A Littelfuse Company

High-Performance 8-Bit Microcontrollers

Z8 Encore! XP® F3224 Series

Product Specification

PS038107-0521

PRELIMINARY

⚠ **Warning:** DO NOT USE THIS PRODUCT IN LIFE SUPPORT SYSTEMS.

LIFE SUPPORT POLICY

ZILOG'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS PRIOR WRITTEN APPROVAL OF THE PRESIDENT AND GENERAL COUNSEL OF ZILOG CORPORATION.

As used herein

Life support devices or systems are devices which (a) are intended for surgical implant into the body, or (b) support or sustain life and whose failure to perform when properly used in accordance with instructions for use provided in the labeling can be reasonably expected to result in a significant injury to the user. A critical component is any component in a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system or to affect its safety or effectiveness.

Document Disclaimer

Revision History

Each instance in the following revision history table reflects a change to this document from its previous version. For more details, refer to the corresponding pages or appropriate links provided in the table.

Date	Revision Level	Description	Page No.
May 2021	07	Added MCT availability for 32-pin package.	1, 423-426
		Removed 20-pin package pending manufacturing support.	2, 10, 46, 48, 400, 422-426
		Added recommendation to alter the CONTCONV and SCAN bits only while the ADC is idle.	312
		Clarified ADC offset compensation and updated ADC gain compensation formula.	315
		Clarified programmable gain connections in OpAmpA block diagram.	331
		Updated absolute maximum ratings for injection current and pin voltage.	400
		Updated typical current consumption specifications.	402
		Updated bandgap voltage.	407
		Updated comparator input offset specification.	411
		FIxed typos.	Various
Oct 2020	06	Corrected minor typos.	Various
		Removed direct DCO control using the DCOCTLL/DCOCTLH registers and the SEEDSEL control bit.	78, 79, 86, 87
		Enhanced UART noise filter description. Corrected Noise Filter Operation figure.	205
		Removed ADC ANA12/ANA13 differential pair.	308, 320
		Clarified ADC wake-up control.	312
		Updated Flash Byte Programming Time.	406
		Updated DCO step size and deleted DCOCTLCH/L step size.	416
		Updated 20-SSOP package information.	422
		Added to Ordering Information Z8F32240100ZCOG and ZMOTIONL400ZCOG.	423, 425
Apr 2020	05	Clarified that PortD0/$\overline{\text{RESET}}$ is not a Stop Mode Recovery source.	48, 64
		Added setting SCKFEN=1 to the System Clock Failure description.	76
		Updated FBP register name to FPROT.	357
		Updated electrical characteristics for: Flash Byte Program Time, I_{VBIAS}, and I_{PROG}. Merged I_{DD}FLL into I_{DD}DCO. Dropped DCO tempco and voltco parameters.	406, 407 416
		Fixed typos and formatting in Chapters 18, 19, 27, and 29. Updated SSOP 20-pin Environmental Flow.	304, 319, 400, 423

Sep 2019	04	Updated AV_{DD}, AV_{SS} pin descriptions.	16
		Updated Table 7 (Flash Information Area Map).	20
		Removed System Clock failure souce limitation.	76, 83
		Updated CLKCTL3 Register reset value.	84
		Updated UART DMX Mode interrupt description.	Various
		Updated ADC starting and stopping conversions.	325
		Updated ADC window detection regarding data formatting.	Various
		Added Comparator hysteresis definition.	411
		Update DCO current consumption.	416
		Corrected minor typos.	Various
Aug 2017	03	Removed 44-pin LQFP packaging option.	Various
		Added QFN EP pin.	CH 2
		Updated System Clock Failure description of clock source detection.	76
		Clarified that the WTO is not affected by SCKDIV.	76, 83
		Updated FLLNDIVH reset value.	80
		Changed ADC REFSEL=10 selection to be reserved.	CH 19, 409
		Updated ADC fixed reference sampling time ($T_{S\_VDD/2}$), wake-up time, and ADC input resistance (R_{IN}). Added VBG ADC sampling time (T_{S_VBG}).	409
		Updated symbols in Tables 233 and 234. Updated Stop-Mode Recovery Latency value.	405
Jun 2017	02	Corrected minor typos/inconsistencies.	Various
		Added internal resistor to DBG pin.	15, 375
		Updated pin configurations for 20-pin SSOP, 32-pin QFN, and 44-pin QFN and LQFP packages.	11, 12
		Updated functionality of PWRCTL1[3:2] affecting ADC and DAC VREF.	45, 313, 314
		Removed FLLRDY bit. Updated FLLDONE and SEEDSEL bit functionality.	CH 8
		Removed ADC translation buffer. Removed 2-pass behavior for 14-bit resolution conversions in favor of user initiated averaging. Eliminated power settings. Removed SST field and functionality.	CH 19, CH 26
		Updated ADC C_{IN}.	314, 410
		Updated comparator window mode connectivity.	337
		Updated T_{VBO} value.	405
		Removed T_{LVD} parameter.	
		Updated LFXO active current. Added feedback resistor parameter.	415
		Updated DCO resolution, DCO control word resolution, $T_{FLADONE.}$.	416
		Updated POR $V_{TH.}$.	404
		Corrected Comparator hysteresis setting inconsistency.	411
Oct 2016	01	Original issue.	n/a

Table of Contents

List of Figures

List of Tables

Chapter 1. Overview

Zilog's F3224 Series MCUs, members of the Z8 Encore! XP® family, are based on Zilog's advanced 8-bit eZ8 CPU core. This microcontroller is optimized for low-power and wireless applications, and supports 1.8V to 3.6V low-voltage operation with extremely low Active, Halt and Stop Mode currents, plus it offers an assortment of speed and low-power options. In addition, the feature-rich analog and digital peripherals of the F3224 Series makes it suitable for a variety of applications including safety and security, utility metering, digital power supervisory, hand-held electronic devices, sensing, motion detection and general motor control.

1.1. Features

Key features of the F3224 Series MCU include:

- 20MHz eZ8 CPU core
- 16KB or 32KB Flash memory with in-circuit programming capability
- 3.75KB internal RAM
- Up to 15-Channel, 12-bit Analog-to-Digital Converter (ADC) that can be configured for internal or external voltage reference and single-ended or differential inputs
- Two on-chip analog comparators
- Two on-chip, low-power operational amplifiers
- 8 Channel Event System provides communication between peripherals for autonomous triggering
- Full-duplex 9-bit UART port with the support of Local Interconnect Network (LIN) and Digital Addressable Lighting Interface (DALI) protocols. RS-485 Multidrop Mode up to 250kbit/sec (DMX Support) integrated with UARTs
- Enhanced Serial Peripheral Interface (SPI) controller
- I^2C controller which supports Master/Slave modes
- One 16-bit timer with 3 functional modes
- Three enhanced 16-bit timers with Capture, Compare, and PWM capability
- One additional basic 16-bit timer with interrupt (shared as UART Baud Rate Generator)
- 16-bit Multi-Channel Timer which supports four Capture/Compare/PWM modules
- Watchdog Timer (WDT)

- 26 to 36 General-Purpose Input/Output (GPIO) pins, depending upon package
- Up to 30 interrupt sources with up to 23 interrupt vectors
- On-Chip Debugger (OCD)
- Power-On Reset (POR) and Voltage Brown-Out (VBO) protection
- Built-in Low-Voltage Detection (LVD) with programmable voltage threshold
- Low Frequency Crystal Oscillator (LFXO) operating at 32.768 kHz with low power consumption
- Internal clock sources and clock multiplication including: Internal Precision Oscillator (IPO), Digitally Controlled Oscillator (DCO), Watchdog Timer Oscillator (WTO) and Frequency Locked Loop (FLL)
- Wide operation voltage range: 1.8 V–3.6 V
- 32-, and 44-pin packages
- –40°C to +85°C (extended) operating temperature range

1.2. Part Selection Guide

Table 1 shows basic features and package styles available for each device within the F3224 Series product line.

Table 1. F3224 Series Family Part Selection Guide

Part Number	Flash (KB)	RAM (B)	I/O	ADC Inputs	SPI	I^2C	UART	Packages
Z8F3224	32	3840	26–36	13-19	1	1	1	32-, and 44-pin
Z8F1624	16	3840	26–36	13-19	1	1	1	32-, and 44-pin

1.3. Block Diagram

Figure 1 shows a block diagram of the F3224 Series architecture.

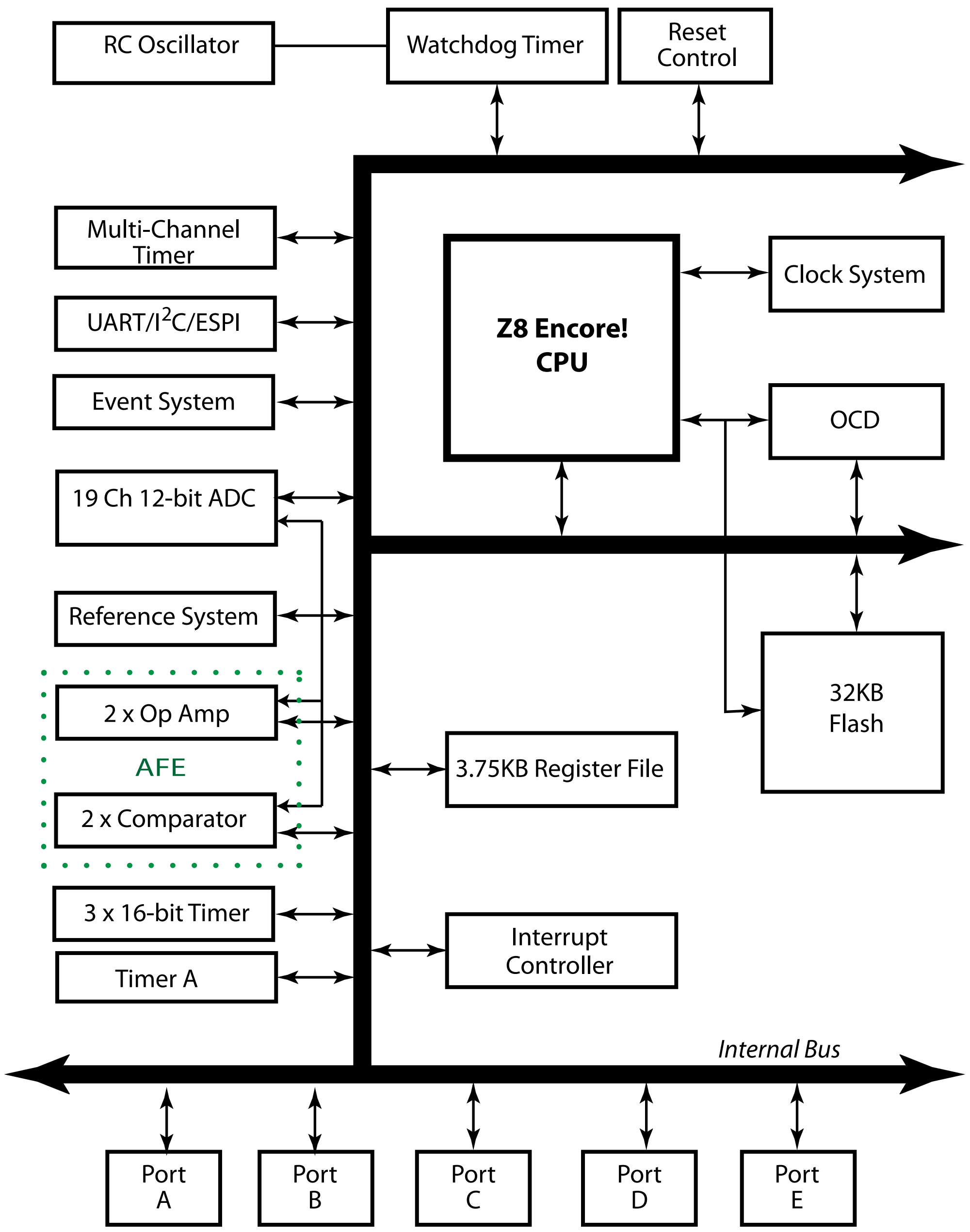


Figure 1. F3224 Series Block Diagram

1.4. eZ8 CPU and Peripheral Overview

Zilog's 8-bit eZ8 CPU meets the continuing demand for faster and more code-efficient microcontrollers. It executes a superset of the original Z8 instruction set. The key features of the eZ8 CPU are:

- Direct register-to-register architecture allows each register to function as an accumulator, improving execution time and decreasing the required Program Memory
- Software stack allows greater depth in subroutine calls and interrupts more than hardware stacks
- Compatible with existing Z8 code
- Expanded internal Register File allows access up to 4KB
- New instructions improve execution efficiency for code developed using higher-level programming languages including C
- Pipelined instruction fetch and execution
- New instructions for improved performance including BIT, BSWAP, BTJ, CPC, LDC, LDCI, LEA, MULT and SRL
- New instructions support 12-bit linear addressing of the register file
- Up to 10MIPS operation
- C Compiler-friendly
- 2 to 9 clock cycles per instruction

To learn more about the eZ8 CPU, refer to the eZ8 CPU Core User Manual (UM0128), which is available free for download from the Zilog website.

1.4.1. General-Purpose Input/Output

The F3224 Series features 14 to 36 port pins (Ports A–E) for general purpose input/output (GPIO). The number of GPIO pins available is a function of package. Each pin is individually programmable.

1.4.2. Flash Controller

The Flash Controller is used to program and erase Flash memory, and supports protection against accidental program and erasure.

1.4.3. Clock System

The clock system generates a System Clock, a low-frequency Peripheral Clock, and the Watchdog Timer Oscillator. It is comprised of:

- Watchdog Timer Oscillator (WTO).
- 32.768 kHz Internal Precision Oscillator (IPO).
- Low Frequency Crystal Oscillator (LFXO) which is a low-power oscillator optimized for use with a 32.768 kHz watch crystal. The IPO and LFXO can be used as clock sources for the Timers in any mode, and as the reference clock for the Frequency Locked Loop (FLL).
- Digitally Controlled Oscillator (DCO).
- Frequency Locked Loop (FLL). The reference clock for the FLL can be either an Internal Precision Oscillator (IPO) or a Low Frequency Crystal Oscillator. The FLL, in conjunction with the DCO can be configured to generate system clock frequencies from 1 to 20 MHz.

1.4.4. Reference System

The Reference System provides programmable and fixed references to the analog peripherals.

1.4.5. 12-Bit Analog-to-Digital Converter

The Analog-to-Digital Converter (ADC) converts an analog input signal to a 12-bit binary number. The ADC supports up to fifteen analog input sources multiplexed with GPIO ports. It is configurable for internal or external voltage reference and single-ended or differential inputs.

1.4.6. Low-Power Operational Amplifiers

Two low-power operational amplifiers (Op Amps) are provided: Op Amp A0 and Op Amp A1. Op Amp Ax (x = 0-1) are low-power, general-purpose operational amplifiers with optional internal programmable gain settings. Each Op Amp output can be internally routed to an op amp input, the ADC, a comparator, or an output pin. These op amps can function in all operating modes, including Stop Mode.

1.4.7. Analog Comparators

The analog comparators compare the signal at an input pin or at other internal signal sources with either an internal programmable voltage reference, an internal fixed reference, or a second input pin. The comparator outputs are used to either drive an output pin, the Event System, or to generate an interrupt. The comparators can function in all operating modes including Stop Mode.

1.4.8. Low-Voltage Detector

The low-voltage detector generates an interrupt when the supply voltage drops below a user-programmable level.

1.4.9. Enhanced SPI

The enhanced SPI is a full-duplex, buffered, synchronous character-oriented channel which supports a four-wire interface.

1.4.10. UART with LIN, DALI, and DMX

A full-duplex 9-bit UART provides serial, asynchronous communication, and supports the Local Interconnect Network (LIN) and Digital Addressable Lighting Interface (DALI) serial communications protocols as well as Asynchronous Serial Digital Data Transmission Standard for Controlling Lighting Equipment and Accessories (DMX). The UART supports 8-bit and 9-bit data modes, selectable parity, and an efficient bus transceiver Driver Enable signal for controlling a multi-transceiver bus, such as a RS-485. The LIN bus is a cost-efficient, single-master, multiple-slave organization which supports speed up to 20 kilobits. Manchester encoding is supported for the DALI protocol.

1.4.11. Master/Slave I^2C

The inter-integrated circuit (I^2C) controller makes the F3224 Series products compatible with the I^2C protocol. The I^2C controller consists of two bidirectional bus lines:

- Serial data (SDA) line
- Serial clock (SCL) line

This I^2C controller also supports Master, Slave, and Multi-Master operations.

1.4.12. Timers

Three enhanced 16-bit reloadable timers are used for timing/counting events or motor control operations. These timers provide a 16-bit programmable reload counter and operate in One-Shot, Triggered One-Shot, Dual Input Triggered One-Shot, Continuous, Counter, PWM Single Output, PWM Dual Output, Capture, Capture Restart, Compare, Gated, Capture and Compare, and Demodulation modes.

In addition to these three enhanced 16-bit timers, there is a 16-bit timer with three functions and there is a basic 16-bit timer both with interrupt functionality. The basic timer is used as a Baud Rate Generator (BRG) when the UART is enabled, and configured as a basic 16-bit timer when the UART is disabled.

1.4.13. Multi-Channel Timer

The multi-channel timer has a 16-bit up/down counter and a 4-channel Capture/Compare/PWM channel array. This timer enables the support of multiple synchronous Capture/Compare/PWM channels based on a single timer.

1.4.14. Interrupt Controller

The F3224 Series of products supports up to thirty interrupt sources with twenty-three interrupt vectors. These interrupts consist of up to twenty-five internal peripheral interrupts and up to sixteen GPIO pin interrupts. These interrupts feature three levels of programmable-interrupt priority.

1.4.15. Reset Controller

The F3224 Series products are reset using the $\overline{\text{RESET}}$ pin, POR, WDT time-out, Stop Mode exit, or VBO warning signal. The $\overline{\text{RESET}}$ pin is bidirectional; i.e., it functions as a reset source as well as a reset indicator.

1.4.16. On-Chip Debugger

The F3224 Series of products features an integrated OCD, which provides a rich set of debugging capabilities such as reading and writing registers, programming Flash memory, setting breakpoints and executing code. The OCD uses one single-pin interface for communication with an external host.

1.4.17. Event System

An 8-channel Event System provides communication between peripherals for autonomous triggering independent of CPU activity. Any Event System source can be selected to drive a signal on an Event System channel. The Event System is active in all operating modes, including Stop Mode.

1.5. Acronyms and Expansions

This document references the acronyms and expansions listed in Table 2.

Table 2. Acronyms and Expansions

Abbreviations/ Acronyms	Expansions
ADC	Analog-to-Digital Converter
CI	Channel Interrupt
CMOS	Complementary Metal-Oxide Semiconductor
DALI	Digital Addressable Lighting Interface
DCO	Digitally Controlled Oscillator
DMX	Asynchronous serial digital data transmission standard for controlling lighting equipment and accessories
Endec	Encoder/decoder
ESPI	Enhanced Serial Peripheral Interface
FLL	Frequency-Locked Loop
GPIO	General-Purpose Input/Output
I^2C	Inter-Integrated Circuit
I^2S	Inter-IC Sound
IPO	Internal Precision Oscillator
IRQ	Interrupt Request
ISR	Interrupt Service Routine
LFXO	Low-Frequency Crystal Oscillator
LIN	Local Interconnect Network
LQFP	Low-Profile Quad Flat Package
LSB	Least-Significant Byte
LVD	Low-Voltage Detection
MSB	Most-Significant Byte
OCD	On-Chip Debugger
Op Amp	Operational Amplifier
PC	Program Counter
POR	Power-On Reset
PWM	Pulse-Width Modulation
QFN	Quad Flat No Lead
SPI	Serial Peripheral Interface

Table 2. Acronyms and Expansions (Continued)

Abbreviations/ Acronyms	Expansions
TI	Timer Interrupt
UART	Universal Asynchronous Receiver/Transmitter
VBO	Voltage Brown-Out
VCO	Voltage Controlled Oscillator
WDT	Watchdog Timer

Chapter 2. Pin Description

The F3224 Series products are available in a variety of package styles and pin configurations. This chapter describes the signals and available pin configurations for each of the package styles. To learn more about the physical package specifications, see the Packaging chapter on page 422.

2.1. Available Packages

Table 3 lists the package styles available for each device in the F3224 Series product line.

Table 3. F3224 Series Package Options

Part Number	32-Pin QFN	44-Pin QFN
Z8F3224	X	X
Z8F1624	X	X

2.2. Pin Configurations

Figures 2 through 3 show the pin configurations of all packages available in the F3224 Series. For signal descriptions, see Table 4 on page 13.

At reset, all port pins default to an input state. In addition, any alternate functionality is not enabled; therefore the pins function as general-purpose input ports until programmed otherwise. At power-up, the Port D0 pin defaults to the $\overline{\text{RESET}}$ alternate function.

The pin configurations listed are preliminary and subject to change based on manufacturing limitations.

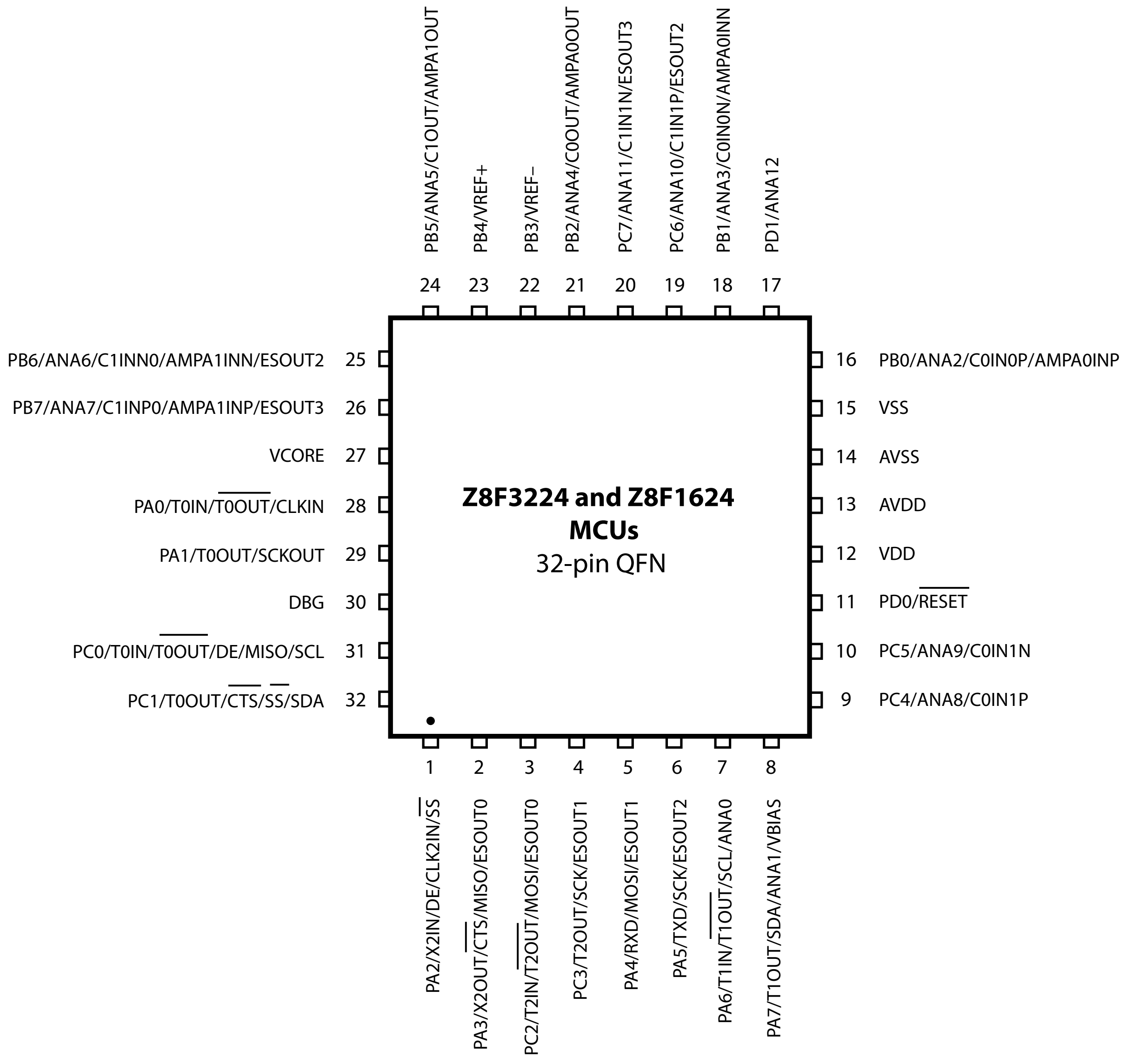


Figure 2. Z8F3224 and Z8F1624 MCUs in 32-Pin Quad Flat No Lead (QFN) Package

---

It is recommended to connect the QFN bottom pad to V_{SS}.

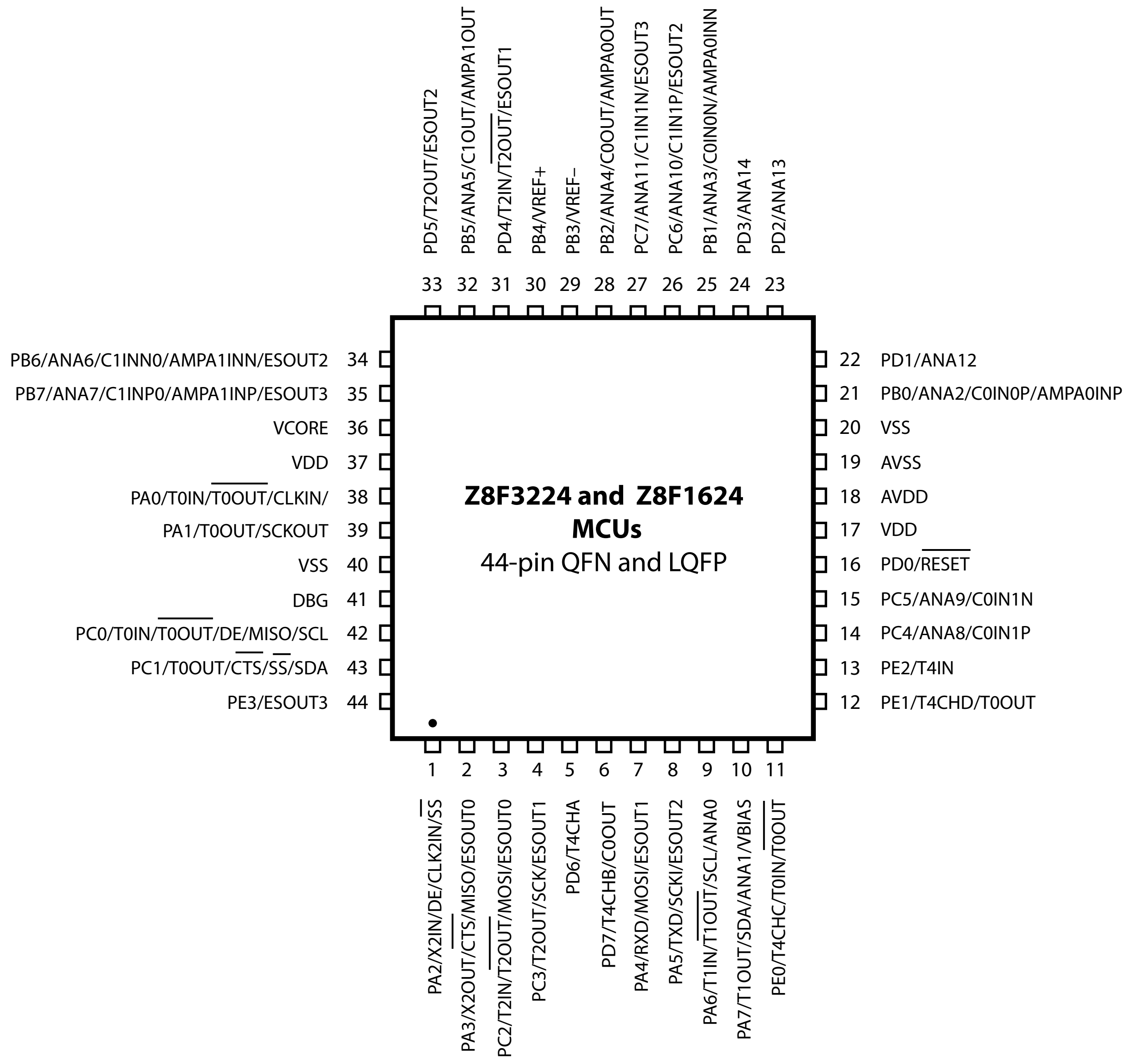


Figure 3. Z8F3224 and Z8F1624 MCUs in 44-Pin Quad Flat No Lead (QFN) Package

---

It is recommended to connect the QFN bottom pad to V_{SS}.

Table 4 lists the pin signals for all F3224 Series MCUs. To determine the signals for a specific package style, see the Pin Configurations section on page 10.

Table 4. Signal Descriptions

Signal Mnemonic	I/O	Description
General-Purpose I/O Ports A–E		
PA[7:0]	I/O	Port A: These pins are used for general-purpose I/O.
PB[7:0]	I/O	Port B: These pins are used for GPIO.
PC[7:0]	I/O	Port C: These pins are used for GPIO.
PD[7:0]	I/O	Port D: These pins are used for GPIO. PD0 is output only.
PE[3:0]	I/O	Port E: These pins are used for GPIO.
UART-LDD Controllers		
TXD	O	**Transmit Data** This signal is the transmit output from the UART.
RXD	I	**Receive Data** This signal is the receive input for the UART.
$\overline{\text{CTS}}$	I	**Clear To Send** This signal is the flow control input for the UART.
DE	O	**Driver Enable** This signal allows automatic control of external RS-485 drivers. This signal is approximately the inverse of the TXE (Transmit Empty) bit in the UART Status 0 Register. The DE signal can be used to ensure the external RS-485 driver is enabled when data is transmitted by the UART.
I^{2}C Controller		
SCL	I/O	**I^{2}C Serial Clock** The I^{2}C Master supplies this signal. If the F3224 Series is the I^{2}C Master, this pin is an output and if it is I^{2}C slave, this pin is an input. When the GPIO pin is configured for alternate function to enable the SCL function, this pin is open-drain.
SDA	I/O	**I^{2}C Serial Data** This open-drain pin transfers data between the I^{2}C and an external I^{2}C Master/Slave. When the GPIO pin is configured for alternate function to enable the SDA function, this pin is open-drain.
ESPI Controller		
$\overline{\text{SS}}$	I/O	**Slave Select** This signal can be an output or an input. If the F3224 Series is the SPI master, $\overline{\text{SS}}$ may be configured as the Slave Select output, and if it is the SPI slave, $\overline{\text{SS}}$ is the input slave select.

Table 4. Signal Descriptions (Continued)

Signal Mnemonic	I/O	Description
SCK	I/O	**SPI Serial Clock** The SPI master 0 supplies this signal. If the F3224 Series is the SPI master, SCLK is output and if it is the SPI slave, SCLK is an input.
MOSI	I/O	**Master Out Slave In** This signal is the data output from the SPI master device and the data input to the SPI slave device.
MISO	I/O	**Master In Slave Out** This pin ise the data input to the SPI master device and the data output from the SPI slave device.
Event System		
ESOUT[3:0]	O	**Event System Outputs**
Timers		
T0OUT/T1OUT/ T2OUT	O	**Timer Output 0–2** These signals are output from the timers.
$\overline{\text{T0OUT}}$/$\overline{\text{T1OUT}}$/ $\overline{\text{T2OUT}}$	O	**Timer Complement Output 0–2** These signals are output from the timers in PWM Dual Output Mode.
T0IN/T1IN/T2IN	I	**Timer Input 0–2** These signals are used as the capture, gating, and counter inputs. The T0IN/T1IN/T2IN signal is multiplexed with $\overline{\text{T0OUT}}$/$\overline{\text{T1OUT}}$/$\overline{\text{T2OUT}}$ signals.
Multichannel Timers		
T4CHA, T4CHB, T4CHC, T4CHD	I/O	**Multichannel Timer Input/Output** These signals function as Capture input or Compare output for channels CHA, CHB, CHC, and CHD.
T4IN	I	**Multichannel Timer clock input** This signal allows external input to serve as the clock source for the Multichannel timer.
Comparators		
C0INxP/C0INxN, C1INxP/C1INxN	I	**Comparator Inputs** These signals are positive and negative inputs to the comparator 0 and comparator 1 (x = 0-1).
C0OUT/C1OUT	O	**Comparator Outputs** These are the output from the comparator 0 and the comparator 1.
Analog		
ANA[14:0]	I	**Analog Port** These signals are used as inputs to the ADC. The ANA2, ANA3, and ANA4 pins can also access the inputs and outputs of the integrated Op Amp A0 The ANA5, ANA6, and ANA7 pins can also access the inputs and outputs of the integrated Op Amp A1.

Table 4. Signal Descriptions (Continued)

Signal Mnemonic	I/O	Description
V_{REF}+/V_{REF}–	I/O	**ADC Reference Voltage** Note: A decoupling capacitor should be connected between V_{REF}+ and V_{REF}– as specified in Table 239 on page 409.
VBIAS	O	**Voltage Bias with Low Current Drive Capability**
Operational Amplifiers, Op Amp A0 and Op Amp A1		
AMPA0INP, AMPA0INN, AMPA1INP, AMPA1INN	I	**Low-Power Operational Amplifier Inputs for Op Amp A0 and Op Amp A1** If enabled, these pins drive the positive and negative Operational Amplifier inputs respectively.
AMPA0OUT AMPA1OUT	O	**Low-Power Operational Amplifier Outputs for Op Amp A0 and Op Amp A1** If enabled, this pin is driven by the on-chip, low-power Operational Amplifier.
Oscillator		
$X2_{IN}$	I	**Low Frequency Crystal Input** The input pin to the Low Frequency Crystal Oscillator that operates at 32.768 kHz. A crystal can be connected between the X2IN and the X2OUT pin to form the oscillator.
$X2_{OUT}$	O	**Low Frequency Crystal Output** This pin is the output from the Low Frequency Crystal Oscillator. A crystal can be connected between the X2IN and the X2OUT pin to form the oscillator.
Clock Input/Output		
CLKIN	I	**Clock Input (Drive) Signal** This pin may be used to input a CMOS-level signal to be used as System Clock.
CLK2IN	I	**Clock 2 Input (Drive) Signal** This pin may be used to input a CMOS-level signal to be used as the Peripheral Clock.
SCK_{OUT}	O	**System Clock Output Signal**
On-Chip Debugger		
DBG	I/O	**Debug** This signal is the control and data input and output of the On-Chip Debugger. **CAUTION:** The DBG pin is open-drain and features an enabled internal pull-up resistor. An additional external pull-up resistor is recommended to ensure proper operation.
Reset		

Table 4. Signal Descriptions (Continued)

Signal Mnemonic	I/O	Description
$\overline{\text{RESET}}$	I/O	**RESET** Generates a Reset when asserted (driven Low). Also serves as a Reset indicator; the Z8 Encore! XP forces this pin Low when in Reset. This pin is open-drain and features an enabled internal pull-up resistor.
Power Supply		
V_{DD}	I	Digital Power Supply.
AV_{DD}	I	Analog Power Supply. AV_{DD} must be at the same potential as V_{DD}.
V_{SS}	I	Digital Ground.
AV_{SS}	I	Analog Ground. AV_{SS} must be at the same potential as V_{SS}.
V_{CORE}	I/O	Regulated core power supply (external current loading not permitted). Note: V_{CORE} should be connected to a 4.7µF decoupling capacitor.
External Pad		
EP		Bottom side Exterior Pad on QFN package This exterior pad is only available on the QFN package and it must be connected to V_{SS}.

2.3. Pin Characteristics

Table 5 lists the characteristics of each available pin on F3224 Series devices. The data in this table is sorted alphabetically by pin symbol mnemonic.

Table 5. Pin Characteristics

Symbol Mnemonic	Direction	Reset Direction	Active Low or Active High	Tristate Output	Internal Pull-Up or Pull-Down	Schmitt Trigger Input	Open-Drain Output
AV_{DD}	N/A	N/A	N/A	N/A	N/A	N/A	N/A
AV_{SS}	N/A	N/A	N/A	N/A	N/A	N/A	N/A
DBG	I/O	I	N/A	Yes	Yes	Yes	Yes
PA[7:0]	I/O	I	N/A	Yes	Programmable pull-up	Yes	Yes, programmable
PB[7:0]	I/O	I	N/A	Yes	Programmable pull-up	Yes	Yes, programmable
PC[7:0]	I/O	I	N/A	Yes	Programmable pull-up	Yes	Yes, programmable
PD[7:1]	I/O	I	N/A	Yes	Programmable pull-up	Yes	Yes, programmable
PD0/ $\overline{\text{RESET}}$	I/O	I/O (defaults to $\overline{\text{RESET}}$)	Low (in RESET mode)	Yes (PD0 only)	Programmable for PD0; always ON for $\overline{\text{RESET}}$	Yes	Programmable for PD0; always ON for $\overline{\text{RESET}}$
PE[3:0]	I/O	I	N/A	Yes	Programmable pull-up	Yes	Yes, programmable
V_{CORE}	N/A	N/A	N/A	N/A	N/A	N/A	N/A
V_{DD}	N/A	N/A	N/A	N/A	N/A	N/A	N/A
V_{SS}	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Chapter 3. Address Space

The eZ8 CPU can access the following three distinct address spaces:

- The Register File contains addresses for general-purpose registers, eZ8 CPU, peripherals, and GPIO port control registers
- The Program Memory contains addresses for all memory locations having executable code and/or data
- The Data Memory contains addresses for all memory locations that contain data only

These three address spaces are covered briefly in the following sections. To learn more about the eZ8 CPU and its address space, refer to the eZ8 CPU Core User Manual (UM0128), which is available free for download from the Zilog website.

3.1. Register File

The Register File address space in the Z8 Encore!® MCU is 4KB (4096 bytes). The Register File is composed of two sections: control registers and general-purpose registers. When instructions are executed, registers defined as sources are read, and registers defined as destinations are written. The architecture of the eZ8 CPU allows all general-purpose registers to function as accumulators, address pointers, index registers, stack areas, or scratch pad memory.

The upper 256 bytes of the 4KB Register File address space are reserved for control of the eZ8 CPU, on-chip peripherals, and the input/output ports. These registers are located in the `F00h` to `FFFh` address range. Some of the addresses within the 256B control register sections are reserved; i.e., unavailable. Reading from a reserved Register File address returns an undefined value. Zilog does not recommend writing to the reserved Register File addresses because doing so can produce unpredictable results.

On-chip Register RAM always begins at address `000h` in the Register File address space. F3224 Series devices contain 3.75KB of on-chip Register RAM.

3.2. Program Memory

The eZ8 CPU supports 64KB of Program Memory address space. The F3224 Series devices contain 16 KB to 32 KB of on-chip Flash memory in the Program Memory address space, depending on the device.

Reading from Program Memory addresses present outside the available Flash memory returns `FFh`. Writing to these unimplemented Program Memory addresses produces no effect. Table 6 lists the Program Memory maps for the F3224 Series products.

Table 6. F3224 Series Program Memory Maps

Program Memory Address (Hex)	Function
Z8F3224 Product	
0000–0001	Flash option bits
0002–0003	Reset vector
0004–0005	WDT interrupt vector
0006–0007	Illegal instruction trap
0008–0047	Interrupt vectors*
0048–0049	Oscillator fail trap*
004A–7FFF	Program Flash
Z8F1624 Product	
0000–0001	Flash option bits
0002–0003	Reset vector
0004–0005	WDT interrupt vector
0006–0007	Illegal instruction trap
0008–0047	Interrupt vectors*
0048–0049	Oscillator fail trap*
004A–3FFF	Program Flash

Note: *See Table 45 on page 90 for a list of interrupt vectors and traps.

3.3. Data Memory

F3224 Series MCUs do not use the eZ8 CPU's 64KB Data Memory address space.

3.4. Flash Information Area

Table 7 lists the F3224 Series Flash Information Area. This 512 B space consists of four pages and is accessed by setting bit 7 of the Flash Page Select Register to 1. When access is enabled, the Flash Information Area is mapped into Program Memory and overlays the `FE00h` to `FFFFh` address range. When Information Area access is enabled, all reads from these Program Memory addresses return the Information Area data rather than the Program Memory data. Access to the Flash Information Area is read-only.

Table 7. F3224 Series Flash Memory Information Area Map

Program Memory Address (Hex)	Function
FE00–FE3F	Zilog option and trim bits.
FE40–FFFF	Reserved.

Chapter 4. Register Map

Table 8 provides the address map for the Register File of F3224 Series devices. Not all devices and package styles in the F3224 Series support the Multi-Channel Timer or all of the GPIO Ports. Consider registers for unimplemented peripherals as Reserved.

Table 8. Register File Address Map

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page #
General-Purpose RAM				
000–EFF	General-Purpose Register File RAM	–	XX	
Special-Purpose Registers				
Timer 0				
F00	Timer 0 High Byte	T0H	00	135
F01	Timer 0 Low Byte	T0L	01	135
F02	Timer 0 Reload High Byte	T0RH	FF	136
F03	Timer 0 Reload Low Byte	T0RL	FF	136
F04	Timer 0 PWM0 High Byte	T0PWM0H	00	137
F05	Timer 0 PWM0 Low Byte	T0PWM0L	00	137
F06	Timer 0 Control 0	T0CTL0	00	139
F07	Timer 0 Control 1	T0CTL1	00	140
F20	Timer 0 PWM1 High Byte	T0PWM1H	00	138
F21	Timer 0 PWM1 Low Byte	T0PWM1L	00	138
F22	Timer 0 Control 2	T0CTL2	00	144
F23	Timer 0 Status	T0STA	00	145
F2C	Timer 0 Noise Filter Control	T0NFC	00	146
Timer 1				
F08	Timer 1 High Byte	T1H	00	135
F09	Timer 1 Low Byte	T1L	01	135
F0A	Timer 1 Reload High Byte	T1RH	FF	136
Timer 1 (Continued)				
F0B	Timer 1 Reload Low Byte	T1RL	FF	136
F0C	Timer 1 PWM0 High Byte	T1PWM0H	00	137
F0D	Timer 1 PWM0 Low Byte	T1PWM0L	00	137

Table 8. Register File Address Map (Continued)

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page #
F0E	Timer 1 Control 0	T1CTL0	00	139
F0F	Timer 1 Control 1	T1CTL1	00	140
F24	Timer 1 PWM1 High Byte	T1PWM1H	00	138
F25	Timer 1 PWM1 Low Byte	T1PWM1L	00	138
F26	Timer 1 Control 2	T1CTL2	00	144
F27	Timer 1 Status	T1STA	00	145
F2D	Timer 1 Noise Filter Control	T1NFC	00	146
Timer 2				
F10	Timer 2 High Byte	T2H	00	135
F11	Timer 2 Low Byte	T2L	01	135
F12	Timer 2 Reload High Byte	T2RH	FF	136
F13	Timer 2 Reload Low Byte	T2RL	FF	136
F14	Timer 2 PWM0 High Byte	T2PWM0H	00	137
F15	Timer 2 PWM0 Low Byte	T2PWM0L	00	137
F16	Timer 2 Control 0	T2CTL0	00	139
F17	Timer 2 Control 1	T2CTL1	00	140
F18–F1F	Reserved	–	XX	
F28	Timer 2 PWM1 High Byte	T2PWM1H	00	140
F29	Timer 2 PWM1 Low Byte	T2PWM1L	00	138
F2A	Timer 2 Control 2	T2CTL2	00	144
F2B	Timer 2 Status	T2STA	00	145
F2E	Timer 2 Noise Filter Control	T2NFC	00	146
F2F	Reserved	–	XX	
Timer A				
F30	Timer A High Byte	TAH	00	153
F31	Timer A Low Byte	TAL	00	153
F32	Timer A Reload High Byte	TARH	FF	154
F33	Timer A Reload Low Byte	TARL	FF	154
F34	Timer A Control	TACTL	00	154
F35	Timer A Prescale	TAPS	00	155
F36-F3F	Reserved	–	XX	

Table 8. Register File Address Map (Continued)

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page #
LDD UART				
F40	LDD UART Transmit Data	UTXD	XX	208
	LDD UART Receive Data	URXD	XX	208
F41	LDD UART Status 0 – Standard UART Mode	USTAT0	0000011Xb	209
	LDD UART Status 0 – LIN Mode	USTAT0	00000110b	210
	LDD UART Status 0 – DALI Mode	USTAT0	0000011Xb	212
	LDD UART Status 0 – DMX Mode	USTAT0	0000011Xb	213
F42	LDD UART Control 0	UCTL0	00	217
F43	LDD UART Control 1 – Multiprocessor Control	UCTL1	00	219
	LDD UART Control 1 – Noise Filter Control	UCTL1	00	221
	LDD UART Control 1 – LIN Control	UCTL1	00	222
	LDD UART Control 1 – DALI Control	UCTL1	00	223
	LDD UART Control 1 – DMX Control	UCTL1	00	224
F44	LIN UART Mode Select and Status	UMDSTAT	00	215
F45	UART Address Compare	UADDR	00	226
F46	UART Baud Rate High Byte	UBRH	FF	226
F47	UART Baud Rate Low Byte	UBRL	FF	227
F48-F4F	Reserved	–	XX	
I^{2}C				
F50	I^{2}C Data	I2CDATA	00	276
F51	I^{2}C Interrupt Status	I2CISTAT	80	277
F52	I^{2}C Control	I2CCTL	00	278
F53	I^{2}C Baud Rate High Byte	I2CBRH	FF	280
F54	I^{2}C Baud Rate Low Byte	I2CBRL	FF	280
F55	I^{2}C State	I2CSTATE	02	281
F56	I^{2}C Mode	I2CMODE	00	284
F57	I^{2}C Slave Address	I2CSLVAD	00	286
F58-F5F	Reserved	–	XX	
Enhanced Serial Peripheral Interface (ESPI)				
F60	ESPI Data	ESPIDATA	XX	245
F61	ESPI Transmit Data Command	ESPITDCR	00	246

Table 8. Register File Address Map (Continued)

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page #
F62	ESPI Control	ESPICTL	00	246
F63	ESPI Mode	ESPIMODE	00	248
F64	ESPI Status	ESPISTAT	81	251
F65	ESPI State	ESPISTATE	00	252
F66	ESPI Baud Rate High Byte	ESPIBRH	FF	254
F67	ESPI Baud Rate Low Byte	ESPIBRL	FF	255
F68-F6F	Reserved	–	XX	
Analog-to-Digital Converter (ADC)				
F70	ADC Control 0	ADCCTL0	00	317
F71	ADC Control 1	ADCCTL1	00	318
F72	ADC Control 2	ADCCTL2	00	319
F73	ADC Input Select High	ADCINSH	00	320
F74	ADC Input Select Low	ADCINSL	00	320
F75	ADC Offset Calibration High	ADCOFF	00	323
F76	ADC Data High	ADCD_H	00	324
F77	ADC Data Low	ADCD_L	00	325
F78	Sample Time	ADCST	00	326
F79	ADC Upper Window Threshold High	ADCUWINH	FF	326
F7A	ADC Upper Window Threshold Low	ADCUWINL	FF	327
F7B	ADC Lower Window Threshold High	ADCLWINH	00	328
F7C	ADC Lower Window Threshold Low	ADCLWINL	00	329
Reference System				
F7D	Fixed Reference Control Register	FREFCTL	00	304
F7E	Programmable Reference 0 Control	PREF0CTL	00	305
F7F	Programmable Reference 1 Control	PREF1CTL	00	305
Low-Power Control				
F80	Power Control 0	PWRCTL0	10	43
F81	Power Control 1	PWRCTL1	00	45
Clock System				
F82	Clock Control 0	CLKCTL0	28	81
F83	Clock Control 1	CLKCTL1	01	83
F84	Clock Control 2	CLKCTL2	00	84

Table 8. Register File Address Map (Continued)

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page #
F85	Clock Control 3	CLKCTL3	3D	84
F86	Clock Control 4	CLKCTL4	00	85
F87	Clock Control 5	CLKCTL5	05	86
F88	Clock Control 6	CLKCTL6	00	87
F89	Clock Control 7	CLKCTL7	00	87
F8A	Clock Control 8	CLKCTL8	XX	88
F8B	Clock Control 9	CLKCTL9	XX	88
F8C-F8E	Reserved	–	XX	
Comparators				
F8F	Comparator Control	CMPCTL	00	341
F90	Comparator 0 Control 0	CMP0CTL0	00	342
F91	Comparator 0 Control 1	CMP0CTL1	00	343
F92	Comparator 1 Control 0	CMP1CTL0	00	344
F93	Comparator 1 Control 1	CMP1CTL1	00	345
Op Amp_A0				
F94	Op Amp_A0 Control 0	AMPA0CTL0	00	334
F95	Op Amp_A0 Control 1	AMPA0CTL1	00	335
Op Amp_A1				
F96	Op Amp_A1 Control 0	AMPA1CTL0	00	334
F97	Op Amp_A1 Control 1	AMPA1CTL1	00	335
Event System				
F98	Event System Source Subaddress	ESSSA	00	294
F99	Event System Source Subdata	ESSSD	00	295
F9A	Event System Destination Subaddress	ESDSA	00	297
F9B	Event System Destination Subdata	ESDSD	00	298
F9C-F9F	Reserved			
Multi-Channel Timer				
FA0	MCT High Byte	MCTH	00	167
FA1	MCT Low Byte	MCTL	00	167
FA2	MCT Reload High Byte	MCTRH	FF	168
FA3	MCT Reload Low Byte	MCTRL	FF	168
FA4	MCT Subaddress	MCTSA	XX	168

Table 8. Register File Address Map (Continued)

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page #
FA5	MCT Subregister 0	MCTSR0	XX	169
FA6	MCT Subregister 1	MCTSR1	XX	169
FA7	MCT Subregister 2	MCTSR2	XX	169
FA8-FBF	Reserved	–	XX	
Interrupt Controller				
FC0	Interrupt Request 0	IRQ0	00	95
FC1	IRQ0 Enable High Bit	IRQ0ENH	00	99
FC2	IRQ0 Enable Low Bit	IRQ0ENL	00	100
FC3	Interrupt Request 1	IRQ1	00	96
FC4	IRQ1 Enable High Bit	IRQ1ENH	00	101
FC5	IRQ1 Enable Low Bit	IRQ1ENL	00	102
Interrupt Controller (Continued)				
FC6	Interrupt Request 2	IRQ2	00	97
FC7	IRQ2 Enable High Bit	IRQ2ENH	00	103
FC8	IRQ2 Enable Low Bit	IRQ2ENL	00	103
FC9	Interrupt Request 3	IRQ3	00	98
FCA	IRQ3 Enable High Bit	IRQ3ENH	00	104
FCB	IRQ3 Enable Low Bit	IRQ3ENL	00	105
FCC	Interrupt Edge Select	IRQES	00	106
FCD	Shared Interrupt Select 0	IRQSS0	00	107
FCE	Reserved	–	XX	
FCF	Interrupt Control	IRQCTL	00	108
GPIO Port A				
FD0	Port A Address	PAADDR	00	59
FD1	Port A Control	PACTL	00	60
FD2	Port A Input Data	PAIN	XX	68
FD3	Port A Output Data	PAOUT	00	69
GPIO Port B				
FD4	Port B Address	PBADDR	00	59
FD5	Port B Control	PBCTL	00	60
FD6	Port B Input Data	PBIN	XX	68
FD7	Port B Output Data	PBOUT	00	69

Table 8. Register File Address Map (Continued)

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page #
GPIO Port C				
FD8	Port C Address	PCADDR	00	59
FD9	Port C Control	PCCTL	00	60
FDA	Port C Input Data	PCIN	XX	68
FDB	Port C Output Data	PCOUT	00	69
GPIO Port D				
FDC	Port D Address	PDADDR	00	59
FDD	Port D Control	PDCTL	00	60
FDE	Port D Input Data	PDIN	XX	68
FDF	Port D Output Data	PDOUT	00	69
GPIO Port E				
FE0	Port E Address	PEADDR	00	59
FE1	Port E Control	PECTL	00	60
FE2	Port E Input Data	PEIN	XX	68
FE3	Port E Output Data	PEOUT	00	69
FE4-FEF	Reserved	–	XX	
Reset				
FF0	Reset Status	RSTSTAT	XX	39
FF1	Reserved	–	XX	
Watchdog Timer				
FF2	Watchdog Timer Reload High Byte	WDTH	FF	179
FF3	Watchdog Timer Reload Low Byte	WDTL	FF	179
FF4–FF5	Reserved	–	XX	
Trim Bit Control				
FF6	Trim Bit Address	TRMADR	00	361
FF7	Trim Data	TRMDR	XX	362
Flash Memory Controller				
FF8	Flash Control	FCTL	00	354
	Flash Status	FSTAT	00	355
FF9	Flash Page Select	FPS	00	356
	Flash Block Protect	FPROT	00	357
FFA-FFB	Reserved	–	XX	

Table 8. Register File Address Map (Continued)

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page #
eZ8 CPU				
FFC	Flags	–	XX	Refer to the eZ8 CPU Core User Manual (UM0128)
FFD	Register Pointer	RP	XX	
FFE	Stack Pointer High Byte	SPH	XX	
FFF	Stack Pointer Low Byte	SPL	XX	

Note: XX = undefined.

Chapter 5. Reset, Stop-Mode Recovery and Low-Voltage Detection

The Reset Controller within the F3224 Series MCU controls Reset and Stop-Mode Recovery operations and provides indication of low-voltage supply conditions. During the operation, the following events cause a Reset:

- Power-On Reset (POR)
- Voltage Brown-Out (VBO) protection
- Watchdog Timer (WDT) time-out (when configured by the WDT_RES Flash option bit to initiate a Reset)
- External $\overline{\text{RESET}}$ pin assertion (when the alternate function, RESET, is enabled by the GPIO register)
- On-Chip Debugger initiated Reset

When the device is in Stop Mode, a Stop-Mode Recovery can be initiated by each of the following triggers:

- Watchdog Timer time-out
- GPIO Port input pin transition on an enabled Stop-Mode Recovery source
- Interrupt from a timer, comparator or Low Voltage Detection operating in Stop Mode

The low-voltage detection circuitry on the device offers the following features:

- The low-voltage detection threshold level is user-defined
- It generates an interrupt when the supply voltage drops below a user-defined level

5.1. Reset Types

The F3224 Series MCU provides multiple types of Reset operation. Stop-Mode Recovery is considered a form of Reset. Table 9 lists the types of Reset and their operating characteristics.

Table 9. Reset, Stop-Mode Recovery Characteristics and Latency

	Reset Characteristics and Latency		
Reset Type	**Control Registers**	**eZ8 CPU**	**Reset Latency (Delay)**
System Reset (non-POR/VBO Reset)	Reset (as applicable)	Reset	10ms Reset Delay
System Reset (POR/VBO Reset)	Reset (as applicable)	Reset	10ms Reset Delay
Stop-Mode Recovery (standard, FRECOV = 0)	Unaffected, except RSTSTAT, CLKCTL0, CLKCTL5, and IRQCTL registers	Reset	6 System Clock (DCO selected) cycles after Stop-Mode Recovery Delay
Stop-Mode Recovery (fast, FRECOV =1)	Unaffected, except RSTSTAT CLKCTL0, CLKCTL5, and IRQCTL registers	Reset	6 System Clock (DCO selected) cycles

5.2. System Reset

During a System Reset, the IPO, DCO and FLL are enabled. Because the control registers are reinitialized by a System Reset, the system clock frequency after reset is approximately 1 MHz based on the DCO configured to run at approximately 8MHz and a system clock division ratio of 8. In addition, the IPO selected as Peripheral Clock (Pclk) to which the FLL locks the DCO. Upon the conclusion of a System Reset, the System Clock source and clock settings can be configured as desired. To learn more, see the Clock System chapter on page 70.

When System Reset occurs due to a VBO condition, the Reset Delay commences when the supply voltage first exceeds the VBO level (discussed later in this chapter). When System Reset occurs due to a POR condition, the Reset Delay commences from when the supply voltage first exceeds both the the POR and VBO levels. If the external $\overline{\text{RESET}}$ pin remains asserted at the end of the Reset period, the device remains in System Reset until the pin is deasserted.

At the beginning of System Reset, all GPIO pins are configured as inputs with pull-up resistor disabled, except PD0 that is shared with the $\overline{\text{RESET}}$ pin. On Reset, the Port D0 pin is configured as a bidirectional open-drain Reset. The pin is internally driven Low during port reset, after which the user code can reconfigure this pin as a general-purpose output.

During Reset, the eZ8 CPU and on-chip peripherals are idle; however, the Internal Precision Oscillator (IPO) continues to function.

On System Reset, control registers within the Register File that have a defined Reset value are loaded with their Reset values. Other control registers (including the Stack Pointer, Register Pointer and Flags) and general-purpose RAM are not initialized and undefined following System Reset. The eZ8 CPU fetches the Reset vector at Program Memory addresses `0002h` and `0003h` and loads that value into the Program Counter. Program execution begins at the Reset vector address.

Note: After a System Reset or Stop-Mode Recovery, an external crystal oscillator may be unstable. Use software to wait until it is stable before using it as a clock source.

Table 10 lists the possible sources of a System Reset.

Table 10. System Reset Sources and Resulting Reset Type

Operating Mode	System Reset Source	Special Conditions
Active or Halt Mode	Power-On Reset	Reset delay begins after supply voltage exceeds POR and VBO levels.
	Voltage Brown-Out	Reset delay begins after supply voltage exceeds VBO level.
	Watchdog Timer time-out when configured for Reset	Reset delay begins upon Watchdog Timer time-out.
	$\overline{\text{RESET}}$ pin assertion	Reset delay begins after $\overline{\text{RESET}}$ pin assertion. All reset pulses less than three system clocks in width are ignored, see the Electrical Characteristics chapter on page 400.
	On-Chip Debugger initiated Reset	OCD is unaffected
Stop Mode	Power-On Reset	Reset delay begins after supply voltage exceeds POR and VBO levels.
	Voltage Brown-Out	Reset delay begins after supply voltage exceeds VBO level.
	$\overline{\text{RESET}}$ pin assertion	Reset delay begins after $\overline{\text{RESET}}$ pin assertion. All reset pulses less than the specified analog delay are ignored, see the Electrical Characteristics chapter on page 400.
	DBG pin driven Low	None.

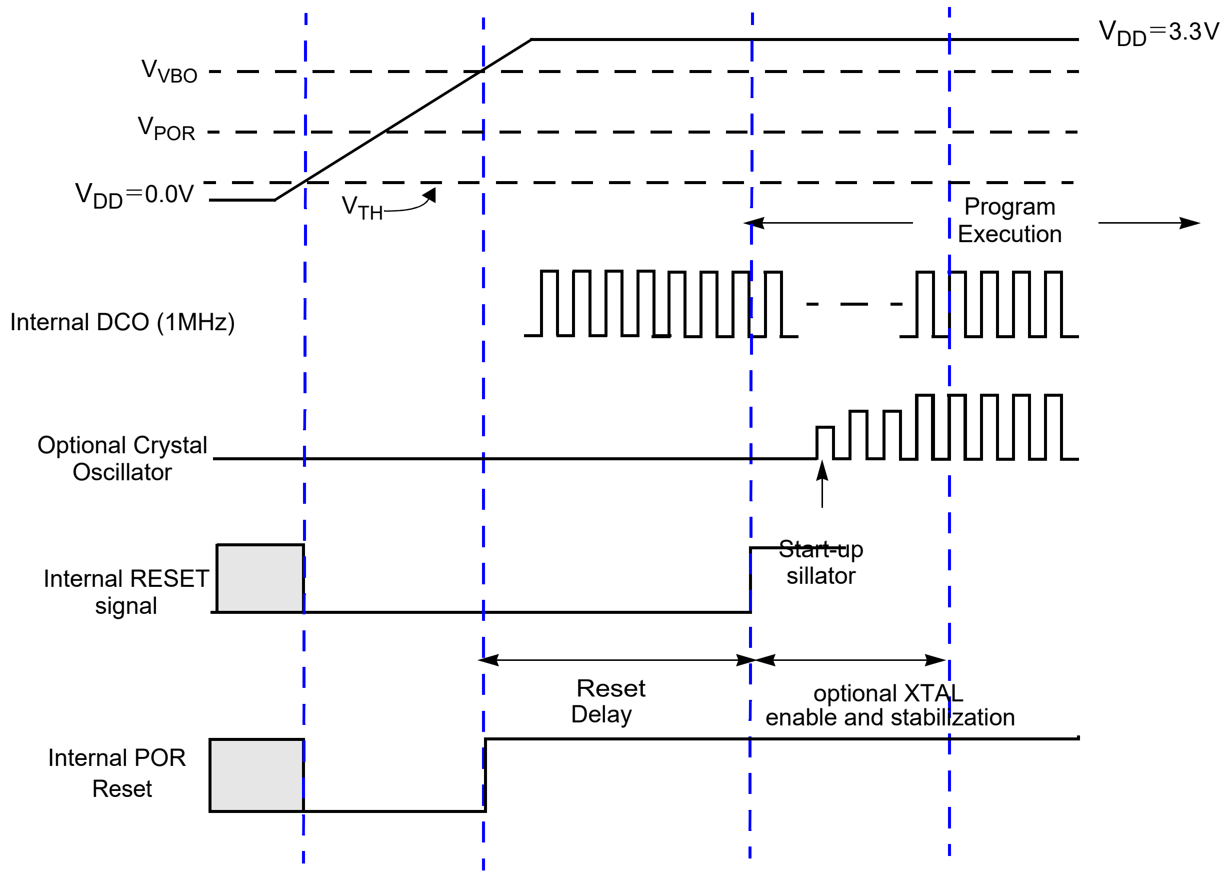
5.2.1. Power-On Reset

Each device in the F3224 Series contains an internal Power-On Reset (POR) circuit. The POR circuit monitors the supply voltage and holds the whole device in the Reset state until the supply voltage reaches a safe circuit operating level when the device is powered on. V_{DD} must be greater than both V_{POR} and V_{VBO} to exit the Reset state.

After power on, the POR circuit keeps idle until the supply voltage drops below V_{TH} voltage. Figure 6 on page 33 shows this POR behavior.

After the F3224 Series MCU exits the POR state, the eZ8 CPU fetches the Reset vector. Following this POR, the POR/VBO status bit in the Reset Status Register is set to 1.

For the POR threshold voltage (V_{POR}) and POR start voltage V_{TH} , see the Electrical Characteristics chapter on page 400.



Notes
1. Not to Scale.
2. Internal Reset and POR Reset are Low active.

▨ undefined

Figure 4. Power-On Reset Operations

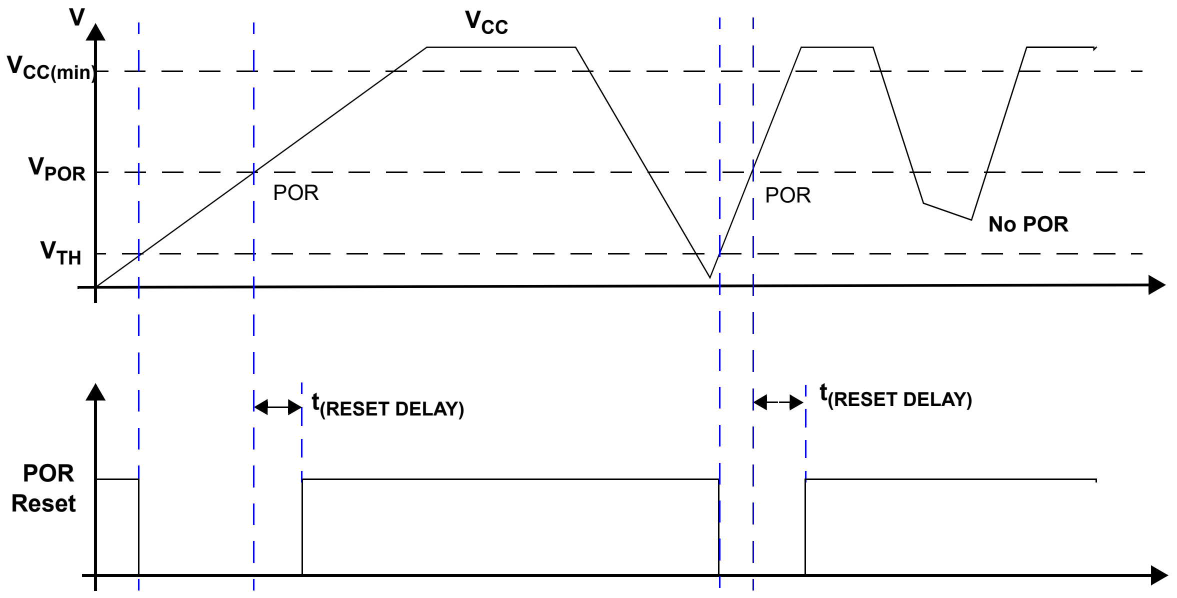


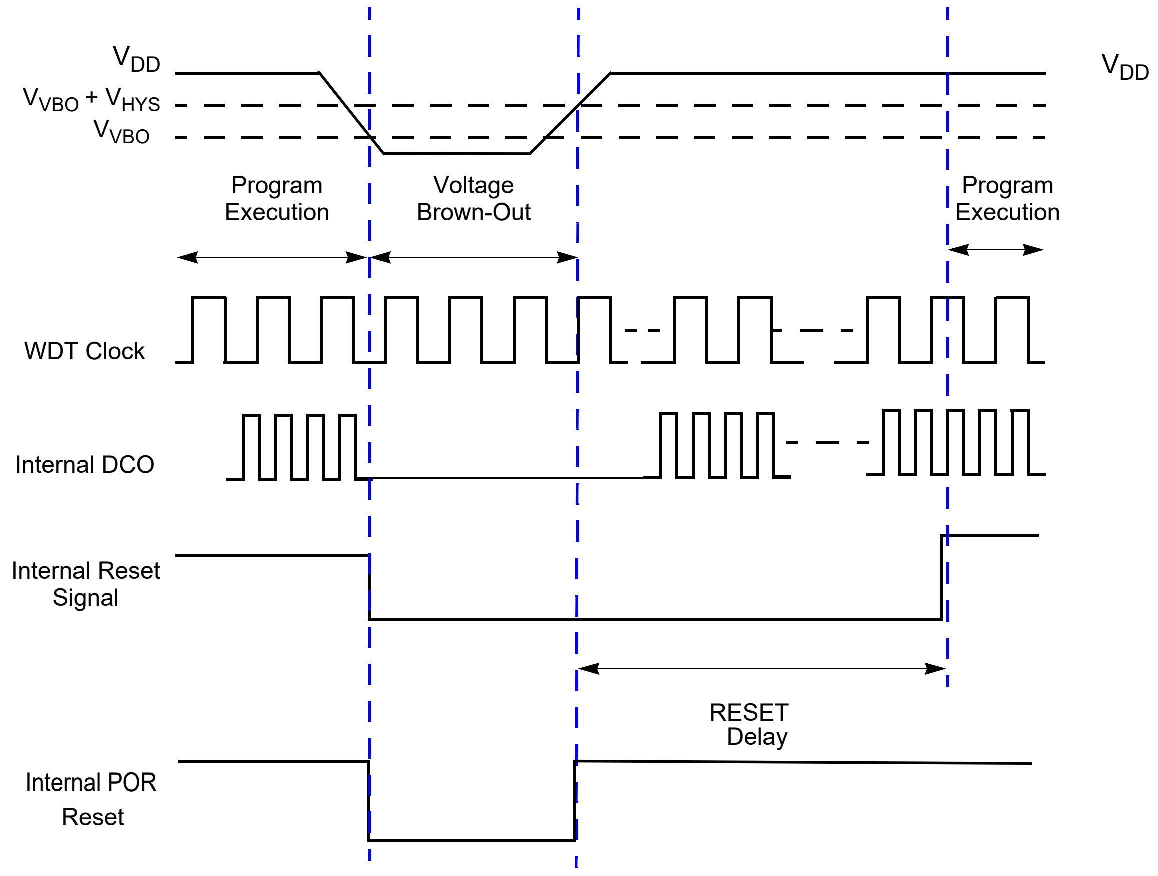
Figure 6. Power-On Reset Behavior

5.2.2. Voltage Brown-Out Reset

The F3224 Series MCU provides a VBO Reset feature for low-voltage protection. The VBO circuit has a preset threshold voltage (V_{VBO}) with a hysteresis of V_{HYS}. The VBO circuit will monitor the power supply voltage if the VBO is enabled. When the VBO Reset circuit detects the power supply voltage falls below the threshold voltage V_{VBO}, the VBO resets the device by pulling the POR Reset from 1 to 0. The VBO will hold the POR Reset until the power supply voltage goes above the V_{VBO+} (V_{VBO}+V_{HYS}), at which time the VBO Reset is released. The device progresses through a System Reset sequence, as occurred with the POR. Following this System Reset sequence, the POR/VBO status bit in the Reset Status (RSTSTAT) Register is set to 1. Figure 7 on page 34 illustrates this VBO Reset operation.

For VBO threshold voltages (V_{VBO}) and VBO hysteresis (V_{HYS}), see the Electrical Characteristics chapter on page 400.

The VBO circuit is either enabled or disabled during Stop Mode. If enabled during Stop Mode, the VBO circuit operates only periodically to reduce current consumption. VBO circuit operation is controlled by the VBOCTL Flash option bits; to learn more, see the Flash Option Bits chapter on page 358. During a POR, the VBO is initially enabled, but is subsequently controlled by VBOCTL upon exit from System Reset.



Note: this figure is not to scale.

Figure 7. Voltage Brown-Out Reset Operation

5.2.3. Watchdog Timer Reset

If the device is in Active Mode or Halt Mode, WDT initiates a System Reset at time-out if the WDT_RES Flash option bit is programmed to 1. This state is the unprogrammed state of the WDT_RES Flash option bit. If the bit is programmed to 0, the WDT is configured to cause an interrupt (WDT_RES = 0 in Flash Option Bits at Program Memory Address `0000h`), not a System Reset at time-out. The WDT status bit in the Reset Status Register is set to signify that the reset was initiated by the WDT.

5.2.4. External Reset Input

The $\overline{\text{RESET}}$ pin has a Schmitt-triggered input and an internal pull-up resistor. When the $\overline{\text{RESET}}$ pin is asserted for a minimum of four system clock cycles, the device progresses through the System Reset sequence. Because of possible asynchronicity of the system clock and reset signals, the required reset duration can be as short as three clock periods and as long as four. A reset pulse three clock cycles in duration could trigger a Reset; a pulse four cycles in duration always triggers a Reset.

While the $\overline{\text{RESET}}$ input pin is asserted Low, the F3224 Series MCU remains in the Reset state. If the $\overline{\text{RESET}}$ pin is held Low beyond the System Reset time-out, the device exits the Reset state on the system clock rising edge following $\overline{\text{RESET}}$ pin deassertion. Following a System Reset initiated by the external $\overline{\text{RESET}}$ pin, the EXT status bit in the RSTSTAT Register is set to 1.

5.2.5. External Reset Indicator

During System Reset, or when enabled by the GPIO logic (see the Port A–E Control Registers section on page 60), the $\overline{\text{RESET}}$ pin functions as an open-drain (active Low) reset mode indicator in addition to the input functionality. This Reset output feature allows the F3224 Series MCU to reset other components to which it is connected, even if that reset is caused by internal sources such as POR, VBO, or WDT events.

After an internal Reset event occurs, the internal circuitry begins driving the $\overline{\text{RESET}}$ pin Low. The $\overline{\text{RESET}}$ pin is held Low by the internal circuitry until the appropriate delay (listed in Table 9 on page 30) has elapsed.

5.2.6. On-Chip Debugger Initiated Reset

A POR can be initiated using the OCD. The OCD block is not reset, but the remainder of the chip goes through a normal System Reset. The RST bit automatically clears during the system reset. Following the System Reset the POR bit in the Reset Status Register is set.

5.3. Stop-Mode Recovery

Stop Mode is entered by execution of a STOP instruction by the eZ8 CPU. For detailed Stop Mode information, see the Low-Power Modes section on page 41. Stop-Mode Recovery does not affect on-chip registers other than the Reset Status (RSTSTAT), Clock Control 0 (CLKCTL0), Clock Control 5 (CLKCTL5) and Interrupt Control (IRQCTL) registers.

During Stop-Mode Recovery, the DCO is configured with the most recent DCO delay control code, and is selected as System Clock with the FLL disabled. If the FLL or another system clock source is required, the Stop-Mode Recovery code must reconfigure the Clock System such that the desired system clock source is enabled and selected. To learn more, see the Clock System chapter on page 70.

Note: After a System Reset or Stop-Mode Recovery, an external crystal oscillator may become unstable. Use software to wait until it is stable before using this crystal as a clock source.

The IPO, LFXO, or external clock drive, when enabled, can be configured to remain operating during Stop Mode (PCKSM = 1 in the CLKCTL1 Register) or to be nonoperating during Stop Mode (PCKSM = 0 in the CLKCTL1 Register). If enabled and configured to to be nonoperating during Stop Mode, the clock source will become operational during Stop-Mode Recovery. The FLL is always disabled by entry into Stop Mode and, if required during Normal Operation, must be enabled by software after Stop-Mode Recovery.

Stop-Mode Recovery latency is a function of FRECOV in the Power Control Register 0 (PWRCTL0, see the Power Control Register Definitions section on page 43), If FRECOV is set, the Stop-Mode Recovery latency is 6 System Clock cycles. If FRECOV is cleared, the Stop-Mode Recovery latency is the Stop-Mode Recovery Delay plus 6 System Clock cycles. To learn more, see Stop-Mode Recovery Delay in the Electrical Characteristics chapter on page 400.

The eZ8 CPU fetches the Reset vector at Program Memory addresses `0002h` and `0003h` and loads that value into the Program Counter. Program execution begins at the Reset vector address. Following Stop-Mode Recovery, the STOP bit in the Reset Status Register is set to 1 and the IRQE bit in the IRQCTL Register is cleared disabling interrupts. Software can enable interrupts by setting the IRQE bit or by issuing the EI instruction. Interrupt capable peripherals running in Stop Mode can initiate a Stop-Mode Recovery only if enabled as an interrupt source. Table 11 lists the Stop-Mode Recovery sources and resulting actions. The text following provides more information about each of the Stop-Mode Recovery sources.

Table 11. Stop-Mode Recovery Sources and Resulting Action

Operating Mode	Stop-Mode Recovery Source	Action
Stop Mode	Watchdog Timer time-out when configured for Reset	Stop-Mode Recovery
	Watchdog Timer time-out when configured for interrupt	Stop-Mode Recovery followed by interrupt (if interrupts are re-enabled)
	Interrupt from timer enabled for Stop Mode operation	Stop-Mode Recovery followed by interrupt (if interrupts are re-enabled)
	Interrupt from comparator enabled for Stop Mode operation	Stop-Mode Recovery followed by interrupt (if interrupts are re-enabled)
	Interrupt from LVD enabled for Stop Mode operation	Stop-Mode Recovery followed by interrupt (if interrupts are re-enabled)
	Data transition on any GPIO Port pin enabled as a Stop-Mode Recovery source	Stop-Mode Recovery followed by interrupt (if the GPIO was previously enabled as an interrupt source and interrupts are re-enabled)
	Assertion of external $\overline{\text{RESET}}$ Pin	System Reset
	Debug Pin driven Low	System Reset

5.3.1. Stop-Mode Recovery Using Watchdog Timer Time-Out

If the WDT times out during Stop Mode, the device undergoes a Stop-Mode Recovery sequence. In the Reset Status Register, the WDT and STOP bits are set to 1. If the WDT is configured to generate an interrupt on time-out and if interrupts are re-enabled (IRQE in IRQCTL is set again), the eZ8 CPU services the WDT interrupt request. Reading the RSTSTAT Register resets the WDT bit and clears the WDT interrupt. As a result, the WDT interrupt vector is executed only if interrupts are re-enabled prior to reading the RSTSTAT Register. Alternatively, the RSTSTAT Register can be read prior to enabling interrupts followed by a call of the desired WDT interrupt code.

5.3.2. Stop-Mode Recovery Using Timer, Comparator, or LVD Interrupt

If a timer, comparator, or LVD enabled for Stop Mode operation asserts during Stop Mode, the device undergoes a Stop-Mode Recovery sequence. Comparator assertion is defined as a high output signal from the Comparator. In the Reset Status Register, the STOP bit is set to 1. If interrupts are re-enabled (IRQE in IRQCTL is set again), the eZ8 CPU services the corresponding interrupt request.

5.3.3. Stop-Mode Recovery Using GPIO Port Pin Transition

Many of the GPIO Port pins can be configured as a Stop-Mode Recovery input source. Which GPIO can be configured as a Stop-Mode Recovery input source is described in the General-Purpose Input/Output chapter on page 46. On any GPIO pin enabled as a Stop-Mode Recovery source, a change in the input pin value (from High to Low or from Low to High) initiates Stop-Mode Recovery. In the Reset Status Register, the STOP bit is set to 1.

If the GPIO is also configured as an interrupt source, an interrupt will occur once interrupts are re-enabled.

Caution: In Stop Mode, the GPIO Port Input Data registers (PxIN) are disabled. The Port Input Data registers record the Port transition only if the signal stays on the Port pin until the end of the Stop-Mode Recovery delay. As a result, short pulses on the Port pin can initiate Stop-Mode Recovery without being written to the Port Input Data Register or without initiating an interrupt (if enabled for that pin).

5.3.4. Stop-Mode Recovery Using External $\overline{\text{RESET}}$ Pin

When the F3224 Series MCU is in Stop Mode and the external $\overline{\text{RESET}}$ pin is driven Low, a System Reset occurs. Because of a glitch filter operating on the $\overline{\text{RESET}}$ pin, the Low pulse must be greater than the minimum width specified, or it is ignored. For details, see the Electrical Characteristics chapter on page 400.

5.4. Low-Voltage Detection

In addition to the VBO Reset described earlier, it is also possible to generate an interrupt when the supply voltage drops below a user-selected value. To learn more about the available Low-Voltage Detection (LVD) threshold levels, see the Flash Option Bits chapter on page 358.

When the supply voltage drops below the LVD threshold, the LVD bit of the RSTSTAT Register is set to 1. This bit remains 1 until the low-voltage condition elapses. Reading or writing this bit does not clear it. The LVD circuit can also generate an interrupt when enabled; see the Interrupt Controller chapter on page 89. The LVD is not latched, so enabling the interrupt is the only way to guarantee detection of a transient low-voltage event.

The LVD circuit is either enabled or disabled by the Power Control Register bit 4. To learn more, see the Power Control Register Definitions section on page 43.

5.5. Reset Register Definitions

The Reset Status (RSTSTAT) Register, shown in Table 12, is a read-only register that indicates the source of the most recent Reset event, Stop-Mode Recovery event and/or WDT time-out. Reading this register resets the upper 4 bits to 0.

Table 13 relates Reset and Stop-Mode recovery events to the Reset Status Register settings.

Table 12. Reset Status Register (RSTSTAT)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	POR/VBO	STOP	WDT	EXT	Reserved			LVD
Reset	See descriptions below			0	0	0	0	0
R/W	R	R	R	R	R	R	R	R
Address	FF0h							

Bit	Description
[7] POR/VBO	**Power-On initiated VBO Reset or general VBO Reset Indicator** If this bit is set to 1, a POR or VBO Reset event occurs. This bit is reset to 0, if a WDT time-out or Stop-Mode Recovery occurs. This bit is also reset to 0 when the register is read.
[6] STOP	**Stop-Mode Recovery Indicator** If this bit is set to 1, a Stop-Mode Recovery occurs. If the STOP and WDT bits are both set to 1, the Stop-Mode Recovery occurs because of a WDT time-out. If the STOP bit is 1 and the WDT bit is 0, the Stop-Mode Recovery is not caused by a WDT time-out. This bit is reset by Power-On Reset or WDT time-out that occurred while not in Stop Mode. Reading this register also resets this bit.
[5] WDT	**Watchdog Timer time-out Indicator** If this bit is set to 1, a WDT time-out occurs. A POR resets this bit. A Stop-Mode Recovery from a Stop-Mode Recovery source other than the WDT also resets this bit. Reading this register resets this bit. This read must occur to clear the WDT interrupt.
[4] EXT	External Reset Indicator If this bit is set to 1, a Reset initiated by the external $\overline{\text{RESET}}$ pin occurs. A POR or a Stop-Mode Recovery from a change in an input pin resets this bit. Reading this register resets this bit.
[3:1]	**Reserved** These bits are reserved and must be programmed to 000.
[0] LVD	**Low-Voltage Detection Indicator** If this bit is set to 1 the current state of the supply voltage is below the low-voltage detection threshold. This value is not latched but is a real-time indicator of the supply voltage level.

Table 13. Reset Status Per Event

Reset or Stop-Mode Recovery Event	POR	STOP	WDT	EXT
Power-On Reset or VBO Reset	1	0	0	0
Reset using $\overline{\text{RESET}}$ pin assertion	0	0	0	1
Reset using Watchdog Timer time-out	0	0	1	0
Reset using the On-Chip Debugger	1	0	0	0
Reset from Stop Mode using DBG Pin driven Low	1	0	0	0
Stop-Mode Recovery using GPIO pin transition	0	1	0	0
Stop-Mode Recovery using Watchdog Timer time-out	0	1	1	0
Stop-Mode Recovery using Timer, Comparator or Low Voltage Detection interrupt	0	1	0	0

Chapter 6. Low-Power Modes

The F3224 Series products have power-saving features. The highest level of power reduction is provided by the Stop Mode. The next lower level of power reduction is provided by the Halt Mode.

Further power savings can be implemented by disabling individual peripheral blocks while in Active Mode.

6.1. Stop Mode

Executing the eZ8 CPU's Stop instruction places the device into Stop Mode. In Stop Mode, the operating characteristics are:

- IRQE in the IRQCTL Register is cleared.
- The FLL is disabled (FLLEN is cleared) and the DCO is stopped; upon recovering from Stop Mode, the FLL remains disabled and the DCO is enabled, see the Clock System chapter on page 70 to learn more.
- If enabled and selected as the Peripheral Clock (PCKSEL in the Clock Control 1 Register), a Pclk source can be configured to operate in Stop Mode, as follows:
  - Internal Precision Oscillator (IPO): PCKSM = 1 in the Clock Control 1 Register and FRECOV = 1 in the Power Control 0 Register
  - Low Frequency Crystal Oscillator (LFXO): PCKSM = 1 in the Clock Control 1 Register
  - External clock drive: PCKSM = 1 in the Clock Control 1 Register
- System Clock is stopped.
- eZ8 CPU is stopped.
- Program counter (PC) stops incrementing.
- If enabled, the Watchdog Timer (WDT) logic continues operating.
- If enabled for operation in Stop Mode, the logic for Timer0-2 and Timer A continues to operate with the selected timer clock source.
- If enabled for operation in Stop Mode by the associated Flash option bits, the VBO protection circuit continues operating; the LVD circuit continues to operate if enabled by the Power Control Register 0.
- Operational Amplifiers and comparators continue to operate if both enabled by the Power Control Register 0 and FRECOV=1.
- All other on-chip peripherals are idle.

To minimize current in Stop Mode, all GPIO pins which are configured as digital inputs must be driven to one of the supply rails (V_{DD} or V_{SS}). The device is brought out of Stop Mode using Stop-Mode Recovery. To learn more about Stop-Mode Recovery, see the Reset, Stop-Mode Recovery and Low-Voltage Detection chapter on page 29.

6.2. Halt Mode

Executing the eZ8 CPU's Halt instruction places the device into Halt Mode. In Halt Mode, the operating characteristics are:

- Any enabled crystal oscillator continues to operate
- System clock is enabled and continues to operate
- eZ8 CPU is stopped
- Program counter (PC) stops incrementing
- If enabled, the WDT continues to operate
- All other on-chip peripherals continue to operate

The eZ8 CPU can be brought out of Halt Mode by any of the following operations:

- Interrupt
- Watchdog Timer time-out (Interrupt or Reset)
- Power-On Reset
- Voltage Brown-Out Reset
- External $\overline{\text{RESET}}$ pin assertion

To minimize current in Halt Mode, all GPIO pins which are configured as inputs must be driven to one of the supply rails (V_{DD}or V_{SS}).

6.3. Peripheral-Level Power Control

In addition to the Stop and Halt modes, it is possible to disable each peripheral on each of the F3224 Series devices. Disabling a given peripheral minimizes its power consumption.

6.4. Power Control Register Definitions

Each bit of the following registers disables a peripheral block, either by gating its system clock input or by removing power from the block.

With the exception of the LVD, the default state of all peripherals is OFF. To use a peripheral, set the peripheral's enable bit. If a peripheral is not offered on a particular product, setting or clearing the corresponding disable bit has no effect on product operation. Some enabled peripherals even run in Stop Mode. If the peripheral is not required in Stop Mode, disable it. Failure to perform this task results in Stop Mode currents greater than specified.

Note: These registers are only reset during a POR/VBO Reset; other System Reset events do not affect these registers.

6.4.1. Power Control Register 0

Table 14. Power Control Register 0 (PWRCTL0)

Bit	7	6	5	4	3	2	1	0
Field	OpAmpA1	OpAmpA0	Reserved	LVD	Reserved	FRECOV	COMP0	COMP1
Reset	0	0	0	1	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F80h							

Bit	**Description**
[7] OpAmpA1	**Op Amp A1 Enable** 0=Op Amp A1 is disabled. 1=Op Amp A1 is enabled (this applies even in Stop Mode if FRECOV=1).
[6] OpAmpA0	**Op Amp A0 Enable** 0=Op Amp A0 is disabled. 1=Op Amp A0 is enabled (this applies even in Stop Mode if FRECOV=1).
[5]	**Reserved** This bit is reserved and must be programmed to 0.
[4] LVD	**Low-Voltage Detection Enable** 0=LVD disabled. 1=LVD enabled (this applies even in Stop Mode).
[3]	**Reserved** This bit is reserved and must be programmed to 0.

Bit	Description (Continued)
[2] FRECOV	**Fast Recovery** 0=Fast Recovery disabled. 1=Fast Recovery enabled. Fast Recovery provides for the shortest Stop-Mode recovery latency at the expense of higher Stop Mode current consumption. See the Reset, Stop-Mode Recovery and Low-Voltage Detection chapter on page 29 to learn more. In addition, this bit must be set for certain peripherals to remain active during Stop Mode as described in this chapter.
[1] COMP0	**Comparator 0 Enable** 0=Comparator 0 is disabled. 1=Comparator 0 is enabled (this applies even in Stop Mode if FRECOV=1).
[0] COMP1	**Comparator 1 Enable** 0=Comparator 1 is disabled). 1=Comparator 1 is enabled (this applies even in Stop Mode if FRECOV=1).

6.4.2. Power Control Register 1

Table 15. Power Control Register 1 (PWRCTL1)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	Reserved				ADC		Reserved	
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F81h							

Bit	**Description**
[7:4]	**Reserved** These bits are reserved and must be programmed to 0.
[3:2] ADC	**ADC and ADC Internal Voltage Reference Buffer Continous Enable** 00: ADC is auto enabled when a conversion is triggered. The ADC wake up time, T_{WAKE_ADC}, is incurred prior to the conversion starting. 01: If selected as the ADC positive reference, the ADC internal voltage reference buffer is continuously enabled (while in Active or Halt modes). This selection is typically used when the ADC internal voltage reference buffer is connected to the V_{REF}+ pin (REFSEL = 11 in the ADCCTL2 Register).The ADC wake up time, T_{WAKE_ADC}, is incurred prior to the conversion starting. 10: The ADC is continuously enabled (while in Active or Halt modes). Additionally, the ADC internal voltage reference buffer is also continuously enabled (while in Active or Halt modes). The ADC wake up time, T_{WAKE_ADC}, is overridden to be zero. 11: Reserved. Defaults to ADC=10 behavior.
[1:0]	**Reserved** These bits are reserved and must be programmed to 00.

Chapter 7. General-Purpose Input/Output

The F3224 Series products support a maximum of 36 port pins (ports A–E) for general-purpose input/output (GPIO) operations. Each port contains control and data registers. The GPIO control registers determine data direction, open-drain, output drive current, programmable pull-ups, Stop-Mode Recovery functionality, and alternate pin functions. Each port pin is individually programmable.

7.1. GPIO Port Availability by Device

Table 16 lists the port pins available with each device and package type.

Table 16. Port Availability by Device and Package Type

Device	Pkg.	12-Bit ADC CH	I^2C	SPI	UART	Port A	Port B	Port C	Port D	Port E	Total I/O
Z8F3224QK, Z8F1624QK	32-pin QFN	12	1	1	1	[7:0]	[7:0]	[7:0]	[1:0]	-	26
Z8F3224QN, Z8F1624QN	44-pin QFN	15	1	1	1	[7:0]	[7:0]	[7:0]	[7:0]	[3:0]	36

7.2. Architecture

Figure 8 shows a simplified block diagram of a GPIO port pin and does not illustrate the ability to accommodate alternate functions and variable port current drive strength.

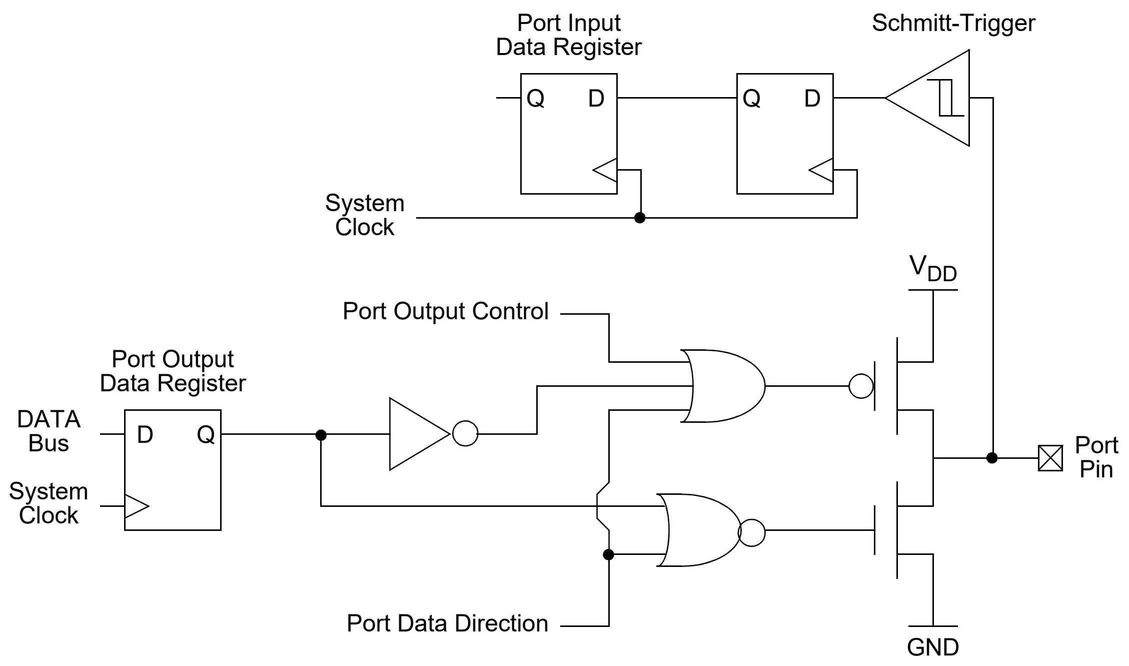


Figure 8. GPIO Port Pin Block Diagram

7.3. GPIO Alternate Functions

Many GPIO port pins are used for GPIO and to access the on-chip peripheral functions like the timers and serial-communication devices. The Port A–E Alternate Function subregisters configure these pins for either GPIO or alternate function operation. When a pin is configured for alternate function, control of port-pin direction (input/output) is passed from Port A–E Data Direction subregisters to the alternate functions assigned to this pin. When the alternate function is an analog function such as ANAx, the control of pull-up enable is also passed from the Port A–E Pull-Up Enable subregister to the alternate functions assigned to this pin. Tables 18 through 19, beginning on page 49, list the alternate functions possible with each port pin for every package. The alternate function associated at a pin is defined through alternate function sets subregisters AFS1 and AFS2. $\overline{\text{T0OUT}}$, $\overline{\text{T1OUT}}$, and $\overline{\text{T2OUT}}$ are output only when the corresponding Timer is in PWM Dual Output Mode.

The 32kHz crystal oscillator function is not controlled by the GPIO block. When the 32kHz secondary oscillator is enabled in the clock system block, the GPIO functionality of PA2 and PA3, is overridden. In such a case, those pins function as input and output for the crystal oscillator.

7.4. Shared Reset Pin

On all devices, the Port D0 pin shares a function with a bidirectional reset pin. Unlike all other I/O pins, this pin does not default to GPIO pin on power-up. This pin acts as a bidirectional, open-drain reset with pull-up until user software reconfigures it. The Port D0 pin is output-only when in GPIO Mode and does not function as a Stop-Mode Recovery source.

7.5. Low Frequency Crystal Oscillator (LFXO) Override

For systems using the LFXO, PA2 and PA3 are used to connect a watch crystal. When the LFXO is enabled, the GPIO settings are overridden and PA2 and PA3 is disabled; see the Clock Control 1 Register (CLKCTL1) on page 83.

7.6. External Clock Setup

For systems using an external CMOS-level drive, PA0 can be selected as the System Clock source by configuring PA0 for alternate function CLKIN and writing the Clock Control 0 Register (see page 81) to select the External Clock Input. PA2 can be selected as the clock source for Pclk by configuring PA2 for alternate function CLK2IN and writing the Clock Control 1 Register (see page 83) to select the External Clock2 Input.

7.7. Port Alternate Function Mapping

Alternate Function subregisters enable the alternate function selection on pins. Tables 18 through 19 indicate the port alternate function mapping.

Table 18. Port Alternate Function Mapping, 32-Pin Parts

Port	Pin	Mnemonic	Alternate Function Description	Alternate Function Set Subregisters
Port A[2]	PA0	T0IN/$\overline{\text{T0OUT}}$	Timer 0 Input/Timer 0 Output Complement	AFS1[0]: 0, AFS2[0]: 0
		CLKIN	External Clock Input	AFS1[0]: 1, AFS2[0]: 0
		Reserved		AFS1[0]: x, AFS2[0]: 1
	PA1	T0OUT	Timer 0 Output	AFS1[1]: 0, AFS2[1]: 0
		SCKOUT	System Clock Out	AFS1[1]: 1, AFS2[1]: 0
		Reserved		AFS1[1]: x, AFS2[1]: 1
	PA2	DE	UART Driver Enable	AFS1[2]: 0, AFS2[2]: 0
		$\overline{\text{SS}}$	SPI Slave Select	AFS1[2]: 1, AFS2[2]: 0
		CLK2IN	External Clock2 Input	AFS1[2]: 0, AFS2[2]: 1
		Reserved		AFS1[2]: x, AFS2[2]: 1
	PA3	$\overline{\text{CTS}}$	UART Clear to Send	AFS1[3]: 0, AFS2[3]: 0
		MISO	SPI Master In/Slave Out	AFS1[3]: 1, AFS2[3]: 0
		ESOUT0	Event System Output 0	AFS1[3]: 0, AFS2[3]: 1
		Reserved		AFS1[3]: 1, AFS2[3]: 1
	PA4	RXD	UART Receive Data	AFS1[4]: 0, AFS2[4]: 0
		MOSI	SPI Master Out/Slave In	AFS1[4]: 1, AFS2[4]: 0
		ESOUT1	Event System Output 1	AFS1[4]: 0, AFS2[4]: 1
		Reserved		AFS1[4]: 1, AFS2[4]: 1
	PA5	TXD	UART Transmit Data	AFS1[5]: 0, AFS2[5]: 0
		SCK	SPI Serial Clock	AFS1[5]: 1, AFS2[5]: 0
		ESOUT2	Event System Output 2	AFS1[5]: 0, AFS2[5]: 1
		Reserved		AFS1[5]: 1, AFS2[5]: 1
	PA6	T1IN/$\overline{\text{T1OUT}}$	Timer 1 Input/Timer 1 Output Complement	AFS1[6]: 0, AFS2[6]: 0
		SCL	I^2C Serial Clock	AFS1[6]: 1, AFS2[6]: 0
		ANA0	ADC Analog Input	AFS1[6]: 0, AFS2[6]: 1
		Reserved		AFS1[6]: 1, AFS2[6]: 1

Notes

1. Because there are at most two choices of alternate function for some pins of Ports B and D, the Alternate Function Set Subregister AFS2 is not implemented. Additionally, alternate function selection, as described in the Port A–E Alternate Function Subregisters (see page 61), must also be enabled.
2. The alternate function selection for Ports A and C, as described in the Port A–E Alternate Function Subregisters (see page 61), must also be enabled.

