Embedded in Life

An IXYS Company

Z8 Encore!® Motor Control Flash MCUs

Z8FMC16100 Series

Product Specification

PS024615-0811

www.zilog.com

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

 Information in this publication concerning the devices, applications, or technology described is intended to suggest possible uses and may be superseded. ZILOG, INC. DOES NOT ASSUME LIABILITY FOR OR PROVIDE A REPRESENTATION OF ACCURACY OF THE INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED IN THIS DOCUMENT. ZILOG ALSO DOES NOT ASSUME LIABILITY FOR INTELLECTUAL PROPERTY INFRINGEMENT RELATED IN ANY MANNER TO USE OF INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED HEREIN OR OTHERWISE. The information contained within this document has been verified according to the general principles of electrical and mechanical engineering.

Revision History

Each instance in Revision History reflects a change to this document from its previous revision. For more details, refer to the corresponding pages or appropriate links listed in the table below.

Date	Revision Level	Description	Page No.
Aug 2011	15	Corrected temperature range in Table 144; reflowed LOT.	261
Aug 2011	14	Added Flash Memory Electrical Characteristics and Timing table.	261
		Corrected PWEN to PWMEN, all instances per CR#13094.	All
		Corrected PWM Fault Mask Register (PWMFM) bits(4:3) reset value from 000 to 00 per CR#13094.	79, 309
Sep 2010	13	Updated Figures 57 and 58.	293, 294
Aug 2010	12	Deleted Trim Option Bits at 0004H (ADCCAL) table in Trim Bit Address 0003H.	226
Apr 2008	11	Updated Ordering Information and Part Number descriptions.	295, 297
Jan 2008	10	Updated Table 145 in the Electrical Characteristics chapter.	262
Jul 2007	09	Added description for Alternate Function registers in the General-Purpose Input/Output chapter; added note under Fault Detection and Protection section in the Pulse Width Modulator chapter; set Fault1 as active Low; updated Table 9 in the Reset and Stop Mode Recovery chapter.	35, 71, All, 28
Dec 2006	08	Corrected Figure 9; corrected CMP0RST bit; updated Figure 8 and Table 55.	68. 312, 67, 82
Nov 2006	07	Updated Table 103; updated Comparator and Op Amp; updated PWM Minimum Pulse Width Filter; updated Table 54; updated Hex Address: F28; updated Tables 136 and 137; added caution statement.	196, 333, 81, 312, 251, 252, 33

Table of Contents

List of Figures

List of Tables

Introduction

The Z8FMC16100 Series MCU MCU is based on Zilog's advanced eZ8 8-bit CPU core and is optimized for motor control applications. It supports control of single- and multi-phase variable-speed motors. Target applications are consumer appliances, HVAC, factory automation, refrigeration and automotive applications.

Features

The Z8FMC16100 Series MCU features include:

- 20MHz eZ8 CPU core
- Up to 16KB Flash program memory
- 512B register RAM
- Fast 8-channel 10-bit Analog-to-Digital Converter (ADC)
- 12-bit Pulse-Width Modulator (PWM) module with three complementary pairs, or six independent PWM outputs with deadband generation and fault trip input
- 16-bit timer with Capture, Compare and PWM capability
- Analog Comparator
- Operational Amplifier
- Inter-Integrated Circuit (I^2C) controller supports master, slave and multimaster modes
- Universal Asynchronous Receiver/Transmitter (UART) with interface support for Local Interconnect Network (LIN) and IrDA
- Serial Peripheral Interface (SPI) controller
- Internal Precision Oscillator (IPO)
- On-chip oscillator supports external crystals, ceramic resonators and clock drivers
- 17 General-Purpose Input/Output (GPIO) pins
- Voltage Brown-Out/Power-On Reset (VBO/POR)
- Watchdog Timer (WDT) with internal RC oscillator
- On-Chip Debugger (OCD)
- In-circuit serial programming
- Operating at 2.7V to 3.6V
- 32-pin packages

- Lead-free packaging
- Standard and extended temperature ranges: 0°C to 70°C (S) and –40°C to 105°C (E)
- Up to 20 interrupts with configurable priority

Block Diagram

Figure 1 presents the architecture of the Z8FMC16100 Series MCU MCU.

