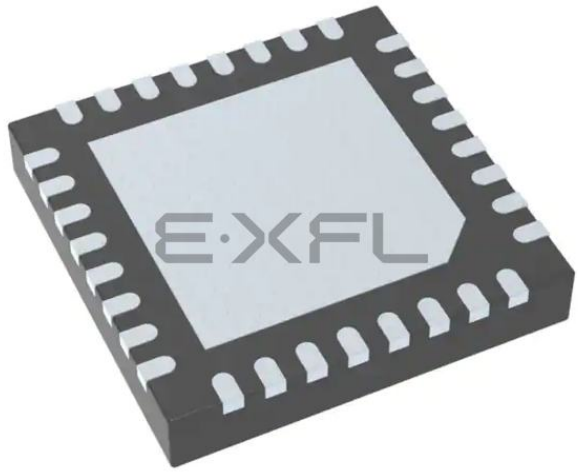


Welcome to E-XFL.COM

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	8KB (4K 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/attiny87-15mz

1.5 Block Diagram

Figure 1-1. Block Diagram

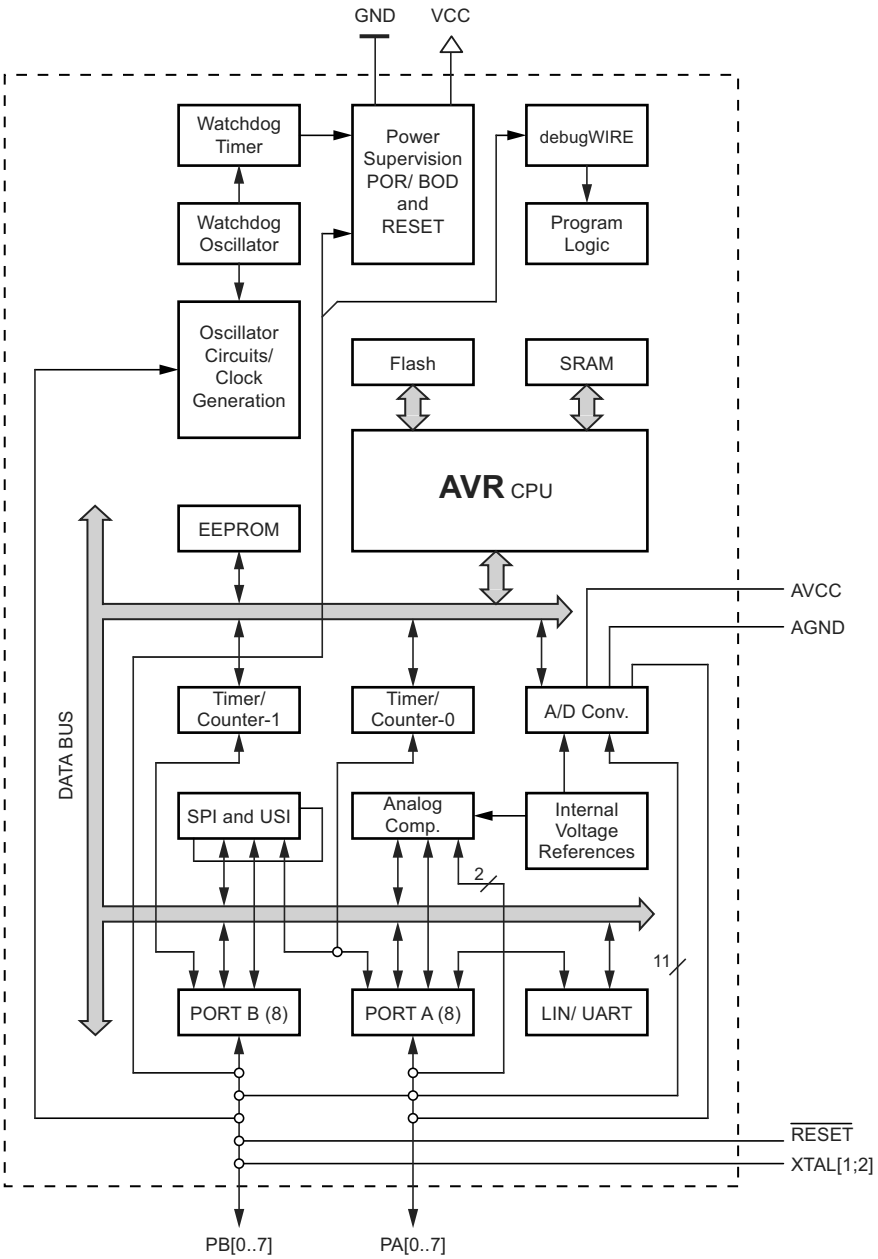
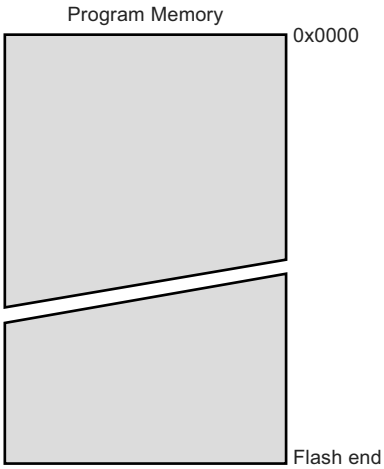


Figure 3-1. Program Memory Map



3.2 SRAM Data Memory

Figure 3-2 shows how the Atmel® ATtiny87/167 SRAM memory is organized.

The ATtiny87/167 is a complex microcontroller with more peripheral units than can be supported within the 64 locations reserved in the opcode for the IN and OUT instructions. For the extended I/O space in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

The data memory locations address both the register file, the I/O memory, extended I/O memory, and the internal data SRAM. The first 32 locations address the register file, the next 64 location the standard I/O memory, then 160 locations of extended I/O memory, and the next locations address the internal data SRAM (see “ISRAM size” in Table 3-1 on page 16).

The five different addressing modes for the data memory cover: direct, indirect with displacement, indirect, indirect with pre-decrement, and indirect with post-increment. In the register file, registers R26 to R31 feature the indirect addressing pointer registers.

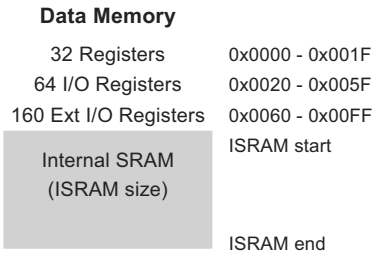
The direct addressing reaches the entire data space.

The indirect with displacement mode reaches 63 address locations from the base address given by the Y- or Z-register.

When using register indirect addressing modes with automatic pre-decrement and post-increment, the address registers X, Y, and Z are decremented or incremented.

The 32 general purpose working registers, 64 I/O Registers, 160 extended I/O registers and the internal data SRAM in the ATtiny87/167 are all accessible through all these addressing modes. The register file is described in Section 2.4 “General Purpose Register File” on page 11.