Table 18. Port Alternate Function Mapping, 32-Pin Parts (Continued)

Port	Pin	Mnemonic	Alternate Function Description	Alternate Function Set Subregisters
Port A (cont'd)	PA7	T1OUT	Timer 1 Output	AFS1[7]: 0, AFS2[7]: 0
		SDA	I^2C Serial Data	AFS1[7]: 1, AFS2[7]: 0
		ANA1/VBIAS	ADC Analog Input or Voltage Bias with low current drive capability	AFS1[7]: 0, AFS2[7]: 1
		Reserved		AFS1[7]: 1, AFS2[7]: 1
Port B[1]	PB0	Reserved		AFS1[0]: 0
		ANA2/C0IN0P/ AMPA0INP	ADC Analog Input//Comparator 0 Input 0 (P)/Op Amp A0 Input (P)	AFS1[0]: 1
	PB1	Reserved		AFS1[1]: 0
		ANA3/C0IN0N/ AMPA0INN	ADC Analog Input/Comparator 0 Input 0 (N)/Op Amp A0 Input (N)	AFS1[1]: 1
	PB2	C0OUT	Comparator 0 Output	AFS1[2]: 0
		ANA4/ AMPA0OUT	ADC Analog Input/Op Amp A0 Output	AFS1[2]: 1
	PB3	Reserved		AFS1[3]: 0
		V_{REF}–	Voltage Reference (M)	AFS1[3]: 1
	PB4	Reserved		AFS1[4]: 0
		V_{REF}+	Voltage Reference (P)	AFS1[4]: 1
	PB5	C1OUT	Comparator 1 Output	AFS1[5]: 0
		ANA5/ AMPA1OUT	ADC Analog Input/Op Amp A1 Output	AFS1[5]: 1
	PB6	ESOUT2	Event System Output 2	AFS1[6]: 0
		ANA6/C1IN0N/ AMPA1INN	ADC Analog Input/Comparator 1 Input 0 (N)/Op Amp A1 Input (N)	AFS1[6]: 1
	PB7	ESOUT3	Event System Output 3	AFS1[7]: 0
		ANA7/C1IN0P/ AMPA1INP	ADC Analog Input/Comparator 1 Input 0 (P)/Op Amp A1 Input (P)	AFS1[7]: 1

Notes
1. Because there are at most two choices of alternate function for some pins of Ports B and D, the Alternate Function Set Subregister AFS2 is not implemented. Additionally, alternate function selection, as described in the Port A–E Alternate Function Subregisters (see page 61), must also be enabled.
2. The alternate function selection for Ports A and C, as described in the Port A–E Alternate Function Subregisters (see page 61), must also be enabled.

Table 18. Port Alternate Function Mapping, 32-Pin Parts (Continued)

Port	Pin	Mnemonic	Alternate Function Description	Alternate Function Set Subregisters
Port C[2]	PC0	T0IN/$\overline{\text{T0OUT}}$	Timer 0 Input/Timer 0 Output Complement	AFS1[0]: 0, AFS2[0]: 0
		DE	UART Driver Enable	AFS1[0]: 1, AFS2[0]: 0
		MISO	SPI Master In/Slave Out	AFS1[0]: 0, AFS2[0]: 1
		SCL	I^{2}C Serial Clock	AFS1[0]: 1, AFS2[0]: 1
	PC1	T0OUT	Timer 0 Output	AFS1[1]: 0, AFS2[1]: 0
		$\overline{\text{CTS}}$	UART Clear to Send	AFS1[1]: 1, AFS2[1]: 0
		$\overline{\text{SS}}$	SPI Slave Select	AFS1[1]: 0, AFS2[1]: 1
		SDA	I^{2}C Serial Data	AFS1[1]: 1, AFS2[1]: 1
	PC2	T2IN/$\overline{\text{T2OUT}}$	Timer 2 Input/Timer 2 Output Complement	AFS1[2]: 0, AFS2[2]: 0
		MOSI	SPI Master Out/Slave In	AFS1[2]: 1, AFS2[2]: 0
		ESOUT0	Event System Output 0	AFS1[2]: 0, AFS2[2]: 1
		Reserved		AFS1[2]: 1, AFS2[2]: 1
	PC3	T2OUT	Timer 2 Output	AFS1[3]: 0, AFS2[3]: 0
		SCK	SPI Serial Clock	AFS1[3]: 1, AFS2[3]: 0
		ESOUT1	Event System Output 1	AFS1[3]: 0, AFS2[3]: 1
		Reserved		AFS1[3]: 1, AFS2[3]: 1
	PC4	Reserved		AFS1[4]: 0, AFS2[4]: 0
		ANA8/C0IN1P	ADC or Comparator 0 Input 1 (P)	AFS1[4]: 1, AFS2[4]: 0
		Reserved		AFS1[4]: x, AFS2[4]: 1
	PC5	Reserved		AFS1[5]: 0, AFS2[5]: 0
		ANA9/C0IN1N	ADC or Comparator 0 Input 1 (N)	AFS1[5]: 1, AFS2[5]: 0
		Reserved		AFS1[5]: x, AFS2[5]: 1
	PC6	Reserved		AFS1[6]: 0, AFS2[6]: 0
		ANA10/C1IN1P	ADC or Comparator 1 Input 1 (P)	AFS1[6]: 1, AFS2[6]: 0
		ESOUT2	Event System Output 2	AFS1[6]: 0, AFS2[6]: 1
		Reserved		AFS1[6]: 1, AFS2[6]: 1

Notes
1. Because there are at most two choices of alternate function for some pins of Ports B and D, the Alternate Function Set Subregister AFS2 is not implemented. Additionally, alternate function selection, as described in the Port A–E Alternate Function Subregisters (see page 61), must also be enabled.
2. The alternate function selection for Ports A and C, as described in the Port A–E Alternate Function Subregisters (see page 61), must also be enabled.

Table 18. Port Alternate Function Mapping, 32-Pin Parts (Continued)

Port	Pin	Mnemonic	Alternate Function Description	Alternate Function Set Subregisters
Port C[2] (cont'd.)	PC7	Reserved		AFS1[7]: 0, AFS2[7]: 0
		ANA11/C1IN1N	ADC or Comparator 1 Input 1 (N)	AFS1[7]: 1, AFS2[7]: 0
		ESOUT3	Event System Output 3	AFS1[7]: 0, AFS2[7]: 1
		Reserved		AFS1[7]: 1, AFS2[7]: 1
Port D[1]	PD0	$\overline{\text{RESET}}$	External Reset	AFS1[0]: 0
		Reserved		AFS1[0]: 1
	PD1	Reserved		AFS1[1]: 0
		ANA12	ADC Analog Input	AFS1[1]: 1

Notes
1. Because there are at most two choices of alternate function for some pins of Ports B and D, the Alternate Function Set Subregister AFS2 is not implemented. Additionally, alternate function selection, as described in the Port A–E Alternate Function Subregisters (see page 61), must also be enabled.
2. The alternate function selection for Ports A and C, as described in the Port A–E Alternate Function Subregisters (see page 61), must also be enabled.

Table 19. Port Alternate Function Mapping (44-Pin Parts)

Port	Pin	Mnemonic	Alternate Function Description	Alternate Function Set Subregisters
Port A[2]	PA0	T0IN/$\overline{\text{T0OUT}}$	Timer 0 Input/Timer 0 Output Complement	AFS1[0]: 0, AFS2[0]: 0
		CLKIN	External Clock Input	AFS1[0]: 1, AFS2[0]: 0
		Reserved		AFS1[0]: x, AFS2[0]: 1
	PA1	T0OUT	Timer 0 Output	AFS1[1]: 0, AFS2[1]: 0
		SCKOUT	System Clock Out	AFS1[1]: 1, AFS2[1]: 0
		Reserved		AFS1[1]: x, AFS2[1]: 1
	PA2	DE	UART Driver Enable	AFS1[2]: 0, AFS2[2]: 0
		$\overline{\text{SS}}$	SPI Slave Select	AFS1[2]: 1, AFS2[2]: 0
		CLK2IN	External Clock2 Input	AFS1[2]: 0, AFS2[2]: 1
		Reserved		AFS1[2]: x, AFS2[2]: 1
	PA3	$\overline{\text{CTS0}}$	UART 0 Clear to Send	AFS1[3]: 0, AFS2[3]: 0
		MISO	SPI Master In/Slave Out	AFS1[3]: 1, AFS2[3]: 0
		ESOUT0	Event System Output 0	AFS1[3]: 0, AFS2[3]: 1
		Reserved		AFS1[3]: 1, AFS2[3]: 1
	PA4	RXD/	UART Receive Data	AFS1[4]: 0, AFS2[4]: 0
		MOSI	SPI Master Out/Slave In	AFS1[4]: 1, AFS2[4]: 0
		ESOUT1	Event System Output 1	AFS1[4]: 0, AFS2[4]: 1
		Reserved		AFS1[4]: 1, AFS2[4]: 1
	PA5	TXD	UART Transmit Data	AFS1[5]: 0, AFS2[5]: 0
		SCK	SPI Serial Clock	AFS1[5]: 1, AFS2[5]: 0
		ESOUT2	Event System Output 2	AFS1[5]: 0, AFS2[5]: 1
		Reserved		AFS1[5]: 1, AFS2[5]: 1
	PA6	T1IN/$\overline{\text{T1OUT}}$	Timer 1 Input/Timer 1 Output Complement	AFS1[6]: 0, AFS2[6]: 0
		SCL	I^2C Serial Clock	AFS1[6]: 1, AFS2[6]: 0
		ANA0	ADC Analog Input	AFS1[6]: 0, AFS2[6]: 1
		Reserved		AFS1[6]: 1, AFS2[6]: 1

Notes
1. Because there are at most two choices of alternate function for some pins of Ports B, D and E, the Alternate Function Set register AFS2 is not implemented. Additionally, alternate function selection, as described in the Port A–E Alternate Function Subregisters (see page 61), must also be enabled.
2. The alternate function selection for Ports A and C, as described in the Port A–E Alternate Function Subregisters (see page 61), must also be enabled.

Table 19. Port Alternate Function Mapping (44-Pin Parts) (Continued)

Port	Pin	Mnemonic	Alternate Function Description	Alternate Function Set Subregisters
Port A (cont'd)	PA7	T1OUT	Timer 1 Output	AFS1[7]: 0, AFS2[7]: 0
		SDA	I^{2}C Serial Data	AFS1[7]: 1, AFS2[7]: 0
		ANA1/VBIAS	ADC Analog Input or Voltage Bias with low current drive capability	AFS1[7]: 0, AFS2[7]: 1
		Reserved		AFS1[7]: 1, AFS2[7]: 1
Port B[1]	PB0	Reserved		AFS1[0]: 0
		ANA2/C0IN0P/ AMPA0INP	ADC Analog Input//Comparator 0 Input 0 (P)/Op Amp A0 Input (P)	AFS1[0]: 1
	PB1	Reserved		AFS1[1]: 0
		ANA3/C0IN0N/ AMPA0INN	ADC Analog Input/Comparator 0 Input 0 (N)/Op Amp A0 Input (N)	AFS1[1]: 1
	PB2	C0OUT	Comparator 0 Output	AFS1[2]: 0
		ANA4/ AMPA0OUT	ADC Analog Input/Op Amp A0 Output	AFS1[2]: 1
	PB3	Reserved		AFS1[3]: 0
		V_{REF}–	Voltage Reference (M)	AFS1[3]: 1
	PB4	Reserved		AFS1[4]: 0
		V_{REF}+	Voltage Reference (P)	AFS1[4]: 1
	PB5	C1OUT	Comparator 1 Output	AFS1[5]: 0
		ANA5/ AMPA1OUT	ADC Analog Input/Op Amp A1 Output	AFS1[5]: 1
	PB6	ESOUT2	Event System Output 2	AFS1[6]: 0
		ANA6/C1IN0N/ AMPA1INN	ADC Analog Input/Comparator 1 Input 0 (N)/Op Amp A1 Input (N)	AFS1[6]: 1
	PB7	ESOUT3	Event System Output 3	AFS1[7]: 0
		ANA7/C1IN0P/ AMPA1INP	ADC Analog Input/Comparator 1 Input 0 (P)/Op Amp A1 Input (P)	AFS1[7]: 1

Notes
1. Because there are at most two choices of alternate function for some pins of Ports B, D and E, the Alternate Function Set register AFS2 is not implemented. Additionally, alternate function selection, as described in the Port A–E Alternate Function Subregisters (see page 61), must also be enabled.
2. The alternate function selection for Ports A and C, as described in the Port A–E Alternate Function Subregisters (see page 61), must also be enabled.

Table 19. Port Alternate Function Mapping (44-Pin Parts) (Continued)

Port	Pin	Mnemonic	Alternate Function Description	Alternate Function Set Subregisters
Port C[2]	PC0	T0IN/$\overline{\text{T0OUT}}$	Timer 0 Input/Timer 0 Output Complement	AFS1[0]: 0, AFS2[0]: 0
		DE	UART Driver Enable	AFS1[0]: 1, AFS2[0]: 0
		MISO	SPI Master In/Slave Out	AFS1[0]: 0, AFS2[0]: 1
		SCL	I^{2}C Serial Clock	AFS1[0]: 1, AFS2[0]: 1
	PC1	T0OUT	Timer 0 Output	AFS1[1]: 0, AFS2[1]: 0
		$\overline{\text{CTS}}$	UART Clear to Send	AFS1[1]: 1, AFS2[1]: 0
		$\overline{\text{SS}}$	SPI Slave Select	AFS1[1]: 0, AFS2[1]: 1
		SDA	I^{2}C Serial Data	AFS1[1]: 1, AFS2[1]: 1
	PC2	T2IN/$\overline{\text{T2OUT}}$	Timer 2 Input/Timer 2 Output Complement	AFS1[2]: 0, AFS2[2]: 0
		MOSI	SPI Master Out/Slave In	AFS1[2]: 1, AFS2[2]: 0
		ESOUT0	Event System Output 0	AFS1[2]: 0, AFS2[2]: 1
		Reserved		AFS1[2]: 1, AFS2[2]: 1
	PC3	T2OUT	Timer 2 Output	AFS1[3]: 0, AFS2[3]: 0
		SCK	SPI Serial Clock	AFS1[3]: 1, AFS2[3]: 0
		ESOUT1	Event System Output 1	AFS1[3]: 0, AFS2[3]: 1
		Reserved		AFS1[2]: 1, AFS2[2]: 1
	PC4	Reserved		AFS1[4]: 0, AFS2[4]: 0
		ANA8/C0IN1P	ADC or Comparator 0 Input 1 (P)	AFS1[4]: 1, AFS2[4]: 0
		Reserved		AFS1[4]: x, AFS2[4]: 1
	PC5	Reserved		AFS1[5]: 0, AFS2[5]: 0
		ANA9/C0IN1N	ADC or Comparator 0 Input 1 (N)	AFS1[5]: 1, AFS2[5]: 0
		Reserved		AFS1[5]: x, AFS2[5]: 1

Notes
1. Because there are at most two choices of alternate function for some pins of Ports B, D and E, the Alternate Function Set register AFS2 is not implemented. Additionally, alternate function selection, as described in the Port A–E Alternate Function Subregisters (see page 61), must also be enabled.
2. The alternate function selection for Ports A and C, as described in the Port A–E Alternate Function Subregisters (see page 61), must also be enabled.

Table 19. Port Alternate Function Mapping (44-Pin Parts) (Continued)

Port	Pin	Mnemonic	Alternate Function Description	Alternate Function Set Subregisters
Port C[2] (cont'd.)	PC6	Reserved		AFS1[6]: 0, AFS2[6]: 0
		ANA10/C1IN1P	ADC or Comparator 1 Input 1 (P)	AFS1[6]: 1, AFS2[6]: 0
		ESOUT2	Event System Output 2	AFS1[6]: 0, AFS2[6]: 1
		Reserved		AFS1[6]: 1, AFS2[6]: 1
	PC7	Reserved		AFS1[7]: 0, AFS2[7]: 0
		ANA11/C1IN1N	ADC or Comparator 1 Input 1 (N)	AFS1[7]: 1, AFS2[7]: 0
		ESOUT3	Event System Output 3	AFS1[7]: 0, AFS2[7]: 1
		Reserved		AFS1[7]: 1, AFS2[7]: 1
Port D[1]	PD0	$\overline{\text{RESET}}$	External Reset	AFS1[0]: 0
		Reserved		AFS1[0]: 1
	PD1	Reserved		AFS1[1]: 0
		ANA12	ADC Analog Input	AFS1[1]: 1
	PD2	Reserved		AFS1[2]: 0
		ANA13	ADC Analog Input	AFS1[2]: 1
	PD3	Reserved		AFS1[3]: 0
		ANA14	ADC Analog Input	AFS1[3]: 1
	PD4	T2IN/$\overline{\text{T2OUT}}$	Timer 2 Input/Timer 2 Output Complement	AFS1[4]: 0
		ESOUT1	Event System Output 1	AFS1[4]: 1
	PD5	T2OUT	Timer 2 Output	AFS1[5]: 0
		ESOUT2	Event System Output 2	AFS1[5]: 1
	PD6	T4CHA	Multi Channel Timer Input/Output A	AFS1[6]: 0
		Reserved		AFS1[6]: 1
	PD7	T4CHB	Multi Channel Timer Input/Output B	AFS1[7]: 0
		C0OUT	Comparator 0 Output	AFS1[7]: 1

Notes
1. Because there are at most two choices of alternate function for some pins of Ports B, D and E, the Alternate Function Set register AFS2 is not implemented. Additionally, alternate function selection, as described in the Port A–E Alternate Function Subregisters (see page 61), must also be enabled.
2. The alternate function selection for Ports A and C, as described in the Port A–E Alternate Function Subregisters (see page 61), must also be enabled.

Table 19. Port Alternate Function Mapping (44-Pin Parts) (Continued)

Port	Pin	Mnemonic	Alternate Function Description	Alternate Function Set Subregisters
Port E[1]	PE0	T4CHC	Multi Channel Timer Input/Output C	AFS1[0]: 0
		T0IN/$\overline{\text{T0OUT}}$	Timer 0 Input/Timer 0 Output Complement	AFS1[0]: 1
	PE1	T4CHD	Multi Channel Timer Input/Output D	AFS1[1]: 0
		T0OUT	Timer 0 Output	AFS1[1]: 1
	PE2	T4IN	Multi Channel Timer Input	AFS1[2]: 0
		Reserved		AFS1[2]: 1
	PE3	Reserved		AFS1[3]: 0
		ESOUT3	Event System Out 3	AFS1[3]: 1

Notes
1. Because there are at most two choices of alternate function for some pins of Ports B, D and E, the Alternate Function Set register AFS2 is not implemented. Additionally, alternate function selection, as described in the Port A–E Alternate Function Subregisters (see page 61), must also be enabled.
2. The alternate function selection for Ports A and C, as described in the Port A–E Alternate Function Subregisters (see page 61), must also be enabled.

Many of the GPIO port pins can be used as interrupt sources. Some port pins can be configured to generate an interrupt request on either the rising edge or falling edge of the pin-input signal. Other port-pin interrupt sources generate an interrupt when any edge occurs (both rising and falling). To learn more about interrupts using the GPIO pins, see the Interrupt Controller chapter on page 89.

7.8. GPIO Control Register Definitions

Four registers for each port provide access to GPIO control, input data, and output data. Table 20 lists these port registers. Use the Port A–E address and control registers together to provide access to their subregisters for port configuration and control.

Table 20. GPIO Port Registers and Subregisters

Port Register Mnemonic	**Port Register Name**
P*x*ADDR	Port A–E Address Register (Selects subregisters)
P*x*CTL	Port A–E Control Register (Provides access to subregisters)
P*x*IN	Port A–E Input Data Register
P*x*OUT	Port A–E Output Data Register
P*x*DD	Data Direction
P*x*AF	Alternate Function
P*x*OC	Output Control (Open-Drain)
P*x*HDE	High Drive Enable
P*x*SMRE	Stop-Mode Recovery Source Enable
P*x*PUE	Pull-up Enable
PxAFS1	Alternate Function Set 1
PxAFS2	Alternate Function Set 2

7.8.1. Port A–E Address Registers

The Port A–E address registers select the GPIO Port functionality accessible through the Port A–E Control registers. The Port A–E Address and Control registers combine to provide access to all GPIO Port controls, see Table 21.

Table 21. Port A–E GPIO Address Registers (P*x*ADDR)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	PADDR[7:0]							
Reset	00h							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	Port A @ FD0h, Port B @ FD4h, Port C @ FD8h, Port D @ FDCh, Port E @ FE0h							
Note: *x = A, B, C, D, or E.*								

Bit	**Description**
[7:0] P*x*ADDR	**Port A Address Registers** The port address selects one of the subregisters, which are accessible through the Port Control Register. 00h: No function; provides some protection against accidental port reconfiguration. 01h: Data Direction. 02h: Alternate Function. 03h: Output Control (Open-Drain). 04h: High Drive Enable. 05h: Stop-Mode Recovery Source Enable. 06h: Pull-up Enable. 07h: Alternate Function Set 1. 08h: Alternate Function Set 2. 09h–FFh: No function.

7.8.2. Port A–E Control Registers

The Port A–E Control registers, shown in Table 22, set the GPIO port operation. The value in the corresponding Port A–E Address Register determines which subregister is read from or written to by a Port A–E Control Register transaction.

Table 22. Port A–E Control Registers (P*x*CTL)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	PCTL							
Reset	00h							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	Port A @ FD1h, Port B @ FD5h, Port C @ FD9h, Port D @ FDDh, Port E @ FE1h							
Note: *x = A, B, C, D, or E.*								

Bit	**Description**
[7:0] P*x*CTL	**Port Control** The Port Control Register provides access to all subregisters that configure GPIO port operation.

7.8.3. Port A–E Data Direction Subregisters

The Port A–E Data Direction Subregister, shown in Table 23, is accessed through the Port A–E Control Register by writing `01h` to the Port A–E Address Register.

Table 23. Port A–E Data Direction Subregisters (P*x*DD)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	DD7	DD6	DD5	DD4	DD3	DD2	DD1	DD0
Reset	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	If 01h in Port A–E Address Register, accessible through the Port A–E Control Register							
Note: *x = A, B, C, D, or E.*								

Bit	**Description**
[7:0] P*x*DD	**Data Direction** These bits control the direction of the associated port pin. Port Alternate Function operation overrides the Data Direction Register setting. 0: Output. Data in the Port A–E Output Data Register is driven onto the port pin. 1: Input. The port pin is sampled and the value written into the Port A–E Input Data Register. The output driver is tristated.

7.8.4. Port A–E Alternate Function Subregisters

The Port A–E Alternate Function Subregister, shown in Table 24, is accessed through the Port A–E Control Register by writing `02h` to the Port A–E Address Register. The Port A–E Alternate Function subregisters enable the alternate function selection on pins. If disabled, pins functions as GPIO. If enabled, select one of the four alternate functions using Alternate Function set subregisters 1 and 2 as described in the Port A–E Alternate Function Set 1 Subregisters section on page 66 and the Alternate function selection on the port pins must also be enabled as described in the Port A–E Alternate Function Subregisters section on page 61. section on page 66**.** To determine the alternate function associated with each port pin, see the GPIO Alternate Functions section on page 47.

Caution: Do not enable alternate functions for GPIO port pins for which there is no associated alternate function. Failure to follow this guideline results in unpredictable operation.

Table 24. Port A–E Alternate Function Subregisters (P*x*AF)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	AF7	AF6	AF5	AF4	AF3	AF2	AF1	AF0
Reset	00h (Ports A–C); 01h (Port D); 00h (Port E)							
R/W	R/W							
Address	If 02h in Port A–E Address Register, accessible through the Port A–E Control Register							
Note: *x = A, B, C, D, or E*.								

Bit	**Description**
[7:0] P*x*AF	**Port Alternate Function Enable** 0: The port pin is in Normal Mode and the DDx bit in the Port A–E Data Direction Subregister determines the direction of the pin. 1: The alternate function selected through Alternate Function set subregisters are enabled. Port pin operation is controlled by the alternate function.

7.8.5. Port A–E Output Control Subregisters

The Port A–E Output Control Subregister, shown in Table 25, is accessed through the Port A–E Control Register by writing `03h` to the Port A–E Address Register. Setting the bits in the Port A–E Output Control subregisters to 1 configures the specified port pins for open-drain operation. These subregisters affect the pins directly and, as a result, alternate functions are also affected.

Table 25. Port A–E Output Control Subregisters (P*x*OC)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	POC7	POC6	POC5	POC4	POC3	POC2	POC1	POC0
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	If 03h in Port A–E Address Register, accessible through the Port A–E Control Register							
Note: *x = A, B, C, D, or E*.								

Bit	**Description**
[7:0] P*x*OC	**Port Output Control** These bits function independently of the alternate function bit and always disable the drains if set to 1. 0: The drains are enabled for any output mode (unless overridden by the alternate function). 1: The drain of the associated pin is disabled (open-drain mode).

7.8.6. Port A–E High Drive Enable Subregisters

The Port A–E High Drive Enable Subregister, shown in Table 26, is accessed through the Port A–E Control Register by writing `04h` to the Port A–E Address Register. Setting the bits in the Port A–E High Drive Enable subregisters to 1 configures the specified port pins for high-current output drive operation. The Port A–E High Drive Enable Subregister affects the pins directly and, as a result, alternate functions are also affected.

Table 26. Port A–E High Drive Enable Subregisters (P*x*HDE)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	PHDE7	PHDE6	PHDE5	PHDE4	PHDE3	PHDE2	PHDE1	PHDE0
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	If 04h in Port A–E Address Register, accessible through the Port A–E Control Register							
Note: *x = A, B, C, D, or E*.								

Bit	Description
[7:0] P*x*HDE	**Port High Drive Enabled** 0: The port pin is configured for standard output current drive. 1: The port pin is configured for high output current drive.

7.8.7. Port A–E Stop-Mode Recovery Source Enable Subregisters

The Port A–E Stop-Mode Recovery Source Enable Subregister, shown in Table 27, is accessed through Port A–E Control Register by writing `05h` to the Port A–E Address Register. Setting the bits in the Port A–E Stop-Mode Recovery Source Enable subregisters to 1 configures the specified Port pins as a Stop-Mode Recovery source. During Stop Mode, any logic transition on a Port pin enabled as a Stop-Mode Recovery source initiates Stop-Mode Recovery.

Table 27. Port A–E Stop-Mode Recovery Source Enable Subregisters (P*x*SMRE)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	PSMRE7	PSMRE6	PSMRE5	PSMRE4	PSMRE3	PSMRE2	PSMRE1	PSMRE0
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	If 05h in Port A–E Address Register, accessible through the Port A–E Control Register							
Note: *x = A, B, C, D, or E*.								

Bit	**Description**
[7:0] P*x*SMRE	**Port Stop-Mode Recovery Source Enabled** 0: The port pin is not configured as a Stop-Mode Recovery source. Transitions on this pin during Stop Mode do not initiate a Stop-Mode Recovery. 1: The port pin is configured as a Stop-Mode Recovery source. Any logic transition on this pin during Stop Mode initiates a Stop-Mode Recovery. **Note:** PortD0/$\overline{\text{RESET}}$ does not function as a Stop-Mode Recovery source.

7.8.8. Port A–E Pull-up Enable Subregisters

The Port A–E Pull-up Enable Subregister, shown in Table 28, is accessed through the Port A–E Control Register by writing `06h` to the Port A–E Address Register. Setting the bits in the Port A–E Pull-up Enable subregisters enables a weak internal resistive pull-up on the specified Port pins.

Table 28. Port A–E Pull-Up Enable Subregisters (P*x*PUE)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	PPUE7	PPUE6	PPUE5	PPUE4	PPUE3	PPUE2	PPUE1	PPUE0
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	If 06h in Port A–E Address Register, accessible through the Port A–E Control Register							
Note: *x = A, B, C, D, or E*.								

Bit	Description
[7:0] P*x*PUE	**Port Pull-up Enabled** 0: The weak pull-up on the port pin is disabled. 1: The weak pull-up on the port pin is enabled.

7.8.9. Port A–E Alternate Function Set 1 Subregisters

The Port A–E Alternate Function Set1 Subregister, shown in Table 29, is accessed through the Port A–E Control Register by writing `07h` to the Port A–E Address Register. The Alternate Function Set 1 subregisters select the alternate function available at a port pin. Alternate Functions selected by setting or clearing bits of this register are defined in the GPIO Alternate Functions section on page 47.

Table 29. Port A–E Alternate Function Set 1 Subregisters (P*x*AFS1)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	PAFS17	PAFS16	PAFS15	PAFS14	PAFS13	PAFS12	PAFS11	PAFS10
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	If 07h in Port A–E Address Register, accessible through the Port A–E Control Register							
Note: *x = A, B, C, D, or E.*								

Bit	**Description**
[7:0] P*x*AFS1	PAFS1[7:0] – Port Alternate Function Set 1 0: Port Alternate Function selected, as defined in Tables 17 through 19 in the GPIO Alternate Functions section on page 47. 1: Port Alternate Function selected, as defined in Tables 17 through 19 in the GPIO Alternate Functions section on page 47.

Note: Alternate function selection on the port pins must also be enabled as described in the Port A–E Alternate Function Subregisters section on page 61.

7.8.10. Port C Alternate Function Set 2 Subregister

The Port C Alternate Function Set 2 Subregister, shown in Table 30, is accessed through the Port C Control Register by writing `08h` to the Port C Address Register. The Alternate Function Set 2 Subregister selects the alternate function available at a port pin. Alternate Functions selected by setting or clearing bits of this register is defined in Tables 18 through 19 in the GPIO Alternate Functions section on page 47.

Table 30. Port C Alternate Function Set 2 Subregisters (PxAFS2)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	PAFS27	PAFS26	PAFS25	PAFS24	PAFS23	PAFS22	PAFS21	PAFS20
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	If 08h in Port A, C Address Register, accessible through the Port A, C Control Register							
Note: *x = A, or C.*								

Bit	**Description**
[7:0] P*x*AFS2	Port Alternate Function Set 2 0: Port Alternate Function selected, as defined in Tables 17 through 19 in the GPIO Alternate Functions section on page 47. 1: Port Alternate Function selected, as defined in Tables 17 through 19 in the GPIO Alternate Functions section on page 47.

Note: Alternate function selection on port pins must also be enabled as described in the Port A–E Alternate Function Subregisters section on page 61.

7.8.11. Port A–E Input Data Registers

Reading from the Port A–E Input Data registers, shown in Table 31, returns the sampled values from the corresponding port pins. The Port A–E Input Data registers are read-only. The value returned for any unused ports is 0. Unused ports include those missing on the packages other than the 80-pin package.

Table 31. Port A–E Input Data Registers (P*x*IN)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	PIN7	PIN6	PIN5	PIN4	PIN3	PIN2	PIN1	PIN0
Reset	X	X	X	X	X	X	X	X
R/W	R	R	R	R	R	R	R	R
Address	Port A @ FD2h, Port B @ FD6h, Port C @ FDAh, Port D @ FDEh, Port E @ FE2h							
Note: *x = A, B, C, D, or E*.								

Bit	Description
[7:0] P*x*IN	**Port Input Data** Sampled data from the corresponding port pin input. 0: Input data is logical 0 (Low). 1: Input data is logical 1 (High).

7.8.12. Port A–E Output Data Register

The Port A–E Output Data Register, shown in Table 32, controls the output data to the pins.

Table 32. Port A–E Output Data Register (P*x*OUT)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	POUT7	POUT6	POUT5	POUT4	POUT3	POUT2	POUT1	POUT0
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	Port A @ FD3h, Port B @ FD7h, Port C @ FDBh, Port D @ FDFh, Port E @ FE3h							
Note: *x = A, B, C, D, or E*.								

Bit	**Description**
[7:0] P*x*OUT	**Port Output Data** These bits contain the data to be driven to the port pins. The values are only driven if the corresponding pin is configured as an output and the pin is not configured for Alternate Function operation. 0: Drive a logical 0 (Low). 1: Drive a logical 1 (High). High value is not driven if the drain has been disabled by setting the corresponding Port Output Control Register bit to 1.

Chapter 8. Clock System

The F3224 Series devices use five possible clock sources, each user-selectable:

- On-chip Internal Precision Oscillator (IPO)
- On-chip Digitally Controlled Oscillator (DCO)
- On-chip Low Frequency Crystal Oscillator (LFXO) using off-chip crystal
- Two external clock drives
- On-chip Watchdog Timer Oscillator (WTO)

An internal clock multiplication system is available consisting of an on-chip Frequency Locked Loop (FLL) which locks the DCO to Peripheral Clock (Pclk, an internal clock).

Three internal clocks exist and are configured to the clock sources as follows:

- System Clock (Sysclk) is the clock input to the CPU as well as other system functions. The four possible clock sources for System Clock are:
  - Peripheral Clock (Pclk), a derived clock
  - DCO which can be locked to Pclk using the FLL
  - External clock drive
  - WTO

- Peripheral Clock (Pclk) is the 32.768 kHz clock input to the DCO and can be selected to clock certain peripherals. The three possible clock sources for Pclk are:
  - IPO
  - LFXO
  - External clock2 drive

- The WTO is the clock input to the Watchdog Timer (WDT) and can be selected to clock certain peripherals

In addition, F3224 Series devices contain:

- A clock failure detection and recovery circuitry allowing continued operation despite a failure of the System Clock source

- A divider circuit to reduce the frequency of the selected System Clock source. The divider selection can be altered dynamically to quickly increase or decrease System Clock frequency to manage performance and power consumption

8.1. Architecture

This chapter discusses the oscillator control system including clock sources such as oscillators, clock multiplication, selection of clock sources for internal clocks and oscillator failure detection. A diagram of the oscillator control system is shown in Figure 9.

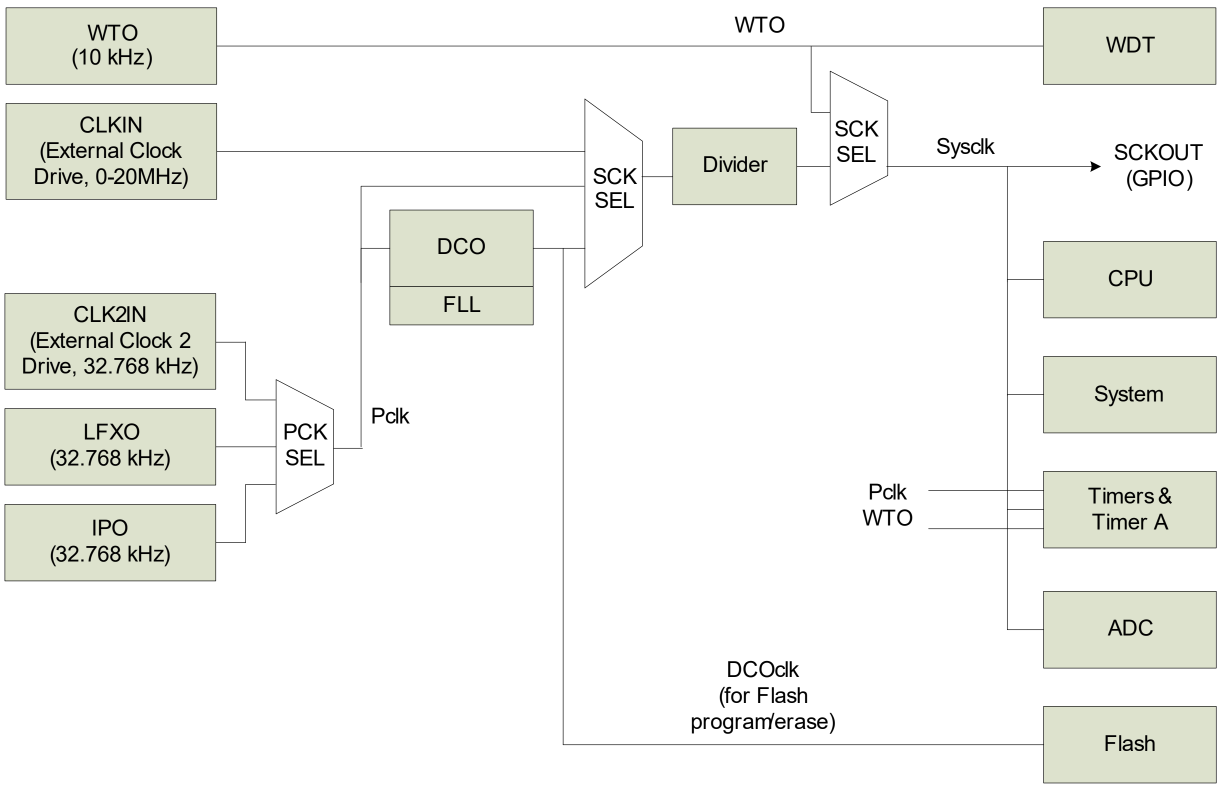


Figure 9. Clock System Block Diagram

8.2. Clock Selection

Clock sources can be selected for System Clock and Peripheral Clock

8.2.1. System Clock Selection

The Clock System selects from the available clock sources. Table 33 details each clock source and its usage.

Table 33. System Clock Configuration and Selection

Clock Source	Characteristics	Required Setup
Digitally Controlled Oscillator (DCO)	Running without FLL • Frequency range 1–20MHz • No external components required	• Unlock the Clock Control registers and configure the DCO • Switch System Clock sources as described in the System Clock Source Switching section on page 73
	Running with FLL • Performs clock multiplication • Frequency range 1–20 MHz • Accuracy 0.25% (vs. Pclk) • No external components required	• Unlock the Clock Control registers • Configure Peripheral Clock (Pclk) • Configure the FLL • Switch System Clock sources as described in the System Clock Source Switching section on page 73
Peripheral Clock	• See Table 34 on page 74	• Unlock the Clock Control registers and configure Peripheral Clock (Pclk) • Switch System Clock sources as described in the System Clock Source Switching section on page 73
External Clock Drive	• 0 to 20 MHz • Accuracy dependent on external clock source	• Write GPIO registers to configure PA0 pin for external clock input, CLKIN • Apply external clock signal to GPIO • Switch System Clock sources as described in the System Clock Source Switching section on page 73
Internal WDT Oscillator (WTO)	• 8.2 kHz nominal • ± 5% accuracy; no external components required • Low power consumption	• Switch System Clock sources as described in the System Clock Source Switching section on page 73

Caution: Unintentional accesses to the Clock System Control registers can result in the use of unintended clock sources and/or clock frequencies. To prevent this condition, the oscillator control block employs a register unlocking/locking scheme.

8.2.1.1. System Clock Source Switching

System Clock source switching provides glitch free operation in that changing the clock source or prescale setting will not cause clock glitches. In accomplishing glitch free clock switching operation, there is a delay from the writing of SCKSEL to the actual switching from the currently active clock source to the new, desired clock source. Reading SCKSEL returns the currently active clock source. Hardware will prevent disabling any clock source that is currently active as System Clock. To switch from one System Clock source to another, observe the following procedure:

1. Unlock the clock control registers
2. Write to the appropriate clock control registers to configure the desired clock source
3. Enable the desired clock source
4. Wait for the newly enabled clock source to stabilize by polling the appropriate ready bit
5. Write CLKCTL0 to select System Clock. The byte written to CLKCTL0 should have CSTAT set if it desired to leave the system clock registers unlocked
6. Poll SCKSEL to confirm that the selected System Clock is now currently active as System Clock; i.e., the value read from SCKSEL matches the value previously written to it)
7. If desired, disable any unnecessary clock sources. Hardware will prevent disabling any clock source that is currently active as System Clock
8. The clock control registers can be locked by clearing CSTAT

8.2.1.2. System Clock Divider

The clock source selected to be System Clock can be divided prior to driving System Clock. The clock divider provides glitch free operation in that changing the divider setting will not cause clock glitches. This allows software quickly to increase or decrease System Clock frequency to manage performance and power consumption.

8.2.2. Peripheral Clock (Pclk) Selection

Pclk operates at 32.768kHz (nominal). It is the DCO clock source and can be selected to be the clock for several perhipherals. Pclk can be configured operate during Stop Mode by setting PCKSM in the Clock Control Register and, if the IPO is selected as the source for

Pclk, by also setting FRECOV in the Power Control 0 Register. If switching the Pclk clock source, Zilog recommends first disabling any peripheral that is clocked by Pclk.

If the DCO is selected as System Clock with the FLL enabled, disable the FLL prior to selecting a new Pclk source. When Pclk is driven by the new source, the FLL can be enabled.

Table 34 summarizes peripheral clock sources and usage.

Table 34. Peripheral Clock Sources and Usage

Pclk Sources	Characteristics	Required Setup
Low Frequency Crystal Oscillator (LFXO)	• Optimized for use with a 32.768 kHz Watch Crystal • Very high accuracy • Dedicated XTAL pins • External components required	• Unlock the Clock Control registers • Enable the LFXO • Switch System Clock sources as described in the Pclk Source Switching section on page 74 • Select Pclk as the clock source for any desired peripheral(s) in the appropriate peripheral control register(s)
Internal Precision Oscillator (IPO)	• 32.768 kHz nominal • ± 2% accuracy with factory trim • No external components required	• Unlock the Clock Control registers • Enable the IPO • Switch System Clock sources as described in the Pclk Source Switching section on page 74 • Select Pclk as the clock source for any desired peripheral(s) in the appropriate peripheral control register(s)
External Clock2 Drive	• 32.768 kHz when used as Pclk source • Accuracy dependent on external clock source	• Write GPIO registers to configure PA2 pin for external clock2 input, CLK2IN • Apply external clock signal to GPIO • Unlock the Clock Control registers • Switch System Clock sources as described in the Pclk Source Switching section on page 74 • Select Pclk as the clock source for any desired peripheral(s) in the appropriate peripheral control register(s)

8.2.2.1. Pclk Source Switching

Pclk source switching provides glitch free operation in that changing the clock source or prescale setting will not cause clock glitches. In accomplishing glitch free clock switching operation, there is a delay from the writing of PCKSEL to the actual switching from the currently active clock source to the new, desired clock source. Reading PCKSEL returns the currently active clock source. Hardware will prevent disabling any clock source that is currently active as Pclk. To switch from one Pclk source to another, observe the following procedure:

1. Unlock the clock control registers.
2. Write to the appropriate clock control registers to configure the desired clock source.
3. Enable the desired clock source.
4. Wait for the newly enabled clock source to stabilize.
5. Write CLKCTL1 to select Pclk.
6. Poll PCKSEL to confirm that the selected Pclk is now currently active as Pclk; i.e., the value read from PCKSEL matches the value previously written to it).
7. If desired, disable any unnecessary clock sources. Hardware will prevent disabling any clock source that is currently active as Pclk.
8. The clock control registers can be locked by clearing CSTAT.

8.2.3. Clock System Control Register Unlocking/Locking

Before writing a clock system control register (CLKCTLx), the clock control registers must be unlocked by making two writes to the CLKCTL0 Register with the value `E7h` followed by the value `18h`. Successful unlocking sets CSTAT. When unlocked, one or more CLKCTLx registers can be written. The clock control registers can be again locked by clearing CSTAT in CLKCTL0. Any other sequence of clock system control register writes has no effect. The values written to unlock the register must be ordered correctly, but are not necessarily consecutive. It is possible to write to or read from other registers within the unlocking/locking operation. To remain unlocked when writing CLKCTL0, the byte written to CLKCTL0 must have CSTAT set.

Note: Before writing the unlock sequence, check that CSTAT is cleared. If the unlock sequence is written while CSTAT is set, the CLKCTL0 Register will be loaded with the unlock sequence values.

When selecting a new clock source, the System Clock failure detection circuitry must be disabled. If SCKFEN is not disabled prior to a clock switch-over, it is possible to generate an interrupt for a System Clock. The Failure detection circuitry can be enabled anytime after a successful write of SCKSEL in the CLKCTL0 Register.

By default, System Clock is configured as the Digitally Controlled Oscillator (DCO), locked to the Internal Precision Oscillator using the Frequency Locked Loop (FLL). If the user code changes to a different clock source, it may be appropriate to disable the IPO for power savings. Disabling the IPO does not occur automatically.

8.3. Clock Failure Detection and Recovery

Clock failure detection and recovery features are provided for the System Clock.

8.3.1. System Clock Failure

The F3224 Series devices can generate nonmaskable interrupt-like events when the System Clock source fails. To maintain system function in this situation, when SCKFEN=1, the clock failure recovery circuitry automatically forces the Watchdog Timer Oscillator (WTO) to drive System Clock. The WTO is enabled automatically to allow the recovery. Although this oscillator runs at a much slower speed than the original system clock, the CPU continues to operate allowing execution of a clock failure vector and software routines that either remedy the oscillator failure or issue a failure alert. This automatic switchover is not available if the WTO is the System Clock source

The System Clock source failure detection circuitry asserts if the System Clock frequency drops below 1 kHz. If an external signal is selected as the System Clock source, it is possible that a very slow but nonfailing clock can generate a failure condition. Under these conditions, do not enable the System Clock failure circuitry (SCKFEN should be cleared).

8.4. Low Frequency Crystal Oscillator (LFXO)

The products in the F3224 Series contain a low-frequency on-chip crystal oscillator (LFXO) for use with an external 32.768 kHz crystal. LFXO features include:

- Optimized for low current consumption
- Selectable as Pclk which can clock peripherals and be used to generate System Clock

Alternatively, the $X2_{IN}$ input pin can also accept a 32.768 kHz CMOS-level clock input signal. If an external clock generator is used, the $X2_{OUT}$ pin must be left unconnected.

8.4.1. LFXO Operation

LFXOEN in the CLKCTL1 Register controls whether the LFXO is enabled. During System Reset, LFXOEN is cleared, disabling the LFXO. When user code sets LFXOEN to enable the crystal oscillator, it should also check that the LFXO is stable, by reading LFXORDY, before using it as the FLL clock source or selecting the LFXO as System Clock. LFXORDY is cleared when the LFXO is disabled

Figure 10 shows a recommended configuration for connection with external load capacitors a fundamental-mode, parallel-resonant crystal operating. See the Electrical Characteristics chapter on page 400 for additional details regarding the characteristics of the LFXO. Printed circuit board layout should minimize crystal pin parasitic capacitance.

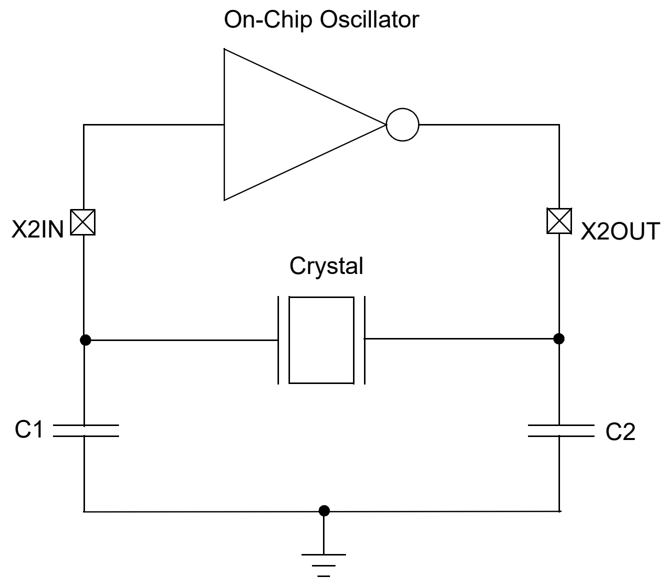


Figure 10. Recommended 32.768 kHz Crystal Oscillator Configuration

8.5. Internal Precision Oscillator (IPO)

The Internal Precision Oscillator (IPO) can be selected as Pclk and is designed for use without external components. IPO features include:

- On-chip RC oscillator that does not require external components
- Elimination of crystals in applications for which high timing accuracy is not required
- 32.768 kHz nominal frequency
- Accuracy: ± 2% over operational temperature and voltage range

8.5.1. Operation

IPOEN in the CLKCTL1 Register controls whether the IPO is enabled. During System Reset, IPOEN is set, enabling the IPO. If the IPO is disabled, user code can set IPOEN to enable the IPO, it should also check that the IPO is stable, by reading IPORDY, before selecting the IPO as Pclk.

The IPO is an RC relaxation oscillator that offers low sensitivity to power supply and temperature variation. At System Reset, the IPO is enabled and selected as Pclk which, in turn, is selected as input to the FLL. If the IPO is not required, it can be disabled by clearing IPOEN in CLKCTL1 to reduce system power consumption.

8.6. Watchdog Timer Oscillator (WTO)

The Watchdog Timer Oscillator (WTO) can be selected as System Clock, the clock source for several peripherals including the Watchdog Timer. The WTO is automatically enabled whenever it is needed.

8.7. Digitally Controlled Oscillator (DCO)

The DCO can be selected as System Clock and can be adjusted to oscillate over a wide frequency range. DCO features include:

- Can be locked to Pclk using the FLL or can free run
- When using the FLL, the DCO is locked to a multiple of Pclk determined by FLLNDIVL and FLLNDIVH
- When locked, the FLL can remain enabled so that the DCO control words continue to converge, tracking any changes in operating conditions, or the FLL can be disabled to free run with the current DCO control words

8.7.1. Operating Modes

The DCO supports two operating modes:

- Free running (FLLEN = 0 in CLKCTL5)
- Locked to Pclk using the FLL (FLLEN=1 in CLKCTL5)

8.7.2. DCO Operation

DCOEN in the CLKCTL5 Register controls whether the DCO is enabled. During System Reset, DCOEN is set, enabling the DCO. After setting DCOEN to enable the DCO, the DCO will oscillate at a frequency determined by the DCO control words as well as device characteristics and operating conditions. A particular set of DCO control words may not provide exactly the same oscillation frequency on all units, or at differing operating conditions for any particular unit.

To achieve a desired DCO operating frequency, the appropriate control word values for a particular unit at its current operating conditions can be determined by enabling the FLL to lock the DCO to Pclk. FLADONE will be set after the convergence of the DCO control words is completed. The FLL can remain enabled so that the DCO control words continue to converge to track any changes in operating conditions, or the FLL can be disabled to run with the current DCO control words.

To learn more about changing DCO frequency while the FLL is enabled, see the Frequency Locked Loop (FLL) section on page 79.

Use caution when free running the DCO, as changes to operating temperature and operating voltage can result in a DCO frequency that differs from the frequency at the time the DCO control word values were converged.

8.8. Frequency Locked Loop (FLL)

The FLL is used to lock the DCO to a frequency multiple of Pclk. FLL features include:

- Converges DCO control words to achieve frequency lock
- Employs both fast locking and linear locking algorithms

8.8.1. Operating Modes

The FLL supports two locking modes:

- A fast locking algorithm is initiated when FLLNDIVH is written. It is well-suited to locking without initial values for the DCO control words. The fast locking algorithm is also initiated automatically during a System Reset.
- A linear locking algorithm is initiated whenever FLLEN is set. If, for example, the FLL is disabled after locking, then re-enabled, linear locking algorithm is initiated.

8.8.2. FLL Operation

The FLL consists of a 10-bit feedback divider, a phase detector and an integrator. A block diagram of the FLL with connections to the DCO is shown in Figure 11.

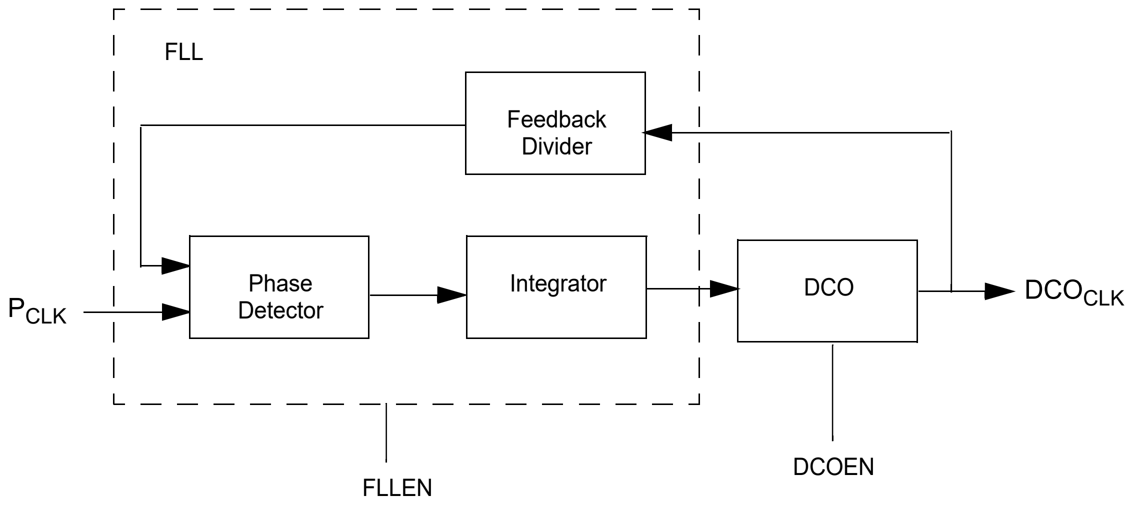


Figure 11. FLL Block Diagram

FLLEN in the CLKCTL5 Register controls whether the FLL is enabled. During System Reset, FLLEN is set, enabling the FLL to lock the DCO to the default frequency. While enabled, the FLL converges the DCO control words to lock the DCO to the multiple of Pclk determined by {FLLNDIVH, FLLNDIVL}. The FLL satisfies the following equation:

DCO_{CLK}=Pclk x {FLLNDIVH, FLLNDIVL}

A change of FLL frequency target can be initiated as follows:

a. Set FLLEN=1.
b. Write FLLNDIVL with the least significant byte of the desired frequency divisor.
c. Write FLLNDIVH with the most significant byte of the desired frequency divisor. Writing to FLLNDIVH will trigger the fast locking algorithm.

Two bits, FLLLL and FLADONE, are provided in the CLKCTL5 Register to describe the FLL status. The FLADONE bit is set when the fast locking algorithm has completed. If the FLL loses lock status, FLLLL is set, otherwise FLLLL is cleared. If the FLL is disabled, FLLLL is cleared. If lock is lost while the FLL is enabled, a system clock fail trap occurs if the FLLIRQE bit in the CLKCTL5 Register is set.

Note: The fast locking algorithm should not be interrupted by entering Stop Mode, clearing the FLLEN bit, or by changing the FLL divider value. After the fast locking algorithm has been initiated, always check to determine that it has completed (as evidenced by FLADONE = 1) before entering Stop Mode, clearing the FLLEN bit, or by changing the FLL divider value. The fast locking algorithm is always initiated automatically during System Reset.

While the FLL is enabled, it continues to converge the DCO control codes, as required, to maintain frequency lock. Although the DCO has fine resolution, the resolution is finite and the FLL operation can result in dithering between two values of the DCO control words. If such dither is undesirable, the FLL can be disabled for dither-free operation and enabled periodically to again converge the DCO control words and therefore account for environmental changes. Upon being enabled, the FLL will converge the DCO control words based on the divisor values in FLLNDIVL and FLLNDIVH, and achieve lock using the linear algorithm.

Immediately after Stop-Mode Recovery, the DCO is enabled and operates with the DCO control words existing upon entry into Stop Mode. In addition, the FLL is disabled. If no frequency change is desired, the FLL can be enabled and will lock using the linear algorithm.

8.9. Clock System Register Definitions

The Clock System registers enable and disable the various oscillator circuits, enable and disable the failure detection and recovery circuitry, and select clock sources for System Clock and Peripheral Clock.

8.9.1. Clock Control 0 Register

The Clock Control 0 (CLKCTL0) Register, shown in Table 35, selects System Clock and System Clock division, enables/disables System Clock failure detection, and provides clock system register locking status and control. SCKSEL is reset by both System Reset and Stop-Mode Recovery.

Before writing CLKCTL0, the clock control registers must be unlocked as described in the Clock System Control Register Unlocking/Locking section on page 75.

Table 35. Clock Control 0 Register (CLKCTL0)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	CSTAT	SCKFEN	SCKDIV			SCKSEL		
Reset	0	0	1	0	1	0	0	0
SMR*	none	none	none	none	none	0	0	0
R/W	R/W0	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F82h							
Note: *SMR = Effect of Stop-Mode Recovery.								

Bit	Description
[7] CSTAT	**Clock System Register Lock Status** 0: Clock system registers are locked. User software can clear CSTAT to lock the clock system registers. 1: Clock system registers are unlocked. Set by hardware when the {E7h, 18h} unlock sequence is written to CLKCTL0. To remain unlocked when writing to CLKCTL0, the byte written to CLKCTL0 must have CSTAT set.
[6] SCKFEN	**System Clock Failure Detection Enable** 0: Failure detection of the System Clock is disabled. 1: Failure detection of the System Clock is enabled.

Bit	Description (Continued)
[5:3] SCKDIV	**System Clock Division Ratio** Effective on all System Clock sources except the WTO. 000: 1. 001: 2. 010: 3. 011: 4. 100: 6. 101: 8. 110: 16. 111: 32. This setting should not be used if the System Clock frequency is ≤48kHz (e.g., Pclk is selected as System Clock) and SCKFEN is set.
[2:0] SCKSEL	**System Clock Source Select** There is a delay from the writing of SCKSEL to the actual switching from the currently active clock source to the new, desired clock source. Reading SCKSEL return the currently active clock source. 000: Digitally Controlled Oscillator (DCO). 001: Peripheral Clock (Pclk). 010: External clock drive, CLKIN, on PA0. 011: Reserved. 100: Watchdog Timer Oscillator (WTO). 101: Reserved. 110: Reserved. 111: Reserved.

8.9.2. Clock Control 1 Register

The Clock Control 1 (CLKCTL1) Register, shown in Table 36, enables/disables the IPO and LFXO, selects Pclk Stop Mode behavior, selects the Pclk source, and contains oscillator ready bits. Before writing CLKCTL1, the clock control registers must be unlocked as described in the Clock System Control Register Unlocking/Locking section on page 75.

Table 36. Clock Control 1 Register (CLKCTL1)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	IPORDY	LFXORDY	Reserved	PCKSEL		PCKSM	LFXOEN	IPOEN
Reset	0	0	0	0	0	0	0	1
R/W	R	R	R	R/W	R/W	R/W	R/W	R/W
Address	F83h							

Bit	Description
[7] IPORDY	**Internal Precision Oscillator (IPO) Ready Flag** 0: The IPO is not ready. 1: The IPO is ready.
[6] LFXORDY	**Low Frequency Crystal Oscillator (LFXO) Ready Flag** 0: The LFXO is not ready. 1: The LFXO is ready. The LFXO Ready Flag is cleared when the LFXO is disabled or when the LFXO sources System Clock (via Pclk) and the system clock failure trap is set.
[5]	**Reserved**
[4:3] PCKSEL	**Pclk Source Select** There is a delay from the writing of PCKSEL to the actual switching from the currently active clock source to the new, desired clock source. Reading PCKSEL returns the currently active clock source. 00: Internal Precision Oscillator (IPO). 01: Low Frequency Crystal Oscillator (LFXO). 10: External clock2 drive, CLK2IN on PA2. 11: Reserved.
[2] PCKSM	**Pclk Stop Mode Operation** 0: Enabled Pclk sources do not operate during Stop Mode. 1: Enabled Pclk source operate during Stop Mode. To enable Internal Precision Oscillator operation during Stop Mode, it is also necessary to set FRECOV in the Power Control 0 Register.
[1] LFXOEN	**Low Frequency Crystal Oscillator (LFXO) Enable** 0: LFXO is disabled. 1: LFXO is enabled.
[0] IPOEN	**Internal Precision Oscillator (IPO) Enable** 0: IPO is disabled. 1: IPO is enabled.

8.9.3. Clock Control 2 Register

The Clock Control 2 (CLKCTL2) Register, shown in Table 37, is reserved.

Table 37. Clock Control 2 Register (CLKCTL2)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	Reserved							
Reset	0	0	0	0	0	0	0	0
R/W	R	R	R	R	R	R	R	R
Address	F84h							

Bit	**Description**
[7:0]	**Reserved** These bits are reserved and must be programmed to 00h.

8.9.4. Clock Control 3 Register

The Clock Control 3 (CLKCTL3) Register, shown in Table 38, selects the FLL N-Divider High byte. Before writing CLKCTL3, the clock control registers must be unlocked as described in the Clock System Control Register Unlocking/Locking section on page 75.

Table 38. Clock Control 3 Register (CLKCTL3)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	FLLNDIVH							
Reset	0	1	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F85h							

Bit	**Description**
[7:0] FLLNDIVH	**Frequency Locked Loop (FLL) N-Divider High Byte** The most significant bits of the FLL N-divider control word, bits 9:2. Refer to the Clock Control 4 Register section on page 85 to learn more about FLL N-divider settings.

8.9.5. Clock Control 4 Register

The Clock Control 4 Register (CLKCTL4) selects the FLL N-Divider Low byte. Before writing CLKCTL4, the clock control registers must be unlocked as described in the Clock System Control Register Unlocking/Locking section on page 75.

Table 39. Clock Control 4 Register (CLKCTL4)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	Reserved						FLLNDIVL	
Reset	0	0	0	0	0	0	0	0
R/W	R	R	R	R	R	R	R/W	R/W
Address	F86h							

Bit	**Description**
[7:2]	**Reserved** These bits are reserved and must be programmed to 000000.
[1:0] FLLNDIVL	**Frequency Locked Loop (FLL) N-Divider Low Byte** 00–11: Least significant bits of the FLL N-divider control word, bits 1:0. To meet minimum and maximum FLL operating frequency requirements, {FLLNDIVH, FLLNDIVL} should not be less than 01Fh (1MHz) nor exceed 262h (20MHz). {FLLNDIVH, FLLNDIV} function as follows: 000h–01Eh=Reserved, 01Fh=Multiply Pclk by 31. 020h=Multiply Pclk by 32. 021h=Multiply Pclk by 33. • • • 260h: Multiply Pclk by 608. 261h: Multiply Pclk by 609. 262h: Multiply Pclk by 610. 263h–3FFh: Reserved.

8.9.6. Clock Control 5 Register

The Clock Control 5 Register (CLKCTL5), shown in Table 40, enables/disables various clock sources and selects controls the DCO and FLL. Entry into Stop Mode and Stop-Mode Recovery (SMR) affects the state of some bits in this register and leaves others unchanged. Before writing CLKCTL5, the clock control registers must be unlocked as described in the Clock System Control Register Unlocking/Locking section on page 75.

Table 40. Clock Control 5 Register (CLKCTL5)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	Reserved	FLLIRQE	FLLLL	Reserved	FLADONE	DCOEN	Reserved	FLLEN
Reset	0	0	0	0	0	1	0	1
STOP*	no change	no change	0	0	no change	0	no change	0
SMR*	no change	no change	0	0	no change	1	no change	0
R/W	R	R/W	R	R	R	R/W	R/W	R/W
Address	F87h							
Notes: *STOP = Effect of entering Stop Mode; SMR = Effect of Stop-Mode Recovery.								

Bit	**Description**
[7]	**Reserved** This bit is reserved and must be programmed to 0.
[6] FLLIRQE	**FLL Lost Lock Interrupt Request Enable** 0: The FLL lost lock condition (FLLLL) does not generate an interrupt request. 1: The FLL lost lock condition (FLLLL) generates an interrupt request using the System Clock failure interrupt.
[5] FLLLL	**FLL Lost Lock** 0: The FLL has not lost lock. 1: The FLL has lost lock.
[4]	**Reserved** This bit is reserved and must be programmed to 0.
[3] FLADONE	**FLL Fast Locking Algorithm Done** 0: The FLL fast locking algorithm is not done. If the fast locking algorithm was initiated, do not enter Stop Mode, clear FLLEN or change FLL divider value until this bit is set. 1: The FLL fast locking algorithm is done.
[2] DCOEN	**Digitally Controlled Oscillator (DCO) Enable** 0: DCO is disabled. 1: DCO is enabled.
[1]	**Reserved** This bit is reserved and must be programmed to 0.

Bit	Description (Continued)
[0] FLLEN	**Frequency Locked Loop (FLL) Enable** 0: FLL is disabled. The FLLLL status bit is cleared when the FLL is disabled. 1: FLL is enabled.

8.9.7. Clock Control 6 Register

The Clock Control 6 (CLKCTL6) Register, shown in Table 41, is reserved and should not be written.

Table 41. Clock Control 6 Register (CLKCTL6)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	Reserved							
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F88h							

Bit	Description
[7:0]	**Reserved**

8.9.8. Clock Control 7 Register

The Clock Control 7 (CLKCTL7) Register, shown in Table 42, is reserved and should not be written.

Table 42. Clock Control 7 Register (CLKCTL7)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	Reserved							
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F89h							

Bit	Description
[7:0]	**Reserved**

8.9.9. Clock Control 8 Register

The Clock Control 8 (CLKCTL8) Register, shown in Table 43, selects the High byte of the DCO converged control word. This register is read-only.

Table 43. Clock Control 8 Register (CLKCTL8)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	DCOCTLCH							
Reset	X	X	X	X	X	X	X	X
R/W	R	R	R	R	R	R	R	R
Address	F8Ah							

Bit	Description
[7:0] DCOCTLCH	**Digitally Controlled Oscillator (DCO) Converged Control Word High Byte** The most-significant bits of the DCO converged control word, bits 15:8.

8.9.10. Clock Control 9 Register

The Clock Control 9 (CLKCTL9) Register, shown in Table 44, selects the low byte of the DCO converged control word. This register is read-only.

Table 44. Clock Control 9 Register (CLKCTL9)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	DCOCTLCL							
Reset	X	X	X	X	X	X	X	X
R/W	R	R	R	R	R	R	R	R
Address	F8Bh							

Bit	Description
[7:0] DCOCTLCL	**Digitally Controlled Oscillator (DCO) converged control word Low Byte** The least-significant bits of the DCO converged control word, bits 7:0.

Chapter 9. Interrupt Controller

The interrupt controller on the F3224 Series products prioritizes the interrupt requests from the on-chip peripherals and the GPIO port pins. The interrupt controller includes the following features:

- Thirty interrupt sources using twenty-three unique interrupt vectors
  - 16 GPIO port pin interrupt sources (seven interrupt vectors are shared, see Table 45)
  - 14 on-chip peripheral interrupt sources (three interrupt vectors are shared, see Table 45)
- Flexible GPIO interrupts
  - Twelve selectable rising and falling edge GPIO interrupts
  - Four dual-edge interrupts
- Three levels of individually programmable interrupt priority
- WDT can be configured to generate an interrupt

Interrupt requests (IRQs) allow peripheral devices to suspend CPU operation in an orderly manner and force the CPU to start an interrupt service routine (ISR). Usually this interrupt service routine is involved with the exchange of data, status information, or control information between the CPU and the interrupting peripheral. When the service routine is completed, the CPU returns to the operation from which it was interrupted.

The eZ8 CPU supports both vectored and polled interrupt handling. For polled interrupts, the interrupt controller has no effect on operation. For more information about interrupt servicing by the eZ8 CPU, refer to the eZ8 CPU Core User Manual (UM0128), which is available free for download from the Zilog website.

9.1. Interrupt Vector Listing

Table 45 lists all of the interrupts available in order of priority. The interrupt vector is stored with the most-significant byte (MSB) at the even Program Memory address and the least-significant byte (LSB) at the following odd Program Memory address.

Note: Some port interrupts are not available on the Z8Fxx24 32-pin package and the Z8Fxx24 20-pin package. The Multi-Channel Timer interrupt source is unavailable on devices not containing that peripheral.

Table 45. Trap and Interrupt Vectors in Order of Priority

Priority*	Program Memory Vector Address	Interrupt or Trap Source
Highest	0002h	Reset (not an interrupt)
	0004h	Watchdog Timer; see the Watchdog Timer chapter on page 176
	0048h	System Clock Fail Trap (not an interrupt, the Clock System chapter on page 70)
	0006h	Illegal Instruction Trap (not an interrupt)
	0008h	Timer 2
	000Ah	Timer 1
	000Ch	Timer 0
	000Eh	UART receiver
	0010h	UART transmitter
	0012h	
	0014h	
	0016h	I^2C
	0018h	
	001Ah	
	001Ch	
	001Eh	
	0020h	ADC
	0022h	SPI
	0024h	
	0026h	
	0028h	Port A7, selectable rising or falling input edge or LVD. To learn more, see the Reset, Stop-Mode Recovery and Low-Voltage Detection chapter on page 29.
	002Ah	Port A6, selectable rising or falling input edge or Comparator 0 Output, selectable rising or falling edge

Table 45. Trap and Interrupt Vectors in Order of Priority (Continued)

Priority*	Program Memory Vector Address	Interrupt or Trap Source
	002Ch	Port A5, selectable rising or falling input edge or Comparator 1 Output, selectable rising or falling edge
	002Eh	Port A4 or Port D4, selectable rising or falling input edge
	0030h	Port A3 or Port D3, selectable rising or falling input edge
	0032h	Port A2 or Port D2, selectable rising or falling input edge
	0034h	Port A1 or Port D1, selectable rising or falling input edge
	0036h	Port A0, selectable rising or falling input edge
	0038h	Timer A
	003Ah	Multi-Channel Timer
	003Ch	
	003Eh	
	0040h	Port C3, both input edges
	0042h	Port C2, both input edges
	0044h	Port C1, both input edges
Lowest	0046h	Port C0, both input edges

Note: *The order of priority is only for identical interrupt level. The priority varies depending on different interrupt level setting. See the Interrupt Vectors and Priority section on page 93 to learn more.

9.2. Architecture

Figure 12 shows the interrupt controller block diagram.

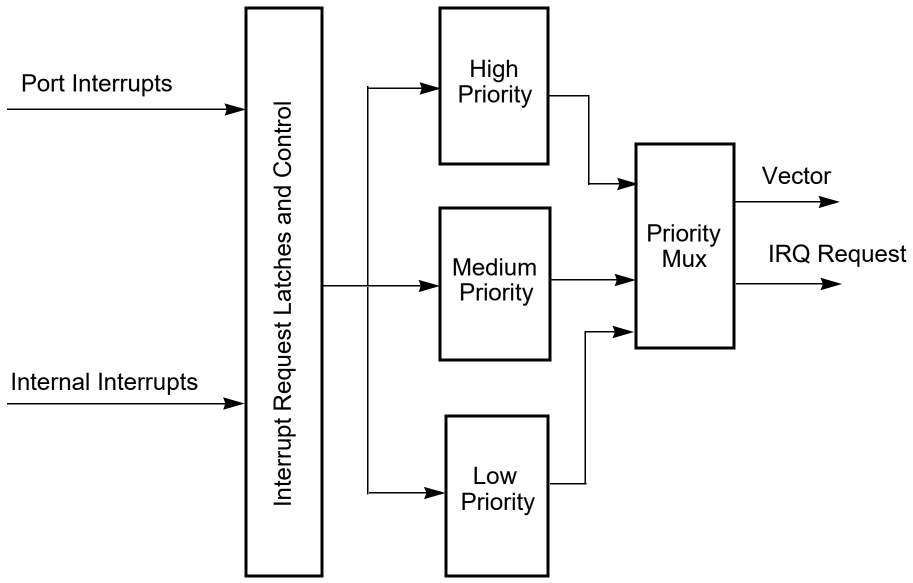


Figure 12. Interrupt Controller Block Diagram

9.3. Operation

The following section describes the master interrupt enable, interrupt vectors and priority, and interrupt assertion and deassertion.

9.3.1. Master Interrupt Enable

The master interrupt enable (IRQE) bit in the interrupt control register globally enables and disables interrupts.

Interrupts are globally enabled by any of the following actions:

- Execution of an Enable Interrupt (EI) instruction
- Execution of an Interrupt Return (IRET) instruction
- Writing 1 to the IRQE bit in the interrupt control register

Interrupts are globally disabled by any of the following actions:

- System Reset

- Stop-Mode Recovery
- Execution of a Disable Interrupt (DI) instruction
- Writing 0 to the IRQE bit in the interrupt control register
- eZ8 CPU acknowledgement of an interrupt service request from the interrupt controller
- Execution of a Trap instruction
- Illegal Instruction Trap
- Primary Oscillator Fail Trap

9.3.2. Interrupt Vectors and Priority

The interrupt controller supports three levels of interrupt priority. Level 3 is the highest priority, Level 2 is the second highest priority, and Level 1 is the lowest priority. If all of the interrupts are enabled with identical interrupt priority (for example, all as Level 2 interrupts), the interrupt priority is assigned from highest to lowest as specified in Table 45 on page 90. Level 3 interrupts are always assigned higher priority than Level 2 interrupts which, in turn, always are assigned higher priority than Level 1 interrupts. Within each interrupt priority level (Level 1, Level 2, or Level 3), priority is assigned as specified in Table 45 on page 90. Reset, Watchdog Timer interrupt (if enabled), Primary Oscillator Fail Trap, and Illegal Instruction Trap always have highest (Level 3) priority.

9.3.3. Interrupt Assertion

Interrupt sources assert their interrupt requests for only a single-system clock period (single pulse). When the interrupt request is acknowledged by the eZ8 CPU, the corresponding bit in the Interrupt Request Register is cleared until the next interrupt occurs. Writing 0 to the corresponding bit in the Interrupt Request Register likewise clears the interrupt request.

Caution: Zilog recommends not using a coding style that clears bits in the Interrupt Request registers. All incoming interrupts received between execution of the first LDX command and the final LDX command are lost. See Example 1, which follows.

Example 1. Poor coding style that can result in lost interrupt requests:

```
LDX r0, IRQ0
AND r0, MASK
LDX IRQ0, r0
```

To avoid missing interrupts, use the coding style in Example 2 to clear bits in the Interrupt Request 0 Register:

Example 2. Good coding style that avoids lost interrupt requests:

```
ANDX IRQ0, MASK
```

9.3.4. Software Interrupt Assertion

Program code can generate interrupts directly. Writing 1 to the correct bit in the Interrupt Request Register triggers an interrupt (assuming that interrupt is enabled). When the interrupt request is acknowledged by the eZ8 CPU, the bit in the Interrupt Request Register is automatically cleared to 0.

Caution: Zilog recommends not using a coding style to generate software interrupts by setting bits in the Interrupt Request registers. All incoming interrupts received between execution of the first LDX command and the final LDX command are lost. See Example 3, which follows.

Example 3. Poor coding style that can result in lost interrupt requests:

```
LDX r0, IRQ0
OR r0, MASK
LDX IRQ0, r0
```

To avoid missing interrupts, use the coding style in Example 4 to set bits in the Interrupt Request registers.

Example 4. Good coding style that avoids lost-interrupt requests:

```
ORX IRQ0, MASK
```

9.4. Interrupt Control Register Definitions

For all interrupts other than the Watchdog Timer interrupt and the Primary Oscillator Fail Trap, the Interrupt Control registers enable individual interrupts, set interrupt priorities, and indicate interrupt requests.

9.4.1. Interrupt Request 0 Register

The Interrupt Request 0 (IRQ0) Register, shown in Table 46, stores the interrupt requests for both vectored and polled interrupts. When a request is presented to the interrupt controller, the corresponding bit in the IRQ0 Register becomes 1. If interrupts are globally enabled (vectored interrupts), the interrupt controller passes an interrupt request to the eZ8 CPU. If interrupts are globally disabled (polled interrupts), the eZ8 CPU can read the Interrupt Request 0 Register to determine if any interrupt requests are pending.

Table 46. Interrupt Request 0 Register (IRQ0)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	T2I	T1I	T0I	URXI	UTXI	Reserved		I2CI
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R	R	R/W
Address	FC0h							

Bit	Description
[7] T2I	**Timer 2 Interrupt Request** 0: No interrupt request is pending for Timer 2. 1: An interrupt request from Timer 2 is awaiting service.
[6] T1I	**Timer 1 Interrupt Request** 0: No interrupt request is pending for Timer 1. 1: An interrupt request from Timer 1 is awaiting service.
[5] T0I	**Timer 0 Interrupt Request** 0: No interrupt request is pending for Timer 0. 1: An interrupt request from Timer 0 is awaiting service.
[4] URXI	**UART 0 Receiver Interrupt Request** 0: No interrupt request is pending for the UART receiver. 1: An interrupt request from the UART receiver is awaiting service.
[3] UTXI	**UART 0 Transmitter Interrupt Request** 0: No interrupt request is pending for the UART transmitter. 1: An interrupt request from the UART transmitter is awaiting service.
[2:1]	**Reserved** These bits are reserved and must be programmed to 00.
[0] I2CI	**I^{2}C Interrupt Request** 0: No interrupt request is pending for the I^{2}C. 1: An interrupt request from I^{2}C is awaiting service.

9.4.2. Interrupt Request 1 Register

The Interrupt Request 1 (IRQ1) Register, shown in Table 47, stores interrupt requests for both vectored and polled interrupts. When a request is presented to the interrupt controller, the corresponding bit in the IRQ1 Register becomes 1. If interrupts are globally enabled (vectored interrupts), the interrupt controller passes an interrupt request to the eZ8 CPU. If interrupts are globally disabled (polled interrupts), the eZ8 CPU can read the Interrupt Request 1 Register to determine if any interrupt requests are pending.

Table 47. Interrupt Request 1 Register (IRQ1)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	Reserved				ADCI	SPII	Reserved	
Reset	0	0	0	0	0	0	0	0
R/W	R	R	R	R	R/W	R/W	R	R
Address	FC3h							

Bit	Description
[7:4]	**Reserved** These bits are reserved and must be programmed to 0000.
[3] ADCI	**ADC Interrupt Request** 0: No interrupt request is pending for the ADC. 1: An interrupt request from the ADC is awaiting service.
[2] SPII	**SPI Interrupt Request** 0: No interrupt request is pending for the SPI. 1: An interrupt request from the SPI is awaiting service.
[1:0]	**Reserved** These bits are reserved and must be programmed to 00.

9.4.3. Interrupt Request 2 Register

The Interrupt Request 2 (IRQ2) Register, shown in Table 48, stores interrupt requests for both vectored and polled interrupts. When a request is presented to the interrupt controller, the corresponding bit in the IRQ2 Register becomes 1. If interrupts are globally enabled (vectored interrupts), the interrupt controller passes an interrupt request to the eZ8 CPU. If interrupts are globally disabled (polled interrupts), the eZ8 CPU can read the Interrupt Request 2 Register to determine if any interrupt requests are pending.

Table 48. Interrupt Request 2 Register (IRQ2)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	PA7VI	PA6CI	PA5CI	PAD4I	PAD3I	PAD2I	PAD1I	PA0I
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	FC6h							

Bit	Description
[7] PA7VI	**Port A7 or LVD Interrupt Request** 0: No interrupt request is pending for GPIO Port A7 or LVD. 1: An interrupt request from GPIO Port A7 or LVD.
[6] PA6CI	**Port A6 or Comparator 0 Interrupt Request** 0: No interrupt request is pending for GPIO Port A6 or Comparator 0. 1: An interrupt request from GPIO Port A6 or Comparator 0.
[5] PA5CI	**Port A5 or Comparator 1 Interrupt Request** 0: No interrupt request is pending for GPIO Port A5 or Comparator 1. 1: An interrupt request from GPIO Port A5 or Comparator 1.
[4:1] PAD*x*I	**Port A*x* or Port D*x* Interrupt Request** 0: No interrupt request is pending for GPIO Port A*x* or Port D*x*; *x* indicates the specific Port A or D number (4–1). 1: An interrupt request from GPIO Port A*x* or Port D*x* is awaiting service.
[0] PA0I	**Port A0 Interrupt Request** For a description of the interrupt source select feature, see the Shared Interrupt Select Register 0 section on page 107. 0: No interrupt request is pending for GPIO Port A0. 1: An interrupt request from GPIO Port A0 is awaiting service.

9.4.4. Interrupt Request 3 Register

The Interrupt Request 3 (IRQ3) Register, shown in Table 49, stores interrupt requests for both vectored and polled interrupts. When a request is presented to the interrupt controller, the corresponding bit in the IRQ3 Register becomes 1. If interrupts are globally enabled (vectored interrupts), the interrupt controller passes an interrupt request to the eZ8 CPU. If interrupts are globally disabled (polled interrupts), the eZ8 CPU can read the Interrupt Request 3 Register to determine if any interrupt requests are pending.

Table 49. Interrupt Request 3 Register (IRQ3)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	TAI	MCTI	Reserved		PC3I	PC2I	PC1I	PC0I
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	FC9h							

Bit	Description
[7] TAI	**Timer A Interrupt Request** 0: No interrupt request is pending for Timer A. 1: An interrupt request from Timer A is awaiting service.
[6] MCTI	**Multi-Channel timer Interrupt Request** 0: No interrupt request is pending for multi-channel timer. 1: An interrupt request from multi-channel timer is awaiting service.
[5:4]	**Reserved** These bits are reserved and must be programmed to 00.
[3:0] PC*x*I/	**Port C*x* Interrupt Request** 0: No interrupt request is pending for GPIO Port C*x*. 1: An interrupt request from GPIO Port C*x*is awaiting service; *x* indicates the specific GPIO Port C bit(3–0).

9.4.5. IRQ0 Enable High and Low Bit Registers

Table 50 presents the priority control for IRQ0. The IRQ0 Enable High and Low Bit registers, shown in Tables 51 and 52, form a priority-encoded enabling for interrupts in the Interrupt Request 0 Register. Priority is generated by setting bits in each register.

Table 50. IRQ0 Enable and Priority Encoding

IRQ0ENH[*x*]	IRQ0ENL[*x*]	Priority	Description
0	0	Disabled	Disabled
0	1	Level 1	Low

Table 50. IRQ0 Enable and Priority Encoding

IRQ0ENH[*x*]	IRQ0ENL[*x*]	Priority	Description
1	0	Level 2	Nominal
1	1	Level 3	High
Note: *x* indicates register bits from 0–7.			

Table 51. IRQ0 Enable High Bit Register (IRQ0ENH)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	T2ENH	T1ENH	T0ENH	URENH	UTENH	Reserved		I2CENH
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R	R	R/W
Address	FC1h							

Bit	Description
[7] T2ENH	**Timer 2 Interrupt Request Enable High Bit**
[6] T1ENH	**Timer 1 Interrupt Request Enable High Bit**
[5] T0ENH	**Timer 0 Interrupt Request Enable High Bit**
[4] URENH	**UART Receive Interrupt Request Enable High Bit**
[3] UTENH	**UART Transmit Interrupt Request Enable High Bit**
[2:1]	**Reserved** These bits are reserved and must be programmed to 00.
[0] I2CENH	**I^{2}C Interrupt Request Enable High Bit**

Table 52. IRQ0 Enable Low Bit Register (IRQ0ENL)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	T2ENL	T1ENL	T0ENL	URENL	UTENL	Reserved		I2CENL
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R	R	R/W
Address	FC2h							

Bit	Description
[7] T2ENL	**Timer 2 Interrupt Request Enable Low Bit**
[6] T1ENL	**Timer 1 Interrupt Request Enable Low Bit**
[5] T0ENL	**Timer 0 Interrupt Request Enable Low Bit**
[4] URENL	**UART Receive Interrupt Request Enable Low Bit**
[3] UTENL	**UART Transmit Interrupt Request Enable Low Bit**
[2:1]	**Reserved** These bits are reserved and must be programmed to 00.
[0] I2CENL	**I^{2}C Interrupt Request Enable Low Bit**

9.4.6. IRQ1 Enable High and Low Bit Registers

Table 53 lists the priority control for IRQ1. The IRQ1 Enable High and Low Bit registers, shown in Tables 54 and 55, form a priority-encoded enabling for interrupts in the Interrupt Request 1 Register. Priority is generated by setting bits in each register.

Table 53. IRQ1 Enable and Priority Encoding

IRQ1ENH[*x*]	IRQ1ENL[*x*]	Priority	Description
0	0	Disabled	Disabled
0	1	Level 1	Low
1	0	Level 2	Nominal
1	1	Level 3	High
Note: *x* indicates register bits from 0–7.			

Table 54. IRQ1 Enable High Bit Register (IRQ1ENH)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	Reserved				ADCENH	SPIENH	Reserved	
Reset	0	0	0	0	0	0	0	0
R/W	R	R	R	R	R/W	R/W	R	R
Address	FC4h							

Bit	Description
[7:4]	**Reserved** These bits are reserved and must be programmed to 0000.
[3] ADCENH	**ADC Interrupt Request Enable High Bit**
[2] SPIENH	**SPI Interrupt Request Enable High Bit**
[1:0]	**Reserved** These bits are reserved and must be programmed to 00.

Table 55. IRQ1 Enable Low Bit Register (IRQ1ENL)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	Reserved				ADCENL	SPIENL	Reserved	
Reset	0	0	0	0	0	0	0	0
R/W	R	R	R	R	R/W	R/W	R	R
Address	FC5h							

Bit	**Description**
[7:4]	**Reserved** These bits are reserved and must be programmed to 0000.
[3] ADCENL	**ADC Interrupt Request Enable Low Bit**
[2] SPIENL	**SPI Interrupt Request Enable Low Bit**
[1:0]	**Reserved** These bits are reserved and must be programmed to 00.

9.4.7. IRQ2 Enable High and Low Bit Registers

Table 56 lists the priority control for IRQ2. The IRQ2 Enable High and Low Bit registers, shown in Tables 57 and 58, form a priority-encoded enabling for interrupts in the Interrupt Request 2 Register. Priority is generated by setting bits in each register.

Table 56. IRQ2 Enable and Priority Encoding

IRQ2ENH[*x*]	**IRQ2ENL[*x*]**	**Priority**	**Description**
0	0	Disabled	Disabled
0	1	Level 1	Low
1	0	Level 2	Nominal
1	1	Level 3	High
Note: *x* indicates the register bits from 0–7.			

Table 57. IRQ2 Enable High Bit Register (IRQ2ENH)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	PA7VENH	PA6C0ENH	PA5C1ENH	PAD4ENH	PAD3ENH	PAD2ENH	PAD1ENH	PA0ENH
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	FC7h							

Bit	Description
[7] PA7VENH	**Port A7 or LVD Interrupt Request Enable High Bit**
[6] PA6C0ENH	**Port A6 or Comparator 0 Interrupt Request Enable High Bit**
[5] PA5C1ENH	**Port A5 or Comparator 1 Interrupt Request Enable High Bit**
[4:1] PAD*x*ENH	**Port A*x* or Port D*x* Interrupt Request Enable High Bit** *x* indicates the specific PAD bit (4–1). See Table 63 on page 107 for a selection of either Port A or Port D as the interrupt source.
[0] PA0ENH	**Port A0 Interrupt Request Enable High Bit** See Table 63 on page 107for a selection of either Port A or Port D as the interrupt source.

Table 58. IRQ2 Enable Low Bit Register (IRQ2ENL)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	PA7VENL	PA6C0ENL	PA5C1ENL	PAD4ENL	PAD3ENL	PAD2ENL	PAD1ENL	PA0ENL
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	FC8h							

Bit	Description
[7] PA7VENL	**Port A7 or LVD Interrupt Request Enable Low Bit**
[6] PA6C0ENL	**Port A6 or Comparator 0 Interrupt Request Enable Low Bit**
[5] PA5C1ENL	**Port A5 or Comparator 1 Interrupt Request Enable Low Bit**

Bit	Description (Continued)
[4:1] PAD*x*ENL	**Port A*x* or Port D*x* Interrupt Request Enable Low Bit** x indicates the specific PAD bit (4–1).
[0] PA0ENL	**Port A0 Interrupt Request Enable Low Bit**

9.4.8. IRQ3 Enable High and Low Bit Registers

Table 59 describes the priority control for IRQ3. The IRQ3 Enable High and Low Bit registers, shown in Tables 60 and 61, form a priority-encoded enabling for interrupts in the Interrupt Request 3 Register. Priority is generated by setting bits in each register.

Table 59. IRQ3 Enable and Priority Encoding

IRQ3ENH[*x*]	IRQ3ENL[*x*]	Priority	Description
0	0	Disabled	Disabled
0	1	Level 1	Low
1	0	Level 2	Nominal
1	1	Level 3	High
Note: *x* indicates register bits from 0–7.			

Table 60. IRQ3 Enable High Bit Register (IRQ3ENH)

Bit	7	6	5	4	3	2	1	0
Field	TAENH	MCTENH	Reserved		C3ENH	C2ENH	C1ENH	C0ENH
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R	R	R/W	R/W	R/W	R/W
Address	FCAh							

Bit	Description
[7] TAENH	**Timer A Interrupt Request Enable High Bit**
[6] MCTENH	**Multi-Channel Timer Interrupt Request Enable High Bit**
[5:4]	**Reserved** These bits are reserved and must be programmed to 00.
[3:0] C*x*ENH	**Port CxInterrupt Request Enable High Bit** *x* indicates the specific Port C bit (3–0)

Table 61. IRQ3 Enable Low Bit Register (IRQ3ENL)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	TAENL	MCTENL	Reserved		C3ENL	C2ENL	C1ENL	C0ENL
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R	R	R/W	R/W	R/W	R/W
Address	FCBh							

Bit	Description
[7] TAENL	**Timer A Interrupt Request Enable Low Bit**
[6] MCTENL	**Multi-Channel Timer Interrupt Request Enable Low Bit**
[5:4]	**Reserved** These bits are reserved and must be programmed to 00.
[3:0] C*x*ENL	**Port C*x* Interrupt Request Enable Low Bit** *x* indicates the specific Port C bit (3–0).

9.4.9. Interrupt Edge Select Register

The Interrupt Edge Select (IRQES) Register, shown in Table 62, determines whether an interrupt is generated for the rising edge or falling edge on the selected GPIO Port A or Port D input pin.

Table 62. Interrupt Edge Select Register (IRQES)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	IES7	IES6	IES5	IES4	IES3	IES2	IES1	IES0
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	FCCh							

Bit	**Description**
[7:0] IES*x*	**Interrupt Edge Select *x*** 0: An interrupt request is generated on the falling edge of the PA*x* input or PD*x* input or Comparator output. 1: An interrupt request is generated on the rising edge of the PA*x* input or PD*x* input or Comparator output.

Note: *x* indicates the specific GPIO port bit number (7–0).

9.4.10. Shared Interrupt Select Register 0

The Shared Interrupt Select 0 (IRQSS0) Register, shown in Table 63, determines the source of the PAD*x*S interrupts. The Shared Interrupt Select Register selects between Port A and alternate sources for the individual interrupts.

Table 63. Shared Interrupt Select Register 0 (IRQSS0)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	PA7VS	PA6CS	PA5CS	PAD4S	PAD3S	PAD2S	PAD1S	Reserved
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	FCDh							

Bit	Description
[7] PA7VS	**PA7/LVD Selection** 0: PA7 is used for the interrupt for PA7VS interrupt request. 1: The LVD is used for the interrupt for PA7VS interrupt request.
[6] PA6CS	**PA6/Comparator 0 Selection** 0: PA6 is used for the interrupt for PA6CS interrupt request. 1: The Comparator 0 is used for the interrupt for PA6CS interrupt request.
[5] PA5CS	**PA5/Comparator 1 Selection** 0: PA5 is used for the interrupt for PA5CS interrupt request. 1: The Comparator 1 is used for the interrupt for PA5CS interrupt request.
[4:1] PAD*x*S	**PAx/PDx Selection** *x* indicates the specific Port A bit number (4–1). 0: PAx is used for the interrupt for PAx/PDx interrupt request. 1: PDx is used for the interrupt for PAx/PDx interrupt request.
[0]	**Reserved** This bit is reserved and must be programmed to 0.

9.4.11. Interrupt Control Register

The Interrupt Control (IRQCTL) Register, shown in Table 64, contains the master enable bit for all interrupts.

Table 64. Interrupt Control Register (IRQCTL)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	IRQE	Reserved						
Reset	0	0	0	0	0	0	0	0
SMR*	0	NC	NC	NC	NC	NC	NC	NC
R/W	R/W	R	R	R	R	R	R	R
Address	FCFh							
Note: *SMR = Effect of Stop-Mode Recovery; NC = No Change.								

Bit	**Description**
[7] IRQE	**Interrupt Request Enable** This bit is set to 1 by executing an Enable Interrupts (EI) or Interrupt Return (IRET) instruction, or by a direct register write of a 1 to this bit. It is reset to 0 by System Reset, Stop-Mode Recovery, executing a Disable Interrupts (DI) instruction, direct register write of a 0 to this bit, eZ8 CPU acknowledgement of an interrupt request, or by a trap. 0: Interrupts are disabled. 1: Interrupts are enabled.
[6:0]	**Reserved** These bits are reserved and must be programmed to 0000000.

Chapter 10. Timers

The F3224 Series products contain three 16-bit reloadable timers that can be used for timing, event counting, or generation of pulse-width modulated signals. The timers' features include:

- 16-bit reload counter
- One-shot timer
- Programmable prescaler with prescale values ranging from 1 to 128
- PWM output generation
- Capture and compare capability
- Two independent capture/compare channels which reference the common timer
- Event System and external input pin for timer input, clock gating, or capture signal. External input pin signal frequency is limited to a maximum of one-fourth the timer clock frequency
- Timer output to Event System and external pin
- Timer interrupt
- Noise Filter on Timer Input 0 signal
- Operation in any mode with Pclk or WTO; operation in Active Mode and Halt Mode with Sysclk, Pclk, or WTO

In addition to the timers described in this chapter, the Baud Rate Generator (BRG) of unused serial port peripherals can also be used to provide basic timing functionality. For more information about using the Baud Rate Generators as additional timers, see the UART-LDD chapter on page 180, the Enhanced Serial Peripheral Interface chapter on page 231 and the I2C Master/Slave Controller chapter on page 256.

10.1. Timer Architecture

Figure 13 shows the architecture of the Timer.

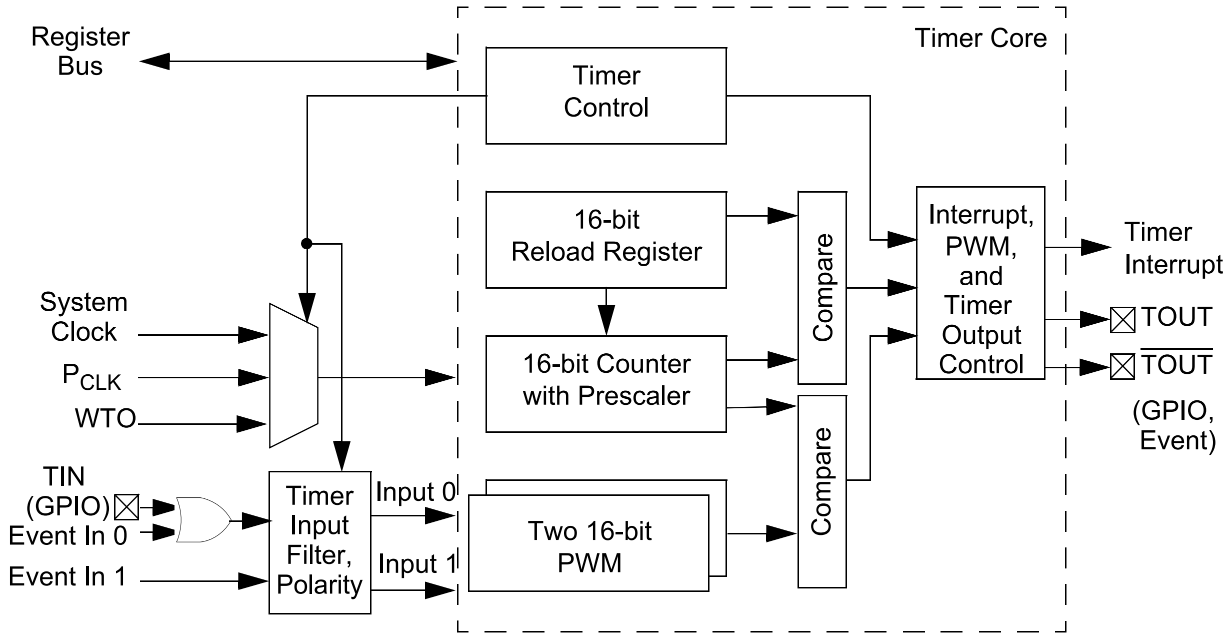


Figure 13. Timer Block Diagram

10.1. Timer Operation

The Timer is a 16-bit up-counter. Minimum time-out delay is set by loading the value `0001h` into the Timer Reload High and Low Byte registers and setting the prescale value to 1. Maximum time-out delay is set by loading the value `0000h` into the Timer Reload High and Low Byte registers and setting the prescale value to 128. If the Timer reaches `FFFFh`, the Timer rolls over to `0000h` and continues counting.

10.1.1. Timer Clock Source

The timer clock source is selected in TCLKS and can come from either the peripheral clock (Pclk), the Watch-dog Timer Oscillator (WTO), or the System Clock.

For timer operation in Stop Mode, Pclk or the WTO must be selected as the clock source. Either Pclk or the WTO can be selected as source for Active, Halt, and Stop Mode operation. System Clock is only for operation in Active and Halt modes.

Caution: When the timer is operating on Pclk or the WTO, the timer clock is asynchronous to System Clock. To ensure error-free operation, disable the timer before modifying its operation (including changing the timer clock source).

When the Timer uses Pclk or the WTO and the Timer is enabled, any read from T*x*H or T*x*L is not recommended, because results can be unpredictable. Disable the Timer first, then read it. If capture, capture/compare, Capture Restart or demodulation mode is selected, any read from T*x*PWM0H, T*x*PWM0L, T*x*PWM1H, T*x*PWM1L, or T*x*STAT must be done after capture interrupt occurs, or results can be unpredictable. INPCAP in the Timer Control 0 Register has the same characteristics as these PWM registers. When the Timer clock selection is System Clock, registers can be written/read at any time.

