-	COUT	CSUT1	CSUT0	CSEL3	CSEL2	CSEL1	CSEL0	CLKSELR
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	~ (CKOUT) fuse	SUT1..0 fuses			CKSEL3..0 fuses			

- **Bit 7– Res: Reserved Bit**

This bit is reserved bit in the Atmel® ATtiny87/167 and will always read as zero.

- **Bit 6 – COUT: Clock Out**

The COUT bit is initialized with ~(CKOUT) fuse bit.
The COUT bit is only used in case of '*CKOUT*' command. Refer to Section 4.2.7 "Clock Output Buffer" on page 32 for using.
In case of '*recover system clock Source*' command, COUT it is not affected (no recovering of this setting).

- **Bits 5:4 – CSUT1:0: Clock Start-up Time**

CSUT bits are initialized with the values of SUT fuse bits.
In case of '*enable/disable clock source*' command, CSUT field provides the code of the clock start-up time. Refer to subdivisions of Section 4.2 "Clock Sources" on page 26 for code of clock start-up times.
In case of '*recover system clock source*' command, CSUT field is not affected (no recovering of SUT code).

- **Bits 3:0 – CSEL3:0: Clock Source Select**

CSEL bits are initialized with the values of CKSEL fuse bits.
In case of '*enable/disable clock source*', '*request for clock availability*' or '*clock source switch*' command, CSEL field provides the code of the clock source. Refer to Table 4-1 on page 26 and subdivisions of Section 4.2 "Clock Sources" on page 26 for clock source codes.
In case of '*recover system clock source*' command, CSEL field contains the code of the clock source used to drive the clock control unit as described in Figure 4-1 on page 25.

6. System Control and Reset

6.1 Reset

6.1.1 Resetting the AVR

During reset, all I/O registers are set to their initial values, and the program starts execution from the reset vector. The instruction placed at the reset vector must be an RJMP – Relative Jump – instruction to the reset handling routine. If the program never enables an interrupt source, the interrupt vectors are not used, and regular program code can be placed at these locations. The circuit diagram in Figure 6-1 shows the reset circuit. Tables in Section 22.5 “RESET Characteristics” on page 225 defines the electrical parameters of the reset circuitry.

The I/O ports of the AVR® are immediately reset to their initial state when a reset source goes active. This does not require any clock source to be running.

After all reset sources have gone inactive, a delay counter is invoked, stretching the internal reset. This allows the power to reach a stable level before normal operation starts. The time-out period of the delay counter is defined by the user through the SUT and CKSEL fuses. The different selections for the delay period are presented in Section 4.2 “Clock Sources” on page 26.

6.1.2 Reset Sources

The Atmel® ATtiny87/167 has four sources of reset:

- Power-on reset. The MCU is reset when the supply voltage is below the power-on reset threshold (V_{POT}).
- External reset. The MCU is reset when a low level is present on the $\overline{\text{RESET}}$ pin for longer than the minimum pulse length.
- Watchdog system reset. The MCU is reset when the watchdog timer period expires and the watchdog system reset mode is enabled.
- Brown-out reset. The MCU is reset when the supply voltage Vcc is below the brown-out reset threshold (V_{BOT}) and the brown-out detector is enabled.

Bit 0 – INT0: External Interrupt Request 0 Enable

When the INT0 bit is set (one) and the I-bit in the status register (SREG) is set (one), the external pin interrupt is enabled. The interrupt sense control0 bits 1/0 (ISC01 and ISC00) in the external interrupt control register A (EICRA) define whether the external interrupt is activated on rising and/or falling edge of the INT0 pin or level sensed. Activity on the pin will cause an interrupt request even if INT0 is configured as an output. The corresponding interrupt of external interrupt request 0 is executed from the INT0 interrupt vector.

8.3.3 External Interrupt Flag Register – EIFR

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	–	–	INTF1	INTF0	EIFR
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7, 2 – Res: Reserved Bits**

These bits are unused bits in the Atmel® ATtiny87/167, and will always read as zero.

- **Bit 1 – INTF1: External Interrupt Flag 1**

When an edge or logic change on the INT1 pin triggers an interrupt request, INTF1 becomes set (one). If the I-bit in SREG and the INT1 bit in EIMSK are set (one), the MCU will jump to the corresponding interrupt vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it. This flag is always cleared when INT1 is configured as a level interrupt.

- **Bit 0 – INTF0: External Interrupt Flag 0**

When an edge or logic change on the INT0 pin triggers an interrupt request, INTF0 becomes set (one). If the I-bit in SREG and the INT0 bit in EIMSK are set (one), the MCU will jump to the corresponding interrupt vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it. This flag is always cleared when INT0 is configured as a level interrupt.

8.3.4 Pin Change Interrupt Control Register – PCICR

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	–	–	PCIE1	PCIE0	PCICR
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7, 2 – Res: Reserved Bits**

These bits are unused bits in the Atmel ATtiny87/167, and will always read as zero.