Figure 3-2. Data Memory Map



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

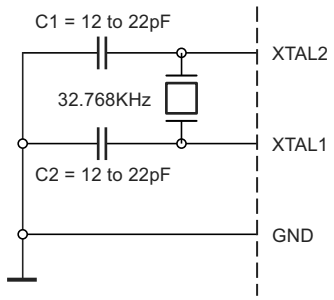
Notes:
1. Flash fuse bits.
2. CLKSELR register bits.
3. These options should only be used when not operating close to the maximum frequency of the device, and only if frequency stability at start-up is not important for the application. These options are not suitable for crystals.
4. These options are 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

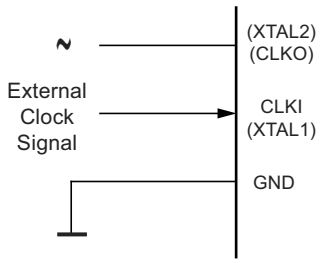
SUT1..0(1) CSUT1..0(2)	Start-up Time from Power-down/save	Additional Delay from Reset (Vcc = 5.0V)	Recommended usage
00	1K(1024)CK(3)	4.1ms	Fast rising power or BOD enabled
01	1K(1024)CK(3)	65ms	Slowly rising power
10	32K(32768)CK	65ms	Stable frequency at start-up
11	Reserved		

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



- **Bit 7 – CLKPCE: Clock Prescaler Change Enable**

The CLKPCE bit must be written to logic one to enable change of the CLKPS bits. The CLKPCE bit is only updated when the other bits in CLKPR are simultaneously written to zero. CLKPCE is cleared by hardware four cycles after it is written or when the CLKPS bits are written. Rewriting the CLKPCE bit within this time-out period does neither extend the time-out period, nor clear the CLKPCE bit.

- **Bits 6:4 – Res: Reserved Bits**

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

- **Bits 3:0 – CLKPS3:0: Clock Prescaler Select Bits 3 - 0**

These bits define the division factor between the selected clock source and the internal system clock. These bits can be written run-time to vary the clock frequency to suit the application requirements. As the divider divides the master clock input to the MCU, the speed of all synchronous peripherals is reduced when a division factor is used. The division factors are given in Table 4-10.

To avoid unintentional changes of clock frequency, a special write procedure must be followed to change the CLKPS bits:

1. Write the Clock Prescaler Change Enable (CLKPCE) bit to one and all other bits in CLKPR to zero.
2. Within four cycles, write the desired value to CLKPS while writing a zero to CLKPCE.

Interrupts must be disabled when changing prescaler setting in order not to disturb the procedure.

The CKDIV8 fuse determines the initial value of the CLKPS bits. If CKDIV8 is unprogrammed, the CLKPS bits will be reset to “0000”. If CKDIV8 is programmed, CLKPS bits are reset to “0011”, giving a division factor of eight at start up. This feature should be used if the selected clock source has a higher frequency than the maximum frequency of the device at the present operating conditions. Note that any value can be written to the CLKPS bits regardless of the CKDIV8 Fuse setting. The application software must ensure that a sufficient division factor is chosen if the selected clock source has a higher frequency than the maximum frequency of the device at the present operating conditions. The device is shipped with the CKDIV8 fuse programmed.

Table 4-10. Clock Prescaler Select

CLKPS3	CLKPS2	CLKPS1	CLKPS0	Clock Division Factor
0	0	0	0	1
0	0	0	1	2
0	0	1	0	4
0	0	1	1	8
0	1	0	0	16
0	1	0	1	32
0	1	1	0	64
0	1	1	1	128
1	0	0	0	256
1	0	0	1	Reserved
1	0	1	0	Reserved
1	0	1	1	Reserved
1	1	0	0	Reserved
1	1	0	1	Reserved
1	1	1	0	Reserved
1	1	1	1	Reserved

5. Power Management and Sleep Modes

Sleep modes enable the application to shut down unused modules in the MCU, thereby saving power. The AVR® provides various sleep modes allowing the user to tailor the power consumption to the application's requirements.

When enabled, the Brown-out Detector (BOD) actively monitors the power supply voltage during the sleep periods. To further save power, it is possible to disable the BOD in some sleep modes. See Section 5.2 "BOD Disable" on page 42 for more details.

5.1 Sleep Modes

Figure 4-1 on page 25 presents the different clock systems in the Atmel® ATtiny87/167, and their distribution. The figure is helpful in selecting an appropriate sleep mode. Table 5-1 shows the different sleep modes, their wake up sources and BOD disable ability.

Table 5-1. Active Clock Domains and Wake-up Sources in the Different Sleep Modes

	Active Clock Domains					Oscillators		Wake-up Sources							
Sleep Mode	clk_{CPU}	clk_{FLASH}	clk_{IO}	clk_{ADC}	clk_{ASY}	Main Clock Source Enabled	Timer0 Osc. Enable	INT1, INT0 and Pin Change	SPM/EEPROM Ready	ADC	WDT	USI Start Condition	Timer0	Other I/O	Software BOD Disable
Idle			X	X	X	X	X	X	X	X	X	X	X	X	
ADC Noise Reduction				X	X	X	X	X(1)	X	X	X	X	X		
Power-down								X(1)			X	X			X
Power-Save					X		X	X(1)			X	X	X		X

Note: 1. For INT1 and INT0, only level interrupt.

To enter any of the four sleep modes, the SE bit in SMCR must be written to logic one and a SLEEP instruction must be executed. The SM1, and SM0 bits in the SMCR register select which sleep mode (Idle, ADC noise reduction, power-down, or power-save) will be activated by the SLEEP instruction. See Table 5-2 on page 45 for a summary.

If an enabled interrupt occurs while the MCU is in a sleep mode, the MCU wakes up. The MCU is then halted for four cycles in addition to the start-up time, executes the interrupt routine, and resumes execution from the instruction following SLEEP. The contents of the Register File and SRAM are unaltered when the device wakes up from sleep. If a reset occurs during sleep mode, the MCU wakes up and executes from the reset vector.

5.2 BOD Disable

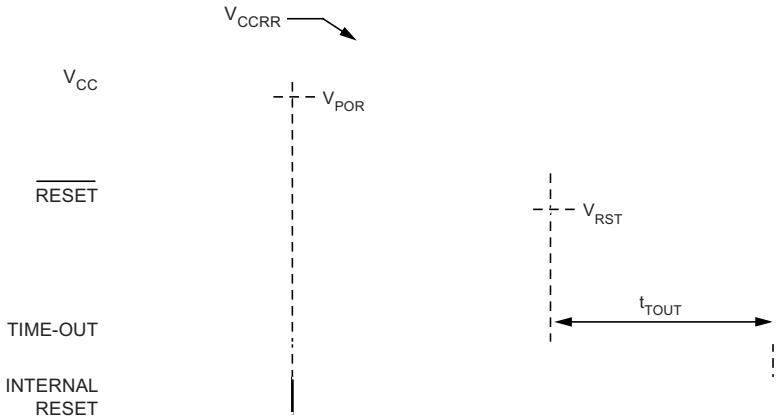
When the brown-out detector (BOD) is enabled by BODLEVEL fuses, Table 21-3 on page 208, the BOD is actively monitoring the power supply voltage during a sleep period. To save power, it is possible to disable the BOD by software for some of the sleep modes, see Table 5-1. The sleep mode power consumption will then be at the same level as when BOD is globally disabled by fuses. If BOD is disabled in software, the BOD function is turned off immediately after entering the sleep mode. Upon wake-up from sleep, BOD is automatically enabled again. This ensures safe operation in case the Vcc level has dropped during the sleep period.