10.1.2. Low-Power Modes

Timers can operate in both Halt Mode and Stop Mode. This section discusses each of these low-power modes.

10.1.2.1. Operation in Halt Mode

When the eZ8 CPU enters Halt Mode, the Timer will continue to operate if enabled. To minimize current in Halt Mode, the Timer can be disabled by clearing the TEN control bit. The Noise Filter, if enabled, will also continue to operate in Halt Mode and rejects any noise on the Timer Input 0.

10.1.2.2. Operation in Stop Mode

When the eZ8 CPU enters Stop Mode, the Timer continues to operate if enabled and Pclk or the WTO is selected as the timer clock. In Stop Mode, the timer interrupt (if enabled) automatically initiates a Stop-Mode Recovery and generates an interrupt request. In the Reset Status Register, the stop bit is set to 1. Also, timer interrupt request bit in Interrupt Request 0 Register is set. Following completion of the Stop-Mode Recovery, if interrupts are enabled, the CPU responds to the interrupt request by fetching the timer interrupt vector. The Noise Filter, if enabled, will also continue to operate in Stop Mode and rejects any noise on the Timer Input 0.

If System Clock is chosen as the timer clock, the Timer ceases to operate as System Clock is disabled in Stop Mode. In this case the registers are not reset and operation will resume after Stop-Mode Recovery occurs.

10.1.2.3. Power Reduction During Operation

Clearing TEN will inhibit clocking of the Timer. The CPU can still read/write registers when TEN is cleared.

10.1.3. Timer Operating Modes

The Timer can be configured to operate in the following modes, each of which is described in this section where indicated in Table 65.

Table 65. Timer Operating Modes

Mode	Page #
One-Shot Mode	112
Triggered One-Shot Mode	113
Dual Input Triggered One-Shot Mode	115
Continuous Mode	117
Counter Mode	118
PWM Single Output Mode	119
PWM Dual Output Mode	121
Capture Mode	122
Capture Restart Mode	124
Compare Mode	125
Gated Mode	126
Capture/Compare Mode	127
Demodulation Mode	128

10.1.3.1. One-Shot Mode

In One-Shot Mode (TMODE= 0000), the Timer counts timer clocks up to the 16-bit reload value stored in the Timer Reload High and Low Byte registers. Upon reaching the reload value, the Timer generates an interrupt, and the count value in the Timer High and Low Byte registers is reset to `0001h`. Then, the Timer is automatically disabled and stops counting.

Additionally, the OUTCTL configuration determines whether the Timer Output changes state for one clock cycle (from Low to High or from High to Low) upon timer reload (OUTCTL=00) or changes state from timer start until timer reload (OUTCTL=01). Use OUTCLT=01 only if TPOL=1. The Timer Output can be connected to the Event System and, if the Timer Output alternate function is enabled, to the Timer Output pin. If it is appropriate to have the Timer Output make a persistent state change on One-Shot time-out, first configure the TPOL bit in the Timer Control 1 Register to the start value before beginning One-Shot Mode. Then, after starting the timer, configure TPOL to the opposite bit value.

Observe the following steps to configure a timer for One-Shot Mode and to initiate the count.

1. Write to the Timer Control 1 Register to:
   - Disable the timer
   - Configure the timer for One-Shot Mode
   - Set the prescale value
   - If using the Timer Output alternate function or the Event System, set the initial output level (High or Low) and configure the output behavior (OUTCTL)

2. Write to the Timer Control 2 Register to choose the timer clock source.
3. Write to the Timer Control 0 Register to set the timer interrupt configuration field TICONFIG.
4. Write to the Timer High and Low Byte registers to set the starting count value.
5. Write to the Timer Reload High and Low Byte registers to set the reload value.
6. If required, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
7. If using the Timer Output function, configure the associated GPIO port pin for the Timer Output alternate function. If using the Event System, configure it to route the Timer Output to the desired destination.
8. Write to the Timer Control 1 Register to enable the timer and initiate counting.

In One-Shot Mode, the timer clock always provides the timer input. The timer period is calculated using the following equation:

$$\text{ONE-SHOT Mode Time-Out Period (s)} = \frac{(\text{Reload Value - Start Value})\text{xPrescale}}{\text{Timer Clock Frequency (Hz)}}$$

10.1.3.2. Triggered One-Shot Mode

In Triggered One-Shot Mode (TMODE=1011), the Timer Input 0 signal triggers counting. Two timer output options and four interrupt options are available. Timer Input 1 can be used in the interrupt control to signal that a trigger event occurred while counting. If a trigger event occurs while counting, INPCAP is set to indicate that the interrupt is due to the trigger event. If a reload occurs, INPCAP is cleared to indicate that the interrupt is not due to a trigger event. The timer operates in the following sequence:

1. The Timer idles until a trigger is received. The Timer trigger, the Input 0 signal, is taken from the GPIO port pin timer input alternate function or from the Event System. If required, enable the Noise Filter and set the Noise Filter control by writing to the relevant bits in the Noise Filter Control Register. The TPOL bit in the Timer Control 1 Register selects whether the triggering occurs on the rising edge or the falling edge of the timer Input 0 signal.

2. Following the trigger event, the Timer counts timer clocks up to the 16-bit reload value stored in the Timer Reload High and Low Byte registers.
3. The timer output polarity is selected by TPOL and timer output behavior is selected in OUTCTL to be one of the following:
   - OUTCTL=00: Pulse at reload lasting one timer clock
   - OUTCTL=01: Pulse from start to reload. Use this setting only with TPOL = 0.
   - OUTCTL=10: Pulse lasting one timer clock upon a timer Input 0 trigger event while counting
   - OUTCTL=11: Pulse lasting one timer clock upon a timer Input 1 trigger event while counting

4. Timer interrupt behavior is selected by TICONFIG to occur upon one of the following:
   - TICONFIG=00: All Reload and Input 1 trigger while counting Events
   - TICONFIG=01: Only on timer Input 0 trigger events while counting
   - TICONFIG=10: Only on timer Input 1 trigger events while counting
   - TICONFIG=11: Only on reload events

5. Upon reaching the reload value, the timer resets the count value in the Timer High and Low Byte registers to `0001h`. The Timer now idles until the next timer Input 0 trigger event.

In Triggered One-Shot Mode, the timer clock always provides the timer input. The timer period is shown in the following equation:

$$\text{Triggered ONE-SHOT Mode Time-Out Period (s)} = \frac{(\text{Reload Value - Start Value}) \times \text{Prescale}}{\text{Timer Clock Frequency (Hz)}}$$

Table 66 provides an example initialization sequence for configuring Timer 0 in Triggered One-Shot Mode and initiating operation.

Table 66. Triggered One-Shot Mode Initialization Example

Register	Value	Comment
T0CTL0	E0h	TMODE[3:0]=1011b selects Triggered One-Shot Mode. TICONFIG[1:0]=11b enables interrupts on Timer reload only. PWMD[2:0]=000b has no effect. INPCAP=0 has no effect. TEN=0 disables the timer. OUTCTL[1:0]=00b selects a single clock output pulse upon timer reload. TPOL=0 enables triggering on rising edge of Timer Input 0 and sets the Timer Output signal to 0. PRES[2:0]=000b sets prescaler to divide by 1. TCLKS=10b selects Pclk as the Timer clock source.
T0CTL1	03h	
T0CTL2	01h	
T0H	00h	Timer starting value=0001h.
T0L	01h	
T0RH	ABh	Timer reload value=ABCDh.
T0RL	CDh	
PAADDR	02h	Selects Port A Alternate Function Register.
PACTL[1:0]	11b	PACTL[0] enables Timer 0 Input Alternate function if also selected by the Alternate Function Set 1 Register. PACTL[1] enables Timer 0 Output Alternate function if also selected by the Alternate Function Set 1 Register.
PAADDR	07h	Selects Port A Alternate Function Set 1 Register.
PACTL[1:0]	00b	PACTL[0] enables Timer 0 Input Alternate function. PACTL[1] enables Timer 0 Output Alternate function.
ESDADDR	10h	Selects the Timer 0 Input 0 Event System Destination.
ESDCTL	00h	Disconnects the Event System Input0 to Timer 0.
IRQ0ENH[5]	0b	Disables the Timer 0 interrupt.
IRQ0ENL[5]	0b	
T0CTL1	83h	TEN=1 enables the timer. All other bits remain in their appropriate settings.

Note: After receiving an input trigger, Timer 0 will:
1. Count ABCDh timer clocks.
2. Upon Timer 0 reload, generate a single clock cycle active High output pulse on the Timer 0 Output.
3. Wait for the next input trigger event.

10.1.3.3. Dual Input Triggered One-Shot Mode

In Dual Input Triggered One-Shot Mode (TMODE=1010), the first to arrive of the timer Input 0 or Input 1 signals triggers counting. In addition, the input that was first to arrive is recorded by the timer for later use in interrupt and output generation. Two timer output options and four interrupt options are available. Previous and current timer input triggers can be used in the interrupt control. If an interrupt is generated due to a trigger event while counting, INPCAP is set. If an interrupt is generated due to reload, INPCAP is cleared. The timer operates in the following sequence:

1. The Timer idles until a trigger is received. The Timer trigger, in essence the first to arrive of the timer Input 0 and Input 1 signals, is taken from the GPIO port pin timer input alternate function or from the Event System. If required, enable the Noise Filter and set the Noise Filter control by writing to the relevant bits in the Noise Filter Control Register. The TPOL bit in the Timer Control 1 Register selects whether the trigger occurs on the rising edge or the falling edge of the timer Input 0 signal.
2. Following the trigger event, the Timer counts timer clocks up to the 16-bit reload value stored in the Timer Reload High and Low Byte registers.
3. The timer output polarity is selected by TPOL and timer output behavior is selected in OUTCTL to be one of the following:
   - OUTCTL=00: Pulse at reload lasting one timer clock
   - OUTCTL=01: Pulse from start to reload. Use this setting only with TPOL = 0.
   - OUTCTL=10: Pulse lasting one timer clock occurring when the same timer input that triggered counting (either timer Input 0 or timer Input 1) triggers while counting
   - OUTCTL=11: Pulse lasting one timer clock occurring when the timer input that did not trigger counting (either timer Input 0 or timer Input 1) triggers while counting
4. Timer interrupt behavior is selected by TICONFIG to occur upon one of the following:
   - TICONFIG=00: All reload events and all trigger events while counting
   - TICONFIG=01: Only on same input trigger events while counting
   - TICONFIG=10: Only on opposite input trigger events while counting
   - TICONFIG=11: Only on reload events
5. Upon reaching the reload value, the timer resets the count value in the Timer High and Low Byte registers to `0001h`. The Timer now idles until the next timer Input 0 or Input 1 trigger event.

In Dual Input Triggered One-Shot Mode, the timer clock always provides the timer input. The timer period is shown in the following equation:

$$\text{DualTriggered ONE-SHOT Mode Time-Out Period (s)} = \frac{(\text{Reload Value - Start Value}) \times \text{Prescale}}{\text{Timer Clock Frequency (Hz)}}$$

10.1.3.4. Continuous Mode

In Continuous Mode (TMODE=0001), the timer counts timer clocks up to the 16-bit reload value stored in the Timer Reload High and Low Byte registers. Upon reaching the reload value, the timer generates an interrupt, the count value in the Timer High and Low Byte registers is reset to `0001h` and counting resumes. Also, the Timer Output changes state (from Low to High or High to Low) on timer reload. The Timer Output can be connected to the Event System and, if the Timer Output alternate function is enabled, to the Timer Output pin.

Observe the following steps to configure a timer for Continuous Mode and initiate the count:

1. Write to the Timer Control 1 Register to:
   - Disable the timer
   - Configure the timer for Continuous Mode
   - Set the prescale value
   - If using the Timer Output Alternate Function or the Event System, set the initial output level (High or Low)
2. Write to the Timer Control 2 Register to choose the timer clock source.
3. Write to the Timer Control 0 Register to set the timer interrupt-configuration field TICONFIG.
4. Write to the Timer High and Low Byte registers to set the starting count value (usually `0001h`). This value only affects the first pass in Continuous Mode. After the first timer reload in Continuous Mode, counting always begins at the reset value of `0001h`.
5. Write to the Timer Reload High and Low Byte registers to set the reload value.
6. If appropriate, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
7. If using the Timer Output function, configure the associated GPIO port pin for the Timer Output alternate function. If using the Event System, configure it to route the Timer Output to the desired destination.
8. Write to the Timer Control 1 Register to enable the timer and initiate counting.

In Continuous Mode, the timer clock always provides the timer input. The timer period is calculated using the following equation:

$$\text{CONTINUOUS Mode Time-Out Period (s)} = \frac{\text{Reload Value} \times \text{Prescale}}{\text{Timer Clock Frequency (Hz)}}$$

If an initial starting value other than `0001h` is loaded into the Timer High and Low Byte registers, the One-Shot Mode equation must be used to determine the first time-out period.

10.1.3.5. Counter Mode

In Counter Mode (TMODE=0010), the timer counts timer Input 0 signal transitions. The timer Input 0 signal is taken from the GPIO Port pin Timer Input alternate function or from the Event System Input 0. The TPOL bit in the Timer Control 1 Register selects whether the count occurs on the rising edge or the falling edge of the Timer Input 0 signal. In Counter Mode, the prescaler is disabled.

Caution: The input frequency of the Timer Input signal must not exceed one-fourth the timer clock frequency.

Upon reaching the reload value stored in the Timer Reload High and Low Byte registers, the timer generates an interrupt, the count value in the Timer High and Low Byte registers is reset to `0001h` and counting resumes. Also, the Timer Output pin changes state (from Low to High or High to Low) at timer reload. The Timer Output can be connected to the Event System and, if the Timer Output alternate function is enabled, to the Timer Output pin.

Observe the following steps to configure a timer for Counter Mode and initiate the count:

1. Write to the Timer Control 1 Register to:
   - Disable the timer.
   - Configure the timer for Counter Mode.
   - Select either the rising edge or falling edge of the Timer Input 0 signal for the count. This also sets the initial logic level (High or Low) for the Timer Output Alternate Function. However, the Timer Output function is not required to be enabled.
2. Write to the Timer Control 2 Register to choose the timer clock source.
3. Write to the Timer Control 0 Register to set the timer interrupt configuration field TICONFIG.
4. Write to the Timer High and Low Byte registers to set the starting count value. This value only affects the first pass in Counter Mode. After the first timer reload in Counter Mode, counting always begins at the reset value of `0001h`. Generally, in Counter Mode the Timer High and Low Byte registers should be written with the value `0001h`.
5. Write to the Timer Reload High and Low Byte registers to set the reload value.
6. If required, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.

7. If required, enable the Noise Filter and set the Noise Filter control by writing to the relevant bits in the Noise Filter Control Register.
8. Configure the associated GPIO port pin for the Timer Input alternate function or configure the desired Event System Timer Input 0.
9. When using the Timer Output, configure the associated GPIO port pin for the Timer Output alternate function or configure the Event System to route the Timer Output to the desired destination.
10. Write to the Timer Control 1 Register to enable the timer.

In Counter Mode, the number of Timer Input 0 transitions since the timer start is calculated using the following equation:

COUNTER Mode Timer Input Transitions = Current Count Value - Start Value

10.1.3.6. PWM Single Output Mode

In PWM Single Output Mode (TMODE=0011), the timer outputs a Pulse Width Modulator output signal through a GPIO Port pin and/or to the Event System. The Timer counts timer clocks up to the 16-bit reload value. The timer first counts up to the 16-bit PWM match value stored in the Timer PWM0 High and Low Byte registers. When the timer count value matches the PWM value, the Timer Output toggles. The timer continues counting until it reaches the reload value stored in the Timer Reload High and Low Byte registers. Upon reaching the reload value, the timer generates an interrupt, the count value in the Timer High and Low Byte registers is reset to `0001h` and counting resumes.

If the TPOL bit in the Timer Control 1 Register is set to 1, the Timer Output signal begins as High (1) and then transitions to Low (0) when the timer value matches the PWM value. The Timer Output signal returns to High (1) after the timer reaches the reload value and is reset to `0001h`.

If the TPOL bit in the Timer Control 1 Register is set to 0, the Timer Output signal begins as Low (0) and then transitions to High (1) when the timer value matches the PWM value. The Timer Output signal returns to Low (0) after the timer reaches the reload value and is reset to `0001h`.

Observe the following steps to configure a timer for PWM Single Output Mode and initiate PWM operation:

1. Write to the Timer Control 1 Register to:
   – Disable the timer
   – Configure the timer for PWM Mode
   – Set the prescale value

- Set the initial logic level (High or Low) and PWM High/Low transition for the Timer Output Alternate Function

2. Write to the Timer Control 2 Register to choose the timer clock source.
3. Write to the Timer Control 0 Register to set the timer interrupt configuration field TICONFIG.
4. Write to the Timer High and Low Byte registers to set the starting count value (typically `0001h`). This value only affects the first pass in PWM Mode. After the first timer reset in PWM Mode, counting always begins at the reset value of `0001h`.
5. Write to the Timer PWM0 High and Low Byte registers to set the PWM value.
6. Write to the Timer Reload High and Low Byte registers to set the reload value (PWM period). The reload value must be greater than the PWM value.
7. If appropriate, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
8. Configure the associated GPIO port pin for the Timer Output alternate function and/or the Event System to route the Timer Output to the desired destination.
9. Write to the Timer Control 1 Register to enable the timer and initiate counting.

The PWM period is calculated using the following equation:

$$\text{PWM Period (s)} = \frac{\text{Reload Value} \times \text{Prescale}}{\text{Timer Clock Frequency (Hz)}}$$

If an initial starting value other than `0001h` is loaded into the Timer High and Low Byte registers, the One-Shot Mode equation must be used to determine the first PWM time-out period.

If TPOL is set to 0, the ratio of the PWM output High time to the total period is calculated using the following equation:

$$\text{PWM Output High Time Ratio (\%)} = \frac{\text{Reload Value - PWM Value}}{\text{Reload Value}} \times 100$$

If TPOL is set to 1, the ratio of the PWM output High time to the total period is calculated using the following equation:

$$\text{PWM Output High Time Ratio (\%)} = \frac{\text{PWM Value}}{\text{Reload Value}} \times 100$$

10.1.3.7. PWM Dual Output Mode

In PWM Dual Output Mode (TMODE=1000), the timer outputs a Pulse Width Modulator output signal and also its complement through two GPIO Port pins and/or to the Event System. The timer first counts up to 16-bit PWM match value stored in the Timer PWM0 High and Low Byte registers. When the timer count value matches the PWM value, the Timer Outputs (TOUT and $\overline{\text{TOUT}}$) toggle. The timer continues counting until it reaches the reload value stored in the Timer Reload High and Low Byte registers. Upon reaching the reload value, the timer generates an interrupt, the count value in the Timer High and Low Byte registers is reset to `0001h` and TOUT and $\overline{\text{TOUT}}$ toggles again and counting resumes.

If the TPOL bit in the Timer Control 1 Register is set to 1, the Timer Output signal begins as High (1) and then transitions to Low (0) when the timer value matches the PWM value. The Timer Output signal returns to High (1) after the timer reaches the reload value and is reset to `0001h`.

If the TPOL bit in the Timer Control 1 Register is set to 0, the Timer Output signal begins as Low (0) and then transitions to High (1) when the timer value matches the PWM value. The Timer Output signal returns to Low (0) after the timer reaches the reload value and is reset to `0001h`.

The timer also generates a second PWM output signal, Timer Output Complement ($\overline{\text{TOUT}}$). $\overline{\text{TOUT}}$ is the complement of the Timer Output PWM signal (TOUT). A programmable deadband delay can be configured to set a time delay (0 to 128 timer clock cycles) when one PWM output transitions from High to Low and the other PWM output transitions from a Low to High. This configuration ensures a time gap between the removal of one PWM output and the assertion of its complement.

Observe the following steps to configure a timer for PWM Dual Output Mode and initiate the PWM operation:

1. Write to the Timer Control 1 Register to:
   - Disable the timer
   - Configure the timer for PWM Dual Output Mode. Setting the mode also involves writing to TMODE[3] bit in the T*x*CTL0 Register
   - Set the prescale value
   - Set the initial logic level (High or Low) and PWM High/Low transition for the Timer Output Alternate Function
2. Write to the Timer High and Low Byte registers to set the starting count value (typically `0001h`). This value only affects the first pass in PWM Mode. After the first timer reset in PWM Mode, counting always begins at the reset value of `0001h`.
3. Write to the Timer PWM0 High and Low Byte registers to set the PWM value.
4. Write to the Timer Control 0 Register:

- To set the PWM deadband delay value
- To choose the timer clock source

5. Write to the Timer Control 0 Register to set the timer interrupt configuration field TICONFIG.
6. Write to the Timer Reload High and Low Byte registers to set the reload value (PWM period). The reload value must be greater than the PWM value.
7. If appropriate, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
8. Configure the associated GPIO port pin for the Timer Output and Timer Output Complement alternate functions and/or the Event System to route the Timer Output to the desired destination.
9. Write to the Timer Control 1 Register to enable the timer and initiate counting.

The PWM period is calculated using the following equation:

$$\text{PWM Period (s)} = \frac{\text{Reload Value} \times \text{Prescale}}{\text{Timer Clock Frequency (Hz)}}$$

If an initial starting value other than `0001h` is loaded into the Timer High and Low Byte registers, the One-Shot Mode equation must be used to determine the first PWM time-out period.

If TPOL is set to 0, the ratio of the PWM output High time to the total period is calculated using the following equation:

$$\text{PWM Output High Time Ratio (\%)} = \frac{\text{Reload Value - PWM Value}}{\text{Reload Value}} \times 100$$

If TPOL is set to 1, the ratio of the PWM output High time to the total period is calculated using the following equation:

$$\text{PWM Output High Time Ratio (\%)} = \frac{\text{PWM Value}}{\text{Reload Value}} \times 100$$

10.1.3.8. Capture Mode

In Capture Mode (TMODE=0100), the current timer count value is recorded when the appropriate external Timer Input 0 transition occurs. The Capture count value is written to the Timer PWM0 High and Low Byte registers. The Timer counts timer clocks up to the 16-bit reload value. The TPOL bit in the Timer Control 1 Register determines if the Capture occurs on a rising edge or a falling edge of the Timer Input 0 signal. When the Capture event occurs, an interrupt is generated and the timer continues counting. The INPCAP bit in Timer Control 0 Register is set to indicate the timer interrupt is due to an input capture event.

The timer continues counting up to the 16-bit reload value stored in the Timer Reload High and Low Byte registers. Upon reaching the reload value, the timer generates an interrupt and continues counting. The INPCAP bit in Timer Control 0 Register is cleared to indicate the timer interrupt is not due to an input capture event.

Observe the following steps to configure a timer for Capture Mode and initiate the count:

1. Write to the Timer Control 1 Register to:
   - Disable the timer
   - Configure the timer for Capture Mode
   - Set the prescale value
   - Set the Capture edge (rising or falling) for the Timer Input 0
2. Write to the Timer Control 2 Register to choose the timer clock source.
3. Write to the Timer Control 0 Register to set the timer interrupt configuration field TICONFIG.
4. Write to the Timer High and Low Byte registers to set the starting count value (typically `0001h`).
5. Write to the Timer Reload High and Low Byte registers to set the reload value.
6. Clear the Timer PWM High and Low Byte registers to `0000h`. This allows user software to determine if interrupts were generated by either a capture event or a reload. If the PWM0 High and Low Byte registers still contain `0000h` after the interrupt, then the interrupt was generated by a Reload.
7. If required, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers. By default, the timer interrupt will be generated for both input capture and reload events. If required, configure the timer interrupt to be generated only at the input capture event or the reload event by setting TICONFIG field of the Timer Control 0 Register.
8. Configure the associated GPIO port pin for the Timer Input alternate function or configure the desired Event System Timer Input 0.
9. Write to the Timer Control 1 Register to enable the timer and initiate counting.

In Capture Mode, the elapsed time from timer start to Capture event can be calculated using the following equation:

$$\text{Capture Elapsed Time (s)} = \frac{(\text{Capture Value - Start Value}) \times \text{Prescale}}{\text{Timer Clock Frequency (Hz)}}$$

10.1.3.9. Capture Restart Mode

In Capture Restart Mode (TMODE=1001), the current timer count value is recorded when the appropriate external Timer Input 0 transition occurs. The Capture count value is written to the Timer PWM0 High and Low Byte registers. The Timer counts timer clocks up to the 16-bit reload value. The TPOL bit in the Timer Control 1 Register determines if the Capture occurs on a rising edge or a falling edge of the Timer Input 0 signal. When the Capture event occurs, an interrupt is generated and the count value in the Timer High and Low Byte registers is reset to `0001h` and counting resumes. The INPCAP bit in Timer Control 0 Register is set to indicate the timer interrupt is due to an input capture event.

If no Capture event occurs, the timer counts up to the 16-bit Compare value stored in the Timer Reload High and Low Byte registers. Upon reaching the reload value, the timer generates an interrupt, the count value in the Timer High and Low Byte registers is reset to `0001h` and counting resumes. The INPCAP bit in Timer Control 0 Register is cleared to indicate the timer interrupt is not due to an input capture event.

Observe the following steps to configure a timer for Capture Restart Mode and initiate the count:

1. Write to the Timer Control 1 Register to:
   - Disable the timer
   - Configure the timer for Capture Restart Mode. Setting the mode also involves writing to TMODE[3] bit in the T*x*CTL0 Register
   - Set the prescale value
   - Set the Capture edge (rising or falling) for the Timer Input 0
2. Write to the Timer Control 2 Register to choose the timer clock source.
3. Write to the Timer Control 0 Register to set the timer interrupt configuration field TICONFIG.
4. Write to the Timer High and Low Byte registers to set the starting count value (typically `0001h`).
5. Write to the Timer Reload High and Low Byte registers to set the reload value.
6. Clear the Timer PWM High and Low Byte registers to `0000h`. This allows user software to determine if interrupts are generated by either a Capture Event or a Reload. If the PWM0 High and Low Byte registers still contain `0000h` after the interrupt, then the interrupt is generated by a Reload.
7. If required, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers. By default, the timer interrupt will be generated for both input capture and reload events. If required, configure the timer interrupt to be generated only at the Input Capture event or the reload event by setting TICONFIG field of the Timer Control 0 Register.

8. Configure the associated GPIO port pin for the Timer Input alternate function or configure the desired Event System Timer Input 0.
9. Write to the Timer Control 1 Register to enable the timer and initiate counting.

In Capture Mode, the elapsed time from Timer start to Capture event can be calculated using the following equation:

$$\text{Capture Elapsed Time (s)} = \frac{(\text{Capture Value - Start Value}) \times \text{Prescale}}{\text{Timer Clock Frequency (Hz)}}$$

10.1.3.10.Compare Mode

In Compare Mode (TMODE=0101), the timer counts timer clocks up to the 16-bit maximum Compare value stored in the Timer Reload High and Low Byte registers. Upon reaching the Compare value, the timer generates an interrupt and counting continues (the timer value is not reset to `0001h`). Also, the Timer Output changes state (from Low to High or from High to Low) on Compare. The Timer Output can be connected to the Event System and, if the Timer Output alternate function is enabled, to the Timer Output pin.

If the Timer reaches `FFFFh`, the timer rolls over to `0000h` and continues counting.

Observe the following steps to configure a timer for Compare Mode and initiate the count:

1. Write to the Timer Control 1 Register to:
   - Disable the timer
   - Configure the timer for Compare Mode
   - Set the prescale value
   - Set the initial logic level (High or Low) for the Timer Output, if required
2. Write to the Timer Control 2 Register to choose the timer clock source.
3. Write to the Timer Control 0 Register to set the timer interrupt configuration field, TICONFIG.
4. Write to the Timer High and Low Byte registers to set the starting count value.
5. Write to the Timer Reload High and Low Byte registers to set the Compare value.
6. If appropriate, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
7. If using the Timer Output function, configure the associated GPIO port pin for the Timer Output alternate function. If using the Event System, configure it to route the Timer Output to the desired destination.
8. Write to the Timer Control 1 Register to enable the timer and initiate counting.

In Compare Mode, the timer clock always provides the timer input. The Compare time is calculated using the following equation:

$$\text{COMPARE Mode Time (s)} = \frac{(\text{Compare Value - Start Value}) \times \text{Prescale}}{\text{Timer Clock Frequency (Hz)}}$$

10.1.3.11.Gated Mode

In Gated Mode (TMODE=0110), the timer counts only when the Timer Input 0 signal is in its active state (asserted) as determined by the TPOL bit in the Timer Control 1 Register. When the Timer Input 0 signal is asserted, counting begins. A Timer Interrupt is generated when the Timer Input 0 signal is deasserted or a timer reload occurs. The INPCAP bit in Timer Control 0 Register is set to indicate the timer interrupt is due to the Timer Input signal.

The timer counts up to the 16-bit reload value stored in the Timer Reload High and Low Byte registers using the timer clock. When reaching the reload value, the timer generates an interrupt, the count value in the Timer High and Low Byte registers is reset to `0001h` and counting resumes (assuming the Timer Input 0 signal is still asserted). Also, the Timer Output changes state (from Low to High or from High to Low) at timer reset. The Timer Output can be connected to the Event System and, if the Timer Output alternate function is enabled, to the Timer Output pin.

Observe the following steps to configure a timer for Gated Mode and initiate the count:

1. Write to the Timer Control 1 Register to:
   - Disable the timer
   - Configure the timer for Gated Mode
   - Set the prescale value
2. Write to the Timer Control 2 Register to choose the timer clock source.
3. Write to the Timer Control 0 Register to set the timer interrupt configuration field TICONFIG.
4. Write to the Timer High and Low Byte registers to set the starting count value. This value only affects the first pass in Gated Mode. After the first timer reset in Gated Mode, counting always begins at the reset value of `0001h`.
5. Write to the Timer Reload High and Low Byte registers to set the reload value.
6. If required, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers. By default, the timer interrupt will be generated for both input deassertion and reload events. If required, configure the timer interrupt to be generated only at the Input Deassertion event or the Reload event by setting TICONFIG field of the Timer Control 0 Register.

7. Configure either the associated GPIO port pin for the Timer Input alternate function or the Event System Timer Input 0.
8. Write to the Timer Control 1 Register to enable the timer.
9. Assert the Timer Input 0 signal to initiate the counting.

10.1.3.12.Capture/Compare Mode

In Capture/Compare Mode (TMODE=0111), the timer begins counting on the first external Timer Input 0 transition. The appropriate transition (rising edge or falling edge) is set by the TPOL bit in the Timer Control 1 Register. The Timer counts timer clocks up to the 16-bit reload value.

Every subsequent appropriate transition (after the first) of the Timer Input 0 signal captures the current count value. The Capture value is written to the Timer PWM0 High and Low Byte registers. When the Capture event occurs, an interrupt is generated, the count value in the Timer High and Low Byte registers is reset to `0001h` and counting resumes. The INPCAP bit in Timer Control 0 Register is set to indicate the timer interrupt is due to an input capture event.

If no Capture event occurs, the timer counts up to the 16-bit Compare value stored in the Timer Reload High and Low Byte registers. Upon reaching the Compare value, the timer generates an interrupt, the count value in the Timer High and Low Byte registers is reset to `0001h` and counting resumes. The INPCAP bit in Timer Control 0 Register is cleared to indicate the timer interrupt is not due to an input capture event.

Observe the following steps to configure a timer for Capture/Compare Mode and initiate the count:

1. Write to the Timer Control 1 Register to:
   - Disable the timer
   - Configure the timer for Capture/Compare Mode
   - Set the prescale value
   - Set the Capture edge (rising or falling) for the Timer Input 0
2. Write to the Timer High and Low Byte registers to set the starting count value (typically `0001h`).
3. Write to the Timer Control 2 Register to choose the timer clock source.
4. Write to the Timer Control 0 Register to set the timer interrupt configuration field TICONFIG.
5. Write to the Timer Reload High and Low Byte registers to set the Compare value.
6. If required, enable the timer interrupt and set the timer-interrupt priority by writing to the relevant interrupt registers. By default, the timer interrupt will be generated for

both input capture and reload events. If required, configure the timer interrupt to be generated only at the input Capture event or the Reload event by setting TICONFIG field of the Timer Control 0 Register.

7. Configure the associated GPIO port pin for the Timer Input alternate function or configure the desired Event System Timer Input 0.
8. Write to the Timer Control 1 Register to enable the timer.
9. Counting begins on the first transition of the Timer Input 0 signal. No interrupt is generated by this first edge.

In Capture/Compare Mode, the elapsed time from timer start to Capture event is calculated using the following equation:

$$\text{apture Elapsed Time (s)} = \frac{(\text{Capture Value - Start Value}) \times \text{Prescale}}{\text{Timer Clock Frequency (Hz)}}$$

10.1.3.13.Demodulation Mode

In Demodulation Mode (TMODE=1100), the timer begins counting on the first external Timer Input 0 transition. The appropriate transition (rising edge or falling edge or both) is set by the TPOL bit in the Timer Control 1 Register and TPOLHI bit in the Timer Control 2 Register. The Timer counts timer clocks up to the 16-bit reload value.

Every subsequent appropriate transition (after the first) of the Timer Input 0 signal captures the current count value. The Capture value is written to the Timer PWM0 High and Low Byte registers for rising input edges of the Timer Input 0 signal. For falling edges the capture count value is written to the Timer PWM1 High and Low Byte registers. The TPOL bit in the Timer Control 1 Register determines if the Capture occurs on a rising edge or a falling edge of the Timer Input 0 signal. If the TPOLHI bit in the Timer Control 2 Register is set, a Capture is executed on both the rising and falling edges of the input signal.

Whenever the Capture event occurs, an interrupt is generated and the timer continues counting. The corresponding event flag bit in the Timer Status Register, PWMxEF, is set to indicate that the timer interrupt is due to an input Capture event.

The timer counts up to the 16-bit Compare value stored in the Timer Reload High and Low Byte registers. Upon reaching the reload value, the timer generates an interrupt, the count value in the Timer High and Low Byte registers is reset to `0001h`, and counting resumes. The RTOEF event flag bit in the Timer Status Register is set to indicate that the timer interrupt is due to a Reload event. Software can use this bit to determine if a Reload occurred prior to a Capture.

Observe the following steps to configure a timer for Demodulation Mode and initiate the count:

1. Write to the Timer Control 1 Register to:

   - Disable the timer.
   - Configure the timer for Demodulation Mode. Setting the mode also involves writing to the TMODEHI bit in the T*x*CTL0 Register.
   - Set the prescale value.
   - Set the TPOL bit to set the Capture edge (rising or falling) for the Timer Input 0. This setting applies only if the TPOLHI bit in the T*x*CTL2 Register is not set.

2. Write to the Timer Control 2 Register to:
   - Choose the timer clock source.
   - Set the TPOLHI bit if the Capture is required on both edges of the input signal.

3. Write to the Timer Control 0 Register to set the timer interrupt configuration field TICONFIG.
4. Write to the Timer High and Low Byte registers to set the starting count value (typically `0001h`).
5. Write to the Timer Reload High and Low Byte registers to set the reload value.
6. Clear the Timer T*x*PWM0 and T*x*PWM1 High and Low Byte registers to `0000h`.
7. If required, enable the Noise Filter and set the Noise Filter control by writing to the relevant bits in the Noise Filter Control Register.
8. If required, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers. By default, the timer interrupt will be generated for both input capture and reload events. If required, configure the timer interrupt to be generated only at the input Capture event or the Reload event by setting TICONFIG field of the Timer Control 0 Register.
9. Configure the associated GPIO Port pin for the Timer Input alternate function or configure the desired Event System Timer Input 0.
10. Write to the Timer Control 1 Register to enable the timer. Counting will start on the occurrence of the first external input transition.

In Demodulation Mode, the elapsed time from timer start to Capture event can be calculated using the following equation:

$$\text{Capture Elapsed Time (s)} = \frac{(\text{Capture Value - Start Value}) \times \text{Prescale}}{\text{Timer Clock Frequency (Hz)}}$$

Table 67 provides an example initialization sequence for configuring Timer 0 in Demodulation Mode and initiating operation.

Table 67. Demodulation Mode Initialization Example

Register	Value	Comment
T0CTL0	C0h	TMODE[3:0]=1100b selects Demodulation Mode. TICONFIG[1:0]=10b enables interrupt only on Capture events. PWMD[2:0]=000b has no effect. INPCAP=0 has no effect. TEN=0 disables the timer. PRES[2:0]=000b sets prescaler to divide by 1. TPOLHI,TPOL=10b enables trigger and Capture on both rising and falling edges of Timer Input. TCLKS=10b enables Pclk as timer clock source
T0CTL1	04h	
T0CTL2	11h	
T0H	00h	Timer starting value=0001h.
T0L	01h	
T0RH	ABh	Timer reload value=ABCDh
T0RL	CDh	
T0PWM0H	00h	Initial PWM0 value=0000h
T0PWM0L	00h	
T0PWM1H	00h	Initial PWM1 value=0000h
T0PWM1H	00h	
T0NFC	70h	NFCTL=0111b enables 8-bit up/down Noise Filter counting
PAADDR	02h	Selects Port A Alternate Function control register.
PACTL[1:0]	11b	PACTL[0] enables Timer 0 Input alternate function. PACTL[1] enables Timer 0 Output alternate function.
PAADDR	07h	Selects Port A Alternate Function Set 1 Register.
PACTL[1:0]	00b	PACTL[0] enables Timer 0 Input Alternate function. PACTL[1] enables Timer 0 Output Alternate function.
ESDADDR	10h	Selects the Timer 0 Input 0 Event System Destination
ESDCTL	00h	Disconnects the Event System Input 0 to Timer 0.
IRQ0ENH[5]	0b	Disables the Timer 0 interrupt.
IRQ0ENL[5]	0b	
T0CTL1	84h	TEN=1 enables the timer. All other bits remain in their appropriate settings.

NotesAfter receiving the input trigger (rising or falling edge), Timer 0 will:

1. Start counting on the timer clock.
2. Upon receiving a Timer 0 Input 0 rising edge, save the Capture value in the T0PWM0 registers, generate an interrupt, and continue to count.
3. Upon receiving a Timer 0 Input 0 falling edge, save the Capture value in the T0PWM1 registers, generate an interrupt, and continue to count.
4. After the timer count to ABCD clocks, set the reload event flag and reset the Timer count to the start value.

10.1.4. Reading the Timer Count Values

The current count value in the timers can be read while counting (enabled). This capability has no effect on timer operation. When the timer is enabled and the Timer High Byte Register is read, the contents of the Timer Low Byte Register are placed in a holding register. A subsequent read from the Timer Low Byte Register returns the value in the holding register. This operation allows accurate reads of the full 16-bit timer count value while enabled. When the timers are not enabled, a read from the Timer Low Byte Register returns the actual value in the counter.

10.1.5. Timer Interrupts

The Timer can generate an interrupt request upon reload and capture. In addition, certain input triggering events can generate an interrupt, as described in the Triggered One-Shot Mode section on page 113 and the Dual Input Triggered One-Shot Mode section on page 115. Use TICONFIG to select whether interrupts are generated due to reload, capture or input triggering events. An interrupt request that is pending when the Timer is disabled is not automatically cleared.

10.1.6. Timer Output Signal Operation

The Timer Outputs, TOUT and $\overline{\text{TOUT}}$, are available as GPIO Port pin alternate functions and a sources to the Event System. $\overline{\text{TOUT}}$ is the complement of TOUT in all timer modes with the special case of DUAL PWM Mode in which the complementary behavior includes deadband insertion. Generally, the Timer Outputs are toggled every time the counter is reloaded. For One-Shot, Triggered One-Shot, and Dual Input Triggered One-Shot modes, the timer output waveforms are controlled by OUTCTL. Output connectivity to Event System destinations is controlled by the Event System registers. GPIO connectivity is controlled by the GPIO alternate function registers.

10.1.7. Timer Input Path and Noise Filter

The timer input path develops two timer inputs from three input signals as followings:

Input 0 is an OR function of a GPIO alternate function (TIN) and Event System Timer Input 0. The result is optionally polarity-adjusted and filtered.

Input 1 is Event System Timer Input 1. A noise filter from another timer can be assigned to this input.

A noise filter circuit is included which filters noise on the Timer Input 0 signal before it reaches the Timer.

The noise filter features the following elements:

- Synchronizes the input signal to the timer clock.

- The Noise Filter Control (NFCTL) input selects whether the Noise Filter is bypassed (NFCTL=`0000`) and the width of the up/down saturating counter digital filter. The available widths range from 2 bits to 11 bits.
- The digital filter output has hysteresis such that the data output does not change until the saturated value is reached.
- Provides an active-Low *Saturated State* output, FiltSatB, which is used as an indication of the presence of noise.
- Available for operation in Stop Mode.

10.1.7.1. Architecture

Figure 14 shows how the Input Path and Noise Filter are integrated with the Timer.

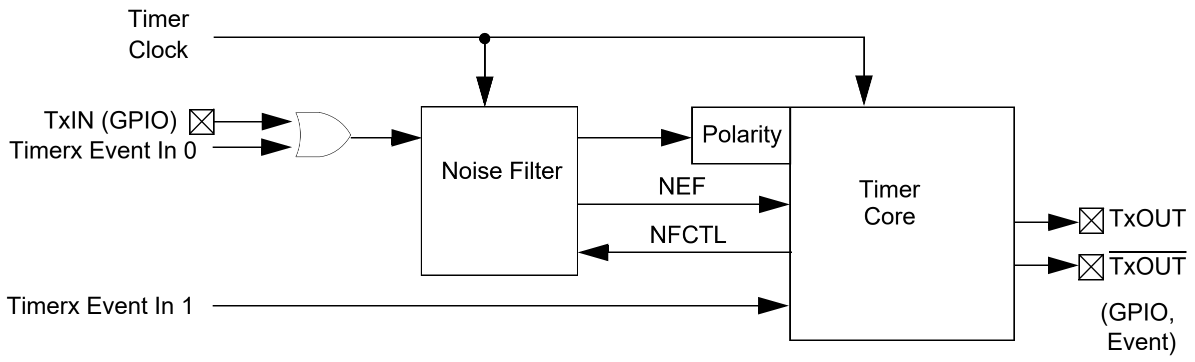


Figure 14. Input Path and Noise Filter System Block Diagram

10.1.7.2. Operation

Three signals are input to the timer: A GPIO alternate function (TIN), Event System Timer Input 0, and Event System Timer Input 1. The GPIO alternate function (TIN) and Event System Timer Input 0 are logically ORed; typically, only one of these is configured to be active at a given time. Any inactive input is held at logic 0.

The ORed signal is operated on by the Noise Filter and polarity adjustment (TPOL) to form the first timer input, Input 0. The Noise Filter is clocked by timer clock and can be used to filter noisy signals or to establish a minimum signal duration. The Event System Timer Input 1 is the second timer input, Input 1.

The OR function at Input 0 allows for special functionality.when using one of the capture modes, (TMODE=0100, 0111, 1100). For example, the delay between pulses on two different inputs can be captured.

Timer(x) Input 1 can also be filtered if the Timer(x–1) Noise Filter is assigned to it by setting NFCON for Timer(x–1). As such, Timer(x–1) Input 0 bypasses the Timer(x–1) Noise Filter, the Timer(x–1) Noise Filter is reassigned to Timer(x) Input 1 and uses the Timer(x) timer clock. When reassigning the Timer 2 Noise Filter, it is connected to Timer 0 Input 1.

Figure 15 shows an configuration example with the Timer0 Noise Filter reassigned to Timer1 (NFCON=1 in the T0NFC Register).

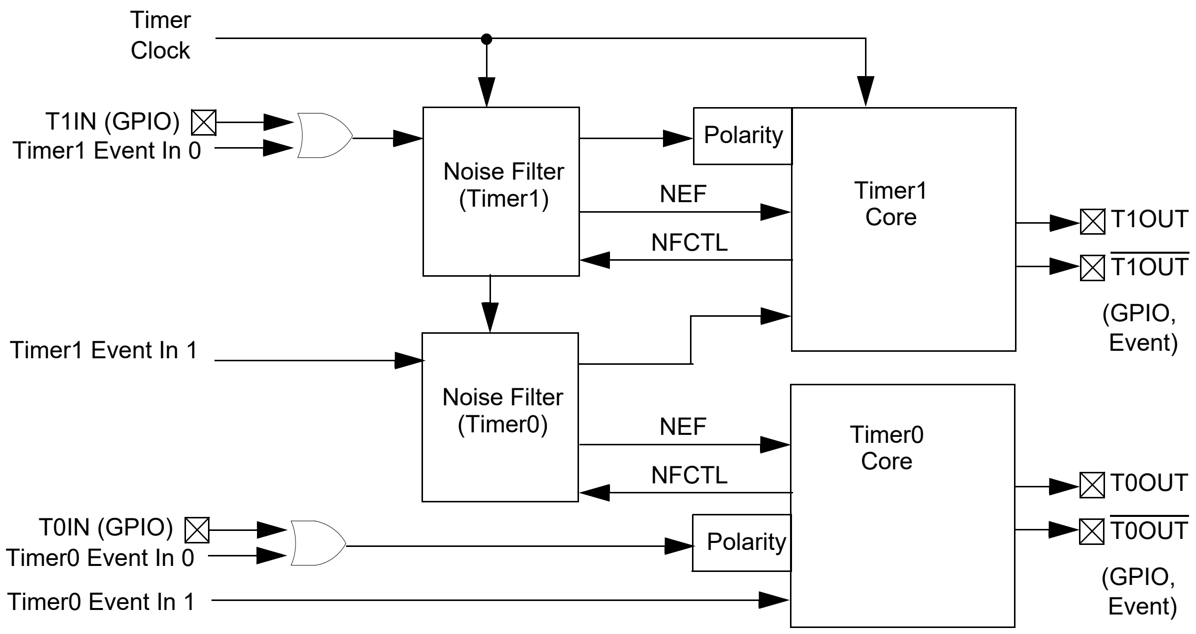


Figure 15. Example with the Timer0 Noise Filter Reassigned to Timer1

Figure 16 shows the operation of the Noise Filter with and without noise. The Noise Filter in this example is a 2-bit up/down counter which saturates at `00` and `11`. A 2-bit counter is described for convenience; the operation of wider counters is similar. The output of the filter switches from 1 to 0 when the counter counts down from `01` to `00` and switches from 0 to 1 when the counter counts up from `10` to `11`. The Noise Filter delays the receive data by three timer clock cycles.

The NEF output signal is checked when the filtered T*x*IN input signal is sampled. The Timer samples the filtered T*x*IN input near the center of the bit time. The NEF signal must be sampled at the same time to detect whether there is noise near the center of the bit time. The presence of noise (NEF=1 at the center of the bit time) does not mean that the sampled data is incorrect; rather, it is intended to be an indicator of the level of noise in the network.

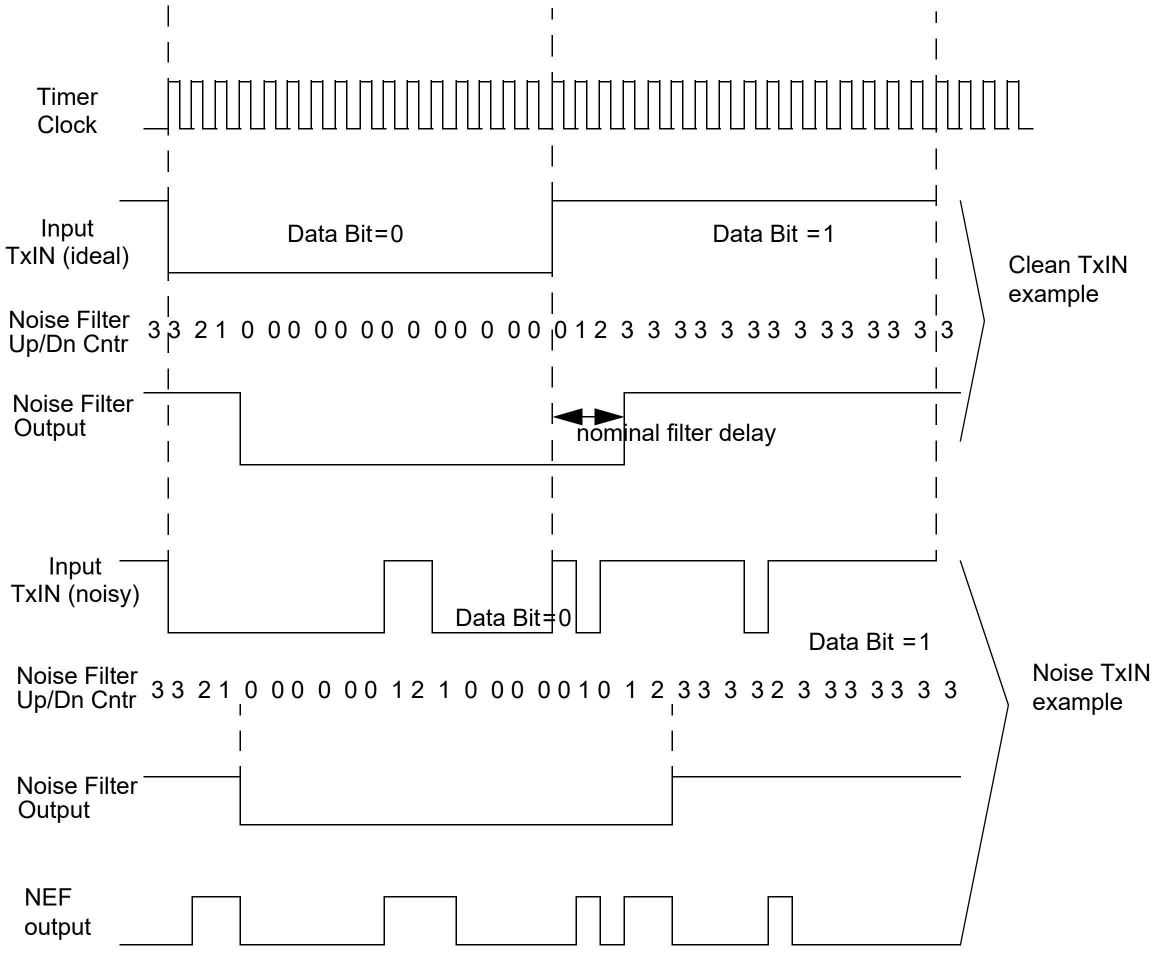


Figure 16. Noise Filter Operation

10.1. Timer Register Definitions

This section describes the following Timer registers.

- Timer 0–2 High and Low Byte Registers – see page 135
- Timer Reload High and Low Byte Registers – see page 136
- Timer 0–2 PWM0 High and Low Byte Registers – see page 137
- Timer 0–2 PWM1 High and Low Byte Registers – see page 138
- Timer 0–2 Control Registers – see page 139
- Timer 0–2 Status Registers – see page 145

- Timer 0–2 Noise Filter Control Registers – see page 146

10.1.1. Timer 0–2 High and Low Byte Registers

The Timer 0–2 High and Low Byte (T*x*H and T*x*L) registers, shown in Tables 68 and 69, contain the current 16-bit timer count value. When the timer is enabled, a read from T*x*H causes the value in T*x*L to be stored in a temporary holding register. A read from T*x*L always returns this temporary register when the timers are enabled. When the timer is disabled, reading from the T*x*L reads the register directly.

Zilog does not recommend writing to the Timer High and Low Byte registers when the timer is enabled. There are no temporary holding registers available for write operations; therefore simultaneous 16-bit writes are not possible. If either the Timer High or Low Byte registers are written during counting, the 8-bit written value is placed in the counter (High or Low Byte) at the next clock edge. The counter continues counting from the new value.

Table 68. Timer 0–2 High Byte Registers (TxH)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	TH							
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	T0H @ F00h, T1H @ F08h, T2H @ F10h							
Note: *x* references bits in the range [2:0].								

Table 69. Timer 0–2 Low Byte Registers (TxL)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	TL							
Reset	0	0	0	0	0	0	0	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	T0L @ F01h, T1L @ F09h, T2L @ F11h							
Note: *x* references bits in the range [2:0].								

Bit	**Description**
[7:0] TH, TL	**Timer High and Low Bytes** These 2 bytes, {TH[7:0], TL[7:0]}, contain the current 16-bit timer count value.

10.1.2. Timer Reload High and Low Byte Registers

The Timer 0–2 Reload High and Low Byte (T*x*RH and T*x*RL) registers, shown in Tables 70 and 71, store a 16-bit reload value, {TRH[7:0], TRL[7:0]}. Values written to these Timer Reload High Byte registers are stored in a temporary holding register. When a write to the Timer Reload Low Byte Register occurs, this temporary holding register value is written to the Timer High Byte Register. This operation allows simultaneous updates of the 16-bit timer reload value.

In Compare Mode, the Timer Reload High and Low Byte registers store the 16-bit Compare value.

Table 70. Timer 0–2 Reload High Byte Registers (T*x*RH)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	TRH							
Reset	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	T0RH @ F02h, T1RH @ F0Ah, T2RH @ F12h							
Note: *x* references bits in the range [2:0].								

Table 71. Timer 0–2 Reload Low Byte Registers (T*x*RL)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	TRL							
Reset	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	T0RL @ F03h, T1RL @ F0Bh, T2RL @ F13h							
Note: *x* references bits in the range [2:0].								

Bit	**Description**
[7:0] TRH, TRL	**Timer Reload Register High and Low** These two bytes form the 16-bit reload value, {TRH[7:0], TRL[7:0]}. This value is used to set the maximum count value which initiates a timer reload to 0001h. In Compare Mode, these two bytes form the 16-bit Compare value.

10.1.3. Timer 0–2 PWM0 High and Low Byte Registers

The Timer 0–2 PWM0 High and Low Byte (T*x*PWM0H and T*x*PWM0L) registers, shown in Tables 72 and 73, are used for Pulse Width Modulator (PWM) operations. These registers also store the Capture values for the Capture, Capture/Compare and Demodulation modes. When the timer is enabled, writes to these registers are buffered, and loading of the registers is delayed until a timer reload to `0001h` occurs; i.e., unless PWM0UE=1.

Table 72. Timer 0–2 PWM0 High Byte Registers (T*x*PWM0H)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	PWM0H							
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	T0PWM0H @ F04h, T1PWM0H @ F0Ch, T2PWM0H @ F14h							
Note: *x* references bits in the range [2:0].								

Table 73. Timer 0–2 PWM0 Low Byte Registers (T*x*PWM0L)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	PWM0L							
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	T0PWM0L @ F05h, T1PWM0L @ F0Dh, T2PWM0L @ F15h							
Note: *x* references bits in the range [2:0].								

Bit	**Description**
[7:0] PWM0H, PWM0L	**Pulse Width Modulator 0 High and Low Bytes** These two bytes, {PWM0H[7:0], PWM0L[7:0]}, form a 16-bit value that is compared to the current 16-bit timer count. When a match occurs, the PWM output changes state. The PWM output value is set by the TPOL bit in the Timer Control 1 Register (T*x*CTL1). The T*x*PWM0H and T*x*PWM0L registers also store the 16-bit captured timer value when operating in Capture, Capture/Compare and Demodulation modes.

10.1.4. Timer 0–2 PWM1 High and Low Byte Registers

The Timer 0–2 PWM1 High and Low Byte (T*x*PWM1H and T*x*PWM1L) registers, shown in Tables 74 and 75, store Capture values for Demodulation Mode.

Table 74. Timer 0–2 PWM1 High Byte Registers (T*x*PWM1H)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	PWM1H							
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	T0PWM1H @ F20h, T1PWM1H @ F24h, T2PWM1H @ F28h							
Note: *x* references bits in the range [2:0].								

Table 75. Timer 0–2 PWM1 Low Byte Registers (T*x*PWM1L)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	PWM1L							
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	T0PWM1L @ F21h, T1PWM1L @ F25h, T2PWM1L @ F29h							
Note: *x* references bits in the range [2:0].								

Bit	**Description**
[7:0] PWM1H, PWM1L	**Pulse Width Modulator 1 High and Low Bytes** These two bytes, {PWM1H[7:0], PWM1L[7:0]}, store the 16-bit captured timer value for Demodulation Mode.

10.1.5. Timer 0–2 Control Registers

This subsection describes the three timer control registers.

10.1.5.1. Timer 0–2 Control 0 Register

The Timer 0–2 Control 0 (T*x*CTL0) registers, shown in Table 76, together with the T*x*CTL1 Register, determine the timer operating mode. These registers also include a programmable PWM deadband delay, two bits to configure timer interrupt definition and a status bit to identify if the last timer interrupt is due to an input capture event.

Table 76. Timer 0–2 Control 0 Registers (T*x*CTL0)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	TMODE[3]	TICONFIG		Reserved	PWMD			INPCAP
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R
Address	T0CTL0 @ F06h, T1CTL0 @ F0Eh, T2CTL0 @ F16h							
Note: *x* references bits in the range [2:0].								

Bit	Description
[7] TMODE[3]	**Timer Mode High Bit** This bit, along with the TMODE[2:0] field in the T*x*CTL1 Register, determines the operating mode of the timer. This bit is the most significant bit of the timer mode selection value. To learn more, see the description of the Timer 0–2 Control 1 Registers section on page 140.
[6:5] TICONFIG	**Timer Interrupt Configuration** This field configures timer interrupt definition. This bit is a function of TMODE. **All TMODE selections except Triggered One-Shot and Dual Input Triggered One-Shot modes.** 0x: Timer Interrupt occurs on all Reload, Compare and Input Events available in the selected mode. 10: Timer Interrupt only on Input Capture/Deassertion Events available in the selected mode. 11: Timer Interrupt only on Reload/Compare Events available in the selected mode. **Triggered One-Shot Mode.** 00: All Reload Events and Input 0 Trigger Events while counting. 01: Only on timer Input 0 Trigger Events while counting. 10: Only on timer Input 1 Trigger Events while counting. 11: Only on Reload Events. **Dual Input Triggered One-Shot Mode** 00: All Reload and Trigger while counting Events. 01: Only on same input Trigger Events while counting. 10: Only on input Trigger Events while counting by the input that did not trigger counting. 11: Only on Reload Events.

Bit	Description (Continued)
[4]	Reserved.
[3:1] PWMD	**PWM Delay Value** This field is a programmable delay to control the number of timer clock cycles time delay before the Timer Output and the Timer Output Complement is forced to their active state. 000: No delay. 001: 2 cycles delay. 010: 4 cycles delay. 011: 8 cycles delay. 100: 16 cycles delay. 101: 32 cycles delay. 110: 64 cycles delay. 111: 128 cycles delay.
[0] INPCAP	**Input Capture Event** This bit should be ignored when TICONFIG=1x. 0: A reload occurred more recently than an Input Trigger, Timer Input Capture or Gate Deassertion Event. 1: An Input Trigger, Timer Input Capture or Gate Deassertion Event occurred more recently than reload.

10.1.5.2. Timer 0–2 Control 1 Registers

The Timer 0–2 Control 1 (T*x*CTL1) registers, shown in Table 77, enable and disable the timers, set the prescaler value, and determine the timer operating mode.

Table 77. Timer 0–2 Control 1 Registers (T*x*CTL1)

Bit	7	6	5	4	3	2	1	0
Field	TEN	TPOL	PRES			TMODE		
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	T0CTL1 @ F07h, T1CTL1 @ F0Fh, T2CTL1 @ F17h							
Note: *x* references bits in the range [2:0].								

Bit	Description
[7] TEN	**Timer Enable** 0=Timer is disabled. 1=Timer enabled to count.
[6] TPOL	**Timer Input/Output Polarity** Operation of this field is a function of the current operating modes of the timer. **One-Shot Mode** When the timer is disabled, the Timer Output signal is set to the value of this bit. When the timer is enabled, the Timer Output signal is complemented upon timer reload.

Bit	Description (Continued)
[6] TPOL (cont'd)	**Continuous Mode** When the timer is disabled, the Timer Output signal is set to the value of this bit. When the timer is enabled, the Timer Output signal is complemented upon timer reload. **Counter Mode** When the timer is disabled, the Timer Output signal is set to the value of this bit. When the timer is enabled, the Timer Output signal is complemented upon timer reload. 0: Count occurs on the rising edge of the Timer Input 0 signal. 1: Count occurs on the falling edge of the Timer Input 0 signal. **PWM Single Output Mode** 0: Timer Output is forced Low (0) when the timer is disabled. When enabled, the Timer Output is forced High (1) on PWM count match and forced Low (0) on Reload. 1: Timer Output is forced High (1) when the timer is disabled. When enabled, the Timer Output is forced Low (0) on PWM count match and forced High (1) on Reload. **Capture Mode** 0: Count is captured on the rising edge of the Timer Input 0 signal. 1: Count is captured on the falling edge of the Timer Input 0 signal. **Compare Mode** When the timer is disabled, the Timer Output signal is set to the value of this bit. When the timer is enabled, the Timer Output signal is complemented on timer reload. **Gated Mode** 0: Timer counts when the Timer Input 0 signal is High (1) and interrupts are generated on the falling edge of the Timer Input 0 signal. 1: Timer counts when the Timer Input 0 signal is Low (0) and interrupts are generated on the rising edge of the Timer Input 0 signal. **Capture/Compare Mode** 0: Counting is started on the first rising edge of the Timer Input 0 signal. The current count is captured on subsequent rising edges of the Timer Input 0 signal. 1: Counting is started on the first falling edge of the Timer Input 0 signal. The current count is captured on subsequent falling edges of the Timer Input 0 signal.

Bit	Description (Continued)
[6] TPOL (cont'd)	**PWM Dual Output Mode** 0: Timer Output is forced Low (0) and Timer Output Complement is forced High (1) when the timer is disabled. When enabled, the Timer Output is forced High (1) upon PWM count match and forced Low (0) upon Reload. When enabled, the Timer Output Complement is forced Low (0) upon PWM count match and forced High (1) upon Reload. The PWMD field in Timer Control 0 Register is a programmable delay to control the number of cycles time delay before the Timer Output and the Timer Output Complement is forced to High (1). 1: Timer Output is forced High (1) and Timer Output Complement is forced Low (0) when the timer is disabled. When enabled, the Timer Output is forced Low (0) upon PWM count match and forced High (1) upon Reload. When enabled, the Timer Output Complement is forced High (1) upon PWM count match and forced Low (0) upon Reload. The PWMD field in Timer Control 0 Register is a programmable delay to control the number of cycles time delay before the Timer Output and the Timer Output Complement is forced to Low (0). **Capture Restart Mode** 0: Count is captured on the rising edge of the Timer Input 0 signal. 1: Count is captured on the falling edge of the Timer Input 0 signal. **Comparator Counter Mode** When the timer is disabled, the Timer Output signal is set to the value of this bit. When the timer is enabled, the Timer Output signal is complemented upon timer reload. **Triggered One-Shot Mode and Dual Input Triggered One-Shot Mode** 0: Timer counting is triggered on the rising edge of the Timer Input 0 or Input 1 signal. 1: Timer counting is triggered on the falling edge of the Timer Input 0 or Input 1 signal. Use this setting only if OUTCTL[1:0] = 00. When the timer is disabled, the Timer Output signal is cleared to Low (0). **Demodulation Mode** This functionality applies only if TPOLHI bit in Timer Control 2 Register is 0. If TPOLHI bit is 1 then timer counting is triggered on any edge of the Timer Input 0 signal and the current count is captured on both edges. The current count is captured into PWM0 registers on rising edges and PWM1 registers on falling edges of the Timer Input 0 signal. 0: Timer counting is triggered on the rising edge of the Timer Input 0 signal. The current count is captured into PWM0 High and Low byte registers on subsequent rising edges of the Timer Input 0 signal. 1: Timer counting is triggered on the falling edge of the Timer Input 0 signal. The current count is captured into PWM1 High and Low byte registers on subsequent falling edges of the Timer Input 0 signal.

Bit	Description (Continued)
[5:3] PRES	**Prescale Value** The timer input clock is divided by PRES; PRES can be set from 0 to 7. The prescaler is reset each time the Timer is disabled. This insures proper clock division each time the Timer is restarted. 000=Divide by 1 001=Divide by 2 010=Divide by 4 011=Divide by 8 100=Divide by 16 101=Divide by 32 110=Divide by 64 111=Divide by 128
[2:0] TMODE[2:0]	**Timer Mode** This field, along with the TMODE[3] bit in the T*x*CTL0 Register, determines the operating mode of the timer. TMODE[3:0] selects among the following modes: 0000=One-Shot Mode. 0001=Continuous Mode. 0010=Counter Mode. 0011=PWM Single Output Mode. 0100=Capture Mode. 0101=Compare Mode. 0110=Gated Mode. 0111=Capture/Compare Mode. 1000=PWM Dual Output Mode. 1001=Capture Restart Mode. 1010=Dual Input Triggered One-Shot Mode. 1011=Triggered One-Shot Mode. 1100=Demodulation Mode.

10.1.5.3. Timer 0–2 Control 2 Registers

The Timer 0–2 Control 2 (T*x*CTL2) registers, shown in Table 78, allow selection of timer clock source and control of timer input polarity in Demodulation Mode.

Table 78. Timer 0–2 Control 2 Registers (T*x*CTL2)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	Reserved		PWM0UE	TPOLHI	OUTCTL		TCLKS	
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	T0CTL2 @ F22h, T1CTL2 @ F26h, T2CTL2 @ F2Ah							
Note: *x* references bits in the range [2:0].								

Bit	Description
[7:6]	**Reserved** These bits are reserved and must be programmed to 00.
[5] PWM0UE	**PWM0 Update Enable** This bit determines whether writes to the PWM0 High and Low Byte registers are buffered when TEN=1. Writes to these registers are not buffered when TEN=0, regardless of the value of this bit. 0: Writes to the Channel High and Low Byte registers are buffered when TEN=1 and only take affect on a timer reload to 0001h. 1: Writes to the Channel High and Low Byte registers are not buffered when TEN=1.
[4] TPOLHI	**Timer Input/Output Polarity High Bit** This bit determines if timer count is triggered and captured on both edges of the input signal. This applies only to Demodulation Mode. 0: Count is captured only on one edge in Demodulation Mode. In this case, edge polarity is determined by TPOL bit in the T*x*CTL1 Register. 1: Count is triggered on any edge and captured on both rising and falling edges of the Timer Input signal in Demodulation Mode.
[3]	**Reserved** This bit is reserved and must be programmed to 0.

Bit	Description (Continued)
[3:2] OUTCTL	**Timer Output Control** These bits determines timer output behavior and applies only to One-Shot, Triggered One-Shot, and Dual Input Triggered One-Shot modes. 00: Pulse at reload lasting one timer clock. 01: Pulse from start to reload. Use this setting only with TPOL = 0. **TMODE = 0000 (One-Shot)**10:Reserved. 11: Reserved. **TMODE = 1011 (Triggered One-Shot)**10:Pulse lasting one timer clock occurring when the Timer Input 0 triggers while counting. 11: Pulse lasting one timer clock occurring when the Timer Input 1 triggers while counting. **TMODE = 1010 (Dual Triggered One-Shot)**10:Pulse lasting one timer clock occurring when the same timer input that triggered counting (either Timer Input 0 or Timer Input 1) triggers while counting. 11: Pulse lasting one timer clock occurring when the timer input that did not trigger counting (either Timer Input 0 or Timer Input 1) triggers while counting.
[1:0] TCLKS	**Timer Clock Source** 00: System Clock. 01: Reserved. Defaults to System Clock. 10: Pclk. 11: WTO.

10.1.6. Timer 0–2 Status Registers

The Timer 0–2 Status (T*x*STAT) Register, shown in Table 79, indicate PWM capture/compare event occurrences, overrun errors, noise event occurrences and reload time-out status.

Table 79. Timer 0–2 Status Register (T*x*STAT)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	NEF	Reserved	PWM1EO	PWM0EO	RTOEF	Reserved	PWM1EF	PWM0EF
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	T0STAT @ F23h, T1STAT @ F27h, T2STAT @ F2Bh							

Bit	Description
[7] NEF	**Noise Event Flag** This status is applicable only if the Timer Noise Filter is enabled. The NEF bit will be asserted if digital noise is detected on the Timer Input 0 when the data is being sampled (center of bit time). If this bit is set, it does not mean that the timer input data is corrupted (though it can be in extreme cases), just that one or more Noise Filter data samples near the center of the bit time did not match the average data value.
[6]	**Reserved** This bit is reserved and must be programmed to 0.
[5:4] PWMxEO	**PWM x Event Overrun** This bit indicates that an overrun error has occurred. An overrun occurs when a new capture/compare event occurs before the previous PWMxEF bit is cleared. Clearing the associated PWMxEF bit in the T*x*STAT Register clears this bit. 0: No Overrun. 1: Capture/Compare Event Flag Overrun.
[3] RTOEF	**Reload Time-Out Event Flag** This flag is set if timer counts up to the reload value and is reset to 0001h. Software can use this bit to determine if a reload occurred prior to a capture. It can also determine if timer interrupt is due to a reload event. 0: No Reload Time-Out event occurred. 1: A Reload Time-Out event occurred.
[2]	**Reserved** This bit is reserved and must be programmed to 0.
[1:0] PWMxEF	**PWM x Event Flag** This bit indicates if a capture/compare event occurred for this PWM channel. Software can use this bit to determine the PWM channel responsible for generating the timer interrupt. This event flag is cleared by writing a 1 to the bit. These bits will be set when an event occurs independent of the setting of the timer interrupt enable bit. 0: No Capture/Compare Event occurred for this PWM channel. 1: A Capture/Compare Event occurred for this PWM channel.

10.1.7. Timer 0–2 Noise Filter Control Registers

The Timer 0–2 Noise Filter Control (T*x*NFC) registers, shown in Table 80, enable and disable the Timer Noise Filter and set noise filter control.

Table 80. Timer 0–2 Noise Filter Control Registers (T*x*NFC)

Bit	7	6	5	4	3	2	1	0

Table 80. Timer 0–2 Noise Filter Control Registers (T*x*NFC)

Field	NFCTL				NFCON	Reserved		
Reset	0	0	0	0	0	0	0	0
R/W	R/W				R/W	R	R	R
Address	T0NFC @ F2Ch, T1NFC @ F2Dh, T2NFC @ F2Eh							
Note: *x* references bits in the range [2:0].								

Bit	Description
[7:4] NFCTL	**Noise Filter Control** This field controls the delay and noise rejection characteristics of the Noise Filter. The wider the counter the more delay that is introduced by the filter and the wider the noise event that will be filtered. 0000: The Noise Filter is disabled. Received inputs bypass the filter 0001: 2-bit up/down counter 0010: 3-bit up/down counter 0011: 4-bit up/down counter 0100: 5-bit up/down counter 0101: 6-bit up/down counter 0110: 7-bit up/down counter 0111: 8-bit up/down counter 1000: 9-bit up/down counter 1001: 10-bit up/down counter 1010: 11-bit up/down counter 1011–1111: Reserved.
[3] NFCON	**Noise Filter Connection** 0: Noise Filter connects to Timer(x) and filters timer Input 0 (T*x*IN ORed with Event System Timer(x) Input 0). 1: Noise Filter is reassigned to Timer(x+1) and filters Event System Timer(x+1) Input 1. Timer(x) Input 0 effectively bypasses the Noise Filter. In the case of Timer 2, its Noise Filter connects to Timer 0 Input 1. With this selection, the Noise Filter is reassigned to the designated timer and uses its timer clock.
[2:0]	**Reserved** This bit is reserved and must be programmed to 0.

Chapter 11. Timer A

The F3224 Series products contain one Timer A timer that can be used for continuous timing, capture, or generation of a pulse-width modulated signal. Timer1's features include:

- 16-bit reload counter for continuous timing
- Capture capability with a rising or falling edge trigger
- PWM output generation
- Programmable prescaler with prescale values ranging from 1 to 2^16
- External input from Event System for capture signal. External input pin signal frequency is limited to a maximum of one-fourth the timer clock frequency
- Timer output to Event System

Operation in any mode with Pclk or WTO; operation in Active Mode and Halt Mode with Sysclk, External Clock Source, Pclk, or WTO

11.1. Timer A Architecture

Figure 17 shows the architecture of the Timer A.

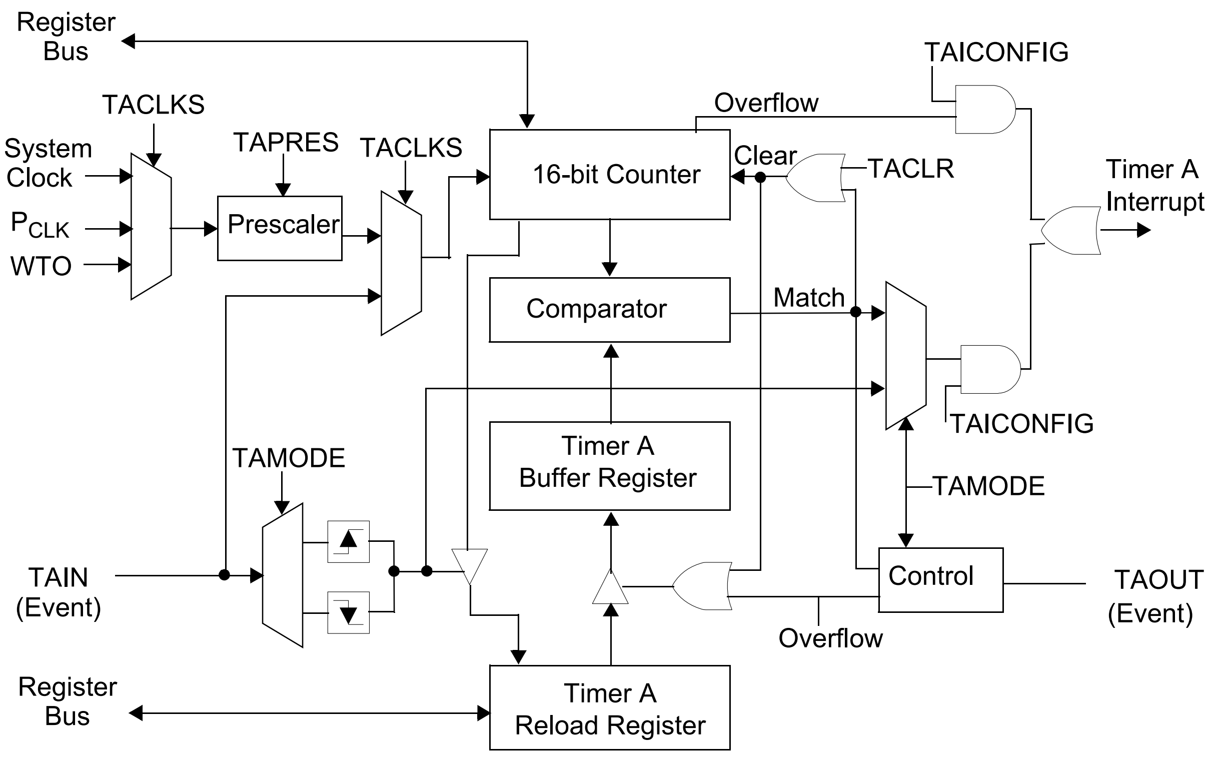


Figure 17. Timer A Block Diagram

11.2. Timer A Operation

Timer A is a 16-bit up-counter. Minimum time-out delay is set by loading the value `0001h` into the Timer A Reload High and Low Byte registers and setting the prescale value to 1. Maximum time-out delay is set by loading the value `0000h` into the Timer Reload High and Low Byte registers and setting the prescale value to 2^16. If the timer reaches `FFFFh`, the timer rolls over to `0000h` and continues counting.

11.2.1. Timer Clock Source and Prescaler

The Timer A clock source is selected using TACLKS and can come from either the System Clock, the External Clock Source (Event System), the peripheral clock (Pclk), or the Watch-dog Timer Oscillator (WTO).

All Timer A clock sources are available for Active or Halt Mode operation. For timer operation in Stop Mode, Pclk or the WTO must be selected as the timer clock source.

Caution: When the timer is operating on Pclk or the WTO, the timer clock is asynchronous to System Clock. To ensure error-free operation, disable the timer before modifying its operation (including changing the timer clock source).

When the timer uses Pclk or the WTO and the Timer is enabled, any read from TAH or TAL is not recommended, because results can be unpredictable. Disable the timer first, then read it. If Capture Mode is selected, any read from TARH, TARL must be done after capture interrupt occurs, or results can be unpredictable. TAINPCAP in the Timer A Control Register has the same characteristics as these registers. When the timer clock selection is System Clock or External Clock Source, registers can be written/read at any time.

Timer A has a clock prescaler with an input clock division range from 1 to 2^16, selectable using TAPRES. It is recommended that the timer is disabled (TAEN=0) while changing Timer A clock source selection and while changing the prescaler selection.

11.2.2. Clearing and Enabling Timer A

Timer A is enabled by setting TAEN in the TACTL Register. Clearing Timer A is accomplished by setting TACLR in the TACTL Register which will reset the count value to zero and load the Timer A Reload Register contents into the Timer A Buffer Register. After clearing Timer A, TACLR is automatically cleared by hardware.

11.2.3. Low-Power Modes

Timer A can operate in both Halt Mode and Stop Mode. This section discusses Timer A operation each of these low-power modes.

11.2.3.1. Operation in Halt Mode

When the eZ8 CPU enters Halt Mode, the timer will continue to operate if enabled. To minimize current in Halt Mode, the timer can be disabled by clearing the TAEN control bit.

11.2.3.2. Operation in Stop Mode

When the eZ8 CPU enters Stop Mode, the timer continues to operate if enabled and Pclk or the WTO is selected as the timer clock. In Stop Mode, the timer interrupt (if enabled) automatically initiates a Stop-Mode Recovery and generates an interrupt request. In the

Reset Status Register, the stop bit is set to 1. Also, the Timer A interrupt request bit in Interrupt Request 3 Register is set. Following completion of the Stop-Mode Recovery, if interrupts are enabled, the CPU responds to the interrupt request by fetching the Timer A interrupt vector.

If System Clock is chosen as the timer clock, the timer ceases to operate as System Clock is disabled in Stop Mode. In this case the registers are not reset and operation will resume after Stop-Mode Recovery occurs.

11.2.3.3. Power Reduction During Operation

Clearing TAEN will inhibit clocking of the timer. The CPU can still read/write registers when TAEN is cleared.

11.2.4. Timer A Operating Modes

The timer can be configured to operate in the following modes, each of which is described in this section where indicated in Table 81.

Table 81. Timer A Operating Modes

Mode	Page #
Continuous Mode	151
Capture Mode	151
Pulse Width Modulation (PWM) Mode	152

11.2.4.1. Continuous Mode

In Continuous Mode (TAMODE=00), the timer counts timer clocks up to the 16-bit reload value stored in the Timer A Reload High and Low Byte registers. Upon reaching the reload value, the timer generates a match interrupt, the count value in the Timer A High and Low Byte registers is reset to `0000h` and counting resumes. Also, the Timer Output changes state (from Low to High or High to Low) on timer reload. The Timer Output can be connected to the Event System. The Timer Output is reset to Low level upon System Reset and whenever the timer is disabled.

11.2.4.2. Capture Mode

In Capture Mode Rising Edge (TAMODE=01) and Capture Mode Falling Edge (TAMODE=10), the current timer count value is recorded when the selected Timer Input (Event System) transition occurs. This capture count value is written to the Timer A Reload High and Low Byte registers. When the capture event occurs, a match interrupt is generated and the timer continues counting up to the 16-bit maximum value of “FFFFh” and then rolls over, generates an overflow interrupt and continues counting.

If TAINPCAP=1, a capture event is the most recent event that occurred, whereas, if TAIN-PCAP=0, a timer overflow is the most recent event that occurred.

By reading the captured data value in the Timer A Reload High and Low Byte registers, and assuming a specific value for the timer A clock frequency, the pulse width (duration) of the Timer Input signal can be calculated.

For Capture Mode, it is necessary to configure the Timer Input as an Event System destination. The Timer Input frequency must not exceed 1/4 the timer clock frequency.

The Timer Output is held at a Low level during Capture Mode.

11.2.4.3. Pulse Width Modulation (PWM) Mode

In PWM Mode (TAMODE=11), the timer outputs a Pulse Width Modulator output signal to the Event System. The timer first counts up to the 16-bit PWM match value stored in the Timer A Reload High and Low Byte registers while holding the Timer Output at a High level. When the timer count value matches the PWM match value, a match interrupt is generated and the Timer Output becomes a Low level. The timer continues counting until it reaches the 16-bit maximum value of “FFFFh”, then rolls over, generates an overflow interrupt, sets the Timer Output to a High level, and then continues to increment from "0000h".

The Timer Output is held to High level while the counter value is less than (<) the PWM reload value. The Timer Output is held to Low level while the counter value is greater than or equal to (≥) the PWM reload value. One PWM period is equal to Timer A clock period x 65536.

If TAINPCAP=1, a match event is the most recent event that occurred, whereas, if TAINPCAP=0, a timer overflow is the most recent event that occurred.

For PWM Mode, it is necessary to configure the Timer A Output as an Event System source.

11.3. Timer A Register Definitions

The Timer A registers are described in the following tables.

11.3.1. Timer A High and Low Byte Registers

The Timer A High and Low Byte (TAH and TAL) registers, shown in Tables 82 and 83, contain the current 16-bit timer count value. When the timer is enabled, a read from TAH causes the value in TAL to be stored in a temporary holding register. A read from TAL always returns this temporary register when the timers are enabled. When the timer is disabled, reading from the TAL reads the register directly.

Zilog does not recommend writing to the Timer High and Low Byte registers while the timer is enabled. There are no temporary holding registers available for write operations; therefore, simultaneous 16-bit writes are not possible. If either the Timer High or Low Byte registers are written during counting, the 8-bit written value is placed in the counter

(High or Low Byte) at the next clock edge. The counter continues counting from the new value.

Table 82. Timer A High Byte Register (TAH)

Bit	7	6	5	4	3	2	1	0
Field	TAH							
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F30h							

Table 83. Timer A Low Byte Register (TAL)

Bit	7	6	5	4	3	2	1	0
Field	TAL							
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F31h							

Bit	Description
[7:0] TAH, TAL	**Timer A High and Low Bytes** These 2 bytes, {TAH[7:0], TAL[7:0]}, contain the current 16-bit timer count value.

11.3.2. Timer A Reload High and Low Byte Registers

The Timer A Reload High and Low Byte (TARH, TARL) registers, shown in Tables 84 and 85, store a 16-bit reload value, {TARH[7:0], TARL[7:0]}. Values written to the Timer A Reload High Byte Register are stored and subsequently loaded into the Timer A Buffer Register upon Timer A overflow or upon Timer A counter clearing due to match or setting of the TACLR bit. This operation provides simultaneous update of the two comparator match value bytes.

In Continuous Mode, the Timer A Reload High and Low Byte registers store the 16-bit compare value.

In Capture Mode, the Timer A Reload High and Low Byte registers store the 16-bit capture count value.

In PWM Mode, the Timer A Reload High and Low Byte registers store the 16-bit PWM match value.

Table 84. Timer A Reload High Byte Register (TARH)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	TARH							
Reset	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F32h							

Table 85. Timer A Reload Low Byte Register (TARL)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	TARL							
Reset	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F33h							

Bit	**Description**
[7:0] TARH, TARL	**Timer A Reload High and Low Bytes** TARH: The Timer A Reload High Byte is the most significant byte, or bits [15:8] of the 16-bit Timer A reload value. TARL: The Timer A Reload Low Byte is the least significant byte, or bits [7:0] of the 16-bit Timer A reload value.

11.3.3. Timer A Control Register

The Timer A Control (TACTL) Register, shown in Table 86, determines the timer operating mode. This register also includes clear, start and interrupt control.

Table 86. Timer A Control Register (TACTL)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	TAMODE		TACLR	TAEN	TAICONFIG		Reserved	TAINPCAP
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	R
Address	F34h							

Bit	Description
[7:6] TAMODE	**Timer A Mode** These bits determines the operating mode of the timer. 00: Continuous Mode. 01: Capture Mode, Rising Edge. As the counter is running, an overflow can occur. 10: Capture Mode, Falling Edge. As the counter is running, an overflow can occur. 11: PWM Mode. As the counter is running, an overflow can occur.
[5] TACLR	**Timer A Clear** 0: No effect. 1: Clear Timer A. After clearing Timer A, this bit is automatically cleared by hardware.
[4] TAEN	**Timer A Enable** 0: Disable Timer A. 1: Enable Timer A.
[3:2] TAICONFIG	**Timer A Interrupt Configuration** 00: Reserved. 01: Only on Overflow Events. 10: Only on Match/Capture Events. 11: All Overflow Events and Match/Capture Events.
[1]	Reserved
[0] TAINPCAP	**Timer A Input Capture Event** This bit is useful when TAICONFIG=11. 0: An Overflow Event occurred more recently than a Match/Capture Event. 1: A Match/Capture Event occurred more recently than an Overflow Event.

11.3.4. Timer A Prescale Register

The Timer A Prescale (TAPS) register, shown in Table 87, determines the timer clock source selection and prescaler value.

Table 87. Timer A Prescale Register (TAPS)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	TACLKS		Reserved	TAPRES				
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F35h							

Bit	Description
[7:6] TACLKS	**Timer A Clock Source** 00: System Clock. 01: External Clock Source (Event System). For this selection, System Clock must be running and the External Clock Source input frequency must not exceed 1/4 the System Clock frequency. 10: Pclk. 11: WTO.
[5]	Reserved
[4:0] TAPRES	**Timer A Prescale Value** The Timer A Input Clock Source is divided by 2^{TAPRES}; TAPRES can be set from 0 to 16. The prescaler is reset each time the Timer A is disabled to ensure proper clock division each time the Timer A is restarted. 00000: Timer A Input Clock Source / 2^0. 00001: Timer A Input Clock Source / 2^1. 00010: Timer A Input Clock Source / 2^2. 00011: Timer A Input Clock Source / 2^3. 00100: Timer A Input Clock Source / 2^4. 00101: Timer A Input Clock Source / 2^5. 00110: Timer A Input Clock Source / 2^6. 00111: Timer A Input Clock Source / 2^7. 01000: Timer A Input Clock Source / 2^8. 01001: Timer A Input Clock Source / 2^9. 01010: Timer A Input Clock Source / 2^10. 01011: Timer A Input Clock Source / 2^11. 01100: Timer A Input Clock Source / 2^12. 01101: Timer A Input Clock Source / 2^13. 01110: Timer A Input Clock Source / 2^14. 01111: Timer A Input Clock Source / 2^15. 10000: Timer A Input Clock Source / 2^16. All other settings are reserved.

Chapter 12. Multi-Channel Timer

The Multi-Channel timer has a 16-bit up/down counter and a 4-channel Capture/Compare/PWM channel array. This timer provides multiple synchronous Capture/Compare/PWM channels based on a single timer. The Multi-Channel Timer features include:

- 16-bit up/down timer counter with programmable prescale
- Selectable clock source (system clock or external input pin)
- Count Modulo and Count up/down counter modes
- Four independent capture/compare channels which reference the common timer
- Channel modes:
  - One-Shot Compare Mode
  - Continuous Compare Mode
  - PWM Output Mode
  - Capture Mode
- Event System and external input pin for timer input

12.1. Architecture

Figure 18 shows the Multi-Channel Timer architecture.

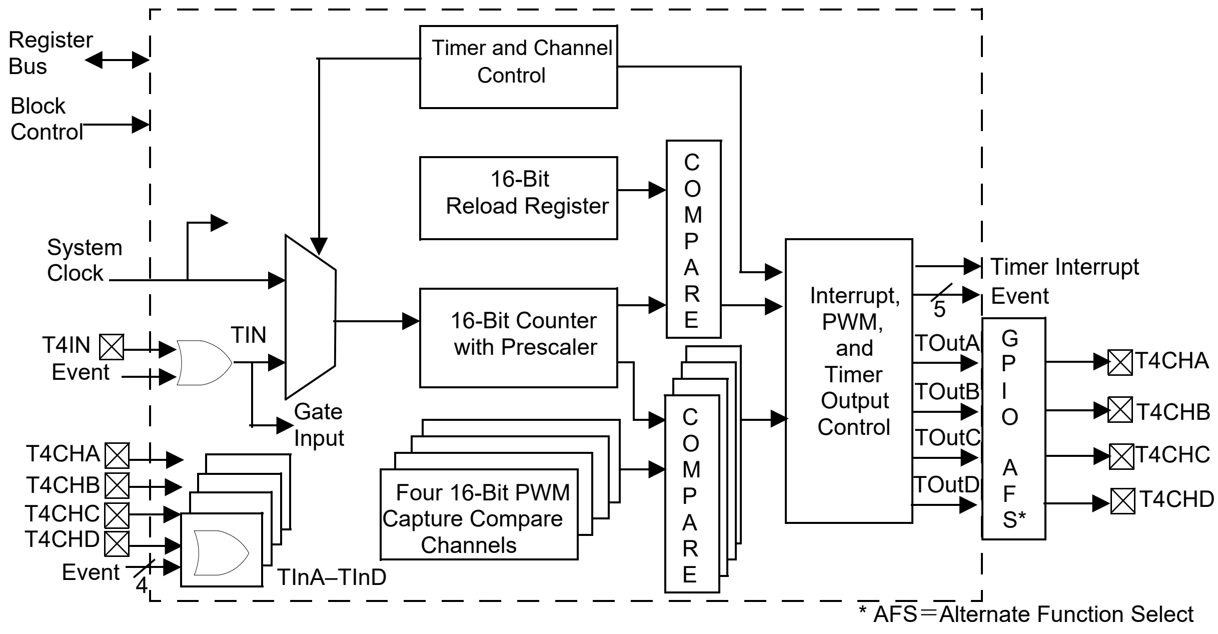


Figure 18. Multi-Channel Timer Block Diagram

12.2. Timer Operation

12.2.1. Multi-Channel Timer Counter

The Multi-Channel Timer is based around a 16-bit up/down counter. The counter, depending on the timer mode, counts up or down with each rising edge of the clock signal. Timer Counter registers MCTH and MCTL can be read/written by software.

12.2.2. Inputs and Outputs

Each GPIO alternate function (T4CHA–T4CHD and T4IN) is logically ORed with a corresponding Event System signal to form an input to the Multi-Channel Timer. Typically, only one of these two signals, either the GPIO input or Event System input, is configured to be active at a given time. Any inactive input is held at logic 0. To learn more about selecting a GPIO input using GPIO alternate function registers, refer to the General-Purpose Input/Output chapter on page 46. For information regarding selecting an Event System input, refer to the Event System chapter on page 287.

Multi-Channel Timer outputs can drive a GPIO and/or be a source to the Event System. To learn more about selecting a GPIO as a Multi-Channel Timer output using GPIO alternate function registers, refer to the General-Purpose Input/Output chapter on page 46. For information regarding selecting a Multi-Channel Timer output as an Event System source, refer to the Event System chapter on page 287.

12.2.3. Clock Source

The Multi-Channel Timer clock source can come from either the System Clock, the System Cock gated by the TIN input, or the TIN input pin operating as a clock input. The TCLKS field in the MCTCTL0 Register selects the timer clock source. When using the TIN input, the associated GPIO pin or Event System channel must be configured as an input to the timer. The TIN frequency cannot exceed one-fourth the system clock frequency.

12.2.4. Multi-Channel Timer Clock Prescaler

The prescaler allows the system clock signal to be decreased by factors of 1, 2, 4, 8, 16, 32, 64, or 128. The PRES[2:0] bit field in the MCTCTL1 Register controls prescaler operation. The PRES field is buffered so that the prescale value is updated only on a MCT end-of-cycle count. The prescaler has no effect when the TIN input is selected as the clock source.

12.2.5. Multi-Channel Timer Start

The Multi-Channel Timer starts counting when TEN bit in MCTCTL1 Register is set and the clock source is active. Timer counting can be stopped without disabling the timer by setting the Reload Register to 0. The timer will then stop when the counter next reaches 0. Writing a nonzero value to the Reload Register restarts the timer counting.

12.2.6. Multi-Channel Timer Mode Control

The Multi-Channel Timer supports two modes of operation: Count Modulo and Count up/down. The timer operating mode is selected with the TMODE[1:0] field in the MCTCTL1 Register. The timer modes are described below in Table 88.

Table 88. Timer Count Modes

TMODE	Timer Mode	Description
00	Count Modulo	Timer counts up to Reload Register value. Then it is reset to 0000h and counting resumes.
01	Reserved	
10	Count Up/Down	Timer counts up to Reload and then counts down to 0000h. The Count up/down cycle continues.
11	Reserved	

12.2.7. Count Modulo Mode

In the Count Modulo Mode, the Timer counts up to the Reload Register value (max value = `FFFFh`). Then it is reset to `0000h` and counting resumes. As shown in Figure 19, the counting cycle continues with Reload + 1 as the period. A timer count interrupt request is generated when the timer count resets from Reload to `0000h`. If Count Modulo is selected when the timer count is greater than Reload, the timer immediately restarts counting from zero.

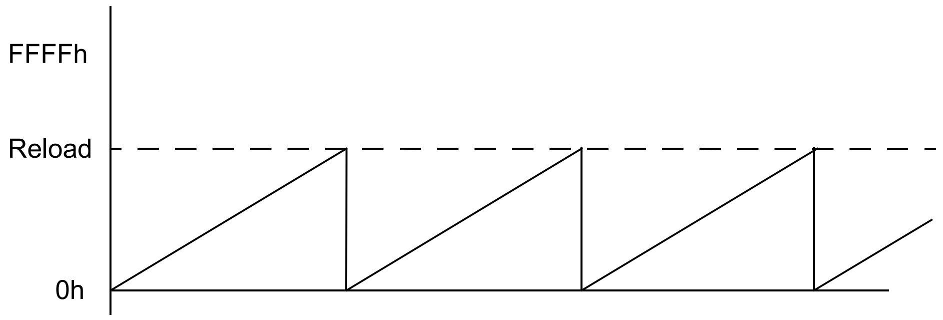


Figure 19. Count Modulo Mode

12.2.8. Count Up/Down Mode

In the Count Up/Down Mode, the timer counts up to the Reload Register value and then counts down to `0000h`. As shown in Figure 20, the counting cycle continues with twice the reload value as the period. A timer count interrupt is generated when the timer count decrements to zero.

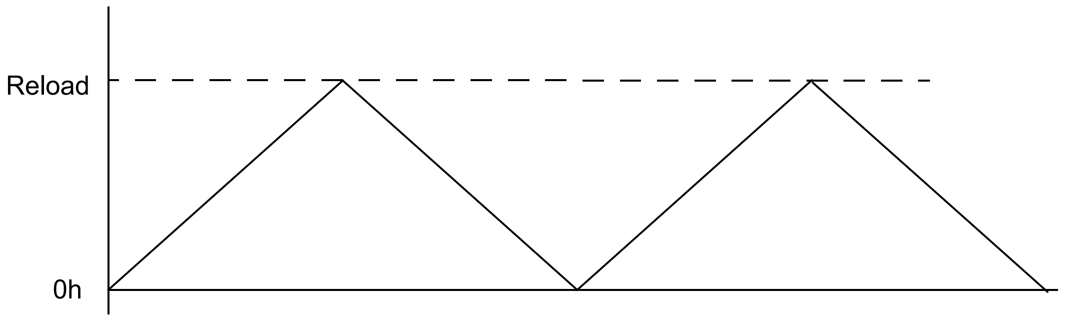


Figure 20. Count Up/Down Mode

12.3. Capture/Compare Channel Operation

The Multi-Channel timer supports four Capture/Compare channels: CHA, CHB, CHC, and CHD. Each channel has the following features:

- A 16-bit Capture/Compare Register (MCTCHyH and MCTCHyL registers) used to capture input event times or to generate time intervals. Any user software update of the Capture/Compare Register value when the timer is running takes effect only at the end of the counting cycle, not immediately. The end of the counting cycle is when the counter transitions from the reload value to 0 (Count Modulo Mode) or from 1 to 0 (Count Up/Down Mode).
- A dedicated bidirectional GPIO pin (T4CHA, B, C, or D) and Event System input/output that can be configured for the input capture function or to generate an output compare match or one-shot pulse.

Each channel is configured to operate in either One-Shot Compare, Continuous Compare, PWM Output, or Capture Mode.

12.3.1. One-Shot Compare Operation

In One-Shot Compare operation, a channel interrupt is generated when the channel compare value matches the timer count. The channel event flag, CHyEF, is set in the Channel Status 1 Register (MCTCHS1) to identify the responsible channel. Then the channel is automatically disabled. The timer continues counting according to the programmed mode. The channel output (TOutA, B, C, or D) changes state for one system clock cycle (from Low to High then back to Low or High to Low then back to High as determined by the CHPOL bit) on match.

12.3.2. Continuous Compare Operation

In Continuous Compare operation, a channel interrupt is generated when the channel compare value matches the timer count. The channel event flag (CHyEF) is set in the Channel Status1 Register (MCTCHS1) and the channel remains enabled. The timer continues counting according to the programmed mode. The channel output (TOutA, B, C, or D) changes state (from Low to High then back to Low, or High to Low then back to High as determined by the CHPOL bit) on match. For proper operation, configure the CHPOL bit prior to setting the CHEN bit.