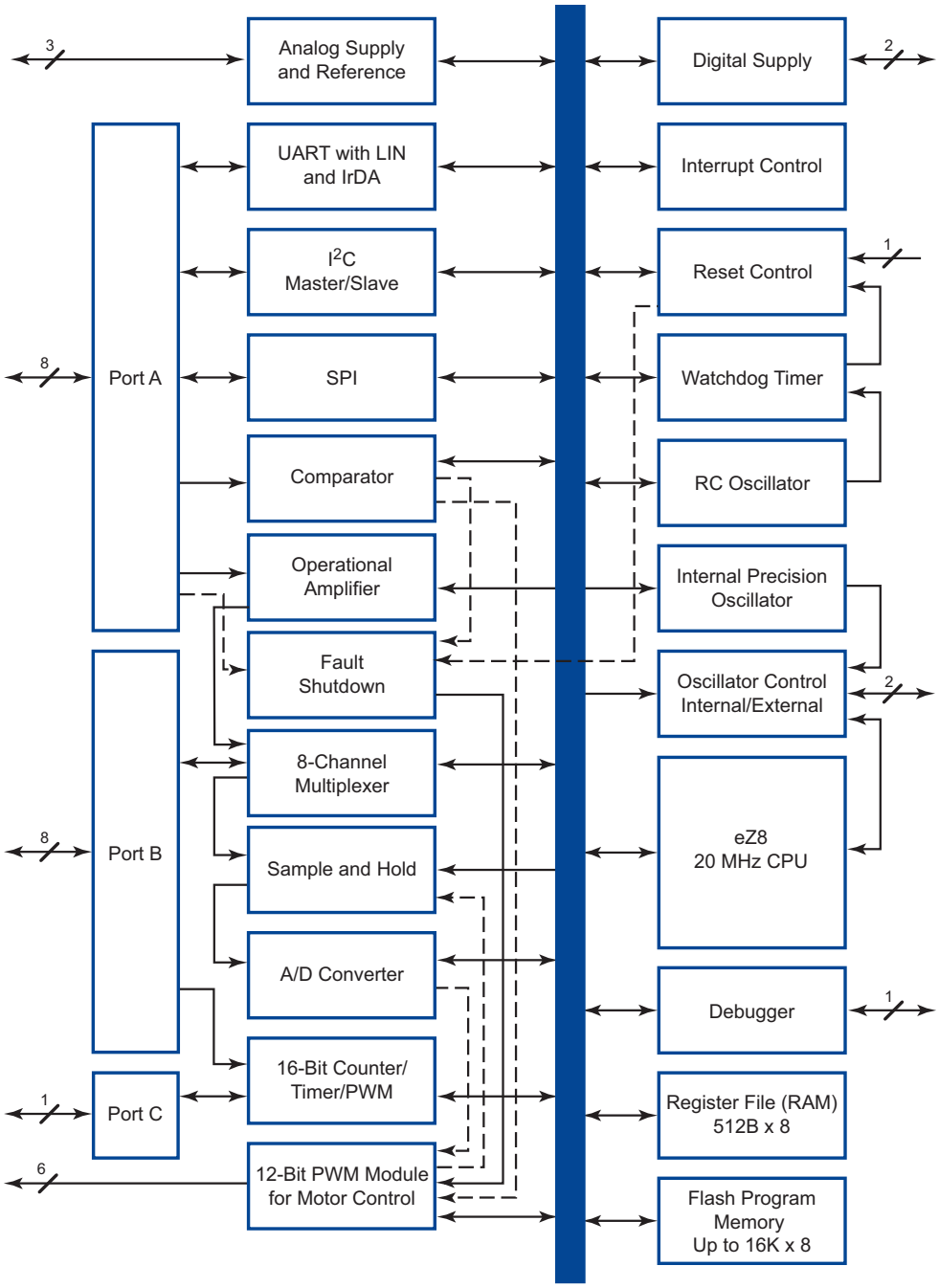


Figure 1. Z8FMC16100 Series MCU MCU Block Diagram

CPU and Peripheral Overview

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

- 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 much greater depth in subroutine calls and interrupts than hardware stacks
- Compatible with existing Z8 assembly code
- New instructions improve execution efficiency for code developed using higher-level programming languages, including 'C' language
- 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 10 MIPS operation
- C-Compiler friendly
- 2-9 clock cycles per instruction

For more details about the eZ8 CPU, refer to the eZ8 CPU Core User Manual (UM0128), which is available for download at www.zilog.com.

Pulse Width Modulator for Motor Control Applications

To rotate a 3-phase motor three voltage and current signals must be supplied, each 120° shifted from each other. To control a 3-phase motor, the MCU must provide 6 PWM outputs. The Z8FMC16100 Series MCU features a flexible PWM module with three complementary pairs or six independent PWM outputs supporting deadband operation and fault protection trip input. These features provide multiphase control capability for various motor types and ensure safe operation of the motor by providing immediate shutdown of the PWM pins during fault condition.

10-Bit Analog-to-Digital Converter

The Z8FMC16100 Series MCU devices feature up to eight channels of 10-bit analog-to-digital conversion.

Analog Comparator

The Z8FMC16100 Series MCU features an on-chip analog comparator with external input pins.

Operational Amplifier

The Z8FMC16100 Series MCU features a two-input and one-output operational amplifier.

General-Purpose Input/Output

The Z8FMC16100 Series MCU features 17 GPIO pins. Each pin is individually programmable.

Flash Controller

The Flash Controller programs and erases the Flash Memory. The Z8FMC16100 Series MCU products contain 16KB of on-chip Flash Memory. A sector protection scheme allows for flexible protection of user code.

Random Access Memory

An internal random access memory (RAM) of 512B provides storage space for data, variables and stack operations.

UART with LIN and IrDA

A full-duplex 9-bit UART provides a serial, asynchronous communication and supports the LIN serial communications protocol. The UART communication is full-duplex and capable of handling asynchronous data transfers. The UARTs support 8- and 9-bit data modes, selectable parity and an efficient bus transceiver Driver Enable signal for controlling a multitransceiver bus, such as RS-485. The LIN bus is a cost-efficient single master, multiple slave organization which supports speed up to 20Kbps.

Serial Peripheral Interface

The SPI allows the Z8FMC16100 Series MCU to exchange data between other peripheral devices such as electrically erasable programmable read-only memory (EEPROM), A/D converters and ISDN devices. The SPI is a full-duplex, synchronous, character-oriented channel that supports a 4-wire interface.

Inter-Integrated Circuit

The I^2C controller makes the Z8FMC16100 Series MCU compatible with the I^2C protocol. The I^2C controller consists of two bidirectional bus lines, a serial data (SDA) line and a serial clock (SCL) line. The I^2C operates as a master and/or slave and supports multi-master bus arbitration.

Internal Precision Oscillator

The IPO provides a stable, accurate time base without the requirement for external components. This reduces system cost in many applications by eliminating the requirement for external crystals or ceramic resonators. The IPO frequency is 5.5296MHz.

Crystal Oscillator

The on-chip crystal oscillator features programmable gain to support crystals and ceramic resonators ranging from 32kHz to 20MHz. The oscillator can also be used with clock drivers.

Standard Timer

The 16-bit reloadable timer can be used for timing/counting events and PWM signal generation. This timer provides a 16-bit programmable reload counter and operate in ONE-SHOT, CONTINUOUS, GATED, CAPTURE, COMPARE, CAPTURE and COMPARE and PWM modes. This timer can measure velocity from a tachometer wheel or read sensor outputs for rotor position for brushless DC motor commutation. The standard timer can also be used for general-purpose timing and counting operations.

Interrupt Controller

The Z8FMC16100 Series MCU supports 3 levels of programmable interrupt priority. The interrupt sources include internal peripherals, GPIO pins and system fault detection.

Reset Controller

The Z8FMC16100 Series MCU is reset using the $\overline{\text{RESET}}$ pin, POR, WDT, Stop Mode Recovery, or VBO warning signal.

On-Chip Debugger

The Z8FMC16100 Series MCU features an integrated On-Chip Debugger (OCD). The single-pin OCD interface provides a rich set of debugging capabilities, like reading and

writing registers, programming the Flash, setting break points and executing code. The OCD simplifies code development and allows easy in-circuit programming.

Signal and Pin Descriptions

The Z8FMC16100 Series MCU products are available in various package styles and pin configurations. This chapter describes the signals and available pin configurations for each package style. For more information about the physical package specifications, see the Packaging chapter on page 293.

Pin Configurations

Figure 2 displays the pin configuration for the packages available in the Z8FMC16100 Series MCU. See Table 1 for description of each signal.