When the BOD has been disabled, the wake-up time from sleep mode will be approximately 60 µs to ensure that the BOD is working correctly before the MCU continues executing code.

BOD disable is controlled by BODS bit (BOD Sleep) in the control register MCUCR, see Section 5.9.2 "MCUCR – MCU Control Register" on page 45. Setting it to one turns off the BOD in relevant sleep modes, while a zero in this bit keeps BOD active. Default setting keeps BOD active, i.e. BODS is cleared to zero.

Writing to the BODS bit is controlled by a timed sequence and an enable bit, see Section 5.9.2 "MCUCR – MCU Control Register" on page 45.

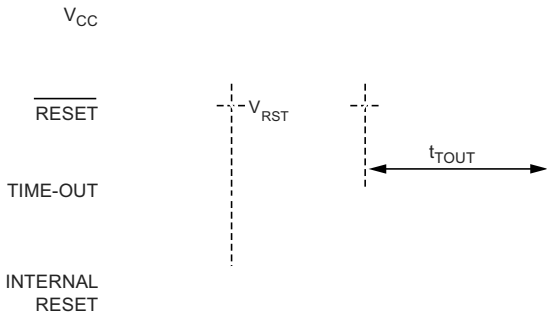
Figure 6-3. MCU Start-up, $\overline{\text{RESET}}$ Extended Externally



6.1.4 External Reset

An external reset is generated by a low level on the $\overline{\text{RESET}}$ pin. Reset pulses longer than the minimum pulse width (see Table 22-3 on page 225) will generate a reset, even if the clock is not running. Shorter pulses are not guaranteed to generate a reset. When the applied signal reaches the reset threshold voltage – V_{RST} – on its positive edge, the delay counter starts the MCU after the time-out period – t_{TOUT} – has expired. The external reset can be disabled by the RSTDISBL fuse, see Table 21-4 on page 208.

Figure 6-4. External Reset During Operation



6.1.5 Brown-out Detection

Atmel® ATtiny87/167 has an on-chip brown-out detection (BOD) circuit for monitoring the Vcc level during operation by comparing it to a fixed trigger level. The trigger level for the BOD can be selected by the BODLEVEL fuses (Section 22-5 “BODLEVEL Fuse Coding” on page 226). The trigger level has a hysteresis to ensure spike free Brown-out Detection. The hysteresis on the detection level should be interpreted as $V_{BOT+} = V_{BOT} + V_{HYST} / 2$ and $V_{BOT-} = V_{BOT} - V_{HYST} / 2$.

- **Bit 2 – BORF: Brown-out Reset Flag**

This bit is set if a brown-out reset occurs. The bit is reset by a power-on reset, or by writing a logic zero to the flag.

- **Bit 1 – EXTRF: External Reset Flag**

This bit is set if an external reset occurs. The bit is reset by a power-on reset, or by writing a logic zero to the flag.

- **Bit 0 – PORF: Power-on Reset Flag**

This bit is set if a power-on reset occurs. The bit is reset only by writing a logic zero to the flag.

To make use of the reset flags to identify a reset condition, the user should read and then reset the MCUSR as early as possible in the program. If the register is cleared before another reset occurs, the source of the reset can be found by examining the reset flags.

6.2 Internal Voltage Reference

Atmel® ATtiny87/167 features an internal bandgap reference. This reference is used for brown-out detection, and it can be used as an input to the analog comparator or the ADC.

6.2.1 Voltage Reference Enable Signals and Start-up Time

The voltage reference has a start-up time that may influence the way it should be used. The start-up time is given in Table 22-7 on page 226. To save power, the reference is not always turned on. The reference is on during the following situations:

1. When the BOD is enabled (by programming the BODLEVEL [2:0] fuses).
2. When the bandgap reference is connected to the analog comparator (by setting the ACIRS bit in ACSR).
3. When the ADC is enabled.

Thus, when the BOD is not enabled, after setting the ACIRS bit or enabling the ADC, the user must always allow the reference to start up before the output from the analog comparator or ADC is used. To reduce power consumption in power-down mode or in power-save, the user can avoid the three conditions above to ensure that the reference is turned off before entering in these power reduction modes.

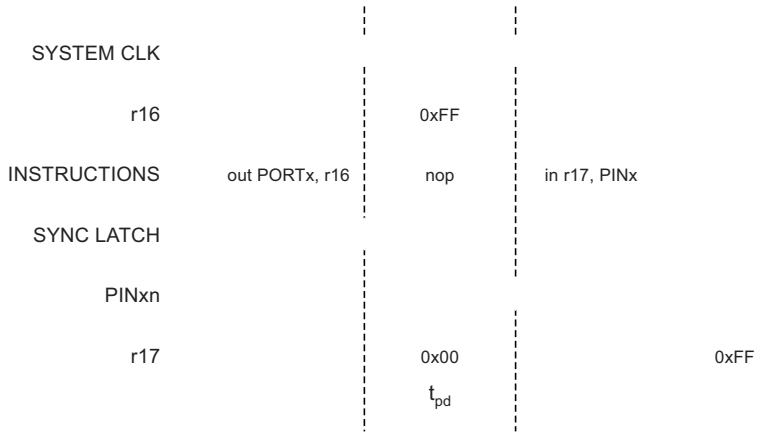
6.3 Watchdog Timer

Atmel ATtiny87/167 has an enhanced watchdog timer (WDT). The main features are:

- Clocked from separate on-chip oscillator
- 4 Operating modes
  - Interrupt
  - System Reset
  - Interrupt and System Reset
  - Clock Monitoring
- Selectable time-out period from 16ms to 8s
- Possible hardware fuse watchdog always on (WDTON) for fail-safe mode

When reading back a software assigned pin value, a nop instruction must be inserted as indicated in Figure 9-5. The out instruction sets the “SYNC LATCH” signal at the positive edge of the clock. In this case, the delay tpd through the synchronizer is 1 system clock period.

Figure 9-5. Synchronization When Reading a Software Assigned Pin Value



The following code example shows how to set port B pins 0 and 1 high, 2 and 3 low, and define the port pins from 4 to 7 as input with pull-ups assigned to port pins 6 and 7. The resulting pin values are read back again, but as previously discussed, a nop instruction is included to be able to read back the value recently assigned to some of the pins.