12.3.3. PWM Output Operation

In PWM Output operation, the timer generates a PWM output signal on the channel output (TOutA, B, C, or D). The channel output toggles whenever the timer count matches the channel compare value (defined in the MCTCHyH and MCTCHyL) registers. In addition, a channel interrupt is generated and the channel event flag is set in the status register. The timer continues counting according to its programmed mode.

The channel output signal begins with the output value=CHPOL and then transitions to $\overline{\text{CHPOL}}$ when timer value matches the PWM value. If the timer mode is Count Modulo Mode, the channel output signal returns to output=CHPOL when timer reaches the reload value and is reset. If the timer mode is Count Up/Down Mode, channel output signal returns to output=CHPOL when the timer count matches the PWM value again (when counting down). For proper operation, configure the CHPOL bit prior to setting the CHEN bit.

12.3.4. Capture Operation

In Capture operation, the current timer count is recorded when the selected transition occurs on TInA, B, C or D. The Capture count value is written to the Channel High and Low Byte registers. In addition, a channel interrupt is generated and the channel event flag (CHyEF) is set in the Channel Status Register. The CHPOL bit in the Channel Control Register determines if the Capture occurs on a rising edge or a falling edge of the Channel Input signal. The timer continues counting according to the programmed mode.

12.4. Multi-Channel Timer Interrupts

The Multi-Channel Timer provides a single interrupt which has five possible sources. These sources are the internal timer and the four timer channels.

12.4.1. Timer Interrupt

If enabled by TCIEN bit of the MCTCTL0 Register, the timer interrupt will be generated when the timer completes a count cycle. This occurs during transition from counter=reload register value to counter=0 in Count Modulo Mode, and occurs during transition from counter=1 to counter=0 in Count Up/Down Mode.

12.4.2. Channel Interrupts

If enabled by the CHIEN bit of the MCTCHyCTL Register, a channel interrupt is generated when the channel compare value matches the timer count while in one of the following channel modes: One-Shot Compare, Continuous Compare, or PWM Output.

In Capture operation, a channel interrupt is generated whenever there is a successful Capture Event on the Timer Channel.

12.5. Low-Power Modes

12.5.1. Operation in Halt Mode

When the eZ8 CPU is operating in Halt Mode, the Multi-Channel Timer will continue to operate if enabled. To minimize current in Halt Mode, the Multi-Channel Timer must be disabled by clearing the TEN control bit.

12.5.2. Operation in Stop Mode

When the eZ8 CPU is operating in Stop Mode, the Multi-Channel Timer ceases to operate as the System Clock is stopped. The registers are not reset and operation will resume after Stop-Mode Recovery occurs.

12.5.3. Power Reduction During Operation

Deassertion of the TEN bit will inhibit clocking of the entire Multi-Channel Timer block. Deassertion of the CHEN bit of individual channels will inhibit clocking of channel specific logic to minimize power consumption of unused channels. The CPU can still read/ write registers when the enable bit(s) are deasserted.

12.6. Multi-Channel Timer Application Examples

12.6.1. PWM Programmable Deadband Generation

The Count Up/Down Mode supports motor control applications that require dead time between output signals. Figure 21 shows dead-time generation between two channels operating in Count Up/Down Mode.

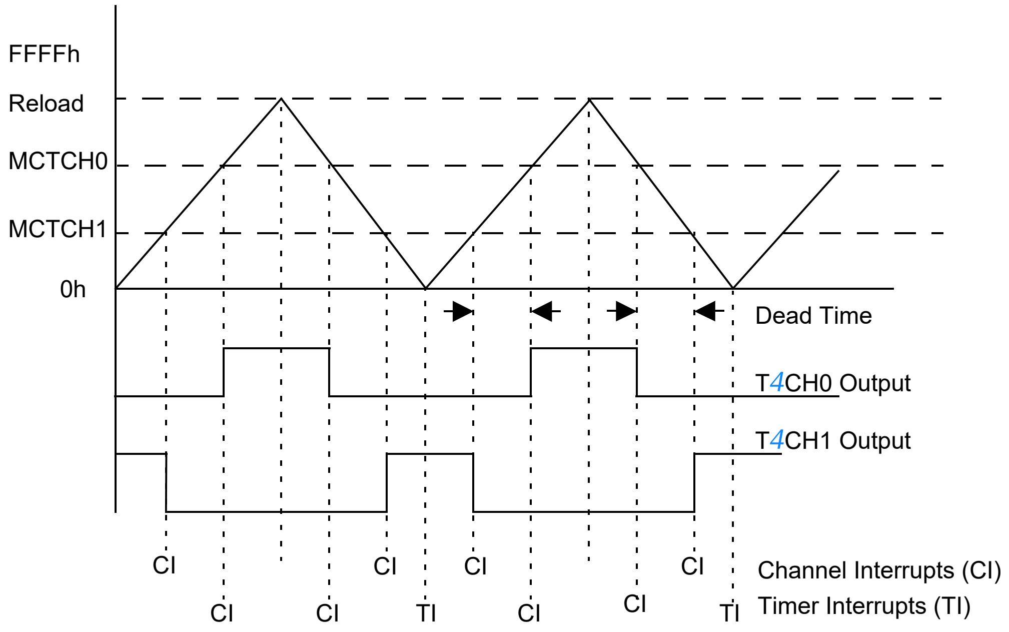


Figure 21. Count Up/Down Mode with PWM Channel Outputs and Deadband

12.6.2. Multiple Timer Intervals Generation

Figure 22 shows generation of two constant time intervals t0 and t1. The timer is in Count Modulo Mode with reload=`FFFFh`. Channels 0 and 1 are set up for Continuous Compare operation. After every channel compare interrupt, the channel Capture/Compare registers are updated in the interrupt service routine by adding a constant equal to the time interval required. This operation requires that the Channel Update Enable (CHUE) bit must be set in channels 0 and 1 so that writes to the Capture/Compare registers take affect immediately.

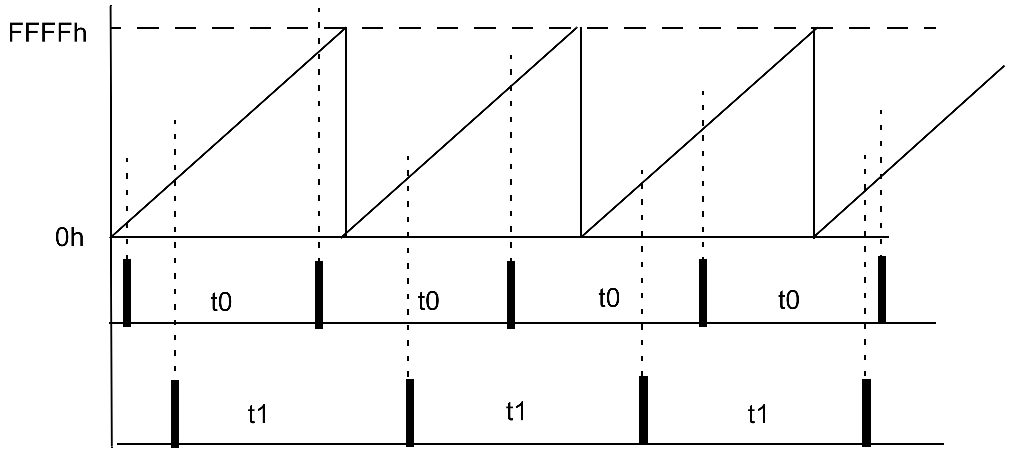


Figure 22. Count Max Mode with Channel Compare

12.7. Multi-Channel Timer Control Register Definitions

12.7.1. Multi-Channel Timer Address Map

Table 89 defines the byte address offsets for the Multi-Channel Timer registers. To save address space, a subaddress is used for the Timer Control 0, Timer Control 1, Channel Status 0, Channel Status 1, Channel-y Control, Channel-y High- and Low-Byte registers. Only the Timer High- and Low-Byte registers and the Reload High- and Low-Byte registers can be directly accessed.

While writing to a subregister, first write its subaddress to the Timer Subaddress Register, then write data to Subregister 0, Subregister 1, or Subregister 2. Reads function the same as writes.

Table 89. Multi-Channel Timer Address Map

Address/ Subaddress	Register/ Subregister Name
Direct Access Register	
FA0	Timer (Counter) High
FA1	Timer (Counter) Low
FA2	Timer Reload High
FA3	Timer Reload Low
FA4	Timer Subaddress
FA5	Subregister 0

Table 89. Multi-Channel Timer Address Map (Continued)

Address/ Subaddress	Register/ Subregister Name
FA6	Subregister 1
FA7	Subregister 2
Subregister 0	
0	Timer Control 0
1	Channel Status 0
2	Channel A Capture/Compare High
3	Channel B Capture/Compare High
4	Channel C Capture/Compare High
5	Channel D Capture/Compare High
Subregister 1	
0	Timer Control 1
1	Channel Status 1
2	Channel A Capture/Compare Low
3	Channel B Capture/Compare Low
4	Channel C Capture/Compare Low
5	Channel D Capture/Compare High
Subregister 2	
0	Reserved
1	Reserved
2	Channel A Control
3	Channel B Control
4	Channel C Control
5	Channel D Control

12.7.2. Multi-Channel Timer High and Low Byte Registers

The High and Low Byte (MCTH and MCTL) registers, shown in Tables 90 and 91, contain the current 16-bit MCT count value. Zilog does not recommend writing to the MCT High and Low Byte registers while the MCT is enabled. If either or both of the MCT High or Low Byte registers are written to during counting, the 8-bit written value is placed in the counter (High and/or Low byte) at the next system clock edge. The counter continues counting from the new value.

When MCT is enabled, a read from MCTH causes the value in MCTL to be stored in a temporary holding register. A read from MCTL returns this temporary register when MCT is enabled. When MCT is disabled, reads from MCTL read the register directory. The MCT High and Low Byte registers are not reset when TEN=0.

Table 90. MCT High Byte Register (MCTH)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	MCTH							
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	FA0h							

Table 91. MCT Low Byte Register (MCTL)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	MCTL							
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	FA1h							

Bit	**Description**
[7:0] MCTH, MCTL	**MCT High and Low Bytes** These 2 bytes, {MCTH[7:0], MCTL[7:0]}, contain the current 16-bit MCT count value.

12.7.3. MCT Reload High and Low Byte Registers

The MCT Reload High and Low Byte (MCTRH and MCTRL) registers, shown in Tables 92 and 93, store a 16-bit reload value, {MCTRH[7:0], MCTRL[7:0]}. When TEN=0, writes to this address update the register on the next clock cycle. When TEN=1, writes to this register are buffered and transferred into the register when the counter reaches the end of the count cycle.

$$\text{Modulo Mode Period} = \frac{\text{Prescale Value} \times (\text{Reload Value} + 1)}{f_{\text{MCTclk}}}$$

$$\text{Up/Down Mode Period} = \frac{2 \times \text{Prescale Value} \times \text{Reload Value}}{f_{\text{MCTclk}}}$$

A value written to the MCTRH is stored in a temporary holding register. When a write to the MCTRL occurs, the temporary holding register value is written to the MCTRH. This operation allows simultaneous updates of the 16-bit MCT reload value.

Table 92. MCT Reload High Byte Register (MCTRH)

Bit	7	6	5	4	3	2	1	0
Field	MCTRH							
RESET	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	FA2h							

Table 93. MCT Reload Low Byte Register (MCTRL)

Bit	7	6	5	4	3	2	1	0
Field	MCTRL							
RESET	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	FA3h							

Bit	Description
[7:0] MCTRH, MCTRL	**MCT Reload Register High and Low** These two bytes form the 16-bit reload value, {MCTRH[7:0], MCTRL[7:0]}. This value sets the MCT period in the Modulo and Up/Down Count modes.

12.7.4. MCT Subaddress Register

The MCT Subaddress Register, shown in Table 94, stores 3-bit subaddresses for subregisters. These three bits are from MCTSA[2:0], all other bits are reserved. When accessing a subregister (writing or reading), first write MCTSA with the subregister address, then access the subregister by writing or reading subregisters 0, 1, or 2.

Table 94. MCT Subaddress Register (MCTSA)

Bit	7	6	5	4	3	2	1	0
Field	Reserved					MCTSA		
RESET	0	0	0	0	0	0	0	0
R/W	R	R	R	R	R	R/W	R/W	R/W
Address	FA4h							

12.7.5. MCT Subregister x (0, 1, or 2)

The MCT subregisters 0, 1, or 2, shown in Table 95, store an 8-bit data write to a subregister or an 8-bit data read from a subregister. The MCT Subaddress Register selects the subregister to be written to or read from.

Table 95. MCT Subregister *x* (MCTSR*x*)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	MCTSR*x*							
RESET	X	X	X	X	X	X	X	X
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	MCTSR0 @ FA5h, MCTSR1 @ FFA6h, MCTSR2 @ FFA7h							
Note: *x* references bits in the range [2:0].								

Bit	**Description**
[7:0] MCTSR*x*	

12.7.6. Multi-Channel Timer Control 0 and Control 1 Registers

The Multi-Channel Timer Control 0 and 1 registers (MCTCTL0, MCTCTL1), shown in Tables 96 and 97, control multi-channel timer operation. Writes to the PRES field of the MCTCTL1 Register are buffered when TEN=1, and will not take effect until the next end-of-cycle count occurs.

Table 96. Multi-Channel Timer Control 0 Register (MCTCTL0)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	TCTST	CHST	TCIEN	Reserved		TCLKS		
RESET	0	0	0	0	0	0	0	0
R/W	R/W1C	R	R/W	R	R	R/W	R/W	R/W
Address	00h in Subaddress Register, accessible through Subregister 0							

Bit	**Description**
[7] TCTST	**Timer Count Status** This bit indicates if a timer count cycle is complete and is cleared by writing 1 to the bit and is cleared when TEN=0. 0: Timer count cycle is not complete. 1: Timer count cycle is complete.

Bit	Description (Continued)
[6] CHST	**Channel Status** This bit indicates if a channel Capture/Compare event occurred. This bit is the logical OR of the CHyEF bits in the MCTCHS1 Register. This bit is cleared when TEN=0. 0: No channel capture/compare event has occurred. 1: A channel capture/compare event has occurred. One or more of the CHDEF, CHCEF, CHBEF, and CHAEF bits in the MCTCHS1 Register are set.
[5] TCIEN	**Timer Count Interrupt Enable** This bit enables generation of timer count interrupt. A timer count interrupt is generated whenever the timer completes a count cycle: counting up to Reload Register value or counting down to zero depending on whether the time r mode is Count Modulo or Count up/down. 0: Timer Count Interrupt is disabled. 1: Timer Count Interrupt is enabled.
[4:3]	**Reserved** These bits are reserved and must be programmed to 00.
[2:0] TCLKS	**Timer Clock Source** 000: System Clock (prescaling enabled). 001: Reserved. 010: System Clock gated by active High Timer Input signal (Prescaling enabled). 011: System Clock gated by active Low Timer Input signal (Prescaling enabled). 100: Timer input rising edge (Prescaler disabled). 101: Timer input falling edge (Prescaler disabled). 110: Reserved. 111: Reserved.

Note: The input frequency of the timer input signal must not exceed one-fourth of the system clock frequency.

Table 97. Multi-Channel Timer Control 1 Register (MCTCTL1)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	TEN	Reserved	PRES			Reserved	TMODE	
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R	R/W	R/W	R/W	R	R/W	R/W
Address	00h in Subaddress Register, accessible through Subregister 1							

Bit	**Description**
[7] TEN	**Timer Enable** 0: Timer is disabled and the counter is reset. 1: Timer is enabled to count.
[6]	**Reserved** This bit is reserved and must be programmed to 0.
[5:3] PRES	**Prescale Value** The system clock is divided by the value selected in PRES. The prescaling operation is not applied when the alternate function input pin is selected as the timer clock source. 000: Divide by 1. 001: Divide by 2. 010: Divide by 4. 011: Divide by 8. 100: Divide by 16. 101: Divide by 32. 110: Divide by 64. 111: Divide by 128.
[2]	**Reserved** This bit is reserved and must be programmed to 0.
[1:0] TMODE	**Timer Mode** 00: Count Modulo: the timer counts up to the reload register value, then is reset to 0000h, and counting up resumes. 01: Reserved. 10: Count up/down: the timer counts up to the reload register value, then counts down to 0000h. The count up and count down cycles continue. 11: Reserved.

12.7.7. Multi-Channel Timer Channel Status 0 and Status 1 Registers

The Multi-Channel Timer Channel Status 0 and Status 1 Registers (MCTCHS0, MCTCHS1), shown in Tables 98 and 99, indicate channel overrun and channel capture/compare events.

Table 98. Multi-Channel Timer Channel Status 0 Register (MCTCHS0)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	Reserved				CHDEO	CHCEO	CHBEO	CHAEO
RESET	0	0	0	0	0	0	0	0
R/W	R	R	R	R	R	R	R	R
Address	01h in Subaddress Register, accessible through Subregister 0							

Bit	**Description**
[7:4]	**Reserved** These bits are reserved and must be programmed to 0000.
[3:0] CHyEO	**Channel y Event Flag Overrun** This bit indicates that an overrun error has occurred. An overrun occurs when a new Capture/Compare event occurs before the previous CHyEF bit is cleared. Clearing the associated CHyEF bit in the MCTCHS1 Register clears this bit. This bit is also cleared when TEN=0 (TEN is the MSB of MCTCTL1). 0: No Overrun. 1: Capture/Compare Event Flag Overrun.

Note: *y* = A, B, C, D.

Table 99. Multi-Channel Timer Channel Status 1 Register (MCTCHS1)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	Reserved				CHDEF	CHCEF	CHBEF	CHAEF
RESET	0	0	0	0	0	0	0	0
R/W	R	R	R	R	R/W1C	R/W1C	R/W1C	R/W1C
Address	01h in Subaddress Register, accessible through Subregister 1							

Bit	**Description**
[7:4]	**Reserved** These bits are reserved and must be programmed to 0000.

Note: *y* = A, B, C, D.

Bit	Description (Continued)
[3:0] CHyEF	**Channel y Event Flag** This bit indicates whether a Capture/Compare event occurred for this channel. Software can use this bit to determine the channel(s) responsible for generating the MCT channel interrupt. This event flag is cleared by writing a 1 to the bit. These bits will be set when an event occurs independently of the setting of the CHIEN bit. This bit is cleared when TEN = 0 (TEN is the MSB of MCTCTL1). 0: No Capture/Compare event occurred for this channel. 1: A Capture/Compare event occurred for this channel.

Note: *y* = A, B, C, D.

12.7.8. Multi-Channel Timer Channel-y Control Registers

Each channel in the Multi-Channel Timer Channel-y Control registers, shown in Table 100, has a control register to enable the channel, select the input/output polarity, enable channel interrupts, and select the channel mode of operation.

Table 100. Multi-Channel Timer Channel Control Register (MCTCHyCTL)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	CHEN	CHPOL	CHIEN	CHUE	Reserved	CHOP		
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R	R/W	R/W	R/W
Address	02h, 03h, 04h, 05h in Subaddress Register, accessible through Subregister 2							

Bit	Description
[7] CHEN	**Channel Enable** 0: Channel is disabled. 1: Channel is enabled.

Note: *y* = A, B, C, D.

Bit	Description (Continued)
[6] CHPOL	**Channel Input/Output Polarity** Operation of this bit is a function of the current operating mode of the channel. For Continuous Compare and PWM Output operation, configure this bit prior to setting the CHEN bit. ***One-Shot Operation*** When the channel is disabled, the Channel Output signal is set to the value of this bit. When the channel is enabled, the Channel Output signal toggles for one system clock on reaching the Channel Capture/Compare Register value. ***Continuous Compare Operation*** When the channel is disabled, the Channel Output signal is set to the value of this bit. When the channel is enabled, the Channel Output signal toggles (from Low to High or High to Low) on reaching the Channel Capture/Compare Register value. ***PWM Output Operation*** 0: Channel Output is forced Low when the channel is disabled. When enabled, the Channel Output is forced High on Channel Capture/Compare Register value match and forced Low on reaching the Timer Reload Register value (Modulo Mode) or counting down through the channel Capture/Compare register value (Count Up/Down Mode). 1: Channel Output is forced Low when the channel is disabled. When enabled, the Channel Output is forced High on Channel Capture/Compare Register value match and forced Low on reaching the Timer Reload Register value (Modulo Mode) or counting down through the channel Capture/Compare register value (Count Up/Down Mode). ***Capture Operation*** 0: Count is captured on the rising edge of the Channel Input signal. 1: Count is captured on the falling edge of the Channel Input signal.
[5] CHIEN	**Channel Interrupt Enable** This bit enables generation of channel interrupt. A channel interrupt is generated whenever there is a capture/compare event on the Timer Channel. 0: Channel interrupt is disabled. 1: Channel interrupt is enabled.
[4] CHUE	**Channel Update Enable** This bit determines whether writes to the Channel High and Low Byte registers are buffered when TEN = 1. Writes to these registers are not buffered when TEN = 0 regardless of the value of this bit. 0: Writes to the Channel High and Low Byte registers are buffered when TEN = 1, and only take affect on the next end-of-cycle count. 1: Writes to the Channel High and Low Byte registers are not buffered when TEN = 1.
[3]	**Reserved** This bit is reserved and must be programmed to 0.

Note: *y* = A, B, C, D.

Bit	Description (Continued)
[2:0] CHOP	**Channel Operation Mode** This field determines the operating mode of the channel. For a description of the operating modes, see the Count Up/Down Mode section on page 160. 00: One-Shot Compare. 001: Continuous Compare. 010: PWM Output. 011: Capture. 100–111: Reserved.
Note: *y* = A, B, C, D.	

12.7.9. Multi-Channel Timer Channel-y High and Low Byte Registers

Each channel has a 16-bit capture/compare register defined here as the Channel-y High and Low Byte registers. When the timer is enabled, writes to these registers are buffered and loading of the registers is delayed till the next timer end count, unless CHUE=1.

Table 101. Multi-Channel Timer Channel-y High Byte Registers (MCTCHyH)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	CHyH							
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	02h, 03h, 04h, 05h in Subaddress Register, accessible through Subregister 0							

Table 102. Multi-Channel Timer Channel-y Low Byte Registers (MCTCHyL)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	CHyL							
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	02h, 03h, 04h, 05h in Subaddress Register, accessible through Subregister 1							

Bit	Description
[7:0] CHyH, CHyL	**Multi-Channel Timer Channel-y High and Low Bytes** During a compare operation, these two bytes, {CHyH[7:0], CHyL[7:0]}, form a 16-bit value that is compared to the current 16-bit timer count. When a match occurs, the Channel Output changes state. The Channel Output value is set by the CHPOL bit in the Channel-y Control subregister. During a capture operation, the current Timer Count is recorded in these two bytes when the desired Channel Input transition occurs.
Note: *y* = A, B, C, D.	

Chapter 13. Watchdog Timer

The Watchdog Timer (WDT) function helps protect against corrupted or unreliable software and other system-level problems that can place the F3224 Series MCU into unsuitable operating states. The WDT includes the following features:

- Clocked by the Watchdog Timer Oscillator (WTO)
- A selectable time-out response: System Reset or Interrupt
- 16-bit programmable time-out value

13.1. Operation

The WDT is a retriggerable one-shot timer that resets or interrupts the F3224 Series MCU when the WDT reaches its terminal count. The WDT uses the Watchdog Timer Oscillator (WTO) as its clock source. The WDT has only two modes of operation: ON and OFF. After it is enabled, the WDT always counts and must be retriggered to prevent a time-out. An enable can be performed by executing the WDT instruction or by writing the WDT_AO option bit to 0. When 0, The WDT_AO bit enables the WDT to operate continuously, even if a WDT instruction has not been executed.

The WDT is a 16-bit reloadable downcounter that uses two 8-bit registers in the eZ8 CPU register space to set the reload value. The nominal WDT time-out period is calculated using the following equation:

$$\text{WDT Time–Out Period (ms)} = \frac{\text{WDT Reload Value}}{10}$$

In the above equation, the WDT reload value is computed using {WDTH[7:0], WDTL[7:0]} and the typical Watchdog Timer RC Oscillator frequency is 8.2 kHz. Users must consider system requirements when selecting the time-out delay. Table 103 indicates the approximate time-out delays for the default and maximum WDT reload values.

Table 103. Watchdog Timer Approximate Time-Out Delays

WDT Reload Value (Hex)	WDT Reload Value (Decimal)	Approximate Time-Out Delay (with 8.2 kHz Typical WDT Oscillator Frequency)	
		Typical	Description
0400	1024	125 ms	Reset default value time-out delay.
FFFF	65,536	7.99 s	Maximum time-out delay.

13.1.1. Watchdog Timer Retrigger

When first enabled, the WDT is loaded with the value in the WDT Reload registers. The WDT then counts down to `0000h` unless a WDT instruction is executed by the eZ8 CPU. Execution of the WDT instruction causes the downcounter to be reloaded with the WDT reload value stored in the WDT Reload registers. Counting resumes following the reload operation.

When the eZ8 CPU is operating in DEBUG Mode (through the OCD), the WDT is continuously retriggered to prevent unnecessary WDT time-outs.

13.1.2. Watchdog Timer Time-Out Response

The WDT times out when the counter reaches `0000h`. A time-out of the WDT generates either an interrupt or a System Reset. The WDT_RES option bit determines the time-out response of the WDT. To learn more about programming the WDT_RES option bit, see the Flash Option Bits chapter on page 358.

13.1.2.1. WDT Interrupt in Active and Halt Modes

If it is configured to generate an interrupt when a time-out occurs, the WDT issues an interrupt request to the Interrupt Controller. The WDT status bit in the Reset Status Register is set. The eZ8 CPU responds to the request by fetching the corresponding interrupt vector and executing code from the vector address. After time-out and interrupt generation, the WDT automatically reloads and continues counting. To clear the WDT interrupt, clear the WDT bit in the Reset Status Register.

13.1.2.2. WDT Interrupt in Stop Mode

The WDT automatically initiates a Stop-Mode Recovery and generates an interrupt request if configured to generate an interrupt when a time-out occurs and the CPU is in Stop Mode. Both the WDT status bit and the STOP bit in the Reset Status Register are set to 1 following a WDT time-out in Stop Mode. After time-out and interrupt generation, the WDT automatically reloads and continues counting.

Upon completion of the Stop-Mode Recovery, the eZ8 CPU responds to the interrupt request by fetching the corresponding interrupt vector and executing code from the vector address. To clear the WDT interrupt, clear the WDT bit in the Reset Status Register.

13.1.2.3. WDT Reset in Active and Halt Modes

The WDT forces the device into the Reset state if it is configured to generate a System Reset when a time-out occurs; the WDT status bit is set to 1 (for details, see the Reset Status Register (RSTSTAT) on page 39). For more information about System Reset and the WDT status bit, see the Reset, Stop-Mode Recovery and Low-Voltage Detection chapter on page 29. Following a System Reset, the WDT Counter is initialized with its reset value.

13.1.2.4. WDT Reset in Stop Mode

If enabled in Stop Mode and configured to generate a System Reset when a time-out occurs and the device is in Stop Mode, the WDT initiates a Stop-Mode Recovery. Both the WDT status bit and the STOP bit in the Reset Status Register (RSTSTAT) (see page 39) are set to 1 following a WDT time-out in Stop Mode. For more information, see the Reset, Stop-Mode Recovery and Low-Voltage Detection section on page 29.

13.1.3. Watchdog Timer Reload Unlock Sequence

Writing the unlock sequence to the Watchdog Timer Reload High (WDTH) Register address unlocks the two Watchdog Timer Reload registers (WDTH and WDTL) to allow changes to the time-out period. These write operations to the WDTH Register address produce no effect on the bits in the WDTH Register. The locking mechanism prevents unwarranted writes to the Reload registers. The following sequence is required to unlock the Watchdog Timer Reload registers (WDTH and WDTL) for write access.

1. Write `55h` to the Watchdog Timer Reload High Register (WDTH).
2. Write `AAh` to the Watchdog Timer Reload High Register (WDTH).
3. Write the appropriate value to the Watchdog Timer Reload High Register (WDTH).
4. Write the appropriate value to the Watchdog Timer Reload Low Register (WDTL). When this write occurs, the Watchdog Timer Reload registers are again locked.

All steps of the WDT Reload Unlock sequence must be written in the order defined above. The values in these WDT Reload registers are loaded into the counter every time a WDT instruction is executed.

13.2. Watchdog Timer Register Definitions

The two Watchdog Timer Reload registers (WDTH and WDTL) are described in the following tables.

13.2.1. Watchdog Timer Reload High and Low Byte Registers

The Watchdog Timer Reload High and Low Byte (WDTH, WDTL) registers, shown in Tables 104 and 105, form the 16-bit reload value that is loaded into the Watchdog Timer when a WDT instruction executes; this 16-bit reload value is {WDTH[7:0], WDTL[7:0]}. Writing to these registers following the unlock sequence sets the appropriate reload value. Reading from these registers returns the current WDT count value.

Table 104. Watchdog Timer Reload High Byte Register (WDTH=FF2h)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	WDTH							
Reset	0	0	0	0	0	1	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	FF2h							

Table 105. Watchdog Timer Reload Low Byte Register (WDTL=FF3h)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	WDTL							
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	FF3h							

Bit	**Description**
[7:0] WDTH, WDTL	**Watchdog Timer Reload High and Low Bytes** WDTH: The WDT Reload High Byte is the most significant byte, or bits [15:8] of the 16-bit WDT reload value. WDTL: The WDT Reload Low Byte is the least significant byte, or bits [7:0] of the 16-bit WDT reload value.

Chapter 14. UART-LDD

The Local Interconnect Network Universal Asynchronous Receiver/Transmitter (UART-LDD) is a full-duplex communication channel capable of handling asynchronous data transfers in standard UART applications and providing LIN, DALI, and DMX protocol support. The UART-LDD is a superset of the standard F3224 Series MCU UART, providing all of its standard features, LIN/DALI/DMX protocol support and a digital noise filter.

UART-LDD includes the following features:

- 8-bit asynchronous data transfer
- Selectable even- and odd-parity generation and checking
- Option of 1 or 2 stop bits
- Selectable Multiprocessor (9-bit) Mode with three configurable interrupt schemes
- Separate transmit and receive interrupts
- Framing, parity, overrun and break detection
- 16-bit baud rate generator (BRG) which can function as a general-purpose timer with interrupt
- Driver Enable output for external bus transceivers
- LIN protocol support for both Master and Slave modes:
  - Break generation and detection
  - Selectable slave autobaud
  - Check Tx vs. Rx data when sending
- DALI protocol support for both Master and Slave modes:
  - Biphase data encoding
  - Slave address matching
- DMX protocol support for both Master and Slave modes:
  - Slave address matching
  - Automatic break generation
- Configurable digital-noise filter on the Receive Data line

14.1. UART-LDD Architecture

The UART-LDD consists of three primary functional blocks: transmitter, receiver and baud-rate generator. The UART-LDD's transmitter and receiver function independently but use the same baud rate and data format. The basic UART operation is enhanced by the Noise Filter. Figure 23 shows the UART-LDD architecture.

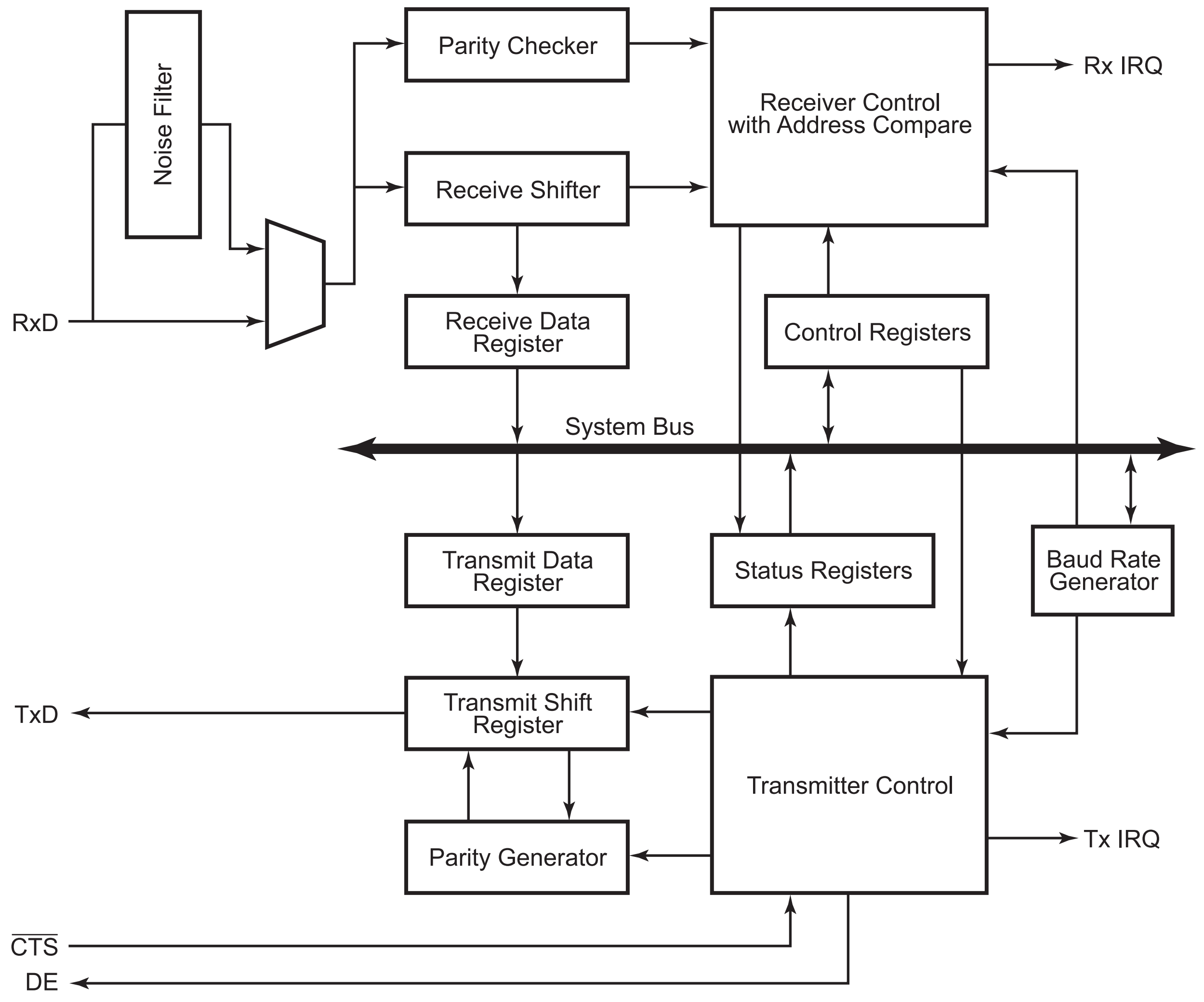


Figure 23. UART-LDD Block Diagram

14.1.1. Data Format for Standard UART Modes

The UART-LDD always transmits and receives data in an 8-bit data format with the least significant bit first. An even-or-odd parity bit or multiprocessor address/data bit can be optionally added to the data stream. Each character begins with an active Low start bit and ends with either one or two active High stop bits. Figures 24 and 25 show the asynchronous data format employed by the UART-LDD without parity and with parity, respectively.

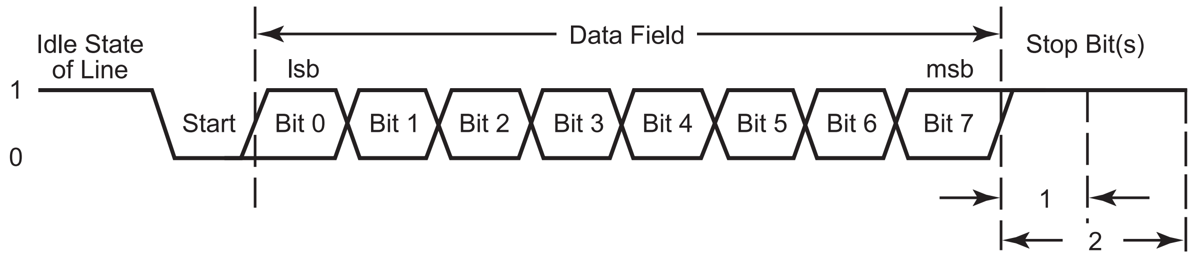


Figure 24. UART-LDD Asynchronous Data Format without Parity

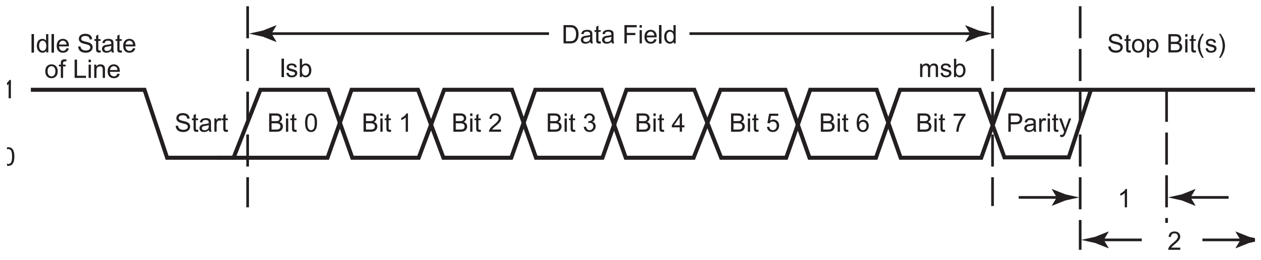


Figure 25. UART-LDD Asynchronous Data Format with Parity

14.1.2. Transmitting Data using the Polled Method

Observe the following steps to transmit data using the polled-operating method:

1. Write to the UART-LDD Baud Rate High and Low Byte registers to set the appropriate baud rate.
2. Enable the UART-LDD pin functions by configuring the associated GPIO port pins for alternate-function operation.
3. If Multiprocessor Mode is appropriate, write to the UART-LDD Control 1 Register to enable Multiprocessor (9-bit) Mode functions.
4. Set the Multiprocessor Mode Select (MPEN) bit to enable Multiprocessor Mode.

5. Write to the UART-LDD Control 0 Register to:
   a. Set the Transmit Enable (TEN) bit to enable the UART-LDD for data transmission.
   b. If parity is appropriate and Multiprocessor Mode is not enabled, set the parity enable (PEN) bit and select either even-or-odd parity (PSEL).
   c. Set or clear the CTSE bit to enable or disable control from the remote receiver using the $\overline{\text{CTS}}$ pin.
6. Check the TDRE bit in the UART-LDD Status 0 Register to determine if the Transmit Data Register is empty (indicated by a 1); if empty, continue to Step 7. If the Transmit Data Register is full (indicated by a 0), continue to monitor the TDRE bit until the Transmit Data Register becomes available to receive new data.
7. If operating in Multiprocessor Mode, write to the UART-LDD Control 1 Register to select the outgoing address bit.
   – Set the Multiprocessor Bit Transmitter (MPBT) if sending an address byte; clear it if sending a data byte.
8. Write the data byte to the UART-LDD Transmit Data Register. The transmitter automatically transfers the data to the Transmit Shift Register and transmits the data.
9. If appropriate – and if Multiprocessor Mode is enabled – changes can be made to the Multiprocessor Bit Transmitter (MPBT) value.
10. To transmit additional bytes, return to Step 6.

14.1.3. Transmitting Data Using Interrupt-Driven Method

The UART-LDD Transmitter interrupt indicates the availability of the Transmit Data Register to accept new data for transmission. Observe the following steps to configure the UART-LDD for interrupt-driven data transmission:

1. Write to the UART-LDD Baud Rate High and Low Byte registers to set the appropriate baud rate.
2. Enable the UART-LDD pin functions by configuring the associated GPIO port pins for alternate function operation.
3. Execute a DI instruction to disable interrupts.
4. Write to the interrupt control registers to enable the UART-LDD Transmitter interrupt and set the appropriate priority.
5. If Multiprocessor Mode is appropriate, write to the UART-LDD Control 1 Register to enable Multiprocessor (9-bit) Mode functions.
6. Set the Multiprocessor Mode Select (MPEN) bit to enable Multiprocessor Mode.
7. Write to the UART-LDD Control 0 Register to:
   a. Set the transmit enable (TEN) bit to enable the UART-LDD for data transmission.

   b. If Multiprocessor Mode is not enabled, then enable parity if appropriate and select either even or odd parity.
   c. Set or clear the CTSE bit to enable or disable control from the remote receiver via the $\overline{\text{CTS}}$ pin.

8. Execute an EI instruction to enable interrupts.

The UART-LDD is now configured for interrupt-driven data transmission. Because the UART-LDD Transmit Data Register is empty, an interrupt is generated immediately. When the UART-LDD Transmit interrupt is detected and there is transmit data ready to send, the associated interrupt service routine (ISR) performs the following:

1. If in Multiprocessor Mode, writes to the UART-LDD Control 1 Register to select the outgoing address bit:
   – Sets the Multiprocessor Bit Transmitter (MPBT) if sending an address byte, clears it if sending a data byte.

2. Writes the data byte to the UART-LDD Transmit Data Register. The transmitter automatically transfers the data to the Transmit Shift Register and transmits the data.
3. Executes the IRET instruction to return from the interrupt-service routine and wait for the Transmit Data Register to again become empty.

If a transmit interrupt occurs and there is no transmit data ready to send, the interrupt service routine executes the IRET instruction. When the application does have data to transmit, software can set the appropriate interrupt request bit in the Interrupt Controller to initiate a new transmit interrupt. Another alternative would be for the software to write the data to the Transmit Data Register instead of invoking the interrupt service routine.

14.1.4. Receiving Data Using Polled Method

Observe the following steps to configure the UART-LDD for polled data reception:

1. Write to the UART-LDD Baud Rate High and Low Byte registers to set the appropriate baud rate.
2. Enable the UART-LDD pin functions by configuring the associated GPIO port pins for alternate function operation.
3. If Multiprocessor Mode is appropriate, write to the UART-LDD Control 1 Register to enable Multiprocessor (9-bit) Mode functions.
4. Write to the UART-LDD Control 0 Register to:
   a. Set the Receive Enable (REN) bit to enable the UART-LDD for data reception.

   b. If Multiprocessor Mode is not enabled, then enable parity (if appropriate), and select either even or odd parity.

5. Check the RDA bit in the UART-LDD Status 0 Register to determine if the Receive Data Register contains a valid data byte (indicated by a 1). If RDA is set to 1 to indicate available data, continue to Step 6. If the Receive Data Register is empty (indicated by a 0), continue to monitor the RDA bit that is awaiting reception of the valid data.
6. Read data from the UART-LDD Receive Data Register. If operating in Multiprocessor (9-bit) Mode, further actions may be required depending on the Multiprocessor Mode bits MPMD[1:0].
7. Return to Step 5 to receive additional data.

14.1.5. Receiving Data Using the Interrupt-Driven Method

The UART-LDD Receiver interrupt indicates the availability of new data (as well as error conditions). Observe the following steps to configure the UART-LDD receiver for interrupt-driven operation:

1. Write to the UART-LDD Baud Rate High and Low Byte registers to set the appropriate baud rate.
2. Enable the UART-LDD pin functions by configuring the associated GPIO port pins for alternate function operation.
3. Execute a DI instruction to disable interrupts.
4. Write to the Interrupt Control registers to enable the UART-LDD Receiver interrupt and set the appropriate priority.
5. Clear the UART-LDD Receiver interrupt in the applicable Interrupt Request Register.
6. Write to the UART-LDD Control 1 Register to enable Multiprocessor (9-bit) Mode functions, if appropriate.
   a. Set the Multiprocessor Mode Select (MPEN) bit to enable Multiprocessor Mode.
   b. Set the Multiprocessor Mode Bits, MPMD[1:0] to select the appropriate address matching scheme.
   c. Configure the UART-LDD to interrupt on received data and errors or errors only (interrupt on errors only is unlikely to be useful for Z8 Encore! devices without a DMA block).
7. Write the device address to the Address Compare Register (automatic Multiprocessor modes only).
8. Write to the UART-LDD Control 0 Register to:
   a. Set the receive enable (REN) bit to enable the UART-LDD for data reception.

   b. If Multiprocessor Mode is not enabled, then enable parity (if appropriate) and select either even or odd parity.

9. Execute an EI instruction to enable interrupts.

The UART-LDD is now configured for interrupt-driven data reception. When the UART-LDD Receiver interrupt is detected, the associated ISR performs the following:

1. Checks the UART-LDD Status 0 Register to determine the source of the interrupt-error, break, or received data.
2. If the interrupt is due to data available, read the data from the UART-LDD Receive Data Register. If operating in Multiprocessor (9-bit) Mode, further actions may be required depending on the Multiprocessor Mode bits MPMD[1:0].
3. Execute the IRET instruction to return from the ISR and await more data.

14.1.6. Clear To Send Operation

The Clear To Send ($\overline{\text{CTS}}$) pin, if enabled by the CTSE bit of the UART-LDD Control 0 Register, performs flow control on the outgoing transmit data stream. The Clear To Send ($\overline{\text{CTS}}$) input pin is sampled one system clock before any new character transmission begins. To delay transmission of the next data character, an external receiver must reduce $\overline{\text{CTS}}$ at least one system clock cycle before a new data transmission begins. For multiple character transmissions, this operation is typically performed during the stop bit transmission. If $\overline{\text{CTS}}$ stops in the middle of a character transmission, the current character is sent completely.

14.1.7. External Driver Enable

The UART-LDD provides a Driver Enable (DE) signal for off-chip bus transceivers. This feature reduces the software overhead associated using a GPIO pin to control the transceiver when communicating on a multitransceiver bus, such as RS-485.

Driver Enable is a programmable polarity signal which envelopes the entire transmitted data frame including parity and stop bits, as illustrated in Figure 26. The Driver Enable signal asserts when a byte is written to the UART-LDD Transmit Data Register. The Driver Enable signal asserts at least one bit period, and no greater than two bit periods, before the start bit is transmitted. This assertion allows a set-up time to enable the transceiver. The Driver Enable signal deasserts one system clock period after the last stop bit is transmitted. This system clock delay allows both time for data to clear the transceiver before disabling it, as well as the ability to determine if another character follows the current character. In the event of back-to-back characters (new data must be written to the Transmit Data Register before the previous character is completely transmitted), the DE signal is not deasserted between characters. The DEPOL bit in the UART-LDD Control Register 1 sets the polarity of the Driver Enable signal.

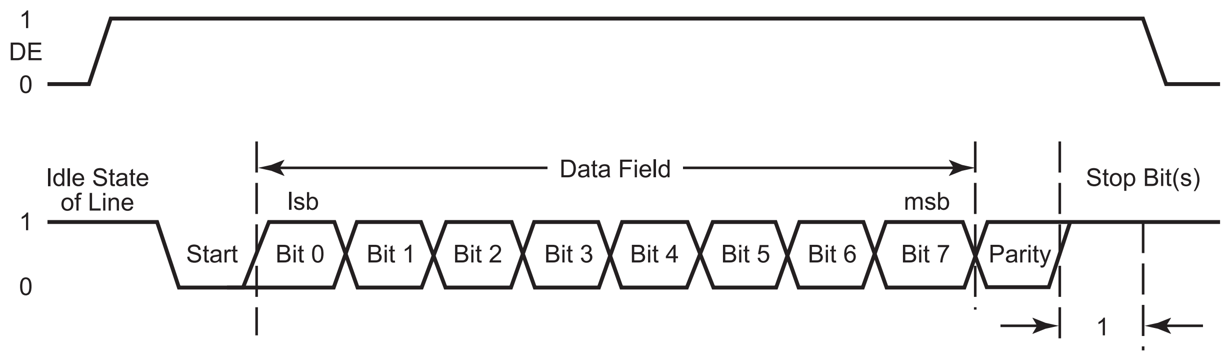


Figure 26. UART-LDD Driver Enable Signal Timing with One Stop Bit and Parity

The Driver Enable to start bit set-up time is calculated as follows:

$$\frac{1}{\text{Baud Rate (Hz)}} \leq \text{DE to Start Bit Set-up Time(s)} \leq \frac{2}{\text{Baud Rate (Hz)}}$$

14.1.8. UART-LDD Special Modes

The special modes of the UART-LDD are:

- Multiprocessor Mode
- LIN Mode
- DALI Mode
- DMX Mode

The UART-LDD features a common control register (Control 0) that has a unique register address and several mode-specific control registers (Multiprocessor Control, Noise Filter Control, LIN Control, DALI Control, and DMX Control) that share a common register address (Control 1). When the Control 1 address is read or written, the MSEL[2:0] (Mode Select) field of the Mode Select and Status Register determines which physical register is accessed. Similarly, there are mode-specific status registers, one of which is returned when the Status 0 Register is read, depending on the MSEL field.

14.1.9. Multiprocessor Mode

The UART-LDD features a Multiprocessor (9-bit) mode that uses an extra (9th) bit for selective communication when a number of processors share a common UART bus. In Multiprocessor Mode (also referred to as 9-bit mode), the multiprocessor (MP) bit is transmitted immediately following the 8 bits of data and immediately preceding the stop bit(s) as shown in Figure 27.

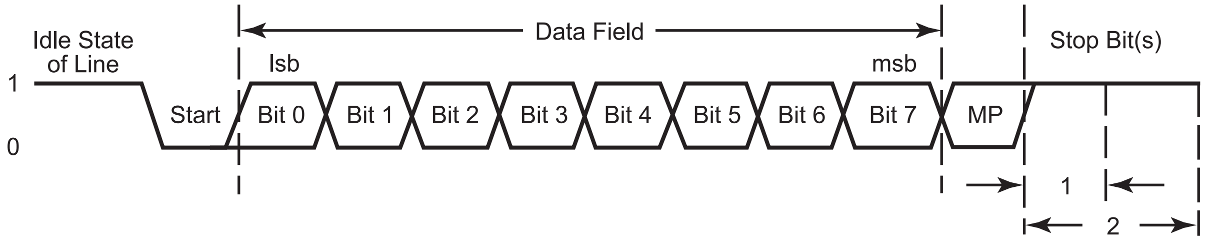


Figure 27. UART-LDD Asynchronous Multiprocessor Mode Data Format

In Multiprocessor (9-bit) Mode, the Parity bit location (9th bit) becomes the Multiprocessor control bit. The UART-LDD Control 1 and Status 1 registers provide Multiprocessor (9-bit) Mode control and status information. If an automatic address matching scheme is enabled, the UART-LDD Address Compare Register holds the network address of the device.

14.1.9.1. Multiprocessor Mode Receive Interrupts

When Multiprocessor (9-bit) Mode is enabled, the UART-LDD processes only frames addressed to it. Determining whether a frame of data is addressed to the UART-LDD can be made in hardware, software or a combination of the two, depending on the multiprocessor configuration bits. In general, the address compare feature reduces the load on the CPU, because it is not required to access the UART-LDD when it receives data directed to other devices on the multinode network. The following three Multiprocessor modes are available in hardware:

- Interrupt on all address bytes
- Interrupt on matched address bytes and correctly framed data bytes
- Interrupt only on correctly framed data bytes

These modes are selected with MPMD[1:0] in the UART-LDD Control 1 Register. For all Multiprocessor modes, the MPEN bit of the UART-LDD Control 1 Register must be set to 1.

The first scheme is enabled by writing `01b` to MPMD[1:0]. In this mode, all incoming address bytes cause an interrupt, while data bytes never cause an interrupt. The interrupt service routine checks the address byte which triggered the interrupt. If it matches the UART-LDD address, the software clears MPMD[0]. At this point, each new incoming byte interrupts the CPU. The software determines the end of the frame and checks for it by reading the MPRX bit of the UART-LDD Status 1 Register for each incoming byte. If

MPRX=1, a new frame begins. If the address of this new frame is different from the UART-LDD's address, then MPMD[0] must be set to 1 by software, causing the UART-LDD interrupts to go inactive until the next address byte. If the new frame's address matches the UART-LDD's address, then the data in the new frame is also processed.

The second scheme is enabled by setting MPMD[1:0] to `10b` and writing the UART-LDD's address into the UART-LDD Address Compare Register. This mode introduces more hardware control, interrupting only on frames that match the UART-LDD's address. When an incoming address byte does not match the UART-LDD's address, it is ignored. All successive data bytes in this frame are also ignored. When a matching address byte occurs, an interrupt is issued and further interrupts occur on each successive data byte. The first data byte in the frame has NEWFRM=1 in the UART-LDD Status 1 Register. When the next address byte occurs, the hardware compares it to the UART-LDD's address. If there is a match, the interrupt occurs and the NEWFRM bit is set to the first byte of the new frame. If there is no match, the UART-LDD ignores all incoming bytes until the next address match.

The third scheme is enabled by setting MPMD[1:0] to `11b` and by writing the UART-LDD's address into the UART-LDD Address Compare Register. This mode is identical to the second scheme, except that there are no interrupts on address bytes. The first data byte of each frame remains accompanied by a NEWFRM assertion.

14.1.10. LIN Protocol Mode

The Local Interconnect Network (LIN) protocol, as supported by the UART-LDD module, is defined in Revision 2.0 of the LIN Specification Package. The LIN protocol specification covers all aspects of transferring information between LIN master and slave devices using *message frames*, including error detection and recovery, SLEEP Mode and wake up from SLEEP Mode. The UART-LDD hardware in LIN Mode provides character transfers to support the LIN protocol including break transmission and detection, wake-up transmission and detection and slave autobauding. Part of the error detection of the LIN protocol is for both master and slave devices to monitor their receive data when transmitting. If the receive and transmit data streams do not match, the UART-LDD asserts the PLE bit (i.e., the physical layer error bit in the Status 0 Register). The *message frame* time-out aspect of the protocol depends on software requiring the use of an additional general-purpose timer. The LIN Mode of the UART-LDD does not provide any hardware support for computing/verifying the checksum field or verifying the contents of the identifier field. These fields are treated as data and are not interpreted by hardware. The checksum calculation/verification can easily be implemented in software via the Add with Carry (ADC) instruction.

The LIN bus contains a single Master and one or more slaves. The LIN master is responsible for transmitting the message frame header which consists of the Break, Synch and Identifier fields. Either the master or one of the slaves transmits the associated *response* section of the message which consists of data characters followed by a checksum character.

In LIN Mode, the interrupts defined for normal UART operation still apply with the following changes:

- A Parity Error (i.e., the PE bit in the Status 0 Register) is redefined as the Physical Layer Error (PLE) bit. The PLE bit indicates that receive data does not match transmit data when the UART-LDD is transmitting. This definition applies to both Master and Slave operating modes.
- The Break Detect interrupt (i.e., the BRKD bit in the Status 0 Register) indicates when a break is detected by the slave (i.e., a break condition for at least 11 bit times). Software can use this interrupt to start a timer checking for message frame time-out. The duration of the break can be read in the RxBreakLength[3:0] field of the Mode Select and Status Register.
- The Break Detect interrupt (BRKD bit in Status 0 Register) indicates when a wake-up message has been received, if the UART-LDD is in a LIN Sleep state.
- In LIN Slave Mode, if the BRG counter overflows while measuring the autobaud period (from the start bit to the beginning of bit 7 of the autobaud character), an Overrun Error is indicated (OE bit in the Status 0 Register). In this case, software sets the LinState field back to `10b`, where the slave ignores the current message and waits for the next break. The Baud Reload High and Low registers are not updated by hardware if this autobaud error occurs. The OE bit is also set if a data overrun error occurs.

14.1.10.1. LIN System Clock Requirements

The LIN Master provides the timing reference for the LIN network and is required to have a clock source with a tolerance of ±0.5%. A slave with autobaud capability is required to have a baud clock matching the master oscillator within ±14%. The slave nodes autobaud to lock onto the master timing reference with an accuracy of ±2%. If a slave does not contain autobaud capability, it must include a baud clock which deviates from the masters by not more than ±1.5%. These accuracy requirements must include the effects such as voltage and temperature drift during operation.

Before sending/receiving messages, the Baud Reload High/Low registers must be initialized. Unlike standard UART modes, the Baud Reload High/Low registers must be loaded with the baud interval rather than 1/16 of the baud interval.

To autobaud with the required accuracy, the LIN slave system clock must be at least 100 times the baud rate.

14.1.10.2. LIN Mode Initialization and Operation

LIN Protocol Mode is selected by setting either the LIN Master (LMST) or LIN Slave (LSLV) and, optionally (for the LIN slave), the Autobaud Enable (ABEN) bits in the LIN Control Register. To access the LIN Control Register, the Mode Select (MSEL) field of the UART-LDD Mode Select/Status Register must be=`010b`. The UART-LDD Control 0 Register must be initialized with TEN=1, REN=1 and all other bits=0.

In addition to the LMST, LSLV and ABEN bits in the LIN Control Register, a Lin-State[1:0] field exists which defines the current state of the LIN logic. This field is initially set by software. In the LIN Slave Mode, the LinState field is updated by hardware as the slave moves through the Wait For Break, AutoBaud and Active states.

14.1.10.3. LIN Master Mode Operation

LIN Master Mode is selected by setting LMST=1, LSLV=0, ABEN=0 and LinState[1:0]=`11b` (*sleep state*). If the LIN bus protocol indicates the bus is required go into the LIN Sleep state, the LinState[1:0] bits must be set to `00b` by software.

The break is the first part of the message frame transmitted by the master, consisting of at least 13 bit periods of logical zero on the LIN bus. During initialization of the LIN master, the duration (in bit times) of the break is written to the TxBreakLength field of the LIN Control Register. The transmission of the break is performed by setting the SBRK bit in the Control 0 Register. The UART-LDD starts the break after the SBRK bit is set and any character transmission currently underway has completed. The SBRK bit is deasserted by hardware until the break is completed.

If it is necessary to generate a break longer than 15 bit times, the SBRK bit can be used in normal UART Mode, in which software times the duration of the break.

The Synch character is transmitted by writing a `55h` to the Transmit Data Register (TDRE must=1 before writing). The Synch character is not transmitted by the hardware until the break is complete.

The identifier character is transmitted by writing the appropriate value to the Transmit Data Register (TDRE must=1 before writing).

If the master is sending the response portion of the message, these data and checksum characters are written to the Transmit Data Register when the TDRE bit asserts. If the Transmit Data Register is written after TDRE asserts, but before TXE asserts, the hardware inserts one or two stop bits between each character as determined by the stop bit in the Control 0 Register. Additional idle time occurs between characters, if TXE asserts before the next character is written.

If the selected slave is sending the response portion of the frame to the master, each receive byte will be signalled by the receive data interrupt (the RDA bit will be set in the Status 0 Register). If the selected slave is sending the response to a different slave, the master can ignore the response characters by deasserting the REN bit in the Control 0 Register until the frame time slot is completed.

14.1.10.4. LIN Sleep Mode

While the LIN bus is in the *sleep state*, the CPU can either be in low-power Stop Mode, in Halt Mode, or in normal operational state. Any device on the LIN bus can issue a wake-up message if it requires the master to initiate a LIN message frame. Following the wake-up message, the master wakes up and initiates a new message. A wake-up message is accomplished by pulling the bus Low for at least 250 µs but less than 5ms. Transmitting a `00h` character is one way to transmit the wake-up message.

If the CPU is in Stop Mode, the UART-LDD is not active and the wake-up message must be detected by a GPIO edge detect Stop-Mode Recovery. The duration of the Stop-Mode Recovery sequence can preclude making an accurate measurement of the wake-up message duration.

If the CPU is in a halt or operational mode, the UART-LDD (if enabled) times the duration of the wake-up and provides an interrupt following the end of the break sequence if the duration is ≥ 3 bit times. The total duration of the wake-up message in bit times can be obtained by reading the RxBreakLength field in the Mode Select and Status Register. After a wake-up message has been detected, the UART-LDD can be placed (by software) either into LIN Master or LIN Slave Wait for Break states, as appropriate. If the break duration exceeds 15 bit times, the RxBreakLength field contains the value `Fh`. If the UART-LDD is disabled, wake-up message is detected via a port pin interrupt and timed by software. If the device is in Stop Mode, the High to Low transition on the port pin will bring the device out of Stop Mode.

The *LIN Sleep state* is selected by software setting LinState[1:0]=00. The decision to move from an active state to sleep state is based on the LIN messages as interpreted by software.

14.1.10.5. LIN Slave Operation

LIN Slave Mode is selected by setting LMST=0, LSLV=1, ABEN=1 or 0 and LinState[1:0]=`01b` (Wait for Break state). The LIN slave detects the start of a new message by the break which appears to the slave as a break of at least 11 bit times in duration. The UART-LDD detects the break and generates an interrupt to the CPU. The duration of the break is observable in the RxBreakLength field of the Mode Select and Status Register. A break of less than 11 bit times in duration does not generate a break interrupt when the UART-LDD is in a Wait for Break state. If the break duration exceeds 15 bit times, the RxBreakLength field contains the value `Fh`.

Following the break, the UART-LDD hardware automatically transits to the *Autobaud state*, where it autobauds by timing the duration of the first 8 bit times of the Synch character as defined in the LIN standard. The duration of the autobaud period is measured by the BRG Counter which will update every 8th system clock cycle between the start bit and the beginning of bit 7 of the autobaud sequence. At the end of the autobaud period, the duration measured by the BRG counter (auto baud period divided by 8) is automatically transferred to the Baud Reload High and Low registers if the ABEN bit of the LIN Control Register is set. If the BRG Counter overflows before reaching the start of bit 7 in the autobaud sequence the Autobaud Overrun Error interrupt occurs, the OE bit in the Status 0 Register is set and the Baud Reload registers are not updated. To autobaud within 2% of the master's baud rate, the slave system clock must be a minimum of 100 times the baud rate. To avoid an autobaud overrun error, the system clock must not be greater than 2^{19}

times the baud rate (16 bit counter following 3-bit prescaler when counting the 8 bit times of the Autobaud sequence).

Following the Synch character, the UART-LDD hardware transits to the *Active* state, in which the identifier character is received and the characters of the *response* section of the message are sent or received. The slave remains in this *Active* state until a break is received or software forces a state change. After it is in an Active state (i.e., autobaud has completed), a break of 10 or more bit times is recognized and causes a transition to the Autobaud state.

If the identifier character indicates that this slave device is not participating in the message, the software sets the LinState[1:0]=`01b` (Wait for Break state) to ignore the remainder of the message. No further receive interrupts will occur until the next break.

14.1.11. DALI Protocol Mode

The Digital Addressable Lighting Interface (DALI) protocol, as supported by the UART-LDD module, is defined in IEC62386-201. This DALI protocol specification covers all aspects of transferring information between DALI master and DALI slave devices. The UART-LDD hardware provides character transfers to support the DALI protocol, including biphase encoding and decoding, message formation, and slave message address extraction for matching with the comparison address (COMP_ADDR).

Forward messages from a DALI master to a DALI slave are typically19-bit messages consisting of a start bit, an 8-bit address, 8-bit data, and 2 stop bits. The start bit, address byte, and data byte are biphase encoded; stop bits are not biphase encoded. Backward messages from a slave to a master are typically 11 bits consisting of 1 start bit, 8 data bits and 2 stop bits. The slave transmits only upon the request of the master. The DALI start bit is encoded as a logical 1 (a Low to High transition) and the stop bits are High levels. The DALI standard frames and biphase bit encoding are shown in Figure 28.

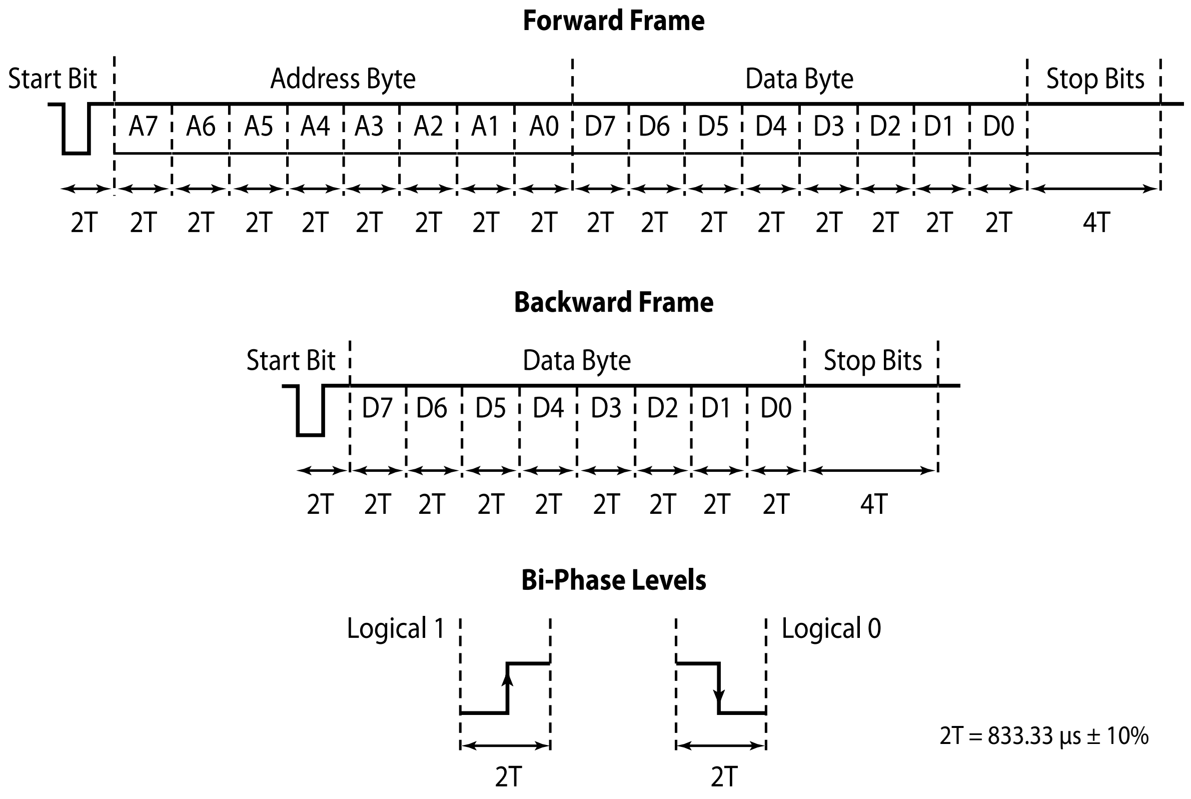


Figure 28. UART-LDD DALI Standard Frames and Biphase Bit Encoding

In DALI Mode, the interrupts defined for normal UART operation still apply, but with the following changes:

- A Parity Error (i.e., the PE bit in the Status 0 Register) is replaced with a biphase error, BPE, which indicates that there was a biphase encode error
- The Break Detect interrupt (i.e., the BRKD bit in the Status 0 Register) is not set
- Framing error checking occurs only for single-byte transfers (i.e, MULTRXE=0 in the DALI Control Register)

14.1.11.1. DALI Clock Requirements

Both the DALI master and DALI slaves are required to have a nominal 1.2 kbit/s bit rate with a tolerance of ±10%.

Before sending/receiving messages, the Baud Reload High/Low registers must be initialized. Unlike standard UART modes, the Baud Reload High/Low registers must be loaded with 1/32 of the baud interval rather than 1/16 of the baud interval.

14.1.11.2. DALI Mode Initialization and Operation

DALI Protocol Mode is selected by setting either the MULTTXE (multiple-byte transmit enable) or MULTRXE (multiple-byte receive enable) in the DALI Control Register. To access the DALI Control Register, the MSEL (Mode Select) field of the UART-LDD Mode Select/Status Register must be=`100b`. The UART-LDD Control 0 Register must be initialized. For DALI transmit operation, configure TEN=1, STOP=1 and all other bits=0. For DALI receive operation, configure REN=1, STOP=1 and all other bits=0.

In the DALI Control Register, several bits affect both transmit and receive operation:

Biphase Encoding Enable (BPEN). BPEN should be set for DALI operation.

DALI Biphase Encoding (BPENC). BPENC has an effect only if BPEN=1. When BPENC=0, DALI encoding is performed such that logic 0 is encoded as biphase 1→0 and logic 1 is encoded as biphase 0→1. When BPENC=1, alternate encoding is performed such that logic 0 is encoded as biphase 0→1 and logic 1 is encoded as biphase 1→0.

Start Bit Polarity (STRTPOL). The start bit value, logic 0 or logic 1, matches the value of this bit. STRTPOL is typically set for DALI.

Bit Order (BITORD). Standard UART bit order is selected when BITORD=0 and the LSB (TxD[0]/RxD[0]) is transmitted/received first. DALI bit order is selected when BITORD=1 and the MSB (TxD[7]/RxD[7]) is transmitted/received first.

DALI Control Register bits that affect only transmit or receive operations are described in the following sections.

14.1.11.3. DALI Transmit Operation

The UART-LDD Control 0 Register must be initialized. For DALI transmit operation, configure TEN=1, STOP=1 and all other bits=0. When the MSEL= `100b` (Mode Select) in the DALI Control Register, DALI transmit operation can be configured for single-byte or multiple-byte transmission. If MULTTXE=0 in the DALI Control Register, single-byte transmit is selected and stop bits will be transmitted after each transmitted byte. This setting is typically selected for DALI slave response messages that are transmitted to the master.

If MULTTXE=1 in the DALI Control Register, multiple-byte transmit is selected and stop bits will be transmitted only after the last transmitted byte. This setting is typically selected for DALI master operation to send an address byte, followed by one or more data bytes. MULTTXE must also be set for DALI master operation. When the UART-LDD Transmit Data Register is written, the UART-LDD will send a start bit, followed by the 8 bits in the UART-LDD Transmit Data Register. As the data is transmitted, the UART-LDD will assert an interrupt request for the next data byte. If the Transmit Data Register is written after TDRE asserts – but before TXE asserts – the hardware will transmit the character in the Transmit Data Register without the intervening stop bits. If TXE asserts before the next character is written in the Transmit Data Register, the DALI Master will transmit two stop bits after the last data bit.

Collision detection is enabled by setting CLSNE in the DALI Control Register. This setting is useful for DALI masters in systems with more than one master, because it is possible for more than one master to start a transmission at the same time. When CLSNE is set, the UART-LDD monitors its own transmission. Because Low (i.e., 0) is the dominant state on the DALI bus, collision detection effectively checks to determine if High (i.e., 1) state transmissions are not corrupted. If a collision is detected, CLSN is set in the Status Register and an interrupt request is generated.

14.1.11.4. DALI Receive Operation

The UART-LDD Control 0 Register must be initialized. For DALI receive operation, configure REN=1, STOP=1 and all other bits=0. When the MSEL= `100b` (Mode Select) in the DALI Control Register, DALI receive operation can be configured for single-byte or multiple-byte reception. If MULTRXE=0 in the DALI Control Register, single-byte receive is selected, and stop bits will be expected after each transmitted byte. This setting is typically selected for a DALI master that will receive a slave response message. Address match checking is not performed when MULTRXE=0.

If MULTRXE=1 in the DALI Control Register, multiple-byte receive is selected, and stop bits will be received to signal the end of the transmission. This setting is typically selected for DALI slave operation to receive an address byte followed by one or more data bytes.

If PARTRXE is set, and if a partial byte has been received, it will be loaded into the UART-LDD Receive Data Register upon receiving the number of stop bits selected by STOP in the UART-LDD Control 0 Register. Software should determine which bits in the received byte are valid.

When MULTRXE=1 in the DALI Control Register, the start bit is detected as the beginning of a new message. The UART-LDD decodes the first byte received as an address, and a status is provided with MODESTAT in the UART-LDD Mode Select and Status Register, as follows:

0–7Fh. Short address, each DALI slave is assigned a short address (MODESTAT=001).

80–9Fh. Group address (MODESTAT=010).

A0–FDh. Special or unrecognized command (MODESTAT=101).

FE–FFh. Broadcast (MODESTAT=100).

Address matching for short addresses is performed by hardware, which compares the short address in the received address byte to the value of COMP_ADDR[5:0] stored in the Comparison Address Register. If a short address is received that does not match the value of COMP_ADDR[5:0], the message is ignored.

Each DALI slave can belong to as many as 4 groups of the 16 available groups. Software should determine whether the slave belongs to the group for which the message is intended, and whether to process the message.

Except for cases in which messages with short addresses do not match the value of COMP_ADDR[5:0], when each byte is received including the first byte, RDA is set in the UART-LDD Status Register, and an interrupt request is generated.

14.1.11.5. DALI Receive During Stop Mode

While the DALI bus is idle, the DALI receiver can either be in low-power Stop Mode, in Halt Mode, or in a normal operational state. While the receiver is in Stop Mode, the UART-LDD is not active, and the start bit must be detected by a GPIO edge-detect Stop-Mode Recovery. A High-to-Low transition on the port pin can be used to bring the device out of Stop Mode. When Stop-Mode Recovery is completed, and if enabled, the UART-LDD will recognize the start bit. The duration of the Stop-Mode Recovery sequence can preclude recognizing the start bit.

14.1.11.6. Control Register Settings for DALI Mode

Example control register settings for DALI Mode operation are shown in Table 106. The baud rate settings in the BRH and BRL registers correspond to a 20MHz System Clock and 1.2 kbits/sec DALI bit rate but other System Clock frequencies and baud rate divisors may be used.

Table 106. Example Control Register Settings for DALI Mode

Control Register	DALI Master	DALI Slave
UMDSTAT	80h	80h
UCTL0	C2h	C2h
UCTL1	ECh	6Ch
BRH	02h	02h
BRL	09h	09h

14.1.12. DMX Protocol Mode

The Asynchronous Serial Digital Data Transmission Standard for Controlling Lighting Equipment and Accessories (DMX) protocol, as supported by the UART-LDD module, is defined in ANSI E1.11-2008. This DMX protocol specification covers all aspects of transferring information from a DMX master to DMX slave devices. The UART-LDD hardware provides character transfers to support the DMX protocol, including break transmission and detection, mark after break transmission and detection, and tracking by the DMX slave of the received data slot number for matching with the comparison address (COMP_ADDR). The DMX mode of the UART-LDD also provides hardware support for recognizing a null start code.

The DMX bus contains a single master and one or more slaves. The DMX protocol consists of a reset sequence followed by up to 512 data slots. The DMX master is responsible for transmitting the reset sequence, which consists of the break, a mark after break, and a start code. The Master then transmits up to 512 DMX data slots. When the start code is the null start code, data values from 0 to 255 are valid. Furthermore, each slave must be configured to identify the data slot(s) containing data intended for it. After transmitting the appropriate number of data slots, the DMX master asserts a mark before break. The DMX frame and data slot are shown in Figure 29.

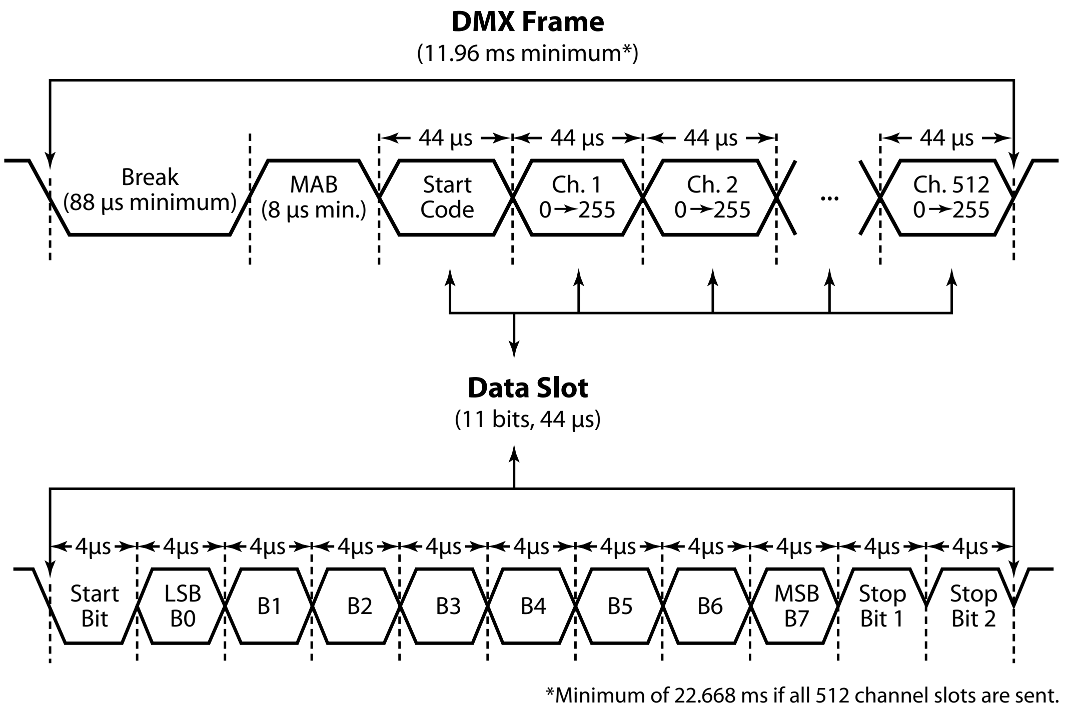


Figure 29. UART-LDD DMX Frame and Data Slot

In DMX Mode, the interrupts defined for normal UART operation still apply, but with the following changes:

- A Parity Error (i.e., the PE bit in the Status 0 Register) is not applicable. Parity Enable (i.e., the PEN bit in the Control 0 Register) is ignored in DMX Mode.
- Framing error checking is not performed; therefore, the framing error status bit (i.e., the FE bit in the Status 0 Register) is reserved in DMX Mode.
- The Break Detect interrupt (i.e, the BRKD bit in the Status 0 Register) indicates when a break is detected by the slave (a break condition for at least 22 bit times). Software can use this interrupt to start a timer checking for lost data input (loss of data tolerance).

14.1.12.1. DMX Clock Requirements

Both a DMX Master and DMX slaves are required to have a nominal 250 kbit/s bit rate with a tolerance of ±2%. Before sending/receiving messages, the Baud Reload High/Low registers must be initialized.

14.1.12.2. DMX Mode Initialization and Operation

DMX Protocol Mode is selected by setting either the DMXMST (DMX master) or DMX-SLV (DMX slave) in the DMX Control Register. To access the DMX Control Register, the Mode Select (MSEL) field of the UART-LDD Mode Select/Status Register must be=`101b`. The UART-LDD Control 0 Register must be initialized. For DMX Master Mode operation, configure TEN=1, STOP=1, and all other bits=0. For DMX slave operation, configure REN=1, STOP=1, and all other bits=0.

14.1.12.3. DMX Master Mode Operation

When the MSEL= `101b` (Mode Select) in the DMX Control Register, DMX Master Mode is selected by setting DMXMST=1 and DMXSLV=0 in the DMX Control Register.

The break is the first part of the DMX protocol transmitted by the master. Hardware can be selected to generate a break consisting of 24 bit periods of logical zero on the DMX bus by first setting AUTOBRK in the DMX Control Register with SBRK in the Control 0 Register cleared. The duration of the break is timed by hardware, and AUTOBRK is deasserted by hardware when the break is completed. Alternatively, if it is necessary to generate a break longer than 24 bit times, a break can be sent manually by first clearing AUTOBRK in the DMX Control Register, then setting SBRK in the Control 0 Register, waiting the appropriate duration, then clearing SBRK. In both cases, UART-LDD starts the break after AUTOBRK or SBRK is set, and any character transmission currently underway has completed.

The mark after break is transmitted automatically for four bit times at the conclusion of the break.

The start code should be written to the Transmit Data Register prior to the conclusion of a mark after break transmission. The start code can be written while generating either the mark before break, the break, or the mark after break, as long as TDRE=1 before writing.

If the Transmit Data Register is written after TDRE asserts but before TXE asserts, the hardware will transmit the character in the Transmit Data Register during the next slot. During each slot, the DMX master will insert a start bit prior to transmitting the character, and will insert two stop bits between each character if STOP=1 in the Control 0 Register. If TXE asserts before the next character is written in the Transmit Data Register, the DMX Master will transmit a mark before break.

14.1.12.4. DMX Slave Operation

When the MSEL= `101b` (Mode Select) in the DMX Control Register, DMX Slave Mode is selected by setting DMXMST=0 and DMXSLV=1. The DMX slave detects the start of a new message by the break which appears to the slave as a break of at least 22 bit times in duration.

Following the break, the UART-LDD hardware automatically checks for a mark after a break lasting at least one bit time. When a mark after this break is detected, hardware will wait for the first slot to be transmitted, which commences with the start bit of the start code. The UART-LDD will also check for the break condition.

The UART-LDD DMX slave decodes the start code and, if it is the null start code, the UART-LDD counts data slots received until the received data slot number matches the value of COMP_ADDR stored in the Comparison Address Register and the DMX Control Register. The slave then receives the data slots. While the REN bit in the UCLT0 Register remains set, the slave receives data slots until a mark before the break is received, and also receives the first byte of the break. After receiving the data slots assigned to the slave, clearing the REN bit in the UCTL0 Register and then setting REN prevents further data reception and associated interrupts until after the next valid break. DMX receive status information is provided in the MODESTAT field of the UART-LDD Mode Select and Status Register.

The UART-LDD can be configured to generate an interrupt upon all received characters, or only upon characters received in data slots equal to or greater than the COMP_ADDR. When the characters in the data slots assigned to the slave have been received, a Wait for Break (WFBRK) can be set to inhibit further receive interrupts until after the next break. To learn more about DMX slave interrupt options, see the Receiver Interrupts section on page 202. Even if WFBRK is set, the Receive Data Register will continue to receive if the REN bit was not temporarily cleared after receiving the data slots assigned to the slave. In this case, a software handler based on the mode status bits in the UMDSTAT Register should discard the Receive Data Register contents between the time the DMX break condition is detected and the reception of a start code is indicated.

14.1.12.5. DMX Slave During Stop Mode

While the DMX bus is in a mark after break condition, the DMX slave can either be in low-power Stop Mode, in Halt Mode, or in a normal operational state. While the slave is in Stop Mode, the UART-LDD is not active, and the break condition must be detected by a GPIO edge-detect Stop-Mode Recovery. A High-to-Low transition on the port pin can be used to bring the device out of Stop Mode. When Stop-Mode Recovery is completed, if enabled, the UART-LDD will time the duration of the break. If the UART-LDD is disabled, the duration of the break can be timed by software. The duration of the Stop-Mode Recovery sequence can preclude making an accurate measurement of a break duration.

14.1.13. UART-LDD Interrupts

The UART-LDD features separate interrupts for the transmitter and receiver. In addition, when the UART-LDD primary functionality is disabled, the Baud Rate Generator can also function as a basic timer with interrupt capability.

14.1.13.1. Transmitter Interrupts

The transmitter generates a single interrupt when the Transmit Data Register Empty (TDRE) bit is set to 1. This interrupt indicates that the transmitter is ready to accept new data for transmission. The TDRE interrupt occurs when the transmitter is initially enabled, and after the Transmit Shift Register has shifted out the first bit of a character. At this point, the Transmit Data Register can be written with the next character to send. As a result of this write, 7 bit periods of latency are provided to load the Transmit Data Register before the Transmit Shift Register completes shifting the current character. Writing to the UART-LDD Transmit Data Register clears the TDRE bit to 0.

In addition, and while transmitting, the UART-LDD can detect a physical layer error (PLE) for LIN Protocol Mode and a collision error (CLSN) for DALI Protocol Mode. The UART-LDD will generate an interrupt if PLE is detected and if CLSN is detected while CLSNE is set.

14.1.13.2. Receiver Interrupts

The receiver generates an interrupt when any one of the following issues occur:

- A data byte has been received and is available in the UART-LDD Receive Data Register. This interrupt can be disabled independently of the other receiver interrupt sources via the RDAIRQ bit in the Multiprocessor Control Register. The received data interrupt occurs after the receive character has been placed in the Receive Data Register. Software must respond to this received data available condition before the next character is completely received to avoid an overrun error.

Note: In Multiprocessor Mode (MPEN = 1), the receive-data interrupts are dependent on the multiprocessor configuration and the most recent address byte.

- A break is received
- A receive data overrun or LIN slave autobaud overrun error is detected
- A data framing error is detected
- A parity error is detected (e.g., if a physical layer error in LIN Mode occurs, software must disable parity error checking for DMX Mode)
- In DALI Mode, a collision error is detected while CLSNE is set in the DALI Control Register
- In DMX Mode, the following slave receive data interrupt events as selected by DMXSIRQ while RDAIRQ is cleared and WFBRK is cleared:
  - Interrupt following start code reception and each received byte.
  - Interrupt following each received byte in a slot ≥ the slave address only if the first received byte was the null start.
  - Interrupt following start code reception.
  - Interrupt following start code reception and each received byte in a slot ≥ the slave address only if the first received byte was the null start.

14.1.13.3. UART-LDD Overrun Errors

When an overrun error condition occurs, the UART-LDD prevents overwriting of the valid data currently in the Receive Data Register. The break detect and overrun status bits are not available until after the valid data has been read.

After the valid data has been read, the OE bit of the Status 0 Register is updated to indicate the overrun condition (and break detect, if applicable). The RDA bit is set to 1 to indicate that the Receive Data Register contains a data byte. However, because the overrun error occurred, this byte cannot contain valid data, and must be ignored. A BRKD bit indicates if the overrun is caused by a break condition on the line. After reading a status byte indicating an overrun error, the Receive Data Register must be read again to clear the error bits in the UART-LDD Status 0 Register.

In LIN Mode, an overrun error is signalled for receive-data overruns as described above, and in the LIN slave if the BRG Counter overflows during the autobaud sequence (the ATB bit will also be set in this case). There is no data associated with the autobaud overflow interrupt; however the Receive Data Register must be read to clear the OE bit. In this case, software must write a `10b` to the LinState field, forcing the LIN slave back to a Wait for Break state.

14.1.13.4. UART-LDD Data- and Error-Handling Procedure

Figures 30 shows the recommended procedure for use in UART-LDD receiver interrupt service routines.

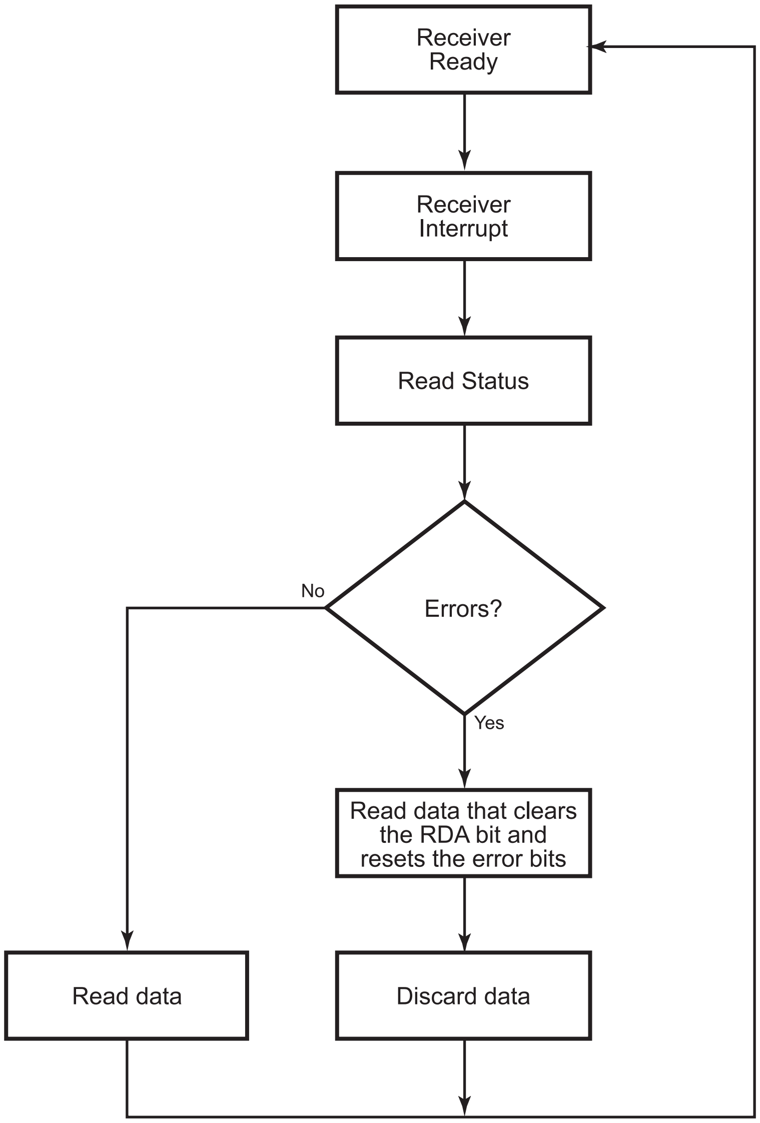


Figure 30. UART-LDD Receiver Interrupt Service Routine Flow

14.1.13.5. Baud Rate Generator Interrupts

If the BRGCTL bit of the Multiprocessor Control Register (UART-LDD Control 1 Register with MSEL=`000b`) is set and the REN bit of the Control 0 Register is 0. The UART-LDD Receiver interrupt asserts when the UART-LDD Baud Rate Generator reloads. This action allows the Baud Rate Generator to function as an additional counter, if the UART-LDD receiver functionality is not employed. The transmitter can be enabled in this mode.

14.1.14. UART-LDD Baud Rate Generator

The UART-LDD Baud Rate Generator creates a lower frequency baud rate clock for data transmission. The input to the Baud Rate Generator is the system clock. The UART-LDD Baud Rate High and Low Byte registers combine to create a 16-bit baud rate divisor value (BRG[15:0]) that sets the data-transmission rate (baud rate) of the UART-LDD. The UART-LDD data rate for normal UART operation and DMX operation is calculated using the following equation:

$$\text{UART Data Rate (bits/s)} = \frac{\text{System Clock Frequency (Hz)}}{\text{16 x UART Baud Rate Divisor Value}}$$

The UART-LDD data rate for LIN Mode UART operation is calculated using the following equation:

$$\text{UART Data Rate (bits/s)} = \frac{\text{System Clock Frequency (Hz)}}{\text{UART Baud Rate Divisor Value}}$$

The UART-LDD data rate for DALI Mode operation is calculated using the following equation:

$$\text{UART Data Rate (bits/s)} = \frac{\text{System Clock Frequency (Hz)}}{\text{32 x UART Baud Rate Divisor Value}}$$

When the UART-LDD is disabled, the BRG functions as a basic 16-bit timer with interrupt on time-out. To configure the BRG as a timer with interrupt on time-out, follow the procedure below:

1. Disable the UART-LDD receiver by clearing the REN bit in the UART-LDD Control 0 Register to 0 (i.e., the TEN bit can be asserted; transmit activity can occur).
2. Load the appropriate 16-bit count value into the UART-LDD Baud Rate High and Low Byte registers.
3. Enable the BRG timer function and the associated interrupt by setting the BRGCTL bit in the UART-LDD Control 1 Register to 1.

14.2. Noise Filter

An included noise filter circuit filters noise on a digital input signal (such as UART Receive Data) before data is sampled by the block. This circuit is likely to be a requirement for protocols that will operate within a noisy environment.

The noise filter contains the following features:

- Synchronizes the receive input data to the System Clock
- Noise Filter Enable (NFEN) input selects whether the noise filter is bypassed (NFEN=0) or included (NFEN=1) in the receive data path
- Noise Filter Control (NFCTL[2:0]) input selects the width of the up/down saturating counter digital filter; the available width ranges from 4 to 11 bits
- The digital filter output features hysteresis
- Provides an active-Low *Saturated State* output, FiltSatB, which is used as an indication of the presence of noise

14.2.1. Architecture

Figure 31 shows an example of how the noise filter is integrated with the UART-LDD on a LIN network.

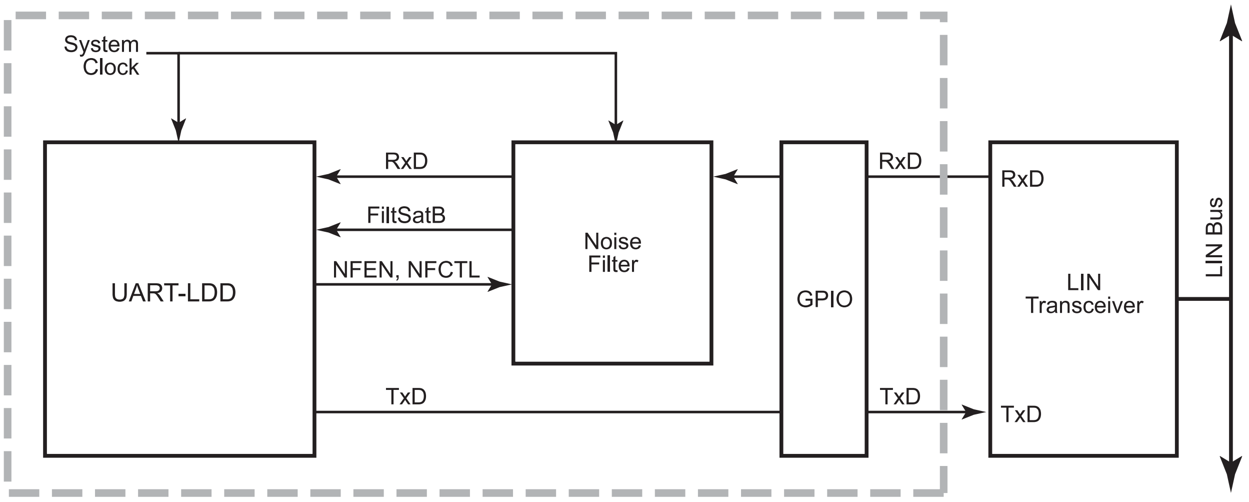


Figure 31. Noise Filter System Block Diagram

14.2.2. Operation

Figure 32 shows the operation of the noise filter both with and without noise. The noise filter in this example is a 2-bit up/down counter which saturates at `00b` and `11b`. A 2-bit counter is shown for convenience; the operation of wider counters is similar. The output of the filter switches from 1 to 0, when the counter counts down from `01b` to `00b`; this output switches from 0 to 1 when the counter counts up from `10b` to `11b`.

In this 2-bit counter example, the noise filter delays the receive data by three System Clock cycles. The receive data delay induced by the noise filter is a function of the counter width:

> Filter delay= 2^n - 1 Sytem Clock cycles, where n is the counter width value selected by NFCTL.

The FiltSatB signal is checked when the filtered RxD is sampled in the center of the bit time. The presence of noise (FiltSatB=1 at the center of the bit time) does not mean that the sampled data is incorrect; instead, the filter is not in its *saturated* state of all ones or all zeroes. If FiltSatB=1, then RxD is sampled during a receive character and the NE bit in the ModeStatus[4:0] field is set. By observing this bit, an indication of the level of noise in the network can be obtained.

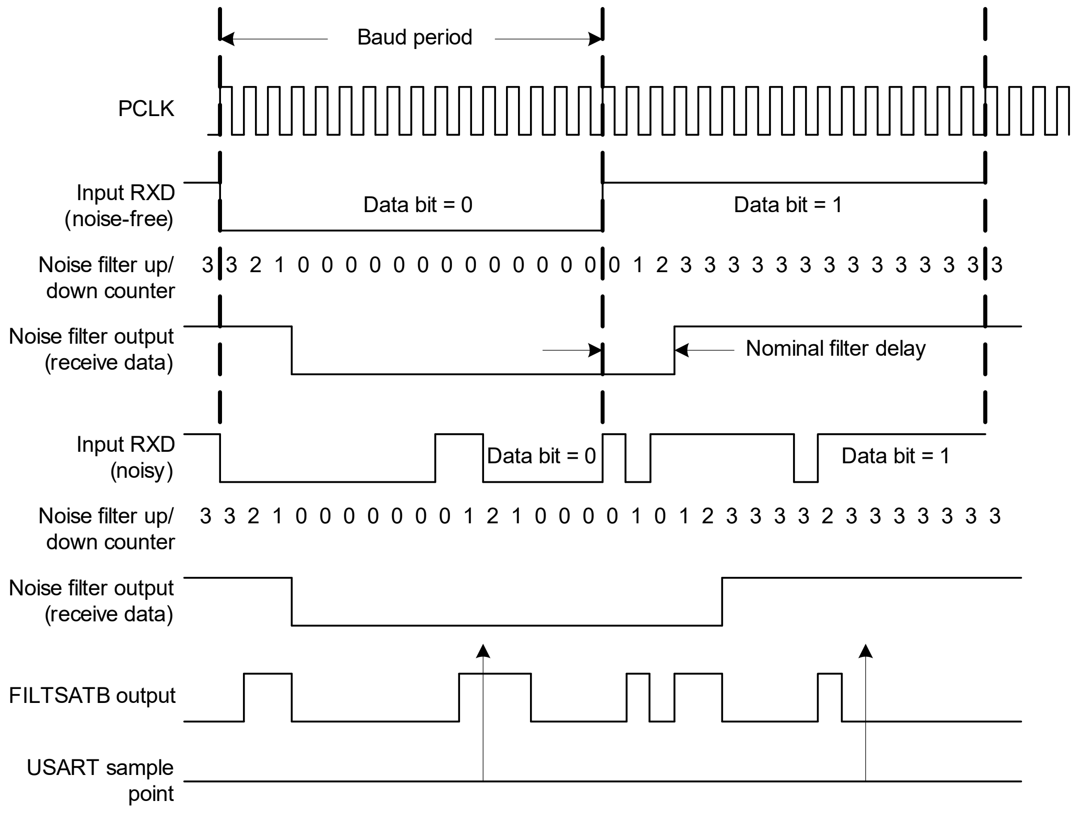


Figure 32. Noise Filter Operation

14.3. UART-LDD Control Register Definitions

The UART-LDD control registers support the UART-LDD and the noise filter.