- **Bit 1 - PCIE1: Pin Change Interrupt Enable 1**

When the PCIE1 bit is set (one) and the I-bit in the status register (SREG) is set (one), pin change interrupt 1 is enabled. Any change on any enabled PCINT15..8 pin will cause an interrupt. The corresponding interrupt of pin change interrupt request is executed from the PCI1 interrupt vector. PCINT15..8 pins are enabled individually by the PCMSK1 register.

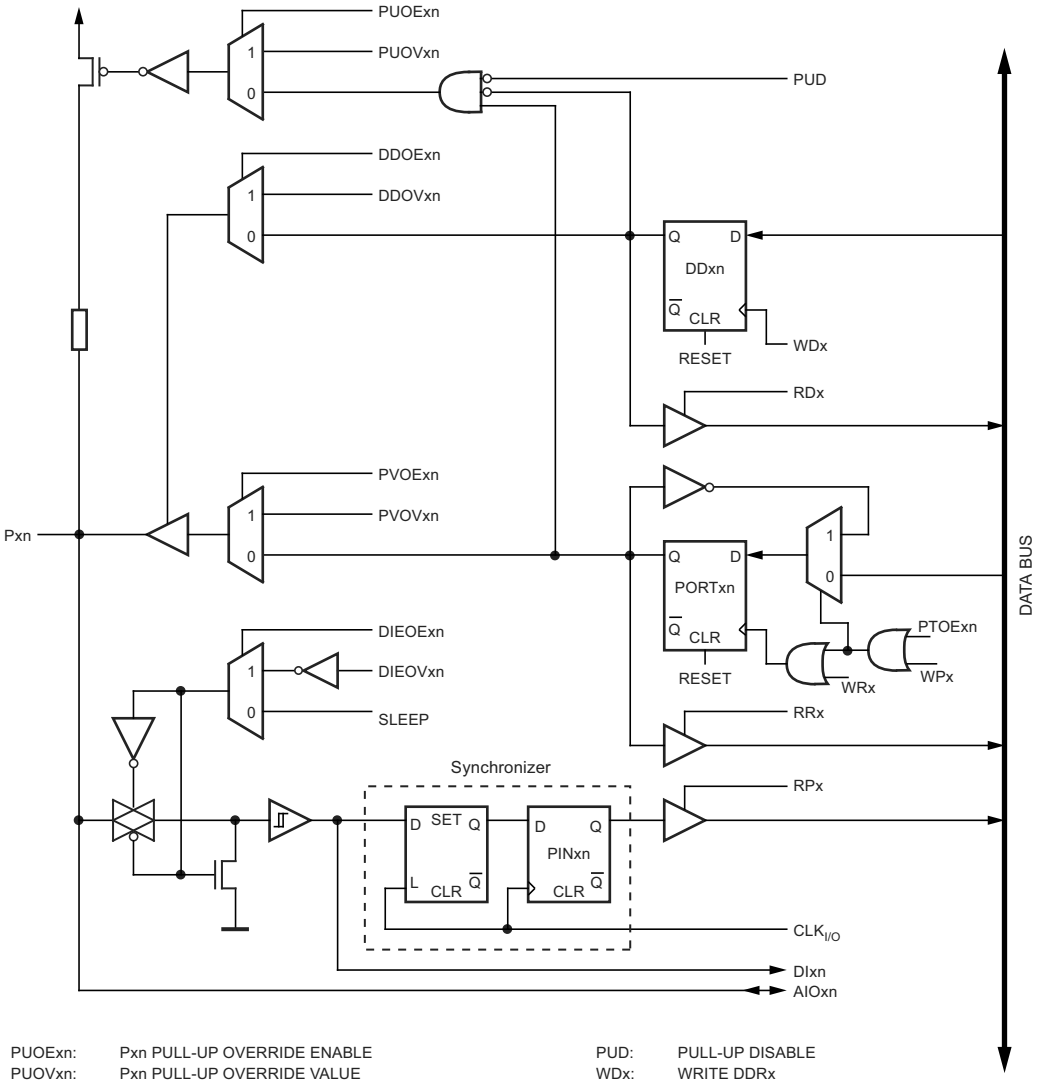
- **Bit 0 - PCIE0: Pin Change Interrupt Enable 0**

When the PCIE0 bit is set (one) and the I-bit in the status register (SREG) is set (one), pin change interrupt 0 is enabled. Any change on any enabled PCINT7..0 pin will cause an interrupt. The corresponding interrupt of pin change interrupt request is executed from the PCI0 Interrupt Vector. PCINT7..0 pins are enabled individually by the PCMSK0 register.

9.3 Alternate Port Functions

Most port pins have alternate functions in addition to being general digital I/Os. Figure 9-6 shows how the port pin control signals from the simplified Figure 9-2 on page 65 can be overridden by alternate functions. The overriding signals may not be present in all port pins, but the figure serves as a generic description applicable to all port pins in the AVR microcontroller family.