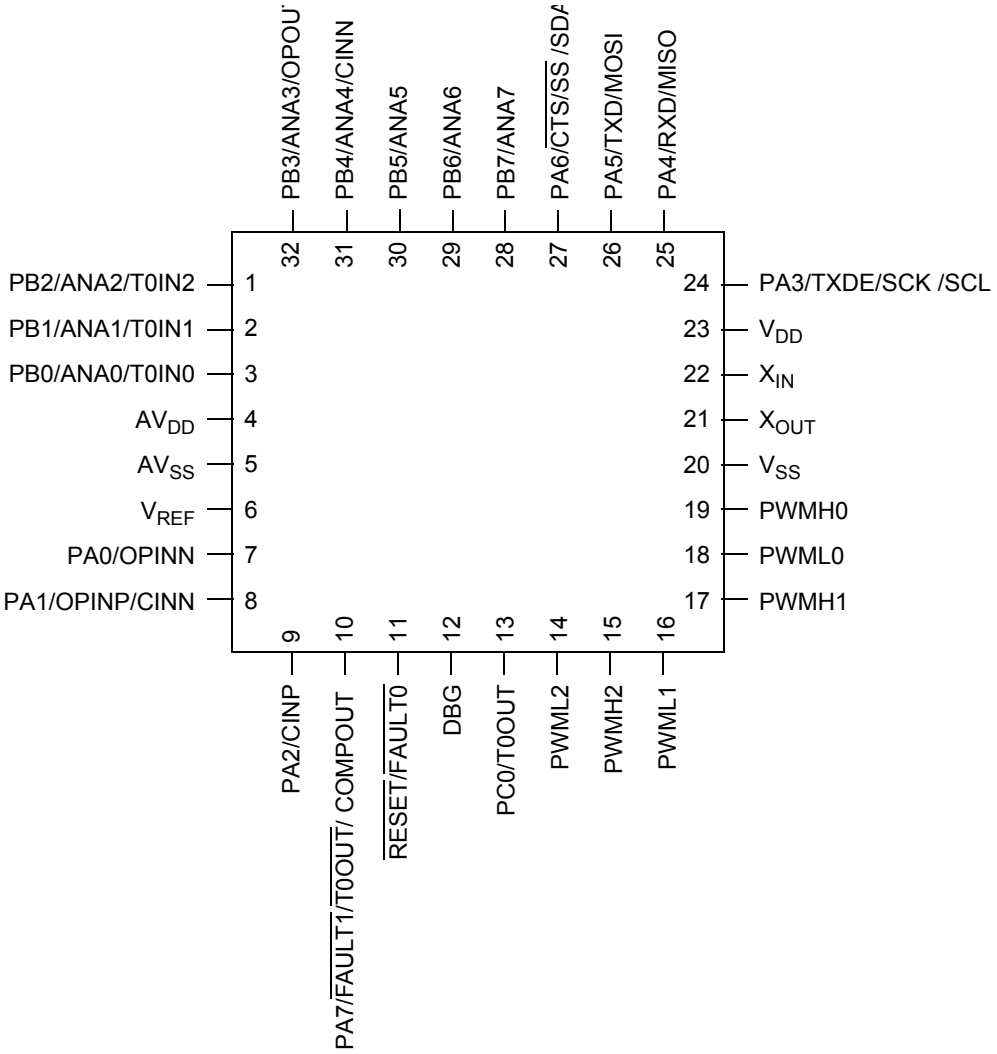


Figure 2. Z8FMC16100 Series MCU in 32-Pin QFN and LQFP Package

Signal Descriptions

Table 1 describes the Z8FMC16100 Series MCU signals.

Table 1. Signal Descriptions

Signal Mnemonic	I/O	Description
General-Purpose Input/Output Ports A–H		
PA[7:0]	I/O	Port A[7:0]: These pins are used for general-purpose I/O.
PB[7:0]	I/O	Port B[7:0]: These pins are used for general-purpose I/O.
PC[0]	I/O	Port C[0]: These pins are used for general-purpose I/O.
Pulse Width Modulator for Motor Control		
PWMH0/PWMH1/ PWMH2	O	PWM High output.
PWML0/PWML1/ PWML2	O	PWM Low output.
$\overline{\text{FAULT0}}$/$\overline{\text{FAULT1}}$	I	PWM FAULT condition input. $\overline{\text{FAULT0}}$ is active Low, $\overline{\text{FAULT1}}$ is active Low.
Serial Peripheral Interface		
MISO	I/O	Master-In/Slave-Out.
MOSI	I/O	Master-Out/Slave-In.
SCLK	I/O	SPI Clock.
SS	I	Slave Select
Timers		
T0OUT, $\overline{\text{T0OUT}}$	O	General-Purpose Timer Outputs.
T0INx	I	General-Purpose Timer Input: This signal is used as the capture, gating and counter inputs.
UART Controller		
TXD	O	Transmit Data: This signal is the transmit output from the UART and IrDA.
RXD	I	Receive Data: This signal is the receiver input for the UART and IrDA.
CTS	I	Clear To Send signal from the receiving device that ready to receive data.
TXDE	O	Driver Enable: This signal allows automatic control of external RS-485 drivers. The DE signal may be used to ensure an external RS-485 driver is enabled when data is transmitted by the UART.

Table 1. Signal Descriptions (Continued)

Signal Mnemonic	I/O	Description
Analog		
ANA[7:0]	I	Analog Input: These signals are inputs to the ADC.
V_{REF}	I/O	Voltage buffer output: This signal provides reference voltage for external components. **Caution:** If using the internal reference voltage generator, a 10 µF capacitor must be placed on this pin to Ground.
CINP	I	Comparator positive input.
CINN	I	Comparator negative input.
COMPOUT	O	Comparator output.
OPINP	I	Operational amplifier positive input.
OPINN	I	Operational amplifier negative input.
OPOUT	O	Operational amplifier output.
Oscillators		
X_{IN}	I	The External Crystal Input is the input pin to the crystal oscillator. A crystal can be connected between it and the X_{OUT} pin to form the oscillator. In addition, this pin is used with external RC networks or external clock drivers to provide the system clock.
X_{OUT}	O	External Crystal Output: This pin is the output of the crystal oscillator. A crystal is connected between it and the X_{IN} pin to form the oscillator. This pin must be left unconnected when not using a crystal.
On-Chip Debugger		
DBG	I/O	Debug: This open-drain pin provides the single-pin control and data interface to the OCD. For operation of the OCD, all power pins (V_{DD}) must be supplied with power and all ground pins (V_{SS}) must be grounded. **Caution:** This pin is open-drain and must have an external pull-up resistor to ensure proper operation.
Reset–PWM Fault		
$\overline{\text{RESET}}$/$\overline{\text{FAULT0}}$	I/O	The default state of the $\overline{\text{RESET}}$ pin is determined by FLTSEL bit7 in user Flash Option2 byte in location 0001 as either 1=RESET or 0=FAULT0. Subsequent change to FLTSEL bit0 in RSTSTAT Register by software can set the pin to either 0=RESET or 1=FAULT0.
Power Supply		
V_{DD}, AV_{DD}	I	Power Supply.
V_{SS}, AV_{SS}	I	Ground.

Pin Characteristics

Table 2 lists the characteristics for each Z8FMC16100 Series MCU's 32 pins. Data in this table is sorted alphabetically by pin symbol mnemonic.