14.3.1. UART-LDD Transmit Data Register

Data bytes written to the UART-LDD Transmit Data Register, shown in Table 107, are shifted out on the TxD pin. This write-only register shares a Register File address with the read-only UART-LDD Receive Data Register.

Table 107. UART-LDD Transmit Data Registers (UTXD)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	TxD							
Reset	X	X	X	X	X	X	X	X
R/W	W	W	W	W	W	W	W	W
Address	F40h							

Note: W=Write; X=undefined; *x* = 0,1.

Bit	**Description**
[7:0] TxD	**Transmit Data** UART-LDD transmitter data byte to be shifted out through the TxD pin.

14.3.2. UART-LDD Receive Data Register

Data bytes received through the RxD pin are stored in the UART-LDD Receive Data Register, as shown in Table 108. This read-only register shares a Register File address with the write-only UART-LDD Transmit Data Register.

Table 108. UART-LDD Receive Data Registers (U*x*RXD)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	RxD							
Reset	X	X	X	X	X	X	X	X
R/W	R	R	R	R	R	R	R	R
Address	F40h							

Note: R=read; X=undefined; *x* = 0,1.

Bit	**Description**
[7:0] RxD	**Receive Data** UART-LDD receiver data byte from the RxD pin.

14.3.3. UART-LDD Status 0 Register

The UART-LDD Status 0 register identifies the current UART-LDD operating configuration and status. Table 109 describes the Status 0 register for standard UART Mode. The Status 0 register is described for LIN Mode in Table 110, for DALI Mode in Table 111, and for DMX Mode in Table 112.

Table 109. UART-LDD Status 0 Register, Standard UART Mode (USTAT0)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	RDA	PE	OE	FE	BRKD	TDRE	TXE	CTS
Reset	0	0	0	0	0	1	1	X
R/W	R	R	R	R	R	R	R	R
Address	F41h							

Note: R = read; X = undefined; *x* = 0,1.

Bit	**Description**
[7] RDA	**Receive Data Available** This bit indicates that the UART-LDD Receive Data Register has received data. Reading the UART-LDD Receive Data Register clears this bit. 0: The UART-LDD Receive Data Register is empty. 1: There is a byte in the UART-LDD Receive Data Register.
[6] PE	**Parity Error** This bit indicates that a parity error has occurred. Reading the Receive Data Register clears this bit. 0: No parity error occurred. 1: A parity error occurred.
[5] OE	**Overrun Error** This bit indicates that an overrun error has occurred. An overrun occurs when new data is received and the Receive Data Register is not read. Reading the Receive Data Register clears this bit. 0: No overrun error occurred. 1: An overrun error occurred.
[4] FE	**Framing Error** This bit indicates that a framing error (no stop bit following data reception) was detected. Reading the Receive Data Register clears this bit. 0: No framing error occurred. 1: A framing error occurred.
[3] BRKD	**Break Detect** This bit indicates that a break occurred. If the data bits, parity/multiprocessor bit and stop bit(s) are all zeroes, then this bit is set to 1. Reading the Receive Data Register clears this bit. 0: No break occurred. 1: A break occurred.

Bit	Description (Continued)
[2] TDRE	**Transmitter Data Register Empty** This bit indicates that the Transmit Data Register is empty and ready for additional data. Writing to the Transmit Data Register resets this bit. 0: Do not write to the Transmit Data Register. 1: The Transmit Data Register is ready to receive an additional byte for transmission.
[1] TXE	**Transmitter Empty** This bit indicates that the Transmit Shift Register is empty and character transmission is finished. 0: Data is currently transmitting. 1: Transmission is complete.
[0] CTS	**Clear to Send Signal** When this bit is read it returns the level of the $\overline{\text{CTS}}$ signal. If LBEN=1, the $\overline{\text{CTS}}$ input signal is replaced by the internal Receive Data Available signal to provide flow control in a loopback mode. CTS only affects transmission if the CTSE bit=1.

Table 110. UART-LDD Status 0 Register, LIN Mode (USTAT0)

Bit	7	6	5	4	3	2	1	0
Field	RDA	PLE	OE	FE	BRKD	TDRE	TXE	ATB
Reset	0	0	0	0	0	1	1	0
R/W	R	R	R	R	R	R	R	R
Address	F41h							

Note: R=read; *x* = 0,1.

Bit	Description
[7] RDA	**Receive Data Available** This bit indicates that the Receive Data Register has received data. Reading the Receive Data Register clears this bit. 0: The Receive Data Register is empty. 1: There is a byte in the Receive Data Register.
[6] PLE	**Physical Layer Error** This bit indicates that transmit and receive data do not match when a LIN slave or master is transmitting. This could be by a fault in the physical layer or multiple devices driving the bus simultaneously. Reading the Status 0 Register or the Receive Data Register clears this bit. 0: Transmit and Receive data match. 1: Transmit and Receive data do not match.

Bit	Description (Continued)
[5] OE	**Receive Data and Autobaud Overrun Error** This bit is set just as in normal UART operation if a receive data overrun error occurs. This bit is also set during LIN slave autobaud if the BRG counter overflows before the end of the autobaud sequence. This indicates that the receive activity is not an autobaud character or the master baud rate is too slow. The ATB status bit will also be set in this case. This bit is cleared by reading the Receive Data Register. 0: No autobaud or data overrun error occurred. 1: An autobaud or data overrun error occurred.
[4] FE	**Framing Error** This bit indicates that a framing error (no stop bit following data reception) is detected. Reading the Receive Data Register clears this bit. 0: No framing error occurred. 1: A framing error occurred.
[3] BRKD	**Break Detect** This bit is set in LIN Mode if: • It is in LIN Sleep state and a break of at least 4 bit times occurred (wake-up event) or • It is in Slave Wait Break state and a break of at least 11 bit times occurred (break event) or • It is in Slave Active state and a break of at least 10 bit times occurs. Reading the Status 0 Register or the Receive Data Register clears this bit. 0: No LIN break occurred. 1: LIN break occurred.
[2] TDRE	**Transmitter Data Register Empty** This bit indicates that the Transmit Data Register is empty and ready for additional data. Writing to the Transmit Data Register resets this bit. 0: Do not write to the Transmit Data Register. 1: The Transmit Data Register is ready to receive an additional byte for transmission.
[1] TXE	**Transmitter Empty** This bit indicates that the Transmit Shift Register is empty and character transmission is completed. 0: Data is currently transmitting. 1: Transmission is complete.
[0] ATB	**LIN Slave Autobaud Complete** This bit is set in LIN Slave Mode when an autobaud character is received. If the ABIEN bit is set in the LIN Control Register, then a receive interrupt is generated when this bit is set. Reading the Status 0 Register clears this bit. This bit will be 0 in LIN Master Mode.

Table 111. UART-LDD Status 0 Register, DALI Mode (USTAT0)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	RDA	BPE	OE	FE	CLSN	TDRE	TXE	CTS
Reset	0	0	0	0	0	1	1	X
R/W	R	R	R	R	R	R	R	R
Address	F41h							

Note: R=read; X=undefined; x = 0,1.

Bit	Description
[7] RDA	**Receive Data Available** This bit indicates that the UART-LDD Receive Data Register has received data. Reading the UART-LDD Receive Data Register clears this bit. 0: The UART-LDD Receive Data Register is empty. 1: There is a byte in the UART-LDD Receive Data Register.
[6] BPE	**Biphase Error** This bit indicates that a biphase error has occurred. Reading the Receive Data Register clears this bit. 0: No biphase error occurred. 1: A biphase error occurred. Biphase format data was expected, but both phases had the same value. This bit is set only if BPEN=1.
[5] OE	**Overrun Error** This bit indicates that an overrun error has occurred. An overrun occurs when new data is received and the Receive Data Register is not read. Reading the Receive Data Register clears this bit. 0: No overrun error occurred. 1: An overrun error occurred.
[4] FE	**Framing Error** This bit indicates that a framing error (no stop bit following data reception) was detected.Checking occurs only for single-byte transfers (MULTRXE=0 in the DALI Control Register). Reading the Receive Data Register clears this bit. 0: No framing error occurred. 1: A framing error occurred.
[3] CLSN	**Collision Detect Error** This bit indicates that a collision was detected. Reading the Receive Data Register clears this bit. 0: No collision was detected. 1: A collision was detected.
[2] TDRE	**Transmitter Data Register Empty** This bit indicates that the Transmit Data Register is empty and ready for additional data. Writing to the Transmit Data Register resets this bit. 0: Do not write to the Transmit Data Register. 1: The Transmit Data Register is ready to receive an additional byte for transmission.

Bit	Description (Continued)
[1] TXE	**Transmitter Empty** This bit indicates that the Transmit Shift Register is empty and character transmission is finished. 0: Data is currently transmitting. 1: Transmission is complete.
[0] CTS	**Clear to Send Signal** When this bit is read it returns the level of the $\overline{\text{CTS}}$ signal. If LBEN=1, the $\overline{\text{CTS}}$ input signal is replaced by the internal Receive Data Available signal to provide flow control in a loopback mode. CTS only affects transmission if the CTSE bit=1.

Table 112. UART-LDD Status 0 Register, DMX Mode (USTAT0)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	RDA	Reserved	OE	Reserved	BRKD	TDRE	TXE	CTS
Reset	0	0	0	0	0	1	1	X
R/W	R	R	R	R	R	R	R	R
Address	F41h							

Note: R=read; X=undefined; x = 0,1.

Bit	Description
[7] RDA	**Receive Data Available** This bit indicates that the UART-LDD Receive Data Register has received data. Reading the UART-LDD Receive Data Register clears this bit. 0: The UART-LDD Receive Data Register is empty. 1: There is a byte in the UART-LDD Receive Data Register.
[6]	**Reserved** This bit is reserved and must be programmed to 0.
[5] OE	**Overrun Error** This bit indicates that an overrun error has occurred. An overrun occurs when new data is received and the Receive Data Register is not read. Reading the Receive Data Register clears this bit. 0: No overrun error occurred. 1: An overrun error occurred.
[4]	**Reserved** This bit is reserved and must be programmed to 0.
[3] BRKD	**Break Detect** This bit indicates that a break occurred. If the break condition exists for at least 22 bit times (all bits are zero) then this bit is set to 1. Reading the Receive Data Register clears this bit. 0: No break occurred. 1: A break occurred.

Bit	Description (Continued)
[2] TDRE	**Transmitter Data Register Empty** This bit indicates that the Transmit Data Register is empty and ready for additional data. Writing to the Transmit Data Register resets this bit. 0: Do not write to the Transmit Data Register. 1: The Transmit Data Register is ready to receive an additional byte for transmission.
[1] TXE	**Transmitter Empty** This bit indicates that the Transmit Shift Register is empty and character transmission is finished. 0: Data is currently transmitting. 1: Transmission is complete.
[0] CTS	**Clear to Send Signal** When this bit is read it returns the level of the $\overline{\text{CTS}}$ signal. If LBEN=1, the $\overline{\text{CTS}}$ input signal is replaced by the internal Receive Data Available signal to provide flow control in a loopback mode. CTS only affects transmission if the CTSE bit=1.

14.3.4. UART-LDD Mode Select and Status Register

The UART-LDD Mode Select and Status register, shown in Table 113, contains mode select and status bits.

Table 113. UART-LDD Mode Select and Status Register (UMDSTAT)

Bit	7	6	5	4	3	2	1	0
Field	MSEL			MODESTAT				
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R	R	R	R	R
Address	F44h							

Note: R=read; R/W=read/write; *x* = 0,1.

Bit	Description
[7:5] MSEL	**Mode Select** This read/write field determines which control register is accessed when performing a write or read to the UART Control 1 Register address. This field also determines which status is returned in the Mode Status field when reading this register. 000: Multiprocessor and normal UART control/status. 001: Noise filter control/status. 010: LIN protocol control/status. 011: Reserved. 100: DALI protocol/status. 101: DMX protocol/status. 110: Reserved. 111: Reserved.

Bit	Description (Continued)
[4:0] MODESTAT	**Mode Status** This read-only field returns status corresponding to one of four modes selected by MSEL. These four modes are described in Table 114 on page 215. When MSEL[2:0] is set to: 000: Multiprocessor Mode status={NE,0,0,NEWFRM, MPRX}. 001: Noise filter status={NE,0,0,0,0}. 010: LIN Mode status={NE, RxBreakLength}. 011: Reserved. 100: DALI Mode status={RDA, CLSN,ADDRD}. 101: DMX Mode status={RDA, NullStartCode, NonNullStartCode, SlvAddrMatch, MABState}. 110: Reserved. 111: Reserved

Table 114. Mode Status Fields

Multiprocessor Mode Status Field; MSEL=000b	**New Frame (NEWFRM)** Status bit denoting the start of a new frame. Reading the UART-LDD Receive Data Register resets this bit to 0. 0: The current byte is not the first data byte of a new frame. 1: The current byte is the first data byte of a new frame. **Multiprocessor Receive (MPRX)** Returns the value of the last multiprocessor bit received. Reading from the UART-LDD Receive Data Register resets this bit to 0. **Noise Event (NE)** This bit is asserted if digital noise is detected on the receive data line when the data is sampled (center of bit-time). If this bit is set, it does not mean that the receive data is corrupted (though it can be in extreme cases), means that one or more of the noise filter data samples near the center of the bit-time did not match the average data value.
Digital Noise Filter Mode Status Field; MSEL=001b	**Noise Event (NE)** See the NE description in this table for Multiprocessor Mode Status Field.
LIN Mode Status Field; MSEL=010b	**Noise Event (NE)** See description in this table for Multiprocessor Mode Status Field. **RxBreakLength** LIN Mode received break length. This field can be read following a break (LIN wake-up or break) so that the software can determine the measured duration of the break. If the break exceeds 15 bit times the value saturates at 1111b.

Table 114. Mode Status Fields

DALI Mode Status Field; MSEL=100b	**Receive Data Available (RDA)** This bit is identical to RDA in the UART-LDD Status 0 Register. **Collision (CLSN)** This bit is identical to CLSN in the UART-LDD Status 0 Register in DALI Mode. **Address Detect (ADDRD)** This field indicates the address detected by the UART-LDD block for the current message. 001: Short address match. 010: Group address. 100: Broadcast. 101: Special or unrecognized command. All other bits are reserved.
DMX Mode Status Field; MSEL=101b	**Receive Data Available (RDA)** This bit is identical to RDA in the UART-LDD Status 0 Register. **Null Start Code Received (NullStartCode)** This bit is asserted upon detecting a null start code and is cleared when reception of the frame ends. **Non-Null Start Code Received (NonNullStartCode)** This bit is asserted upon detecting a non-null start code and is cleared when reception of the frame ends. **Slave Address Match (SlvAddrMatch)** This bit is asserted upon detecting slave address match and is cleared when reception of the frame ends. **Mark Before/After Break Condition (MABState)** This bit is asserted while the Mark Before/After Break Condition is detected.

14.3.5. UART-LDD Control 0 Register

The UART-LDD Control 0 register, shown in Table 115, configures the basic properties of UART-LDD's transmit and receive operations. A more detailed discussion of each bit follows the table.

Table 115. UART-LDD Control 0 Register (UCTL0)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	TEN	REN	CTSE	PEN	PSEL	SBRK	STOP	LBEN
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F42h							

Note: R/W=read/write; *x* = 0,1.

Bit	Description
[7] TEN	**Transmit Enable** This bit enables or disables the transmitter. The enable is also controlled by the $\overline{\text{CTS}}$ signal and the CTSE bit. If the $\overline{\text{CTS}}$ signal is Low and the CTSE bit is 1, the transmitter is enabled. 0: Transmitter disabled. 1: Transmitter enabled.
[6] REN	**Receive Enable** This bit enables or disables the receiver. 0: Receiver disabled. 1: Receiver enabled.
[5] CTSE	**Clear To Send Enable** 0: The $\overline{\text{CTS}}$ signal has no effect on the transmitter. 1: The UART-LDD recognizes the $\overline{\text{CTS}}$ signal as an enable control for the transmitter.
[4] PEN	**Parity Enable** This bit enables or disables parity and should be cleared for DALI (MSEL=100) and DMX (MSEL=101) modes. Even or odd is determined by the PSEL bit. 0: Parity is disabled. This bit is overridden by the MPEN bit. 1: The transmitter sends data with an additional parity bit and the receiver receives an additional parity bit.
[3] PSEL	**Parity Select** 0: Even parity is sent as an additional parity bit for the transmitter/receiver. 1: Odd parity is sent as an additional parity bit for the transmitter/receiver.

Bit	Description (Continued)
[2] SBRK	**Send Break** This bit pauses or breaks data transmission. Sending a break interrupts any transmission in progress, so ensure that the transmitter has completed sending data before setting this bit. In standard UART Mode, the duration of the break is determined by how long the software leaves this bit asserted. Also the duration of any required stop bits following the break must be timed by software before writing a new byte to be transmitted to the Transmit Data Register. In LIN Mode, the master sends a break character by asserting SBRK. The duration of the break is timed by hardware and the SBRK bit is deasserted by hardware when the break is completed. The duration of the break is determined by the TxBreakLength field of the LIN Control Register. One or two stop bits are automatically provided by the hardware in LIN Mode, as defined by the stop bit. In DALI Mode and DMX Mode, this bit pauses or breaks data transmission just as in standard UART Mode. In DMX Mode, hardware can time the duration of the break if AUTOBRK is set in the DMX Control Register. 0: No break is sent. 1: A break is sent (the output of the transmitter is 0).
[1] STOP	**Stop Bit Select** 0: The transmitter sends one stop bit. The receiver framing error check expects one stop bit. 1: The transmitter sends two stop bits. The receiver framing error check expects two stop bits.
[0] LBEN	**Loop Back Enable** 0: Normal operation. 1: All transmitted data is looped back to the receiver.

14.3.6. UART-LDD Control 1 Registers

Multiple registers are accessible by a single bus address. The register selected is determined by the Mode Select (MSEL) field. These registers provide additional control over UART-LDD operation.

14.3.6.1. Multiprocessor Control Register

When MSEL=`000b`, the Multiprocessor Control Register, shown in Table 116, provides control for UART Multiprocessor Mode and Baud Rate Timer Mode, as well as other features that can apply to multiple modes.

Table 116. Multiprocessor Control Register (UCTL1 with MSEL=000b)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	MPMD1	MPEN	MPMD0	MPBT	DEPOL	BRGCTL	RDAIRQ	Reserved
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R
Address	F43h							

Note: R=read; R/W=read/write; *x* = 0,1.

Bit	Description
[7,5] MPMD[1:0]	**Multiprocessor (9-Bit) Mode** 00: The UART-LDD generates an interrupt request on all data and address bytes. 01: The UART-LDD generates an interrupt request only on received address bytes. 10: The UART-LDD generates an interrupt request when a received address byte matches the value stored in the Address Compare Register and on all successive data bytes until an address mismatch occurs. 11: The UART-LDD generates an interrupt request on all received data bytes for which the most recent address byte matched the value in the Address Compare Register.
[6] MPEN	**Multiprocessor Enable** This bit is used to enable Multiprocessor (9-bit) Mode. 0: Disable Multiprocessor (9-bit) Mode. 1: Enable Multiprocessor (9-bit) Mode.
[4] MPBT	**Multiprocessor Bit Transmit** This bit is applicable only when Multiprocessor (9-bit) Mode is enabled. 0: Send a 0 in the multiprocessor bit location of the data stream (9th bit). 1: Send a 1 in the multiprocessor bit location of the data stream (9th bit).
[3] DEPOL	**Driver Enable Polarity** 0: DE signal is active High. 1: DE signal is active Low.

Bit	Description (Continued)
[2] BRGCTL	**Baud Rate Generator Control** This bit causes different UART-LDD behavior depending on whether the UART-LDD receiver is enabled (REN = 1 in the UART-LDD Control 0 Register). When the UART-LDD receiver is not enabled, this bit determines whether the Baud Rate Generator issues interrupts. When the UART-LDD receiver is enabled, this bit allows Reads from the baud rate registers to return the BRG count value instead of the reload value. When the UART-LDD receiver is not enabled: 0: BRG is disabled. Reads from the Baud Rate High and Low Byte registers return the BRG reload value. 1: BRG is enabled and counting. The Baud Rate Generator generates a receive interrupt when it counts down to 0. Reads from the Baud Rate High and Low Byte registers return the current BRG count value. When the UART-LDD receiver is enabled: 0: Reads from the Baud Rate High and Low Byte registers return the BRG reload value. 1: Reads from the Baud Rate High and Low Byte registers return the current BRG count value. Unlike the timers, there is no mechanism to latch the High Byte when the Low Byte is read.
[1] RDAIRQ	**Receive Data Interrupt** 0: Received data and receiver errors generates an interrupt request to the Interrupt controller. Note that RDAIRQ also affects DALI and DMX modes. In DMX Mode, the received data interrupts are governed by DMXSIRQ. 1: Received data does not generate an interrupt request to the Interrupt controller. Only receiver errors generate an interrupt request.
[0]	**Reserved** This bit is reserved and must be programmed to 0.

14.3.7. Noise Filter Control Registers

When MSEL=`001b`, the Noise Filter Control Register, shown in Table 117, provides control for the digital noise filter.

Table 117. Noise Filter Control Register (UCTL1 with MSEL=001b)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	NFEN	NFCTL			Reserved			
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R	R	R	R
Address	F43h							

Note: R=read; R/W=read/write; *x* = 0,1.

Bit	Description
[7] NFEN	**Noise Filter Enable** 0: Noise filter is disabled. 1: Noise filter is enabled. Receive data is preprocessed by the noise filter.
[6:4] NFCTL	**Noise Filter Control** This field controls the delay and noise rejection characteristics of the noise filter. The wider the counter is, the more delay is introduced by the filter and the wider the noise event is filtered. 000: 4-bit up/down counter. 001: 5-bit up/down counter. 010: 6-bit up/down counter. 011: 7-bit up/down counter. 100: 8-bit up/down counter. 101: 9-bit up/down counter. 110: 10-bit up/down counter. 111: 11-bit up/down counter.
[3:0]	**Reserved** These bits are reserved and must be programmed to 0000.

14.3.8. LIN Control Register

When MSEL=`010b`, the LIN Control registers provides control for the LIN Mode of operation.

Table 118. LIN Control Register (UCTL1 with MSEL=010b)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	LMST	LSLV	ABEN	ABIEN	LinState[1:0]		TxBreakLength	
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F43h							

Note: R/W=read/write; *x* = 0,1.

Bit	**Description**
[7] LMST	**LIN Master Mode** 0: LIN Master Mode not selected. 1: LIN Master Mode selected (if MPEN, PEN, LSLV=0).
[6] LSLV	**LIN Slave Mode** 0: LIN Slave Mode not selected. 1: LIN Slave Mode selected (if MPEN, PEN, LMST=0).
[5] ABEN	**Autobaud Enable** 0: Autobaud not enabled. 1: Autobaud enabled, if in LIN Slave Mode.
[4] ABIEN	**Autobaud Interrupt Enable** 0: Interrupt following autobaud does not occur. 1: Interrupt following autobaud enabled, if in LIN Slave Mode. When the autobaud character is received, a receive interrupt is generated and the ATB bit is set in the Status0 Register.
[3:2] LINSTATE[1:0]	**LIN State Machine** The LinState is controlled by both hardware and software. Software can force a state change at any time if necessary. In normal operation, software moves the state in and out of Sleep state. For a LIN slave, software changes the state from Sleep to Wait for Break, after which hardware cycles through the Wait for Break, Autobaud and Active states. Software changes the state from one of the active states to Sleep state, if the LIN bus goes into Sleep Mode. For a LIN master, software changes the state from Sleep to Active, where it remains until the software sets it back to the Sleep state. After configuration, software does not alter the LinState field during operation. 00: Sleep state (either LMST or LSLV can be set). 01: Wait for Break state (only valid for LSLV=1). 10: Autobaud state (only valid for LSLV=1). 11: Active state (either LMST or LSLV can be set).

Bit	Description (Continued)
[1:0] TxBreakLength	**TxBreakLength** Used in LIN Mode by the master to control the duration of the transmitted break. 00: 13 bit times. 01: 14 bit times. 10: 15 bit times. 11: 16 bit times.

14.3.9. DALI Control Register

The DALI Control Register (U*x*CTL1), shown in Table 119, provides control for the DALI Mode of operation.

Table 119. DALI Control Register (UCTL1 with MSEL=100b)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	MULTTXE	MULTRXE	BPEN	BPENC	STRTPOL	BITORD	CLSNE	PARTRXE
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F43h							

Note: R/W=read/write; *x* = 0,1.

Bit	Description
[7] MULTTXE	**Multiple-Byte Transmit Enable** TEN in the UART-LDD Control Register must also be set to enable transmit. 0: Transmit stop bit(s) after each byte transmitted. 1: Transmit multiple bytes without stop bit(s) inserted between the bytes as long as data is available in the UART-LDD Transmit Data Register. Stop bit(s) are transmitted after the last byte is transmitted.
[6] MULTRXE	**Multiple-Byte Receive Enable** REN in the UART-LDD Control Register must also be set to enable receive. 0: Stop bit(s) are expected after each byte received. 1: Multiple bytes can be received without stop bit(s) inserted between the bytes.
[5] BPEN	**Biphase Encoding Enable** 0: Biphase encoding not enabled. 1: Biphase encoding enabled. BPEN should be set for DALI operation.
[4 BPENC	**Biphase Encoding** BPENC has an effect only if BPEN=1. 0: DALI biphase encoding. Logic 0 is encoded as biphase 1→0 and logic 1 is encoded as biphase 0 → 1. 1: Alternate biphase encoding. Logic 0 is encoded as biphase 0→1 and logic 1 is encoded as biphase 1 → 0.

Bit	Description (Continued)
[3] STRTPOL	**Start Bit Polarity** 0: Start bit is a logic 0. 1: Start bit is a logic 1. STRTPOL is typically set for DALI.
[2] BITORD	**Bit Order** 0: Standard UART bit order with the LSB (TxD[0]/RxD[0]) transmitted/received first. 1: DALI bit order with the MSB (TxD[7]/RxD[7]) transmitted/received first.
[1] CLSNE	**Collision Detection Enable** 0: Collision detection is disabled. 1: Collision detection is enabled. CLSNE should be set only for DALI master transmissions. If a collision occurs while CLSNE is set, CLSN will be set in the UART-LDD Status Register and an interrupt will be generated.
[0] PARTRXE	**Partial Byte Reception Enable** PARTRXE has an effect only when receiving. 0: Partial bytes are not loaded into RXDATA. 1: Partial bytes are loaded into RXDATA if a partial byte has been received upon receiving a stop bit.

14.3.10. DMX Control Register

When MSEL=`101b`, the DMX Control Register, shown in Table 120, provides control for the DMX Mode of operation.

Table 120. DMX Control Register (UCTL1 with MSEL=101b)

Bit	7	6	5	4	3	2	1	0
Field	DMXMST	DMXSLV	DMXSIRQ		Reserved	AUTOBRK	WFBRK	COMP_ ADDR[8]
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R	R/W	R/W	R/W
Address	F43h							

Note: R/W=read/write; *x* = 0,1.

Bit	Description
[7] DMXMST	**DMX Master Mode** 0: DMX Master Mode not selected. 1: DMX Master Mode selected.
[6] DMXSLV	**DMX Slave Mode** 0: DMX Slave Mode not selected. 1: DMX Slave Mode selected.

Bit	Description (Continued)
[5:4] DMXSIRQ	**DMX Slave Interrupt Control** DMXSIRQ has an effect only for a DMX slave (DMXSLV = 1) and if both RDAIRQ = 0 and WFBRK = 0. 00: Interrupt following break detect, start code reception, and each received byte. 01: Interrupt following break detect and each received byte in a slot ≥ the slave address only if the first received byte was the null start. 10: Interrupt following break detect and start code reception. 11: Interrupt following break detect, start code reception, and each received byte in a slot ≥ the slave address only if the first received byte was the null start.
[3]	**Reserved** This bit is reserved and must be programmed to 0.
[2] AUTOBRK	**Automatic Break** AUTOBRK has an effect only for a DMX Master (DMXMST = 1). 0: No automatic break transmission. Manually send a break using SBRK 1: Automatic break transmission of 24 bit times (96us at 250 kHz) upon being set. AUTOBRK is cleared by hardware upon completing the break transmission. Clearing AUTOBRK during a break transmission does not terminate the break transmission.
[1] WFBRK	**Wait for Break** WFBRK has an effect only for a DMX slave (DMXSLV = 1). 0: Do not wait for break. Continue to generate receive data interrupts as configured by DMXSIRQ and RDAIRQ. 1: Wait for break. Do not generate received data interrupts until the next break is received. Upon receiving a break, WFBRK is cleared by hardware.
[0] COMP_AD DR[8]	**Comparison Address bit 8** COMP_ADDR[8] has an effect only for a DMX slave (MODE = 101, DMXSLV = 1). 0–1: Combined with COMP_ADDR[7:0] in UART-LDD Address Compare Register to form a 9-bit comparison address. For a DMX slave, the comparison address is compared against the received data slot number.

14.3.11. UART-LDD Address Compare Registers

The UART-LDD Address Compare Register, shown in Table 121, stores the multinode network address of the UART-LDD. When the MPMD[1] bit of the UART-LDD Multiprocessor Control Register is set, all incoming address bytes are compared to the value stored in this Address Compare Register. Receive interrupts and RDA assertions only occur in the event of a match.

When RDAIRQ is cleared, DMXSLV is set, and DMXSIRQ = x1, the data slot number is compared to the value of COMP_ADDR is stored in this address compare register and in the DMX Control Register. Interrupts can be configured to occur upon receiving characters in data slots equal to or greater than the COMP_ADDR value.

Table 121. UART-LDD Address Compare Register (UADDR)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	COMP_ADDR							
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F45h							

Note: R/W=read/write; *x* = 0,1.

Bit	**Description**
[7:0] COMP_ADDR	**Compare Address** This 8-bit value is compared to the incoming address bytes. For DMX slaves, a 9th bit, COMP_ADDR[8] in the DMX Control Register is also used.

14.3.12. UART-LDD Baud Rate High and Low Byte Registers

The UART-LDD Baud Rate High and Low Byte registers, shown in Tables 122 and 123, combine to create a 16-bit baud rate divisor value (BRG[15:0]) that sets the data transmission rate (baud rate) of the UART-LDD.

Table 122. UART-LDD Baud Rate High Byte Register (UBRH)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	BRH							
Reset	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F46h							

Note: R/W=read/write; *x* = 0,1.

Bit	**Description**
[7:0] BRH	**Baud Rate High** These bits set the High byte of the baud rate divisor value.

Table 123. UART-LDD Baud Rate Low Byte Register (UBRL)

Bit	7	6	5	4	3	2	1	0
Field	BRL							
Reset	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F47h							

Note: R/W=read/write; *x* = 0,1.

Bit	Description
[7:0] BRL	**Baud Rate Low** These bits set the Low Byte of the baud rate divisor value.

The UART-LDD data rate is calculated using the following equation for standard UART and DMX modes. For the LIN protocol, the Baud Rate registers must be programmed with the baud period rather than 1/16th of the baud period.

Note: The UART must be disabled when updating the Baud Rate registers because the high and low registers must be written independently.

The UART-LDD data rate is calculated using the following equation for standard UART and DMX Mode operation:

$$\text{UART Data Rate (bits/s)} = \frac{\text{System Clock Frequency (Hz)}}{16 \times \text{UART Baud Rate Divisor Value}}$$

The UART-LDD data rate is calculated using the following equation for LIN Mode UART operation:

$$\text{UART Data Rate (bits/s)} = \frac{\text{System Clock Frequency (Hz)}}{\text{UART Baud Rate Divisor Value}}$$

The UART-LDD data rate is calculated using the following equation for DALI Mode operation:

$$\text{UART Data Rate (bits/s)} = \frac{\text{System Clock Frequency (Hz)}}{32 \times \text{UART Baud Rate Divisor Value}}$$

For a given UART-LDD data rate, the integer baud rate divisor value is calculated using the following equation for standard UART or DMX Mode operation:

$$\text{UART Baud Rate Divisor Value (BRG)} = \text{Round}\left(\frac{\text{System Clock Frequency (Hz)}}{16\text{xUART Data Rate (bits/s)}}\right)$$

For a given UART-LDD data rate, the integer baud rate divisor value is calculated using the following equation for LIN Mode UART operation:

$$\text{UART Baud Rate Divisor Value (BRG)} = \text{Round}\left(\frac{\text{System Clock Frequency (Hz)}}{\text{UART Data Rate (bits/s)}}\right)$$

For a given UART-LDD data rate, the integer baud rate divisor value is calculated using the following equation for DALI Mode operation:

$$\text{UART Baud Rate Divisor Value (BRG)} = \text{Round}\left(\frac{\text{System Clock Frequency (Hz)}}{32\text{xUART Data Rate (bits/s)}}\right)$$

The baud rate error relative to the appropriate baud rate is calculated using the following equation:

$$\text{UART Baud Rate Error (\%)} = 100 \times \left(\frac{\text{Actual Data Rate} - \text{Desired Data Rate}}{\text{Desired Data Rate}}\right)$$

For reliable communication, the UART-LDD baud rate error must never exceed 5 percent in standard UART modes. Tables 124 through 128 provide error data for popular baud rates and commonly-used crystal oscillator frequencies for standard UART and DMX modes of operation.

Table 124. UART-LDD Baud Rates, 20.0 MHz System Clock

Applicable Rate (kHz)	BRG Divisor (Decimal)	Actual Rate (kHz)	Error (%)	Applicable Rate (kHz)	BRG Divisor (Decimal)	Actual Rate (kHz)	Error (%)
1250.0	1	1250.0	0.00	9.60	130	9.62	0.16
625.0	2	625.0	0.00	4.80	260	4.81	0.16
250.0	5	250.0	0.00	2.40	521	2.40	–0.03
115.2	11	113.64	–1.36	1.20	1042	1.20	–0.03
57.6	22	56.82	–1.36	0.60	2083	0.60	0.02
38.4	33	37.88	–1.36	0.30	4167	0.30	–0.01
19.2	65	19.23	0.16				

Table 125. UART-LDD Baud Rates, 19.99848 MHz System Clock

Applicable Rate (kHz)	BRG Divisor (Decimal)	Actual Rate (kHz)	Error (%)	Applicable Rate (kHz)	BRG Divisor (Decimal)	Actual Rate (kHz)	Error (%)
1250.0	1	1250.0	–0.06	9.60	130	9.61	0.10
625.0	2	625.0	–0.06	4.80	260	4.81	0.10
250.0	5	250.0	–0.06	2.40	521	2.40	–0.09
115.2	11	113.6	–1.41	1.20	1042	1.20	0.01
57.6	22	56.79	–1.41	0.60	2083	0.60	0.01
38.4	33	37.86	–1.41	0.30	4167	0.30	0.01
19.2	65	19.2	0.10				

Table 126. UART-LDD Baud Rates, 10.0 MHz System Clock

Applicable Rate (kHz)	BRG Divisor (Decimal)	Actual Rate (kHz)	Error (%)	Applicable Rate (kHz)	BRG Divisor (Decimal)	Actual Rate (kHz)	Error (%)
1250.0	N/A	N/A	N/A	9.60	65	9.62	0.16
625.0	1	625.0	0.00	4.80	130	4.81	0.16
250.0	3	208.3	–16.67	2.40	260	2.40	0.16
115.2	5	125.0	8.51	1.20	521	1.20	–0.03
57.6	11	56.8	–1.36	0.60	1042	0.60	–0.03
38.4	16	39.1	1.73	0.30	2083	0.30	0.2
19.2	33	18.9	–1.36				

Table 127. UART-LDD Baud Rates, 7.3728 MHz System Clock

Applicable Rate (kHz)	BRG Divisor (Decimal)	Actual Rate (kHz)	Error (%)	Applicable Rate (kHz)	BRG Divisor (Decimal)	Actual Rate (kHz)	Error (%)
1250.0	N/A	N/A	N/A	9.60	48	9.60	0.00
625.0	N/A	N/A	N/A	4.80	96	4.80	0.00
250.0	2	230.4	–7.84	2.40	192	2.40	0.00
115.2	4	115.2	0.00	1.20	384	1.20	0.00
57.6	8	57.6	0.00	0.60	768	0.60	0.00
38.4	12	38.4	0.00	0.30	1536	0.30	0.00
19.2	24	19.2	0.00				

Table 128. UART-LDD Baud Rates, 2.4576 MHz System Clock

Applicable Rate (kHz)	BRG Divisor (Decimal)	Actual Rate (kHz)	Error (%)	Applicable Rate (kHz)	BRG Divisor (Decimal)	Actual Rate (kHz)	Error (%)
1250.0	N/A	N/A	N/A	9.60	16	9.60	0.00
625.0	N/A	N/A	N/A	4.80	32	4.80	0.00
250.0	N/A	N/A	N/A	2.40	64	2.40	0.00
115.2	N/A	N/A	N/A	1.20	128	1.20	0.00
57.6	3	57.6	–11.11	0.60	256	0.60	0.00
38.4	4	38.4	0.00	0.30	512	0.30	0.00
19.2	8	19.2	0.00				

Chapter 15. Enhanced Serial Peripheral Interface

The Enhanced Serial Peripheral Interface (ESPI) supports the Serial Peripheral Interface (SPI) and Inter-IC Sound (I^2S). ESPI includes the following features:

- Full-duplex, synchronous, character-oriented communication
- Four-wire interface ($\overline{SS}$, SCK, MOSI and MISO)
- Transmit and receive buffer registers to enable high throughput
- Master Mode transfer rates up to a maximum of one-half the system clock frequency
- Slave Mode transfer rates up to a maximum of one-eighth the system clock frequency
- Error detection
- Dedicated Programmable Baud Rate Generator
- Data transfer control via polling or interrupt

15.1. Architecture

The ESPI is a full-duplex, synchronous, character-oriented channel that supports a four-wire interface (serial clock, transmit data, receive data and slave select). The ESPI block consists of a shift register, data buffer register, a baud rate (clock) generator, control/status registers and a control state machine. Transmit and receive transfers are in synch because there is a single shift register for both transmitting and receiving data. Figure 33 shows a diagram of the ESPI block.

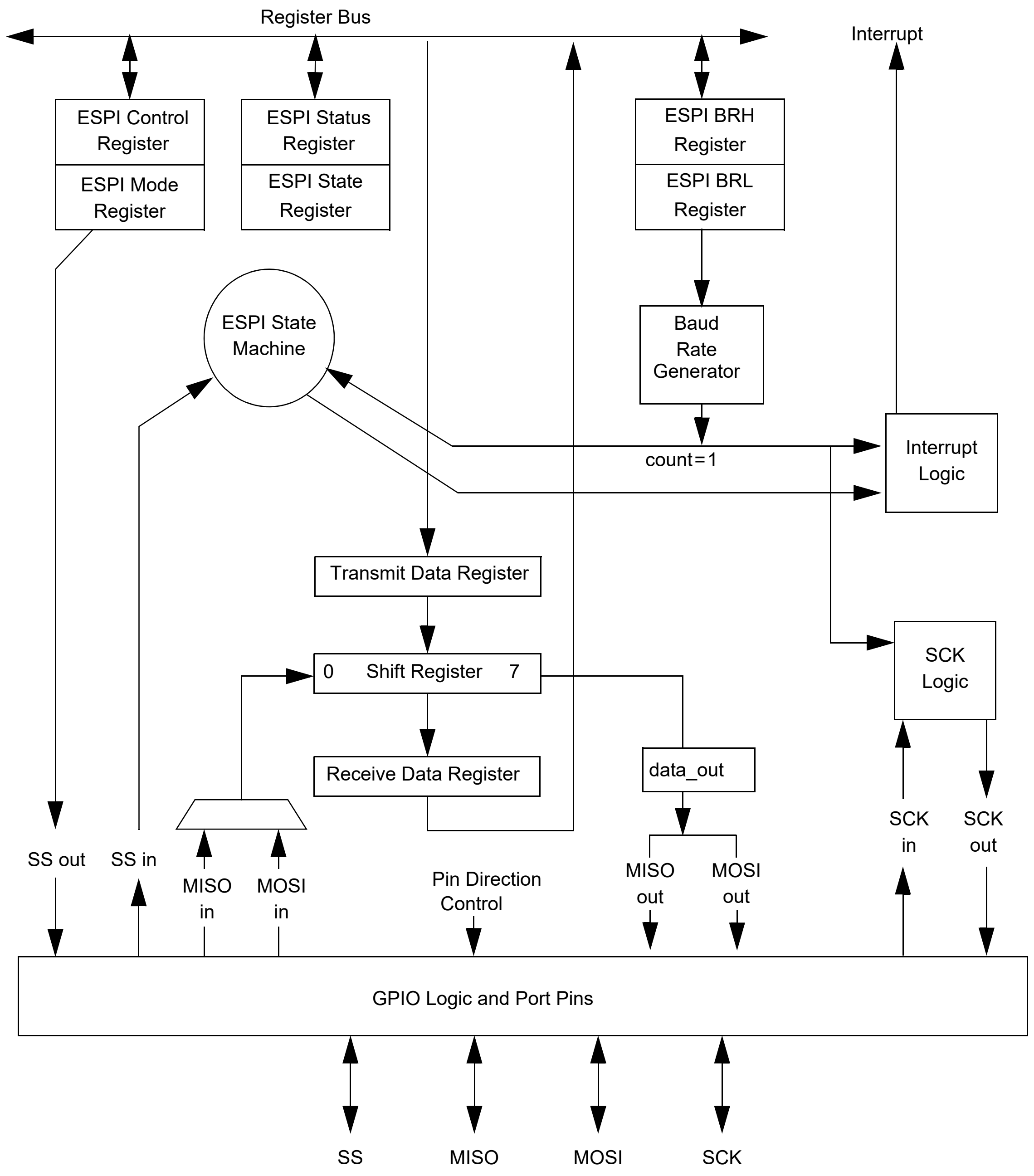


Figure 33. ESPI Block Diagram

15.2. ESPI Signals

The four ESPI signals are:

- Master-In/Slave-Out (MISO)
- Master-Out/Slave-In (MOSI)
- Serial Clock (SCK)
- Slave Select ($\overline{\text{SS}}$)

The following paragraphs discuss these signals as they operate in both Master and Slave modes.

15.2.1. Master-In/Slave-Out

The Master-In/Slave-Out (MISO) pin is configured as an input in a master device and as an output in a slave device. Data is transferred most significant bit first. The MISO pin of a slave device is placed in a high-impedance state if the slave is not selected. When the ESPI is not enabled, this signal is in a high-impedance state. The direction of this pin is controlled by the MMEN bit of the ESPI Control Register.

15.2.2. Master-Out/Slave-In

The Master-Out/Slave-In (MOSI) pin is configured as an output in a master device and as an input in a slave device. Data is transferred most significant bit first. When the ESPI is not enabled, this signal is in a high-impedance state. The direction of this pin is controlled by the MMEN bit of the ESPI Control Register.

15.2.3. Serial Clock

The Serial Clock (SCK) synchronizes data movement both in and out of the Shift Register via the MOSI and MISO pins. In Master Mode (MMEN=1), the ESPI's Baud Rate Generator creates the serial clock and drives it out on its SCK pin to the slave devices. In Slave Mode, the SCK pin is an input. Slave devices ignore the SCK signal unless their $\overline{\text{SS}}$ pin is asserted.

The master and slave are each capable of exchanging a character of data during a sequence of NUMBITS clock cycles; see the ESPI Mode Register (ESPIMODE) on page 248 for details. In both master and slave ESPI devices, data is shifted on one edge of the SCK, and is sampled on the opposite edge where data is stable. SCK phase and polarity is determined by the PHASE and CLKPOL bits in the ESPI Control Register.

15.2.4. Slave Select

The Slave Select signal is a bidirectional framing signal with several modes of operation to support SPI and other synchronous serial interface protocols. Slave Select Mode is selected by the SSMD field of the ESPI Mode Register. The direction of the $\overline{SS}$ signal is controlled by the SSIO bit of the ESPI Mode Register. The $\overline{SS}$ signal is an input on slave devices, and is an output on the active master device. Slave devices ignore transactions on the bus unless their Slave Select input is asserted. In SPI Master Mode, additional GPIO pins are required to provide Slave Selects if there is more than one slave device.

15.3. Operation

During a transfer, data is sent and received simultaneously by both the master and slave devices. Separate signals are required for transmit data, receive data, and the serial clock. When a transfer occurs, a multi-bit (typically 8-bit) character is shifted out one data pin, and a multi-bit character is simultaneously shifted in on second data pin. An 8-bit shift register in the master and an 8-bit shift register in the slave are connected as a circular buffer. The ESPI Shift Register is buffered to support back-to-back character transfers in high-performance applications.

Though the hardware is inherently full-duplex during an SPI transaction, software may choose to use the SPI to send only, to receive only, or to both send and receive data. The ESPIEN1 and ESPIEN0 bits in the Control Register are used to enable data movement in transmit and receive directions. Only the data interrupt(s) associated with the enabled direction(s) will be asserted. If transmit is enabled by ESPIEN1=1, then the TDRE bit in the status register can be set by the SPI and assert the SPI interrupt if DIRQS=1. If receive is enabled by ESPIEN0=1, then the RDRNE bit in the status register can be set by the SPI and will assert the SPI interrupt if DIRQS=1.

When ESPIEN1=0 (transmit is disabled), transmit data will be all 1s and transmit interrupt will not be asserted. When ESPIEN0=0 (receive is disabled), RDRNE will not be set; therefore, a receive interrupt will not be asserted. When both ESPIEN1 and ESPIEN0 are set in Master Mode following a character transfer, interrupts must be serviced before the next transaction will start. These transmit and receive requests can be serviced in either order. To support back-to-back transfers without an intervening pause, the receive and transmit interrupts must be serviced while the current character is being transferred.

The master sources the Serial Clock (SCK) and Slave Select signal ($\overline{SS}$) during the transfer.

15.3.1. Throughput

In Master Mode, the maximum SCK rate supported is one-half the system clock frequency. This frequency is achieved by programming the value `0001h` into the Baud Rate High/Low subregister pair. In SPI Master Mode, if the software transfers do not keep up with the SPI baud rate, there will be a pause between characters. In I^2S Master Mode, the transfer will be terminated if new data is not available to send.

In Slave Mode, the transfer rate is controlled by the master. As long as the TDRE and RDRNE interrupt are serviced before the next character transfer completes, the slave will keep up with the master. The master's baud rate should be set for compatibility with all slave devices so that transmit underruns and receive overruns do not occur. In Slave Mode, the baud rate must be restricted to a maximum of one-eighth of the system clock frequency to allow for synchronization of the SCK input to the internal system clock.

15.3.2. ESPI Clock Phase and Polarity Control

The ESPI supports four combinations of serial clock phase and polarity using two bits in the ESPI Control Register. The clock polarity bit, CLKPOL, selects an active High or active Low clock, and has no effect on the transfer format. Table 128 lists the ESPI clock phase and polarity operation parameters. The clock phase bit, PHASE, selects one of two fundamentally different transfer formats. The data is output a half-cycle before the receive clock edge, which provides a half cycle of setup and hold time.

Table 128. ESPI Clock Phase (PHASE) and Clock Polarity (CLKPOL) Operation

PHASE	CLKPOL	SCK Transmit Edge	SCK Receive Edge	SCK Idle State
0	0	Falling	Rising	Low
0	1	Rising	Falling	High
1	0	Rising	Falling	Low
1	1	Falling	Rising	High

15.3.2.1. Transfer Format when Phase Equals Zero

Figure 34 shows a timing diagram for an SPI-type transfer, in which PHASE=0. For SPI transfers, the clock only toggles during a character transfer. The two SCK waveforms show polarity with CLKPOL=0 and CLKPOL=1. The diagram can be interpreted as either a master or slave timing diagram, because the SCK Master-In/Slave-Out (MISO) and Master-Out/Slave-In (MOSI) pins are directly connected between the master and the slave.

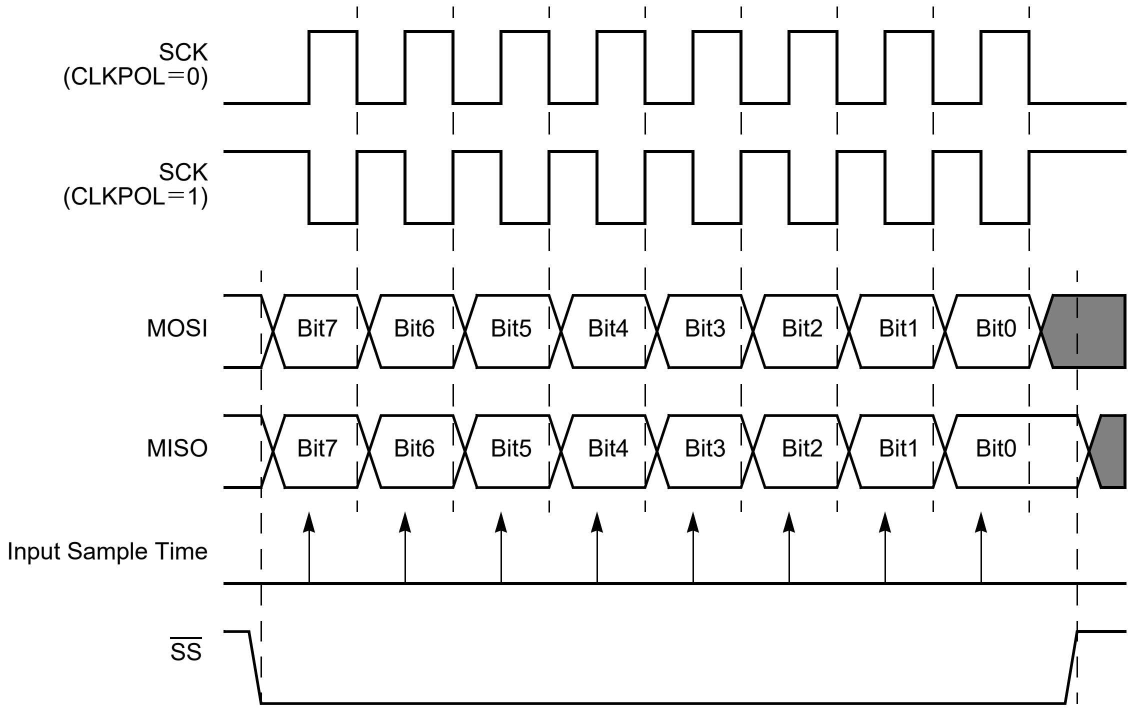


Figure 34. ESPI Timing when PHASE=0

15.3.2.2. Transfer Format When Phase Equals One

Figure 35 shows a timing diagram for an SPI-type transfer in which PHASE is 1. For SPI transfers, the clock only toggles during a character transfer. Two waveforms are depicted for SCK: one for CLKPOL=0 and another for CLKPOL=1.

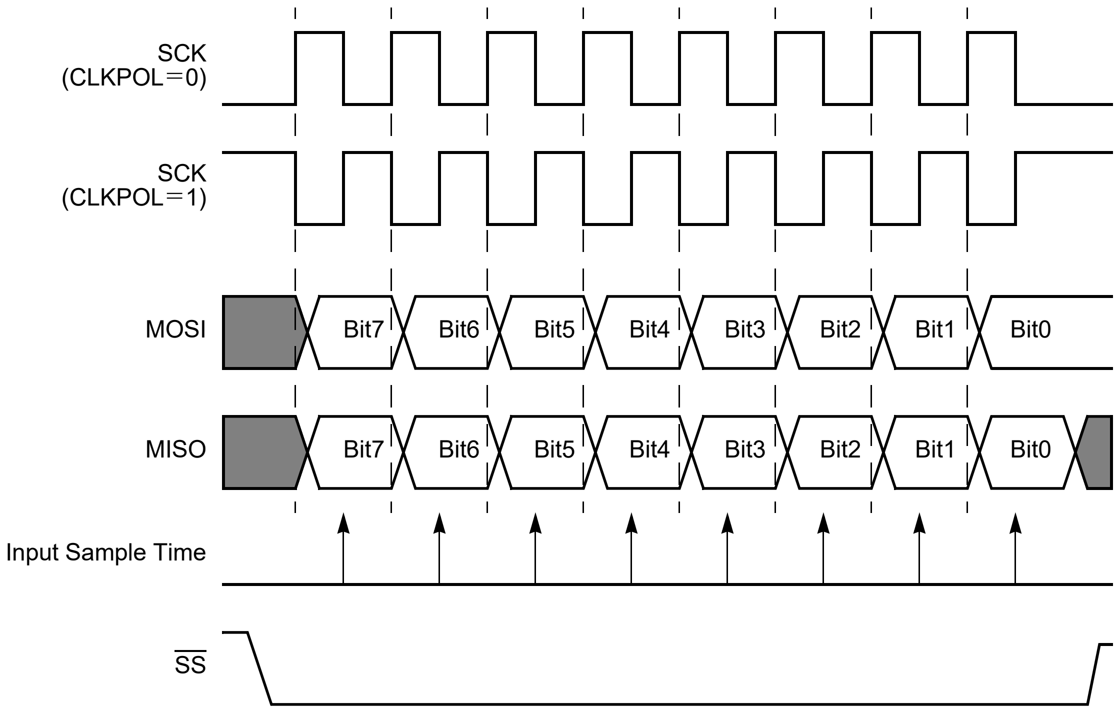


Figure 35. ESPI Timing when PHASE=1

15.3.3. Slave Select Modes of Operation

This section describes the different modes of data transfer supported by the ESPI block. The mode is selected by the Slave Select Mode (SSMD) field of the Mode Register.

15.3.3.1. SPI Mode

SPI Mode is selected by setting the SSMD field of the Mode Register to 000. In this mode, software controls the assertion of the $\overline{SS}$ signal directly via the SSV and TEOF bits of the SPI Transmit Data Command Register. Software can be used to control an SPI Mode transaction. Prior to writing the first transmit data byte, software sets the SSV bit.

$\overline{SS}$ will remain asserted when one or more characters are transferred. There are two mechanisms for deasserting $\overline{SS}$ at the end of the transaction. One method used by software is to set the TEOF bit of the Transmit Data Command Register, when the last TDRE interrupt is being serviced (set TEOF before writing the last data byte). After the last bit of the last character is transmitted, the hardware will automatically deassert the SSV and TEOF bits. The second method is for software to directly clear the SSV bit after the transaction completes. If software clears the SSV bit directly it is not necessary for software to also set the TEOF bit on the last transmit byte. After writing the last transmit byte, the end of the transaction can be detected by waiting for the last RDRNE interrupt or by monitoring the TFST bit in the ESPI Status Register.

Transmit underrun and receive overrun errors will not occur in an SPI Mode master. If the RDRNE and TDRE requests have not been serviced before the current byte transfer completes, SCK will be paused until the Data Register is read and written. The transmit underrun and receive overrun errors will occur in an SPI Mode slave if the slave's software does not keep up with the master's data rate. In this case, the shift register in the slave will be loaded with all 1s to serve as transmit data.

In SPI Mode, the SCK is active only for the data transfer, with one SCK period per bit transferred. If the SPI bus has multiple slaves, the Slave Select lines to all slaves – or all but one of the slaves – must be controlled independently by software using GPIO pins. Figure 36 shows a typical multiple character transfer in SPI Mode.

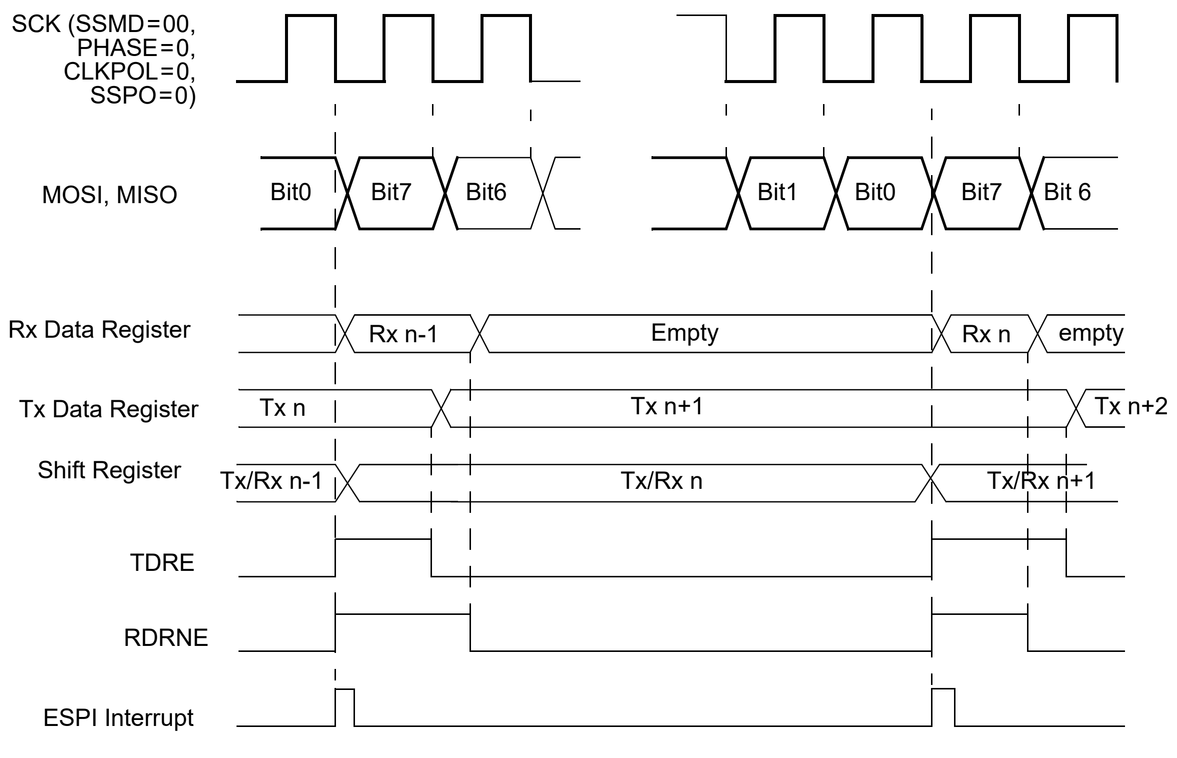


Figure 36. SPI Mode (SSMD=000)

Note: When character *n* is transferred via the Shift Register, software responds to the receive request for character *n*–1 and the transmit request for character *n*+1.

15.3.3.2. Inter-IC Sound (I^2S) Mode

This mode is selected by setting the SSMD field of the Mode Register to 010. The PHASE and CLKPOL bits of the Control Register must be cleared to 0 and the ESPIEN1 bit must be set=1 (ESPIEN0 can be either 1 or 0). If the ESPI is being used to both send and receive I^2S data, the TDRE interrupt should be serviced before the RDRF interrupt. Figure 37 shows I^2S Mode, with $\overline{SS}$ alternating between consecutive frames. Each audio frame consists of a fixed number of bits, typically a multiple of 8 bits such as 16.

The SSV indicates whether the corresponding bytes are left or right channel data. The SSV value must be updated when servicing the TDRE interrupt/request for the first byte in a left or right channel frame. This update can be made by performing a byte write to update SSV, followed by a byte Write to the Data Register. The $\overline{SS}$ signal will lead the data by one SCK period.

A transaction is terminated when the master has no more data to transmit. After the last bit is transferred, SCK will stop, and $\overline{SS}$ and SSV will return to their default states. A transmit underrun error will occur at this point.

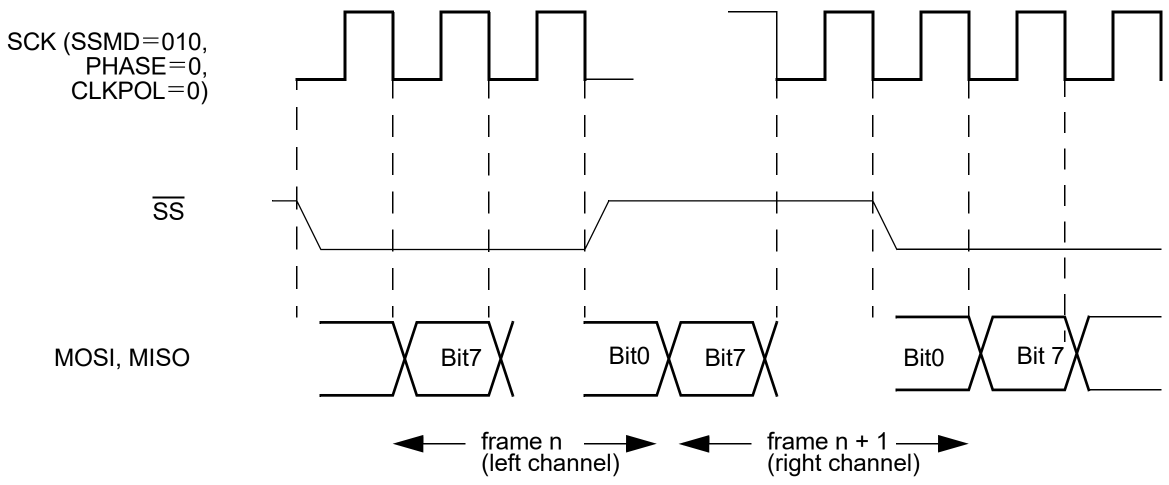


Figure 37. I^2S Mode (SSMD=010), Multiple Frames

15.3.4. SPI Protocol Configuration

This section describes how to configure the ESPI block for the SPI protocol. In the SPI protocol, the master sources the SCK and asserts Slave Select signals to one or more slaves. The Slave Select signals are typically active Low.

15.3.4.1. SPI Master Operation

The ESPI block is configured for Master Mode operation by setting the MMEN bit=1 in the ESPICTL Register. The SSMD field of the ESPI Mode Register is set to 000 for SPI

Protocol Mode. The PHASE, CLKPOL and WOR bits in the ESPICTL Register and the NUMBITS field in the ESPI Mode Register must be set to be consistent with the slave SPI devices. Typically, for an SPI master, SSIO=1 and SSPO=0.

The appropriate GPIO pins are configured for the ESPI alternate function on the MOSI, MISO and SCK pins. Typically, the GPIO for the ESPI $\overline{\text{SS}}$ pin is configured in an alternate function mode, though the software can use any GPIO pin(s) to drive one or more slave select lines. If the ESPI $\overline{\text{SS}}$ signal is not used to drive a Slave Select, the SSIO bit should still be set to 1 in a single-master system. Figures 38 and 39 show block diagrams of the ESPI configured as an SPI master.

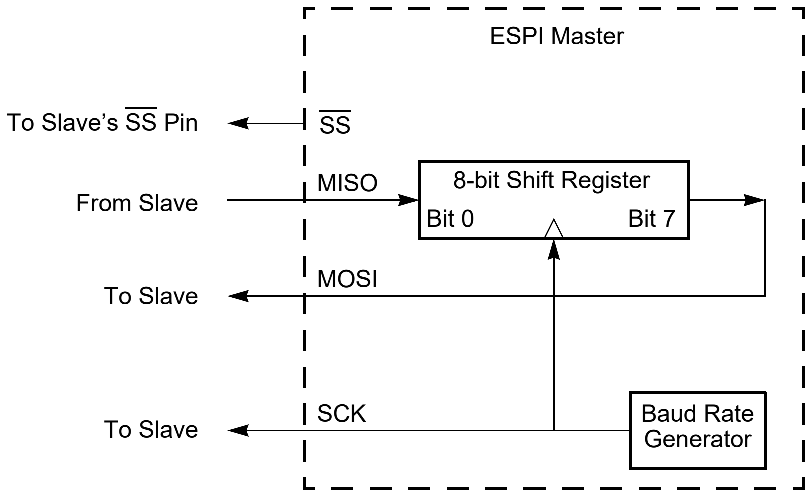


Figure 38. ESPI Configured as an SPI Master in a Single Master, Single Slave System

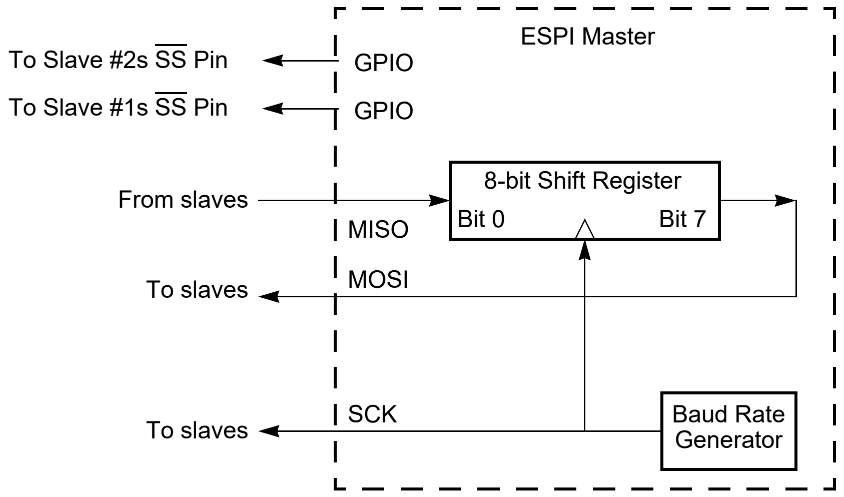


Figure 39. ESPI Configured as an SPI Master in a Single Master, Multiple Slave System

15.3.4.2. Multi-Master SPI Operation

In a multi-master SPI system, all SCK pins are tied together, all MOSI pins are tied together, and all MISO pins are tied together. All SPI pins must be configured in open-drain mode to prevent bus contention. At any time, only one SPI device can be configured as a master, and all other devices on the bus are configured as slaves. The master asserts the $\overline{\text{SS}}$ pin on a selected slave. Next, the active master drives the clock and transmits data on the SCK and MOSI pins to the SCK and MOSI pins on the slave (including those slaves which are not enabled). The enabled slave drives data out its MISO pin to the MISO Master pin.

When the ESPI is configured as a master in a multi-master SPI system, the $\overline{\text{SS}}$ pin must be configured as an input. The $\overline{\text{SS}}$ input signal on a device configured as a master should remain High. If the $\overline{\text{SS}}$ signal on the active master goes Low (indicating that another master is accessing this device as a slave), a collision error flag is set in the ESPI Status Register. The slave select outputs on a master in a multi-master system must come from GPIO pins.

15.3.4.3. SPI Slave Operation

The ESPI block is configured for Slave Mode operation by setting the MMEN bit=0 in the ESPICTL Register and setting the SSIO bit=0 in the ESPIMODE Register. The SSMD field of the ESPI Mode Register is set to 000 for SPI Protocol Mode. The PHASE, CLKPOL and WOR bits in the ESPICTL Register and the NUMBITS field in the ESPIMODE Register must be set to be consistent with the other SPI devices. Typically, for an SPI slave, SSPO=0.

If the slave has data to send to the master, the data must be written to the Transmit Data Register before the transaction starts (first edge of SCK when $\overline{SS}$ is asserted). If the Transmit Data Register is not written prior to the slave transaction, the MISO pin outputs all ones.

Due to the delay resulting from synchronization of the $\overline{SS}$ and SCK input signals to the internal system clock, the maximum SCK baud rate that can be supported in Slave Mode is the system clock frequency divided by 8. This rate is controlled by the SPI master. Figure 40 shows the ESPI configuration in SPI Slave Mode.

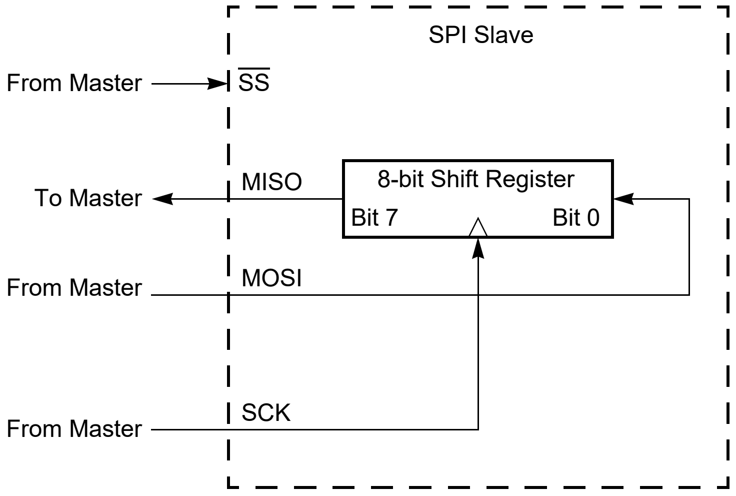


Figure 40. ESPI Configured as an SPI Slave

15.3.5. Error Detection

Error events detected by the ESPI block are described in this section. Error events generate an ESPI interrupt and set a bit in the ESPI Status Register. Read or write a 1 to clear the error bits of the ESPI Status Register.

15.3.5.1. Transmit Underrun

A transmit underrun error occurs for a master with SSMD=010 when a character transfer is completed and when TDRE=1. When a transmit underrun occurs in these modes, the transfer will be aborted (i.e, SCK will halt and SSV will be deasserted). For a master in SPI Mode (SSMD=000), a transmit underrun is not signaled, because SCK will pause and wait for the Transmit Data Register to be written.

In Slave Mode, a transmit underrun error occurs if TDRE=1 and ESPIEN1 = 1 (transmit is enabled) at the start of a transfer. When a transmit underrun occurs in Slave Mode, ESPI will transmit a character of all ones.

A transmit underrun sets the TUND bit in the ESPI Status Register to 1. Writing a 1 to TUND clears this error flag.

15.3.5.2. Mode Fault (Multi-Master Collision)

A mode fault indicates when more than one master is trying to communicate simultaneously (i.e., a multi-master collision) in SPI Mode. The mode fault is detected when the enabled master's $\overline{\text{SS}}$ input pin is asserted. For this assertion to occur, the Control and Mode registers must be configured with MMEN=1, SSIO=0 ($\overline{\text{SS}}$ is an input) and $\overline{\text{SS}}$ input=0. A mode fault sets the COL bit in the ESPI Status Register to 1. Writing a 1 to COL clears this error flag.

15.3.5.3. Receive Overrun

A receive overrun error occurs when a transfer completes and the RDRNE bit is still set from the previous transfer. A receive overrun sets the ROVR bit in the ESPI Status Register to 1. Writing a 1 to ROVR clears this error flag. The Receive Data Register is not overwritten and will contain the data from the transfer which initially set the RDRNE bit. Subsequent received data is lost until the RDRNE bit is cleared.

In SPI Master Mode, a receive overrun will not occur. Instead, the SCK will be paused until software responds to the previous RDRNE/TDRE requests.

15.3.5.4. Slave Mode Abort

In Slave Mode, if the $\overline{\text{SS}}$ pin deasserts before all bits in a character have been transferred, the transaction is aborted. When this condition occurs, the ABT bit is set in the ESPI Status Register. A slave abort error resets the slave control logic to idle state.

A slave abort error is also asserted in Slave Mode if BRGCTL=1 and a baud rate generator time-out occurs. When BRGCTL=1 in Slave Mode, the baud rate generator functions as a watchdog timer monitoring the SCK signal. The BRG counter is reloaded every time a transition on SCK occurs while $\overline{\text{SS}}$ is asserted. The Baud Rate Reload registers must be programmed with a value longer than the expected time between the $\overline{\text{SS}}$ assertion and the first SCK edge, between SCK transitions while $\overline{\text{SS}}$ is asserted, and between the last SCK edge and $\overline{\text{SS}}$ deassertion. A time-out indicates that the master is stalled or disabled. Writing a 1 to ABT clears this error flag.

15.3.6. ESPI Interrupts

ESPI has a single interrupt output which is asserted when any of the TDRE, TUND, COL, ABT, ROVR or RDRNE bits are set in the ESPI Status Register. The setting of TDRE will only generate an interrupt if transmit is enabled (ESPIEN1 = 1). The interrupt is a pulse which is generated when any one of the source bits initially sets. The TDRE and RDRNE interrupts can be enabled/disabled via the Data Interrupt Request Select (DIRQS) bit of the ESPI Control Register.

A transmit interrupt is asserted by the TDRE status bit when the ESPI block is enabled and the DIRQS bit is set. The TDRE bit in the status register is cleared automatically when the

Transmit Data Register is written or the ESPI block is disabled. After the Transmit Data Register is loaded into the Shift Register to start a new transfer, the TDRE bit will be set again, causing a new transmit interrupt. In Master or Slave modes, if information is being received but not transmitted, the transmit interrupts can be eliminated by selecting Receive Only Mode (ESPIEN1,0=01).

A receive interrupt is generated by the RDRNE status bit when the ESPI block is enabled, the DIRQS bit is set, and a character transfer completes. At the end of the character transfer, the contents of the Shift Register are transferred into the Receive Data Register, causing the RDRNE bit to assert. The RDRNE bit is cleared when the Data Buffer is read as empty. If information is being transmitted but not received by the software application, the receive interrupt can be eliminated by selecting Transmit Only Mode (ESPIEN1,0=10) in either Master or Slave modes. When information is being sent and received under interrupt control, RDRNE and TDRE will both assert simultaneously at the end of a character transfer. In this case, RDRNE and TDRE can be serviced in either order.

ESPI error interrupts occur if any of the TUND, COL, ABT and ROVR bits in the ESPI Status Register are set. These bits are cleared by writing a 1. If the ESPI is disabled (ESPIEN1, 0=00), an ESPI interrupt can be generated by a Baud Rate Generator time-out. This timer function must be enabled by setting the BRGCTL bit in the ESPICTL Register. This timer interrupt does not set any of the bits of the ESPI Status Register.

15.3.7. ESPI Baud Rate Generator

In ESPI Master Mode, the Baud Rate Generator creates a lower frequency serial clock (SCK) for data transmission synchronization between the master and the external slave. The input to the Baud Rate Generator is the system clock. The ESPI Baud Rate High and Low Byte registers combine to form a 16-bit reload value, BRG[15:0], for the ESPI Baud Rate Generator. The ESPI baud rate is calculated using the following equation:

$$\text{SPI Baud Rate (bits/s)} = \frac{\text{System Clock Frequency (Hz)}}{2 \times \text{BRG[15:0]}}$$

The minimum baud rate is obtained by setting BRG[15:0] to `0000h` for a clock divisor value of (2 x 65536=131072).

When the ESPI is disabled, the Baud Rate Generator can function as a basic 16-bit timer with interrupt on time-out. Observe the following steps to configure the Baud Rate Generator as a timer with interrupt on time-out:

1. Disable the ESPI by clearing the ESPIEN1,0 bits in the ESPI Control Register.
2. Load the appropriate 16-bit count value into the ESPI Baud Rate High and Low Byte registers.
3. Enable the Baud Rate Generator timer function and associated interrupt by setting the BRGCTL bit in the ESPI Control Register to 1.

When configured as a general-purpose timer, the SPI BRG interrupt interval is calculated using the following equation:

$$\text{SPI BRG Interrupt Interval (s)} = \text{System Clock Period (s)} \times \text{BRG[15:0]}$$

15.4. ESPI Control Register Definitions

The ESPI control registers are defined in this section.

15.4.1. ESPI Data Register

The ESPI Data Register, shown in Table 129, addresses both the outgoing Transmit Data Register and the incoming Receive Data Register. Reads from the ESPI Data Register return the contents of the Receive Data Register. The Receive Data Register is updated with the contents of the Shift Register at the end of each transfer. Writes to the ESPI Data Register load the Transmit Data Register unless TDRE=0. Data is shifted out starting with bit 7. The last bit received resides in bit position 0. In either the Master or Slave modes, if TDRE=0, writes to this register are ignored.

When the character length is less than 8 bits (as set by the NUMBITS field in the ESPI Mode Register), the transmit character must be left-justified in the ESPI Data Register. A received character of less than 8 bits is right-justified (i.e., the last bit received is in bit position 0). For example, if the ESPI is configured for 4-bit characters, the transmit characters must be written to ESPIDATA[7:4] and the received characters are read from ESPIDATA[3:0].

Table 129. ESPI Data Register (ESPIDATA)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	DATA							
Reset	X	X	X	X	X	X	X	X
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F60h							
Note: *x* references bits in the range [1:0].								

Bit	**Description**
[7:0] DATA	**Data** Transmit and/or receive data. Writes to the ESPIDATA Register load the Shift Register. Reads from the ESPIDATA Register return the value of the Receive Data Register.

15.4.2. ESPI Transmit Data Command Register

The ESPI Transmit Data Command Register, shown in Table 130, provides control of the $\overline{SS}$ pin when it is configured as an output (Master Mode).

Table 130. ESPI Transmit Data Command Register (ESPITDCR)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	Reserved						TEOF	SSV
Reset	0	0	0	0	0	0	0	0
R/W	R	R	R	R	R	R	R/W	R/W
Address	F61h							
Note: *x* references bits in the range [1:0].								

Bit	Description
[7:2]	These bits are reserved and must be written to 000000.
[1] TEOF	**Transmit End of Frame** This bit is used in Master Mode to indicate that the data in the Transmit Data Register is the last byte of the transfer or frame. When the last byte has been sent $\overline{SS}$ (and SSV) will change state, and TEOF will automatically clear. 0: The data in the Transmit Data Register is not the last character in the message. 1: The data in the Transmit Data Register is the last character in the message.
[0] SSV	**Slave Select Value** When SSIO=1, writes to this register will control the value output on the $\overline{SS}$ pin. To learn more, see the SSMD field of the ESPI Mode Register section on page 248.

15.4.3. ESPI Control Register

The ESPI Control Register, shown in Table 131, configures the ESPI for transmit and receive operations.

Table 131. ESPI Control Register (ESPICTL)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	DIRQS	ESPIEN1	BRGCTL	PHASE	CLKPOL	WOR	MMEN	ESPIEN0
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F62h							
Note: *x* references bits in the range [1:0].								

Bit	Description
[7] DIRQS	**Data Interrupt Request Select** This bit is used to disable or enable data (TDRE and RDRNE) interrupts. Disabling the data interrupts is required to control data transfer by polling. Error interrupts are not disabled. To block all ESPI interrupt sources, clear the ESPI interrupt enable bit in the Interrupt Controller. 0: TDRE and RDRNE assertions do not cause an interrupt. Use this setting if controlling data transfer by software polling of TDRE and RDRNE. The TUND, COL, ABT and ROVR bits will cause an interrupt. 1: TDRE and RDRNE assertions will cause an interrupt. TUND, COL, ABT and ROVR will also cause interrupts. Use this setting when controlling data transfer via interrupt handlers.
[6,0] ESPIEN1, ESPIEN0	**ESPI Enable and Direction Control** 00: The ESPI block is disabled. BRG can be used as a general-purpose timer by setting BRGCTL=1. 01: Receive Only Mode. Use this setting in Master or Slave Mode if software application is receiving data but not sending. Transmit interrupt requests will not be asserted due to TDRE being set. Transmitted data will be all ones. Receive Only Mode for a master is not valid for I^{2}S operation (SSMD=010). 10: Transmit Only Mode Use this setting in Master or Slave Mode when the software application is sending data but not receiving. RDRNE will not assert. Receive interrupt requests will not be asserted. 11: Transmit/Receive Mode Use this setting if the software application is both sending and receiving information. Both TDRE and RDRNE will be active.
[5] BRGCTL	**Baud Rate Generator Control** The function of this bit depends upon ESPIEN1,0. When ESPIEN1,0=00, this bit allows enabling the BRG to provide periodic interrupts. **If the ESPI is disabled** 0: The Baud Rate Generator timer function is disabled. Reading the Baud Rate High and Low registers returns the BRG reload value. 1: The Baud Rate Generator timer function and time-out interrupt is enabled. Reading the Baud Rate High and Low registers returns the BRG Counter value. **If the ESPI is enabled:** 0: Reading the Baud Rate High and Low registers returns the BRG reload value. If MMEN=1, the BRG is enabled to generate SCK. If MMEN=0, the BRG is disabled. 1: Reading the Baud Rate High and Low registers returns the BRG Counter value. If MMEN=1, the BRG is enabled to generate SCK. If MMEN=0 the BRG is enabled to provide a Slave SCK time-out. See the error description in the Slave Mode Abort section on page 243. **CAUTION:** If reading the counter one byte at a time while the BRG is counting, keep in mind that the values will not be in sync.
[4] PHASE	**Phase Select** Sets the phase relationship of the data to the clock. For more information about operation of the PHASE bit, see the ESPI Clock Phase and Polarity Control section on page 235.

Bit	Description (Continued)
[3] CLKPOL	**Clock Polarity** 0: SCK idles Low (0). 1: SCK idles High (1).
[2] WOR	**Wire OR (Open-Drain) Mode Enabled** 0: ESPI signal pins not configured for open-drain. 1: All four ESPI signal pins (SCK, $\overline{SS}$, MISO and MOSI) configured for open-drain function. This setting is typically used for multi-master and/or multi-slave configurations.
[1] MMEN	**ESPI Master Mode Enable** This bit controls the data I/O pin selection and SCK direction. 0: Data out on MISO, data in on MOSI (used in SPI Slave Mode), SCK is an input. 1: Data out on MOSI, data in on MISO (used in SPI Master Mode), SCK is an output.

15.4.4. ESPI Mode Register

The ESPI Mode Register, shown in Table 132, configures the character bit width and mode of the ESPI I/O pins.

Table 132. ESPI Mode Register (ESPIMODE)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	SSMD			NUMBITS[2:0]			SSIO	SSPO
Reset	000			0	0	0	0	0
R/W	R/W			R/W	R/W	R/W	R/W	R/W
Address	F63h							
Note: *x* references bits in the range [1:0].								

Bit	Description
[7:5] SSMD	**Slave Select Mode** This field selects the behavior of $\overline{SS}$ as a framing signal. For a description of these modes, see the Slave Select section on page 233. **000 = SPI Mode** When SSIO=1, the $\overline{SS}$ pin is driven directly from the SSV bit in the Transmit Data Command Register. The master software should set SSV (or a GPIO output if the $\overline{SS}$ pin is not connected to the appropriate slave) to the asserted state prior to or on the same clock cycle that the Transmit Data Register is written with the initial byte. At the end of a frame (after the last RDRNE event), SSV will be automatically deasserted by hardware. In this mode, SCK is active only for data transfer (one clock cycle per bit transferred). **001 = Loopback Mode** When ESPI is configured as master (MMEN=1), the outputs are deasserted and data is looped from Shift Register Out to Shift Register In. When ESPI is configured as a slave (MMEN=0) and $\overline{SS}$ in asserts, MISO (slave output) is tied to MOSI (slave input) to provide an asynchronous remote loop back (echo) function. **010 = I^{2}S Mode** In this mode, the value from SSV will be output by the master on the $\overline{SS}$ pin with one SCK period before the data and will remain in that state until the start of the next frame. Typically, this mode is used to send back to back frames with $\overline{SS}$ alternating on each frame. A frame boundary is indicated in the master when SSV changes. A frame boundary is detected in the slave by $\overline{SS}$ changing state. The $\overline{SS}$ framing signal will lead the frame by one SCK period. In this mode, SCK will run continuously, starting with the initial $\overline{SS}$ assertion. Frames will run back-to-back as long as software continues to provide data. An example of this mode is the I^{2}S protocol (Inter IC Sound) which is used to carry left and right channel audio data with the $\overline{SS}$ signal indicating which channel is being sent. In Slave Mode, the change in state of $\overline{SS}$ (Low to High or High to Low) triggers the start of a transaction on the next SCK cycle.

Bit	Description (Continued)
[4:2] NUMBITS[2:0]	**Number of Data Bits Per Character to Transfer** This field contains the number of bits to shift for each character transfer. To learn more about valid bit positions when the character length is less than 8 bits, see the description of the ESPI Data Register section on page 245. 000: 8 bits. 001: 1 bit. 010: 2 bits. 011: 3 bits. 100: 4 bits. 101: 5 bits. 110: 6 bits. 111: 7 bits.
[1] SSIO	**Slave Select I/O** This bit controls the direction of the $\overline{SS}$ pin. In single Master Mode, SSIO is set to 1 even if a separate GPIO pin is being used to provide the $\overline{SS}$ output function. In the SPI slave or multi-master configuration, SSIO is set to 0. 0: $\overline{SS}$ pin configured as an input (SPI slave and multi-master modes). 1: $\overline{SS}$ pin configured as an output (SPI single-master mode).
[0] SSPO	**Slave Select Polarity** This bit controls the polarity of the $\overline{SS}$ pin. 0: $\overline{SS}$ is active Low. (SSV=1 corresponds to $\overline{SS}$=0). 1: $\overline{SS}$ is active High. (SSV=1 corresponds to $\overline{SS}$=1).

15.4.5. ESPI Status Register

The ESPI Status Register, shown in Table 133, indicates the current state of the ESPI. All bits revert to their Reset state if the ESPI is disabled.

Table 133. ESPI Status Register (ESPISTAT)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	TDRE	TUND	COL	ABT	ROVR	RDRNE	TFST	SLAS
Reset	1	0	0	0	0	0	0	1
R/W	R	R/W*	R/W*	R/W*	R/W*	R	R	R
Address	F64h							
Note: *****R/W=read access; write a 1 to clear the bit to 0; *x* references bits in the range [1:0].								

Bit	Description
[7] TDRE	**Transmit Data Register Empty** 0: Transmit Data Register is full or ESPI is disabled. 1: Transmit Data Register is empty. A write to the ESPI (Transmit) Data Register clears this bit.
[6] TUND	**Transmit Underrun** 0: A Transmit Underrun error has not occurred. 1: A Transmit Underrun error has occurred.
[5] COL	**Collision** 0: A multi-master collision (mode fault) has not occurred. 1: A multi-master collision (mode fault) has occurred.
[4] ABT	**Slave Mode Transaction Abort** This bit is set if the ESPI is configured in Slave Mode, a transaction is occurring and $\overline{\text{SS}}$ deasserts before all bits of a character have been transferred as defined by the NUMBITS field of the ESPIMODE Register. This bit can also be set in Slave Mode by an SCK monitor time-out (MMEN=0, BRGCTL=1). 0: A Slave Mode transaction abort has not occurred. 1: A Slave Mode transaction abort has occurred.
[3] ROVR	**Receive Overrun** 0: A Receive Overrun error has not occurred. 1: A Receive Overrun error has occurred.
[2] RDRNE	**Receive Data Register Not Empty** 0: Receive Data Register is empty. 1: Receive Data Register is not empty.

Bit	Description (Continued)
[1] TFST	**Transfer Status** 0: No data transfer is currently in progress. 1: Data transfer is currently in progress.
[0] SLAS	**Slave Select** Reading this bit returns the current value of the $\overline{SS}$ pin. 0: The $\overline{SS}$ pin input is Low. 1: The $\overline{SS}$ pin input is High.

15.4.6. ESPI State Register

The ESPI State Register, shown in Table 134, lets you observe the ESPI clock, data and internal state.

Table 134. ESPI State Register (ESPISTATE)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	SCKI	SDI	ESPISTATE					
Reset	0	0	0					
R/W	R	R	R					
Address	F65h							
Note: *x* references bits in the range [1:0].								

Bit	Description
[7] SCKI	**Serial Clock Input** This bit reflects the state of the serial clock pin. 0: The SCK input pin is Low. 1: The SCK input pin is High.
[6] SDI	**Serial Data Input** This bit reflects the state of the serial data input (MOSI or MISO depending on the MMEN bit). 0: The serial data input pin is Low. 1: The serial data input pin is High.
[5:0] ESPISTATE	**ESPI State Machine** Indicates the current state of the internal ESPI State Machine. This information is intended for manufacturing test purposes. The state values may change in future hardware revisions and are not intended to be used by a software driver.

Table 135 defines the valid ESPI states.

Table 135. ESPISTATE Values

ESPISTATE Value	Description
00_0000	Idle.
00_0001	Slave Wait for SCK.
00_0010	I^{2}S Slave Mode start delay.
00_0011	I^{2}S Slave Mode start delay.
01_0000	SPI Master Mode start delay.
11_0001	I^{2}S Master Mode start delay.
11_0010	I^{2}S Master Mode start delay.
10_1110	Bit 7 Receive.
10_1111	Bit 7 Transmit.
10_1100	Bit 6 Receive.
10_1101	Bit 6 Transmit.
10_1010	Bit 5 Receive.
10_1011	Bit 5 Transmit.
10_1000	Bit 4 Receive.
10_1001	Bit 4 Transmit.
10_0110	Bit 3 Receive.
10_0111	Bit 3 Transmit.
10_0100	Bit 2 Receive.
10_0101	Bit 2 Transmit.
10_0010	Bit 1 Receive.
10_0011	Bit 1 Transmit.
10_0000	Bit 0 Receive.
10_0001	Bit 0 Transmit.

15.4.7. ESPI Baud Rate High and Low Byte Registers

The ESPI Baud Rate High and Low Byte registers, shown in Tables 136 and 137, combine to form a 16-bit reload value, BRG[15:0], for the ESPI Baud Rate Generator. The ESPI baud rate is calculated using the following equation:

$$\text{SPI Baud Rate (bits/s)} = \frac{\text{System Clock Frequency (Hz)}}{2 \times \text{BRG[15:0]}}$$

The minimum baud rate is obtained by setting BRG[15:0] to `0000h` for a clock divisor value of (2 x 65536=131072).

When the ESPI function is disabled, the BRG functions as a basic 16-bit timer with interrupt on time-out.

Follow the procedure below to configure the BRG as a general-purpose timer with interrupt on timeout:

1. Disable the ESPI by setting ESPIEN[1:0]=00 in the SPI Control Register.
2. Load the appropriate 16-bit count value into the ESPI Baud Rate High and Low Byte registers.
3. Enable the BRG timer function and associated interrupt by setting the BRGCTL bit in the ESPI Control Register to 1.

When configured as a general-purpose timer, the SPI BRG interrupt interval is calculated using the following equation:

$$\text{SPI BRG Interrupt Interval (s)} = \text{System Clock Period (s)} \times \text{BRG[15:0]}$$

Table 136. ESPI Baud Rate High Byte Register (ESPIBRH)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	BRH							
Reset	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F66							
Note: *x* references bits in the range [1:0].								

Bit	**Description**
[7:0] BRH	**ESPI Baud Rate High Byte** The most significant byte, BRG[15:8], of the ESPI Baud Rate Generator's reload value.

Table 137. ESPI Baud Rate Low Byte Register (ESPIBRL)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	BRL							
Reset	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F67h							
Note: *x* references bits in the range [1:0].								

Bit	Description
[7:0] BRL	**ESPI Baud Rate Low Byte** The least significant byte, BRG[7:0], of the ESPI Baud Rate Generator's reload value.

Chapter 16. I^{2}C Master/Slave Controller

The I^{2}C Master/Slave Controller ensures that the F3224 Series devices are bus-compatible with the I^{2}C protocol. The I^{2}C bus consists of the serial data signal (SDA) and a serial clock signal (SCL) bidirectional lines. The features of the I^{2}C controller include:

- Operates in MASTER/SLAVE or SLAVE ONLY modes
- Supports arbitration in a multimaster environment (MASTER/SLAVE Mode)
- Supports data rates up to 400 Kbps
- 7-bit or 10-bit slave address recognition (interrupt-only on address match)
- Optional general call address recognition
- Optional digital filter on receive SDA, SCL lines
- Optional interactive receive mode allows software interpretation of each received address and/or data byte before acknowledging
- Unrestricted number of data bytes per transfer
- Baud Rate Generator can be used as a general-purpose timer with an interrupt if the I^{2}C controller is disabled

16.1. Architecture

Figure 41 shows the architecture of the I^{2}C controller.

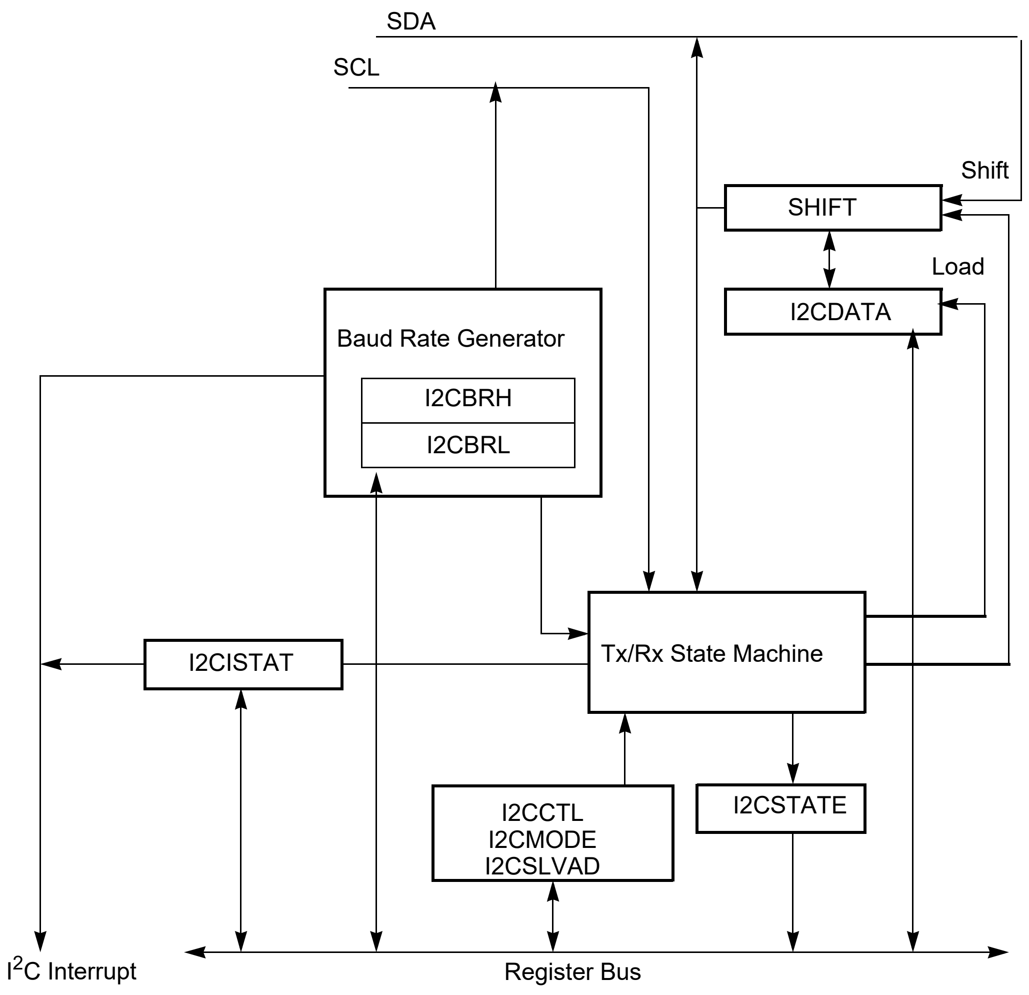


Figure 41. I^{2}C Controller Block Diagram

16.1.1. I^{2}C Master/Slave Controller Registers

Table 138 summarizes the I^{2}C Master/Slave controller's software-accessible registers.

Table 138. I^{2}C Master/Slave Controller Registers

Name	Abbreviation	Description
I^{2}C Data	I2CDATA	Transmit/receive data register.
I^{2}C Interrupt Status	I2CISTAT	Interrupt status register.
I^{2}C Control	I2CCTL	Control register: basic control functions.

Table 138. I^2C Master/Slave Controller Registers (Continued)

Name	Abbreviation	Description
I^2C Baud Rate High	I2CBRH	High byte of baud rate generator initialization value.
I^2C Baud Rate Low	I2CBRL	Low byte of baud rate generator initialization value.
I^2C State	I2CSTATE	State register.
I^2C Mode	I2CMODE	Selects MASTER or SLAVE modes, 7-bit or 10-bit addressing; configure address recognition, define slave address bits [9:8].
I^2C Slave Address	I2CSLVAD	Defines slave address bits [7:0].

16.2. Operation

The I^2C Master/Slave Controller operates in MASTER/SLAVE Mode, SLAVE ONLY Mode, or with master arbitration. In MASTER/SLAVE Mode, it can be used as the only master on the bus or as one of the several masters on the bus, with arbitration. In a multi-master environment, the controller switches from MASTER to SLAVE Mode on losing arbitration.

Though slave operation is fully supported in MASTER/SLAVE Mode, if a device is intended to operate only as a slave, then SLAVE ONLY Mode can be selected. In SLAVE ONLY Mode, the device will not initiate a transaction, even if the software inadvertently sets the start bit.

16.2.1. SDA and SCL Signals

The I^2C circuit sends all addresses, Data and Acknowledge signals over the SDA line, with most-significant bit first. SCL is the clock for the I^2C bus. When the SDA and SCL pin alternate functions are selected for their respective GPIO ports, the pins are automatically configured for open-drain operation.

The master is responsible for driving the SCL clock signal. During the Low period of the clock, a slave can hold the SCL signal Low to suspend the transaction if it is not ready to proceed. The master releases the clock at the end of the Low period and notices that the clock remains Low instead of returning to a High level. When the slave releases the clock, the I^2C master continues the transaction. All data is transferred in bytes; there is no limit to the amount of data transferred in one operation. When transmitting an address, data, or an Acknowledge, the SDA signal changes in the middle of the Low period of SCL**.** When receiving an address, data, or an Acknowledge; the SDA signal is sampled in the middle of the High period of SCL.