Figure 9-6. Alternate Port Functions(1)



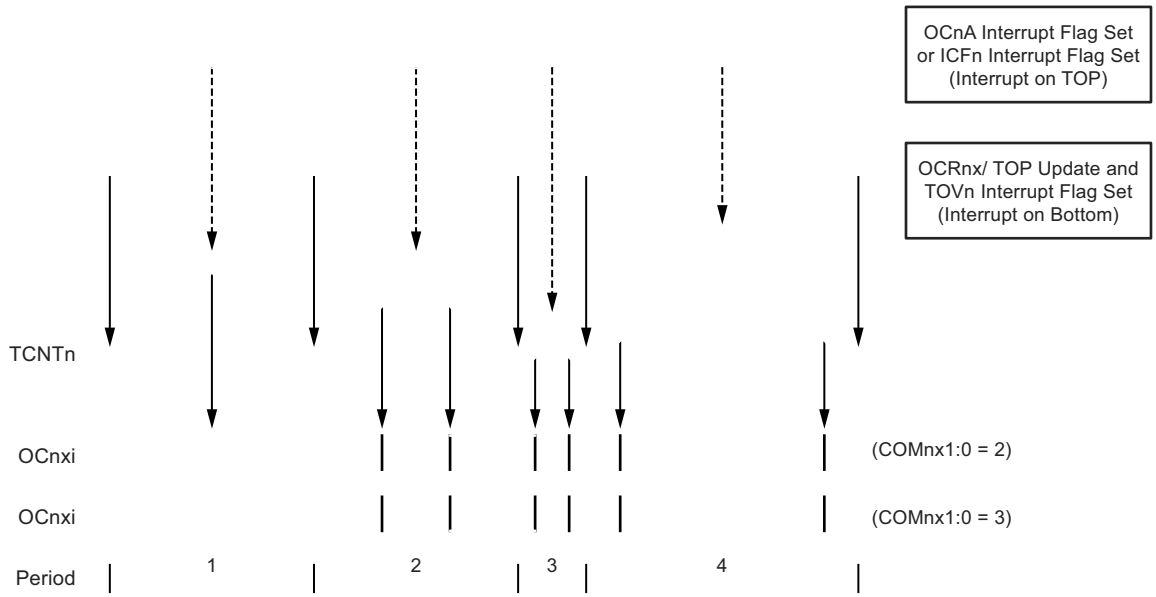
PUOExn:	Pxn PULL-UP OVERRIDE ENABLE	PUD:	PULL-UP DISABLE
PUOVxn:	Pxn PULL-UP OVERRIDE VALUE	WDx:	WRITE DDRx
DDOExn:	Pxn DATA DIRECTION OVERRIDE ENABLE	RDx:	READ DDRx
DDOVxn:	Pxn DATA DIRECTION OVERRIDE VALUE	RRx:	READ PORTx REGISTER
PVOExn:	Pxn PORT VALUE OVERRIDE ENABLE	WRx:	WRITE PORTx
PVOVxn:	Pxn PORT VALUE OVERRIDE VALUE	RPx:	READ PORTx PIN
DIEOExn:	Pxn DIGITAL INPUT ENABLE OVERRIDE ENABLE	WPx:	WRITE PINx
DIEOVxn:	Pxn DIGITAL INPUT ENABLE OVERRIDE VALUE	$CLK_{I/O}$:	I/O CLOCK
SLEEP:	SLEEP CONTROL	DIxn:	DIGITAL INPUT PIN n ON PORTx
PTOExn:	Pxn, PORT TOGGLE OVERRIDE ENABLE	AIOxn:	ANALOG INPUT/OUTPUT PIN n ON PORTx

Note: 1. WRx, WPx, WDx, RRx, RPx, and RDx are common to all pins within the same port. $clk_{I/O}$, SLEEP, and PUD are common to all ports. All other signals are unique for each pin.

Table 9-8. Overriding Signals for Alternate Functions in PB3..PB0

Signal Name	PB3/PCINT11/ OC1BV	PB2/PCINT10/ OC1AV/USCK/SCL	PB1/PCINT9/ OC1BU/DO	PB0/IPCINT8/ OC1AU/DI/SDA
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	(USI_2_WIRE **&** $\overline{\text{USIPOS}}$)	0	(USI_2_WIRE **&** $\overline{\text{USIPOS}}$)
DDOV	0	(USI_SCL_HOLD **|** $\overline{\text{PORTB2}}$) **&** DDRB2	0	($\overline{\text{USI\_SHIFTOUT}}$ **|** $\overline{\text{PORTB0}}$) **&** DDRB0)
PVOE	OC1B_ENABLE **&** OC1BV	(USI_2_WIRE **&** $\overline{\text{USIPOS}}$ **&** DDRB2) **|** (OC1A_ENABLE **&** OC1AV)	($\overline{\text{USI\_2\_WIRE}}$ **&** USI_3_WIRE **&** $\overline{\text{USIPOS}}$) **|** (OC1B_ENABLE **&** OC1BU)	(USI_2_WIRE **&** $\overline{\text{USIPOS}}$ **&** DDRB0) **|** (OC1A_ENABLE **&** OC1AU)
PVOV	OC1B	**{** (USI_2_WIRE **&** $\overline{\text{USIPOS}}$ **&** DDRB2) **?** (0) **:** (OC1A) **}**	**{** ($\overline{\text{USI\_2\_WIRE}}$ **&** USI_3_WIRE **&** $\overline{\text{USIPOS}}$) **?** (USI_SHIFTOUT) **:** (OC1B) **}**	**{** (USI_2_WIRE **&** $\overline{\text{USIPOS}}$ **&** DDRB0) **?** (0) **:** (OC1A) **}**
PTOE	0	USI_PTOE **&** $\overline{\text{USIPOS}}$	0	0
DIEOE	PCIE1 **&** PCMSK11	(USISIE **&** $\overline{\text{USIPOS}}$) **|** (PCIE1 **&** PCMSK10)	PCIE1 **&** PCMSK9	(USISIE **&** $\overline{\text{USIPOS}}$) **|** (PCIE1 **&** PCMSK8)
DIEOV	1	(USISIE **&** $\overline{\text{USIPOS}}$) **|** (PCIE1 **&** PCMSK10)	1	(USISIE **&** $\overline{\text{USIPOS}}$) **|** (PCIE1 **&** PCMSK8)
DI	PCINT11	PCINT10 **-/-** USCK **-/-** SCL	PCINT9	PCINT8 **-/-** DI **-/-** SDA
AIO	0	0	0	0