Table 2. Pin Characteristics of the Z8FMC16100 Series MCU

Symbol Mnemonic	Direction	Reset Direction	Active Low or Active High	Tri-State Output	Internal Pull-Up or Pull-Down	Schmitt Trigger Input	Open Drain Output
DBG	I/O	I	N/A	Yes	Pull-up	Yes	Yes
PA[7:0]	I/O	I	N/A	Yes	Pull-up, programmable	Yes	Yes, programmable
PB[7:0]	I/O	I	N/A	Yes	Pull-up, programmable	Yes	Yes, programmable
PC[0]	I/O	I	N/A	Yes	Pull-up, programmable	Yes	Yes, programmable
PWMxH	I/O	Tristate	N/A	Yes	No	Yes	No
PWMxL	I/O	Tristate	N/A	Yes	No	Yes	No
$\overline{\text{RESET}}$	I	I	Low	N/A	Pull-up	Yes	N/A
X_{IN}	I	I	N/A	N/A	No	No	N/A
V_{REF}	I/O	I	N/A	Yes	N/A	N/A	N/A
V_{DD}, AV_{DD}	Supply	N/A	N/A	N/A	No	No	N/A
V_{SS}, AV_{SS}	Supply	N/A	N/A	N/A	No	No	N/A

Note: *x* represents integers 0, 1,... to indicate multiple pins with symbol mnemonics which differ only by an integer.

Address Space

The eZ8 CPU can access three distinct address spaces:

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

For more information about the eZ8 CPU and its address space, refer to the eZ8 CPU Core User Manual (UM0128), which is available for download at www.zilog.com.

Register File

The Z8FMC16100 Series MCU supports up to 512B of internal RAM within the Register File address space. The Register File is composed of two sections: Control Registers and General-Purpose Registers (RAM). When instructions are executed, registers are read when defined as sources and written when defined as destinations. The eZ8 CPU architecture 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 I/O ports. These registers are located at addresses ranging from F00h to FFFh. Some of the addresses within the 256-byte control register section are reserved (unavailable). Reading from reserved Register File addresses returns an undefined value.

Caution: Writing to reserved Register File addresses is not recommended and can produce unpredictable results.

The on-chip RAM always begins at address 000h in the Register File address space. The Z8FMC16100 Series MCU devices provide 512B of on-chip RAM depending upon the device. Reading from Register File addresses outside the available RAM addresses (and not within the control register address space) returns an undefined value. Writing to these Register File addresses produces no effect. To determine the amount of RAM available for the specific Z8FMC16100 Series MCU device, see the Ordering Information chapter on page 295.

Program Memory

The Z8FMC16100 Series MCU products contain 16KB of on-chip Flash Memory in the Program Memory address space, depending upon the device (The Z8FMC08100 MCU contains 8KB and the Z8FMC04100 MCU contains 4KB). Reading from Program Memory addresses outside the available Flash Memory addresses returns FFh. Writing to these unimplemented Program Memory addresses yields no effect.

Table 3 lists the Program Memory map for the Z8FMC16100 Series MCU products.

Table 3. Z8FMC16100 Series MCU **Program Memory Maps**

Program Memory Address (Hex)	Function
Z8FMC16000 Products	
0000-0001	Option Bits
0002-0003	Reset Vector
0004-0007	System Exception Vectors
0008-003F	Interrupt Vectors
0040-3FFF	Program Memory

Note: See Table 29 on page 49 for a list of interrupt vectors.

Data Memory

The Z8FMC16100 Series MCU do not use the eZ8 CPUs 64KB Data Memory address space.

Information Area

Table 4 describes the Z8FMC16100 Series MCU Information Area. This 512 byte Information Area is accessed by setting bit 7 of the Flash Page Select Register to 1. When access is enabled, the Information Area is mapped into the Program Memory and overlays the 512 bytes at addresses FE00h to FFFFh. When the Information Area access is enabled, execution of LDC and LDCI instructions from these Program Memory addresses return the Information Area data rather than the Program Memory data. Reads of these addresses through the OCD also returns the Information Area data. Code execution from these addresses continues to correctly use the Program Memory. Access to the Information Area is read-only.

Table 4. Z8FMC16100 Series Information Area Map

Program Memory Address (Hex)	Function
FE00h–FE3Fh	Reserved.
FE40h–FE5Fh	**Part Number** • 20-character ASCII alphanumeric code • Left justified and filled with zeros (ASCII Null character).
FE60h–FFFFh	Reserved.

Register File Address Map

Table 5 provides the address map for the Register File of the Z8FMC16100 Series MCU.

Table 5. Register File Address Map

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page No
General Purpose RAM: Z8FMC16 devices with 512B On-Chip RAM				
000–1FF	General-Purpose Register File RAM	—	XX	13
200–EFF	Reserved	—	XX	
Timer 0				
F00	Timer 0 High Byte Register (T0H)	T0H	00	98
F01	Timer 0 Low Byte Register (T0L)	T0L	01	98
F02	Timer 0 Reload High Byte Register (T0RH)	T0RH	FF	99
F03	Timer 0 Reload Low Byte Register (T0RL)	T0RL	FF	99
F04	Timer 0 PWM High Byte Register (T0PWMH)	T0PWMH	00	100
F05	Timer 0 PWM Low Byte Register (T0PWML)	T0PWML	00	100
F06	Timer 0 Control 0 Register (T0CTL0)	T0CTL0	00	101
F07	Timer 0 Control 1 Register (T0CTL1)	T0CTL1	00	102
F08	ADC Timer 0 Capture Register, High Byte	ADCTCAP_H	XX	208
F09	ADC Timer 0 Capture Register, Low Byte	ADCTCAP_L	XX	208
F0A–F1F	Reserved	—	XX	
Pulse-Width Modulator				
F20	PWM Control 0 Register	PWMCTL0	00	75
F21	PWM Control 1 Register	PWMCTL1	00	77
F22	PWM Deadband Register	PWMDB	00	78
F23	PWM Minimum Pulse Width Filter	PWMMPF	00	78
F24	PWM Fault Mask Register	PWMFM	00	79
F25	PWM Fault Status Register	PWMFSTAT	X0X0-0XXXb	80
F26	PWM Input Sample Register	PWMIN	00	82
F27	PWM Output Control Register	PWMOUT	00	83
F28	PWM Fault Control Register	PWMFCTL	00	81
F29	Current Sense ADC Trigger Control Register	PWMSHC	00	83

Note: XX = undefined.