A low-pass digital filter can be applied to the SDA and SCL receive signals by setting the Filter Enable (FILTEN) bit in the I^2C Control Register. When the filter is enabled, any glitch that is less than a system clock period in width will be rejected. This filter should be enabled when running in I^{2}C FAST Mode (400 KBps) and can also be used at lower data rates.

16.2.2. I^{2}C Interrupts

The I^{2}C controller contains multiple interrupt sources that are combined into one interrupt request signal to the interrupt controller. If the I^{2}C controller is enabled, the source of the interrupt is determined by which bits are set in the I2CISTAT Register. If the I^{2}C controller is disabled, the BRG controller is used to generate general-purpose timer interrupts.

Each interrupt source, other than the baud rate generator interrupt, features an associated bit in the I2CISTAT Register that automatically clears when software reads the register or performs another task, such as reading/writing the Data Register.

16.2.2.1. Transmit Interrupts

Transmit interrupts (i.e., the TDRE bit = 1 in the I2CISTAT Register) occur under the following conditions, both of which must be true:

- The Transmit Data Register is empty and the TXI bit=1 in the I^{2}C Control Register.
- The I^{2}C controller is enabled with one of the following elements:
  - The first bit of a 10-bit address is shifted out.
  - The first bit of the final byte of an address is shifted out and the RD bit is deasserted.
  - The first bit of a data byte is shifted out.

Writing to the I^{2}C Data Register always clears the TRDE bit to 0.

16.2.2.2. Receive Interrupts

Receive interrupts (i.e., the RDRF bit=1 in the I2CISTAT Register) occur when a byte of data has been received by the I^{2}C controller. The RDRF bit is cleared by reading from the I^{2}C Data Register. If the RDRF interrupt is not serviced prior to the completion of the next receive byte, the I^{2}C controller holds SCL Low during the final data bit of the next byte until RDRF is cleared, to prevent receive overruns. A receive interrupt does not occur when a slave receives an address byte or when data bytes following a slave address do not match. An exception is if the Interactive Receive Mode (IRM) bit is set in the I2CMODE Register, in which case receive interrupts occur for all receive address and data bytes in SLAVE Mode.

16.2.2.3. Slave Address Match Interrupts

Slave address match interrupts (i.e., the SAM bit=1 in the I2CISTAT Register) occur when the I^{2}C controller is in SLAVE Mode and a received address matches the unique slave address. The General Call Address (`0000_0000`) and STARTBYTE (`0000_0001`) are recognized if the GCE bit=1 in the I2CMODE Register. The software checks the RD bit in the I2CISTAT Register to determine if the transaction is a read or write transaction. The General Call Address and STARTBYTE address are also distinguished by the RD bit. The General Call Address (GCA) bit of the I2CISTAT Register indicates whether the address match occurred on the unique slave address or the General Call/STARTBYTE address. The SAM bit clears automatically when the I2CISTAT Register is read.

If configured via the MODE[1:0] field of the I^2C Mode Register for 7-bit slave addressing, the most significant 7 bits of the first byte of a transaction are compared against the SLA[6:0] bits of the Slave Address Register. If configured for 10-bit slave addressing, the first byte of the transaction is compared against {11110, SLA[9:8], $R/\overline{W}$} and the second byte is compared against SLA[7:0].

16.2.2.4. Arbitration Lost Interrupts

Arbitration Lost interrupts (i.e., the ARBLST bit=1 in the I2CISTAT Register) occur when the I^2C controller is in MASTER Mode and loses arbitration (outputs 1 on SDA and receives 0 on SDA). The I^2C controller switches to SLAVE Mode when this instance occurs. This bit clears automatically when the I2CISTAT Register is read.

16.2.2.5. Stop/Restart Interrupts

A stop/restart event interrupt (i.e., the SPRS bit=1 in the I2CISTAT Register) occurs when the I^2C controller is in SLAVE Mode and a stop or restart condition is received, indicating the end of the transaction. The RSTR bit in the I^2C State Register indicates whether the bit is set due to a stop or restart condition. When a restart occurs, a new transaction by the same master is expected to follow. This bit is automatically cleared when the I2CISTAT Register is read. The stop/restart interrupt occurs only on a selected (address match) slave.

16.2.2.6. Not Acknowledge Interrupts

Not Acknowledge interrupts (i.e., the NCKI bit=1 in the I2CISTAT Register) occur in MASTER Mode when a Not Acknowledge is received or sent by the I^2C controller and the start or stop bit is not set in the I^2C Control Register. In MASTER Mode, the Not Acknowledge interrupt clears by setting the start or stop bit. When this interrupt occurs in MASTER Mode, the I^2C controller waits until it is cleared before performing any action. In SLAVE Mode, the Not Acknowledge interrupt occurs when a Not Acknowledge is received in response to data sent. The NCKI bit clears in SLAVE Mode when software reads the I2CISTAT Register.

16.2.2.7. General-Purpose Timer Interrupt from Baud Rate Generator

If the I^2C controller is disabled (i.e., the IEN bit in the I2CCTL Register=0) and the BIRQS bit in the I2CCTL Register=1, an interrupt is generated when the baud rate generator (BRG) counts down to 1. The baud rate generator reloads and continues counting, providing a periodic interrupt. None of the bits in the I2CISTAT Register are set, allowing the BRG in the I^2C controller to be used as a general-purpose timer when the I^2C controller is disabled.

16.2.3. Start and Stop Conditions

The master generates the start and stop conditions to start or end a transaction. To start a transaction, the I^2C controller generates a start condition by pulling the SDA signal Low while SCL is High. To complete a transaction, the I^2C controller generates a stop condition by creating a Low-to-High transition of the SDA signal while the SCL signal is High. These start and stop events occur when the start and stop bits in the I^2C Control Register are written by software to begin or end a transaction. Any byte transfer currently under way including the Acknowledge phase finishes before the start or stop condition occurs.

16.2.4. Software Control of I^2C Transactions

The I^2C controller is configured via the I^2C Control and I^2C Mode registers. The MODE[1:0] field of the I^2C Mode Register allows the configuration of the I^2C controller for MASTER/SLAVE or SLAVE ONLY Mode, and configures the slave for 7-bit or 10-bit addressing recognition.

MASTER/SLAVE Mode can be used for:

- MASTER ONLY operation in a *single master/one or more slave* I^2C system
- MASTER/SLAVE in a *multimaster/multislave* I^2C system
- SLAVE ONLY operation in an I^2C system

In SLAVE ONLY Mode, the start bit of the I^2C Control Register is ignored (i.e., software cannot initiate a master transaction by accident) and operation to SLAVE ONLY Mode is restricted thereby preventing accidental operation in MASTER Mode. The software controls I^2C transactions by enabling the I^2C controller interrupt in the interrupt controller or by polling the I^2C Status Register.

To use interrupts, the I^2C interrupt must be enabled in the interrupt controller and followed by executing an EI instruction. The TXI bit in the I^2C Control Register must be set to enable transmit interrupts. An I^2C interrupt service routine then checks the I^2C Status Register to determine the cause of the interrupt.

To control transactions by polling, the TDRE, RDRF, SAM, ARBLST, SPRS and NCKI interrupt bits in the I^2C Status Register should be polled. The TDRE bit asserts regardless of the state of the TXI bit.

16.2.5. Master Transactions

The following sections describe master read and write transactions to both 7-bit and 10-bit slaves.

16.2.5.1. Master Arbitration

If a master loses arbitration during the address byte it releases the SDA line, switches to SLAVE Mode and monitors the address to determine if it is selected as a slave. If a master loses arbitration during the transmission of a data byte, it releases the SDA line and waits for the next stop or start condition.

The master detects a loss of arbitration when a 1 is transmitted but a 0 is received from the bus in the same bit-time. This loss occurs if more than one master is simultaneously accessing the bus. Loss of arbitration occurs during the address phase (two or more Masters accessing different slaves) or during the data phase, when the masters are attempting to write different data to the same slave.

When a master loses arbitration, the software is informed by means of the Arbitration Lost interrupt. The software can repeat the same transaction at a later time.

A special case can occur when a slave transaction starts just before the software attempts to start a new master transaction by setting the start bit. In this case, the state machine enters its slave states before the start bit is set and as a result the I^2C controller will not arbitrate. If a slave address match occurs and the I^2C controller receives/transmits data, the start bit is cleared and an Arbitration Lost interrupt is asserted. The software can minimize the chance of this instance occurring by checking the BUSY bit in the I2CSTATE Register before initiating a master transaction. If a slave address match does not occur, the Arbitration Lost interrupt will not occur and the start bit will not be cleared. The I^2C controller will initiate the master transaction after the I^2C bus is no longer busy.

16.2.5.2. Master Address-Only Transactions

It is sometimes preferable to perform an address-only transaction to determine if a particular slave device is able to respond. This transaction can be performed by monitoring the ACKV bit in the I2CSTATE Register after the address has been written to the I2CDATA Register and the start bit has been set. After the ACKV bit is set, the ACK bit in the I2CSTATE Register determines if the slave is able to communicate. The stop bit must be set in the I2CCTL Register to terminate the transaction without transferring data. For a 10-bit slave address, if the first address byte is acknowledged, the second address byte should also be sent to determine if the preferred slave is responding.

Another approach is to set both the stop and start bits (for sending a 7-bit address). After both bits have been cleared (7-bit address has been sent and transaction is complete), the ACK bit can be read to determine if the slave has acknowledged. For a 10-bit slave, set the stop bit after the second TDRE interrupt (which indicates that the second address byte is being sent).

16.2.5.3. Master Transaction Diagrams

In the following transaction diagrams, the shaded regions indicate the data that is transferred from the master to the slave, and the unshaded regions indicate the data that is transferred from the slave to the master. The transaction field labels are defined as follows:

S	Start
W	Write

A	Acknowledge
A	Not Acknowledge
P	Stop

16.2.5.4. Master Write Transaction with a 7-Bit Address

Figure 42 shows the data transfer format from a master to a 7-bit addressed slave.

S	Slave Address	W=0	A	Data	A	Data	A	Data	A/$\overline{A}$	P/S

Figure 42. Data Transfer Format, Master Write Transaction with a 7-Bit Address

Observe the following steps for a master transmit operation to a 7-bit addressed slave:

1. The software initializes the MODE field in the I^{2}C Mode Register for MASTER/SLAVE Mode with either a 7-bit or 10-bit slave address. The MODE field selects the address width for this mode when addressed as a slave (but not for the remote slave). The software asserts the IEN bit in the I^{2}C Control Register.
2. The software asserts the TXI bit of the I^{2}C Control Register to enable transmit interrupts.
3. The I^{2}C interrupt asserts, because the I^{2}C Data Register is empty.
4. The software responds to the TDRE bit by writing a 7-bit slave address plus the write bit (which is cleared to 0) to the I^{2}C Data Register.
5. The software sets the start bit of the I^{2}C Control Register.
6. The I^{2}C controller sends a start condition to the I^{2}C slave.
7. The I^{2}C controller loads the I^{2}C Shift Register with the contents of the I^{2}C Data Register.
8. After one bit of the address has been shifted out by the SDA signal, the transmit interrupt asserts.
9. The software responds by writing the transmit data into the I^{2}C Data Register.
10. The I^{2}C controller shifts the remainder of the address and the write bit out via the SDA signal.
11. The I^{2}C slave sends an Acknowledge (by pulling the SDA signal Low) during the next High period of SCL. The I^{2}C controller sets the ACK bit in the I^{2}C Status Register.

    If the slave does not acknowledge the address byte, the I^{2}C controller sets the NCKI bit in the I^{2}C Status Register, sets the ACKV bit and clears the ACK bit in the I^{2}C State Register. The software responds to the Not Acknowledge interrupt by setting the stop bit and clearing the TXI bit. The I^{2}C controller flushes the Transmit Data Register, sends a stop condition on the bus and clears the stop and NCKI bits. The transaction is complete and the following steps can be ignored.

12. The I^{2}C controller loads the contents of the I^{2}C Shift Register with the contents of the I^{2}C Data Register.
13. The I^{2}C controller shifts the data out via the SDA signal. After the first bit is sent, the transmit interrupt asserts.
14. If more bytes remain to be sent, return to Step 9.
15. When there is no more data to be sent, the software responds by setting the stop bit of the I^{2}C Control Register (or the start bit to initiate a new transaction).
16. If no additional transaction is queued by the master, the software can clear the TXI bit of the I^{2}C Control Register.
17. The I^{2}C controller completes transmission of the data on the SDA signal.
18. The I^{2}C controller sends a stop condition to the I^{2}C bus.

Note: If the slave terminates the transaction early by responding with a Not Acknowledge during the transfer, the I^{2}C controller asserts the NCKI interrupt and halts. The software must terminate the transaction by setting either the stop bit (end transaction) or the start bit (end this transaction, start a new one). In this case, it is not necessary for software to set the FLUSH bit of the I2CCTL Register to flush the data that was previously written but not transmitted. The I^{2}C controller hardware automatically flushes transmit data in the Not Acknowledge case.

16.2.5.5. Master Write Transaction with a 10-Bit Address

Figure 43 shows the data transfer format from a master to a 10-bit addressed slave.

S	Slave Address 1st Byte	W=0	A	Slave Address 2nd Byte	A	Data	A	Data	$A/\overline{A}$	F/S

Figure 43. Data Transfer Format, Master Write Transaction with a 10-Bit Address

The first 7 bits transmitted in the first byte are `11110xx`. The 2 `xx` bits are the two most significant bits of the 10-bit address. The lowest bit of the first byte transferred is the read/write control bit (which is cleared to 0). The transmit operation is performed in the same manner as 7-bit addressing.

Observe the following steps for a master transmit operation to a 10-bit addressed slave:

1. The software initializes the MODE field in the I^{2}C Mode Register for MASTER/SLAVE Mode with 7- or 10-bit addressing (the I^{2}C bus protocol allows the mixing of slave address types). The MODE field selects the address width for this mode when addressed as a slave (but not for the remote slave). The software asserts the IEN bit in the I^{2}C Control Register.
2. The software asserts the TXI bit of the I^{2}C Control Register to enable transmit interrupts.
3. The I^{2}C interrupt asserts because the I^{2}C Data Register is empty.
4. The software responds to the TDRE interrupt by writing the first slave address byte (`11110xx0`). The least-significant bit must be 0 for the write operation.
5. The software asserts the start bit of the I^{2}C Control Register.
6. The I^{2}C controller sends a start condition to the I^{2}C slave.
7. The I^{2}C controller loads the I^{2}C Shift Register with the contents of the I^{2}C Data Register.
8. After one bit of the address is shifted out by the SDA signal, the transmit interrupt asserts.
9. The software responds by writing the second byte of address into the contents of the I^{2}C Data Register.
10. The I^{2}C controller shifts the remainder of the first byte of the address and the write bit out via the SDA signal.
11. The I^{2}C slave sends an Acknowledge by pulling the SDA signal Low during the next High period of SCL. The I^{2}C controller sets the ACK bit in the I^{2}C Status Register.

    If the slave does not acknowledge the first address byte, the I^{2}C controller sets the NCKI bit in the I^{2}C Status Register, sets the ACKV bit and clears the ACK bit in the I^{2}C State Register. The software responds to the Not Acknowledge interrupt by setting the stop bit and clearing the TXI bit. The I^{2}C controller flushes the second address byte from the Data Register, sends a stop condition on the bus, and clears the stop and NCKI bits. The transaction is complete and the following steps can be ignored.
12. The I^{2}C controller loads the I^{2}C Shift Register with the contents of the I^{2}C Data Register (2nd address byte).
13. The I^{2}C controller shifts the second address byte out via the SDA signal. After the first bit has been sent, the transmit interrupt asserts.
14. The software responds by writing the data to be written out to the I^{2}C Control Register.
15. The I^{2}C controller shifts out the remainder of the second byte of the slave address (or ensuring data bytes, if looping) via the SDA signal.

16. The I^{2}C slave sends an Acknowledge by pulling the SDA signal Low during the next High period of SCL. The I^{2}C controller sets the ACK bit in the I^{2}C Status Register. If the slave does not acknowledge, see the second paragraph of Step 11.
17. The I^{2}C controller shifts the data out by the SDA signal. After the first bit is sent, the transmit interrupt asserts.
18. If more bytes remain to be sent, return to Step 14.
19. The software responds by asserting the stop bit of the I^{2}C Control Register.
20. The I^{2}C controller completes transmission of the data on the SDA signal.
21. The I^{2}C controller sends a stop condition to the I^{2}C bus.

Note: If the slave responds with a Not Acknowledge during the transfer, the I^{2}C controller asserts the NCKI bit, sets the ACKV bit, clears the ACK bit in the I^{2}C State Register and halts. The software terminates the transaction by setting either the stop bit (end transaction) or the start bit (end this transaction, start a new one). The Transmit Data Register is flushed automatically.

16.2.5.6. Master Read Transaction with a 7-Bit Address

Figure 44 shows the data transfer format for a read operation to a 7-bit addressed slave.

S	Slave Address	R=1	A	Data	A	Data	$\overline{A}$	P/S

Figure 44. Data Transfer Format, Master Read Transaction with a 7-Bit Address

Observe the following steps for a Master Read operation to a 7-bit addressed slave:

1. The software initializes the MODE field in the I^{2}C Mode Register for MASTER/ SLAVE Mode with 7- or 10-bit addressing (the I^{2}C bus protocol allows the mixing of slave address types). The MODE field selects the address width for this mode when addressed as a slave (but not for the remote slave). The software asserts the IEN bit in the I^{2}C Control Register.
2. The software writes the I^{2}C Data Register with a 7-bit slave address, plus the read bit (which is set to 1).
3. The software asserts the start bit of the I^{2}C Control Register.
4. If this operation is a single-byte transfer, the software asserts the NAK bit of the I^{2}C Control Register so that after the first byte of data has been read by the I^{2}C controller, a Not Acknowledge instruction is sent to the I^{2}C slave.
5. The I^{2}C controller sends a start condition.

6. The I^2C controller sends the address and read bit out via the SDA signal.
7. The I^2C slave acknowledges the address by pulling the SDA signal Low during the next High period of SCL.

   If the slave does not acknowledge the address byte, the I^2C controller sets the NCKI bit in the I^2C Status Register, sets the ACKV bit and clears the ACK bit in the I^2C State Register. The software responds to the Not Acknowledge interrupt by setting the stop bit. The I^2C controller sends a stop condition on the bus and clears the stop and NCKI bits. The transaction is complete and the following steps can be ignored.
8. The I^2C controller shifts in a byte of data from the I^2C slave.
9. The I^2C controller asserts the receive interrupt.
10. The software responds by reading the I^2C Data Register. If the next data byte is to be the final byte, the software must set the NAK bit of the I^2C Control Register.
11. The I^2C controller sends an Acknowledge or Not Acknowledge to the I^2C slave based on the value of the NAK bit.
12. If there are more bytes to transfer, the I^2C controller returns to Step 8.
13. A NAK interrupt (NCKI bit in I2CISTAT) is generated by the I^2C controller.
14. The software responds by setting the stop bit of the I^2C Control Register.
15. A stop condition is sent to the I^2C slave.

16.2.5.7. Master Read Transaction with a 10-Bit Address

Figure 45 shows the read transaction format for a 10-bit addressed slave.

S	Slave Address 1st Byte	W=0	A	Slave Address 2nd Byte	A	S	Slave Address 1st Byte	R=1	A	Data	A	Data	$\overline{A}$	P

Figure 45. Data Transfer Format, Master Read Transaction with a 10-Bit Address

The first 7 bits transmitted in the first byte are `11110xx`. The two `xx` bits are the two most-significant bits of the 10-bit address. The lowest bit of the first byte transferred is the write control bit.

Observe the following data transfer procedure for a read operation to a 10-bit addressed slave:

1. The software initializes the MODE field in the I^2C Mode Register for MASTER/ SLAVE Mode with 7- or 10-bit addressing (the I^2C bus protocol allows the mixing of slave address types). The MODE field selects the address width for this mode when addressed as a slave (but not for the remote slave). The software asserts the IEN bit in the I^2C Control Register.

2. The software writes `11110b`, followed by the two most-significant address bits and a 0 (write) to the I^2C Data Register.
3. The software asserts the start bit of the I^2C Control Register.
4. The I^2C controller sends a start condition.
5. The I^2C controller loads the I^2C Shift Register with the contents of the I^2C Data Register.
6. After the first bit has been shifted out, a transmit interrupt is asserted.
7. The software responds by writing the least significant eight bits of address to the I^2C Data Register.
8. The I^2C controller completes shifting of the first address byte.
9. The I^2C slave sends an Acknowledge by pulling the SDA signal Low during the next High period of SCL.

   If the slave does not acknowledge the address byte, the I^2C controller sets the NCKI bit in the I^2C Status Register, sets the ACKV bit and clears the ACK bit in the I^2C State Register. The software responds to the Not Acknowledge interrupt by setting the stop bit. The I^2C controller flushes the Transmit Data Register, sends the stop condition on the bus and clears the stop and NCKI bits. The transaction is complete and the following steps can be ignored.
10. The I^2C controller loads the I^2C Shift Register with the contents of the I^2C Data Register (the lower byte of the 10-bit address).
11. The I^2C controller shifts out the next eight bits of the address. After the first bit shifts, the I^2C controller generates a transmit interrupt.
12. The software responds by setting the start bit of the I^2C Control Register to generate a repeated start condition.
13. The software writes `11110b`, followed by the 2-bit slave address and a 1 (read) to the I^2C Data Register.
14. If the user chooses to read-only one byte, the software responds by setting the NAK bit of the I^2C Control Register.
15. After the I^2C controller shifts out the address bits listed in Step 9 (the second address transfer), the I^2C slave sends an Acknowledge by pulling the SDA signal Low during the next High period of SCL.

    If the slave does not acknowledge the address byte, the I^2C controller sets the NCKI bit in the I^2C Status Register, sets the ACKV bit and clears the ACK bit in the I^2C State Register. The software responds to the Not Acknowledge interrupt by setting the stop bit. The I^2C controller flushes the Transmit Data Register, sends the stop condition on the bus and clears the stop and NCKI bits. The transaction is complete and the following steps can be ignored.

16. The I^{2}C controller sends a repeated start condition.
17. The I^{2}C controller loads the I^{2}C Shift Register with the contents of the I^{2}C Data Register (the third address transfer).
18. The I^{2}C controller sends `11110b`, followed by the two most-significant bits of the slave read address and a 1 (read).
19. The I^{2}C slave sends an Acknowledge by pulling the SDA signal Low during the next High period of SCL.
20. The I^{2}C controller shifts in a byte of data from the slave.
21. The I^{2}C controller asserts the Receive interrupt.
22. The software responds by reading the I^{2}C Data Register. If the next data byte is to be the final byte, the software must set the NAK bit of the I^{2}C Control Register.
23. The I^{2}C controller sends an Acknowledge or Not Acknowledge to the I^{2}C slave, based on the value of the NAK bit.
24. If there are more bytes to transfer, the I^{2}C controller returns to Step 20.
25. The I^{2}C controller generates a NAK interrupt (the NCKI bit in the I2CISTAT Register).
26. The software responds by setting the stop bit of the I^{2}C Control Register.
27. A stop condition is sent to the I^{2}C slave.

16.2.6. Slave Transactions

The following subsections describe read and write transactions to the I^{2}C controller configured for 7-bit and 10-bit slave modes.

16.2.6.1. Slave Address Recognition

The following two slave address recognition options are supported; a description of each follows.

- Slave 7-Bit Address Recognition Mode
- Slave 10-Bit Address Recognition Mode

Slave 7-Bit Address Recognition Mode. If IRM=0 during the address phase and the controller is configured for MASTER/SLAVE or SLAVE 7-BIT ADDRESS Mode, the hardware detects a match to the 7-bit slave address defined in the I2CSLVAD Register and generates the slave address match interrupt (the SAM bit=1 in the I2CISTAT Register). The I^{2}C controller automatically responds during the Acknowledge phase with the value in the NAK bit of the I2CCTL Register.

Slave 10-Bit Address Recognition Mode. If IRM=0 during the address phase and the controller is configured for MASTER/SLAVE or SLAVE 10-BIT ADDRESS Mode, the hardware detects a match to the 10-bit slave address defined in the I2CMODE and I2CSLVAD registers and generates the slave address match interrupt (the SAM bit=1 in the I2CISTAT Register). The I^2C controller automatically responds during the Acknowledge phase with the value in the NAK bit of the I2CCTL Register.

16.2.6.2. General Call and Start Byte Address Recognition

If GCE=1 and IRM=0 during the address phase and the controller is configured for master/slave or slave in either 7- or 10-bit address modes, the hardware detects a match to the General Call Address or the start byte and generates the slave address match interrupt. A General Call Address is a 7-bit address of all zeroes, with the $R/\overline{W}$ bit=0. A start byte is a 7-bit address of all zeroes, with the $R/\overline{W}$ bit=1. The SAM and GCA bits are set in the I2CISTAT Register. The RD bit in the I2CISTAT Register distinguishes a General Call Address from a start byte which is cleared to 0 for a General Call Address). For a General Call Address, the I^2C controller automatically responds during the address acknowledge phase with the value in the NAK bit of the I2CCTL Register. If the software is set to process the data bytes associated with the GCA bit, the IRM bit can optionally be set following the SAM interrupt to allow the software to examine each received data byte before deciding to set or clear the NAK bit. A start byte will not be acknowledged – a requirement of the I^2C specification.

16.2.6.3. Software Address Recognition

To disable hardware address recognition, the IRM bit must be set to 1 prior to the reception of the address byte(s). When IRM=1, each received byte generates a receive interrupt (RDRF=1 in the I2CISTAT Register). The software must examine each byte and determine whether to set or clear the NAK bit. The slave holds SCL Low during the acknowledge phase until the software responds by writing to the I2CCTL Register. The value written to the NAK bit is used by the controller to drive the I^2C bus, then releasing the SCL. The SAM and GCA bits are not set when IRM=1 during the address phase, but the RD bit is updated based on the first address byte.

16.2.6.4. Slave Transaction Diagrams

In the following transaction diagrams, the shaded regions indicate data transferred from the master to the slave and the unshaded regions indicate the data transferred from the slave to the master. The transaction field labels are defined as follows:

S	Start
W	Write
A	Acknowledge
A	Not Acknowledge
P	Stop

16.2.6.5. Slave Receive Transaction with 7-Bit Address

The data transfer format for writing data from a master to a slave in 7-bit address mode is shown in Figure 46. The procedure that follows describes the I^{2}C Master/Slave Controller operating as a slave in 7-bit addressing mode and receiving data from the bus master.

S	Slave Address	W=0	A	Data	A	Data	A	Data	$A/\overline{A}$	P/S

Figure 46. Data Transfer Format, Slave Receive Transaction with 7-Bit Address

1. The software configures the controller for operation as a slave in 7-bit addressing mode, as follows:
   a. Initialize the MODE field in the I^{2}C Mode Register for either SLAVE ONLY Mode or MASTER/SLAVE Mode with 7-bit addressing.
   b. Optionally set the GCE bit.
   c. Initialize the SLA[6:0] bits in the I^{2}C Slave Address Register.
   d. Set IEN=1 in the I^{2}C Control Register. Set NAK=0 in the I^{2}C Control Register.
2. The bus master initiates a transfer, sending the address byte. In SLAVE Mode, the I^{2}C controller recognizes its own address and detects that $R/\overline{W}$ bit=0 (written from the master to the slave). The I^{2}C controller acknowledges, indicating it is available to accept the transaction. The SAM bit in the I2CISTAT Register is set to 1, causing an interrupt. The RD bit in the I2CISTAT Register is cleared to 0, indicating a write to the slave. The I^{2}C controller holds the SCL signal Low, waiting for the software to load the first data byte.
3. The software responds to the interrupt by reading the I2CISTAT Register (which clears the SAM bit). After seeing the SAM bit to 1, the software checks the RD bit. Because RD=0, no immediate action is required until the first byte of data is received. If software is only able to accept a single byte, it sets the NAK bit in the I2CCTL Register at this time.
4. The master detects the Acknowledge and sends the byte of data.
5. The I^{2}C controller receives the data byte and responds with an Acknowledge or a Not Acknowledge, depending on the state of the NAK bit in the I2CCTL Register. The I^{2}C controller generates the receive data interrupt by setting the RDRF bit in the I2CISTAT Register.
6. The software responds by reading the I2CISTAT Register, finding the RDRF bit=1 and reading the I2CDATA Register clearing the RDRF bit. If software can accept only one more data byte, it sets the NAK bit in the I2CCTL Register.
7. The master and slave loops through Step 4 to Step 6 until the master detects a Not Acknowledge instruction or runs out of data to send.

8. The master sends the stop or restart signal on the bus. Either of these signals can cause the I^{2}C controller to assert a stop interrupt (the stop bit=1 in the I2CISTAT Register). Because the slave received data from the master, the software takes no action in response to the stop interrupt other than reading the I2CISTAT Register to clear the stop bit in the I2CISTAT Register.

16.2.6.6. Slave Receive Transaction with 10-Bit Address

The data transfer format for writing data from a master to a slave with 10-bit addressing is shown in Figure 47. The procedure that follows describes the I^{2}C Master/Slave Controller operating as a slave in 10-bit addressing mode and receiving data from the bus master.

S	Slave Address 1st Byte	W=0	A	Slave Address 2nd Byte	A	Data	A	Data	$A/\overline{A}$	P/S

Figure 47. Data Transfer Format, Slave Receive Transaction with 10-Bit Address

1. The software configures the controller for operation as a slave in 10-bit addressing mode, as follows:
   a. Initialize the MODE field in the I2CMODE Register for either SLAVE ONLY Mode or MASTER/SLAVE Mode with 10-bit addressing.
   b. Optionally set the GCE bit.
   c. Initialize the SLA[7:0] bits in the I2CSLVAD Register and the SLA[9:8] bits in the I2CMODE Register.
   d. Set IEN=1 in the I2CCTL Register. Set NAK=0 in the I^{2}C Control Register.

2. The master initiates a transfer, sending the first address byte. The I^{2}C controller recognizes the start of a 10-bit address with a match to SLA[9:8] and detects R/$\overline{W}$ bit=0 (a write from the master to the slave). The I^{2}C controller acknowledges, indicating it is available to accept the transaction.
3. The master sends the second address byte. The SLAVE Mode I^{2}C controller detects an address match between the second address byte and SLA[7:0]. The SAM bit in the I2CISTAT Register is set to 1, thereby causing an interrupt. The RD bit is cleared to 0, indicating a write to the slave. The I^{2}C controller acknowledges, indicating it is available to accept the data.
4. The software responds to the interrupt by reading the I2CISTAT Register, which clears the SAM bit. Because RD=0, no immediate action is taken by the software until the first byte of data is received. If the software is only able to accept a single byte, it sets the NAK bit in the I2CCTL Register.
5. The master detects the Acknowledge and sends the first byte of data.

6. The I^2C controller receives the first byte and responds with Acknowledge or Not Acknowledge, depending on the state of the NAK bit in the I2CCTL Register. The I^2C controller generates the receive data interrupt by setting the RDRF bit in the I2CISTAT Register.
7. The software responds by reading the I2CISTAT Register, finding the RDRF bit=1 and then reading the I2CDATA Register, which clears the RDRF bit. If the software can accept only one more data byte, it sets the NAK bit in the I2CCTL Register.
8. The master and slave loops through Step 5 to Step 7 until the master detects a Not Acknowledge instruction or runs out of data to send.
9. The master sends the stop or restart signal on the bus. Either of these signals can cause the I^2C controller to assert the stop interrupt (the stop bit=1 in the I2CISTAT Register). Because the slave received data from the master, the software takes no action in response to the stop interrupt other than reading the I2CISTAT Register to clear the stop bit.

16.2.6.7. Slave Transmit Transaction With 7-Bit Address

The data transfer format for a master reading data from a slave in 7-bit address mode is shown in Figure 48. The procedure that follows describes the I^2C Master/Slave Controller operating as a slave in 7-bit addressing mode and transmitting data to the bus master.

S	Slave Address	R=1	A	Data	A	Data	$\overline{A}$	P/S

Figure 48. Data Transfer Format, Slave Transmit Transaction with 7-bit Address

1. The software configures the controller for operation as a slave in 7-bit addressing mode, as follows:
   a. Initialize the MODE field in the I^2C Mode Register for either SLAVE ONLY Mode or MASTER/SLAVE Mode with 7-bit addressing.
   b. Optionally set the GCE bit.
   c. Initialize the SLA[6:0] bits in the I^2C Slave Address Register.
   d. Set IEN=1 in the I^2C Control Register. Set NAK=0 in the I^2C Control Register.

2. The master initiates a transfer by sending the address byte. The SLAVE Mode I^2C controller finds an address match and detects that the R/$\overline{W}$ bit=1 (read by the master from the slave). The I^2C controller acknowledges, indicating that it is ready to accept the transaction. The SAM bit in the I2CISTAT Register is set to 1, causing an interrupt. The RD bit is set to 1, indicating a read from the slave.
3. The software responds to the interrupt by reading the I2CISTAT Register, thereby clearing the SAM bit. Because RD=1, the software responds by loading the first data byte into the I2CDATA Register. The software sets the TXI bit in the I2CCTL Regis-

ter to enable transmit interrupts. When the master initiates the data transfer, the I^{2}C controller holds SCL Low until the software has written the first data byte to the I2CDATA Register.

4. SCL is released and the first data byte is shifted out.
5. After the first bit of the first data byte has been transferred, the I^{2}C controller sets the TDRE bit, which asserts the transmit data interrupt.
6. The software responds to the transmit data interrupt (TDRE=1) by loading the next data byte into the I2CDATA Register, which clears TDRE.
7. After the data byte has been received by the master, the master transmits an Acknowledge instruction (or a Not Acknowledge instruction if this byte is the final data byte).
8. The bus cycles through Step 5 to Step 7 until the final byte has been transferred. If the software has not yet loaded the next data byte when the master brings SCL Low to transfer the most significant data bit, the slave I^{2}C controller holds SCL Low until the Data Register has been written. When a Not Acknowledge instruction is received by the slave, the I^{2}C controller sets the NCKI bit in the I2CISTAT Register, causing the Not Acknowledge interrupt to be generated.
9. The software responds to the Not Acknowledge interrupt by clearing the TXI bit in the I2CCTL Register, and by asserting the FLUSH bit of the I2CCTL Register to *empty* the Data Register.
10. When the master has completed the final acknowledge cycle, it asserts a stop or restart condition on the bus.
11. The slave I^{2}C controller asserts the stop/restart interrupt (i.e., sets the SPRS bit in the I2CISTAT Register).
12. The software responds to the stop/restart interrupt by reading the I2CISTAT Register, which clears the SPRS bit.

16.2.6.8. Slave Transmit Transaction With 10-Bit Address

The data transfer format for a master reading data from a slave with 10-bit addressing is shown in Figure 49. The following procedure describes the I^{2}C Master/Slave Controller operating as a slave in 10-bit addressing mode, transmitting data to the bus master.

S	Slave Address 1st Byte	W=0	A	Slave Address 2nd Byte	A	S	Slave Address 1st Byte	R=1	A	Data	A	Data	$\overline{A}$	P

Figure 49. Data Transfer Format, Slave Transmit Transaction with 10-Bit Address

1. The software configures the controller for operation as a slave in 10-bit addressing mode.

a. Initialize the MODE field in the I^{2}C Mode Register for either SLAVE ONLY Mode or MASTER/SLAVE Mode with 10-bit addressing.
b. Optionally set the GCE bit.
c. Initialize the SLA[7:0] bits in the I2CSLVAD Register and SLA[9:8] in the I^{2}C MODE Register.
d. Set IEN=1 and NAK=0 in the I^{2}C Control Register.

2. The master initiates a transfer by sending the first address byte. The SLAVE Mode I^{2}C controller recognizes the start of a 10-bit address with a match to SLA[9:8], and detects R/$\overline{W}$ bit=0 (i.e., a write from the master to the slave). The I^{2}C controller acknowledges, indicating it is available to accept the transaction.
3. The master sends the second address byte. The SLAVE Mode I^{2}C controller compares the second address byte with the value in SLA[7:0]. If there is a match, the SAM bit in the I2CISTAT Register is set=1, causing a slave address match interrupt. The RD bit is set=0, indicating a write to the slave. If a match occurs, the I^{2}C controller acknowledges on the I^{2}C bus, indicating it is available to accept the data.
4. The software responds to the slave address match interrupt by reading the I2CISTAT Register, which clears the SAM bit. Because the RD bit=0, no further action is required.
5. The master sees the Acknowledge and sends a restart instruction, followed by the first address byte with R/$\overline{W}$ set to 1. The SLAVE Mode I^{2}C controller recognizes the restart instruction, follows with the first address byte with a match to SLA[9:8], and detects R/$\overline{W}$=1 (i.e, the master reads from the slave). The slave I^{2}C controller sets the SAM bit in the I2CISTAT Register, which causes the slave address match interrupt. The RD bit is set=1. The SLAVE Mode I^{2}C controller acknowledges on the bus.
6. The software responds to the interrupt by reading the I2CISTAT Register and clearing the SAM bit. The software loads the initial data byte into the I2CDATA Register and sets the TXI bit in the I2CCTL Register.
7. The master starts the data transfer by asserting SCL Low. After the I^{2}C controller has data available to transmit, the SCL is released and the master proceeds to shift the first data byte.
8. After the first bit of the first data byte has been transferred, the I^{2}C controller sets the TDRE bit which asserts the transmit data interrupt.
9. The software responds to the transmit data interrupt by loading the next data byte into the I2CDATA Register.
10. The I^{2}C master shifts in the remainder of the data byte. The master transmits the Acknowledge (or Not Acknowledge, if this byte is the final data byte).
11. The bus cycles through Step 7 to Step 10 until the final byte is transferred. If the software has not yet loaded the next data byte when the master brings SCL Low to trans-

fer the most significant data bit, the slave I²C controller holds SCL Low until the Data Register is written.

When a Not Acknowledge is received by the slave, the I²C controller sets the NCKI bit in the I2CISTAT Register, causing the NAK interrupt to be generated.

12. The software responds to the NAK interrupt by clearing the TXI bit in the I2CCTL Register and by asserting the FLUSH bit of the I2CCTL Register.
13. When the master has completed the Acknowledge cycle of the last transfer, it asserts a stop or restart condition on the bus.
14. The slave I²C controller asserts the stop/restart interrupt (i.e., sets the SPRS bit in the I2CISTAT Register).
15. The software responds to the stop interrupt by reading the I2CISTAT Register and clearing the SPRS bit.

16.3. I²C Control Register Definitions

The I²C Control registers are described in this section.

16.3.1. I²C Data Register

The I²C Data Register listed in Table 139 contains the data that is to be loaded into the Shift Register to transmit onto the I²C bus. This register also contains data that is loaded from the Shift Register after it is received from the I²C bus. The I²C Shift Register is not accessible in the Register File address space, but is used only to buffer incoming and outgoing data.

Writes by the software to the I2CDATA Register are blocked if a slave write transaction is underway (the I²C controller is in SLAVE Mode and data is being received).

Table 139. I²C Data Register (I2CDATA = F50h)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	Data 7	Data 6	Data 5	Data 4	Data 3	Data 2	Data 1	Data 0
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F50h							

Bit	**Description**
[7:0] DATA	**I²C Data Byte**

16.3.2. I²C Interrupt Status Register

The read-only I²C Interrupt Status Register, shown in Table 140, indicates the cause of any current I²C interrupt and provides the status of the I²C controller. When an interrupt occurs, one or more of the TDRE, RDRF, SAM, ARBLST, SPRS or NCKI bits is set. The GCA and RD bits do not generate an interrupt, but instead provide the status associated with the SAM bit interrupt.

Table 140. I²C Interrupt Status Register (I2CISTAT=F51h)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	TDRE	RDRF	SAM	GCA	RD	ARBLST	SPRS	NCKI
Reset	1	0	0	0	0	0	0	0
R/W	R	R	R	R	R	R	R	R
Address	F51h							

Bit	Description
[7] TDRE	**Transmit Data Register Empty** When the I²C controller is enabled, this bit is 1 when the I²C Data Register is empty. When set, this bit causes the I²C controller to generate an interrupt, except when the I²C controller is shifting in data during the reception of a byte or when shifting an address and the RD bit is set. This bit clears by writing to the I2CDATA Register.
[6] RDRF	**Receive Data Register Full** This bit is set=1 when the I²C controller is enabled and the I²C controller has received a byte of data. When asserted, this bit causes the I²C controller to generate an interrupt. This bit clears by reading the I2CDATA Register.
[5] SAM	**Slave Address Match** This bit is set=1 if the I²C controller is enabled in SLAVE Mode and an address is received that matches the unique slave address or General Call Address (if enabled by the GCE bit in the I²C Mode Register). In 10-bit addressing mode, this bit is not set until a match is achieved on both address bytes. When this bit is set, the RD and GCA bits are also valid. This bit clears by reading the I2CISTAT Register.
[4] GCA	**General Call Address** This bit is set in SLAVE Mode when the General Call Address or start byte is recognized (in either 7 or 10 bit SLAVE Mode). The GCE bit in the I²C Mode Register must be set to enable recognition of the General Call Address and start byte. This bit clears when IEN=0 and is updated following the first address byte of each SLAVE Mode transaction. A General Call Address is distinguished from a start byte by the value of the RD bit (RD=0 for General Call Address, 1 for start byte).
[3] RD	**Read** This bit indicates the direction of transfer of the data. It is set when the master is reading data from the slave. This bit matches the least-significant bit of the address byte after the start condition occurs (for both MASTER and SLAVE modes). This bit clears when IEN=0, and is updated following the first address byte of each transaction.

Bit	Description (Continued)
[2] ARBLST	**Arbitration Lost** This bit is set when the I^{2}C controller is enabled in MASTER Mode and loses arbitration (i.e., outputs a 1 on SDA and receives a 0 on SDA). The ARBLST bit clears when the I2CISTAT Register is read.
[1] SPRS	**Stop/Restart Condition Interrupt** This bit is set when the I^{2}C controller is enabled in SLAVE Mode, and detects a stop or restart condition during a transaction directed to this slave. This bit clears when the I2CISTAT Register is read. Read the RSTR bit of the I2CSTATE Register to determine whether the interrupt was caused by a stop or restart condition.
[0] NCKI	**NAK Interrupt** In MASTER Mode, this bit is set when a Not Acknowledge condition is received or sent, and neither the start nor the stop bit is active. In MASTER Mode, this bit can only be cleared by setting the start or stop bits. In SLAVE Mode, this bit is set when a Not Acknowledge condition is received (i.e., a master reading data from a slave), indicating that the master is finished reading. A stop or restart condition follows. In SLAVE Mode this bit clears when the I2CISTAT Register is read.

16.3.3. I^{2}C Control Register

The I^{2}C Control Register, shown in Table 141, enables and configures I^{2}C operation.

Note: The R/W1 bit can be set (written to 1) when IEN=1, but cannot be cleared (written to 0).

Table 141. I^{2}C Control Register (I2CCTL)

Bit	7	6	5	4	3	2	1	0
Field	IEN	START	STOP	BIRQS	TXI	NAK	FLUSH	FILTEN
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W1	R/W1	R/W	R/W	R/W1	W	R/W
Address	F52h							

Bit	Description
[7] IEN	**I^{2}C Enable** This bit enables the I^{2}C controller.

Bit	Description (Continued)
[6] START	**Send Start Condition** When set, this bit causes the I^{2}C controller (when configured as the master) to send a start condition. After it is asserted, this bit is cleared by the I^{2}C controller after it sends the start condition or by deasserting the IEN bit. If this bit is 1, it cannot be cleared by writing to the bit. After this bit is set, a start condition is sent if there is data in the I2CDATA or I^{2}C Shift Register. If there is no data in one of these registers, the I^{2}C controller waits until data is loaded. If this bit is set while the I^{2}C controller is shifting out data, it generates a restart condition after the byte shifts and the Acknowledge phase completes. If the stop bit is also set, it waits until the stop condition is sent before the start condition. If start is set while a SLAVE Mode transaction is underway to this device, the start bit will be cleared and ARBLST bit in the Interrupt Status Register will be set.
[5] STOP	**Send Stop Condition** When set, this bit causes the I^{2}C controller (when configured as the master) to send the stop condition after the byte in the I^{2}C Shift Register has completed transmission or after a byte is received in a receive operation. When set, this bit is reset by the I^{2}C controller after a stop condition has been sent or by deasserting the IEN bit. If this bit is 1, it cannot be cleared to 0 by writing to the register. If a stop is set while a SLAVE Mode transaction is underway, the stop bit is cleared by hardware.
[4] BIRQS	**Baud Rate Generator Interrupt Request Select** This bit is ignored when the I^{2}C controller is enabled. If this bit is set=1 when the I^{2}C controller is disabled (IEN=0), the baud rate generator is used as an additional timer causing an interrupt to occur every time the baud rate generator counts down to one. The baud rate generator runs continuously in this mode, generating periodic interrupts.
[3] TXI	**Enable TDRE Interrupts** This bit enables interrupts when the I^{2}C Data Register is empty.
[2] NAK	**Send NAK** Setting this bit sends a Not Acknowledge condition after the next byte of data has been received. It is automatically deasserted after the Not Acknowledge is sent or the IEN bit is cleared. If this bit is 1, it cannot be cleared to 0 by writing to the register.
[1] FLUSH	**Flush Data** Setting this bit clears the I^{2}C Data Register and sets the TDRE bit to 1. This bit allows flushing of the I^{2}C Data Register when an NAK condition is received after the next data byte is written to the I^{2}C Data Register. Reading this bit always returns 0.
[0] FILTEN	**I^{2}C Signal Filter Enable** Setting this bit enables low-pass digital filters on the SDA and SCL input signals. This function provides the spike suppression filter required in I^{2}C Fast Mode. These filters reject any input pulse with periods less than a full system clock cycle. The filters introduce a 3-system clock cycle latency on the inputs.

16.3.4. I^{2}C Baud Rate High and Low Byte Registers

The I^{2}C Baud Rate High and Low Byte registers, shown in Tables 142 and 143, combine to form a 16-bit reload value, BRG[15:0], for the I^{2}C Baud Rate Generator.

The I^2C baud rate is calculated using the following equation.

$$I^2C\text{ Baud Rate (bits/s)} = \frac{\text{System Clock Frequency (Hz)}}{4 \times \text{BRG[15:0]}}$$

When configured as a general purpose timer, the I^2C baud rate generator interrupt interval is calculated using the following equation.

$$I^2C\text{ Baud Rate (bits/s)} = \frac{\text{System Clock Frequency (Hz)}}{\text{BRG[15:0]}}$$

Note: If BRG=`0000h`, then use `10000h` in the equation.

Table 142. I^2C Baud Rate High Byte Register (I2CBRH=53h)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	BRH							
Reset	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F53h							

Bit	**Description**
[7:0] BRH	**I^2C Baud Rate High Byte** The most significant byte, BRG[15:8], of the I^2C Baud Rate Generator's reload value.

Note: If the DIAG bit in the I^2C Mode Register is set to 1, a read of the I2CBRH Register returns the current value of the I^2C Baud Rate Counter[15:8].

Table 143. I^2C Baud Rate Low Byte Register (I2CBRL=F54h)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	BRL							
Reset	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F54h							

Bit	Description
[7:0] BRL	**I²C Baud Rate Low Byte** The least significant byte, BRG[7:0], of the I²C Baud Rate Generator's reload value.

Note: If the DIAG bit in the I²C Mode Register is set to 1, a read of the I2CBRL Register returns the current value of the I²C Baud Rate Counter[7:0].

16.3.5. I²C State Register

The read-only I²C State Register, shown in Table 144, provides information about the state of the I²C bus and the I²C bus controller. When the DIAG bit of the I²C Mode Register is cleared, this register provides information about the internal state of the I²C controller and I²C bus; see Table 146.

When the DIAG bit of the I²C Mode Register is set, this register returns the value of the I²C controller state machine.

Table 144. I²C State Register (I2CSTATE), Description when DIAG = 1

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	I2CSTATE_H				I2CSTATE_L			
Reset	0	0	0	0	0	0	0	0
R/W	R	R	R	R	R	R	R	R
Address	F55h							

Bit	Description
[7:4] I2CSTATE_H	**I²C State** This field defines the current state of the I²C controller. It is the most significant nibble of the internal state machine. Table 146 defines the states for this field.
[3:0] I2CSTATE_L	**Least Significant Nibble of the I²C State Machine** This field defines the substates for the states defined by I2CSTATE_H. Table 147 defines the values for this field.

Table 145. I^{2}C State Register (I2CSTATE), Description when DIAG=0

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	ACKV	ACK	AS	DS	10B	RSTR	SCLOUT	BUSY
Reset	0	0	0	0	0	0	1	0
R/W	R	R	R	R	R	R	R	R
Address	F55h							

Bit	**Description**
[7] ACKV	**ACK Valid** This bit is set, if sending data (master or slave) and the ACK bit in this register is valid for the byte just transmitted. This bit can be monitored if it is appropriate for software to verify the ACK value before writing the next byte to be sent. To operate in this mode, the Data Register must not be written when TDRE asserts; instead, the software waits for ACKV to assert. This bit clears when transmission of the next byte begins or the transaction is ended by a stop or restart condition.
[6] ACK	**Acknowledge** This bit indicates the status of the Acknowledge for the last byte transmitted or received. This bit is set for an Acknowledge and cleared for a Not Acknowledge condition.
[5] AS	**Address State** This bit is active High while the address is being transferred on the I^{2}C bus.
[4] DS	**Data State** This bit is active High while the data is being transferred on the I^{2}C bus.
[3] 10B	**10B** This bit indicates whether a 7-bit or 10-bit address is being transmitted when operating as a master. After the start bit is set, if the five most-significant bits of the address are 11110b, this bit is set. When set, it is Reset after the address has been sent.
[2] RSTR	**Restart** This bit is updated each time a stop or restart interrupt occurs (SPRS bit set in I2CISTAT Register). 0: Stop condition. 1: Restart condition.
[1] SCLOUT	**Serial Clock Output** Current value of Serial Clock being output onto the bus. The actual values of the SCL and SDA signals on the I^{2}C bus can be observed via the GPIO Input Register.
[0] BUSY	**I^{2}C Bus Busy** 0: No activity on the I^{2}C Bus. 1: A transaction is underway on the I^{2}C bus.

Table 146. I2CSTATE_H

State Encoding	State Name	State Description
0000	Idle	I^{2}C bus is idle or I^{2}C controller is disabled.
0001	Slave Start	I^{2}C controller has received a start condition.
0010	Slave Bystander	Address did not match; ignore remainder of transaction.
0011	Slave Wait	Waiting for stop or restart condition after sending a Not Acknowledge instruction.
0100	Master Stop2	Master completing stop condition (SCL=1, SDA=1).
0101	Master Start/Restart	MASTER Mode sending start condition (SCL=1, SDA=0).
0110	Master Stop1	Master initiating stop condition (SCL=1, SDA=0).
0111	Master Wait	Master received a Not Acknowledge instruction, waiting for software to assert stop or start control bits.
1000	Slave Transmit Data	Nine substates, one for each data bit and one for the Acknowledge.
1001	Slave Receive Data	Nine substates, one for each data bit and one for the Acknowledge.
1010	Slave Receive Addr1	Slave receiving first address byte (7- and 10-bit addressing) Nine substates, one for each address bit and one for the Acknowledge.
1011	Slave Receive Addr2	Slave receiving second address byte (10-bit addressing) nine substates, one for each address bit and one for the Acknowledge.
1100	Master Transmit Data	Nine substates, one for each data bit and one for the Acknowledge.
1101	Master Receive Data	Nine substates, one for each data bit and one for the Acknowledge.
1110	Master Transmit Addr1	Master sending first address byte (7- and 10-bit addressing) nine substates, one for each address bit and one for the Acknowledge.
1111	Master Transmit Addr2	Master sending second address byte (10-bit addressing) nine substates, one for each address bit and one for the Acknowledge.

Table 147. I2CSTATE_L

State I2CSTATE_H	Substate I2CSTATE_L	Substate Name	State Description
0000–0100	0000	–	There are no substates for these I2CSTATE_H values.
0110–0111	0000	–	There are no substates for these I2CSTATE_H values.
0101	0000	Master Start	Initiating a new transaction
	0001	Master Restart	Master is ending one transaction and starting a new one without letting the bus go idle.

Table 147. I2CSTATE_L (Continued)

State I2CSTATE_H	Substate I2CSTATE_L	Substate Name	State Description
1000–1111	0111	Send/Receive bit 7	Sending/Receiving most significant bit.
	0110	Send/Receive bit 6	
	0101	Send/Receive bit 5	
	0100	Send/Receive bit 4	
	0011	Send/Receive bit 3	
	0010	Send/Receive bit 2	
	0001	Send/Receive bit 1	
	0000	Send/Receive bit 0	Sending/Receiving least significant bit.
	1000	Send/Receive Acknowledge	Sending/Receiving Acknowledge.

16.3.6. I^{2}C Mode Register

The I^{2}C Mode Register, shown in Table 148, provides control over master vs. slave operating mode, slave address and diagnostic modes.

Table 148. I^{2}C Mode Register (I^{2}C Mode=F56h)

Bit	7	6	5	4	3	2	1	0
Field	Reserved	MODE[1:0]		IRM	GCE	SLA[9:8]		DIAG
Reset	0	0		0	0	0		0
R/W	R/W	R/W		R/W	R/W	R/W		R/W
Address	F56h							

Bit	Description
[7]	**Reserved. Must be 0.**
[6:5] MODE[1:0]	**Selects the I^{2}C Controller Operational Mode** 00: MASTER/SLAVE capable (supports multi-master arbitration) with 7-bit slave address. 01: MASTER/SLAVE capable (supports multi-master arbitration) with 10-bit slave address. 10: Slave Only capable with 7-bit address. 11: Slave Only capable with 10-bit address.

Bit	Description (Continued)
[4] IRM	**Interactive Receive Mode** Valid in SLAVE Mode when software must interpret each received byte before acknowledging. This bit is useful for processing the data bytes following a General Call Address or if software wants to disable hardware address recognition. 0: Acknowledge occurs automatically and is determined by the value of the NAK bit of the I2CCTL Register. 1: A receive interrupt is generated for each byte received (address or data). The SCL is held Low during the Acknowledge cycle until software writes to the I2CCTL Register. The value written to the NAK bit of the I2CCTL Register is output on SDA. This value allows software to Acknowledge or Not Acknowledge after interpreting the associated address/ data byte.
[3] GCE	**General Call Address Enable** Enables reception of messages beginning with the General Call Address or start byte. 0: Do not accept a message with the General Call Address or start byte. 1: Do accept a message with the General Call Address or start byte. When an address match occurs, the GCA and RD bits in the I^{2}C Status Register indicates whether the address matched the General Call Address/start byte or not. Following the General Call Address byte, the software can set the IRM bit that allows software to examine the following data byte(s) before acknowledging.
[2:1] SLA[9:8]	**Slave Address Bits 9 and 8** Initialize with the appropriate slave address value when using 10-bit slave addressing. These bits are ignored when using 7-bit slave addressing.
[0] DIAG	**Diagnostic Mode** Selects read back value of the Baud Rate Reload and State registers. 0: Reading the baud rate registers returns the baud rate register values. Reading the state register returns I^{2}C controller state information. 1: Reading the Baud Rate registers returns the current value of the baud rate counter. Reading the state register returns additional state information.

16.3.7. I²C Slave Address Register

The I²C Slave Address Register, shown in Table 149, provides control over the lower-order address bits used in 7- and 10-bit slave address recognition.

Table 149. I²C Slave Address Register (I2CSLVAD = 57h)

Bit	7	6	5	4	3	2	1	0
Field	SLA[7:0]							
Reset	00h							
R/W	R/W							
Address	F57h							

Bit	Description
[7:0] SLA[7:0]	**Slave Address Bits** Initialize with the appropriate slave address value. When using 7-bit slave addressing, SLA[9:7] are ignored.

Chapter 17. Event System

The F3224 Series devices provide an eight-channel Event System that can route up to eight signals independent of any CPU activity. Any Event System source can be selected to drive a signal on an Event System channel.

These Event System sources are:

- Software
- Timers 0, 1, 2
- Timer A
- Multi-Channel Timer
- Comparators
- GPIO

Event System Destinations:

- Timers 0, 1, 2
- Timer A
- Multi-Channel Timer
- ADC
- GPIO

The Event System is active in all operating modes, including Stop Mode.

17.4. Architecture

This chapter discusses the Event System, including Event System sources, channels, and destinations. A diagram of the Event System is shown in Figure 50.

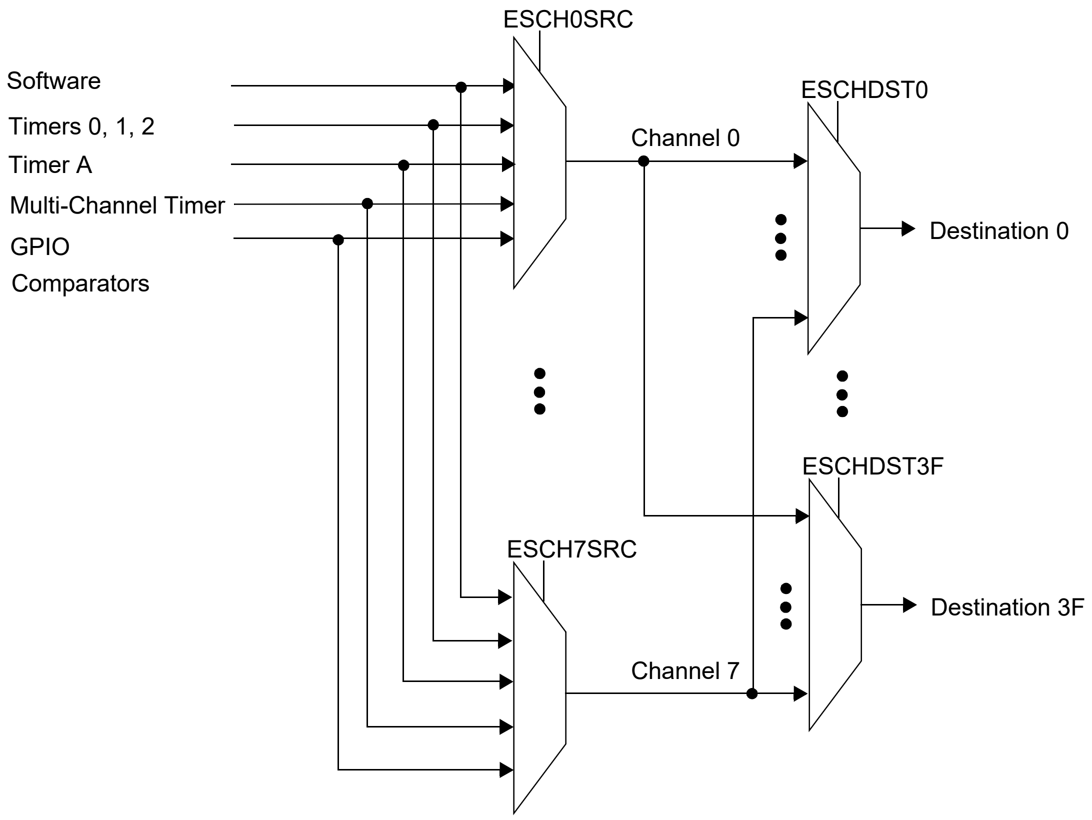


Figure 50. Event System Block Diagram

The Event System provides autonomous communication between peripherals, independently of any CPU activity. The Event System is available in all operating modes, including Stop Mode, to allow for the energy-efficient triggering of peripherals. Furthermore, the direct peripheral-to-peripheral communication reduces software overhead, and allows critical timing signals to pass directly without incurring interrupt latencies.

The Event System can route up to eight signals simultaneously. An event system channel is enabled by writing a nonzero value to the corresponding ESCHxSRC Subregister. Each Event System channel can service one or more destinations.

17.5. Source Selection

The Channel Source subregisters, ESCHxSRC, are used to enable Event System channels, and to select each channel signal source with CHSRCSEL. When enabled, each Event System channel is driven by a single, selectable signal source.

A variety of peripherals and GPIOs can be selected to be an Event System channel signal source. In addition, software can assert a channel (High) by writing CHSRCSEL=`01h`. In this case, the channel will remain asserted (High) until CHSRCSEL is written with a value other than `01h`. When selecting a GPIO as an Event System channel signal source, no additional configuration of the GPIO port alternate function selection registers is required.

The ESCHxSRC subregisters are accessed using the Event System Source Subaddress Register (ESSSA) and Event System Source Subdata Register (ESSSD). An ESCHxSRC Subregister is selected by ESSSA in the ESSSA Register and is accessed by writing/reading ESSSD in the ESSSD Register.

Table 150 lists the available Event System signal sources.

Table 150. Event System Signal Sources

Peripheral	Output Signal
Software	ESCHxCTL=01h
Timer 0	Timer 0 out
	$\overline{\text{Timer 0 out}}$
Timer 1	Timer 1 out
	$\overline{\text{Timer 1 out}}$
Timer 2	Timer 2 out
	$\overline{\text{Timer 2 out}}$
Timer A	Timer A out
Multi-Channel Timer	Multi-Channel Timer out A
	Multi-Channel Timer out B
	Multi-Channel Timer out C
	Multi-Channel Timer out D
Comparator 0	Comparator 0 out
Comparator 1	Comparator 1 out
Comparator 0/1	Comparator 0/1 window detection (C01)
Port A	Port A[7:0] pin
Port C	Port C[7:4] pin
Port E	Port E[3:0] pin

17.6. Destination Selection

A variety of peripherals and GPIOs can be selected to be a destination for an Event System channel. Connecting a destination to an Event System channel is controlled in the Event System Destination Channel subregisters, ESDSTxCH. If DSTCON in the ESDSTxCH

Subregister is set, the destination is connected to the Event System channel specified in DSTCHSEL; otherwise, the destination is connected to logic 0. Because each destination can connect to any Event System channel, multiple peripherals can be a destination for the same Event System channel.

The ESDSTxCH subregisters are accessed using the Event System Destination Subaddress Register (ESDSA) and the Event System Destination Subdata Register (ESDSD). An ESDSTxCH Subregister is selected by ESDSD in the ESDSA Register, and is accessed by writing/reading the ESDSD bit in the ESDSD Register.

Table 151 lists the Event System destinations.

Table 151. Event System Destinations

Peripheral	Signal Connection
Timer 0	Input 0
	Input 1
Timer 1	Input 0
	Input 1
Timer 2	Input 0
	Input 1
Timer A	Input
Multi-Channel Timer	Multi-Channel Timer in A
	Multi-Channel Timer in B
	Multi-Channel Timer in C
	Multi-Channel Timer in D
	Multi-Channel Timer in
ADC	Convert
Port A3, Port C2	ESOUT0
Port A4, Port C3, Port D4	ESOUT1
Port A5, Port B6, Port C6, Port D5	ESOUT2
Port B7, Port C7, Port E3	ESOUT3

Four Event System GPIO outputs, ESOUT[3:0], are available to the port pins listed in Table 151. The GPIO PxAF, PxAFS1, and PxAFS2 subregisters select the Event System output, ESOUT[3:0], that is available for a particular port pin. See the General-Purpose Input/Output chapter on page 46 to learn more regarding alternate function selection.

17.7. Timing Considerations

The Event System essentially performs a multiplexing function. Signals sourced to Event System channels do not go through a synchronization process within the Event System. As such, the signals on Event System channels must be sufficient in duration for detection by their corresponding destinations. Any source and destination pair that are using the same clock, one of Sysclk, Pclk, or the WTO, can be connected to each other via the Event System without concern about source signal duration. When the Event System source and destination pair are using dissimilar clocks, the Event System source signal should be at least 1.5 times the duration of the clock period of the Event System destination to assure that the destination will detect the signal from the source.

17.8. Event System Usage Examples

To illustrate the usage of the Event System, let's examine the following two examples.

Example 1: Triggering Periodic ADC Conversions Using a Timer. In this example, a timer serves as the signal source to Event System channel 0 and the this channel is selected to trigger ADC conversions.

- Select Timer 0 out as the signal source for Event System channel 0, as follows:
  - Write `00h` to the ESSSA Register to select the ESCH0SRC Subregister.
  - Write `10h` to the ESSSD Register. This accesses the ESCH0SRC Subregister to select Timer 0 out as the Event System channel 0 source.

- Configure the ADC conversion parameters. For instance, the ADC could be configured with a window function such that interrupts are generated only when the window is exceeded.
- Configure the ADC to respond to Event System channel 0, as follows:
  - Write `04h` to the ESDSA Register to select the ESDST04CH Subregister.
  - Write `08h` to the ESDSD Register. This accesses ESDST04CH Subregister to enable Event System connection to the ADC and to select channel 0 as input to the ADC.

- Set Timer 0 to Counter Mode with half the desired ADC conversion periodicity, then enable Timer 0. Each Timer 0 Out rising edge will be detected by the ADC, resulting in a new ADC conversion.

Example 2: Observing a Event System Channel on a GPIO. This example builds on the previous example by additionally routing Event System channel 0 to GPIO Port C7.

- Select Timer 0 out as the signal source for channel 0, as follows:

   - Write `00h` to the ESSSA Register to select the ESCH0SRC Subregister.
   - Write `10h` to the ESSSD Register. This accesses ESCH0SRC Subregister to select Timer 0 out as the Event System channel 0 source.

- Configure the ADC conversion parameters. For instance, the ADC could be configured with a window function such that interrupts are generated only when the window is exceeded.
- Configure the ADC to respond to Event System channel 0, as follows:
   - Write `04h` to the ESDSA Register to select the ESDST04CH Subregister.
   - Write `08h` to the ESDSD Register. This accesses the ESDST04CH Subregister to enable Event System connection to the ADC and to select channel 0 as input to the ADC.
- Configure Port C7 to respond to Event System channel 0, as follows:
   - Write `02h` to the GPIO PCADDR Register to select the PCAF Subregister.
   - Write `80h` to the GPIO PCCTL Register to enable alternate function for Port C7.
   - Write `07h` to the GPIO PCADDR Register to select the PCAFS1 Subregister.
   - Write `00h` to the GPIO PCCTL Register as partial selection of ESOUT1 for Port C7.
   - Write `08h` to the GPIO PCADDR Register to select the PCAFS2 Subregister.
   - Write `80h` to the GPIO PCCTL Register to complete selection of ESOUT1 for Port C7.
   - Write `31h` to the ESDSA Register to select the ESDST31CH Subregister.
   - Write `08h` to the ESDSD Register. This accesses the ESDST31CH Subregister to enable Event System connection to the ESOUT1 and to select channel 0 as input to ESOUT1.

- Configure Timer 0 to Counter Mode with half the desired ADC conversion periodicity, then enable Timer 0. Each Timer 0 Out rising edge will be detected by the ADC, resulting in a new ADC conversion; the Timer 0 Out signal will be available on Port C7.

17.9. Event System Register Definitions

Four register addresses provide access to the Event System subregisters that control source and destination selection for each Event System channel. Table 152 lists these Event System registers and subregisters.

Table 152. Event System Registers and Subregisters

Event System Source Selection Registers and Subregisters		
Register Mnemonic	**Address**	**Register Name**
ESSSA	F98h	Event System Source Subaddress Register
ESSSD	F99h	Event System Source Subdata Register
Subregister Mnemonic	**Subregister Address***	**Subregister Name**
ESCHxSRC	0–7h	Event System Channel 0–7 Source subregisters

Event System Destination Selection Registers and Subregisters		
Register Mnemonic	**Address**	**Register Name**
ESDSA	F9Ah	Event System Destination Subaddress Register
ESDSD	F9Bh	Event System Destination Subdata Register
Subregister Mnemonic	**Subregister Address***	**Subregister Name**
ESCHDSTx	0–3Fh	Event System Destination 0–3F Channel subregisters

Note: *The ESSSA Register contains the ESCHxSRC Subregister address; the ESDSA Register contains the ESCH-DSTx Subregister address.

17.9.1. Event System Source Subaddress Register

The Event System Source Subaddress Register (ESSSA), shown in Table 153, selects the Channel Source Subregister (ESCHxSRC) that is accessible through the Event System Source Subdata Register (ESSSD).

Table 153. Event System Source Subaddress Register (ESSSA)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	Reserved					ESSSA		
Reset	0	0	0	0	0	0	0	0
R/W	R	R	R	R	R	R/W	R/W	R/W
Address	F98h							

Bit	Description
[7:4]	**Reserved** These bits are reserved and must be programmed to 0000.
[3:0] ESSSA	**Event System Source Subaddress for Channel Source Selection** 000=Channel 0. 001=Channel 1. 010=Channel 2. 011=Channel 3. 100=Channel 4. 101=Channel 5. 110=Channel 6. 111=Channel 7.

17.9.2. Event System Source Subdata Register

The Event System Source Subdata Register (ESSSD), shown in Table 154, sets Channel Source Subregister (ESCHxSRC) operation. The value in the ESSSA Register determines which ESCHxSRC subregister is accessed.

Table 154. Event System Source Subdata Register (ESSSD)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	Reserved	ESSSD						
Reset	0	0	0	0	0	0	0	0
R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F99h							

Bit	Description
[7]	**Reserved** This bit is reserved and must be programmed to 0.
[6:0] ESSSD	**Event System Source Subregister Data**

17.9.3. Event System Channel 0–7 Source Subregisters

The Event System Channel 0–7 Source (ESCHxSRC) subregisters, shown in Table 155, enable the channel and select the source that drives the channel.

Table 155. Event System Channel 0–7 Source Subregisters (ESCHxSRC)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	Reserved	CHSRCSEL						
Reset	0	0	0	0	0	0	0	0
R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	If 00h–07h in Event System Source Subaddress Register, accessible through the Event System Source Subdata Register							

Bit	**Description**
[7]	**Reserved** This bit is reserved and must be programmed to 0.
[6:0] CHSRCSEL	**Event System Channel Source Selection** 00h: Channel Disabled (Low). 01h: Software assertion (High). The channel remains asserted (High) until a different source is selected. 02h–0Fh: Reserved. 10h: Timer 0 out. 11h: Timer 0 out. 12h–13h: Reserved. 14h: Timer 1 out. 15h: Timer 1 out. 16h–17h: Reserved. 18h: Timer 2 out. 19h: Timer 2 out. 1Ah: Timer A out. 1Bh: Reserved. 1Ch: Multi-Channel Timer Output A. 1Dh: Multi-Channel Timer Output B. 1Eh: Multi-Channel Timer Output C. 1Fh: Multi-Channel Timer Output D. 20h–3Fh: Reserved. 40h: Comparator 0 out. 41h: Comparator 1 out. 42h: Comparator 0/1 window detection (C01). 43h–4Fh: Reserved. 50h: Port A0 pin. 51h: Port A1 pin. 52h: Port A2 pin.

Bit	Description (Continued)
[6:0] CHSRCSEL (cont'd.)	53h: Port A3 pin. 54h: Port A4 pin. 55h: Port A5 pin. 56h: Port A6 pin. 57h: Port A7 pin. 58h–63h: Reserved. 64h: Port C4 pin. 65h: Port C5 pin. 66h: Port C6 pin. 67h: Port C7 pin. 68h–72h: Reserved. 73h: Port E0 pin. 74h: Port E1 pin. 75h: Port E2 pin. 76h: Port E3 pin. 77h–7Fh: Reserved.

17.9.4. Event System Destination Subaddress Register

The Event System Destination Subaddress Register (ESDSA), shown in Table 156, selects the Event System Destination 0–3F Channel Subregister (ESDSTxCH) that is accessible through the Event System Destination Subdata Register (ESDSD).

Table 156. Event System Destination Subaddress Register (ESDSA)

Bit	7	6	5	4	3	2	1	0
Field	Reserved		ESDSA					
Reset	0	0	0	0	0	0	0	0
R/W	R	R	R/W	R/W	R/W	R/W	R/W	R/W
Address	F9Ah							

Bit	Description
[7:6]	**Reserved** This bit is reserved and must be programmed to 0.
[5:0] ESDSA	**Event System Destination Subaddress for Destination Selection** 00h–03h: Reserved. 04h: ADC convert input. 05h–0Fh: Reserved. 10h: Timer 0 input 0. 11h: Timer 0 input 1. 12h–13h: Reserved.

Bit	Description (Continued)
[5:0] ESDSA (cont'd.)	14h: Timer 1 input 0. 15h: Timer 1 input 1. 16h–17h: Reserved. 18h: Timer 2 input 0. 19h: Timer 2 input 1. 1Ah: Timer A input. 1Bh: Reserved. 1Ch: Multi-Channel Timer Input A. 1Dh: Multi-Channel Timer Input B. 1Eh: Multi-Channel Timer Input C. 1Fh: Multi-Channel Timer Input D. 20h: Multi-Channel Timer Input. 21h–2Fh: Reserved. 30h: ESOUT0. Event System GPIO Out 0. 31h: ESOUT1. Event System GPIO Out 1. 32h: ESOUT2. Event System GPIO Out 2. 33h: ESOUT3. Event System GPIO Out 3. 34h–3Fh: Reserved.

17.9.5. Event System Destination Subdata Register

The Event System Destination Subdata Register (ESDSD), shown in Table 157, sets the Event System Destination 0–3F Channel Subregister (ESDSTxCH) operation. The value in the ESDSA Register determines which ESDSTxCH Subregister is accessed.

Table 157. Event System Destination Subdata Register (ESDSD)

Bit	7	6	5	4	3	2	1	0
Field	Reserved				ESDSD			
Reset	0	0	0	0	0	0	0	0
R/W	R	R	R	R	R/W	R/W	R/W	R/W
Address	F9Bh							

Bit	Description
[7:4]	**Reserved** This bit is reserved and must be programmed to 0.
[3:0] ESDSD	**Event System Destination Subregister Data**

17.9.6. Event System Destination 0–3F Channel Subregisters

The Event System Destination 0–3F Channel (ESDSTxCH) subregisters, shown in Table 158, determine whether each destination is connected to the Event System and select the Event System channel to connect to the destination.

Table 158. Event System Destination 0–3F Channel Subregisters (ESDSTxCH)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	Reserved				DSTCON	DSTCHSEL		
Reset	0	0	0	0	0	0	0	0
R/W	R	R	R	R	R/W	R/W	R/W	R/W
Address	If 00h -3Fh in Event System Destination Subaddress Register, accessible through the Event System Destination Subdata Register							

Bit	Description
[7:4]	**Reserved** This bit is reserved and must be programmed to 0.
[3] DSTCON	**Event System Destination Connection** 0: The selected Event System channel is not connected to the addressed destination. The destination is connected to logic 0. 1: The selected Event System channel is connected to the addressed destination.
[2:0] DSTCHSEL	**Event System Destination Channel Selection** 000: Channel 0 is connected to the destination if DSTCON=1. 001: Channel 1 is connected to the destination if DSTCON=1. 010: Channel 2 is connected to the destination if DSTCON=1. 011: Channel 3 is connected to the destination if DSTCON=1. 100: Channel 4 is connected to the destination if DSTCON=1. 101: Channel 5 is connected to the destination if DSTCON=1. 110: Channel 6 is connected to the destination if DSTCON=1. 111: Channel 7 is connected to the destination if DSTCON=1.

Chapter 18. Reference System

The F3224 Series devices feature a reference system that provides reference levels for the comparators, ADC and VBIAS output.

Features of the Reference System are as follows:

- Reference Generator:
  - Fixed reference voltages including: the bandgap voltage, $AV_{DD}/2$, 0.50V, 0.65V, 0.75V, 1.0V, 1.25V, 1.35V, and 1.5V, that are available to certain internal functions
  - VBIAS with four selectable levels (2.5V, 2.0V, 1.5V, 1.25V) that is available as an internal positive voltage reference for the ADC and as a GPIO alternate function to provide a low-power external reference
- Two programmable references provide 32 taps (steps), with the highest tap selectable as either VBIAS or AV_{DD}

18.1. Architecture

Figure 51 shows a simplified block diagram of the Reference System, which includes the Reference Generator and two programmable references.

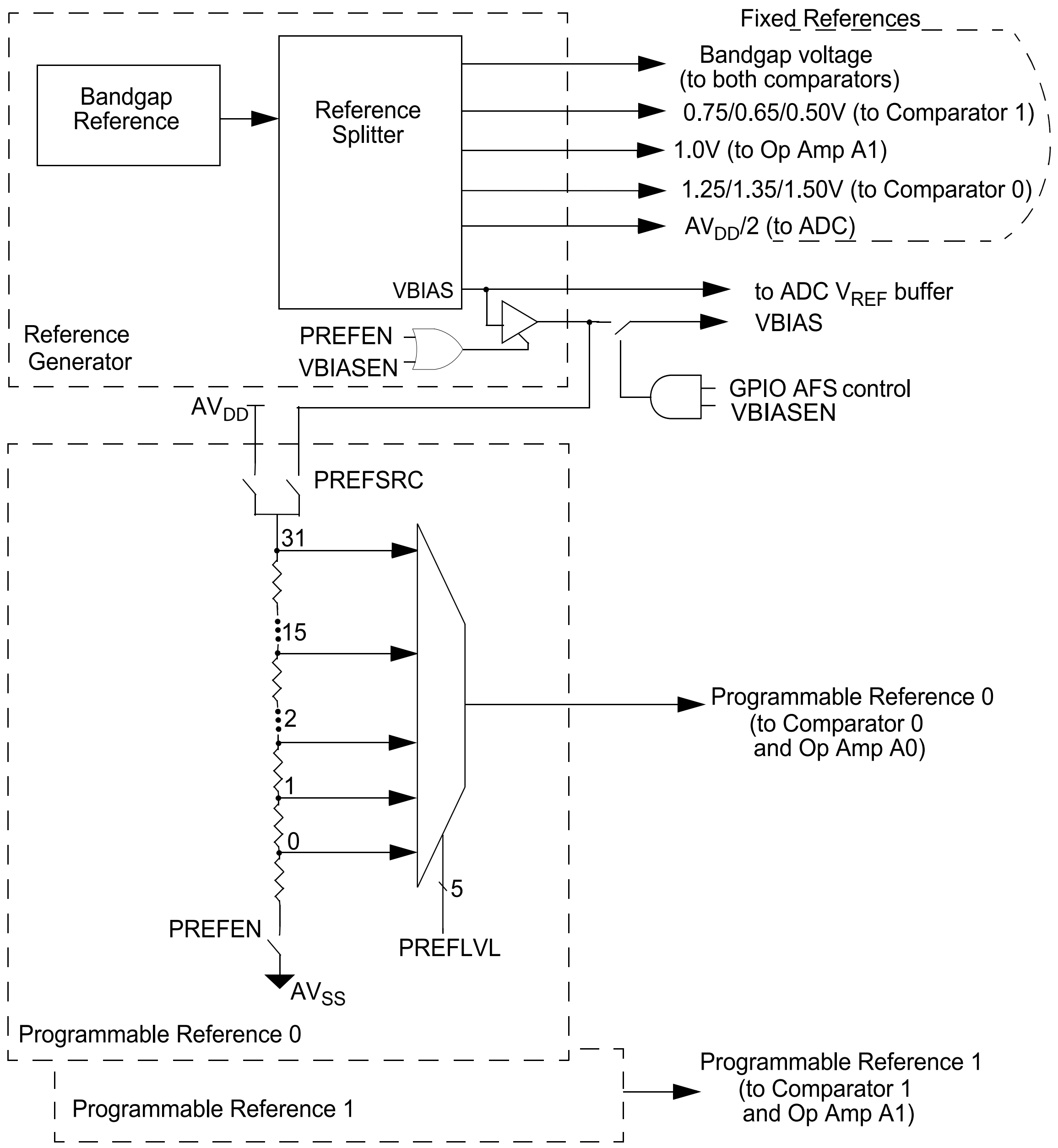


Figure 51. Reference System Block Diagram

18.2. Reference System Operation

The Reference System provides predetermined fixed voltage levels and two programmable references that provide user-selectable voltage levels. Features of the Reference System include:

- Reference Generator:
  - Fixed reference voltages including: the bandgap voltage, $AV_{DD}/2$, one of three low fixed reference levels (0.50V, 0.65V, 0.75V), 1.0V, and one of three high fixed reference levels (1.25V, 1.35V, 1.50V), that are available to certain internal functions
  - VBIAS with four selectable levels (2.5V, 2.0V, 1.5V, 1.25V) that is available as an internal positive voltage reference for the ADC and as a GPIO alternate function to provide a low-power external reference
- Two programmable references provide 32 taps (steps), with the highest tap selectable as either VBIAS or AV_{DD}

The Reference Generator behavior during Active Mode and Halt Mode is as follows:

- Fixed reference voltages are always available
- VBIAS is available if any of the following events are true:
  - VBIASEN is set in the FREFCTL Register
  - A programmable reference is enabled
  - An internal reference voltage is selected for the ADC

To output VBIAS on the VBIAS pin, select the corresponding GPIO alternate function and set the VBIASEN bit in the FREFCTL Register.

The Reference Generator behavior during Stop Mode is as follows:

- Fixed reference voltages are available if FRECOV=1
- VBIAS is available if FRECOV=1 and any of the following events are true:
  - VBIASEN is set in the FREFCTL Register
  - A programmable reference is enabled by setting PREFEN in the PREFxCTL Register
  - An internal reference voltage is selected for the ADC.

To output VBIAS on the VBIAS pin, select the corresponding GPIO alternate function and set the VBIASEN bit in the FREFCTL Register.

When enabled, the Reference Generator provides low current consumption. In addition, any particular fixed reference is automatically enabled upon selection in the function that uses the fixed reference. The low and high fixed reference levels are selected by FREFLVL in the FREFCTL Register.

Each internal programmable reference features an independent enable, PREFEN, that eliminates current consumption if a particular programmable reference is not required. Each programmable reference can drive its assigned comparator and/or its assigned op amp. Programmable Reference 0 can be selected as an input to Comparator 0 and/or Op Amp A0. Programmable Reference 1 can be selected as an input to Comparator 1 and/or Op Amp A1.

Each programmable reference provides 32 levels, selectable with PREFLVL. The uppermost level can be selected as VBIAS from the Reference Generator (PREFSRC=0) or selected as A_{VDD} (PREFSRC=1). The level of the VBIAS is determined by VBIASLVL in the FREFCTL Register. VBIAS also serves as a voltage reference for the ADC and VBIAS pin.

18.3. Reference System Register Definitions

This section defines the features of the following Reference System registers:

Fixed Reference Control Register at address `F7Dh`

Programmable Reference 0 Control Register at address `F7Eh`

Programmable Reference 1 Control Register at address `F7Fh`

18.3.1. Fixed Reference Control Register

The Fixed Reference Control Register, shown in Table 159, provides fixed reference low/high VBIAS control.

Table 159. Fixed Reference Control Register (FREFCTL)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	Reserved			FREFLVL		VBIASEN	VBIASLVL	
Reset	0	0	0	0	0	0	0	0
R/W	R	R	R	R/W	R/W	R/W	R/W	R/W
Address	F7Dh							

Bit	**Description**
[7:5]	**Reserved** These bits are reserved and must be programmed to 000.
[4:3]	**Fixed Reference Low/High Level Select** 00: 0.75V/1.25 V for fixed reference low/high. 01: 0.65V/1.35 V for fixed reference low/high. 10: 0.50V/1.50 V for fixed reference low/high. 11: Reserved.
[2] VBIASEN	**VBIAS Enable** VBIAS is automatically enabled in Active and Halt modes whenever VBIASEN is set, an internal programmable reference is enabled or the buffered VBIAS is selected as the positive reference voltage for the ADC. To make VBIASEN available to the VBIAS pin as a GPIO alternate function, VBIASEN must be set. 0: VBIAS disabled. 1: VBIAS enabled.
[1:0] VBIASLVL	**VBIAS Level Select** 00: 1.25V. 01: 1.5 V. 10: 2.0 V. 11: 2.5 V.

18.3.2. Programmable Reference 0 Control Register

The Programmable Reference 0 Control Register is shown in Table 160. It provides control for Programmable Reference 0.

Table 160. Programmable Reference 0 Control (PREF0CTL)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	Reserved	PREFEN	PREFSRC	PREFLVL				
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F7Eh							

Bit	**Description**
[7]	**Reserved** This bit is reserved and must be programmed to 0.
[6] PREFEN	**Programmable Reference Enable** 0: Programmable Reference disabled. 1: Programmable Reference enabled as defined by PREFSRC and PREFLVL.
[5] PREFSRC	**Programmable Reference Source Selection** 0: VBIAS is the highest tap of the Programmable Reference. 1: AV_{DD} is the highest tap of the Programmable Reference.
[4:0] PREFLVL	**Programmable Reference Level Selection** 00000 to 11111: Programmable reference level=(PREFSRC selection) * (PREFLVL + 1) ÷ 32.

18.3.3. Programmable Reference 1 Control Register

The Programmable Reference 1 Control Register is shown in Table 161. It provides control for Programmable Reference 1.

Table 161. Programmable Reference 1 Control (PREF1CTL)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	Reserved	PREFEN	PREFSRC	PREFLVL				
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F7Fh							

Bit	Description
[7]	**Reserved** This bit is reserved and must be programmed to 0.
[6] PREFEN	**Programmable Reference Enable** 0: Programmable reference disabled. 1: Programmable reference enabled as defined by PREFSRC and PREFLVL.
[5] PREFSRC	**Programmable Reference Source Selection** 0: VBIAS is the highest tap of the Programmable Reference. 1: AV_{DD} is the highest tap of the Programmable Reference.
[4:0] PREFLVL	**Programmable Reference Level Selection** 00000 to 11111:Programmable reference level=(PREFSRC selection) * (PREFLVL + 1) ÷ 32.

Chapter 19. Analog-to-Digital Converter

The F3224 Series MCUs include a seventeen-channel Successive Approximation Register Analog-to-Digital Converter (SAR ADC). This ADC converts an analog input signal to a 12-bit or 14-bit binary number, and includes the following additional features:

- 12-bit or 14-bit resolution
- Fifteen analog input sources multiplexed with general-purpose I/O ports
- Four internal analog input sources including: Op Amp A0 output, Op Amp A1 output, bandgap, and $A_{VDD}/2$ fixed reference
- Two input modes: single-ended mode and differential mode
- Conversion initiated by software or Event System input
- Channel scanning function
- Optional conversion averaging of 2, 4, 8, 16 samples
- Continuous conversion function that can be used with or without channel sequencing
- Fast conversion time, as low as 3 µs
- Programmable timing controls including ADC clock prescaler
- Window check function
- Interrupt on conversion complete or outside window
- Internal voltage reference selections of A_{VDD} or buffered VBIAS from the Reference System (2.5V, 2.0V, 1.5V, 1.25V)
- Buffered VBIAS internal reference voltage can be driven externally on $V_{REF}+$
- Ability to utilize external reference voltage
- In-situ calibration for all operating modes
- Auto-disable

19.1. Architecture

The ADC can be operated with either single-ended inputs or differential inputs. The architecture, shown in Figure 52, consists of input multiplexers, sample-and-hold, an internal voltage reference buffer, and a 12-bit SAR ADC. The ADC digitizes the signal on a selected channel and stores the digitized data in the ADC data registers. In environments with high electrical noise, an external RC filter must be added at the input pins to reduce high-frequency noise.

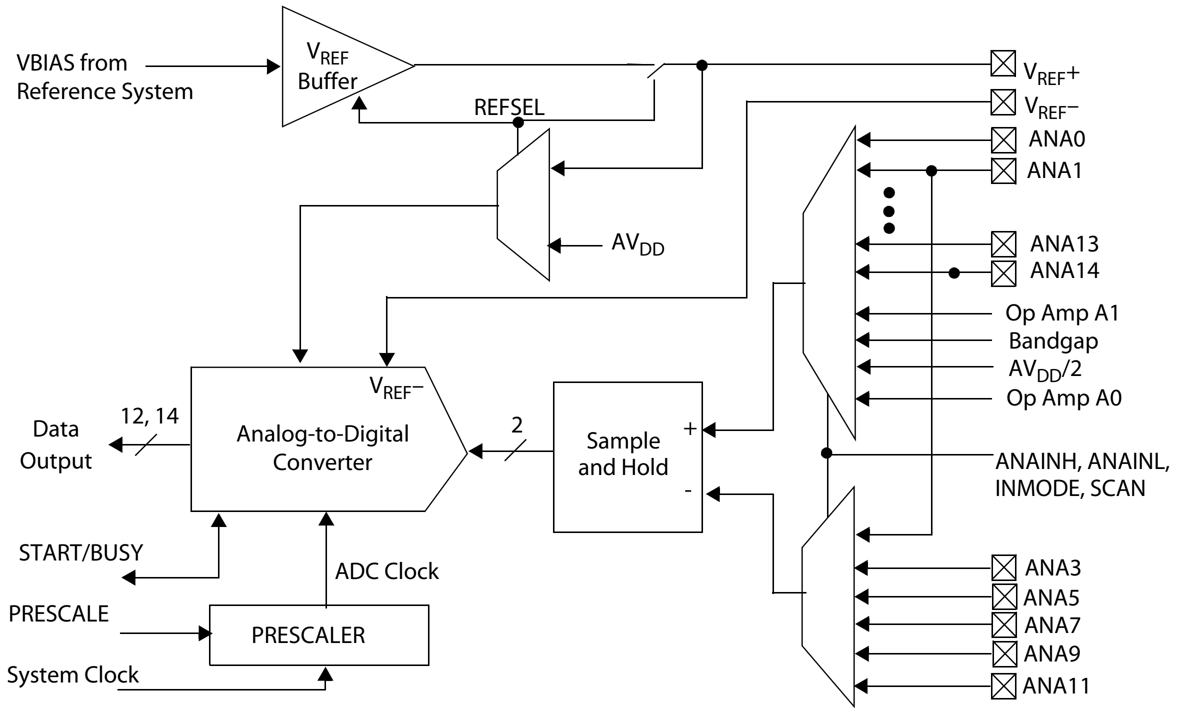


Figure 52. Analog-to-Digital Converter Block Diagram

19.2. Operation

The ADC converts the analog input to a digital representation. The ADC has selectable input modes, resolution, data format, conversion options, window detection, and voltage reference options.

Assuming zero gain and offset errors, any voltage outside the ADC input limits of V_{REF-} and V_{REF+} returns the minimum ADC output code or the maximum ADC output code, respectively.

19.2.1. Input Modes

Two input modes are available:

- Single-ended input
- Differential inputs

Single-ended input mode is selected by configuring INMODE=0. In this mode, one of 19 positive inputs can be selected using both ANAINH and ANAINL, and are referenced to V_{REF-}. In this mode, V_{REF+} can range up to A_{VDD}.

Differential input mode is selected by configuring INMODE=1. In this mode, one of 6 positive input pairs can be selected using ANAINL, and the inputs are treated as differential, in that the positive input can be higher or lower than the negative input. In this mode, V_{REF}+ can range up to A_{VDD}.

19.2.2. ADC Data Format

The ADC supports two data formats, unsigned and signed, selected by DFORMAT in the ADC Control 1 Register. When using signed data format, negative values are sign extended. Figures 53 through 54 show the relationship between data formats at 12-bit resolution, and ADC output data for the selectable input modes. The equation for calculating the ADC output data value is a function of input mode, resolution, and data format. The following equations can be used to calculate an ADC output data value for common combinations of input mode, resolution, and data format.

Single-ended input mode (INMODE=0), unsigned (DFORMAT=0):

$$\text{ADC Output} = \text{FSR} \times ((\text{ANAx} - V_{REF-}) \div (V_{REF+} - V_{REF-}))$$

In the equation above, FSR (full-scale range) is 4095 for 12-bit conversions, and 16383 for 14-bit conversions.

Differential input mode (INMODE=1), signed (DFORMAT=1):

$$\text{ADC Output} = \text{FSR} \times ((\text{ANAx} - \text{ANAx+1}) \div (V_{REF+} - V_{REF-}))$$

In the equation above, 12-bit conversion FSR (full scale range) is –2048 for negative inputs and +2047 for positive inputs; 14-bit conversion FSR is –8192 for negative inputs and +8191 for positive inputs.

Data can be left-justified or right-justified as defined by the JUSTIFY bit in the ADC Control 1 Register. These justifications apply to the data registers and window registers. Conversion resolution can be configured to be 12-bit or 14-bit, as defined by the RESOLUT bit in the ADC Control 1 Register. Note that bit 0 of the ACDCTL1 Register must be set for proper ADC operation.

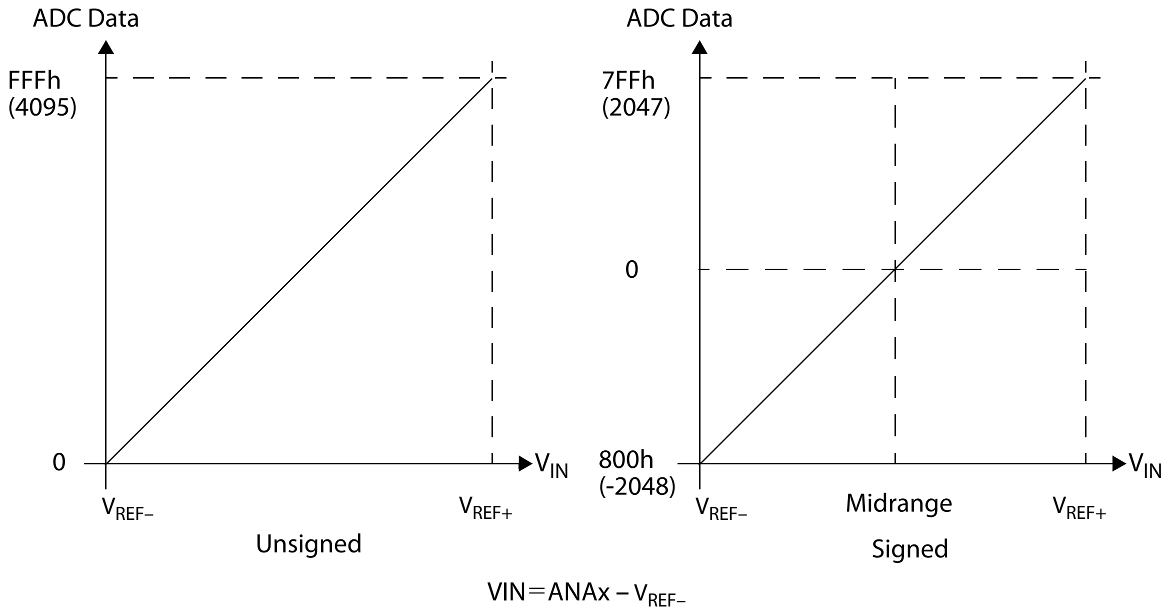


Figure 53. ADC Data (12-bit) vs. Input Voltage for Single-Ended Input Mode

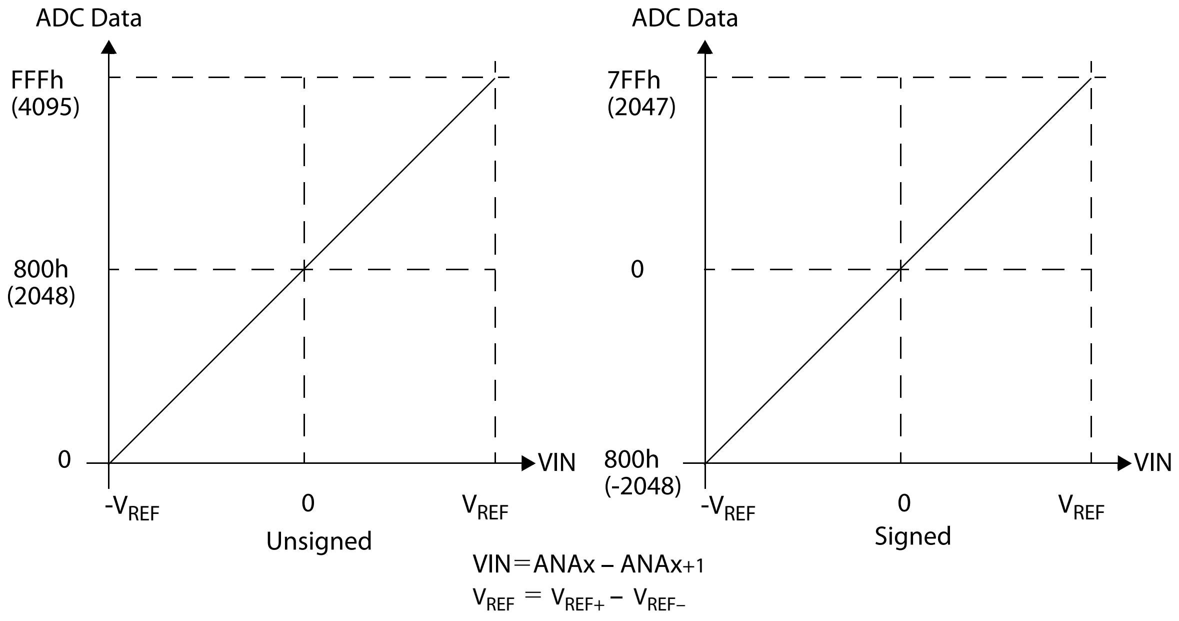


Figure 54. ADC Data (12-bit) vs. Input Voltage for Differential Input Mode

19.2.3. Conversion Options

Five ADC conversion options are available, and are configured in the ADC Control 0 Register. These conversion options are independent from each other and can be selected in any combination. Furthermore, these conversion options can be enabled for any input mode. Each of these five options is described in the following subsections.

