Universidad de Costa Rica
Facultad de Ingeniería
Escuela de Ingeniería Eléctrica
IE-0624: Laboratorio de Microcontroladores

II ciclo 2024

Laboratorio # 2
GPIOs, Timers y FSM

Christabel Alvarado Anchía - B80286
Erick Marín Rojas - B94544
Profesor: MSc. Marco Villalta Fallas

18 de setiembre de 2024

Índice

Índice de figuras

Índice de tablas

1. Resumen

En este laboratorio se tiene como objetivo principal familiarizarse con el funcionamiento del ATtiny 4313. Para lograr esto se desarrolló un juego de Simón dice en el que se cuenta con 4 leds de distintos colores y 4 botones correspondientes, el sistema ilumina los leds en cierto orden y el usuario debe presionar los botones correspondientes siguiendo el mismo patrón. Además, el juego emite sonidos al iluminar los leds y mientras no se encuentre en una partida presionar cualquier botón permite iniciar la partida. El diseño e implementación de este juego permitió a los estudiantes comprender de mejor manera el uso de GPIOs, interrupciones, timers y FSM. En este informe se resumen los aspectos más importantes del ATtiny 4313; luego, se explica y analiza tanto el hardware como el software diseñados para simular el juego Simon dice.

Repositorio de GitHub `https://github.com/ErickMaRi/Laboratorios-Grupales-IE0624`

2. Nota teórica

2.1. ATtiny4313

El microcontrolador de 8 bits, ATtiny4313, pertenece a la familia AVR (basados en arquitectura RISC). Según la hoja del fabricante [1] se considera un microcontrolador de alto rendimiento y bajo consumo, con 120 instrucciones. Además, cuenta con una memoria flash de 4k bytes, una SRAM de 256 bytes y una EEPROM de 256 bytes. Respecto a pines de entrada y salida cuenta con 18 GPIOs, 1 USART full duplex, USI, cuatro canales PWM, un timer de 8 bits y otro de 16 bits.

2.1.1. Diagrama de pines

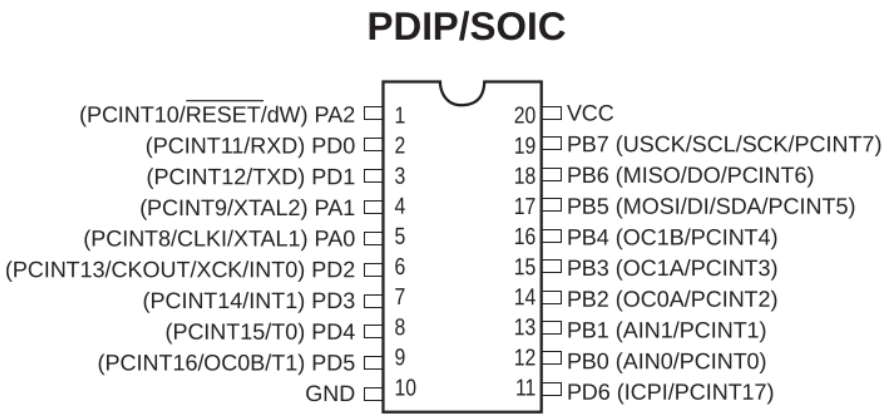


Figura 1: Diagrama de pines de ATtiny4313 [1]

2.1.2. Diagrama de bloques

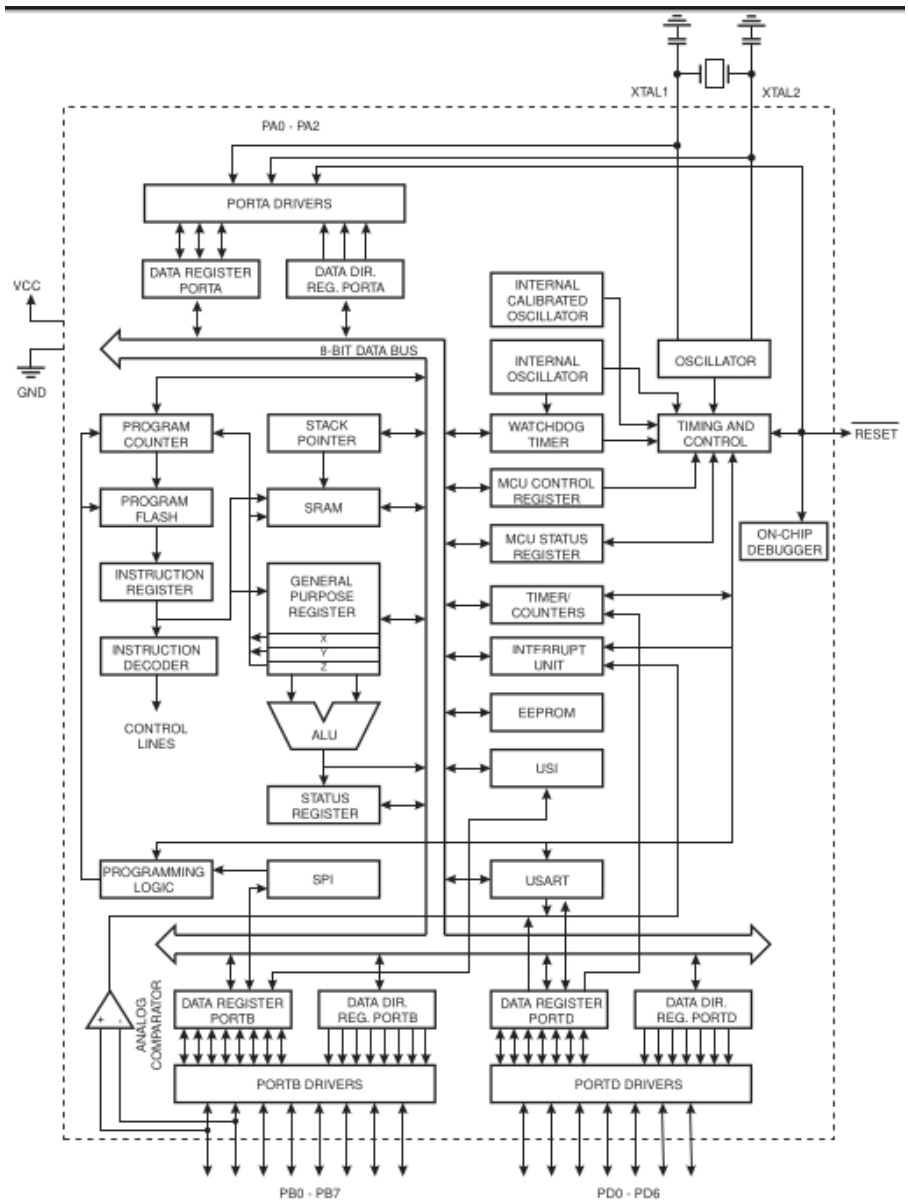


Figura 2: Diagrama de bloques de ATtiny4313 [1]

2.1.3. Registros relevantes

Para el desarrollo de este laboratorio resulta necesario conocer detalles sobre varios registros del ATtiny4313, entre estos destacan:

- **DDRB:** Se utiliza para definir los pines del puerto B como salidas o entradas. Es un registro de 8 bits donde 1 corresponde a salida y 0 a entrada. DDRA y DDRD se comportan de manera similar.

Bit	7	6	5	4	3	2	1	0	
0x17 (0x37)	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
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	

Figura 3: Descripción de DDRB [1]

- **PORTB:** Permite encender y apagar los pines del puerto B, contiene resistencias de pull up internas. PORTA y PORTD se comportan de manera similar.

Bit	7	6	5	4	3	2	1	0	
0x18 (0x38)	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
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	

Figura 4: Descripción de PORTB [1]

- **GIMSK:** Este registro es utilizado para habilitar o deshabilitar distintas interrupciones.

Bit	7	6	5	4	3	2	1	0	
0x3B (0x5B)	INT1	INT0	PCIE0	PCIE2	PCIE1	–	–	–	GIMSK
Read/Write	R/W	R/W	R/W	R/W	R/W	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

Figura 5: Descripción de GIMSK [1]

- **PCMSK1:** Permite habilitar o deshabilitar las interrupciones por cambio de estado en los pines de PORTD. Tiene un comportamiento similar a PCMSK2

Bit	7	6	5	4	3	2	1	0	
0x04 (0x24)	–	–	–	–	–	PCINT10	PCINT9	PCINT8	PCMSK1
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figura 6: Descripción de PCMSK1 [1]

- **MCUCR:** Permite habilitar los modos de interpretación de interrupciones.

Bit	7	6	5	4	3	2	1	0	
0x35 (0x55)	PUD	SM1	SE	SM0	ISC11	ISC10	ISC01	ISC00	MCUCR
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	

Figura 7: Descripción de MCUCR [1]

- **TIMSK:** Permite seleccionar el tipo de interrupción generada por el contador 0 (overflow).

Bit	7	6	5	4	3	2	1	0	
0x39 (0x59)	TOIE1	OCIE1A	OCIE1B	–	ICIE1	OCIE0B	TOIE0	OCIE0A	TIMSK
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figura 8: Descripción de TIMSK [1]

- **TCNT0:** Permite acceder, en modo escritura o lectura, al contador de 8 bits del timer.

Bit	7	6	5	4	3	2	1	0	
0x32 (0x52)	TCNT0[7:0]								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	

Figura 9: Descripción de TCNT0 [1]

- **TCCR0B:** Permite configurar la fuente de reloj del registro B. TCCR0A tiene un comportamiento similar.

Bit	7	6	5	4	3	2	1	0	
0x33 (0x53)	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	TCCR0B
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figura 10: Descripción de TCCR0B [1]

2.2. Conceptos importantes para el diseño del circuito

2.2.1. FSM: Maquina de estados finita

Una maquina de estados consiste en un algoritmo que indica una serie de instrucciones a ser ejecutadas según el estado que se ejecute. Cada estado representa un comportamiento específico del sistema, y las transiciones entre distintos estados se activan por eventos o entradas específicas. En C las FSM pueden ser implementadas usando if-else o switch.
Las maquinas de estado finitas se pueden dividir en deterministas y no deterministas. Las deterministas se caracteriza porque desde cualquier estado, solo hay una transición para cualquier entrada permitida. [2] Mientras que las no deterministas varían en que dada una entrada desde un estado particular, esta puede conducir a más de un estado diferente. [2]

2.2.2. Resistencias limitadoras de corriente

En el diseño de circuitos es común conectar una resistencia en serie con los LEDs a utilizar para proteger al mismo de posibles excesos de corriente. Para calcular el valor adecuado de estos resistores se utiliza la siguiente ecuación:

$$R = \frac{V_{DD} - V_{LED}}{I_{max}} \tag{1}$$

Para efectos de este laboratorio ese sabe que el voltaje del ATtiny4313 es 5V , la corriente forward, según el simulador, es de 20mA y el voltaje forward es 2V. Con esto se obtiene que la resistencia debe ser de 150Ω; sin embargo, se decidió utilizar resistencias mayores para evitar errores, considerando que se trabajó con múltiples LED.

2.2.3. Filtro paso bajo

Los filtros paso bajo permiten limitar las señales que lo atraviesan, de manera que solamente pasan las frecuencias bajas. En el procesamiento de audio utilizar filtros de paso bajo trae como beneficio un sonido más suave o dulce. [3]. Además, ayuda a eliminar picos y ruido en la señal y a evitar sobrepaso de corriente.

2.3. Componentes

Para el diseño del circuito se deben utilizar los siguientes componentes, con un precio total de ₡3251. Para encontrar los precios se utilizaron sitios de venta de componentes electrónicos dentro de Costa Rica, como MicroJPM y Mouser.

Componente	Cantidad	Precio por unidad
ATtiny4313	1	₡833
Resistor 220Ω	4	₡26
Resistor 18$k\Omega$	1	₡26
Capacitor 47nF	1	₡52
LED rojo	1	₡259
LED azul	1	₡259
LED amarillo	1	₡259
LED verde	1	₡259
Botón	4	₡105
Buzzer	1	₡780
Amplificador	1	₡494

Tabla 1: Componentes utilizados

3. Desarrollo

Se detalla el procedimiento llevado a cabo para la implementar el juego de Simon en el microcontrolador ATtiny4313. Se explica tanto la configuración de hardware como la implementación del código en C.

3.1. Configuración de Hardware

El hardware se configuró de la siguiente manera:

- **LEDs:** Se conectaron cuatro LEDs (rojo, azul, amarillo y verde) a los pines PB0, PB1, PB2 y PB3 respectivamente, utilizando resistencias de 220Ω para limitar la corriente.
- **Botones:** Se utilizaron cuatro botones conectados a los pines PA1, PD1, PD2 y PD3. Estos pines se configuraron como entradas con resistencias pull-up internas activadas, para detectar la pulsación de los botones, que referirían el pin a la tierra.
- **Buzzer:** Un buzzer se conectó al pin PB4, permitiendo la generación de sonidos mediante modulación de frecuencia.
- **Alimentación:** El microcontrolador se alimenta con 5V por defecto en el ambiente de SimulIDE.

Las conexiones se pueden apreciar en la figura 11.

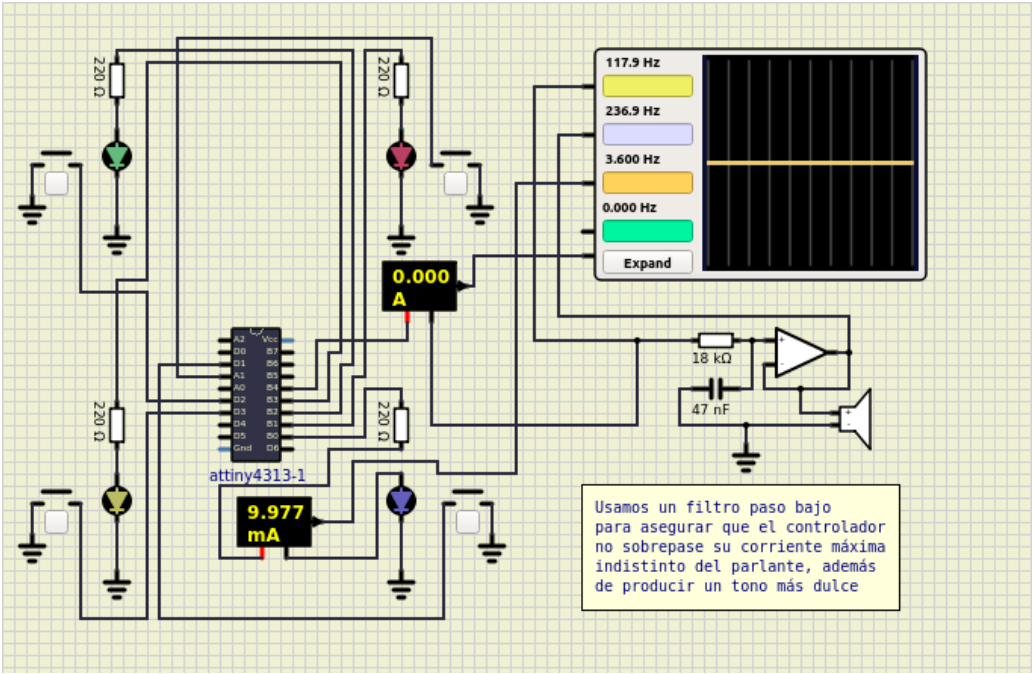


Figura 11: Podemos observar los LEDS de los distintos colores encendidos al inicio del juego, cada uno usando casi 10 mA de los 60 disponibles.

3.2. Implementación del Código

El código fue desarrollado en C, utilizando las librerías estándar del AVR. Se detallan las partes más importantes del código y su función dentro del programa. Se encienden los LEDs mientras se produce una nota, para las entradas y salidas.

3.2.1. Variables Globales y Definiciones

Se definieron constantes para los estados de la máquina de estados finitos (FSM) y para las frecuencias de las notas musicales utilizadas:

```
#define ESPERA 0
#define INICIO 1
#define MOSTRAR 2
#define REVISAR 3
#define FIN 4

#define NOTA1 392 // G4
#define NOTA2 440 // A4
#define NOTA3 493 // B4
#define NOTA4 523 // C5
```

Se eligieron las notas sin partir de un acorde, arpegio o arreglo de frecuencias en específico, simplemente se seleccionaron cuatro entre diez iniciales, que sonaban bien juntas, debido a la baja fidelidad del atraso en microsegundos.

Las variables globales principales son:

- **estado:** Indica el estado actual.
- **entrada_usuario:** Almacena el botón que el usuario presionó.
- **secuencia[13]:** Almacena la secuencia de LEDs que el usuario debe repetir.
- **turno:** Lleva el conteo del nivel actual en el juego para controlar la dificultad.
- **indice_secuencia:** Utilizado para comparar la entrada del usuario con la secuencia.
- **enable y overflow_cont:** Utilizados para el control de retardos con el temporizador.

3.2.2. Configuración de Puertos y Registros

En el método `main()`, se realiza la configuración de los puertos y registros necesarios:

- **Configuración de salidas:** Se configuran los pines PB0 a PB3 y PB4 como salidas para controlar los LEDs y el buzzer.
- **Configuración de entradas:** Los pines PA1, PD1, PD2 y PD3 se configuran como entradas con resistencias pull-up activadas.
- **Configuración de interrupciones:** Se habilitan las interrupciones por cambio de pin (*Pin Change Interrupts*) en los puertos A y D, permitiendo detectar cuando se presiona un botón.
- **Configuración del Timer0:** Se configura el Timer0 en modo normal con un prescaler de 64, y se habilita la interrupción por desbordamiento para contar los retardos.

Los registros utilizados para estas configuraciones son:

- **DDRB, DDRA, DDRD:** Configuran los pines como entradas o salidas.

- **PORTA, PORTD:** Activan las resistencias pull-up internas.
- **GIMSK, PCMSK1, PCMSK2:** Configuran las interrupciones por cambio de pin.
- **MCUCR:** Configura las interrupciones externas.
- **TIMSK, TCNT0, TCCR0A, TCCR0B:** Configuran el Timer0 y sus interrupciones.

3.2.3. Funciones Principales

- `parpadear(int n)`: Hace que todos los LEDs parpadeen `n` veces, utilizado para indicar el inicio o fin del juego.
- `encenderLed(int n)`: Enciende el LED correspondiente al número `n` y reproduce una nota asociada. Utiliza la función `tocarNota()` para generar el sonido.
- `delay(int ms)`: Implementa un retardo de `ms` milisegundos utilizando el Timer0 y las interrupciones por desbordamiento.
- `delay_us(unsigned int us)`: Implementa un retardo en la escala de los microsegundos declarándo tres enteros volátiles.
- `tocarNota(int frecuencia)`: Genera un tono en el buzzer a la frecuencia especificada, controlando el ciclo de encendido y apagado del pin PB4. Utiliza `delay_us(unsigned int us)` para controlar la frecuencia.
- `tonadaInicio()` **y** `tonadaError()`: Reproducen melodías específicas al inicio del juego y cuando ocurre un error, respectivamente, usan `tocarNota(int frecuencia)` para producir las tonadas.
- `imprimirSecuencia()`: Muestra la secuencia de LEDs que el usuario debe memorizar, ajustando la duración según el nivel actual.
- `iniciarsSecuencia()`: Genera una secuencia aleatoria de números entre 0 y 3 utilizando una semilla fija.
- `FSM()`: Implementa la máquina de estados finitos que controla la lógica del juego, gestionando los estados ESPERA, INICIO, MOSTRAR, REVISAR y FIN.

3.2.4. Interrupciones

Se implementaron las siguientes rutinas de servicio de interrupción (ISR):

- `ISR(TIMER0_OVF_vect)`: Maneja el desbordamiento del Timer0, incrementando el contador de overflows para controlar los retardos en milisegundos.
- `ISR(PCINT1_vect)` **y** `ISR(PCINT2_vect)`: Detectan cambios en los pines de entrada (botones presionados) y actualizan la variable `entrada_usuario` con el botón presionado, además de reproducir la nota correspondiente.

3.2.5. Máquina de Estados Finita (FSM)

La FSM se implementa en la función `FSM()`, y controla el flujo del juego mediante los siguientes estados:

1. **ESPERA:** El juego espera a que el usuario presione cualquier botón para iniciar.
2. **INICIO:** Se reproduce la tonada de inicio y los LEDs parpadean dos veces. Se inicializa la secuencia aleatoria y se reinicia el contador de turnos.
3. **MOSTRAR:** Se muestra la secuencia de LEDs al usuario. La velocidad de presentación disminuye ligeramente con cada turno para aumentar la dificultad.
4. **REVISAR:** El juego espera la entrada del usuario y compara cada botón presionado con la secuencia. Si el usuario acierta, continúa; si falla, cambia al estado FIN.
5. **FIN:** Se reproduce la tonada de error y los LEDs parpadean tres veces. El juego regresa al estado ESPERA.

3.2.6. Procedimiento de Juego

El flujo del juego es el siguiente:

1. **Inicio en ESPERA:** El programa inicia y espera que el usuario presione un botón.
2. **Transición a INICIO:** Al detectar una entrada, el juego comienza. Se reproducen señales visuales y sonoras para indicar el inicio.
3. **Generamos y Mostramos Respuesta:** Se genera una secuencia aleatoria de LEDs y se muestra al usuario, incrementando en longitud en cada turno.
4. **Revisamos la Entrada del Usuario:** El usuario intenta repetir la secuencia presionando los botones correspondientes. El programa verifica la exactitud de cada entrada en tiempo real.
5. **Progresión o Finalización:** Si el usuario completa correctamente la secuencia, el juego avanza al siguiente turno. Si el usuario falla, se indica el error y el juego termina, regresamos a ESPERA.

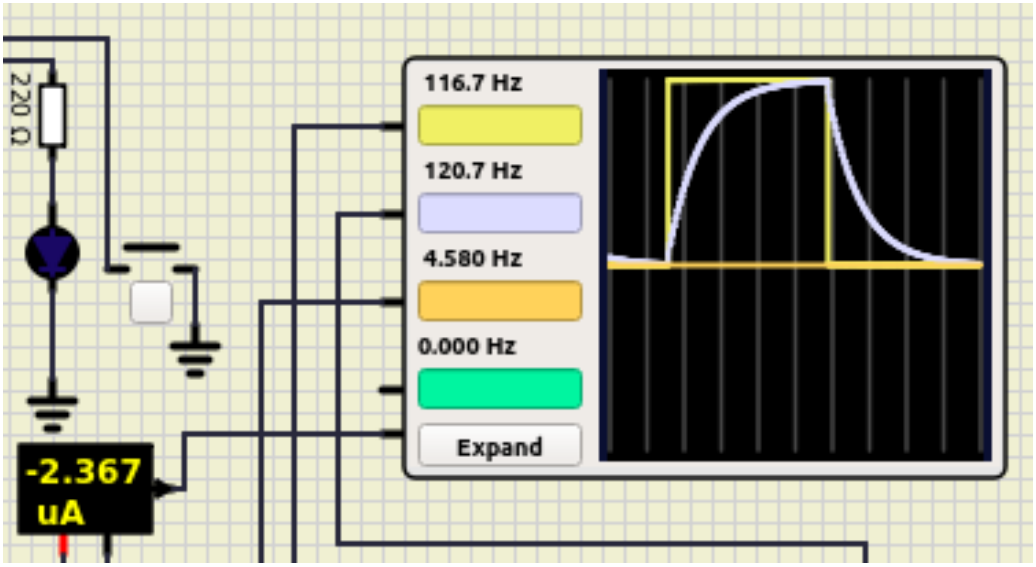


Figura 12: Se filtran los componentes de alta frecuencia de la señal cuadrada para producir un tono más dulce, se observa una corriente baja en el pin del controlador debido a la alta impedancia del circuito paso bajo usado. Aunque a corriente de entrada al filtro oscila bastante, lo hace por debajo de 1 mA.

3.3. Pruebas y Resultados

Se realizaron pruebas para verificar el correcto funcionamiento del juego. Usando el sistema diseñado en un ambiente simulado se comprobó que:

- El sistema espera a que el usuario presione un botón para iniciar el juego.
- Los LEDs y el buzzer responden correctamente a las señales del programa, siguiendo el tiempo que el enunciado solicita.
- Se muestra la secuencia generada al usario exitosamente.
- Se reciben las entradas del usuario a través de los botones.
- El juego incrementa la dificultad de forma progresiva, con 200 milisegundos menos por vez.
- La máquina de estados realiza las transiciones necesarias para poder jugar Simon.
- Para asegurar que el microcontrolador no sobrepase los límites de corriente indistinto del parlante a usar, se le separa con un op-amp del actuador, esto también se usa para producir una señal con un tono más dulce, como podemos observar en la figura 12.
- Se usa un buzzer para realimentad un sonido distinto por cada LED-botón, realimentando la idea de que están relacionados y haciendo el juguete más entretenido.

3.4. Consideraciones Adicionales

- **Generación de Números Aleatorios:** Se utilizó una semilla fija para `rand()`, por lo que la secuencia utilizada siempre es la misma, esto se hizo así para efectos de desarrollo pero para nivel de producción la semilla se debe leer de algún temporizador.

- **Temporización Imprecisa:** El `_delay_us()` aproxima un delay en la escala de los microsegundos instanciando enteros volátiles tres veces, por lo que no se debe fiar de la temporización para aplicaciones sensibles.
- **Limitación de la Corriente en los LEDs:** Para efectos de SimulIDE la corriente en los LEDs no cambia respecto al color por lo que no se dimensiona una resistencia distinta por color.

4. Conclusiones y recomendaciones

- Se logró simular de manera exitosa el juego simón dice según las especificaciones indicadas en el enunciado del laboratorio.
- Se implementó una máquina de estados finita de manera correcta, logrando representar el juego simón dice de manera clara.
- El uso de amperímetros y el osciloscopio permite observar el comportamiento correcto de las diversas señales del sistema.
- Es de suma importancia comprender el funcionamiento de los registros e interrupciones del microcontrolador para poder generar el código necesario de manera exitosa. Por esto se recomienda leer detalladamente la hoja del fabricante previo a iniciar el desarrollo del proyecto.
- Separando el controlador del resto del circuito logramos alcanzar corrientes en los pines bajas para la modulación de ancho de pulso que se usó en el buzzer.

Referencias

[1] A. Corporation, *ATtiny2313A/4313 Data Sheet*, 2011. [Online]. Available: https://ww1.microchip.com/downloads/en/DeviceDoc/doc8246.pdf

[2] M. Shead, "State machines: the ultimate guide to automata in computer science," 2018, accessed: 2024-09-18. [Online]. Available: https://www.freecodecamp.org/news/state-machines-basics-of-computer-science-d42855debc66/

[3] M. Alessi, "¿qué es un filtro paso bajo y cómo se utiliza en las mezclas?" 2024, accessed: 2024-09-18. [Online]. Available: https://emastered.com/es/blog/low-pass-filter

5. Anexos

Features

- High Performance, Low Power AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  - 120 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 20 MIPS Throughput at 20 MHz
- Data and Non-volatile Program and Data Memories
  - 2/4K Bytes of In-System Self Programmable Flash
    - Endurance 10,000 Write/Erase Cycles
  - 128/256 Bytes In-System Programmable EEPROM
    - Endurance: 100,000 Write/Erase Cycles
  - 128/256 Bytes Internal SRAM
  - Programming Lock for Flash Program and EEPROM Data Security
- Peripheral Features
  - One 8-bit Timer/Counter with Separate Prescaler and Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare and Capture Modes
  - Four PWM Channels
  - On-chip Analog Comparator
  - Programmable Watchdog Timer with On-chip Oscillator
  - USI – Universal Serial Interface
  - Full Duplex USART
- Special Microcontroller Features
  - debugWIRE On-chip Debugging
  - In-System Programmable via SPI Port
  - External and Internal Interrupt Sources
  - Low-power Idle, Power-down, and Standby Modes
  - Enhanced Power-on Reset Circuit
  - Programmable Brown-out Detection Circuit
  - Internal Calibrated Oscillator
- I/O and Packages
  - 18 Programmable I/O Lines
  - 20-pin PDIP, 20-pin SOIC, 20-pad MLF/VQFN
- Operating Voltage
  - 1.8 – 5.5V
- Speed Grades
  - 0 – 4 MHz @ 1.8 – 5.5V
  - 0 – 10 MHz @ 2.7 – 5.5V
  - 0 – 20 MHz @ 4.5 – 5.5V
- Industrial Temperature Range: -40°C to +85°C
- Low Power Consumption
  - Active Mode
    - 190 µA at 1.8V and 1MHz
  - Idle Mode
    - 24 µA at 1.8V and 1MHz
  - Power-down Mode
    - 0.1 µA at 1.8V and +25°C

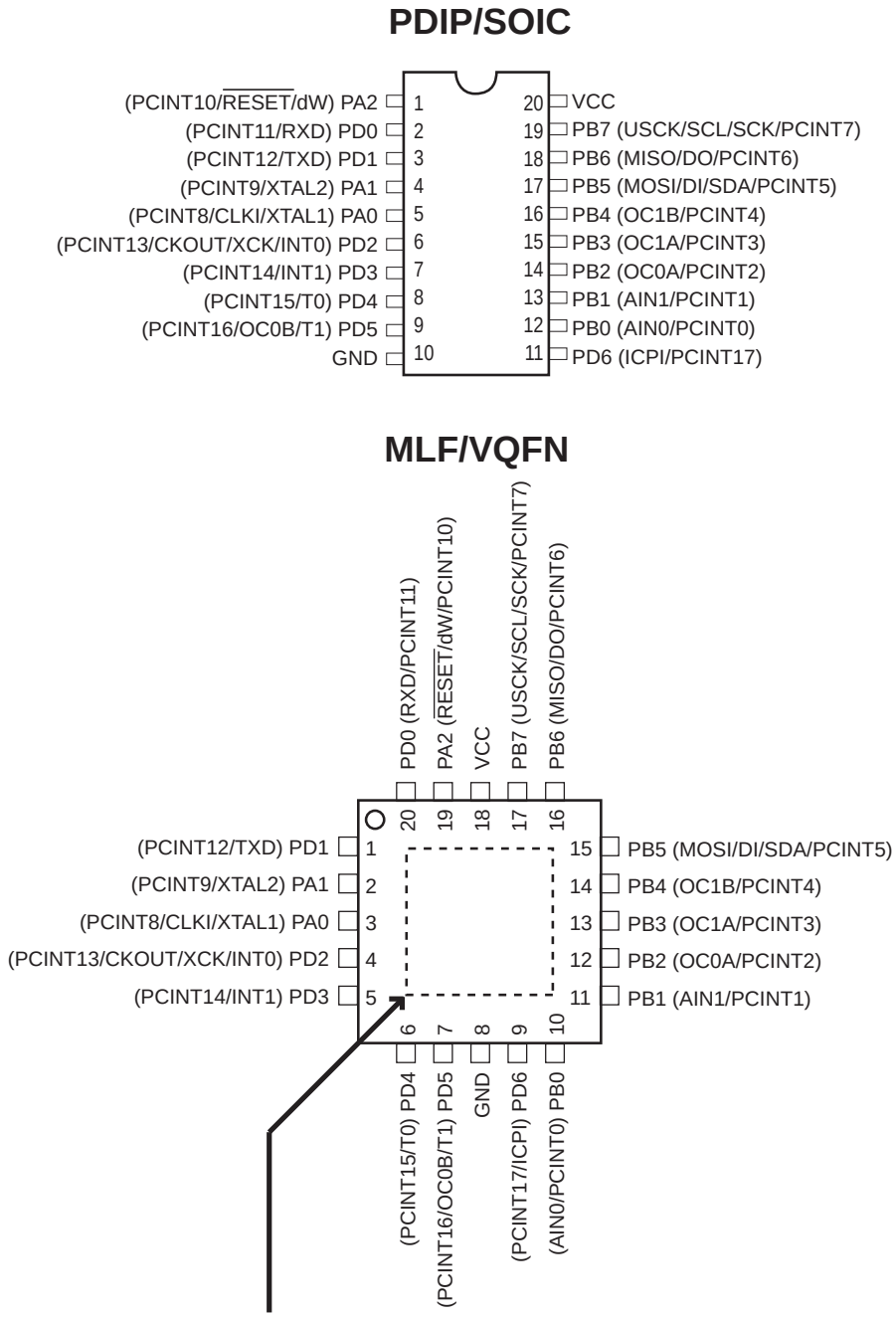
8-bit AVR® Microcontroller with 2/4K Bytes In-System Programmable Flash

ATtiny2313A
ATtiny4313

Rev. 8246B–AVR–09/11

1. Pin Configurations

Figure 1-1. Pinout ATtiny2313A/4313



NOTE: Bottom pad should be soldered to ground.

1.1 Pin Descriptions

1.1.1 VCC

Digital supply voltage.

1.1.2 GND

Ground.

1.1.3 Port A (PA2..PA0)

Port A is a 3-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, except PA2 which has the $\overline{\text{RESET}}$ capability. To use pin PA2 as I/O pin, instead of RESET pin, program (“0”) RSTDISBL fuse. 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 ATtiny2313A/4313 as listed on page 62.

1.1.4 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 ATtiny2313A/4313 as listed on page 63.

1.1.5 Port D (PD6..PD0)

Port D is a 7-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port D also serves the functions of various special features of the ATtiny2313A/4313 as listed on page 67.

1.1.6 $\overline{\text{RESET}}$

Reset input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running and provided that the reset pin has not been disabled. The minimum pulse length is given in Table 22-3 on page 201. Shorter pulses are not guaranteed to generate a reset. The Reset Input is an alternate function for PA2 and dW.

The reset pin can also be used as a (weak) I/O pin.

1.1.7 XTAL1

Input to the inverting Oscillator amplifier and input to the internal clock operating circuit. XTAL1 is an alternate function for PA0.

1.1.8 XTAL2

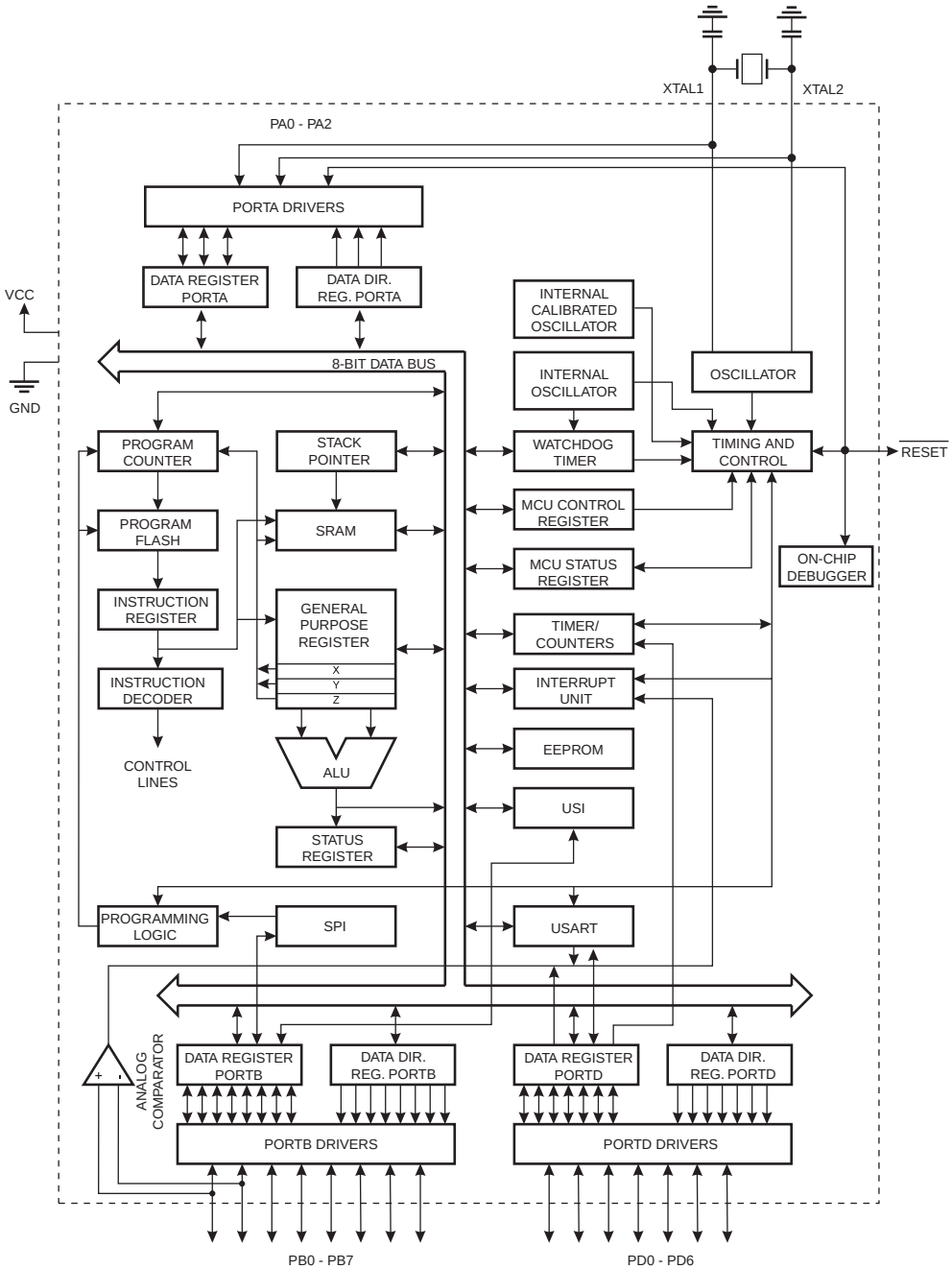
Output from the inverting Oscillator amplifier. XTAL2 is an alternate function for PA1.

2. Overview

The ATtiny2313A/4313 is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATtiny2313A/4313 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

2.1 Block Diagram

Figure 2-1. Block Diagram



The AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The ATtiny2313A/4313 provides the following features: 2/4K bytes of In-System Programmable Flash, 128/256 bytes EEPROM, 128/256 bytes SRAM, 18 general purpose I/O lines, 32 general purpose working registers, a single-wire Interface for On-chip Debugging, two flexible Timer/Counters with compare modes, internal and external interrupts, a serial programmable USART, Universal Serial Interface with Start Condition Detector, a programmable Watchdog Timer with internal Oscillator, and three software selectable power saving modes. The Idle mode stops the CPU while allowing the SRAM, Timer/Counters, and interrupt system to continue functioning. The Power-down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next interrupt or hardware reset. In Standby mode, the crystal/resonator Oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low-power consumption.

The device is manufactured using Atmel's high density non-volatile memory technology. The On-chip ISP Flash allows the program memory to be reprogrammed In-System through an SPI serial interface, or by a conventional non-volatile memory programmer. By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the Atmel ATtiny2313A/4313 is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications.

The ATtiny2313A/4313 AVR is supported with a full suite of program and system development tools including: C Compilers, Macro Assemblers, Program Debugger/Simulators, In-Circuit Emulators, and Evaluation kits.

2.2 Comparison Between ATtiny2313A and ATtiny4313

The ATtiny2313A and ATtiny4313 differ only in memory sizes. Table 2-1 summarizes the different memory sizes for the two devices.

Table 2-1. Memory Size Summary

Device	Flash	EEPROM	RAM
ATtiny2313A	2K Bytes	128 Bytes	128 Bytes
ATtiny4313	4K Bytes	256 Bytes	256 Bytes

3. About

3.1 Resources

A comprehensive set of drivers, application notes, data sheets and descriptions on development tools are available for download at http://www.atmel.com/avr.

3.2 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.

For I/O Registers located in the extended I/O map, “IN”, “OUT”, “SBIS”, “SBIC”, “CBI”, and “SBI” instructions must be replaced with instructions that allow access to extended I/O. Typically, this means “LDS” and “STS” combined with “SBRS”, “SBRC”, “SBR”, and “CBR”. Note that not all AVR devices include an extended I/O map.

3.3 Data Retention

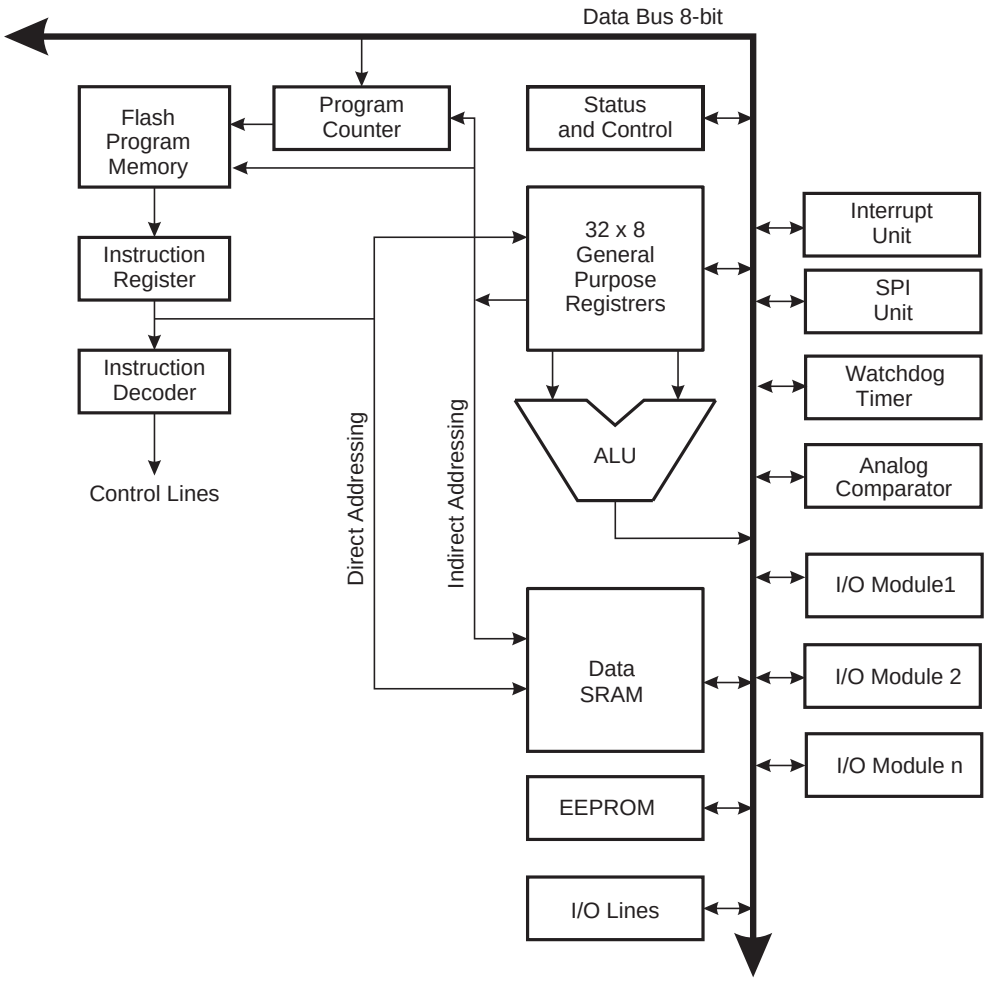
Reliability Qualification results show that the projected data retention failure rate is much less than 1 PPM over 20 years at 85°C or 100 years at 25°C.

4. CPU Core

This section discusses the AVR core architecture in general. The main function of the CPU core is to ensure correct program execution. The CPU must therefore be able to access memories, perform calculations, control peripherals, and handle interrupts.

4.1 Architectural Overview

Figure 4-1. Block Diagram of the AVR Architecture



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

The fast-access Register File contains 32 x 8-bit general purpose working registers with a single clock cycle access time. This allows single-cycle Arithmetic Logic Unit (ALU) operation. In a typical ALU operation, two operands are output from the Register File, the operation is executed, and the result is stored back in the Register File – in one clock cycle.

Six of the 32 registers can be used as three 16-bit indirect address register pointers for Data Space addressing – enabling efficient address calculations. One of the these address pointers can also be used as an address pointer for look up tables in Flash program memory. These added function registers are the 16-bit X-, Y-, and Z-register, described later in this section.

The ALU supports arithmetic and logic operations between registers or between a constant and a register. Single register operations can also be executed in the ALU. After an arithmetic operation, the Status Register is updated to reflect information about the result of the operation.

Program flow is provided by conditional and unconditional jump and call instructions, able to directly address the whole address space. Most AVR instructions have a single 16-bit word format. Every program memory address contains a 16- or 32-bit instruction.

During interrupts and subroutine calls, the return address Program Counter (PC) is stored on the Stack. The Stack is effectively allocated in the general data SRAM, and consequently the Stack size is only limited by the total SRAM size and the usage of the SRAM. All user programs must initialize the SP in the Reset routine (before subroutines or interrupts are executed). The Stack Pointer (SP) is read/write accessible in the I/O space. The data SRAM can easily be accessed through the five different addressing modes supported in the AVR architecture.

The memory spaces in the AVR architecture are all linear and regular memory maps.

A flexible interrupt module has its control registers in the I/O space with an additional Global Interrupt Enable bit in the Status Register. All interrupts have a separate Interrupt Vector in the Interrupt Vector table. The interrupts have priority in accordance with their Interrupt Vector position. The lower the Interrupt Vector address, the higher the priority.

The I/O memory space contains 64 addresses for CPU peripheral functions as Control Registers, and other I/O functions. The I/O Memory can be accessed directly, or as the Data Space locations following those of the Register File, 0x20 - 0x5F.

4.2 ALU – Arithmetic Logic Unit

The high-performance AVR ALU operates in direct connection with all the 32 general purpose working registers. Within a single clock cycle, arithmetic operations between general purpose registers or between a register and an immediate are executed. The ALU operations are divided into three main categories – arithmetic, logical, and bit-functions. Some implementations of the architecture also provide a powerful multiplier supporting both signed/unsigned multiplication and fractional format. See the “Instruction Set” section for a detailed description.

4.3 Status Register

The Status Register contains information about the result of the most recently executed arithmetic instruction. This information can be used for altering program flow in order to perform conditional operations. Note that the Status Register is updated after all ALU operations, as specified in the Instruction Set Reference. This will in many cases remove the need for using the dedicated compare instructions, resulting in faster and more compact code.

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

The AVR Status Register – SREG – is defined as:

Bit	7	6	5	4	3	2	1	0	
0x3F (0x5F)	I	T	H	S	V	N	Z	C	SREG
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 – I: Global Interrupt Enable**

The Global Interrupt Enable bit must be set for the interrupts to be enabled. The individual interrupt enable control is then performed in separate control registers. If the Global Interrupt Enable Register is cleared, none of the interrupts are enabled independent of the individual interrupt enable settings. The I-bit is cleared by hardware after an interrupt has occurred, and is set by the RETI instruction to enable subsequent interrupts. The I-bit can also be set and cleared by the application with the SEI and CLI instructions, as described in the instruction set reference.

- **Bit 6 – T: Bit Copy Storage**

The Bit Copy instructions BLD (Bit LoaD) and BST (Bit STore) use the T-bit as source or destination for the operated bit. A bit from a register in the Register File can be copied into T by the BST instruction, and a bit in T can be copied into a bit in a register in the Register File by the BLD instruction.

- **Bit 5 – H: Half Carry Flag**

The Half Carry Flag H indicates a Half Carry in some arithmetic operations. Half Carry Is useful in BCD arithmetic. See the “Instruction Set Description” for detailed information.

- **Bit 4 – S: Sign Bit, S = N ⊕ V**

The S-bit is always an exclusive or between the negative flag N and the Two’s Complement Overflow Flag V. See the “Instruction Set Description” for detailed information.

- **Bit 3 – V: Two’s Complement Overflow Flag**

The Two’s Complement Overflow Flag V supports two’s complement arithmetics. See the “Instruction Set Description” for detailed information.

- **Bit 2 – N: Negative Flag**

The Negative Flag N indicates a negative result in an arithmetic or logic operation. See the “Instruction Set Description” for detailed information.

- **Bit 1 – Z: Zero Flag**

The Zero Flag Z indicates a zero result in an arithmetic or logic operation. See the “Instruction Set Description” for detailed information.

- **Bit 0 – C: Carry Flag**

The Carry Flag C indicates a carry in an arithmetic or logic operation. See the “Instruction Set Description” for detailed information.

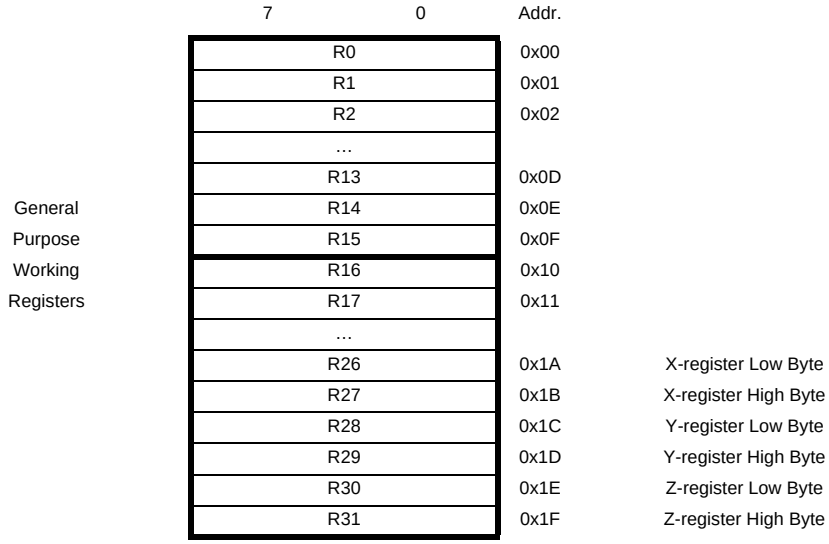
4.4 General Purpose Register File

The Register File is optimized for the AVR Enhanced RISC instruction set. In order to achieve the required performance and flexibility, the following input/output schemes are supported by the Register File:

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

Figure 4-2 shows the structure of the 32 general purpose working registers in the CPU.

Figure 4-2. AVR CPU General Purpose Working Registers



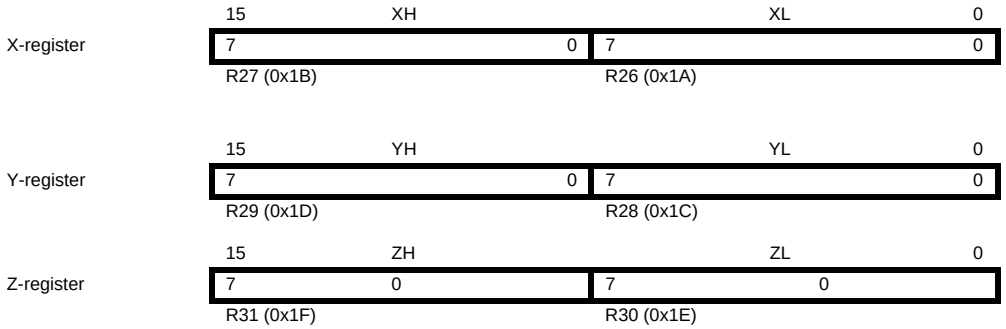
Most of the instructions operating on the Register File have direct access to all registers, and most of them are single cycle instructions.

As shown in Figure 4-2, each register is also assigned a data memory address, mapping them directly into the first 32 locations of the user Data Space. Although not being physically implemented as SRAM locations, this memory organization provides great flexibility in access of the registers, as the X-, Y- and Z-pointer registers can be set to index any register in the file.

4.4.1 The X-register, Y-register, and Z-register

The registers R26..R31 have some added functions to their general purpose usage. These registers are 16-bit address pointers for indirect addressing of the data space. The three indirect address registers X, Y, and Z are defined as described in Figure 4-3.

Figure 4-3. The X-, Y-, and Z-registers



In the different addressing modes these address registers have functions as fixed displacement, automatic increment, and automatic decrement (see the instruction set reference for details).

4.5 Stack Pointer

The Stack is mainly used for storing temporary data, for storing local variables and for storing return addresses after interrupts and subroutine calls. The Stack Pointer Register always points to the top of the Stack. Note that the Stack is implemented as growing from higher memory locations to lower memory locations. This implies that a Stack PUSH command decreases the Stack Pointer.

The Stack Pointer points to the data SRAM Stack area where the Subroutine and Interrupt Stacks are located. This Stack space in the data SRAM must be defined by the program before any subroutine calls are executed or interrupts are enabled. The Stack Pointer must be set to point above 0x60. The Stack Pointer is decremented by one when data is pushed onto the Stack with the PUSH instruction, and it is decremented by two when the return address is pushed onto the Stack with subroutine call or interrupt. The Stack Pointer is incremented by one when data is popped from the Stack with the POP instruction, and it is incremented by two when data is popped from the Stack with return from subroutine RET or return from interrupt RETI.

The Stack Pointer is implemented as one 8-bit register in the I/O space.

Bit	7	6	5	4	3	2	1	0	
0x3D (0x5D)	**SP7**	**SP6**	**SP5**	**SP4**	**SP3**	**SP2**	**SP1**	**SP0**	**SPL**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	

4.6 Instruction Execution Timing

This section describes the general access timing concepts for instruction execution. The AVR CPU is driven by the CPU clock clk_{CPU}, directly generated from the selected clock source for the chip. No internal clock division is used.

Figure 4-4 shows the parallel instruction fetches and instruction executions enabled by the Harvard architecture and the fast-access Register File concept. This is the basic pipelining concept to obtain up to 1 MIPS per MHz with the corresponding unique results for functions per cost, functions per clocks, and functions per power-unit.

Figure 4-4. The Parallel Instruction Fetches and Instruction Executions

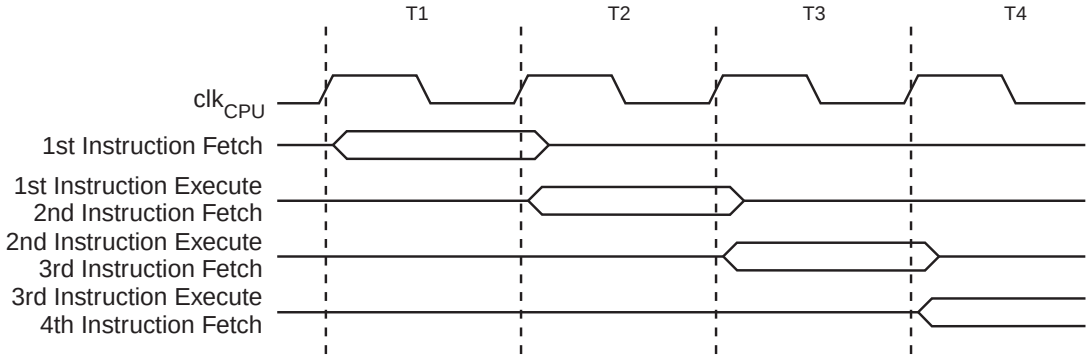
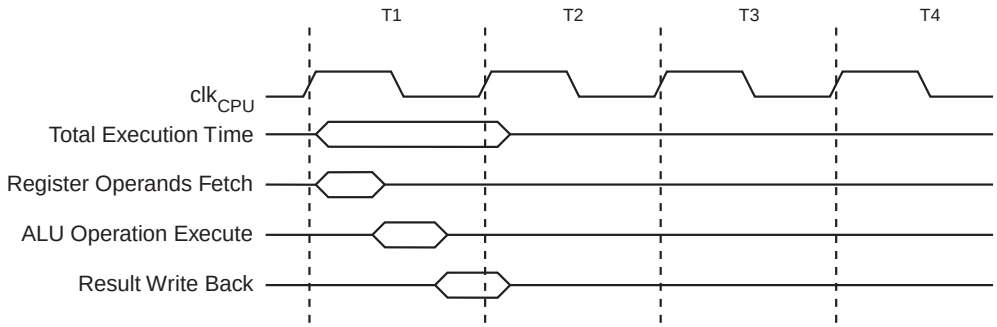


Figure 4-5 shows the internal timing concept for the Register File. In a single clock cycle an ALU operation using two register operands is executed, and the result is stored back to the destination register.

Figure 4-5. Single Cycle ALU Operation



4.7 Reset and Interrupt Handling

The AVR provides several different interrupt sources. These interrupts and the separate Reset Vector each have a separate program vector in the program memory space. All interrupts are assigned individual enable bits which must be written logic one together with the Global Interrupt Enable bit in the Status Register in order to enable the interrupt.

The lowest addresses in the program memory space are by default defined as the Reset and Interrupt Vectors. The complete list of vectors is shown in “Interrupts” on page 48. The list also determines the priority levels of the different interrupts. The lower the address the higher is the priority level. RESET has the highest priority, and next is INT0 – the External Interrupt Request 0. Refer to “Interrupts” on page 48 for more information.

When an interrupt occurs, the Global Interrupt Enable I-bit is cleared and all interrupts are disabled. The user software can write logic one to the I-bit to enable nested interrupts. All enabled interrupts can then interrupt the current interrupt routine. The I-bit is automatically set when a Return from Interrupt instruction – RETI – is executed.

There are basically two types of interrupts. The first type is triggered by an event that sets the interrupt flag. For these interrupts, the Program Counter is vectored to the actual Interrupt Vector in order to execute the interrupt handling routine, and hardware clears the corresponding interrupt flag. Interrupt flags can also be cleared by writing a logic one to the flag bit position(s) to be

cleared. If an interrupt condition occurs while the corresponding interrupt enable bit is cleared, the interrupt flag will be set and remembered until the interrupt is enabled, or the flag is cleared by software. Similarly, if one or more interrupt conditions occur while the Global Interrupt Enable bit is cleared, the corresponding interrupt flag(s) will be set and remembered until the Global Interrupt Enable bit is set, and will then be executed by order of priority.

The second type of interrupts will trigger as long as the interrupt condition is present. These interrupts do not necessarily have interrupt flags. If the interrupt condition disappears before the interrupt is enabled, the interrupt will not be triggered.

When the AVR exits from an interrupt, it will always return to the main program and execute one more instruction before any pending interrupt is served.

Note that the Status Register is not automatically stored when entering an interrupt routine, nor restored when returning from an interrupt routine. This must be handled by software.

When using the CLI instruction to disable interrupts, the interrupts will be immediately disabled. No interrupt will be executed after the CLI instruction, even if it occurs simultaneously with the CLI instruction. The following example shows how this can be used to avoid interrupts during the timed EEPROM write sequence..

Assembly Code Example

```
in  r16, SREG     ; store SREG value
cli    ; disable interrupts during timed sequence
sbi EECR, EEMPE   ; start EEPROM write
sbi EECR, EEPE
out SREG, r16     ; restore SREG value (I-bit)
```

C Code Example

```
char cSREG;
cSREG = SREG; /* store SREG value */
/* disable interrupts during timed sequence */
__disable_interrupt();
EECR |= (1<<EEMPE); /* start EEPROM write */
EECR |= (1<<EEPE);
SREG = cSREG; /* restore SREG value (I-bit) */
```

When using the SEI instruction to enable interrupts, the instruction following SEI will be executed before any pending interrupts, as shown in this example.

Assembly Code Example

```
sei  ; set Global Interrupt Enable
sleep; enter sleep, waiting for interrupt
; note: will enter sleep before any pending
; interrupt(s)
```

C Code Example

```
__enable_interrupt(); /* set Global Interrupt Enable */
__sleep(); /* enter sleep, waiting for interrupt */
/* note: will enter sleep before any pending interrupt(s) */
```

4.7.1 Interrupt Response Time

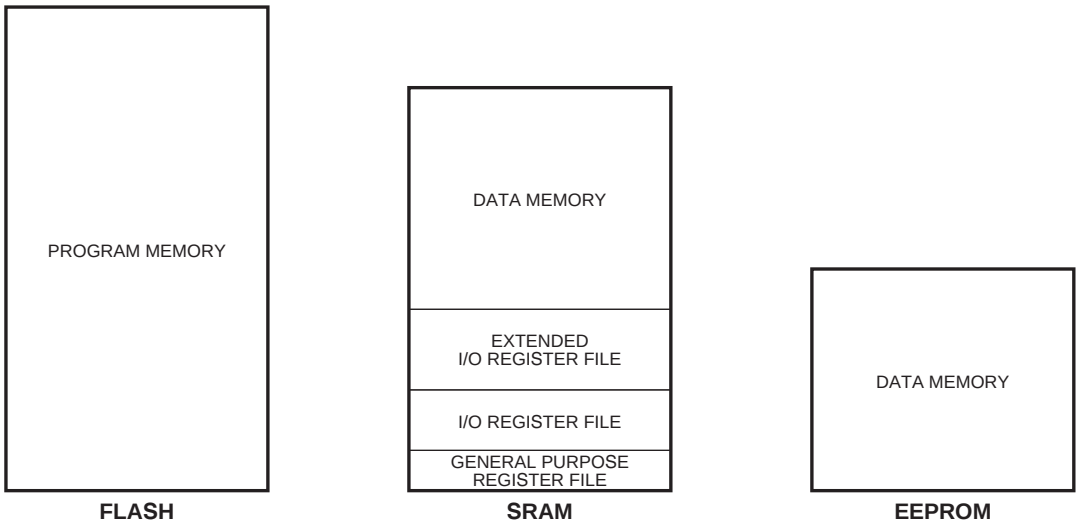
The interrupt execution response for all the enabled AVR interrupts is four clock cycles minimum. After four clock cycles the program vector address for the actual interrupt handling routine is executed. During this four clock cycle period, the Program Counter is pushed onto the Stack. The vector is normally a jump to the interrupt routine, and this jump takes three clock cycles. If an interrupt occurs during execution of a multi-cycle instruction, this instruction is completed before the interrupt is served. If an interrupt occurs when the MCU is in sleep mode, the interrupt execution response time is increased by four clock cycles. This increase comes in addition to the start-up time from the selected sleep mode.

A return from an interrupt handling routine takes four clock cycles. During these four clock cycles, the Program Counter (two bytes) is popped back from the Stack, the Stack Pointer is incremented by two, and the I-bit in SREG is set.

5. Memories

The AVR architecture makes a distinction between program memory and data memory, locating each memory type in a separate address space. Executable code is located in non-volatile program memory (Flash), whereas data can be placed in either volatile (SRAM) or non-volatile memory (EEPROM). See Figure 5-1, below.

Figure 5-1. Memory Overview.



All memory spaces are linear and regular.

5.1 Program Memory (Flash)

ATtiny2313A/4313 contains 2/4K byte of on-chip, in-system reprogrammable Flash memory for program storage. Flash memories are non-volatile, i.e. they retain stored information even when not powered.

Since all AVR instructions are 16 or 32 bits wide, the Flash is organized as 1024/2048 x 16 bits. The Program Counter (PC) is 10/11 bits wide, thus capable of addressing all 1024/2048 locations of program memory, as illustrated in Table 5-1, below.

Table 5-1. Size of Program Memory (Flash).

Device	Flash Size		Address Range
ATtiny2313A	2KB	1024 words	0x0000 – 0x03FF
ATtiny4313	4KB	2048 words	0x0000 – 0x07FF

Constant tables can be allocated within the entire address space of program memory. See instructions LPM (Load Program Memory), and SPM (Store Program Memory) in “Instruction Set Summary” on page 257. Flash program memory can also be programmed from an external device, as described in “External Programming” on page 184.

Timing diagrams for instruction fetch and execution are presented in “Instruction Execution Timing” on page 12.

The Flash memory has a minimum endurance of 10,000 write/erase cycles.

5.2 Data Memory (SRAM) and Register Files

Table 5-2 shows how the data memory and register files of ATtiny2313A/4313 are organized. These memory areas are volatile, i.e. they do not retain information when power is removed.

Table 5-2. Layout of Data Memory and Register Area.

Device	Memory Area	Size	Long Address (1)	Short Address (2)
ATtiny2313A	General purpose register file	32B	0x000 – 0x01F	n/a
	I/O register file	64B	0x020 – 0x05F	0x00 – 0x3F
	Data SRAM	128B	0x060 – 0x0E0	n/a
ATtiny4313	General purpose register file	32B	0x000 – 0x01F	n/a
	I/O register file	64B	0x020 – 0x05F	0x00 – 0x3F
	Data SRAM	256B	0x060 – 0x160	n/a

Note: 1. Also known as data address. This mode of addressing covers the entire data memory and register area. The address is contained in a 16-bit area of two-word instructions.

2. Also known as direct I/O address. This mode of addressing covers part of the register area, only. It is used by instructions where the address is embedded in the instruction word.

The 224/352 memory locations include the general purpose register file, I/O register file, extended I/O register file, and the internal data memory.

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

5.2.1 General Purpose Register File

The first 32 locations are reserved for the general purpose register file. These registers are described in detail in “General Purpose Register File” on page 11.

5.2.2 I/O Register File

Following the general purpose register file, the next 64 locations are reserved for I/O registers. Registers in this area are used mainly for communicating with I/O and peripheral units of the device. Data can be transferred between I/O space and the general purpose register file using instructions such as IN, OUT, LD, ST, and derivatives.

All I/O registers in this area can be accessed with the instructions IN and OUT. These I/O specific instructions address the first location in the I/O register area as 0x00 and the last as 0x3F.

The low 32 registers (address range 0x00...0x1F) are accessible by some bit-specific instructions. In these registers, bits are easily set and cleared using SBI and CBI, while bit-conditional branches are readily constructed using instructions SBIC, SBIS, SBRC, and SBRS.

Registers in this area may also be accessed with instructions LD/LDD/LDI/LDS and ST/STD/STS. These instructions treat the entire volatile memory as one data space and, therefore, address I/O registers starting at 0x20.

See “Instruction Set Summary” on page 257.

ATtiny2313A/4313 also contains three general purpose I/O registers that can be used for storing any information. See GPIOR0, GPIOR1 and GPIOR2 in “Register Summary” on page 255. These general purpose I/O registers are particularly useful for storing global variables and status

flags, since they are accessible to bit-specific instructions such as SBI, CBI, SBIC, SBIS, SBRC, and SBRS.

5.2.3 Data Memory (SRAM)

Following the general purpose register file and the I/O register file, the remaining 128/256 locations are reserved for the internal data SRAM.

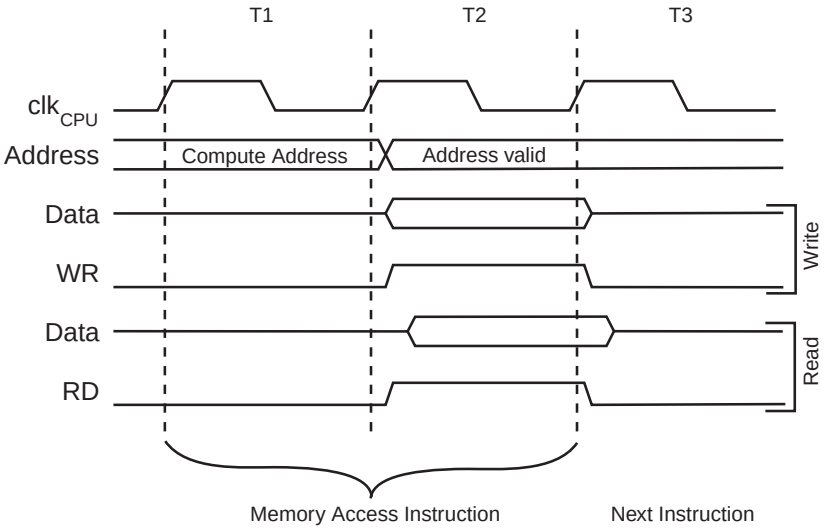
There are five addressing modes available:

- Direct. This mode of addressing reaches the entire data space.
- Indirect.
- Indirect with Displacement. This mode of addressing reaches 63 address locations from the base address given by the Y- or Z-register.
- Indirect with Pre-decrement. In this mode the address register is automatically decremented before access. Address pointer registers (X, Y, and Z) are located in the general purpose register file, in registers R26 to R31. See “General Purpose Register File” on page 11.
- Indirect with Post-increment. In this mode the address register is automatically incremented after access. Address pointer registers (X, Y, and Z) are located in the general purpose register file, in registers R26 to R31. See “General Purpose Register File” on page 11.

All addressing modes can be used on the entire volatile memory, including the general purpose register file, the I/O register files and the data memory.

Internal SRAM is accessed in two clk_{CPU} cycles, as illustrated in Figure 5-2, below.

Figure 5-2. On-chip Data SRAM Access Cycles



5.3 Data Memory (EEPROM)

ATtiny2313A/4313 contains 128 bytes of non-volatile data memory. This EEPROM is organized as a separate data space, in which single bytes can be read and written. All access registers are located in the I/O space.

The EEPROM memory layout is summarised in Table 5-3, below.

Table 5-3. Size of Non-Volatile Data Memory (EEPROM).

Device	EEPROM Size	Address Range
ATtiny2313A	128B	0x00 – 0x7F
ATtiny4313	256B	0x00 – 0xFF

The internal 8MHz oscillator is used to time EEPROM operations. The frequency of the oscillator must be within the requirements described in “OSCCAL – Oscillator Calibration Register” on page 32.

When powered by heavily filtered supplies, the supply voltage, V_{CC}, is likely to rise or fall slowly on power-up and power-down. Slow rise and fall times may put the device in a state where it is running at supply voltages lower than specified. To avoid problems in situations like this, see “Preventing EEPROM Corruption” on page 20.

The EEPROM has a minimum endurance of 100,000 write/erase cycles.

5.3.1 Programming Methods

There are two methods for EEPROM programming:

- Atomic byte programming. This is the simple mode of programming, where target locations are erased and written in a single operation. In this mode of operation the target is guaranteed to always be erased before writing but programmin times are longer.
- Split byte programming. It is possible to split the erase and write cycle in two different operations. This is useful when short access times are required, for example when supply voltage is falling. In order to take advantage of this method target locations must be erased before writing to them. This can be done at times when the system allows time-critical operations, typically at start-up and initialisation.

The programming method is selected using the EEPROM Programming Mode bits (EEPM1 and EEPM0) in EEPROM Control Register (EECR). See Table 5-4 on page 24. Write and erase times are given in the same table.

Since programming takes some time the application must wait for one operation to complete before starting the next. This can be done by either polling the EEPROM Program Enable bit (EEPE) in EEPROM Control Register (EECR), or via the EEPROM Ready Interrupt. The EEPROM interrupt is controlled by the EEPROM Ready Interrupt Enable (EERIE) bit in EECR.

5.3.2 Read

To read an EEPROM memory location follow the procedure below:

- Poll the EEPROM Program Enable bit (EEPE) in EEPROM Control Register (EECR) to make sure no other EEPROM operations are in process. If set, wait to clear.
- Write target address to EEPROM Address Registers (EEARH/EEARL).
- Start the read operation by setting the EEPROM Read Enable bit (EERE) in the EEPROM Control Register (EECR). During the read operation, the CPU is halted for four clock cycles before executing the next instruction.
- Read data from the EEPROM Data Register (EEDR).

5.3.3 Erase

In order to prevent unintentional EEPROM writes, a specific procedure must be followed to erase memory locations. To erase an EEPROM memory location follow the procedure below:

- Poll the EEPROM Program Enable bit (EEPE) in EEPROM Control Register (EECR) to make sure no other EEPROM operations are in process. If set, wait to clear.
- Set mode of programming to erase by writing EEPROM Programming Mode bits (EEPM0 and EEPM1) in EEPROM Control Register (EECR).
- Write target address to EEPROM Address Registers (EEARH/EEARL).
- Enable erase by setting EEPROM Master Program Enable (EEMPE) in EEPROM Control Register (EECR). Within four clock cycles, start the erase operation by setting the EEPROM Program Enable bit (EEPE) in the EEPROM Control Register (EECR). During the erase operation, the CPU is halted for two clock cycles before executing the next instruction.

The EEPE bit remains set until the erase operation has completed. While the device is busy programming, it is not possible to perform any other EEPROM operations.

5.3.4 Write

In order to prevent unintentional EEPROM writes, a specific procedure must be followed to write to memory locations.

Before writing data to EEPROM the target location must be erased. This can be done either in the same operation or as part of a split operation. Writing to an unerased EEPROM location will result in corrupted data.

To write an EEPROM memory location follow the procedure below:

- Poll the EEPROM Program Enable bit (EEPE) in EEPROM Control Register (EECR) to make sure no other EEPROM operations are in process. If set, wait to clear.
- Set mode of programming by writing EEPROM Programming Mode bits (EEPM0 and EEPM1) in EEPROM Control Register (EECR). Alternatively, data can be written in one operation or the write procedure can be split up in erase, only, and write, only.
- Write target address to EEPROM Address Registers (EEARH/EEARL).
- Write target data to EEPROM Data Register (EEDR).
- Enable write by setting EEPROM Master Program Enable (EEMPE) in EEPROM Control Register (EECR). Within four clock cycles, start the write operation by setting the EEPROM Program Enable bit (EEPE) in the EEPROM Control Register (EECR). During the write operation, the CPU is halted for two clock cycles before executing the next instruction.

The EEPE bit remains set until the write operation has completed. While the device is busy with programming, it is not possible to do any other EEPROM operations.

5.3.5 Preventing EEPROM Corruption

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

At low supply voltages data in EEPROM can be corrupted in two ways:

- The supply voltage is too low to maintain proper operation of an otherwise legitimate EEPROM program sequence.
- The supply voltage is too low for the CPU and instructions may be executed incorrectly.

EEPROM data corruption is avoided by keeping the device in reset during periods of insufficient power supply voltage. This is easily done by enabling the internal Brown-Out Detector (BOD). If BOD detection levels are not sufficient for the design, an external reset circuit for low V_{CC} can be used.

Provided that supply voltage is sufficient, an EEPROM write operation will be completed even when a reset occurs.

5.3.6 Program Examples

The following code examples show one assembly and one C function for erase, write, or atomic write of the EEPROM. The examples assume that interrupts are controlled (e.g., by disabling interrupts globally) so that no interrupts occur during execution of these functions.

Assembly Code Example

```
EEPROM_write:
  ; Wait for completion of previous write
  sbic EECR, EEPE
  rjmp EEPROM_write
  ; Set Programming mode
  ldi  r16, (0<<EEPM1)|(0<<EEPM0)
  out  EECR, r16
  ; Set up address (r18:r17) in address registers
  out  EEARH, r18
  out  EEARL, r17
  ; Write data (r19) to data register
  out  EEDR, r19
  ; Write logical one to EEMPE
  sbi  EECR, EEMPE
  ; Start eeprom write by setting EEPE
  sbi  EECR, EEPE
  ret
```

Note: See “Code Examples” on page 7.

C Code Example

```
void EEPROM_write(unsigned int ucAddress, unsigned char ucData)
{
  /* Wait for completion of previous write */
  while(EECR & (1<<EEPE));
  /* Set Programming mode */
  EECR = (0<<EEPM1)|(0<<EEPM0);
  /* Set up address and data registers */
  EEAR = ucAddress;
  EEDR = ucData;
  /* Write logical one to EEMPE */
  EECR |= (1<<EEMPE);
  /* Start eeprom write by setting EEPE */
  EECR |= (1<<EEPE);
}
```

Note: See “Code Examples” on page 7.

The next code examples show assembly and C functions for reading the EEPROM. The examples assume that interrupts are controlled so that no interrupts will occur during execution of these functions.

Assembly Code Example

```
EEPROM_read:
  ; Wait for completion of previous write
  sbic EECR, EEPE
  rjmp EEPROM_read
  ; Set up address (r18:r17) in address registers
  out  EEARH, r18
  out  EEARL, r17
  ; Start eeprom read by writing EERE
  sbi  EECR, EERE
  ; Read data from data register
  in   r16, EEDR
  ret
```

Note: See “Code Examples” on page 7.

C Code Example

```
unsigned char EEPROM_read(unsigned int ucAddress)
{
  /* Wait for completion of previous write */
  while(EECR & (1<<EEPE));
  /* Set up address register */
  EEAR = ucAddress;
  /* Start eeprom read by writing EERE */
  EECR |= (1<<EERE);
  /* Return data from data register */
  return EEDR;
}
```

Note: See “Code Examples” on page 7.

5.4 Register Description

5.4.1 EEAR – EEPROM Address Register

Bit	7	6	5	4	3	2	1	0	
0x1E (0x3E)	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	EEAR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	X	X	X	X	X	X	X	X	

- **Bit 7 – EEAR7: EEPROM Address**

This is the most significant EEPROM address bit of ATtiny4313. In devices with less EEPROM, i.e. ATtiny2313A, this bit is reserved and will always read zero. The initial value of the EEPROM Address Register (EEAR) is undefined and a proper value must therefore be written before the EEPROM is accessed.

- **Bits 6..0 – EEAR6..0: EEPROM Address**

These are the (low) bits of the EEPROM Address Register. The EEPROM data bytes are addressed linearly in the range 0...(128-1). The initial value of EEAR is undefined and a proper value must be therefore be written before the EEPROM may be accessed.

5.4.2 EEDR – EEPROM Data Register

Bit	7	6	5	4	3	2	1	0	
0x1D (0x3D)	EEDR7	EEDR6	EEDR5	EEDR4	EEDR3	EEDR2	EEDR1	EEDR0	EEDR
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	

- **Bits 7..0 – EEDR7..0: EEPROM Data**

For the EEPROM write operation the EEDR Register contains the data to be written to the EEPROM in the address given by the EEAR Register. For the EEPROM read operation, the EEDR contains the data read out from the EEPROM at the address given by EEAR.

5.4.3 EECR – EEPROM Control Register

Bit	7	6	5	4	3	2	1	0	
0x1C (0x3C)	–	–	EEPM1	EEPM0	EERIE	EEMPE	EEPE	EERE	EECR
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	X	X	0	0	X	0	

- **Bits 7, 6 – Res: Reserved Bit**

These bits are reserved and will always read zero.

- **Bits 5, 4 – EEPM1 and EEPM0: EEPROM Programming Mode Bits**

The EEPROM Programming mode bits setting defines which programming action that will be triggered when writing EEPE. It is possible to program data in one atomic operation (erase the old value and program the new value) or to split the Erase and Write operations in two different operations. The Programming times for the different modes are shown in Table 5-4.

Table 5-4. EEPROM Programming Mode Bits and Programming Times

EEPM1	EEPM0	Programming Time	Operation
0	0	3.4 ms	Erase and Write in one operation (Atomic Operation)
0	1	1.8 ms	Erase Only
1	0	1.8 ms	Write Only
1	1	–	Reserved for future use

When EEPE is set any write to EEPMn will be ignored.

During reset, the EEPMn bits will be reset to 0b00 unless the EEPROM is busy programming.

- **Bit 3 – EERIE: EEPROM Ready Interrupt Enable**

Writing this bit to one enables the EEPROM Ready Interrupt. Provided the I-bit in SREG is set, the EEPROM Ready Interrupt is triggered when non-volatile memory is ready for programming.

Writing this bit to zero disables the EEPROM Ready Interrupt.

- **Bit 2 – EEMPE: EEPROM Master Program Enable**

The EEMPE bit determines whether writing EEPE to one will have effect or not.

When EEMPE is set and EEPE written within four clock cycles the EEPROM at the selected address will be programmed. Hardware clears the EEMPE bit to zero after four clock cycles.

If EEMPE is zero the EEPE bit will have no effect.

- **Bit 1 – EEPE: EEPROM Program Enable**

This is the programming enable signal of the EEPROM. The EEMPE bit must be set before EEPE is written, or EEPROM will not be programmed.

When EEPE is written, the EEPROM will be programmed according to the EEPMn bit settings. When EEPE has been set, the CPU is halted for two cycles before the next instruction is executed. After the write access time has elapsed, the EEPE bit is cleared by hardware.

Note that an EEPROM write operation blocks all software programming of Flash, fuse bits, and lock bits.

- **Bit 0 – EERE: EEPROM Read Enable**

This is the read strobe of the EEPROM. When the target address has been set up in the EEAR, the EERE bit must be written to one to trigger the EEPROM read operation.

EEPROM read access takes one instruction, and the requested data is available immediately. When the EEPROM is read, the CPU is halted for four cycles before the next instruction is executed.

The user should poll the EEPE bit before starting the read operation. If a write operation is in progress, it not possible to read the EEPROM, or to change the address register (EEAR).

5.4.4 GPIOR2 – General Purpose I/O Register 2

Bit	7	6	5	4	3	2	1	0	
0x15 (0x35)	MSB							LSB	GPIOR2
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	

5.4.5 GPIOR1 – General Purpose I/O Register 1

Bit	7	6	5	4	3	2	1	0	
0x14 (0x34)	MSB							LSB	GPIOR1
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	

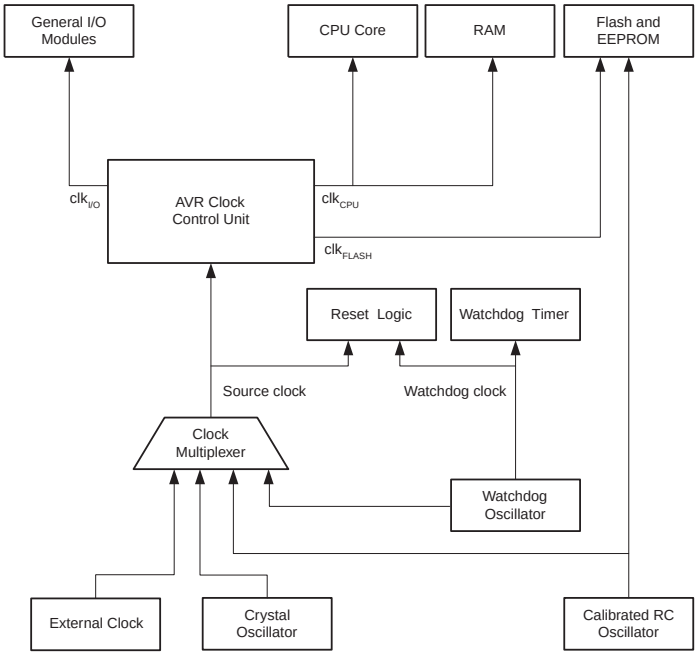
5.4.6 GPIOR0 – General Purpose I/O Register 0

Bit	7	6	5	4	3	2	1	0	
0x13 (0x33)	MSB							LSB	GPIOR0
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	

6. Clock System

Figure 6-1 presents the principal clock systems in the AVR and their distribution. All of the clocks need not be active at a given time. In order to reduce power consumption, the clocks to modules not being used can be halted by using different sleep modes, as described in “Power Management and Sleep Modes” on page 34.

Figure 6-1. Clock Distribution



6.1 Clock Subsystems

6.1.1 CPU Clock – clk_{CPU}

The CPU clock is routed to parts of the system concerned with operation of the AVR core. Examples of such modules are the General Purpose Register File, the Status Register and the data memory holding the Stack Pointer. Halting the CPU clock inhibits the core from performing general operations and calculations.

6.1.2 I/O Clock – $clk_{I/O}$

The I/O clock is used by the majority of the I/O modules, like Timer/Counters, and USART. The I/O clock is also used by the External Interrupt module, but note that some external interrupts are detected by asynchronous logic, allowing such interrupts to be detected even if the I/O clock is halted. Also note that start condition detection in the USI module is carried out asynchronously when $clk_{I/O}$ is halted, enabling USI start condition detection in all sleep modes.

6.1.3 Flash Clock – clk_{FLASH}

The Flash clock controls operation of the Flash interface. The Flash clock is usually active simultaneously with the CPU clock.

6.2 Clock Sources

The device has the following clock source options, selectable by Flash Fuse bits as shown below. The clock from the selected source is input to the AVR clock generator, and routed to the appropriate modules.

Table 6-1. Device Clocking Select

Device Clocking Option	CKSEL3..0[1]
External Clock (see page 27)	0000
Calibrated Internal RC Oscillator 4 MHz (see page 28)	0010
Calibrated internal RC Oscillator 8 MHz (see page 28)	0100
128 kHz Internal Oscillator (see page 29)	0110
External Crystal/Ceramic Resonator (see page 30)	1000 - 1111
Reserved	0001/0011/0101/0111

Note: 1. For all fuses “1” means unprogrammed while “0” means programmed.

The various choices for each clocking option is given in the following sections. When the CPU wakes up from Power-down, the selected clock source is used to time the start-up, ensuring stable Oscillator operation before instruction execution starts. When the CPU starts from reset, there is an additional delay allowing the power to reach a stable level before commencing normal operation. The Watchdog Oscillator is used for timing this real-time part of the start-up time.

6.2.1 Default Clock Source

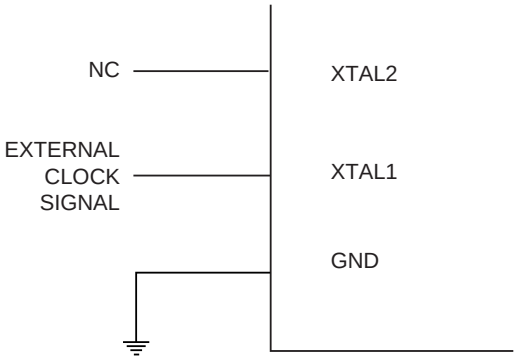
The device is shipped with CKSEL = “0100”, SUT = “10”, and CKDIV8 programmed. The default clock source setting is the Internal RC Oscillator with longest start-up time and an initial system clock prescaling of 8, resulting in 1.0 MHz system clock. This default setting ensures that all users can make their desired clock source setting using an In-System or Parallel programmer.

For low-voltage devices it should be noted that unprogramming the CKDIV8 fuse may result in overclocking. At low voltages (below 2.7V) the devices are rated for maximum 4 MHz operation (see Section 22.3 on page 200), but routing the clock signal from the internal oscillator directly to the system clock line will run the device at 8 MHz.

6.2.2 External Clock

To drive the device from an external clock source, XTAL1 should be driven as shown in Figure 6-2. To run the device on an external clock, the CKSEL Fuses must be programmed to “0000”.

Figure 6-2. External Clock Drive Configuration



When this clock source is selected, start-up times are determined by the SUT Fuses as shown in Table 6-2.

Table 6-2. Start-up Times for the External Clock Selection

SUT1..0	Start-up Time from Power-down and Power-save	Additional Delay from Reset	Recommended Usage
00	6 CK	14CK	BOD enabled
01	6 CK	14CK + 4 ms	Fast rising power
10	6 CK	14CK + 64 ms	Slowly rising power
11	Reserved		

When applying an external clock, it is required to avoid sudden changes in the applied clock frequency to ensure stable operation of the MCU. A variation in frequency of more than 2% from one clock cycle to the next can lead to unpredictable behavior. It is required to ensure that the MCU is kept in Reset during such changes in the clock frequency.

Note that the System Clock Prescaler can be used to implement run-time changes of the internal clock frequency while still ensuring stable operation. See “System Clock Prescaler” on page 31 for details.

6.2.3 Calibrated Internal RC Oscillator

The calibrated internal RC Oscillator provides a fixed 8.0 MHz clock. The frequency is nominal value at 3V and 25°C. If 8 MHz frequency exceeds the speed specification of the device (depends on V_{CC}), the CKDIV8 Fuse must be programmed in order to divide the internal frequency by 8 during start-up. The device is shipped with the CKDIV8 Fuse programmed. This clock may be selected as the system clock by programming the CKSEL Fuses as shown in Table 6-3. If selected, it will operate with no external components. During reset, hardware loads the calibration byte into the OSCCAL Register and thereby automatically calibrates the RC Oscillator. At 3V and 25°C, this calibration gives a frequency within ± 10% of the nominal frequency. Using calibration methods as described in application notes available at www.atmel.com/avr it is possible to achieve ± 2% accuracy at any given V_{CC} and Temperature. When this Oscillator is used as the chip clock, the Watchdog Oscillator will still be used for the

Watchdog Timer and for the Reset Time-out. For more information on the pre-programmed calibration value, see the section “Calibration Byte” on page 181.

Table 6-3. Internal Calibrated RC Oscillator Operating Modes

CKSEL3..0	Nominal Frequency
0010	4.0 MHz
0100	8.0 MHz(1)

Note: 1. The device is shipped with this option selected.

When this Oscillator is selected, start-up times are determined by the SUT Fuses as shown in Table 6-4.

Table 6-4. Start-up Times for the Internal Calibrated RC Oscillator Clock Selection

SUT1..0	Start-up Time from Power-down and Power-save	Additional Delay from Reset	Recommended Usage
00	6 CK	14CK(1)	BOD enabled
01	6 CK	14CK + 4 ms	Fast rising power
10(2)	6 CK	14CK + 64 ms	Slowly rising power
11	Reserved		

Notes: 1. If the RSTDISBL fuse is programmed, this start-up time will be increased to 14CK + 4 ms to ensure programming mode can be entered.
2. The device is shipped with this option selected.

6.2.4 128 kHz Internal Oscillator

The 128 kHz Internal Oscillator is a low power Oscillator providing a clock of 128 kHz. The frequency is nominal at 3 V and 25°C. This clock may be selected as the system clock by programming the CKSEL Fuses to 0110.

When this clock source is selected, start-up times are determined by the SUT Fuses as shown in Table 6-5.

Table 6-5. Start-up Times for the 128 kHz Internal Oscillator

SUT1..0	Start-up Time from Power-down and Power-save	Additional Delay from Reset	Recommended Usage
00	6 CK	14CK(1)	BOD enabled
01	6 CK	14CK + 4 ms	Fast rising power
10	6 CK	14CK + 64 ms	Slowly rising power
11	Reserved		

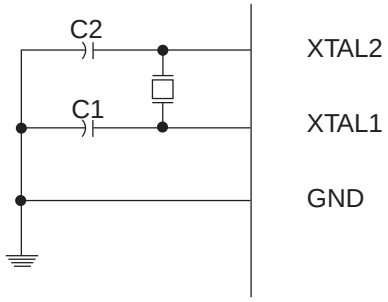
Note: 1. If the RSTDISBL fuse is programmed, this start-up time will be increased to 14CK + 4 ms to ensure programming mode can be entered.

6.2.5 Crystal Oscillator

XTAL1 and XTAL2 are input and output, respectively, of an inverting amplifier which can be configured for use as an On-chip Oscillator, as shown in Figure 6-3 on page 30. Either a quartz crystal or a ceramic resonator may be used.

C1 and C2 should always be equal for both crystals and resonators. The optimal value of the capacitors depends on the crystal or resonator in use, the amount of stray capacitance, and the electromagnetic noise of the environment. Some initial guidelines for choosing capacitors for use with crystals are given in Table 6-6 on page 30. For ceramic resonators, the capacitor values given by the manufacturer should be used.

Figure 6-3. Crystal Oscillator Connections



The Oscillator can operate in three different modes, each optimized for a specific frequency range. The operating mode is selected by the fuses CKSEL3..1 as shown in Table 6-6.

Table 6-6. Crystal Oscillator Operating Modes

CKSEL3..1	Frequency Range (MHz)	Recommended Range for Capacitors C1 and C2 for Use with Crystals (pF)
100[1]	0.4 - 0.9	–
101	0.9 - 3.0	12 - 22
110	3.0 - 8.0	12 - 22
111	8.0 -	12 - 22

Note: 1. This option should not be used with crystals, only with ceramic resonators.

The CKSEL0 Fuse together with the SUT1..0 Fuses select the start-up times as shown in Table 6-7.

Table 6-7. Start-up Times for the Crystal Oscillator Clock Selection

CKSEL0	SUT1..0	Start-up Time from Power-down and Power-save	Additional Delay from Reset (V_{CC} = 5.0V)	Recommended Usage
0	00	258 CK(1)	14CK + 4 ms	Ceramic resonator, fast rising power
0	01	258 CK(1)	14CK + 64 ms	Ceramic resonator, slowly rising power
0	10	1K CK(2)	14CK	Ceramic resonator, BOD enabled
0	11	1K CK(2)	14CK + 4 ms	Ceramic resonator, fast rising power
1	00	1K CK(2)	14CK + 64 ms	Ceramic resonator, slowly rising power
1	01	16K CK	14CK	Crystal Oscillator, BOD enabled
1	10	16K CK	14CK + 4 ms	Crystal Oscillator, fast rising power
1	11	16K CK	14CK + 64 ms	Crystal Oscillator, slowly rising power

Notes:
1. 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.
2. 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.