Assembly Code Example[1]

```
...
; Define pull-ups and set outputs high
; Define directions for port pins
ldi  r16,(1<<PB7)|(1<<PB6)|(1<<PB1)|(1<<PB0)
ldi  r17,(1<<DDB3)|(1<<DDB2)|(1<<DDB1)|(1<<DDB0)
out  PORTB,r16
out  DDRB,r17
; Insert nop for synchronization
nop
; Read port pins
in   r16,PINB
...
```

C Code Example

```
unsigned char i;
...
/* Define pull-ups and set outputs high */
/* Define directions for port pins */
PORTB = (1<<PB7)|(1<<PB6)|(1<<PB1)|(1<<PB0);
DDRB = (1<<DDB3)|(1<<DDB2)|(1<<DDB1)|(1<<DDB0);
/* Insert nop for synchronization*/
__no_operation();
/* Read port pins */
i = PINB;
...
```

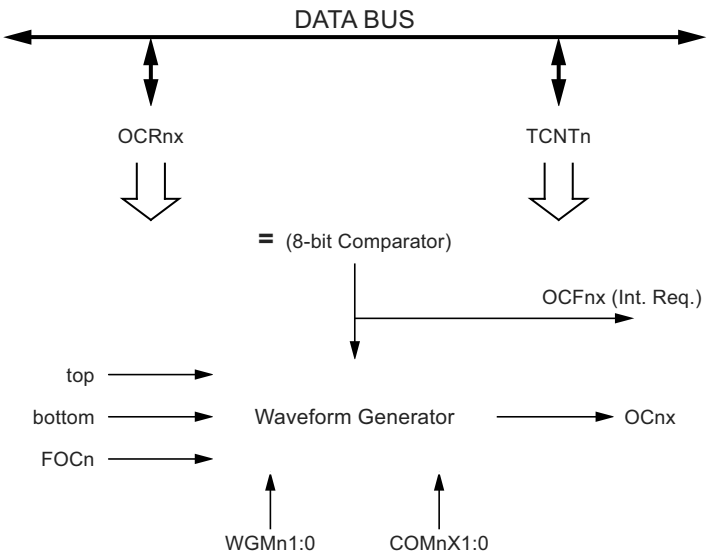
Note: 1. For the assembly program, two temporary registers are used to minimize the time from pull-ups are set on pins 0, 1, 6, and 7, until the direction bits are correctly set, defining bit 2 and 3 as low and redefining bits 0 and 1 as strong high drivers.

10.5 Output Compare Unit

The 8-bit comparator continuously compares TCNT0 with the output compare register (OCR0A). Whenever TCNT0 equals OCR0A, the comparator signals a match. A match will set the output compare flag (OCF0A) at the next timer clock cycle. If enabled (OCIE0A = 1), the output compare flag generates an output compare interrupt. The OCF0A flag is automatically cleared when the interrupt is executed. Alternatively, the OCF0A flag can be cleared by software by writing a logical one to its I/O bit location. The waveform generator uses the match signal to generate an output according to operating mode set by the WGM01:0 bits and compare output mode (COM0A1:0) bits. The max and bottom signals are used by the waveform generator for handling the special cases of the extreme values in some modes of operation (Section 10.7 “Modes of Operation” on page 88).

Figure 10-3 shows a block diagram of the output compare unit.

Figure 10-3. Output Compare Unit, Block Diagram



The OCR0A register is double buffered when using any of the pulse width modulation (PWM) modes. For the normal and clear timer on compare (CTC) modes of operation, the double buffering is disabled. The double buffering synchronizes the update of the OCR0A compare register to either top or bottom of the counting sequence. The synchronization prevents the occurrence of odd-length, non-symmetrical PWM pulses, thereby making the output glitch-free.

The OCR0A register access may seem complex, but this is not case. When the double buffering is enabled, the CPU has access to the OCR0A buffer register, and if double buffering is disabled the CPU will access the OCR0A directly.

10.5.1 Force Output Compare

In non-PWM waveform generation modes, the match output of the comparator can be forced by writing a one to the force output compare (FOC0A) bit. Forcing compare match will not set the OCF0A flag or reload/clear the timer, but the OC0A pin will be updated as if a real compare match had occurred (the COM0A1:0 bits settings define whether the OC0A pin is set, cleared or toggled).

10.5.2 Compare Match Blocking by TCNT0 Write

All CPU write operations to the TCNT0 register will block any compare match that occurs in the next timer clock cycle, even when the timer is stopped. This feature allows OCR0A to be initialized to the same value as TCNT0 without triggering an interrupt when the timer/counter clock is enabled.

10.5.3 Using the Output Compare Unit

Since writing TCNT0 in any mode of operation will block all compare matches for one timer clock cycle, there are risks involved when changing TCNT0 when using the output compare channel, independently of whether the timer/counter is running or not. If the value written to TCNT0 equals the OCR0A value, the compare match will be missed, resulting in incorrect waveform generation. Similarly, do not write the TCNT0 value equal to BOTTOM when the counter is down counting.

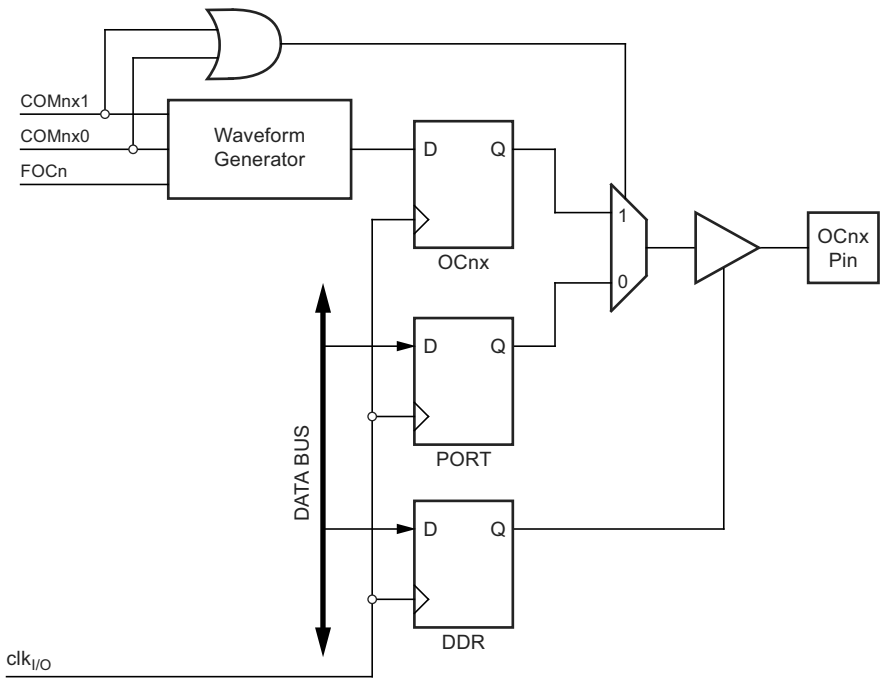
The setup of the OC0A should be performed before setting the data direction register for the port pin to output. The easiest way of setting the OC0A value is to use the force output compare (FOC0A) strobe bit in normal mode. The OC0A register keeps its value even when changing between waveform generation modes.