Table 5. Register File Address Map (Continued)

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page No
F2A–B	Reserved	—	XX	
F2C	PWM High Byte Register (PWMH)	PWMH	00	73
F2D	PWM Low Byte Register (PWML)	PWML	00	73
F2E	PWM Reload High Byte Register (PWMRH)	PWMRH	0F	74
F2F	PWM Reload Low Byte Register (PWMRL)	PWMRL	FF	74
F30	PWM 0 High Side Duty Cycle High Byte	PWMH0Dh	00	75
F31	PWM 0 High Side Duty Cycle Low Byte	PWMH0DL	00	75
F32	PWM 0 Low Side Duty Cycle High Byte	PWML0Dh	00	75
F33	PWM 0 Low Side Duty Cycle Low Byte	PWML0DL	00	75
F34	PWM 1 High Side Duty Cycle High Byte	PWMH1DH	00	75
F35	PWM 1 High Side Duty Cycle Low Byte	PWMH1DL	00	75
F36	PWM 1 Low Side Duty Cycle High Byte	PWML1DH	00	75
F37	PWM 1 Low Side Duty Cycle Low Byte	PWML1DL	00	75
F38	PWM 2 High Side Duty Cycle High Byte	PWMH2DH	00	75
F39	PWM 2 High Side Duty Cycle Low Byte	PWMH2DL	00	75
F3A	PWM 2 Low Side Duty Cycle High Byte	PWML2DH	00	75
F3B	PWM 2 Low Side Duty Cycle Low Byte	PWML2DL	00	75
F3C–F3F	Reserved	—	XX	

Note: XX = undefined.

Table 5. Register File Address Map (Continued)

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page No
LIN-UART				
F40	LIN-UART Transmit Data Register	U0TXD	XX	125
	LIN-UART Receive Data Register	U0RXD	XX	125
F41	LIN-UART Status 0 Register	U0STAT0	0000_011Xb	126
F42	LIN-UART Control 0 Register	U0CTL0	00	130
F43	LIN-UART Control 1	U0CTL1	00	132
F44	LIN-UART Mode Select and Status	U0MDSTAT	00	128
F45	LIN-UART Address Compare	U0ADDR	00	136
F46	LIN-UART Baud Rate High Byte	U0BRH	FF	137
F47	LIN-UART Baud Rate Low Byte	U0BRL	FF	137
F48–F5F	Reserved	—	XX	
Inter-Integrated Circuit				
F50	I2C Data Register	I2CDATA	00	181
F51	I2C Interrupt Status Register	I2CISTAT	80	182
F52	I2C Control Register	I2CCTL	00	184
F53	I2C Baud Rate High Byte Register (I2CBRH)	I2CBRH	FF	185
F54	I2C Baud Rate Low Byte Register (I2CBRL)	I2CBRL	FF	186
F55	I2C State Register (I2CSTATE) - Description when DIAG = 0	I2CSTATE	0X	186
	I2C State Register (I2CSTATE) - Description when DIAG = 1	I2CSTATE	00	188
F56	I2C Mode Register	I2CMODE	00	190
F57	I2C Slave Address Register	I2CSLVAD	00	192
F58–F5F	Reserved	—	XX	
Serial Peripheral Interface				
F60	SPI Data Register	SPIDATA	XX	154
F61	SPI Control Register	SPICTL	00	155
F62	SPI Status Register	SPISTAT	01	156
F63	SPI Mode Register	SPIMODE	00	157
F64	SPI Diagnostic State Register	SPIDST	00	158
F65	Reserved	—	XX	
F66	SPI Baud Rate High Byte Register (SPIBRH)	SPIBRH	FF	159

Note: XX = undefined.

Table 5. Register File Address Map (Continued)

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page No
F67	SPI Baud Rate Low Byte Register (SPIBRL)	SPIBRL	FF	159
F68–F6F	Reserved	—	XX	
Analog-to-Digital Converter				
F70	ADC Control Register 0	ADCCTL0	20	201
F71	ADC Raw Data High Byte Register	ADCRD_H	XX	202
F72	ADC Data High Byte Register	ADCD_H	XX	203
F73	ADC Data Low Bits Register	ADCD_L	XX	204
F74	Sample Settling Time Register	ADCSST	1F	205
F75	Sample Time Register	ADCST	A0	206
F76	ADC Clock Prescale Register	ADCCP	00	207
F77–F85	Reserved	—	XX	
Oscillator Control				
F86	Oscillator Control Register	OSCCTL	A0	229
F87	Oscillator Divide Register	OSCDIV	00	231
Trim Control				
F88–F8F	Reserved	—	XX	
Comparator and Op Amp				
F90	Comparator and Op Amp Control Register	CMPOPC	00	196
F91–FBF	Reserved	—	XX	
Interrupt Controller				
FC0	Interrupt Request 0 Register	IRQ0	00	52
FC1	IRQ0 Enable High Bit Register (IRQ0ENH)	IRQ0ENH	00	55
FC2	IRQ0 Enable Low Bit Register (IRQ0ENL)	IRQ0ENL	00	56
FC3	Interrupt Request 1 Register	IRQ1	00	54
FC4	IRQ1 Enable High Bit Register (IRQ1ENH)	IRQ1ENH	00	57
FC5	IRQ1 Enable Low Bit Register (IRQ1ENL)	IRQ1ENL	00	58
FC9–FCE	Reserved	—	XX	
FCF	Interrupt Control Register	IRQCTL	00	59
GPIO Port A				
FD0	Port A Address	PAADDR	00	38
FD1	Port A Control	PACTL	00	39
FD2	Port A Input Data	PAIN	XX	47

Note: XX = undefined.

Table 5. Register File Address Map (Continued)

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page No
FD3	Port A Output Data	PAOUT	00	47
GPIO Port B				
FD4	Port B Address	PBADDR	00	38
FD5	Port B Control	PBCTL	00	39
FD6	Port B Input Data	PBIN	XX	47
FD7	Port B Output Data	PBOUT	00	47
GPIO Port C				
FD8	Port C Address	PCADDR	00	38
FD9	Port C Control	PCCTL	00	39
FDA	Port C Input Data	PCIN	XX	47
FDB	Port C Output Data	PCOUT	00	47
Reset and Watchdog Timer				
FF0	Reset Status and Control Register	RSTSTAT	see Table 10	28
FF1	Reserved	—	XX	
FF2	Watchdog Timer Reload High Byte Register (WDTH)	WDTH	04	63
FF3	Watchdog Timer Reload Low Byte Register (WDTL)	WDTL	00	63
FF4–FF5	Reserved	—	XX	

Note: XX = undefined.

Table 5. Register File Address Map (Continued)

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page No
Trim				
FF6	Trim Bit Address Register	TRMADR	00	224
FF7	Trim Bit Data Register	TRMDR	00	225
Flash Memory Controller				
FF8	Flash Control Register Definitions	FCTL	00	216
FF8	Flash Status Register	FSTAT	00	217
FF9	Flash Page Select Register	FPS	00	218
FF9 (if enabled)	Flash Sector Protect Register	FPROT	00	219
FFA	Flash Frequency High and Low Byte Registers (FFREQH, FFREQL)	FFREQH	00	220
FFB	Flash Frequency Low Byte Register	FFREQL	00	219
eZ8 CPU				
FFC	Flags	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.