6.3 System Clock Prescaler

The ATtiny2313A/4313 has a system clock prescaler, and the system clock can be divided by setting the "CLKPR – Clock Prescale Register" on page 32. This feature can be used to decrease the system clock frequency and the 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_{CPU}, and clk_{FLASH} are divided by a factor as shown in Table 6-8 on page 33.

6.3.1 Switching Time

When switching between prescaler settings, the System Clock Prescaler ensures that no glitches occurs in the clock system. It also ensures 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 the other 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, two active clock edges are produced. Here, T1 is the previous clock period, and T2 is the period corresponding to the new prescaler setting.

6.4 Clock Output Buffer

The device can output the system clock on the CKOUT pin. To enable this, the CKOUT fuse has to be programmed. This mode of operation is useful when the chip clock is needed to drive other circuits in the system.

Note that the clock will not be output during reset and that the normal operation of the I/O pin will be overridden when the fuse is programmed. Any clock source, including the internal RC Oscillator, can be selected when the clock is output on CKOUT pin.

If the System Clock Prescaler is used, it is the divided system clock that is output.

6.5 Register Description

6.5.1 OSCCAL – Oscillator Calibration Register

Bit	7	6	5	4	3	2	1	0	
0x31 (0x51)	–	CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0	OSCCAL
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	Device Specific Calibration Value							

- **Bit 7 – Res: Reserved Bit**

This bit is reserved bit in ATtiny2313A/4313 and it will always read zero.

- **Bits 6:0 – CAL[6:0]: Oscillator Calibration Value**

Writing the calibration byte to this address will trim the internal Oscillator to remove process variations from the Oscillator frequency. This is done automatically during Chip Reset. When OSCCAL is zero, the lowest available frequency is chosen. Writing non-zero values to this register will increase the frequency of the internal Oscillator. Writing 0x7F to the register gives the highest available frequency.

The calibrated Oscillator is used to time EEPROM and Flash access. If EEPROM or Flash is written, do not calibrate to more than 10% above the nominal frequency. Otherwise, the EEPROM or Flash write may fail. Note that the Oscillator is intended for calibration to 8 MHz or 4 MHz. Tuning to other values is not guaranteed, as indicated in Table 6-8 below.

To ensure stable operation of the MCU the calibration value should be changed in small steps. A variation in frequency of more than 2% from one cycle to the next can lead to unpredicatble behavior. Changes in OSCCAL should not exceed 0x20 for each calibration. It is required to ensure that the MCU is kept in Reset during such changes in the clock frequency.

6.5.2 CLKPR – Clock Prescale Register

Bit	7	6	5	4	3	2	1	0	
0x26 (0x46)	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				

- **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 ATtiny2313A/4313 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 6-8 on page 33.

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 to make sure the write procedure is not interrupted. 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 6-8. 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

7. Power Management and Sleep Modes

The high performance and industry leading code efficiency makes the AVR microcontrollers an ideal choise for low power applications. In addition, 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.

7.1 Sleep Modes

Figure 6-1 on page 26 presents the different clock systems and their distribution in ATtiny2313A/4313. The figure is helpful in selecting an appropriate sleep mode. Table 7-1 shows the different sleep modes and their wake up sources.

Table 7-1. Active Clock Domains and Wake-up Sources in Different Sleep Modes

	Active Clock Domains			Oscillators	Wake-up Sources				
Sleep Mode	clk_{CPU}	clk_{FLASH}	clk_{IO}	Main Clock Source Enabled	INT0, INT1 and Pin Change	USI Start Condition	SPM/EEPROM Ready Interrupt	Other I/O	Watchdog Interrupt
Idle			X	X	X	X	X	X	X
Power-down					X(1)	X			X
Stand-by				X	X(1)	X			X

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

To enter any of the three sleep modes, the SE bit in MCUCR must be written to logic one and a SLEEP instruction must be executed. The SM1..0 bits in the MCUCR Register select which sleep mode (Idle, Standby or Power-down) will be activated by the SLEEP instruction. See Table 7-2 on page 37 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.

Note that if a level triggered interrupt is used for wake-up the changed level must be held for some time to wake up the MCU (and for the MCU to enter the interrupt service routine). See "External Interrupts" on page 49 for details.

7.1.1 Idle Mode

When the SM1..0 bits are written to 00, the SLEEP instruction makes the MCU enter Idle mode, stopping the CPU but allowing Analog Comparator, USI, Timer/Counter, Watchdog, and the interrupt system to continue operating. This sleep mode basically halts clk_{CPU} and clk_{FLASH}, while allowing the other clocks to run.

Idle mode enables the MCU to wake up from external triggered interrupts as well as internal ones like the Timer Overflow. If wake-up from the Analog Comparator interrupt is not required, the Analog Comparator can be powered down by setting the ACD bit in "ACSR – Analog Com-

parator Control and Status Register" on page 168. This will reduce power consumption in Idle mode.

7.1.2 Power-Down Mode

When the SM1:0 bits are written to 01/11, the SLEEP instruction makes the MCU enter Power-down mode. In this mode, the Oscillator is stopped, while the external interrupts, and the Watchdog continue operating (if enabled). Only an External Reset, a Watchdog Reset, a Brown-out Reset, an external level interrupt on INT0 and INT1, or a pin change interrupt can wake up the MCU. This sleep mode halts all generated clocks, allowing operation of asynchronous modules only.

7.1.3 Standby Mode

When the SM1..0 bits are 10 and an external crystal/resonator clock option is selected, the SLEEP instruction makes the MCU enter Standby mode. This mode is identical to Power-down with the exception that the Oscillator is kept running. From Standby mode, the device wakes up in six clock cycles.

7.2 Software BOD Disable

When the Brown-out Detector (BOD) is enabled by BODLEVEL fuses (see Table 20-4 on page 179), the BOD is actively monitoring the supply voltage during a sleep period. In some devices it is possible to save power by disabling the BOD by software in Power-Down and Stand-By sleep modes. The sleep mode power consumption will then be at the same level as when BOD is globally disabled by fuses.

If BOD is disabled by 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 V_{CC} level has dropped during the sleep period.

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

BOD disable is controlled by the BODS (BOD Sleep) bit of BOD Control Register, see "BODCR – Brown-Out Detector Control Register" on page 38. Writing this bit to one turns off BOD in Power-Down and Stand-By, while writing a zero keeps the BOD active. The default setting is zero, i.e. BOD active.

Writing to the BODS bit is controlled by a timed sequence and an enable bit, see "BODCR – Brown-Out Detector Control Register" on page 38.

7.3 Power Reduction Register

The Power Reduction Register (PRR), see "PRR – Power Reduction Register" on page 37, provides a method to reduce power consumption by stopping the clock to individual peripherals. When the clock for a peripheral is stopped then:

- The current state of the peripheral is frozen.
- The associated registers can not be read or written.
- Resources used by the peripheral will remain occupied.

The peripheral should in most cases be disabled before stopping the clock. Clearing the PRR bit wakes up the peripheral and puts it in the same state as before shutdown.

Peripheral shutdown can be used in Idle mode and Active mode to significantly reduce the overall power consumption. See “Effect of Power Reduction” on page 206 for examples. In all other sleep modes, the clock is already stopped.

7.4 Minimizing Power Consumption

There are several issues to consider when trying to minimize the power consumption in an AVR controlled system. In general, sleep modes should be used as much as possible, and the sleep mode should be selected so that as few as possible of the device’s functions are operating. All functions not needed should be disabled. In particular, the following modules may need special consideration when trying to achieve the lowest possible power consumption.

7.4.1 Analog Comparator

When entering Idle mode, the Analog Comparator should be disabled if not used. In the other sleep modes, the Analog Comparator is automatically disabled. However, if the Analog Comparator is set up to use the Internal Voltage Reference as input, the Analog Comparator should be disabled in all sleep modes. Otherwise, the Internal Voltage Reference will be enabled, independent of sleep mode. See “Analog Comparator” on page 168 for details on how to configure the Analog Comparator.

7.4.2 Internal Voltage Reference

The Internal Voltage Reference will be enabled when needed by the Brown-out Detection or the Analog Comparator. If these modules are disabled as described in the sections above, the internal voltage reference will be disabled and it will not be consuming power. When turned on again, the user must allow the reference to start up before the output is used. If the reference is kept on in sleep mode, the output can be used immediately. See “Internal Voltage Reference” on page 42 for details on the start-up time.

7.4.3 Brown-out Detector

If the Brown-out Detector is not needed in the application, this module should be turned off. If the Brown-out Detector is enabled by the BODLEVEL Fuses, it will be enabled in all sleep modes, and hence, always consume power. In the deeper sleep modes, this will contribute significantly to the total current consumption. See “Brown-out Detection” on page 41 and “Software BOD Disable” on page 35 for details on how to configure the Brown-out Detector.

7.4.4 Watchdog Timer

If the Watchdog Timer is not needed in the application, this module should be turned off. If the Watchdog Timer is enabled, it will be enabled in all sleep modes, and hence, always consume power. In the deeper sleep modes, this will contribute significantly to the total current consumption. See “Interrupts” on page 48 for details on how to configure the Watchdog Timer.

7.4.5 Port Pins

When entering a sleep mode, all port pins should be configured to use minimum power. The most important thing is then to ensure that no pins drive resistive loads. In sleep modes where the I/O clock ($clk_{I/O}$) is stopped, the input buffers of the device will be disabled. This ensures that no power is consumed by the input logic when not needed. In some cases, the input logic is needed for detecting wake-up conditions, and it will then be enabled. See the section “Digital Input Enable and Sleep Modes” on page 58 for details on which pins are enabled. If the input buffer is enabled and the input signal is left floating or has an analog signal level close to $V_{CC}/2$, the input buffer will use excessive power.

For analog input pins, the digital input buffer should be disabled at all times. An analog signal level close to $V_{CC}/2$ on an input pin can cause significant current even in active mode. Digital input buffers can be disabled by writing to the Digital Input Disable Register (DIDR). See "DIDR – Digital Input Disable Register" on page 169 for details.

7.5 Register Description

7.5.1 MCUCR – MCU Control Register

The Sleep Mode Control Register contains control bits for power management.

Bit	7	6	5	4	3	2	1	0	
0x35 (0x55)	PUD	SM1	SE	SM0	ISC11	ISC10	ISC01	ISC00	MCUCR
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	

- **Bits 6, 4 – SM1..0: Sleep Mode Select Bits 1 and 0**

These bits select between the four available sleep modes as shown in Table 7-2.

Table 7-2. Sleep Mode Select

SM1	SM0	Sleep Mode
0	0	Idle
0	1	Power-down
1	0	Standby
1	1	Power-down

Note: Standby mode is only recommended for use with external crystals or resonators.

- **Bit 5 – SE: Sleep Enable**

The SE bit must be written to logic one to make the MCU enter the sleep mode when the SLEEP instruction is executed. To avoid the MCU entering the sleep mode unless it is the programmer's purpose, it is recommended to write the Sleep Enable (SE) bit to one just before the execution of the SLEEP instruction and to clear it immediately after waking up.

7.5.2 PRR – Power Reduction Register

The Power Reduction Register provides a method to reduce power consumption by allowing peripheral clock signals to be disabled.

Bit	7	6	5	4	3	2	1	0	
0x06 (0x26)	–	–	–	–	PRTIM1	PRTIM0	PRUSI	PRUSART	PRR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7..4 – Res: Reserved Bits**

These bits are reserved and will always read zero.

- **Bit 3 – PRTIM1: Power Reduction Timer/Counter1**

Writing a logic one to this bit shuts down the Timer/Counter1 module. When the Timer/Counter1 is enabled, operation will continue like before the shutdown.

- **Bit 2 – PRTIM0: Power Reduction Timer/Counter0**

Writing a logic one to this bit shuts down the Timer/Counter0 module. When the Timer/Counter0 is enabled, operation will continue like before the shutdown.

- **Bit 1 – PRUSI: Power Reduction USI**

Writing a logic one to this bit shuts down the USI by stopping the clock to the module. When waking up the USI again, the USI should be re initialized to ensure proper operation.

- **Bit 0 – PRUSART: Power Reduction USART**

Writing a logic one to this bit shuts down the USI by stopping the clock to the module. When waking up the USI again, the USI should be re initialized to ensure proper operation.

7.5.3 BODCR – Brown-Out Detector Control Register

The BOD Control Register contains control bits for disabling the BOD by software.

Bit	7	6	5	4	3	2	1	0	
0x07 (0x27)	—	—	—	—	—	—	BODS	BODSE	BODCR
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 1 – BODS: BOD Sleep**

In order to disable BOD during sleep the BODS bit must be written to logic one. This is controlled by a timed sequence and the enable bit, BODSE. First, both BODS and BODSE must be set to one. Second, within four clock cycles, BODS must be set to one and BODSE must be set to zero. The BODS bit is active three clock cycles after it is set. A sleep instruction must be executed while BODS is active in order to turn off the BOD for the actual sleep mode. The BODS bit is automatically cleared after three clock cycles.

- **Bit 0 – BODSE: BOD Sleep Enable**

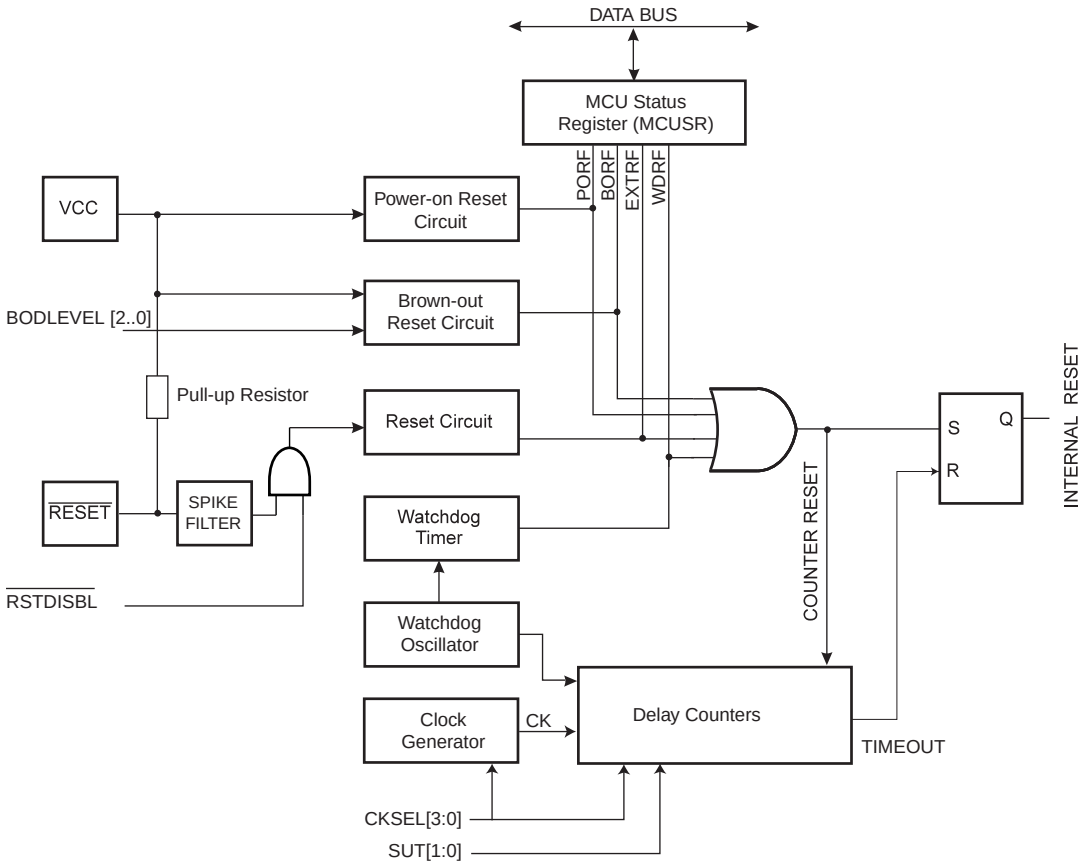
The BODSE bit enables setting of BODS control bit, as explained on BODS bit description. BOD disable is controlled by a timed sequence.

8. System Control and Reset

8.1 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 a 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 8-1 shows the reset logic. Electrical parameters of the reset circuitry are given in Table 22-3 on page 201.

Figure 8-1. Reset Logic



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 "Clock Sources" on page 27.

8.2 Reset Sources

The ATtiny2313A/4313 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 when $\overline{\text{RESET}}$ function is enabled
- Watchdog Reset. The MCU is reset when the Watchdog Timer period expires and the Watchdog is enabled
- Brown-out Reset. The MCU is reset when the supply voltage V_{CC} is below the Brown-out Reset threshold (V_{BOT}) and the Brown-out Detector is enabled

8.2.1 Power-on Reset

A Power-on Reset (POR) pulse is generated by an on-chip detection circuit. The detection level is defined in "System and Reset Characteristics" on page 201. The POR is activated whenever V_{CC} is below the detection level. The POR circuit can be used to trigger the Start-up Reset, as well as to detect a failure in supply voltage.

A Power-on Reset (POR) circuit ensures that the device is reset from Power-on. Reaching the Power-on Reset threshold voltage invokes the delay counter, which determines how long the device is kept in reset after V_{CC} rise. The reset signal is activated again, without any delay, when V_{CC} decreases below the detection level.

Figure 8-2. MCU Start-up, $\overline{\text{RESET}}$ Tied to V_{CC}

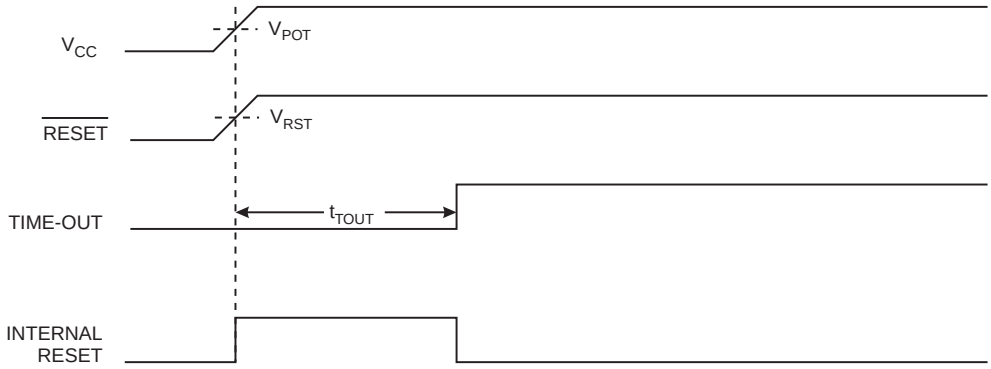
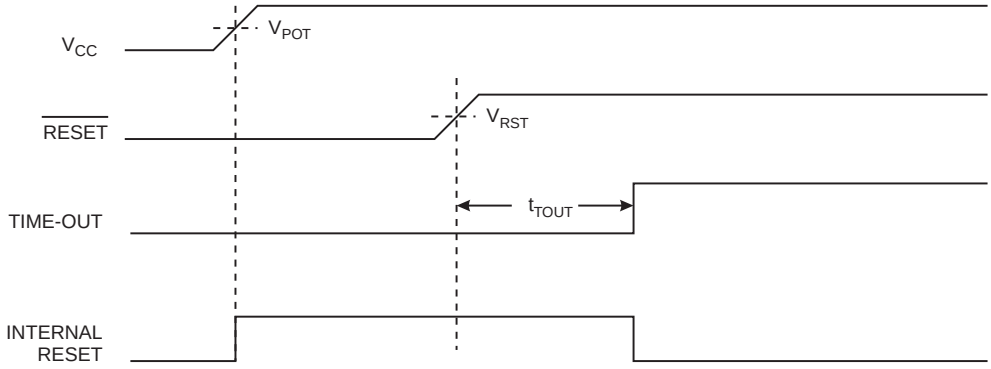


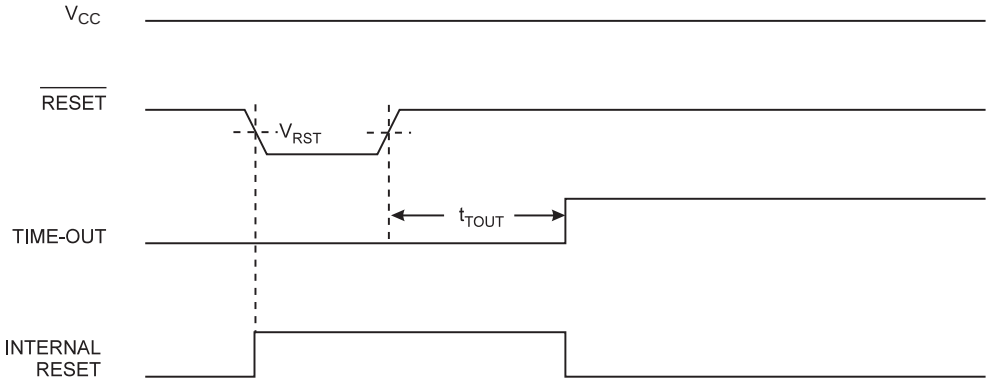
Figure 8-3. MCU Start-up, $\overline{\text{RESET}}$ Extended Externally



8.2.2 External Reset

An External Reset is generated by a low level on the $\overline{\text{RESET}}$ pin if enabled. Reset pulses longer than the minimum pulse width (see "System and Reset Characteristics" on page 201) 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.

Figure 8-4. External Reset During Operation



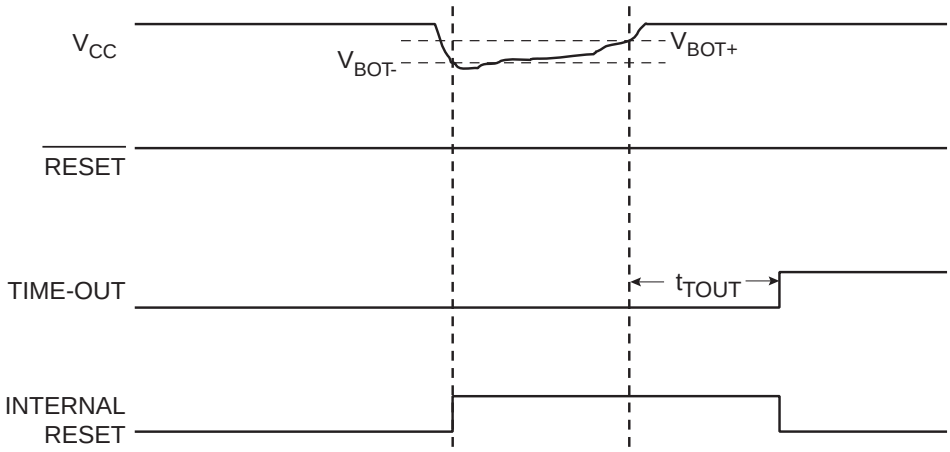
8.2.3 Brown-out Detection

ATtiny2313A/4313 has an On-chip Brown-out Detection (BOD) circuit for monitoring the V_{CC} level during operation by comparing it to a fixed trigger level. The trigger level for the BOD can be selected by the BODLEVEL Fuses. 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 $V_{BOT-} = V_{BOT} - V_{HYST}/2$.

When the BOD is enabled, and V_{CC} decreases to a value below the trigger level (V_{BOT-} in Figure 8-5 on page 41), the Brown-out Reset is immediately activated. When V_{CC} increases above the trigger level (V_{BOT+} in Figure 8-5 on page 41), the delay counter starts the MCU after the Time-out period t_{TOUT} has expired.

The BOD circuit will only detect a drop in V_{CC} if the voltage stays below the trigger level for longer than t_{BOD} given in "System and Reset Characteristics" on page 201.

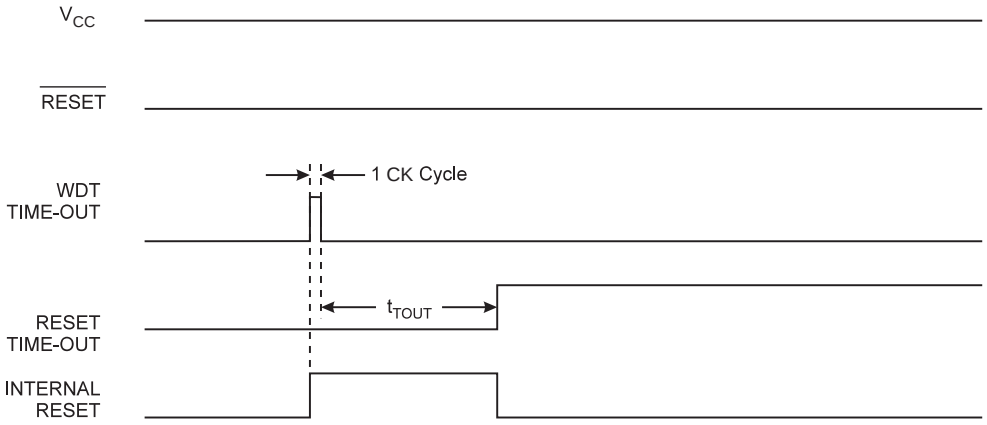
Figure 8-5. Brown-out Reset During Operation



8.2.4 Watchdog Reset

When the Watchdog times out, it will generate a short reset pulse of one CK cycle duration. On the falling edge of this pulse, the delay timer starts counting the Time-out period t_{TOUT}. See "Interrupts" on page 48 for details on operation of the Watchdog Timer.

Figure 8-6. Watchdog Reset During Operation



8.3 Internal Voltage Reference

ATtiny2313A/4313 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. The bandgap voltage varies with supply voltage and temperature.

8.3.1 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 "System and Reset Characteristics" on page 201. 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] Fuse).
2. When the internal reference is connected to the Analog Comparator (by setting the ACBG bit in ACSR).

Thus, when the BOD is not enabled, after setting the ACBG bit, the user must always allow the reference to start up before the output from the Analog Comparator. To reduce power consumption in Power-down mode, the user can avoid the three conditions above to ensure that the reference is turned off before entering Power-down mode.

8.4 Watchdog Timer

The Watchdog Timer is clocked from an On-chip Oscillator which runs at 128 kHz. By controlling the Watchdog Timer prescaler, the Watchdog Reset interval can be adjusted as shown in Table 8-3 on page 47. The WDR – Watchdog Reset – instruction resets the Watchdog Timer. The Watchdog Timer is also reset when it is disabled and when a Chip Reset occurs. Ten different clock cycle periods can be selected to determine the reset period. If the reset period expires without another Watchdog Reset, the ATtiny2313A/4313 resets and executes from the Reset Vector. For timing details on the Watchdog Reset, refer to Table 8-3 on page 47.

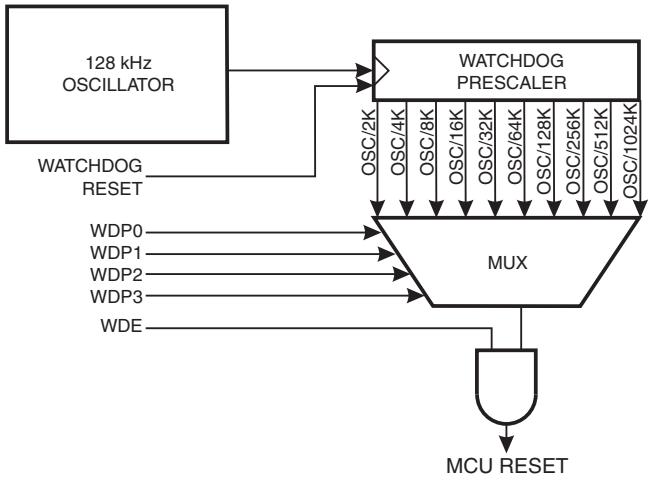
The Watchdog Timer can also be configured to generate an interrupt instead of a reset. This can be very helpful when using the Watchdog to wake-up from Power-down.

To prevent unintentional disabling of the Watchdog or unintentional change of time-out period, two different safety levels are selected by the fuse WDTON as shown in Table 8-1 See “Timed Sequences for Changing the Configuration of the Watchdog Timer” on page 43 for details.

Table 8-1. WDT Configuration as a Function of the Fuse Settings of WDTON

WDTON	Safety Level	WDT Initial State	How to Disable the WDT	How to Change Time-out
Unprogrammed	1	Disabled	Timed sequence	No limitations
Programmed	2	Enabled	Always enabled	Timed sequence

Figure 8-7. Watchdog Timer



8.4.1 Timed Sequences for Changing the Configuration of the Watchdog Timer

The sequence for changing configuration differs slightly between the two safety levels. Separate procedures are described for each level.

- Safety Level 1

  In this mode, the Watchdog Timer is initially disabled, but can be enabled by writing the WDE bit to one without any restriction. A timed sequence is needed when disabling an enabled Watchdog Timer. To disable an enabled Watchdog Timer, the following procedure must be followed:

  a. In the same operation, write a logic one to WDCE and WDE. A logic one must be written to WDE regardless of the previous value of the WDE bit
  b. Within the next four clock cycles, in the same operation, write the WDE and WDP bits as desired, but with the WDCE bit cleared

- Safety Level 2

  In this mode, the Watchdog Timer is always enabled, and the WDE bit will always read as one. A timed sequence is needed when changing the Watchdog Time-out period. To change the Watchdog Time-out, the following procedure must be followed:

  a. In the same operation, write a logical one to WDCE and WDE. Even though the WDE always is set, the WDE must be written to one to start the timed sequence
  b. Within the next four clock cycles, in the same operation, write the WDP bits as desired, but with the WDCE bit cleared. The value written to the WDE bit is irrelevant

8.4.2 Code Example

The following code example shows one assembly and one C function for turning off the WDT. The example assumes that interrupts are controlled (e.g., by disabling interrupts globally) so that no interrupts will occur during execution of these functions.

Assembly Code Example[1]

```
WDT_off:
  wdr
  ; Clear WDRF in MCUSR
  ldi  r16, (0<<WDRF)
  out  MCUSR, r16
  ; Write logical one to WDCE and WDE
  ; Keep old prescaler setting to prevent unintentional Watchdog Reset
  in   r16, WDTCSR
  ori  r16, (1<<WDCE)|(1<<WDE)
  out  WDTCSR, r16
  ; Turn off WDT
  ldi  r16, (0<<WDE)
  out  WDTCSR, r16
  ret
```

C Code Example[1]

```
void WDT_off(void)
{
  _WDR();
  /* Clear WDRF in MCUSR */
  MCUSR = 0x00;
  /* Write logical one to WDCE and WDE */
  WDTCSR |= (1<<WDCE) | (1<<WDE);
  /* Turn off WDT */
  WDTCSR = 0x00;
}
```

Note: 1. See “Code Examples” on page 7.

8.5 Register Description

8.5.1 MCUSR – MCU Status Register

The MCU Status Register provides information on which reset source caused an MCU Reset.

Bit	7	6	5	4	3	2	1	0	
0x34 (0x54)	–	–	–	–	WDRF	BORF	EXTRF	PORF	MCUSR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	See Bit Description				

- **Bits 7..4 – Res: Reserved Bits**

These bits are reserved bits in the ATtiny2313A/4313 and will always read as zero.

- **Bit 3 – WDRF: Watchdog Reset Flag**

This bit is set if a Watchdog Reset occurs. The bit is reset by a Power-on Reset, or by writing a logic zero to the flag.

- **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.

8.5.2 WDTCSR – Watchdog Timer Control and Status Register

Bit	7	6	5	4	3	2	1	0	
0x21 (0x41)	WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0	WDTCSR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	X	0	0	0	

- **Bit 7 – WDIF: Watchdog Timeout Interrupt Flag**

This bit is set when a time-out occurs in the Watchdog Timer and the Watchdog Timer is configured for interrupt. WDIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, WDIF is cleared by writing a logic one to the flag. When the I-bit in SREG and WDIE are set, the Watchdog Time-out Interrupt is executed.

- **Bit 6 – WDIE: Watchdog Timeout Interrupt Enable**

When this bit is written to one, WDE is cleared, and the I-bit in the Status Register is set, the Watchdog Time-out Interrupt is enabled. In this mode the corresponding interrupt is executed instead of a reset if a timeout in the Watchdog Timer occurs.

If WDE is set, WDIE is automatically cleared by hardware when a time-out occurs. This is useful for keeping the Watchdog Reset security while using the interrupt. After the WDIE bit is cleared,

the next time-out will generate a reset. To avoid the Watchdog Reset, WDIE must be set after each interrupt.

Table 8-2. Watchdog Timer Configuration

WDE	WDIE	Watchdog Timer State	Action on Time-out
0	0	Stopped	None
0	1	Running	Interrupt
1	0	Running	Reset
1	1	Running	Interrupt

- **Bit 4 – WDCE: Watchdog Change Enable**

This bit must be set when the WDE bit is written to logic zero. Otherwise, the Watchdog will not be disabled. Once written to one, hardware will clear this bit after four clock cycles. See the description of the WDE bit for a Watchdog disable procedure. This bit must also be set when changing the prescaler bits. See “Timed Sequences for Changing the Configuration of the Watchdog Timer” on page 43.

- **Bit 3 – WDE: Watchdog Enable**

When the WDE is written to logic one, the Watchdog Timer is enabled, and if the WDE is written to logic zero, the Watchdog Timer function is disabled. WDE can only be cleared if the WDCE bit has logic level one. To disable an enabled Watchdog Timer, the following procedure must be followed:

1. In the same operation, write a logic one to WDCE and WDE. A logic one must be written to WDE even though it is set to one before the disable operation starts.
2. Within the next four clock cycles, write a logic 0 to WDE. This disables the Watchdog.

In safety level 2, it is not possible to disable the Watchdog Timer, even with the algorithm described above. See “Timed Sequences for Changing the Configuration of the Watchdog Timer” on page 43.

In safety level 1, WDE is overridden by WDRF in MCUSR. See “MCUSR – MCU Status Register” on page 45 for description of WDRF. This means that WDE is always set when WDRF is set. To clear WDE, WDRF must be cleared before disabling the Watchdog with the procedure described above. This feature ensures multiple resets during conditions causing failure, and a safe start-up after the failure.

Note: If the watchdog timer is not going to be used in the application, it is important to go through a watchdog disable procedure in the initialization of the device. If the Watchdog is accidentally enabled, for example by a runaway pointer or brown-out condition, the device will be reset, which in turn will lead to a new watchdog reset. To avoid this situation, the application software should always clear the WDRF flag and the WDE control bit in the initialization routine.

- **Bits 5, 2..0 – WDP3..0: Watchdog Timer Prescaler 3, 2, 1, and 0**

The WDP3..0 bits determine the Watchdog Timer prescaling when the Watchdog Timer is enabled. The different prescaling values and their corresponding Timeout Periods are shown in Table 8-3 on page 47.

Table 8-3. Watchdog Timer Prescale Select

WDP3	WDP2	WDP1	WDP0	Number of WDT Oscillator Cycles	Typical Time-out at V_{CC} = 5.0V
0	0	0	0	2K cycles	16 ms
0	0	0	1	4K cycles	32 ms
0	0	1	0	8K cycles	64 ms
0	0	1	1	16K cycles	0.125 s
0	1	0	0	32K cycles	0.25 s
0	1	0	1	64K cycles	0.5 s
0	1	1	0	128K cycles	1.0 s
0	1	1	1	256K cycles	2.0 s
1	0	0	0	512K cycles	4.0 s
1	0	0	1	1024K cycles	8.0 s
1	0	1	0	Reserved[1]	
1	0	1	1		
1	1	0	0		
1	1	0	1		
1	1	1	0		
1	1	1	1		

Note: 1. If selected, one of the valid settings below 0b1010 will be used.

9. Interrupts

This section describes the specifics of the interrupt handling as performed in ATtiny2313A/4313. For a general explanation of the AVR interrupt handling, refer to “Reset and Interrupt Handling” on page 13.

9.1 Interrupt Vectors

The interrupt vectors of ATtiny2313A/4313 are described in Table 9-1 below

Table 9-1. Reset and Interrupt Vectors

Vector No.	Program Address	Label	Interrupt Source
1	0x0000	RESET	External Pin, Power-on Reset, Brown-out Reset, and Watchdog Reset
2	0x0001	INT0	External Interrupt Request 0
3	0x0002	INT1	External Interrupt Request 1
4	0x0003	TIMER1 CAPT	Timer/Counter1 Capture Event
5	0x0004	TIMER1 COMPA	Timer/Counter1 Compare Match A
6	0x0005	TIMER1 OVF	Timer/Counter1 Overflow
7	0x0006	TIMER0 OVF	Timer/Counter0 Overflow
8	0x0007	USART0, RX	USART0, Rx Complete
9	0x0008	USART0, UDRE	USART0 Data Register Empty
10	0x0009	USART0, TX	USART0, Tx Complete
11	0x000A	ANALOG COMP	Analog Comparator
12	0x000B	PCINT0	Pin Change Interrupt Request 0
13	0x000C	TIMER1 COMPB	Timer/Counter1 Compare Match B
14	0x000D	TIMER0 COMPA	Timer/Counter0 Compare Match A
15	0x000E	TIMER0 COMPB	Timer/Counter0 Compare Match B
16	0x000F	USI START	USI Start Condition
17	0x0010	USI OVERFLOW	USI Overflow
18	0x0011	EE READY	EEPROM Ready
19	0x0012	WDT OVERFLOW	Watchdog Timer Overflow
20	0x0013	PCINT1	Pin Change Interrupt Request 1
21	0x0014	PCINT2	Pin Change Interrupt Request 2

In case the program never enables an interrupt source, the Interrupt Vectors will not be used and, consequently, regular program code can be placed at these locations.

The most typical and general setup for the Interrupt Vector Addresses in ATtiny2313A/4313 shown below:

```
Address  Labels Code                 Comments
0x0000         rjmp   RESET          ; Reset Handler
0x0001         rjmp   INT0           ; External Interrupt0 Handler
0x0002         rjmp   INT1           ; External Interrupt1 Handler
0x0003         rjmp   TIM1_CAPT      ; Timer1 Capture Handler
0x0004         rjmp   TIM1_COMPA     ; Timer1 CompareA Handler
0x0005         rjmp   TIM1_OVF       ; Timer1 Overflow Handler
0x0006         rjmp   TIM0_OVF       ; Timer0 Overflow Handler
0x0007         rjmp   USART0_RXC     ; USART0 RX Complete Handler
0x0008         rjmp   USART0_DRE     ; USART0,UDR Empty Handler
0x0009         rjmp   USART0_TXC     ; USART0 TX Complete Handler
0x000A         rjmp   ANA_COMP       ; Analog Comparator Handler
0x000B         rjmp   PCINT0         ; PCINT0 Handler
0x000C         rjmp   TIMER1_COMPB   ; Timer1 Compare B Handler
0x000D         rjmp   TIMER0_COMPA   ; Timer0 Compare A Handler
0x000E         rjmp   TIMER0_COMPB   ; Timer0 Compare B Handler
0x000F         rjmp   USI_START      ; USI Start Handler
0x0010         rjmp   USI_OVERFLOW   ; USI Overflow Handler
0x0011         rjmp   EE_READY       ; EEPROM Ready Handler
0x0012         rjmp   WDT_OVERFLOW   ; Watchdog Overflow Handler
0x0013         rjmp   PCINT1         ; PCINT1 Handler
0x0014         rjmp   PCINT2         ; PCINT2 Handler
;
0x0013  RESET: ldi    r16, low(RAMEND); Main program start
0x0014         out    SPL,r16          Set Stack Pointer to top of RAM
0x0015         sei                   ; Enable interrupts
0x0016         <instr>  xxx
  ...     ...    ...  ...
```

9.2 External Interrupts

External Interrupts are triggered by the INT0 or INT1 pin or any of the PCINT17..0 pins. Observe that, if enabled, the interrupts will trigger even if the INT0, INT1 or PCINT17..0 pins are configured as outputs. This feature provides a way of generating a software interrupt. Pin change 0 interrupts PCI0 will trigger if any enabled PCINT7..0 pin toggles. Pin change 1 interrupts PCI1 will trigger if any enabled PCINT10..8 pin toggles. Pin change 2 interrupts PCI2 will trigger, if any enabled PCINT17..11 pin toggles. The PCMSK0, PCMSK1, and PCMSK2 Registers control which pins contribute to the pin change interrupts. Pin change interrupts on PCINT17..0 are detected asynchronously, which means that these interrupts can be used for waking the part also from sleep modes other than Idle mode.

The INT0 and INT1 interrupts can be triggered by a falling or rising edge or a low level. This is set up as shown in “MCUCR – MCU Control Register” on page 51. When the INT0 or INT1 interrupt is enabled and configured as level triggered, the interrupt will trigger as long as the pin is held low. Note that recognition of falling or rising edge interrupts on INT0 or INT1 requires the presence of an I/O clock, as described in “Clock Sources” on page 27.

9.2.1 Low Level Interrupt

A low level interrupt on INT0 or INT1 is detected asynchronously. This means that the interrupt source can be used for waking the part also from sleep modes other than Idle (the I/O clock is halted in all sleep modes except Idle).

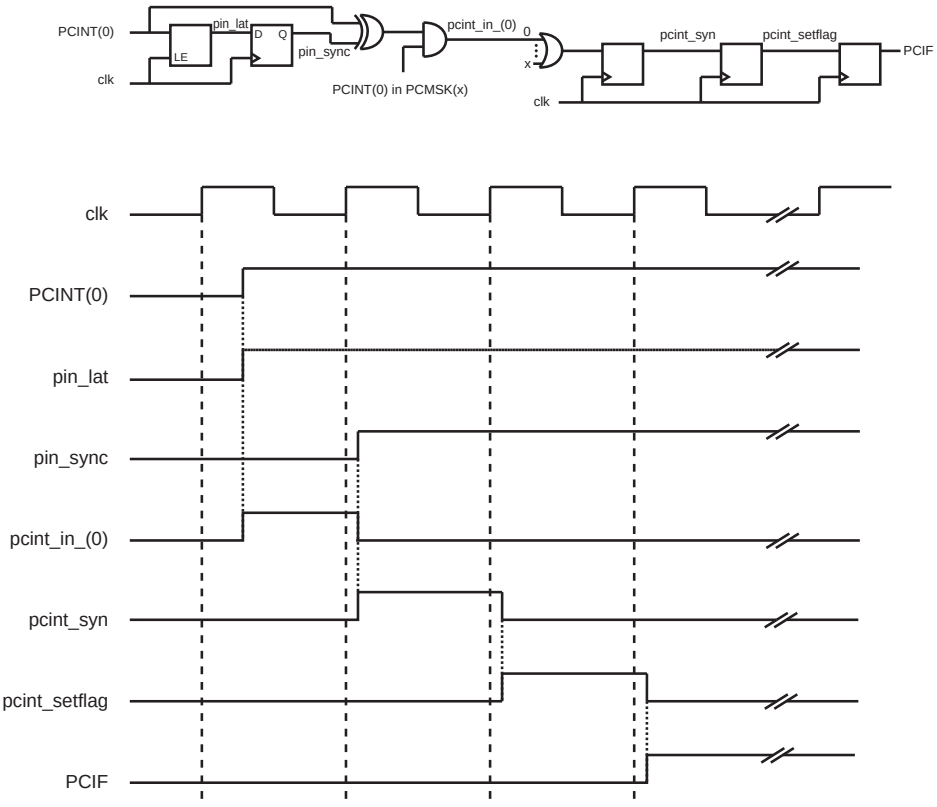
Note that if a level triggered interrupt is used for wake-up from Power-down, the required level must be held long enough for the MCU to complete the wake-up to trigger the level interrupt. If the level disappears before the end of the Start-up Time, the MCU will still wake up, but no interrupt will be generated. The start-up time is defined by the SUT and CKSEL fuses, as described in “Clock System” on page 26.

If the low level on the interrupt pin is removed before the device has woken up then program execution will not be diverted to the interrupt service routine but continue from the instruction following the SLEEP command.

9.2.2 Pin Change Interrupt Timing

A timing example of a pin change interrupt is shown in Figure 9-1.

Figure 9-1. Timing of pin change interrupts



9.3 Register Description

9.3.1 MCUCR – MCU Control Register

The External Interrupt Control Register contains control bits for interrupt sense control.

Bit	7	6	5	4	3	2	1	0	
0x35 (0x55)	PUD	SM1	SE	SM0	ISC11	ISC10	ISC01	ISC00	MCUCR
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 3, 2 – ISC11, ISC10: Interrupt Sense Control 1 Bit 1 and Bit 0**

The External Interrupt 1 is activated by the external pin INT1 if the SREG I-flag and the corresponding interrupt mask are set. The level and edges on the external INT1 pin that activate the interrupt are defined in Table 9-2. The value on the INT1 pin is sampled before detecting edges. If edge or toggle interrupt is selected, pulses that last longer than one clock period will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt. If low level interrupt is selected, the low level must be held until the completion of the currently executing instruction to generate an interrupt

Table 9-2. Interrupt 1 Sense Control

ISC11	ISC10	Description
0	0	The low level of INT1 generates an interrupt request.
0	1	Any logical change on INT1 generates an interrupt request.
1	0	The falling edge of INT1 generates an interrupt request.
1	1	The rising edge of INT1 generates an interrupt request.

- **Bits 1, 0 – ISC01, ISC00: Interrupt Sense Control 0 Bit 1 and Bit 0**

The External Interrupt 0 is activated by the external pin INT0 if the SREG I-flag and the corresponding interrupt mask are set. The level and edges on the external INT0 pin that activate the interrupt are defined in Table 9-3. The value on the INT0 pin is sampled before detecting edges. If edge or toggle interrupt is selected, pulses that last longer than one clock period will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt. If low level interrupt is selected, the low level must be held until the completion of the currently executing instruction to generate an interrupt.

Table 9-3. Interrupt 0 Sense Control

ISC01	ISC00	Description
0	0	The low level of INT0 generates an interrupt request.
0	1	Any logical change on INT0 generates an interrupt request.
1	0	The falling edge of INT0 generates an interrupt request.
1	1	The rising edge of INT0 generates an interrupt request.

9.3.2 GIMSK – General Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
0x3B (0x5B)	INT1	INT0	PCIE0	PCIE2	PCIE1	–	–	–	GIMSK
Read/Write	R/W	R/W	R/W	R/W	R/W	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 2..0 – Res: Reserved Bits**

These bits are reserved and will always read as zero.

- **Bit 7 – INT1: External Interrupt Request 1 Enable**

When the INT1 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 Control bits (ISC11 and ISC10) in the External Interrupt Control Register A (EICRA) define whether the external interrupt is activated on rising and/or falling edge of the INT1 pin or level sensed. Activity on the pin will cause an interrupt request even if INT1 is configured as an output. The corresponding interrupt of External Interrupt Request 1 is executed from the INT1 Interrupt Vector.

- **Bit 6 – 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 Control bits (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.

- **Bit 5 – 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.

- **Bit 4 – PCIE2: Pin Change Interrupt Enable 2**

When the PCIE2 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), pin change interrupt 2 is enabled. Any change on any enabled PCINT17..11 pin will cause an interrupt. The corresponding interrupt of Pin Change Interrupt Request is executed from the PCI2 Interrupt Vector. PCINT17..11 pins are enabled individually by the PCMSK2 Register.

- **Bit 3 – 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 PCINT10..8 pin will cause an interrupt. The corresponding interrupt of Pin Change Interrupt Request is executed from the PCI1 Interrupt Vector. PCINT10..8 pins are enabled individually by the PCMSK1 Register.

9.3.3 GIFR – General Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x3A (0x5A)	INTF1	INTF0	PCIF0	PCIF2	PCIF1	–	–	–	GIFR
Read/Write	R/W	R/W	R/W	R/W	R/W	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 2..0 – Res: Reserved Bits**

These bits are reserved and will always read as zero.

- **Bit 7 – 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 GIMSK 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 6 – 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 GIMSK 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.

- **Bit 5 – PCIF0: Pin Change Interrupt Flag 0**

When a logic change on any PCINT7..0 pin triggers an interrupt request, PCIF becomes set (one). If the I-bit in SREG and the PCIE0 bit in GIMSK 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.

- **Bit 4 – PCIF2: Pin Change Interrupt Flag 2**

When a logic change on any PCINT17..11 pin triggers an interrupt request, PCIF2 becomes set (one). If the I-bit in SREG and the PCIE2 bit in GIMSK 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.

- **Bit 3 – PCIF1: Pin Change Interrupt Flag 1**

When a logic change on any PCINT10..8 pin triggers an interrupt request, PCIF1 becomes set (one). If the I-bit in SREG and the PCIE1 bit in GIMSK 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.

9.3.4 PCMSK2 – Pin Change Mask Register 2

Bit	7	6	5	4	3	2	1	0	
0x05 (0x25)	–	PCINT17	PCINT16	PCINT15	PCINT14	PCINT13	PCINT12	PCINT11	PCMSK2
Read/Write	R	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 – Res: Reserved Bit**

These bits are reserved and will always read as zero.

- **Bits 6..0 – PCINT17..11: Pin Change Enable Mask 17..11**

Each PCINT17..11 bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT17..11 is set and the PCIE1 bit in GIMSK is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT17..11 is cleared, pin change interrupt on the corresponding I/O pin is disabled.

9.3.5 PCMSK1 – Pin Change Mask Register 1

Bit	7	6	5	4	3	2	1	0	
0x04 (0x24)	–	–	–	–	–	PCINT10	PCINT9	PCINT8	PCMSK1
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:3 – Res: Reserved Bits**

These bits are reserved and will always read as zero.

- **Bits 2..0 – PCINT10..8: Pin Change Enable Mask 10..8**

Each PCINT10..8 bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT10..8 is set and the PCIE1 bit in GIMSK is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT10..8 is cleared, pin change interrupt on the corresponding I/O pin is disabled.

9.3.6 PCMSK0 – Pin Change Mask Register 0

Bit	7	6	5	4	3	2	1	0	
0x20 (0x40)	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	PCMSK0
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	

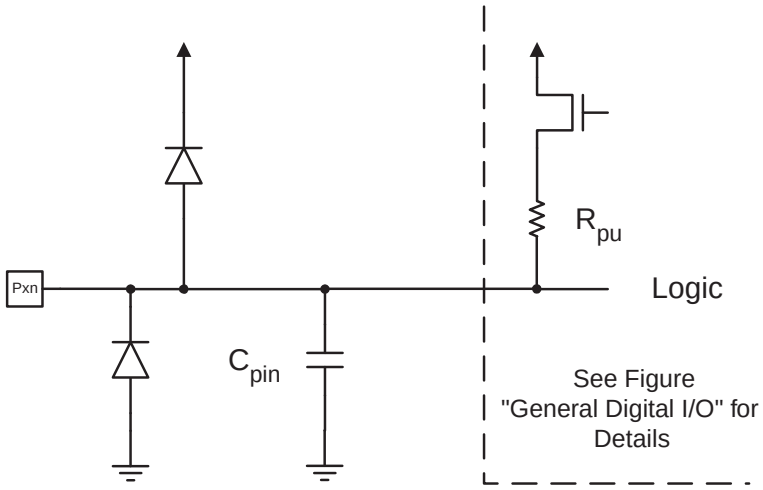
- **Bits 7..0 – PCINT7..0: Pin Change Enable Mask 7..0**

Each PCINT7..0 bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT7..0 is set and the PCIE0 bit in GIMSK is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT7..0 is cleared, pin change interrupt on the corresponding I/O pin is disabled.

10. I/O-Ports

All AVR ports have true Read-Modify-Write functionality when used as general digital I/O ports. This means that the direction of one port pin can be changed without unintentionally changing the direction of any other pin with the SBI and CBI instructions. The same applies when changing drive value (if configured as output) or enabling/disabling of pull-up resistors (if configured as input). Each output buffer has symmetrical drive characteristics with both high sink and source capability. The pin driver is strong enough to drive LED displays directly. All port pins have individually selectable pull-up resistors with a supply-voltage invariant resistance. All I/O pins have protection diodes to both V_{CC} and Ground as indicated in Figure 10-1 on page 55. See "Electrical Characteristics" on page 198 for a complete list of parameters.

Figure 10-1. I/O Pin Equivalent Schematic



All registers and bit references in this section are written in general form. A lower case "x" represents the numbering letter for the port, and a lower case "n" represents the bit number. However, when using the register or bit defines in a program, the precise form must be used. For example, PORTB3 for bit no. 3 in Port B, here documented generally as PORTxn. The physical I/O Registers and bit locations are listed in "Register Description" on page 69.

Three I/O memory address locations are allocated for each port, one each for the Data Register – PORTx, Data Direction Register – DDRx, and the Port Input Pins – PINx. The Port Input Pins I/O location is read only, while the Data Register and the Data Direction Register are read/write. However, writing a logic one to a bit in the PINx Register, will result in a toggle in the corresponding bit in the Data Register. In addition, the Pull-up Disable – PUD bit in MCUCR disables the pull-up function for all pins in all ports when set.

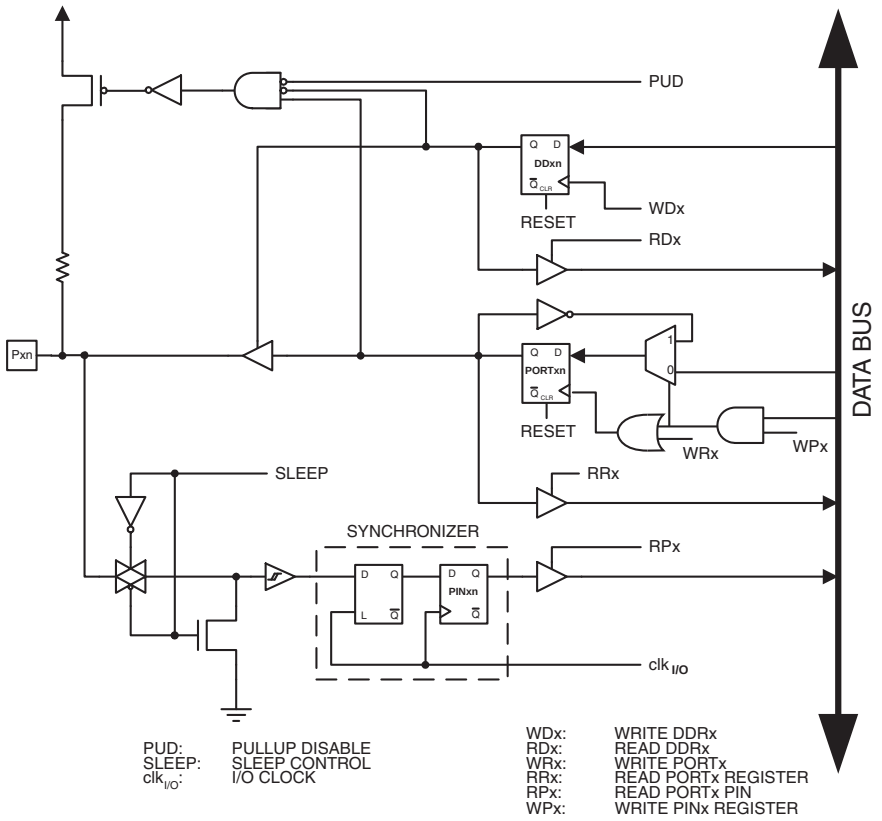
Using the I/O port as General Digital I/O is described in "Ports as General Digital I/O" on page 56. Most port pins are multiplexed with alternate functions for the peripheral features on the device. How each alternate function interferes with the port pin is described in "Alternate Port Functions" on page 60. Refer to the individual module sections for a full description of the alternate functions.

Note that enabling the alternate function of some of the port pins does not affect the use of the other pins in the port as general digital I/O.

10.1 Ports as General Digital I/O

The ports are bi-directional I/O ports with optional internal pull-ups. Figure 10-2 shows a functional description of one I/O-port pin, here generically called Pxn.

Figure 10-2. General Digital I/O[1]



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.

10.1.1 Configuring the Pin

Each port pin consists of three register bits: DDxn, PORTxn, and PINxn. As shown in "Register Description" on page 69, the DDxn bits are accessed at the DDRx I/O address, the PORTxn bits at the PORTx I/O address, and the PINxn bits at the PINx I/O address.

The DDxn bit in the DDRx Register selects the direction of this pin. If DDxn is written logic one, Pxn is configured as an output pin. If DDxn is written logic zero, Pxn is configured as an input pin.

If PORTxn is written logic one when the pin is configured as an input pin, the pull-up resistor is activated. To switch the pull-up resistor off, PORTxn has to be written logic zero or the pin has to be configured as an output pin. The port pins are tri-stated when reset condition becomes active, even if no clocks are running.

If PORTxn is written logic one when the pin is configured as an output pin, the port pin is driven high (one). If PORTxn is written logic zero when the pin is configured as an output pin, the port pin is driven low (zero).

10.1.2 Toggling the Pin

Writing a logic one to PINxn toggles the value of PORTxn, independent on the value of DDRxn. Note that the SBI instruction can be used to toggle one single bit in a port.

10.1.3 Switching Between Input and Output

When switching between tri-state ({DDxn, PORTxn} = 0b00) and output high ({DDxn, PORTxn} = 0b11), an intermediate state with either pull-up enabled {DDxn, PORTxn} = 0b01) or output low ({DDxn, PORTxn} = 0b10) must occur. Normally, the pull-up enabled state is fully acceptable, as a high-impedant environment will not notice the difference between a strong high driver and a pull-up. If this is not the case, the PUD bit in the MCUCR Register can be set to disable all pull-ups in all ports.

Switching between input with pull-up and output low generates the same problem. The user must use either the tri-state ({DDxn, PORTxn} = 0b00) or the output high state ({DDxn, PORTxn} = 0b10) as an intermediate step.

Table 10-1 summarizes the control signals for the pin value.

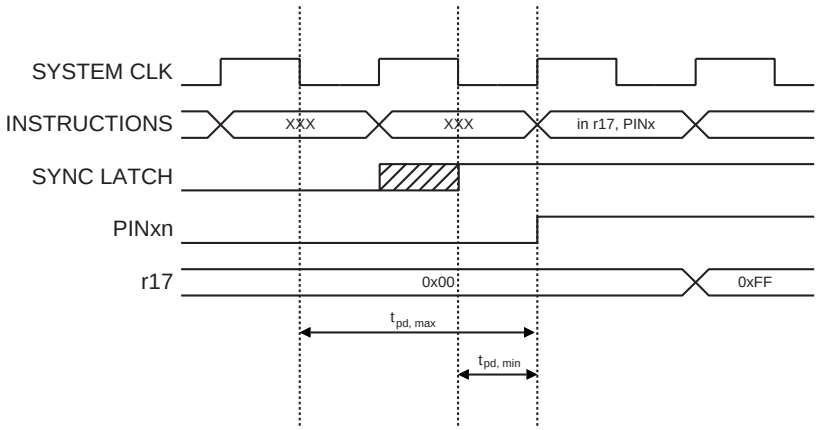
Table 10-1. Port Pin Configurations

DDxn	PORTxn	PUD (in MCUCR)	I/O	Pull-up	Comment
0	0	X	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	Pxn will source current if ext. pulled low
0	1	1	Input	No	Tri-state (Hi-Z)
1	0	X	Output	No	Output Low (Sink)
1	1	X	Output	No	Output High (Source)

10.1.4 Reading the Pin Value

Independent of the setting of Data Direction bit DDxn, the port pin can be read through the PINxn Register bit. As shown in Figure 10-2 on page 56, the PINxn Register bit and the preceding latch constitute a synchronizer. This is needed to avoid metastability if the physical pin changes value near the edge of the internal clock, but it also introduces a delay. Figure 10-3 shows a timing diagram of the synchronization when reading an externally applied pin value. The maximum and minimum propagation delays are denoted $t_{pd,max}$ and $t_{pd,min}$ respectively.

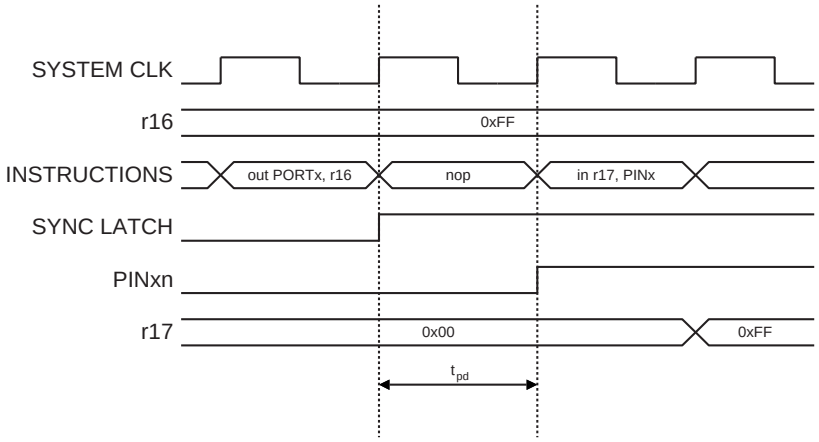
Figure 10-3. Synchronization when Reading an Externally Applied Pin value



Consider the clock period starting shortly after the first falling edge of the system clock. The latch is closed when the clock is low, and goes transparent when the clock is high, as indicated by the shaded region of the “SYNC LATCH” signal. The signal value is latched when the system clock goes low. It is clocked into the PINxn Register at the succeeding positive clock edge. As indicated by the two arrows tpd,max and tpd,min, a single signal transition on the pin will be delayed between ½ and 1½ system clock period depending upon the time of assertion.

When reading back a software assigned pin value, a nop instruction must be inserted as indicated in Figure 10-4 on page 58. 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 one system clock period.

Figure 10-4. Synchronization when Reading a Software Assigned Pin Value



10.1.5 Digital Input Enable and Sleep Modes

As shown in Figure 10-2 on page 56, the digital input signal can be clamped to ground at the input of the schmitt-trigger. The signal denoted SLEEP in the figure, is set by the MCU Sleep Controller in Power-down and Standby modes to avoid high power consumption if some input signals are left floating, or have an analog signal level close to $V_{CC}/2$.

SLEEP is overridden for port pins enabled as external interrupt pins. If the external interrupt request is not enabled, SLEEP is active also for these pins. SLEEP is also overridden by various other alternate functions as described in “Alternate Port Functions” on page 60.

If a logic high level (“one”) is present on an asynchronous external interrupt pin configured as “Interrupt on Rising Edge, Falling Edge, or Any Logic Change on Pin” while the external interrupt is *not* enabled, the corresponding External Interrupt Flag will be set when resuming from the above mentioned Sleep mode, as the clamping in these sleep mode produces the requested logic change.

10.1.6 Unconnected Pins

If some pins are unused, it is recommended to ensure that these pins have a defined level. Even though most of the digital inputs are disabled in the deep sleep modes as described above, floating inputs should be avoided to reduce current consumption in all other modes where the digital inputs are enabled (Reset, Active mode and Idle mode).

The simplest method to ensure a defined level of an unused pin, is to enable the internal pull-up. In this case, the pull-up will be disabled during reset. If low power consumption during reset is

important, it is recommended to use an external pull-up or pulldown. Connecting unused pins directly to V_{CC} or GND is not recommended, since this may cause excessive currents if the pin is accidentally configured as an output.

10.1.7 Program Examples

The following code example shows how to set port A pins 0 and 1 high, 2 and 3 low, and define the port pins from 4 to 7 as input with a pull-up assigned to port pin 4. 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

```
...
; Define pull-ups and set outputs high
; Define directions for port pins
ldi  r16,(1<<PB4)|(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
...
```

Note: Two temporary registers are used to minimize the time from pull-ups are set on pins 0, 1 and 4, until the direction bits are correctly set, defining bit 2 and 3 as low and redefining bits 0 and 1 as strong high drivers.

C Code Example[1]

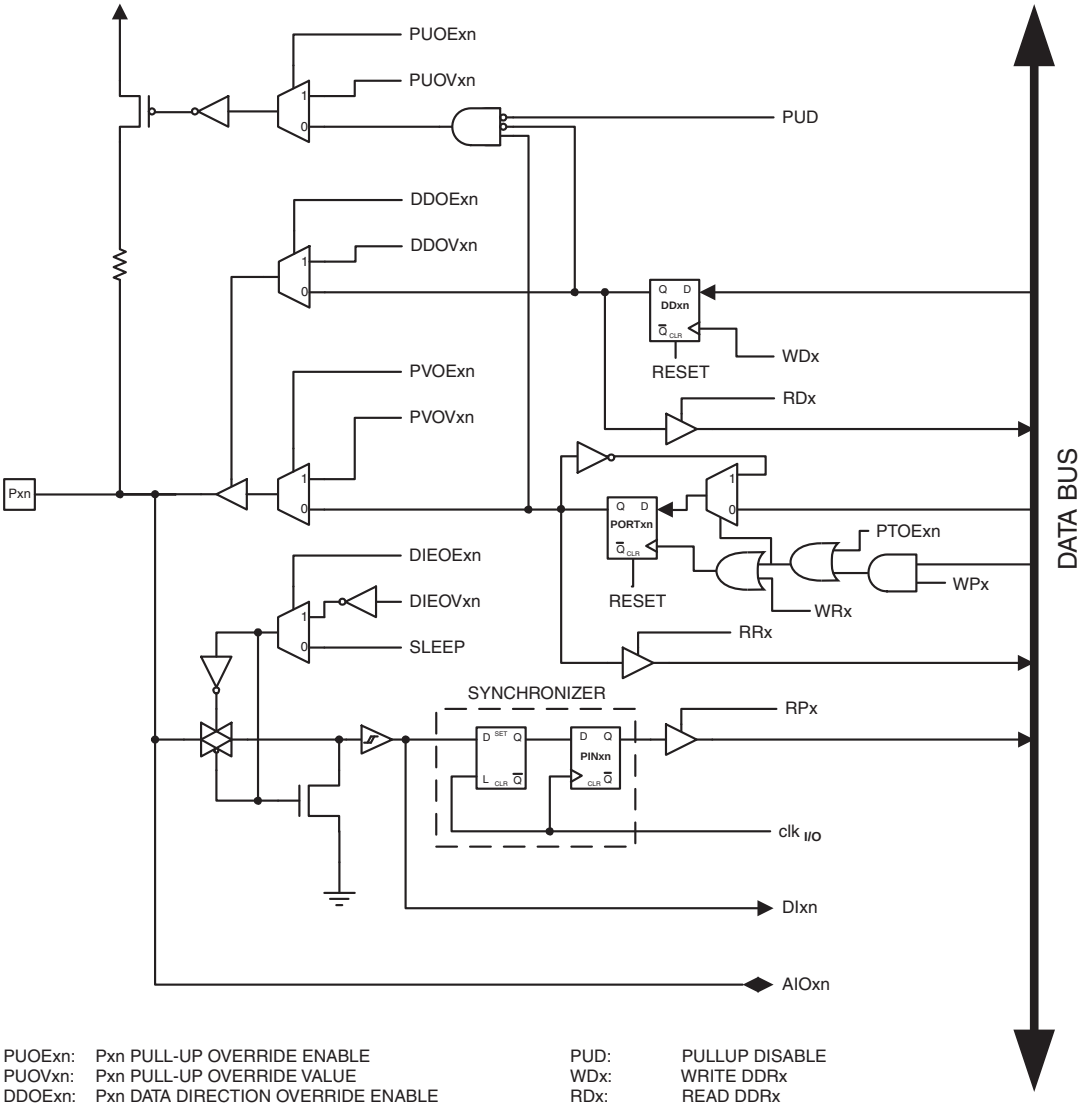
```
unsigned char i;
  ...
  /* Define pull-ups and set outputs high */
  /* Define directions for port pins */
  PORTB = (1<<PB4)|(1<<PB1)|(1<<PB0);
  DDRB = (1<<DDB3)|(1<<DDB2)|(1<<DDB1)|(1<<DDB0);
  /* Insert nop for synchronization*/
  _NOP();
  /* Read port pins */
  i = PINB;
  ...
```

Note: 1. See “Code Examples” on page 7.

10.2 Alternate Port Functions

Most port pins have alternate functions in addition to being general digital I/Os. In Figure 10-5 below is shown how the port pin control signals from the simplified Figure 10-2 on page 56 can be overridden by alternate functions.

Figure 10-5. Alternate Port Functions[1]



PUOExn:	Pxn PULL-UP OVERRIDE ENABLE	PUD:	PULLUP 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.

The illustration in the figure above serves as a generic description applicable to all port pins in the AVR microcontroller family. Some overriding signals may not be present in all port pins.

Table 10-2 summarizes the function of the overriding signals. The pin and port indexes from Figure 10-5 are not shown in the succeeding tables. The overriding signals are generated internally in the modules having the alternate function.

Table 10-2. Generic Description of Overriding Signals for Alternate Functions

Signal Name	Full Name	Description
PUOE	Pull-up Override Enable	If this signal is set, the pull-up enable is controlled by the PUOV signal. If this signal is cleared, the pull-up is enabled when {DDxn, PORTxn, PUD} = 0b010.
PUOV	Pull-up Override Value	If PUOE is set, the pull-up is enabled/disabled when PUOV is set/cleared, regardless of the setting of the DDxn, PORTxn, and PUD Register bits.
DDOE	Data Direction Override Enable	If this signal is set, the Output Driver Enable is controlled by the DDOV signal. If this signal is cleared, the Output driver is enabled by the DDxn Register bit.
DDOV	Data Direction Override Value	If DDOE is set, the Output Driver is enabled/disabled when DDOV is set/cleared, regardless of the setting of the DDxn Register bit.
PVOE	Port Value Override Enable	If this signal is set and the Output Driver is enabled, the port value is controlled by the PVOV signal. If PVOE is cleared, and the Output Driver is enabled, the port Value is controlled by the PORTxn Register bit.
PVOV	Port Value Override Value	If PVOE is set, the port value is set to PVOV, regardless of the setting of the PORTxn Register bit.
PTOE	Port Toggle Override Enable	If PTOE is set, the PORTxn Register bit is inverted.
DIEOE	Digital Input Enable Override Enable	If this bit is set, the Digital Input Enable is controlled by the DIEOV signal. If this signal is cleared, the Digital Input Enable is determined by MCU state (Normal mode, sleep mode).
DIEOV	Digital Input Enable Override Value	If DIEOE is set, the Digital Input is enabled/disabled when DIEOV is set/cleared, regardless of the MCU state (Normal mode, sleep mode).
DI	Digital Input	This is the Digital Input to alternate functions. In the figure, the signal is connected to the output of the schmitt-trigger but before the synchronizer. Unless the Digital Input is used as a clock source, the module with the alternate function will use its own synchronizer.
AIO	Analog Input/Output	This is the Analog Input/Output to/from alternate functions. The signal is connected directly to the pad, and can be used bi-directionally.

The following subsections shortly describe the alternate functions for each port, and relate the overriding signals to the alternate function. Refer to the alternate function description for further details.

10.2.1 Alternate Functions of Port A

The Port A pins with alternate function are shown in Table 10-3.

Table 10-3. Port A Pins Alternate Functions

Port Pin	Alternate Function
PA0	XTAL1: Crystal Oscillator Input CLKI: External Clock Input PCINT8: Pin Change Interrupt 1, Source 8
PA1	XTAL2: Crystal Oscillator Output PCINT9: Pin Change Interrupt 1, Source 9
PA2	$\overline{\text{RESET}}$: Reset pin dW: debugWire I/O PCINT10: Pin Change Interrupt 1, Source 10

- **Port A, Bit 0 – XTAL1/CLKI/PCINT8**
  - XTAL1: Chip Clock Oscillator pin 1. Used for all chip clock sources except internal calibratable RC oscillator. When used as a clock pin, the pin can not be used as an I/O pin. When using internal calibratable RC Oscillator as a chip clock source, PA0 serves as an ordinary I/O pin.
  - CLKI: Clock Input from an external clock source, see “External Clock” on page 27.
  - PCINT8: Pin Change Interrupt source 8. The PA0 pin can serve as an external interrupt source for pin change interrupt 1.

- **Port A, Bit 1 – XTAL2/PCINT9**
  - XTAL2: Chip Clock Oscillator pin 2. Used as clock pin for all chip clock sources except internal calibratable RC Oscillator and external clock. When used as a clock pin, the pin can not be used as an I/O pin. When using internal calibratable RC Oscillator or External clock as a Chip clock sources, PA1 serves as an ordinary I/O pin.
  - PCINT9: Pin Change Interrupt source 9. The PA1 pin can serve as an external interrupt source for pin change interrupt 1.

- **Port A, Bit 2 – $\overline{\text{RESET}}$/dW/PCINT10**
  - $\overline{\text{RESET}}$: External $\overline{\text{Reset}}$ input is active low and enabled by unprogramming (“1”) the RSTDISBL Fuse. Pullup is activated and output driver and digital input are deactivated when the pin is used as the $\overline{\text{RESET}}$ pin.
  - dW: When the debugWIRE Enable (DWEN) Fuse is programmed and Lock bits are unprogrammed, the debugWIRE system within the target device is activated. The RESET port pin is configured as a wire-AND (open-drain) bi-directional I/O pin with pull-up enabled and becomes the communication gateway between target and emulator.
  - PCINT10: Pin Change Interrupt source 10. The PA2 pin can serve as an external interrupt source for pin change interrupt 1.

Table 10-4 relates the alternate functions of Port A to the overriding signals shown in Figure 10-5 on page 60.

Table 10-4. Overriding Signals for Alternate Functions in PA2..PA0

Signal Name	PA2/RESET/dW/PCINT10	PA1/XTAL2/PCINT9	PA0/XTAL1/PCINT8
PUOE	$\overline{\text{RSTDISBL}}$[1] + DEBUGWIRE_ENABLE[2]	EXT_OSC[3]	EXT_CLOCK[4] + EXT_OSC[3]
PUOV	1	0	0
DDOE	$\overline{\text{RSTDISBL}}$[1] + DEBUGWIRE_ENABLE[2]	EXT_OSC[3]	EXT_CLOCK[4] + EXT_OSC[3]
DDOV	DEBUGWIRE_ENABLE[2] • $\overline{\text{debugWire Transmit}}$	0	0
PVOE	$\overline{\text{RSTDISBL}}$[1] + DEBUGWIRE_ENABLE[2]	EXT_OSC[3]	EXT_CLOCK[4] + EXT_OSC[3]
PVOV	0	0	0
PTOE	0	0	0
DIEOE	$\overline{\text{RSTDISBL}}$[1] + DEBUGWIRE_ENABLE[2] + PCINT10 • PCIE1	EXT_OSC[3] + PCINT9 • PCIE1	EXT_CLOCK[4] + EXT_OSC[3] + (PCINT8 • PCIE1)
DIEOV	DEBUGWIRE_ENABLE[2] + ($\overline{\text{RSTDISBL}}$[1] • PCINT10 • PCIE1)	$\overline{\text{EXT\_OSC}}$[3] + PCINT9 • PCIE1	(EXT_CLOCK[4] • $\overline{\text{PWR\_DOWN}}$) + ($\overline{\text{EXT\_CLOCK}}$[4] • $\overline{\text{EXT\_OSC}}$[3] • PCINT8 • PCIE1)
DI	dW/PCINT10 Input	PCINT9 Input	CLKI/PCINT8 Input
AIO		XTAL2	XTAL1

Notes:
1. RSTDISBL is 1 when the fuse is “0” (Programmed).
2. DebugWIRE is enabled when DWEN Fuse is programmed and Lock bits are unprogrammed.
3. EXT_OSC = crystal oscillator or low frequency crystal oscillator is selected as system clock.
4. EXT_CLOCK = external closk is selected as system clock.

10.2.2 Alternate Functions of Port B

The Port B pins with alternate function are shown in Table 10-5.

Table 10-5. Port B Pins Alternate Functions

Port Pin	Alternate Function
PB0	AIN0: Analog Comparator, Positive Input PCINT0: Pin Change Interrupt 0, Source 0
PB1	AIN1: Analog Comparator, Negative Input PCINT1: Pin Change Interrupt 0, Source 1
PB2	OC0A:: Timer/Counter0 Compare Match AOutput PCINT2: Pin Change Interrupt 0, Source 2
PB3	OC1A: Timer/Counter1 Compare Match A Output PCINT3: Pin Change Interrupt 0, Source 3

Table 10-5. Port B Pins Alternate Functions

Port Pin	Alternate Function
PB4	OC1B: Timer/Counter1 Compare Match B Output PCINT4: Pin Change Interrupt 0, Source 4
PB5	DI: USI Data Input (Three Wire Mode) SDA: USI Data Input (Two Wire Mode) PCINT5: Pin Change Interrupt 0, Source 5
PB6	DO: USI Data Output (Three Wire Mode) PCINT6: Pin Change Interrupt 0, Source 6
PB7	USCK: USI Clock (Three Wire Mode) SCL : USI Clock (Two Wire Mode) PCINT7: Pin Change Interrupt 0, Source 7

- **Port B, Bit 0 – AIN0/PCINT0**
  - AIN0: Analog Comparator Positive input. Configure the port pin as input with the internal pull-up switched off to avoid the digital port function from interfering with the function of the Analog Comparator.
  - PCINT0: Pin Change Interrupt Source 0. The PB0 pin can serve as an external interrupt source for pin change interrupt 0.

- **Port B, Bit 1 – AIN1/PCINT1**
  - AIN1: Analog Comparator Negative input. Configure the port pin as input with the internal pull-up switched off to avoid the digital port function from interfering with the function of the analog comparator.
  - PCINT1: Pin Change Interrupt Source 1. The PB1 pin can serve as an external interrupt source for pin change interrupt 0.

- **Port B, Bit 2 – OC0A/PCINT2**
  - OC0A: Output Compare Match A output. The PB2 pin can serve as an external output for the Timer/Counter0 Output Compare A. The pin has to be configured as an output (DDB2 set (one)) to serve this function. The OC0A pin is also the output pin for the PWM mode timer function.
  - PCINT2: Pin Change Interrupt Source 2. The PB2 pin can serve as an external interrupt source for pin change interrupt 0.

- **Port B, Bit 3 – OC1A/PCINT3**
  - OC1A: Output Compare Match A output: The PB3 pin can serve as an external output for the Timer/Counter1 Output Compare A. The pin has to be configured as an output (DDB3 set (one)) to serve this function. The OC1A pin is also the output pin for the PWM mode timer function.
  - PCINT3: Pin Change Interrupt Source 3: The PB3 pin can serve as an external interrupt source for pin change interrupt 0.

- **Port B, Bit 4 – OC1B/PCINT4**
  - OC1B: Output Compare Match B output: The PB4 pin can serve as an external output for the Timer/Counter1 Output Compare B. The pin has to be configured as an output (DDB4 set

(one)) to serve this function. The OC1B pin is also the output pin for the PWM mode timer function.

- PCINT4: Pin Change Interrupt Source 4. The PB4 pin can serve as an external interrupt source for pin change interrupt 0.

- **Port B, Bit 5 – DI/SDA/PCINT5**
  - DI: Three-wire mode Universal Serial Interface Data input. Three-wire mode does not override normal port functions, so pin must be configured as an input. SDA: Two-wire mode Serial Interface Data.
  - PCINT5: Pin Change Interrupt Source 5. The PB5 pin can serve as an external interrupt source for pin change interrupt 0.

- **Port B, Bit 6 – DO/PCINT6**
  - DO: Three-wire mode Universal Serial Interface Data output. Three-wire mode Data output overrides PORTB6 value and it is driven to the port when data direction bit DDB6 is set (one). However the PORTB6 bit still controls the pull-up enabling pull-up, if direction is input and PORTB6 is set (one).
  - PCINT6: Pin Change Interrupt Source 6. The PB6 pin can serve as an external interrupt source for pin change interrupt 0.

- **Port B, Bit 7 – USCK/SCL/PCINT7**
  - USCK: Three-wire mode Universal Serial Interface Clock.
  - SCL: Two-wire mode Serial Clock for USI Two-wire mode.
  - PCINT7: Pin Change Interrupt source 7. The PB7 pin can serve as an external interrupt source for pin change interrupt 0.

Table 10-6 and Table 10-7 relate the alternate functions of Port B to the overriding signals shown in Figure 10-5 on page 60. SPI MSTR INPUT and SPI SLAVE OUTPUT constitute the MISO signal, while MOSI is divided into SPI MSTR OUTPUT and SPI SLAVE INPUT.

Table 10-6. Overriding Signals for Alternate Functions in PB7..PB4

Signal Name	PB7/USCK/ SCL/PCINT7	PB6/DO/PCINT6	PB5/SDA/ DI/PCINT5	PB4/OC1B/ PCINT4
PUOE	USI_TWO_WIRE	0	0	0
PUOV	0	0	0	0
DDOE	USI_TWO_WIRE	0	USI_TWO_WIRE	0
DDOV	(USI_SCL_HOLD+ $\overline{\text{PORTB7}}$)•DDB7	0	($\overline{\text{SDA}}$ + $\overline{\text{PORTB5}}$)• DDB5	0
PVOE	USI_TWO_WIRE • DDB7	USI_THREE_WIRE	USI_TWO_WIRE • DDB5	OC1B_PVOE
PVOV	0	DO	0	0OC1B_PVOV
PTOE	USI_PTOE	0	0	0
DIEOE	(PCINT7•PCIE) +USISIE	(PCINT6•PCIE)	(PCINT5•PCIE) + USISIE	(PCINT4•PCIE)
DIEOV	1	1	1	1
DI	PCINT7 Input USCK Input SCL Input	PCINT6 Input	PCINT5 Input SDA Input DI Input	PCINT4 Input
AIO	–	–	–	–

Table 10-7. Overriding Signals for Alternate Functions in PB3..PB0

Signal Name	PB3/OC1A/ PCINT3	PB2/OC0A/ PCINT2	PB1/AIN1/ PCINT1	PB0/AIN0/ PCINT0
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	OC1A_PVOE	OC0A_PVOE	0	0
PVOV	OC1A_PVOV	OC0A_PVOV	0	0
PTOE	0	0	0	0
DIEOE	(PCINT3 • PCIE)	(PCINT2 • PCIE)	(PCINT1 • PCIE)	(PCINT0 • PCIE)
DIEOV	1	1	1	1
DI	PCINT7 Input	PCINT6 Input	PCINT5 Input	PCINT4 Input
AIO	–	–	AIN1	AIN0

10.2.3 Alternate Functions of Port D

The Port D pins with alternate functions are shown in Table 10-8..

Table 10-8. Port D Pins Alternate Functions

Port Pin	Alternate Function
PD0	RXD: UART Data Receiver PCINT11:Pin Change Interrupt 2, Source 11
PD1	TXD: UART Data Transmitter PCINT12:Pin Change Interrupt 2, Source 12
PD2	INT0: External Interrupt 0 Input XCK: USART Transfer Clock CKOUT: System Clock Output PCINT13:Pin Change Interrupt 2, Source 13
PD3	INT1: External Interrupt 1 Input PCINT14:Pin Change Interrupt 2, Source 14
PD4	T0: Timer/Counter0 Clock Source PCINT15:Pin Change Interrupt 2, Source 15
PD5	OC0B: Timer/Counter0 Compare Match B output T1: Timer/Counter1 Clock Source PCINT16:Pin Change Interrupt 2, Source 16
PD6	ICPI: Timer/Counter1 Input Capture Pin PCINT17:Pin Change Interrupt 2, Source 17

- **Port D, Bit 0 – RXD/PCINT11**
  - RXD: UART Data Receiver.
  - PCINT11: Pin Change Interrupt Source 11. The PD0 pin can serve as an external interrupt source for pin change interrupt 2.
- **Port D, Bit 1 – TXD/PCINT12**
  - TXD: UART Data Transmitter.
  - PCINT12: Pin Change Interrupt Source 12. The PD1 pin can serve as an external interrupt source for pin change interrupt 2.
- **Port D, Bit 2 – INT0/XCK/CKOUT/PCINT13**
  - INT0: External Interrupt Source 0. The PD2 pin can serve as en external interrupt source to the MCU.
  - XCK: USART Transfer Clock used only by Synchronous Transfer mode.
  - CKOUT: System Clock Output.
  - PCINT13: Pin Change Interrupt Source 13. The PD2 pin can serve as an external interrupt source for pin change interrupt 2.
- **Port D, Bit 3 – INT1/PCINT14**
  - INT1: External Interrupt Source 1. The PD3 pin can serve as an external interrupt source to the MCU.
  - PCINT14: Pin Change Interrupt Source 14. The PD3 pin can serve as an external interrupt source for pin change interrupt 2.

- **Port D, Bit 4 – T0/PCINT15**
  - T0: Timer/Counter0 External Counter Clock input is enabled by setting (one) the bits CS02 and CS01 in the Timer/Counter0 Control Register (TCCR0).
  - PCINT15: Pin Change Interrupt Source 15. The PD4 pin can serve as an external interrupt source for pin change interrupt 2.

- **Port D, Bit 5 – OC0B/T1/PCINT16**
  - OC0B: Output Compare Match B output: The PD5 pin can serve as an external output for the Timer/Counter0 Output Compare B. The pin has to be configured as an output (DDD5 set (one)) to serve this function. The OC0B pin is also the output pin for the PWM mode timer function.
  - T1: Timer/Counter1 External Counter Clock input is enabled by setting (one) the bits CS02 and CS01 in the Timer/Counter1 Control Register (TCCR1).
  - PCINT16: Pin Change Interrupt Source 16. The PD5 pin can serve as an external interrupt source for pin change interrupt 2.

- **Port D, Bit 6 – ICPI/PCINT17**
  - ICPI: Timer/Counter1 Input Capture Pin. The PD6 pin can act as an Input Capture pin for Timer/Counter1.
  - PCINT17: Pin Change Interrupt Source 17. The PD6 pin can serve as an external interrupt source for pin change interrupt 2.

Table 10-9 and Table 10-10 relates the alternate functions of Port D to the overriding signals shown in Figure 10-5 on page 60.

Table 10-9. Overriding Signals for Alternate Functions PD6..PD4

Signal Name	PD6/ICPI/PCINT17	PD5/OC1B/T1/PCINT16	PD4/T0/PCINT15
PUOE	0	0	0
PUOV	0	0	0
DDOE	0	0	0
DDOV	0	0	0
PVOE	0	OC1B_PVOE	0
PVOV	0	OC1B_PVOV	0
PTOE	0	0	0
DIEOE	ICPI Enable + PCINT17	T1 Enable + PCINT16	T0 Enable + PCINT15
DIEOV	PCINT17	PCINT16	PCINT15
DI	ICPI Input/PCINT17	T1 Input/PCINT16	T0 Input/PCINT15
AIO	–	–	AIN1

Table 10-10. Overriding Signals for Alternate Functions in PD3..PD0

Signal Name	PD3/INT1/ PCINT14	PD2/INT0/XCK/CKOUT/ PCINT13	PD1/TXD/ PCINT12	PD0/RXD/PCINT11
PUOE	0	0	TXD_OE	RXD_OE
PUOV	0	0	0	PORTD0 • $\overline{\text{PUD}}$
DDOE	0	0	TXD_OE	RXD_EN
DDOV	0	0	1	0
PVOE	0	XCKO_PVOE	TXD_OE	0
PVOV	0	XCKO_PVOV	TXD_PVOV	0
PTOE	0	0	0	0
DIEOE	INT1 Enable + PCINT14	INT0 Enable/ XCK Input Enable/PCINT13	PCINT12	PCINT11
DIEOV	PCINT14	PCINT13	PCINT12	PCINT11
DI	INT1 Input/ PCINT14	INT0 Input/XCK Input/ PCINT13	PCINT12	RXD Input/PCINT11
AIO	–	–	–	–

10.3 Register Description

10.3.1 MCUCR – MCU Control Register

Bit	7	6	5	4	3	2	1	0	
0x35 (0x55)	**PUD**	**SM1**	**SE**	**SM0**	**ISC11**	**ISC10**	**ISC01**	**ISC00**	MCUCR
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 – PUD: Pull-up Disable**

When this bit is written to one, the pull-ups in the I/O ports are disabled even if the DDxn and PORTxn Registers are configured to enable the pull-ups ({DDxn, PORTxn} = 0b01). See “Configuring the Pin” on page 56 for more details about this feature.

10.3.2 PORTA – Port A Data Register

Bit	7	6	5	4	3	2	1	0	
0x1B (0x3B)	–	–	–	–	–	**PORTA2**	**PORTA1**	**PORTA0**	**PORTA**
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

10.3.3 DDRA – Port A Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x1A (0x3A)	–	–	–	–	–	**DDA2**	**DDA1**	**DDA0**	**DDRA**
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

10.3.4 PINA – Port A Input Pins Address

Bit	7	6	5	4	3	2	1	0	
0x19 (0x39)	–	–	–	–	–	PINA2	PINA1	PINA0	PINA
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

10.3.5 PORTB – Port B Data Register

Bit	7	6	5	4	3	2	1	0	
0x18 (0x38)	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
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	

10.3.6 DDRB – Port B Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x17 (0x37)	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
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	

10.3.7 PINB – Port B Input Pins Address

Bit	7	6	5	4	3	2	1	0	
0x16 (0x36)	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

10.3.8 PORTD – Port D Data Register

Bit	7	6	5	4	3	2	1	0	
0x12 (0x32)	–	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	PORTD
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

10.3.9 DDRD – Port D Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x11 (0x31)	–	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	DDRD
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

10.3.10 PIND – Port D Input Pins Address

Bit	7	6	5	4	3	2	1	0	
0x10 (0x30)	–	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	PIND
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

11. 8-bit Timer/Counter0 with PWM

11.1 Features

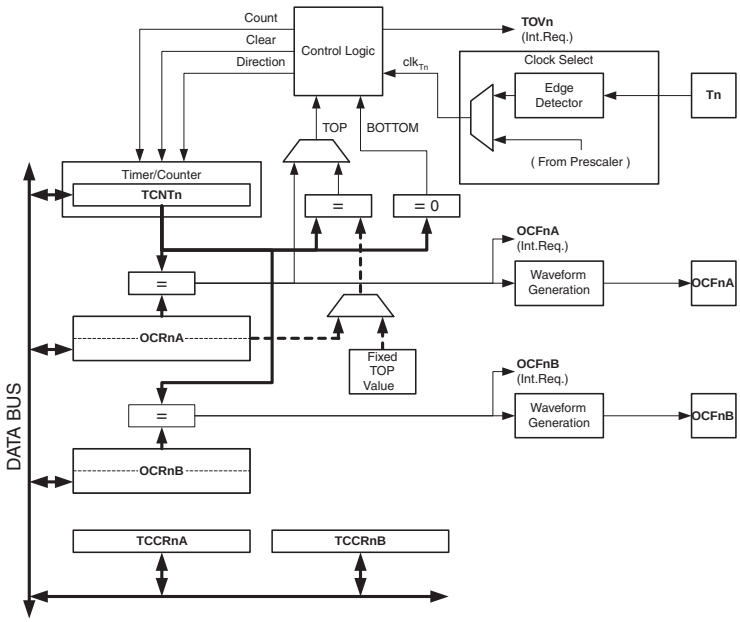
- **Two Independent Output Compare Units**
- **Double Buffered Output Compare Registers**
- **Clear Timer on Compare Match (Auto Reload)**
- **Glitch Free, Phase Correct Pulse Width Modulator (PWM)**
- **Variable PWM Period**
- **Frequency Generator**
- **Three Independent Interrupt Sources (TOV0, OCF0A, and OCF0B)**

11.2 Overview

Timer/Counter0 is a general purpose 8-bit Timer/Counter module, with two independent Output Compare Units, and with PWM support. It allows accurate program execution timing (event management) and wave generation.