Be aware that the COM0A1:0 bits are not double buffered together with the compare value. Changing the COM0A1:0 bits will take effect immediately.

10.6 Compare Match Output Unit

The compare output mode (COM0A1:0) bits have two functions. The waveform generator uses the COM0A1:0 bits for defining the output compare (OC0A) state at the next compare match. Also, the COM0A1:0 bits control the OC0A pin output source. Figure 10-4 shows a simplified schematic of the logic affected by the COM0A1:0 bit setting. The I/O Registers, I/O bits, and I/O pins in the figure are shown in bold. Only the parts of the general I/O port control registers (DDR and PORT) that are affected by the COM0A1:0 bits are shown. When referring to the OC0A state, the reference is for the internal OC0A register, not the OC0A pin.

Figure 10-4. Compare Match Output Logic



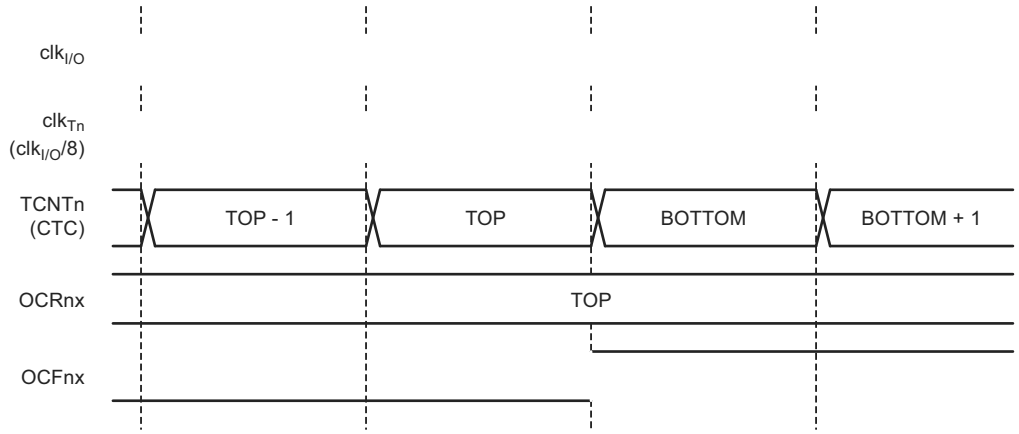
10.6.1 Compare Output Function

The general I/O port function is overridden by the output compare (OC0A) from the waveform generator if either of the COM0A1:0 bits are set. However, the OC0A pin direction (input or output) is still controlled by the data direction register (DDR) for the port pin. The data direction register bit for the OC0A pin (DDR_OC0A) must be set as output before the OC0A value is visible on the pin. The port override function is independent of the waveform generation mode.

The design of the output compare pin logic allows initialization of the OC0A state before the output is enabled. Note that some COM0A1:0 bit settings are reserved for certain modes of operation. Section 10.11 “8-bit Timer/Counter Register Description” on page 95

Figure 10-11 shows the setting of OCF0A and the clearing of TCNT0 in CTC mode.

Figure 10-11.Timer/Counter Timing Diagram, Clear Timer on Compare Match mode, with Prescaler ($f_{clk\_I/O}$/8)



10.9 Asynchronous Operation of Timer/Counter0

When timer/counter0 operates asynchronously, some considerations must be taken.

Warning: When switching between asynchronous and synchronous clocking of timer/counter0, the timer registers TCNT0, OCR0A, and TCCR0A might be corrupted. A safe procedure for switching clock source is:

1. Disable the timer/counter0 interrupts by clearing OCIE0A and TOIE0.
2. Select clock source by setting AS0 and EXCLK as appropriate.
3. Write new values to TCNT0, OCR0A, and TCCR0A.
4. To switch to asynchronous operation: wait for TCN0UB, OCR0UB, and TCR0UB.
5. Clear the timer/counter0 interrupt flags.
6. Enable interrupts, if needed.

- If an 32.768kHz watch crystal is used, the CPU main clock frequency must be more than four times the oscillator or external clock frequency.
- When writing to one of the registers TCNT0, OCR0A, or TCCR0A, the value is transferred to a temporary register, and latched after two positive edges on TOSC1. The user should not write a new value before the contents of the temporary register have been transferred to its destination. Each of the three mentioned registers have their individual temporary register, which means that e.g. writing to TCNT0 does not disturb an OCR0A write in progress. To detect that a transfer to the destination register has taken place, the asynchronous status register – ASSR has been implemented.
- When entering power-save mode after having written to TCNT0, OCR0A, or TCCR0A, the user must wait until the written register has been updated if timer/counter0 is used to wake up the device. Otherwise, the MCU will enter sleep mode before the changes are effective. This is particularly important if the output compare0 interrupt is used to wake up the device, since the output compare function is disabled during writing to OCR0A or TCNT0. If the write cycle is not finished, and the MCU enters sleep mode before the OCR0UB bit returns to zero, the device will never receive a compare match interrupt, and the MCU will not wake up.

10.11.2 Timer/Counter0 Register – TCNT0

Bit	7	6	5	4	3	2	1	0	
	TCNT07	TCNT06	TCNT05	TCNT04	TCNT03	TCNT02	TCNT01	TCNT00	TCNT0
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	0	

The timer/counter register gives direct access, both for read and write operations, to the timer/counter unit 8-bit counter. writing to the TCNT0 register blocks (removes) the compare match on the following timer clock. Modifying the counter (TCNT0) while the counter is running, introduces a risk of missing a compare match between TCNT0 and the OCR0x register.

10.11.3 Output Compare Register A – OCR0A

Bit	7	6	5	4	3	2	1	0	
	OCR0A7	OCR0A6	OCR0A5	OCR0A4	OCR0A3	OCR0A2	OCR0A1	OCR0A0	OCR0A
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	0	

The output compare register A contains an 8-bit value that is continuously compared with the counter value (TCNT0). A match can be used to generate an output compare interrupt, or to generate a waveform output on the OC0A pin.

10.11.4 Asynchronous Status Register – ASSR

Bit	7	6	5	4	3	2	1	0	
	–	EXCLK	AS0	TCN0UB	OCR0AUB	–	TCR0AUB	TCR0BUB	ASSR
Read/Write	R	R/W	R/W	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – Res: Reserved Bit**

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

- **Bit 6 – EXCLK: Enable External Clock Input**

When EXCLK is written to one, and asynchronous clock is selected, the external clock input buffer is enabled and an external clock can be input on XTAL1 pin instead of an external crystal. Writing to EXCLK should be done before asynchronous operation is selected. Note that the crystal oscillator will only run when this bit is zero.

- **Bit 5 – AS0: Asynchronous Timer/Counter0**

When AS0 is written to zero, timer/counter0 is clocked from the I/O clock, clkI/O and the timer/counter0 acts as a synchronous peripheral.