Reset and Stop Mode Recovery

The Reset Controller within the Z8FMC16100 Series MCU controls RESET and Stop Mode Recovery operation. In typical operation, the following events cause a RESET to occur:

- Power-On Reset
- Voltage Brown-Out (VBO)
- Watchdog Timer time-out (when configured through the WDT_RES option bit to initiate a Reset)
- External $\overline{\text{RESET}}$ pin assertion
- OCD initiated RESET (OCDCTL[0] set to 1)
- Fault detect logic

When the Z8FMC16100 Series MCU is in STOP Mode, a Stop Mode Recovery is initiated by either of the following occurrences:

- WDT time-out
- GPIO port input pin transition on an enabled Stop Mode Recovery source

Reset Types

The Z8FMC16100 Series MCU provides two different types of reset operation: System Reset and Stop Mode Recovery. The type of reset is a function of both the current operating mode of the Z8FMC16100 Series MCU and the source of the reset. Table 6 lists the types of resets and their operating characteristics.

Table 6. Reset and Stop Mode Recovery Characteristics and Latency

	Reset Characteristics and Latency		
Reset Type	**Peripheral Control Registers**	**eZ8 CPU**	**Reset Latency (Delay)**
System Reset	Reset (as applicable)	RESET	A minimum of 66 IPO cycles.
Stop Mode Recovery	Unaffected, except RSTSRC and OSCCTL registers	Reset	A minimum of 66 IPO cycles.

System Reset

During a system reset, the Z8FMC16100 Series MCU is held in RESET for 66 cycles of internal precision oscillator (IPO). At the beginning of RESET, all GPIO pins are configured as inputs. All GPIO programmable pull-ups are disabled.

At the beginning of a System Reset, the motor control PWM outputs are forced to high-impedance momentarily. When the Option Bits which control the off-state have been properly evaluated the PWM outputs are forced to the programmed off-state.

During RESET, the eZ8 CPU and on-chip peripherals are idle; however, the IPO and WDT oscillator continue to run. During the first 50 clock cycles the internal option bit registers are initialized after which the system clock for the core and peripherals begins operating. The eZ8 CPU and on-chip peripherals remain idle through the next 16 cycles of the system clock after which time the internal reset signal is deasserted.

On RESET, control registers within the Register File which have a defined reset value are loaded with their reset values. Other control registers (including the Flags) and general-purpose RAM are undefined following 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.

Table 7 lists the system reset sources as a function of the operating mode. The text following provides more detailed information about the individual RESET sources.

Note: A POR/VBO event always has priority over all other possible reset sources to ensure a full system reset occurs.

Table 7. System Reset Sources and Resulting Reset Action

Operating Mode	System Reset Source	Action
NORMAL or HALT modes	POR/VBO.	System Reset.
	WDT time-out when configured for reset.	System Reset.
	$\overline{\text{RESET}}$ pin assertion.	System Reset.
	Write OCDCTL[0] to 1.	System Reset except the OCD is not reset.
	Fault detect logic reset.	System Reset.
STOP Mode	Power-on reset/Voltage Brown-Out.	System Reset.
	$\overline{\text{RESET}}$ pin assertion.	System Reset.
	Fault detect logic reset.	System Reset.

Power-On Reset

Each device in the Z8FMC16100 Series MCU contains an internal Power-On Reset (POR) circuit. The POR circuit monitors the supply voltage and holds the device in the Reset state until the supply voltage reaches a safe operating level. After the supply voltage exceeds the POR voltage threshold (V_{POR}) and has stabilized, the POR Counter is enabled and counts 50 cycles of the IPO. At this point, the System Clock is enabled and the POR Counter counts a total of 16 system clock pulses. The device is held in the Reset state until this second POR Counter sequence has timed out. Once the Z8FMC16100 Series MCU exits the Power-On Reset state, the eZ8 CPU fetches the Reset vector. Following POR, the POR status bit in the Reset Status and Control Register is set to 1.

Figure 3 displays the POR operation. For POR threshold voltage (V_{POR}), see the On-Chip Peripheral AC and DC Electrical Characteristics chapter on page 258.

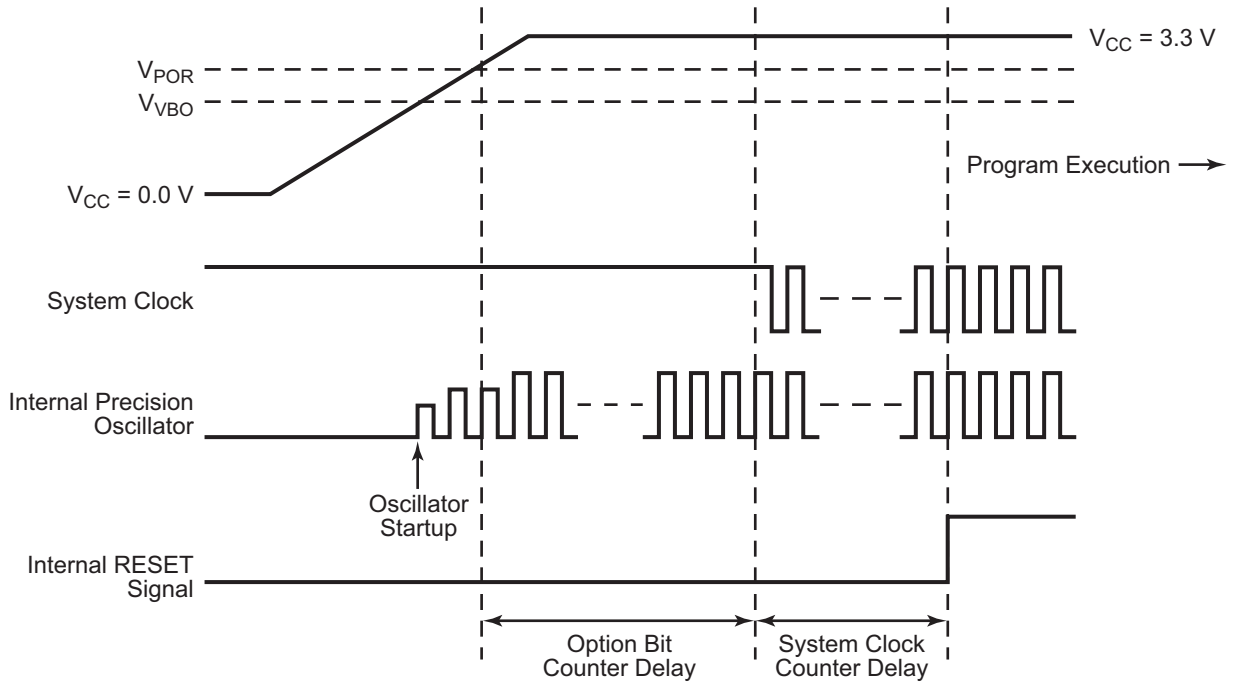


Figure 3. Power-On Reset Operation

Voltage Brown-Out Reset

The Z8FMC16100 Series MCU provides low VBO protection. The VBO circuit senses when the supply voltage drops to an unsafe level (below the VBO threshold voltage) and forces the device into a RESET state. While the supply voltage remains below the POR voltage threshold (V_{POR}), the VBO holds the device in the Reset state.