A simplified block diagram of the 8-bit Timer/Counter is shown in Figure 11-1. For the actual placement of I/O pins, refer to “Pinout ATtiny2313A/4313” on page 2. CPU accessible I/O Registers, including I/O bits and I/O pins, are shown in bold. The device-specific I/O Register and bit locations are listed in the “Register Description” on page 82.

Figure 11-1. 8-bit Timer/Counter Block Diagram



11.2.1 Registers

The Timer/Counter (TCNT0) and Output Compare Registers (OCR0A and OCR0B) are 8-bit registers. Interrupt request (abbreviated to Int.Req. in the figure) signals are all visible in the Timer Interrupt Flag Register (TIFR). All interrupts are individually masked with the Timer Interrupt Mask Register (TIMSK). TIFR and TIMSK are not shown in the figure.

The Timer/Counter can be clocked internally, via the prescaler, or by an external clock source on the T0 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_{T0}).

The double buffered Output Compare Registers (OCR0A and OCR0B) is compared with the Timer/Counter value at all times. 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 (OC0A and OC0B). See “Output Compare Unit” on page 73. for details. The Compare Match event will also set the Compare Flag (OCF0A or OCF0B) which can be used to generate an Output Compare interrupt request.

11.2.2 Definitions

Many register and bit references in this section are written in general form. A lower case “n” replaces the Timer/Counter number, in this case 0. A lower case “x” replaces the Output Compare Unit, in this case Compare Unit A or Compare Unit B. However, when using the register or bit defines in a program, the precise form must be used, i.e., TCNT0 for accessing Timer/Counter0 counter value and so on.

The definitions in Table 11-1 are also used extensively throughout the document.

Table 11-1. Definitions

Constant	Description
BOTTOM	The counter reaches BOTTOM when it becomes 0x00
MAX	The counter reaches its MAXimum when it becomes 0xFF (decimal 255)
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 the fixed value 0xFF (MAX) or the value stored in the OCR0A Register. The assignment depends on the mode of operation

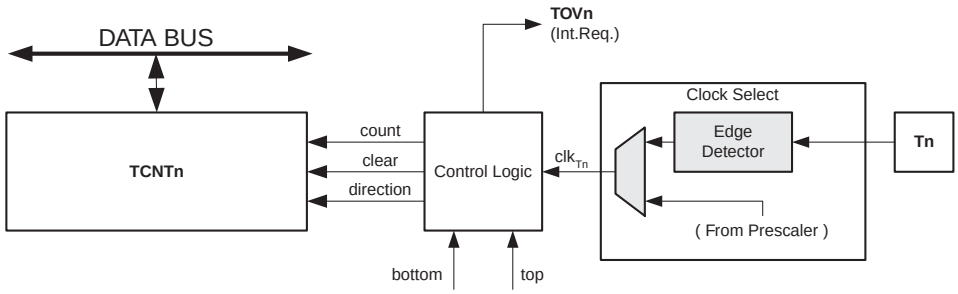
11.3 Clock Sources

The Timer/Counter can be clocked by an internal or an external clock source. The clock source is selected by the Clock Select logic which is controlled by the Clock Select (CS02:0) bits located in the Timer/Counter Control Register (TCCR0B). For details on clock sources and prescaler, see “Timer/Counter0 and Timer/Counter1 Prescalers” on page 118.

11.4 Counter Unit

The main part of the 8-bit Timer/Counter is the programmable bi-directional counter unit. Figure 11-2 shows a block diagram of the counter and its surroundings.

Figure 11-2. Counter Unit Block Diagram



Signal description (internal signals):

count	Increment or decrement TCNT0 by 1.
direction	Select between increment and decrement.
clear	Clear TCNT0 (set all bits to zero).
clk_{Tn}	Timer/Counter clock, referred to as clk_{T0} in the following.
top	Signalize that TCNT0 has reached maximum value.
bottom	Signalize that TCNT0 has reached minimum value (zero).

Depending of the mode of operation used, the counter is cleared, incremented, or decremented at each timer clock (clk_{T0}). clk_{T0} can be generated from an external or internal clock source, selected by the Clock Select bits (CS02:0). When no clock source is selected (CS02:0 = 0) the timer is stopped. However, the TCNT0 value can be accessed by the CPU, regardless of whether clk_{T0} is present or not. A CPU write overrides (has priority over) all counter clear or count operations.

The counting sequence is determined by the setting of the WGM01 and WGM00 bits located in the Timer/Counter Control Register (TCCR0A) and the WGM02 bit located in the Timer/Counter Control Register B (TCCR0B). There are close connections between how the counter behaves (counts) and how waveforms are generated on the Output Compare output OC0A. For more details about advanced counting sequences and waveform generation, see “Modes of Operation” on page 97.

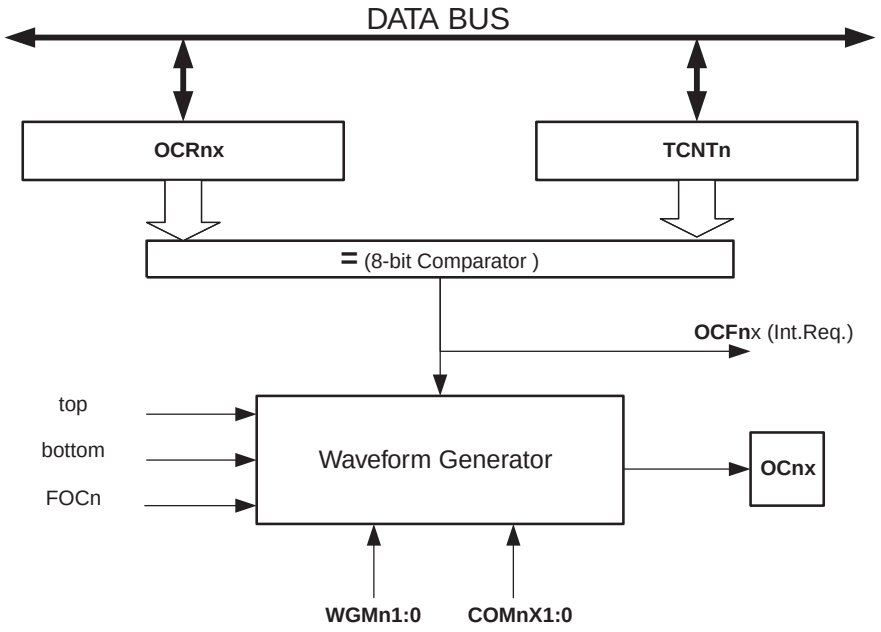
The Timer/Counter Overflow Flag (TOV0) is set according to the mode of operation selected by the WGM01:0 bits. TOV0 can be used for generating a CPU interrupt.

11.5 Output Compare Unit

The 8-bit comparator continuously compares TCNT0 with the Output Compare Registers (OCR0A and OCR0B). Whenever TCNT0 equals OCR0A or OCR0B, the comparator signals a match. A match will set the Output Compare Flag (OCF0A or OCF0B) at the next timer clock cycle. If the corresponding interrupt is enabled, the Output Compare Flag generates an Output Compare interrupt. The Output Compare Flag is automatically cleared when the interrupt is executed. Alternatively, the 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 WGM02:0 bits and Compare Output mode (COM0x1:0) 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 (see “Modes of Operation” on page 97).

Figure 11-3 shows a block diagram of the Output Compare unit.

Figure 11-3. Output Compare Unit, Block Diagram



The OCR0x Registers are 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 OCR0x Compare Registers 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 OCR0x Register access may seem complex, but this is not case. When the double buffering is enabled, the CPU has access to the OCR0x Buffer Register, and if double buffering is disabled the CPU will access the OCR0x directly.

11.5.1 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 (FOC0x) bit. Forcing Compare Match will not set the OCF0x Flag or reload/clear the timer, but the OC0x pin will be updated as if a real Compare Match had occurred (the COM0x1:0 bits settings define whether the OC0x pin is set, cleared or toggled).

11.5.2 Compare Match Blocking by TCNT0 Write

All CPU write operations to the TCNT0 Register will block any Compare Match that occur in the next timer clock cycle, even when the timer is stopped. This feature allows OCR0x to be initialized to the same value as TCNT0 without triggering an interrupt when the Timer/Counter clock is enabled.

11.5.3 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 Unit, independently of whether the Timer/Counter is running or not. If the value written to TCNT0 equals the OCR0x 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.

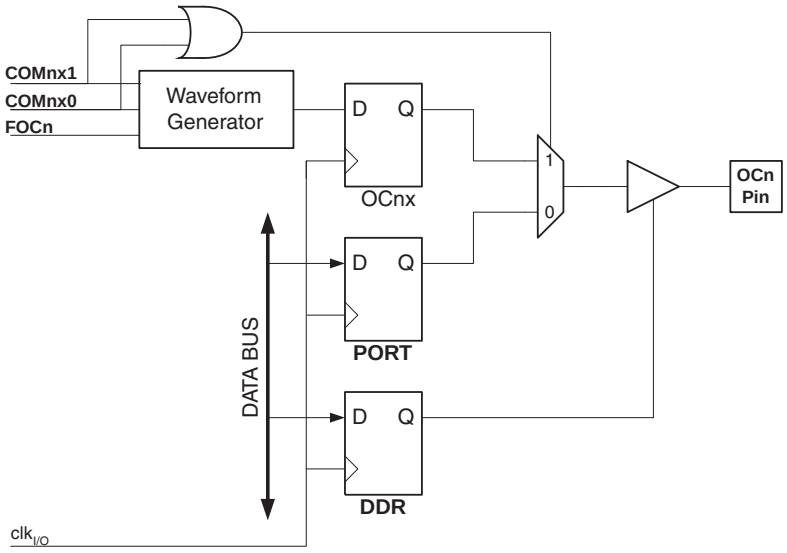
The setup of the OC0x should be performed before setting the Data Direction Register for the port pin to output. The easiest way of setting the OC0x value is to use the Force Output Compare (FOC0x) strobe bits in Normal mode. The OC0x Registers keep their values even when changing between Waveform Generation modes.

Be aware that the COM0x1:0 bits are not double buffered together with the compare value. Changing the COM0x1:0 bits will take effect immediately.

11.6 Compare Match Output Unit

The Compare Output mode (COM0x1:0) bits have two functions. The Waveform Generator uses the COM0x1:0 bits for defining the Output Compare (OC0x) state at the next Compare Match. Also, the COM0x1:0 bits control the OC0x pin output source. Figure 11-4 shows a simplified schematic of the logic affected by the COM0x1:0 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 COM0x1:0 bits are shown. When referring to the OC0x state, the reference is for the internal OC0x Register, not the OC0x pin. If a system reset occur, the OC0x Register is reset to “0”.

Figure 11-4. Compare Match Output Unit, Schematic



The general I/O port function is overridden by the Output Compare (OC0x) from the Waveform Generator if either of the COM0x1:0 bits are set. However, the OC0x 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 OC0x pin (DDR_OC0x) must be set as output before the OC0x 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 OC0x state before the output is enabled. Note that some COM0x1:0 bit settings are reserved for certain modes of operation. See “Register Description” on page 82.

11.6.1 Compare Output Mode and Waveform Generation

The Waveform Generator uses the COM0x1:0 bits differently in Normal, CTC, and PWM modes. For all modes, setting the COM0x1:0 = 0 tells the Waveform Generator that no action on the OC0x Register is to be performed on the next Compare Match. For compare output actions in the non-PWM modes refer to Figure 11-3 on page 74. For fast PWM mode, refer to Table 10-6 on page 66, and for phase correct PWM refer to Table 10-7 on page 66.

A change of the COM0x1:0 bits state will have effect at the first Compare Match after the bits are written. For non-PWM modes, the action can be forced to have immediate effect by using the FOC0x strobe bits.

11.7 Modes of Operation

The mode of operation, i.e., the behavior of the Timer/Counter and the Output Compare pins, is defined by the combination of the Waveform Generation mode (WGM02:0) and Compare Output mode (COM0x1:0) bits. The Compare Output mode bits do not affect the counting sequence, while the Waveform Generation mode bits do. The COM0x1:0 bits control whether the PWM output generated should be inverted or not (inverted or non-inverted PWM). For non-PWM modes the COM0x1:0 bits control whether the output should be set, cleared, or toggled at a Compare Match (See “Compare Match Output Unit” on page 75.).

For detailed timing information refer to Figure 11-8, Figure 11-9, Figure 11-10 and Figure 11-11 in “Timer/Counter Timing Diagrams” on page 80.

11.7.1 Normal Mode

The simplest mode of operation is the Normal mode (WGM02:0 = 0). In this mode the counting direction is always up (incrementing), and no counter clear is performed. The counter simply overruns when it passes its maximum 8-bit value (TOP = 0xFF) and then restarts from the bottom (0x00). In normal operation the Timer/Counter Overflow Flag (TOV0) will be set in the same timer clock cycle as the TCNT0 becomes zero. The TOV0 Flag in this case behaves like a ninth bit, except that it is only set, not cleared. However, combined with the timer overflow interrupt that automatically clears the TOV0 Flag, the timer resolution can be increased by software. There are no special cases to consider in the Normal mode, a new counter value can be written anytime.

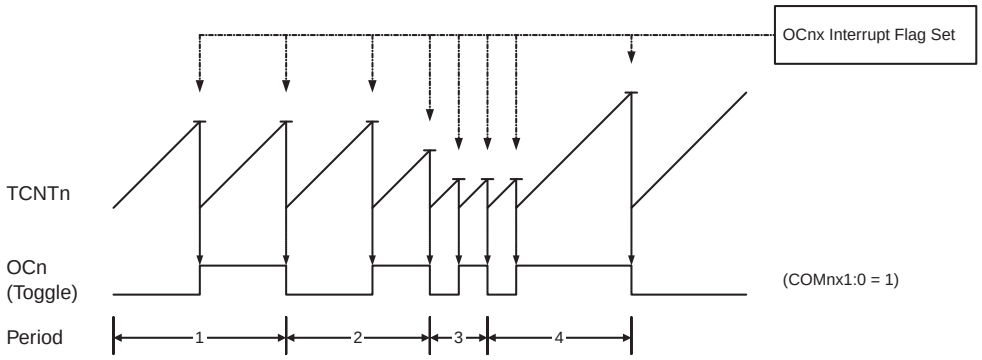
The Output Compare Unit can be used to generate interrupts at some given time. Using the Output Compare to generate waveforms in Normal mode is not recommended, since this will occupy too much of the CPU time.

11.7.2 Clear Timer on Compare Match (CTC) Mode

In Clear Timer on Compare or CTC mode (WGM02:0 = 2), the OCR0A Register is used to manipulate the counter resolution. In CTC mode the counter is cleared to zero when the counter value (TCNT0) matches the OCR0A. The OCR0A defines the top value for the counter, hence also its resolution. This mode allows greater control of the Compare Match output frequency. It also simplifies the operation of counting external events.

The timing diagram for the CTC mode is shown in Figure 11-5. The counter value (TCNT0) increases until a Compare Match occurs between TCNT0 and OCR0A, and then counter (TCNT0) is cleared.

Figure 11-5. CTC Mode, Timing Diagram



An interrupt can be generated each time the counter value reaches the TOP value by using the OCF0A Flag. If the interrupt is enabled, the interrupt handler routine can be used for updating the TOP value. However, changing TOP to a value close to BOTTOM when the counter is running with none or a low prescaler value must be done with care since the CTC mode does not have the double buffering feature. If the new value written to OCR0A is lower than the current value of TCNT0, the counter will miss the Compare Match. The counter will then have to count to its maximum value (0xFF) and wrap around starting at 0x00 before the Compare Match can occur.

For generating a waveform output in CTC mode, the OC0A output can be set to toggle its logical level on each Compare Match by setting the Compare Output mode bits to toggle mode (COM0A1:0 = 1). The OC0A value will not be visible on the port pin unless the data direction for the pin is set to output. The waveform generated will have a maximum frequency of $f_{OC0} = f_{clk\_I/O}/2$ when OCR0A is set to zero (0x00). The waveform frequency is defined by the following equation:

$$f_{OCnx} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot (1 + OCRnx)}$$

The *N* variable represents the prescale factor (1, 8, 64, 256, or 1024).

As for the Normal mode of operation, the TOV0 Flag is set in the same timer clock cycle that the counter counts from MAX to 0x00.

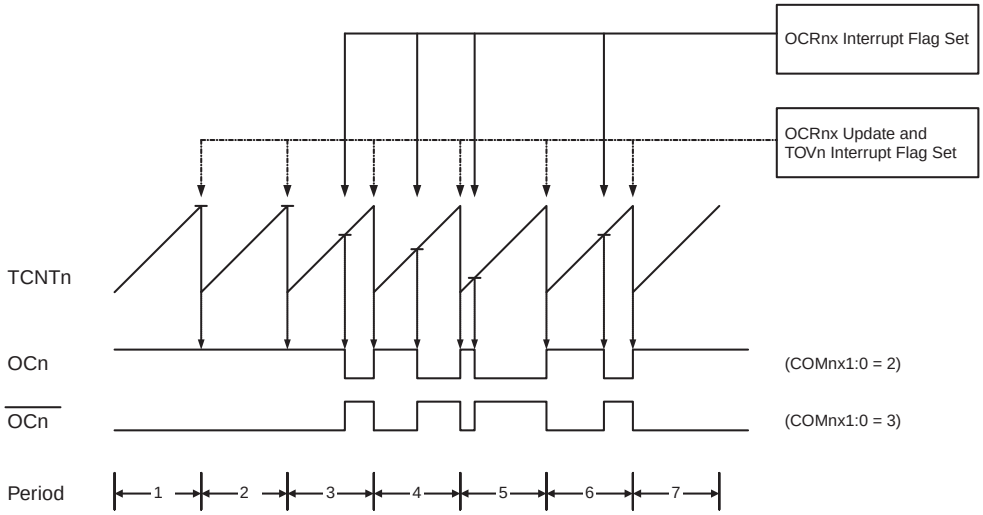
11.7.3 Fast PWM Mode

The fast Pulse Width Modulation or fast PWM mode (WGM02:0 = 3 or 7) provides a high frequency PWM waveform generation option. The fast PWM differs from the other PWM option by its single-slope operation. The counter counts from BOTTOM to TOP then restarts from BOTTOM. TOP is defined as 0xFF when WGM2:0 = 3, and OCR0A when WGM2:0 = 7. In non-inverting Compare Output mode, the Output Compare (OC0x) is cleared on the Compare Match between TCNT0 and OCR0x, and set at BOTTOM. In inverting Compare Output mode, the output is set on Compare Match and cleared at BOTTOM. Due to the single-slope operation, the operating frequency of the fast PWM mode can be twice as high as the phase correct PWM mode that use dual-slope operation. This high frequency makes the fast PWM mode well suited for power regulation, rectification, and DAC applications. High frequency allows physically small sized external components (coils, capacitors), and therefore reduces total system cost.

In fast PWM mode, the counter is incremented until the counter value matches the TOP value. The counter is then cleared at the following timer clock cycle. The timing diagram for the fast

PWM mode is shown in Figure 11-4. The TCNT0 value is in the timing diagram shown as a histogram for illustrating the single-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT0 slopes represent Compare Matches between OCR0x and TCNT0.

Figure 11-6. Fast PWM Mode, Timing Diagram



The Timer/Counter Overflow Flag (TOV0) is set each time the counter reaches TOP. If the interrupt is enabled, the interrupt handler routine can be used for updating the compare value.

In fast PWM mode, the compare unit allows generation of PWM waveforms on the OC0x pins. Setting the COM0x1:0 bits to two will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COM0x1:0 to three: Setting the COM0A1:0 bits to one allows the AC0A pin to toggle on Compare Matches if the WGM02 bit is set. This option is not available for the OC0B pin (See Table 10-6 on page 66). The actual OC0x value will only be visible on the port pin if the data direction for the port pin is set as output. The PWM waveform is generated by setting (or clearing) the OC0x Register at the Compare Match between OCR0x and TCNT0, and clearing (or setting) the OC0x Register at the timer clock cycle the counter is cleared (changes from TOP to BOTTOM).

The PWM frequency for the output can be calculated by the following equation:

$$f_{OCnxPWM} = \frac{f_{clk\_I/O}}{N \cdot 256}$$

The *N* variable represents the prescale factor (1, 8, 64, 256, or 1024).

The extreme values for the OCR0A Register represents special cases when generating a PWM waveform output in the fast PWM mode. If the OCR0A is set equal to BOTTOM, the output will be a narrow spike for each MAX+1 timer clock cycle. Setting the OCR0A equal to MAX will result in a constantly high or low output (depending on the polarity of the output set by the COM0A1:0 bits.)

A frequency (with 50% duty cycle) waveform output in fast PWM mode can be achieved by setting OC0x to toggle its logical level on each Compare Match (COM0x1:0 = 1). The waveform generated will have a maximum frequency of $f_{OC0} = f_{clk\_I/O}/2$ when OCR0A is set to zero. This

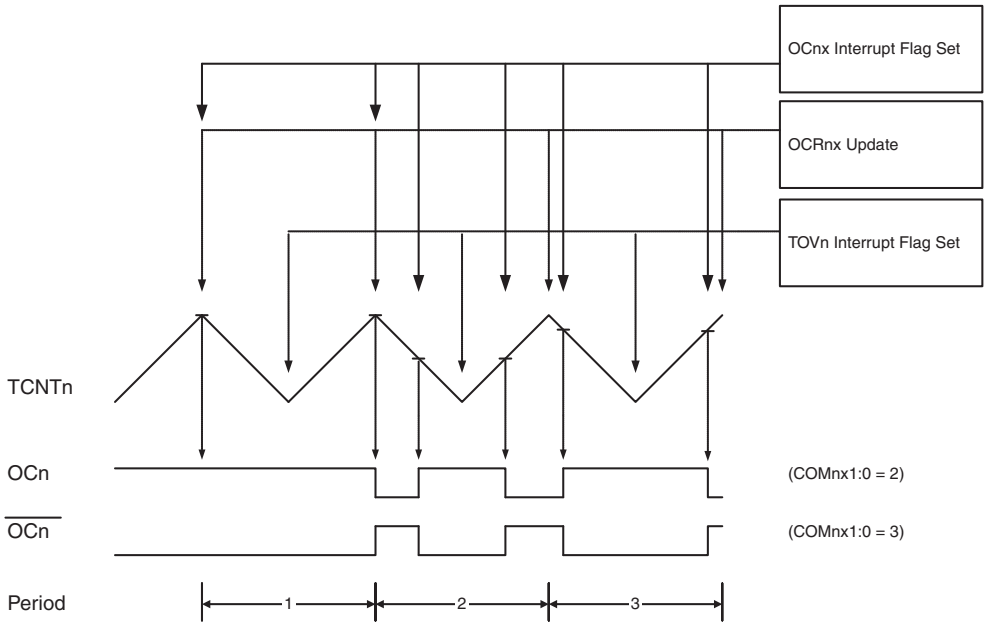
feature is similar to the OC0A toggle in CTC mode, except the double buffer feature of the Output Compare unit is enabled in the fast PWM mode.

11.7.4 Phase Correct PWM Mode

The phase correct PWM mode (WGM02:0 = 1 or 5) provides a high resolution phase correct PWM waveform generation option. The phase correct PWM mode is based on a dual-slope operation. The counter counts repeatedly from BOTTOM to TOP and then from TOP to BOTTOM. TOP is defined as 0xFF when WGM2:0 = 1, and OCR0A when WGM2:0 = 5. In non-inverting Compare Output mode, the Output Compare (OC0x) is cleared on the Compare Match between TCNT0 and OCR0x while upcounting, and set on the Compare Match while downcounting. In inverting Output Compare mode, the operation is inverted. The dual-slope operation has lower maximum operation frequency than single slope operation. However, due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

In phase correct PWM mode the counter is incremented until the counter value matches TOP. When the counter reaches TOP, it changes the count direction. The TCNT0 value will be equal to TOP for one timer clock cycle. The timing diagram for the phase correct PWM mode is shown on Figure 11-7. The TCNT0 value is in the timing diagram shown as a histogram for illustrating the dual-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT0 slopes represent Compare Matches between OCR0x and TCNT0.

Figure 11-7. Phase Correct PWM Mode, Timing Diagram



The Timer/Counter Overflow Flag (TOV0) is set each time the counter reaches BOTTOM. The Interrupt Flag can be used to generate an interrupt each time the counter reaches the BOTTOM value.

In phase correct PWM mode, the compare unit allows generation of PWM waveforms on the OC0x pins. Setting the COM0x1:0 bits to two will produce a non-inverted PWM. An inverted PWM output can be generated by setting the COM0x1:0 to three: Setting the COM0A0 bits to

one allows the OC0A pin to toggle on Compare Matches if the WGM02 bit is set. This option is not available for the OC0B pin (See Table 10-7 on page 66). The actual OC0x value will only be visible on the port pin if the data direction for the port pin is set as output. The PWM waveform is generated by clearing (or setting) the OC0x Register at the Compare Match between OCR0x and TCNT0 when the counter increments, and setting (or clearing) the OC0x Register at Compare Match between OCR0x and TCNT0 when the counter decrements. The PWM frequency for the output when using phase correct PWM can be calculated by the following equation:

$$f_{OCnxPCPWM} = \frac{f_{clk\_I/O}}{N \cdot 510}$$

The N variable represents the prescale factor (1, 8, 64, 256, or 1024).

The extreme values for the OCR0A Register represent special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCR0A is set equal to BOTTOM, the output will be continuously low and if set equal to MAX the output will be continuously high for non-inverted PWM mode. For inverted PWM the output will have the opposite logic values.

At the very start of period 2 in Figure 11-7 on page 79 OCn has a transition from high to low even though there is no Compare Match. The point of this transition is to guarantee symmetry around BOTTOM. There are two cases that give a transition without Compare Match.

- OCR0A changes its value from MAX, like in Figure 11-7 on page 79. When the OCR0A value is MAX the OCn pin value is the same as the result of a down-counting Compare Match. To ensure symmetry around BOTTOM the OCn value at MAX must correspond to the result of an up-counting Compare Match.
- The timer starts counting from a value higher than the one in OCR0A, and for that reason misses the Compare Match and hence the OCn change that would have happened on the way up.

11.8 Timer/Counter Timing Diagrams

The Timer/Counter is a synchronous design and the timer clock (clk_{T0}) is therefore shown as a clock enable signal in the following figures. The figures include information on when Interrupt Flags are set. Figure 11-8 contains timing data for basic Timer/Counter operation. The figure shows the count sequence close to the MAX value in all modes other than phase correct PWM mode.

Figure 11-8. Timer/Counter Timing Diagram, no Prescaling

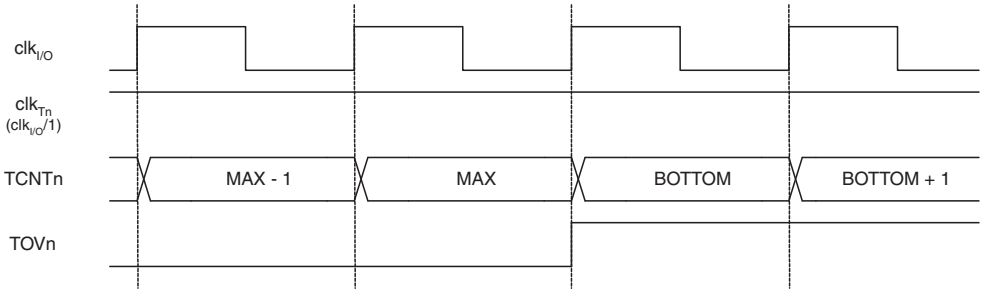


Figure 11-9 on page 81 shows the same timing data, but with the prescaler enabled.

Figure 11-9. Timer/Counter Timing Diagram, with Prescaler ($f_{clk\_I/O}/8$)

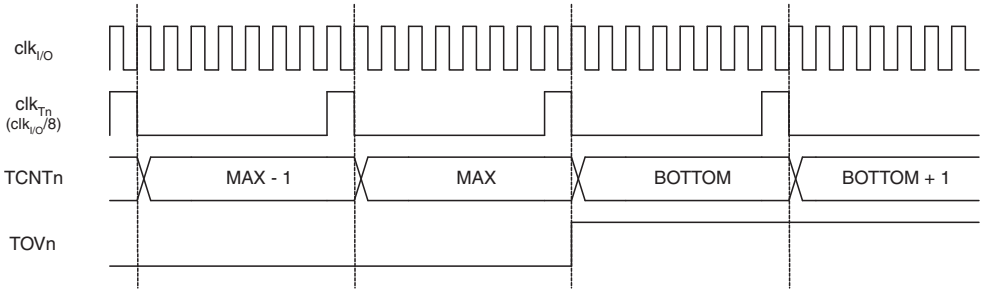


Figure 11-10 shows the setting of OCF0B in all modes and OCF0A in all modes except CTC mode and PWM mode, where OCR0A is TOP.

Figure 11-10. Timer/Counter Timing Diagram, Setting of OCF0x, with Prescaler ($f_{clk\_I/O}/8$)

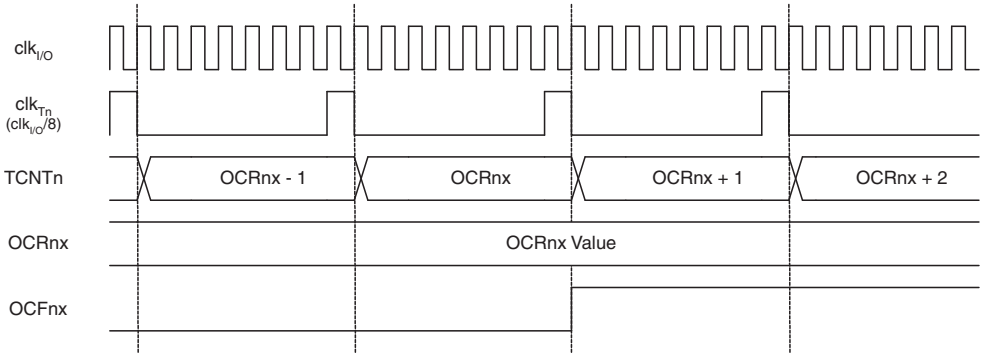
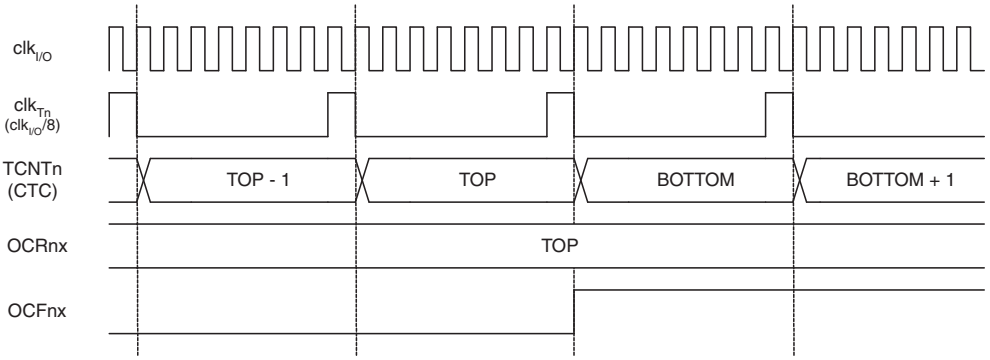


Figure 11-11 shows the setting of OCF0A and the clearing of TCNT0 in CTC mode and fast PWM mode where OCR0A is TOP.

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



11.9 Register Description

11.9.1 TCCR0A – Timer/Counter Control Register A

Bit	7	6	5	4	3	2	1	0	
0x30 (0x50)	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00	TCCR0A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:6 – COM0A1:0: Compare Match Output A Mode**

These bits control the Output Compare pin (OC0A) behavior. If one or both of the COM0A1:0 bits are set, the OC0A output overrides the normal port functionality of the I/O pin it is connected to. However, note that the Data Direction Register (DDR) bit corresponding to the OC0A pin must be set in order to enable the output driver.

When OC0A is connected to the pin, the function of the COM0A1:0 bits depends on the WGM02:0 bit setting. Table 11-2 shows the COM0A1:0 bit functionality when the WGM02:0 bits are set to a normal or CTC mode (non-PWM).

Table 11-2. Compare Output Mode, non-PWM Mode

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	Toggle OC0A on Compare Match
1	0	Clear OC0A on Compare Match
1	1	Set OC0A on Compare Match

Table 11-3 shows the COM0A1:0 bit functionality when the WGM01:0 bits are set to fast PWM mode.

Table 11-3. Compare Output Mode, Fast PWM Mode(1)

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	WGM02 = 0: Normal Port Operation, OC0A Disconnected. WGM02 = 1: Toggle OC0A on Compare Match.
1	0	Clear OC0A on Compare Match, set OC0A at TOP
1	1	Set OC0A on Compare Match, clear OC0A at TOP

Note: 1. A special case occurs when OCR0A equals TOP and COM0A1 is set. In this case, the Compare Match is ignored, but the set or clear is done at TOP. See “Fast PWM Mode” on page 77 for more details.

Table 11-4 shows the COM0A1:0 bit functionality when the WGM02:0 bits are set to phase correct PWM mode.

Table 11-4. Compare Output Mode, Phase Correct PWM Mode(1)

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	WGM02 = 0: Normal Port Operation, OC0A Disconnected. WGM02 = 1: Toggle OC0A on Compare Match.
1	0	Clear OC0A on Compare Match when up-counting. Set OC0A on Compare Match when down-counting.
1	1	Set OC0A on Compare Match when up-counting. Clear OC0A on Compare Match when down-counting.

Note: 1. A special case occurs when OCR0A equals TOP and COM0A1 is set. In this case, the Compare Match is ignored, but the set or clear is done at TOP. See “Phase Correct PWM Mode” on page 79 for more details.

- **Bits 5:4 – COM0B1:0: Compare Match Output B Mode**

These bits control the Output Compare pin (OC0B) behavior. If one or both of the COM0B1:0 bits are set, the OC0B output overrides the normal port functionality of the I/O pin it is connected to. However, note that the Data Direction Register (DDR) bit corresponding to the OC0B pin must be set in order to enable the output driver.

When OC0B is connected to the pin, the function of the COM0B1:0 bits depends on the WGM02:0 bit setting. Table 11-5 shows the COM0B1:0 bit functionality when the WGM02:0 bits are set to a normal or CTC mode (non-PWM).

Table 11-5. Compare Output Mode, non-PWM Mode

COM0B1	COM0B0	Description
0	0	Normal port operation, OC0B disconnected.
0	1	Toggle OC0B on Compare Match
1	0	Clear OC0B on Compare Match
1	1	Set OC0B on Compare Match

Table 11-6 shows the COM0B1:0 bit functionality when the WGM02:0 bits are set to fast PWM mode.

Table 11-6. Compare Output Mode, Fast PWM Mode(1)

COM0B1	COM0B0	Description
0	0	Normal port operation, OC0B disconnected.
0	1	Reserved
1	0	Clear OC0B on Compare Match, set OC0B at TOP
1	1	Set OC0B on Compare Match, clear OC0B at TOP

Note: 1. A special case occurs when OCR0B equals TOP and COM0B1 is set. In this case, the Compare Match is ignored, but the set or clear is done at TOP. See “Fast PWM Mode” on page 77 for more details.

Table 11-7 shows the COM0B1:0 bit functionality when the WGM02:0 bits are set to phase correct PWM mode.

Table 11-7. Compare Output Mode, Phase Correct PWM Mode[1]

COM0B1	COM0B0	Description
0	0	Normal port operation, OCR0B disconnected.
0	1	Reserved
1	0	Clear ORC0B on Compare Match when up-counting. Set OCR0B on Compare Match when down-counting.
1	1	Set OCR0B on Compare Match when up-counting. Clear OCR0B on Compare Match when down-counting.

Note: 1. A special case occurs when OCR0B equals TOP and COM0B1 is set. In this case, the Compare Match is ignored, but the set or clear is done at TOP. See “Phase Correct PWM Mode” on page 79 for more details.

- **Bits 3, 2 – Res: Reserved Bits**

These bits are reserved bits in the ATtiny2313A/4313 and will always read as zero.

- **Bits 1:0 – WGM01:0: Waveform Generation Mode**

Combined with the WGM02 bit found in the TCCR0B Register, these bits control the counting sequence of the counter, the source for maximum (TOP) counter value, and what type of waveform generation to be used, see Table 11-8. Modes of operation supported by the Timer/Counter unit are: Normal mode (counter), Clear Timer on Compare Match (CTC) mode, and two types of Pulse Width Modulation (PWM) modes (see “Modes of Operation” on page 76).

Table 11-8. Waveform Generation Mode Bit Description

Mode	WGM2	WGM1	WGM0	Timer/Counter Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on[1][2]
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCR0A	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	TOP	MAX
4	1	0	0	Reserved	–	–	–
5	1	0	1	PWM, Phase Correct	OCR0A	TOP	BOTTOM
6	1	1	0	Reserved	–	–	–
7	1	1	1	Fast PWM	OCR0A	TOP	TOP

Notes: 1. MAX = 0xFF
2. BOTTOM = 0x00

11.9.2 TCCR0B – Timer/Counter Control Register B

Bit	7	6	5	4	3	2	1	0	
0x33 (0x53)	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	TCCR0B
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – FOC0A: Force Output Compare A**

The FOC0A bit is only active when the WGM bits specify a non-PWM mode.

However, for ensuring compatibility with future devices, this bit must be set to zero when TCCR0B is written when operating in PWM mode. When writing a logical one to the FOC0A bit, an immediate Compare Match is forced on the Waveform Generation unit. The OC0A output is changed according to its COM0A1:0 bits setting. Note that the FOC0A bit is implemented as a strobe. Therefore it is the value present in the COM0A1:0 bits that determines the effect of the forced compare.

A FOC0A strobe will not generate any interrupt, nor will it clear the timer in CTC mode using OCR0A as TOP.

The FOC0A bit is always read as zero.

- **Bit 6 – FOC0B: Force Output Compare B**

The FOC0B bit is only active when the WGM bits specify a non-PWM mode.

However, for ensuring compatibility with future devices, this bit must be set to zero when TCCR0B is written when operating in PWM mode. When writing a logical one to the FOC0B bit, an immediate Compare Match is forced on the Waveform Generation unit. The OC0B output is changed according to its COM0B1:0 bits setting. Note that the FOC0B bit is implemented as a strobe. Therefore it is the value present in the COM0B1:0 bits that determines the effect of the forced compare.

A FOC0B strobe will not generate any interrupt, nor will it clear the timer in CTC mode using OCR0B as TOP.

The FOC0B bit is always read as zero.

- **Bits 5:4 – Res: Reserved Bits**

These bits are reserved bits in the ATtiny2313A/4313 and will always read as zero.

- **Bit 3 – WGM02: Waveform Generation Mode**

See the description in the “TCCR0A – Timer/Counter Control Register A” on page 82.

- **Bits 2:0 – CS02:0: Clock Select**

The three Clock Select bits select the clock source to be used by the Timer/Counter. See Table 11-9 on page 86.

Table 11-9. Clock Select Bit Description

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/(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 T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

If external pin modes are used for the Timer/Counter0, transitions on the T0 pin will clock the counter even if the pin is configured as an output. This feature allows software control of the counting.

11.9.3 TCNT0 – Timer/Counter Register

Bit	7	6	5	4	3	2	1	0	
0x32 (0x52)	TCNT0[7:0]								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 Registers.

11.9.4 OCR0A – Output Compare Register A

Bit	7	6	5	4	3	2	1	0	
0x36 (0x56)	OCR0A[7:0]								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.

11.9.5 OCR0B – Output Compare Register B

Bit	7	6	5	4	3	2	1	0	
0x3C (0x5C)	OCR0B[7:0]								OCR0B
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 B 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 OC0B pin.

11.9.6 TIMSK – Timer/Counter Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
0x39 (0x59)	TOIE1	OCIE1A	OCIE1B	–	ICIE1	OCIE0B	TOIE0	OCIE0A	TIMSK
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 4 – Res: Reserved Bit**

This bit is reserved bit in the ATtiny2313A/4313 and will always read as zero.

- **Bit 2 – OCIE0B: Timer/Counter0 Output Compare Match B Interrupt Enable**

When the OCIE0B bit is written to one, and the I-bit in the Status Register is set, the Timer/Counter Compare Match B interrupt is enabled. The corresponding interrupt is executed if a Compare Match in Timer/Counter occurs, i.e., when the OCF0B bit is set in the Timer/Counter Interrupt Flag Register – TIFR.

- **Bit 1 – TOIE0: Timer/Counter0 Overflow Interrupt Enable**

When the TOIE0 bit is written to one, and the I-bit in the Status Register is set, the Timer/Counter0 Overflow interrupt is enabled. The corresponding interrupt is executed if an overflow in Timer/Counter0 occurs, i.e., when the TOV0 bit is set in the Timer/Counter 0 Interrupt Flag Register – TIFR.

- **Bit 0 – OCIE0A: Timer/Counter0 Output Compare Match A Interrupt Enable**

When the OCIE0A bit is written to one, and the I-bit in the Status Register is set, the Timer/Counter0 Compare Match A interrupt is enabled. The corresponding interrupt is executed if a Compare Match in Timer/Counter0 occurs, i.e., when the OCF0A bit is set in the Timer/Counter 0 Interrupt Flag Register – TIFR.

11.9.7 TIFR – Timer/Counter Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x38 (0x58)	TOV1	OCF1A	OCF1B	–	ICF1	OCF0B	TOV0	OCF0A	TIFR
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 4 – Res: Reserved Bit**

This bit is reserved bit in the ATtiny2313A/4313 and will always read as zero.

- **Bit 2 – OCF0B: Output Compare Flag 0 B**

The OCF0B bit is set when a Compare Match occurs between the Timer/Counter and the data in OCR0B – Output Compare Register0 B. OCF0B is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF0B is cleared by writing a logic one to the flag. When the I-bit in SREG, OCIE0B (Timer/Counter Compare B Match Interrupt Enable), and OCF0B are set, the Timer/Counter Compare Match Interrupt is executed.

- **Bit 1 – TOV0: Timer/Counter0 Overflow Flag**

The bit TOV0 is set when an overflow occurs in Timer/Counter0. TOV0 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, TOV0 is cleared by writing a logic one to the flag. When the SREG I-bit, TOIE0 (Timer/Counter0 Overflow Interrupt Enable), and TOV0 are set, the Timer/Counter0 Overflow interrupt is executed.

The setting of this flag is dependent of the WGM02:0 bit setting. Refer to Table 11-8, “Waveform Generation Mode Bit Description” on page 84.

- **Bit 0 – OCF0A: Output Compare Flag 0 A**

The OCF0A bit is set when a Compare Match occurs between the Timer/Counter0 and the data in OCR0A – Output Compare Register0 A. OCF0A is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF0A is cleared by writing a logic one to the flag. When the I-bit in SREG, OCIE0A (Timer/Counter0 Compare Match Interrupt Enable), and OCF0A are set, the Timer/Counter0 Compare Match Interrupt is executed.

12. 16-bit Timer/Counter1

12.1 Features

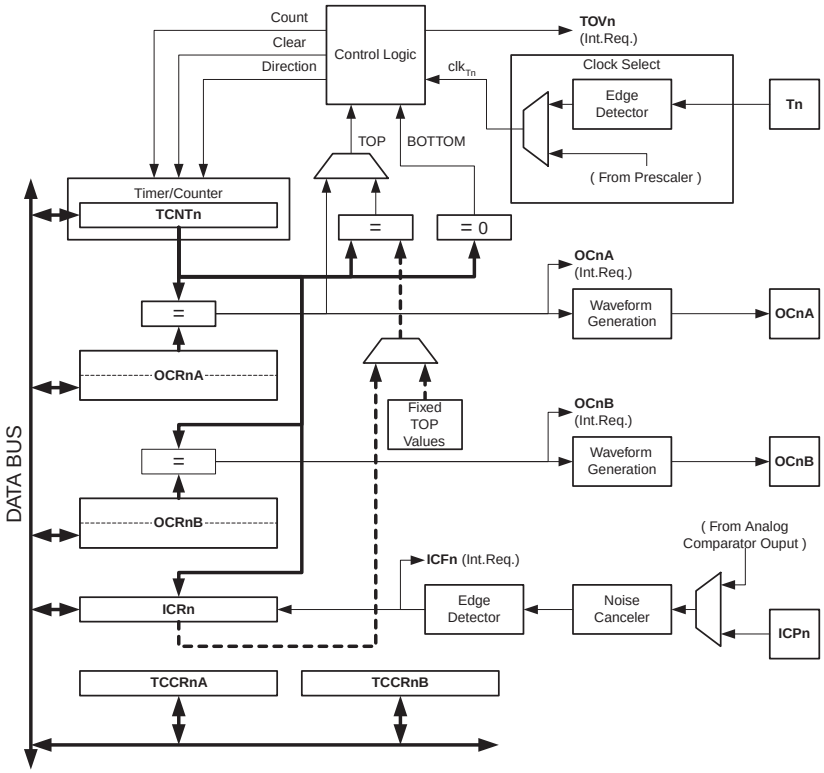
- **True 16-bit Design (i.e., Allows 16-bit PWM)**
- **Two independent Output Compare Units**
- **Double Buffered Output Compare Registers**
- **One Input Capture Unit**
- **Input Capture Noise Canceler**
- **Clear Timer on Compare Match (Auto Reload)**
- **Glitch-free, Phase Correct Pulse Width Modulator (PWM)**
- **Variable PWM Period**
- **Frequency Generator**
- **External Event Counter**
- **Four independent interrupt Sources (TOV1, OCF1A, OCF1B, and ICF1)**

12.2 Overview

The 16-bit Timer/Counter unit allows accurate program execution timing (event management), wave generation, and signal timing measurement.

A simplified block diagram of the 16-bit Timer/Counter is shown in Figure 12-1. For the actual placement of I/O pins, refer to “Pinout ATtiny2313A/4313” on page 2. CPU accessible I/O Registers, including I/O bits and I/O pins, are shown in bold. The device-specific I/O Register and bit locations are listed in the “Register Description” on page 111.

Figure 12-1. 16-bit Timer/Counter Block Diagram(Note:)



Note: Refer to Figure 1-1 on page 2 for Timer/Counter1 pin placement and description.

Most register and bit references in this section are written in general form. A lower case “n” replaces the Timer/Counter number, and a lower case “x” replaces the Output Compare unit channel. However, when using the register or bit defines in a program, the precise form must be used, i.e., TCNT1 for accessing Timer/Counter1 counter value and so on.

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 “Accessing 16-bit Registers” on page 107. 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* (TIFR). All interrupts are individually masked with the *Timer Interrupt Mask Register* (TIMSK). TIFR and TIMSK are not shown in the figure.

The Timer/Counter can be clocked internally, via the prescaler, or by an external clock source on the T1 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_{T1}).

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 pin (OC1A/B). See “Output Compare Units” on page 94.. 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 “Analog Comparator” on page 168.) 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:

Table 12-1. Definitions

Constant	Description
BOTTOM	The counter reaches BOTTOM when it becomes 0x0000
MAX	The counter reaches its MAXimum when it becomes 0xFFFF (decimal 65535)
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 depends on the mode of operation

12.2.3 Compatibility

The 16-bit Timer/Counter has been updated and improved from previous versions of the 16-bit AVR Timer/Counter. This 16-bit Timer/Counter is fully compatible with the earlier version regarding:

- All 16-bit Timer/Counter related I/O Register address locations, including Timer Interrupt Registers.
- Bit locations inside all 16-bit Timer/Counter Registers, including Timer Interrupt Registers.
- Interrupt Vectors.

The following control bits have changed name, but have same functionality and register location:

- PWM10 is changed to WGM10.
- PWM11 is changed to WGM11.
- CTC1 is changed to WGM12.

The following bits are added to the 16-bit Timer/Counter Control Registers:

- FOC1A and FOC1B are added to TCCR1A.
- WGM13 is added to TCCR1B.

The 16-bit Timer/Counter has improvements that will affect the compatibility in some special cases.

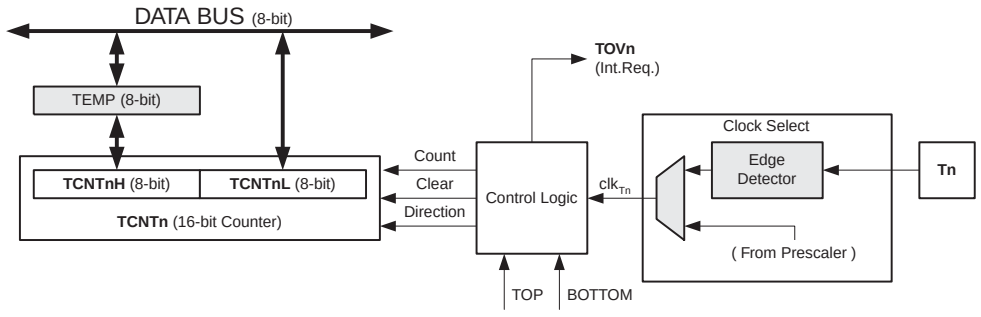
12.3 Timer/Counter Clock Sources

The Timer/Counter can be clocked by an internal or an external clock source. The clock source is selected by the Clock Select logic which is controlled by the *Clock Select* (CS12:0) bits located in the *Timer/Counter control Register B* (TCCR1B). For details on clock sources and prescaler, see “Timer/Counter0 and Timer/Counter1 Prescalers” on page 118.

12.4 Counter Unit

The main part of the 16-bit Timer/Counter is the programmable 16-bit bi-directional counter unit. Figure 12-2 shows a block diagram of the counter and its surroundings.

Figure 12-2. Counter Unit Block Diagram



Signal description (internal signals):

Count	Increment or decrement TCNT1 by 1.
Direction	Select between increment and decrement.
Clear	Clear TCNT1 (set all bits to zero).

clk_{T1}	Timer/Counter clock.
TOP	Signalize that TCNT1 has reached maximum value.
BOTTOM	Signalize that TCNT1 has reached minimum value (zero).

The 16-bit counter is mapped into two 8-bit I/O memory locations: *Counter High* (TCNT1H) containing the upper eight bits of the counter, and *Counter Low* (TCNT1L) containing the lower eight bits. The TCNT1H Register can only be indirectly accessed by the CPU. When the CPU does an access to the TCNT1H I/O location, the CPU accesses the high byte temporary register (TEMP). The temporary register is updated with the TCNT1H value when the TCNT1L is read, and TCNT1H is updated with the temporary register value when TCNT1L is written. This allows the CPU to read or write the entire 16-bit counter value within one clock cycle via the 8-bit data bus. It is important to notice that there are special cases of writing to the TCNT1 Register when the counter is counting that will give unpredictable results. The special cases are described in the sections where they are of importance.

Depending on the mode of operation used, the counter is cleared, incremented, or decremented at each *timer clock* (clk_{T1}). The clk_{T1} can be generated from an external or internal clock source, selected by the *Clock Select* bits (CS12:0). When no clock source is selected (CS12:0 = 0) the timer is stopped. However, the TCNT1 value can be accessed by the CPU, independent of whether clk_{T1} is present or not. A CPU write overrides (has priority over) all counter clear or count operations.

The counting sequence is determined by the setting of the *Waveform Generation mode* bits (WGM13:0) located in the *Timer/Counter Control Registers* A and B (TCCR1A and TCCR1B). There are close connections between how the counter behaves (counts) and how waveforms are generated on the Output Compare outputs OC1x. For more details about advanced counting sequences and waveform generation, see “Modes of Operation” on page 97.

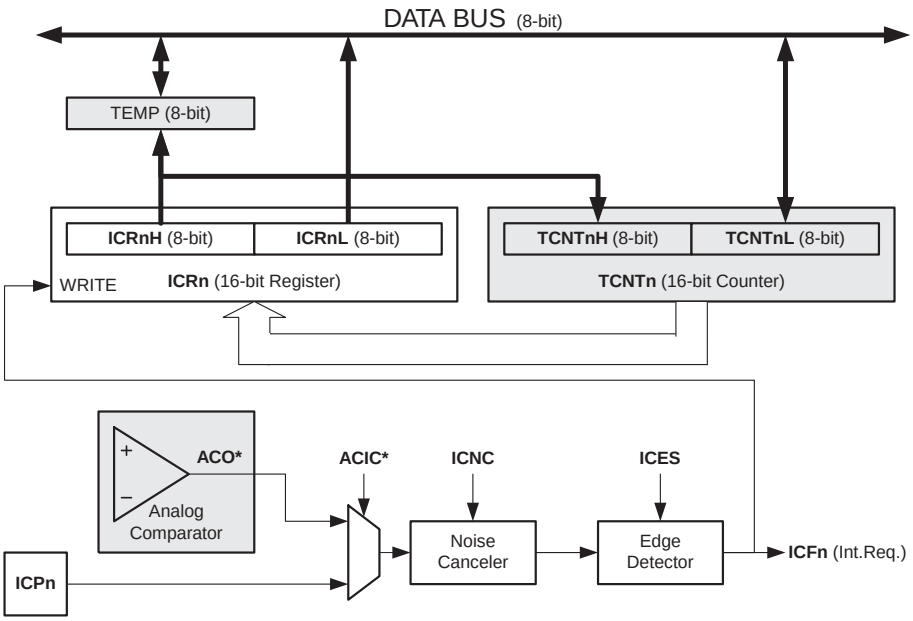
The Timer/Counter Overflow Flag (TOV1) is set according to the mode of operation selected by the WGM13:0 bits. TOV1 can be used for generating a CPU interrupt.

12.5 Input Capture Unit

The Timer/Counter incorporates an Input Capture unit that can capture external events and give them a time-stamp indicating time of occurrence. The external signal indicating an event, or multiple events, can be applied via the ICP1 pin or alternatively, via the analog-comparator unit. The time-stamps can then be used to calculate frequency, duty-cycle, and other features of the signal applied. Alternatively the time-stamps can be used for creating a log of the events.

The Input Capture unit is illustrated by the block diagram shown in Figure 12-3. The elements of the block diagram that are not directly a part of the Input Capture unit are gray shaded. The small “n” in register and bit names indicates the Timer/Counter number.

Figure 12-3. Input Capture Unit Block Diagram



When a change of the logic level (an event) occurs on the *Input Capture pin* (ICP1), alternatively on the *Analog Comparator output* (ACO), and this change confirms to the setting of the edge detector, a capture will be triggered. When a capture is triggered, the 16-bit value of the counter (TCNT1) is written to the *Input Capture Register* (ICR1). The *Input Capture Flag* (ICF1) is set at the same system clock as the TCNT1 value is copied into ICR1 Register. If enabled (ICIE1 = 1), the Input Capture Flag generates an Input Capture interrupt. The ICF1 flag is automatically cleared when the interrupt is executed. Alternatively the ICF1 flag can be cleared by software by writing a logical one to its I/O bit location.

Reading the 16-bit value in the *Input Capture Register* (ICR1) is done by first reading the low byte (ICR1L) and then the high byte (ICR1H). When the low byte is read the high byte is copied into the high byte temporary register (TEMP). When the CPU reads the ICR1H I/O location it will access the TEMP Register.

The ICR1 Register can only be written when using a Waveform Generation mode that utilizes the ICR1 Register for defining the counter's TOP value. In these cases the *Waveform Generation mode* (WGM13:0) bits must be set before the TOP value can be written to the ICR1 Register. When writing the ICR1 Register the high byte must be written to the ICR1H I/O location before the low byte is written to ICR1L.

For more information on how to access the 16-bit registers refer to "Accessing 16-bit Registers" on page 107.

12.5.1 Input Capture Trigger Source

The main trigger source for the Input Capture unit is the *Input Capture pin* (ICP1). Timer/Counter1 can alternatively use the Analog Comparator output as trigger source for the Input Capture unit. The Analog Comparator is selected as trigger source by setting the *Analog Comparator Input Capture* (ACIC) bit in the *Analog Comparator Control and Status Register* (ACSR). Be aware that changing trigger source can trigger a capture. The Input Capture Flag must therefore be cleared after the change.

Both the *Input Capture pin* (ICP1) and the *Analog Comparator output* (ACO) inputs are sampled using the same technique as for the T1 pin (Figure 13-1 on page 118). The edge detector is also identical. However, when the noise canceler is enabled, additional logic is inserted before the edge detector, which increases the delay by four system clock cycles. Note that the input of the noise canceler and edge detector is always enabled unless the Timer/Counter is set in a Waveform Generation mode that uses ICR1 to define TOP.

An Input Capture can be triggered by software by controlling the port of the ICP1 pin.

12.5.2 Noise Canceler

The noise canceler improves noise immunity by using a simple digital filtering scheme. The noise canceler input is monitored over four samples, and all four must be equal for changing the output that in turn is used by the edge detector.

The noise canceler is enabled by setting the *Input Capture Noise Canceler* (ICNC1) bit in *Timer/Counter Control Register B* (TCCR1B). When enabled the noise canceler introduces additional four system clock cycles of delay from a change applied to the input, to the update of the ICR1 Register. The noise canceler uses the system clock and is therefore not affected by the prescaler.

12.5.3 Using the Input Capture Unit

The main challenge when using the Input Capture unit is to assign enough processor capacity for handling the incoming events. The time between two events is critical. If the processor has not read the captured value in the ICR1 Register before the next event occurs, the ICR1 will be overwritten with a new value. In this case the result of the capture will be incorrect.

When using the Input Capture interrupt, the ICR1 Register should be read as early in the interrupt handler routine as possible. Even though the Input Capture interrupt has relatively high priority, the maximum interrupt response time is dependent on the maximum number of clock cycles it takes to handle any of the other interrupt requests.

Using the Input Capture unit in any mode of operation when the TOP value (resolution) is actively changed during operation, is not recommended.

Measurement of an external signal's duty cycle requires that the trigger edge is changed after each capture. Changing the edge sensing must be done as early as possible after the ICR1 Register has been read. After a change of the edge, the Input Capture Flag (ICF1) must be cleared by software (writing a logical one to the I/O bit location). For measuring frequency only, the clearing of the ICF1 flag is not required (if an interrupt handler is used).

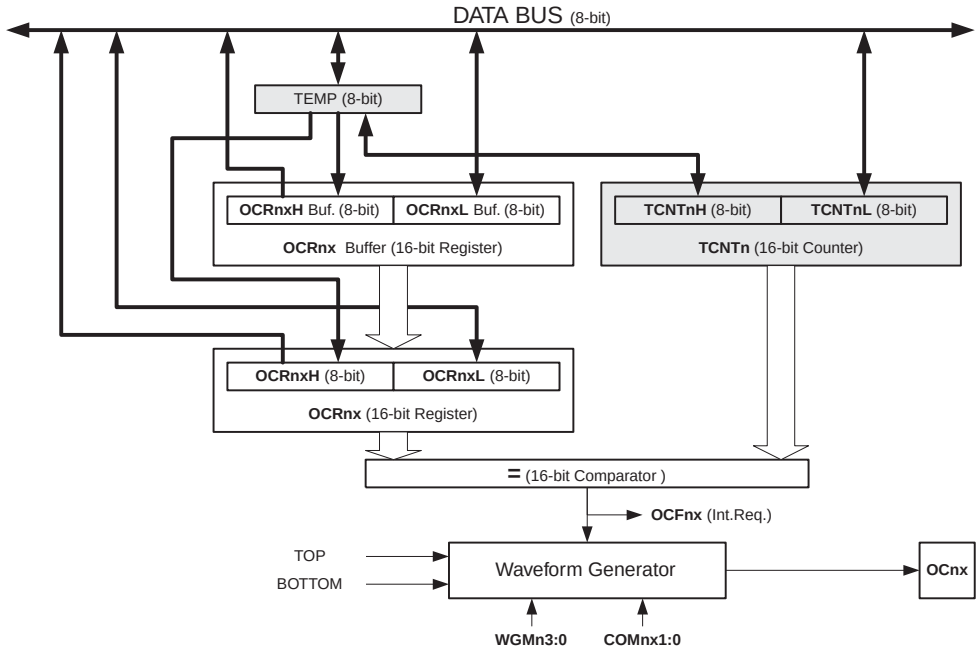
12.6 Output Compare Units

The 16-bit comparator continuously compares TCNT1 with the *Output Compare Register* (OCR1x). If TCNT equals OCR1x the comparator signals a match. A match will set the *Output Compare Flag* (OCF1x) at the next timer clock cycle. If enabled (OCIE1x = 1), the Output Compare Flag generates an Output Compare interrupt. The OCF1x flag is automatically cleared when the interrupt is executed. Alternatively the OCF1x 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 *Waveform Generation mode* (WGM13:0) bits and *Compare Output mode* (COM1x1:0) bits. The TOP and BOTTOM signals are used by the Waveform Generator for handling the special cases of the extreme values in some modes of operation (See "Modes of Operation" on page 97.)

A special feature of Output Compare unit A allows it to define the Timer/Counter TOP value (i.e., counter resolution). In addition to the counter resolution, the TOP value defines the period time for waveforms generated by the Waveform Generator.

Figure 12-4 shows a block diagram of the Output Compare unit. The small “n” in the register and bit names indicates the device number (n = 1 for Timer/Counter 1), and the “x” indicates Output Compare unit (A/B). The elements of the block diagram that are not directly a part of the Output Compare unit are gray shaded.

Figure 12-4. Output Compare Unit, Block Diagram



The OCR1x Register is double buffered when using any of the twelve *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 OCR1x 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 OCR1x Register access may seem complex, but this is not case. When the double buffering is enabled, the CPU has access to the OCR1x Buffer Register, and if double buffering is disabled the CPU will access the OCR1x directly. The content of the OCR1x (Buffer or Compare) Register is only changed by a write operation (the Timer/Counter does not update this register automatically as the TCNT1 and ICR1 Register). Therefore OCR1x is not read via the high byte temporary register (TEMP). However, it is a good practice to read the low byte first as when accessing other 16-bit registers. Writing the OCR1x Registers must be done via the TEMP Register since the compare of all 16 bits is done continuously. The high byte (OCR1xH) has to be written first. When the high byte I/O location is written by the CPU, the TEMP Register will be updated by the value written. Then when the low byte (OCR1xL) is written to the lower eight bits, the high byte will be copied into the upper 8-bits of either the OCR1x buffer or OCR1x Compare Register in the same system clock cycle.

For more information of how to access the 16-bit registers refer to “Accessing 16-bit Registers” on page 107.

12.6.1 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* (FOC1x) bit. Forcing compare match will not set the OCF1x flag or reload/clear the timer, but the OC1x pin will be updated as if a real compare match had occurred (the COM11:0 bits settings define whether the OC1x pin is set, cleared or toggled).

12.6.2 Compare Match Blocking by TCNT1 Write

All CPU writes to the TCNT1 Register will block any compare match that occurs in the next timer clock cycle, even when the timer is stopped. This feature allows OCR1x to be initialized to the same value as TCNT1 without triggering an interrupt when the Timer/Counter clock is enabled.

12.6.3 Using the Output Compare Unit

Since writing TCNT1 in any mode of operation will block all compare matches for one timer clock cycle, there are risks involved when changing TCNT1 when using any of the Output Compare channels, independent of whether the Timer/Counter is running or not. If the value written to TCNT1 equals the OCR1x value, the compare match will be missed, resulting in incorrect waveform generation. Do not write the TCNT1 equal to TOP in PWM modes with variable TOP values. The compare match for the TOP will be ignored and the counter will continue to 0xFFFF. Similarly, do not write the TCNT1 value equal to BOTTOM when the counter is downcounting.

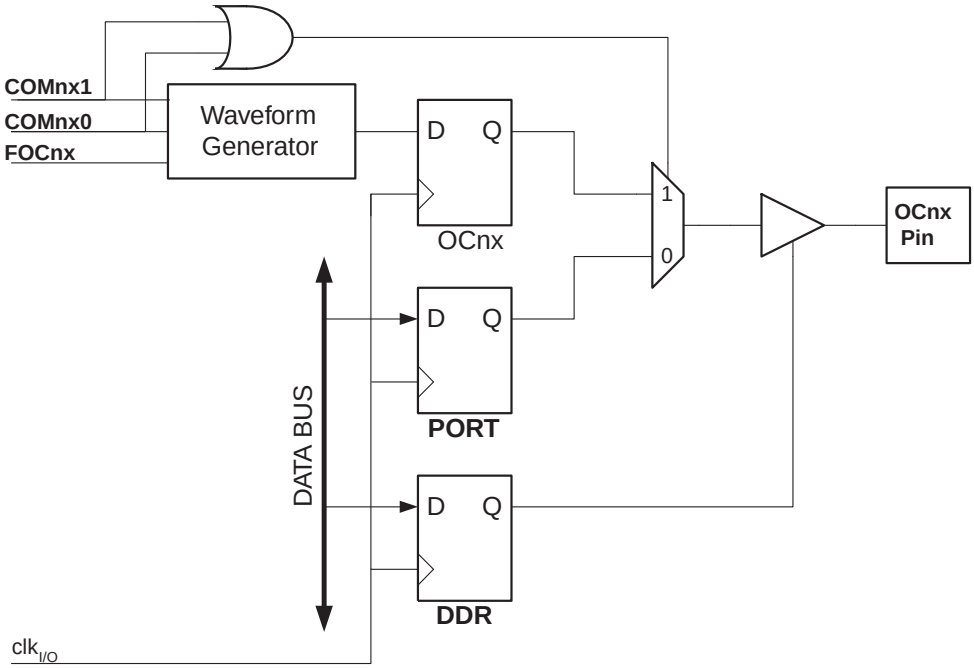
The setup of the OC1x should be performed before setting the Data Direction Register for the port pin to output. The easiest way of setting the OC1x value is to use the Force Output Compare (FOC1x) strobe bits in Normal mode. The OC1x Register keeps its value even when changing between Waveform Generation modes.

Be aware that the COM1x1:0 bits are not double buffered together with the compare value. Changing the COM1x1:0 bits will take effect immediately.

12.7 Compare Match Output Unit

The *Compare Output mode* (COM1x1:0) bits have two functions. The Waveform Generator uses the COM1x1:0 bits for defining the Output Compare (OC1x) state at the next compare match. Secondly the COM1x1:0 bits control the OC1x pin output source. Figure 12-5 shows a simplified schematic of the logic affected by the COM1x1:0 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 COM1x1:0 bits are shown. When referring to the OC1x state, the reference is for the internal OC1x Register, not the OC1x pin. If a system reset occur, the OC1x Register is reset to “0”.

Figure 12-5. Compare Match Output Unit, Schematic



The general I/O port function is overridden by the Output Compare (OC1x) from the Waveform Generator if either of the COM1x1:0 bits are set. However, the OC1x 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 OC1x pin (DDR_OC1x) must be set as output before the OC1x value is visible on the pin. The port override function is generally independent of the Waveform Generation mode, but there are some exceptions. Refer to Table 12-2, Table 12-3 and Table 12-4 for details.

The design of the Output Compare pin logic allows initialization of the OC1x state before the output is enabled. Note that some COM1x1:0 bit settings are reserved for certain modes of operation. See “Register Description” on page 111.

The COM1x1:0 bits have no effect on the Input Capture unit.

12.7.1 Compare Output Mode and Waveform Generation

The Waveform Generator uses the COM1x1:0 bits differently in normal, CTC, and PWM modes. For all modes, setting the COM1x1:0 = 0 tells the Waveform Generator that no action on the OC1x Register is to be performed on the next compare match. For compare output actions in the non-PWM modes refer to Table 12-2 on page 111. For fast PWM mode refer to Table 12-3 on page 111, and for phase correct and phase and frequency correct PWM refer to Table 12-4 on page 112.

A change of the COM1x1:0 bits state will have effect at the first compare match after the bits are written. For non-PWM modes, the action can be forced to have immediate effect by using the FOC1x strobe bits.

12.8 Modes of Operation

The mode of operation, i.e., the behavior of the Timer/Counter and the Output Compare pins, is defined by the combination of the *Waveform Generation mode* (WGM13:0) and *Compare Output*

mode (COM1x1:0) bits. The Compare Output mode bits do not affect the counting sequence, while the Waveform Generation mode bits do. The COM1x1:0 bits control whether the PWM output generated should be inverted or not (inverted or non-inverted PWM). For non-PWM modes the COM1x1:0 bits control whether the output should be set, cleared or toggle at a compare match (See “Compare Match Output Unit” on page 96.)

For detailed timing information refer to “Timer/Counter Timing Diagrams” on page 105.

12.8.1 Normal Mode

The simplest mode of operation is the *Normal mode* (WGM13:0 = 0). In this mode the counting direction is always up (incrementing), and no counter clear is performed. The counter simply overruns when it passes its maximum 16-bit value (MAX = 0xFFFF) and then restarts from the BOTTOM (0x0000). In normal operation the *Timer/Counter Overflow Flag* (TOV1) will be set in the same timer clock cycle as the TCNT1 becomes zero. The TOV1 flag in this case behaves like a 17th bit, except that it is only set, not cleared. However, combined with the timer overflow interrupt that automatically clears the TOV1 flag, the timer resolution can be increased by software. There are no special cases to consider in the Normal mode, a new counter value can be written anytime.

The Input Capture unit is easy to use in Normal mode. However, observe that the maximum interval between the external events must not exceed the resolution of the counter. If the interval between events are too long, the timer overflow interrupt or the prescaler must be used to extend the resolution for the capture unit.

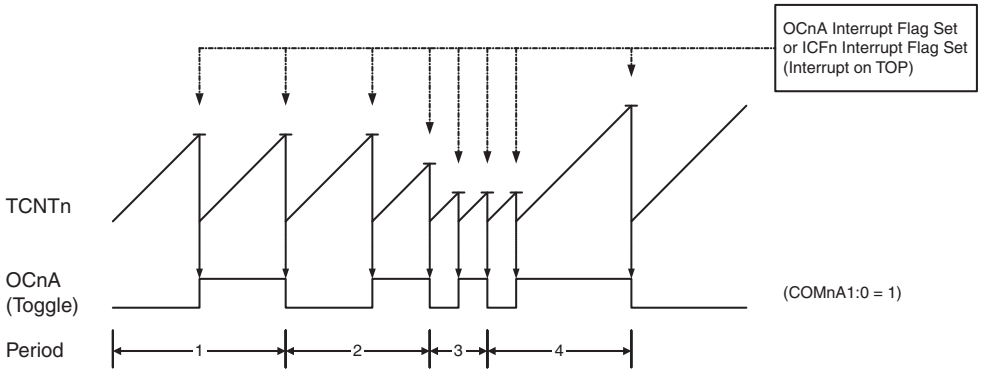
The Output Compare units can be used to generate interrupts at some given time. Using the Output Compare to generate waveforms in Normal mode is not recommended, since this will occupy too much of the CPU time.

12.8.2 Clear Timer on Compare Match (CTC) Mode

In *Clear Timer on Compare* or CTC mode (WGM13:0 = 4 or 12), the OCR1A or ICR1 Register are used to manipulate the counter resolution. In CTC mode the counter is cleared to zero when the counter value (TCNT1) matches either the OCR1A (WGM13:0 = 4) or the ICR1 (WGM13:0 = 12). The OCR1A or ICR1 define the top value for the counter, hence also its resolution. This mode allows greater control of the compare match output frequency. It also simplifies the operation of counting external events.

The timing diagram for the CTC mode is shown in Figure 12-6. The counter value (TCNT1) increases until a compare match occurs with either OCR1A or ICR1, and then counter (TCNT1) is cleared.

Figure 12-6. CTC Mode, Timing Diagram



An interrupt can be generated at each time the counter value reaches the TOP value by either using the OCF1A or ICF1 flag according to the register used to define the TOP value. If the interrupt is enabled, the interrupt handler routine can be used for updating the TOP value. However, changing the TOP to a value close to BOTTOM when the counter is running with none or a low prescaler value must be done with care since the CTC mode does not have the double buffering feature. If the new value written to OCR1A or ICR1 is lower than the current value of TCNT1, the counter will miss the compare match. The counter will then have to count to its maximum value (0xFFFF) and wrap around starting at 0x0000 before the compare match can occur. In many cases this feature is not desirable. An alternative will then be to use the fast PWM mode using OCR1A for defining TOP (WGM13:0 = 15) since the OCR1A then will be double buffered.

For generating a waveform output in CTC mode, the OCFA output can be set to toggle its logical level on each compare match by setting the Compare Output mode bits to toggle mode (COM1A1:0 = 1). The OCF1A value will not be visible on the port pin unless the data direction for the pin is set to output (DDR_OCF1A = 1). The waveform generated will have a maximum frequency of $f_{OC1A} = f_{clk\_I/O}/2$ when OCR1A is set to zero (0x0000). The waveform frequency is defined by the following equation:

$$f_{OCnA} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot (1 + OCRnA)}$$

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

As for the Normal mode of operation, the TOV1 flag is set in the same timer clock cycle that the counter counts from MAX to 0x0000.

12.8.3 Fast PWM Mode

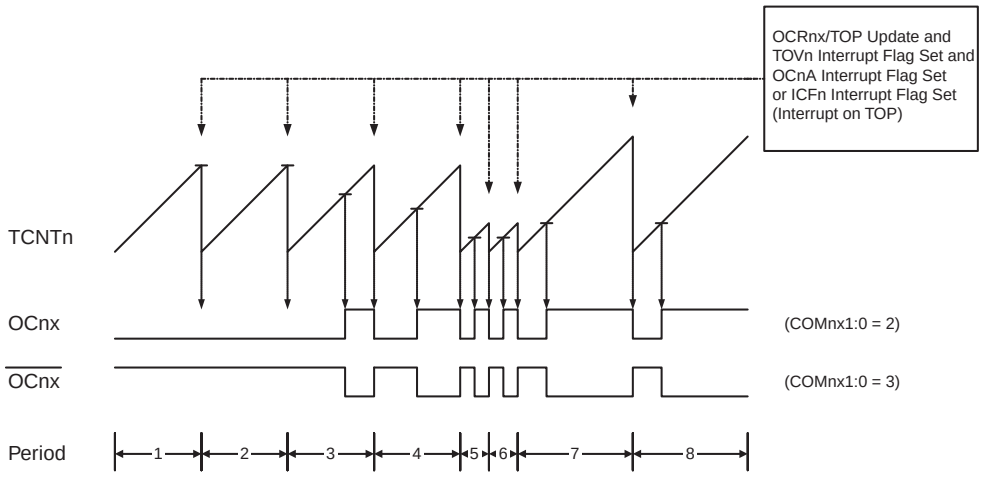
The *fast Pulse Width Modulation* or fast PWM mode (WGM13:0 = 5, 6, 7, 14, or 15) provides a high frequency PWM waveform generation option. The fast PWM differs from the other PWM options by its single-slope operation. The counter counts from BOTTOM to TOP then restarts from BOTTOM. In non-inverting Compare Output mode, the Output Compare (OC1x) is set on the compare match between TCNT1 and OCR1x, and cleared at TOP. In inverting Compare Output mode output is cleared on compare match and set at TOP. Due to the single-slope operation, the operating frequency of the fast PWM mode can be twice as high as the phase correct and phase and frequency correct PWM modes that use dual-slope operation. This high frequency makes the fast PWM mode well suited for power regulation, rectification, and DAC applications. High frequency allows physically small sized external components (coils, capacitors), hence reduces total system cost.

The PWM resolution for fast PWM can be fixed to 8-, 9-, or 10-bit, or defined by either ICR1 or OCR1A. The minimum resolution allowed is 2-bit (ICR1 or OCR1A set to 0x0003), and the maximum resolution is 16-bit (ICR1 or OCR1A set to MAX). The PWM resolution in bits can be calculated by using the following equation:

$$R_{FPWM} = \frac{\log(TOP+1)}{\log(2)}$$

In fast PWM mode the counter is incremented until the counter value matches either one of the fixed values 0x00FF, 0x01FF, or 0x03FF (WGM13:0 = 5, 6, or 7), the value in ICR1 (WGM13:0 = 14), or the value in OCR1A (WGM13:0 = 15). The counter is then cleared at the following timer clock cycle. The timing diagram for the fast PWM mode is shown in Figure 12-7. The figure shows fast PWM mode when OCR1A or ICR1 is used to define TOP. The TCNT1 value is in the timing diagram shown as a histogram for illustrating the single-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT1 slopes represent compare matches between OCR1x and TCNT1. The OC1x interrupt flag will be set when a compare match occurs.

Figure 12-7. Fast PWM Mode, Timing Diagram



The Timer/Counter Overflow Flag (TOV1) is set each time the counter reaches TOP. In addition the OCF1A or ICF1 flag is set at the same timer clock cycle as TOV1 is set when either OCR1A or ICR1 is used for defining the TOP value. If one of the interrupts are enabled, the interrupt handler routine can be used for updating the TOP and compare values.

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 OCR1x. Note that when using fixed TOP values the unused bits are masked to zero when any of the OCR1x Registers are written.

The procedure for updating ICR1 differs from updating OCR1A when used for defining the TOP value. The ICR1 Register is not double buffered. This means that if ICR1 is changed to a low value when the counter is running with none or a low prescaler value, there is a risk that the new ICR1 value written is lower than the current value of TCNT1. The result will then be that the counter will miss the compare match at the TOP value. The counter will then have to count to the MAX value (0xFFFF) and wrap around starting at 0x0000 before the compare match can occur.

The OCR1A Register however, is double buffered. This feature allows the OCR1A I/O location to be written anytime. When the OCR1A I/O location is written the value written will be put into the OCR1A Buffer Register. The OCR1A Compare Register will then be updated with the value in the Buffer Register at the next timer clock cycle the TCNT1 matches TOP. The update is done at the same timer clock cycle as the TCNT1 is cleared and the TOV1 flag is set.

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 fast PWM mode, the compare units allow generation of PWM waveforms on the OC1x pins. Setting the COM1x1:0 bits to two will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COM1x1:0 to three (see Table 12-2 on page 111). The actual OC1x value will only be visible on the port pin if the data direction for the port pin is set as output (DDR_OC1x). The PWM waveform is generated by setting (or clearing) the OC1x Register at the compare match between OCR1x and TCNT1, and clearing (or setting) the OC1x Register at the timer clock cycle the counter is cleared (changes from TOP to BOTTOM).

The PWM frequency for the output can be calculated by the following equation:

$$f_{OCnxPWM} = \frac{f_{clk\_I/O}}{N \cdot (1 + TOP)}$$

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

The extreme values for the OCR1x Register represents special cases when generating a PWM waveform output in the fast PWM mode. If the OCR1x is set equal to BOTTOM (0x0000) the output will be a narrow spike for each TOP+1 timer clock cycle. Setting the OCR1x equal to TOP will result in a constant high or low output (depending on the polarity of the output set by the COM1x1:0 bits.)

A frequency (with 50% duty cycle) waveform output in fast PWM mode can be achieved by setting OCF1A to toggle its logical level on each compare match (COM1A1:0 = 1). The waveform generated will have a maximum frequency of $f_{OC1A} = f_{clk\_I/O}/2$ when OCR1A is set to zero (0x0000). This feature is similar to the OCF1A toggle in CTC mode, except the double buffer feature of the Output Compare unit is enabled in the fast PWM mode.

12.8.4 Phase Correct PWM Mode

The *phase correct Pulse Width Modulation* or phase correct PWM mode (WGM13:0 = 1, 2, 3, 10, or 11) provides a high resolution phase correct PWM waveform generation option. The phase correct PWM mode is, like the phase and frequency correct PWM mode, based on a dual-slope operation. The counter counts repeatedly from BOTTOM (0x0000) to TOP and then from TOP to BOTTOM. In non-inverting Compare Output mode, the Output Compare (OC1x) is cleared on the compare match between TCNT1 and OCR1x while upcounting, and set on the compare match while downcounting. In inverting Output Compare mode, the operation is inverted. The dual-slope operation has lower maximum operation frequency than single slope operation. However, due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

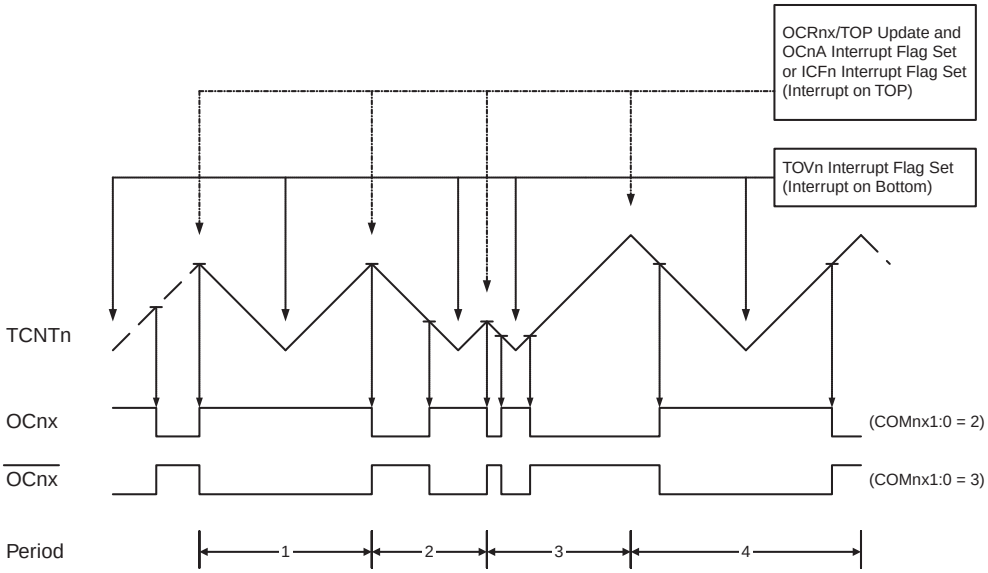
The PWM resolution for the phase correct PWM mode can be fixed to 8-, 9-, or 10-bit, or defined by either ICR1 or OCR1A. The minimum resolution allowed is 2-bit (ICR1 or OCR1A set to

0x0003), and the maximum resolution is 16-bit (ICR1 or OCR1A set to MAX). The PWM resolution in bits can be calculated by using the following equation:

$$R_{PCPWM} = \frac{\log(TOP + 1)}{\log(2)}$$

In phase correct PWM mode the counter is incremented until the counter value matches either one of the fixed values 0x00FF, 0x01FF, or 0x03FF (WGM13:0 = 1, 2, or 3), the value in ICR1 (WGM13:0 = 10), or the value in OCR1A (WGM13:0 = 11). The counter has then reached the TOP and changes the count direction. The TCNT1 value will be equal to TOP for one timer clock cycle. The timing diagram for the phase correct PWM mode is shown on Figure 12-8 on page 102. The figure shows phase correct PWM mode when OCR1A or ICR1 is used to define TOP. The TCNT1 value is in the timing diagram shown as a histogram for illustrating the dual-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT1 slopes represent compare matches between OCR1x and TCNT1. The OC1x interrupt flag will be set when a compare match occurs.

Figure 12-8. Phase Correct PWM Mode, Timing Diagram



The Timer/Counter Overflow Flag (TOV1) is set each time the counter reaches BOTTOM. When either OCR1A or ICR1 is used for defining the TOP value, the OCF1A or ICF1 flag is set accordingly at the same timer clock cycle as the OCR1x Registers are updated with the double buffer value (at TOP). The interrupt flags can 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 OCR1x. Note that when using fixed TOP values, the unused bits are masked to zero when any of the OCR1x Registers are written. As the third period shown in Figure 12-8 illustrates, changing the TOP actively while the Timer/Counter is running in the phase correct mode can result in an unsymmetrical output. The reason for this can be found in the time of update of the OCR1x Register. Since the OCR1x update occurs at TOP, the PWM period starts and ends at TOP. This

implies that the length of the falling slope is determined by the previous TOP value, while the length of the rising slope is determined by the new TOP value. When these two values differ the two slopes of the period will differ in length. The difference in length gives the unsymmetrical result on the output.

It is recommended to use the phase and frequency correct mode instead of the phase correct mode when changing the TOP value while the Timer/Counter is running. When using a static TOP value there are practically no differences between the two modes of operation.

In phase correct PWM mode, the compare units allow generation of PWM waveforms on the OC1x pins. Setting the COM1x1:0 bits to two will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COM1x1:0 to three (See Table 12-3 on page 111). The actual OC1x value will only be visible on the port pin if the data direction for the port pin is set as output (DDR_OC1x). The PWM waveform is generated by setting (or clearing) the OC1x Register at the compare match between OCR1x and TCNT1 when the counter increments, and clearing (or setting) the OC1x Register at compare match between OCR1x and TCNT1 when the counter decrements. The PWM frequency for the output when using phase correct PWM can be calculated by the following equation:

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

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

The extreme values for the OCR1x Register represent special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCR1x is set equal to BOTTOM the output will be continuously low and if set equal to TOP the output will be continuously high for non-inverted PWM mode. For inverted PWM the output will have the opposite logic values.

12.8.5 Phase and Frequency Correct PWM Mode

The *phase and frequency correct Pulse Width Modulation,* or phase and frequency correct PWM mode (WGM13:0 = 8 or 9) provides a high resolution phase and frequency correct PWM waveform generation option. The phase and frequency correct PWM mode is, like the phase correct PWM mode, based on a dual-slope operation. The counter counts repeatedly from BOTTOM (0x0000) to TOP and then from TOP to BOTTOM. In non-inverting Compare Output mode, the Output Compare (OC1x) is cleared on the compare match between TCNT1 and OCR1x while upcounting, and set on the compare match while downcounting. In inverting Compare Output mode, the operation is inverted. The dual-slope operation gives a lower maximum operation frequency compared to the single-slope operation. However, due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

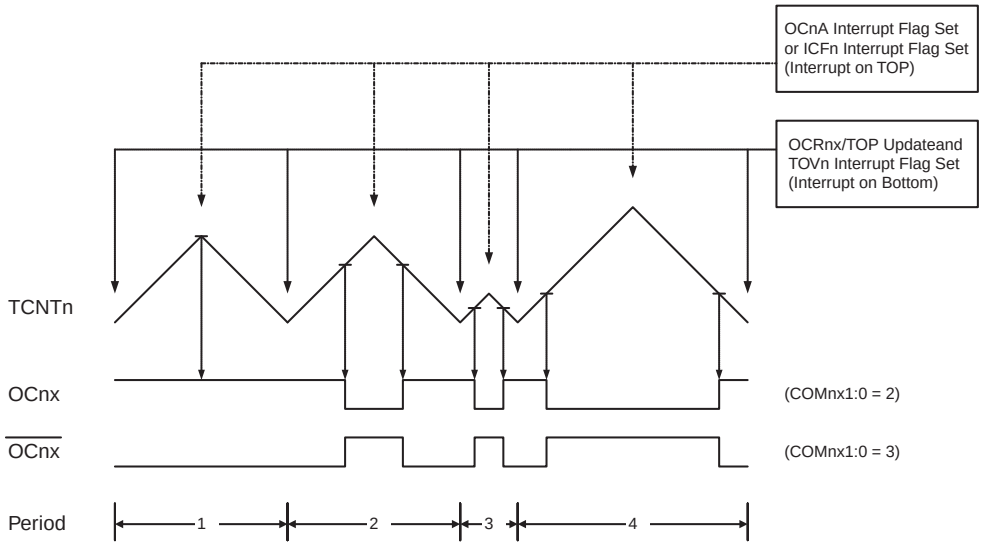
The main difference between the phase correct, and the phase and frequency correct PWM mode is the time the OCR1x Register is updated by the OCR1x Buffer Register, (see Figure 12-8 on page 102 and Figure 12-9 on page 104).

The PWM resolution for the phase and frequency correct PWM mode can be defined by either ICR1 or OCR1A. The minimum resolution allowed is 2-bit (ICR1 or OCR1A set to 0x0003), and the maximum resolution is 16-bit (ICR1 or OCR1A set to MAX). The PWM resolution in bits can be calculated using the following equation:

$$R_{PFCPWM} = \frac{\log(TOP + 1)}{\log(2)}$$

In phase and frequency correct PWM mode the counter is incremented until the counter value matches either the value in ICR1 (WGM13:0 = 8), or the value in OCR1A (WGM13:0 = 9). The counter has then reached the TOP and changes the count direction. The TCNT1 value will be equal to TOP for one timer clock cycle. The timing diagram for the phase correct and frequency correct PWM mode is shown on Figure 12-9 on page 104. The figure shows phase and frequency correct PWM mode when OCR1A or ICR1 is used to define TOP. The TCNT1 value is in the timing diagram shown as a histogram for illustrating the dual-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT1 slopes represent compare matches between OCR1x and TCNT1. The OC1x interrupt flag will be set when a compare match occurs.

Figure 12-9. Phase and Frequency Correct PWM Mode, Timing Diagram



The Timer/Counter Overflow Flag (TOV1) is set at the same timer clock cycle as the OCR1x Registers are updated with the double buffer value (at BOTTOM). When either OCR1A or ICR1 is used for defining the TOP value, the OCF1A 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 OCR1x.

As Figure 12-9 shows the output generated is, in contrast to the phase correct mode, symmetrical in all periods. Since the OCR1x 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 OC1x pins. Setting the COM1x1:0 bits to two will produce a non-inverted PWM and

an inverted PWM output can be generated by setting the COM1x1:0 to three (See Table 12-4 on page 112). The actual OC1Fx value will only be visible on the port pin if the data direction for the port pin is set as output (DDR_OCF1x). The PWM waveform is generated by setting (or clearing) the OCF1x Register at the compare match between OCR1x and TCNT1 when the counter increments, and clearing (or setting) the OCF1x Register at compare match between OCR1x 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:

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

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

The extreme values for the OCR1x Register represents special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCR1x 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.

12.9 Timer/Counter Timing Diagrams

The Timer/Counter is a synchronous design and the timer clock (clk_{T1}) is therefore shown as a clock enable signal in the following figures. The figures include information on when interrupt flags are set, and when the OCR1x Register is updated with the OCR1x buffer value (only for modes utilizing double buffering). Figure 12-10 shows a timing diagram for the setting of OCF1x.

Figure 12-10. Timer/Counter Timing Diagram, Setting of OCF1x, no Prescaling

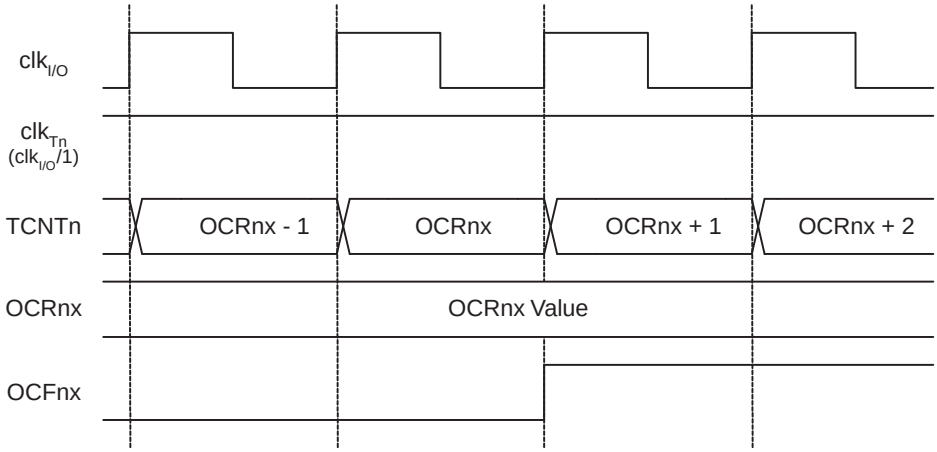


Figure 12-11 shows the same timing data, but with the prescaler enabled.