Figure 12-10.Phase and Frequency Correct PWM Mode, Timing Diagram



The timer/counter overflow flag (TOV1) is set at the same timer clock cycle as the OCR1A/B registers are updated with the double buffer value (at BOTTOM). When either OCR1A or ICR1 is used for defining the TOP value, the OC1A or ICF1 flag set when TCNT1 has reached TOP. The interrupt flags can then be used to generate an interrupt each time the counter reaches the TOP or BOTTOM value.

When changing the TOP value the program must ensure that the new TOP value is higher or equal to the value of all of the compare registers. If the TOP value is lower than any of the compare registers, a compare match will never occur between the TCNT1 and the OCR1A/B.

As Figure 12-10 shows the output generated is, in contrast to the phase correct mode, symmetrical in all periods. Since the OCR1A/B registers are updated at BOTTOM, the length of the rising and the falling slopes will always be equal. This gives symmetrical output pulses and is therefore frequency correct.

Using the ICR1 register for defining TOP works well when using fixed TOP values. By using ICR1, the OCR1A register is free to be used for generating a PWM output on OC1A. However, if the base PWM frequency is actively changed by changing the TOP value, using the OCR1A as TOP is clearly a better choice due to its double buffer feature.

In phase and frequency correct PWM mode, the compare units allow generation of PWM waveforms on the OC1A/B pins. Setting the COM1A/B1:0 bits to two will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COM1A/B1:0 to three (See Table on page 125). The actual OC1A/B value will only be visible on the port pin if the data direction for the port pin is set as output (DDR_OC1A/B) and OC1A/Bi is set. The PWM waveform is generated by setting (or clearing) the OC1A/B register at the compare match between OCR1A/B and TCNT1 when the counter increments, and clearing (or setting) the OC1A/B register at compare match between OCR1A/B and TCNT1 when the counter decrements. The PWM frequency for the output when using phase and frequency correct PWM can be calculated by the following equation:

$$f_{OCnxPFCPWM} = \frac{f_{\text{clk\_I/O}}}{2 \cdot N \cdot TOP}$$

The N variable represents the prescaler divider (1, 8, 64, 256, or 1024).

The extreme values for the OCR1A/B register represents special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCR1A/B is set equal to BOTTOM the output will be continuously low and if set equal to TOP the output will be set to high for non-inverted PWM mode. For inverted PWM the output will have the opposite logic values.

The 4-bit counter can be both read and written via the data bus, and can generate an overflow interrupt. Both the USI data register and the counter are clocked simultaneously by the same clock source. This allows the counter to count the number of bits received or transmitted and generate an interrupt when the transfer is complete. Note that when an external clock source is selected the counter counts both clock edges. In this case the counter counts the number of edges, and not the number of bits. The clock can be selected from three different sources: The USCK pin, timer/counter0 compare match or from software.

The Two-wire clock control unit can generate an interrupt when a start condition is detected on the two-wire bus. It can also generate wait states by holding the clock pin low after a start condition is detected, or after the counter overflows.

14.3 Functional Descriptions

14.3.1 Three-wire Mode

The USI three-wire mode is compliant to the serial peripheral interface (SPI) mode 0 and 1, but does not have the slave select ($\overline{SS}$) pin functionality. However, this feature can be implemented in software if necessary. Pin names used by this mode are: DI, DO, and USCK.