After the supply voltage again exceeds the POR voltage threshold and stabilized, the device progresses through a full System Reset sequence, as described in the Power-on reset section. Following POR, the POR status bit in the Reset Source Register is set to 1. Figure 4 displays the VBO operation. For VBO and POR threshold voltages (V_{VBO} and V_{POR}), see the On-Chip Peripheral AC and DC Electrical Characteristics chapter on page 258.

The VBO circuit can be either enabled or disabled during STOP Mode. Operation during STOP Mode is controlled by the VBO_AO option bit. For information about configuring VBO_AO, see the Option Bits chapter on page 221.

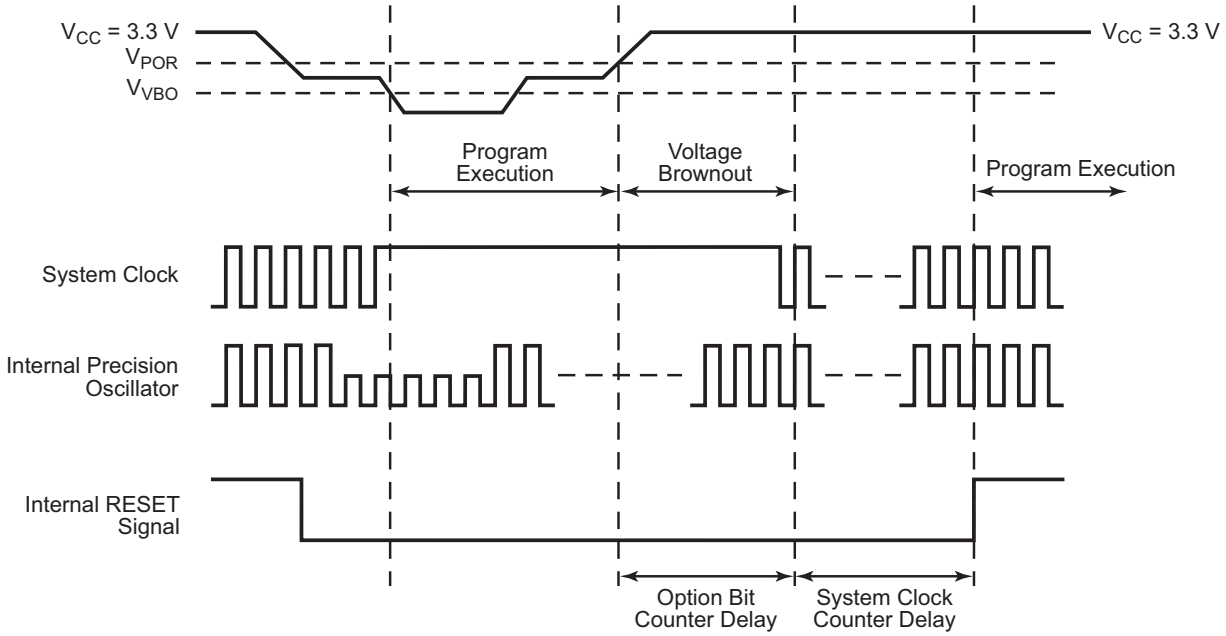


Figure 4. Voltage Brown-Out Reset Operation

Watchdog Timer Reset

If the device is in normal or HALT Mode, the WDT can initiate a System Reset at time-out if the WDT_RES option bit is set to 1. This setting is the default (unprogrammed) setting of the WDT_RES option bit. The WDT status bit in the Reset Status and Control Register is set to signify that the reset was initiated by the WDT.

External Pin Reset

The input-only $\overline{\text{RESET}}$ pin has a Schmitt-triggered input, an internal pull-up, an analog filter and a digital filter to reject noise. When the $\overline{\text{RESET}}$ pin is asserted for at least 4

system clock cycles, the device progresses through the System Reset sequence. While the $\overline{\text{RESET}}$ input pin is asserted Low, the Z8FMC16100 Series MCU device continues to be held 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 16 system clock cycles following $\overline{\text{RESET}}$ pin deassertion. If the $\overline{\text{RESET}}$ pin is released before the System Reset time-out, the $\overline{\text{RESET}}$ pin is driven Low by the chip until the completion of the time-out as described in the next section. In STOP Mode the digital filter is bypassed because the System Clock is disabled.

Following a System Reset initiated by the external $\overline{\text{RESET}}$ pin, the EXT status bit in the Reset Status and Control Register is set to 1.

External Reset Indicator

During System Reset, 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 a Z8FMC16100 Series MCU device to reset other connected components , even if the reset is caused by internal sources such as POR, VBO, or WDT events and as an indication of when the reset sequence completes.

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 6 has elapsed.

On-Chip Debugger Initiated Reset

A System Reset may be initiated through the OCD by setting RST bit of the OCDCTL Register. The OCD is not reset but the rest 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 and Control Register is set.

Fault Detect Logic Reset

Fault detect circuitry exists to detect illegal state changes which may be caused by transient power or electrostatic discharge events. When such a fault is detected, a system reset is forced. Following the system reset, the FLTD bit in the Reset Status and Control Register is set.

Stop Mode Recovery

The STOP Mode is entered by execution of a STOP instruction by the eZ8 CPU. For detailed STOP Mode information, see the Low-Power Modes chapter on page 30. During Stop Mode Recovery, the device is held in reset for 66 cycles of the IPO. Stop Mode Recovery only affects the contents of the Reset Status and Control Register and Oscillator

Control Register. Stop Mode Recovery does not affect any other values in the Register File, including the Stack Pointer, Register Pointer, Flags, peripheral control registers and general-purpose RAM.

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 and Control Register is set to 1. Table 8 lists the Stop Mode Recovery sources and resulting actions.

Table 8. Stop Mode Recovery Sources and Resulting Action

Operating Mode	Stop Mode Recovery Source	Action
Stop Mode	WDT time-out when configured for Reset	Stop Mode Recovery
	WDT time-out when configured for System Exception	Stop Mode Recovery followed by WDT System Exception
	Data transition on any GPIO port pin enabled as a Stop Mode Recovery source	Stop Mode Recovery

Stop Mode Recovery Using Watchdog Timer Time-Out

If the WDT times out during STOP Mode, the device undergoes a SMR sequence. In the Reset Status and Control Register, the WDT and STOP bits are set to 1. If the WDT is configured to generate a System Exception upon time-out, the eZ8 CPU services the WDT System Exception following the normal Stop Mode Recovery sequence.

Stop Mode Recovery Using a GPIO Port Pin Transition

Each of the GPIO port pins may be configured as a Stop Mode Recovery input source. 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. The GPIO Stop Mode Recovery signals are filtered to reject pulses less than 10 ns (typical) in duration. In the Reset Status and Control Register, the STOP bit is set to 1.