Figure 12-11. Timer/Counter Timing Diagram, Setting of OCF1x, with Prescaler ($f_{clk\_I/O}/8$)

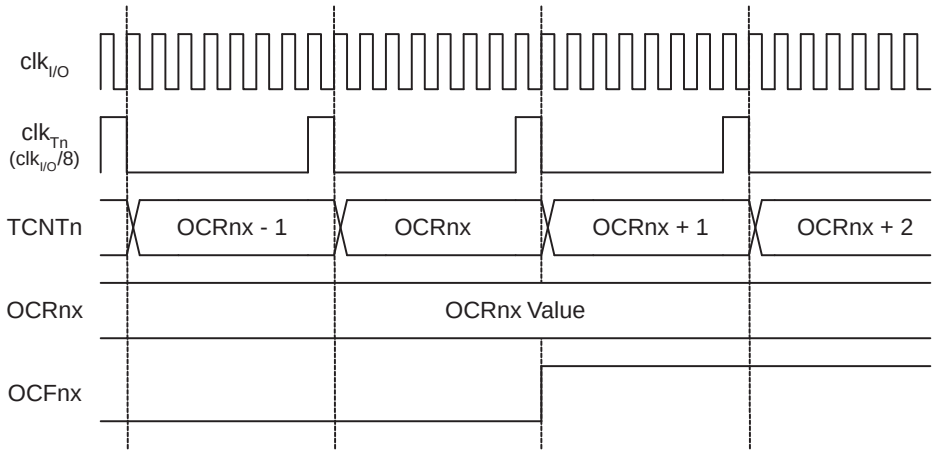


Figure 12-12 on page 106 shows the count sequence close to TOP in various modes. When using phase and frequency correct PWM mode the OCR1x Register is updated at BOTTOM. The timing diagrams will be the same, but TOP should be replaced by BOTTOM, TOP-1 by BOTTOM+1 and so on. The same renaming applies for modes that set the TOV1 flag at BOTTOM.

Figure 12-12. Timer/Counter Timing Diagram, no Prescaling

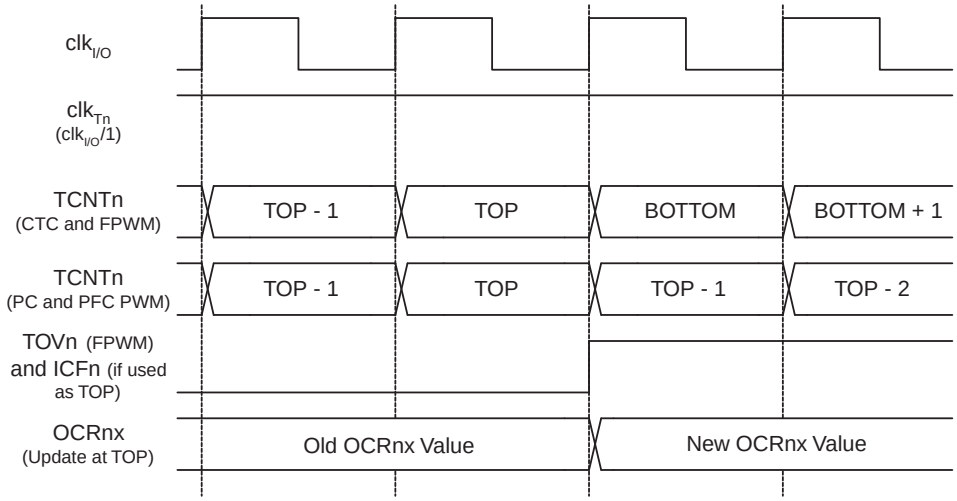
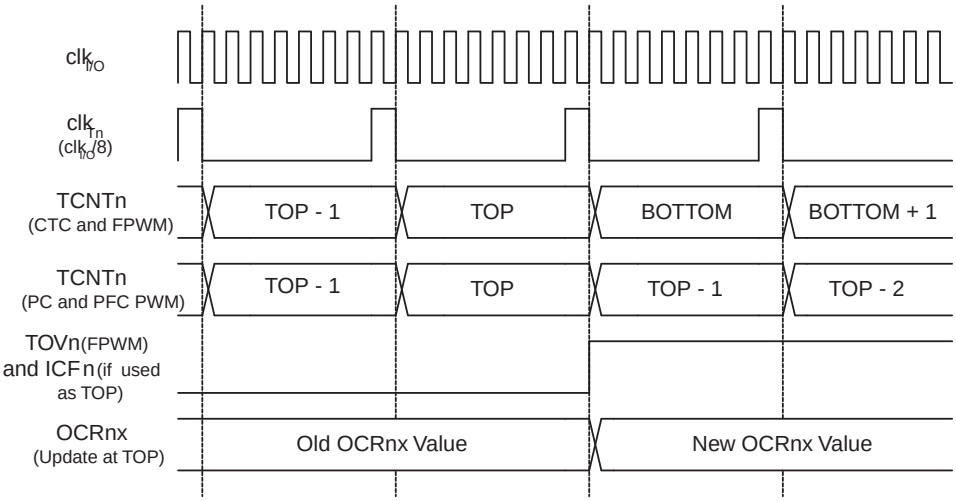


Figure 12-13 shows the same timing data, but with the prescaler enabled.

Figure 12-13. Timer/Counter Timing Diagram, with Prescaler ($f_{clk\_I/O}/8$)



12.10 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.

The following code examples show how to access the 16-bit timer registers assuming that no interrupts updates the temporary register. The same principle can be used directly for accessing

the OCR1A/B and ICR1 Registers. Note that when using “C”, the compiler handles the 16-bit access.

Assembly Code Examples[1]

```
...
; Set TCNT1 to 0x01FF
ldi r17,0x01
ldi r16,0xFF
out TCNT1H,r17
out TCNT1L,r16
; Read TCNT1 into r17:r16
in  r16,TCNT1L
in  r17,TCNT1H
...
```

C Code Examples[1]

```
unsigned int i;
...
/* Set TCNT1 to 0x01FF */
TCNT1 = 0x1FF;
/* Read TCNT1 into i */
i = TCNT1;
...
```

Note: 1. See “Code Examples” on page 7.

The assembly code example returns the TCNT1 value in the r17:r16 register pair.

It is important to notice that accessing 16-bit registers are atomic operations. If an interrupt occurs between the two instructions accessing the 16-bit register, and the interrupt code updates the temporary register by accessing the same or any other of the 16-bit timer registers, then the result of the access outside the interrupt will be corrupted. Therefore, when both the main code and the interrupt code update the temporary register, the main code must disable the interrupts during the 16-bit access.

The following code examples show how to do an atomic read of the TCNT1 Register contents. Reading any of the OCR1A/B or ICR1 Registers can be done by using the same principle.

Assembly Code Example[1]

```
TIM16_ReadTCNT1:
  ; Save global interrupt flag
  in  r18,SREG
  ; Disable interrupts
  cli
  ; Read TCNT1 into r17:r16
  in  r16,TCNT1L
  in  r17,TCNT1H
  ; Restore global interrupt flag
  out SREG,r18
  ret
```

C Code Example[1]

```
unsigned int TIM16_ReadTCNT1( void )
{
  unsigned char sreg;
  unsigned int i;
  /* Save global interrupt flag */
  sreg = SREG;
  /* Disable interrupts */
  __disable_interrupt();
  /* Read TCNT1 into i */
  i = TCNT1;
  /* Restore global interrupt flag */
  SREG = sreg;
  return i;
}
```

Note: 1. See “Code Examples” on page 7.

The assembly code example returns the TCNT1 value in the r17:r16 register pair.

The following code examples show how to do an atomic write of the TCNT1 Register contents. Writing any of the OCR1A/B or ICR1 Registers can be done by using the same principle.

Assembly Code Example[1]

```
TIM16_WriteTCNT1:
  ; Save global interrupt flag
  in  r18,SREG
  ; Disable interrupts
  cli
  ; Set TCNT1 to r17:r16
  out TCNT1H,r17
  out TCNT1L,r16
  ; Restore global interrupt flag
  out SREG,r18
  ret
```

C Code Example[1]

```
void TIM16_WriteTCNT1( unsigned int i )
{
  unsigned char sreg;
  unsigned int i;
  /* Save global interrupt flag */
  sreg = SREG;
  /* Disable interrupts */
  __disable_interrupt();
  /* Set TCNT1 to i */
  TCNT1 = i;
  /* Restore global interrupt flag */
  SREG = sreg;
}
```

Note: 1. See “Code Examples” on page 7.

The assembly code example requires that the r17:r16 register pair contains the value to be written to TCNT1.

12.10.1 Reusing the Temporary High Byte Register

If writing to more than one 16-bit register where the high byte is the same for all registers written, then the high byte only needs to be written once. However, note that the same rule of atomic operation described previously also applies in this case.

12.11 Register Description

12.11.1 TCCR1A – Timer/Counter1 Control Register A

Bit	7	6	5	4	3	2	1	0	
0x2F (0x4F)	COM1A1	COM1A0	COM1B1	COM1B0	–	–	WGM11	WGM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – COM1A1:0: Compare Output Mode for Channel A**

- **Bit 5:4 – COM1B1:0: Compare Output Mode for Channel B**

The COM1A1:0 and COM1B1:0 control the Output Compare pins (OC1A and OC1B respectively) behavior. If one or both of the COM1A1:0 bits are written to one, the OC1A output overrides the normal port functionality of the I/O pin it is connected to. If one or both of the COM1B1:0 bit are written to one, the OC1B output overrides the normal port functionality of the I/O pin it is connected to. However, note that the *Data Direction Register* (DDR) bit corresponding to the OC1A or OC1B pin must be set in order to enable the output driver.

When the OC1A or OC1B is connected to the pin, the function of the COM1x1:0 bits is dependent of the WGM13:0 bits setting. Table 12-2 shows the COM1x1:0 bit functionality when the WGM13:0 bits are set to a Normal or a CTC mode (non-PWM).

Table 12-2. Compare Output Mode, non-PWM

COM1A1/COM1B1	COM1A0/COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	Toggle OC1A/OC1B on Compare Match.
1	0	Clear OC1A/OC1B on Compare Match (Set output to low level).
1	1	Set OC1A/OC1B on Compare Match (Set output to high level).

Table 12-3 shows the COM1x1:0 bit functionality when the WGM13:0 bits are set to the fast PWM mode.

Table 12-3. Compare Output Mode, Fast PWM[1]

COM1A1/COM1B1	COM1A0/COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	WGM13=0: Normal port operation, OC1A/OC1B disconnected. WGM13=1: Toggle OC1A on Compare Match, OC1B reserved.
1	0	Clear OC1A/OC1B on Compare Match, set OC1A/OC1B at TOP
1	1	Set OC1A/OC1B on Compare Match, clear OC1A/OC1B at TOP

Note: 1. A special case occurs when OCR1A/OCR1B equals TOP and COM1A1/COM1B1 is set. In this case the compare match is ignored, but the set or clear is done at TOP. See “Fast PWM Mode” on page 99. for more details.

Table 12-4 shows the COM1x1:0 bit functionality when the WGM13:0 bits are set to the phase correct or the phase and frequency correct, PWM mode.

Table 12-4. Compare Output Mode, Phase Correct and Phase and Frequency Correct PWM[1]

COM1A1/COM1B1	COM1A0/COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	WGM13=0: Normal port operation, OC1A/OC1B disconnected. WGM13=1: Toggle OC1A on Compare Match, OC1B reserved.
1	0	Clear OC1A/OC1B on Compare Match when up-counting. Set OC1A/OC1B on Compare Match when downcounting.
1	1	Set OC1A/OC1B on Compare Match when up-counting. Clear OC1A/OC1B on Compare Match when downcounting.

Note: 1. A special case occurs when OCR1A/OCR1B equals TOP and COM1A1/COM1B1 is set. See “Phase Correct PWM Mode” on page 101. for more details.

- **Bit 1:0 – WGM11:0: Waveform Generation Mode**

Combined with the WGM13:2 bits found in the TCCR1B Register, these bits control the counting sequence of the counter, the source for maximum (TOP) counter value, and what type of waveform generation to be used, see Table 12-5. Modes of operation supported by the Timer/Counter unit are: Normal mode (counter), Clear Timer on Compare match (CTC) mode, and three types of Pulse Width Modulation (PWM) modes. (See “Modes of Operation” on page 97.).

Table 12-5. Waveform Generation Mode Bit Description[1]

Mode	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Timer/Counter Mode of Operation	TOP	Update of OCR1x at	TOV1 Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	TOP	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	TOP	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	TOP	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICR1	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	Immediate	MAX
13	1	1	0	1	(Reserved)	–	–	–
14	1	1	1	0	Fast PWM	ICR1	TOP	TOP
15	1	1	1	1	Fast PWM	OCR1A	TOP	TOP

Note: 1. The CTC1 and PWM11:0 bit definition names are obsolete. Use the WGM12:0 definitions. However, the functionality and location of these bits are compatible with previous versions of the timer.

12.11.2 TCCR1B – Timer/Counter1 Control Register B

Bit	7	6	5	4	3	2	1	0	
0x2E (0x4E)	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 12-10 on page 105 and Figure 12-11 on page 106.

Table 12-6. 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.

12.11.3 TCCR1C – Timer/Counter1 Control Register C

Bit	7	6	5	4	3	2	1	0	
0x22 (ox42)	FOC1A	FOC1B	–	–	–	–	–	–	TCCR1C
Read/Write	W	W	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – FOC1A: Force Output Compare for Channel A**

- **Bit 6 – FOC1B: Force Output Compare for Channel B**

The FOC1A/FOC1B bits are only active when the WGM13:0 bits specifies a non-PWM mode. However, for ensuring compatibility with future devices, these bits must be set to zero when TCCR1A is written when operating in a PWM mode. When writing a logical one to the FOC1A/FOC1B bit, an immediate compare match is forced on the Waveform Generation unit. The OC1A/OC1B output is changed according to its COM1x1:0 bits setting. Note that the FOC1A/FOC1B bits are implemented as strobes. Therefore it is the value present in the COM1x1:0 bits that determine the effect of the forced compare.

A FOC1A/FOC1B strobe will not generate any interrupt nor will it clear the timer in Clear Timer on Compare match (CTC) mode using OCR1A as TOP.

The FOC1A/FOC1B bits are always read as zero.

12.11.4 TCNT1H and TCNT1L – Timer/Counter1

Bit	7	6	5	4	3	2	1	0	
0x2D (0x4D)	TCNT1[15:8]								TCNT1H
0x2C (0x4C)	TCNT1[7:0]								TCNT1L
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 two *Timer/Counter* I/O locations (TCNT1H and TCNT1L, combined TCNT1) give direct access, both for read and for write operations, to the Timer/Counter unit 16-bit counter. To ensure that both the high and low bytes are read and written simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary high byte register (TEMP). This temporary register is shared by all the other 16-bit registers. See “Accessing 16-bit Registers” on page 107.

Modifying the counter (TCNT1) while the counter is running introduces a risk of missing a compare match between TCNT1 and one of the OCR1x Registers.

Writing to the TCNT1 Register blocks (removes) the compare match on the following timer clock for all compare units.

12.11.5 OCR1AH and OCR1AL – Output Compare Register 1 A

Bit	7	6	5	4	3	2	1	0	
0x2B (0x4B)	OCR1A[15:8]								OCR1AH
0x2A (0x4A)	OCR1A[7:0]								OCR1AL
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	

12.11.6 OCR1BH and OCR1BL – Output Compare Register 1 B

Bit	7	6	5	4	3	2	1	0	
0x29 (0x49)	OCR1B[15:8]								OCR1BH
0x28 (0x48)	OCR1B[7:0]								OCR1BL
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 Registers contain a 16-bit value that is continuously compared with the counter value (TCNT1). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC1x pin.

The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary high byte register (TEMP). This temporary register is shared by all the other 16-bit registers. See “Accessing 16-bit Registers” on page 107.

12.11.7 ICR1H and ICR1L – Input Capture Register 1

Bit	7	6	5	4	3	2	1	0	
0x25 (0x45)	ICR1[15:8]								ICR1H
0x24 (0x44)	ICR1[7:0]								ICR1L
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 Input Capture is updated with the counter (TCNT1) value each time an event occurs on the ICP1 pin (or optionally on the Analog Comparator output for Timer/Counter1). The Input Capture can be used for defining the counter TOP value.

The Input Capture Register is 16-bit in size. To ensure that both the high and low bytes are read simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary high byte register (TEMP). This temporary register is shared by all the other 16-bit registers. See “Accessing 16-bit Registers” on page 107.

12.11.8 TIMSK – Timer/Counter Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
0x39 (0x59)	TOIE1	OCIE1A	OCIE1B	–	ICIE1	OCIE0B	TOIE0	OCIE0A	TIMSK
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – TOIE1: Timer/Counter1, Overflow Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter1 Overflow interrupt is enabled. The corresponding Interrupt Vector (See “Interrupts” on page 48.) is executed when the TOV1 flag, located in TIFR, is set.

- **Bit 6 – OCIE1A: Timer/Counter1, Output Compare A Match Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter1 Output Compare A Match interrupt is enabled. The corresponding Interrupt Vector (See “Interrupts” on page 48.) is executed when the OCF1A flag, located in TIFR, is set.

- **Bit 5 – OCIE1B: Timer/Counter1, Output Compare B Match Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter1 Output Compare B Match interrupt is enabled. The corresponding Interrupt Vector (See “Interrupts” on page 48.) is executed when the OCF1B flag, located in TIFR, is set.

- **Bit 3 – ICIE1: Timer/Counter1, Input Capture Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter1 Input Capture interrupt is enabled. The corresponding Interrupt Vector (See “Interrupts” on page 48.) is executed when the ICF1 flag, located in TIFR, is set.

12.11.9 TIFR – Timer/Counter Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x38 (0x58)	TOV1	OCF1A	OCF1B	–	ICF1	OCF0B	TOV0	OCF0A	TIFR
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – TOV1: Timer/Counter1, Overflow Flag**

The setting of this flag is dependent of the WGM13:0 bits setting. In Normal and CTC modes, the TOV1 flag is set when the timer overflows. Refer to Table 12-5 on page 113 for the TOV1 flag behavior when using another WGM13:0 bit setting.

TOV1 is automatically cleared when the Timer/Counter1 Overflow Interrupt Vector is executed. Alternatively, TOV1 can be cleared by writing a logic one to its bit location.

- **Bit 6 – OCF1A: Timer/Counter1, Output Compare A Match Flag**

This flag is set in the timer clock cycle after the counter (TCNT1) value matches the Output Compare Register A (OCR1A).

Note that a Forced Output Compare (FOC1A) strobe will not set the OCF1A flag.

OCF1A is automatically cleared when the Output Compare Match A Interrupt Vector is executed. Alternatively, OCF1A can be cleared by writing a logic one to its bit location.

- **Bit 5 – OCF1B: Timer/Counter1, Output Compare B Match Flag**

This flag is set in the timer clock cycle after the counter (TCNT1) value matches the Output Compare Register B (OCR1B).

Note that a Forced Output Compare (FOC1B) strobe will not set the OCF1B flag.

OCF1B is automatically cleared when the Output Compare Match B Interrupt Vector is executed. Alternatively, OCF1B can be cleared by writing a logic one to its bit location.

- **Bit 3 – ICF1: Timer/Counter1, Input Capture Flag**

This flag is set when a capture event occurs on the ICP1 pin. When the Input Capture Register (ICR1) is set by the WGM13:0 to be used as the TOP value, the ICF1 flag is set when the counter reaches the TOP value.

ICF1 is automatically cleared when the Input Capture Interrupt Vector is executed. Alternatively, ICF1 can be cleared by writing a logic one to its bit location.

13. Timer/Counter0 and Timer/Counter1 Prescalers

Timer/Counter1 and Timer/Counter0 share the same prescaler module, but the Timer/Counters can have different prescaler settings. The description below applies to both Timer/Counter1 and Timer/Counter0.

13.1 Internal Clock Source

The Timer/Counter can be clocked directly by the system clock (by setting the CSn2:0 = 1). This provides the fastest operation, with a maximum Timer/Counter clock frequency equal to system clock frequency ($f_{CLK\_I/O}$). Alternatively, one of four taps from the prescaler can be used as a clock source. The prescaled clock has a frequency of either $f_{CLK\_I/O}/8$, $f_{CLK\_I/O}/64$, $f_{CLK\_I/O}/256$, or $f_{CLK\_I/O}/1024$.

13.2 Prescaler Reset

The prescaler is free running, i.e., operates independently of the Clock Select logic of the Timer/Counter, and it is shared by Timer/Counter1 and Timer/Counter0. Since the prescaler is not affected by the Timer/Counter's clock select, the state of the prescaler will have implications for situations where a prescaled clock is used. One example of prescaling artifacts occurs when the timer is enabled and clocked by the prescaler (6 > CSn2:0 > 1). The number of system clock cycles from when the timer is enabled to the first count occurs can be from 1 to N+1 system clock cycles, where N equals the prescaler divisor (8, 64, 256, or 1024).

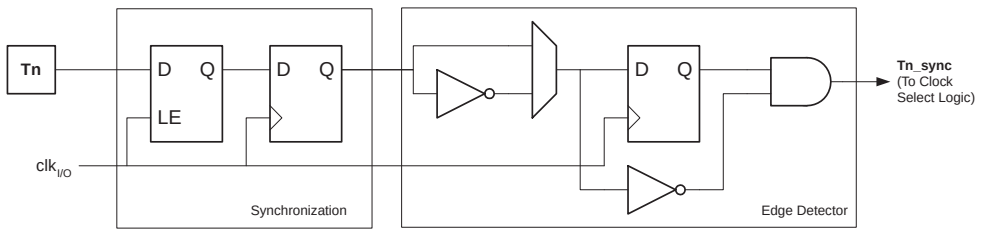
It is possible to use the prescaler reset for synchronizing the Timer/Counter to program execution. However, care must be taken if the other Timer/Counter that shares the same prescaler also uses prescaling. A prescaler reset will affect the prescaler period for all Timer/Counters it is connected to.

13.3 External Clock Source

An external clock source applied to the T1/T0 pin can be used as Timer/Counter clock (clk_{T1}/clk_{T0}). The T1/T0 pin is sampled once every system clock cycle by the pin synchronization logic. The synchronized (sampled) signal is then passed through the edge detector. Figure 13-1 shows a functional equivalent block diagram of the T1/T0 synchronization and edge detector logic. The registers are clocked at the positive edge of the internal system clock ($clk_{I/O}$). The latch is transparent in the high period of the internal system clock.

The edge detector generates one clk_{T1}/clk_{T0} pulse for each positive (CSn2:0 = 7) or negative (CSn2:0 = 6) edge it detects.

Figure 13-1. T1/T0 Pin Sampling



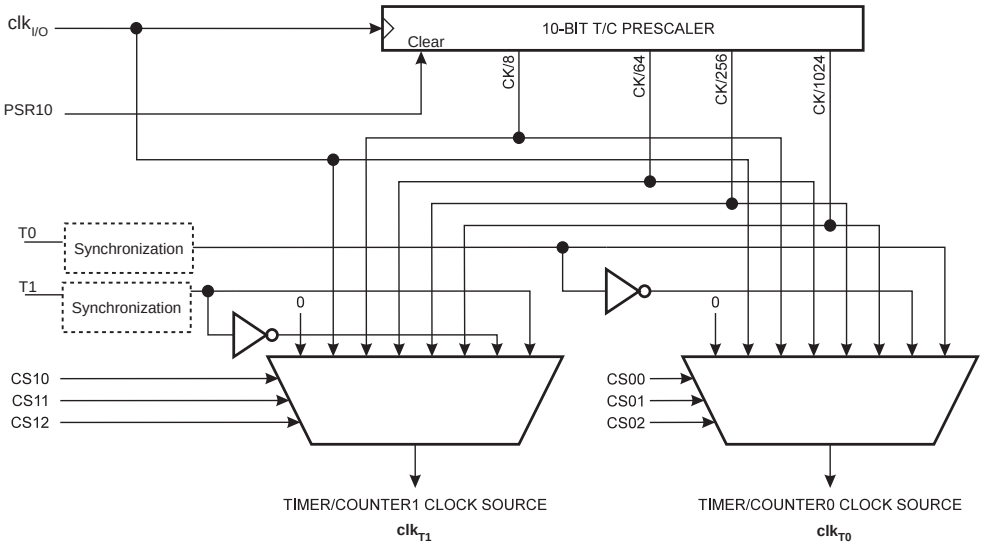
The synchronization and edge detector logic introduces a delay of 2.5 to 3.5 system clock cycles from an edge has been applied to the T1/T0 pin to the counter is updated.

Enabling and disabling of the clock input must be done when T1/T0 has been stable for at least one system clock cycle, otherwise it is a risk that a false Timer/Counter clock pulse is generated.

Each half period of the external clock applied must be longer than one system clock cycle to ensure correct sampling. The external clock must be guaranteed to have less than half the system clock frequency ($f_{ExtClk} < f_{clk\_I/O}/2$) given a 50/50% duty cycle. Since the edge detector uses sampling, the maximum frequency of an external clock it can detect is half the sampling frequency (Nyquist sampling theorem). However, due to variation of the system clock frequency and duty cycle caused by Oscillator source (crystal, resonator, and capacitors) tolerances, it is recommended that maximum frequency of an external clock source is less than $f_{clk\_I/O}/2.5$.

An external clock source can not be prescaled.

Figure 13-2. Prescaler for Timer/Counter0 and Timer/Counter1[(1)]



Note: 1. The synchronization logic on the input pins (T1/T0) is shown in Figure 13-1 on page 118.

13.4 Register Description

13.4.1 GTCCR – General Timer/Counter Control Register

Bit	7	6	5	4	3	2	1	0	
0x23 (0x43)	—	—	—	—	—	—	—	PSR10	GTCCR
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7..1 – Res: Reserved Bits**

These bits are reserved bits in the ATtiny2313A/4313 and will always read as zero.

- **Bit 0 – PSR10: Prescaler Reset Timer/Counter1 and Timer/Counter0**

When this bit is one, Timer/Counter1 and Timer/Counter0 prescaler will be Reset. This bit is normally cleared immediately by hardware. Note that Timer/Counter1 and Timer/Counter0 share the same prescaler and a reset of this prescaler will affect both timers.

14. USART

14.1 Features

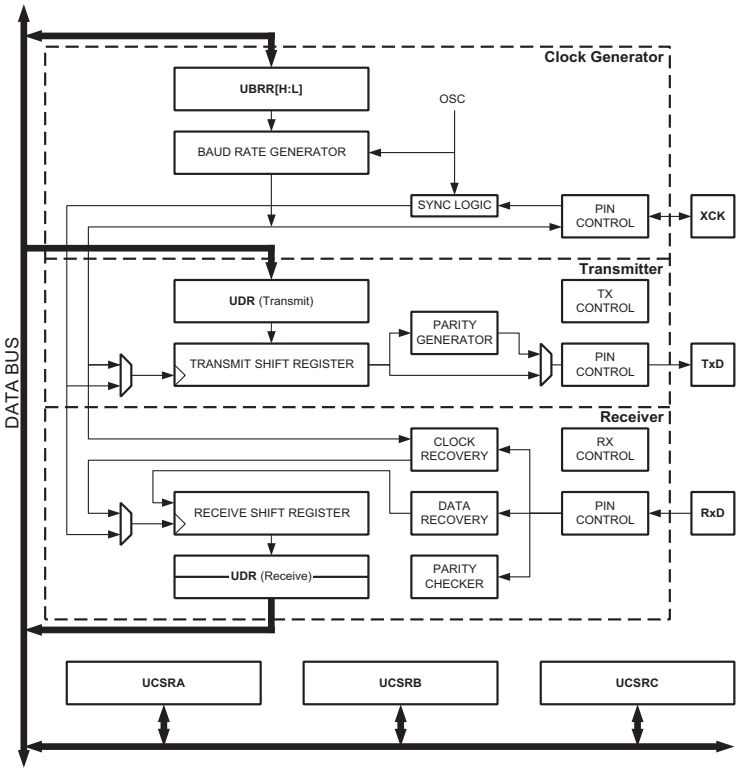
- **Full Duplex Operation (Independent Serial Receive and Transmit Registers)**
- **Asynchronous or Synchronous Operation**
- **Master or Slave Clocked Synchronous Operation**
- **High Resolution Baud Rate Generator**
- **Supports Serial Frames with 5, 6, 7, 8, or 9 Data Bits and 1 or 2 Stop Bits**
- **Odd or Even Parity Generation and Parity Check Supported by Hardware**
- **Data OverRun Detection**
- **Framing Error Detection**
- **Noise Filtering Includes False Start Bit Detection and Digital Low Pass Filter**
- **Three Separate Interrupts on TX Complete, TX Data Register Empty and RX Complete**
- **Multi-processor Communication Mode**
- **Double Speed Asynchronous Communication Mode**

14.2 Overview

The Universal Synchronous and Asynchronous serial Receiver and Transmitter (USART) is a highly flexible serial communication device.

A simplified block diagram of the USART Transmitter is shown in Figure 14-1. CPU accessible I/O Registers and I/O pins are shown in bold.

Figure 14-1. USART Block Diagram[1]



Note: 1. Refer to Figure 1-1 on page 2, Table 10-9 on page 68, and Table 10-6 on page 66 for USART pin placement.

The dashed boxes in the block diagram separate the three main parts of the USART (listed from the top): Clock Generator, Transmitter and Receiver. Control registers are shared by all units. The Clock Generation logic consists of synchronization logic for external clock input used by synchronous slave operation, and the baud rate generator. The XCK (Transfer Clock) pin is only used by synchronous transfer mode. The Transmitter consists of a single write buffer, a serial Shift Register, Parity Generator and Control logic for handling different serial frame formats. The write buffer allows a continuous transfer of data without any delay between frames. The Receiver is the most complex part of the USART module due to its clock and data recovery units. The recovery units are used for asynchronous data reception. In addition to the recovery units, the Receiver includes a Parity Checker, Control logic, a Shift Register and a two level receive buffer (UDR). The Receiver supports the same frame formats as the Transmitter, and can detect Frame Error, Data OverRun and Parity Errors.

14.2.1 AVR USART vs. AVR UART – Compatibility

The USART is fully compatible with the AVR UART regarding:

- Bit locations inside all USART Registers.
- Baud Rate Generation.
- Transmitter Operation.
- Transmit Buffer Functionality.
- Receiver Operation.

However, the receive buffering has two improvements that will affect the compatibility in some special cases:

- A second Buffer Register has been added. The two Buffer Registers operate as a circular FIFO buffer. Therefore the UDR must only be read once for each incoming data! More important is the fact that the error flags (FE and DOR) and the ninth data bit (RXB8) are buffered with the data in the receive buffer. Therefore the status bits must always be read before the UDR Register is read. Otherwise the error status will be lost since the buffer state is lost.
- The Receiver Shift Register can now act as a third buffer level. This is done by allowing the received data to remain in the serial Shift Register (see Figure 14-1) if the Buffer Registers are full, until a new start bit is detected. The USART is therefore more resistant to Data OverRun (DOR) error conditions.

The following control bits have changed name, but have same functionality and register location:

- CHR9 is changed to UCSZ2.
- OR is changed to DOR.

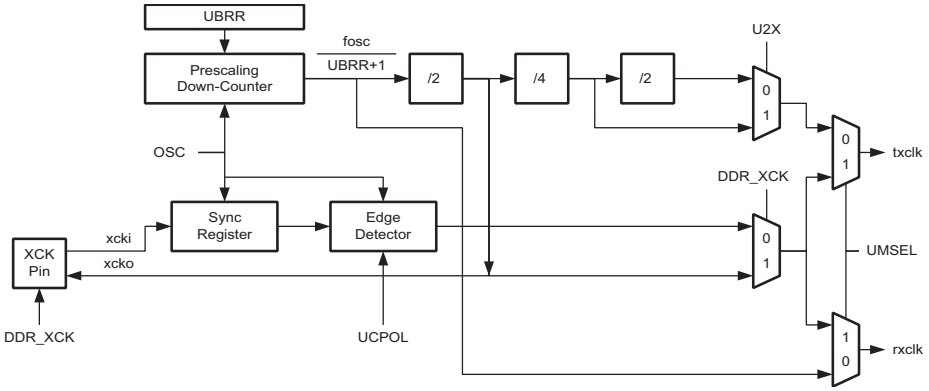
14.3 Clock Generation

The Clock Generation logic generates the base clock for the Transmitter and Receiver. The USART supports four modes of clock operation: Normal asynchronous, Double Speed asynchronous, Master synchronous and Slave synchronous mode. The UMSEL bit in USART Control and Status Register C (UCSRC) selects between asynchronous and synchronous operation. Double Speed (asynchronous mode only) is controlled by the U2X found in the UCSRA Register. When using synchronous mode (UMSEL = 1), the Data Direction Register for the XCK

pin (DDR_XCK) controls whether the clock source is internal (Master mode) or external (Slave mode). The XCK pin is only active when using synchronous mode.

Figure 14-2 shows a block diagram of the clock generation logic.

Figure 14-2. Clock Generation Logic, Block Diagram



Signal description:

txclk	Transmitter clock (Internal Signal).
rxclk	Receiver base clock (Internal Signal).
xcki	Input from XCK pin (internal Signal). Used for synchronous slave operation.
xcko	Clock output to XCK pin (Internal Signal). Used for synchronous master operation.
fosc	XTAL pin frequency (System Clock).

14.3.1 Internal Clock Generation – The Baud Rate Generator

Internal clock generation is used for the asynchronous and the synchronous master modes of operation. The description in this section refers to Figure 14-2.

The USART Baud Rate Register (UBRR) and the down-counter connected to it function as a programmable prescaler or baud rate generator. The down-counter, running at system clock (f_{osc}), is loaded with the UBRR value each time the counter has counted down to zero or when the UBRRL Register is written. A clock is generated each time the counter reaches zero. This clock is the baud rate generator clock output (= f_{osc}/(UBRR+1)). The Transmitter divides the baud rate generator clock output by 2, 8 or 16 depending on mode. The baud rate generator output is used directly by the Receiver's clock and data recovery units. However, the recovery units use a state machine that uses 2, 8 or 16 states depending on mode set by the state of the UMSEL, U2X and DDR_XCK bits.

Table 14-1 contains equations for calculating the baud rate (in bits per second) and for calculating the UBRR value for each mode of operation using an internally generated clock source.

Table 14-1. Equations for Calculating Baud Rate Register Setting

Operating Mode	Equation for Calculating Baud Rate[1]	Equation for Calculating UBRR Value
Asynchronous Normal mode (U2X = 0)	$BAUD = \frac{f_{OSC}}{16(UBRR+1)}$	$UBRR = \frac{f_{OSC}}{16BAUD} - 1$
Asynchronous Double Speed mode (U2X = 1)	$BAUD = \frac{f_{OSC}}{8(UBRR+1)}$	$UBRR = \frac{f_{OSC}}{8BAUD} - 1$
Synchronous Master mode	$BAUD = \frac{f_{OSC}}{2(UBRR+1)}$	$UBRR = \frac{f_{OSC}}{2BAUD} - 1$

Note: 1. The baud rate is defined to be the transfer rate in bit per second (bps)

BAUD Baud rate (in bits per second, bps)

f_{osc} System Oscillator clock frequency

UBRR Contents of the UBRRH and UBRRL Registers, (0-4095)

Some examples of UBRR values for some system clock frequencies are found in Table 14-9 (see page 142).

14.3.2 Double Speed Operation (U2X)

The transfer rate can be doubled by setting the U2X bit in UCSRA. Setting this bit only has effect for the asynchronous operation. Set this bit to zero when using synchronous operation.

Setting this bit will reduce the divisor of the baud rate divider from 16 to 8, effectively doubling the transfer rate for asynchronous communication. Note however that the Receiver will in this case only use half the number of samples (reduced from 16 to 8) for data sampling and clock recovery, and therefore a more accurate baud rate setting and system clock are required when this mode is used. For the Transmitter, there are no downsides.

14.3.3 External Clock

External clocking is used by the synchronous slave modes of operation. The description in this section refers to Figure 14-2 for details.

External clock input from the XCK pin is sampled by a synchronization register to minimize the chance of meta-stability. The output from the synchronization register must then pass through an edge detector before it can be used by the Transmitter and Receiver. This process introduces a two CPU clock period delay and therefore the maximum external XCK clock frequency is limited by the following equation:

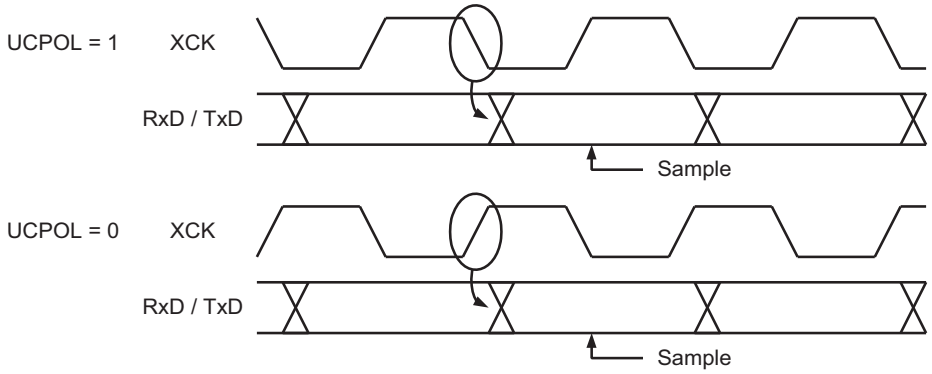
$$f_{XCK} < \frac{f_{OSC}}{4}$$

Note that f_{osc} depends on the stability of the system clock source. It is therefore recommended to add some margin to avoid possible loss of data due to frequency variations.

14.3.4 Synchronous Clock Operation

When synchronous mode is used (UMSEL = 1), the XCK pin will be used as either clock input (Slave) or clock output (Master). The dependency between the clock edges and data sampling or data change is the same. The basic principle is that data input (on RxD) is sampled at the opposite XCK clock edge of the edge the data output (TxD) is changed.

Figure 14-3. Synchronous Mode XCK Timing.



The UCPOL bit UCRSC selects which XCK clock edge is used for data sampling and which is used for data change. As Figure 14-3 shows, when UCPOL is zero the data will be changed at rising XCK edge and sampled at falling XCK edge. If UCPOL is set, the data will be changed at falling XCK edge and sampled at rising XCK edge.

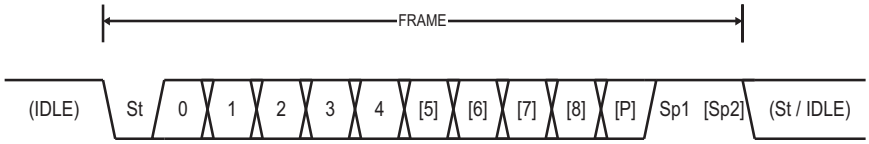
14.4 Frame Formats

A serial frame is defined to be one character of data bits with synchronization bits (start and stop bits), and optionally a parity bit for error checking. The USART accepts all 30 combinations of the following as valid frame formats:

- 1 start bit
- 5, 6, 7, 8, or 9 data bits
- no, even or odd parity bit
- 1 or 2 stop bits

A frame starts with the start bit followed by the least significant data bit. Then the next data bits, up to a total of nine, are succeeding, ending with the most significant bit. If enabled, the parity bit is inserted after the data bits, before the stop bits. When a complete frame is transmitted, it can be directly followed by a new frame, or the communication line can be set to an idle (high) state. Figure 14-4 illustrates the possible combinations of the frame formats. Bits inside brackets are optional.

Figure 14-4. Frame Formats



St	Start bit, always low.
(n)	Data bits (0 to 8).
P	Parity bit. Can be odd or even.
Sp	Stop bit, always high.
IDLE	No transfers on the communication line (RxD or TxD). An IDLE line must be high.

The frame format used by the USART is set by the UCSZ2:0, UPM1:0 and USBS bits in UCSRB and UCSRC. The Receiver and Transmitter use the same setting. Note that changing the setting of any of these bits will corrupt all ongoing communication for both the Receiver and Transmitter.

The USART Character SiZe (UCSZ2:0) bits select the number of data bits in the frame. The USART Parity mode (UPM1:0) bits enable and set the type of parity bit. The selection between one or two stop bits is done by the USART Stop Bit Select (USBS) bit. The Receiver ignores the second stop bit. An FE (Frame Error) will therefore only be detected in the cases where the first stop bit is zero.

14.4.1 Parity Bit Calculation

The parity bit is calculated by doing an exclusive-or of all the data bits. If odd parity is used, the result of the exclusive or is inverted. The relation between the parity bit and data bits is as follows:

$$P_{even} = d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 0$$
$$P_{odd} = d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 1$$

$\mathbf{P_{even}}$	Parity bit using even parity
$\mathbf{P_{odd}}$	Parity bit using odd parity
$\mathbf{d_n}$	Data bit n of the character

If used, the parity bit is located between the last data bit and first stop bit of a serial frame.

14.5 USART Initialization

The USART has to be initialized before any communication can take place. The initialization process normally consists of setting the baud rate, setting frame format and enabling the Transmitter or the Receiver depending on the usage. For interrupt driven USART operation, the Global Interrupt Flag should be cleared (and interrupts globally disabled) when doing the initialization.

Before doing a re-initialization with changed baud rate or frame format, be sure that there are no ongoing transmissions during the period the registers are changed. The TXC flag can be used to check that the Transmitter has completed all transfers, and the RXC flag can be used to check that there are no unread data in the receive buffer. Note that the TXC flag must be cleared before each transmission (before UDR is written) if it is used for this purpose.

The following simple USART initialization code examples show one assembly and one C function that are equal in functionality. The examples assume asynchronous operation using polling (no interrupts enabled) and a fixed frame format. The baud rate is given as a function parameter. For the assembly code, the baud rate parameter is assumed to be stored in the r17:r16 Registers.

Assembly Code Example[1]

```
USART_Init:
  ; Set baud rate
  out   UBRRH, r17
  out   UBRRL, r16
  ; Enable receiver and transmitter
  ldi   r16, (1<<RXEN)|(1<<TXEN)
  out   UCSRB,r16
  ; Set frame format: 8data, 2stop bit
  ldi   r16, (1<<USBS)|(3<<UCSZ0)
  out   UCSRC,r16
  ret
```

C Code Example[1]

```
void USART_Init( unsigned int baud )
{
  /* Set baud rate */
  UBRRH = (unsigned char)(baud>>8);
  UBRRL = (unsigned char)baud;
  /* Enable receiver and transmitter */
  UCSRB = (1<<RXEN)|(1<<TXEN);
  /* Set frame format: 8data, 2stop bit */
  UCSRC = (1<<USBS)|(3<<UCSZ0);
}
```

Note: 1. See “Code Examples” on page 7.

More advanced initialization routines can be made that include frame format as parameters, disable interrupts and so on. However, many applications use a fixed setting of the baud and control registers, and for these types of applications the initialization code can be placed directly in the main routine, or be combined with initialization code for other I/O modules.

14.6 Data Transmission – The USART Transmitter

The USART Transmitter is enabled by setting the *Transmit Enable* (TXEN) bit in the UCSRB Register. When the Transmitter is enabled, the normal port operation of the TxD pin is overridden by the USART and given the function as the Transmitter’s serial output. The baud rate, mode of operation and frame format must be set up once before doing any transmissions. If synchronous operation is used, the clock on the XCK pin will be overridden and used as transmission clock.

14.6.1 Sending Frames with 5 to 8 Data Bit

A data transmission is initiated by loading the transmit buffer with the data to be transmitted. The CPU can load the transmit buffer by writing to the UDR I/O location. The buffered data in the transmit buffer will be moved to the Shift Register when the Shift Register is ready to send a new frame. The Shift Register is loaded with new data if it is in idle state (no ongoing transmission) or immediately after the last stop bit of the previous frame is transmitted. When the Shift Register is loaded with new data, it will transfer one complete frame at the rate given by the Baud Register, U2X bit or by XCK depending on mode of operation.

The following code examples show a simple USART transmit function based on polling of the *Data Register Empty* (UDRE) flag. When using frames with less than eight bits, the most significant bits written to the UDR are ignored. The USART has to be initialized before the function can be used. For the assembly code, the data to be sent is assumed to be stored in Register R16

Assembly Code Example[1]

```
USART_Transmit:
  ; Wait for empty transmit buffer
  sbis UCSRA,UDRE
  rjmp USART_Transmit
  ; Put data (r16) into buffer, sends the data
  out  UDR,r16
  ret
```

C Code Example[1]

```
void USART_Transmit( unsigned char data )
{
  /* Wait for empty transmit buffer */
  while ( !( UCSRA & (1<<UDRE)) )
       ;
  /* Put data into buffer, sends the data */
  UDR = data;
}
```

Note: 1. See “Code Examples” on page 7.

The function simply waits for the transmit buffer to be empty by checking the UDRE flag, before loading it with new data to be transmitted. If the Data Register Empty interrupt is utilized, the interrupt routine writes the data into the buffer.

14.6.2 Sending Frames with 9 Data Bit

If 9-bit characters are used (UCSZ = 7), the ninth bit must be written to the TXB8 bit in UCSRB before the low byte of the character is written to UDR. The following code examples show a transmit function that handles 9-bit characters. For the assembly code, the data to be sent is assumed to be stored in registers R17:R16.

Assembly Code Example[1]

```
USART_Transmit:
  ; Wait for empty transmit buffer
  sbis UCSRA,UDRE
  rjmp USART_Transmit
  ; Copy 9th bit from r17 to TXB8
  cbi  UCSRB,TXB8
  sbrc r17,0
  sbi  UCSRB,TXB8
  ; Put LSB data (r16) into buffer, sends the data
  out  UDR,r16
  ret
```

C Code Example[1]

```
void USART_Transmit( unsigned int data )
{
  /* Wait for empty transmit buffer */
  while ( !( UCSRA & (1<<UDRE)) )
       ;
  /* Copy 9th bit to TXB8 */
  UCSRB &= ~(1<<TXB8);
  if ( data & 0x0100 )
    UCSRB |= (1<<TXB8);
  /* Put data into buffer, sends the data */
  UDR = data;
}
```

Note: 1. See “Code Examples” on page 7.

2. These transmit functions are written to be general functions. They can be optimized if the contents of the UCSRB is static. For example, only the TXB8 bit of the UCSRB Register is used after initialization.

The ninth bit can be used for indicating an address frame when using multi processor communication mode or for other protocol handling as for example synchronization.

14.6.3 Transmitter Flags and Interrupts

The USART Transmitter has two flags that indicate its state: USART Data Register Empty (UDRE) and Transmit Complete (TXC). Both flags can be used for generating interrupts.

The Data Register Empty (UDRE) flag indicates whether the transmit buffer is ready to receive new data. This bit is set when the transmit buffer is empty, and cleared when the transmit buffer

contains data to be transmitted that has not yet been moved into the Shift Register. For compatibility with future devices, always write this bit to zero when writing the UCSRA Register.

When the Data Register Empty Interrupt Enable (UDRIE) bit in UCSRB is written to one, the USART Data Register Empty Interrupt will be executed as long as UDRE is set (provided that global interrupts are enabled). UDRE is cleared by writing UDR. When interrupt-driven data transmission is used, the Data Register Empty interrupt routine must either write new data to UDR in order to clear UDRE or disable the Data Register Empty interrupt, otherwise a new interrupt will occur once the interrupt routine terminates.

The Transmit Complete (TXC) flag bit is set one when the entire frame in the Transmit Shift Register has been shifted out and there are no new data currently present in the transmit buffer. The TXC flag bit is automatically cleared when a transmit complete interrupt is executed, or it can be cleared by writing a one to its bit location. The TXC flag is useful in half-duplex communication interfaces (like the RS-485 standard), where a transmitting application must enter receive mode and free the communication bus immediately after completing the transmission.

When the Transmit Compete Interrupt Enable (TXCIE) bit in UCSRB is set, the USART Transmit Complete Interrupt will be executed when the TXC flag becomes set (provided that global interrupts are enabled). When the transmit complete interrupt is used, the interrupt handling routine does not have to clear the TXC flag, this is done automatically when the interrupt is executed.

14.6.4 Parity Generator

The Parity Generator calculates the parity bit for the serial frame data. When parity bit is enabled (UPM1 = 1), the transmitter control logic inserts the parity bit between the last data bit and the first stop bit of the frame that is sent.

14.6.5 Disabling the Transmitter

The disabling of the Transmitter (setting the TXEN to zero) will not become effective until ongoing and pending transmissions are completed, i.e., when the Transmit Shift Register and Transmit Buffer Register do not contain data to be transmitted. When disabled, the Transmitter will no longer override the TxD pin.

14.7 Data Reception – The USART Receiver

The USART Receiver is enabled by writing the Receive Enable (RXEN) bit in the UCSRB Register to one. When the Receiver is enabled, the normal pin operation of the RxD pin is overridden by the USART and given the function as the Receiver's serial input. The baud rate, mode of operation and frame format must be set up once before any serial reception can be done. If synchronous operation is used, the clock on the XCK pin will be used as transfer clock.

14.7.1 Receiving Frames with 5 to 8 Data Bits

The Receiver starts data reception when it detects a valid start bit. Each bit that follows the start bit will be sampled at the baud rate or XCK clock, and shifted into the Receive Shift Register until the first stop bit of a frame is received. A second stop bit will be ignored by the Receiver. When the first stop bit is received, i.e., a complete serial frame is present in the Receive Shift Register, the contents of the Shift Register will be moved into the receive buffer. The receive buffer can then be read by reading the UDR I/O location.

The following code example shows a simple USART receive function based on polling of the Receive Complete (RXC) flag. When using frames with less than eight bits the most significant bits of the data read from the UDR will be masked to zero. The USART has to be initialized before the function can be used.

Assembly Code Example[1]

```
USART_Receive:
  ; Wait for data to be received
  sbis UCSRA, RXC
  rjmp USART_Receive
  ; Get and return received data from buffer
  in   r16, UDR
  ret
```

C Code Example[1]

```
unsigned char USART_Receive( void )
{
  /* Wait for data to be received */
  while ( !(UCSRA & (1<<RXC)) )
       ;
  /* Get and return received data from buffer */
  return UDR;
}
```

Note: 1. See "Code Examples" on page 7.

The function simply waits for data to be present in the receive buffer by checking the RXC flag, before reading the buffer and returning the value.

14.7.2 Receiving Frames with 9 Data Bits

If 9-bit characters are used (UCSZ=7) the ninth bit must be read from the RXB8 bit in UCSRB **before** reading the low bits from the UDR. This rule applies to the FE, DOR and UPE Status Flags as well. Read status from UCSRA, then data from UDR. Reading the UDR I/O location will

change the state of the receive buffer FIFO and consequently the TXB8, FE, DOR and UPE bits, which all are stored in the FIFO, will change.

The following code example shows a simple USART receive function that handles both nine bit characters and the status bits.

Assembly Code Example[1]

```
USART_Receive:
  ; Wait for data to be received
  sbis UCSRA, RXC
  rjmp USART_Receive
  ; Get status and 9th bit, then data from buffer
  in   r18, UCSRA
  in   r17, UCSRB
  in   r16, UDR
  ; If error, return -1
  andi r18,(1<<FE)|(1<<DOR)|(1<<UPE)
  breq USART_ReceiveNoError
  ldi  r17, HIGH(-1)
  ldi  r16, LOW(-1)
USART_ReceiveNoError:
  ; Filter the 9th bit, then return
  lsr  r17
  andi r17, 0x01
  ret
```

C Code Example[1]

```
unsigned int USART_Receive( void )
{
  unsigned char status, resh, resl;
  /* Wait for data to be received */
  while ( !(UCSRA & (1<<RXC)) )
       ;
  /* Get status and 9th bit, then data */
  /* from buffer */
  status = UCSRA;
  resh = UCSRB;
  resl = UDR;
  /* If error, return -1 */
  if ( status & (1<<FE)|(1<<DOR)|(1<<UPE) )
    return -1;
  /* Filter the 9th bit, then return */
  resh = (resh >> 1) & 0x01;
  return ((resh << 8) | resl);
}
```

Note: 1. See “Code Examples” on page 7.

The receive function example reads all the I/O Registers into the Register File before any computation is done. This gives an optimal receive buffer utilization since the buffer location read will be free to accept new data as early as possible.

14.7.3 Receive Compete Flag and Interrupt

The USART Receiver has one flag that indicates the Receiver state.

The Receive Complete (RXC) flag indicates if there are unread data present in the receive buffer. This flag is one when unread data exist in the receive buffer, and zero when the receive buffer is empty (i.e., does not contain any unread data). If the Receiver is disabled (RXEN = 0), the receive buffer will be flushed and consequently the RXC bit will become zero.

When the Receive Complete Interrupt Enable (RXCIE) in UCSRB is set, the USART Receive Complete interrupt will be executed as long as the RXC flag is set (provided that global interrupts are enabled). When interrupt-driven data reception is used, the receive complete routine must read the received data from UDR in order to clear the RXC flag, otherwise a new interrupt will occur once the interrupt routine terminates.

14.7.4 Receiver Error Flags

The USART Receiver has three error flags: Frame Error (FE), Data OverRun (DOR) and Parity Error (UPE). All can be accessed by reading UCSRA. Common for the error flags is that they are located in the receive buffer together with the frame for which they indicate the error status. Due to the buffering of the error flags, the UCSRA must be read before the receive buffer (UDR), since reading the UDR I/O location changes the buffer read location. Another equality for the error flags is that they can not be altered by software doing a write to the flag location. However, all flags must be set to zero when the UCSRA is written for upward compatibility of future USART implementations. None of the error flags can generate interrupts.

The Frame Error (FE) flag indicates the state of the first stop bit of the next readable frame stored in the receive buffer. The FE flag is zero when the stop bit was correctly read (as one), and the FE flag will be one when the stop bit was incorrect (zero). This flag can be used for detecting out-of-sync conditions, detecting break conditions and protocol handling. The FE flag is not affected by the setting of the USBS bit in UCSRC since the Receiver ignores all, except for the first, stop bits. For compatibility with future devices, always set this bit to zero when writing to UCSRA.

The Data OverRun (DOR) flag indicates data loss due to a receiver buffer full condition. A Data OverRun occurs when the receive buffer is full (two characters), it is a new character waiting in the Receive Shift Register, and a new start bit is detected. If the DOR flag is set there was one or more serial frame lost between the frame last read from UDR, and the next frame read from UDR. For compatibility with future devices, always write this bit to zero when writing to UCSRA. The DOR flag is cleared when the frame received was successfully moved from the Shift Register to the receive buffer.

The Parity Error (UPE) Flag indicates that the next frame in the receive buffer had a Parity Error when received. If Parity Check is not enabled the UPE bit will always be read zero. For compatibility with future devices, always set this bit to zero when writing to UCSRA. For more details see “Parity Bit Calculation” on page 125 and “Parity Checker” on page 132.

14.7.5 Parity Checker

The Parity Checker is active when the high USART Parity mode (UPM1) bit is set. Type of Parity Check to be performed (odd or even) is selected by the UPM0 bit. When enabled, the Parity

Checker calculates the parity of the data bits in incoming frames and compares the result with the parity bit from the serial frame. The result of the check is stored in the receive buffer together with the received data and stop bits. The Parity Error (UPE) flag can then be read by software to check if the frame had a Parity Error.

The UPE bit is set if the next character that can be read from the receive buffer had a Parity Error when received and the Parity Checking was enabled at that point (UPM1 = 1). This bit is valid until the receive buffer (UDR) is read.

14.7.6 Disabling the Receiver

In contrast to the Transmitter, disabling of the Receiver will be immediate. Data from ongoing receptions will therefore be lost. When disabled (i.e., the RXEN is set to zero) the Receiver will no longer override the normal function of the RxD port pin. The Receiver buffer FIFO will be flushed when the Receiver is disabled. Remaining data in the buffer will be lost

14.7.7 Flushing the Receive Buffer

The receiver buffer FIFO will be flushed when the Receiver is disabled, i.e., the buffer will be emptied of its contents. Unread data will be lost. If the buffer has to be flushed during normal operation, due to for instance an error condition, read the UDR I/O location until the RXC flag is cleared. The following code example shows how to flush the receive buffer.

Assembly Code Example[1]

```
USART_Flush:
  sbis UCSRA, RXC
  ret
  in   r16, UDR
  rjmp USART_Flush
```

C Code Example[1]

```
void USART_Flush( void )
{
  unsigned char dummy;
  while ( UCSRA & (1<<RXC) ) dummy = UDR;
}
```

Note: 1. See “Code Examples” on page 7.

14.8 Asynchronous Data Reception

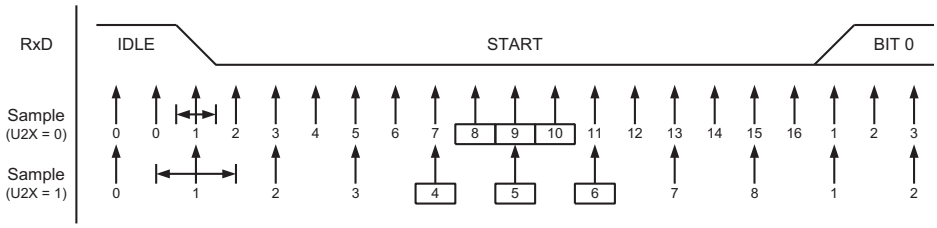
The USART includes a clock recovery and a data recovery unit for handling asynchronous data reception. The clock recovery logic is used for synchronizing the internally generated baud rate clock to the incoming asynchronous serial frames at the RxD pin. The data recovery logic samples and low pass filters each incoming bit, thereby improving the noise immunity of the Receiver. The asynchronous reception operational range depends on the accuracy of the internal baud rate clock, the rate of the incoming frames, and the frame size in number of bits.

14.8.1 Asynchronous Clock Recovery

The clock recovery logic synchronizes internal clock to the incoming serial frames. Figure 14-5 illustrates the sampling process of the start bit of an incoming frame. The sample rate is 16 times

the baud rate for Normal mode, and eight times the baud rate for Double Speed mode. The horizontal arrows illustrate the synchronization variation due to the sampling process. Note the larger time variation when using the Double Speed mode (U2X = 1) of operation. Samples denoted zero are samples done when the RxD line is idle (i.e., no communication activity).

Figure 14-5. Start Bit Sampling

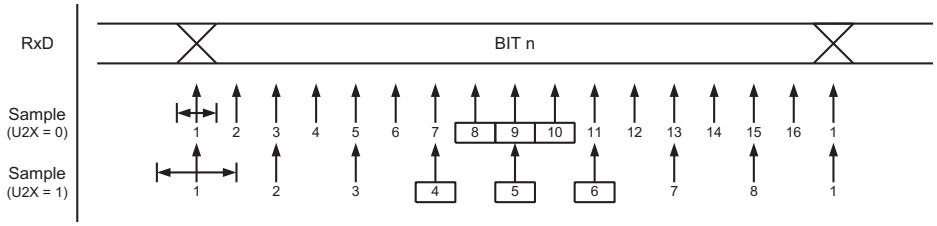


When the clock recovery logic detects a high (idle) to low (start) transition on the RxD line, the start bit detection sequence is initiated. Let sample 1 denote the first zero-sample as shown in the figure. The clock recovery logic then uses samples 8, 9, and 10 for Normal mode, and samples 4, 5, and 6 for Double Speed mode (indicated with sample numbers inside boxes on the figure), to decide if a valid start bit is received. If two or more of these three samples have logical high levels (the majority wins), the start bit is rejected as a noise spike and the Receiver starts looking for the next high to low-transition. If however, a valid start bit is detected, the clock recovery logic is synchronized and the data recovery can begin. The synchronization process is repeated for each start bit.

14.8.2 Asynchronous Data Recovery

When the receiver clock is synchronized to the start bit, the data recovery can begin. The data recovery unit uses a state machine that has 16 states for each bit in Normal mode and eight states for each bit in Double Speed mode. Figure 14-6 shows the sampling of the data bits and the parity bit. Each of the samples is given a number that is equal to the state of the recovery unit.

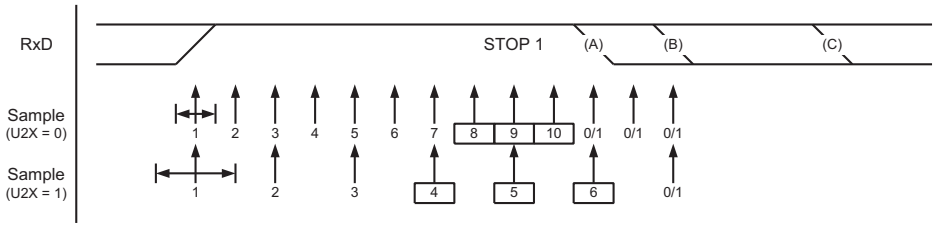
Figure 14-6. Sampling of Data and Parity Bit



The decision of the logic level of the received bit is taken by doing a majority voting of the logic value to the three samples in the center of the received bit. The center samples are emphasized on the figure by having the sample number inside boxes. The majority voting process is done as follows: If two or all three samples have high levels, the received bit is registered to be a logic 1. If two or all three samples have low levels, the received bit is registered to be a logic 0. This majority voting process acts as a low pass filter for the incoming signal on the RxD pin. The recovery process is then repeated until a complete frame is received. Including the first stop bit. Note that the Receiver only uses the first stop bit of a frame.

Figure 14-7 shows the sampling of the stop bit and the earliest possible beginning of the start bit of the next frame.

Figure 14-7. Stop Bit Sampling and Next Start Bit Sampling



The same majority voting is done to the stop bit as done for the other bits in the frame. If the stop bit is registered to have a logic 0 value, the Frame Error (FE) flag will be set.

A new high to low transition indicating the start bit of a new frame can come right after the last of the bits used for majority voting. For Normal Speed mode, the first low level sample can be at point marked (A) in Figure 14-7. For Double Speed mode the first low level must be delayed to (B). (C) marks a stop bit of full length. The early start bit detection influences the operational range of the Receiver.

14.8.3 Asynchronous Operational Range

The operational range of the Receiver is dependent on the mismatch between the received bit rate and the internally generated baud rate. If the Transmitter is sending frames at too fast or too slow bit rates, or the internally generated baud rate of the Receiver does not have a similar (see Table 14-2) base frequency, the Receiver will not be able to synchronize the frames to the start bit.

The following equations can be used to calculate the ratio of the incoming data rate and internal receiver baud rate.

$$R_{slow} = \frac{(D+1)S}{S-1+D\cdot S+S_F} \qquad R_{fast} = \frac{(D+2)S}{(D+1)S+S_M}$$

D Sum of character size and parity size (D = 5 to 10 bit)

S Samples per bit. S = 16 for Normal Speed mode and S = 8 for Double Speed mode.

S_F First sample number used for majority voting. S_F = 8 for normal speed and S_F = 4 for Double Speed mode.

S_M Middle sample number used for majority voting. S_M = 9 for normal speed and S_M = 5 for Double Speed mode.

R_{slow} is the ratio of the slowest incoming data rate that can be accepted in relation to the receiver baud rate. R_{fast} is the ratio of the fastest incoming data rate that can be accepted in relation to the receiver baud rate.

Table 14-2 and Table 14-3 list the maximum receiver baud rate error that can be tolerated. Note that Normal Speed mode has higher toleration of baud rate variations.

Table 14-2. Recommended Maximum Receiver Baud Rate Error for Normal Speed Mode (U2X = 0)

D # (Data+Parity Bit)	R_{slow} (%)	R_{fast} (%)	Max Total Error (%)	Recommended Max Receiver Error (%)
5	93.20	106.67	+6.67/-6.8	± 3.0
6	94.12	105.79	+5.79/-5.88	± 2.5
7	94.81	105.11	+5.11/-5.19	± 2.0
8	95.36	104.58	+4.58/-4.54	± 2.0
9	95.81	104.14	+4.14/-4.19	± 1.5
10	96.17	103.78	+3.78/-3.83	± 1.5

Table 14-3. Recommended Maximum Receiver Baud Rate Error for Double Speed Mode (U2X = 1)

D # (Data+Parity Bit)	R_{slow} (%)	R_{fast} (%)	Max Total Error (%)	Recommended Max Receiver Error (%)
5	94.12	105.66	+5.66/-5.88	± 2.5
6	94.92	104.92	+4.92/-5.08	± 2.0
7	95.52	104,35	+4.35/-4.48	± 1.5
8	96.00	103.90	+3.90/-4.00	± 1.5
9	96.39	103.53	+3.53/-3.61	± 1.5
10	96.70	103.23	+3.23/-3.30	± 1.0

The recommendations of the maximum receiver baud rate error was made under the assumption that the Receiver and Transmitter equally divides the maximum total error.

There are two possible sources for the receivers baud rate error. The Receiver's system clock (XTAL) will always have some minor instability over the supply voltage range and the temperature range. When using a crystal to generate the system clock, this is rarely a problem, but for a resonator the system clock may differ more than 2% depending of the resonators tolerance. The second source for the error is more controllable. The baud rate generator can not always do an exact division of the system frequency to get the baud rate wanted. In this case an UBRR value that gives an acceptable low error can be used if possible.

14.9 Multi-processor Communication Mode

Setting the Multi-processor Communication mode (MPCM) bit in UCSRA enables a filtering function of incoming frames received by the USART Receiver. Frames that do not contain address information will be ignored and not put into the receive buffer. This effectively reduces the number of incoming frames that has to be handled by the CPU, in a system with multiple MCUs that communicate via the same serial bus. The Transmitter is unaffected by the MPCM setting, but has to be used differently when it is a part of a system utilizing the Multi-processor Communication mode.

If the Receiver is set up to receive frames that contain 5 to 8 data bits, then the first stop bit indicates if the frame contains data or address information. If the Receiver is set up for frames with nine data bits, then the ninth bit (RXB8) is used for identifying address and data frames. When

the frame type bit (the first stop or the ninth bit) is one, the frame contains an address. When the frame type bit is zero the frame is a data frame.

The Multi-processor Communication mode enables several slave MCUs to receive data from a master MCU. This is done by first decoding an address frame to find out which MCU has been addressed. If a particular slave MCU has been addressed, it will receive the following data frames as normal, while the other slave MCUs will ignore the received frames until another address frame is received.

14.9.1 Using MPCM

For an MCU to act as a master MCU, it can use a 9-bit character frame format (UCSZ = 7). The ninth bit (TXB8) must be set when an address frame (TXB8 = 1) or cleared when a data frame (TXB = 0) is being transmitted. The slave MCUs must in this case be set to use a 9-bit character frame format.

The following procedure should be used to exchange data in Multi-processor Communication mode:

1. All Slave MCUs are in Multi-processor Communication mode (MPCM in UCSRA is set).
2. The Master MCU sends an address frame, and all slaves receive and read this frame. In the Slave MCUs, the RXC flag in UCSRA will be set as normal.
3. Each Slave MCU reads the UDR Register and determines if it has been selected. If so, it clears the MPCM bit in UCSRA, otherwise it waits for the next address byte and keeps the MPCM setting.
4. The addressed MCU will receive all data frames until a new address frame is received. The other Slave MCUs, which still have the MPCM bit set, will ignore the data frames.
5. When the last data frame is received by the addressed MCU, the addressed MCU sets the MPCM bit and waits for a new address frame from master. The process then repeats from 2.

Using any of the 5- to 8-bit character frame formats is possible, but impractical since the Receiver must change between using n and n+1 character frame formats. This makes full-duplex operation difficult since the Transmitter and Receiver uses the same character size setting. If 5- to 8-bit character frames are used, the Transmitter must be set to use two stop bit (USBS = 1) since the first stop bit is used for indicating the frame type.

Do not use Read-Modify-Write instructions (SBI and CBI) to set or clear the MPCM bit. The MPCM bit shares the same I/O location as the TXC flag and this might accidentally be cleared when using SBI or CBI instructions.

14.10 Register Description

14.10.1 UDR – USART I/O Data Register

Bit	7	6	5	4	3	2	1	0	
0x0C (0x2C)	RXB[7:0]								UDR (Read)
0x0C (0x2C)	TXB[7:0]								UDR (Write)
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 USART Transmit Data Buffer Register and USART Receive Data Buffer Registers share the same I/O address referred to as USART Data Register or UDR. The Transmit Data Buffer Register (TXB) will be the destination for data written to the UDR Register location. Reading the UDR Register location will return the contents of the Receive Data Buffer Register (RXB).

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 UDRE flag in the UCSRA Register is set. Data written to UDR when the UDRE 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. Then the data will be serially transmitted on the TxD pin.

The receive buffer consists of a two level FIFO. The FIFO will change its state whenever the receive buffer is accessed. Due to this behavior of the receive buffer, do not use Read-Modify-Write instructions (SBI and CBI) on this location. Be careful when using bit test instructions (SBIC and SBIS), since these also will change the state of the FIFO.

14.10.2 UCSRA – USART Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
0x0B (0x2B)	**RXC**	**TXC**	**UDRE**	**FE**	**DOR**	**UPE**	**U2X**	**MPCM**	**UCSRA**
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

- **Bit 7 – RXC: USART Receive Complete**

This flag bit is set when there are unread data in the receive buffer and cleared when the receive buffer is empty (i.e., does not contain any unread data). If the Receiver is disabled, the receive buffer will be flushed and consequently the RXC bit will become zero. The RXC flag can be used to generate a Receive Complete interrupt (see description of the RXCIE bit).

- **Bit 6 – TXC: USART Transmit Complete**

This flag bit is set when the entire frame in the Transmit Shift Register has been shifted out and there are no new data currently present in the transmit buffer (UDR). The TXC flag bit is automatically cleared when a transmit complete interrupt is executed, or it can be cleared by writing a one to its bit location. The TXC flag can generate a Transmit Complete interrupt (see description of the TXCIE bit).

- **Bit 5 – UDRE: USART Data Register Empty**

The UDRE flag indicates if the transmit buffer (UDR) is ready to receive new data. If UDRE is one, the buffer is empty, and therefore ready to be written. The UDRE flag can generate a Data Register Empty interrupt (see description of the UDRIE bit).

UDRE is set after a reset to indicate that the Transmitter is ready.

- **Bit 4 – FE: Frame Error**

This bit is set if the next character in the receive buffer had a Frame Error when received. I.e., when the first stop bit of the next character in the receive buffer is zero. This bit is valid until the receive buffer (UDR) is read. The FE bit is zero when the stop bit of received data is one. Always set this bit to zero when writing to UCSRA.

- **Bit 3 – DOR: Data OverRun**

This bit is set if a Data OverRun condition is detected. A Data OverRun occurs when the receive buffer is full (two characters), it is a new character waiting in the Receive Shift Register, and a new start bit is detected. This bit is valid until the receive buffer (UDR) is read. Always set this bit to zero when writing to UCSRA.

- **Bit 2 – UPE: USART Parity Error**

This bit is set if the next character in the receive buffer had a Parity Error when received and the Parity Checking was enabled at that point (UPM1 = 1). This bit is valid until the receive buffer (UDR) is read. Always set this bit to zero when writing to UCSRA.

- **Bit 1 – U2X: Double the USART Transmission Speed**

This bit only has effect for the asynchronous operation. Write this bit to zero when using synchronous operation.

Writing this bit to one will reduce the divisor of the baud rate divider from 16 to 8 effectively doubling the transfer rate for asynchronous communication.

- **Bit 0 – MPCM: Multi-processor Communication Mode**

This bit enables the Multi-processor Communication mode. When the MPCM bit is written to one, all the incoming frames received by the USART Receiver that do not contain address information will be ignored. The Transmitter is unaffected by the MPCM setting. For more detailed information see “Multi-processor Communication Mode” on page 136.

14.10.3 UCSRB – USART Control and Status Register B

Bit	7	6	5	4	3	2	1	0	
0x0A (0x2A)	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	UCSRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – RXCIE: RX Complete Interrupt Enable**

Writing this bit to one enables interrupt on the RXC flag. A USART Receive Complete interrupt will be generated only if the RXCIE bit is written to one, the Global Interrupt Flag in SREG is written to one and the RXC bit in UCSRA is set.

- **Bit 6 – TXCIE: TX Complete Interrupt Enable**

Writing this bit to one enables interrupt on the TXC flag. A USART Transmit Complete interrupt will be generated only if the TXCIE bit is written to one, the Global Interrupt Flag in SREG is written to one and the TXC bit in UCSRA is set.

- **Bit 5 – UDRIE: USART Data Register Empty Interrupt Enable**

Writing this bit to one enables interrupt on the UDRE flag. A Data Register Empty interrupt will be generated only if the UDRIE bit is written to one, the Global Interrupt Flag in SREG is written to one and the UDRE bit in UCSRA is set.

- **Bit 4 – RXEN: Receiver Enable**

Writing this bit to one enables the USART Receiver. The Receiver will override normal port operation for the RxD pin when enabled. Disabling the Receiver will flush the receive buffer invalidating the FE, DOR, and UPE Flags.

- **Bit 3 – TXEN: Transmitter Enable**

Writing this bit to one enables the USART Transmitter. The Transmitter will override normal port operation for the TxD pin when enabled. The disabling of the Transmitter (writing TXEN to zero) will not become effective until ongoing and pending transmissions are completed, i.e., when the Transmit Shift Register and Transmit Buffer Register do not contain data to be transmitted. When disabled, the Transmitter will no longer override the TxD port.

- **Bit 2 – UCSZ2: Character Size**

The UCSZ2 bits combined with the UCSZ1:0 bit in UCSRC sets the number of data bits (Character SiZe) in a frame the Receiver and Transmitter use.

- **Bit 1 – RXB8: Receive Data Bit 8**

RXB8 is the ninth data bit of the received character when operating with serial frames with nine data bits. Must be read before reading the low bits from UDR.

- **Bit 0 – TXB8: Transmit Data Bit 8**

TXB8 is the ninth data bit in the character to be transmitted when operating with serial frames with nine data bits. Must be written before writing the low bits to UDR.

14.10.4 UCSRC – USART Control and Status Register C

Bit	7	6	5	4	3	2	1	0	
0x03 (0x23)	**UMSEL1**	**UMSEL0**	**UPM1**	**UPM0**	**USBS**	**UCSZ1**	**UCSZ0**	**UCPOL**	**UCSRC**
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	1	1	0	

- **Bits 7:6 – UMSEL1:0: USART Mode Select**

These bits select the mode of operation of the USART as shown in Table 14-4.

Table 14-4. UMSEL Bit Settings

UMSEL1	UMSEL0	Mode
0	0	Asynchronous USART
0	1	Synchronous USART
1	0	Reserved
1	1	Master SPI (MSPIM)(1)

Note: 1. For full description of the Master SPI Mode (MSPIM) Operation, see “USART in SPI Mode” on page 146.

- **Bits 5:4 – UPM1:0: Parity Mode**

These bits enable and set type of parity generation and check. If enabled, the Transmitter will automatically generate and send the parity of the transmitted data bits within each frame. The Receiver will generate a parity value for the incoming data and compare it to the UPM0 setting. If a mismatch is detected, the UPE Flag in UCSRA will be set.

Table 14-5. UPM Bits Settings

UPM1	UPM0	Parity Mode
0	0	Disabled
0	1	Reserved
1	0	Enabled, Even Parity
1	1	Enabled, Odd Parity

- **Bit 3 – USBS: Stop Bit Select**

This bit selects the number of stop bits to be inserted by the Transmitter. The Receiver ignores this setting.

Table 14-6. USBS Bit Settings

USBS	Stop Bit(s)
0	1-bit
1	2-bit

- **Bit 2:1 – UCSZ1:0: Character Size**

The UCSZ1:0 bits combined with the UCSZ2 bit in UCSRB sets the number of data bits (Character SiZe) in a frame the Receiver and Transmitter use. See Table 14-7.

Table 14-7. UCSZ Bits Settings

UCSZ2	UCSZ1	UCSZ0	Character Size
0	0	0	5-bit
0	0	1	6-bit
0	1	0	7-bit
0	1	1	8-bit
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	9-bit

- **Bit 0 – UCPOL: Clock Polarity**

This bit is used for synchronous mode only. Write this bit to zero when asynchronous mode is used. The UCPOL bit sets the relationship between data output change and data input sample, and the synchronous clock (XCK).

Table 14-8. UCPOL Bit Settings

UCPOL	Transmitted Data Changed (Output of TxD Pin)	Received Data Sampled (Input on RxD Pin)
0	Rising XCK Edge	Falling XCK Edge
1	Falling XCK Edge	Rising XCK Edge

14.10.5 UBRRL and UBRRH – USART Baud Rate Registers

Bit	15	14	13	12	11	10	9	8	
0x02 (0x22)	–	–	–	–	UBRR[11:8]				UBRRH
0x09 (0x29)	UBRR[7:0]								UBRRL
Bit	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
	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	
	0	0	0	0	0	0	0	0	

- **Bit 15:12 – Reserved Bits**

These bits are reserved for future use. For compatibility with future devices, these bit must be written to zero when UBRRH is written.

- **Bit 11:0 – UBRR11:0: USART Baud Rate Register**

This is a 12-bit register which contains the USART baud rate. The UBRRH contains the four most significant bits, and the UBRRL contains the eight least significant bits of the USART baud rate. Ongoing transmissions by the Transmitter and Receiver will be corrupted if the baud rate is changed. Writing UBRRL will trigger an immediate update of the baud rate prescaler.

14.11 Examples of Baud Rate Setting

For standard crystal and resonator frequencies, the most commonly used baud rates for asynchronous operation can be generated by using the UBRR settings, see the following tables from Table 14-9 to Table 14-12. UBRR values which yield an actual baud rate differing less than 0.5% from the target baud rate, are bold in the table. Higher error ratings are acceptable, but the Receiver will have less noise resistance when the error ratings are high, especially for large serial frames (see “Asynchronous Operational Range” on page 135). The error values are calculated using the following equation:

$$\text{Error}[\%] = \left(\frac{\text{BaudRate}_{\text{Closest Match}}}{\text{BaudRate}} - 1\right) \bullet 100\%$$

Table 14-9. Examples of UBRR Settings for Commonly Used Oscillator Frequencies

Baud Rate (bps)	f_{osc} = 1.0000 MHz				f_{osc} = 1.8432 MHz				f_{osc} = 2.0000 MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	25	0.2%	51	0.2%	47	0.0%	95	0.0%	51	0.2%	103	0.2%
4800	12	0.2%	25	0.2%	23	0.0%	47	0.0%	25	0.2%	51	0.2%
9600	6	-7.0%	12	0.2%	11	0.0%	23	0.0%	12	0.2%	25	0.2%
14.4k	3	8.5%	8	-3.5%	7	0.0%	15	0.0%	8	-3.5%	16	2.1%
19.2k	2	8.5%	6	-7.0%	5	0.0%	11	0.0%	6	-7.0%	12	0.2%
28.8k	1	8.5%	3	8.5%	3	0.0%	7	0.0%	3	8.5%	8	-3.5%
38.4k	1	-18.6%	2	8.5%	2	0.0%	5	0.0%	2	8.5%	6	-7.0%
57.6k	0	8.5%	1	8.5%	1	0.0%	3	0.0%	1	8.5%	3	8.5%
76.8k	–	–	1	-18.6%	1	-25.0%	2	0.0%	1	-18.6%	2	8.5%
115.2k	–	–	0	8.5%	0	0.0%	1	0.0%	0	8.5%	1	8.5%
230.4k	–	–	–	–	–	–	0	0.0%	–	–	–	–
250k	–	–	–	–	–	–	–	–	–	–	0	0.0%
Max. (1)	62.5 kbps		125 kbps		115.2 kbps		230.4 kbps		125 kbps		250 kbps	

1. UBRR = 0, Error = 0.0%

Table 14-10. Examples of UBRR Settings for Commonly Used Oscillator Frequencies (Continued)

Baud Rate (bps)	f_{osc} = 3.6864 MHz				f_{osc} = 4.0000 MHz				f_{osc} = 7.3728 MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	95	0.0%	191	0.0%	103	0.2%	207	0.2%	191	0.0%	383	0.0%
4800	47	0.0%	95	0.0%	51	0.2%	103	0.2%	95	0.0%	191	0.0%
9600	23	0.0%	47	0.0%	25	0.2%	51	0.2%	47	0.0%	95	0.0%
14.4k	15	0.0%	31	0.0%	16	2.1%	34	-0.8%	31	0.0%	63	0.0%
19.2k	11	0.0%	23	0.0%	12	0.2%	25	0.2%	23	0.0%	47	0.0%
28.8k	7	0.0%	15	0.0%	8	-3.5%	16	2.1%	15	0.0%	31	0.0%
38.4k	5	0.0%	11	0.0%	6	-7.0%	12	0.2%	11	0.0%	23	0.0%
57.6k	3	0.0%	7	0.0%	3	8.5%	8	-3.5%	7	0.0%	15	0.0%
76.8k	2	0.0%	5	0.0%	2	8.5%	6	-7.0%	5	0.0%	11	0.0%
115.2k	1	0.0%	3	0.0%	1	8.5%	3	8.5%	3	0.0%	7	0.0%
230.4k	0	0.0%	1	0.0%	0	8.5%	1	8.5%	1	0.0%	3	0.0%
250k	0	-7.8%	1	-7.8%	0	0.0%	1	0.0%	1	-7.8%	3	-7.8%
0.5M	–	–	0	-7.8%	–	–	0	0.0%	0	-7.8%	1	-7.8%
1M	–	–	–	–	–	–	–	–	–	–	0	-7.8%
Max. (1)	230.4 kbps		460.8 kbps		250 kbps		0.5 Mbps		460.8 kbps		921.6 kbps	

1. UBRR = 0, Error = 0.0%

Table 14-11. Examples of UBRR Settings for Commonly Used Oscillator Frequencies (Continued)

Baud Rate (bps)	f_{osc} = 8.0000 MHz				f_{osc} = 11.0592 MHz				f_{osc} = 14.7456 MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	207	0.2%	416	-0.1%	287	0.0%	575	0.0%	383	0.0%	767	0.0%
4800	103	0.2%	207	0.2%	143	0.0%	287	0.0%	191	0.0%	383	0.0%
9600	51	0.2%	103	0.2%	71	0.0%	143	0.0%	95	0.0%	191	0.0%
14.4k	34	-0.8%	68	0.6%	47	0.0%	95	0.0%	63	0.0%	127	0.0%
19.2k	25	0.2%	51	0.2%	35	0.0%	71	0.0%	47	0.0%	95	0.0%
28.8k	16	2.1%	34	-0.8%	23	0.0%	47	0.0%	31	0.0%	63	0.0%
38.4k	12	0.2%	25	0.2%	17	0.0%	35	0.0%	23	0.0%	47	0.0%
57.6k	8	-3.5%	16	2.1%	11	0.0%	23	0.0%	15	0.0%	31	0.0%
76.8k	6	-7.0%	12	0.2%	8	0.0%	17	0.0%	11	0.0%	23	0.0%
115.2k	3	8.5%	8	-3.5%	5	0.0%	11	0.0%	7	0.0%	15	0.0%
230.4k	1	8.5%	3	8.5%	2	0.0%	5	0.0%	3	0.0%	7	0.0%
250k	1	0.0%	3	0.0%	2	-7.8%	5	-7.8%	3	-7.8%	6	5.3%
0.5M	0	0.0%	1	0.0%	–	–	2	-7.8%	1	-7.8%	3	-7.8%
1M	–	–	0	0.0%	–	–	–	–	0	-7.8%	1	-7.8%
Max. (1)	0.5 Mbps		1 Mbps		691.2 kbps		1.3824 Mbps		921.6 kbps		1.8432 Mbps	

1. UBRR = 0, Error = 0.0%

Table 14-12. Examples of UBRR Settings for Commonly Used Oscillator Frequencies (Continued)

Baud Rate (bps)	f_{osc} = 16.0000 MHz			
	U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error
2400	416	-0.1%	832	0.0%
4800	207	0.2%	416	-0.1%
9600	103	0.2%	207	0.2%
14.4k	68	0.6%	138	-0.1%
19.2k	51	0.2%	103	0.2%
28.8k	34	-0.8%	68	0.6%
38.4k	25	0.2%	51	0.2%
57.6k	16	2.1%	34	-0.8%
76.8k	12	0.2%	25	0.2%
115.2k	8	-3.5%	16	2.1%
230.4k	3	8.5%	8	-3.5%
250k	3	0.0%	7	0.0%
0.5M	1	0.0%	3	0.0%
1M	0	0.0%	1	0.0%
Max. (1)	1 Mbps		2 Mbps	

1. UBRR = 0, Error = 0.0%

15. USART in SPI Mode

15.1 Features

- **Full Duplex, Three-wire Synchronous Data Transfer**
- **Master Operation**
- **Supports all four SPI Modes of Operation (Mode 0, 1, 2, and 3)**
- **LSB First or MSB First Data Transfer (Configurable Data Order)**
- **Queued Operation (Double Buffered)**
- **High Resolution Baud Rate Generator**
- **High Speed Operation ($f_{XCKmax} = f_{CK}/2$)**
- **Flexible Interrupt Generation**

15.2 Overview

The Universal Synchronous and Asynchronous serial Receiver and Transmitter (USART) can be set to a master SPI compliant mode of operation.

Setting both UMSEL1:0 bits to one enables the USART in MSPIM logic. In this mode of operation the SPI master control logic takes direct control over the USART resources. These resources include the transmitter and receiver shift register and buffers, and the baud rate generator. The parity generator and checker, the data and clock recovery logic, and the RX and TX control logic is disabled. The USART RX and TX control logic is replaced by a common SPI transfer control logic. However, the pin control logic and interrupt generation logic is identical in both modes of operation.

The I/O register locations are the same in both modes. However, some of the functionality of the control registers changes when using MSPIM.

15.3 Clock Generation

The Clock Generation logic generates the base clock for the Transmitter and Receiver. For USART MSPIM mode of operation only internal clock generation (i.e. master operation) is supported. The Data Direction Register for the XCK pin (DDR_XCK) must therefore be set to one (i.e. as output) for the USART in MSPIM to operate correctly. Preferably the DDR_XCK should be set up before the USART in MSPIM is enabled (i.e. TXEN and RXEN bit set to one).

The internal clock generation used in MSPIM mode is identical to the USART synchronous master mode. The baud rate or UBRR setting can therefore be calculated using the same equations, see Table 15-1:

Table 15-1. Equations for Calculating Baud Rate Register Setting

Operating Mode	Equation for Calculating Baud Rate[1]	Equation for Calculating UBRRn Value
Synchronous Master mode	$BAUD = \frac{f_{OSC}}{2(\boldsymbol{UBRR}+1)}$	$\boldsymbol{UBRR} = \frac{f_{OSC}}{2BAUD} - 1$

Note: 1. The baud rate is defined to be the transfer rate in bit per second (bps)

BAUD	Baud rate (in bits per second, bps)
f_{osc}	System Oscillator clock frequency
UBRR	Contents of the UBRRH and UBRRL Registers, (0-4095)

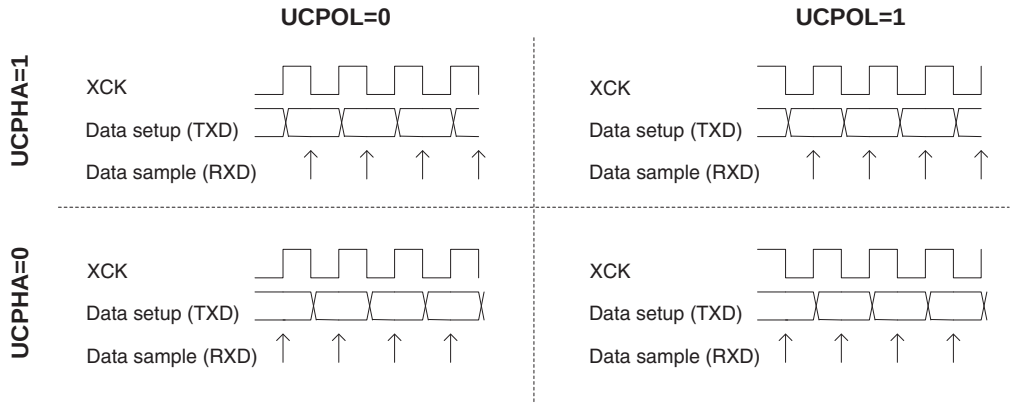
15.4 SPI Data Modes and Timing

There are four combinations of XCK (SCK) phase and polarity with respect to serial data, which are determined by control bits UCPHA and UCPOL. The data transfer timing diagrams are shown in Figure 15-1. Data bits are shifted out and latched in on opposite edges of the XCK signal, ensuring sufficient time for data signals to stabilize. The UCPOL and UCPHA functionality is summarized in Table 15-2. Note that changing the setting of any of these bits will corrupt all ongoing communication for both the Receiver and Transmitter.

Table 15-2. UCPOL and UCPHA Functionality-

UCPOL	UCPHA	SPI Mode	Leading Edge	Trailing Edge
0	0	0	Sample (Rising)	Setup (Falling)
0	1	1	Setup (Rising)	Sample (Falling)
1	0	2	Sample (Falling)	Setup (Rising)
1	1	3	Setup (Falling)	Sample (Rising)

Figure 15-1. UCPHA and UCPOL data transfer timing diagrams.



15.5 Frame Formats

A serial frame for the MSPIM is defined to be one character of 8 data bits. The USART in MSPIM mode has two valid frame formats:

- 8-bit data with MSB first
- 8-bit data with LSB first

A frame starts with the least or most significant data bit. Then the next data bits, up to a total of eight, are succeeding, ending with the most or least significant bit accordingly. When a complete frame is transmitted, a new frame can directly follow it, or the communication line can be set to an idle (high) state.

The UDORD bit in UCSRC sets the frame format used by the USART in MSPIM mode. The Receiver and Transmitter use the same setting. Note that changing the setting of any of these bits will corrupt all ongoing communication for both the Receiver and Transmitter.

16-bit data transfer can be achieved by writing two data bytes to UDR. A UART transmit complete interrupt will then signal that the 16-bit value has been shifted out.

15.5.1 USART MSPIM Initialization

The USART in MSPIM mode has to be initialized before any communication can take place. The initialization process normally consists of setting the baud rate, setting master mode of operation (by setting DDR_XCK to one), setting frame format and enabling the Transmitter and the Receiver. Only the transmitter can operate independently. For interrupt driven USART operation, the Global Interrupt Flag should be cleared (and thus interrupts globally disabled) when doing the initialization.

Note: To ensure immediate initialization of the XCK output the baud-rate register (UBRR) must be zero at the time the transmitter is enabled. Contrary to the normal mode USART operation the UBRR must then be written to the desired value after the transmitter is enabled, but before the first transmission is started. Setting UBRR to zero before enabling the transmitter is not necessary if the initialization is done immediately after a reset since UBRR is reset to zero.

Before doing a re-initialization with changed baud rate, data mode, or frame format, be sure that there is no ongoing transmissions during the period the registers are changed. The TXC Flag can be used to check that the Transmitter has completed all transfers, and the RXC Flag can be used to check that there are no unread data in the receive buffer. Note that the TXC Flag must be cleared before each transmission (before UDR is written) if it is used for this purpose.

The following simple USART initialization code examples show one assembly and one C function that are equal in functionality. The examples assume polling (no interrupts enabled). The

baud rate is given as a function parameter. For the assembly code, the baud rate parameter is assumed to be stored in the r17:r16 registers.