Figure 14-2. Three-wire Mode Operation, Simplified Diagram

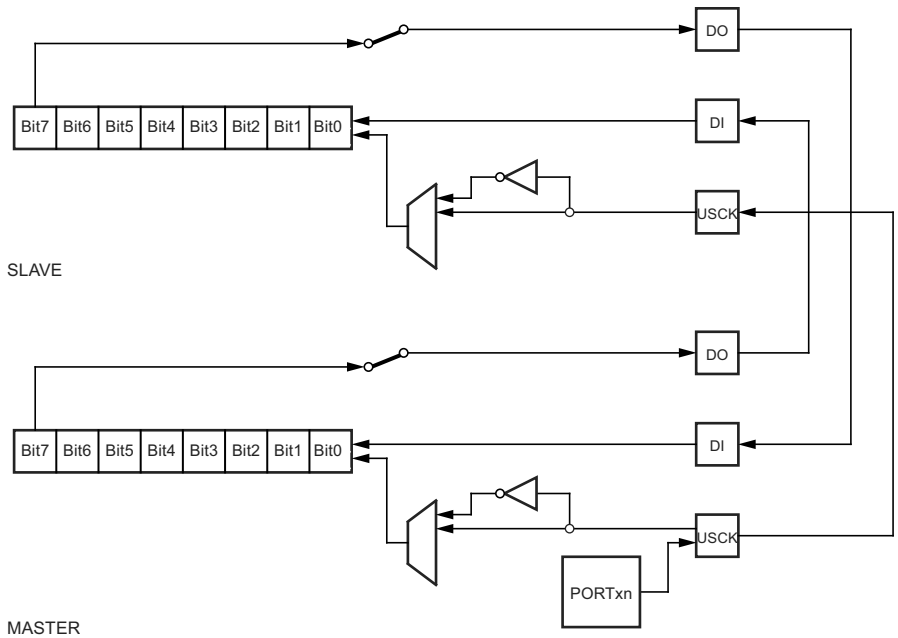


Figure 14-2 shows two USI units operating in three-wire mode, one as master and one as slave. The two USI data register are interconnected in such way that after eight USCK clocks, the data in each register are interchanged. The same clock also increments the USI's 4-bit counter. The counter overflow (interrupt) flag, or USIOIF, can therefore be used to determine when a transfer is completed.

The clock is generated by the master device software by toggling the USCK pin via the PORT register or by writing a one to the USITC bit in USICR.

Table 14-1. Relations between USIWM1..0 and the USI Operation

USIWM1	USIWM0	Description
0	0	Outputs, clock hold, and start detector disabled. Port pins operates as normal.
0	1	Three-wire mode. Uses DO, DI, and USCK pins. The *data output* (DO) pin overrides the corresponding bit in the PORT register in this mode. However, the corresponding DDR bit still controls the data direction. When the port pin is set as input the pins pull-up is controlled by the PORT bit. The *data input* (DI) and *serial clock* (USCK) pins do not affect the normal port operation. When operating as master, clock pulses are software generated by toggling the PORT register, while the data direction is set to output. The USITC bit in the USICR register can be used for this purpose.
1	0	Two-wire mode. Uses SDA (DI) and SCL (USCK) pins(1). The *serial data* (SDA) and the *serial clock* (SCL) pins are bi-directional and uses open-collector output drives. The output drivers are enabled by setting the corresponding bit for SDA and SCL in the DDR register. When the output driver is enabled for the SDA pin, the output driver will force the line SDA low if the output of the USI data register or the corresponding bit in the PORT register is zero. Otherwise the SDA line will not be driven (i.e., it is released). When the SCL pin output driver is enabled the SCL line will be forced low if the corresponding bit in the PORT register is zero, or by the start detector. Otherwise the SCL line will not be driven. The SCL line is held low when a start detector detects a start condition and the output is enabled. Clearing the start condition flag (USISIF) releases the line. The SDA and SCL pin inputs is not affected by enabling this mode. Pull-ups on the SDA and SCL port pin are disabled in Two-wire mode.
1	1	Two-wire mode. Uses SDA and SCL pins. Same operation as for the two-wire mode described above, except that the SCL line is also held low when a counter overflow occurs, and is held low until the counter overflow flag (USIOIF) is cleared.

Note: 1. The DI and USCK pins are renamed to *serial data* (SDA) and *serial clock* (SCL) respectively to avoid confusion between the modes of operation.

- **Bit 3:2 – USICS1:0: Clock Source Select**

These bits set the clock source for the USI data register and counter. The data output latch ensures that the output is changed at the opposite edge of the sampling of the data input (DI/SDA) when using external clock source (USCK/SCL). When software strobe or timer/counter0 compare match clock option is selected, the output latch is transparent and therefore the output is changed immediately. Clearing the USICS1:0 bits enables software strobe option. When using this option, writing a one to the USICLK bit clocks both the USI data register and the counter. For external clock source (USICS1 = 1), the USICLK bit is no longer used as a strobe, but selects between external clocking and software clocking by the USITC strobe bit.

Table 14-2 on page 148 shows the relationship between the USICS1..0 and USICLK setting and clock source used for the USI data register and the 4-bit counter.

15.4.7.2 Rx Service

Once this service is enabled, the user is warned of an in-coming character by the LRXOK flag of LINSIR register. Reading LINDAT register automatically clears the flag and makes free the second stage of the buffer. If the user considers that the in-coming character is irrelevant without reading it, he directly can clear the flag (see specific flag management described in Section 15.6.2 “LIN Status and Interrupt Register - LINSIR” on page 167).

The intrinsic structure of the Rx service offers a 2-byte buffer. The fist one is used for serial to parallel conversion, the second one receives the result of the conversion. This second buffer byte is reached reading LINDAT register. If the 2-byte buffer is full, a new in-coming character will overwrite the second one already recorded. An OVRERR error in LINERR register will then accompany this character when read.