19.2.3.1. Single-Shot or Continuous Conversion

The ADC can be configured for single-shot or continuous conversion. When the CONTCONV bit is cleared, starting the ADC will produce a single result. When CONTCONV is set, starting the ADC will produce a continuous stream of results until CONTCONV is cleared. An interrupt can be generated for each conversion result.

19.2.3.2. Channel Scanning

The ADC can be configured to automatically scan multiple channels. If SCAN is cleared, channel scanning is not performed, and only the input (or input pair) defined in ANAINL is sampled for conversion. Note that while SCAN=1, a read of ANAINH/ANAINL will return the channel associated with the data in ADCD_H/ADCD_L.

If the SCAN bit is set, channel scanning is enabled, and the configuration of ANAINL and ANAINH determines which channels are scanned. If the bit corresponding to a particular channel is set, the channel will be included in the scan. Scanning commences with the LSB of ANAINL and completes with the MSB of ANAINH. ADC conversions are performed only on the channels selected in ANAINL and ANAINH. Channels that are not selected are skipped.

The ADC configuration is identical for each channel scanned as defined in the ADC control registers. Timing parameters, and ST, should be configured for the requirements of the worst-case channel. If single-shot conversion is selected (CONTCONV=0), the ADC performs the scan sequence once. If continuous conversion is enabled, the ADC repeats the scan sequence in a continuous fashion.

An interrupt can be generated for each channel conversion result.

19.2.3.3. Conversion Averaging

The ADC is capable of processing data from multiple individual conversions to form an averaged result. When averaging is enabled by setting the AVE bit, AVESAMP determines whether 2, 4, 8, or 16 samples are averaged to produce a result. An interrupt will be generated only when a final sample is obtained and processed into the average. If channel scanning is enabled, the averaged result is obtained sequentially for each channel being scanned.

19.2.3.4. Resolution

When the RESOLUT bit is cleared, 12-bit conversions are performed. For applications that require even higher resolution, 14-bit resolution conversions are performed when

RESOLUT is set. 14-bit conversions involve somewhat longer timing than 12-bit conversions as described in the Conversion Timing section on page 313.

19.2.4. Starting and Stopping Conversions

ADC activity is initiated by writing the START bits in the ADC Control 0 Register to perform a conversion (START=01), offset calibration (START=10), or gain calibration (START=11). When SCAN=1, starting a conversion by writing START=01 initiates channel scanning. Additionally, the Event System can trigger a new conversion and cause START to be set to 01. When a calibration is initiated, SCAN and CONTCONV are ignored.

The ADC can be configured to be constantly awake (ADC=10 in the PWRCTL1 Register) or to awaken upon a trigger event (ADC=0x in the PWRCTL1 Register). For the latter, when ADC activity is triggered, the ADC will wake up for the duration of the ADC wake-up period, T_{WAKE_ADC}, prior to performing the conversion (see the Electrical Characteristics chapter on page 400). When the conversion is completed, the ADC's auto-disable feature will automatically disable the ADC when no further conversions are scheduled. If performing multiple sequential conversions due to averaging, scanning or continuous conversion, the wake-up time is incurred only prior to the first conversion.

When a conversion or calibration completes, START is cleared to 00 automatically by hardware. To avoid disrupting a conversion already in progress, START can be read to indicate ADC operation status (busy or available). If the result from a previous ADC operation is not read before the result from a subsequent ADC operation is complete, the previous result is overwritten.

A conversion or calibration can be aborted by clearing the START bits to 00. When a conversion or calibration is in progress, the value of the START bits cannot be changed to another value other than 00. Upon entering Stop Mode, any ADC conversion or calibration that is in progress is aborted.

It is recommended to change the value of the SCAN and CONTCONV bits only while the ADC is inactive (START = 00).

19.2.5. Voltage References

The ADC positive voltage reference is selected with REFSEL. The ADC negative reference should always be configured as V_{REF-} using the GPIO Alternate Function Selection described in the General-Purpose Input/Output chapter on page 46. ADC positive voltage reference selection options are:

- A_{VDD} (REFSEL=00)
- External voltage reference on V_{REF}+ (REFSEL=01)
- Internal voltage reference buffer connected to V_{REF}+ which buffers VBIAS from the Reference System (REFSEL=11)

VBIAS in the Reference System offers four possible level settings that are selected with VBIASLVL in the FREFCTL Register, namely: 1.25V, 1.5V, 2.0V, and 2.5V. Care should be exercised to ensure that A_{VDD} is always at least 0.5V greater than the selected VBIAS level. Wake-up of the internal voltage reference buffer can be performed automatically or manually.

Automatic Wake-Up. If the ADC bits cleared to 00 in the PWRCTL1 Register, then when ADC activity is triggered, the ADC internal voltage reference buffer will wake up for the duration of the ADC wake-up period, T_{WAKE_ADC}, prior to the ADC conversion being performed (see the Electrical Characteristics chapter on page 400). When the conversion is completed, the ADC auto-disable feature will automatically disable the internal voltage reference buffer when no further conversions are scheduled. If performing multiple sequential conversions due to averaging, scanning or continuous conversion, the wake-up time is incurred only prior to the first conversion.

Manual Wake-Up. When the ADC = 01 or 10 in the PWRCTL1 Register, the internal voltage reference buffer is continuously enabled if it is selected as the ADC positive voltage reference. This is typically used when the ADC internal voltage reference buffer is connected to the V_{REF}+ pin (REFSEL=11 in the ADCCTL2 Register).

When using the internal voltage reference buffer connected to V_{REF}+ (REFSEL=11), an external bypass capacitor is required, as defined in the Electrical Characteristics chapter on page 400.

19.2.6. ADC Timing

System Clock can be prescaled to form the ADC clock with the divisor defined by PRESCALE. ADC timing is a function of resolution, as described in the following sections. When the ADC exits the idle state to perform a conversion, a wake-up time, T_{WAKE_ADC}, is incurred, as defined in the Electrical Characteristics chapter on page 400.

19.2.6.1. Conversion Timing

Each ADC conversion consists of 2 phases:

1. Input sampling time, as defined by ST, is a function of source impedance and the desired accuracy, as discussed later in this section. The minimum input sampling period is 200ns.
2. Sample conversion time is 13 ADC clock cycles for 12-bit resolution (RESOLUT=0) and 15 ADC clock cycles for 14-bit resolution (RESOLUT=1). The maximum ADC clock frequency is shown in Table 239 on page 409. When performing 14-bit conversions, it is recommended to average at least two samples.

Figure 55 shows the timing of an ADC conversion.

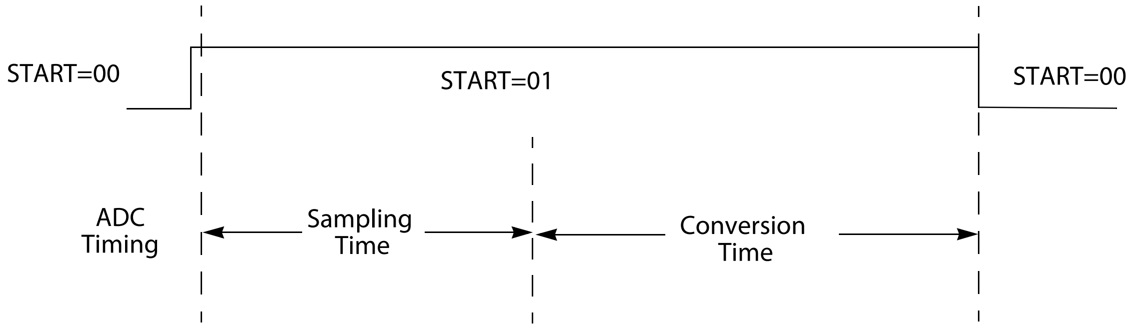


Figure 55. ADC Conversion Timing Diagram

19.2.6.2. ADC Wake-up and Sampling

As the ADC is designed for low-power applications, the ADC core is powered down when no further conversions are scheduled and ADC is cleared to 00 in the PWRCTL1 Register. When starting a new conversion while the ADC is idle, the ADC wake-up delay is automatically inserted prior to sampling. Furthermore, if the internal voltage reference buffer is selected with auto-enable, the internal voltage reference buffer wake-up delay is automatically inserted prior to sampling. The ADC wake-up time can be avoided by setting ADC=10 in the PWRCTL1 Register to continuously enable the ADC.

The sample period is a function of source impedance and the desired accuracy. While sampling is occurring, the ADC input impedance changes from high impedance to a series 2KΩ (max) resistance and a shunt 19pF (max) between the signal source and the ADC input, as shown in Figure 56. Sufficient sampling time should be allotted to charge the capacitance to the desired accuracy. The following equation describes the minimum sampling time required to charge the capacitor to 1/2 LSB for resolution, n:

$$ST > (Rs + Ri) \times \ln(2^{n+1}) \times Ci + STmin$$

In this equation, ST is the sampling time, Rs is the source resistance, Ri is the ADC series input resistance, Ci is the ADC shunt input capacitance, n is the resolution and STmin is the minimum sampling time specified.

For example, to achieve 12-bit resolution with INMODE=0, Rs=10KΩ, and maximum internal input resistance and capacitance, consider the following equation:

$$ST > (10K\Omega + 2K\Omega) \times \ln(2^{12+1}) \times 19pF + 200ns$$

In this equation, ST > 2.25 µs.

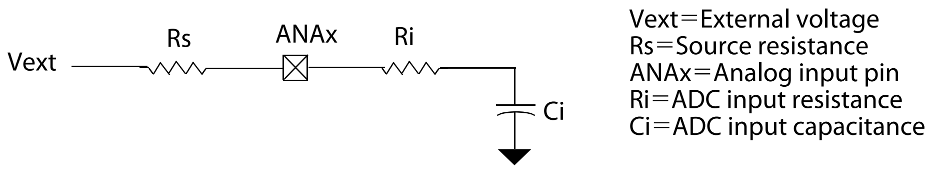


Figure 56. ADC Input Equivalent Circuit

19.2.7. Window Detection

Window detection is available using the window upper threshold registers, ADCUWINH and ADCUWINL, and the window lower threshold registers, ADCLWINH and ADCLWINL. An interrupt is generated if the ADC result is outside the window, meaning that the result is either greater than the value in the window upper threshold registers, or lower than the value in the window lower threshold registers. If only below-threshold detection is desired, configure the upper threshold registers to be the maximum conversion value (all 1s). If only above-upper-threshold detection is desired, configure the lower threshold registers to be the minimum conversion value (all 0s).

Data loaded into the window threshold registers must be consistent with data loaded into the ADC data registers (ADCD_H, ADCD_L) in terms of DFORMAT, RESOLUT, and JUSTIFY. Reserved data bit locations are ignored during window detection. Window comparison occurs only upon a new ADC result; changes to a window threshold register's contents will not result in a comparison against residual ADC output data.

19.2.8. ADC Interrupts

The ADC can generate an interrupt request upon each new ADC result for any completed conversion or calibration (START= 01, 10, 11). The ADC can also generate an interrupt if the conversion result is outside the range defined by the window threshold registers (ADCUWINH, ADCUWINL, ADCLWINH, ADCLWINL). Use the IRQ bit in the ADC Control 0 Register to select whether interrupts are generated due to only exceeding the window thresholds or due to both exceeding the window thresholds and end of convert. An interrupt request that is pending when the ADC is disabled or idle (ready) is not automatically cleared.

19.2.9. Calibration and Compensation

Both gain and offset calibration can be performed in situ to achieve even higher accuracy than specified in the Electrical Characteristics chapter on page 400. These calibration operations are performed using the current ADC configuration, as defined by the INMODE, PRESCALE, and ST bits. After these parameters are reconfigured, initiating calibration prior to performing conversions can optimize results. Only initiate calibration when continuous conversion is not selected (i.e., CONTCONV = 0).

Offset calibration is performed when the START bits are written to 10. The offset calibration result is stored in the OFFSET field of the ADCOFF Register and is a 14-bit two's complement value (RESOLUT is ignored during offset calibration). The calibration is complete when START is cleared to 00.

The value in the OFFSET field is automatically applied to compensate subsequent conversions by hardware. Offset correction by hardware is effective for signed mode (DFORMAT = 1) only. For unsigned mode (DFORMAT = 0), software should store the value of OFFSET, clear OFFSET to `00h`, and perform any desired offset compensation.

As each ADC configuration can exhibit a unique offset calibration, the offset calibration value for each ADC configuration of interest can be stored. Stored offset values can be used to compensate both 12-bit and 14-bit resolution conversions as shown in Table 162.

Table 162. Compensating ADC results for ADC offset

ADC offset calibration RESOLUT value	ADC conversion RESOLUT value	ADC offset compensated result formula using stored OFFSET value
RESOLUT=x	RESOLUT=0 (12-bit)	{ADCD_H, ADCD_L} - stored_ADCOFF/4
RESOLUT=x	RESOLUT=1 (14-bit)	{ADCD_H, ADCD_L} - stored_ADCOFF

Gain calibration is performed when the START bits are written to 11. The calibration is complete when START is cleared to 00. To utilize the gain factor, first read it from the ADCD_H and ADCD_L registers following a gain calibration and store it. The gain factor can then be applied to a raw ADC or an offset compensated ADC result by software to produce a gain-compensated result, as follows:

ADC gain-compensated result = ADC result x ((full scale range x 0.75) ÷ (stored_gain_factor))

19.3. ADC Control Register Definitions

The registers that control analog-to-digital conversion functions are defined in this section.

19.3.1. ADC Control 0 Register

The ADC Control 0 Register, shown in Table 163, initiates the A/D conversion, provides ADC status information and contains conversion control options.

Table 163. ADC Control 0 Register (ADCCTL0)

Bits	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	START		Reserved	IRQ	CONTCONV	AVE	AVESAMP	
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W
Address	F70h							

Bit	**Description**
[7:6] START	**ADC Start/Busy** 00: Writing 00 aborts the conversion or calibration that is in progress. Reading 00 indicates the ADC is available to begin a conversion or calibration. 01: Writing 01 starts a conversion. Reading 01 indicates that a conversion is currently in progress. When START=01, START can be written only with 00, which aborts the conversion. 10: Writing 10 starts an offset calibration. Reading 10 indicates that an offset calibration is currently in progress. When START=10, START can be written only with 00, which aborts the calibration. Set RESOLUT = 1 prior to initiating an offset calibration. 11: Writing 11 starts a gain calibration. Reading 11 indicates that a gain calibration is currently in progress. When START=11, START can be written only with 00, which aborts the calibration.
[5]	**Reserved** This bit is reserved and must be programmed to 0.
[4] IRQ	**Interrupt Control** 0: Outside window. 1: Both end of convert and outside window.
[3] CONTCONV	**Continuous Conversion Enable** 0: Single-shot conversion. 1: Continuous conversion.
[2] AVE	**Averaging Enable** 0: Averaging of ADC samples is disabled. 1: Averaging of ADC samples is enabled. The number of samples to convert to form each ADC result are determined by AVESAMP.
[1:0] AVESAMP	**Averaging Samples** If AVE = 1, ADC samples are averaged to form an ADC result. 00: 2 samples are converted to form each ADC result. 01: 4 samples are converted to form each ADC result. 10: 8 samples are converted to form each ADC result. 11: 16 samples are converted to form each ADC result.

19.3.2. ADC Control 1 Register

The ADC Control 1 Register, shown in Table 164, contains control for the ADC input mode and other ADC features. Note that bit 0 of this register must be set to 1 for proper ADC operation.

Table 164. ADC Control 1 Register (ADCCTL1)

Bits	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	Reserved		SCAN	Reserved	INMODE	DFORMAT	RESOLUT	JUSTIFY
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F71h							

Bit	Description
[7:6]	**Reserved** These bits are reserved and must be programmed to 00.
[5] SCAN	**Channel Scanning Enable** The channels to be scanned are determined by ANAINH and ANAINL. 0: Channel scanning is disabled. 1: Channel scanning is enabled.
[4]	**Reserved** This bit is reserved and must be programmed to 0.
[3] INMODE	**Input Mode** 0: Single-ended. 1: Differential.
[2] DFORMAT	**Data Format** 0: Data is unsigned (binary). 1: Data is signed (two's complement). Negative values are sign-extended.
[1] RESOLUT	**ADC Conversion Resolution** 0: 12-bit resolution. 1: 14-bit resolution.
[0] JUSTIFY	**Data Register Justification** 0: Left-justified. 1: Right-justified.

19.3.3. ADC Control 2 Register

The ADC Control 2 Register, shown in Table 165, contains control for the ADC prescaler and reference selection.

Table 165. ADC Control 2 Register (ADCCTL2)

Bits	7	6	5	4	3	2	1	0
Field	REFSEL		Reserved		PRESCALE			
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R	R	R/W	R/W	R/W	R/W
Address	F72h							

Bit	Description
[7:6] REFSEL	**ADC Positive Voltage Reference Select** 00: Internal connection to AV_{DD}. 01: V_{REF+} pin driven by an external source. 10: Reserved. 11: Buffered VBIAS from the Reference System drives the V_{REF}+ pin.
[5:4]	**Reserved** These bits are reserved and must be programmed to 00.
[3:0] PRESCALE	**ADC Clock Prescale Divider** 0000: ADC Clock is System Clock divided by 1. 0001: ADC Clock is System Clock divided by 2. 0010: ADC Clock is System Clock divided by 3. 0011: ADC Clock is System Clock divided by 4. 0100: ADC Clock is System Clock divided by 5. 0101: ADC Clock is System Clock divided by 6. 0110: ADC Clock is System Clock divided by 7. 0111: ADC Clock is System Clock divided by 8. 1000: ADC Clock is System Clock divided by 9. 1001: ADC Clock is System Clock divided by 10. 1010: ADC Clock is System Clock divided by 11. 1011: ADC Clock is System Clock divided by 12. 1100: ADC Clock is System Clock divided by 13. 1101: ADC Clock is System Clock divided by 14. 1110: ADC Clock is System Clock divided by 15. 1111: ADC Clock is System Clock divided by 16.

19.3.4. ADC Input Select High Register

The ADC Input Select High Register, shown in Table 166, selects the ADC input(s) for conversion. This register is used only when SCAN is set and the ADC inputs to be scanned are defined both in ADCINSH and ADCINSL.

Table 166. ADC Input Select High Register (ADCINSH)

Bits	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	Reserved	ANAINH						
Reset	0	0	0	0	0	0	0	0
R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F73h							

Bit	**Description**
[7]	**Reserved** This bit is reserved and must be programmed to 0.
[6:0] ANAINH	**Analog Input Selection High** ADC Input Selection is a function of SCAN and INMODE.
	SCAN=0, INMODE = x All bits are reserved.
[6:0] ANAINH (cont'd.)	**SCAN=1, INMODE=0** xxxxxx1: ANA8 input is selected for ADC scanning. Additional inputs may be selected. xxxxx1x: ANA9 input is selected for ADC scanning. Additional inputs may be selected. xxxx1xx: ANA10 input is selected for ADC scanning. Additional inputs may be selected. xxx1xxx: ANA11 input is selected for ADC scanning. Additional inputs may be selected. xx1xxxx: ANA12input is selected for ADC scanning. Additional inputs may be selected. x1xxxxx: Op Amp A0 output is selected for ADC scanning. Additional inputs may be selected. 1xxxxxx: Op Amp A1 output is selected for ADC scanning. Additional inputs may be selected. **Note:** When SCAN=1, a read returns the channel corresponding to ADCD_H/ADCD_L.
	SCAN=1, INMODE=1 All bits are reserved.

19.3.5. ADC Input Select Low Register

The ADC Input Select Low Register, shown in Table 167, selects the ADC input(s) for conversion. If SCAN is set, ADCINSH and ADCINSL are both used to define the ADC inputs to be scanned, otherwise, only ADCINSL is used to select the ADC input(s).

Table 167. ADC Input Select Low Register (ADCINSL)

Bits	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	ANAINL							
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F74h							

Bit	Description
[7:0] ANAINL	**Analog Input Selection Low** ADC Input Selection is a function of SCAN and INMODE.
	SCAN=0, INMODE=0 00000000: ANA0 input is selected for analog-to-digital conversion. 00000001: ANA1 input is selected for analog-to-digital conversion. 00000010: ANA2 input is selected for analog-to-digital conversion. 00000011: ANA3 input is selected for analog-to-digital conversion. 00000100: ANA4 input is selected for analog-to-digital conversion. 00000101: ANA5 input is selected for analog-to-digital conversion. 00000110: ANA6 input is selected for analog-to-digital conversion. 00000111: ANA7 input is selected for analog-to-digital conversion. 00001000: ANA8 input is selected for analog-to-digital conversion. 00001001: ANA9 input is selected for analog-to-digital conversion.

Bit	Description (Continued)
[7:0] ANAINL (cont'd.)	00001010: ANA10 input is selected for analog-to-digital conversion. 00001011: ANA11 input is selected for analog-to-digital conversion. 00001100: ANA12 input is selected for analog-to-digital conversion. 00001101: ANA13 input is selected for analog-to-digital conversion. 00001110: ANA14 input is selected for analog-to-digital conversion. 00001111: Op Amp A0 output is selected for analog-to-digital conversion. 00010000: Op Amp A1 output is selected for analog-to-digital conversion. 00010001: AVDD/2 Fixed Reference is selected for analog-to-digital conversion. 00010010: Bandgap reference is selected for analog-to-digital conversion. All other bits are reserved.
	SCAN=0, INMODE=1 0000000x: ANA0 input is selected as the positive input for analog-to-digital conversion. ANA1 input is selected as the negative input for analog-to-digital conversion. 0000001x: ANA2 input is selected as the positive input for analog-to-digital conversion. ANA3 input is selected as the negative input for analog-to-digital conversion. 0000010x: ANA4 input is selected as the positive input for analog-to-digital conversion. ANA5 input is selected as the negative input for analog-to-digital conversion. 0000011x: ANA6 input is selected as the positive input for analog-to-digital conversion. ANA7 input is selected as the negative input for analog-to-digital conversion. 0000100x: ANA8 input is selected as the positive input for analog-to-digital conversion. ANA9 input is selected as the negative input for analog-to-digital conversion. 0000101x: ANA10 input is selected as the positive input for analog-to-digital conversion. ANA11 input is selected as the negative input for analog-to-digital conversion. All other bits are reserved.
	SCAN=1, INMODE=0 xxxxxxx1: ANA0 input is selected for ADC scanning. Additional inputs may be selected. xxxxxx1x: ANA1 input is selected for ADC scanning. Additional inputs may be selected. xxxxx1xx: ANA2 input is selected for ADC scanning. Additional inputs may be selected. xxxx1xxx: ANA3 input is selected for ADC scanning. Additional inputs may be selected. xxx1xxxx: ANA4 input is selected for ADC scanning. Additional inputs may be selected. xx1xxxxx: ANA5 input is selected for ADC scanning. Additional inputs may be selected. x1xxxxxx: ANA6 input is selected for ADC scanning. Additional inputs may be selected. 1xxxxxxx: ANA7 input is selected for ADC scanning. Additional inputs may be selected. **Note:** When SCAN=1, a read returns the channel corresponding to the data currently in the ADCD_H/ADCD_L registers.
	SCAN=1, INMODE=1 0xxxxxx1: ANA0 and ANA1 are selected as a differential input pair for ADC scanning. ANA0 input is selected as the positive input for analog-to-digital conversion. ANA1 input is selected as the negative input for analog-to-digital conversion

Bit	Description (Continued)
[7:0] ANAINL (cont'd.)	**SCAN=1, INMODE=1 (continued)**
	0xxxxx1x: ANA2 and ANA3 are selected as a differential input pair for ADC scanning. ANA2 input is selected as the positive input for analog-to-digital conversion. ANA3 input is selected as the negative input for analog-to-digital conversion.
	0xxxx1xx: ANA4 and ANA5 are selected as a differential input pair for ADC scanning. ANA4 input is selected as the positive input for analog-to-digital conversion. ANA5 input is selected as the negative input for analog-to-digital conversion.
	0xxx1xxx: ANA6 and ANA7 are selected as a differential input pair for ADC scanning. ANA6 input is selected as the positive input for analog-to-digital conversion. ANA7 input is selected as the negative input for analog-to-digital conversion.
	0xx1xxxx: ANA8 and ANA9 are selected as a differential input pair for ADC scanning. ANA8 input is selected as the positive input for analog-to-digital conversion. ANA9 input is selected as the negative input for analog-to-digital conversion.
	0x1xxxxx: ANA10 and ANA11 are selected as a differential input pair for ADC scanning. ANA10 input is selected as the positive input for analog-to-digital conversion. ANA11 input is selected as the negative input for analog-to-digital conversion.
	Note: When SCAN=1, a read returns the channel corresponding to the data currently in the ADCD_H/ADCD_L registers.

19.3.6. ADC Offset Calibration Register

The ADC Offset Calibration Register, shown in Table 168, contains the ADC offset calibration value.

Table 168. ADC Offset Calibration Register (ADCOFF)

Bits	7	6	5	4	3	2	1	0
Field	OFFSET							
Reset	00h							
R/W	R/W							
Address	F75h							

Bit	Description
OFFSET	**ADC Offset Calibration Value** The ADC Offset Calibration Value is a function of RESOLUT. OFFSET is in two's complement format and is applied by the ADC to compensate conversions (START=01) for offset errors. The ADC places the result from offset calibration (START=10) in OFFSET. The value can be read by software and re-written to OFFSET at a later time, for example, when using multiple input modes, each with a unique offset calibration value. Offset correction by hardware is effective for signed mode (DFORMAT = 1) only. For unsigned mode (DFORMAT = 0), software should store the value of OFFSET, clear OFFSET to 00h, and perform any desired offset compensation.
	RESOLUT = 0 (12-bit)

[7:2]	00–3F:	2's complement offset value.
[1:0]	0–1:	Reserved.
RESOLUT = 1 (14-bit)		
[7:0]	00–FF:	2's complement offset value.

19.3.7. ADC Data High Register

The ADC Data High Register, shown in Table 169, contains the MSBs of the ADC result. Access to the ADC Data High Register is read-only. Reading the ADC Data High Register latches data in the ADC Low Register and, if SCAN=1, corresponding scan channel information in the ADCINSH/ADCINSL registers.

Table 169. ADC Data High Register (ADCD_H)

Bits	7	6	5	4	3	2	1	0
Field	ADCDH							
Reset	00h							
R/W	R							
Address	F76h							

Bit	Description	
ADCDH	**ADC Data High** ADC Data High is a function of RESOLUT and JUSTIFY.	
JUSTIFY = 0 (Left-Justified), RESOLUT = x		
[7:0]	00–FF:	The 8 MSBs of the last conversion result are held in this data register until the next ADC conversion has completed.
JUSTIFY = 1 (Right-Justified), RESOLUT = 0 (12-bit)		
[7:4]	Reserved.	
[3:0]	0–F:	The 4 MSBs of the last conversion result are held in the 4 LSBs of this data register until the next ADC conversion has completed.
JUSTIFY = 1 (Right-Justified), RESOLUT = 1 (14-bit)		
[7:6]	Reserved.	
[5:0]	00–3F:	The 6 MSBs of the last conversion result are held in the 6 LSBs of this data register until the next ADC conversion has completed.

19.3.8. ADC Data Low Register

The ADC Data Low Register, shown in Table 170, contains the LSBs of the ADC result. Access to the ADC Data Low Register is read-only. Reading the ADC Data High Register latches data in the ADC Low Register.

Table 170. ADC Data Low Register (ADCD_L)

Bits	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	ADCDL							
Reset	00h							
R/W	R							
Address	F77h							

Bit	**Description**	
ADCDL	**ADC Data Low** ADC Data Low is a function of RESOLUT and JUSTIFY.	
JUSTIFY = 0 (Left-Justified), RESOLUT = 0 (12-bit)		
[7:4]	0–F:	The 4 LSBs of the last conversion result are latched into the 4 MSBs of this register whenever the ADC Data High Byte register is read.
[3:0]	Reserved.	
JUSTIFY = 0 (Left-Justified), RESOLUT = 1 (14-bit)		
[7:2]	00–3F:	The 6 LSBs of the last conversion result are latched into the 6 MSBs of this register whenever the ADC Data High Byte register is read.
[1:0]	Reserved.	
JUSTIFY = 1 (Right-Justified), RESOLUT = x		
[7:0]	00–FF:	The 8 LSBs of the last conversion result are latched into this data register whenever the ADC Data High Byte register is read.

19.3.9. Sample Time Register

The Sample Time Register, shown in Table 171, is used to program the length of the sampling time once a conversion is initiated by setting the START=01 in the ADC Control 0 Register or is initiated by the Event System. The number of ADC clock cycles required for sample time varies from system to system, depending on the impedance of the external source and the ADC clock period used. The system designer should program this register to contain the number of ADC clocks required to meet accuracy requirements as described in the ADC Timing section on page 313.

When using the internal voltage reference buffer connected to V_{REF}+ (REFSEL=11), an external bypass capacitor is required, as defined in the Electrical Characteristics chapter on page 400.

Table 171. Sample Time (ADCST)

Bits	7	6	5	4	3	2	1	0
Field	ST				Reserved			
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R	R	R	R
Address	F78h							

Bit	Description
[7:4] ST	**Sampling Time** 0000: 2 ADC clocks. 0001: 2 ADC clocks. 0010: 4 ADC clocks. 0011: 8 ADC clocks. 0100: 16 ADC clocks. 0101: 32 ADC clocks. 0110: 64 ADC clocks. 0111: 96 ADC clocks. 1000: 128 ADC clocks. 1001: 192 ADC clocks. 1010: 256 ADC clocks. 1011: 320 ADC clocks. 1100: 384 ADC clocks. 1101: 512 ADC clocks. 1110: 768 ADC clocks. 1111: 1024 ADC clocks.
[3:0]	**Reserved** These bits are reserved and must be programmed to 0000.

19.3.10. ADC Window Upper Threshold High Register

The ADC Window Upper Threshold High Register, shown in Table 172, contains the unsigned MSBs of the ADC window upper threshold. This register is used in conjunction with ADCUWINL to define the ADC window upper threshold.

Table 172. ADC Window Upper Threshold High Register (ADCUWINH)

Bits	7	6	5	4	3	2	1	0
Field	UWINH							
Reset	FFh							
R/W	R/W							
Address	F79h							

Bit	Description
UWINH	**ADC Window Upper Threshold High** ADC Window Upper Threshold High is a function of RESOLUT and JUSTIFY. Interrupt is asserted if the ADC result is greater than the value of UWINH and UWINL.
JUSTIFY = 0 (Left-Justified), RESOLUT = x	
[7:0]	00–FF: The 8 MSBs of the last conversion result are compared against the 8 bits of this data register.
JUSTIFY = 1 (Right-Justified), RESOLUT = 0 (12-bit)	
[7:4]	Reserved: these bits must be programmed to 0000..
[3:0]	0–F: The 4 MSBs of the last conversion result are compared against the 4 LSBs of this data register.
JUSTIFY = 1 (Right-Justified), RESOLUT = 1 (14-bit)	
[7:6]	Reserved: these bits must be programmed to 00.
[5:0]	00–3F: The 6 MSBs of the last conversion result are compared against the 6 LSBs of this data register.

19.3.11. ADC Window Upper Threshold Low Register

The ADC Window Upper Threshold Low Register, shown in Table 173, contains the unsigned LSBs of the ADC window upper threshold. This register is used in conjunction with ADCUWINH to define the ADC window upper threshold.

Table 173. ADC Window Upper Threshold Low Register (ADCUWINL)

Bits	7	6	5	4	3	2	1	0
Field	UWINL							
Reset	FFh							
R/W	R/W							
Address	F7Ah							

Bit	Description
UWINL	**ADC Window Upper Threshold Low** ADC Window Upper Threshold Low is a function of RESOLUT and JUSTIFY. Interrupt is asserted if the ADC result is greater than the value of UWINH and UWINL.
JUSTIFY = 0 (Left-Justified), RESOLUT = 0 (12-bit)	
[7:4]	0–F: The 4 LSBs of the last conversion result of the last conversion result are compared against the 4 MSBs of this data register.
[3:0]	Reserved: these bits must be programmed to 0000.
JUSTIFY = 0 (Left-Justified), RESOLUT = 1 (14-bit)	

Bit	Description (Continued)
[7:2]	00–3F: The 6 LSBs of the last conversion result of the last conversion result are compared against the 6 MSBs of this data register.
[1:0]	Reserved: these bits must be programmed to 00.
JUSTIFY = 1 (Right-Justified), RESOLUT = x	
[7:0]	00–FF: The 8 LSBs of the last conversion result are compared against the 8 bits of this data register.

19.3.12. ADC Window Lower Threshold High Register

The ADC Window Lower Threshold High Register, shown in Table 174, contains the unsigned MSBs of the ADC window lower threshold. This register is used in conjunction with ADCLWINL to set the ADC window lower threshold.

Table 174. ADC Window Lower Threshold High Register (ADCLWINH)

Bits	7	6	5	4	3	2	1	0
Field	LWINH							
Reset	00h							
R/W	R/W							
Address	F7Bh							

Bit	Description
LWINH	**ADC Window Lower Threshold High** ADC Window Lower Threshold High is a function of RESOLUT and JUSTIFY. Interrupt is asserted if the ADC result is lower than the value of LWINH and LWINL.
JUSTIFY = 0 (Left-Justified), RESOLUT = x	
[7:0]	00–FF: The 8 MSBs of the last conversion result are compared against the 8 bits of this data register.
JUSTIFY = 1 (Right-Justified), RESOLUT = 0 (12-bit)	
[7:4]	Reserved: these bits must be programmed to 0000.
[3:0]	0–F: The 4 MSBs of the last conversion result are compared against the 4 LSBs of this data register.
JUSTIFY = 1 (Right-Justified), RESOLUT = 1 (14-bit)	
[7:6]	Reserved: these bits must be programmed to 00.
[5:0]	00–3F: The 6 MSBs of the last conversion result are compared against the 6 LSBs of this data register.

19.3.13. ADC Window Lower Threshold Low Register

The ADC Window Lower Threshold Low Register, shown in Table 175, contains the unsigned LSBs of the ADC window lower threshold. This register is used in conjunction with ADCLWINH to set the ADC window lower threshold.

Table 175. ADC Window Lower Threshold Low Register (ADCLWINL)

Bits	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	LWINL							
Reset	00h							
R/W	R/W							
Address	F7Ch							

Bit	**Description**	
LWINL	**ADC Window Lower Threshold Low** ADC Window Lower Threshold Low is a function of RESOLUT and JUSTIFY. Interrupt is asserted if the ADC result is lower than the value of LWINH and LWINL.	
JUSTIFY = 0 (Left-Justified), RESOLUT = 0 (12-bit)		
[7:4]	0–F:	The 4 LSBs of the last conversion result of the last conversion result are compared against the 4 MSBs of this data register.
[3:0]	Reserved: these bits must be programmed to 0000.	
JUSTIFY = 0 (Left-Justified), RESOLUT = 1 (14-bit)		
[7:2]	00–3F:	The 6 LSBs of the last conversion result of the last conversion result are compared against the 6 MSBs of this data register.
[1:0]	Reserved: these bits must be programmed to 00.	
JUSTIFY = 1 (Right-Justified), RESOLUT = x		
[7:0]	00–FF:	The 8 LSBs of the last conversion result are compared against the 8 bits of this data register.

Chapter 20. Operational Amplifiers

Two low-power operational amplifiers (op amps) are available with Zilog's F3224 Series MCUs: Op Amp A0 and Op Amp A1. These amplifiers are identical to each other and each can be configured internally with various voltage gain settings or be internally configured to provide unity gain feedback. Each op amp input and output is accessible from the package pins.

Features include:

- Two general-purpose op amps (Op Amp A0 and Op Amp A1), individually enabled and configured
- Rail-to-rail inputs and outputs
- Two power vs. bandwidth settings featuring low active currents of 1 µA and 30 µA
- Flexible multiplexed op amp inputs and outputs
- Outputs can drive selectable internal destinations such as the ADC, comparators and op amp inputs without consuming a GPIO
- Internal input and output connections available to conserve pins
- Can be internally configured as a unity gain buffer
- Can be configured as a programmable gain amplifier using an internal programmable resistive feedback network that provides 16 gain steps

20.1. Architecture

Figure 60 shows a simplified block diagram of Op Amp A, including input connections and feedback paths for unity gain and programmable gain.

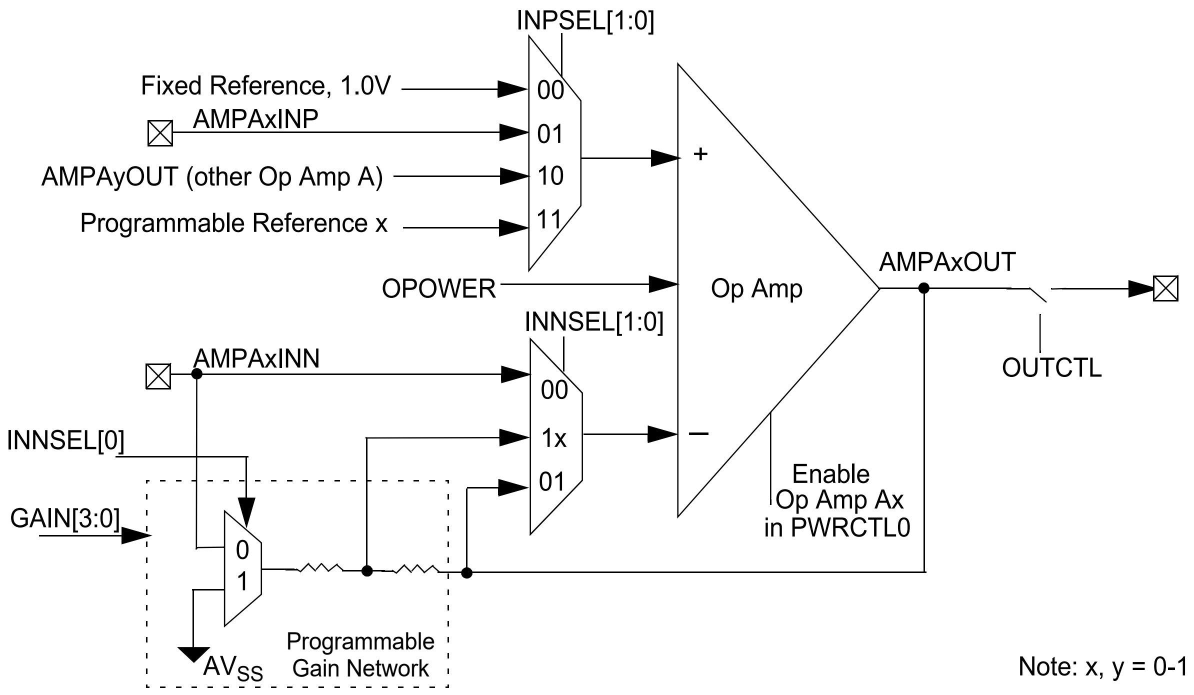


Figure 60. Op Amp A Block Diagram

20.2. Operation

The identical Op Amp A0 and Op Amp A1 amplifiers feature independent control, and provide rail-to-rail operation for both inputs and outputs. They are enabled by setting OpAmpA0 and OpAmpA1, respectively, in the PWRCTL0 Register, which is described in the Low-Power Modes chapter on page 41. If enabled, an amplifier remains active in all modes, even in Stop Mode. If the amplifier is not required in Stop Mode, disable it. Failing to disable it results in higher Stop Mode current than necessary.

Op amp power consumption and bandwidth can be adjusted by setting or clearing the OPOWER bit. Low power/bandwidth, nominally 1 µA and 40 kHz unity gain bandwidth, is selected by clearing this OPOWER bit, whereas normal power/bandwidth, nominally 30 µA and 620 kHz unity gain bandwidth, is selected by setting OPOWER. With these settings, the op amps can support many analog front-end sensing applications, providing signal conditioning ahead of any digital conversion with the integrated ADC or comparators.

All inputs and outputs can be selected to connect to assigned GPIO pins to support user external feedback and coupling networks to meet analog front-end acquisition requirements. To connect GPIOs to an op amp, configure the appropriate alternate function, as described in the General-Purpose Input/Output chapter on page 46, and configure the

OUTCTL bit. In addition, to reduce the demand for external pins or components, op amps can be configured for the following internal connections:

- Unity gain buffer
- Programmable gain amplifier using an internal programmable gain network with 16 available gain selections, 1.5x to 64x, that are selected by writing to the GAIN bit
- Inputs from the reference system, including the internal programmable references and the 1.0V internal fixed reference (Op Amp A1)
- Outputs to op amp inputs, Comparator 0, Comparator 1, and the ADC

The internal connections can also improve performance by eliminating external board and connectivity from loading while going off- and on-chip.

The op amp outputs share package pin connections with ADC inputs. When making an ADC measurement that does not involve an op amp on such a shared pin, either disable the op amp or disconnect it from the GPIO with the appropriate OUTCTL setting.

As shown in Figure 60 on page 331, four positive and four negative inputs are available. The positive Op Amp Ax inputs are selected with the INPSEL bit in the AMPACTL0 Register, and include:

- 1.0V internal fixed reference; see the Reference System chapter on page 300 to learn more
- A GPIO pin used as the Op Amp Ax positive input, AMPAxINP (x = 0-1)
- Op Amp A1 output for Op Amp A0 input, Op Amp A0 output for Op Amp A1 input. This selection provides an internal connection that does not involve the GPIO used as the Op Amp Ax output, AMPAxOUT
- Internal Programmable Reference 0 for Op Amp A0, Internal Programmable Reference 1 for Op Amp A1, with level selected by the PREFLVL bit, and source selected by the PREFSRC bit in the PREF0CTL Register; see Table 160 on page 305 to learn more

The negative Op Amp Ax inputs are selected with the INNSEL bit in the AMPACTL1 Register, and include:

- GPIO pin used as Op Amp Ax negative input, AMPAINN
- Op Amp Ax output, a unity gain configuration using an internal connection
- Op Amp Ax output through internal feedback network using an internal connection to AMPAINN with gain, defined by the GAIN bit
- Op Amp Ax output through internal feedback network using an internal connection to AV_{SS} with gain, defined by the GAIN bit

The Op Amp Ax output, AMPAxOUT, can be selected as an internal input to another op amp, Comparator 0, Comparator 1, and the ADC. Additionally, it can be connected to the GPIO used as AMPAxOUT by setting the OUTCTL bit in the AMPAxCTL0 Register, and configuring the appropriate alternate function, as described in the General-Purpose Input/Output chapter on page 46. This GPIO can also be selected as an input to the ADC.

20.3. Op Amp Register Definitions

The four op amp registers are briefly summarized in Table 176; their bits are defined in Tables 177 through 178.

Table 176. Op Amp Register Summary

Name	Address	Description
AMPA*x*CTL0	F94h, F96h	Configuration for Op Amp A*x*
AMPA*x*CTL1	F95h, F97h	Programmable Gain & Configuration for Op Amp A*x*

20.3.1. Op Amp A0-1 Control 0 Register

The Op Amp A0-1 Control 0 Register, shown in Table 177, contains configuration for Op Amp Ax.

Table 177. Op Amp A0-1 Control 0 Register (AMPA*x*CTL0)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	OUTCTL	Reserved					INPSEL	
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	AMPA0CTL0 @ F94h, AMPA1CTL0 @ F96h							

Bit	**Description**
[7] OUTCTL	**Output Control** 0: Op Amp A*x* output is disconnected from the GPIO used as AMPA*x*OUT. AMPA*x*OUT can be selected as an internal input to another op amp, Comparator 0, Comparator 1, and the ADC without consuming a GPIO. 1: Op Amp A*x* output is connected to the GPIO used as AMPA*x*OUT. It is also necessary to configure the appropriate alternate function, as described in the General-Purpose Input/Output chapter on page 46. This GPIO can also be selected as an input to the ADC.
[6:2]	**Reserved** These bits are reserved and must be programmed to 00000
[1:0] INPSEL	**Positive Input Signal Select** 00: 1.0 V (nominal) reference from the Reference System. 01: GPIO pin used as Op Amp A positive input, AMPAINP. 10: Op Amp A*x* output. This selection provides an internal connection that does not involve the GPIO used as Op Amp A*x* output, AMPA*x*OUT. The configuration is as follows: – AMPA0OUT -> AMPA1INP – AMPA1OUT -> AMPA0INP 11: Internal Programmable Reference *x*, with level selected by PREFLVL and source selected by PREFSRC in the PREF*x*CTL Register; see the Reference System chapter on page 300 to learn more.The configuration is as follows: – Programmable Reference 0 -> Op Amp A0 – Programmable Reference 1 -> Op Amp A1

20.3.2. Op Amp A0-1 Control 1 Register

The Op Amp A0-1 Control 1 Register, shown in Table 178, contains configuration for programmable gain and Op Amp Ax.

Table 178. Op Amp A0-1 Control 1 Register (AMPA*x*CTL1)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	GAIN				OPOWER	Reserved	INNSEL	
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	R/W
Address	AMPA0CTL1 @ F95h, AMPA1CTL1 @ F97h							

Bit	**Description**
[7:4] GAIN	**Internal Voltage Gain Setting** GAIN is effective only when INNSEL[1]=1. 0000: 1.5x. 0001: 2.0x. 0010: 2.5x. 0011: 3.0x. 0100: 3.75x. 0101: 4.0x. 0110: 5.0x. 0111: 6.0x. 1000: 7.5x. 1001: 8.0x. 1010: 10x. 1011: 12x. 1100: 15x. 1101: 20x. 1110: 30x. 1111: 60x.
[3] OPOWER	**Op Amp Power/Speed Select** 0: Low power, 1 µA current, 40 kHz unity gain bandwidth (nominal values). 1: Normal power, 30 µA current, 620 kHz unity gain bandwidth (nominal values).
[2]	**Reserved** This bit is reserved and must be programmed to 0.
1:0 INNSEL	**Negative Input Signal Select** 00: GPIO pin used as Op Amp Ax negative input, AMPA*x*INN. 01: Op Amp A*x* output, unity gain configuration using internal connection. 10: Op Amp A*x* output through internal gain network using internal connection to AMPA*x*INN with gain defined by GAIN. 11: Op Amp A*x* output through internal gain network using internal connection to AV_{SS} with gain defined by GAIN.

Chapter 21. Comparators

The F3224 Series devices feature two identical general-purpose, rail-to-rail comparators, each of which compares two analog input signals with four speed-vs.-power settings and three hysteresis options. A 4-to-1 input multiplexer exists on each comparator positive input and each comparator negative input. Multiplexing can be configured such that a GPIO (C0INxP/C1INxP) pin provides a positive comparator input and/or a GPIO (C0INxN/C1INxN) provides a negative input. The output of each comparator is available as an interrupt source and can be routed to an external pin using the GPIO multiplex, as well as to the Event System.

Features for each comparator include:

- Positive input selections offering two GPIOs and op amp outputs AMPA0OUT and AMPA1OUT
- Negative input selections offering two GPIOs, fixed internal reference levels, and a programmable internal reference,
- Output can be an interrupt source
- Output can drive an external pin and/or be an Event System source
- Operation in Stop Mode
- Power-vs.-speed control with four available settings
- Hysteresis control with three available settings
- Window detection: signal above window, signal inside window, signal below window
- Additional output in the form of a logical OR of each comparator output, is an Event System source, is useful for window detection signaling

21.1. Architecture

Figure 61 shows a simplified block diagram of the comparators, including input and output connections. Each of the two comparators is identical.

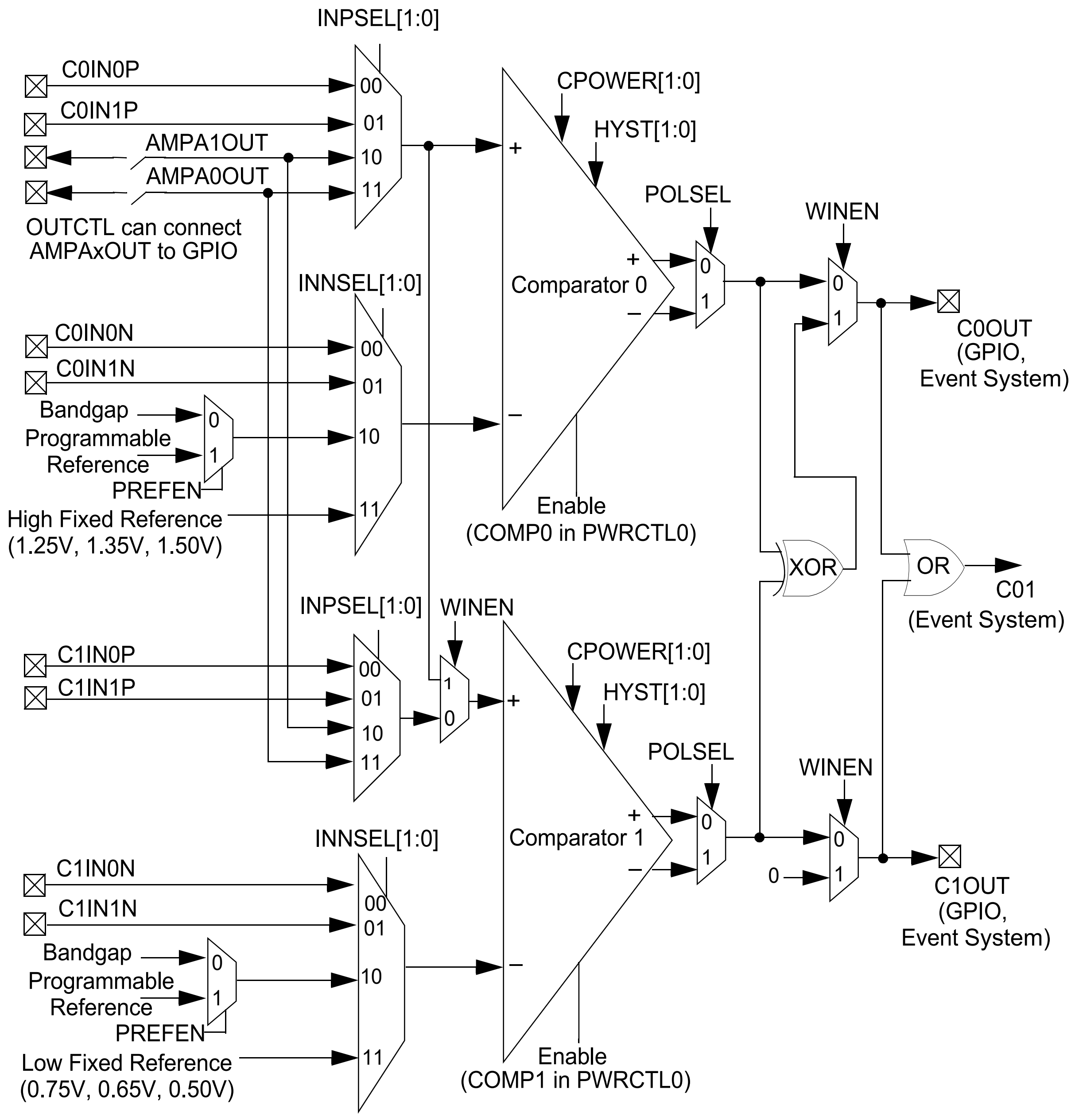

INPSEL[1:0]
C0IN0P
C0IN1P
AMPA1OUT
AMPA0OUT
OUTCTL can connect AMPAxOUT to GPIO
CPOWER[1:0]
HYST[1:0]
POLSEL
WINEN
INNSEL[1:0]
Comparator 0
C0IN0N
C0IN1N
Bandgap
Programmable Reference
PREFEN
High Fixed Reference (1.25V, 1.35V, 1.50V)
Enable (COMP0 in PWRCTL0)
C0OUT (GPIO, Event System)
XOR
OR
C01 (Event System)
C1IN0P
C1IN1P
Comparator 1
C1IN0N
C1IN1N
Low Fixed Reference (0.75V, 0.65V, 0.50V)
Enable (COMP1 in PWRCTL0)
C1OUT (GPIO, Event System)

Figure 61. Comparators Block Diagram

21.2. Comparator Operation

Two identical general-purpose CMOS analog comparators each provide rail-to-rail operation with four speed-vs.-power settings and three hysteresis options. These comparators are enabled by setting the COMP0 and COMP1 bits in the PWRCTL0 Register, which is described in the Low-Power Modes chapter on page 41. The power setting is determined by the CPOWER bit, which selects current consumption ranging from 27 µA, with a propagation delay of 150 ns, to 0.2 µA, with a propagation delay of 10 µs. The low power settings can allow for continuous comparator usage in low-power systems. Hysteresis is selected by the HYST bit; selections range from no hysteresis to 40 mV.

A 4-to-1 input multiplexer exists on each comparator positive input and each comparator negative input. The positive input is selected using the INPSEL bit to be either one of two GPIOs or one of the op amp outputs, AMPA0OUT or AMPA1OUT. The negative input is selected using the INNSEL and PREFEN bits to be either one of two GPIOs, a fixed reference, the bandgap voltage, or a programmable internal reference. Multiplexing can be configured such that a GPIO (C0INxP/C1INxP) pin provides the positive comparator input and/or a GPIO (C0INxN/C1INxN) provides the negative input. When connecting to GPIO, use the appropriate GPIO alternate function selection, as described in the General-Purpose Input/Output chapter on page 46.

The comparator output polarity is determined by the POLSEL bit. When POLSEL = 0, the comparator output is noninverted such that the comparator output is High when the positive comparator input voltage is greater than the negative comparator input voltage. When POLSEL = 1, the comparator output is inverted such that the comparator output is Low when the positive comparator input voltage is greater than the negative comparator input voltage.

The output of each comparator can be routed to a GPIO pin, C0OUT or C1OUT, as well as to the Event System. When connecting to GPIO, use the appropriate GPIO alternate function selection, as described in the General-Purpose Input/Output chapter on page 46. Additionally, the comparator output state can be read directly from the CSTATUS bit in the CMPCTL Register. An additional output, C01, is the logical OR of each comparator output, and is an Event System source; it is useful for window detection signaling.

A window compare feature provides coordinated detection reporting for the two comparators. WINEN = 1 selects Window Mode. The impact of WINEN and POLSEL on the comparator outputs is summarized in Table 179 on page 339.

Table 179. Effect of WINEN and POLSEL on Comparator Outputs

POLSEL		Input Condition (Positive Input vs. Negative Input)		WINEN=0				WINEN=1			
COMP0	**COMP1**	**COMP0**	**COMP1**	**C0 OUT**	**C1 OUT**	**C01**	**CSTAT**	**C0 OUT**	**C1 OUT**	**C01**	**CSTAT, Window State***
0 (noninv)	0 (noninv)	+ < –	+ < –	0	0		00	0	0	0	01, Below
		+ < –	+ > –	0	1	1	01	1	0	1	00, Inside
		+ > –	+ < –	1	0	1	10	1	0	1	00, Inside
		+ > –	+ > –	1	1	1	11	0	0	0	10, Above
0 (noninv)	1 (inv)	+ < –	+ < –	0	1	1	01	1	0	1	00, Inside
		+ < –	+ > –	0	0	0	00	0	0	0	01, Below
		+ > –	+ < –	1	1	1	11	0	0	0	10, Above
		+ > –	+ > –	1	0	1	10	1	0	1	00, Inside
1 (inv)	0 (noninv)	+ < –	+ < –	1	0	1	10	1	0	1	00, Inside
		+ < –	+ > –	1	1	1	11	0	0	0	10, Above
		+ > –	+ < –	0	0	0	00	0	0	0	01, Below
		+ > –	+ > –	0	1	1	01	1	0	1	00, Inside
1 (inv)	1 (inv)	+ < –	+ < –	1	1	1	11	0	0	0	10, Above
		+ < –	+ > –	1	0	1	10	1	0	1	00, Inside
		+ > –	+ < –	0	1	1	01	1	0	1	00, Inside
		+ > –	+ > –	0	0	0	00	0	0	0	01, Below

Note: *Window state naming is from the perspective of noninverted polarity (POLSEL = 0) for both comparators.

In support of Window Mode, the positive input for both comparators can be configured to be a common signal in the following three ways:

- Configure WINEN=1 (Window Mode) which selects C0INP to be input to both comparators.
- Select AMPA0OUT as the positive input for both comparators
- Select AMPA1OUT as the positive input for both comparator

The comparator outputs are used to provide interrupts, as described in the Interrupt Controller chapter on page 89.

The comparator can be powered down to save supply current or can continue to operate in Stop Mode. For details, see the Power Control Register 0 on page 43. In Stop Mode, the comparator interrupt, if enabled, automatically initiates a Stop-Mode Recovery and generates an interrupt request. In the Reset Status Register (RSTSTAT) (see page 39), the stop bit is set to 1. Additionally, the Comparator request bit in the Interrupt Request 2 Register (see page 97) is set. Following completion of the Stop-Mode Recovery, and if interrupts are enabled, the CPU responds to the interrupt request by fetching the comparator interrupt vector.

Caution: Because of the propagation delay of the comparator, spurious interrupts can result after enabling the comparator. Zilog recommends not enabling the comparator without first disabling interrupts, then waiting for the comparator output to settle.

The following code example shows how to safely enable the comparator:

```
di
ldx CMP0CTL0,r0 ; set-up comparator
ldx CMP0CTL1,r1 ; set-up comparator
ldx PWRCTL0,r2  ; enable comparators
nop
nop             ; wait for output to settle
ldx IRQ2,#0     ; clear any spurious interrupts pending
ei
```

21.3. Comparator Register Definitions

This section defines the features of the following Comparator and Reference System registers:

Comparator Control Register (CMPCTL) at address `F8Fh`

Comparator 0 Control 0 Register (CMP0CTL0) at address `F90h`

Comparator 0 Control 1 Register (CMP0CTL1) at address `F91h`

Comparator 1 Control 0 Register (CMP1CTL0) at address `F92h`

Comparator 1 Control 1 Register (CMP1CTL1) at address `F93h`

21.3.1. Comparator Control Register

The Comparator Control Register, shown in Table 180, provides global control and status for both comparators.

Table 180. Comparator Control Register (CMPCTL)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	Reserved					WINEN	CSTATUS	
Reset	0	0	0	0	0	0	0	0
R/W	R	R	R	R	R	R/W	R	R
Address	F8Fh							

Bit	Description
[7:3]	**Reserved** These bits are reserved and must be programmed to 000.
[2] WINEN	**Window Mode Enable** 0: Normal mode, Comparator output (COUT), STATUS and comparator interrupts are based on independent comparators. Comparator 1 positive input is selected using its INPSEL. 1: Window Mode. Comparator output (COUT), STATUS and comparator interrupts are based on a window logic function of both comparators. Comparator 1 positive input is the selected Comparator 0 positive input.
[1:0] CSTATUS	**Comparator Status** Status is dependent upon the state of WINEN. **WINEN=0** 0x: Comparator 0 Output (C0OUT) is Low*. 1x: Comparator 0 Output (C0OUT) is High*. x0: Comparator 1 Output (C1OUT) is Low*. x1: Comparator 1 Output (C1OUT) is High*. **WINEN=1** Window state naming is from the perspective of noninverted polarity (POLSEL = 0) for both comparators 01: Inside Window (C0OUT ≠ C1OUT)*. 01: Below Window (C0OUT=C1OUT=0)*. 10: Above Window (C0OUT=C1OUT=1)*. 11: Reserved.

Note: *C0OUT and C1OUT include the effect of POLSEL.

21.3.2. Comparator 0 Control 0 Register

The Comparator 0 Control 0 Register is shown in Table 181.

Table 181. Comparator 0 Control 0 Register (CMP0CTL0)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	CPOWER		HYST		INNSEL		INPSEL	
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F90h							

Bit	**Description**
[7:6] CPOWER	**Comparator Power/Speed Select** 00: Ultra-low power, current= 200nA, Tpd=10µs (nominal values). 01: Low power, current=1µA, Tpd=1.5µs (nominal values). 10: Normal power, current =4µA, Tpd= 700ns (nominal values). 11: High speed/power, current = 27µA, Tpd =150ns (nominal values).
[5:4] HYST	**Hysteresis Level Select** 00: None, 0mV. 01: 15mV (nominal). 10: Reserved. 11: 40mv (nominal).
[3:2] INNSEL	**Negative Input Signal Select** 00: GPIO pin used as Comparator 0 negative input, C0IN0N. 01: GPIO pin used as Comparator 0 negative input, C0IN1N. 10: If PREFEN=0, bandgap reference from the Reference Generator. If PREFEN=1, Programmable Reference 0, with level selected by PREFLVL and source selected by PREFSRC. 11: 1.25V, 1.35v, or 1.50V (nominal) reference from the Reference Generator.
[1:0] INPSEL	**Positive Input Signal Select** 00: GPIO pin used as Comparator 0 positive input, C0IN0P. 01: GPIO pin used as Comparator 0 positive input, C0IN1P. 10: Op Amp A1 output. This selection provides an internal connection that does not involve the GPIO used as Op Amp A1 output, AMPA1OUT. 11: Op Amp A0 output. This selection provides an internal connection that does not involve the GPIO used as Op Amp A0 output, AMPA0OUT.

21.3.3. Comparator 0 Control 1 Register

The Comparator 0 Control 1 Register is shown in Table 182. It provides control for Comparator 0 and Programmable Reference 0.

Table 182. Comparator 0 Control 1 Register (CMP0CTL1)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	POLSEL	Reserved						
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R	R	R	R	R	R	R
Address	F91h							

Bit	**Description**
[7] POLSEL	**Polarity Select** 0: Noninverted comparator output. The comparator output is High when the positive comparator input voltage is greater than the negative comparator input voltage. 1: Inverted comparator output. The comparator output is Low when the positive comparator input voltage is greater than the negative comparator input voltage.
[6:0]	**Reserved** These bits are reserved and must be programmed to 00H.

21.3.4. Comparator 1 Control 0 Register

The Comparator 1 Control 0 Register is shown in Table 183.

Table 183. Comparator 1 Control 0 Register (CMP1CTL0)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	CPOWER		HYST		INNSEL		INPSEL	
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F92h							

Bit	**Description**
[7:6] CPOWER	**Comparator Power/Speed Select** 00: Ultra-low power, current=200nA, Tpd=10µs (nominal values). 01: Low power, current=1µA, Tpd=1.5µs (nominal values). 10: Normal, current=4µA, Tpd=700ns (nominal values). 11: High Speed/Power, current=27µA, Tpd=150ns (nominal values).
[5:4] HYST	**Hysteresis Level Select** 00 None, 0mV. 01: 15mV (nominal). 10: Reserved. 11: 40mv (nominal).
[3:2] INNSEL	**Negative Input Signal Select** 00: GPIO pin used as Comparator 1 negative input, C1IN0N. 01: GPIO pin used as Comparator 1 negative input, C1IN1N. 10: If PREFEN=0, bandgap reference from the Reference Generator. If PREFEN=1, Programmable Reference 1, with level selected by PREFLVL and source selected by PREFSRC. 11: 0.75V, 0.65V, or 0.50V (nominal) reference from the Reference Generator.
[1:0] INPSEL	**Positive Input Signal Select** Comparator 1 Positive Input Signal select is dependent upon the state of WINEN. **WINEN=0:** 00: GPIO pin used as Comparator 1 positive input, C1IN0P. 01: GPIO pin used as Comparator 1 positive input, C1IN1P. 10: Op Amp A1 output. This selection provides an internal connection that does not involve the GPIO used as Op Amp A1 output, AMPA1OUT. 11: Op Amp A0 output. This selection provides an internal connection that does not involve AMPA0OUT, which is the GPIO used as the Op Amp A0 output. **WINEN=1:** xx: The selected Comparator 0 positive input.

21.3.5. Comparator 1 Control 1 Register

The Comparator 1 Control 1 Register is shown in Table 184. It provides control for Comparator 1 and Programmable Reference 1.

Table 184. Comparator 1 Control 1 Register (CMP1CTL1)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	POLSEL	Reserved						
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R	R	R	R	R	R	R
Address	F93h							

Bit	**Description**
[7] POLSEL	**Polarity Select** 0: Noninverted comparator output. The comparator output is High when the positive comparator input voltage is greater than the negative comparator input voltage. 1: Inverted comparator output. The comparator output is Low when the positive comparator input voltage is greater than the negative comparator input voltage.
[6:0]	**Reserved** These bits are reserved and must be programmed to 00H.

Chapter 22. Flash Memory

The products in the F3224 Series feature either 32 KB (32768) or 16KB (16384) of non-volatile Flash memory with read/write/erase capability. This Flash memory can be programmed and erased in-circuit by either user code or through the On-Chip Debugger.

The Flash memory array is arranged in pages with 512 bytes per page. The 512-byte page is the minimum Flash memory area that can be erased. Each page is divided into 4 rows of 128 bytes.

For program/data protection, a block of Flash memory can be protected. The size of the protected block is configured to be at the desired page boundary.

The first 2 bytes of Flash Program Memory are used as Flash option bits. For more information about their operation, see the Option Bit Types section on page 358.

Table 185 lists the Flash memory configuration for each device in the F3224 Series; Figure 62 shows the Flash memory arrangement.

Table 185. F3224 Series Flash Memory Configurations

Part Number	Flash Size in KB (Bytes)	Flash Pages	Program Memory Addresses
Z8F3224	32 (32768)	64	0000h–7FFFh
Z8F1624	16 (16384)	32	0000h–3FFFh

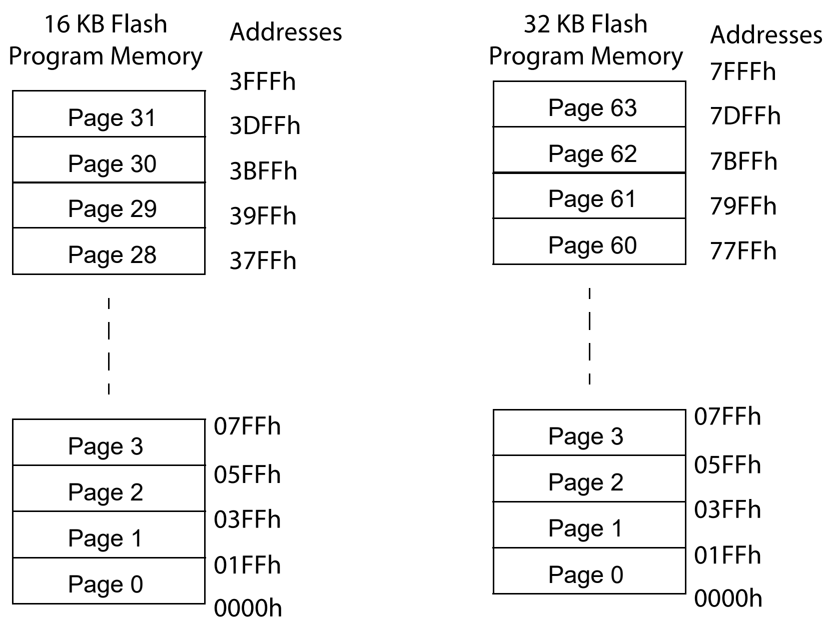


Figure 62. Flash Memory Arrangement

22.1. Flash Information Area

The Flash Information Area is separate from Program Memory and is mapped to the page in the address range `FE00h` to `FFFFh`. This area is used primarily for factory trimming purposes and is not user-accessible. The factory trim bits' working values can be accessed using the Trim Bit Address and Trim Bit Data registers, as described in the Flash Option Bits chapter on page 358.

To map the Flash Information Area to Program Memory address range `FE00h` to `FFFFh`, set the INFO_EN bit in the Flash Page Select Register (FPS).

22.2. Operation

The Flash Controller programs and erases Flash memory, and provides the proper Flash controls and timing for byte programming, Page Erase, and Mass Erase operations in Flash memory. The Flash Controller also contains several protection mechanisms to prevent accidental programming or erasure; these mechanisms operate on the page, block, and full-memory levels.

The flow chart in Figure 63 shows basic Flash Controller operation. The following sections provide details about the Lock, Unlock, Byte Programming, Page Protect, Page Unprotect, Page Select Page Erase, and Mass Erase operations listed in Figure 63.

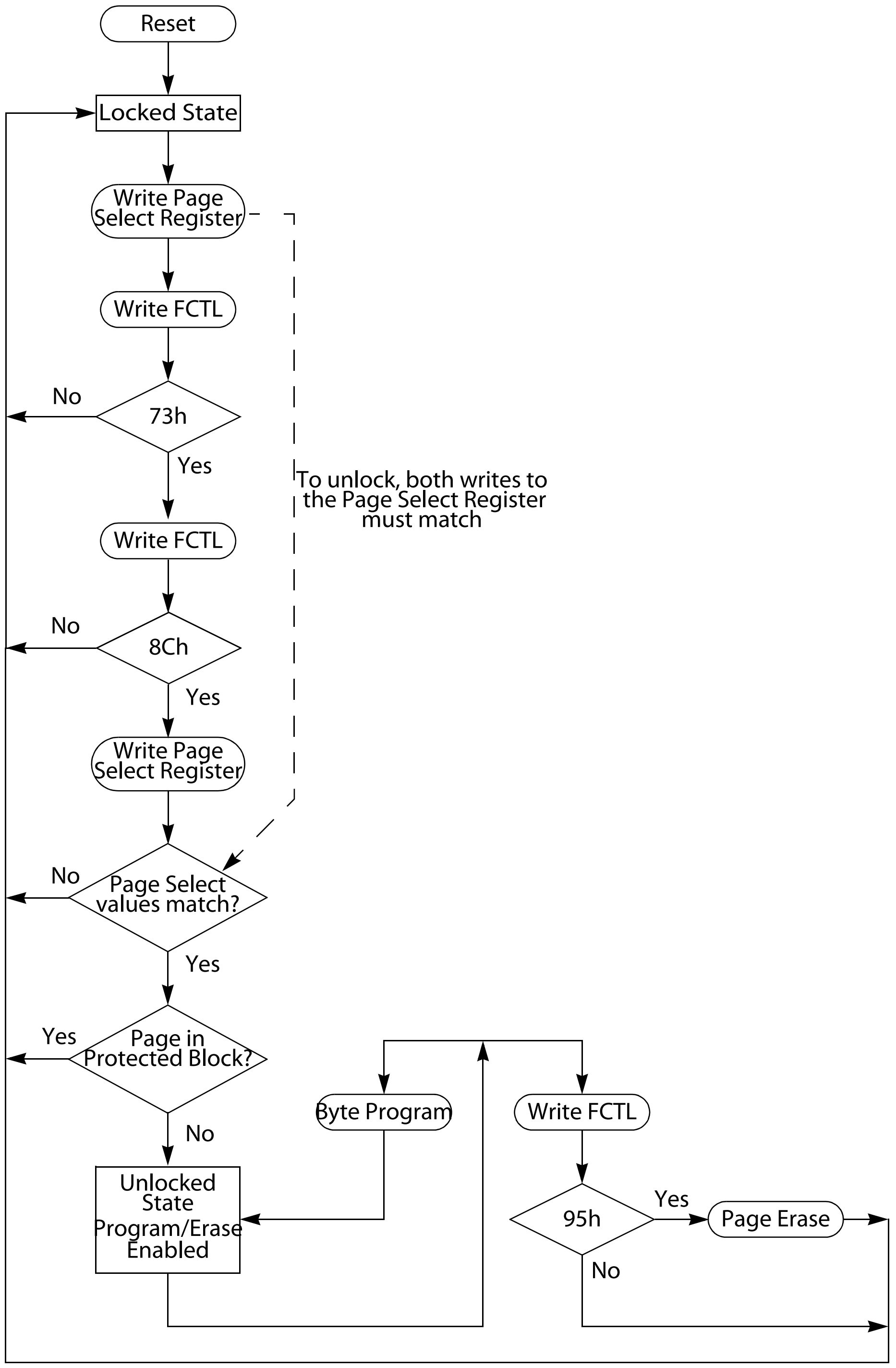

Reset
Locked State
Write Page Select Register
Write FCTL
73h
No
Yes
Write FCTL
8Ch
No
Yes
Write Page Select Register
To unlock, both writes to the Page Select Register must match
Page Select values match?
No
Yes
Page in Protected Block?
Yes
No
Unlocked State Program/Erase Enabled
Byte Program
Write FCTL
95h
Yes
No
Page Erase

Figure 63. Flash Controller Operation Flowchart

22.2.1. Flash Operation Timing

Before performing either a program or erase operation on Flash memory, the Digitally Controlled Oscillator (DCO) must be running and must be locked using the Frequency Locked Loop (FLL) to a minimum frequency of 1 MHz.

22.2.2. Flash Code Protection Against External Access

The user code contained within Flash memory can be protected against external access with the On-Chip Debugger. Programming the RDP Flash option bit prevents the reading of user code with the On-Chip Debugger. To learn more, see the Option Bit Types section on page 358 and the On-Chip Debugger chapter on page 372.

22.2.3. Flash Code Protection Against Accidental Program and Erasure

The F3224 Series provides several levels of protection against accidental program and erasure of the Flash memory contents. This protection is provided by a combination of the Flash option bits, the register locking mechanism, the page select redundancy, and the block level protection control of the Flash Controller.

22.2.3.1. Flash Code Protection Using the Flash Option Bits

The WRP Flash option bit provides Flash Program Memory protection as listed in Table 186. To learn more, see the Option Bit Types section on page 358.

Table 186. Flash Code Protection Using the Flash Option Bit

WRP	Flash Code Protection Description
0	User code programming and erasing are disabled for all of Flash Program Memory. Mass Erase is available through the On-Chip Debugger.
1	User code programming and Page Erase are enabled for all of Flash Program Memory. Mass Erase is available through the On-Chip Debugger.

22.2.3.2. Flash Code Protection Using the Flash Controller

At Reset, the Flash Controller locks to prevent accidental program or erasure of Flash memory. Observe the following procedure to unlock the Flash Controller from user code:

1. Write the Page Select Register with the target page.
2. Write the first unlock command, `73h`, to the Flash Control Register.
3. Write the second unlock command, `8Ch`, to the Flash Control Register.
4. Rewrite the Page Select Register with the target page previously stored in this register in Step 1.

If the two Page Select writes do not match, the controller reverts to a locked state. If the two writes match, the selected page becomes active. For details, see Figure 63 on page 349.

Note: The Programming, Page Erase and Mass Erase operations will not be allowed if the WRP bit is cleared or if the page resides in a protected block.

After unlocking a specific page, Byte Programing or Page Erase may be performed. At the conclusion of a Page Erase, the Flash Controller is automatically locked. To lock the Flash Controller after byte programming, write the Flash Control Register with any value other than the Page Erase or Mass Erase commands.

22.2.3.3. Flash Block Protection

The final protection mechanism is implemented on a block basis. Any number of contiguous pages in Flash memory, starting from page 0, can be protected. When set, the FBP_EN bit in the Flash Block Protection Register enables Flash block protection. When Flash block protection is enabled, the FBPS field in the Flash Block Protection Register identifies the page number of the first page that is not protected. All pages below this page are protected.

The Flash Block Protect Register is shared with the Page Select Register, and is selected for access by writing the `5Eh` command byte to the Flash Control Register while the Flash Controller is locked. When selected, any subsequent read or write to the Page Select Register targets the Flash Block Protect Register. To deselect the Flash Block Protect Register, write any value to the Flash Control Register.

The Flash Block Protect Register is initialized to 0 on Reset, putting each page into an unprotected state. When the FBP_EN bit in the Flash Block Protect Register is written to 1, the block of Flash pages up to – but not including – the page number in the FBPS field can no longer be written or erased. After the FBP_EN bit of the Flash Block Protect Register has been set, it cannot be cleared except by a System Reset.

22.2.4. Programming

Flash memory is enabled for byte programming on the active page after unlocking the Flash Controller. Erase the address(es) to be programmed using either the Page Erase or Mass Erase command prior to programming. An erased Flash byte contains all ones (`FFh`). The programming operation can only be used to change bits from 1 to 0. To change a Flash bit (or multiple bits) from 0 to 1 requires execution of either the Page Erase or Mass Erase command.

Programming can be performed using the On-Chip Debugger's Write Memory command or an eZ8 CPU execution of the LDC or LDCI instructions. For a description of these LDC and LDCI instructions, refer to the eZ8 CPU Core User Manual (UM0128), which is available free for download from the Zilog website. While the Flash Controller programs Flash memory, the eZ8 CPU remains idle, but the system clock and on-chip peripherals continue to operate.

After an address is written, the page remains unlocked, allowing for subsequent writes to other addresses on the same page. To exit programming mode and lock Flash memory, write any value to the Flash Control Register except for the Mass Erase or Page Erase commands.

22.2.5. Page Erase

Flash memory can be erased one page (512 bytes) at a time. Page erasing Flash memory sets all bytes in the active page to the value `FFh`. The Flash Page Select Register identifies the page to be erased. Only a page residing outside the protected block can be erased. With the Flash Controller unlocked, writing the value `95h` to the Flash Control Register initiates the Page Erase operation on the active page. While the Flash Controller executes the Page Erase operation, the eZ8 CPU remains idle, but the system clock and on-chip peripherals continue to operate. The eZ8 CPU resumes operation after the Page Erase operation completes. If the Page Erase operation is performed using the OCD, poll the Flash Status Register to determine when the Page Erase operation is complete. When the Page Erase is complete, the Flash Controller returns to its locked state.

22.2.6. Mass Erase

Flash memory can also be mass erased using the Flash Controller, but only by using the On-Chip Debugger. Mass erasing Flash memory sets all bytes to the value `FFh`. With the Flash Controller unlocked, writing the value `63h` to the Flash Control Register initiates the Mass Erase operation. While the Flash Controller executes the Mass Erase operation, the eZ8 CPU remains idle, but the system clock and on-chip peripherals continue to operate. Using the On-Chip Debugger, poll the Flash Status Register to determine when the Mass Erase operation is complete. When the Mass Erase is complete, the Flash Controller returns to its locked state.

Mass Erase does not affect the user page in the Flash Information Area. Use Page Erase to erase the user page in the Flash Information Area.

22.2.7. Flash Controller Behavior in Debug Mode

The following changes in the behavior of the Flash Controller occur when the Flash Controller is accessed using the On-Chip Debugger:

- The Flash Write Protect option bit is ignored
- The Flash Block Protect Register is ignored for programming operations
- Programming operations are not limited to the page selected in the Page Select Register
- Bits in the Flash Block Protect Register can be written to 1 or 0
- The second write of the Page Select Register to unlock the Flash Controller is not necessary in Debug Mode and neither write is necessary in Debug Mode Tool Option.
- The Page Select Register can be written when the Flash Controller is unlocked.
- The Mass Erase command is enabled through the Flash Control Register. Alternatively, the OCD Mass Erase command can be utilized.
- If read protected (RDP=0), page erase by the On-Chip Debugger is disabled.

Caution: For security reasons, the Flash controller allows only a single page to be opened for write/erase. When writing multiple Flash pages, the Flash controller must go through the unlock sequence again to select another page.

22.3. Flash Control Register Definitions

This section defines the features of the following Flash Control registers.

Flash Control Register

Flash Status Register: see page 355

Flash Page Select Register: see page 356

Flash Block Protect Register: see page 357

22.3.1. Flash Control Register

The Flash Controller must remain unlocked when using the Flash Control Register (shown in Table 187) before programming or erasing Flash memory. The Flash Controller is unlocked by writing the Flash Page Select Register, then `73h 8Ch`, sequentially, to the Flash Control Register. A final write must then be made to the Flash Page Select Register with the same value as the previous write. When the Flash Controller is unlocked, Mass Erase or Page Erase can be initiated by writing the appropriate command to the FCTL. Page Erase applies only to the active page selected in the Flash Page Select Register. Mass Erase is enabled only through the On-Chip Debugger. Writing an invalid value or an invalid sequence returns the Flash Controller to its locked state. The write-only Flash Control Register shares its Register File address with the read-only Flash Status Register.

Table 187. Flash Control Register (FCTL)

Bits	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	FCMD							
Reset	0	0	0	0	0	0	0	0
R/W	W	W	W	W	W	W	W	W
Address	FF8h							

Bit	**Description**
[7:0] FCMD	**Flash Command** 73h: First unlock command. 8Ch: Second unlock command. 95h: Page Erase command (must be third command in sequence to initiate Page Erase). 63h: Mass Erase command (must be third command in sequence to initiate Mass Erase). 5Eh: Enable Flash Block Protect Register Access.

22.3.2. Flash Status Register

The Flash Status Register, shown in Table 188, indicates the current state of the Flash Controller. This register can be read at any time. The read-only-only Flash Status Register shares its Register File address with the write-only Flash Control Register.

Table 188. Flash Status Register (FSTAT)

Bits	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	Reserved					FSTAT		
Reset	0	0	0	0	0	0	0	0
R/W	R	R	R	R	R	R	R	R
Address	FF8h							

Bit	**Description**
[7:3	**Reserved** These bits are reserved and must be programmed to 00000.
[2:0] FSTAT	**Flash Controller Status** 000: Flash Controller locked. 001: First unlock command received (73h written). 010: Second unlock command received (8Ch written). 011: Flash Controller unlocked. 100: Block Protect Register selected. 101: Program operation in progress. 110: Page erase operation in progress. 111: Mass erase operation in progress.

22.3.3. Flash Page Select Register

The Flash Page Select Register, shown in Table 189, shares address space with the Flash Block Protect Register. Unless the Flash controller was last written with `5Eh`, writes to this address target the Flash Page Select Register.

This register is used to select one of the Flash memory pages to be programmed or erased. Each Flash page contains 512 bytes of Flash memory. During a Page Erase operation, all Flash memory having addresses with the most significant 6 bits provided by FPS[5:0] are chosen for program/erase operation.

Table 189. Flash Page Select Register (FPS)

Bits	7	6	5	4	3	2	1	0
Field	INFO_EN	Reserved	PAGE					
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W
Address	FF9h							

Bit	Description
[7] INFO_EN	**Information Area Enable** 0: Information Area is not selected. 1: Information Area is selected. The Information Area is mapped into the Program Memory address space in the FE00h–FFFFh address range.
[6]	**Reserved** This bit is reserved and must be programmed to 0
[5:0] PAGE	**Page Select** This 6-bit field identifies the Flash memory page for Page Erase and page unlocking. Program Memory Address[14:9]=PAGE[5:0]. For Z8F1624 devices, the upper bit must always be 0.

22.3.4. Flash Block Protect Register

The Flash Block Protect Register, shown in Table 190, is shared with the Flash Page Select Register. This register is selected for access when the Flash Control Register (see page 354) is written with `5Eh` while the Flash Controller is locked. When selected, any subsequent read or write to this address targets the Flash Sector Protect Register. To deselect this register, write any value to the Flash Control Register.

This register selects the size of the Flash memory block to be protected. The reset state of the Flash Block Protect Register is such that Block Protect is not enabled. After the selected block is protected by setting the FBP_EN bit, it can only be unprotected (i.e., the register bits can only be cleared) by a System Reset.

Table 190. Flash Block Protect Register (FPROT)

Bits	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	FBP_EN	Reserved	FBPS					
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W
Address	FF9h							

Bit	Description (Continued)
[7] FBP_EN	**Flash Block Protection Enable** 0: Block Protection is not enabled. 1: Block Protection is enabled.
[6]	**Reserved** This bit is reserved and must be programmed to 0
[5:0] FBPS	**Flash Block Protection Size** This 6-bit field identifies the page number of the first page that is not protected. If the FBP_EN bit is set, all pages below the page number in this field are protected.

Chapter 23. Flash Option Bits

Programmable Flash option bits allow user configuration of certain aspects of F3224 Series MCU operation. The configuration data are stored in Flash Program Memory and are read during Reset. The features available for control through the Flash Option Bits include:

- Watchdog Timer time-out response selection – interrupt or System Reset
- Watchdog Timer enabled at Reset
- The ability to prevent unwanted read access to user code in Program Memory
- The ability to prevent accidental programming and erasure of all or a portion of the user code in Program Memory
- The VBO can be configured as always enabled, enabled only during Active and Halt modes to reduce Stop Mode power consumption, or disabled
- LVD voltage threshold selection
- Factory trimming information for multiple analog functions

23.1. Operation

The following sections describe Flash option bit operation.

23.1.1. Option Bit Configuration by Reset

Each time Flash option bits are programmed or erased, the device must be Reset for changes to take effect. During any Reset operation (System Reset or Stop-Mode Recovery), these Flash option bits are automatically read from Flash Program Memory and written to the Option Configuration registers. These Option Configuration registers control operation of the devices within the F3224 Series MCU. Option bit control is established before the device exits System Reset and before the eZ8 CPU begins code execution. The Option Configuration registers are not part of the Register File and are not accessible for read or write access.

23.1.2. Option Bit Types

The following sections describe the option bit types.

23.1.2.1. User Option Bits

The user option bits are contained in the first two bytes of Program Memory. Zilog provides user access to these bits because these locations contain application-specific device configurations. The information contained here is lost when page 0 of the Program memory is erased.

23.1.2.2. Trim Option Bits

The trim option bits are contained in the information area of Flash memory. These bits are factory-programmed values required to optimize the operation of onboard analog circuitry and cannot be permanently altered by the user. Program memory can be erased without endangering these values. It is possible to alter working values of these bits by accessing the Trim Bit Address and Data registers, but these working values are lost after a power loss.

There are 32 bytes of trim data. To modify one of these values, the user code must first write a value between `00h` and `1Fh` into the Trim Bit Address Register. The next write to the Trim Bit Data Register changes the working value of the target trim data byte.

Reading trim data requires the user code to write a value between `00h` and `1Fh` into the Trim Bit Address Register. The next read from the Trim Bit Data Register returns the working value of the target trim data byte.

Note: The trim address ranges from information address `20-3F` only. The remainder of the information area is not accessible via the trim bit address and data registers.

23.1.2.3. Zilog Option Bits

The Zilog option bits are also contained in the information area of Flash memory. These bits are factory-programmed values that configure device peripherals and cannot be altered by the user. Program memory can be erased without endangering these values. Prior to locking the Flash Information Area, it is possible to alter working values of these bits using the OCD Write Option Bits command, but these working values are lost after a power loss. The working value of these bits can be read by using the OCD Read Option Bits command. The programmed value of these bits can be read after selecting the lower information page using the Flash Page Select Register by reading program memory addresses `FE00h`–`FE1Fh`.

23.1.2.4. Zilog Device Data

Zilog device data are also contained in the lower information page of Flash memory. These bits are factory-programmed values that contain a part number and other manufacturing information; these values cannot be altered by the user. Program memory can be erased without endangering these values. The value of these bits can be read after selecting the lower information page using the Flash Page Select Register by reading program memory addresses `FE40h`–`FE57h`.

23.2. Flash Option Bit Control Register Definitions

This section defines the features of the following Flash option bit registers.

23.2.1. Trim Bit Address Register

The Trim Bit Address Register, shown in Table 191, contains the target address for access to the trim option bits. Trim Bit addresses in the range `00h–1Fh` map to the Information Area address range `20h–3Fh`, as indicated in Table 192.

Table 191. Trim Bit Address Register (TRMADR)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	Reserved			TRMADR				
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	FF6h							

Bit	**Description**
[7:5]	**Reserved** These bits are reserved.
[4:0] TRMADR	**Trim Bit Address** 00–1F: Selects the trim option bit register accessed when TRMDR is written; see the map in Table 192.

Table 192. Trim Bit Address Map

Trim Bit Address	Information Area Address
00h	20h
01h	21h
02h	22h
03h	23h
:	:
1Fh	3Fh

23.2.2. Trim Bit Data Register

The Trim Bit Data Register, shown in Table 193, contains the read or write data for access to the trim option bits.

Table 193. Trim Bit Data Register (TRMDR)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	TRMDR							
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	FF7h							

Bit	**Description**
[7:0] TRMDR	**Trim Bit Data** 00–FF: TRMDR is a portal providing access to all trim option bit registers as selected by the TRMADR Register.

23.3. User Option Bit Address Space

The first two bytes of Flash Program Memory, at addresses `0000h` and `0001h`, are reserved for the user-programmable Flash Option bits. See Table 194.

Table 194. Flash Option Bits at Program Memory Address 0000h

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	WDT_RES	WDT_AO	Reserved		VBOCTL		RDP	WRP
Reset	X	X	X	X	X	X	X	X
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	Program Memory 0000h							
Note: X=undefined; R/W=read/write.								

Bit	**Description**
[7] WDT_RES	**Watchdog Timer Reset** 0: Watchdog Timer time-out generates an interrupt request. Interrupts must be globally enabled for the eZ8 CPU to acknowledge the interrupt request. 1: Watchdog Timer time-out causes a System Reset. This setting is the default for unprogrammed (erased) Flash.

Bit	Description (Continued)
[6] WDT_AO	**Watchdog Timer Always ON** 0: Watchdog Timer is automatically enabled upon application of system power. Watchdog Timer cannot be disabled. 1: Watchdog Timer is enabled upon execution of the WDT instruction. After it is enabled, the Watchdog Timer can only be disabled by a Reset or Stop-Mode Recovery. This setting is the default for unprogrammed (erased) Flash.
[5:4]	**Reserved** These bits are reserved and must be prrogrammed to 11.
[3:2] VBOCTL	**Voltage Brown-Out Protection Control** 00: Reserved (defaults to disabled). 01: Voltage Brown-Out Protection is disabled. 10: Voltage Brown-Out Protection is enabled in Active and Halt modes, but is disabled in Stop Mode. 11: Voltage Brown-Out Protection is always enabled. This setting is the default for unprogrammed (erased) Flash.
[1] RDP	**Flash Read Protect** 0: User program code is inaccessible. Limited control features are available through the On-Chip Debugger. 1: User program code is accessible. All On-Chip Debugger commands are enabled. This setting is default for unprogrammed (erased) Flash.
[0] WRP	**Flash Write Protect** This option bit provides Flash Program Memory protection: 0: Flash Program Memory Programming and Page Erase through user software are disabled. Mass Erase is available using the On-Chip Debugger. 1: Programming, Page Erase and Mass Erase are enabled for all of Flash Program Memory.

Table 195. Flash Option Bits at Program Memory Address 0001h

Bit	7	6	5	4	3	2	1	0
Field	Reserved							
Reset	X	X	X	X	X	X	X	X
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	Program Memory 0001h							
Note: X=undefined; R/W=read/write.								

Bit	Description
[7:0]	**Reserved** These bits are reserved and must be programmed to FFh.

23.3.1. Trim Bit Address Space

All available trim bit addresses and their functions are summarized in Table 196. For details about each, see Tables 199 through 209. For related information, the Voltage Reference System and Trimming Order section on page 500.

Table 196. Trim Bit Address Description

Address	Function
0000h	Reserved
0001h	Reserved
0002h	VBG (Bandgap reference)
0003h	IPO (Internal Precision Oscillator)
0004h	VBO and LVD
0005h	ADC VREF (Reference Voltage)
0006h	VREG (Voltage Regulator)
0007h	Reserved
0008h	Reserved
0009h	Reserved
000Ah	Reserved
000Bh	Reserved
000Ch	VBIAS
000Dh–001Fh	Reserved

23.3.1.1. Trim Bit Address Register 0000h

The Trim Option Bits Register at address `0000h`, shown in Table 197, is reserved.

Table 197. Trim Option Bits at Address 0000h (TBA0)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	Reserved							
Reset	0	0	0	0	0	0	0	0
R/W	R	R	R	R	R	R	R	R
Address	Information Page Memory 0020h							
Note: X = undefined; R/W = read/write; R = read-only.								

Bit	Description
[7:0]	**Reserved** All bits are reserved.

23.3.1.2. Trim Bit Address 0001h

The Trim Option Bits register at address `0001h`, shown in Table 198, is reserved

Table 198. Trim Option Bits at Address 0001h

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	Reserved							
Reset	X	X	X	X	X	X	X	0
R/W	R	R	R	R	R	R	R	R
Address	Information Page Memory 0021h							
Note: X = undefined; R/W = read/write; R = read-only.								

Bit	Description
[7:0]	**Reserved** These bits are reserved.

23.3.1.3. Trim Bit Address 0002h

The Trim Option Bits Register at address `0002h`, shown in Table 199, governs control of the bandgap reference trim bits.

Table 199. Trim Option Bits at Address 0002h (TVBG)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	Reserved				VBG_TRIM			
Reset	X	X	X	X	X	X	X	X
R/W	R	R	R	R	R/W	R/W	R/W	R/W
Address	Information Page Memory 0022h							
Note: X = undefined; R/W = read/write; R = read-only.								

Bit	Description
[7:4]	**Reserved** These bits are reserved.
[3:0] VBG_TRIM	**Bandgap Voltage Trim**

23.3.1.4. Trim Bit Address 0003h

The Trim Option Bits Register at address `0003h`, shown in Table 200, governs control of the Internal Precision Oscillator trim bits.

Table 200. Trim Option Bits at Address 0003h (TIPO)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	IPO_TRIM							
Reset	X	X	X	X	X	X	X	X
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	Information Page Memory 0023h							
Note: X = undefined; R/W = read/write; R = read-only.								

Bit	Description
[7:0] IPO_TRIM	**Internal Precision Oscillator Trim Bits**

23.3.1.5. Trim Bit Address 0004h

The Trim Option Bits Register at address `0004h`, shown in Table 201, governs control of the Voltage Brown-Out and Low Voltage Detect trim bits.

Table 201. Trim Option Bits at Address 0004h (TLVD_VBO)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	VBO_TRIM			LVD_TRIM				
Reset*	X	X	X	X	X	X	X	X
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	Information Page Memory 0024h							
Note: *RESET = POR reset only; X=undefined; R/W = read/write; R = read-only.								

Bit	Description
[7:5] VBO_TRIM	**Voltage Brown Out Trim**
[4:0] LVD_TRIM	**Low Voltage Detect Trim** This trimming affects the low-voltage detection threshold. Each LSB represents a 50 mV change in the threshold level. Alternatively, the low voltage threshold can be computed from the options bit value by the following equation: LVD_LVL=3.3V – LVD_TRIM * 0.1V Typical LVD_TRIM values are listed in Table 202.

Table 202. LVD_Trim Values

	LVD Threshold (V)			
LVD_TRIM	Minimum	Typical	Maximum	Description
00000		3.25		Maximum LVD threshold
00001		3.20		
00010		3.15		
00011		3.10		
00100		3.05		
00101		3.00		
00110		2.95		
00111		2.90		
01000		2.85		
01001		2.80		
01010		2.75		
01011		2.70		
01100		2.65		
01101		2.60		
01110		2.55		
01111		2.50		
10000		2.45		
10001		2.40		
10010		2.35		
10011		2.30		
10100		2.25		
10101		2.20		
10110		2.15		
10111		2.10		
11000		2.05		
11001		2.00		
11010		1.95		
11011		1.90		
11100		1.85		
11101		1.80		
11110		1.75		
11111		1.70		Minimum LVD threshold; default on Reset.

23.3.1.6. Trim Bit Address 0005h

In the Trim Option Bits Register at address `0005h` and shown in Table 203, govern control of the ADC Voltage Reference (V_{REF}).

Table 203. Trim Option Bits at Address 0005h (TVREF)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	Reserved				VREF_TRIM			
Reset	X	X	X	X	X	X	X	X
R/W	R	R	R	R	R	R/W	R/W	R/W
Address	Information Page Memory 0025h							
Note: X = undefined; R/W = read/write; R = read-only.								

Bit	Description
[7:4]	**Reserved.** These bits are reserved.
[3:0] VREF_TRIM	**ADC Voltage Reference Trim**

23.3.1.7. Trim Bit Address 0006h

The Trim Option Bits Register at address `0006h`, shown in Table 204, governs control of the voltage regulator trim bits.

Table 204. Trim Option Bits at Address 0006h (TVREG)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**

Table 204. Trim Option Bits at Address 0006h (TVREG)

Field	Reserved					VREG_TRIM		
Reset	X	X	X	X	X	X	X	X
R/W	R	R	R	R	R	R/W	R/W	R/W
Address	Information Page Memory 0026h							
Note: X=undefined; R/W = read/write.								

Bit	Description
[7:3]	**Reserved** These bits are reserved.
[2:0] VREG_TRIM	**Voltage Regulator Trim**

23.3.1.8. Trim Bit Address 0007h

The Trim Option Bits Register at address `0007h` (see Table 205), is reserved for future use.

Table 205. Trim Option Bits at Address 0007h

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	Reserved							
Reset	X	X	X	X	X	X	X	0
R/W	R	R	R	R	R	R	R	R
Address	Information Page Memory 0027h							
Note: X = undefined; R/W = read/write; R = read-only.								

Bit	Description
[7:0]	**Reserved** These bits are reserved.

23.3.1.9. Trim Bit Addresses 0008h and 0009h

The Trim Option Bits registers at addresses `0008h` and `00009h`, shown in Tables 206 and 207, are reserved for future use.

Table 206. Trim Option Bits at Address 0008h

Bit	7	6	5	4	3	2	1	0
Field	Reserved							
Reset	X	X	X	X	X	X	X	0
R/W	R	R	R	R	R	R	R	R
Address	Information Page Memory 0028h							
Note: X = undefined; R/W = read/write; R = read-only.								