Assembly Code Example[1]

```
USART_Init:
  clr r18
  out UBRRH,r18
  out UBRRL,r18
  ; Setting the XCK port pin as output, enables master mode.
  sbi XCK_DDR, XCK
  ; Set MSPI mode of operation and SPI data mode 0.
  ldi r18, (1<<UMSEL1)|(1<<UMSEL0)|(0<<UCPHA)|(0<<UCPOL)
  out UCSRC,r18
  ; Enable receiver and transmitter.
  ldi r18, (1<<RXEN)|(1<<TXEN)
  out UCSRB,r18
  ; Set baud rate.
  ; IMPORTANT: The Baud Rate must be set after the transmitter is enabled!
  out UBRRH, r17
  out UBRRL, r18
  ret
```

C Code Example[1]

```
void USART_Init( unsigned int baud )
{
  UBRR = 0;
  /* Setting the XCK port pin as output, enables master mode. */
  XCK_DDR |= (1<<XCK);
  /* Set MSPI mode of operation and SPI data mode 0. */
  UCSRC = (1<<UMSEL1)|(1<<UMSEL0)|(0<<UCPHA)|(0<<UCPOL);
  /* Enable receiver and transmitter. */
  UCSRB = (1<<RXEN)|(1<<TXEN);
  /* Set baud rate. */
  /* IMPORTANT: The Baud Rate must be set after the transmitter is enabled
  */
  UBRR = baud;
}
```

Note: 1. See ”Code Examples” on page 7.

15.6 Data Transfer

Using the USART in MSPI mode requires the Transmitter to be enabled, i.e. the TXEN bit in the UCSRB register is set to one. When the Transmitter is enabled, the normal port operation of the TxD pin is overridden and given the function as the Transmitter's serial output. Enabling the receiver is optional and is done by setting the RXEN bit in the UCSRB register to one. When the receiver is enabled, the normal pin operation of the RxD pin is overridden and given the function as the Receiver's serial input. The XCK will in both cases be used as the transfer clock.

After initialization the USART is ready for doing data transfers. A data transfer is initiated by writing to the UDR I/O location. This is the case for both sending and receiving data since the transmitter controls the transfer clock. The data written to UDR is moved from the transmit buffer to the shift register when the shift register is ready to send a new frame.

Note: To keep the input buffer in sync with the number of data bytes transmitted, the UDR register must be read once for each byte transmitted. The input buffer operation is identical to normal USART mode, i.e. if an overflow occurs the character last received will be lost, not the first data in the buffer. This means that if four bytes are transferred, byte 1 first, then byte 2, 3, and 4, and the UDR is not read before all transfers are completed, then byte 3 to be received will be lost, and not byte 1.

The following code examples show a simple USART in MSPIM mode transfer function based on polling of the Data Register Empty (UDRE) Flag and the Receive Complete (RXC) Flag. The USART has to be initialized before the function can be used. For the assembly code, the data to be sent is assumed to be stored in Register R16 and the data received will be available in the same register (R16) after the function returns.

The function simply waits for the transmit buffer to be empty by checking the UDRE Flag, before loading it with new data to be transmitted. The function then waits for data to be present in the receive buffer by checking the RXC Flag, before reading the buffer and returning the value.

Assembly Code Example[1]

```
USART_MSPIM_Transfer:
  ; Wait for empty transmit buffer
  sbis UCSRA, UDRE
  rjmp USART_MSPIM_Transfer
  ; Put data (r16) into buffer, sends the data
  out UDR,r16
  ; Wait for data to be received
USART_MSPIM_Wait_RXC:
  sbis UCSRA, RXC
  rjmp USART_MSPIM_Wait_RXC
  ; Get and return received data from buffer
  in r16, UDR
  ret
```

C Code Example[1]

```
unsigned char USART_Receive( void )
{
  /* Wait for empty transmit buffer */
  while ( !( UCSRA & (1<<UDRE)) );
  /* Put data into buffer, sends the data */
  UDR = data;
  /* Wait for data to be received */
  while ( !(UCSRA & (1<<RXC)) );
  /* Get and return received data from buffer */
  return UDR;
}
```

Note: 1. See “Code Examples” on page 7.

15.6.1 Transmitter and Receiver Flags and Interrupts

The RXC, TXC, and UDRE flags and corresponding interrupts in USART in MSPIM mode are identical in function to the normal USART operation. However, the receiver error status flags (FE, DOR, and PE) are not in use and is always read as zero.

15.6.2 Disabling the Transmitter or Receiver

The disabling of the transmitter or receiver in USART in MSPIM mode is identical in function to the normal USART operation.

15.7 AVR USART MSPIM vs. AVR SPI

The USART in MSPIM mode is fully compatible with the AVR SPI regarding:

- Master mode timing diagram.
- The UCPOL bit functionality is identical to the SPI CPOL bit.
- The UCPHA bit functionality is identical to the SPI CPHA bit.
- The UDORD bit functionality is identical to the SPI DORD bit.

However, since the USART in MSPIM mode reuses the USART resources, the use of the USART in MSPIM mode is somewhat different compared to the SPI. In addition to differences of the control register bits, and that only master operation is supported by the USART in MSPIM mode, the following features differ between the two modules:

- The USART in MSPIM mode includes (double) buffering of the transmitter. The SPI has no buffer.
- The USART in MSPIM mode receiver includes an additional buffer level.
- The SPI WCOL (Write Collision) bit is not included in USART in MSPIM mode.
- The SPI double speed mode (SPI2X) bit is not included. However, the same effect is achieved by setting UBRRn accordingly.
- Interrupt timing is not compatible.
- Pin control differs due to the master only operation of the USART in MSPIM mode.

A comparison of the USART in MSPIM mode and the SPI pins is shown in Table 15-3 on page 152.

Table 15-3. Comparison of USART in MSPIM mode and SPI pins.

USART_MSPIM	SPI	Comment
TxD	MOSI	Master Out only
RxD	MISO	Master In only
XCK	SCK	(Functionally identical)
(N/A)	$\overline{SS}$	Not supported by USART in MSPIM

15.8 Register Description

The following section describes the registers used for SPI operation using the USART.

15.8.1 UDR – USART MSPIM I/O Data Register

The function and bit description of the USART data register (UDR) in MSPI mode is identical to normal USART operation. See “UDR – USART I/O Data Register” on page 137.

15.8.2 UCSRA – USART MSPIM Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
0x0B (0x2B)	RXC	TXC	UDRE	-	-	-	-	-	UCSRA
Read/Write	R	R/W	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	1	1	0	

- **Bit 7 – RXC: USART Receive Complete**

This flag bit is set when there are unread data in the receive buffer and cleared when the receive buffer is empty (i.e., does not contain any unread data). If the Receiver is disabled, the receive buffer will be flushed and consequently the RXC bit will become zero. The RXC Flag can be used to generate a Receive Complete interrupt (see description of the RXCIE bit).

- **Bit 6 – TXC: USART Transmit Complete**

This flag bit is set when the entire frame in the Transmit Shift Register has been shifted out and there are no new data currently present in the transmit buffer (UDR). The TXC Flag bit is automatically cleared when a transmit complete interrupt is executed, or it can be cleared by writing a one to its bit location. The TXC Flag can generate a Transmit Complete interrupt (see description of the TXCIE bit).

- **Bit 5 – UDRE: USART Data Register Empty**

The UDRE Flag indicates if the transmit buffer (UDR) is ready to receive new data. If UDRE is one, the buffer is empty, and therefore ready to be written. The UDRE Flag can generate a Data Register Empty interrupt (see description of the UDRIE bit). UDRE is set after a reset to indicate that the Transmitter is ready.

- **Bit 4:0 – Reserved Bits in MSPI mode**

When in MSPI mode, these bits are reserved for future use. For compatibility with future devices, these bits must be written to zero when UCSRA is written.

15.8.3 UCSRB – USART MSPIM Control and Status Register B

Bit	7	6	5	4	3	2	1	0	
0x0A (0x2A)	RXCIE	TXCIE	UDRIE	RXEN	TXEN	-	-	-	UCSRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R	R	R	
Initial Value	0	0	0	0	0	1	1	0	

- **Bit 7 – RXCIE: RX Complete Interrupt Enable**

Writing this bit to one enables interrupt on the RXC Flag. A USART Receive Complete interrupt will be generated only if the RXCIE bit is written to one, the Global Interrupt Flag in SREG is written to one and the RXC bit in UCSRA is set.

- **Bit 6 – TXCIE: TX Complete Interrupt Enable**

Writing this bit to one enables interrupt on the TXC Flag. A USART Transmit Complete interrupt will be generated only if the TXCIE bit is written to one, the Global Interrupt Flag in SREG is written to one and the TXC bit in UCSRA is set.

- **Bit 5 – UDRIE: USART Data Register Empty Interrupt Enable**

Writing this bit to one enables interrupt on the UDRE Flag. A Data Register Empty interrupt will be generated only if the UDRIE bit is written to one, the Global Interrupt Flag in SREG is written to one and the UDRE bit in UCSRA is set.

- **Bit 4 – RXEN: Receiver Enable**

Writing this bit to one enables the USART Receiver in MSPIM mode. The Receiver will override normal port operation for the RxD pin when enabled. Disabling the Receiver will flush the receive buffer. Only enabling the receiver in MSPI mode (i.e. setting RXEN=1 and TXEN=0) has no meaning since it is the transmitter that controls the transfer clock and since only master mode is supported.

- **Bit 3 – TXEN: Transmitter Enable**

Writing this bit to one enables the USART Transmitter. The Transmitter will override normal port operation for the TxD pin when enabled. The disabling of the Transmitter (writing TXEN to zero) will not become effective until ongoing and pending transmissions are completed, i.e., when the Transmit Shift Register and Transmit Buffer Register do not contain data to be transmitted. When disabled, the Transmitter will no longer override the TxD port.

- **Bit 2:0 – Reserved Bits in MSPI mode**

When in MSPI mode, these bits are reserved for future use. For compatibility with future devices, these bits must be written to zero when UCSRB is written.

15.8.4 UCSRC – USART MSPIM Control and Status Register C

Bit	7	6	5	4	3	2	1	0	
0x03 (0x23)	**UMSEL1**	**UMSEL0**	**-**	**-**	**-**	**UDORD**	**UCPHA**	**UCPOL**	**UCSRC**
Read/Write	R/W	R/W	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	1	1	0	

- **Bit 7:6 – UMSEL1:0: USART Mode Select**

These bits select the mode of operation of the USART as shown in Table 15-4. See “UCSRC – USART Control and Status Register C” on page 140 for full description of the normal USART operation. The MSPIM is enabled when both UMSEL bits are set to one. The UDORD, UCPHA, and UCPOL can be set in the same write operation where the MSPIM is enabled.

Table 15-4. UMSEL Bits Settings

UMSEL1	UMSEL0	Mode
0	0	Asynchronous USART
0	1	Synchronous USART
1	0	Reserved
1	1	Master SPI (MSPIM)

- **Bit 5:3 – Reserved Bits in MSPI mode**

When in MSPI mode, these bits are reserved for future use. For compatibility with future devices, these bits must be written to zero when UCSRC is written.

- **Bit 2 – UDORD: Data Order**

When set to one the LSB of the data word is transmitted first. When set to zero the MSB of the data word is transmitted first. Refer to the Frame Formats section page 4 for details.

- **Bit 1 – UCPHA: Clock Phase**

The UCPHA bit setting determine if data is sampled on the leasing edge (first) or tailing (last) edge of XCK. Refer to the SPI Data Modes and Timing section page 4 for details.

- **Bit 0 – UCPOL: Clock Polarity**

The UCPOL bit sets the polarity of the XCK clock. The combination of the UCPOL and UCPHA bit settings determine the timing of the data transfer. Refer to the SPI Data Modes and Timing section page 4 for details.

15.8.5 UBRRL and UBRRH – USART MSPIM Baud Rate Registers

The function and bit description of the baud rate registers in MSPI mode is identical to normal USART operation. See “UBRRL and UBRRH – USART Baud Rate Registers” on page 141.

16. USI – Universal Serial Interface

16.1 Features

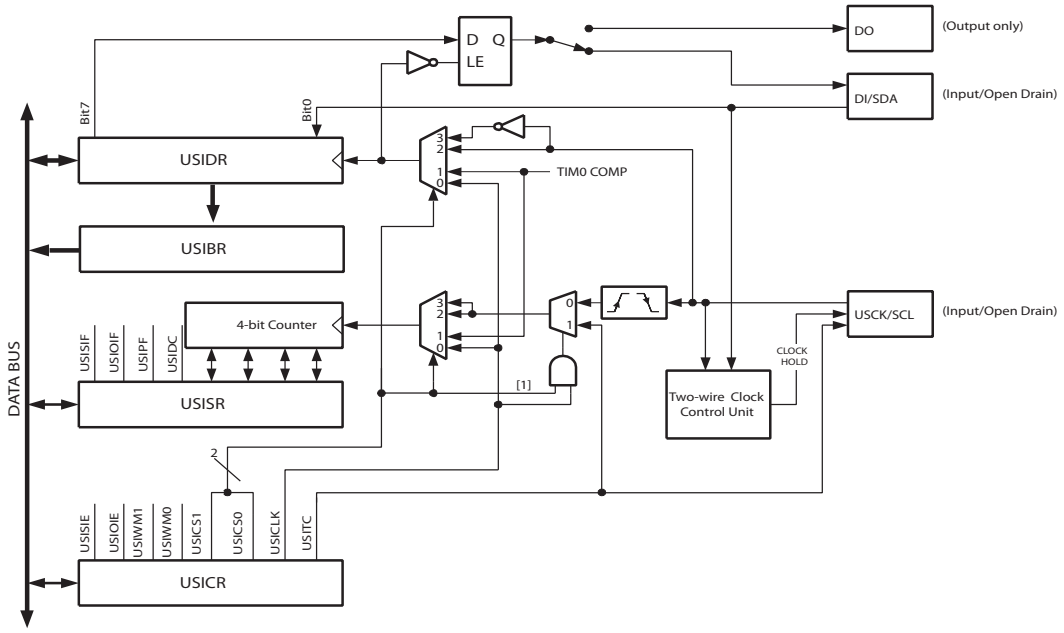
- **Two-wire Synchronous Data Transfer (Master or Slave)**
- **Three-wire Synchronous Data Transfer (Master or Slave)**
- **Data Received Interrupt**
- **Wakeup from Idle Mode**
- **In Two-wire Mode: Wake-up from All Sleep Modes, Including Power-down Mode**
- **Two-wire Start Condition Detector with Interrupt Capability**

16.2 Overview

The Universal Serial Interface (USI), provides the basic hardware resources needed for serial communication. Combined with a minimum of control software, the USI allows significantly higher transfer rates and uses less code space than solutions based on software only. Interrupts are included to minimize the processor load.

A simplified block diagram of the USI is shown in Figure 16-1 For actual placement of I/O pins refer to “Pinout ATtiny2313A/4313” on page 2. Device-specific I/O Register and bit locations are listed in the “Register Description” on page 163.

Figure 16-1. Universal Serial Interface, Block Diagram



The 8-bit USI Data Register (USIDR) contains the incoming and outgoing data. It is directly accessible via the data bus but a copy of the contents is also placed in the USI Buffer Register (USIBR) where it can be retrieved later. If reading the USI Data Register directly, the register must be read as quickly as possible to ensure that no data is lost.

The most significant bit of the USI Data Register is connected to one of two output pins (depending on the mode configuration, see “Analog Comparator” on page 168). There is a transparent latch between the output of the USI Data Register and the output pin, which delays the change

of data output to the opposite clock edge of the data input sampling. The serial input is always sampled from the Data Input (DI) pin independent of the configuration.

The 4-bit counter can be both read and written via the data bus, and it 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. This means the counter registers the number of clock edges and not the number of data 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 be configured to generate an interrupt when a start condition has been detected on the two-wire bus. It can also be set to generate wait states by holding the clock pin low after a start condition is detected, or after the counter overflows.

16.3 Functional Descriptions

16.3.1 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 (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 16-2. Three-wire Mode Operation, Simplified Diagram

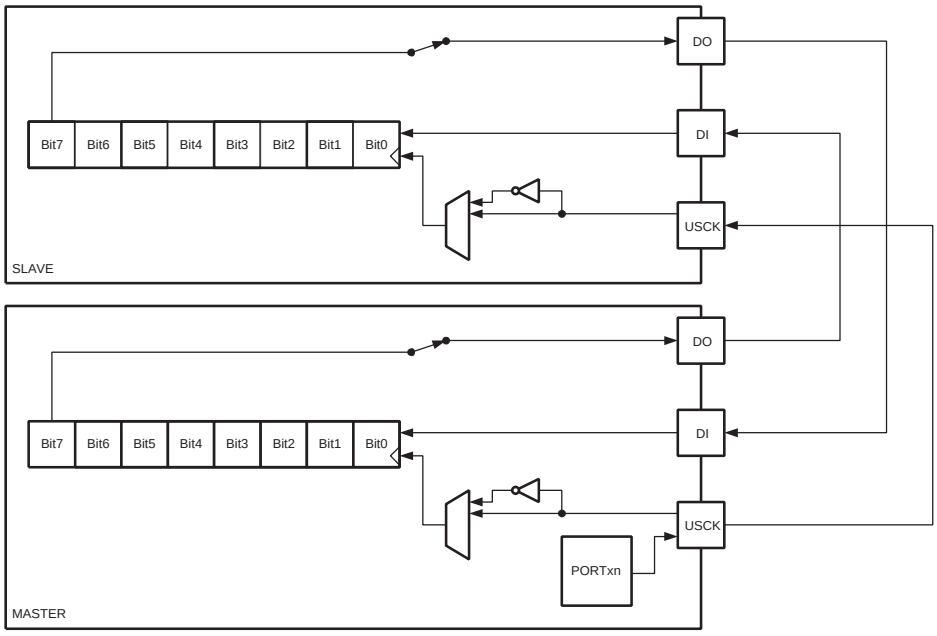
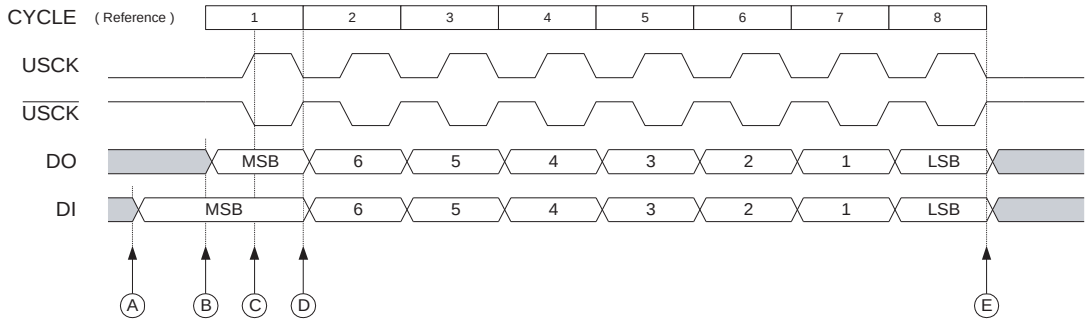


Figure 16-2 shows two USI units operating in three-wire mode, one as Master and one as Slave. The two USI Data Registers are interconnected in such way that after eight USCK clocks, the data in each register has been interchanged. The same clock also increments the USI's 4-bit counter. The Counter Overflow (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 PORTA register or by writing a one to bit USITC bit in USICR.

Figure 16-3. Three-wire Mode, Timing Diagram



The three-wire mode timing is shown in Figure 16-3 At the top of the figure is a USCK cycle reference. One bit is shifted into the USI Data Register (USIDR) for each of these cycles. The USCK timing is shown for both external clock modes. In external clock mode 0 (USICS0 = 0), DI is sampled at positive edges, and DO is changed (USI Data Register is shifted by one) at negative edges. In external clock mode 1 (USICS0 = 1) the opposite edges with respect to mode 0 are used. In other words, data is sampled at negative and output is changed at positive edges. The USI clock modes corresponds to the SPI data mode 0 and 1.

Referring to the timing diagram (Figure 16-3), a bus transfer involves the following steps:

1. The slave and master devices set up their data outputs and, depending on the protocol used, enable their output drivers (mark A and B). The output is set up by writing the data to be transmitted to the USI Data Register. The output is enabled by setting the corresponding bit in the Data Direction Register of Port A. Note that there is not a preferred order of points A and B in the figure, but both must be at least one half USCK cycle before point C, where the data is sampled. This is in order to ensure that the data setup requirement is satisfied. The 4-bit counter is reset to zero.
2. The master software generates a clock pulse by toggling the USCK line twice (C and D). The bit values on the data input (DI) pins are sampled by the USI on the first edge (C), and the data output is changed on the opposite edge (D). The 4-bit counter will count both edges.
3. Step 2. is repeated eight times for a complete register (byte) transfer.
4. After eight clock pulses (i.e., 16 clock edges) the counter will overflow and indicate that the transfer has been completed. If USI Buffer Registers are not used the data bytes that have been transferred must now be processed before a new transfer can be initiated. The overflow interrupt will wake up the processor if it is set to Idle mode. Depending on the protocol used the slave device can now set its output to high impedance.

16.3.2 SPI Master Operation Example

The following code demonstrates how to use the USI module as a SPI Master:

```
SPITransfer:
   out    USIDR,r16
   ldi    r16,(1<<USIOIF)
   out    USISR,r16
   ldi    r17,(1<<USIWM0)|(1<<USICS1)|(1<<USICLK)|(1<<USITC)

<continues>
```

```
<continued>

SPITransfer_loop:
   out    USICR,r17
   in     r16, USISR
   sbrs   r16, USIOIF
   rjmp   SPITransfer_loop
   in     r16,USIDR
   ret
```

The code is size optimized using only eight instructions (plus return). The code example assumes that the DO and USCK pins have been enabled as outputs in DDRA. The value stored in register r16 prior to the function is called is transferred to the slave device, and when the transfer is completed the data received from the slave is stored back into the register r16.

The second and third instructions clear the USI Counter Overflow Flag and the USI counter value. The fourth and fifth instructions set three-wire mode, positive edge clock, count at USITC strobe, and toggle USCK. The loop is repeated 16 times.

The following code demonstrates how to use the USI as an SPI master with maximum speed ($f_{SCK} = f_{CK}/2$):

```
SPITransfer_Fast:
   out    USIDR,r16
   ldi    r16,(1<<USIWM0)|(0<<USICS0)|(1<<USITC)
   ldi    r17,(1<<USIWM0)|(0<<USICS0)|(1<<USITC)|(1<<USICLK)

   out    USICR,r16 ; MSB
   out    USICR,r17
   out    USICR,r16
   out    USICR,r17
   out    USICR,r16
   out    USICR,r17
   out    USICR,r16
   out    USICR,r17
   out    USICR,r16
   out    USICR,r17
   out    USICR,r16
   out    USICR,r17
   out    USICR,r16
   out    USICR,r17
   out    USICR,r16 ; LSB
   out    USICR,r17

   in     r16,USIDR
ret
```

16.3.3 SPI Slave Operation Example

The following code demonstrates how to use the USI module as a SPI Slave:

```
init:
    ldi     r16,(1<<USIWM0)|(1<<USICS1)
    out     USICR,r16
...
SlaveSPITransfer:
    out     USIDR,r16
    ldi     r16,(1<<USIOIF)
    out     USISR,r16
SlaveSPITransfer_loop:
    in      r16, USISR
    sbrs    r16, USIOIF
    rjmp    SlaveSPITransfer_loop
    in      r16,USIDR
    ret
```

The code is size optimized using only eight instructions (plus return). The code example assumes that the DO and USCK pins have been enabled as outputs in DDRA. The value stored in register r16 prior to the function is called is transferred to the master device, and when the transfer is completed the data received from the master is stored back into the register r16.

Note that the first two instructions are for initialization, only, and need only be executed once. These instructions set three-wire mode and positive edge clock. The loop is repeated until the USI Counter Overflow Flag is set.

16.3.4 Two-wire Mode

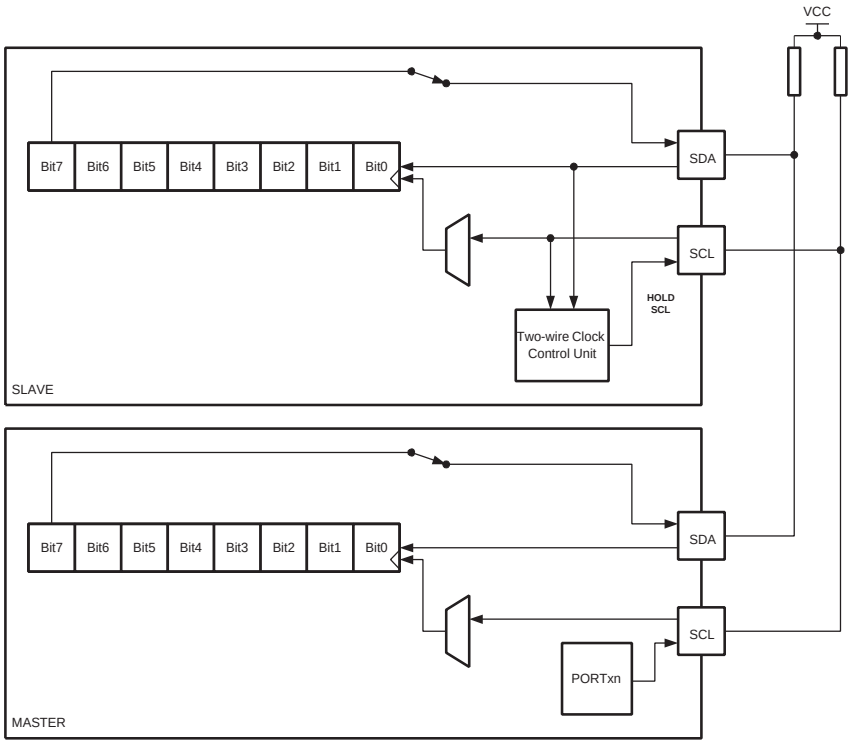
The USI two-wire mode is compliant to the Inter IC (TWI) bus protocol, but without slew rate limiting on outputs and without input noise filtering. Pin names used in this mode are SCL and SDA.

Figure 16-4 shows two USI units operating in two-wire mode, one as master and one as slave. It is only the physical layer that is shown since the system operation is highly dependent of the communication scheme used. The main differences between the master and slave operation at this level is the serial clock generation which is always done by the master. Only the slave uses the clock control unit.

Clock generation must be implemented in software, but the shift operation is done automatically in both devices. Note that clocking only on negative edges for shifting data is of practical use in this mode. The slave can insert wait states at start or end of transfer by forcing the SCL clock low. This means that the master must always check if the SCL line was actually released after it has generated a positive edge.

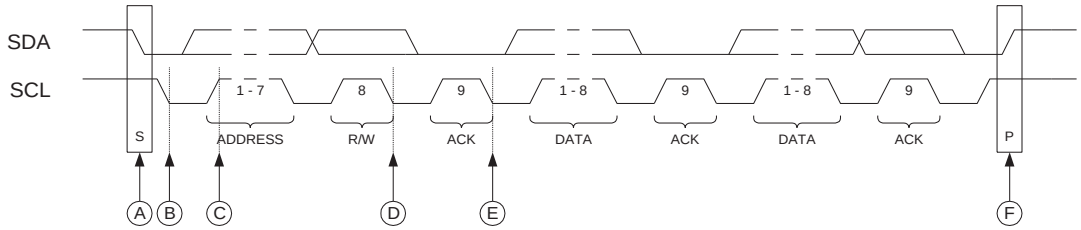
Since the clock also increments the counter, a counter overflow can be used to indicate that the transfer is completed. The clock is generated by the master by toggling the USCK pin via the PORTA register.

Figure 16-4. Two-wire Mode Operation, Simplified Diagram



The data direction is not given by the physical layer. A protocol, like the one used by the TWI-bus, must be implemented to control the data flow.

Figure 16-5. Two-wire Mode, Typical Timing Diagram



Referring to the timing diagram (Figure 16-5), a bus transfer involves the following steps:

1. The start condition is generated by the master by forcing the SDA low line while keeping the SCL line high (A). SDA can be forced low either by writing a zero to bit 7 of the USI Data Register, or by setting the corresponding bit in the PORTA register to zero. Note that the Data Direction Register bit must be set to one for the output to be enabled. The start detector logic of the slave device (see Figure 16-6 on page 162) detects the start condition and sets the USISIF Flag. The flag can generate an interrupt if necessary.
2. In addition, the start detector will hold the SCL line low after the master has forced a negative edge on this line (B). This allows the slave to wake up from sleep or complete other tasks before setting up the USI Data Register to receive the address. This is done by clearing the start condition flag and resetting the counter.

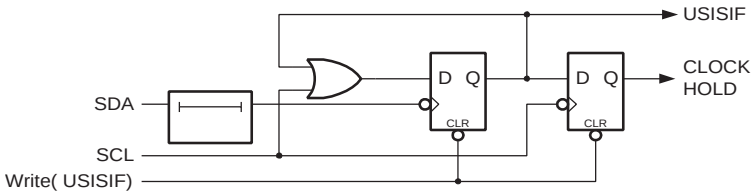
3. The master set the first bit to be transferred and releases the SCL line (C). The slave samples the data and shifts it into the USI Data Register at the positive edge of the SCL clock.
4. After eight bits containing slave address and data direction (read or write) have been transferred, the slave counter overflows and the SCL line is forced low (D). If the slave is not the one the master has addressed, it releases the SCL line and waits for a new start condition.
5. When the slave is addressed, it holds the SDA line low during the acknowledgment cycle before holding the SCL line low again (i.e., the USI Counter Register must be set to 14 before releasing SCL at (D)). Depending on the R/W bit the master or slave enables its output. If the bit is set, a master read operation is in progress (i.e., the slave drives the SDA line) The slave can hold the SCL line low after the acknowledge (E).
6. Multiple bytes can now be transmitted, all in same direction, until a stop condition is given by the master (F), or a new start condition is given.

If the slave is not able to receive more data it does not acknowledge the data byte it has last received. When the master does a read operation it must terminate the operation by forcing the acknowledge bit low after the last byte transmitted.

16.3.5 Start Condition Detector

The start condition detector is shown in Figure 16-6. The SDA line is delayed (in the range of 50 to 300 ns) to ensure valid sampling of the SCL line. The start condition detector is only enabled in two-wire mode.

Figure 16-6. Start Condition Detector, Logic Diagram



The start condition detector works asynchronously and can therefore wake up the processor from power-down sleep mode. However, the protocol used might have restrictions on the SCL hold time. Therefore, when using this feature the oscillator start-up time (set by CKSEL fuses, see “Clock Sources” on page 27) must also be taken into consideration. Refer to the description of the USISIF bit on page 165 for further details.

16.3.6 Clock speed considerations

Maximum frequency for SCL and SCK is f_{CK} / 2. This is also the maximum data transmit and receive rate in both two- and three-wire mode. In two-wire slave mode the Two-wire Clock Control Unit will hold the SCL low until the slave is ready to receive more data. This may reduce the actual data rate in two-wire mode.

16.4 Alternative USI Usage

The flexible design of the USI allows it to be used for other tasks when serial communication is not needed. Below are some examples.

16.4.1 Half-Duplex Asynchronous Data Transfer

Using the USI Data Register in three-wire mode it is possible to implement a more compact and higher performance UART than by software, only.

16.4.2 4-Bit Counter

The 4-bit counter can be used as a stand-alone counter with overflow interrupt. Note that if the counter is clocked externally, both clock edges will increment the counter value.

16.4.3 12-Bit Timer/Counter

Combining the 4-bit USI counter with one of the 8-bit timer/counters creates a 12-bit counter.

16.4.4 Edge Triggered External Interrupt

By setting the counter to maximum value (F) it can function as an additional external interrupt. The Overflow Flag and Interrupt Enable bit are then used for the external interrupt. This feature is selected by the USICS1 bit.

16.4.5 Software Interrupt

The counter overflow interrupt can be used as a software interrupt triggered by a clock strobe.

16.5 Register Description

16.5.1 USICR – USI Control Register

Bit	7	6	5	4	3	2	1	0	
0x0D (0x2D)	**USISIE**	**USIOIE**	**USIWM1**	**USIWM0**	**USICS1**	**USICS0**	**USICLK**	**USITC**	**USICR**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	W	W	
Initial Value	0	0	0	0	0	0	0	0	

The USI Control Register includes bits for interrupt enable, setting the wire mode, selecting the clock and clock strobe.

- **Bit 7 – USISIE: Start Condition Interrupt Enable**

Setting this bit to one enables the start condition detector interrupt. If there is a pending interrupt and USISIE and the Global Interrupt Enable Flag are set to one the interrupt will be executed immediately. Refer to the USISIF bit description on page 165 for further details.

- **Bit 6 – USIOIE: Counter Overflow Interrupt Enable**

Setting this bit to one enables the counter overflow interrupt. If there is a pending interrupt and USIOIE and the Global Interrupt Enable Flag are set to one the interrupt will be executed immediately. Refer to the USIOIF bit description on page 166 for further details.

- **Bit 5:4 – USIWM1:0: Wire Mode**

These bits set the type of wire mode to be used, as shown in Table 16-1 on page 164.

Basically, only the function of the outputs are affected by these bits. Data and clock inputs are not affected by the mode selected and will always have the same function. The counter and USI

Data Register can therefore be clocked externally and data input sampled, even when outputs are disabled.

Table 16-1. Relationship between USIWM1:0 and USI Operation

USIWM1	USIWM0	Description
0	0	**Outputs, clock hold, and start detector disabled.** Port pins operate as normal.
0	1	**Three-wire mode. Uses DO, DI, and USCK pins.** The *Data Output* (DO) pin overrides the corresponding bit in the PORTA register. However, the corresponding DDRA bit still controls the data direction. When the port pin is set as input the pin pull-up is controlled by the PORTA 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 PORTA 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 use open-collector output drives. The output drivers are enabled by setting the corresponding bit for SDA and SCL in the DDRA 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 PORTA 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 PORTA 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 in two-wire mode above, except that the SCL line is also held low when a counter overflow occurs, and 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 16-2 shows the relationship between the USICS1:0 and USICLK setting and clock source used for the USI Data Register and the 4-bit counter.

Table 16-2. Relationship between the USICS1:0 and USICLK Setting

USICS1	USICS0	USICLK	Clock Source	4-bit Counter Clock Source
0	0	0	No Clock	No Clock
0	0	1	Software clock strobe (USICLK)	Software clock strobe (USICLK)
0	1	X	Timer/Counter0 Compare Match	Timer/Counter0 Compare Match
1	0	0	External, positive edge	External, both edges
1	1	0	External, negative edge	External, both edges
1	0	1	External, positive edge	Software clock strobe (USITC)
1	1	1	External, negative edge	Software clock strobe (USITC)

- **Bit 1 – USICLK: Clock Strobe**

Writing a one to this bit location strobes the USI Data Register to shift one step and the counter to increment by one, provided that the software clock strobe option has been selected by writing USICS1:0 bits to zero. The output will change immediately when the clock strobe is executed, i.e., during the same instruction cycle. The value shifted into the USI Data Register is sampled the previous instruction cycle.

When an external clock source is selected (USICS1 = 1), the USICLK function is changed from a clock strobe to a Clock Select Register. Setting the USICLK bit in this case will select the USITC strobe bit as clock source for the 4-bit counter (see Table 16-2).

The bit will be read as zero.

- **Bit 0 – USITC: Toggle Clock Port Pin**

Writing a one to this bit location toggles the USCK/SCL value either from 0 to 1, or from 1 to 0. The toggling is independent of the setting in the Data Direction Register, but if the PORT value is to be shown on the pin the corresponding DDR pin must be set as output (to one). This feature allows easy clock generation when implementing master devices.

When an external clock source is selected (USICS1 = 1) and the USICLK bit is set to one, writing to the USITC strobe bit will directly clock the 4-bit counter. This allows an early detection of when the transfer is done when operating as a master device.

The bit will read as zero.

16.5.2 USISR – USI Status Register

Bit	7	6	5	4	3	2	1	0	
0x0E (0x2E)	USISIF	USIOIF	USIPF	USIDC	USICNT3	USICNT2	USICNT1	USICNT0	USISR
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The Status Register contains interrupt flags, line status flags and the counter value.

- **Bit 7 – USISIF: Start Condition Interrupt Flag**

When two-wire mode is selected, the USISIF Flag is set (to one) when a start condition has been detected. When three-wire mode or output disable mode has been selected any edge on the SCK pin will set the flag.

If USISIE bit in USICR and the Global Interrupt Enable Flag are set, an interrupt will be generated when this flag is set. The flag will only be cleared by writing a logical one to the USISIF bit. Clearing this bit will release the start detection hold of USCL in two-wire mode.

A start condition interrupt will wakeup the processor from all sleep modes.

- **Bit 6 – USIOIF: Counter Overflow Interrupt Flag**

This flag is set (one) when the 4-bit counter overflows (i.e., at the transition from 15 to 0). If the USIOIE bit in USICR and the Global Interrupt Enable Flag are set an interrupt will also be generated when the flag is set. The flag will only be cleared if a one is written to the USIOIF bit. Clearing this bit will release the counter overflow hold of SCL in two-wire mode.

A counter overflow interrupt will wakeup the processor from Idle sleep mode.

- **Bit 5 – USIPF: Stop Condition Flag**

When two-wire mode is selected, the USIPF Flag is set (one) when a stop condition has been detected. The flag is cleared by writing a one to this bit. Note that this is not an interrupt flag. This signal is useful when implementing two-wire bus master arbitration.

- **Bit 4 – USIDC: Data Output Collision**

This bit is logical one when bit 7 in the USI Data Register differs from the physical pin value. The flag is only valid when two-wire mode is used. This signal is useful when implementing Two-wire bus master arbitration.

- **Bits 3:0 – USICNT3:0: Counter Value**

These bits reflect the current 4-bit counter value. The 4-bit counter value can directly be read or written by the CPU.

The 4-bit counter increments by one for each clock generated either by the external clock edge detector, by a Timer/Counter0 Compare Match, or by software using USICLK or USITC strobe bits. The clock source depends on the setting of the USICS1:0 bits.

For external clock operation a special feature is added that allows the clock to be generated by writing to the USITC strobe bit. This feature is enabled by choosing an external clock source (USICS1 = 1) and writing a one to the USICLK bit.

Note that even when no wire mode is selected (USIWM1..0 = 0) the external clock input (USCK/SCL) can still be used by the counter.

16.5.3 USIDR – USI Data Register

Bit	7	6	5	4	3	2	1	0	
0x0F (0x2F)	MSB							LSB	USIDR
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 USI Data Register can be accessed directly but a copy of the data can also be found in the USI Buffer Register.

Depending on the USICS1:0 bits of the USI Control Register a (left) shift operation may be performed. The shift operation can be synchronised to an external clock edge, to a Timer/Counter0 Compare Match, or directly to software via the USICLK bit. If a serial clock occurs at the same cycle the register is written, the register will contain the value written and no shift is performed.

Note that even when no wire mode is selected (USIWM1:0 = 0) both the external data input (DI/SDA) and the external clock input (USCK/SCL) can still be used by the USI Data Register.

The output pin (DO or SDA, depending on the wire mode) is connected via the output latch to the most significant bit (bit 7) of the USI Data Register. The output latch ensures that data input is sampled and data output is changed on opposite clock edges. The latch is open (transparent) during the first half of a serial clock cycle when an external clock source is selected (USICS1 = 1) and constantly open when an internal clock source is used (USICS1 = 0). The output will be changed immediately when a new MSB is written as long as the latch is open.

Note that the Data Direction Register bit corresponding to the output pin must be set to one in order to enable data output from the USI Data Register.

16.5.4 USIBR – USI Buffer Register

Bit	7	6	5	4	3	2	1	0	
0x00 (0x20)	**MSB**							**LSB**	**USIBR**
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

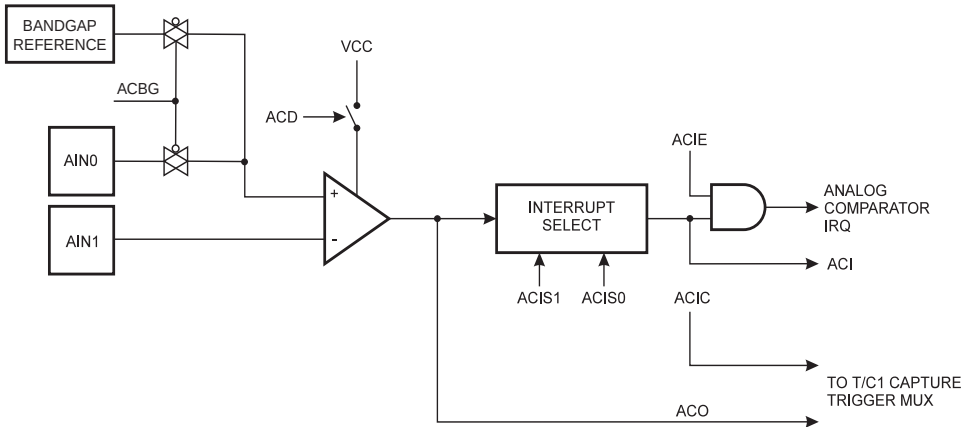
Instead of reading data from the USI Data Register the USI Buffer Register can be used. This makes controlling the USI less time critical and gives the CPU more time to handle other program tasks. USI flags as set similarly as when reading the USIDR register.

The content of the USI Data Register is loaded to the USI Buffer Register when the transfer has been completed.

17. Analog Comparator

The Analog Comparator compares the input values on the positive pin AIN0 and negative pin AIN1. When the voltage on the positive pin AIN0 is higher than the voltage on the negative pin AIN1, the Analog Comparator output, ACO, is set. The comparator's output can be set to trigger the Timer/Counter1 Input Capture function. In addition, the comparator can trigger a separate interrupt, exclusive to the Analog Comparator. The user can select Interrupt triggering on comparator output rise, fall or toggle. A block diagram of the comparator and its surrounding logic is shown in Figure 17-1.

Figure 17-1. Analog Comparator Block Diagram



17.1 Register Description

17.1.1 ACSR – Analog Comparator Control and Status Register

Bit	7	6	5	4	3	2	1	0	
0x08 (0x28)	**ACD**	**ACBG**	**ACO**	**ACI**	**ACIE**	**ACIC**	**ACIS1**	**ACIS0**	**ACSR**
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	N/A	0	0	0	0	0	

- **Bit 7 – ACD: Analog Comparator Disable**

When this bit is written logic one, the power to the Analog Comparator is switched off. This bit can be set at any time to turn off the Analog Comparator. This will reduce power consumption in Active and Idle mode. When changing the ACD bit, the Analog Comparator Interrupt must be disabled by clearing the ACIE bit in ACSR. Otherwise an interrupt can occur when the bit is changed.

- **Bit 6 – ACBG: Analog Comparator Bandgap Select**

When this bit is set, a fixed bandgap reference voltage replaces the positive input to the Analog Comparator. When this bit is cleared, AIN0 is applied to the positive input of the Analog Comparator. When the bandgap reference is used as input to the Analog Comparator, it will take a certain time for the voltage to stabilize. If not stibilized, the first conversion may give a wrong value. See "Internal Voltage Reference" on page 42.

- **Bit 5 – ACO: Analog Comparator Output**

The output of the Analog Comparator is synchronized and then directly connected to ACO. The synchronization introduces a delay of 1 - 2 clock cycles.

- **Bit 4 – ACI: Analog Comparator Interrupt Flag**

This bit is set by hardware when a comparator output event triggers the interrupt mode defined by ACIS1 and ACIS0. The Analog Comparator interrupt routine is executed if the ACIE bit is set and the I-bit in SREG is set. ACI is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, ACI is cleared by writing a logic one to the flag.

- **Bit 3 – ACIE: Analog Comparator Interrupt Enable**

When the ACIE bit is written logic one and the I-bit in the Status Register is set, the Analog Comparator interrupt is activated. When written logic zero, the interrupt is disabled.

- **Bit 2 – ACIC: Analog Comparator Input Capture Enable**

When written logic one, this bit enables the input capture function in Timer/Counter1 to be triggered by the Analog Comparator. The comparator output is in this case directly connected to the input capture front-end logic, making the comparator utilize the noise canceler and edge select features of the Timer/Counter1 Input Capture interrupt. When written logic zero, no connection between the Analog Comparator and the input capture function exists. To make the comparator trigger the Timer/Counter1 Input Capture interrupt, the ICIE1 bit in the Timer Interrupt Mask Register (TIMSK) must be set.

- **Bits 1, 0 – ACIS1, ACIS0: Analog Comparator Interrupt Mode Select**

These bits determine which comparator events that trigger the Analog Comparator interrupt. The different settings are shown in Table 17-1.

Table 17-1. ACIS1/ACIS0 Settings

ACIS1	ACIS0	Interrupt Mode
0	0	Comparator Interrupt on Output Toggle.
0	1	Reserved
1	0	Comparator Interrupt on Falling Output Edge.
1	1	Comparator Interrupt on Rising Output Edge.

When changing the ACIS1/ACIS0 bits, the Analog Comparator Interrupt must be disabled by clearing its Interrupt Enable bit in the ACSR Register. Otherwise an interrupt can occur when the bits are changed.

17.1.2 DIDR – Digital Input Disable Register

Bit	7	6	5	4	3	2	1	0	
0x01 (0x21)	–	–	–	–	–	–	AIN1D	AIN0D	DIDR
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 1, 0 – AIN1D, AIN0D: AIN1, AIN0 Digital Input Disable**

When this bit is written logic one, the digital input buffer on the AIN1/0 pin is disabled. The corresponding PIN Register bit will always read as zero when this bit is set. When an analog signal is applied to the AIN1/0 pin and the digital input from this pin is not needed, this bit should be written logic one to reduce power consumption in the digital input buffer.

18. debugWIRE On-chip Debug System

18.1 Features

- **Complete Program Flow Control**
- **Emulates All On-chip Functions, Both Digital and Analog, except RESET Pin**
- **Real-time Operation**
- **Symbolic Debugging Support (Both at C and Assembler Source Level, or for other HLLs)**
- **Unlimited Number of Program Break Points (Using Software Break Points)**
- **Non-intrusive Operation**
- **Electrical Characteristics Identical to Real Device**
- **Automatic Configuration System**
- **High-Speed Operation**
- **Programming of Non-volatile Memories**

18.2 Overview

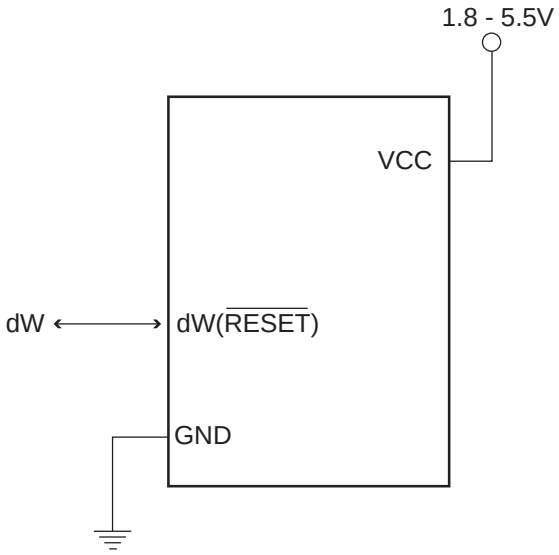
The debugWIRE On-chip debug system uses a One-wire, bi-directional interface to control the program flow, execute AVR instructions in the CPU and to program the different non-volatile memories.

18.3 Physical Interface

When the debugWIRE Enable (DWEN) Fuse is programmed and Lock bits are unprogrammed, the debugWIRE system within the target device is activated. The RESET port pin is configured as a wire-AND (open-drain) bi-directional I/O pin with pull-up enabled and becomes the communication gateway between target and emulator.

Figure 18-1 shows the schematic of a target MCU, with debugWIRE enabled, and the emulator connector. The system clock is not affected by debugWIRE and will always be the clock source selected by the CKSEL Fuses.

Figure 18-1. The debugWIRE Setup



When designing a system where debugWIRE will be used, the following observations must be made for correct operation:

- Pull-Up resistor on the dW/(RESET) line must be larger than 10k. However, the pull-up resistor is optional.
- Connecting the RESET pin directly to V_{CC} will not work.
- Capacitors inserted on the RESET pin must be disconnected when using debugWire.
- All external reset sources must be disconnected.

18.4 Software Break Points

debugWIRE supports Program memory Break Points by the AVR Break instruction. Setting a Break Point in AVR Studio® will insert a BREAK instruction in the Program memory. The instruction replaced by the BREAK instruction will be stored. When program execution is continued, the stored instruction will be executed before continuing from the Program memory. A break can be inserted manually by putting the BREAK instruction in the program.

The Flash must be re-programmed each time a Break Point is changed. This is automatically handled by AVR Studio through the debugWIRE interface. The use of Break Points will therefore reduce the Flash Data retention. Devices used for debugging purposes should not be shipped to end customers.

18.5 Limitations of debugWIRE

The debugWIRE communication pin (dW) is physically located on the same pin as External Reset (RESET). An External Reset source is therefore not supported when the debugWIRE is enabled.

The debugWIRE system accurately emulates all I/O functions when running at full speed, i.e., when the program in the CPU is running. When the CPU is stopped, care must be taken while accessing some of the I/O Registers via the debugger (AVR Studio). See the debugWIRE documentation for detailed description of the limitations.

The debugWIRE interface is asynchronous, which means that the debugger needs to synchronize to the system clock. If the system clock is changed by software (e.g. by writing CLKPS bits) communication via debugWIRE may fail. Also, clock frequencies below 100 kHz may cause communication problems.

A programmed DWEN Fuse enables some parts of the clock system to be running in all sleep modes. This will increase the power consumption while in sleep. Thus, the DWEN Fuse should be disabled when debugWire is not used.

18.6 Register Description

The following section describes the registers used with the debugWire.

18.6.1 DWDR – debugWire Data Register

Bit	7	6	5	4	3	2	1	0	
	DWDR[7:0]								DWDR
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 DWDR Register provides a communication channel from the running program in the MCU to the debugger. This register is only accessible by the debugWIRE and can therefore not be used as a general purpose register in the normal operations.

19. Self-Programming

19.1 Features

- **Self-Programming Enables MCU to Erase, Write and Reprogram Application Memory**
- **Efficient Read-Modify-Write Support**
- **Lock Bits Allow Application Memory to Be Securely Closed for Further Access**

19.2 Overview

The device provides a self-programming mechanism for downloading and uploading program code by the MCU itself. Self-Programming can use any available data interface and associated protocol to read code and write (program) that code into program memory.

19.3 Lock Bits

Program memory can be protected from internal or external access. See “Lock Bits” on page 178.

19.4 Self-Programming the Flash

Program memory is updated in a page by page fashion. Before programming a page with the data stored in the temporary page buffer, the page must be erased. The temporary page buffer is filled one word at a time using SPM and the buffer can be filled either before the Page Erase command or between a Page Erase and a Page Write operation:

1. Either, fill the buffer before a Page Erase:
   a. Fill temporary page buffer
   b. Perform a Page Erase
   c. Perform a Page Write
2. Or, fill the buffer after Page Erase:
   a. Perform a Page Erase
   b. Fill temporary page buffer
   c. Perform a Page Write

If only a part of the page needs to be changed, the rest of the page must be stored (for example in the temporary page buffer) before the erase, and then be re-written.

The temporary page buffer can be accessed in a random sequence.

It is essential that the page address used in both the Page Erase and Page Write operation is addressing the same page.

The SPM instruction is disabled by default but it can be enabled by programming the SELFPRGEN fuse (to “0”).

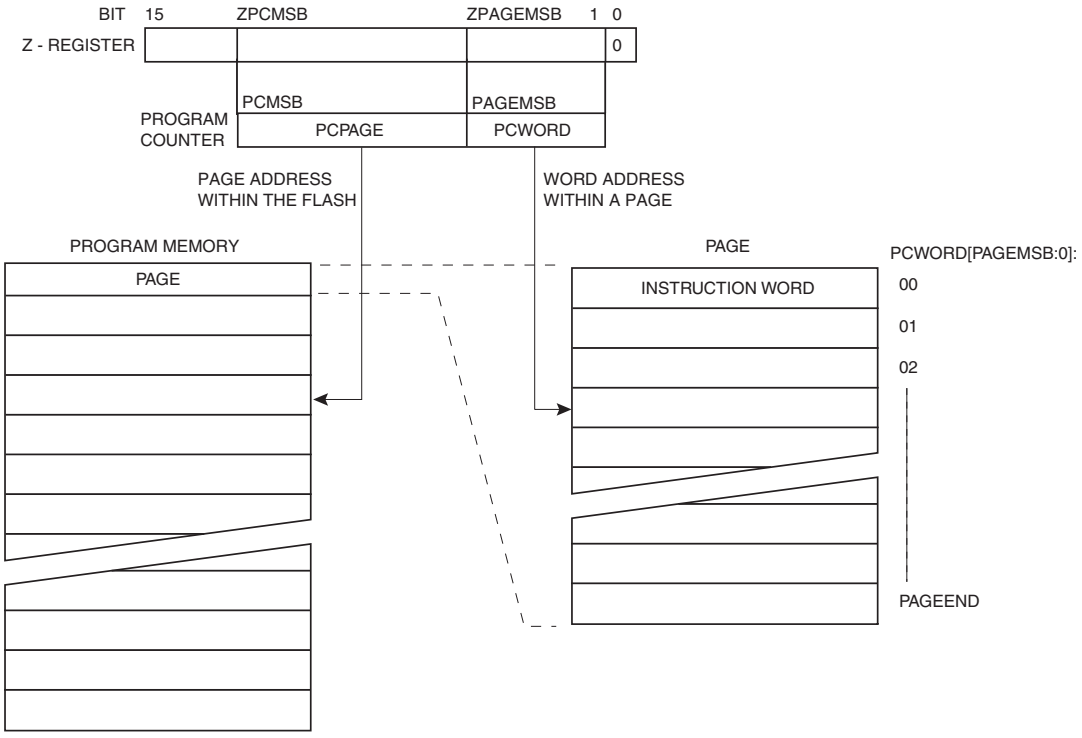
19.4.1 Addressing the Flash During Self-Programming

The Z-pointer is used to address the SPM commands.

Bit	15	14	13	12	11	10	9	8
ZH (R31)	**Z15**	**Z14**	**Z13**	**Z12**	**Z11**	**Z10**	**Z9**	**Z8**
ZL (R30)	**Z7**	**Z6**	**Z5**	**Z4**	**Z3**	**Z2**	**Z1**	**Z0**
	7	6	5	4	3	2	1	0

Since the Flash is organized in pages (see Table 21-1 on page 184), the Program Counter can be treated as having two different sections. One section, consisting of the least significant bits, is addressing the words within a page, while the most significant bits are addressing the pages. This is shown in Figure 19-1, below.

Figure 19-1. Addressing the Flash During SPM



Variables used in Figure 19-1 are explained in Table 19-1, below.

Table 19-1. Variables Used in Flash Addressing

Variable	Description
PCPAGE	Program Counter page address. Selects page of words and is used with Page Erase and Page Write operations. See Table 21-1 on page 184
PCMSB	The most significant bit of the Program Counter. See Table 21-1 on page 184
ZPCMSB	The bit in the Z register that is mapped to PCMSB. Because Z[0] is not used, ZPCMSB = PCMSB + 1. Z register bits above ZPCMSB are ignored
PCWORD	Program Counter word address. Selects the word within a page. This is used for filling the temporary buffer and must be zero during page write operations. See Table 21-1 on page 184
PAGEMSB	The most significant bit used to address the word within one page
ZPAGEMSB	The bit in the Z register that is mapped to PAGEMSB. Because Z[0] is not used, ZPAGEMSB = PAGEMSB + 1

Note that the Page Erase and Page Write operations are addressed independently. Therefore it is of major importance that the software addresses the same page in both the Page Erase and Page Write operation.

Although the least significant bit of the Z-register (Z0) should be zero for SPM, it should be noted that the LPM instruction addresses the Flash byte-by-byte and uses Z0 as a byte select bit.

Once a programming operation is initiated, the address is latched and the Z-pointer can be used for other operations.

19.4.2 Page Erase

To execute Page Erase:

- Set up the address in the Z-pointer
- Write “00000011” to SPMCSR
- Execute an SPM instruction within four clock cycles after writing SPMCSR

The data in R1 and R0 is ignored. The page address must be written to PCPAGE in the Z-register. Other bits in the Z-pointer are ignored during this operation.

If an interrupt occurs during the timed sequence above the four cycle access cannot be guaranteed. In order to ensure atomic operation interrupts should be disabled before writing to SPMCSR.

The CPU is halted during the Page Erase operation.

19.4.3 Page Load

To write an instruction word:

- Set up the address in the Z-pointer
- Set up the data in R1:R0
- Write “00000001” to SPMCSR
- Execute an SPM instruction within four clock cycles after writing SPMCSR

The content of PCWORD in the Z-register is used to address the data in the temporary buffer. The temporary buffer will auto-erase after a Page Write operation, or by writing the RWWSRE bit in SPMCSR. It is also erased after a system reset.

Note that it is not possible to write more than one time to each address without erasing the temporary buffer.

If the EEPROM is written in the middle of an SPM Page Load operation, all data loaded will be lost.

19.4.4 Page Write

To execute Page Write:

- Set up the address in the Z-pointer
- Write “00000101” to SPMCSR
- Execute an SPM instruction within four clock cycles after writing SPMCSR

The data in R1 and R0 is ignored. The page address must be written to PCPAGE. Other bits in the Z-pointer must be written to zero during this operation.

The CPU is halted during the Page Write operation.

19.4.5 SPMCSR Can Not Be Written When EEPROM is Programmed

Note that an EEPROM write operation will block all software programming to Flash. Reading 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 EECR and verifies that it is cleared before writing to SPMCSR.

19.5 Preventing Flash Corruption

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

A Flash program corruption can be caused by two situations when the voltage is too low. First, a regular write sequence to the Flash requires a minimum voltage to operate correctly. Secondly, the CPU itself can execute instructions incorrectly, if the supply voltage for executing instructions is too low.

Flash corruption can easily be avoided by following these design recommendations (one is sufficient):

1. Keep the AVR RESET active (low) during periods of insufficient power supply voltage. This can be done by enabling the internal Brown-out Detector (BOD) if the operating voltage matches the detection level. If not, an external low V_{CC} reset protection circuit can be used. If a reset occurs while a write operation is in progress, the write operation will be completed provided that the power supply voltage is sufficient.
2. Keep the AVR core in Power-down sleep mode during periods of low V_{CC}. This will prevent the CPU from attempting to decode and execute instructions, effectively protecting the SPMCSR Register and thus the Flash from unintentional writes.

19.6 Programming Time for Flash when Using SPM

Flash access is timed using the internal, calibrated 8MHz oscillator. Typical Flash programming times for the CPU are shown in Table 19-2.

Table 19-2. SPM Programming Time

Operation	Min (1)	Max (1)
SPM: Flash Page Erase, Flash Page Write, and lock bit write	3.7 ms	4.5 ms

Note: 1. Min and max programming times are per individual operation.

19.7 Register Description

19.7.1 SPMCSR – Store Program Memory Control and Status Register

The Store Program Memory Control and Status Register contains the control bits needed to control the Program memory operations.

Bit	7	6	5	4	3	2	1	0	
0x37 (0x57)	–	–	RSIG	CTPB	RFLB	PGWRT	PGERS	SPMEN	SPMCSR
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7, 6 – Res: Reserved Bits**

These bits are reserved bits in the ATtiny2313A/4313 and always read as zero.

- **Bit 5 – RSIG: Read Device Signature Imprint Table**

Issuing an LPM instruction within three cycles after RSIG and SPMEN bits have been set in SPMCSR will return the selected data (depending on Z-pointer value) from the device signature imprint table into the destination register. See “Device Signature Imprint Table” on page 180 for details.

- **Bit 4 – CTPB: Clear Temporary Page Buffer**

If the CTPB bit is written while filling the temporary page buffer, the temporary page buffer will be cleared and the data will be lost.

- **Bit 3 – RFLB: Read Fuse and Lock Bits**

An LPM instruction within three cycles after RFLB and SPMEN are set in the SPMCSR Register, will read either the Lock bits or the Fuse bits (depending on Z0 in the Z-pointer) into the destination register. See “SPMCSR Can Not Be Written When EEPROM is Programmed” on page 176 for details.

- **Bit 2 – PGWRT: Page Write**

If this bit is written to one at the same time as SPMEN, the next SPM instruction within four clock cycles executes Page Write, with the data stored in the temporary buffer. The page address is taken from the high part of the Z-pointer. The data in R1 and R0 are ignored. The PGWRT bit will auto-clear upon completion of a Page Write, or if no SPM instruction is executed within four clock cycles. The CPU is halted during the entire Page Write operation.

- **Bit 1 – PGERS: Page Erase**

If this bit is written to one at the same time as SPMEN, the next SPM instruction within four clock cycles executes Page Erase. The page address is taken from the high part of the Z-pointer. The data in R1 and R0 are ignored. The PGERS bit will auto-clear upon completion of a Page Erase, or if no SPM instruction is executed within four clock cycles. The CPU is halted during the entire Page Write operation.

- **Bit 0 – SPMEN: Self Programming Enable**

This bit enables the SPM instruction for the next four clock cycles. If written to one together with either RSIG, CTPB, RFLB, PGWRT, or PGERS, the following SPM instruction will have a special meaning, see description above. If only SPMEN is written, the following SPM instruction will store the value in R1:R0 in the temporary page buffer addressed by the Z-pointer. The LSB of the Z-pointer is ignored. The SPMEN bit will auto-clear upon completion of an SPM instruction, or if no SPM instruction is executed within four clock cycles. During Page Erase and Page Write, the SPMEN bit remains high until the operation is completed.

Writing any other combination than “100001”, “010001”, “001001”, “000101”, “000011” or “000001” in the lower six bits will have no effect.

20. Lock Bits, Fuse Bits and Device Signature

20.1 Lock Bits

ATtiny2313A/4313 provides the program and data memory lock bits listed in Table 20-1.

Table 20-1. Lock Bit Byte

Lock Bit Byte	Bit No	Description	See	Default Value [1]
–	7	–		1 (unprogrammed)
–	6	–		1 (unprogrammed)
–	5	–		1 (unprogrammed)
–	4	–		1 (unprogrammed)
–	3	–		1 (unprogrammed)
–	2	–		1 (unprogrammed)
LB2	1	Lock bit	Below	1 (unprogrammed)
LB1	0			1 (unprogrammed)

Notes: 1. “1” means unprogrammed, “0” means programmed.

Lock bits can be left unprogrammed (“1”) or can be programmed (“0”) to obtain the additional features listed in Table 20-2.

Table 20-2. Lock Bit Protection Modes.

Lock Bits [1]		
LB2	LB1	Mode of Protection
1	1	No memory lock features enabled
1	0	Further programming of Flash and EEPROM is disabled in parallel and serial programming mode. Fuse bits are locked in both serial and parallel programming mode [2]
0	1	Reserved
0	0	Further reading and programming of Flash and EEPROM is disabled in parallel and serial programming mode. Fuse bits are locked in both serial and parallel programming mode [2]

Notes: 1. “1” means unprogrammed, “0” means programmed.
2. Program fuse bits before programming LB1 and LB2.

When programming the lock bits, the mode of protection can be increased, only. Writing the same, or lower, mode of protection automatically results in maximum protection.

Lock bits can be erased to “1” with the Chip Erase command, only.

The ATtiny2313A/4313 has no separate boot loader section. The SPM instruction is enabled for the whole Flash if the SELFPRGEN fuse is programmed (“0”), otherwise it is disabled.

20.2 Fuse Bits

Fuse bits are described in Table 20-3, Table 20-4, and Table 20-5. Note that programmed fuses read as zero.

Table 20-3. Extended Fuse Byte

Bit #	Bit Name	Use	See	Default Value
7	–	–		1 (unprogrammed)
6	–	–		1 (unprogrammed)
5	–	–		1 (unprogrammed)
4	–	–		1 (unprogrammed)
3	–	–		1 (unprogrammed)
2	–	–		1 (unprogrammed)
1	–	–		1 (unprogrammed)
0	SELFPRGEN	Enables SPM instruction	Page 173	1 (unprogrammed)

Table 20-4. High Fuse Byte

Bit #	Bit Name	Use	See	Default Value
7	DWEN	Enables debugWIRE (1)	Page 170	1 (unprogrammed)
6	EESAVE	Preserves EEPROM memory during Chip Erase operation	Page 187	1 (unprogrammed) (2)
5	SPIEN	Enables serial programming and downloading of data to device (3)		0 (programmed) (4)
4	WDTON	Sets watchdog timer permanently on	Page 45	1 (unprogrammed)
3	BODLEVEL2	Sets BOD trigger level	Page 202	1 (unprogrammed)
2	BODLEVEL1			1 (unprogrammed)
1	BODLEVEL0			1 (unprogrammed)
0	RSTDISBL	Disables external reset (1)	Page 41	1 (unprogrammed)

Notes:
1. Programming this fuse bit will change the functionality of the $\overline{\text{RESET}}$ pin and render further programming via the serial interface impossible. The fuse bit can be unprogrammed using the parallel programming algorithm (see page 184).
2. This setting does not preserve EEPROM.
3. This fuse bit is not accessible in serial programming mode.
4. This setting enables SPI programming.

Table 20-5. Low Fuse Byte

Bit #	Bit Name	Use	See	Default Value
7	CKDIV8	Divides clock by 8 (1)	Page 28	0 (programmed)
6	CKOUT	Outputs system clock on port pin	Page 32	1 (unprogrammed)
5	SUT1	–	Pages 28, 29, and 31	1 (unprogrammed)
4	SUT0	Sets system start-up time		0 (programmed) (2)
3	CKSEL3	Selects clock source	Page 27	0 (programmed) (3)
2	CKSEL2			0 (programmed) (3)
1	CKSEL1			1 (unprogrammed) (3)
0	CKSEL0			0 (programmed) (3)

Note: 1. Unprogramming this fuse at low voltages may result in overclocking. See Section 22.3 on page 200 for device speed versus supply voltage.
2. This setting results in maximum start-up time for the default clock source.
3. This setting selects Calibrated Internal RC Oscillator.

Fuse bits are locked when Lock Bit 1 (LB1) is programmed. Hence, fuse bits must be programmed before lock bits.

Fuse bits are not affected by a Chip Erase.

20.2.1 Latching of Fuses

The fuse values are latched when the device enters programming mode and changes of the fuse values will have no effect until the part leaves Programming mode. This does not apply to the EESAVE fuse, which will take effect once it is programmed. The fuses are also latched on Power-up in Normal mode.

20.3 Device Signature Imprint Table

The device signature imprint table is a dedicated read only memory area used for storing miscellaneous device information, such as the device signature and oscillator calibration data. Most of this memory segment is reserved for internal use, as outlined in Table 20-6.

Byte addresses are used when the device itself reads the data with the LPM command. External programming devices must use word addresses.

Table 20-6. Contents of Device Signature Imprint Table.

Word Address	Byte Address	High Byte
0x00	0x00	Signature byte 0 (1)
	0x01	Calibration data for internal 8 MHz oscillator (2)
0x01	0x02	Signature byte 1 (1)
	0x03	Calibration data for internal 4 MHz oscillator (2)
0x02	0x04	Signature byte 2 (1)
	0x05 ...	Reserved for internal use

Notes: 1. See section “Signature Bytes” for more information.

2. See section “Calibration Byte” for more information.

20.3.1 Calibration Byte

The signature area of the ATtiny2313A/4313 contains two bytes of calibration data for the internal oscillator. The calibration data in the high byte of address 0x00 is for use with the oscillator set to 8.0 MHz operation. During reset, this byte is automatically written into the OSCCAL register to ensure correct frequency of the oscillator.

There is a separate calibration byte for the internal oscillator in 4.0 MHz mode of operation but this data is not loaded automatically. The hardware always loads the 8.0 MHz calibration data during reset. To use separate calibration data for the oscillator in 4.0 MHz mode the OSCCAL register must be updated by firmware. The calibration data for 4.0 MHz operation is located in the high byte at address 0x01 of the signature area.

20.3.2 Signature Bytes

All Atmel microcontrollers have a three-byte signature code which identifies the device. This code can be read in both serial and parallel mode, also when the device is locked. The three bytes reside in a separate address space.

For the ATtiny2313A the signature bytes are:

1. 0x000: 0x1E (indicates manufactured by Atmel).
2. 0x001: 0x91 (indicates 2KB Flash memory).
3. 0x002: 0x0A (indicates ATtiny2313A device when 0x001 is 0x91).

For the ATtiny4313 the signature bytes are:

1. 0x000: 0x1E (indicates manufactured by Atmel).
2. 0x001: 0x92 (indicates 4KB Flash memory).
3. 0x002: 0x0D (indicates ATtiny4313 device when 0x001 is 0x92).

20.4 Reading Lock Bits, Fuse Bits and Signature Data from Software

Fuse and lock bits can be read by device firmware. Programmed fuse and lock bits read zero. unprogrammed as one. See “Lock Bits” on page 178 and “Fuse Bits” on page 179.

In addition, firmware can also read data from the device signature imprint table. See “Device Signature Imprint Table” on page 180.

20.4.1 Lock Bit Read

Lock bit values are returned in the destination register after an LPM instruction has been issued within three CPU cycles after RFLB and SPMEN bits have been set in SPMCSR (see page 176). The RFLB and SPMEN bits automatically clear upon completion of reading the lock bits, or if no LPM instruction is executed within three CPU cycles, or if no SPM instruction is executed within four CPU cycles. When RFLB and SPMEN are cleared LPM functions normally.

To read the lock bits, follow the below procedure:

1. Load the Z-pointer with 0x0001.
2. Set RFLB and SPMEN bits in SPMCSR.
3. Issue an LPM instruction within three clock cycles.
4. Read the lock bits from the LPM destination register.

If successful, the contents of the destination register are as follows.

Bit	7	6	5	4	3	2	1	0
Rd	–	–	–	–	–	–	LB2	LB1

See section “Parallel Programming” on page 184 for more information.

20.4.2 Fuse Bit Read

The algorithm for reading fuse bytes is similar to the one described above for reading lock bits, only the addresses are different.

To read the Fuse Low Byte (FLB), follow the below procedure:

1. Load the Z-pointer with 0x0000.
2. Set RFLB and SPMEN bits in SPMCSR.
3. Issue an LPM instruction within three clock cycles.
4. Read the FLB from the LPM destination register.

If successful, the contents of the destination register are as follows.

Bit	7	6	5	4	3	2	1	0
Rd	FLB7	FLB6	FLB5	FLB4	FLB3	FLB2	FLB1	FLB0

Refer to Table 20-5 on page 180 for a detailed description and mapping of the Fuse Low Byte.

To read the Fuse High Byte (FHB), simply replace the address in the Z-pointer with 0x0003 and repeat the procedure above. If successful, the contents of the destination register are as follows.

Bit	7	6	5	4	3	2	1	0
Rd	FHB7	FHB6	FHB5	FHB4	FHB3	FHB2	FHB1	FHB0

Refer to Table 20-4 on page 179 for detailed description and mapping of the Fuse High Byte.

To read the Fuse Extended Byte (FEB), replace the address in the Z-pointer with 0x0002 and repeat the previous procedure. If successful, the contents of the destination register are as follows.

Bit	7	6	5	4	3	2	1	0
Rd	FEB7	FEB6	FEB5	FEB4	FEB3	FEB2	FEB1	FEB0

Refer to Table 20-3 on page 179 for detailed description and mapping of the Fuse Extended Byte.

20.4.3 Device Signature Imprint Table Read

To read the contents of the device signature imprint table, follow the below procedure:

1. Load the Z-pointer with the table index.
2. Set RSIG and SPMEN bits in SPMCSR.
3. Issue an LPM instruction within three clock cycles.

4. Wait three clock cycles for SPMEN bits to be cleared.
5. Read table data from the LPM destination register.

If successful, the contents of the destination register are as described in section “Device Signature Imprint Table” on page 180.

The RSIG and SPMEN bits will auto-clear after three CPU cycles. When RSIG and SPMEN are cleared, LPM will work as described in the “AVR Instruction Set” description.

See program example below.

Assembly Code Example

```
DSIT_read:
  ; Uses Z-pointer as table index
  ldi  ZH, 0
  ldi  ZL, 1
  ; Preload SPMCSR bits into R16, then write to SPMCSR
  ldi  r16, (1<<RSIG)|(1<<SPMEN)
  out  SPMCSR, r16
  ; Issue LPM. Table data will be returned into r17
  lpm  r17, Z
  ret
```

Note: See “Code Examples” on page 7.

21. External Programming

This section describes how to program and verify Flash memory, EEPROM, lock bits, and fuse bits in ATtiny2313A/4313.

21.1 Memory Parametrics

Flash memory parametrics are summarised in Table 21-1, below.

Table 21-1. No. of Words in a Page and No. of Pages in the Flash

Device	Flash Size	Page Size	PCWORD[1]	Pages	PCPAGE	PCMSB
ATtiny2313A	1K word (2K bytes)	16 words	PC[3:0]	64	PC[9:4]	9
ATtiny4313	2K words (4K bytes)	32 words	PC[4:0]	64	PC[10:5]	10

Note: 1. See Table 19-1 on page 174.

EEPROM parametrics are summarised in Table 21-2, below.

Table 21-2. No. of Words in a Page and No. of Pages in the EEPROM

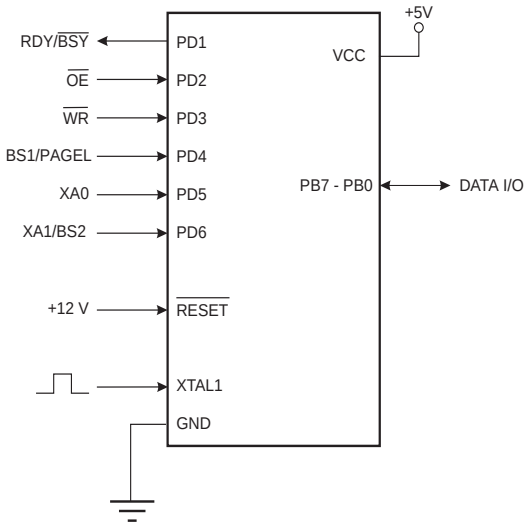
Device	EEPROM Size	Page Size	PCWORD[1]	Pages	PCPAGE[1]	EEAMSB
ATtiny2313A	128 bytes	4 bytes	EEA[1:0]	32	EEA[6:2]	6
ATtiny4313	256 bytes	4 bytes	EEA[1:0]	64	EEA[7:2]	7

Note: 1. See Table 19-1 on page 174.

21.2 Parallel Programming

Parallel programming signals and connections are illustrated in Figure 21-1, below.

Figure 21-1. Parallel Programming



Signals are described in Table 21-3, below. Pins not listed in the table are referenced by pin names.

Table 21-3. Pin and Signal Names Used in Programming Mode

Signal Name	Pin(s)	I/O	Function
RDY/$\overline{\text{BSY}}$	PD1	O	0: Device is busy programming, 1: Device is ready for new command.
$\overline{\text{OE}}$	PD2	I	Output Enable (Active low).
$\overline{\text{WR}}$	PD3	I	Write Pulse (Active low).
BS1/PAGEL	PD4	I	Byte Select 1 (“0” selects low byte, “1” selects high byte). Program Memory and EEPROM Data Page Load.
XA0	PD5	I	XTAL Action Bit 0
XA1/BS2	PD6	I	XTAL Action Bit 1. Byte Select 2 (0: low byte, 1: 2nd high byte).
DATA I/O	PB7-0	I/O	Bi-directional Data bus (Output when $\overline{\text{OE}}$ is low).

Pulses are assumed to be at least 250 ns, unless otherwise noted.

Table 21-4. Pin Values Used to Enter Programming Mode

Pin	Symbol	Value
XA1	Prog_enable[3]	0
XA0	Prog_enable[2]	0
BS1	Prog_enable[1]	0
WR	Prog_enable[0]	0

Pins XA1 and XA0 determine the action when CLKI is given a positive pulse, as shown in Table 21-5.

Table 21-5. XA1 and XA0 Coding

XA1	XA0	Action when CLKI is Pulsed
0	0	Load Flash or EEPROM address (high or low address byte, determined by BS1)
0	1	Load data (high or low data byte for Flash, determined by BS1)
1	0	Load command
1	1	No action, idle

When pulsing $\overline{WR}$ or $\overline{OE}$, the command loaded determines the action executed. The different command options are shown in Table 21-6.

Table 21-6. Command Byte Bit Coding

Command Byte	Command
1000 0000	Chip Erase
0100 0000	Write fuse bits
0010 0000	Write lock bits
0001 0000	Write Flash
0001 0001	Write EEPROM
0000 1000	Read signature bytes and calibration byte
0000 0100	Read fuse and lock bits
0000 0010	Read Flash
0000 0011	Read EEPROM

21.2.1 Enter Programming Mode

The following algorithm puts the device in Parallel (High-voltage) Programming mode:

1. Set Prog_enable pins listed in Table 21-4 on page 185 to “0000”, RESET pin and V_{CC} to 0V.
2. Apply 4.5 - 5.5V between V_{CC} and GND.
3. Ensure that V_{CC} reaches at least 1.8V within the next 20 µs.
4. Wait 20 - 60 µs, and apply 11.5 - 12.5V to RESET.
5. Keep the Prog_enable pins unchanged for at least 10µs after the High-voltage has been applied to ensure the Prog_enable Signature has been latched.
6. Wait at least 300 µs before giving any parallel programming commands.
7. Exit Programming mode by power the device down or by bringing RESET pin to 0V.

If the rise time of the V_{CC} is unable to fulfill the requirements listed above, the following alternative algorithm can be used.

1. Set Prog_enable pins listed in Table 21-4 on page 185 to “0000”, RESET pin to 0V and V_{CC} to 0V.
2. Apply 4.5 - 5.5V between V_{CC} and GND.
3. Monitor V_{CC}, and as soon as V_{CC} reaches 0.9 - 1.1V, apply 11.5 - 12.5V to RESET.
4. Keep the Prog_enable pins unchanged for at least 10µs after the High-voltage has been applied to ensure the Prog_enable Signature has been latched.
5. Wait until V_{CC} actually reaches 4.5 -5.5V before giving any parallel programming commands.
6. Exit Programming mode by power the device down or by bringing RESET pin to 0V.

21.2.2 Considerations for Efficient Programming

Loaded commands and addresses are retained in the device during programming. For efficient programming, the following should be considered.

- When writing or reading multiple memory locations, the command needs only be loaded once
- Do not write the data value 0xFF, since this already is the contents of the entire Flash and EEPROM (unless the EESAVE Fuse is programmed) after a Chip Erase
- Address high byte needs only be loaded before programming or reading a new 256 word window in Flash or 256 byte EEPROM. This also applies to reading signature bytes

21.2.3 Chip Erase

A Chip Erase must be performed before the Flash and/or EEPROM are reprogrammed. The Chip Erase command will erase all Flash and EEPROM plus lock bits. If the EESAVE fuse is programmed, the EEPROM is not erased.

Lock bits are not reset until the program memory has been completely erased. Fuse bits are not changed.

The Chip Erase command is loaded as follows:

1. Set XA1, XA0 to “10”. This enables command loading.
2. Set BS1 to “0”.
3. Set DATA to “1000 0000”. This is the command for Chip Erase.
4. Give XTAL1 a positive pulse. This loads the command.
5. Give $\overline{WR}$ a negative pulse. This starts the Chip Erase. RDY/$\overline{BSY}$ goes low.
6. Wait until RDY/$\overline{BSY}$ goes high before loading a new command.

21.2.4 Programming the Flash

Flash is organized in pages, as shown in Table 21-1 on page 184. When programming the Flash, the program data is first latched into a page buffer. This allows one page of program data to be programmed simultaneously. The following procedure describes how to program the entire Flash memory:

A. Load Command “Write Flash”

1. Set XA1, XA0 to “10”. This enables command loading.
2. Set BS1 to “0”.
3. Set DATA to “0001 0000”. This is the command for Write Flash.
4. Give XTAL1 a positive pulse. This loads the command.

B. Load Address Low byte

1. Set XA1, XA0 to “00”. This enables address loading.
2. Set BS1 to “0”. This selects low address.
3. Set DATA = Address low byte (0x00 - 0xFF).
4. Give XTAL1 a positive pulse. This loads the address low byte.

C. Load Data Low Byte

1. Set XA1, XA0 to “01”. 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 “01”. This enables data loading.
3. Set DATA = Data high byte (0x00 - 0xFF).
4. Give XTAL1 a positive pulse. This loads the data byte.

E. 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 on page 189. 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.

F. Load Address High byte

1. Set XA1, XA0 to “00”. 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.

G. 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 189 for signal waveforms).

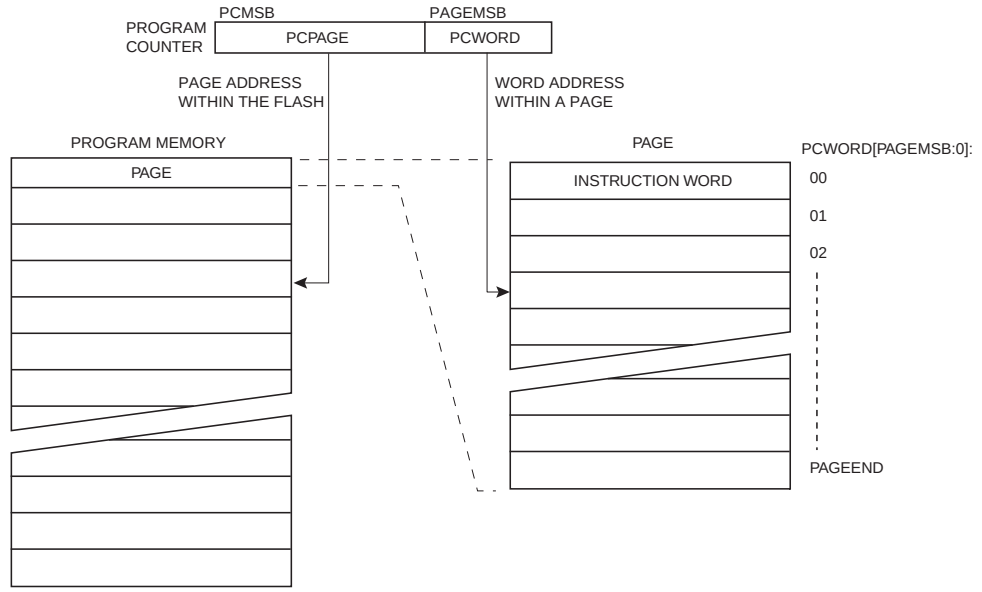
H. Repeat B through H until the entire Flash is programmed or until all data has been programmed.

I. End Page Programming

1. 1. Set XA1, XA0 to “10”. This enables command loading.
2. Set DATA to “0000 0000”. This is the command for No Operation.
3. Give XTAL1 a positive pulse. This loads the command, and the internal write signals are reset.

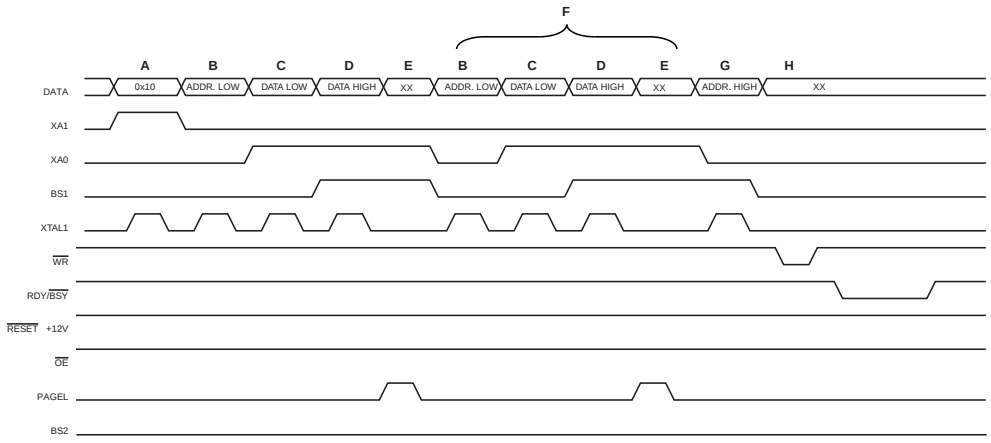
Flash page addressing is illustrated in Figure 21-2, below. Symbols used are described in Table 19-1 on page 174.

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



Flash programming waveforms are illustrated in Figure 21-3, where XX means “don’t care” and letters refer to the programming steps described earlier.

Figure 21-3. Programming the Flash Waveforms



21.2.5 Programming the EEPROM

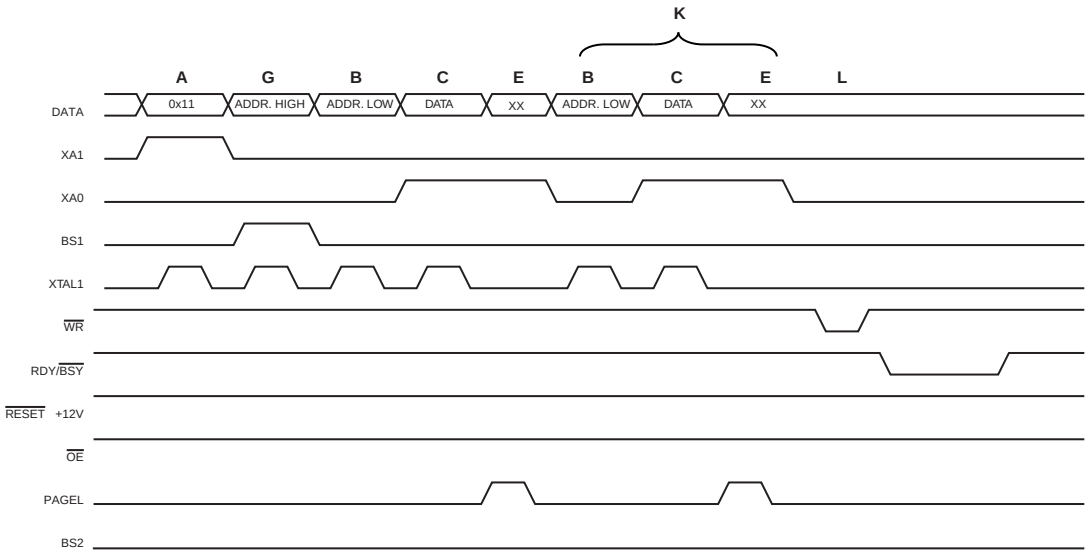
The EEPROM is organized in pages, see Table 21-2 on page 184. When programming the EEPROM, the program data is latched into a page buffer. This allows one page of data to be programmed simultaneously. The programming algorithm for the EEPROM data memory is as follows (see “Programming the Flash” on page 187 for details on loading command, address and data):

- A: Load Command “0001 0001”.
- G: Load Address High Byte (0x00 - 0xFF).
- B: Load Address Low Byte (0x00 - 0xFF).

- C: Load Data (0x00 - 0xFF).
- J: Repeat 3 through 4 until the entire buffer is filled.
- K: Program EEPROM page
  - Set BS1 to “0”.
  - Give $\overline{WR}$ a negative pulse. This starts programming of the EEPROM page. RDY/$\overline{BSY}$ goes low.
  - Wait until to RDY/$\overline{BSY}$ goes high before programming the next page (See Figure 21-4 for signal waveforms).

EEPROM programming waveforms are illustrated in Figure 21-4, where XX means “don’t care” and letters refer to the programming steps described above.

Figure 21-4. EEPROM Programming Waveforms



21.2.6 Reading the Flash

The algorithm for reading the Flash memory is as follows (see “Programming the Flash” on page 187 for details on command and address loading):

- A: Load Command “0000 0010”.
- G: Load Address High Byte (0x00 - 0xFF).
- B: Load Address Low Byte (0x00 - 0xFF).
- Set $\overline{OE}$ to “0”, and BS1 to “0”. The Flash word low byte can now be read at DATA.
- Set BS1 to “1”. The Flash word high byte can now be read at DATA.
- Set $\overline{OE}$ to “1”.

21.2.7 Reading the EEPROM

The algorithm for reading the EEPROM memory is as follows (see “Programming the Flash” on page 187 for details on command and address loading):

- A: Load Command “0000 0011”.
- G: Load Address High Byte (0x00 - 0xFF).

- B: Load Address Low Byte (0x00 - 0xFF).
- Set $\overline{OE}$ to “0”, and BS1 to “0”. The EEPROM Data byte can now be read at DATA.
- Set $\overline{OE}$ to “1”.

21.2.8 Programming Low Fuse Bits

The algorithm for programming the low fuse bits is as follows (see “Programming the Flash” on page 187 for details on command and data loading):

- A: Load Command “0100 0000”.
- C: Load Data Low Byte. Bit n = “0” programs and bit n = “1” erases the Fuse bit.
- Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.

21.2.9 Programming High Fuse Bits

The algorithm for programming the high fuse bits is as follows (see “Programming the Flash” on page 187 for details on command and data loading):

- A: Load Command “0100 0000”.
- C: Load Data Low Byte. Bit n = “0” programs and bit n = “1” erases the Fuse bit.
- Set BS1 to “1” and BS2 to “0”. This selects high data byte.
- Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.
- Set BS1 to “0”. This selects low data byte.

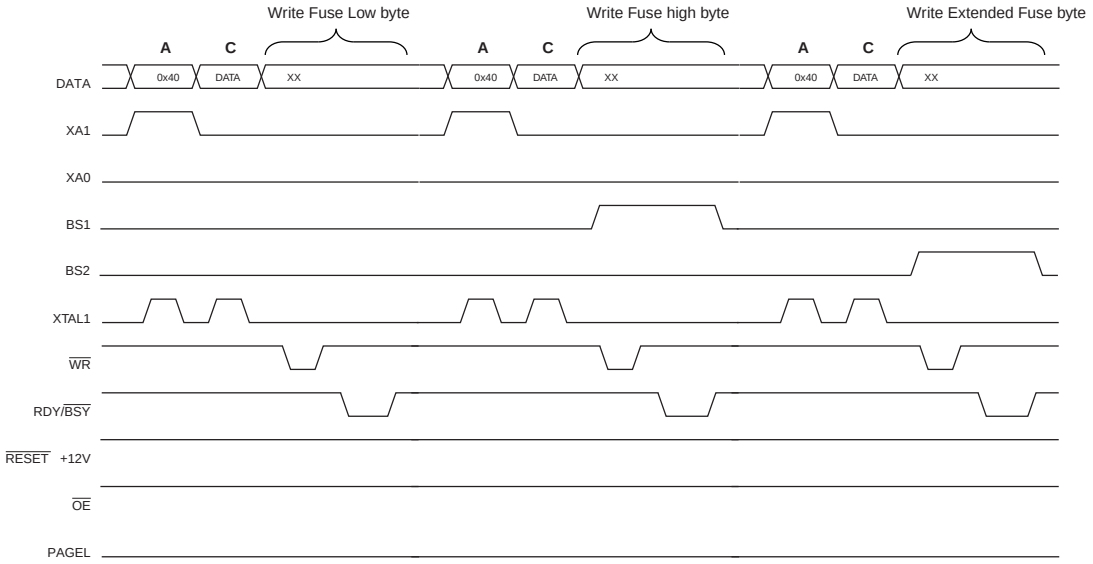
21.2.10 Programming Extended Fuse Bits

The algorithm for programming the extended fuse bits is as follows (see “Programming the Flash” on page 187 for details on command and data loading):

- A: Load Command “0100 0000”.
- C: Load Data Low Byte. Bit n = “0” programs and bit n = “1” erases the Fuse bit.
- Set BS1 to “0” and BS2 to “1”. This selects extended data byte.
- Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.
- Set BS2 to “0”. This selects low data byte.