A FERR error in LINERR register will be set in case of framing error.

15.4.7.3 Tx Service

If this service is enabled, the user sends a character by writing in LINDAT register. Automatically the LTXOK flag of LINSIR register is cleared. It will rise at the end of the serial transmission. If no new character has to be sent, LTXOK flag can be cleared separately (see specific flag management described in Section 15.6.2 “LIN Status and Interrupt Register - LINSIR” on page 167).

There is no transmit buffering.

No error is detected by this service.

15.5 LIN/UART Description

15.5.1 Reset

The AVR® core reset logic signal also resets the LIN/UART controller. Another form of reset exists, a software reset controlled by LSWRES bit in LINCR register. This self-reset bit performs a partial reset as shown in Table 15-2.

Table 15-2. Reset of LIN/UART Registers

Register	Name	Reset Value	LSWRES Value	Comment
LIN control reg.	LINCR	0000 0000 $_b$	0000 0000 $_b$	x=unknown u=unchanged
LIN status & interrupt reg.	LINSIR	0000 0000 $_b$	0000 0000 $_b$	
LIN enable interrupt reg.	LINENIR	0000 0000 $_b$	xxxx 0000 $_b$	
LIN error reg.	LINERR	0000 0000 $_b$	0000 0000 $_b$	
LIN bit timing reg.	LINBTR	0010 0000 $_b$	0010 0000 $_b$	
LIN baud rate reg. low	LINBRRL	0000 0000 $_b$	uuuu uuuu $_b$	
LIN baud rate reg. high	LINBRRH	0000 0000 $_b$	xxxx uuuu $_b$	
LIN data length reg.	LINDLR	0000 0000 $_b$	0000 0000 $_b$	
LIN identifier reg.	LINIDR	1000 0000 $_b$	1000 0000 $_b$	
LIN data buffer selection	LINSEL	0000 0000 $_b$	xxxx 0000 $_b$	
LIN data	LINDAT	0000 0000 $_b$	0000 0000 $_b$	

15.5.2 Clock

The I/O clock signal ($clk_{i/o}$) also clocks the LIN/UART controller. It is its unique clock.

15.5.3 LIN Protocol Selection

LIN13 bit in LINCR register is used to select the LIN protocol:

- LIN13 = 0 (default): LIN 2.1 protocol,
- LIN13 = 1: LIN 1.3 protocol.

The controller checks the LIN13 bit in computing the checksum (enhanced checksum in LIN2.1 / classic checksum in LIN 1.3). This bit is irrelevant for UART commands.

15.5.4 Configuration

Depending on the mode (LIN or UART), LCONF[1..0] bits of the LINCR register set the controller in the following configuration (see Table 15-3).

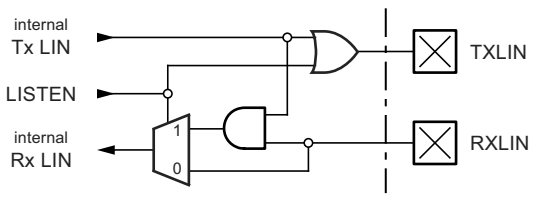
Table 15-3. Configuration Table versus Mode

Mode	LCONF[1..0]	Configuration
LIN	00_b	LIN standard configuration (default)
	01_b	No CRC field detection or transmission
	10_b	Frame_time_out disable
	11_b	Listening mode
UART	00_b	8-bit data, no parity and 1 stop-bit
	01_b	8-bit data, even parity and 1 stop-bit
	10_b	8-bit data, odd parity and 1 stop-bit
	11_b	Listening mode, 8-bit data, no parity and 1 stop-bit

The LIN configuration is independent of the programmed LIN protocol.

The listening mode connects the internal Tx LIN and the internal Rx LIN together. In this mode, the TXLIN output pin is disabled and the RXLIN input pin is always enabled. The same scheme is available in UART mode.

Figure 15-6. Listening Mode

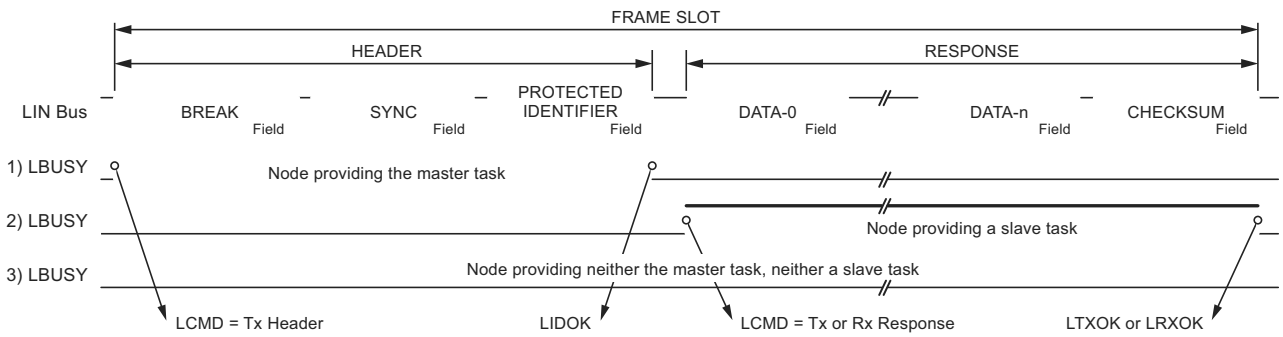


15.5.5 Busy Signal

LBUSY bit flag in LINSIR register is the image of the BUSY signal. It is set and cleared by hardware. It signals that the controller is busy with LIN or UART communication.