When AS0 is written to one, timer/counter0 is clocked from the low-frequency crystal oscillator (see Section 4.2.5 “Low-frequency Crystal Oscillator” on page 31) or from external clock on XTAL1 pin (see Section 4.2.6 “External Clock” on page 31) depending on EXCLK setting. When the value of AS0 is changed, the contents of TCNT0, OCR0A, and TCCR0A might be corrupted.

AS0 also acts as a flag: timer/counter0 is clocked from the low-frequency crystal or from external clock ONLY IF the calibrated internal RC oscillator or the internal watchdog oscillator is used to drive the system clock. After setting AS0, if the switching is available, AS0 remains to 1, else it is forced to 0.

- **Bit 4 – TCN0UB: Timer/Counter0 Update Busy**

When timer/counter0 operates asynchronously and TCNT0 is written, this bit becomes set. When TCNT0 has been updated from the temporary storage register, this bit is cleared by hardware. A logical zero in this bit indicates that TCNT0 is ready to be updated with a new value.

12.2.1 Registers

The timer/counter (TCNT1), output compare registers (OCR1A/B), and input capture register (ICR1) are all 16-bit registers. Special procedures must be followed when accessing the 16-bit registers. These procedures are described in the section Section 12.3 “Accessing 16-bit Registers” on page 105. The timer/counter control registers (TCCR1A/B) are 8-bit registers and have no CPU access restrictions. Interrupt requests (abbreviated to Int.Req. in the figure) signals are all visible in the timer interrupt flag register (TIFR1). All interrupts are individually masked with the timer interrupt mask register (TIMSK1). TIFR1 and TIMSK1 are not shown in the figure.

The timer/counter can be clocked internally, via the prescaler, or by an external clock source on the Tn pin. The clock select logic block controls which clock source and edge the timer/counter uses to increment (or decrement) its value. The timer/counter is inactive when no clock source is selected. The output from the clock select logic is referred to as the timer clock (clk_{Tn}).

The double buffered output compare registers (OCR1A/B) are compared with the timer/counter value at all time. The result of the compare can be used by the waveform generator to generate a PWM or variable frequency output on the output compare pins, see Section 12.7 “Output Compare Units” on page 111. The compare match event will also set the compare match flag (OCF1A/B) which can be used to generate an output compare interrupt request.

The input capture register can capture the timer/counter value at a given external (edge triggered) event on either the input capture pin (ICP1) or on the analog comparator pins (see Section 18. “AnaComp - Analog Comparator” on page 194). The input capture unit includes a digital filtering unit (noise canceler) for reducing the chance of capturing noise spikes.

The TOP value, or maximum timer/counter value, can in some modes of operation be defined by either the OCR1A register, the ICR1 register, or by a set of fixed values. When using OCR1A as TOP value in a PWM mode, the OCR1A register can not be used for generating a PWM output. However, the TOP value will in this case be double buffered allowing the TOP value to be changed in run time. If a fixed TOP value is required, the ICR1 register can be used as an alternative, freeing the OCR1A to be used as PWM output.

12.2.2 Definitions

The following definitions are used extensively throughout the section:

- BOTTOM: The counter reaches the BOTTOM when it becomes 0x0000.
- MAX: The counter reaches its MAXimum when it becomes 0xFFFF (decimal 65,535).
- TOP: The counter reaches the TOP when it becomes equal to the highest value in the count sequence. The TOP value can be assigned to be one of the fixed values: 0x00FF, 0x01FF, or 0x03FF, or to the value stored in the OCR1A or ICR1 register. The assignment is dependent of the mode of operation.

12.3 Accessing 16-bit Registers

The TCNT1, OCR1A/B, and ICR1 are 16-bit registers that can be accessed by the AVR® CPU via the 8-bit data bus. The 16-bit register must be byte accessed using two read or write operations. Each 16-bit timer has a single 8-bit register for temporary storing of the high byte of the 16-bit access. The same temporary register is shared between all 16-bit registers within each 16-bit timer. Accessing the low byte triggers the 16-bit read or write operation. When the low byte of a 16-bit register is written by the CPU, the high byte stored in the temporary register, and the low byte written are both copied into the 16-bit register in the same clock cycle. When the low byte of a 16-bit 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.

Not all 16-bit accesses uses the temporary register for the high byte. Reading the OCR1A/B 16-bit registers does not involve using the temporary register.

To do a 16-bit write, the high byte must be written before the low byte. For a 16-bit read, the low byte must be read before the high byte.

12.11.2 Timer/Counter1 Control Register B – TCCR1B

Bit	7	6	5	4	3	2	1	0	
	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – ICNC1: Input Capture Noise Canceler**

Setting this bit (to one) activates the input capture noise canceler. When the noise canceler is activated, the input from the input capture pin (ICP1) is filtered. The filter function requires four successive equal valued samples of the ICP1 pin for changing its output. The input capture is therefore delayed by four oscillator cycles when the noise canceler is enabled.

- **Bit 6 – ICES1: Input Capture Edge Select**

This bit selects which edge on the input capture pin (ICP1) that is used to trigger a capture event. When the ICES1 bit is written to zero, a falling (negative) edge is used as trigger, and when the ICES1 bit is written to one, a rising (positive) edge will trigger the capture.

When a capture is triggered according to the ICES1 setting, the counter value is copied into the input capture register (ICR1). The event will also set the input capture flag (ICF1), and this can be used to cause an Input capture interrupt, if this interrupt is enabled.

When the ICR1 is used as TOP value (see description of the WGM13:0 bits located in the TCCR1A and the TCCR1B register), the ICP1 is disconnected and consequently the input capture function is disabled.

- **Bit 5 – Reserved Bit**

This bit is reserved for future use. For ensuring compatibility with future devices, this bit must be written to zero when TCCR1B is written.

- **Bit 4:3 – WGM13:2: Waveform Generation Mode**

See TCCR1A register description.

- **Bit 2:0 – CS12:0: Clock Select**

The three Clock Select bits select the clock source to be used by the Timer/Counter, see Figure 12-11 on page 122 and Figure 12-12 on page 122.

Table 12-5. Clock Select Bit Description

CS12	CS11	CS10	Description
0	0	0	No clock source (timer/counter stopped).
0	0	1	$clk_{I/O}$/1 (no prescaling)
0	1	0	$clk_{I/O}$/8 (from prescaler)
0	1	1	$clk_{I/O}$/64 (from prescaler)
1	0	0	$clk_{I/O}$/256 (from prescaler)
1	0	1	$clk_{I/O}$/1024 (from prescaler)
1	1	0	External clock source on T1 pin. Clock on falling edge.
1	1	1	External clock source on T1 pin. Clock on rising edge.

If external pin modes are used for the timer/counter1, transitions on the T1 pin will clock the counter even if the pin is configured as an output. This feature allows software control of the counting.