Fuse programming waveforms are illustrated in Figure 21-5, where XX means “don’t care” and letters refer to the programming steps described above.

Figure 21-5. Fuses Programming Waveforms



21.2.11 Programming the Lock Bits

The algorithm for programming the lock bits is as follows (see “Programming the Flash” on page 187 for details on command and data loading):

1. A: Load Command “0010 0000”.
2. C: Load Data Low Byte. Bit n = “0” programs the Lock bit. If LB mode 3 is programmed (LB1 and LB2 is programmed), it is not possible to program the Boot Lock bits by any External Programming mode.
3. Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.

The Lock bits can only be cleared by executing Chip Erase.

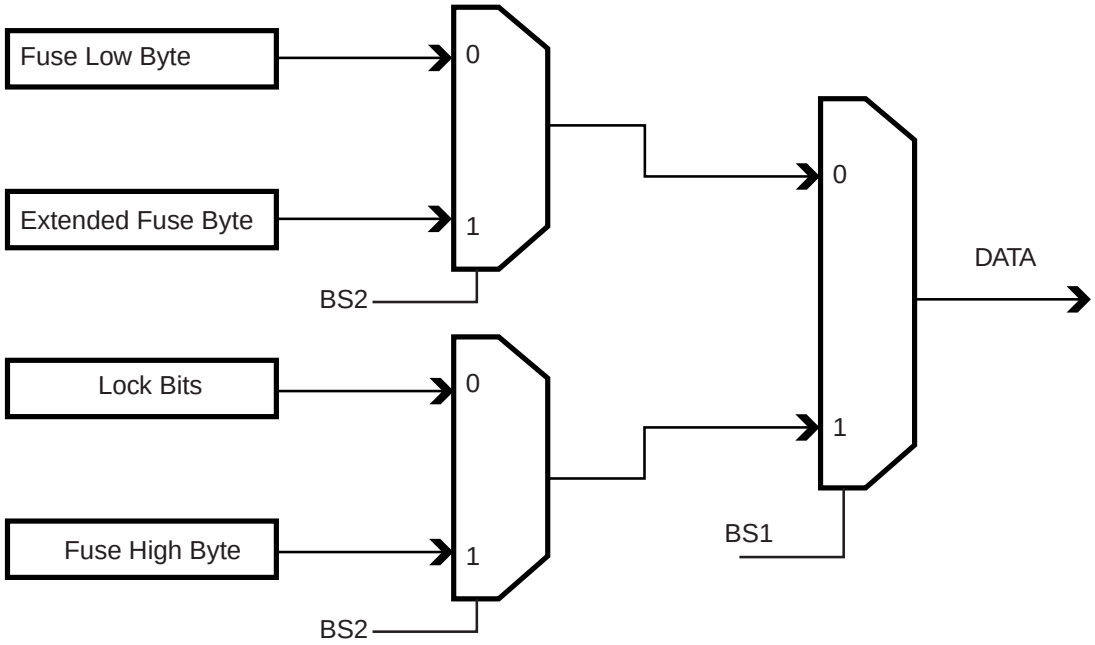
21.2.12 Reading Fuse and Lock Bits

The algorithm for reading fuse and lock bits is as follows (see “Programming the Flash” on page 187 for details on command loading):

- A: Load Command “0000 0100”.
- Set $\overline{OE}$ to “0”, BS2 to “0” and BS1 to “0”. The status of the Fuse Low bits can now be read at DATA (“0” means programmed).
- Set $\overline{OE}$ to “0”, BS2 to “1” and BS1 to “1”. The status of the Fuse High bits can now be read at DATA (“0” means programmed).
- Set OE to “0”, BS2 to “1”, and BS1 to “0”. The status of the Extended Fuse bits can now be read at DATA (“0” means programmed).
- Set $\overline{OE}$ to “0”, BS2 to “0” and BS1 to “1”. The status of the Lock bits can now be read at DATA (“0” means programmed).
- Set $\overline{OE}$ to “1”.

Fuse and lock bit mapping is illustrated in Figure 21-6, below.

Figure 21-6. Mapping Between BS1, BS2 and the Fuse and Lock Bits During Read



21.2.13 Reading Signature Bytes

The algorithm for reading the signature bytes is as follows (see “Programming the Flash” on page 187 for details on command and address loading):

1. A: Load Command “0000 1000”.
2. B: Load Address Low Byte (0x00 - 0x02).
3. Set $\overline{OE}$ to “0”, and BS1 to “0”. The selected Signature byte can now be read at DATA.
4. Set $\overline{OE}$ to “1”.

21.2.14 Reading the Calibration Byte

The algorithm for reading the calibration byte is as follows (see “Programming the Flash” on page 187 for details on command and address loading):

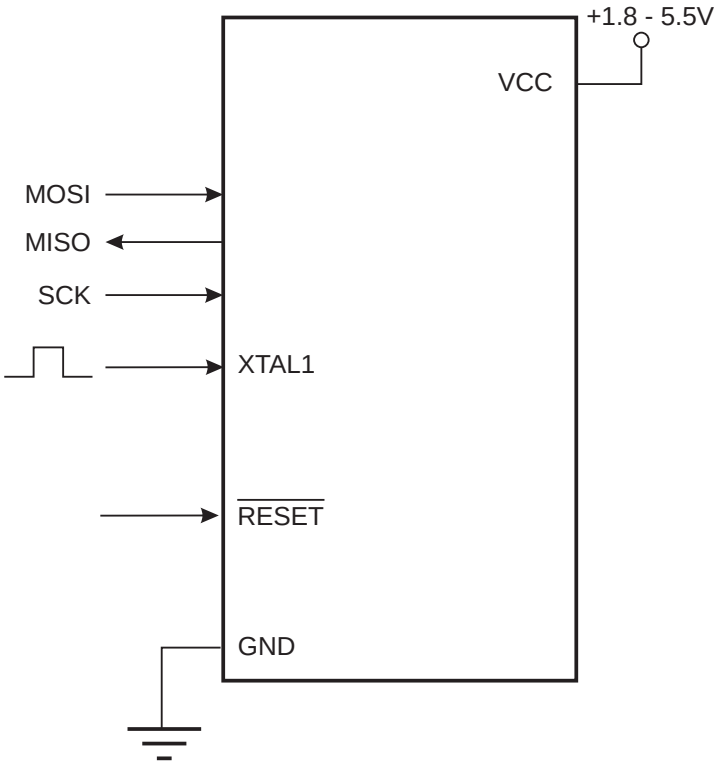
1. A: Load Command “0000 1000”.
2. B: Load Address Low Byte, 0x00.
3. Set $\overline{OE}$ to “0”, and BS1 to “1”. The Calibration byte can now be read at DATA.
4. Set $\overline{OE}$ to “1”.

21.3 Serial Programming

Flash and EEPROM memory arrays can both be programmed using the serial SPI bus while $\overline{RESET}$ is pulled to GND. The serial interface consists of pins SCK, MOSI (input) and MISO (output). After $\overline{RESET}$ is set low, the Programming Enable instruction needs to be executed before program/erase operations can be executed.

Serial programming signals and connections are illustrated in Figure 21-7, below. The pin mapping is listed in Table 21-7 on page 195.

Figure 21-7. Serial Programming Signals



Note: If the device is clocked by the internal oscillator there is no need to connect a clock source to the CLKI pin.

When programming the EEPROM, an auto-erase cycle is built into the self-timed programming operation and there is no need to first execute the Chip Erase instruction. This applies for serial programming mode, only.

The Chip Erase operation turns the content of every memory location in Flash and EEPROM arrays into 0xFF.

Depending on CKSEL fuses, a valid clock must be present. The minimum low and high periods for the serial clock (SCK) input are defined as follows:

- Minimum low period of serial clock:
  - When f_{ck} < 12MHz: > 2 CPU clock cycles
  - When f_{ck} >= 12MHz: 3 CPU clock cycles
- Minimum high period of serial clock:
  - When f_{ck} < 12MHz: > 2 CPU clock cycles
  - When f_{ck} >= 12MHz: 3 CPU clock cycles

21.3.1 Pin Mapping

The pin mapping is listed in Table 21-7. Note that not all parts use the SPI pins dedicated for the internal SPI interface.

Table 21-7. Pin Mapping Serial Programming

Symbol	Pins	I/O	Description
MOSI	PB5	I	Serial Data in
MISO	PB6	O	Serial Data out
SCK	PB7	I	Serial Clock

21.3.2 Programming Algorithm

When writing serial data to the ATtiny2313A/4313, data is clocked on the rising edge of SCK. When reading data from the ATtiny2313A/4313, data is clocked on the falling edge of SCK. See Figure 22-6 on page 205 and Figure 22-7 on page 205 for timing details.

To program and verify the ATtiny2313A/4313 in the serial programming mode, the following sequence is recommended (See Table 21-8, “Serial Programming Instruction Set,” on page 196):

1. Power-up sequence: apply power between V_{CC} and GND while $\overline{\text{RESET}}$ and SCK are set to “0”
   – In some systems, the programmer can not guarantee that SCK is held low during power-up. In this case, $\overline{\text{RESET}}$ must be given a positive pulse after SCK has been set to '0'. The duration of the pulse must be at least t_{RST} plus two CPU clock cycles. See Table 22-3 on page 201 for definition of minimum pulse width on $\overline{\text{RESET}}$ pin, t_{RST}
2. Wait for at least 20 ms and then enable serial programming by sending the Programming Enable serial instruction to the MOSI pin
3. The serial programming instructions will not work if the communication is out of synchronization. When in sync, the second byte (0x53) will echo back when issuing the third byte of the Programming Enable instruction
   – Regardless if the echo is correct or not, all four bytes of the instruction must be transmitted
   – If the 0x53 did not echo back, give $\overline{\text{RESET}}$ a positive pulse and issue a new Programming Enable command
4. The Flash is programmed one page at a time. The memory page is loaded one byte at a time by supplying the 4 LSB of the address and data together with the Load Program Memory Page instruction.
   – To ensure correct loading of the page, the data low byte must be loaded before data high byte is applied for a given address
   – The Program Memory Page is stored by loading the Write Program Memory Page instruction with the 6 MSB of the address
   – If polling (RDY/$\overline{\text{BSY}}$) is not used, the user must wait at least t_{WD_FLASH} before issuing the next page. (See Table 21-9 on page 197). Accessing the serial programming interface before the Flash write operation completes can result in incorrect programming.
5. The EEPROM can be programmed one byte or one page at a time.

– **A**: Byte programming. The EEPROM array is programmed one byte at a time by supplying the address and data together with the Write instruction. EEPROM memory locations are automatically erased before new data is written. If polling (RDY/$\overline{BSY}$) is not used, the user must wait at least t_{WD_EEPROM} before issuing the next byte (See Table 21-9). In a chip erased device, no 0xFFs in the data file(s) need to be programmed

– **B**: Page programming (the EEPROM array is programmed one page at a time). The memory page is loaded one byte at a time by supplying the 6 LSB of the address and data together with the Load EEPROM Memory Page instruction. The EEPROM memory page is stored by loading the Write EEPROM Memory Page Instruction with the 7 MSB of the address. When using EEPROM page access only byte locations loaded with the Load EEPROM Memory Page instruction are altered and the remaining locations remain unchanged. If polling (RDY/$\overline{BSY}$) is not used, the user must wait at least t_{WD_EEPROM} before issuing the next byte (See Table 21-9). In a chip erased device, no 0xFF in the data file(s) need to be programmed

6. Any memory location can be verified by using the Read instruction, which returns the content at the selected address at the serial output pin (MISO)
7. At the end of the programming session, $\overline{RESET}$ can be set high to commence normal operation
8. Power-off sequence (if required): set $\overline{RESET}$ to “1”, and turn V_{CC} power off

21.3.3 Programming Instruction Set

The instruction set for serial programming is described in Table 21-8.

Table 21-8. Serial Programming Instruction Set

Instruction	Instruction Format				Operation
	Byte 1	Byte 2	Byte 3	Byte4	
Programming Enable	1010 1100	0101 0011	xxxx xxxx	xxxx xxxx	Enable Serial Programming after $\overline{RESET}$ goes low.
Chip Erase	1010 1100	100x xxxx	xxxx xxxx	xxxx xxxx	Chip Erase EEPROM and Flash.
Read Program Memory	0010 **H**000	0000 00**aa**	**bbbb bbbb**	**oooo oooo**	Read **H** (high or low) data **o** from Program memory at word address **a**:**b**.
Load Program Memory Page	0100 **H**000	000x xxxx	xxxx **bbbb**	**iiii iiii**	Write **H** (high or low) data **i** to Program Memory page at word address **b**. Data low byte must be loaded before Data high byte is applied within the same address.
Write Program Memory Page	0100 1100	0000 00**aa**	**bbbb** xxxx	xxxx xxxx	Write Program Memory Page at address **a**:**b**.
Read EEPROM Memory	1010 0000	000x xxxx	x**bbb bbbb**	**oooo oooo**	Read data **o** from EEPROM memory at address **b**.
Write EEPROM Memory	1100 0000	000x xxxx	x**bbb bbbb**	**iiii iiii**	Write data **i** to EEPROM memory at address **b**.
Load EEPROM Memory Page (page access)	1100 0001	0000 0000	0000 00**bb**	**iiii iiii**	Load data **i** to EEPROM memory page buffer. After data is loaded, program EEPROM page.

Table 21-8. Serial Programming Instruction Set

Instruction	Instruction Format				Operation
	Byte 1	Byte 2	Byte 3	Byte4	
Write EEPROM Memory Page (page access)	1100 0010	00xx xxxx	x**bbb bb**00	xxxx xxxx	Write EEPROM page at address **b**.
Read Lock bits	0101 1000	0000 0000	xxxx xxxx	xx**oo oooo**	Read Lock bits. “0” = programmed, “1” = unprogrammed. See Table 20-1 on page 178 for details.
Write Lock bits	1010 1100	111x xxxx	xxxx xxxx	11**ii iiii**	Write Lock bits. Set bits = “0” to program Lock bits. See Table 20-1 on page 178 for details.
Read Signature Byte	0011 0000	000x xxxx	xxxx xx**bb**	**oooo oooo**	Read Signature Byte **o** at address **b**.
Write Fuse bits	1010 1100	1010 0000	xxxx xxxx	**iiii iiii**	Set bits = “0” to program, “1” to unprogram.
Write Fuse High bits	1010 1100	1010 1000	xxxx xxxx	**iiii iiii**	Set bits = “0” to program, “1” to unprogram.
Write Extended Fuse Bits	1010 1100	1010 0100	xxxx xxxx	xxxx xxx**i**	Set bits = “0” to program, “1” to unprogram.
Read Fuse bits	0101 0000	0000 0000	xxxx xxxx	**oooo oooo**	Read Fuse bits. “0” = programmed, “1” = unprogrammed.
Read Fuse High bits	0101 1000	0000 1000	xxxx xxxx	**oooo oooo**	Read Fuse High bits. “0” = programmed, “1” = unprogrammed.
Read Extended Fuse Bits	0101 0000	0000 1000	xxxx xxxx	**oooo oooo**	Read Extended Fuse bits. “0” = programmed, “1” = unprogrammed.
Read Calibration Byte	0011 1000	000x xxxx	0000 000**b**	**oooo oooo**	Read Calibration Byte at address **b**.
Poll RDY/$\overline{\text{BSY}}$	1111 0000	0000 0000	xxxx xxxx	xxxx xxx**o**	If **o** = “1”, a programming operation is still busy. Wait until this bit returns to “0” before applying another command.

Note: **a** = address high bits, **b** = address low bits, **H** = 0 - Low byte, 1 - High Byte, **o** = data out, **i** = data in, x = don’t care

If the LSB of RDY/$\overline{\text{BSY}}$ data byte out is ‘1’, a programming operation is still pending. Wait until this bit returns ‘0’ before the next instruction is carried out.

Within the same page, the low data byte must be loaded prior to the high data byte.

After data is loaded to the page buffer, program the EEPROM page.

21.4 Programming Time for Flash and EEPROM

Flash and EEPROM wait times are listed in Table 21-9.

Table 21-9. Minimum Wait Delay Before Writing the Next Flash or EEPROM Location

Symbol	Minimum Wait Delay
t_{WD_FLASH}	4.5 ms
t_{WD_EEPROM}	4.0 ms
t_{WD_ERASE}	9.0 ms
t_{WD_FUSE}	4.5 ms

22. Electrical Characteristics

22.1 Absolute Maximum Ratings*

Operating Temperature	-55°C to +125°C
Storage Temperature	-65°C to +150°C
Voltage on any Pin except $\overline{\text{RESET}}$ with respect to Ground	-0.5V to V_{CC}+0.5V
Voltage on $\overline{\text{RESET}}$ with respect to Ground	-0.5V to +13.0V
Maximum Operating Voltage	6.0V
DC Current per I/O Pin	40.0 mA
DC Current V_{CC} and GND Pins	200.0 mA

*NOTICE: Stresses beyond those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

22.2 DC Characteristics

T_A = -40°C to 85°C, V_{CC} = 1.8V to 5.5V (unless otherwise noted)

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
V_{IL}	Input Low Voltage except XTAL1 and $\overline{\text{RESET}}$ pin	V_{CC} = 1.8V - 2.4V V_{CC} = 2.4V - 5.5V	-0.5		$0.2V_{CC}$ $0.3V_{CC}$	V
V_{IH}	Input High-voltage except XTAL1 and $\overline{\text{RESET}}$ pins	V_{CC} = 1.8V - 2.4V V_{CC} = 2.4V - 5.5V	$0.7V_{CC}$ (1) $0.6V_{CC}$ (1)		V_{CC} +0.5 (2)	V
V_{IL1}	Input Low Voltage XTAL1 pin	V_{CC} = 1.8V - 5.5V	-0.5		$0.1V_{CC}$	V
V_{IH1}	Input High-voltage XTAL1 pin	V_{CC} = 1.8V - 2.4V V_{CC} = 2.4V - 5.5V	$0.8V_{CC}$ (1) $0.7V_{CC}$ (1)		V_{CC} +0.5 (2)	V
V_{IL2}	Input Low Voltage $\overline{\text{RESET}}$ pin	V_{CC} = 1.8V - 5.5V	-0.5		$0.2V_{CC}$	V
V_{IH2}	Input High-voltage $\overline{\text{RESET}}$ pin	V_{CC} = 1.8V - 5.5V	$0.9V_{CC}$ (1)		V_{CC} +0.5 (2)	V
V_{IL3}	Input Low Voltage $\overline{\text{RESET}}$ pin as I/O	V_{CC} = 1.8V - 2.4V V_{CC} = 2.4V - 5.5V	-0.5		$0.2V_{CC}$ $0.3V_{CC}$	V
V_{IH3}	Input High-voltage $\overline{\text{RESET}}$ pin as I/O	V_{CC} = 1.8V - 2.4V V_{CC} = 2.4V - 5.5V	$0.7V_{CC}$ (1) $0.6V_{CC}$ (1)		V_{CC} +0.5 (2)	V
V_{OL}	Output Low Voltage(3) (Except Reset Pin)(5)	I_{OL} = 20 mA, V_{CC} = 5V I_{OL} = 10mA, V_{CC} = 3V			0.8 0.6	V V
V_{OH}	Output High-voltage(4) (Except Reset Pin)(5)	I_{OH} = -20 mA, V_{CC} = 5V I_{OH} = -10 mA, V_{CC} = 3V	4.2 2.4			V V
I_{IL}	Input Leakage Current I/O Pin	V_{CC} = 5.5V, pin low (absolute value)			1(6)	µA
I_{IH}	Input Leakage Current I/O Pin	V_{CC} = 5.5V, pin high (absolute value)			1(6)	µA
R_{RST}	Reset Pull-up Resistor		30		60	kΩ
R_{pu}	I/O Pin Pull-up Resistor		20		50	kΩ

T_A = -40°C to 85°C, V_{CC} = 1.8V to 5.5V (unless otherwise noted) (Continued)

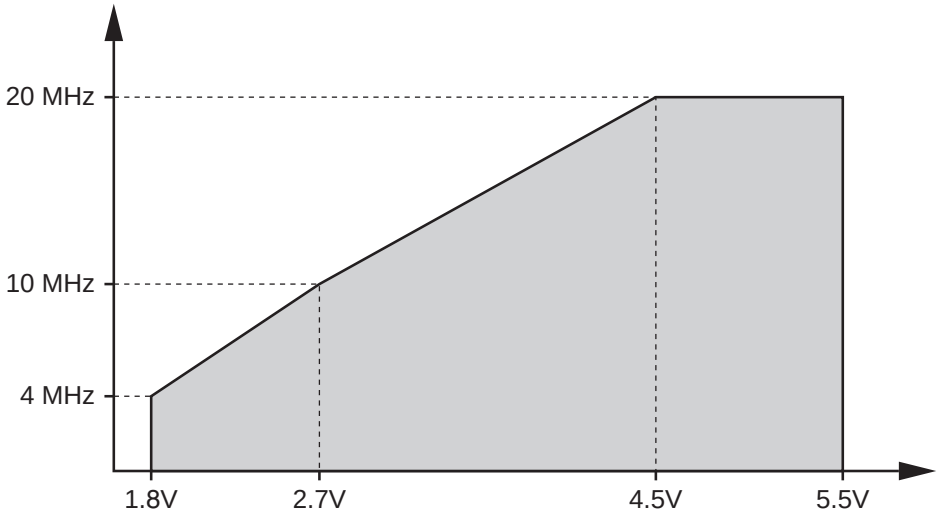
Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
I_{CC}	Power Supply Current	Active 1MHz, V_{CC} = 2V(7)		0.2	0.55	mA
		Active 4MHz, V_{CC} = 3V(7)		1.3	2.5	mA
		Active 8MHz, V_{CC} = 5V(7)		3.9	7	mA
		Idle 1MHz, V_{CC} = 2V(7)		0.03	0.15	mA
		Idle 4MHz, V_{CC} = 3V(7)		0.25	0.6	mA
		Idle 8MHz, V_{CC} = 5V(7)		1	2	mA
	Power-down mode	WDT enabled, V_{CC} = 3V(8)		4	10	µA
		WDT disabled, V_{CC} = 3V(8)		< 0.15	2	µA

Notes:
1. “Min” means the lowest value where the pin is guaranteed to be read as high.
2. “Max” means the highest value where the pin is guaranteed to be read as low.
3. Although each I/O port can sink more than the test conditions (20 mA at V_{CC} = 5V, 10 mA at V_{CC} = 3V) under steady state conditions (non-transient), the following must be observed:
   1] The sum of all IOL, for all ports, should not exceed 60 mA.
   If IOL exceeds the test condition, VOL may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test condition.
4. Although each I/O port can source more than the test conditions (20 mA at V_{CC} = 5V, 10 mA at V_{CC} = 3V) under steady state conditions (non-transient), the following must be observed:
   1] The sum of all IOH, for all ports, should not exceed 60 mA.
   If IOH exceeds the test condition, VOH may exceed the related specification. Pins are not guaranteed to source current greater than the listed test condition.
5. The $\overline{\text{RESET}}$ pin must tolerate high voltages when entering and operating in programming modes and, as a consequence, has a weak drive strength as compared to regular I/O pins. See Figure 23-29 and Figure 23-30.
6. These are test limits, which account for leakage currents of the test environment. Actual device leakage currents are lower.
7. Values using methods described in “Minimizing Power Consumption” on page 36. Power Reduction is enabled (PRR = 0xFF), the external clock is selected (CKSEL = 0000), and there is no I/O drive.
8. BOD Disabled.

22.3 Speed

The maximum operating frequency of the device is dependent on supply voltage, V_{CC}. The relationship between supply voltage and maximum operating frequency is piecewise linear, as shown in Figure 22-1.

Figure 22-1. Maximum Frequency vs. V_{CC}



22.4 Clock Characteristics

22.4.1 Calibrated Internal RC Oscillator Accuracy

It is possible to manually calibrate the internal oscillator to be more accurate than default factory calibration. Note that the oscillator frequency depends on temperature and voltage. Voltage and temperature characteristics can be found in Figure 23-46 on page 229, and Figure 23-47 on page 230.

Table 22-1. Calibration Accuracy of Internal RC Oscillator

Calibration Method	Target Frequency	V_{CC}	Temperature	Accuracy at given Voltage & Temperature[1]
Factory Calibration	4.0 / 8.0MHz	3V	25°C	±10%
User Calibration	Fixed frequency within: 3.1 – 4.7 MHz / 7.3 – 9.1MHz	Fixed voltage within: 1.8V – 5.5V	Fixed temperature within: -40°C – 85°C	±2%

Notes: 1. Accuracy of oscillator frequency at calibration point (fixed temperature and fixed voltage).

22.4.2 External Clock Drive

Figure 22-2. External Clock Drive Waveform

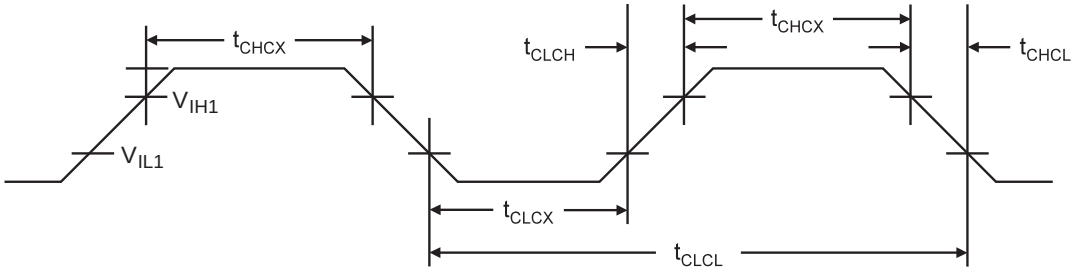


Table 22-2. External Clock Drive

Symbol	Parameter	V_{CC} = 1.8 - 5.5V		V_{CC} = 2.7 - 5.5V		V_{CC} = 4.5 - 5.5V		Units
		Min.	Max.	Min.	Max.	Min.	Max.	
$1/t_{CLCL}$	Clock Frequency	0	4	0	10	0	20	MHz
t_{CLCL}	Clock Period	250		100		50		ns
t_{CHCX}	High Time	100		40		20		ns
t_{CLCX}	Low Time	100		40		20		ns
t_{CLCH}	Rise Time		2.0		1.6		0.5	µs
t_{CHCL}	Fall Time		2.0		1.6		0.5	µs
Δt_{CLCL}	Change in period from one clock cycle to the next		2		2		2	%

22.5 System and Reset Characteristics

Table 22-3. Reset, Brown-out, and Internal Voltage Characteristics

Symbol	Parameter	Condition	Min	Typ	Max	Units
V_{RST}	$\overline{\text{RESET}}$ Pin Threshold Voltage		0.2 V_{CC}		0.8V_{CC}	V
t_{RST}	Minimum pulse width on $\overline{\text{RESET}}$ Pin (1)(2)	V_{CC} = 1.8 - 5.5V			2.5	µs
V_{HYST}	Brown-out Detector Hysteresis (2)			50		mV
t_{BOD}	Min Pulse Width on Brown-out Reset (2)			2		µs
V_{BG}	Internal bandgap reference voltage	V_{CC} = 2.7V T_A = 25°C	1.0	1.1	1.2	V
t_{BG}	Internal bandgap reference start-up time (2)	V_{CC} = 2.7V T_A = 25°C		40	70	µs
I_{BG}	Internal bandgap reference current consumption (2)	V_{CC} = 2.7V T_A = 25°C		15		µA

Notes: 1. When $\overline{\text{RESET}}$ pin used as reset (not as I/O).
2. Not tested in production.

22.5.1 Enhanced Power-On Reset

Table 22-4. Characteristics of Enhanced Power-On Reset. T_A = -40 – 85°C

Symbol	Parameter	Min(1)	Typ(1)	Max(1)	Units
V_{POR}	Release threshold of power-on reset (2)	1.1	1.4	1.6	V
V_{POA}	Activation threshold of power-on reset (3)	0.6	1.3	1.6	V
SR_{ON}	Power-On Slope Rate	0.01			V/ms

Notes: 1. Values are guidelines, only.
2. Threshold where device is released from reset when voltage is rising.
3. The Power-on Reset will not work unless the supply voltage has been below V_{POA}.

22.5.2 Brown-Out Detection

Table 22-5. V_{BOT} vs. BODLEVEL Fuse Coding

BODLEVEL [1:0] Fuses	Min(1)	Typ(1)	Max(1)	Units
11	BOD Disabled			
10	1.7	1.8	2.0	V
01	2.5	2.7	2.9	
00	4.1	4.3	4.5	

Note: 1. V_{BOT} may be below nominal minimum operating voltage for some devices. For devices where this is the case, the device is tested down to $V_{CC} = V_{BOT}$ during the production test. This guarantees that a Brown-out Reset will occur before V_{CC} drops to a voltage where correct operation of the microcontroller is no longer guaranteed.

22.6 Analog Comparator Characteristics

Table 22-6. Analog Comparator Characteristics, T_A = -40°C - 85°C

Symbol	Parameter	Condition	Min	Typ	Max	Units
V_{ACIO}	Input Offset Voltage	V_{CC} = 5V, V_{IN} = V_{CC} / 2		< 10	40	mV
I_{ACLK}	Input Leakage Current	V_{CC} = 5V, V_{IN} = V_{CC} / 2	-50		50	nA
t_{ACPD}	Analog Propagation Delay (from saturation to slight overdrive)	V_{CC} = 2.7V		750		ns
		V_{CC} = 4.0V		500		
	Analog Propagation Delay (large step change)	V_{CC} = 2.7V		100		
		V_{CC} = 4.0V		75		
t_{DPD}	Digital Propagation Delay	V_{CC} = 1.8V - 5.5		1	2	CLK

Note: All parameters are based on simulation results and they are not tested in production

22.7 Parallel Programming Characteristics

Table 22-7. Parallel Programming Characteristics, V_{CC} = 5V ± 10%

Symbol	Parameter	Min	Typ	Max	Units
V_{PP}	Programming Enable Voltage	11.5		12.5	V
I_{PP}	Programming Enable Current			250	µA
t_{DVXH}	Data and Control Valid before XTAL1 High	67			ns
t_{XLXH}	XTAL1 Low to XTAL1 High	200			ns
t_{XHXL}	XTAL1 Pulse Width High	150			ns
t_{XLDX}	Data and Control Hold after XTAL1 Low	67			ns
t_{XLWL}	XTAL1 Low to $\overline{WR}$ Low	0			ns
t_{XLPH}	XTAL1 Low to PAGEL high	0			ns
t_{PLXH}	PAGEL low to XTAL1 high	150			ns
t_{BVPH}	BS1 Valid before PAGEL High	67			ns
t_{PHPL}	PAGEL Pulse Width High	150			ns
t_{PLBX}	BS1 Hold after PAGEL Low	67			ns
t_{WLBX}	BS2/1 Hold after $\overline{WR}$ Low	67			ns
t_{PLWL}	PAGEL Low to $\overline{WR}$ Low	67			ns
t_{BVWL}	BS1 Valid to $\overline{WR}$ Low	67			ns
t_{WLWH}	$\overline{WR}$ Pulse Width Low	150			ns
t_{WLRL}	$\overline{WR}$ Low to RDY/$\overline{BSY}$ Low	0		1	µs
t_{WLRH}	$\overline{WR}$ Low to RDY/$\overline{BSY}$ High[1]	3.7		4.5	ms
t_{WLRH_CE}	$\overline{WR}$ Low to RDY/$\overline{BSY}$ High for Chip Erase[2]	7.5		9	ms
t_{XLOL}	XTAL1 Low to $\overline{OE}$ Low	0			ns
t_{BVDV}	BS1 Valid to DATA valid	0		1000	ns
t_{OLDV}	$\overline{OE}$ Low to DATA Valid			1000	ns
t_{OHDZ}	$\overline{OE}$ High to DATA Tri-stated			1000	ns

Notes: 1. t_{WLRH} is valid for the Write Flash, Write EEPROM, Write Fuse bits and Write Lock bits commands.

2. t_{WLRH_CE} is valid for the Chip Erase command.

Figure 22-3. Parallel Programming Timing, Including some General Timing Requirements

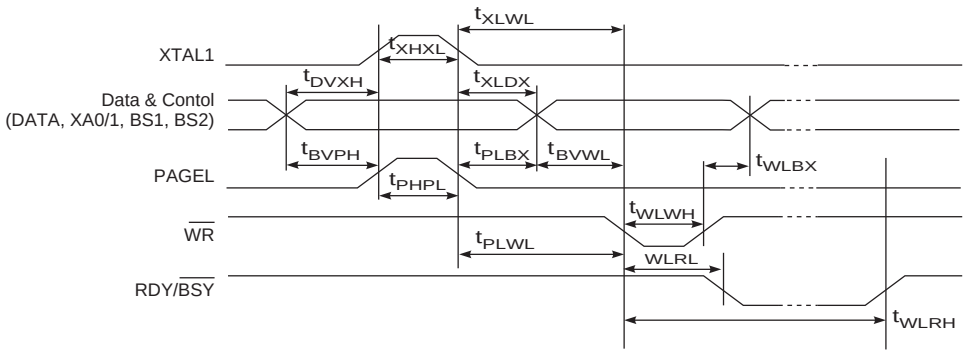
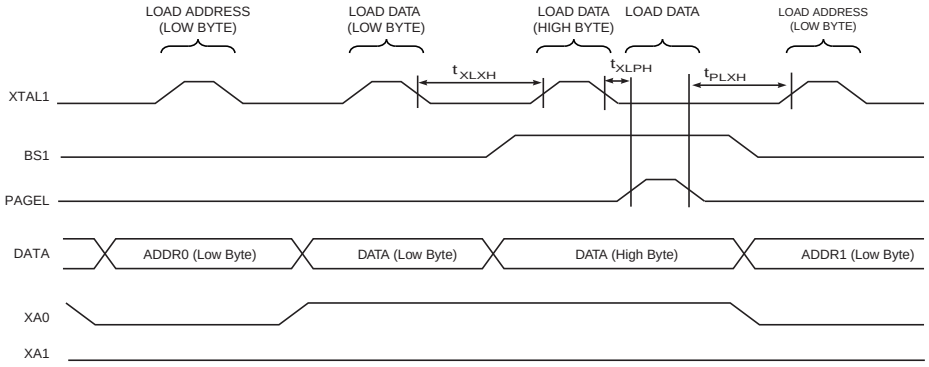
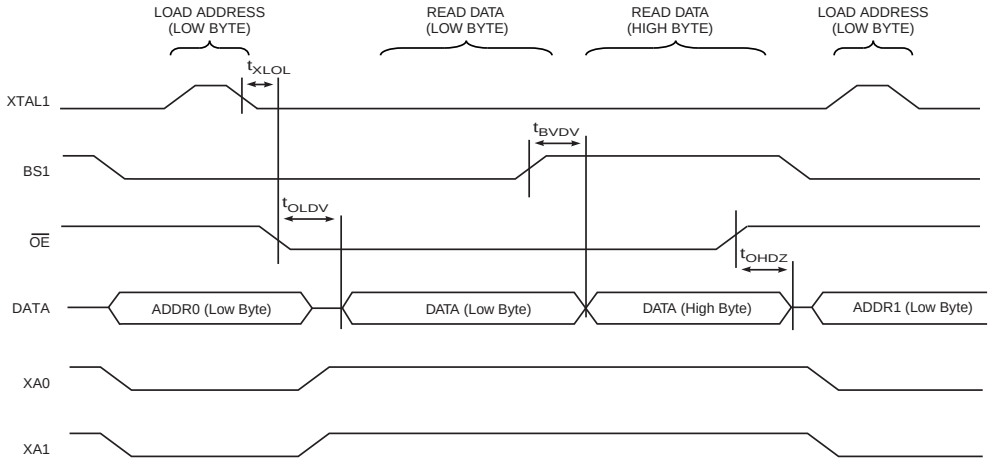


Figure 22-4. Parallel Programming Timing, Loading Sequence with Timing Requirements[1]



Note: 1. The timing requirements shown in Figure 22-3 (i.e., t_{DVXH}, t_{XHXL}, and t_{XLDX}) also apply to loading operation.

Figure 22-5. Parallel Programming Timing, Reading Sequence (within the Same Page) with Timing Requirements[1]



Note: 1. The timing requirements shown in Figure 22-3 (i.e., t_{DVXH}, t_{XHXL}, and t_{XLDX}) also apply to reading operation.

22.8 Serial Programming Characteristics

Figure 22-6. Serial Programming Timing

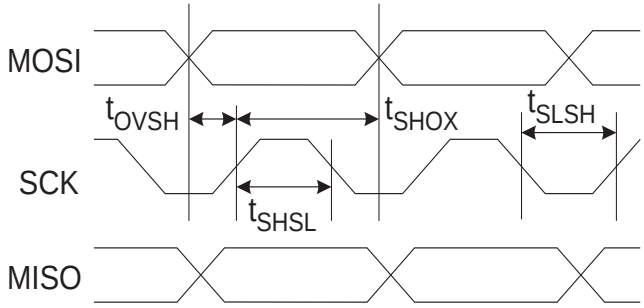


Figure 22-7. Serial Programming Waveform

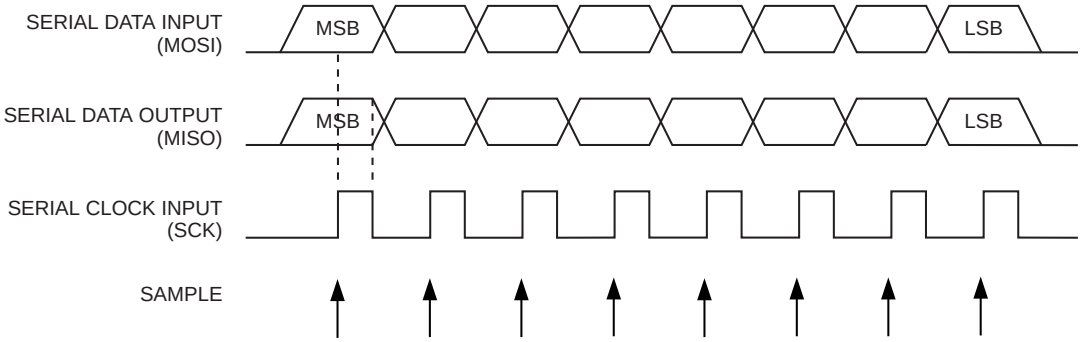


Table 22-8. Serial Programming Characteristics, T_A = -40°C to 85°C, V_{CC} = 1.8 - 5.5V (Unless Otherwise Noted)

Symbol	Parameter	Min	Typ	Max	Units
$1/t_{CLCL}$	Oscillator Frequency (ATtiny2313A/4313)	0		4	MHz
t_{CLCL}	Oscillator Period (ATtiny2313A/4313)	250			ns
$1/t_{CLCL}$	Oscillator Frequency (ATtiny2313A/4313, V_{CC} = 4.5V - 5.5V)	0		20	MHz
t_{CLCL}	Oscillator Period (ATtiny2313A/4313, V_{CC} = 4.5V - 5.5V)	50			ns
t_{SHSL}	SCK Pulse Width High	2 t_{CLCL}*			ns
t_{SLSH}	SCK Pulse Width Low	2 t_{CLCL}*			ns
t_{OVSH}	MOSI Setup to SCK High	t_{CLCL}			ns
t_{SHOX}	MOSI Hold after SCK High	2 t_{CLCL}			ns
t_{SLIV}	SCK Low to MISO Valid			100	ns

Note: 2 t_{CLCL} for f_{ck} < 12 MHz, 3 t_{CLCL} for f_{ck} >= 12 MHz

23. Typical Characteristics

The data contained in this section is largely based on simulations and characterization of similar devices in the same process and design methods. Thus, the data should be treated as indications of how the part will behave.

The following charts show typical behavior. These figures are not tested during manufacturing. During characterisation devices are operated at frequencies higher than test limits but they are not guaranteed to function properly at frequencies higher than the ordering code indicates.

All current consumption measurements are performed with all I/O pins configured as inputs and with internal pull-ups enabled. Current consumption is a function of several factors such as operating voltage, operating frequency, loading of I/O pins, switching rate of I/O pins, code executed and ambient temperature. The dominating factors are operating voltage and frequency.

A sine wave generator with rail-to-rail output is used as clock source but current consumption in Power-Down mode is independent of clock selection. The difference between current consumption in Power-Down mode with Watchdog Timer enabled and Power-Down mode with Watchdog Timer disabled represents the differential current drawn by the Watchdog Timer.

The current drawn from pins with a capacitive load may be estimated (for one pin) as follows:

$$I_{CP} \approx V_{CC} \times C_L \times f_{SW}$$

where V_{CC} = operating voltage, C_L = load capacitance and f_{SW} = average switching frequency of I/O pin.

23.1 Effect of Power Reduction

Peripheral modules are enabled and disabled via control bits in the Power Reduction Register. See “Power Reduction Register” on page 35 for details.

Table 23-1. Additional Current Consumption (Absolute) for Peripherals of ATtiny2313A/4313

	Typical numbers		
PRR bit	**V_{CC} = 2V, f = 1MHz**	**V_{CC} = 3V, f = 4MHz**	**V_{CC} = 5V, f = 8MHz**
PRTIM0	2 µA	11 µA	50 µA
PRTIM1	5 µA	30 µA	120 µA
PRUSI	2 µA	11 µA	50 µA
PRUSART	4 µA	22 µA	95 µA

23.2 ATtiny2313A

23.2.1 Current Consumption in Active Mode

Figure 23-1. Active Supply Current vs. Low Frequency (0.1 - 1.0 MHz)

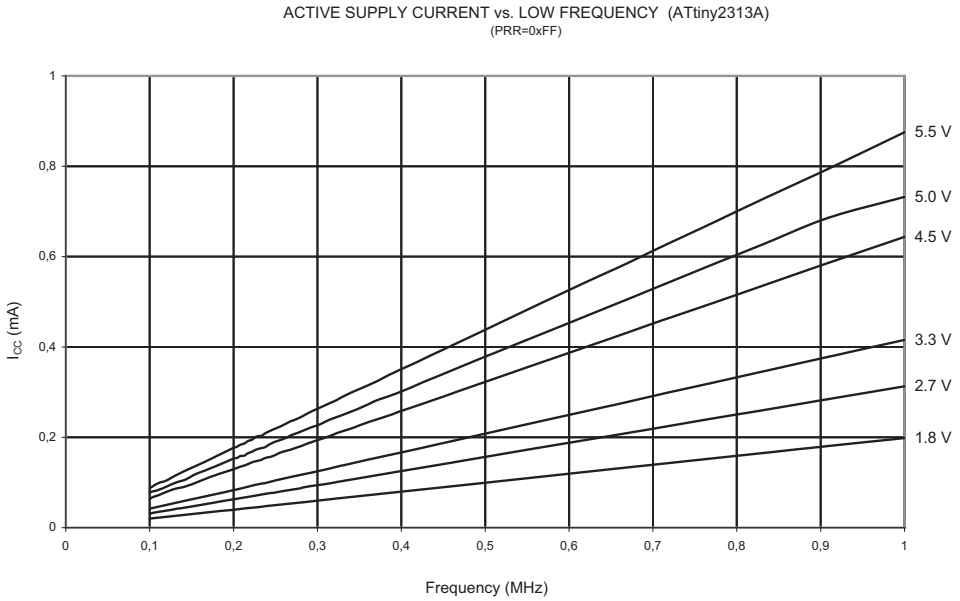


Figure 23-2. Active Supply Current vs. Frequency (1 - 20 MHz)

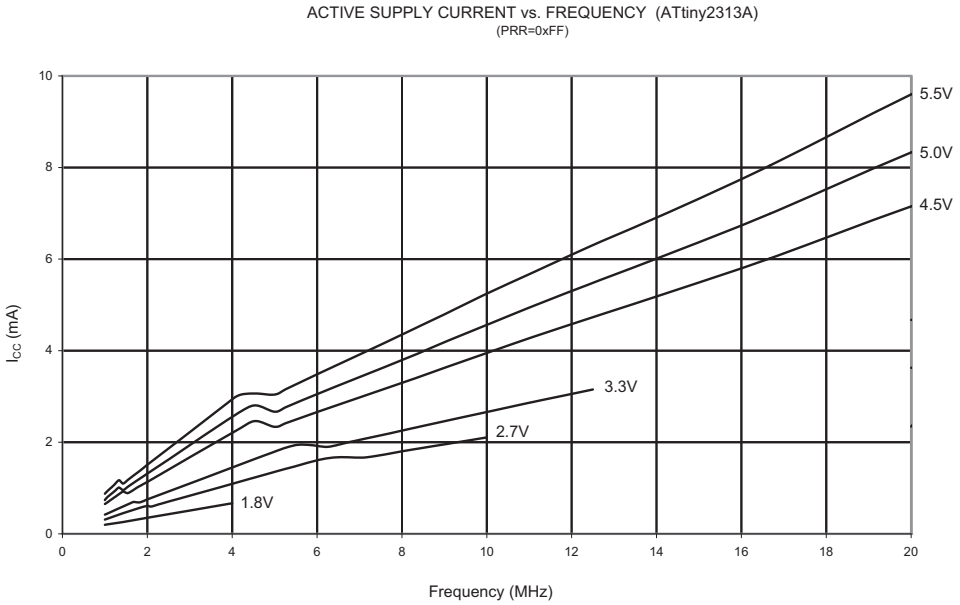


Figure 23-3. Active Supply Current vs. V_{CC} (Internal RC Oscillator, 8 MHz)

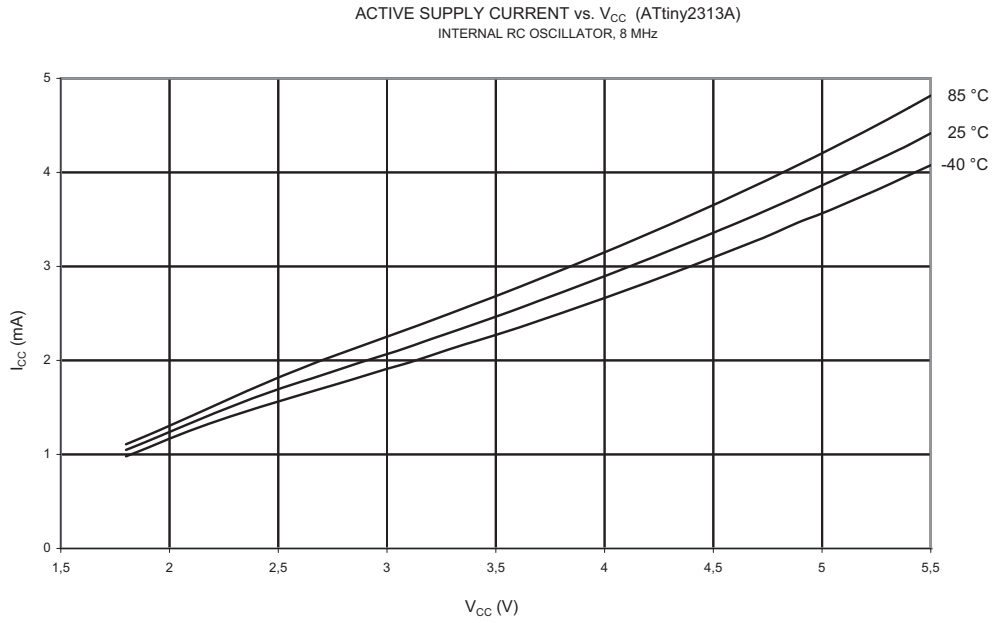


Figure 23-4. Active Supply Current vs. V_{CC} (Internal RC Oscillator, 1 MHz)

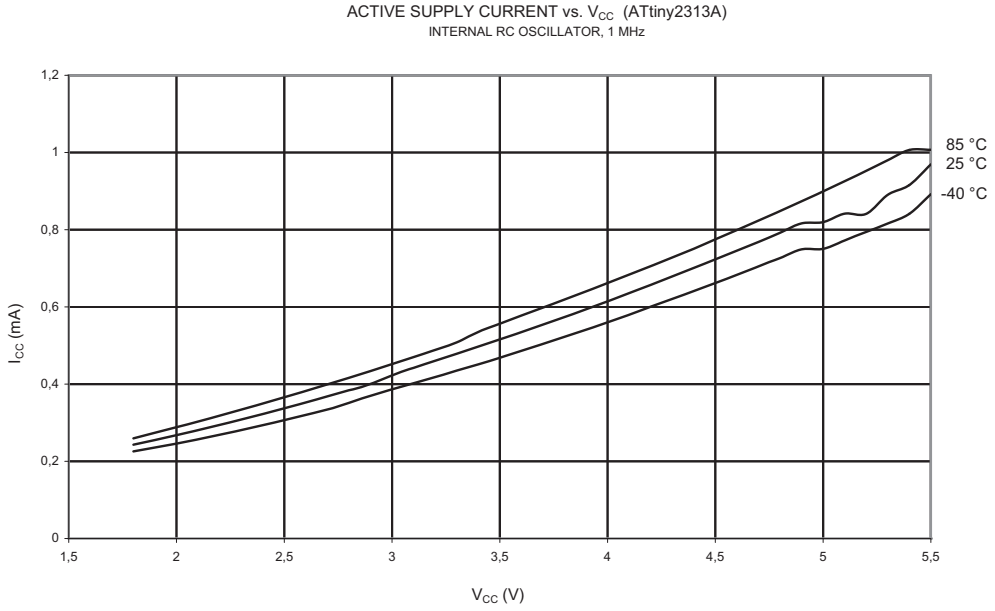
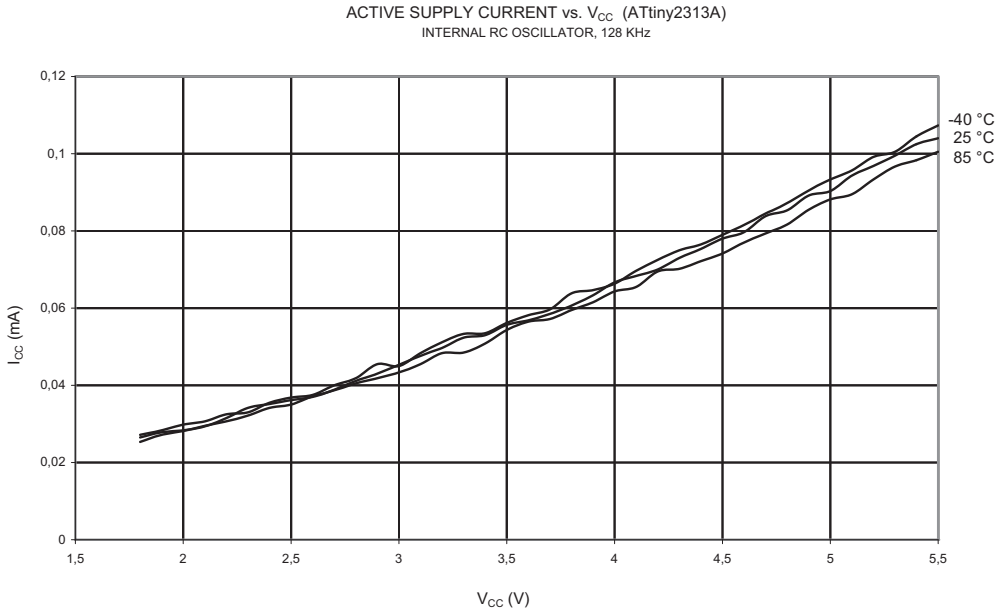


Figure 23-5. Active Supply Current vs. V_{CC} (Internal RC Oscillator, 128 KHz)



23.2.2 Current Consumption in Idle Mode

Figure 23-6. Idle Supply Current vs. Low Frequency (0.1 - 1.0 MHz)

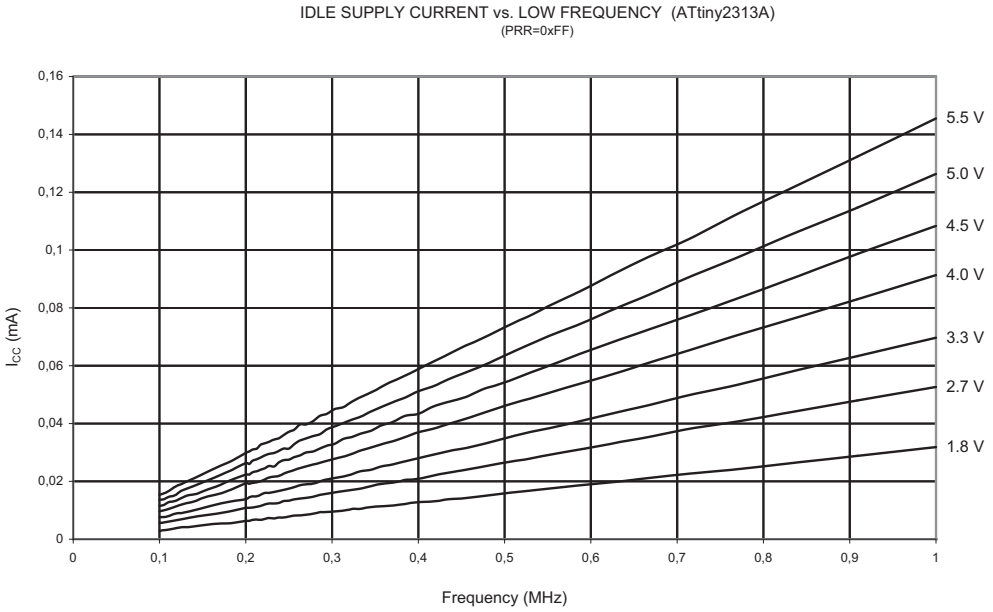


Figure 23-7. Idle Supply Current vs. Frequency (1 - 20 MHz)

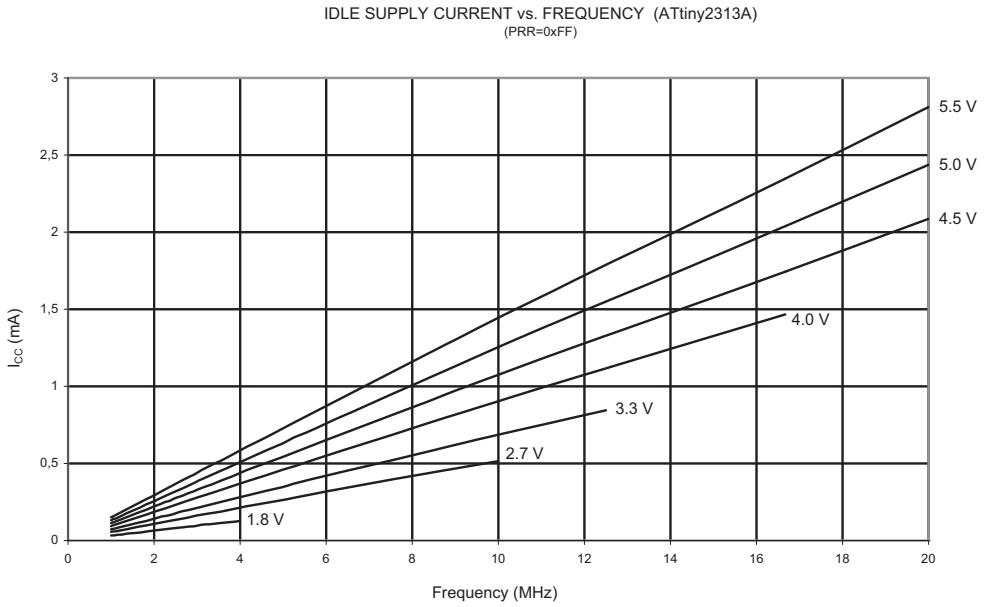


Figure 23-8. Idle Supply Current vs. V_{CC} (Internal RC Oscillator, 8 MHz)

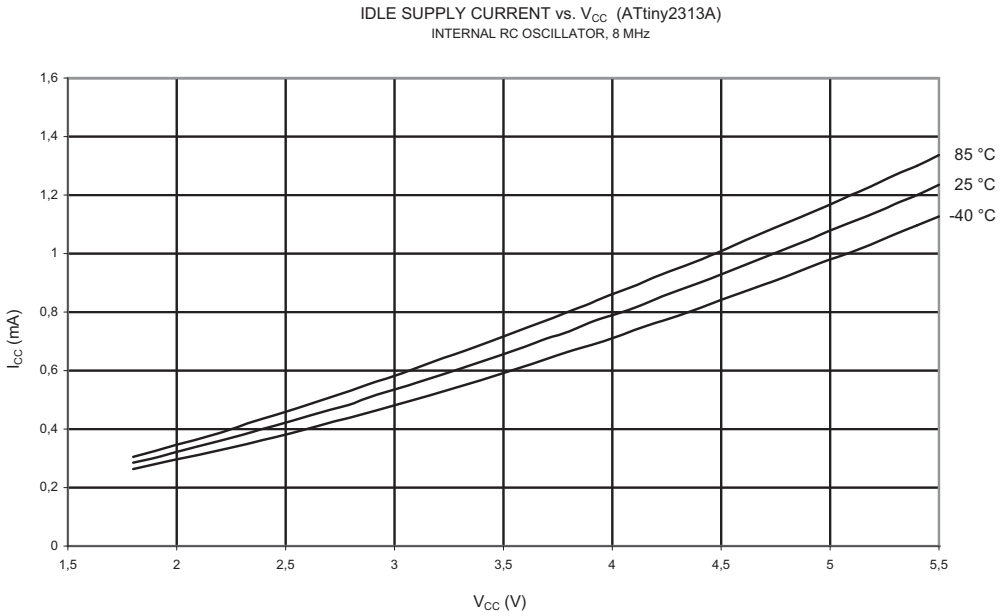


Figure 23-9. Idle Supply Current vs. V_{CC} (Internal RC Oscillator, 1 MHz)

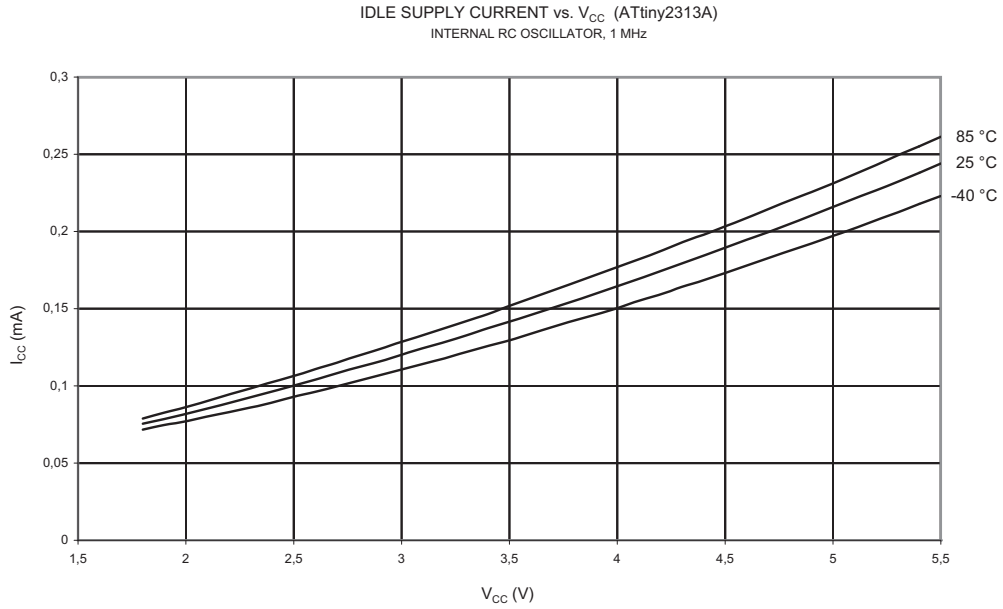
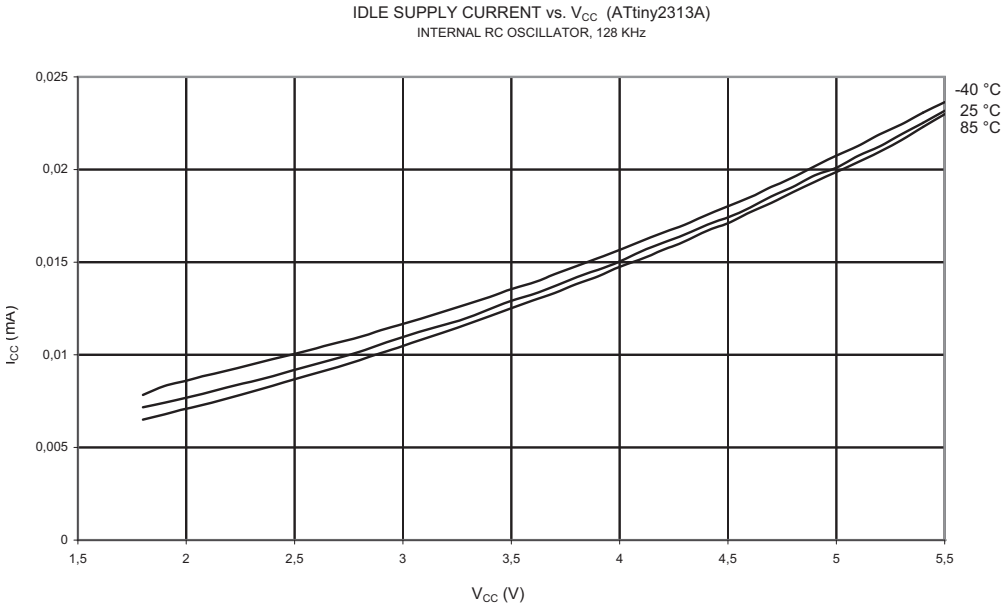


Figure 23-10. Idle Supply Current vs. V_{CC} (Internal RC Oscillator, 128 KHz)



23.2.3 Current Consumption in Power-down Mode

Figure 23-11. Power-down Supply Current vs. V_{CC} (Watchdog Timer Disabled)

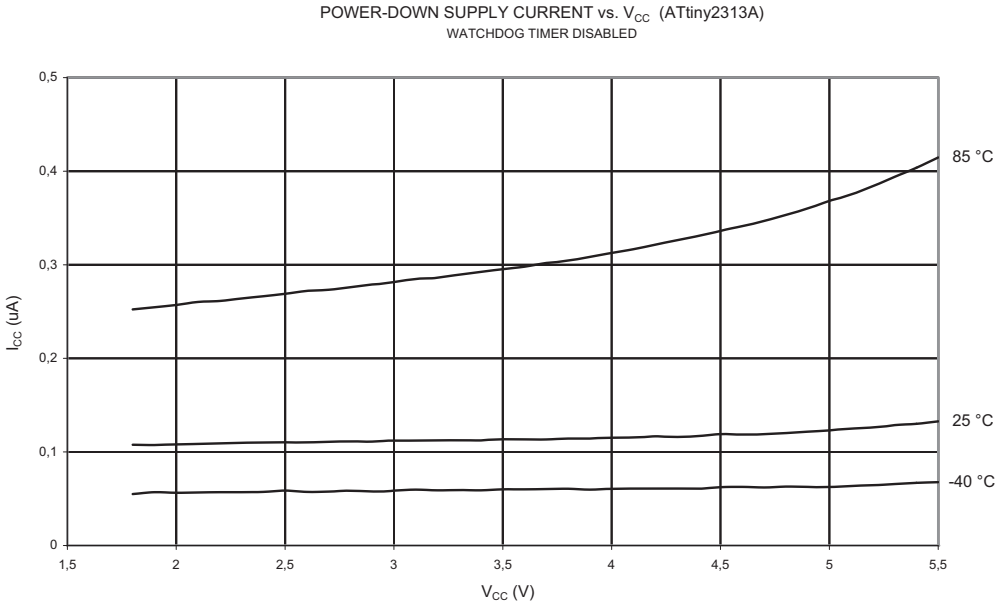
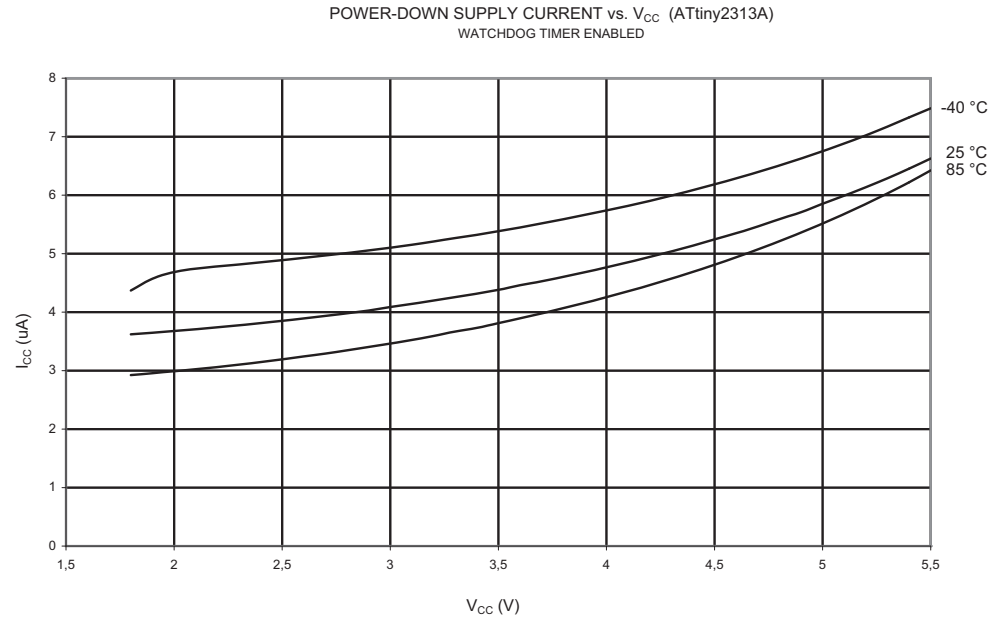


Figure 23-12. Power-down Supply Current vs. V_{CC} (Watchdog Timer Enabled)



23.2.4 Current Consumption in Reset

Figure 23-13. Reset Supply Current vs. V_{CC} (0.1 - 1.0 MHz, Excluding Current Through The Reset Pull-up)

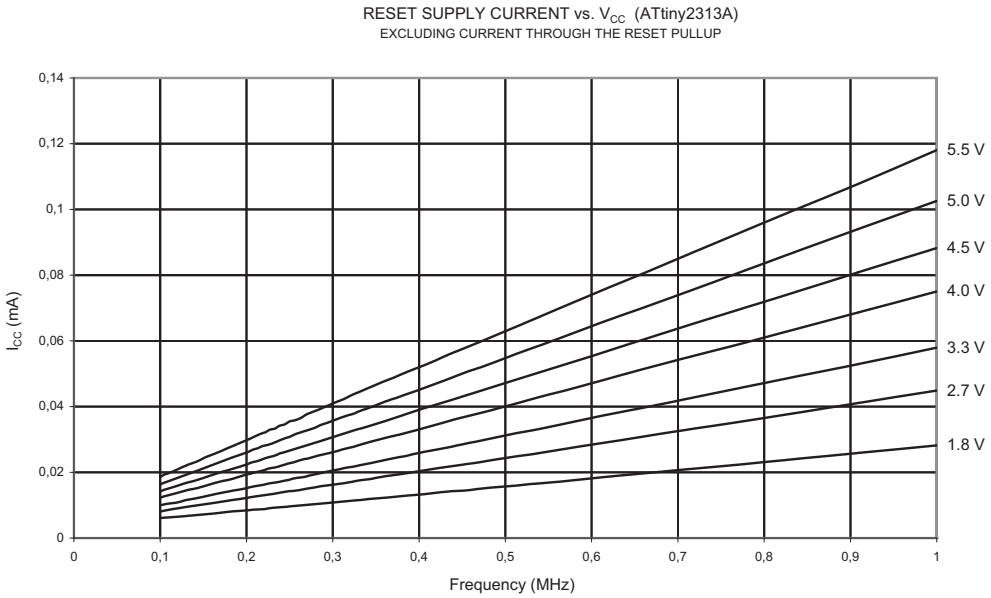
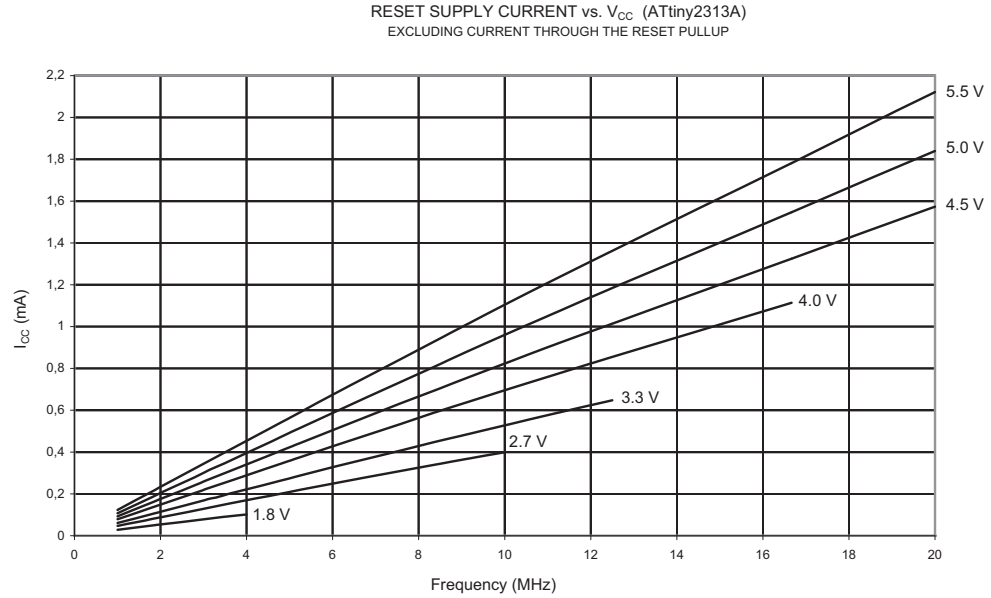


Figure 23-14. Reset Supply Current vs. V_{CC} (1 - 20 MHz, Excluding Current Through The Reset Pull-up)



23.2.5 Current Consumption of Peripheral Units

Figure 23-15. Brownout Detector Current vs. V_{CC}

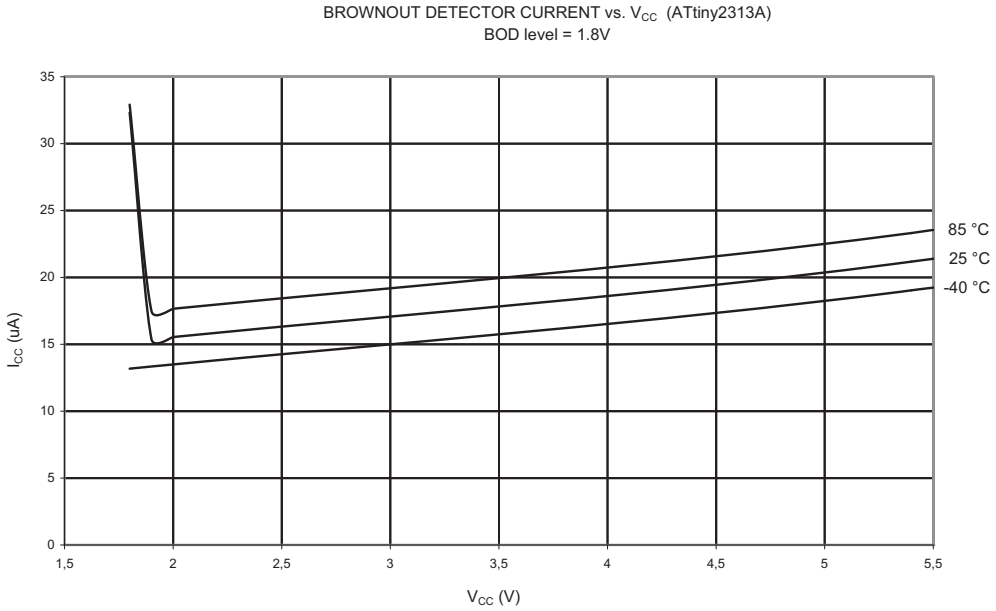
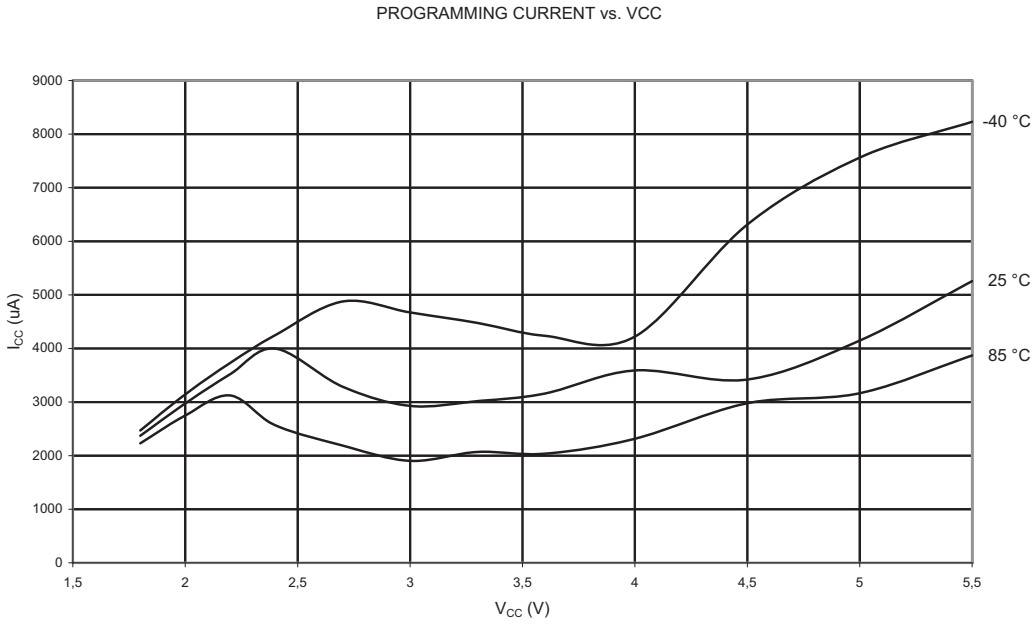


Figure 23-16. Programming Current vs. V_{CC} (ATtiny2313A)



Note: Above programming current based on simulation and characterisation of similar device (ATtiny24A).

23.2.6 Pull-up Resistors

Figure 23-17. Pull-up Resistor Current vs. Input Voltage (I/O Pin, V_{CC} = 1.8V)

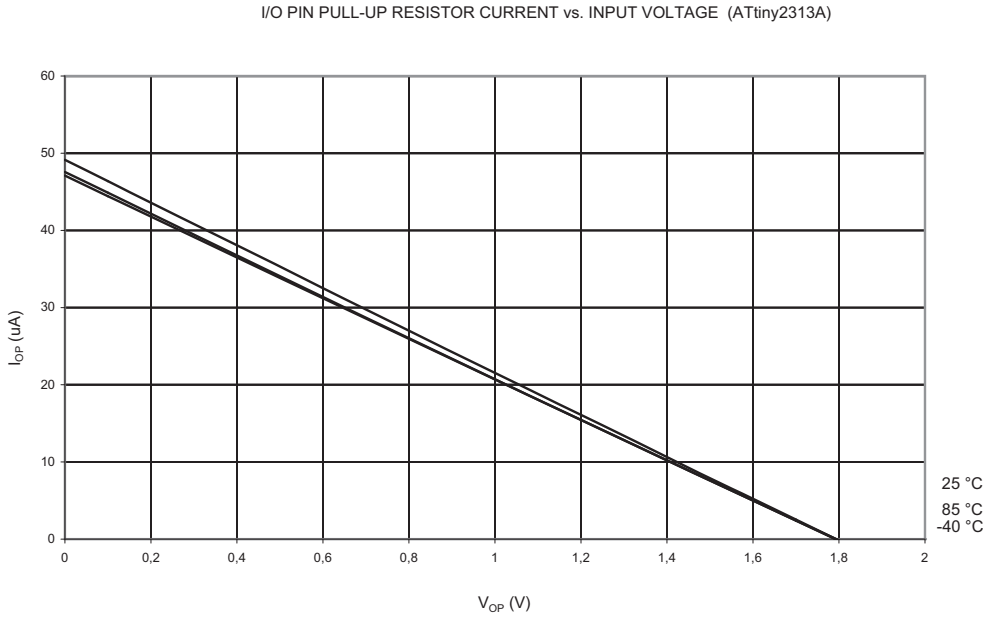


Figure 23-18. Pull-up Resistor Current vs. Input Voltage (I/O Pin, V_{CC} = 2.7V)

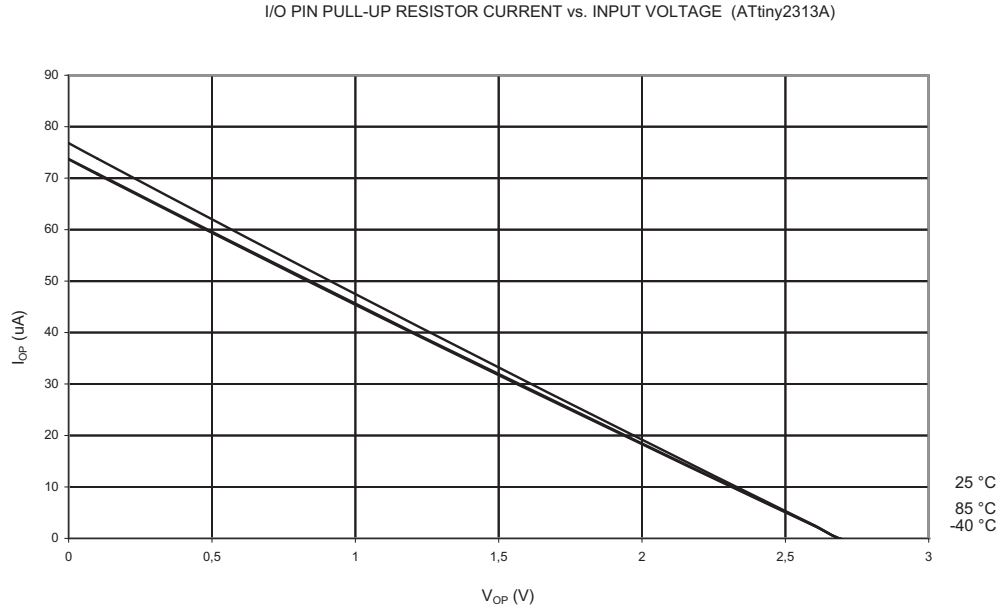


Figure 23-19. Pull-up Resistor Current vs. Input Voltage (I/O Pin, V_{CC} = 5V)

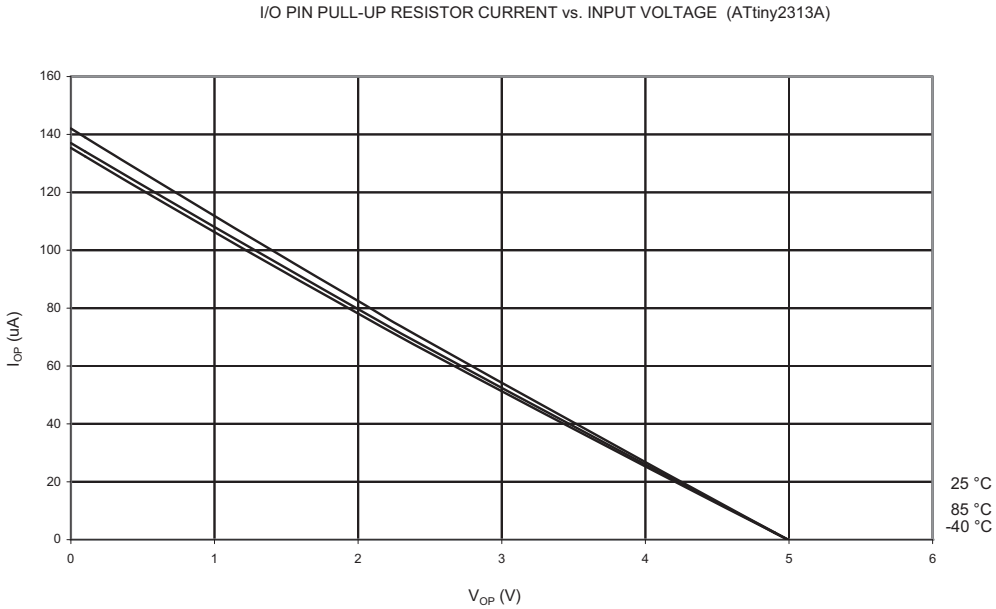


Figure 23-20. Reset Pull-up Resistor Current vs. Reset Pin Voltage (V_{CC} = 1.8V)

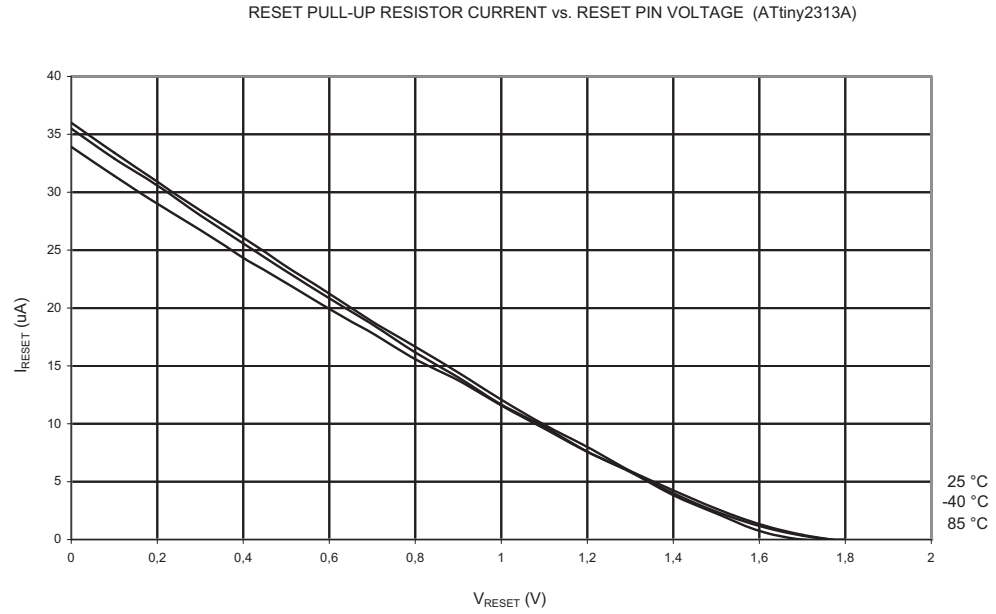


Figure 23-21. Reset Pull-up Resistor Current vs. Reset Pin Voltage (V_{CC} = 2.7V)

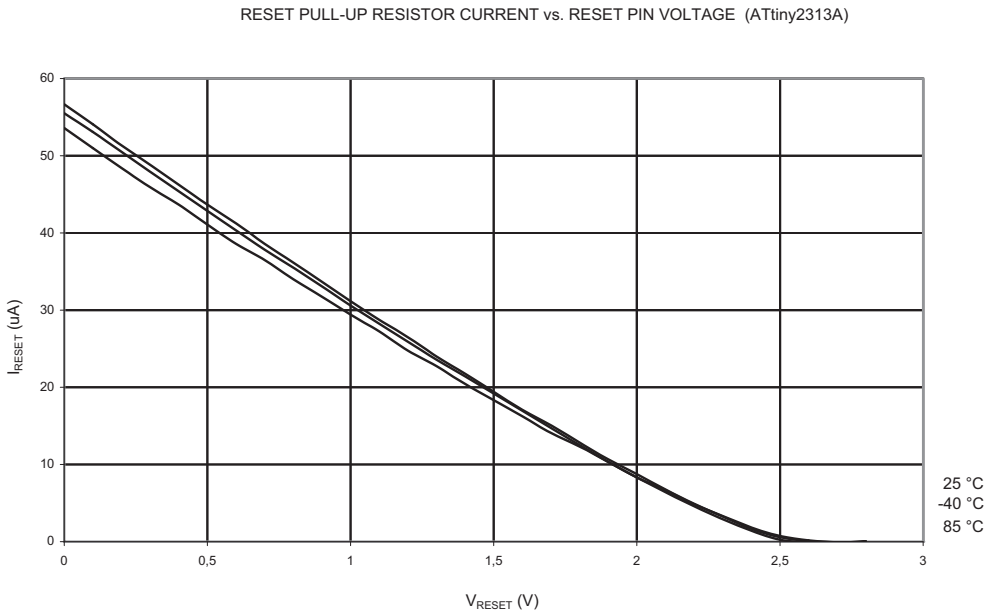
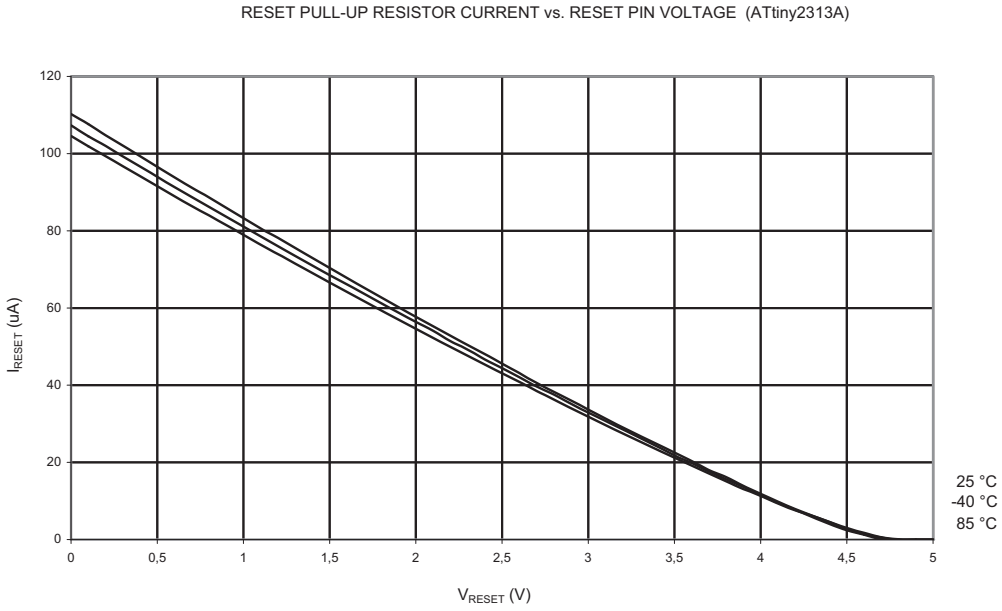


Figure 23-22. Reset Pull-up Resistor Current vs. Reset Pin Voltage (V_{CC} = 5V)



23.2.7 Output Driver Strength

Figure 23-23. V_{OL}: Output Voltage vs. Sink Current (I/O Pin, V_{CC} = 1.8V)

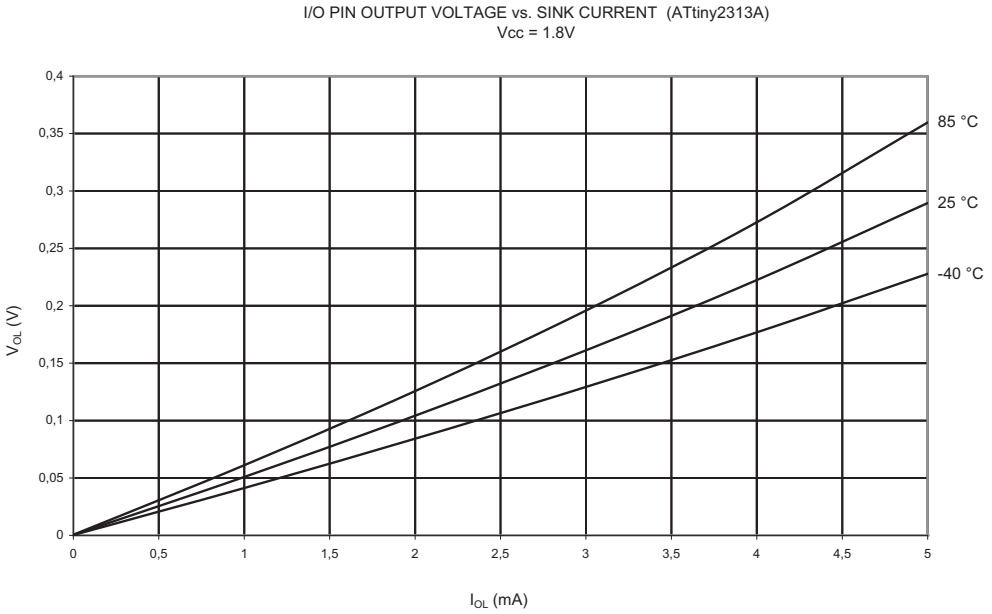


Figure 23-24. V_{OL}: Output Voltage vs. Sink Current (I/O Pin, V_{CC} = 3V)

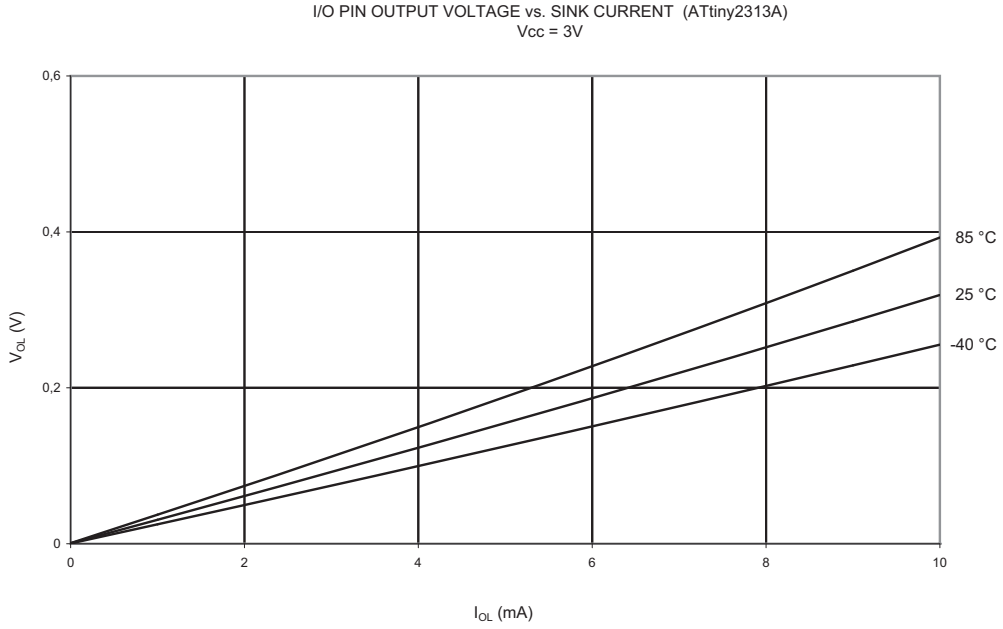


Figure 23-25. V_{OL}: Output Voltage vs. Sink Current (I/O Pin, V_{CC} = 5V)