15.5.5.1 Busy Signal in LIN Mode

Figure 15-7. Busy Signal in LIN Mode



16.2.2 Current Source for Low Cost Transducer

An external transducer based on a variable resistor can be connected to the current source.
This can be, for instance:

- A thermistor, or temperature-sensitive resistor, used as a temperature sensor,
- A CdS photoconductive cell, or luminosity-sensitive resistor, used as a luminosity sensor,
- ...

Using the current source with this type of transducer eliminates the need for additional parts otherwise required in resistor network or wheatstone bridge.

16.2.3 Voltage Reference for External Devices

An external resistor used in conjunction with the current source can be used as voltage reference for external devices. Using a resistor in serie with a lower tolerance than the current source accuracy (≤ 2%) is recommended. Table 16-2 on page 174 gives an example of voltage references using standard values of resistors.

16.2.4 Threshold Reference for Internal Analog Comparator

An external resistor used in conjunction with the current source can be used as threshold reference for internal analog Comparator (see Section 18. “AnaComp - Analog Comparator” on page 194). This can be connected to AIN0 (negative analog compare input pin) as well as AIN1 (positive analog compare input pin). Using a resistor in serie with a lower tolerance than the current source accuracy (≤ 2%) is recommended. Table 16-2 on page 174 gives an example of threshold references using standard values of resistors.

16.3 Control Register

16.3.1 AMISCR – Analog Miscellaneous Control Register

Bit	7	6	5	4	3	2	1	0	
	-	-	-	-	-	**AREFEN**	**XREFEN**	**ISRCEN**	**AMISCR**
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 0 – ISRCEN: Current Source Enable**

Writing this bit to one enables the current source as shown in Figure 16-1 on page 173. It is recommended to use DIDR register bit function when ISRCEN is set. It also recommended to turn off the current source as soon as possible (ex: once the ADC measurement is done).

```
Rdloop:
  lpm    r0, Z+
  ld     r1, Y+
  cpse   r0, r1
  rjmp   Error
  sbiw   loophi:looplo, 1         ; use subi for PAGESIZEB<=256
  brne   Rdloop

  ; To ensure compatibility with devices supporting Read-While-Write
  ; Return to RWW section
  ; Verify that RWW section is safe to read

Return:
  in     temp1, SPMCSR
  sbrs   temp1, RWWSB     ; If RWWSB is set, the RWW section is not ready yet
  ret
  ; Clear temporary page buffer
  ldi    spmcsrval, (1<<CPTB) | (1<<SELFPGEN)
  call   Do_spm
  rjmp   Return

Do_spm:
  ; Check for previous SPM complete
Wait_spm:
  in     temp1, SPMCSR
  sbrc   temp1, SELFPGEN
  rjmp   Wait_spm
  ; Input: spmcsrval determines SPM action
  ; Disable interrupts if enabled, store status
  in     temp2, SREG
  cli
  ; Check that no EEPROM write access is present
Wait_ee:
  sbic   EECR, EEPE
  rjmp   Wait_ee
  ; SPM timed sequence
  out    SPMCSR, spmcsrval
  spm
  ; Restore SREG (to enable interrupts if originally enabled)
  out    SREG, temp2
  ret
```

22.8 ADC Characteristics

Table 22-9. ADC Characteristics, Single Ended Channels (–40°C/+125°C)

Parameter	Condition	Symbol	Min	Typ	Max	Units
Resolution	Single ended conversion			10		Bits
Absolute accuracy	Vcc = 4V, VRef = 4V, ADC clock = 200kHz	TUE		2.0	3.5	LSB
Integral non linearity	Vcc = 4V, VRef = 4V, ADC clock = 200kHz	INL		0.6	2.0	LSB
Differential non linearity	Vcc = 4V, VRef = 4V, ADC clock = 200kHz	DNL		0.3	0.8	LSB
Gain error	Vcc = 4V, VRef = 4V, ADC clock = 200kHz		-6.0	-2.5	2.0	LSB
Offset error	Vcc = 4V, VRef = 4V, ADC clock = 200kHz		-3.5	1.5	3.5	LSB
Ref voltage		V_{REF}	2.56		AVcc	V
Input bandwidth				38.5		kHz
Internal voltage		V_{INT}	2.4	2.56	2.7	V
Reference input resistance		R_{REF}		32		kΩ
Analog input resistance		R_{AIN}		100		MΩ

Table 22-10. ADC Characteristics, Differential Channels (–40°C/+125°C)