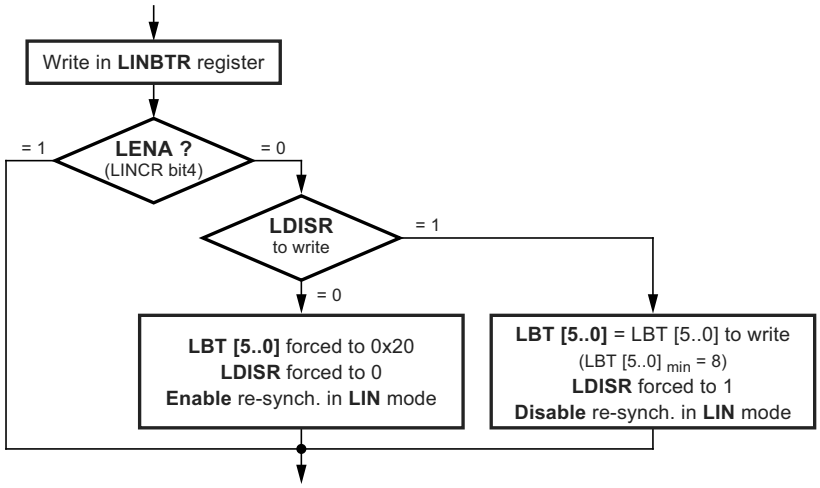
15.5.6.3 Handling LBT[5..0]

LDISR bit of LINBTR register is used to:

- Disable the re-synchronization (for instance in the case of LIN MASTER node),
- To enable the setting of LBT[5..0] (to manually adjust the baud rate especially in the case of UART mode). A minimum of 8 is required for LBT[5..0] due to the sampling operation.

Note that the LENA bit of LINCR register is important for this handling (see Figure 15-8).

Figure 15-8. Handling LBT[5..0]



15.5.7 Data Length

Section 15.4.6 “LIN Commands” on page 154 describes how to set or how are automatically set the LRXDL[3..0] or LTXDL[3..0] fields of LINDLR register before receiving or transmitting a response.

In the case of Tx response the LRXDL[3..0] will be used by the hardware to count the number of bytes already successfully sent.

In the case of Rx response the LTXDL[3..0] will be used by the hardware to count the number of bytes already successfully received.

If an error occurs, this information is useful to the programmer to recover the LIN messages.

15.5.7.1 Data Length in LIN 2.1

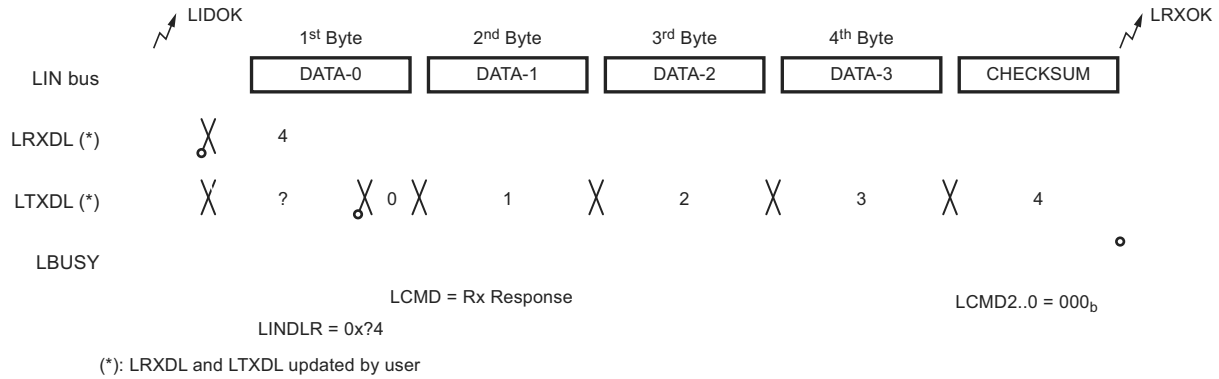
- If LTXDL[3..0]=0 only the CHECKSUM will be sent,
- If LRXDL[3..0]=0 the first byte received will be interpreted as the CHECKSUM,
- If LTXDL[3..0] or LRXDL[3..0] >8, values will be forced to 8 after the command setting and before sending or receiving of the first byte.

15.5.7.2 Data Length in LIN 1.3

- LRXDL and LTXDL fields are both hardware updated before setting LIDOK by decoding the data length code contained in the received PROTECTED IDENTIFIER (LRXDL = LTXDL).
- Via the above mechanism, a length of 0 or >8 is not possible.

15.5.7.3 Data Length in Rx Response

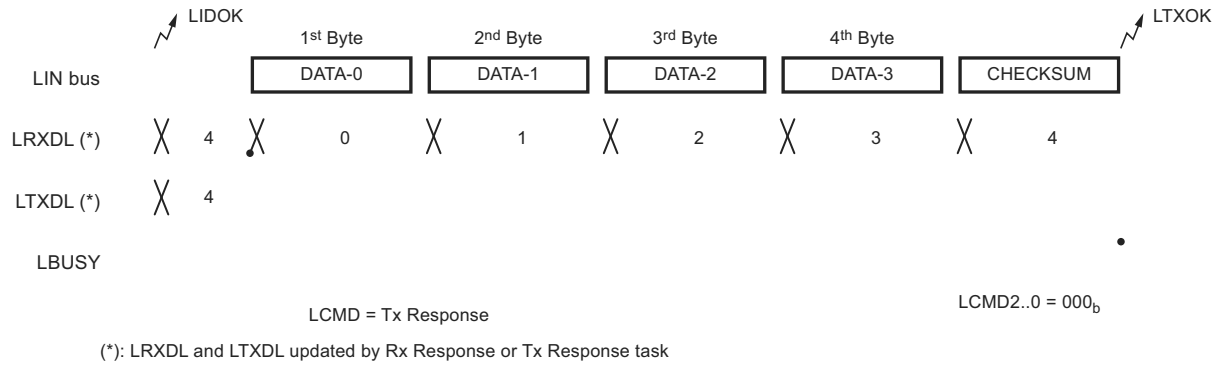
Figure 15-9. LIN2.1 - Rx Response - No error



- The user initializes LRXDL field before setting the Rx response command,
- After setting the Rx response command, LTXDL is reset by hardware,
- LRXDL field will remain unchanged during Rx (during busy signal),
- LTXDL field will count the number of received bytes (during busy signal),
- If an error occurs, Rx stops, the corresponding error flag is set and LTXDL will give the number of received bytes without error,
- If no error occurs, LRXOK is set after the reception of the CHECKSUM, LRXDL will be unchanged (and LTXDL = LRXDL).

15.5.7.4 Data Length in Tx Response

Figure 15-10. LIN1.3 - Tx Response - No Error



- The user initializes LTXDL field before setting the Tx response command,
- After setting the Tx response command, LRXDL is reset by hardware,
- LTXDL will remain unchanged during Tx (during busy signal),
- LRXDL will count the number of transmitted bytes (during busy signal),
- If an error occurs, Tx stops, the corresponding error flag is set and LRXDL will give the number of transmitted bytes without error,
- If no error occurs, LTXOK is set after the transmission of the CHECKSUM, LTXDL will be unchanged (and LRXDL = LTXDL).