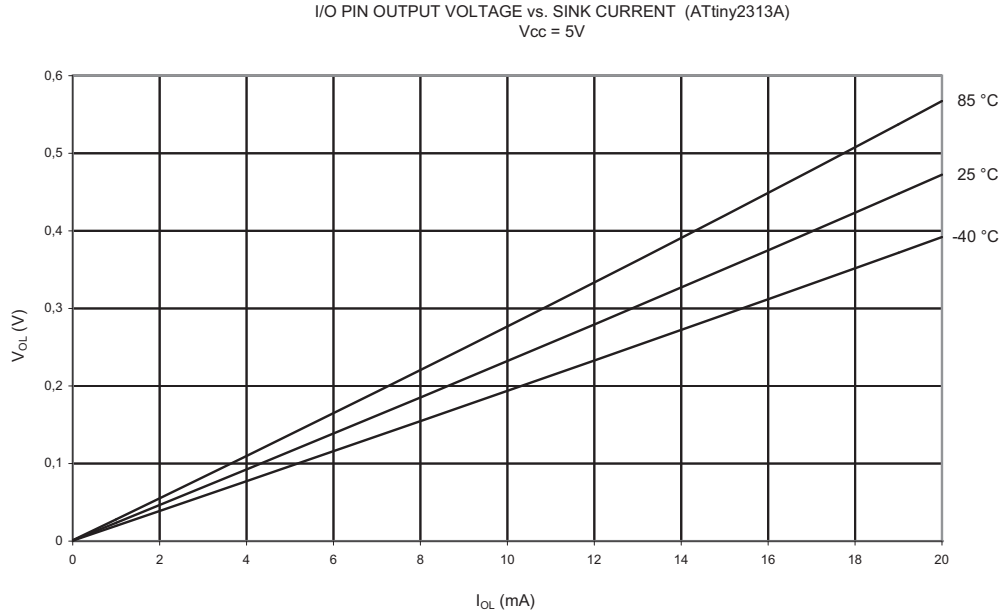


Figure 23-26. V_{OH}: Output Voltage vs. Source Current (I/O Pin, V_{CC} = 1.8V)

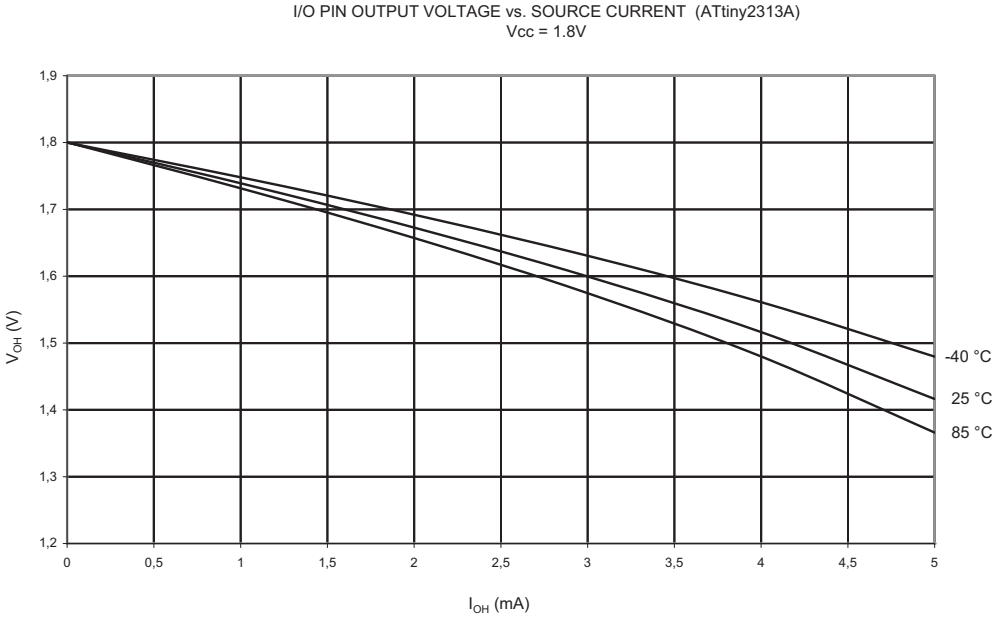


Figure 23-27. V_{OH}: Output Voltage vs. Source Current (I/O Pin, V_{CC} = 3V)

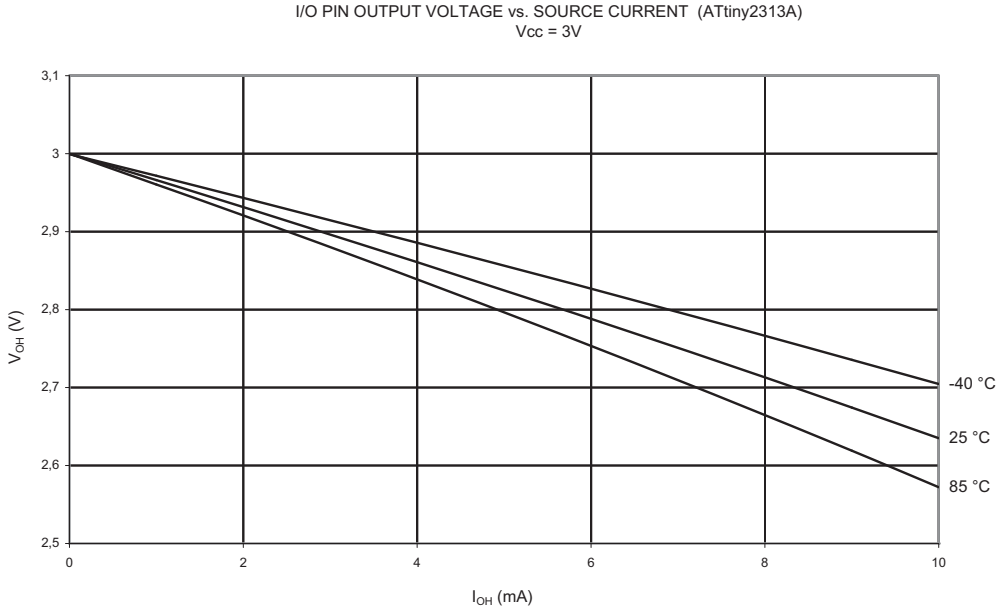


Figure 23-28. V_{OH}: Output Voltage vs. Source Current (I/O Pin, V_{CC} = 5V)

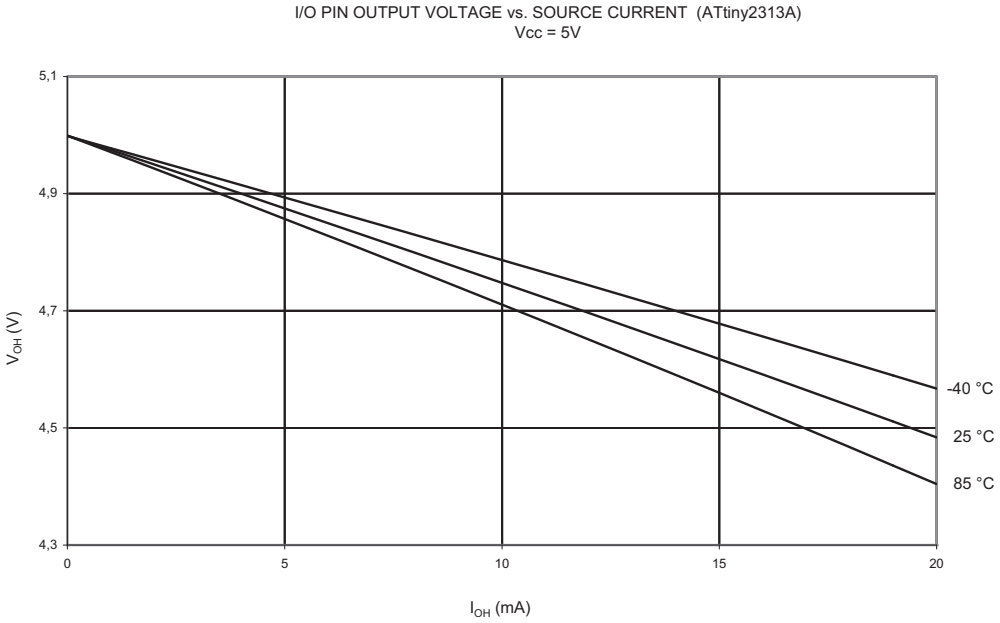


Figure 23-29. V_{OL}: Output Voltage vs. Sink Current (Reset Pin as I/O, T = 25°C)

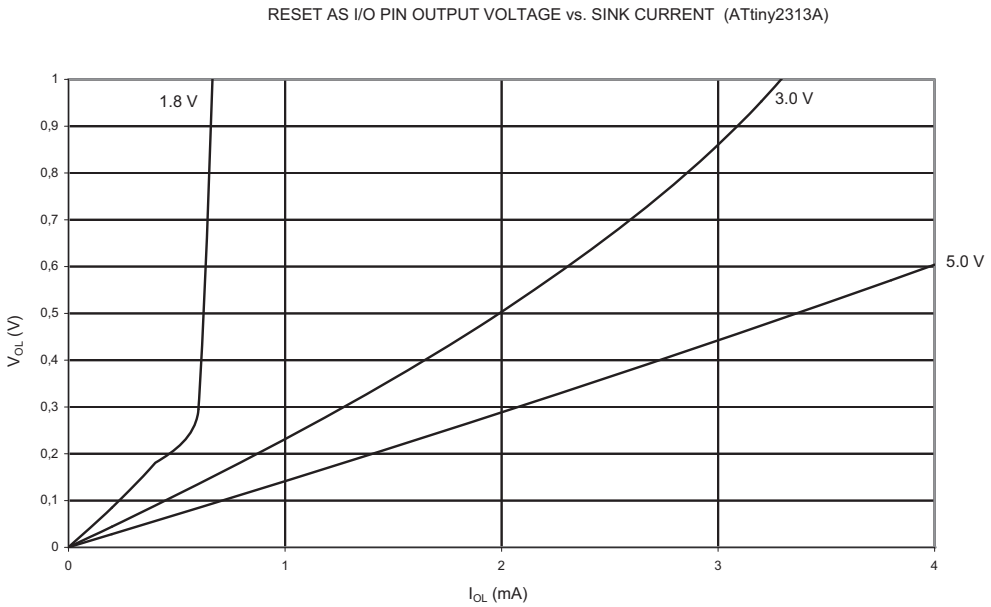
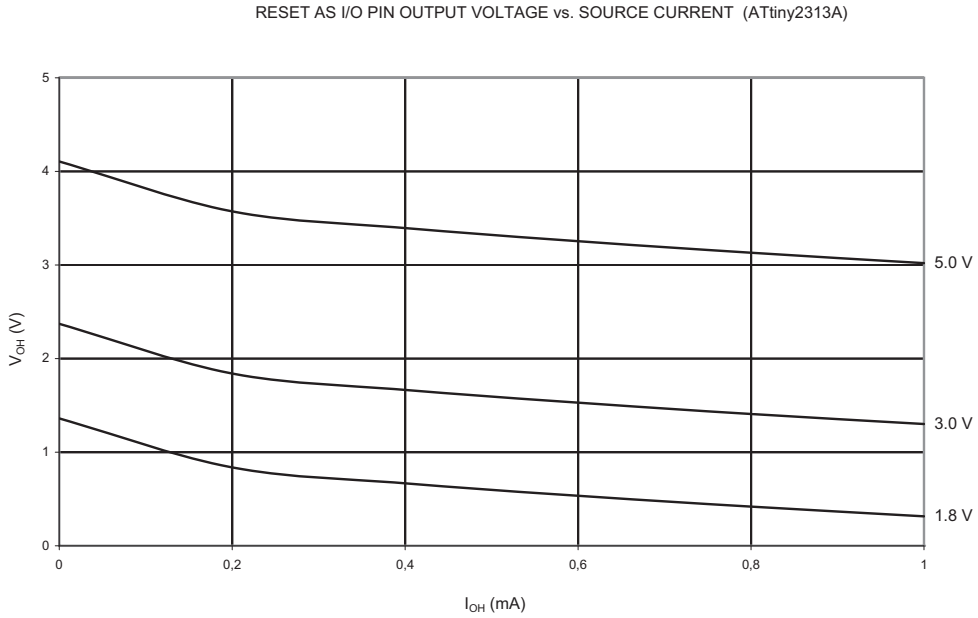


Figure 23-30. V_{OH}: Output Voltage vs. Source Current (Reset Pin as I/O, T = 25°C)



23.2.8 Input Thresholds and Hysteresis (for I/O Ports)

Figure 23-31. V_{IH}: Input Threshold Voltage vs. V_{CC} (I/O Pin Read as '1')

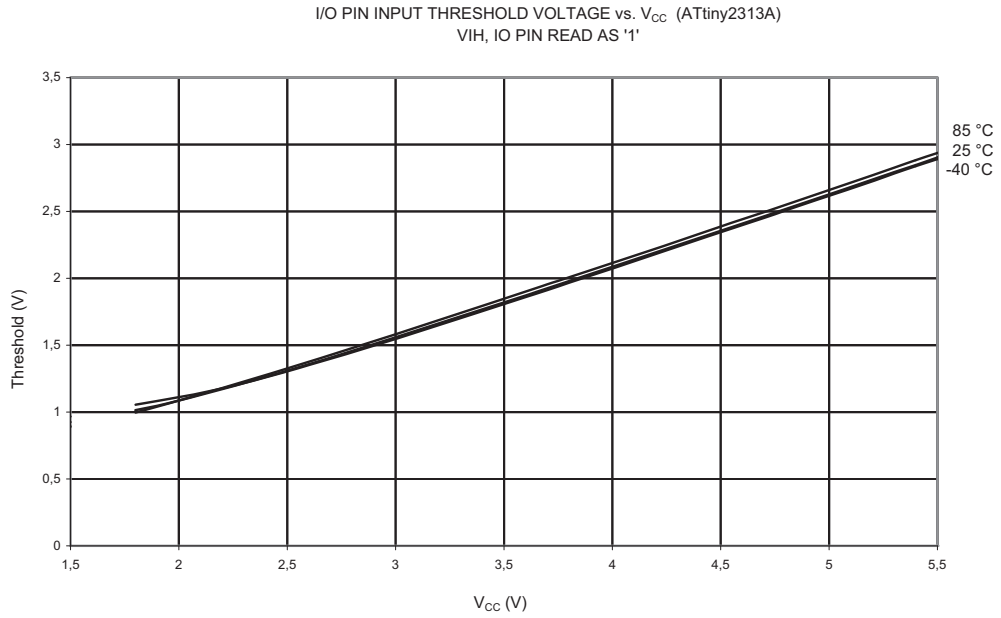


Figure 23-32. V_{IL}: Input Threshold Voltage vs. V_{CC} (I/O Pin, Read as '0')

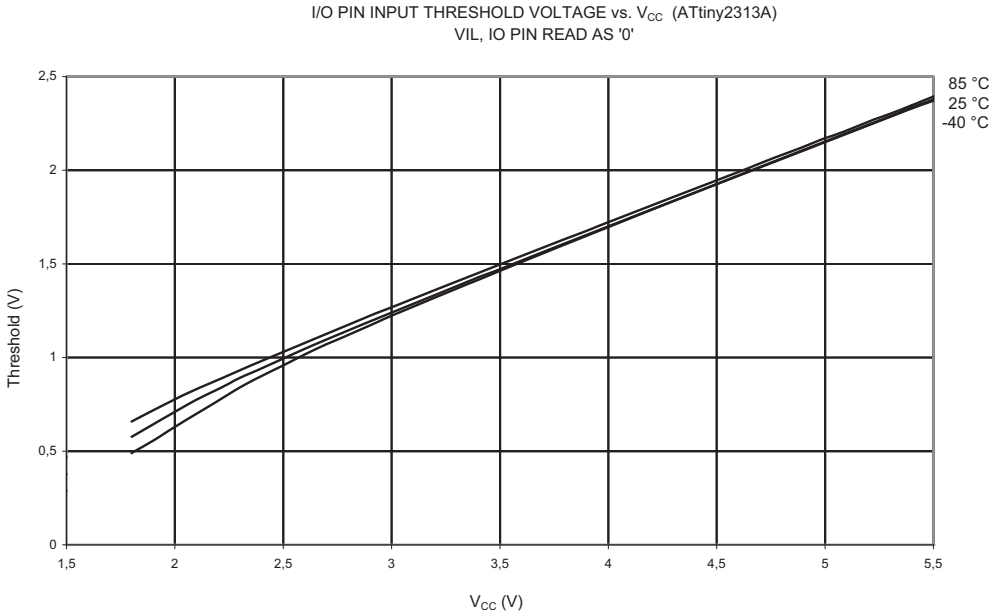


Figure 23-33. V_{IH}-V_{IL}: Input Hysteresis vs. V_{CC} (I/O Pin)

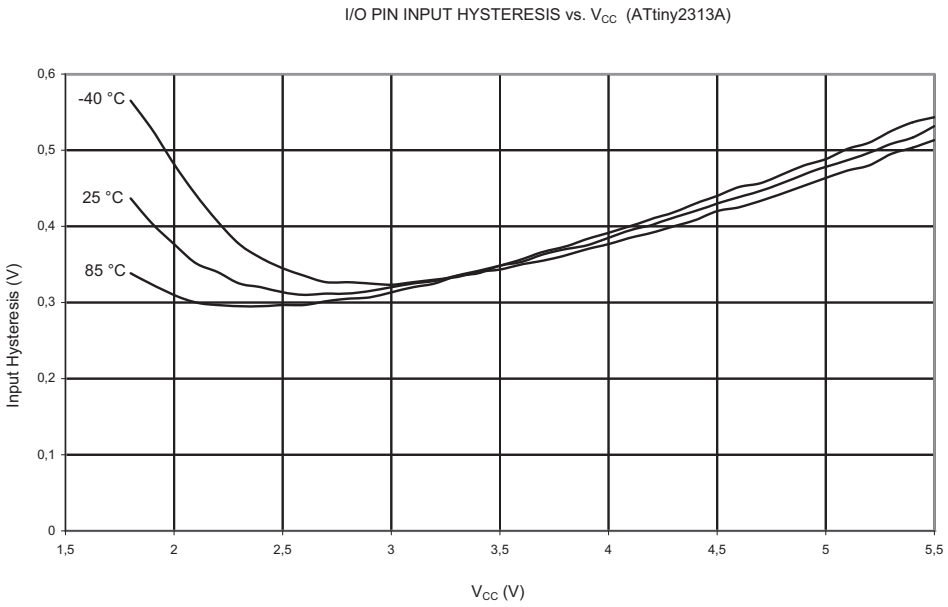


Figure 23-34. V_{IH}: Input Threshold Voltage vs. V_{CC} (Reset Pin as I/O, Read as ‘1’)

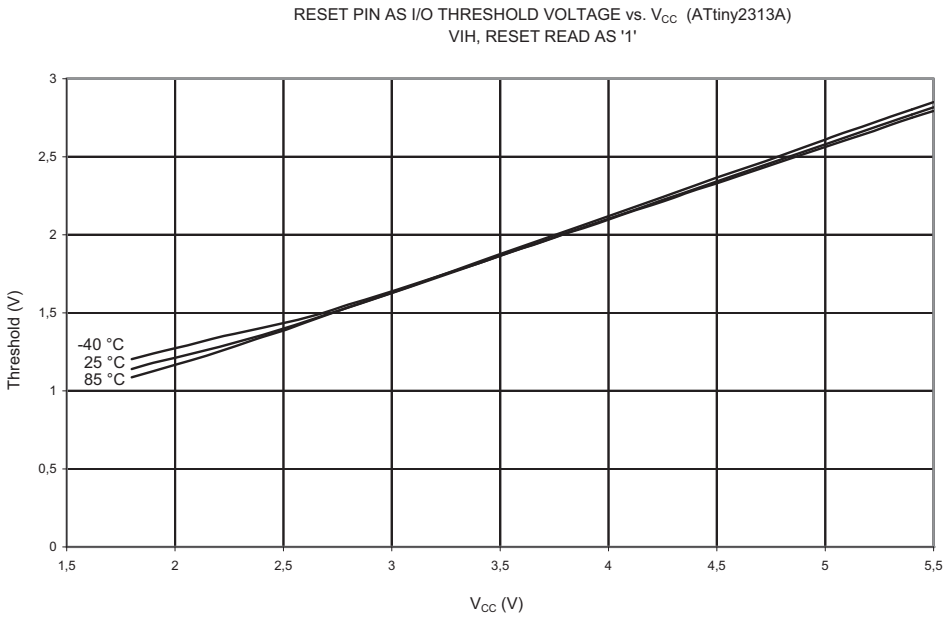


Figure 23-35. V_{IL}: Input Threshold Voltage vs. V_{CC} (Reset Pin as I/O, Read as ‘0’)

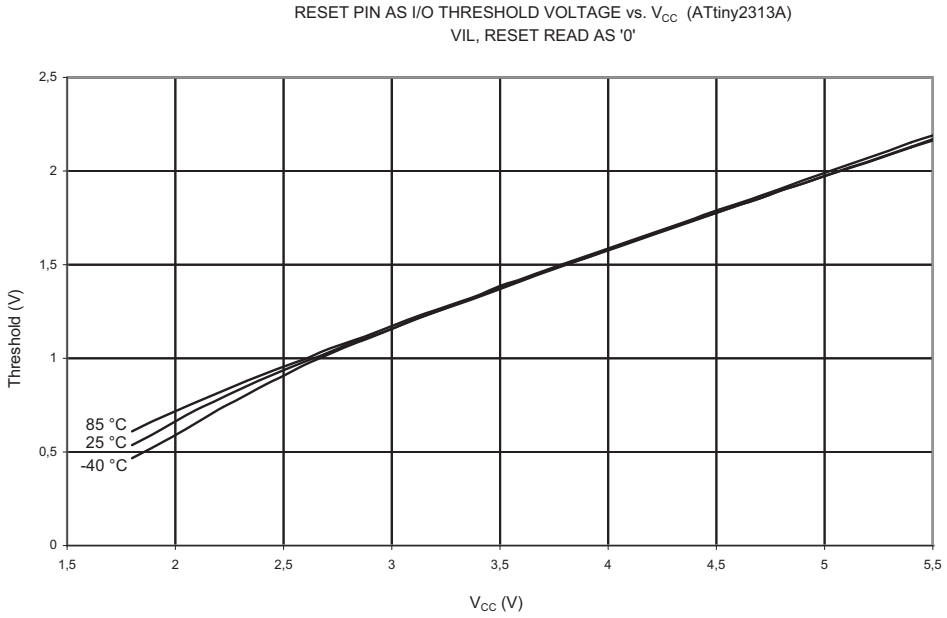
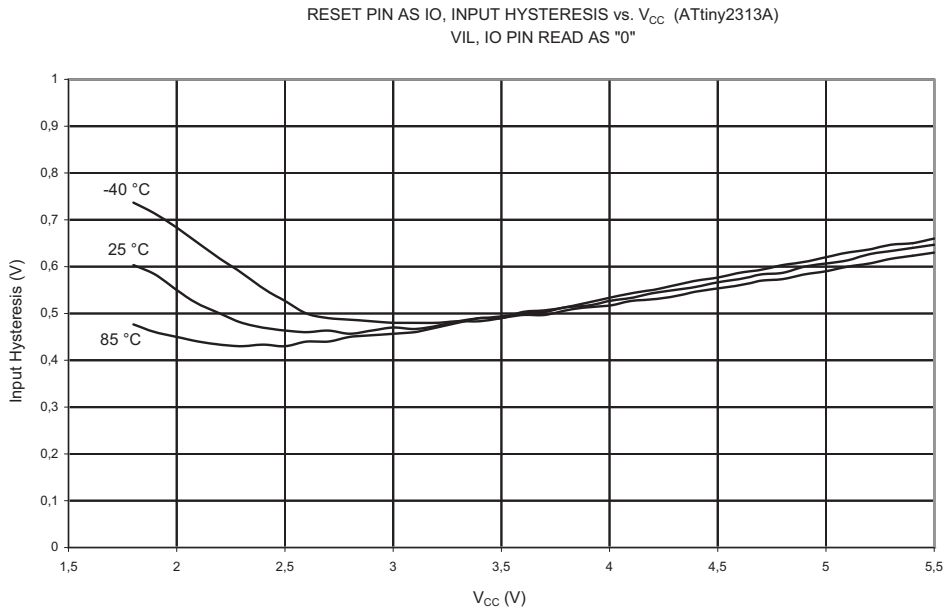


Figure 23-36. V_{IH}-V_{IL}: Input Hysteresis vs. V_{CC} (Reset Pin as I/O)



23.2.9 BOD, Bandgap and Reset

Figure 23-37. BOD Thresholds vs. Temperature (BOD Level is 4.3V)

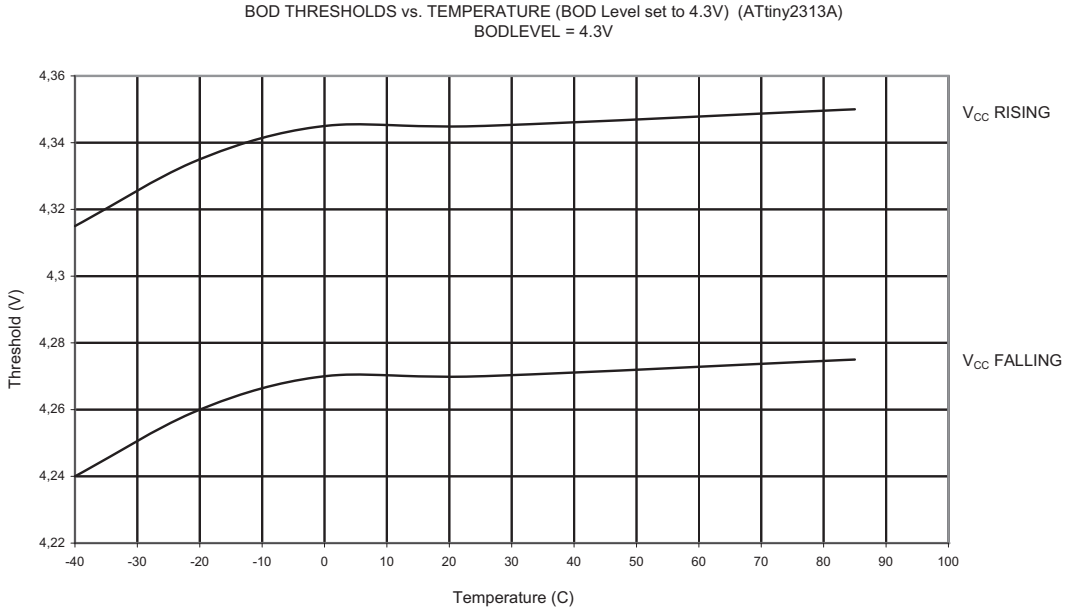


Figure 23-38. BOD Thresholds vs. Temperature (BOD Level is 2.7V)

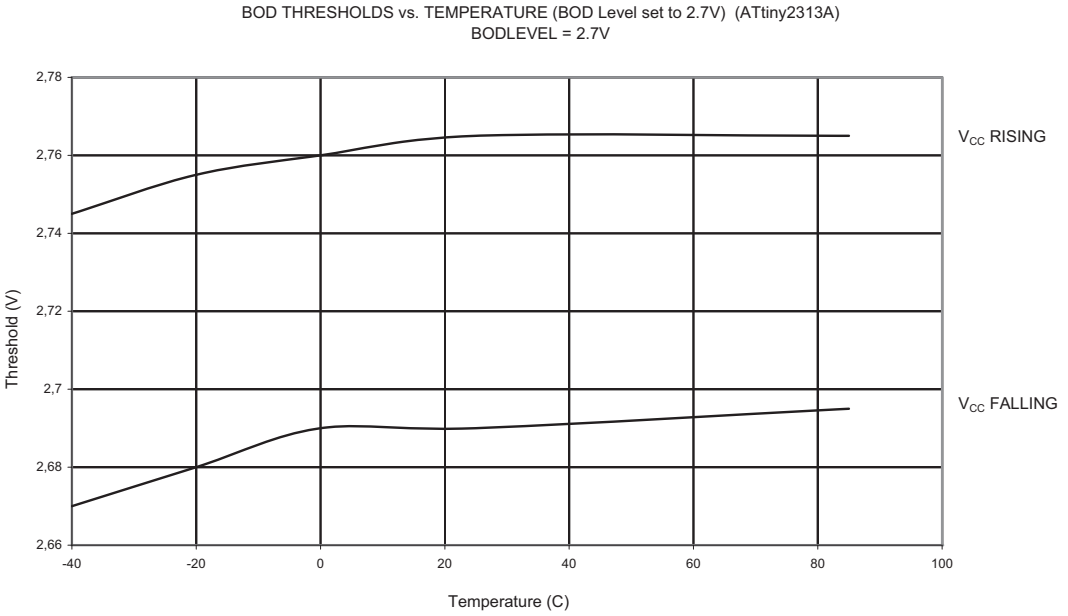


Figure 23-39. BOD Thresholds vs. Temperature (BOD Level is 1.8V)

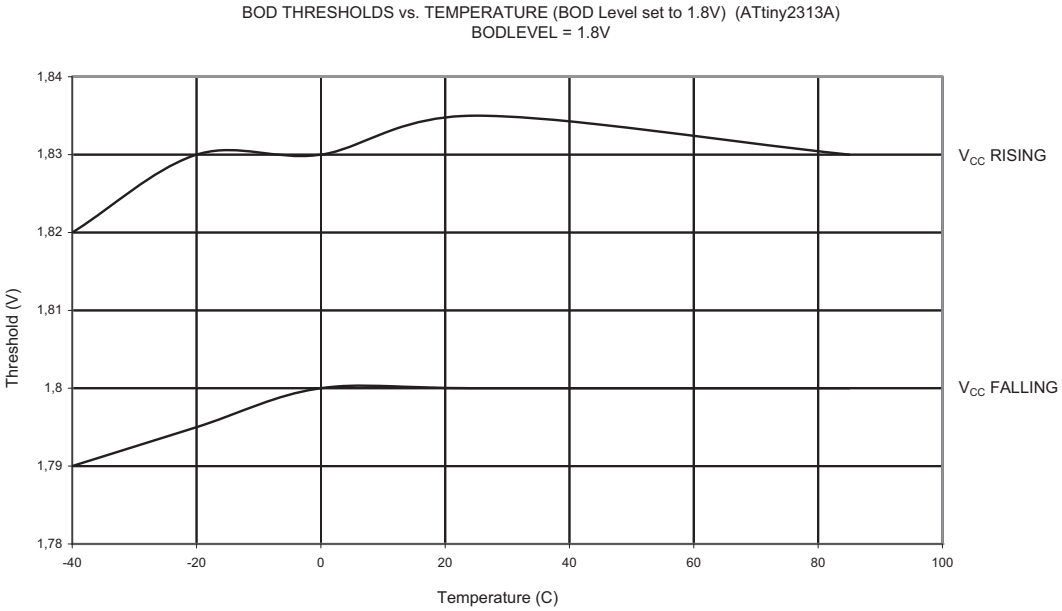


Figure 23-40. Bandgap Voltage vs. Supply Voltage

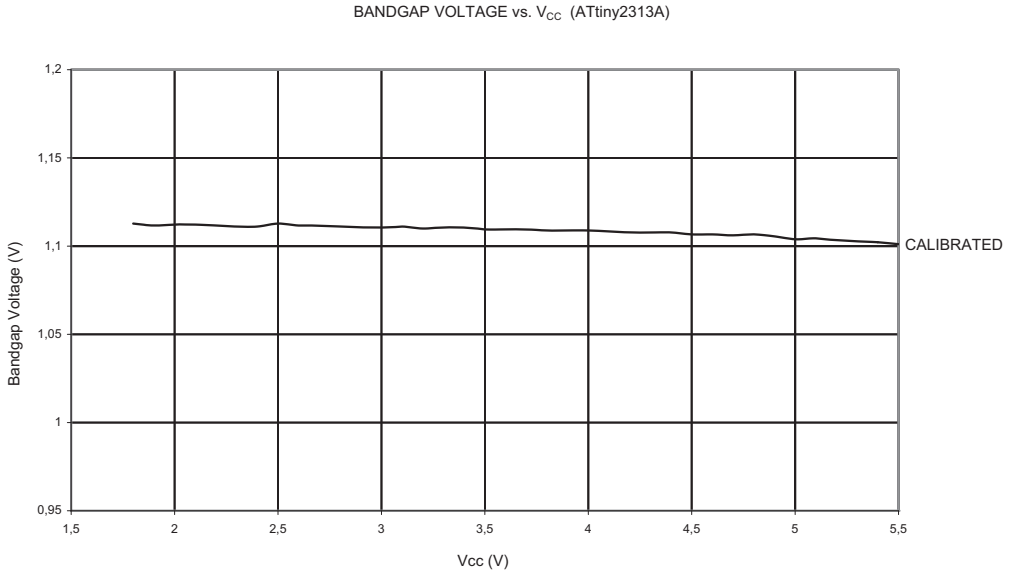


Figure 23-41. Bandgap Voltage vs. Temperature

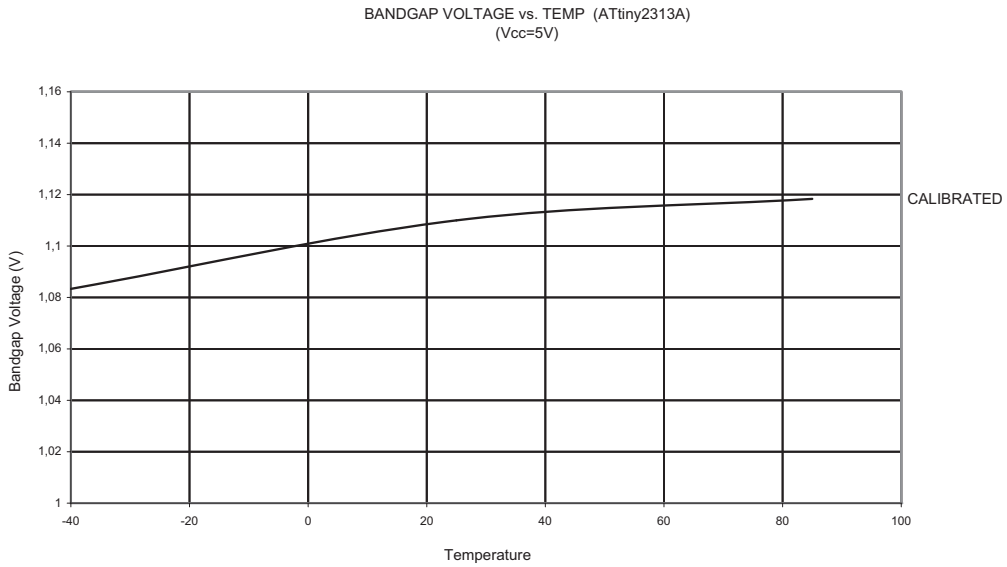


Figure 23-42. V_{IH}: Input Threshold Voltage vs. V_{CC} (Reset Pin, Read as ‘1’)

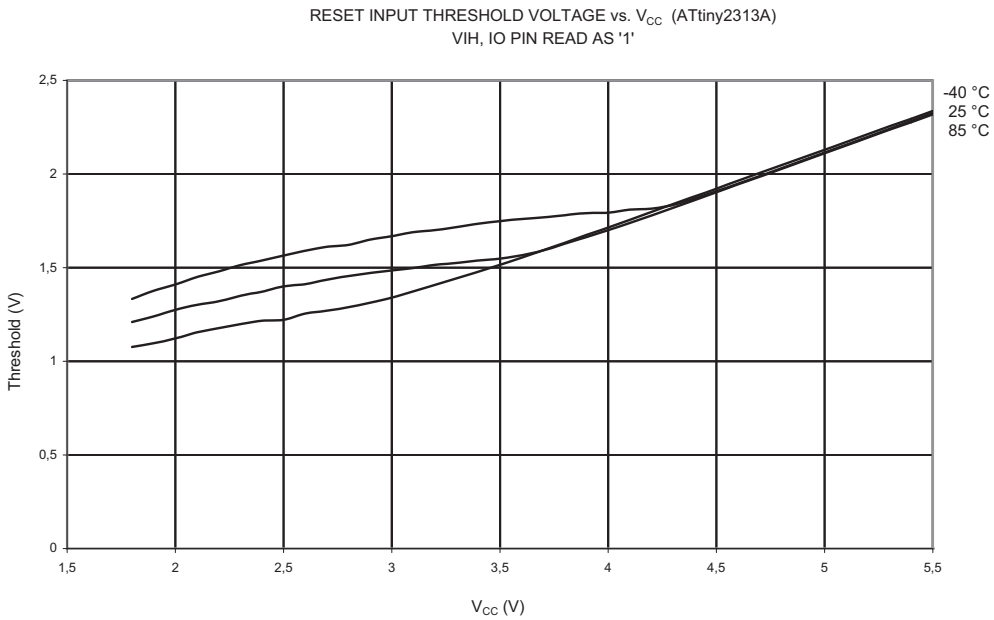


Figure 23-43. V_{IL}: Input Threshold Voltage vs. V_{CC} (Reset Pin, Read as ‘0’)

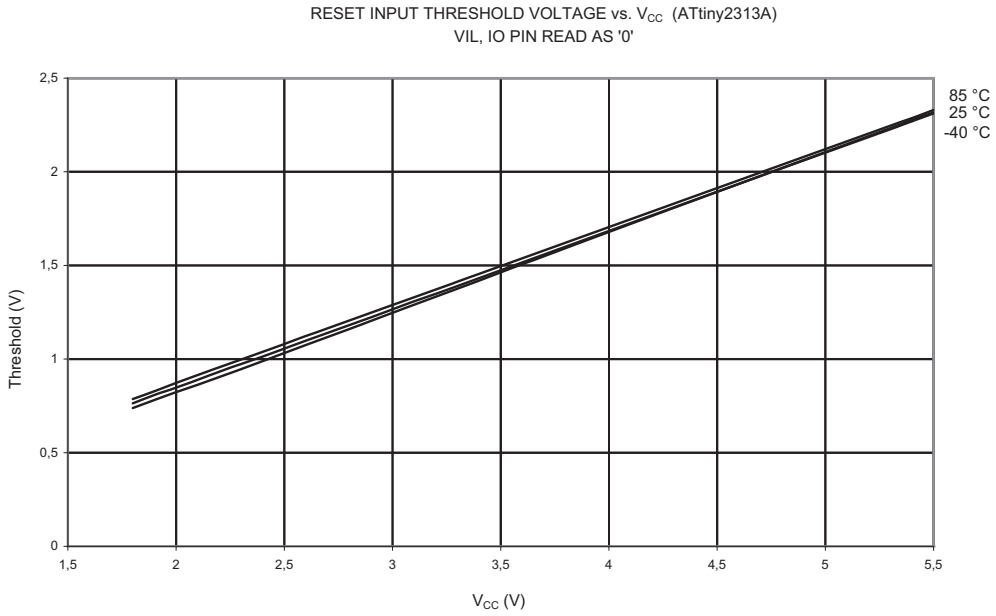


Figure 23-44. V_{IH}-V_{IL}: Input Hysteresis vs. V_{CC} (Reset Pin)

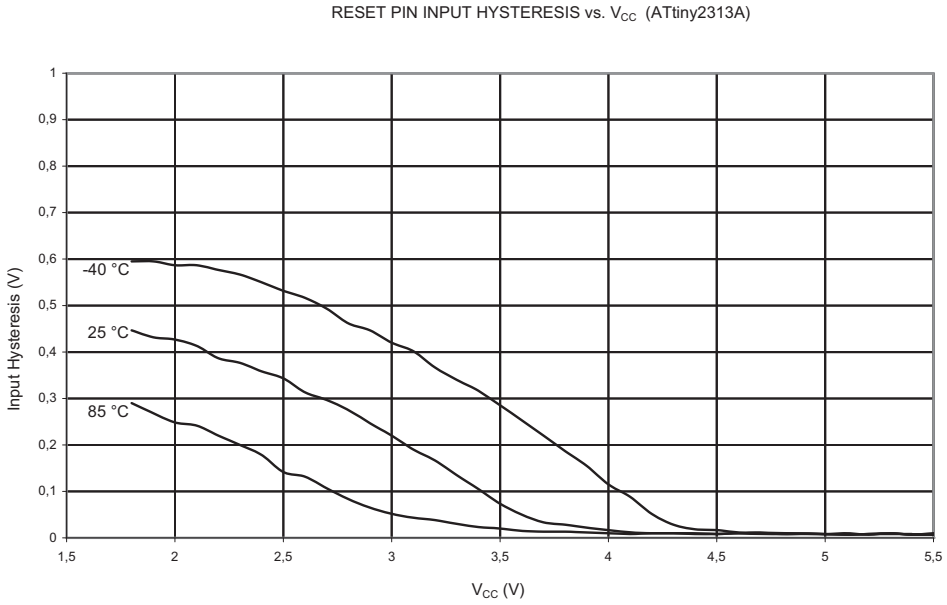
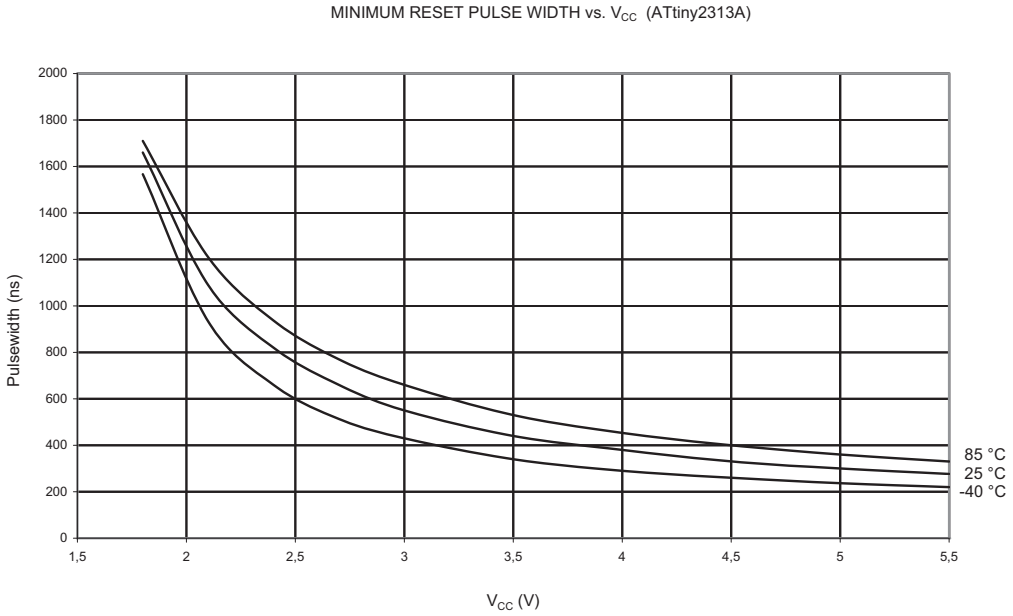


Figure 23-45. Minimum Reset Pulse Width vs. V_{CC}



23.2.10 Internal Oscillator Speed

Figure 23-46. Calibrated 8 MHz RC Oscillator Frequency vs. V_{CC}

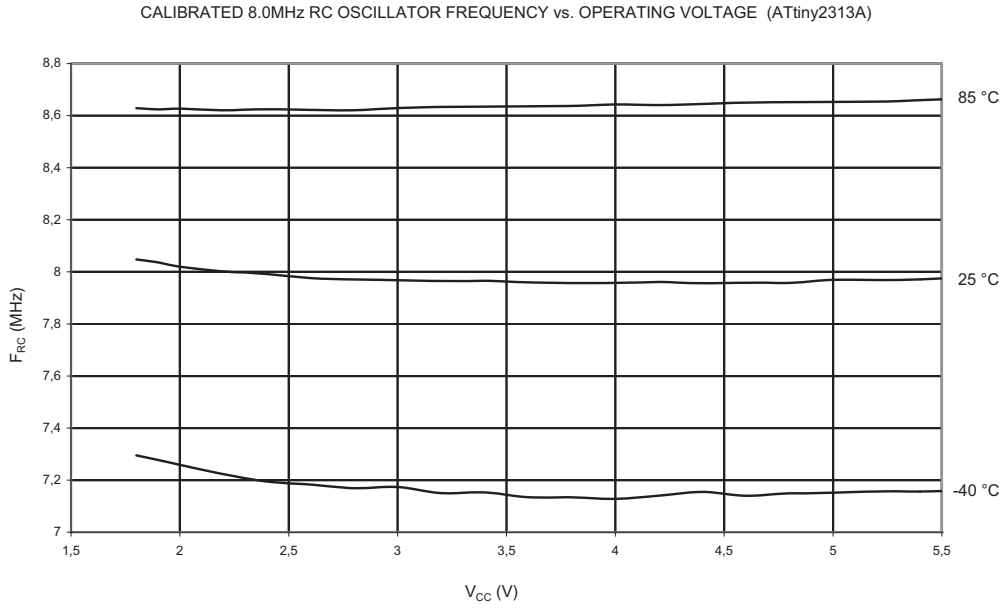


Figure 23-47. Calibrated 8 MHz RC Oscillator Frequency vs. Temperature

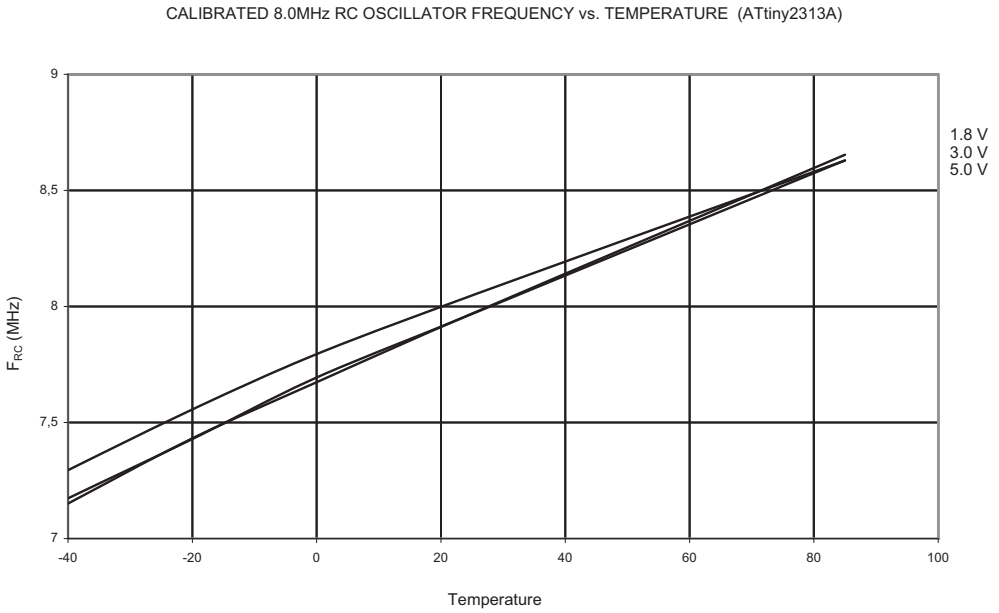
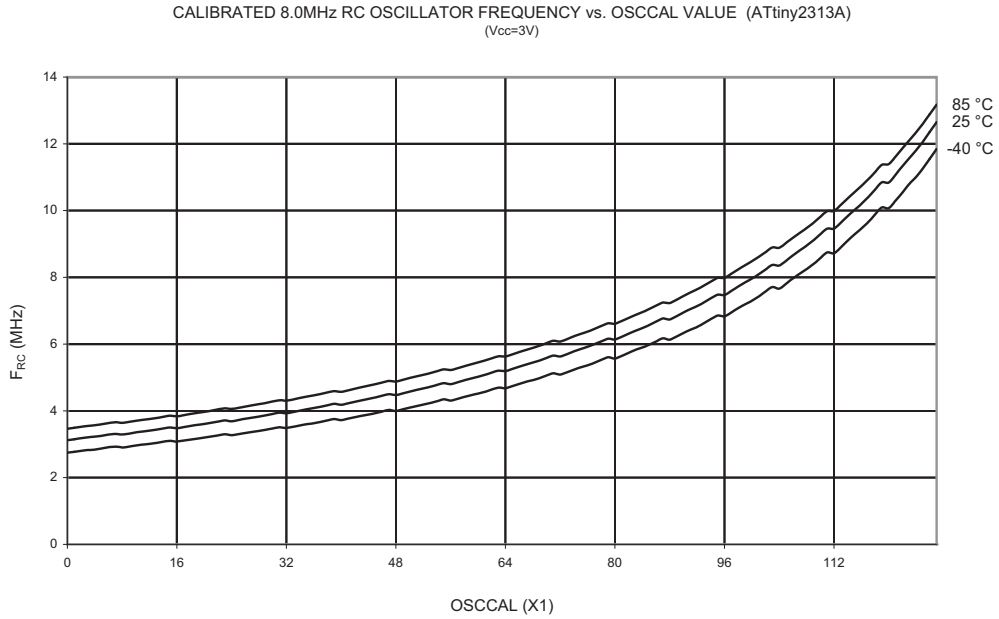


Figure 23-48. Calibrated 8 MHz RC Oscillator Frequency vs. Osccal Value



23.3 ATtiny4313

23.3.1 Current Consumption in Active Mode

Figure 23-49. Active Supply Current vs. Low Frequency (0.1 - 1.0 MHz)

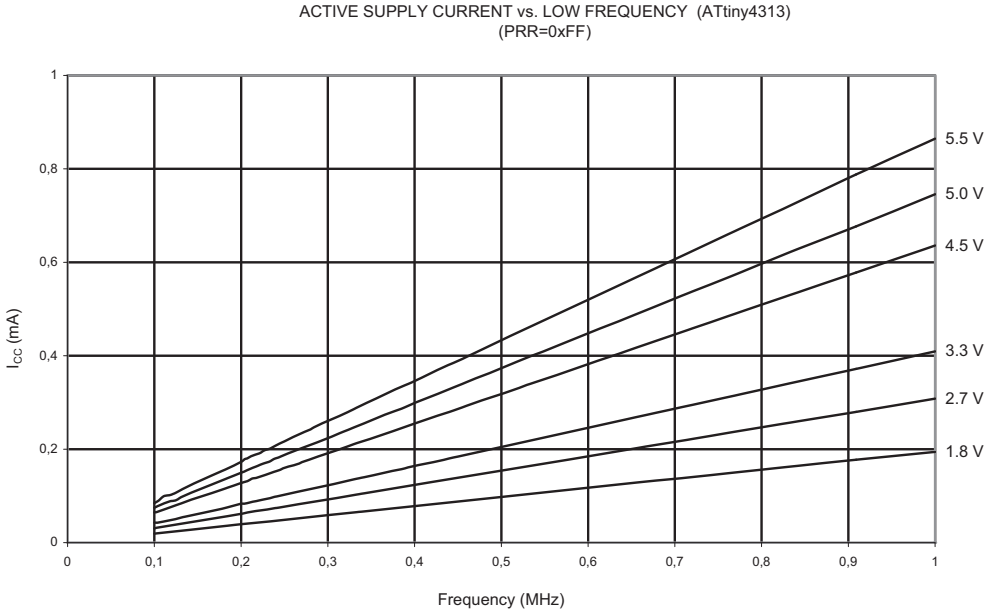


Figure 23-50. Active Supply Current vs. Frequency (1 - 20 MHz)

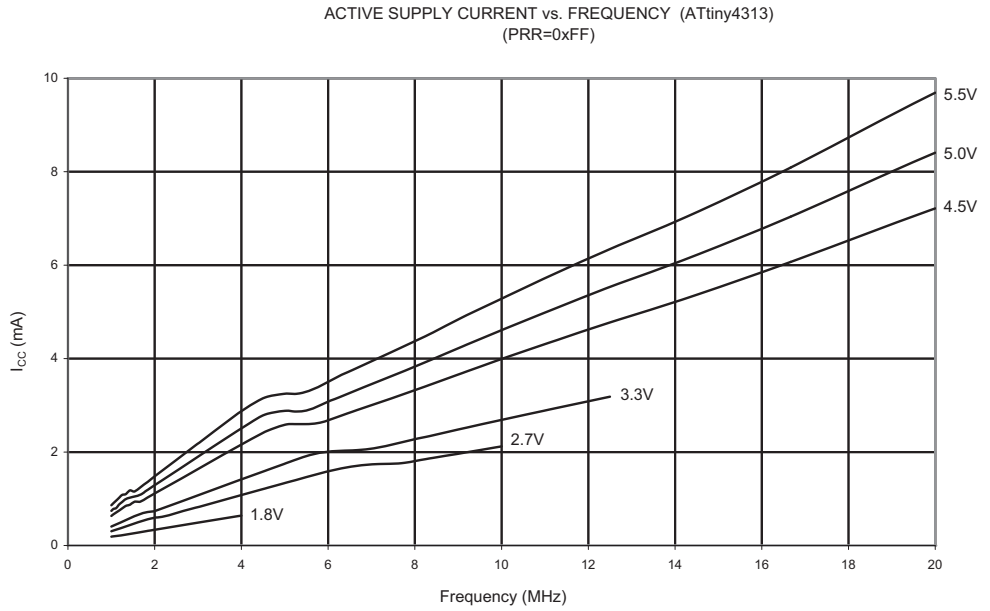


Figure 23-51. Active Supply Current vs. V_{CC} (Internal RC Oscillator, 8 MHz)

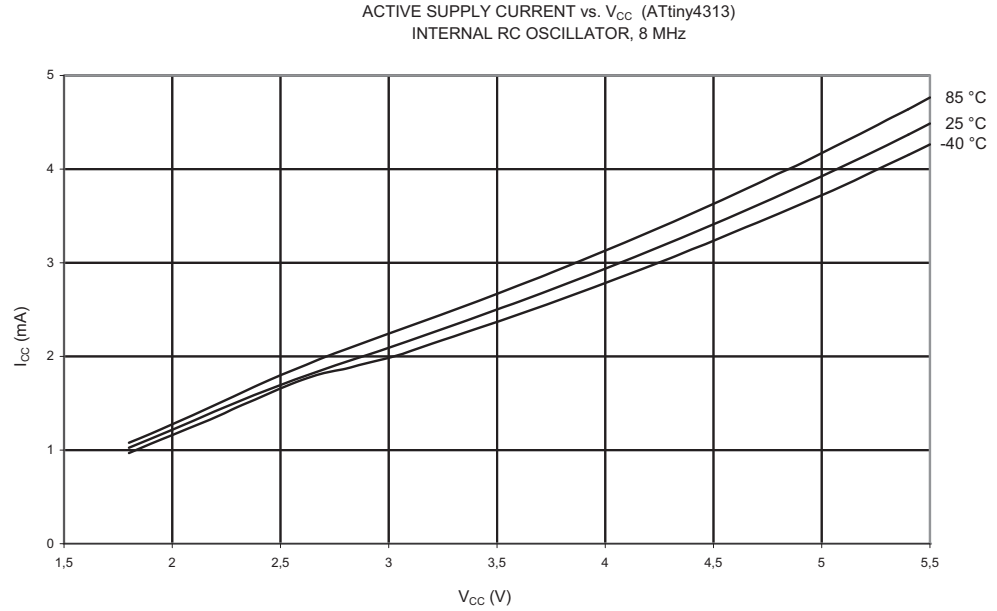


Figure 23-52. Active Supply Current vs. V_{CC} (Internal RC Oscillator, 1 MHz)

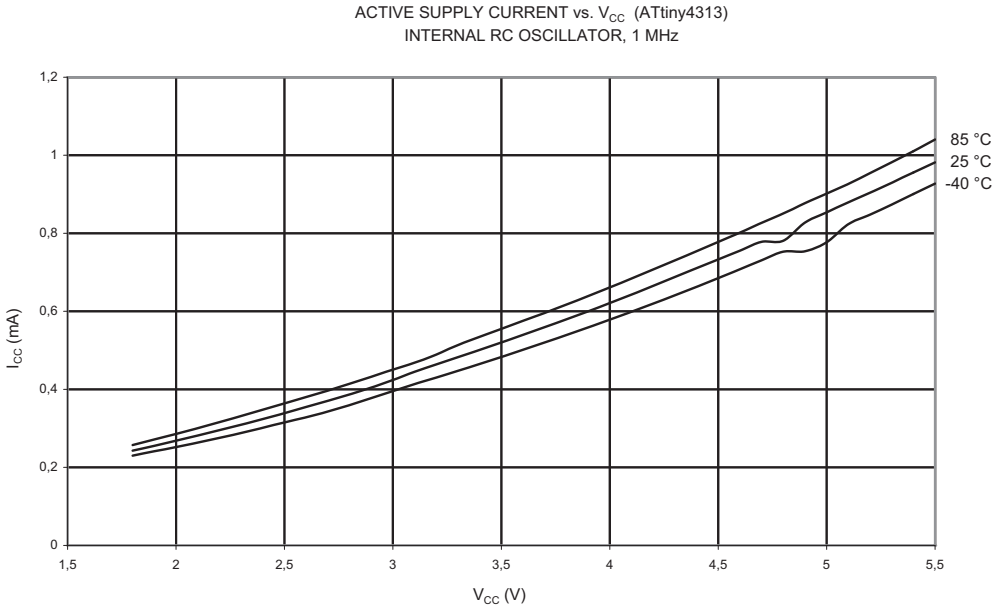
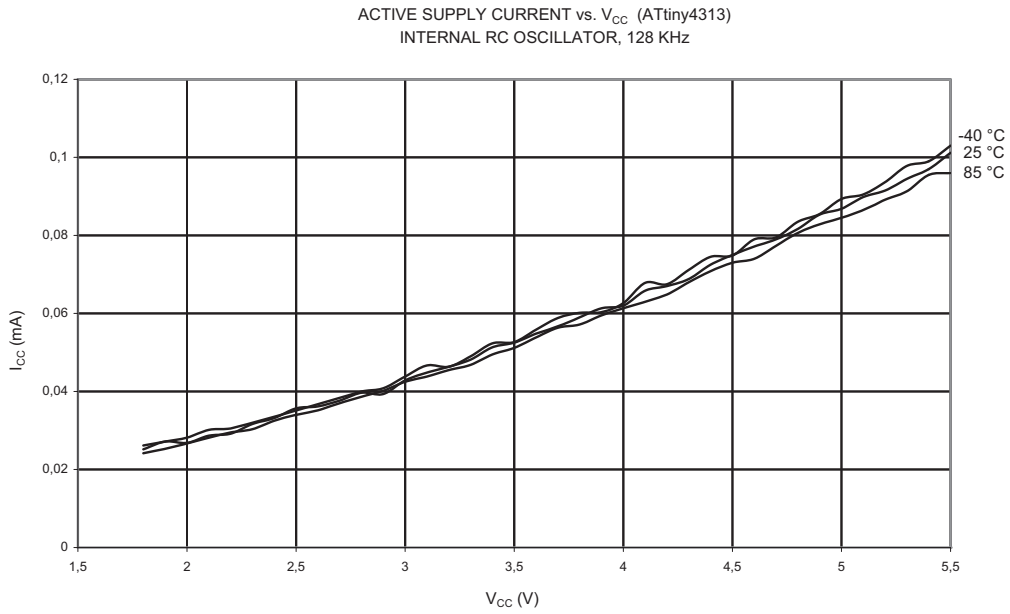


Figure 23-53. Active Supply Current vs. V_{CC} (Internal RC Oscillator, 128 KHz)



23.3.2 Current Consumption in Idle Mode

Figure 23-54. Idle Supply Current vs. Low Frequency (0.1 - 1.0 MHz)

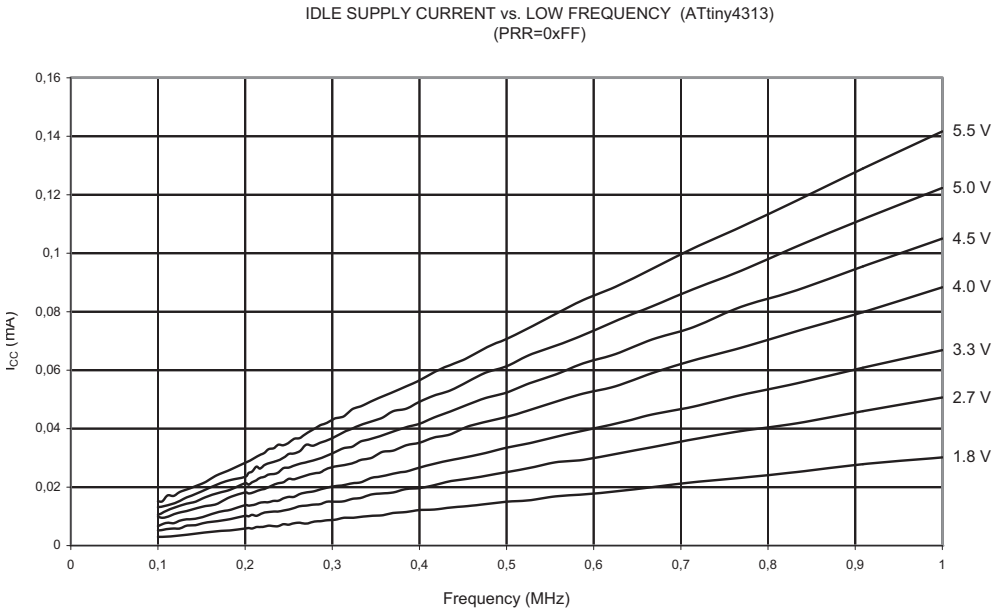


Figure 23-55. Idle Supply Current vs. Frequency (1 - 20 MHz)

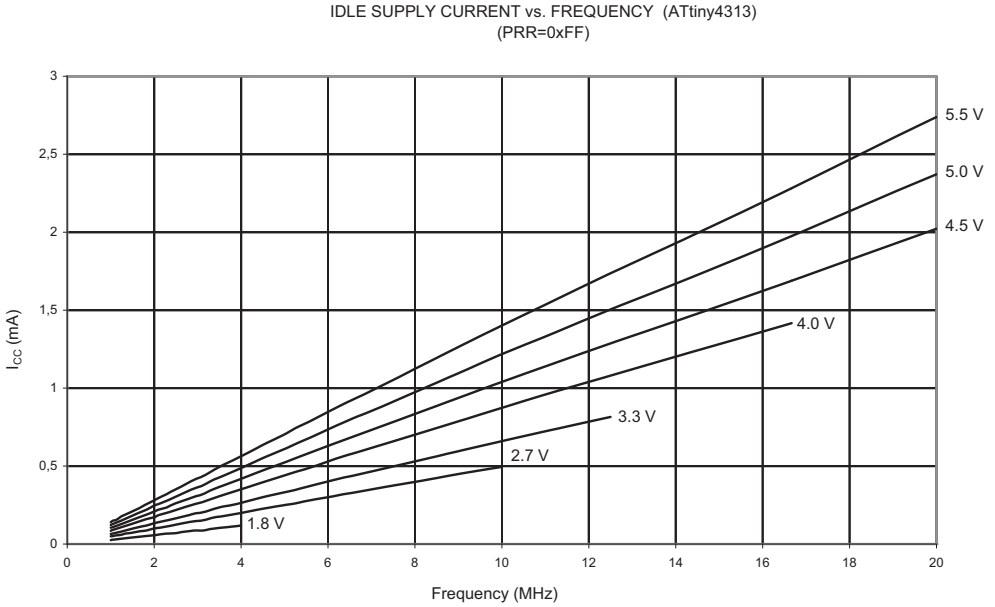


Figure 23-56. Idle Supply Current vs. V_{CC} (Internal RC Oscillator, 8 MHz)

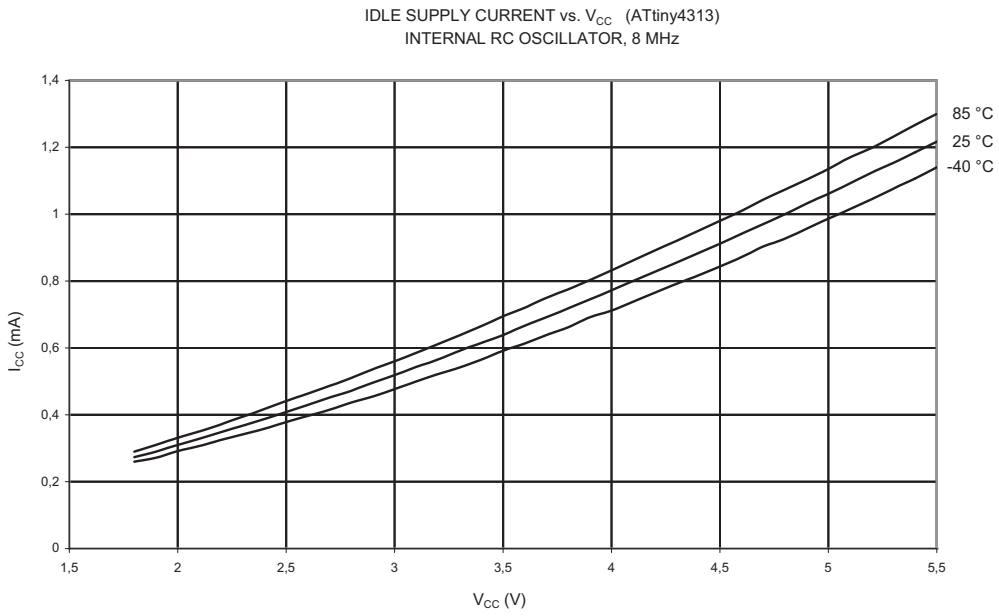


Figure 23-57. Idle Supply Current vs. V_{CC} (Internal RC Oscillator, 1 MHz)

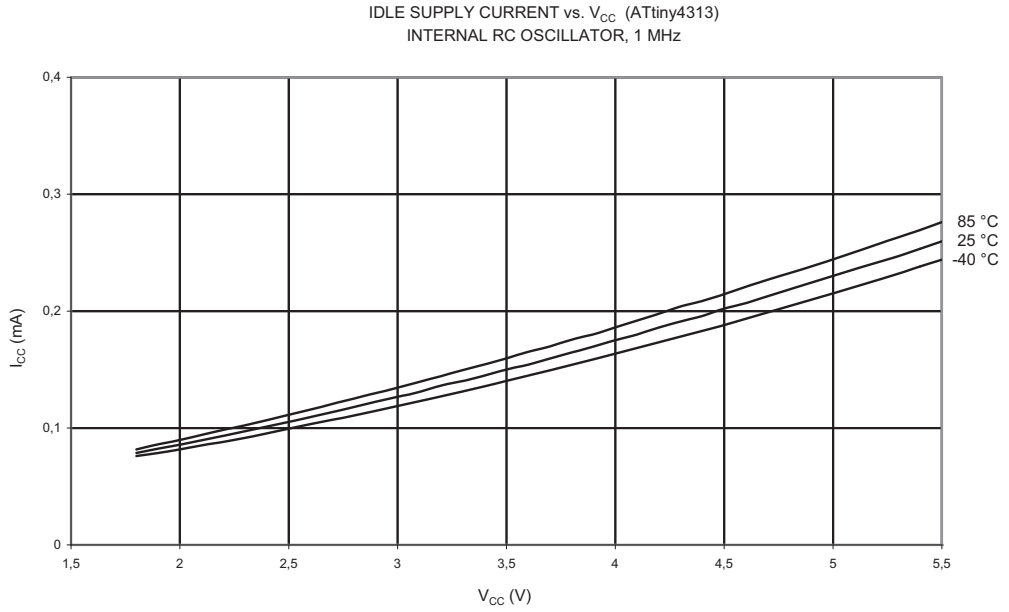
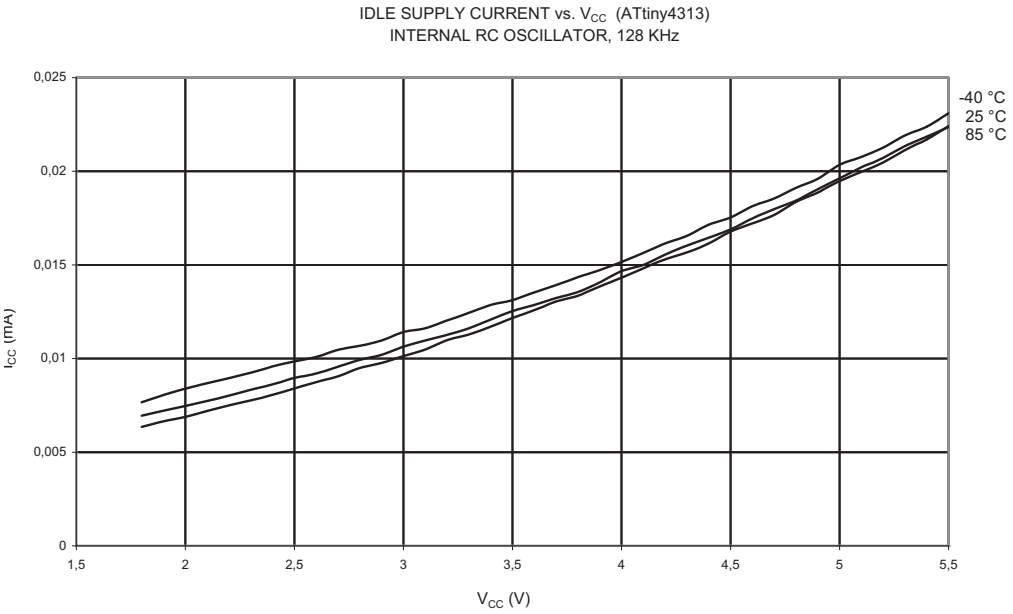


Figure 23-58. Idle Supply Current vs. V_{CC} (Internal RC Oscillator, 128 KHz)



23.3.3 Current Consumption in Power-down Mode

Figure 23-59. Power-down Supply Current vs. V_{CC} (Watchdog Timer Disabled)

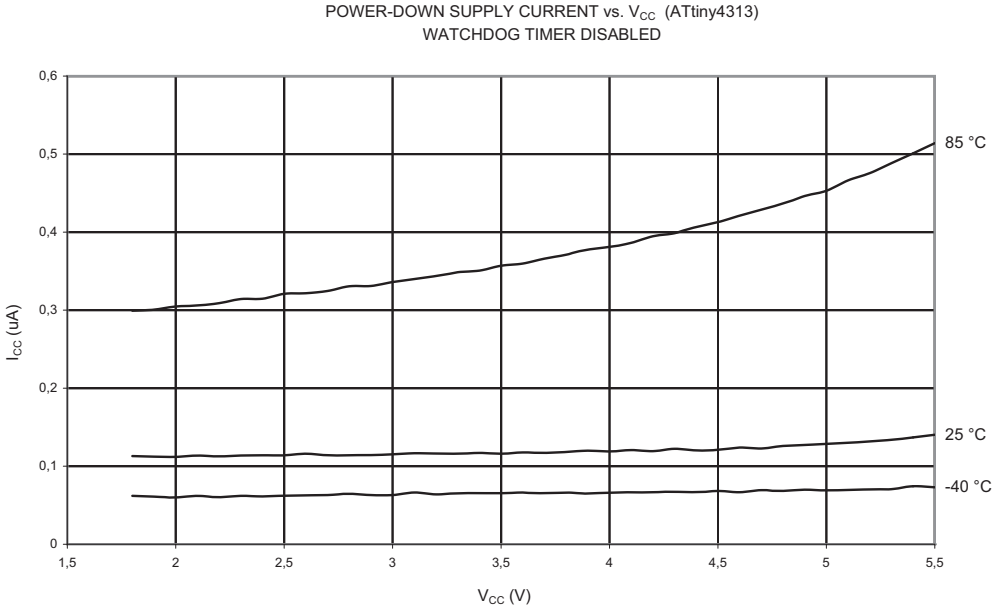
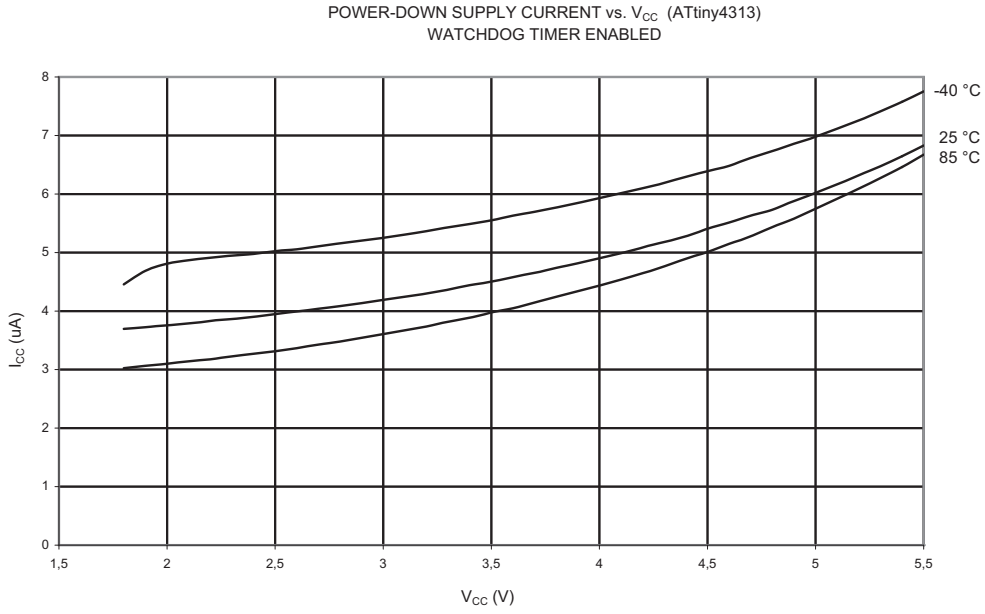


Figure 23-60. Power-down Supply Current vs. V_{CC} (Watchdog Timer Enabled)



23.3.4 Current Consumption in Reset

Figure 23-61. Reset Supply Current vs. V_{CC} (0.1 - 1.0 MHz, Excluding Current Through The Reset Pull-up)

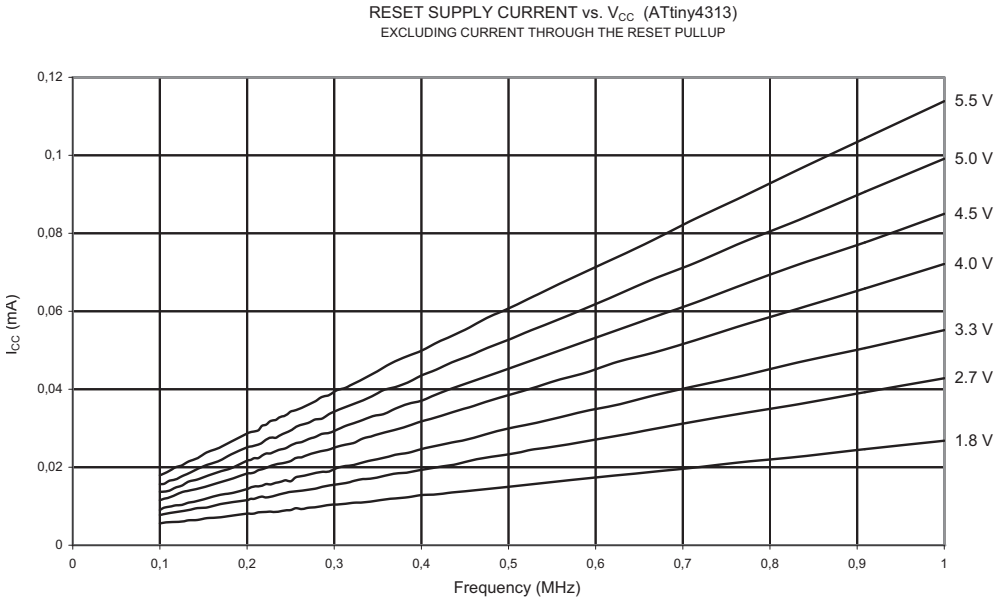
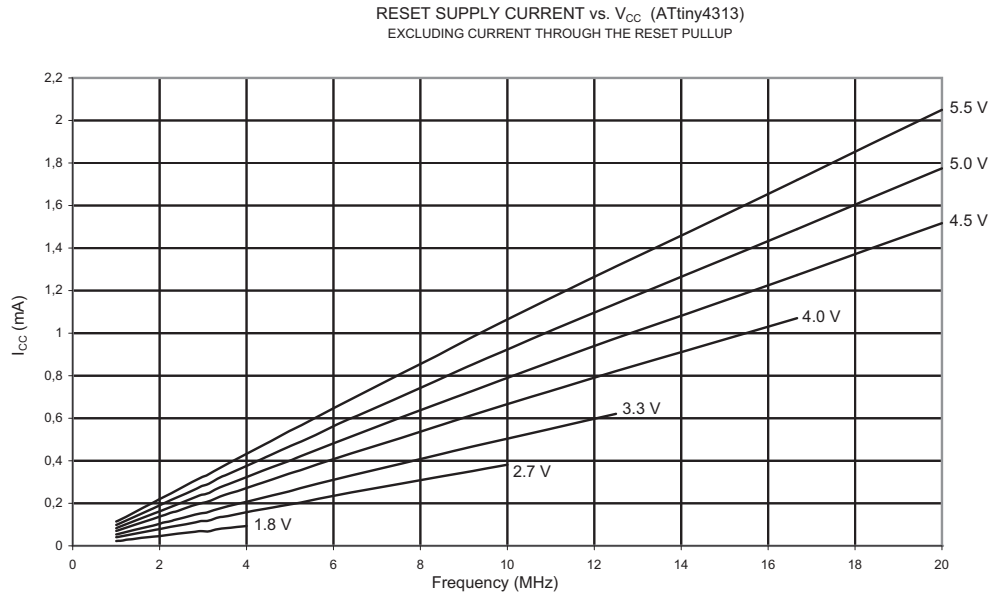


Figure 23-62. Reset Supply Current vs. V_{CC} (1 - 20 MHz, Excluding Current Through The Reset Pull-up)



23.3.5 Current Consumption of Peripheral Units

Figure 23-63. Brownout Detector Current vs. V_{CC}

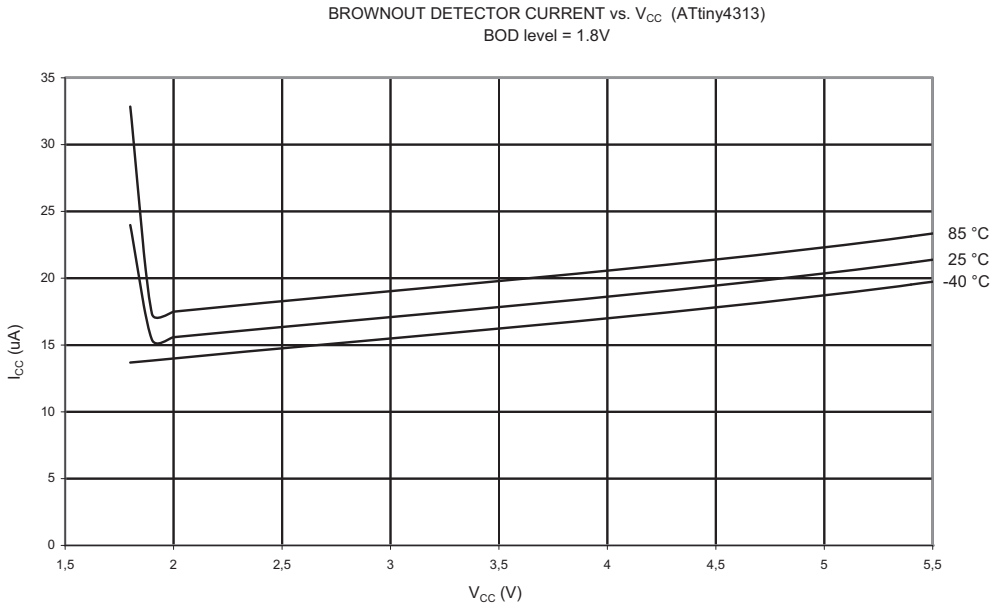
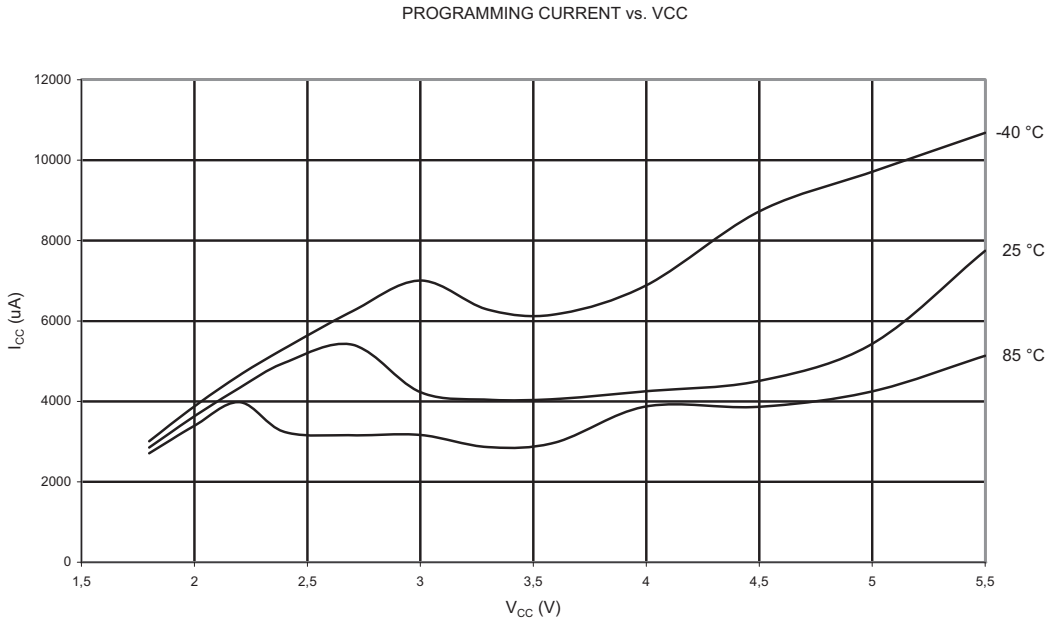


Figure 23-64. Programming Current vs. V_{CC} (ATtiny4313)



Note: Above programming current based on simulation and characterisation of similar device (ATtiny44A).