Parameter	Condition	Symbol	Min	Typ	Max	Units
Resolution	Differential conversion			8		
Absolute accuracy	Gain = 8x, BIPOLAR V_{REF} = 4V, Vcc = 5V ADC clock = 200kHz	TUE		1.0	3.0	LSB
	Gain = 20x, BIPOLAR V_{REF} = 4V, Vcc = 5V ADC clock = 200kHz			1.5	3.5	
	Gain = 8x, UNIPOLAR V_{REF} = 4V, Vcc = 5V ADC clock = 200kHz			2.0	4.5	
	Gain = 20x, UNIPOLAR V_{REF} = 4V, Vcc = 5V ADC clock = 200kHz			2.0	6.0	
Integral non linearity	Gain = 8x, BIPOLAR V_{REF} = 4V, Vcc = 5V ADC clock = 200kHz	INL		0.2	1.0	LSB
	Gain = 20x, BIPOLAR V_{REF} = 4V, Vcc = 5V ADC clock = 200kHz			0.4	1.5	
	Gain = 8x, UNIPOLAR V_{REF} = 4V, Vcc = 5V ADC clock = 200kHz			0.5	2.0	
	Gain = 20x, UNIPOLAR V_{REF} = 4V, Vcc = 5V ADC clock = 200kHz			1.6	5.0	

25. Register Summary (Continued)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
0x2D (0x4D)	SPSR	SPIF	WCOL	–	–	–	–	–	SPI2X	135
0x2C (0x4C)	SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	134
0x2B (0x4B)	GPIOR2	GPIOR27	GPIOR26	GPIOR25	GPIOR24	GPIOR23	GPIOR22	GPIOR21	GPIOR20	24
0x2A (0x4A)	GPIOR1	GPIOR17	GPIOR16	GPIOR15	GPIOR14	GPIOR13	GPIOR12	GPIOR11	GPIOR10	24
0x29 (0x49)	Reserved									
0x28 (0x48)	OCR0A	OCR0A7	OCR0A6	OCR0A5	OCR0A4	OCR0A3	OCR0A2	OCR0A1	OCR0A0	98
0x27 (0x47)	TCNT0	TCNT07	TCNT06	TCNT05	TCNT04	TCNT03	TCNT02	TCNT01	TCNT00	98
0x26 (0x46)	TCCR0B	FOC0A	–	–	–	–	CS02	CS01	CS00	97
0x25 (0x45)	TCCR0A	COM0A1	COM0A0	–	–	–	–	WGM01	WGM00	95
0x24 (0x44)	Reserved									
0x23 (0x43)	GTCCR	TSM	–	–	–	–	–	PSR0	PSR1	100, 102
0x22 (0x42)	EEARH(1)	–	–	–	–	–	–	–	EEAR8	22
0x21 (0x41)	EEARL	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	22
0x20 (0x40)	EEDR	EEDR7	EEDR6	EEDR5	EEDR4	EEDR3	EEDR2	EEDR1	EEDR0	23
0x1F (0x3F)	EECR	–	–	EEPM1	EEPM0	EERIE	EEMPE	EEPE	EERE	23
0x1E (0x3E)	GPIOR0	GPIOR07	GPIOR06	GPIOR05	GPIOR04	GPIOR03	GPIOR02	GPIOR01	GPIOR00	24
0x1D (0x3D)	EIMSK	–	–	–	–	–	–	INT1	INT0	61
0x1C (0x3C)	EIFR	–	–	–	–	–	–	INTF1	INTF0	62
0x1B (0x3B)	PCIFR	–	–	–	–	–	–	PCIF1	PCIF0	63
0x1A (0x3A)	Reserved									
0x19 (0x39)	Reserved									
0x18 (0x38)	Reserved									
0x17 (0x37)	Reserved									
0x16 (0x36)	TIFR1	–	–	ICF1	–	–	OCF1B	OCF1A	TOV1	129
0x15 (0x35)	TIFR0	–	–	–	–	–	–	OCF0A	TOV0	99
0x14 (0x34)	Reserved									
0x13 (0x33)	Reserved									
0x12 (0x32)	PORTCR	–	–	BBMB	BBMA	–	–	PUDB	PUDA	72
0x11 (0x31)	Reserved									
0x10 (0x30)	Reserved									
0x0F (0x2F)	Reserved									
0x0E (0x2E)	Reserved									
0x0D (0x2D)	Reserved									
0x0C (0x2C)	Reserved									
0x0B (0x2B)	Reserved									
0x0A (0x2A)	Reserved									

Notes:
1. Address bits exceeding EEAMSB (Table 21-8 on page 210) are don't care.
2. For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.
3. 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.
4. Some of the status flags are cleared by writing a logical one to them. Note that, unlike most other AVRs, 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 to 0x1F only.
5. When using the I/O specific commands IN and OUT, the I/O addresses 0x00 - 0x3F must be used. When addressing I/O registers as data space using LD and ST instructions, 0x20 must be added to these addresses. The Atmel® ATtiny87/167 is a complex microcontroller with more peripheral units than can be supported within the 64 location reserved in opcode for the IN and OUT instructions. For the extended I/O space from 0x60 - 0xFF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.