15.6.1 LIN Control Register - LINCR

Bit	7	6	5	4	3	2	1	0	
	LSWRES	LIN13	LCONF1	LCONF0	LENA	LCMD2	LCMD1	LCMD0	LINCR
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	0	

- **Bit 7 - LSWRES: Software Reset**
  - 0 = No action,
  - 1 = Software reset (this bit is self-reset at the end of the reset procedure).

- **Bit 6 - LIN13: LIN 1.3 mode**
  - 0 = LIN 2.1 (default),
  - 1 = LIN 1.3.

- **Bit 5:4 - LCONF[1:0]: Configuration**

  a. LIN mode (default = 00):
  - 00 = LIN standard configuration (listen mode “off”, CRC “on” and frame_time_out “on”,
  - 01 = no CRC, no frame_time_out (listen mode “off”),
  - 10 = no frame_time_out (listen mode “off” and CRC “on”),
  - 11 = listening mode (CRC “on” and frame_time_out “on”).

  b. UART mode (default = 00):
  - 00 = 8-bit, no parity (listen mode “off”),
  - 01 = 8-bit, even parity (listen mode “off”),
  - 10 = 8-bit, odd parity (listen mode “off”),
  - 11 = listening mode, 8-bit, no parity.

- **Bit 3 - LENA: Enable**
  - 0 = disable (both LIN and UART modes),
  - 1 = enable (both LIN and UART modes).

- **Bit 2:0 - LCMD[2..0]: Command and mode**

The command is only available if LENA is set.
  - 000 = LIN Rx header - LIN abort,
  - 001 = LIN Tx header,
  - 010 = LIN Rx response,
  - 011 = LIN Tx response,
  - 100 = UART Rx and Tx byte disable,
  - 11x = UART Rx byte enable,
  - 1x1 = UART Tx byte enable.

20.2.2 EEPROM Write Prevents Writing to SPMCSR

Note that an EEPROM write operation will block all software programming to flash. Reading the fuses and lock bits from software will also be prevented during the EEPROM write operation. It is recommended that the user checks the status bit (EEPE) in the EECR register and verifies that the bit is cleared before writing to the SPMCSR register.

20.2.3 Reading the Fuse and Lock Bits from Software

It is possible to read both the fuse and lock bits from software. To read the lock bits, load the Z-pointer with 0x0001 and set the RFLB and SPMEN bits in SPMCSR. When an LPM instruction is executed within three CPU cycles after the RFLB and SPMEN bits are set in SPMCSR, the value of the lock bits will be loaded in the destination register. The RFLB and SPMEN bits will auto-clear upon completion of reading the lock bits or if no LPM instruction is executed within three CPU cycles or no SPM instruction is executed within four CPU cycles. When RFLB and SPMEN are cleared, LPM will work as described in the instruction set manual.

Bit	7	6	5	4	3	2	1	0
Rd (Z=0x0001)	**–**	**–**	**–**	**–**	**–**	**–**	**LB2**	**LB1**

The algorithm for reading the fuse low byte is similar to the one described above for reading the lock bits. To read the fuse low byte, load the Z-pointer with 0x0000 and set the RFLB and SPMEN bits in SPMCSR. When an LPM instruction is executed within three cycles after the RFLB and SPMEN bits are set in the SPMCSR, the value of the fuse low byte (FLB) will be loaded in the destination register as shown below. See Table 21-5 on page 209 for a detailed description and mapping of the fuse low byte.

Bit	7	6	5	4	3	2	1	0
Rd (Z=0x0000)	**FLB7**	**FLB6**	**FLB5**	**FLB4**	**FLB3**	**FLB2**	**FLB1**	**FLB0**

Similarly, when reading the fuse high byte (FHB), load 0x0003 in the Z-pointer. When an LPM instruction is executed within three cycles after the RFLB and SPMEN bits are set in the SPMCSR, the value of the fuse high byte will be loaded in the destination register as shown below. See Table 21-4 on page 208 for detailed description and mapping of the fuse high byte.

Bit	7	6	5	4	3	2	1	0
Rd (Z=0x0003)	**FHB7**	**FHB6**	**FHB5**	**FHB4**	**FHB3**	**FHB2**	**FHB1**	**FHB0**

Similarly, when reading the extended fuse byte (EFB), load 0x0002 in the Z-pointer. When an LPM instruction is executed within three cycles after the RFLB and SPMEN bits are set in the SPMCSR, the value of the extended fuse byte will be loaded in the destination register as shown below. See Table 21-3 on page 208 for detailed description and mapping of the extended fuse byte.

Bit	7	6	5	4	3	2	1	0
Rd (Z=0x0002)	**–**	**–**	**–**	**–**	**–**	**–**	**–**	**EFB0**

Fuse and lock bits that are programmed, will be read as zero. Fuse and lock bits that are unprogrammed, will be read as one.

C. Load Data Low Byte

1. Set XA1, XA0 to “0,1”. This enables data loading.
2. Set DATA = data low byte (0x00 - 0xFF).
3. Give XTAL1 a positive pulse. This loads the data byte.

D. Load Data High Byte

1. Set BS1 to “1”. This selects high data byte.
2. Set XA1, XA0 to “0,1”. This enables data loading.
3. Set DATA = data high byte (0x00 - 0xFF).
4. Give XTAL1 a positive pulse. This loads the data byte.

E. Latch Data

1. Set BS1 to “1”. This selects high data byte.
2. Give PAGEL a positive pulse. This latches the data bytes. (See Figure 21-3 on page 214 for signal waveforms)

F. Repeat B through E until the entire buffer is filled or until all data within the page is loaded.

While the lower bits in the address are mapped to words within the page, the higher bits address the pages within the FLASH. This is illustrated in Figure 21-2. Note that if less than eight bits are required to address words in the page (pagesize < 256), the most significant bit(s) in the address low byte are used to address the page when performing a page write.

G. Load Address High byte

1. Set XA1, XA0 to “0,0”. This enables address loading.
2. Set BS1 to “1”. This selects high address.
3. Set DATA = address high byte (0x00 - 0xFF).
4. Give XTAL1 a positive pulse. This loads the address high byte.

H. Program Page

1. Give $\overline{WR}$ a negative pulse. This starts programming of the entire page of data. RDY/$\overline{BSY}$ goes low.
2. Wait until RDY/$\overline{BSY}$ goes high (See Figure 21-3 on page 214 for signal waveforms).

I. Repeat B through H until the entire flash is programmed or until all data has been programmed.

J. End Page Programming

1. 1. Set XA1, XA0 to “1,0”. This enables command loading.
2. Set DATA to “0000 0000$_b$”. This is the command for no operation.
3. Give XTAL1 a positive pulse. This loads the command, and the internal write signals are reset.

Figure 21-2. Addressing the Flash Which is Organized in Pages