23.3.6 Pull-up Resistors

Figure 23-65. Pull-up Resistor Current vs. Input Voltage (I/O Pin, V_{CC} = 1.8V)

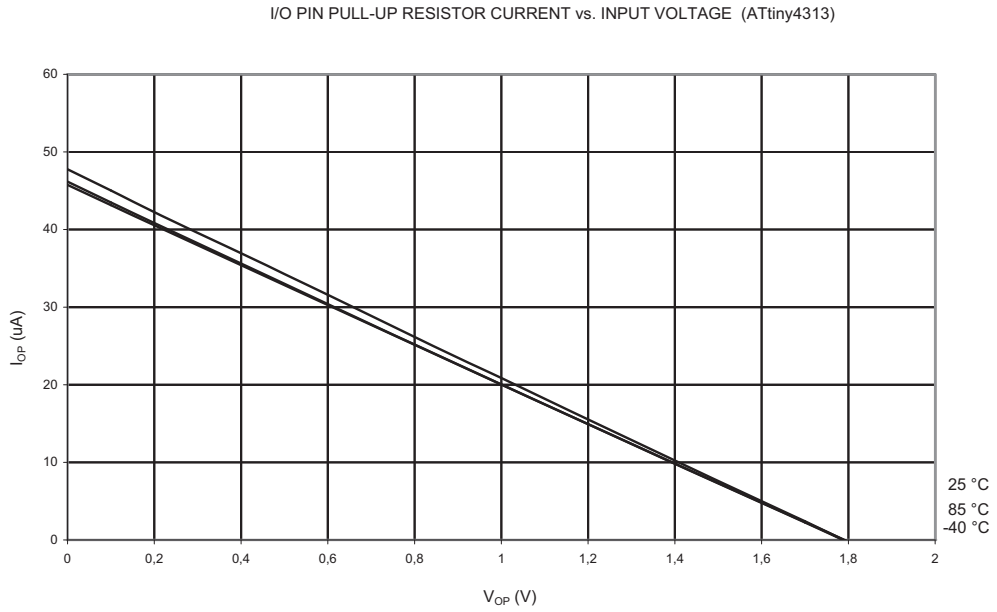


Figure 23-66. Pull-up Resistor Current vs. Input Voltage (I/O Pin, V_{CC} = 2.7V)

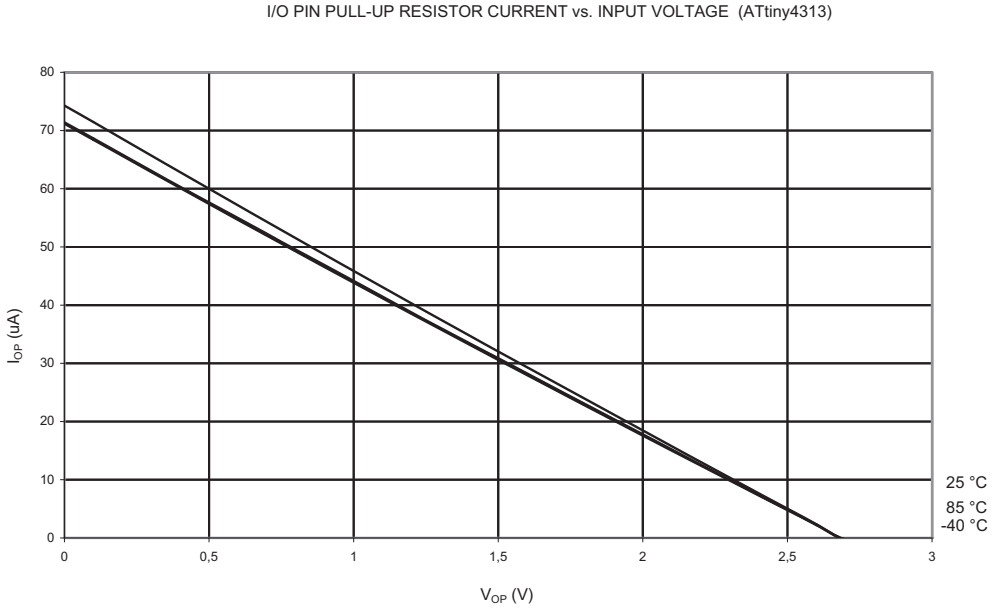


Figure 23-67. Pull-up Resistor Current vs. Input Voltage (I/O Pin, V_{CC} = 5V)

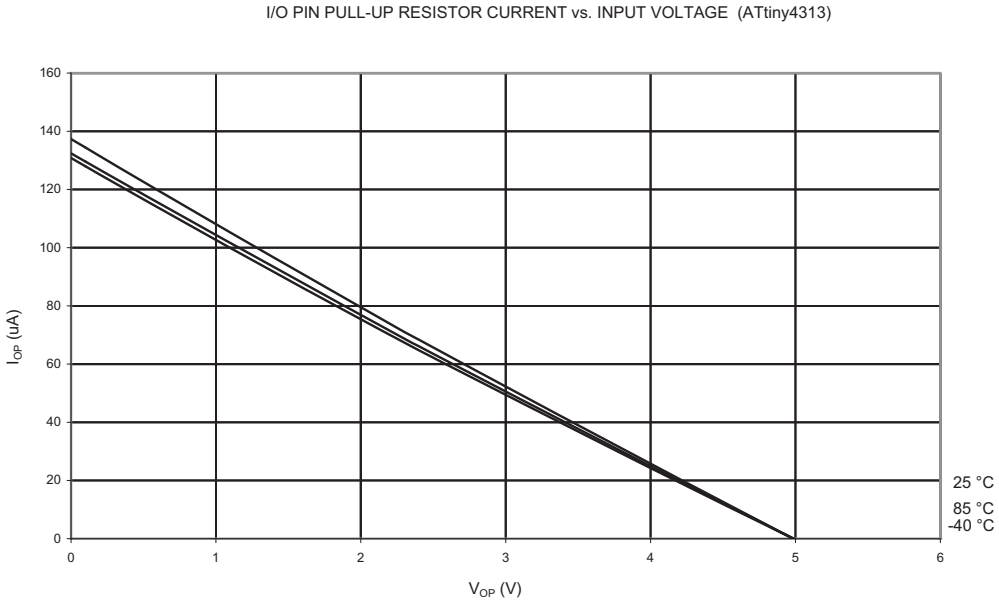


Figure 23-68. Reset Pull-up Resistor Current vs. Reset Pin Voltage (V_{CC} = 1.8V)

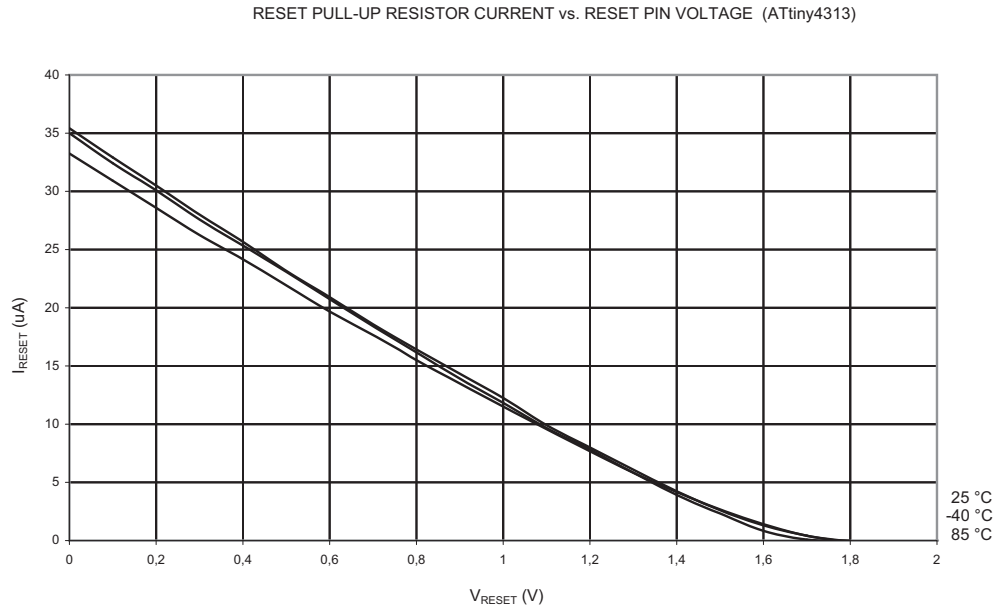


Figure 23-69. Reset Pull-up Resistor Current vs. Reset Pin Voltage (V_{CC} = 2.7V)

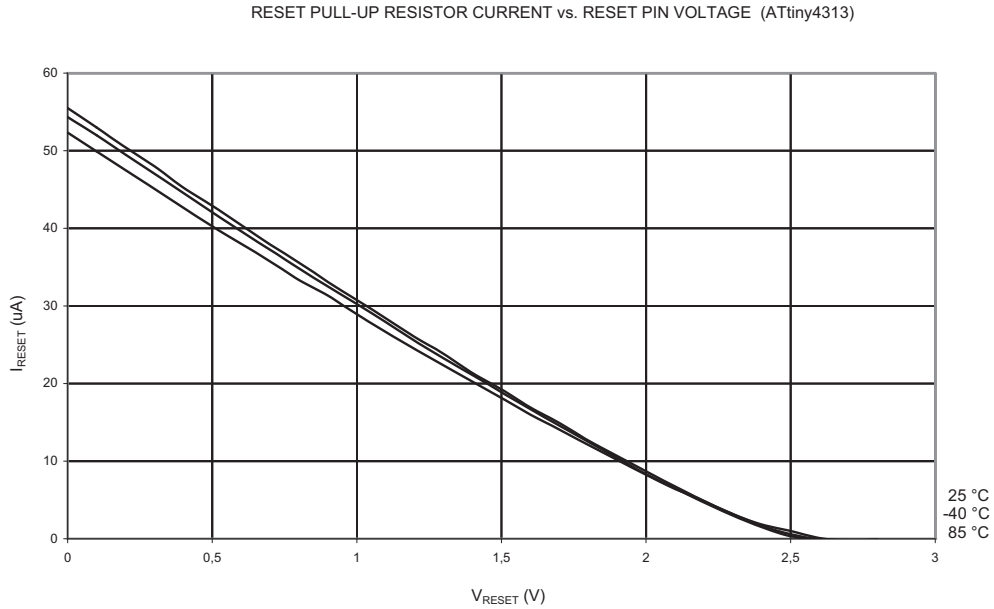
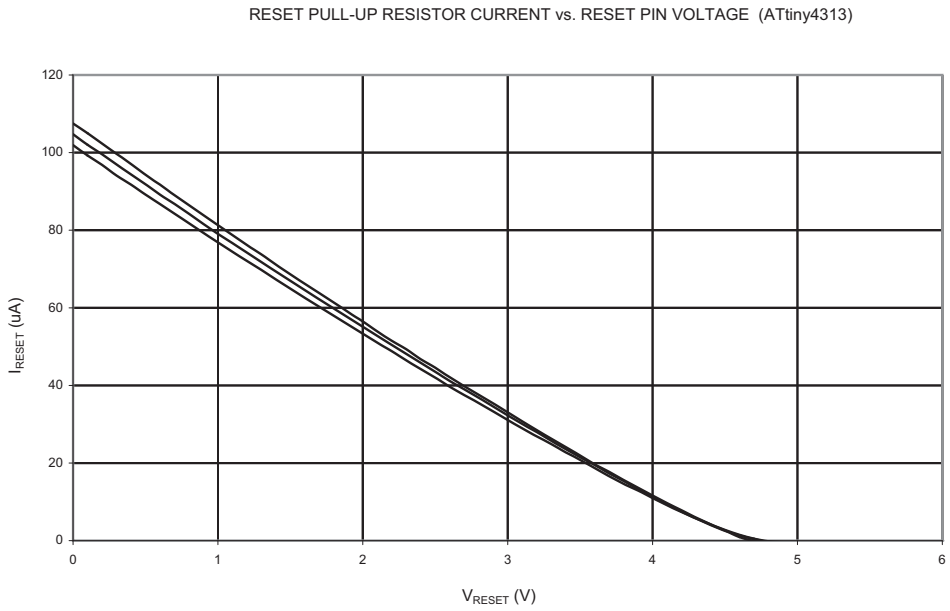


Figure 23-70. Reset Pull-up Resistor Current vs. Reset Pin Voltage (V_{CC} = 5V)



23.3.7 Output Driver Strength

Figure 23-71. V_{OL}: Output Voltage vs. Sink Current (I/O Pin, V_{CC} = 1.8V)

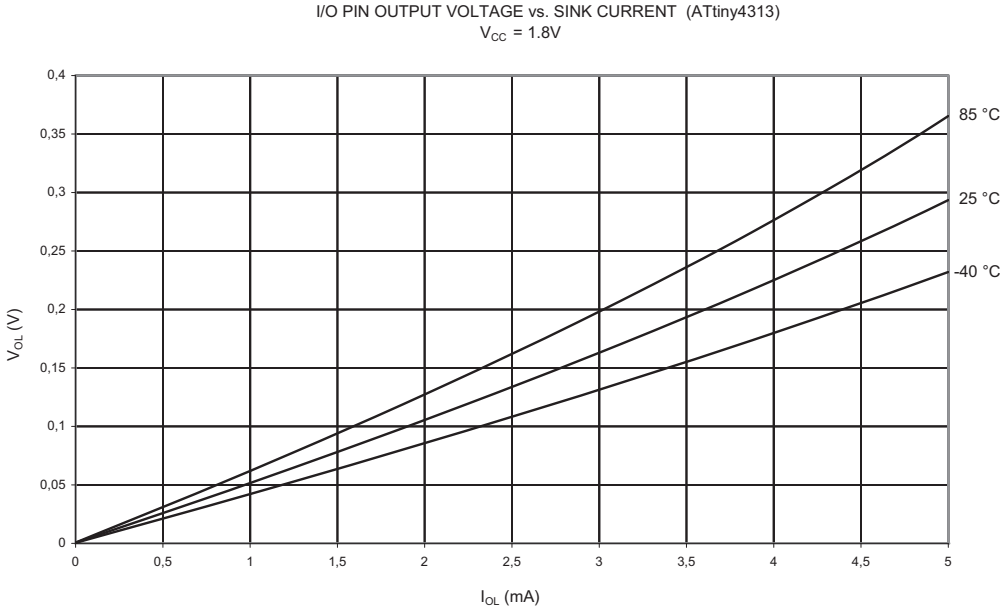


Figure 23-72. V_{OL}: Output Voltage vs. Sink Current (I/O Pin, V_{CC} = 3V)

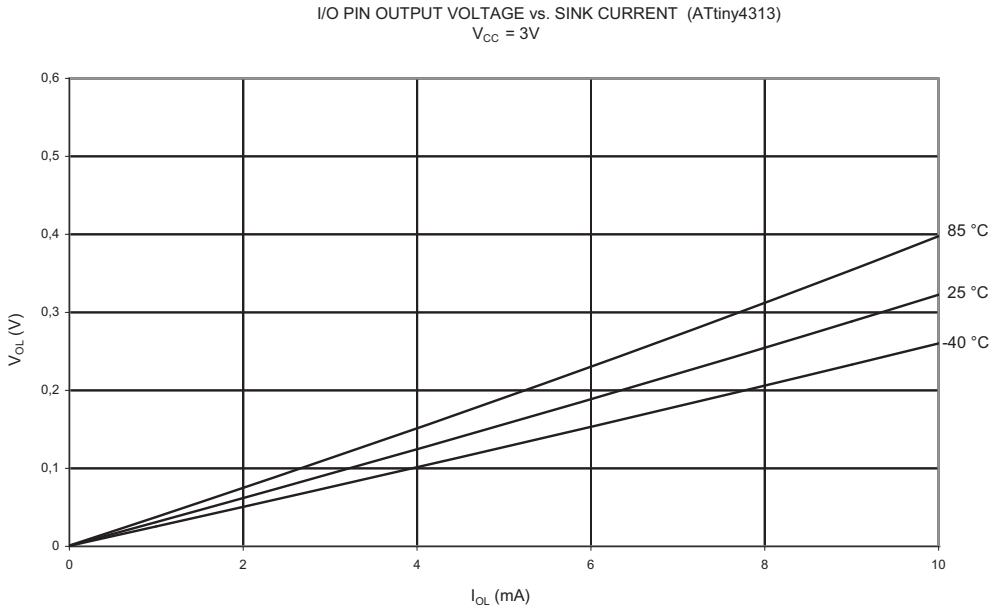


Figure 23-73. V_{OL}: Output Voltage vs. Sink Current (I/O Pin, V_{CC} = 5V)

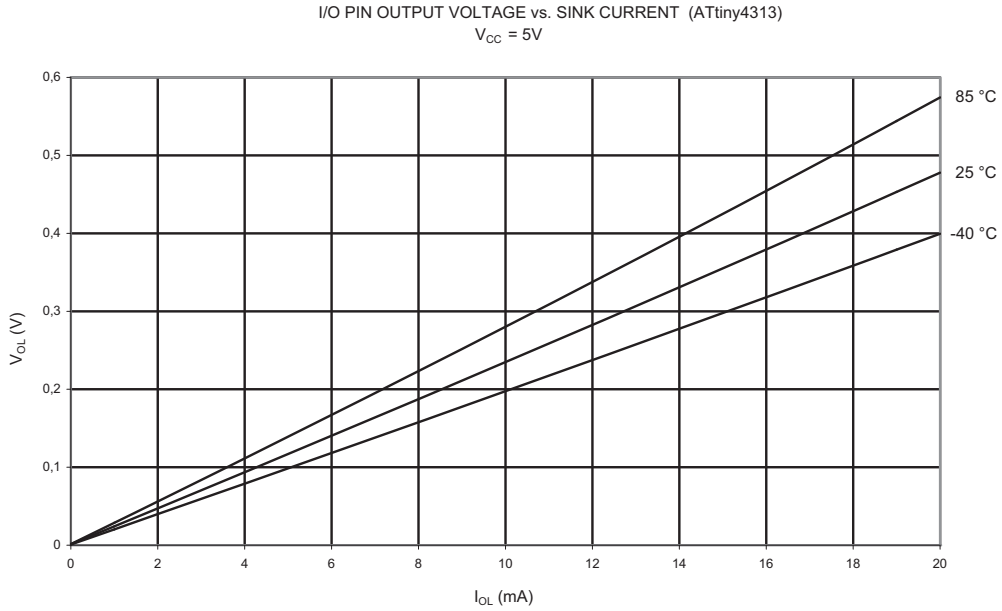


Figure 23-74. V_{OH}: Output Voltage vs. Source Current (I/O Pin, V_{CC} = 1.8V)

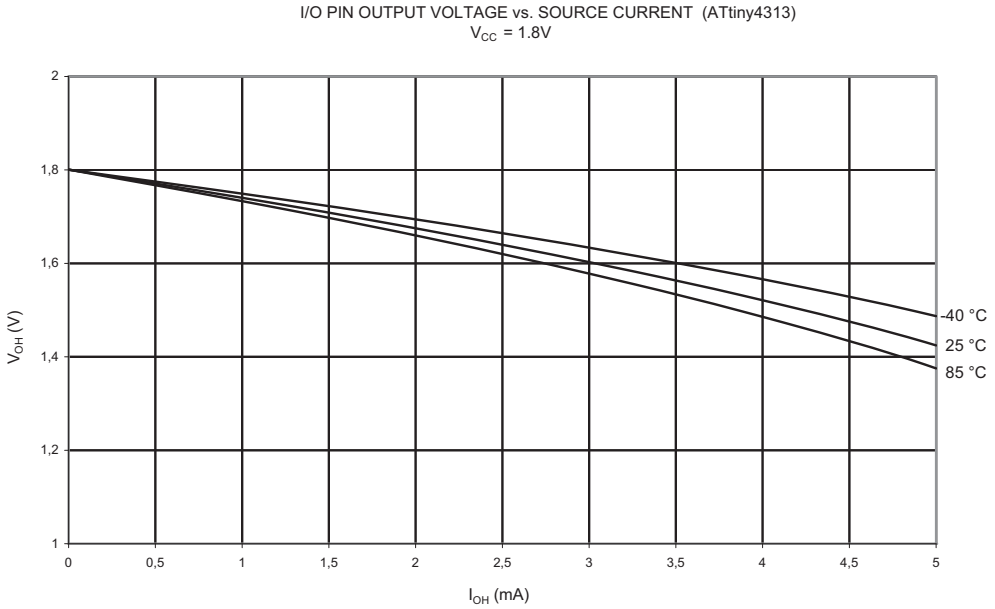


Figure 23-75. V_{OH}: Output Voltage vs. Source Current (I/O Pin, V_{CC} = 3V)

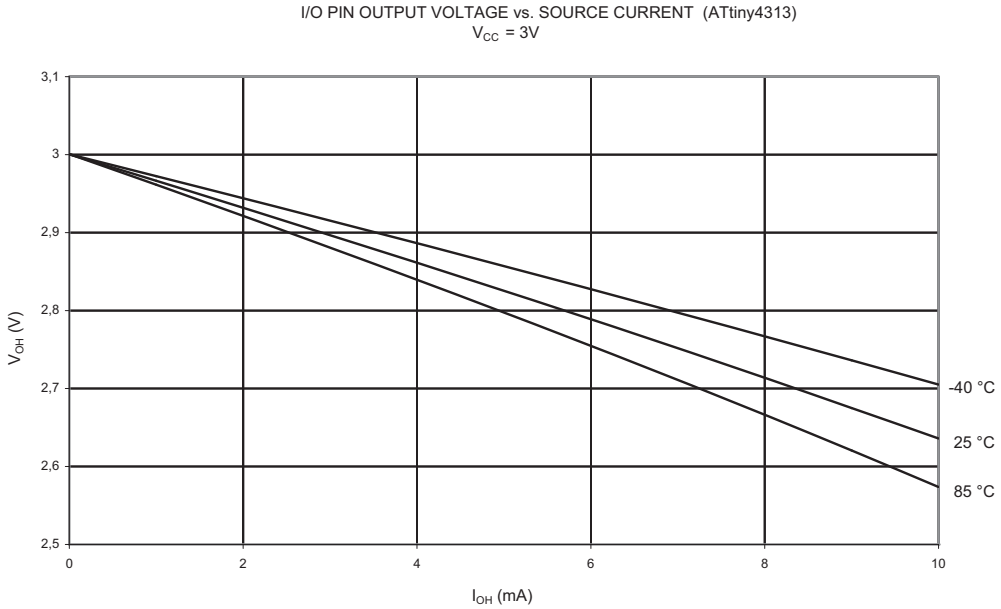


Figure 23-76. V_{OH}: Output Voltage vs. Source Current (I/O Pin, V_{CC} = 5V)

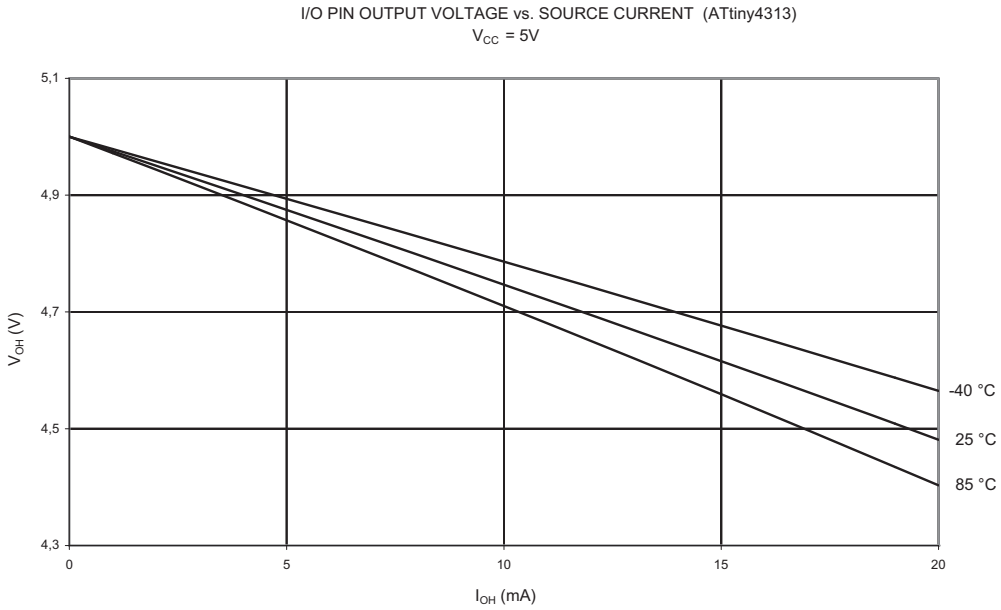


Figure 23-77. V_{OL}: Output Voltage vs. Sink Current (Reset Pin as I/O, T = 25°C)

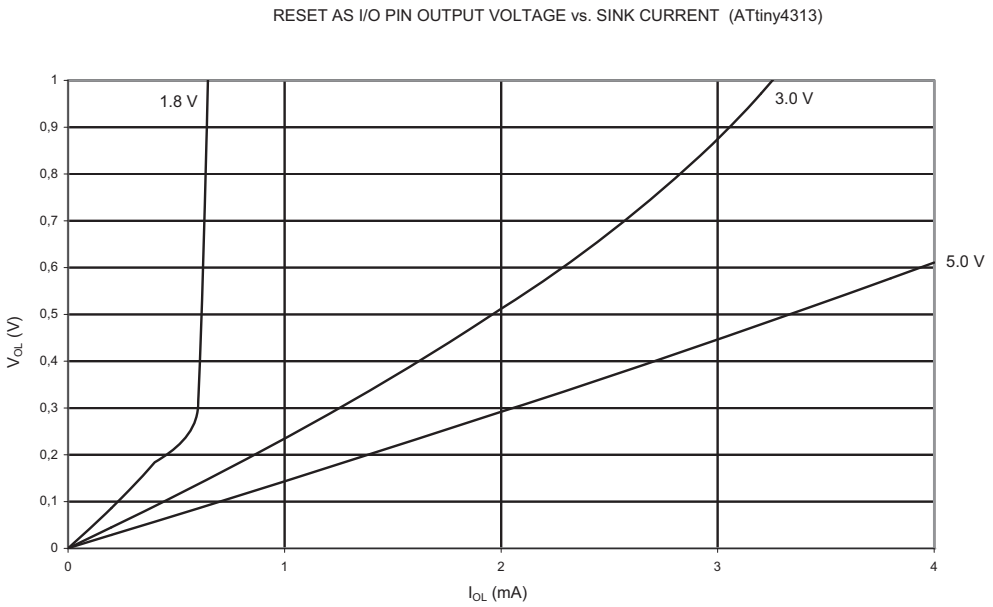
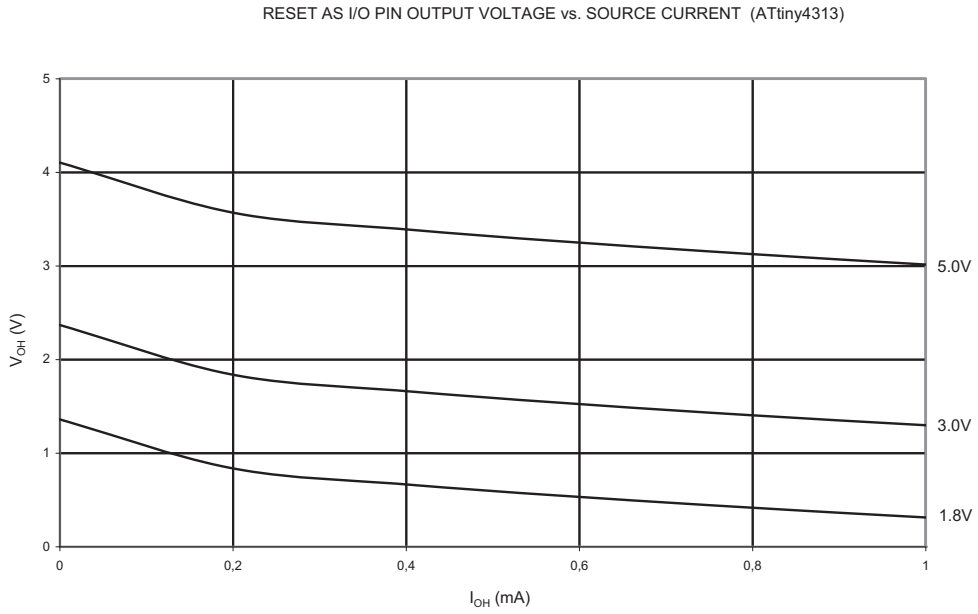


Figure 23-78. V_{OH}: Output Voltage vs. Source Current (Reset Pin as I/O, T = 25°C)



23.3.8 Input Thresholds and Hysteresis (for I/O Ports)

Figure 23-79. V_{IH}: Input Threshold Voltage vs. V_{CC} (I/O Pin Read as ‘1’)

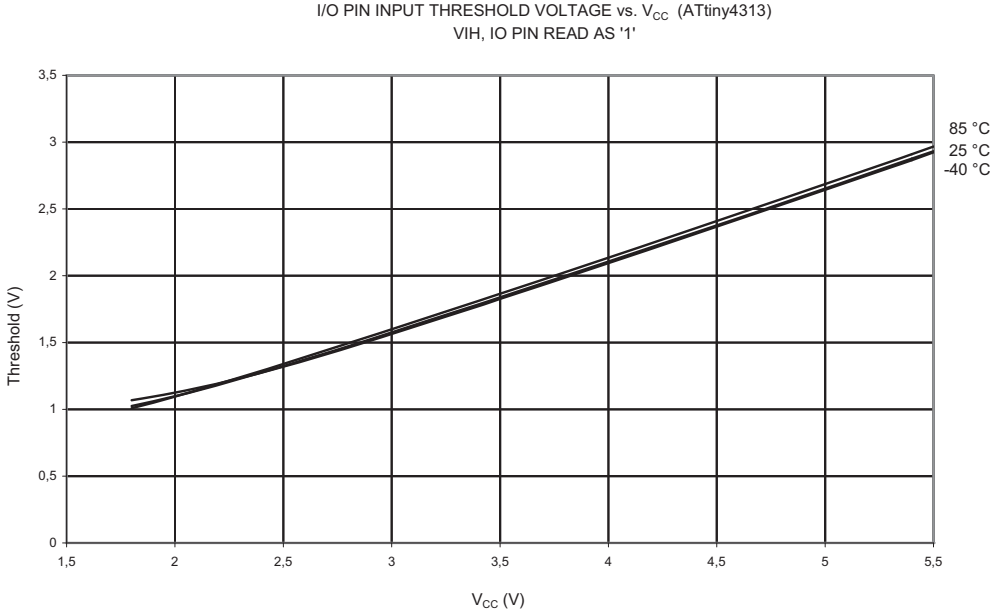


Figure 23-80. V_{IL}: Input Threshold Voltage vs. V_{CC} (I/O Pin, Read as ‘0’)

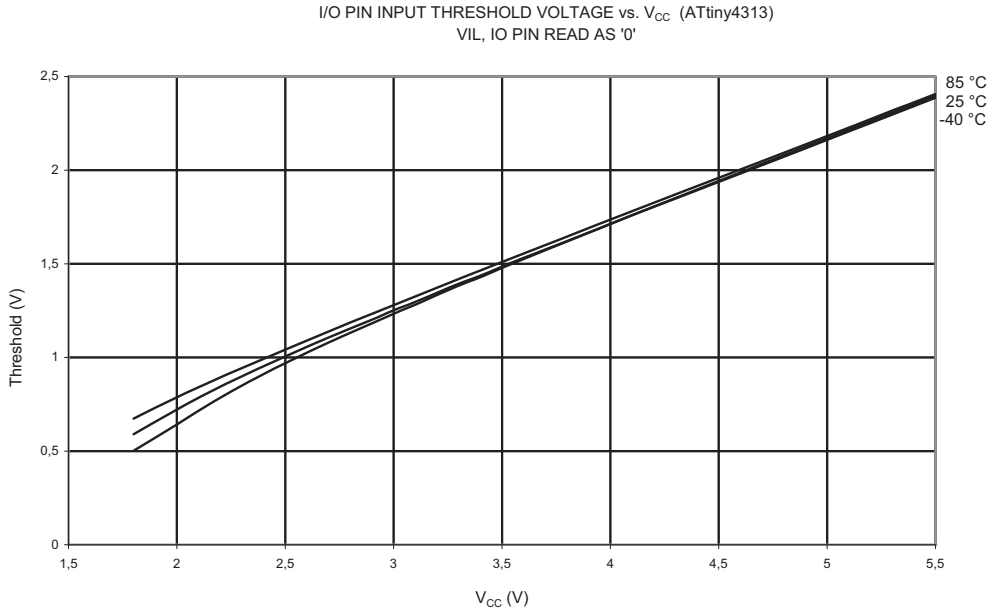


Figure 23-81. V_{IH}-V_{IL}: Input Hysteresis vs. V_{CC} (I/O Pin)

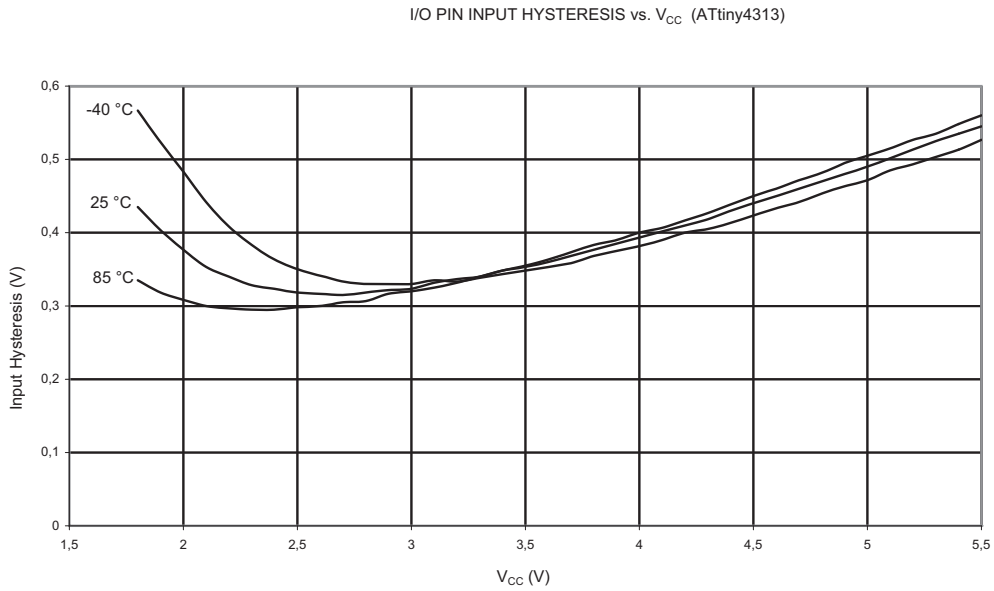


Figure 23-82. V_{IH}: Input Threshold Voltage vs. V_{CC} (Reset Pin as I/O, Read as ‘1’)

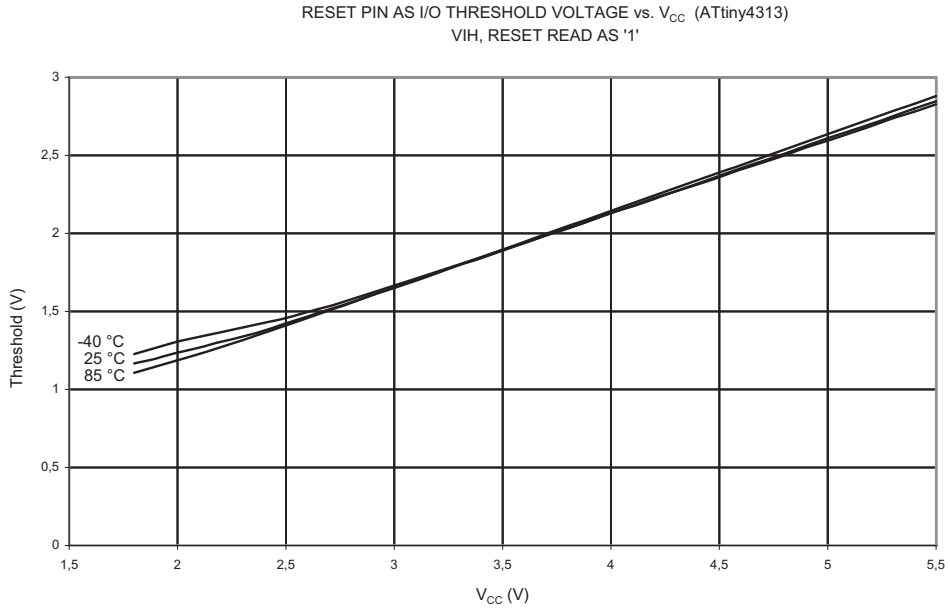


Figure 23-83. V_{IL}: Input Threshold Voltage vs. V_{CC} (Reset Pin as I/O, Read as ‘0’)

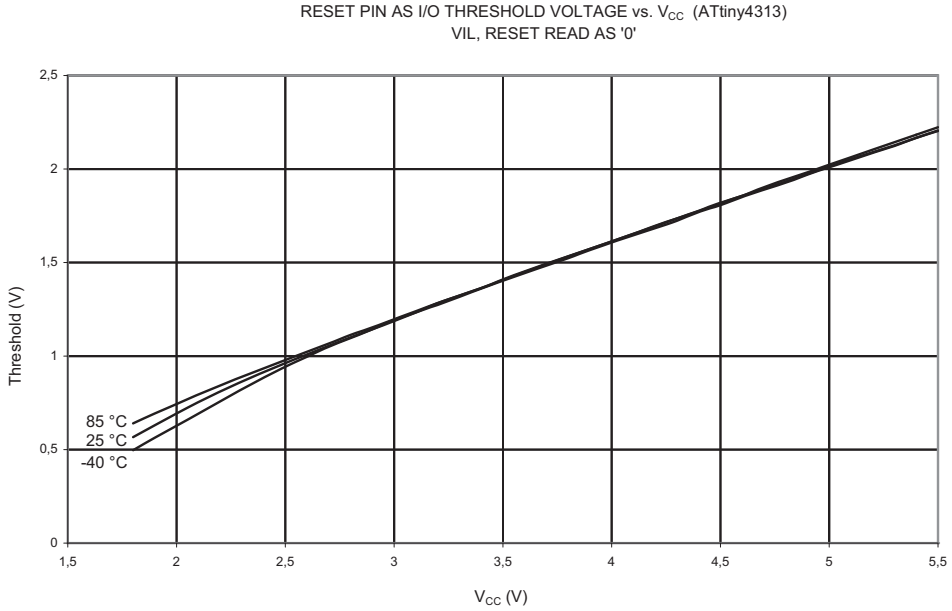
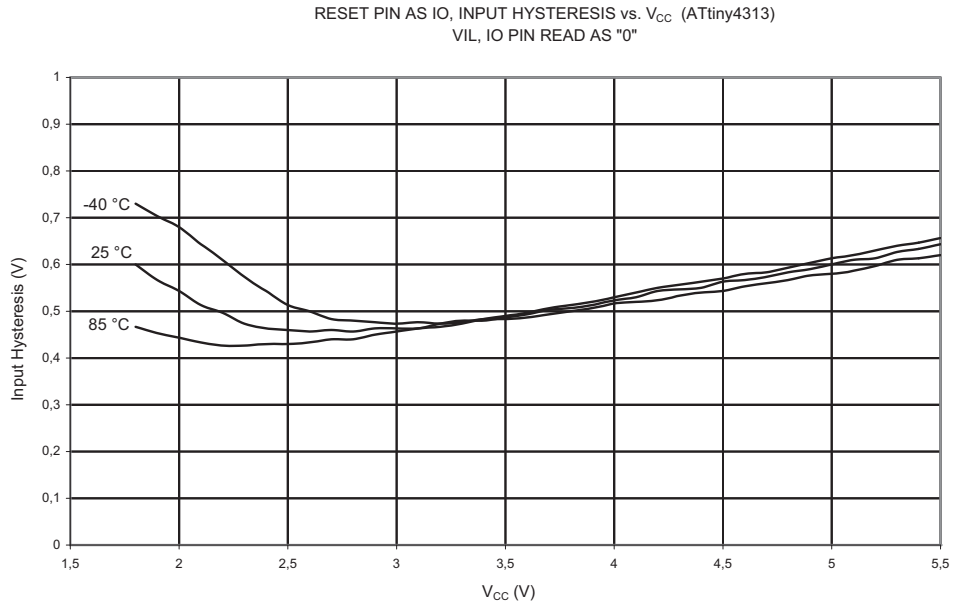


Figure 23-84. V_{IH}-V_{IL}: Input Hysteresis vs. V_{CC} (Reset Pin as I/O)



23.3.9 BOD, Bandgap and Reset

Figure 23-85. BOD Thresholds vs. Temperature (BOD Level is 4.3V)

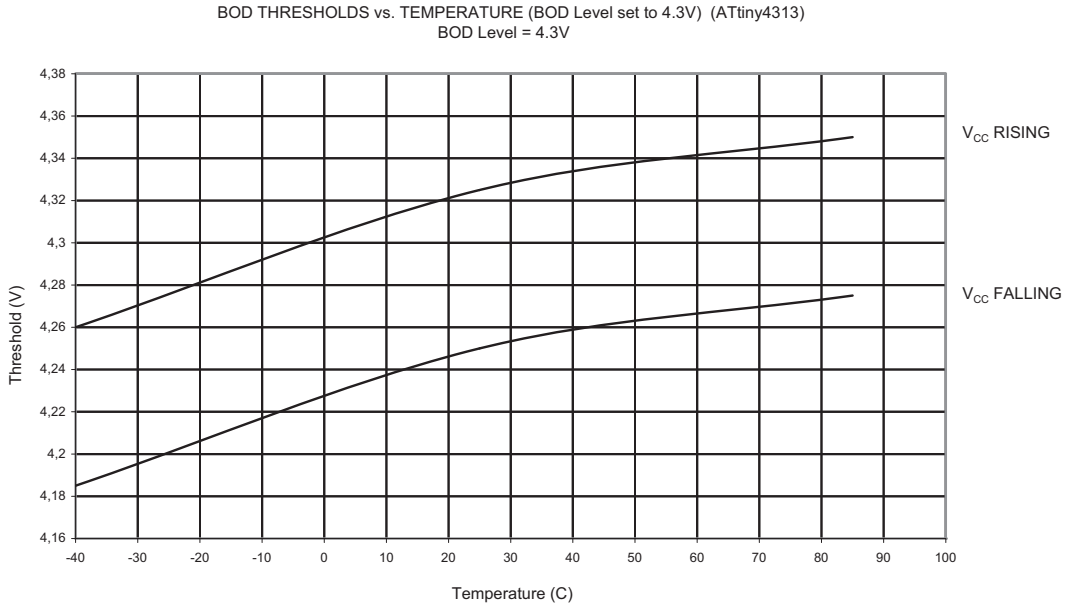


Figure 23-86. BOD Thresholds vs. Temperature (BOD Level is 2.7V)

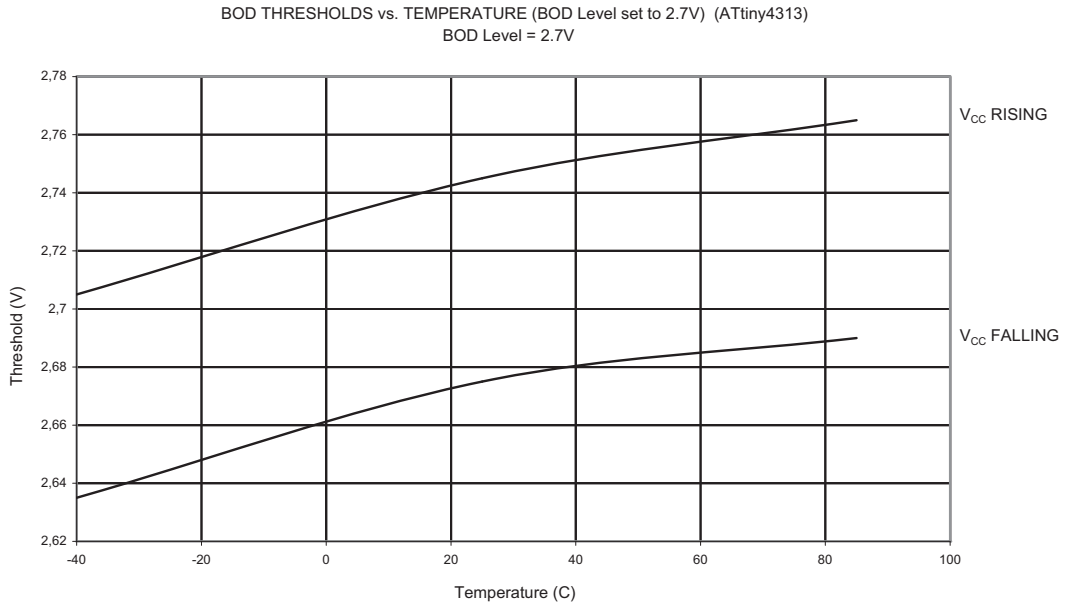


Figure 23-87. BOD Thresholds vs. Temperature (BOD Level is 1.8V)

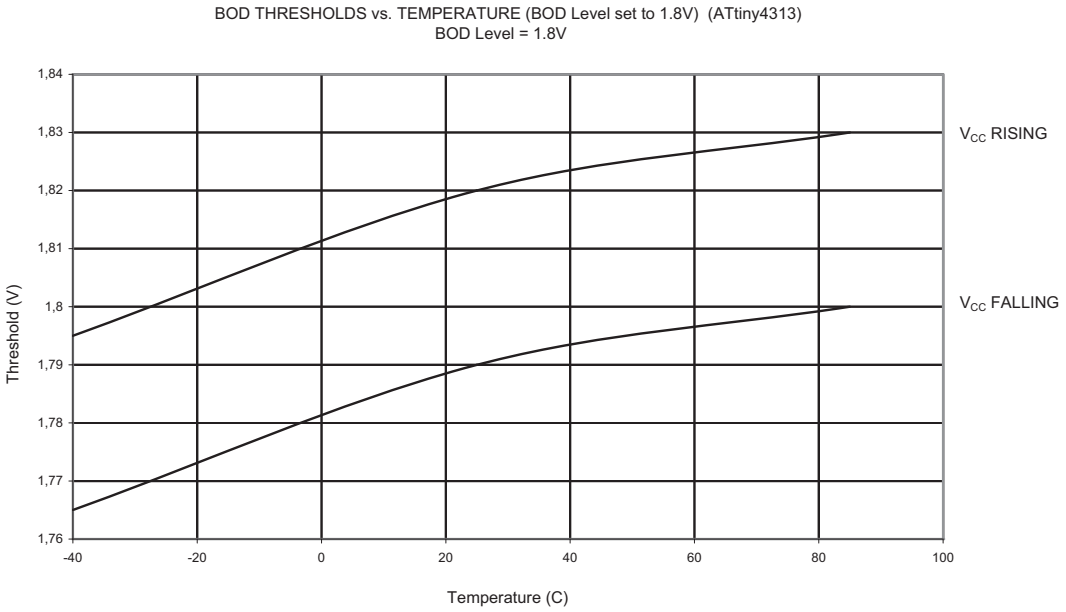


Figure 23-88. Bandgap Voltage vs. Supply Voltage

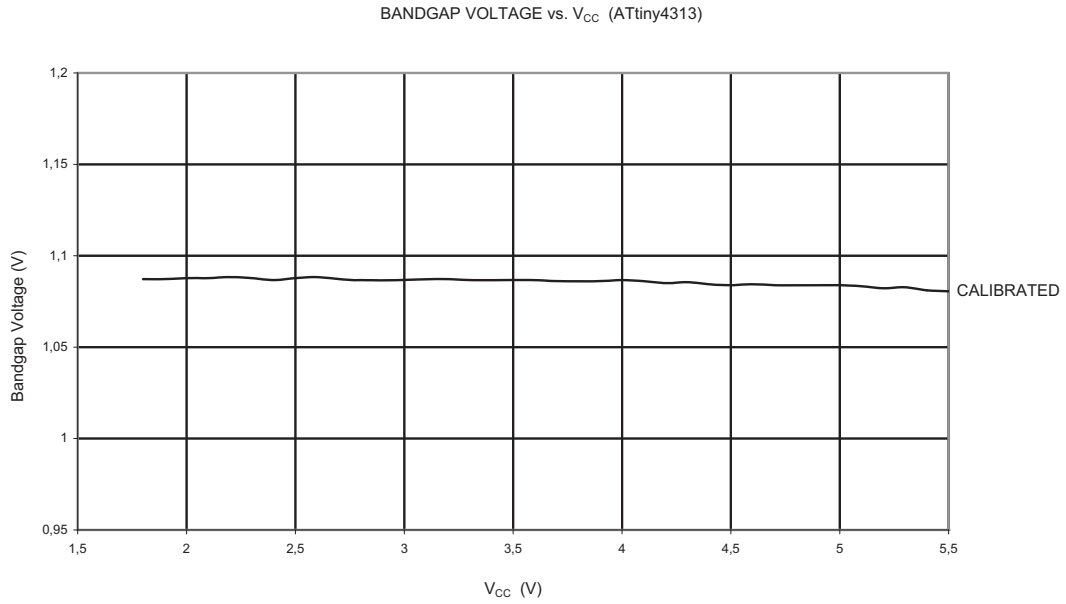


Figure 23-89. Bandgap Voltage vs. Temperature

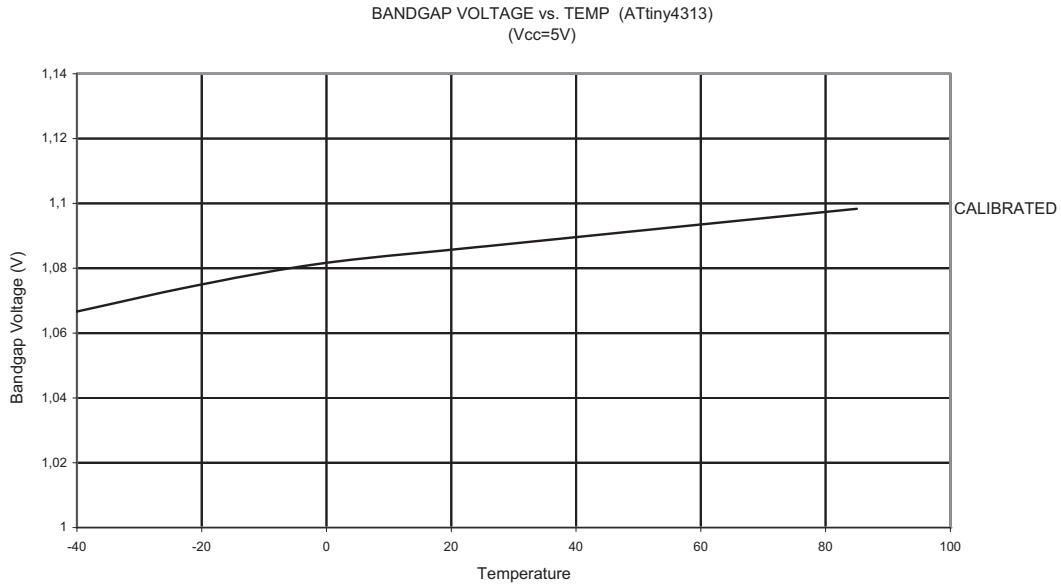


Figure 23-90. V_{IH}: Input Threshold Voltage vs. V_{CC} (Reset Pin, Read as ‘1’)

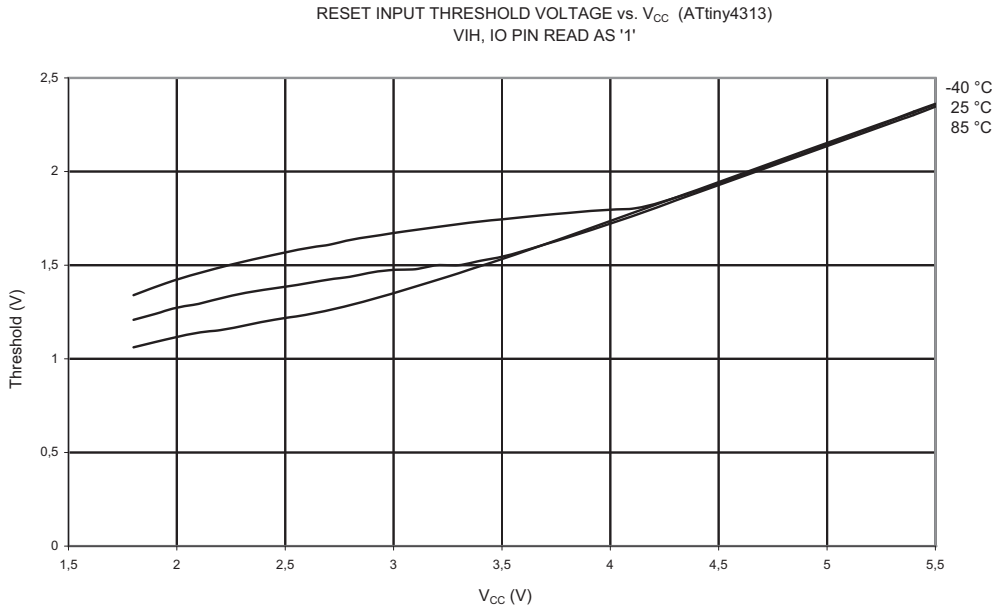


Figure 23-91. V_{IL}: Input Threshold Voltage vs. V_{CC} (Reset Pin, Read as ‘0’)

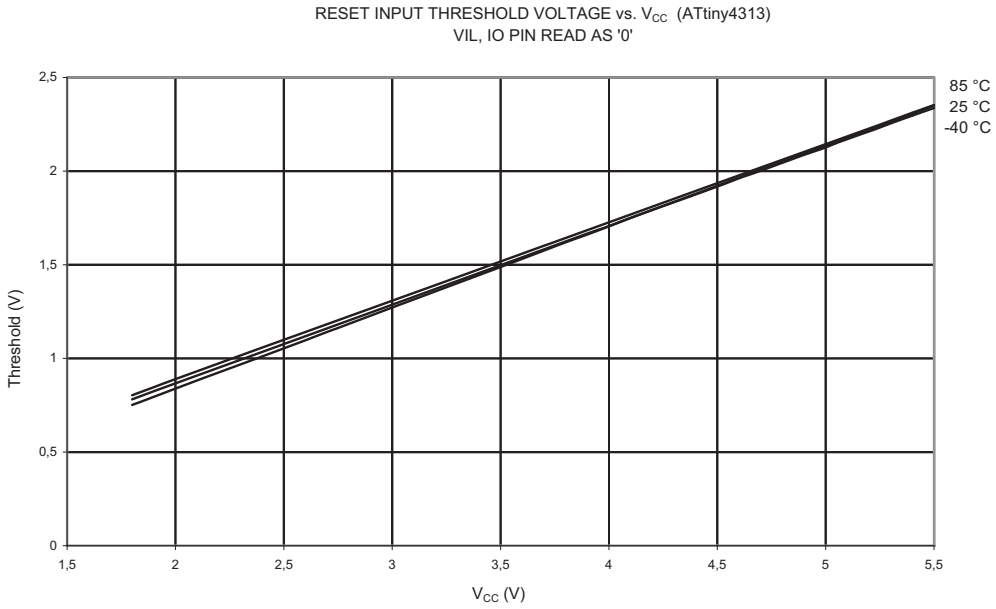


Figure 23-92. V_{IH}-V_{IL}: Input Hysteresis vs. V_{CC} (Reset Pin)

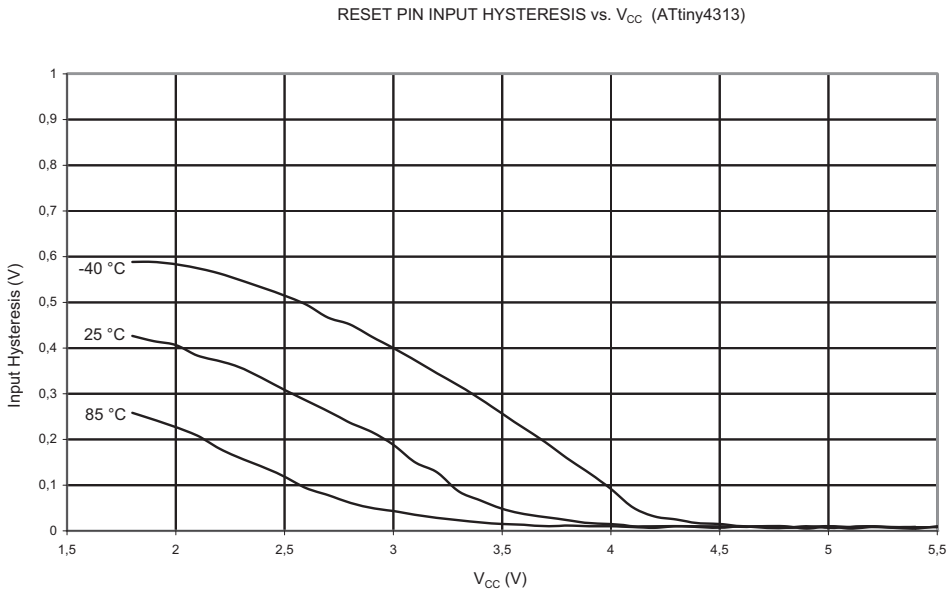
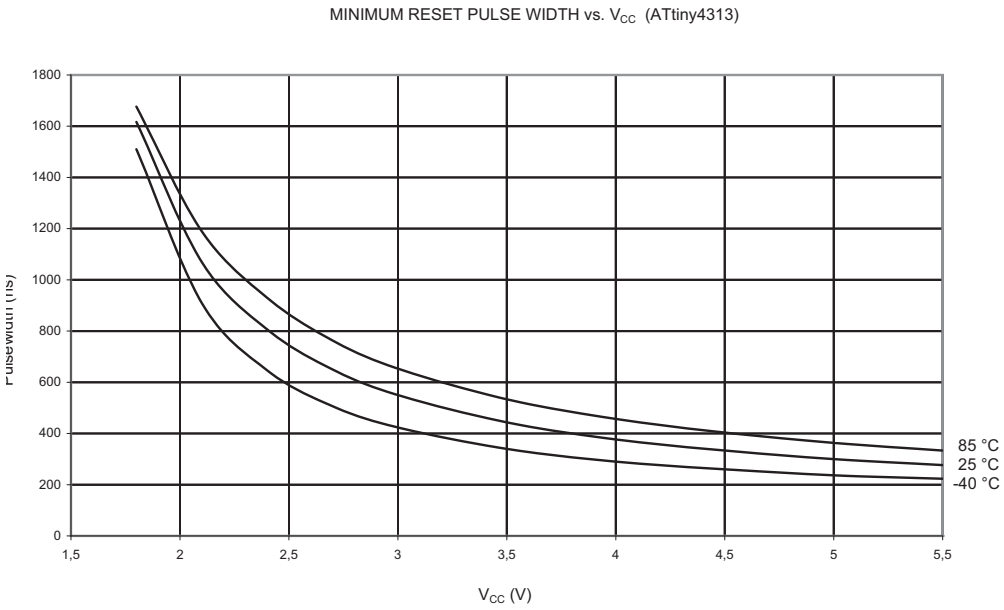


Figure 23-93. Minimum Reset Pulse Width vs. V_{CC}



23.3.10 Internal Oscillator Speed

Figure 23-94. Calibrated 8 MHz RC Oscillator Frequency vs. V_{CC}

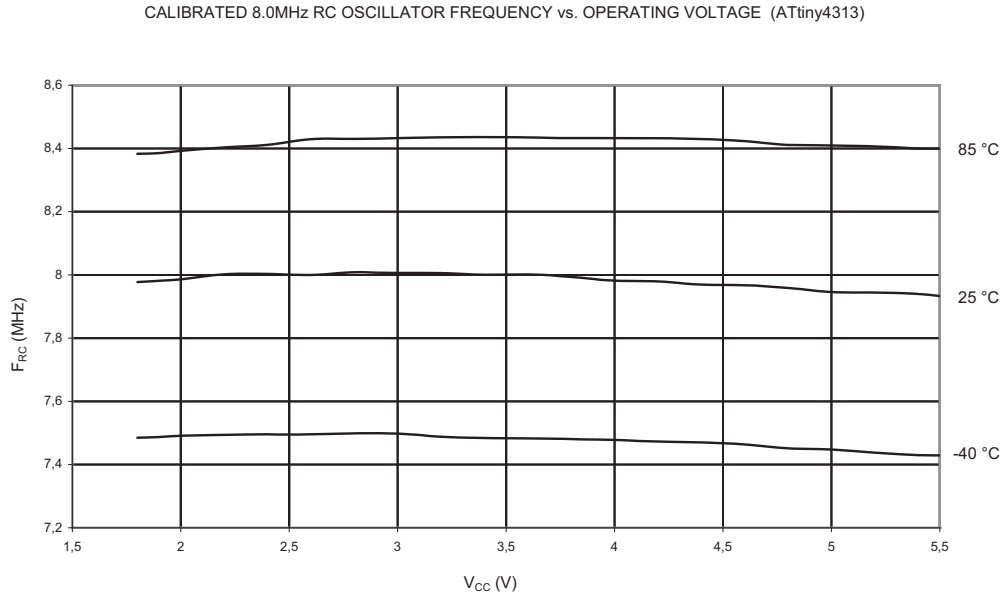


Figure 23-95. Calibrated 8 MHz RC Oscillator Frequency vs. Temperature

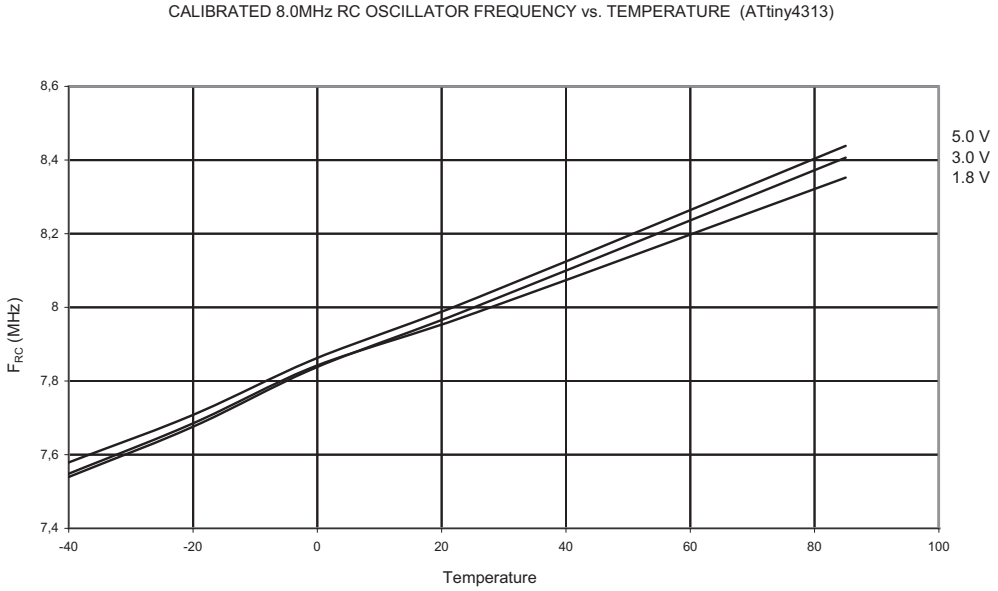
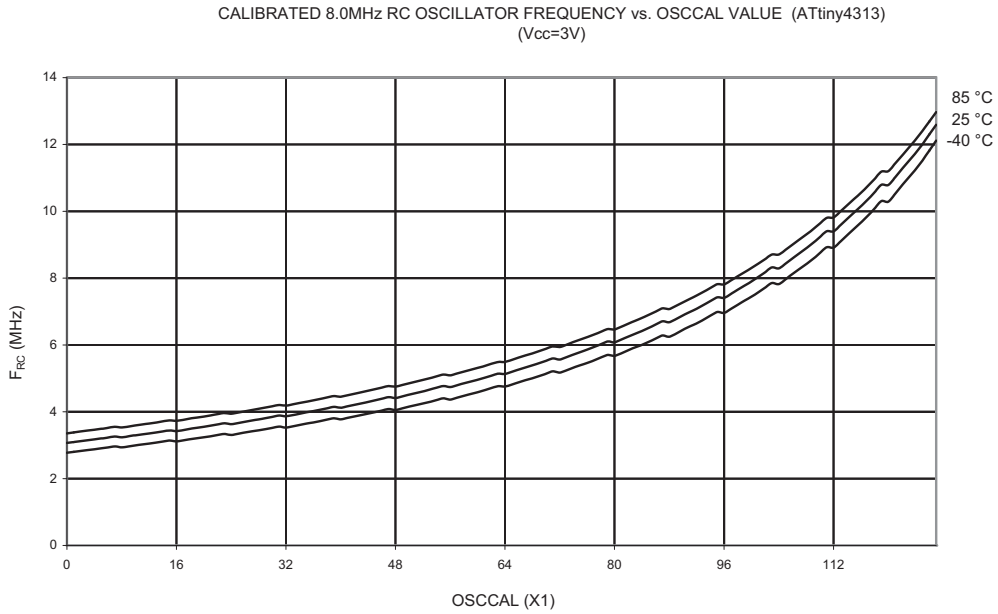


Figure 23-96. Calibrated 8 MHz RC Oscillator Frequency vs. Osccal Value



24. Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
0x3F (0x5F)	SREG	I	T	H	S	V	N	Z	C	9
0x3E (0x5E)	Reserved	–	–	–	–	–	–	–	–	
0x3D (0x5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	12
0x3C (0x5C)	OCR0B	Timer/Counter0 – Compare Register B								86
0x3B (0x5B)	GIMSK	INT1	INT0	PCIE0	PCIE2	PCIE1	–	–	–	52
0x3A (0x5A)	GIFR	INTF1	INTF0	PCIF0	PCIF2	PCIF1	–	–	–	53
0x39 (0x59)	TIMSK	TOIE1	OCIE1A	OCIE1B	–	ICIE1	OCIE0B	TOIE0	OCIE0A	87, 116
0x38 (0x58)	TIFR	TOV1	OCF1A	OCF1B	–	ICF1	OCF0B	TOV0	OCF0A	87, 117
0x37 (0x57)	SPMCSR	–	–	RSIG	CTPB	RFLB	PGWRT	PGERS	SPMEN	176
0x36 (0x56)	OCR0A	Timer/Counter0 – Compare Register A								86
0x35 (0x55)	MCUCR	PUD	SM1	SE	SM0	ISC11	ISC10	ISC01	ISC00	37, 51, 69
0x34 (0x54)	MCUSR	–	–	–	–	WDRF	BORF	EXTRF	PORF	45
0x33 (0x53)	TCCR0B	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	85
0x32 (0x52)	TCNT0	Timer/Counter0 (8-bit)								86
0x31 (0x51)	OSCCAL	–	CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0	32
0x30 (0x50)	TCCR0A	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00	82
0x2F (0x4F)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	–	–	WGM11	WGM10	111
0x2E (0x4E)	TCCR1B	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	113
0x2D (0x4D)	TCNT1H	Timer/Counter1 – Counter Register High Byte								115
0x2C (0x4C)	TCNT1L	Timer/Counter1 – Counter Register Low Byte								115
0x2B (0x4B)	OCR1AH	Timer/Counter1 – Compare Register A High Byte								115
0x2A (0x4A)	OCR1AL	Timer/Counter1 – Compare Register A Low Byte								115
0x29 (0x49)	OCR1BH	Timer/Counter1 – Compare Register B High Byte								115
0x28 (0x48)	OCR1BL	Timer/Counter1 – Compare Register B Low Byte								115
0x27 (0x47)	Reserved	–	–	–	–	–	–	–	–	
0x26 (0x46)	CLKPR	CLKPCE	–	–	–	CLKPS3	CLKPS2	CLKPS1	CLKPS0	32
0x25 (0x45)	ICR1H	Timer/Counter1 - Input Capture Register High Byte								116
0x24 (0x44)	ICR1L	Timer/Counter1 - Input Capture Register Low Byte								116
0x23 (0x43)	GTCCR	–	–	–	–	–	–	–	PSR10	119
0x22 (0x42)	TCCR1C	FOC1A	FOC1B	–	–	–	–	–	–	114
0x21 (0x41)	WDTCSR	WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0	45
0x20 (0x40)	PCMSK0	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	54
0x1F (0x3F)	Reserved	–	–	–	–	–	–	–	–	
0x1E (0x3E)	EEAR	–	EEPROM Address Register							24
0x1D (0x3D)	EEDR	EEPROM Data Register								23
0x1C (0x3C)	EECR	–	–	EEPM1	EEPM0	EERIE	EEMPE	EEPE	EERE	24
0x1B (0x3B)	PORTA	–	–	–	–	–	PORTA2	PORTA1	PORTA0	69
0x1A (0x3A)	DDRA	–	–	–	–	–	DDA2	DDA1	DDA0	69
0x19 (0x39)	PINA	–	–	–	–	–	PINA2	PINA1	PINA0	70
0x18 (0x38)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	70
0x17 (0x37)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	70
0x16 (0x36)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	70
0x15 (0x35)	GPIOR2	General Purpose I/O Register 2								25
0x14 (0x34)	GPIOR1	General Purpose I/O Register 1								25
0x13 (0x33)	GPIOR0	General Purpose I/O Register 0								25
0x12 (0x32)	PORTD	–	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	70
0x11 (0x31)	DDRD	–	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	70
0x10 (0x30)	PIND	–	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	70
0x0F (0x2F)	USIDR	USI Data Register								166
0x0E (0x2E)	USISR	USISIF	USIOIF	USIPF	USIDC	USICNT3	USICNT2	USICNT1	USICNT0	165
0x0D (0x2D)	USICR	USISIE	USIOIE	USIWM1	USIWM0	USICS1	USICS0	USICLK	USITC	163
0x0C (0x2C)	UDR	UART Data Register (8-bit)								137
0x0B (0x2B)	UCSRA	RXC	TXC	UDRE	FE	DOR	UPE	U2X	MPCM	138
0x0A (0x2A)	UCSRB	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	139
0x09 (0x29)	UBRRL	UBRRH[7:0]								141
0x08 (0x28)	ACSR	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	168
0x07 (0x27)	BODCR	–	–	–	–	–	–	BODS	BODSE	38
0x06 (0x26)	PRR	–	–	–	–	PRTIM1	PRTIM0	PRUSI	PRUSART	37
0x05 (0x25)	PCMSK2	–	PCINT17	PCINT16	PCINT15	PCINT14	PCINT13	PCINT12	PCINT11	53
0x04 (0x24)	PCMSK1	–	–	–	–	–	PCINT10	PCINT9	PCINT8	54
0x03 (0x23)	UCSRC	UMSEL1	UMSEL0	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL	140
0x02 (0x22)	UBRRH	–	–	–	–	UBRRH[11:8]				141
0x01 (0x21)	DIDR	–	–	–	–	–	–	AIN1D	AIN0D	169
0x00 (0x20)	USIBR	USI Buffer Register								167

Notes:
1. For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.
2. 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.
3. 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.
4. 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.

25. Instruction Set Summary

Mnemonics	Operands	Description	Operation	Flags	#Clocks
ARITHMETIC AND LOGIC INSTRUCTIONS					
ADD	Rd, Rr	Add two Registers	Rd ← Rd + Rr	Z,C,N,V,H	1
ADC	Rd, Rr	Add with Carry two Registers	Rd ← Rd + Rr + C	Z,C,N,V,H	1
ADIW	Rdl,K	Add Immediate to Word	Rdh:Rdl ← Rdh:Rdl + K	Z,C,N,V,S	2
SUB	Rd, Rr	Subtract two Registers	Rd ← Rd - Rr	Z,C,N,V,H	1
SUBI	Rd, K	Subtract Constant from Register	Rd ← Rd - K	Z,C,N,V,H	1
SBC	Rd, Rr	Subtract with Carry two Registers	Rd ← Rd - Rr - C	Z,C,N,V,H	1
SBCI	Rd, K	Subtract with Carry Constant from Reg.	Rd ← Rd - K - C	Z,C,N,V,H	1
SBIW	Rdl,K	Subtract Immediate from Word	Rdh:Rdl ← Rdh:Rdl - K	Z,C,N,V,S	2
AND	Rd, Rr	Logical AND Registers	Rd ← Rd • Rr	Z,N,V	1
ANDI	Rd, K	Logical AND Register and Constant	Rd ← Rd • K	Z,N,V	1
OR	Rd, Rr	Logical OR Registers	Rd ← Rd v Rr	Z,N,V	1
ORI	Rd, K	Logical OR Register and Constant	Rd ← Rd v K	Z,N,V	1
EOR	Rd, Rr	Exclusive OR Registers	Rd ← Rd ⊕ Rr	Z,N,V	1
COM	Rd	One's Complement	Rd ← 0xFF – Rd	Z,C,N,V	1
NEG	Rd	Two's Complement	Rd ← 0x00 – Rd	Z,C,N,V,H	1
SBR	Rd,K	Set Bit(s) in Register	Rd ← Rd v K	Z,N,V	1
CBR	Rd,K	Clear Bit(s) in Register	Rd ← Rd • (0xFF - K)	Z,N,V	1
INC	Rd	Increment	Rd ← Rd + 1	Z,N,V	1
DEC	Rd	Decrement	Rd ← Rd – 1	Z,N,V	1
TST	Rd	Test for Zero or Minus	Rd ← Rd • Rd	Z,N,V	1
CLR	Rd	Clear Register	Rd ← Rd ⊕ Rd	Z,N,V	1
SER	Rd	Set Register	Rd ← 0xFF	None	1
BRANCH INSTRUCTIONS					
RJMP	k	Relative Jump	PC ← PC + k + 1	None	2
IJMP		Indirect Jump to (Z)	PC ← Z	None	2
RCALL	k	Relative Subroutine Call	PC ← PC + k + 1	None	3
ICALL		Indirect Call to (Z)	PC ← Z	None	3
RET		Subroutine Return	PC ← STACK	None	4
RETI		Interrupt Return	PC ← STACK	I	4
CPSE	Rd,Rr	Compare, Skip if Equal	if (Rd = Rr) PC ← PC + 2 or 3	None	1/2/3
CP	Rd,Rr	Compare	Rd – Rr	Z, N,V,C,H	1
CPC	Rd,Rr	Compare with Carry	Rd – Rr – C	Z, N,V,C,H	1
CPI	Rd,K	Compare Register with Immediate	Rd – K	Z, N,V,C,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 is Set	if (Rr(b)=1) PC ← PC + 2 or 3	None	1/2/3
SBIC	P, b	Skip if Bit in I/O Register Cleared	if (P(b)=0) PC ← PC + 2 or 3	None	1/2/3
SBIS	P, b	Skip if Bit in I/O Register is Set	if (P(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
BRPL	k	Branch if Plus	if (N = 0) then PC ← PC + k + 1	None	1/2
BRGE	k	Branch if Greater or Equal, Signed	if (N ⊕ V= 0) then PC ← PC + k + 1	None	1/2
BRLT	k	Branch if Less Than Zero, Signed	if (N ⊕ V= 1) then PC ← PC + k + 1	None	1/2
BRHS	k	Branch if Half Carry Flag Set	if (H = 1) then PC ← PC + k + 1	None	1/2
BRHC	k	Branch if Half Carry Flag Cleared	if (H = 0) then PC ← PC + k + 1	None	1/2
BRTS	k	Branch if T Flag Set	if (T = 1) then PC ← PC + k + 1	None	1/2
BRTC	k	Branch if T Flag Cleared	if (T = 0) then PC ← PC + k + 1	None	1/2
BRVS	k	Branch if Overflow Flag is Set	if (V = 1) then PC ← PC + k + 1	None	1/2
BRVC	k	Branch if Overflow Flag is Cleared	if (V = 0) then PC ← PC + k + 1	None	1/2
BRIE	k	Branch if Interrupt Enabled	if (I = 1) then PC ← PC + k + 1	None	1/2
BRID	k	Branch if Interrupt Disabled	if (I = 0) then PC ← PC + k + 1	None	1/2
BIT AND BIT-TEST INSTRUCTIONS					
SBI	P,b	Set Bit in I/O Register	I/O(P,b) ← 1	None	2
CBI	P,b	Clear Bit in I/O Register	I/O(P,b) ← 0	None	2
LSL	Rd	Logical Shift Left	Rd(n+1) ← Rd(n), Rd(0) ← 0	Z,C,N,V	1
LSR	Rd	Logical Shift Right	Rd(n) ← Rd(n+1), Rd(7) ← 0	Z,C,N,V	1
ROL	Rd	Rotate Left Through Carry	Rd(0)←C,Rd(n+1)← Rd(n),C←Rd(7)	Z,C,N,V	1

Mnemonics	Operands	Description	Operation	Flags	#Clocks
ROR	Rd	Rotate Right Through Carry	Rd(7)←C,Rd(n)← Rd(n+1),C←Rd(0)	Z,C,N,V	1
ASR	Rd	Arithmetic Shift Right	Rd(n) ← Rd(n+1), n=0..6	Z,C,N,V	1
SWAP	Rd	Swap Nibbles	Rd(3..0)←Rd(7..4),Rd(7..4)←Rd(3..0)	None	1
BSET	s	Flag Set	SREG(s) ← 1	SREG(s)	1
BCLR	s	Flag Clear	SREG(s) ← 0	SREG(s)	1
BST	Rr, b	Bit Store from Register to T	T ← Rr(b)	T	1
BLD	Rd, b	Bit load from T to Register	Rd(b) ← T	None	1
SEC		Set Carry	C ← 1	C	1
CLC		Clear Carry	C ← 0	C	1
SEN		Set Negative Flag	N ← 1	N	1
CLN		Clear Negative Flag	N ← 0	N	1
SEZ		Set Zero Flag	Z ← 1	Z	1
CLZ		Clear Zero Flag	Z ← 0	Z	1
SEI		Global Interrupt Enable	I ← 1	I	1
CLI		Global Interrupt Disable	I ← 0	I	1
SES		Set Signed Test Flag	S ← 1	S	1
CLS		Clear Signed Test Flag	S ← 0	S	1
SEV		Set Twos Complement Overflow.	V ← 1	V	1
CLV		Clear Twos Complement Overflow	V ← 0	V	1
SET		Set T in SREG	T ← 1	T	1
CLT		Clear T in SREG	T ← 0	T	1
SEH		Set Half Carry Flag in SREG	H ← 1	H	1
CLH		Clear Half Carry Flag in SREG	H ← 0	H	1
DATA TRANSFER INSTRUCTIONS					
MOV	Rd, Rr	Move Between Registers	Rd ← Rr	None	1
MOVW	Rd, Rr	Copy Register Word	Rd+1:Rd ← Rr+1:Rr	None	1
LDI	Rd, K	Load Immediate	Rd ← K	None	1
LD	Rd, X	Load Indirect	Rd ← (X)	None	2
LD	Rd, X+	Load Indirect and Post-Inc.	Rd ← (X), X ← X + 1	None	2
LD	Rd, - X	Load Indirect and Pre-Dec.	X ← X - 1, Rd ← (X)	None	2
LD	Rd, Y	Load Indirect	Rd ← (Y)	None	2
LD	Rd, Y+	Load Indirect and Post-Inc.	Rd ← (Y), Y ← Y + 1	None	2
LD	Rd, - Y	Load Indirect and Pre-Dec.	Y ← Y - 1, Rd ← (Y)	None	2
LDD	Rd,Y+q	Load Indirect with Displacement	Rd ← (Y + q)	None	2
LD	Rd, Z	Load Indirect	Rd ← (Z)	None	2
LD	Rd, Z+	Load Indirect and Post-Inc.	Rd ← (Z), Z ← Z+1	None	2
LD	Rd, -Z	Load Indirect and Pre-Dec.	Z ← Z - 1, Rd ← (Z)	None	2
LDD	Rd, Z+q	Load Indirect with Displacement	Rd ← (Z + q)	None	2
LDS	Rd, k	Load Direct from SRAM	Rd ← (k)	None	2
ST	X, Rr	Store Indirect	(X) ← Rr	None	2
ST	X+, Rr	Store Indirect and Post-Inc.	(X) ← Rr, X ← X + 1	None	2
ST	- X, Rr	Store Indirect and Pre-Dec.	X ← X - 1, (X) ← Rr	None	2
ST	Y, Rr	Store Indirect	(Y) ← Rr	None	2
ST	Y+, Rr	Store Indirect and Post-Inc.	(Y) ← Rr, Y ← Y + 1	None	2
ST	- Y, Rr	Store Indirect and Pre-Dec.	Y ← Y - 1, (Y) ← Rr	None	2
STD	Y+q,Rr	Store Indirect with Displacement	(Y + q) ← Rr	None	2
ST	Z, Rr	Store Indirect	(Z) ← Rr	None	2
ST	Z+, Rr	Store Indirect and Post-Inc.	(Z) ← Rr, Z ← Z + 1	None	2
ST	-Z, Rr	Store Indirect and Pre-Dec.	Z ← Z - 1, (Z) ← Rr	None	2
STD	Z+q,Rr	Store Indirect with Displacement	(Z + q) ← Rr	None	2
STS	k, Rr	Store Direct to SRAM	(k) ← Rr	None	2
LPM		Load Program Memory	R0 ← (Z)	None	3
LPM	Rd, Z	Load Program Memory	Rd ← (Z)	None	3
LPM	Rd, Z+	Load Program Memory and Post-Inc	Rd ← (Z), Z ← Z+1	None	3
SPM		Store Program Memory	(Z) ← R1:R0	None	-
IN	Rd, P	In Port	Rd ← P	None	1
OUT	P, Rr	Out Port	P ← Rr	None	1
PUSH	Rr	Push Register on Stack	STACK ← Rr	None	2
POP	Rd	Pop Register from Stack	Rd ← STACK	None	2
MCU CONTROL INSTRUCTIONS					
NOP		No Operation		None	1
SLEEP		Sleep	(see specific descr. for Sleep function)	None	1
WDR		Watchdog Reset	(see specific descr. for WDR/timer)	None	1
BREAK		Break	For On-chip Debug Only	None	N/A

26. Ordering Information

26.1 ATtiny2313A

Speed (MHz) [1]	Supply Voltage (V)	Temperature Range	Package [2]	Ordering Code [3]
20	1.8 – 5.5	Industrial (-40°C to +85°C) [4]	20P3	ATtiny2313A-PU
			20S	ATtiny2313A-SU
				ATtiny2313A-SUR
			20M1	ATtiny2313A-MU
				ATtiny2313A-MUR
			20M2 [5][6]	ATtiny2313A-MMH
				ATtiny2313A-MMHR

Notes:
1. For speed vs. supply voltage, see section 22.3 “Speed” on page 200.
2. All packages are Pb-free, halide-free and fully green, and they comply with the European directive for Restriction of Hazardous Substances (RoHS).
3. Code indicators:
   - H: NiPdAu lead finish
   - U or N: matte tin
   - R: tape & reel
4. Can also be supplied in wafer form. Contact your local Atmel sales office for ordering information and minimum quantities.
5. NiPdAu finish
6. Topside markings :
   - 1st Line: T2313
   - 2nd Line: Axx
   - 3rd Line: xxx

Package Type	
20P3	20-lead, 0.300" Wide, Plastic Dual Inline Package (PDIP)
20S	20-lead, 0.300" Wide, Plastic Gull Wing Small Outline Package (SOIC)
20M1	20-pad, 4 x 4 x 0.8 mm Body, Quad Flat No-Lead / Micro Lead Frame Package (MLF)
20M2	20-pad, 3 x 3 x 0.85 mm Body, Very Thin Quad Flat No Lead Package (VQFN)

26.2 ATtiny4313

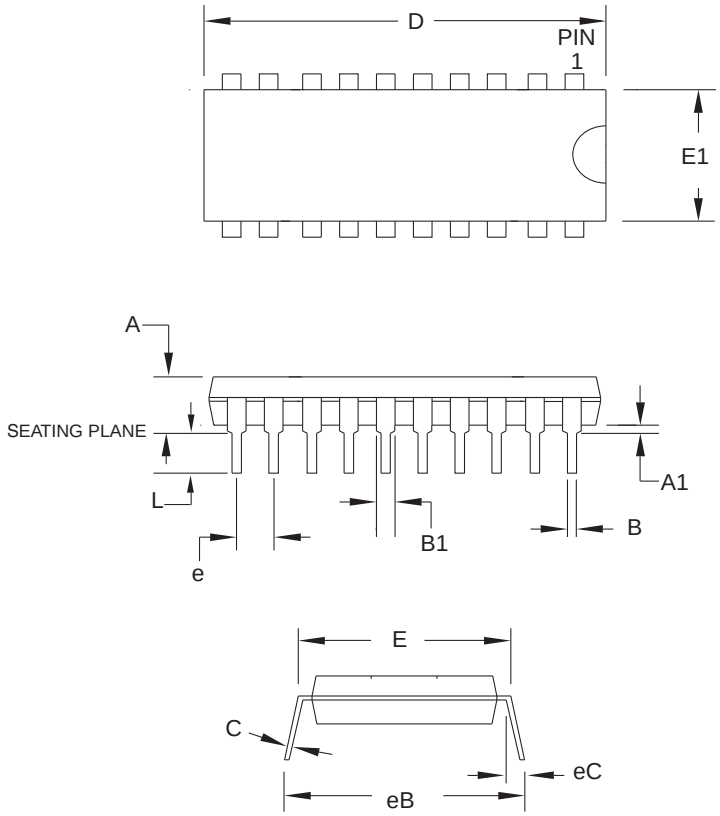
Speed (MHz) [1]	Supply Voltage (V)	Temperature Range	Package [2]	Ordering Code [3]
20	1.8 – 5.5	Industrial (-40°C to +85°C) [4]	20P3	ATtiny4313-PU
			20S	ATtiny4313-SU
				ATtiny4313-SUR
			20M1	ATtiny4313-MU
				ATtiny4313-MUR
			20M2 [5][6]	ATtiny4313-MMH
				ATtiny4313-MMHR

Notes:
1. For speed vs. supply voltage, see section 22.3 “Speed” on page 200.
2. All packages are Pb-free, halide-free and fully green, and they comply with the European directive for Restriction of Hazardous Substances (RoHS).
3. Code indicators:
   - H: NiPdAu lead finish
   - U or N: matte tin
   - R: tape & reel
4. Can also be supplied in wafer form. Contact your local Atmel sales office for ordering information and minimum quantities.
5. NiPdAu finish
6. Topside markings:
   - 1st Line: T4313
   - 2nd Line: Axx
   - 3rd Line: xxx

Package Type	
20P3	20-lead, 0.300" Wide, Plastic Dual Inline Package (PDIP)
20S	20-lead, 0.300" Wide, Plastic Gull Wing Small Outline Package (SOIC)
20M1	20-pad, 4 x 4 x 0.8 mm Body, Quad Flat No-Lead/Micro Lead Frame Package (MLF)
20M2	20-pad, 3 x 3 x 0.85 mm Body, Very Thin Quad Flat No Lead Package (VQFN)

27. Packaging Information

27.1 20P3



COMMON DIMENSIONS
(Unit of Measure = mm)

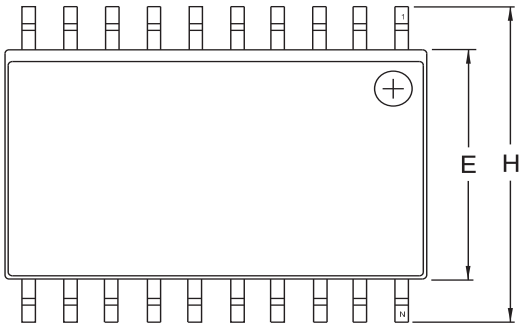
SYMBOL	MIN	NOM	MAX	NOTE
A	–	–	5.334	
A1	0.381	–	–	
D	25.493	–	25.984	Note 2
E	7.620	–	8.255	
E1	6.096	–	7.112	Note 2
B	0.356	–	0.559	
B1	1.270	–	1.551	
L	2.921	–	3.810	
C	0.203	–	0.356	
eB	–	–	10.922	
eC	0.000	–	1.524	
e	2.540 TYP			

Notes:
1. This package conforms to JEDEC reference MS-001, Variation AD.
2. Dimensions D and E1 do not include mold Flash or Protrusion. Mold Flash or Protrusion shall not exceed 0.25 mm (0.010").

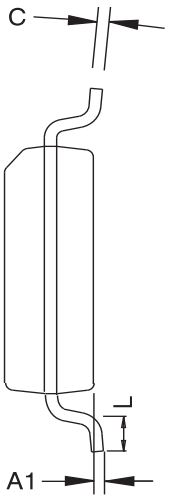
2010-10-19

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**20P3**, 20-lead (0.300"/7.62 mm Wide) Plastic Dual Inline Package (PDIP)	20P3	D

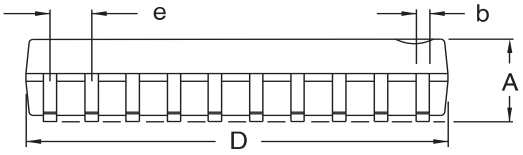
27.2 20S



Top View



End View



Side View

COMMON DIMENSIONS
(Unit of Measure – mm)

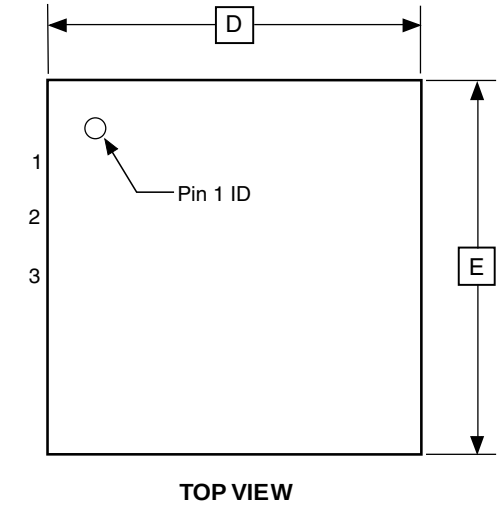
SYMBOL	MIN	NOM	MAX	NOTE
A	2.35		2.65	
A1	0.10		0.30	
b	0.33		0.51	4
C	0.23		0.32	
D	12.60		13.00	1
E	7.40		7.60	2
H	10.00		10.65	
L	0.40		1.27	3
e	1.27 BSC			

Notes.
1. This drawing is for general information only; refer to JEDEC Drawing MS-013, Variation AC for additional information.
2. Dimension 'D' does not include mold Flash, protrusions or gate burrs. Mold Flash, protrusions and gate burrs shall not exceed 0.15 mm (0.006') per side.
3. Dimension 'E' does not include inter-lead Flash or protrusion. Inter-lead Flash and protrusions shall not exceed 0.25 mm (0.010') per side.
4. 'L' is the length of the terminal for soldering to a substrate.
5. The lead width 'b', as measured 0.36 mm (0.014') or greater above the seating plane, shall not exceed a maximum value of 0.61 mm (0.024') per side.

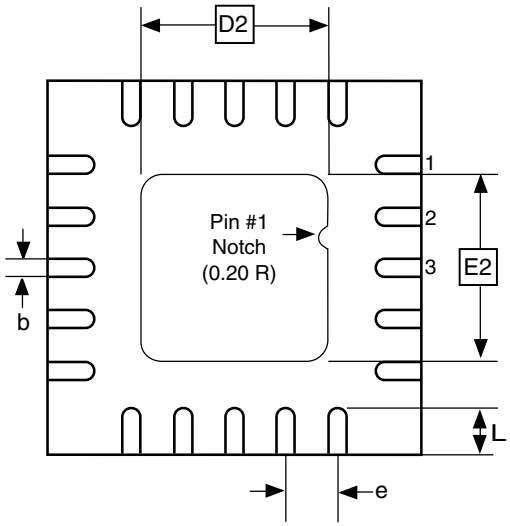
11/6/06

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**20S2**, 20-lead, 0.300' Wide Body, Plastic Gull Wing Small Outline Package (SOIC)	20S2	B

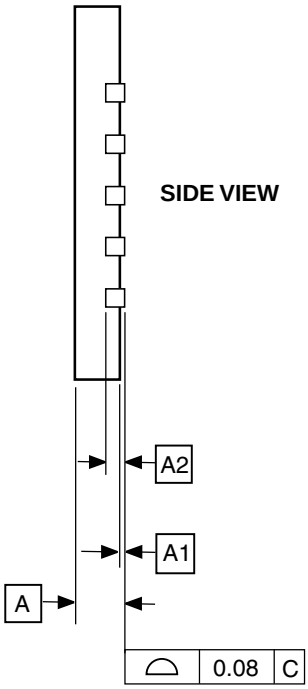
27.3 20M1



TOP VIEW



BOTTOM VIEW



COMMON DIMENSIONS
(Unit of Measure = mm)

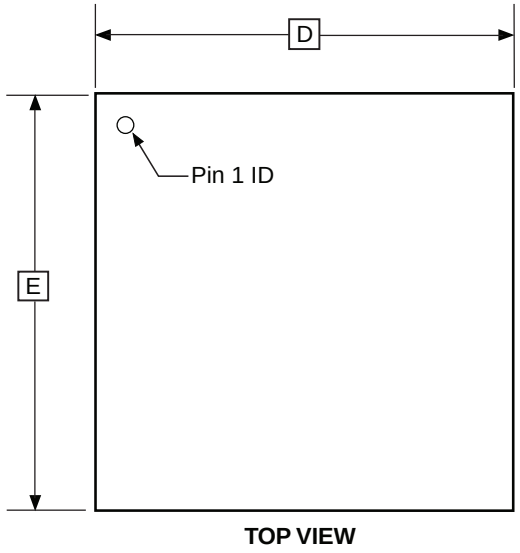
SYMBOL	MIN	NOM	MAX	NOTE
A	0.70	0.75	0.80	
A1	–	0.01	0.05	
A2		0.20 REF		
b	0.18	0.23	0.30	
D		4.00 BSC		
D2	2.45	2.60	2.75	
E		4.00 BSC		
E2	2.45	2.60	2.75	
e		0.50 BSC		
L	0.35	0.40	0.55	

Note: Reference JEDEC Standard MO-220, Fig. 1 (SAW Singulation) WGGD-5.

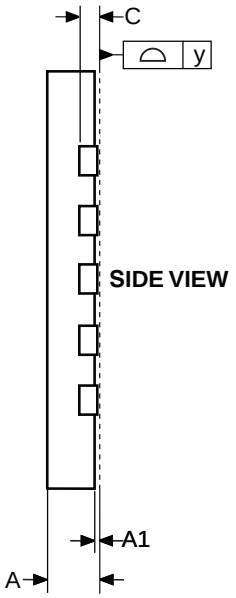
10/27/04

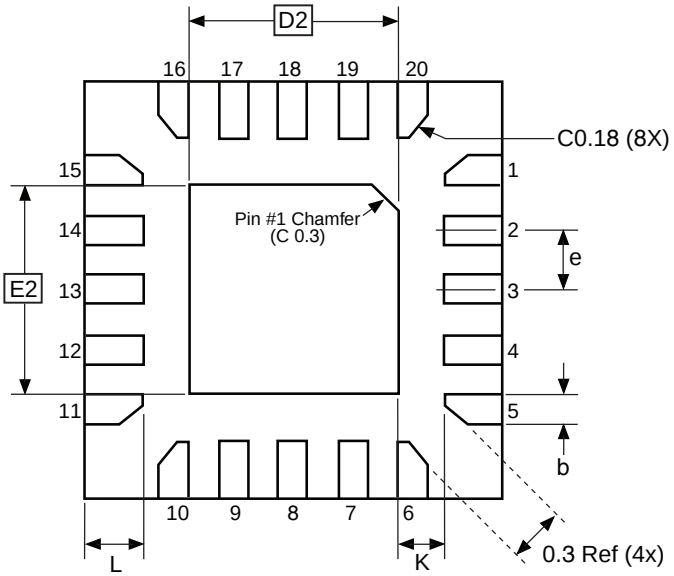
2325 Orchard Parkway San Jose, CA 95131	**TITLE** **20M1**, 20-pad, 4 x 4 x 0.8 mm Body, Lead Pitch 0.50 mm, 2.6 mm Exposed Pad, Micro Lead Frame Package (MLF)	**DRAWING NO.** 20M1	**REV.** A

27.4 20M2



TOP VIEW





BOTTOM VIEW

COMMON DIMENSIONS
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	0.75	0.80	0.85	
A1	0.00	0.02	0.05	
b	0.17	0.22	0.27	
C		0.152		
D	2.90	3.00	3.10	
D2	1.40	1.55	1.70	
E	2.90	3.00	3.10	
E2	1.40	1.55	1.70	
e	–	0.45	–	
L	0.35	0.40	0.45	
K	0.20	–	–	
y	0.00	–	0.08	

10/24/08

Package Drawing Contact: packagedrawings@atmel.com	TITLE	GPC	DRAWING NO.	REV.
	20M2, 20-pad, 3 x 3 x 0.85 mm Body, Lead Pitch 0.45 mm, 1.55 x 1.55 mm Exposed Pad, Thermally Enhanced Plastic Very Thin Quad Flat No Lead Package (VQFN)	ZFC	20M2	B

28. Errata

The revision letters in this section refer to the revision of the corresponding ATtiny2313A/4313 device.

28.1 ATtiny2313A

28.1.1 Rev. D

No known errata.

28.1.2 Rev. A – C

These device revisions were referred to as ATtiny2313/ATtiny2313V.

28.2 ATtiny4313

28.2.1 Rev. A

No known errata.

29. Datasheet Revision History

29.1 Rev. 8246B – 10/11

1. Updated device status from Preliminary to Final.
2. Updated document template.
3. Added order codes for tape&reel devices, on page 259 and page 260
4. Updated figures:
   – Figure 23-33 on page 223
   – Figure 23-44 on page 228
   – Figure 23-81 on page 247
   – Figure 23-92 on page 252
5. Updated sections:
   – Section 5. “Memories” on page 16
   – Section 19. “Self-Programming” on page 173
   – Section 20. “Lock Bits, Fuse Bits and Device Signature” on page 178
   – Section 21. “External Programming” on page 184
   – Section 26. “Ordering Information” on page 259

29.2 Rev. 8246A – 11/09

1. Initial revision. Created from document 2543_t2313.
2. Updated datasheet template.
3. Added VQFN in the Pinout Figure 1-1 on page 2.
4. Added Section 7.2 “Software BOD Disable” on page 35.
5. Added Section 7.3 “Power Reduction Register” on page 35.
6. Updated Table 7-2, “Sleep Mode Select,” on page 37.
7. Added Section 7.5.3 “BODCR – Brown-Out Detector Control Register” on page 38.
8. Added reset disable function in Figure 8-1 on page 39.
9. Added pin change interrupts PCINT1 and PCINT2 in Table 9-1 on page 48.
10. Added PCINT17..8 and PCMSK2..1 in Section 9.2 “External Interrupts” on page 49.
11. Added Section 9.3.4 “PCMSK2 – Pin Change Mask Register 2” on page 53.
12. Added Section 9.3.5 “PCMSK1 – Pin Change Mask Register 1” on page 54.
13. Updated Section 10.2.1 “Alternate Functions of Port A” on page 62.
14. Updated Section 10.2.2 “Alternate Functions of Port B” on page 63.
15. Updated Section 10.2.3 “Alternate Functions of Port D” on page 67.
16. Added UMSEL1 and UMSEL0 in Section 14.10.4 “UCSRC – USART Control and Status Register C” on page 140.
17. Added Section 15. “USART in SPI Mode” on page 146.
18. Added USI Buffer Register (USIBR) in Section 16.2 “Overview” on page 156 and in Figure 16-1 on page 156.
19. Added Section 16.5.4 “USIBR – USI Buffer Register” on page 167.
20. Updated Section 19.6.3 “Reading Device Signature Imprint Table from Firmware” on page 175.

21. Updated Section 19.7.1 “SPMCSR – Store Program Memory Control and Status Register” on page 176.
22. Added Section 20.3 “Device Signature Imprint Table” on page 180.
23. Updated Section 20.3.1 “Calibration Byte” on page 181.
24. Changed BS to BS1 in Section 20.6.13 “Reading the Signature Bytes” on page 189.
25. Updated Section 22.2 “DC Characteristics” on page 198.
26. Added Section 23.1 “Effect of Power Reduction” on page 206.
27. Updated characteristic plots in Section 23. “Typical Characteristics” for ATtiny2313A (pages 207 - 230), and added plots for ATtiny4313 (pages 231 - 254).
28. Updated Section 24. “Register Summary” on page 255 .
29. Updated Section 26. “Ordering Information” on page 259, added the package type 20M2 and the ordering code -MMH (VQFN), and added the topside marking note.

Table of Contents

Headquarters

Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131
USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

International

Atmel Asia Limited
Unit 01-5 & 16, 19/F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
HONG KONG
Tel: (852) 2245-6100
Fax: (852) 2722-1369

Atmel Munich GmbH
Business Campus
Parkring 4
D-85748 Garching b. Munich
GERMANY
Tel: (+49) 89-31970-0
Fax: (+49) 89-3194621

Atmel Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
JAPAN
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Product Contact

Web Site
www.atmel.com

Technical Support
avr@atmel.com

Sales Contact
www.atmel.com/contacts

Literature Requests
www.atmel.com/literature