Bit	Description
[7:0]	**Reserved** These bits are reserved.

Table 207. Trim Option Bits at Address 0009h

Bit	7	6	5	4	3	2	1	0
Field	Reserved							
Reset	X	X	X	X	X	X	X	0
R/W	R	R	R	R	R	R	R	R
Address	Information Page Memory 0029h							
Note: X = undefined; R/W = read/write; R = read-only.								

Bit	Description
[7:0]	**Reserved** These bits are reserved.

23.3.1.10.Trim Bit Address 000Ah

All bits in the Trim Option Bits registers at addresses `000Ah` and `0000Bh`, shown in Tables 208 and 209, are reserved for future use.

Table 208. Trim Option Bits at Address 000Ah

Bit	7	6	5	4	3	2	1	0
Field	Reserved							
Reset	X	X	X	X	X	X	X	X
R/W	R	R	R	R	R	R	R	R/W
Address	Information Page Memory 002Ah							
Note: X = undefined; R/W = read/write; R = read-only.								

Bit	Description
[7:0]	**Reserved** All bits are reserved.

23.3.1.11.Trim Bit Address 000Bh

Table 209. Trim Option Bits at Address 000Bh

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	Reserved							
Reset	0	0	0	0	0	0	0	0
R/W	R	R	R	R	R	R	R	R
Address	Information Page Memory 002Bh							
Note: X = undefined; R/W = read/write; R = read-only.								

Bit	Description
[7:0]	**Reserved** These bits are reserved.

23.3.1.12.Trim Bit Address 000Ch

In the Trim Option Bits Register at address `000Ch` and shown in Table 210, govern control of the 1.25V VBIAS Voltage Reference (V_{BIAS}).

Table 210. Trim Option Bits at Address 000Ch (VBIAS)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	Reserved					VBIAS_TRIM		
Reset	X	X	X	X	X	X	X	X
R/W	R	R	R	R	R	R/W	R/W	R/W
Address	Information Page Memory 002Ch							
Note: X = undefined; R/W = read/write; R = read-only.								

Bit	Description
[7:3]	**Reserved** These bits are reserved.
[2:0] VBIAS_TRIM	**VBIAS (1.25V) Reference Trim**

Chapter 24. On-Chip Debugger

The F3224 Series device contains an integrated On-Chip Debugger (OCD) that provides advanced debugging features that include:

- Reading and writing of the Register File
- Reading and writing of Program and Data Memory
- Setting of breakpoints
- Executing CPU instructions

24.1. Architecture

The OCD consists of four primary functional blocks:

- Transmitter
- Receiver
- Auto-baud detector/generator
- Debug controller

Figure 64 shows the architecture of the On-Chip Debugger.

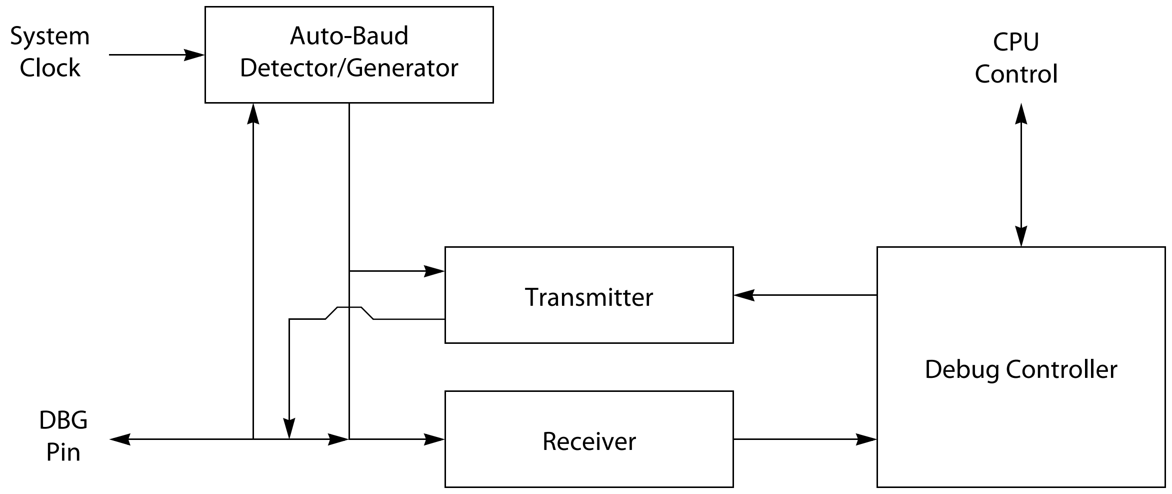


Figure 64. On-Chip Debugger Block Diagram

24.2. Operation

This section describes the development system and OCD interface s well as baud rates, OCD data formats, and breakpoints.

24.2.1. Development System Overview

The development system consists of:

- Host computer running Zilog Developer Studio II (ZDS II), a full-featured Integrated Development Environment (IDE)
- USB SmartCable to interface the computer running ZDS II to the DBG connector on the system under development
- The system under development with a 6-pin DBG connector to access the Zilog microcontroller DBG pin which is used by the On-Chip Debugger (OCD) for communication with an external host

Figure 65 depicts the development system.

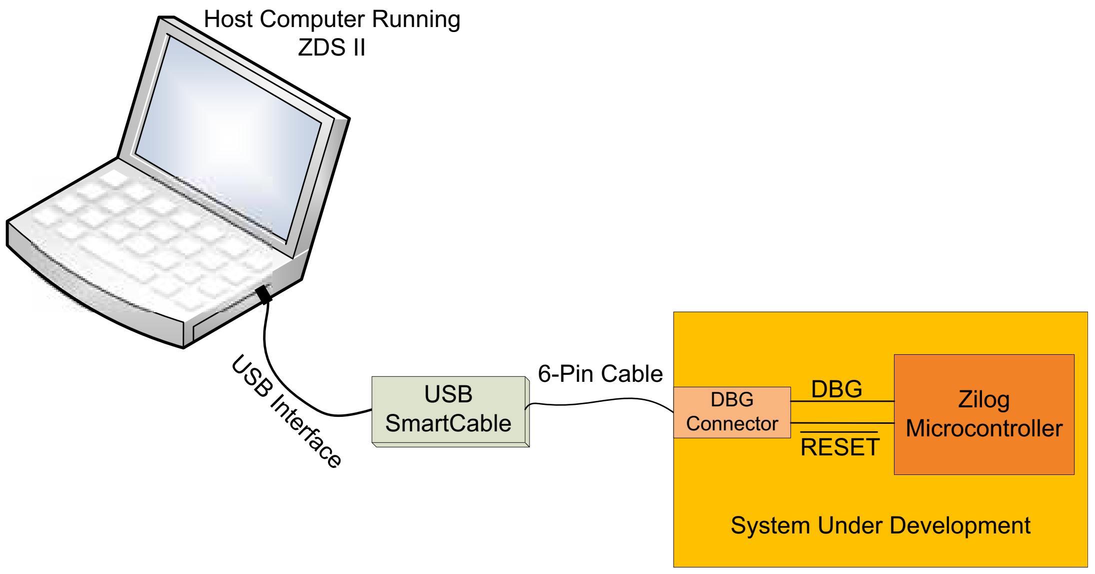


Figure 65. Development System

24.2.2. On-Chip Debugger Interface

The On-Chip Debugger (OCD) uses the DBG pin for communication with an external host. This one-pin interface is a bidirectional open-drain interface that transmits and receives data. Data transmission is half-duplex, in that transmit and receive cannot occur simultaneously. The serial data on the DBG pin is sent using the standard asynchronous data format defined in RS-232. This pin interfaces the F3224 Series device to the serial port of a host PC using minimal external hardware. Figure 66 shows the connections between the debug connector and the Zilog microcontroller.

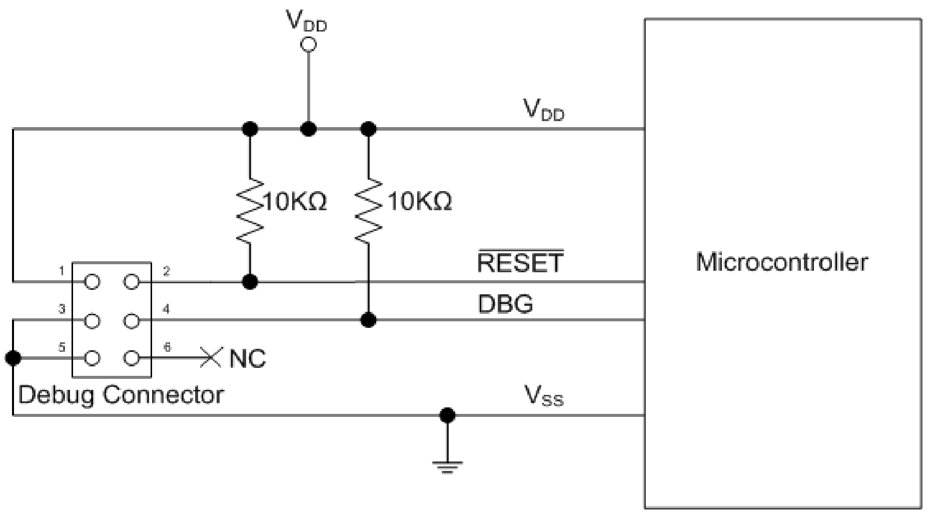


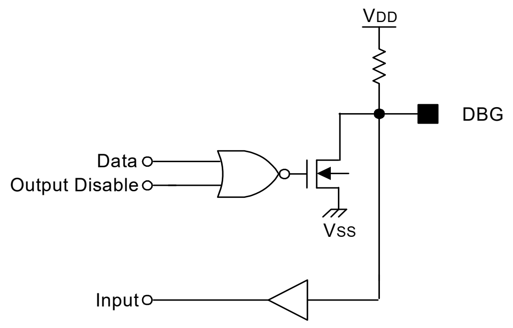
Figure 66. Target OCD Connector Interface

Caution: For proper operation, all power pins (V_{DD} and AV_{DD}) must be supplied with power, and all ground pins (V_{SS} and AV_{SS}) must be properly grounded. The DBG pin should always be connected to V_{DD} through an external pull-up resistor.

The Serial Smart Cable (SSC) does not work with the F3224 device series because it does not fully support the OCD. Use the Opto-Isolated USB, USB, or Ethernet Smart Cables when using in conjunction with ZDSII.

24.2.3. Pin Description

Table 211. Debug Interface pin Description

Pin	Pin Type	Pin Description
DBG	I/O	Debug signal pin 
$\overline{\text{RESET}}$	I (at power on reset)	System reset signal input pin
V_{DD}	-	Power supply input pin. For proper operation, all power pins (V_{DD} and AV_{DD}) must be supplied with power
V_{SS}	-	Ground pin. For proper operation, all ground pins (V_{SS} and AV_{SS}) must be properly grounded

24.2.4. Debug Mode

The operating characteristics of the F3224 Series device in Debug Mode are:

- The CPU fetch unit stops, thereby idling the CPU, unless directed by the OCD to execute specific instructions
- The System Clock operates unless in Stop Mode
- All enabled on-chip peripherals operate unless in Stop Mode or otherwise defined by the on-chip peripheral to disable in Debug Mode
- Automatically exits Halt Mode
- Constantly refreshes the Watchdog Timer, if enabled

24.2.4.1. Entering Debug Mode

The device enters Debug Mode following any of these operations:

- A command to enter Debug Mode is received from the host computer
- CPU execution of a breakpoint (BRK) instruction (when enabled)
- Match of progran counter to a host computer specified value (when enabled)
- After a host computer specified number of System Clocks (when enabled)

24.2.4.2. Exiting Debug Mode

The device exits Debug Mode following any of these operations:

- A command to exit Debug Mode is received from the host computer
- Power-on reset
- Voltage Brown-Out reset
- Asserting the $\overline{\text{RESET}}$ pin Low to initiate a System Reset
- Driving the DBG pin Low when the device is in Stop Mode initiates a System Reset

24.2.5. OCD Data Format

The On-Chip Debugger (OCD) interface uses the asynchronous data format defined for RS-232. Each character is transmitted as 1 start bit, 8 data bits (least-significant bit first), and 1 stop bit; see Figure 67.

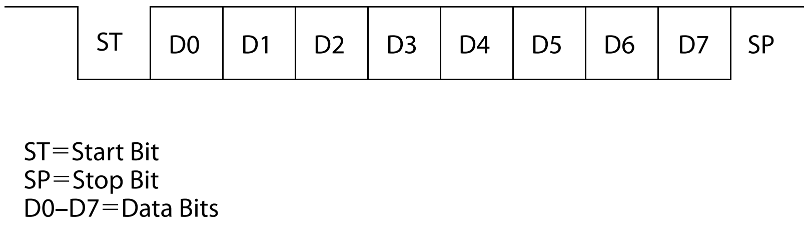


Figure 67. OCD Data Format

24.2.6. OCD Auto-Baud Detector/Generator

To run over a range of baud rates (bits per second) with differing System Clock frequencies, the On-Chip Debugger has an Auto-Baud Detector/Generator. After a reset, the OCD is idle until it receives data. The OCD requires that the first character sent from the host is the character `80h`, which contains eight continuous bits Low (1 start bit plus 7 data bits). The Auto-Baud Detector measures this period and sets the OCD Baud Rate Generator accordingly.

The Auto-Baud Detector/Generator is clocked by the system clock. The minimum baud rate is the system clock frequency divided by 512. If the data can be synchronized with the system clock, the auto-baud generator can run as high as the system clock frequency (1 clock/bit). The maximum recommended baud rate is the system clock frequency divided by 8. Table 212 lists minimum and recommended maximum baud rates for sample System Clock frequencies.

Table 212. OCD Baud-Rate Limits

System Clock Frequency	Maximum Asynchronous Baud Rate (bits/s)	Minimum Asynchronous Baud Rate (bits/s)
20.0MHz	2.5M	39.1k
1.0MHz	125k	1.96k
32kHz	4096	64

24.2.7. Breakpoints

Execution breakpoints are generated using the BRK instruction (op code `00h`). When the CPU decodes a BRK instruction, it signals the On-Chip Debugger. If breakpoints are enabled, the OCD idles the CPU and enters Debug Mode. If breakpoints are not enabled, the OCD ignores the BRK signal and the BRK instruction operates as a NOP instruction.

Chapter 25. eZ8 CPU Instruction Set

This chapter describes assembly language programming and syntax, plus all facets of the eZ8 instruction set.

25.1. Assembly Language Programming Introduction

The eZ8 CPU assembly language provides a means for writing an application program without concern for actual memory addresses or machine instruction formats. A program written in assembly language is called a source program. Assembly language allows the use of symbolic addresses to identify memory locations. It also allows mnemonic codes (op codes and operands) to represent the instructions themselves. The op codes identify the instruction while the operands represent memory locations, registers, or immediate data values.

Each assembly language program consists of a series of symbolic commands called statements. Each statement contains labels, operations, operands and comments.

Labels are assigned to a particular instruction step in a source program. The label identifies that step in the program as an entry point for use by other instructions.

The assembly language also includes assembler directives that supplement the machine instruction. The assembler directives, or *pseudo-ops*, are not translated into a machine instruction. Rather, these pseudo-ops are interpreted as directives that control or assist the assembly process.

A source program is processed (assembled) by the assembler to obtain a machine language program called the *object code*; this object code is executed by the eZ8 CPU. An example segment of an assembly language program is presented in Table 214.

Table 214. Assembly Language Source Program Example

JP START	; Everything after the semicolon is a comment.
START:	; A label called START. The first instruction (JP START) in this ; example causes program execution to jump to the point within the ; program where the START label occurs.
LD R4, R7	; A Load (LD) instruction with two operands. The first operand, ; Working Register R4, is the destination. The second operand, ; Working Register R7, is the source. The contents of R7 is ; written into R4.

Table 214. Assembly Language Source Program Example (Continued)

LD 234h, #%01	; Another Load (LD) instruction with two operands. ; The first operand, Extended Mode Register Address 234h, ; identifies the destination. The second operand, Immediate Data ; value 01h, is the source. The value 01h is written into the ; Register at address 234h.

25.2. Assembly Language Syntax

For proper instruction execution, eZ8 CPU assembly language syntax requires that the operands be written as destination and source. After assembly, the object code usually orders the operands as *source*, *destination*, but ordering is op code-dependent. The two instruction examples that follow illustrate the format of some basic assembly instructions and the resulting object code produced by the assembler. You must follow this binary format if you prefer manual program coding or intend to implement your own assembler.

Example 1. If the contents of registers `43h` and `08h` are added, and the result is stored in `43h`; the assembly syntax and resulting object code is as listed in Table 215.

Table 215. Assembly Language Syntax Example 1

Assembly Language Code	ADD	43h,	08h	(ADD dst, src)
Object Code	04	08	43	(OPC src, dst)

Example 2. In general, when an instruction format requires an 8-bit register address, that address can specify any register location in the range 0–255 or, using Escaped Mode Addressing, a Working Register R0–R15. If the contents of Register `43h` and Working Register R8 are added and the result is stored in `43h`, the assembly syntax and resulting object code is as listed in Table 216.

Table 216. Assembly Language Syntax Example 2

Assembly Language Code	ADD	43h,	R8	(ADD dst, src)
Object Code	04	E8	43	(OPC src, dst)

The register file size varies depending on the device type.

25.3. eZ8 CPU Instruction Notation

In the eZ8 CPU Instruction Summary section on page 387, the operands, condition codes, status flags and address modes are represented by the notational shorthand provided in Table 217.

Table 217. Notational Shorthand

Notation	Description	Operand	Range
b	Bit	b	b represents a value from 0 to 7 (000b to 111b)
cc	Condition Code	–	See the Condition Codes overview in the eZ8 CPU Core User Manual (UM0128)
DA	Direct Address	Addrs	Addrs. represents a number in the range of 0000h to FFFFh
ER	Extended Addressing Register	Reg	Reg. represents a number in the range of 000h to FFFh
IM	Immediate Data	#Data	Data is a number between 00h to FFh
Ir	Indirect Working Register	@Rn	n=0 –15
IR	Indirect Register	@Reg	Reg. represents a number in the range of 00h to FFh
Irr	Indirect Working Register Pair	@RRp	p=0, 2, 4, 6, 8, 10, 12 or 14
IRR	Indirect Register Pair	@Reg	Reg. represents an even number in the range 00h to FEh
p	Polarity	p	Polarity is a single bit binary value of either 0b or 1b.
r	Working Register	Rn	n=0–15
R	Register	Reg	Reg. represents a number in the range of 00h to FFh
RA	Relative Address	X	X represents an index in the range of +127 to –128, which is an offset relative to the address of the next instruction
rr	Working Register Pair	RRp	p=0, 2, 4, 6, 8, 10, 12 or 14
RR	Register Pair	Reg	Reg. represents an even number in the range of 00h to FEh
Vector	Vector Address	Vector	Vector represents a number in the range of 00h to FFh
X	Indexed	#Index	The register or register pair to be indexed is offset by the signed Index value (#Index) in a +127 to –128 range.

Table 218 contains additional symbols that are used throughout the eZ8 CPU Instruction Summary section on page 387.

Table 218. Additional Symbols

Symbol	Definition
dst	Destination Operand
src	Source Operand
@	Indirect Address Prefix
SP	Stack Pointer
PC	Program Counter
FLAGS	Flags Register
RP	Register Pointer
#	Immediate Operand Prefix
B	Binary Number Suffix
%	Hexadecimal Number Prefix
H	Hexadecimal Number Suffix

Assignment of a value is indicated by an arrow. For example, the statement:

dst ← dst + src

indicates that the source data is added to the destination data and the result is stored in the destination location.

25.4. eZ8 CPU Instruction Classes

eZ8 CPU instructions are divided functionally into the following groups:

- Arithmetic
- Bit Manipulation
- Block Transfer
- CPU Control
- Load
- Logical
- Program Control
- Rotate and Shift

Tables 219 through 226 contain the instructions belonging to each group and the number of operands required for each instruction. Some instructions appear in more than one table, because these instructions should be considered as a subset of more than one category. Within these tables, the source operand is identified as *src*, the destination operand is *dst*, and the condition code is *cc*.

Table 219. Arithmetic Instructions

Mnemonic	Operands	Instruction
ADC	dst, src	Add with Carry
ADCX	dst, src	Add with Carry using Extended Addressing
ADD	dst, src	Add
ADDX	dst, src	Add using Extended Addressing
CP	dst, src	Compare
CPC	dst, src	Compare with Carry
CPCX	dst, src	Compare with Carry using Extended Addressing
CPX	dst, src	Compare using Extended Addressing
DA	dst	Decimal Adjust
DEC	dst	Decrement
DECW	dst	Decrement Word
INC	dst	Increment
INCW	dst	Increment Word
MULT	dst	Multiply
SBC	dst, src	Subtract with Carry
SBCX	dst, src	Subtract with Carry using Extended Addressing
SUB	dst, src	Subtract
SUBX	dst, src	Subtract using Extended Addressing

Table 220. Bit Manipulation Instructions

Mnemonic	Operands	Instruction
BCLR	bit, dst	Bit Clear
BIT	p, bit, dst	Bit Set or Clear
BSET	bit, dst	Bit Set
BSWAP	dst	Bit Swap
CCF	–	Complement Carry Flag
RCF	–	Reset Carry Flag
SCF	–	Set Carry Flag
TCM	dst, src	Test Complement Under Mask
TCMX	dst, src	Test Complement Under Mask using Extended Addressing
TM	dst, src	Test Under Mask
TMX	dst, src	Test Under Mask using Extended Addressing

Table 221. Block Transfer Instructions

Mnemonic	Operands	Instruction
LDCI	dst, src	Load Constant to/from Program Memory and Autoincrement Addresses
LDEI	dst, src	Load External Data to/from Data Memory and Autoincrement Addresses

Table 222. CPU Control Instructions

Mnemonic	Operands	Instruction
ATM	–	Atomic Execution
CCF	–	Complement Carry Flag
DI	–	Disable Interrupts
EI	–	Enable Interrupts
HALT	–	Halt Mode
NOP	–	No Operation
RCF	–	Reset Carry Flag
SCF	–	Set Carry Flag
SRP	src	Set Register Pointer

Table 222. CPU Control Instructions (Continued)

Mnemonic	Operands	Instruction
STOP	–	Stop Mode
WDT	–	Watchdog Timer Refresh

Table 223. Load Instructions

Mnemonic	Operands	Instruction
CLR	dst	Clear
LD	dst, src	Load
LDC	dst, src	Load Constant to/from Program Memory
LDCI	dst, src	Load Constant to/from Program Memory and Autoincrement Addresses
LDE	dst, src	Load External Data to/from Data Memory
LDEI	dst, src	Load External Data to/from Data Memory and Autoincrement Addresses
LDWX	dst, src	Load Word using Extended Addressing
LDX	dst, src	Load using Extended Addressing
LEA	dst, X(src)	Load Effective Address
POP	dst	Pop
POPX	dst	Pop using Extended Addressing
PUSH	src	Push
PUSHX	src	Push using Extended Addressing

Table 224. Logical Instructions

Mnemonic	Operands	Instruction
AND	dst, src	Logical AND
ANDX	dst, src	Logical AND using Extended Addressing
COM	dst	Complement
OR	dst, src	Logical OR
ORX	dst, src	Logical OR using Extended Addressing
XOR	dst, src	Logical Exclusive OR
XORX	dst, src	Logical Exclusive OR using Extended Addressing

Table 225. Program Control Instructions

Mnemonic	Operands	Instruction
BRK	–	On-Chip Debugger Break
BTJ	p, bit, src, DA	Bit Test and Jump
BTJNZ	bit, src, DA	Bit Test and Jump if Non-Zero
BTJZ	bit, src, DA	Bit Test and Jump if Zero
CALL	dst	Call Procedure
DJNZ	dst, src, RA	Decrement and Jump Non-Zero
IRET	–	Interrupt Return
JP	dst	Jump
JP cc	dst	Jump Conditional
JR	DA	Jump Relative
JR cc	DA	Jump Relative Conditional
RET	–	Return
TRAP	vector	Software Trap

Table 226. Rotate and Shift Instructions

Mnemonic	Operands	Instruction
BSWAP	dst	Bit Swap
RL	dst	Rotate Left
RLC	dst	Rotate Left through Carry
RR	dst	Rotate Right
RRC	dst	Rotate Right through Carry
SRA	dst	Shift Right Arithmetic
SRL	dst	Shift Right Logical
SWAP	dst	Swap Nibbles

25.5. eZ8 CPU Instruction Summary

Table 227 summarizes the eZ8 CPU instructions. This table identifies the addressing modes employed by the instruction, the effect upon the Flags Register, the number of CPU clock cycles required for the instruction fetch and the number of CPU clock cycles required for the instruction execution.

Table 227. eZ8 CPU Instruction Summary

Assembly Mnemonic	Symbolic Operation	Address Mode		Op Code(s) (Hex)	Flags						Fetch Cycles	Instr. Cycles
		dst	src		C	Z	S	V	D	H		
ADC dst, src	dst ⟵ dst + src + C	r	r	12	*	*	*	*	0	*	2	3
		r	Ir	13							2	4
		R	R	14							3	3
		R	IR	15							3	4
		R	IM	16							3	3
		IR	IM	17							3	4
ADCX dst, src	dst ⟵ dst + src + C	ER	ER	18	*	*	*	*	0	*	4	3
		ER	IM	19							4	3
ADD dst, src	dst ⟵ dst + src	r	r	02	*	*	*	*	0	*	2	3
		r	Ir	03							2	4
		R	R	04							3	3
		R	IR	05							3	4
		R	IM	06							3	3
		IR	IM	07							3	4
ADDX dst, src	dst ⟵ dst + src	ER	ER	08	*	*	*	*	0	*	4	3
		ER	IM	09							4	3

Note: Flags notation:
*=Value is a function of the result of the operation.
–=Unaffected.
X=Undefined.
0=Reset to 0.
1=Set to 1.

Table 227. eZ8 CPU Instruction Summary (Continued)

Assembly Mnemonic	Symbolic Operation	Address Mode dst	Address Mode src	Op Code(s) (Hex)	Flags C	Z	S	V	D	H	Fetch Cycles	Instr. Cycles
AND dst, src	dst ← dst AND src	r	r	52	–	*	*	0	–	–	2	3
		r	Ir	53							2	4
		R	R	54							3	3
		R	IR	55							3	4
		R	IM	56							3	3
		IR	IM	57							3	4
ANDX dst, src	dst ← dst AND src	ER	ER	58	–	*	*	0	–	–	4	3
		ER	IM	59							4	3
ATM	Block all interrupt and DMA requests during execution of the next 3 instructions			2F	–	–	–	–	–	–	1	2
BCLR bit, dst	dst[bit] ← 0	r		E2	–	*	*	0	–	–	2	2
BIT p, bit, dst	dst[bit] ← p	r		E2	–	*	*	0	–	–	2	2
BRK	Debugger Break			00	–	–	–	–	–	–	1	1
BSET bit, dst	dst[bit] ← 1	r		E2	–	*	*	0	–	–	2	2
BSWAP dst	dst[7:0] ← dst[0:7]	R		D5	X	*	*	0	–	–	2	2
BTJ p, bit, src, dst	if src[bit]=p PC ← PC + X		r	F6	–	–	–	–	–	–	3	3
			Ir	F7							3	4
BTJNZ bit, src, dst	if src[bit]=1 PC ← PC + X		r	F6	–	–	–	–	–	–	3	3
			Ir	F7							3	4
BTJZ bit, src, dst	if src[bit]=0 PC ← PC + X		r	F6	–	–	–	–	–	–	3	3
			Ir	F7							3	4
CALL dst	SP ← SP -2 @SP ← PC PC ← dst	IRR		D4	–	–	–	–	–	–	2	6
		DA		D6							3	3
CCF	C ← ~C			EF	*	–	–	–	–	–-	1	2

Note: Flags notation:
*=Value is a function of the result of the operation.
–=Unaffected.
X=Undefined.
0=Reset to 0.
1=Set to 1.

Table 227. eZ8 CPU Instruction Summary (Continued)

Assembly Mnemonic	Symbolic Operation	Address Mode dst	Address Mode src	Op Code(s) (Hex)	Flags C	Z	S	V	D	H	Fetch Cycles	Instr. Cycles
CLR dst	dst ⟵ 00h	R		B0	–	–	–	–	–	–	2	2
		IR		B1							2	3
COM dst	dst ⟵ ~dst	R		60	–	*	*	0	–	–	2	2
		IR		61							2	3
CP dst, src	dst – src	r	r	A2	*	*	*	*	–	–	2	3
		r	Ir	A3							2	4
		R	R	A4							3	3
		R	IR	A5							3	4
		R	IM	A6							3	3
		IR	IM	A7							3	4
CPC dst, src	dst – src – C	r	r	1F A2	*	*	*	*	–	–	3	3
		r	Ir	1F A3							3	4
		R	R	1F A4							4	3
		R	IR	1F A5							4	4
		R	IM	1F A6							4	3
		IR	IM	1F A7							4	4
CPCX dst, src	dst – src – C	ER	ER	1F A8	*	*	*	*	–	–	5	3
		ER	IM	1F A9							5	3
CPX dst, src	dst – src	ER	ER	A8	*	*	*	*	–	–	4	3
		ER	IM	A9							4	3
DA dst	dst ⟵ DA(dst)	R		40	*	*	*	X	–	–	2	2
		IR		41							2	3
DEC dst	dst ⟵ dst – 1	R		30	–	*	*	*	–	–	2	2
		IR		31							2	3
DECW dst	dst ⟵ dst – 1	RR		80	–	*	*	*	–	–	2	5
		IRR		81							2	6
DI	IRQCTL[7] ← 0			8F	–	–	–	–	–	–	1	2

Note: Flags notation:
*=Value is a function of the result of the operation.
–=Unaffected.
X=Undefined.
0=Reset to 0.
1=Set to 1.

Table 227. eZ8 CPU Instruction Summary (Continued)

Assembly Mnemonic	Symbolic Operation	Address Mode		Op Code(s) (Hex)	Flags						Fetch Cycles	Instr. Cycles
		dst	src		C	Z	S	V	D	H		
DJNZ dst, RA	dst ⟵ dst – 1 if dst ≠ 0 PC ⟵ PC + X	r		0A–FA	–	–	–	–	–	–	2	3
EI	IRQCTL[7] ⟵ 1			9F	–	–	–	–	–	–	1	2
HALT	Halt Mode			7F	–	–	–	–	–	–	1	2
INC dst	dst ⟵ dst + 1	R		20	–	*	*	–	–	–	2	2
		IR		21							2	3
		r		0E–FE							1	2
INCW dst	dst ⟵ dst + 1	RR		A0	–	*	*	*	–	–	2	5
		IRR		A1							2	6
IRET	FLAGS ⟵ @SP SP ⟵ SP + 1 PC ⟵ @SP SP ⟵ SP + 2 IRQCTL[7] ⟵ 1			BF	*	*	*	*	*	*	1	5
JP dst	PC ⟵ dst	DA		8D	–	–	–	–	–	–	3	2
		IRR		C4							2	3
JP cc, dst	if cc is true, PC ⟵ dst	DA		0D–FD	–	–	–	–	–	–	3	2
JR dst	PC ⟵ PC + X	DA		8B	–	–	–	–	–	–	2	2
JR cc, dst	if cc is true, PC ⟵ PC + X	DA		0B–FB	–	–	–	–	–	–	2	2

Note: Flags notation:
*=Value is a function of the result of the operation.
–=Unaffected.
X=Undefined.
0=Reset to 0.
1=Set to 1.

Table 227. eZ8 CPU Instruction Summary (Continued)

Assembly Mnemonic	Symbolic Operation	Address Mode dst	Address Mode src	Op Code(s) (Hex)	Flags C	Z	S	V	D	H	Fetch Cycles	Instr. Cycles
LD dst, rc	dst ⟵ src	r	IM	0C–FC	–	–	–	–	–	–	2	2
		r	X(r)	C7							3	3
		X(r)	r	D7							3	4
		r	Ir	E3							2	3
		R	R	E4							3	2
		R	IR	E5							3	4
		R	IM	E6							3	2
		IR	IM	E7							3	3
		Ir	r	F3							2	3
		IR	R	F5							3	3
LDC dst, src	dst ⟵ src	r	Irr	C2	–	–	–	–	–	–	2	5
		Ir	Irr	C5							2	9
		Irr	r	D2							2	5
LDCI dst, src	dst ⟵ src r ⟵ r + 1 rr ⟵ rr + 1	Ir	Irr	C3	–	–	–	–	–	–	2	9
		Irr	Ir	D3							2	9
LDE dst, src	dst ⟵ src	r	Irr	82	–	–	–	–	–	–	2	5
		Irr	r	92							2	5
LDEI dst, src	dst ⟵ src r ⟵ r + 1 rr ⟵ rr + 1	Ir	Irr	83	–	–	–	–	–	–	2	9
		Irr	Ir	93							2	9
LDWX dst, src	dst ⟵ src	ER	ER	1FE8	–	–	–	–	–	–	5	4

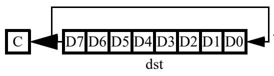
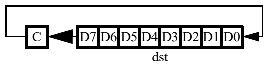
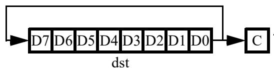
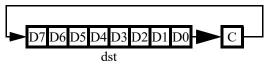
Note: Flags notation:
*=Value is a function of the result of the operation.
–=Unaffected.
X=Undefined.
0=Reset to 0.
1=Set to 1.

Table 227. eZ8 CPU Instruction Summary (Continued)

Assembly Mnemonic	Symbolic Operation	Address Mode dst	Address Mode src	Op Code(s) (Hex)	Flags C	Z	S	V	D	H	Fetch Cycles	Instr. Cycles
LDX dst, src	dst ⟵ src	r	ER	84	–	–	–	–	–	–	3	2
		Ir	ER	85							3	3
		R	IRR	86							3	4
		IR	IRR	87							3	5
		r	X(rr)	88							3	4
		X(rr)	r	89							3	4
		ER	r	94							3	2
		ER	Ir	95							3	3
		IRR	R	96							3	4
		IRR	IR	97							3	5
		ER	ER	E8							4	2
		ER	IM	E9							4	2
LEA dst, X(src)	dst ⟵ src + X	r	X(r)	98	–	–	–	–	–	–	3	3
		rr	X(rr)	99							3	5
MULT dst	dst[15:0] ⟵ dst[15:8] * dst[7:0]	RR		F4	–	–	–	–	–	–	2	8
NOP	No operation			0F	–	–	–	–	–	–	1	2
OR dst, src	dst ⟵ dst OR src	r	r	42	–	*	*	0	–	–	2	3
		r	Ir	43							2	4
		R	R	44							3	3
		R	IR	45							3	4
		R	IM	46							3	3
		IR	IM	47							3	4
ORX dst, src	dst ⟵ dst OR src	ER	ER	48	–	*	*	0	–	–	4	3
		ER	IM	49							4	3
POP dst	dst ⟵ @SP SP ⟵ SP + 1	R		50	–	–	–	–	–	–	2	2
		IR		51							2	3

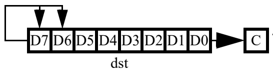
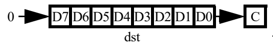
Note: Flags notation:
*=Value is a function of the result of the operation.
–=Unaffected.
X=Undefined.
0=Reset to 0.
1=Set to 1.

Table 227. eZ8 CPU Instruction Summary (Continued)

Assembly Mnemonic	Symbolic Operation	Address Mode dst	Address Mode src	Op Code(s) (Hex)	Flags C	Z	S	V	D	H	Fetch Cycles	Instr. Cycles
POPX dst	dst ⟵ @SP SP ⟵ SP + 1	ER		D8	–	–	–	–	–	–	3	2
PUSH src	SP ⟵ SP – 1 @SP ⟵ src	R		70	–	–	–	–	–	–	2	2
		IR		71							2	3
		IM		IF70							3	2
PUSHX src	SP ⟵ SP – 1 @SP ⟵ src	ER		C8	–	–	–	–	–	–	3	2
RCF	C ⟵ 0			CF	0	–	–	–	–	–	1	2
RET	PC ⟵ @SP SP ⟵ SP + 2			AF	–	–	–	–	–	–	1	4
RL dst		R		90	*	*	*	*	–	–	2	2
		IR		91							2	3
RLC dst		R		10	*	*	*	*	–	–	2	2
		IR		11							2	3
RR dst		R		E0	*	*	*	*	–	–	2	2
		IR		E1							2	3
RRC dst		R		C0	*	*	*	*	–	–	2	2
		IR		C1							2	3
SBC dst, src	dst ⟵ dst – src – C	r	r	32	*	*	*	*	1	*	2	3
		r	Ir	33							2	4
		R	R	34							3	3
		R	IR	35							3	4
		R	IM	36							3	3
		IR	IM	37							3	4
SBCX dst, src	dst ⟵ dst – src – C	ER	ER	38	*	*	*	*	1	*	4	3
		ER	IM	39							4	3
SCF	C ⟵ 1			DF	1	–	–	–	–	–	1	2

Note: Flags notation:
*=Value is a function of the result of the operation.
–=Unaffected.
X=Undefined.
0=Reset to 0.
1=Set to 1.

Table 227. eZ8 CPU Instruction Summary (Continued)

Assembly Mnemonic	Symbolic Operation	Address Mode dst	Address Mode src	Op Code(s) (Hex)	Flags C	Z	S	V	D	H	Fetch Cycles	Instr. Cycles
SRA dst		R		D0	*	*	*	0	–	–	2	2
		IR		D1							2	3
SRL dst		R		1F C0	*	*	0	*	–	–	3	2
		IR		1F C1							3	3
SRP src	RP ← src		IM	01	–	–	–	–	–	–	2	2
STOP	Stop Mode			6F	–	–	–	–	–	–	1	2
SUB dst, src	dst ← dst – src	r	r	22	*	*	*	*	1	*	2	3
		r	Ir	23							2	4
		R	R	24							3	3
		R	IR	25							3	4
		R	IM	26							3	3
		IR	IM	27							3	4
SUBX dst, src	dst ← dst – src	ER	ER	28	*	*	*	*	1	*	4	3
		ER	IM	29							4	3
SWAP dst	dst[7:4] ↔ dst[3:0]	R		F0	X	*	*	X	–	–	2	2
		IR		F1							2	3
TCM dst, src	(NOT dst) AND src	r	r	62	–	*	*	0	–	–	2	3
		r	Ir	63							2	4
		R	R	64							3	3
		R	IR	65							3	4
		R	IM	66							3	3
		IR	IM	67							3	4
TCMX dst, src	(NOT dst) AND src	ER	ER	68	–	*	*	0	–	–	4	3
		ER	IM	69							4	3

Note: Flags notation:
*=Value is a function of the result of the operation.
–=Unaffected.
X=Undefined.
0=Reset to 0.
1=Set to 1.

Table 227. eZ8 CPU Instruction Summary (Continued)

Assembly Mnemonic	Symbolic Operation	Address Mode dst	Address Mode src	Op Code(s) (Hex)	C	Z	S	V	D	H	Fetch Cycles	Instr. Cycles
TM dst, src	dst AND src	r	r	72	–	*	*	0	–	–	2	3
		r	Ir	73							2	4
		R	R	74							3	3
		R	IR	75							3	4
		R	IM	76							3	3
		IR	IM	77							3	4
TMX dst, src	dst AND src	ER	ER	78	–	*	*	0	–	–	4	3
		ER	IM	79							4	3
TRAP Vector	SP ⟵ SP – 2 @SP ⟵ PC SP ⟵ SP – 1 @SP ⟵ FLAGS PC ⟵ @Vector		Vector	F2	–	–	–	–	–	–	2	6
WDT				5F	–	–	–	–	–	–	1	2
XOR dst, src	dst ⟵ dst XOR src	r	r	B2	–	*	*	0	–	–	2	3
		r	Ir	B3							2	4
		R	R	B4							3	3
		R	IR	B5							3	4
		R	IM	B6							3	3
		IR	IM	B7							3	4
XORX dst, src	dst ⟵ dst XOR src	ER	ER	B8	–	*	*	0	–	–	4	3
		ER	IM	B9							4	3

Note: Flags notation:
*=Value is a function of the result of the operation.
–=Unaffected.
X=Undefined.
0=Reset to 0.
1=Set to 1.

Chapter 26. Op Code Maps

Figure 68 shows a description of the op code map data and the abbreviations. Table 228 lists Op Code Map abbreviations.

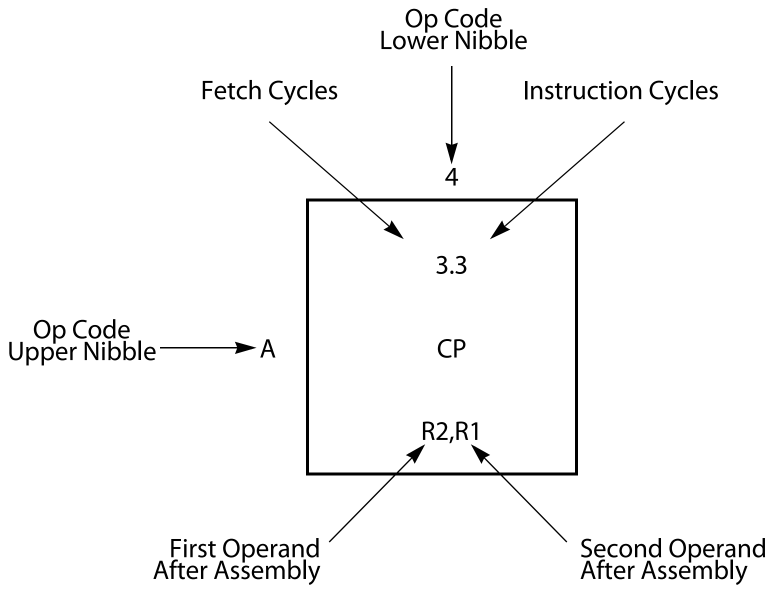


Figure 68. Op Code Map Cell Description

Table 228. Op Code Map Abbreviations

Abbreviation	Description	Abbreviation	Description
b	Bit position	IRR	Indirect Register Pair
cc	Condition code	p	Polarity (0 or 1)
X	8-bit signed index or displacement	r	4-bit Working Register
DA	Destination address	R	8-bit register
ER	Extended Addressing register	r1, R1, Ir1, Irr1, IR1, rr1, RR1, IRR1, ER1	Destination address
IM	Immediate data value	r2, R2, Ir2, Irr2, IR2, rr2, RR2, IRR2, ER2	Source address
Ir	Indirect Working Register	RA	Relative
IR	Indirect register	rr	Working Register Pair
Irr	Indirect Working Register Pair	RR	Register Pair

Figures 69 and 70 provide information about each of the eZ8 CPUinstructions.

Upper Nibble (Hex)	Lower Nibble (Hex)															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	1.1 **BRK**	2.2 **SRP** IM	2.3 **ADD** r1,r2	2.4 **ADD** r1,Ir2	3.3 **ADD** R2,R1	3.4 **ADD** IR2,R1	3.3 **ADD** R1,IM	3.4 **ADD** IR1,IM	4.3 **ADDX** ER2,ER1	4.3 **ADDX** IM,ER1	2.3 **DJNZ** r1,X	2.2 **JR** cc,X	2.2 **LD** r1,IM	3.2 **JP** cc,DA	1.2 **INC** r1	1.2 **NOP**
1	2.2 **RLC** R1	2.3 **RLC** IR1	2.3 **ADC** r1,r2	2.4 **ADC** r1,Ir2	3.3 **ADC** R2,R1	3.4 **ADC** IR2,R1	3.3 **ADC** R1,IM	3.4 **ADC** IR1,IM	4.3 **ADCX** ER2,ER1	4.3 **ADCX** IM,ER1	↓	↓	↓	↓	↓	See 2nd Op Code Map
2	2.2 **INC** R1	2.3 **INC** IR1	2.3 **SUB** r1,r2	2.4 **SUB** r1,Ir2	3.3 **SUB** R2,R1	3.4 **SUB** IR2,R1	3.3 **SUB** R1,IM	3.4 **SUB** IR1,IM	4.3 **SUBX** ER2,ER1	4.3 **SUBX** IM,ER1	↓	↓	↓	↓	↓	1, 2 **ATM**
3	2.2 **DEC** R1	2.3 **DEC** IR1	2.3 **SBC** r1,r2	2.4 **SBC** r1,Ir2	3.3 **SBC** R2,R1	3.4 **SBC** IR2,R1	3.3 **SBC** R1,IM	3.4 **SBC** IR1,IM	4.3 **SBCX** ER2,ER1	4.3 **SBCX** IM,ER1	↓	↓	↓	↓	↓	
4	2.2 **DA** R1	2.3 **DA** IR1	2.3 **OR** r1,r2	2.4 **OR** r1,Ir2	3.3 **OR** R2,R1	3.4 **OR** IR2,R1	3.3 **OR** R1,IM	3.4 **OR** IR1,IM	4.3 **ORX** ER2,ER1	4.3 **ORX** IM,ER1	↓	↓	↓	↓	↓	
5	2.2 **POP** R1	2.3 **POP** IR1	2.3 **AND** r1,r2	2.4 **AND** r1,Ir2	3.3 **AND** R2,R1	3.4 **AND** IR2,R1	3.3 **AND** R1,IM	3.4 **AND** IR1,IM	4.3 **ANDX** ER2,ER1	4.3 **ANDX** IM,ER1	↓	↓	↓	↓	↓	1.2 **WDT**
6	2.2 **COM** R1	2.3 **COM** IR1	2.3 **TCM** r1,r2	2.4 **TCM** r1,Ir2	3.3 **TCM** R2,R1	3.4 **TCM** IR2,R1	3.3 **TCM** R1,IM	3.4 **TCM** IR1,IM	4.3 **TCMX** ER2,ER1	4.3 **TCMX** IM,ER1	↓	↓	↓	↓	↓	1.2 **STOP**
7	2.2 **PUSH** R2	2.3 **PUSH** IR2	2.3 **TM** r1,r2	2.4 **TM** r1,Ir2	3.3 **TM** R2,R1	3.4 **TM** IR2,R1	3.3 **TM** R1,IM	3.4 **TM** IR1,IM	4.3 **TMX** ER2,ER1	4.3 **TMX** IM,ER1	↓	↓	↓	↓	↓	1.2 **HALT**
8	2.5 **DECW** RR1	2.6 **DECW** IRR1	2.5 **LDE** r1,Irr2	2.9 **LDEI** Ir1,Irr2	3.2 **LDX** r1,ER2	3.3 **LDX** Ir1,ER2	3.4 **LDX** IRR2,R1	3.5 **LDX** IRR2,IR1	3.4 **LDX** r1,rr2,X	3.4 **LDX** rr1,r2,X	↓	↓	↓	↓	↓	1.2 **DI**
9	2.2 **RL** R1	2.3 **RL** IR1	2.5 **LDE** r2,Irr1	2.9 **LDEI** Ir2,Irr1	3.2 **LDX** r2,ER1	3.3 **LDX** Ir2,ER1	3.4 **LDX** R2,IRR1	3.5 **LDX** IR2,IRR1	3.3 **LEA** r1,r2,X	3.5 **LEA** rr1,rr2,X	↓	↓	↓	↓	↓	1.2 **EI**
A	2.5 **INCW** RR1	2.6 **INCW** IRR1	2.3 **CP** r1,r2	2.4 **CP** r1,Ir2	3.3 **CP** R2,R1	3.4 **CP** IR2,R1	3.3 **CP** R1,IM	3.4 **CP** IR1,IM	4.3 **CPX** ER2,ER1	4.3 **CPX** IM,ER1	↓	↓	↓	↓	↓	1.4 **RET**
B	2.2 **CLR** R1	2.3 **CLR** IR1	2.3 **XOR** r1,r2	2.4 **XOR** r1,Ir2	3.3 **XOR** R2,R1	3.4 **XOR** IR2,R1	3.3 **XOR** R1,IM	3.4 **XOR** IR1,IM	4.3 **XORX** ER2,ER1	4.3 **XORX** IM,ER1	↓	↓	↓	↓	↓	1.5 **IRET**
C	2.2 **RRC** R1	2.3 **RRC** IR1	2.5 **LDC** r1,Irr2	2.9 **LDCI** Ir1,Irr2	2.3 **JP** IRR1	2.9 **LDC** Ir1,Irr2		3.4 **LD** r1,r2,X	3.2 **PUSHX** ER2		↓	↓	↓	↓	↓	1.2 **RCF**
D	2.2 **SRA** R1	2.3 **SRA** IR1	2.5 **LDC** r2,Irr1	2.9 **LDCI** Ir2,Irr1	2.6 **CALL** IRR1	2.2 **BSWAP** R1	3.3 **CALL** DA	3.4 **LD** r2,r1,X	3.2 **POPX** ER1		↓	↓	↓	↓	↓	1.2 **SCF**
E	2.2 **RR** R1	2.3 **RR** IR1	2.2 **BIT** p,b,r1	2.3 **LD** r1,Ir2	3.2 **LD** R2,R1	3.3 **LD** IR2,R1	3.2 **LD** R1,IM	3.3 **LD** IR1,IM	4.2 **LDX** ER2,ER1	4.2 **LDX** IM,ER1	↓	↓	↓	↓	↓	1.2 **CCF**
F	2.2 **SWAP** R1	2.3 **SWAP** IR1	2.6 **TRAP** Vector	2.3 **LD** Ir1,r2	2.8 **MULT** RR1	3.3 **LD** R2,IR1	3.3 **BTJ** p,b,r1,X	3.4 **BTJ** p,b,Ir1,X			↓	↓	↓	↓	↓	

Figure 69. First Op Code Map

	Lower Nibble (Hex)															
Upper Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0																
1																
2																
3																
4																
5																
6																
7	3, 2 **PUSH** IM															
8																
9																
A			3.3 **CPC** r1,r2	3.4 **CPC** r1,Ir2	4.3 **CPC** R2,R1	4.4 **CPC** IR2,R1	4.3 **CPC** R1,IM	4.4 **CPC** IR1,IM	5.3 **CPCX** ER2,ER1	5.3 **CPCX** IM,ER1						
B																
C	3.2 **SRL** R1	3.3 **SRL** IR1														
D																
E									5, 4 **LDWX** ER2,ER1							
F																

Figure 70. Second Op Code Map after 1Fh

Chapter 27. Electrical Characteristics

The data in this chapter has been tabulated prior to qualification (i.e., prequalification) and precharacterization of the F3224 Series product, and is subject to change. Additional electrical characteristics can be found in the individual chapters of this document.

27.1. Absolute Maximum Ratings

Stresses greater than those listed in Table 229 can cause permanent damage to the device. These ratings are stress ratings only. Operation of the device at any condition outside those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods can affect device reliability. For improved reliability, tie unused inputs to one of the supply voltages (V_{DD} or V_{SS}).

Table 229. Absolute Maximum Ratings

Parameter	Min	Max	Units
Ambient temperature under bias	–40	+125	°C
Storage temperature	–65	+150	°C
Supply voltage on V_{DD}	–0.3	+4.0	V
Voltage on any pin (except V_{CORE})	–0.3	+4.0	V
Maximum injection current on input and/or inactive output pin	–250	+250	µA
Maximum output current from active output pin	–25	+25	mA
Maximum current into V_{DD} or out of V_{SS}			
32-pin QFN		80	mA
44-Pin QFN		160	mA

27.2. DC Characteristics

Table 230 lists the DC characteristics of the F3224 Series products. All voltages are referenced to V_{SS}, which is the primary system ground.

Table 230. DC Characteristics

		T_A=–40°C to +85°C				
Symbol	**Parameter**	**Min**	**Typical**[1]	**Max**	**Units**	**Conditions**
V_{DD}	Supply Voltage	1.8		3.6	V	
V_{IL1}	Low Level Input Voltage	–0.3		0.3*V_{DD}	V	For all input pins except $X2_{IN}$
V_{IL2}	Low Level Input Voltage	–0.3		–	V	For $X2_{IN}$
V_{IH1}	High Level Input Voltage	0.7*V_{DD}		V_{DD}+0.3	V	All input pins
V_{OL1}	Low Level Output Voltage			0.5	V	I_{OL}=4 mA; V_{DD}=3.3V High Output Drive disabled.
V_{OH1}	High Level Output Voltage	V_{DD}–0.8			V	I_{OH}=–4 mA; V_{DD}=3.3V High Output Drive disabled.
V_{OL2}	Low Level Output Voltage	–		0.5	V	I_{OL}=8 mA; V_{DD}=3.3V High Output Drive enabled.
V_{OH2}	High Level Output Voltage	V_{DD}–0.8			V	I_{OH}=–8 mA; V_{DD}=3.3V High Output Drive enabled.
I_{IL}	Input Leakage Current	–50		+50	nA	V_{DD}=3.6V; V_{IN}=V_{DD} or V_{SS}[2]
I_{TL}	Tristate Leakage Current	–50		+50	nA	V_{DD}=3.6V
C_{PAD}	GPIO Port Pad Capacitance		≤8.0		pF	
C_{X2IN}	X_{IN} Pad Capacitance		≤8.0		pF	
C_{X2OUT}	X_{OUT} Pad Capacitance		≤9.5		pF	
R_{PU}	Weak Pull-up Resistance	22	35	50	kΩ	

Notes:
1. These values are provided for design guidance only and are not tested in production.
2. This condition excludes all pins that have on-chip pull-ups, when driven Low.

The currents in Table 231 represent the power consumption without any peripherals active (unless otherwise noted). For design guidance, total power consumption will be the sum of all active peripheral currents plus the appropriate current characteristics shown below.

Table 231. Supply Current Characteristics

		T_A=–40°C to +85°C				
Symbol	**Parameter**	**Min**	**Typical**[1]	**Max**	**Units**	**Conditions**[2]
I_{DDA}	Active Mode Device Current		2.0		mA	20MHz[3,4,5,6]
			0.85		mA	8 MHz[3,4,5,6]
			0.16		mA	1 MHz[3,4,5,6]
			0.015		mA	32kHz[3,4,5,6]
I_{DDH}	Halt Mode Device Current		1.0		mA	20MHz[3,4,5]
			0.41		mA	8MHz[3,4,5,6]
			0.1		mA	1MHz[3,4,5,6]
			0.012		mA	32kHz[3,4,5,6]
I_{DDS1}	Stop Mode Device Current with FRECOV = 1		3.4		µA	All peripherals disabled including VBO and WDT[3,4,6]
I_{DDS2}	Stop Mode Device Current with FRECOV = 0		0.7		µA	All peripherals disabled including VBO and WDT[3,4,6]

Notes:
1. These values are provided for design guidance only and are not tested in production.
2. Typical conditions are defined as 3.0V at 25°C, unless otherwise noted.
3. All internal pull ups are disabled and all push-pull outputs are unloaded.
4. All open-drain outputs are pulled up to V_{DD}/AV_{DD} and are at a High state.
5. System clock source is an external square wave clock signal driven through the CLKIN pin.
6. All inputs are at V_{DD}/AV_{DD} or V_{SS}/AV_{SS} as appropriate.

27.3. AC Characteristics

Table 232 lists the AC characteristics and timing of the F3224 Series products. All AC timing information assumes a standard load of 50pF on all outputs.

Table 232. AC Characteristics

		V_{DD}=1.8 to 3.6V T_A=–40°C to +85°C				
Symbol	Parameter	Min	Typ	Max	Units	Conditions
F_{Sysclk}	System Clock Frequency			20	MHz	See Figure 71
T_{XIN}	CLKIN Period	50			ns	T_{CLK}=1/F_{Sysclk}
T_{XINH}	CLKIN High Time	20		30	ns	TCLK=50 ns
T_{XINL}	CLKIN Low Time	20		30	ns	TCLK=50 ns
T_{XINR}	CLKIN Rise Time			3	ns	TCLK=50 ns
T_{XINF}	CLKIN Fall Time			3	ns	TCLK=50 ns
T_{XIN2}	CLK2IN Period		30.5		µs	T_{CLK}=1/F_{PCLK}
F_{PCLK}	Peripheral Clock Frequency		32.768		kHz	

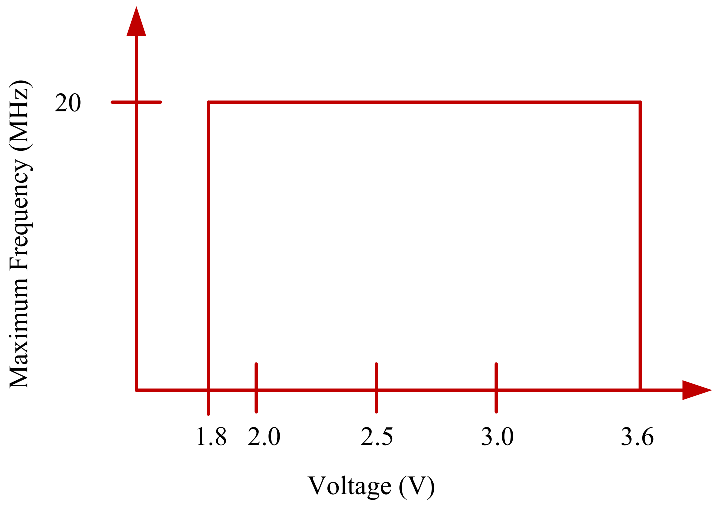


Figure 71. Maximum System Clock Frequency vs. V_{DD}

27.4. On-Chip Peripheral AC and DC Electrical Characteristics

On-chip peripheral AC and DC electrical characteristics are listed in Tables 233 through 240.

27.4.1. Power-On Reset

Table 233 presents electrical and timing data for the F3224 Series' Power-On Reset function.

Table 233. Power-On Reset Electrical Characteristics and Timing

		T_A=–40°C to +85°C				
Symbol	**Parameter**	**Min**	**Typ**[1]	**Max**	**Units**	**Conditions**
$I_{DD}POR$	POR Active Current		<0.01		µA	
V_{POR}	Power-On Reset Voltage Threshold	1.30	1.55	1.75	V	$V_{DD}=V_{POR}$
V_{TH}	POR Start Voltage		0.4		V	
T_{RAMP}	V_{DD} Ramp Up Time	0.01		1000	ms	

Note: 1. Data in the Typical column is from characterization at 3.0V and 25°C. These values are provided for design guidance only and are not tested in production.

27.4.2. Voltage Brown-Out

Table 234 presents electrical and timing data for the F3224 Series' Voltage Brown-Out function.

Table 234. Voltage Brown-Out Electrical Characteristics and Timing

		T_A=–40°C to +85°C				
Symbol	**Parameter**	**Min**	**Typ**[1]	**Max**	**Units**	**Conditions**
I_{DD}VBO	VBO Active Current		2		µA	Active and Halt modes
			0.07		µA	Stop Mode
V_{VBO}	Voltage Brown-Out Reset Voltage Threshold	1.8	1.85	1.9	V	V_{DD}=V_{VBO}
V_{HYS}	Hysteresis of V_{VBO}		100		mV	
T_{VBOPW}	Voltage Brown-Out Pulse Width Rejection		1.05		µs	
T_{RESET}	Reset Delay		10		ms	$V_{DD} > V_{POR}$, $V_{DD} > V_{VBO}$

Note: 1. Data in the Typical column is from characterization at 3.0V and 25°C. These values are provided for design guidance only and are not tested in production.

27.4.3. Stop-Mode Recovery

Table 235 presents electrical and timing data for the F3224 Series' Stop-Mode Recovery function.

Table 235. Stop-Mode Recovery (SMR) Timing

		V_{DD}=1.8V to 3.6V T_A=–40°C to +85°C				
Symbol	**Parameter**	**Min**	**Typ**[1]	**Max**	**Units**	**Conditions**
T_{SMRL}	Stop-Mode Recovery Latency		50 Sysclks + T_{SMRD}			FRECOV=0
			50 Sysclks			FRECOV=1
T_{SMRD}	Stop-Mode Recovery Delay		300		µs	

Note: 1. Data in the Typical column is from characterization at 3.0V and 25°C. These values are provided for design guidance only and are not tested in production.

27.4.4. Flash Memory

Table 236 presents electrical and timing data for the F3224 Series' Flash Memory function.

Table 236. Flash Memory Electrical Characteristics and Timing

	V_{DD}=1.8V to 3.6V T_A=–40°C to +85°C				
Parameter	**Min**	**Typ**	**Max**	**Units**	**Conditions**
Program Current			3	mA	
Erase Current			2	mA	
DCOCLK Frequency for Program or Erasure of Flash Memory	1		20	MHz	FLL locked
Flash Byte Program Time		55		µs	
Flash Page Erase Time	4		6	ms	
Flash Mass Erase Time	20		41	ms	
Data Retention	100			years	25°C
Endurance	10,000			cycles	Program/erase cycles

27.4.5. Watchdog Timer

Table 237 presents electrical and timing data for the F3224 Series' Watchdog Timer function.

Table 237. Watchdog Timer Electrical Characteristics and Timing

		V_{DD}=1.8V to 3.6V TA=–40°C to +85°C				
Symbol	**Parameter**	**Min**	**Typ*[1]**	**Max**	**Unit**	**Conditions**
I_{DD}WDT	WDT Active Current		0.15	–	µA	
F_{WDT}	WDT Oscillator Frequency	-5%	8.2	+5%	kHz	

Note: 1. Data in the Typical column is from characterization at 3.0V and 25°C. These values are provided for design guidance only and are not tested in production.

27.4.6. Reference System

Table 238 presents electrical and timing data for the F3224 Series' Reference System.

Table 238. Reference System Electrical Characteristics

		V_{DD}=1.8V to 3.6V T_A=–40°C to +85°C				
Symbol	**Parameter**	**Min**	**Typ[1]**	**Max**	**Units**	**Conditions**
V_{PREF}	Programmable Internal Reference Voltage Range	VBIAS/ 32		VBIAS	V	PREFSRC=0
		V_{DD}/32		V_{DD}	V	PREFSRC=1
$V_{PREFTOL}$	Programmable Internal Reference Voltage Tolerance	Larger of –1.5% or –5mV	V_{PREF}[2]	Larger of 1.5% or 5mV		PREFSRC=0 (VBIAS is source)
		Larger of –0.5% or –5mV	V_{PREF}[2]	Larger of 0.5% or 5mV		PREFSRC=1 (AV_{DD} is source)
V_{FREF}	Fixed Internal Reference Voltage	–1.5%	V_{FREFL}[3]	+1.5%	V	Low fixed reference
		–1.5%	1.0	+1.5%	V	1.0V fixed reference
		–1.5%	V_{FREFH}[3]	+1.5%	V	High fixed reference
		–1.5%	1.20	+1.5%	V	Bandgap voltage
		–0.5%	V_{DD}/2	+0.5%	V	V_{DD}/2 fixed reference
V_{VBIAS}	VBIAS Reference Voltage	–1.5%	1.25	+1.5%	V	VBIASLVL=00, AV_{DD} ≥ 1.8V
		–1.5%	1.50	+1.5%	V	VBIASLVL=01, AV_{DD} ≥ 2.0V
		–1.5%	2.00	+1.5%	V	VBIASLVL=10, AV_{DD} ≥ 2.5V
		–1.5%	2.50	+1.5%	V	VBIASLVL=11, AV_{DD} ≥ 3.0V
I_{VBIAS}	VBIAS Active Current		2.5		µA	
I_{VBIAS_OUT}	VBIAS Sourced Reference Current	-20		0	µA	

Notes:
1. Data in the Typical column is from characterization at 3.0V and 25°C. These values are provided for design guidance only and are not tested in production.
2. V_{PREF} is a user-set programmable reference voltage. See Tables 160 and 161 in the Reference System chapter on page 300.
3. $V_{FREFH/L}$ is a user-selected fixed high/low reference voltage. See Table 159 in the Reference System chapter on page 300.

Table 238. Reference System Electrical Characteristics (Continued)

		V_{DD}=1.8V to 3.6V T_A=–40°C to +85°C				
Symbol	**Parameter**	**Min**	**Typ**[1]	**Max**	**Units**	**Conditions**
I_{PROG}	Programmable Reference Active Current		1.6		µA	VBIAS=2.0V. I_{VBIAS} is also consumed when one or more programmable reference are enabled
T_{WAKEF}	Time for Wake up, fixed references		2	5	µs	Upon selection of a fixed reference
T_{WAKEP}	Time for Wake up, VBIAS or programmable internal references			2	ms	Upon enable of VBIAS or Programmable Internal Reference

Notes:
1. Data in the Typical column is from characterization at 3.0V and 25°C. These values are provided for design guidance only and are not tested in production.
2. V_{PREF} is a user-set programmable reference voltage. See Tables 160 and 161 in the Reference System chapter on page 300.
3. $V_{FREFH/L}$ is a user-selected fixed high/low reference voltage. See Table 159 in the Reference System chapter on page 300.

27.4.7. Analog-to-Digital Converter

Table 239 presents electrical and timing data for the F3224 Series' Analog-to-Digital Converter function.

Table 239. Analog-to-Digital Converter Electrical Characteristics and Timing

		V_{DD}=1.8V to 3.6V T_A=–40°C to +85°C				
Symbol	**Parameter**	**Min**	**Typ**[1]	**Max**	**Units**	**Conditions**
N	Resolution			12	Bit	RESOLUT=0
				14	Bit	RESOLUT=1
INL	Integral Nonlinearity			±2	LSB	RESOLUT=0
				±5	LSB	RESOLUT=1, INMODE=0
DNL	Differential Nonlinearity			≤±1	LSB	No missing codes RESOLUT=0
				≤±2	LSB	RESOLUT=1, INMODE=1
	Gain Error			±4	LSB	RESOLUT=0, No gain calibration run
				–	LSB	RESOLUT=0, Gain calibration run
				±16	LSB	RESOLUT=1, No gain calibration run
				TBD	LSB	RESOLUT=1, Gain calibration run
	Offset Error			±3	LSB	RESOLUT=0, No offset calibration run
				±1	LSB	RESOLUT=0, Offset calibration run
				±12	LSB	RESOLUT=1, No offset calibration run
				TBD	LSB	RESOLUT=1, Offset calibration run
I_{DD}ADCI	ADC Active Current with Internal Reference (REFSEL=1x)		215		µA	

Note: 1. Data in the Typical column is from characterization at 3.0V and 25°C. These values are provided for design guidance only and are not tested in production.

Table 239. Analog-to-Digital Converter Electrical Characteristics and Timing (Continued)

		V_{DD}=1.8V to 3.6V T_A=–40°C to +85°C				
Symbol	**Parameter**	**Min**	**Typ**[1]	**Max**	**Units**	**Conditions**
I_{DD}ADCE	ADC Active Current with External Reference or V_{DD} Reference (REFSEL = 0x) including I_{DD}VEXT		195		µA	
V_{EXT_REFP}	External Positive Reference Voltage	1.25		AV_{DD}	V	REFSEL = 01
V_{EXT_REFN}	External Negative Reference Voltage	AV_{SS}	AV_{SS}	VREFP – 1.25V	V	
I_{DD}VEXT	External Reference Active Current (included in I_{DD}ADCE)		45	65	µa	
C_{VREF}	V_{REF} Capacitance		1		µF	
V_{INANA}	Analog Input Range	VREFN		VREFP	V	
C_{IN}	Analog Input Capacitance	13	16	19	pF	
R_{IN}	Analog Input Resistance		1600	4500	Ω	1.8V ≤ V_{DD} < 2.2V
			1350	2000	Ω	2.2V ≤ V_{DD} ≤ 3.6V
T_S	Sampling Time	0.2			µs	
$T_{S\_VDD/2}$	Sampling Time for V_{DD}/2 Fixed Reference		70	380	µs	
T_{S_VBG}	Sampling Time for bandgap Fixed Reference		100	220	µs	
T_{CONV}	Conversion Time		13		ADC clock cycles	12-bit (RESOLUT = 0)
			15			14-bit (RESOLUT = 1)
T_{WAKE_AR}	Time for Wake up, Internal ADC Reference		0.5	1.1	ms	REFSEL = 1x, C_{VREFP} = 1 µF
T_{WAKE_ADC}	Time for Wake up, ADC		34		ADC clock cycles	
f_{ADC_CLK}	Frequency of ADC clock			5	MHz	12-bit (RESOLUT = 0);
				4	MHz	14-bit (RESOLUT = 1)

Note: 1. Data in the Typical column is from characterization at 3.0V and 25°C. These values are provided for design guidance only and are not tested in production.

27.4.8. Comparator

Table 240 presents electrical and timing data for the F3224 Series' Comparator function.

Table 240. Comparator Electrical Characteristics

		V_{DD}=1.8V to 3.6V T_A=–40°C to +85°C				
Symbol	**Parameter**	**Min**	**Typ**[1]	**Max**	**Units**	**Conditions**
I_{DD}COMP	Comparator Active Current		0.2		µA	CPOWER=00
			1		µA	CPOWER=01
			4		µA	CPOWER=10
			27		µA	CPOWER=11
V_{ICM}	Input Common Mode Voltage	V_{SS}		V_{DD}	V	
V_{OS}	Input DC Offset		±3	±10	mV	
V_{HYS}[2]	Input Hysteresis		0		mV	HYST=00
			15		mV	HYST=01
			40		mV	HYST=1x
T_{PROP}	Propagation Delay		5		µs	CPOWER=00
			1.5		µs	CPOWER=01
			700		ns	CPOWER=10
			150		ns	CPOWER=11
T_{WAKE}	Time for Wake up			100	µs	

Notes:
1. Data in the Typical column is from characterization at 3.0V and 25°C. These values are provided for design guidance only and are not tested in production.
2. Hysteresis is defined as overdrive of one input relative to the other to cause output transition, thus is half of the full hysteresis band.

27.4.9. Operational Amplifier, Op Amp A

Table 241 presents electrical and timing data for the F3224 Series' Operational Amplifier function, Op Amp A.

Table 241. Operational Amplifier Electrical Characteristics

		V_{DD}=1.8V to 3.6V T_A=–40°C to +85°C				
Symbol	**Parameter**	**Min**	**Typ[1]**	**Max**	**Units**	**Conditions**
I_{DD}OA	Op Amp Active Current		1		µA	OPOWER=0 (low power)
			32		µA	OPOWER=1 (normal power)
V_{ICM}	Input Common Mode Voltage	V_{SS}		V_{DD}	V	
V_{OS}	Input Offset Voltage	–4		4	mV	
V_{OSTA}	Input Offset Temperature Drift	–	1	10	µV/°C	
V_{OCM}	Output Common Mode Voltage	V_{SS}		V_{DD}	V	
AV	DC Gain		72		dB	OPOWER=0
			98		dB	OPOWER=1
PM	Phase Margin		70		deg	OPOWER=0, 50 pF loading
			85		deg	OPOWER=1, 50 pF loading
GBW	Gain Bandwidth Product		40		kHz	OPOWER=0, 50 pF loading
			625		kHz	OPOWER=1, 50 pF loading
I_{OUT}OA	DC Load Current	–50		50	µA	OPOWER=0
		–250		250	µA	OPOWER=1
GAINTOL	Internal Gain Network DC Gain Tolerance		0.5		%	INNSEL=01, GAIN select ≤ 10x
			1.0		%	INNSEL=01, GAIN select ≥ 12x
R_{GAIN}	Internal Gain Network Resistance		200k		Ω	INNSEL=01,

Note: 1. Data in the Typical column is from characterization at 3.0V and 25°C. These values are provided for design guidance only and are not tested in production.

Table 241. Operational Amplifier Electrical Characteristics (Continued)

		V_{DD}=1.8V to 3.6V T_A=–40°C to +85°C				
Symbol	**Parameter**	**Min**	**Typ**[1]	**Max**	**Units**	**Conditions**
R_{OUT}	Output Resistance		375	750	Ω	$V_{SS} \le V_{AMPOUT} \le V_{DD}$, OPOWER=0
			225	550	Ω	$V_{SS} \le V_{AMPOUT} \le V_{DD}$, OPOWER=1
			3	6	Ω	V_{SS}+0.3V ≤ V_{AMPOUT} ≤ V_{DD}–0.3V, OPOWER=0
			1	3	Ω	V_{SS}+0.3V ≤ V_{AMPOUT} ≤ V_{DD}–0.3V, OPOWER=1
SR	Slew Rate	15	30		mV/µs	OPOWER=0
		260	525		mV/µs	OPOWER=1
T_{WAKE}	Time for Wake up		2	200	µs	

Note: 1. Data in the Typical column is from characterization at 3.0V and 25°C. These values are provided for design guidance only and are not tested in production.

27.4.10. Low Voltage Detect

Table 242 presents electrical and timing data for the F3224 Series' Low Voltage Detect function.

Table 242. Low Voltage Detect Electrical Characteristics

		V_{DD}=1.8V to 3.6V T_A=–40°C to +85°C				
Symbol	**Parameter**	**Min**	**Typ**[1]	**Max**	**Units**	**Conditions**
I_{DD}LVDV	LVD Active Current when VBO enabled		1.0		µA	Active or Halt modes
			0.035		µA	Stop Mode
I_{DD}LVD	LVD Active Current when VBO disabled		2.0		µA	Active or Halt modes
			0.07		µA	Stop Mode
V_{TH}	Detected Source Voltage	–5%	V_{TP}[2]	+5%	V	
T_{DELAY}	Delay from source voltage falling lower than V_{TP} to LVD output logic High	50	1000		ns	

Notes:
1. Data in the Typical column is from characterization at 3.0V and 25°C. These values are provided for design guidance only and are not tested in production.
2. V_{TP} is a user-set threshold voltage to be detected. See Table 202 on page 367 in the Flash Option Bits chapter on page 358.

27.4.11. Internal Precision Oscillator

Table 243 presents electrical and timing data for the F3224 Series' Internal Precision Oscillator (IPO) function.

Table 243. IPO Electrical Characteristics

		V_{DD}=1.8 to 3.6V T_A=–40°C to +85°C				
Symbol	**Parameter**	**Min**	**Typ[1]**	**Max**	**Units**	**Conditions**
I_{DD}IPO	IPO Active Supply Current		0.3		µA	
F_{IPO}	Output Frequency	–2%	32.768	+2%	kHz	
	Duty Cycle of Output	45		55	%	
T_{WAKE}	Time for Wake up		25		µs	

Note: 1. Data in the Typical column is from characterization at 3.0V and 25°C. These values are provided for design guidance only and are not tested in production.

27.4.12. Low Frequency Crystal Oscillator

Table 244 presents electrical and timing data for the F3224 Series' Low Frequency Crystal Oscillator function.

Table 244. Low Frequency Oscillator (LFXO) Characteristics

		V_{DD}=1.8V to 3.6V T_A=–40°C to +85°C				
		V_{DD}=1.8 to 3.6V				
Symbol	**Parameter**	**Min**	**Typ[1]**	**Max**	**Units**	**Conditions**
I_{DD}LFXO	LFXO Active Current		0.5		µA	C_{LOAD}=12.5 pF
F_{XTAL2}	LFXO Frequency		32.768		kHz	
gm	Oscillator Transconductance	5			µA/V	
Rfdbk	Feedback Resistor		5		MΩ	
T_{WAKE}	Time for Wake up		400	1000	ms	
DC_{LFXO}	Duty Cycle	40	50	60	%	

Note: 1. Data in the Typical column is from characterization at 3.0V and 25°C. These values are provided for design guidance only and are not tested in production.

27.4.13. Digitally Controlled Oscillator and Frequency-Locked Loop

Table 245 presents electrical and timing data for the F3224 Series' Digitally Controlled Oscillator (DCO) and Frequency-Locked Loop functions.

Table 245. DCO and FLL Electrical Characteristics

		V_{DD}=1.8 to 3.6V T_A=–40°C to +85°C				
Symbol	**Parameter**	**Min**	**Typ**[1]	**Max**	**Units**	**Conditions**
I_{DD}DCO	DCO Active Supply Current		50		µA	F_{DCO}=1MHz
			100		µA	F_{DCO}=10MHz
			200		µA	F_{DCO}=20MHz
F_{DCO}	DCO Frequency	1		20	MHz	
$F_{FLLCLKIN}$	FLL Input Clock Frequency		32.768		kHz	
DC_{DCO}	Duty Cycle of DCO Output	45		55	%	
RES_{DCO}	DCO Resolution		200		ps	
$T_{FLADONE}$	FLL Fast Lock Algorithm Time (FLADONE=1)		3.4		ms	Pclk=32.768kHz; fast locking selected

Note: 1. Data in the Typical column is from characterization at 3.0V and 25°C. These values are provided for design guidance only and are not tested in production.

27.4.14. General-Purpose I/O Port Input Data Sample Timing

Figure 72 and Table 246 present the timing of the GPIO port input sampling. The input value on a GPIO port pin is sampled on the rising edge of the system clock. The port value is available to the eZ8 CPU on the second rising clock edge following the change of the port value.

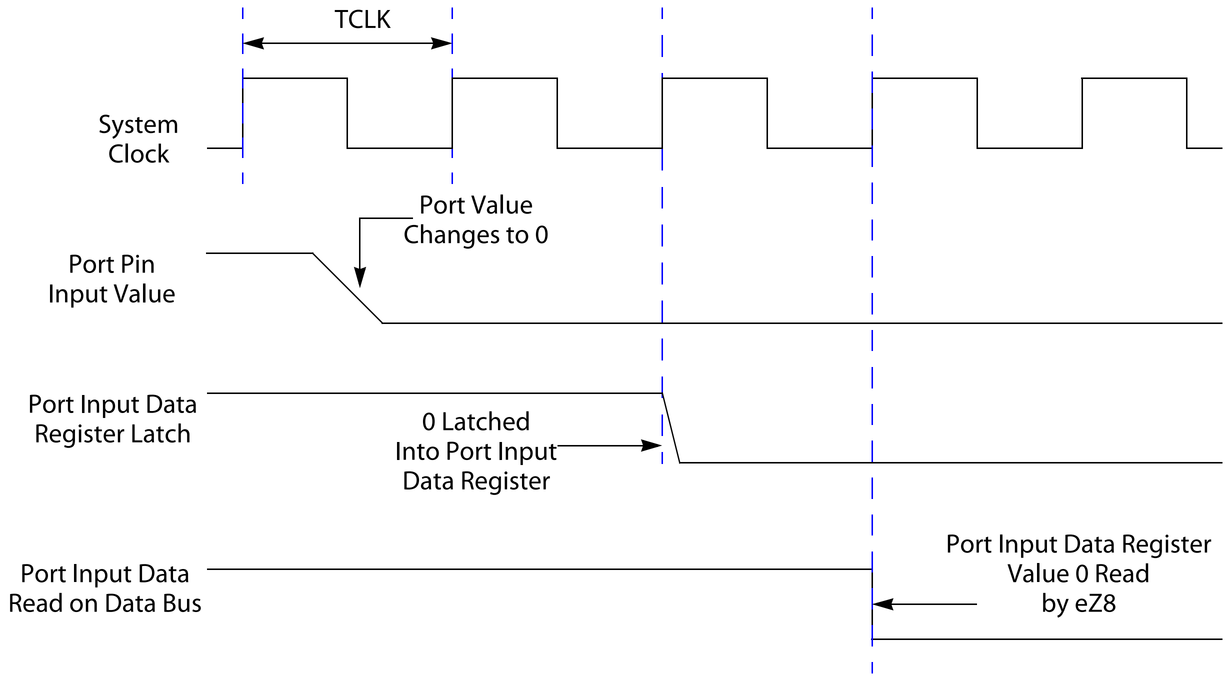

TCLK
System Clock
Port Value Changes to 0
Port Pin Input Value
Port Input Data Register Latch
0 Latched Into Port Input Data Register
Port Input Data Read on Data Bus
Port Input Data Register Value 0 Read by eZ8

Figure 72. Port Input Sample Timing

Table 246. GPIO Port Input Timing

		Delay (ns)	
Parameter	**Abbreviation**	**Min**	**Max**
T_{S_PORT}	Port Input Transition to X_{IN} Rise Setup Time (Not pictured)	5	–
T_{H_PORT}	X_{IN} Rise to Port Input Transition Hold Time (Not pictured)	0	–
T_{SMRPW}	GPIO Port Pin Pulse Width to ensure Stop-Mode Recovery (for GPIO port pins enabled as SMR sources)	125	

27.4.15. General-Purpose I/O Port Output Timing

Figure 73 and Table 247 provide timing information for the GPIO port pins.

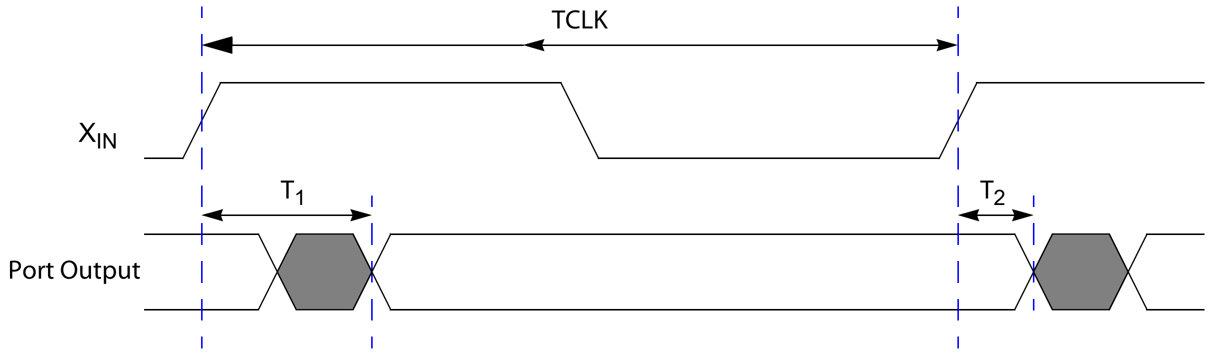


Figure 73. GPIO Port Output Timing

Table 247. GPIO Port Output Timing

		Delay (ns)	
Parameter	**Abbreviation**	**Min**	**Max**
GPIO Port Pins			
T_1	X_{IN} Rise to Port Output Valid Delay	–	15
T_2	X_{IN} Rise to Port Output Hold Time	2	–

27.4.16. On-Chip Debugger Timing

Figure 74 and Table 248 provide timing information for the DBG pin. The DBG pin timing specifications assume a 4ns maximum rise and fall time.

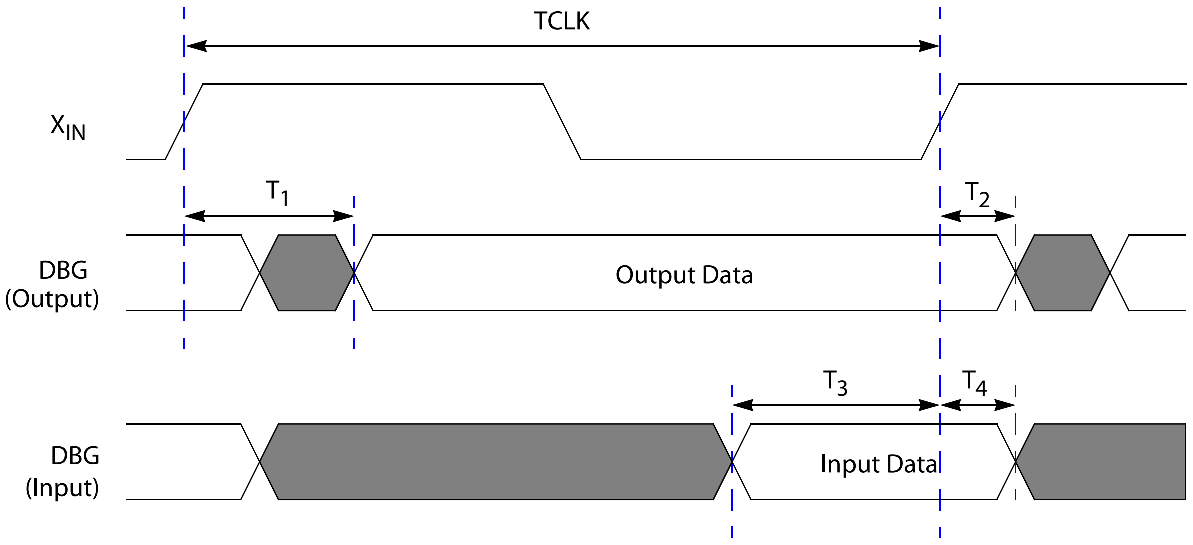


Figure 74. On-Chip Debugger Timing

Table 248. On-Chip Debugger Timing

		Delay (ns)	
Parameter	**Abbreviation**	**Min**	**Max**
DBG			
T_1	X_{IN} Rise to DBG Valid Delay	–	15
T_2	X_{IN} Rise to DBG Output Hold Time	2	–
T_3	DBG to X_{IN} Rise Input Setup Time	5	–
T_{DBGPW}	DBGPulse Width to be Recognized	125	

27.4.17. UART Timing

Figure 75 and Table 249 provide timing information for the UART pins for situations in which CTS is used for flow control. The CTS to DE assertion delay (T1) assumes that the Transmit Data Register has been loaded with data prior to CTS assertion.

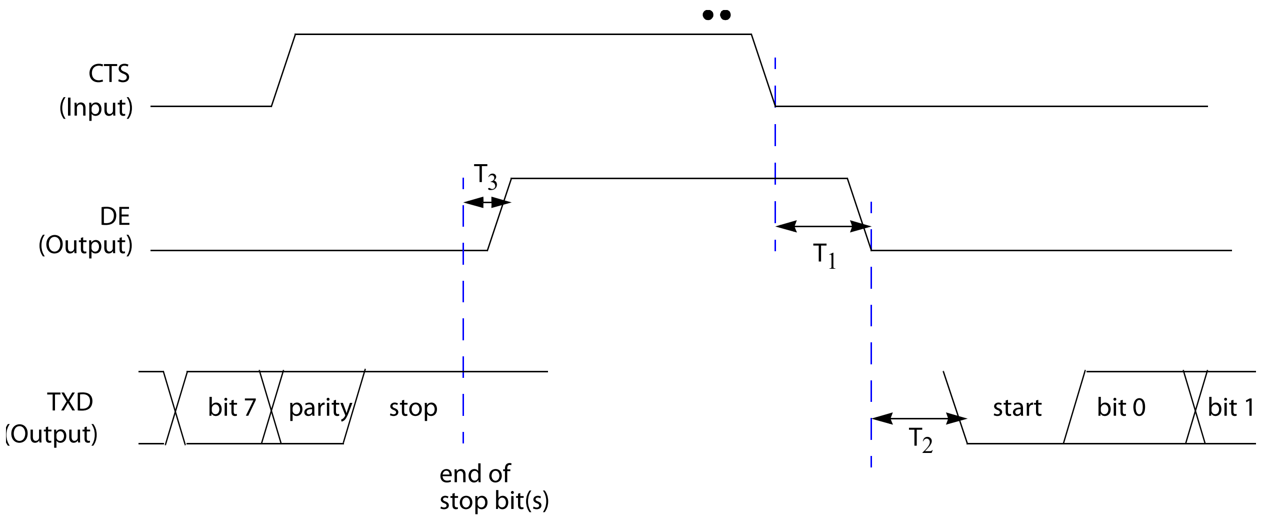


Figure 75. UART Timing With CTS

Table 249. UART Timing with CTS

Parameter	Abbreviation	Delay (ns)	
		Min	Max
T_1	CTS Fall to DE output delay	2 * X_{IN} period	2 * X_{IN} period + 1 bit time
T_2	DE assertion to TXD falling edge (start bit) delay	± 5	
T_3	End of Stop Bit(s) to DE deassertion delay	± 5	

Figure 76 and Table 250 provide timing information for the UART pins for situations in which CTS is not used for flow control. DE asserts after the Transmit Data Register has been written. DE remains asserted for multiple characters as long as the Transmit Data Register is written with the next character before the current character has completed.

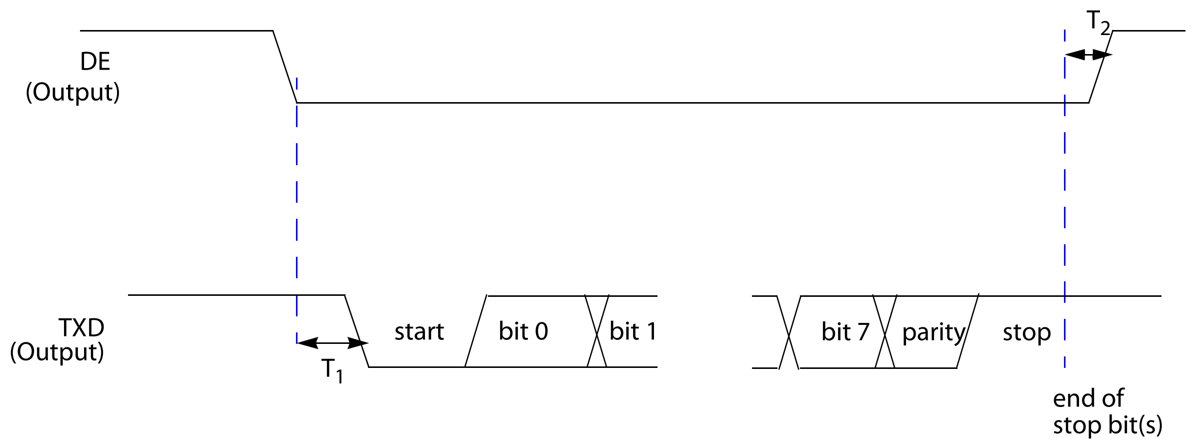


Figure 76. UART Timing Without CTS

Table 250. UART Timing Without CTS

		Delay (ns)	
Parameter	**Abbreviation**	**Min**	**Max**
T_1	DE assertion to TXD falling edge (start bit) delay	1 * Sysclk period	1 bit time
T_2	End of Stop Bit(s) to DE deassertion delay (Tx Data Register is empty)	± 5	

Chapter 28. Packaging

Zilog's F3224 Series of MCUs are available in the following packages:

- 32-pin Quad Flat No Lead (QFN)
- 44-pin Quad Flat No Lead (QFN))

Current diagrams for each of these packages are published in Zilog's Packaging Product Specification (PS0072), which is available free for download from the Zilog website.

Chapter 29. Ordering Information

29.1. Part Number Listing

Order your F3224 Series devices from Zilog using the part numbers listed in Table 250. For more information about ordering, please consult your local Zilog sales office. The Zilog website lists all regional offices and provides additional Z8 Encore! XP product information.

Table 250. F3224 Series Ordering Information

Part Number	Flash	Register RAM	I^2C Master/Slave Controller	ESPI	I/O Lines	Interrupt Vectors	16-Bit Timers w/ PWM	12-Bit A/D Channels	UART with LIN/DALI/DMX	Comparator	Op Amp	Multichannel Timer	Description
F3224 Series with 32KB Flash, 12/14-Bit Analog-to-Digital Converter													
Extended Temperature: –40 °C to 85 °C													
Z8F3224QN020XK	32KB	3.75KB	1	1	36	23	3	15	1	2	2	1	QFN 44-pin package
Z8F3224QK020XK	32KB	3.75KB	1	1	26	22	3	12	1	2	2	1	QFN 32-pin package

Table 250. F3224 Series Ordering Information (Continued)

Part Number	Flash	Register RAM	I^2C Master/Slave Controller	ESPI	I/O Lines	Interrupt Vectors	16-Bit Timers w/ PWM	12-Bit A/D Channels	UART with LIN/DALI/DMX	Comparator	Op Amp	Multichannel Timer	Description
Z8 Encore! XP F1624 Series with 16KB Flash, 12/14-Bit Analog-to-Digital Converter													
Extended Temperature: –40 °C to 85 °C													
Z8F1624QN020XK	16KB	3.75KB	1	1	36	23	3	15	1	2	2	1	QFN 44-pin package
Z8F1624QK020XK	16KB	3.75KB	1	1	26	22	3	12	1	2	2	1	QFN 32-pin package
Z8 Encore! XP F3224 Series Development Kit													
Z8F32240100ZCOG	Z8 Encore! XP F3224 Series 44-Pin Development Kit												
ZUSBESC0200ZACG	Encore! Smart Cable Accessory Kit												
ZUSBSC00100ZACG	USB Smart Cable Accessory Kit												
ZUSBOPTSC01ZACG	USB Opto-Isolated Smart Cable Accessory Kit												
ZENETSC0100ZACG	Ethernet Smart Cable Accessory Kit												

Table 251 shows the F3224 Series of microcontrollers that specifically support the ZMOTION Engine Library. A specific version of the device must be used for the Library to operate correctly. This version is identified by the 2258 suffix on the device part number.

Please refer to UM0275 for more information about the Z8F3224 ZMOTION Library.

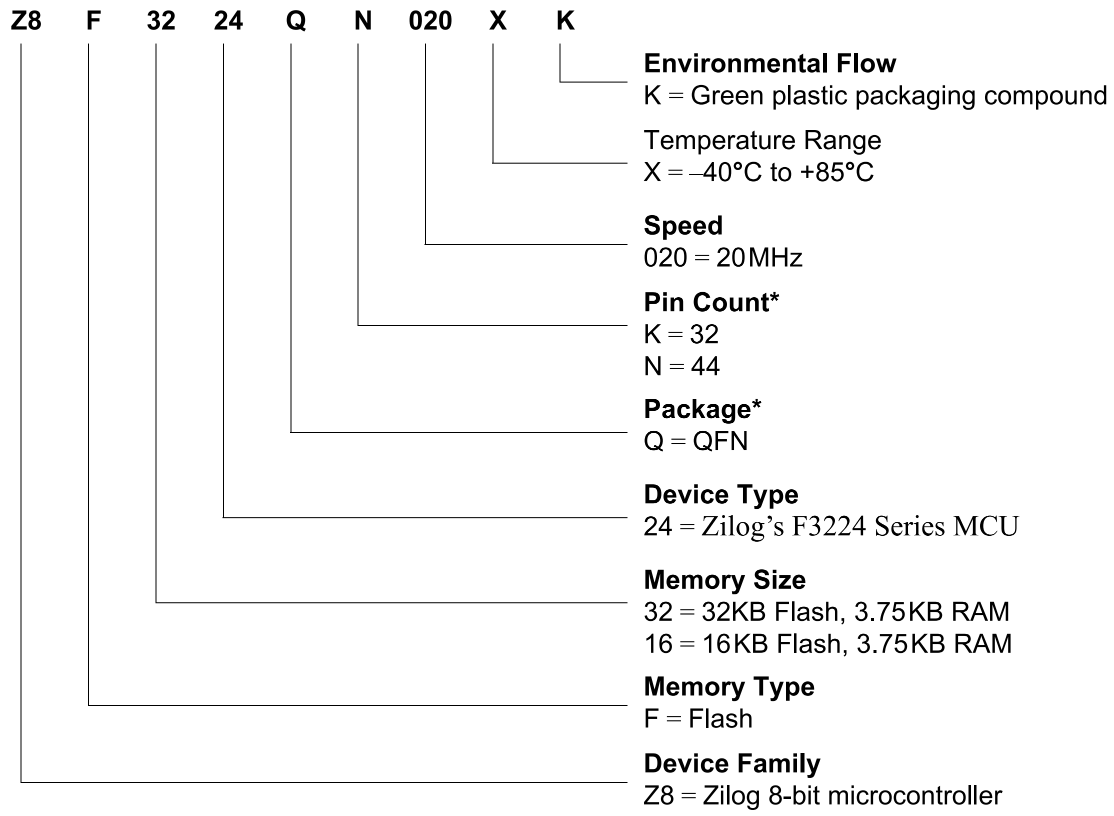
Table 251. F3224 ZMOTION Series Ordering Information

Part Number	Package	Memory
Z8 Encore! XP F3224 ZMOTION Series Devices		
Z8F3224QN020XK2258	44-Pin QFN	32KB Flash, 3.75KB RAM
Z8F3224QK020XK2258	32-Pin QFN	32KB Flash, 3.75KB RAM
Z8 Encore! F3224 Series Development Kit		
ZMOTIONL400ZCOG	Z8F3224 ZMOTION Development Kit	

29.2. Part Number Suffix Designations

Zilog part numbers consist of a number of components, as indicated in the following example.

Example. Part number Z8F3224QN020XK is an 8-bit, 20 MHz Flash microcontroller with 32 KB of Flash memory in an 44-pin QFN package, operating within a –40°C to +85°C temperature range and built using lead-free solder.



Note: * See Table 252 for the combination of package and pin count.

Table 252. Package and Pin Count Description

		Pin	
		32	44
Package	QFN	√	√

29.3. Precharacterization Product

The product represented by this document is newly introduced, and Zilog has not completed the full characterization of the product. This document states all information that Zilog knows about this product at this time, but additional features or nonconformance with some aspects of the document might be found, either by Zilog or its customers, in the course of further application and characterization work. In addition, Zilog cautions that delivery might be uncertain at times, because of start-up yield issues.

Customer Support

To share comments, get your technical questions answered, or report issues you may be experiencing with our products, please visit Zilog's Technical Support page at http://support.zilog.com.

To learn more about this product, find additional documentation, or to discover other facets about Zilog product offerings, please visit the Zilog Knowledge Base at http://zilog.com/kb.

This publication is subject to replacement by a later edition. To determine whether a later edition exists, please visit the Zilog website at http://www.zilog.com.

Index