Caution: Short pulses on the Port pin can initiate Stop Mode Recovery without initiating an interrupt (if enabled for that pin).

PWM $\overline{\text{Fault0}}$ and $\overline{\text{Reset}}$ Pin Selection

The $\overline{\text{RESET}}$ pin can be set to function as a PWM Fault input. When selected, the $\overline{\text{RESET}}$ input function is disabled. The FLTSEL bit in the Reset Status and Control Register allows

software selection of the $\overline{\text{RESET}}$ pin function. The pin function is selected by writing the the unlock sequence followed by the mode to this register. A software write to the FLTSEL bit overrides the value set by the FLTSEL user option bit.

Reset Control Register Definitions

Writing the 14h, 92h unlock sequence to the Reset Status and Control Register address unlocks access to the FLTSEL bit. The locking mechanism prevents spurious writes to this bit. The following sequence is required to unlock this register and write the FLTSEL bit:

1. Write 14h to the Reset Status and Control Register (RSTSTAT).
2. Write 92h to the Reset Status and Control Register (RSTSTAT).
3. Write the FLTSEL bit.

All steps of the unlock sequence must be written in the order listed.

Reset Status and Control Register

The Reset Status and Control Register, shown in Table 9, records the cause of the most recent Reset or Stop Mode Recovery. All status bits are updated on each Reset or Stop Mode Recovery event. Table 10 indicates the possible states of the Reset status bits following a Reset or Stop Mode Recovery event. The $\overline{\text{RESET}}$ pin function is also select with FLTSEL bit.

Table 9. Reset Status and Control Register (RSTSTAT)

Bit	7	6	5	4	3	2	1	0
Field	POR	STOP	WDT	EXT	FLT	Reserved		FLTSEL
RESET	See Table 10 on page 29							u
R/W	R	R	R	R	R	R		R/W
Address	FF0H							

The FLTSEL bit in this register allows software selection of the $\overline{\text{RESET}}$ pin. The pin function is selected by writing the unlock sequence followed by the mode to this register. A software write to the FLTSEL bit 0 in the RSTSTAT Register overrides the value set by the FLTSEL bit 7 in the User Flash Option2 byte location.

0 = $\overline{\text{RESET}}$/$\overline{\text{Fault0}}$ pin, configured as a $\overline{\text{RESET}}$ input.

1 = $\overline{\text{RESET}}$/$\overline{\text{Fault0}}$ pin, configured as a $\overline{\text{Fault0}}$ input.

Note: If the FLTSEL bit 7 in the Flash Option2 location is programmed to 0 then after reset, the FLTSEL bit 0 in the RSTSTAT is set to 1. If the FLTSEL bit 7 in the Flash Option2 location is unprogrammed (1) then after reset, the FLTSEL bit 0 in the RSTSTAT is set to 0.

Table 10. Reset Status Register Values following Reset

Reset or Stop Mode Recovery Event	POR	STOP	WDT	EXT	FLT
Power-On Reset	1	0	0	0	0
Reset using $\overline{\text{RESET}}$ pin assertion	0	0	0	1	0
Reset using Watchdog Timer time-out	0	0	1	1	0
Reset by OCD writing OCDCTL[0] to 1	1	0	0	1	0
Reset from Fault Detect Logic	0	0	0	1	1
Stop Mode Recovery using GPIO pin transition	0	1	0	0	0
Stop Mode Recovery using WDT time-out	0	1	1	0	0

Note: Additional bits may be set depending on the number of resets simultaneously occurring.

Low-Power Modes

The Z8FMC16100 Series MCU products contain power saving features. The highest level of power reduction is provided by STOP Mode. The next level of power reduction is provided by HALT Mode.

STOP Mode

Executing eZ8 CPU's STOP instruction places the Z8FMC16100 Series MCU into STOP Mode. In STOP Mode, the operating characteristics include:

- Primary crystal oscillator and IPO are stopped; X_{IN} and X_{OUT} pins are driven to V_{SS}
- System clock is stopped
- eZ8 CPU is stopped
- Program counter (PC) stops incrementing
- If enabled for operation during STOP Mode, the WDT and its internal RC oscillator continue to operate
- If enabled for operation in STOP Mode through the associated option bit, the VBO circuit continues to operate
- Comparators and voltage reference operate unless disabled
- 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_{CC} or GND), the VBO and WDT must be disabled. The device can be brought out of STOP Mode using Stop Mode Recovery. For more information, see the Reset and Stop Mode Recovery chapter on page 22.

Caution: To prevent excess current consumption, STOP Mode must not be used if the device is driven with an external clock source.

HALT Mode

Execution of the eZ8 CPU's HALT instruction places the device into HALT Mode. In HALT Mode, the operating characteristics are:

- Primary crystal oscillator is enabled and continues to operate
- System clock is enabled and continues to operate
- eZ8 CPU is stopped
- Program counter (PC) stops incrementing
- WDT's internal RC oscillator continues to operate
- 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 or System Exception
- WDT time-out (System Exception or Reset)
- Power-On Reset
- Voltage Brown-Out Reset
- External $\overline{\text{RESET}}$ pin assertion
- HALT Mode Recovery time is less than 5µs

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

Peripheral-Level Power Control

In addition to the STOP and HALT modes, it is possible to disable unused on-chip analog peripherals of the Z8FMC16100 Series MCU device during operation. Disabling an unused analog peripheral minimizes power consumption. Power consumption of the unused on-chip digital peripherals is automatically minimized when not in use.

Power Control Register 0

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

Table 11. Power Control Register 0 (PWRCTL0)

Bit	7	6	5	4	3	2	1	0
Field	Reserved			VBODIS	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	F80H							

Bit	Description
[7:5]	**Reserved** These bits are reserved and must be programmed to 000.
[4] VBODIS	**VBO Detector Disable** This bit and the VBO_AO option bit must be enabled for the VBO to be active. 0 = VBO Detector is enabled. 1 = VBO Detector is disabled.
[3:0]	**Reserved** These bits are reserved and must be programmed to 0000.

General-Purpose Input/Output

The Z8FMC16100 Series MCU contains general-purpose input/output pins (GPIO) arranged as Ports A–C. Each port contains control and data registers. The GPIO control registers determine data direction, open-drain, output drive current, pull-up and alternate pin functions. Each port pin is individually programmable.

GPIO Port Availability By Device

Table 12 lists the port pins available with each device and package type.

Table 12. Port Availability by Device and Package Type

Package	Port A	Port B	Port C
32-pin	[7:0]	[7:0]	[0]

Caution: All Port B, Port A1, Port C0, PA0 and PA2 are not 5V tolerant. Any voltage above 3.6V will affect the analog functions.

Architecture

Figure 5 displays a simplified block diagram of a GPIO port pin. The figure displays the ability to accommodate alternate functions and variable port current drive strength are not illustrated.

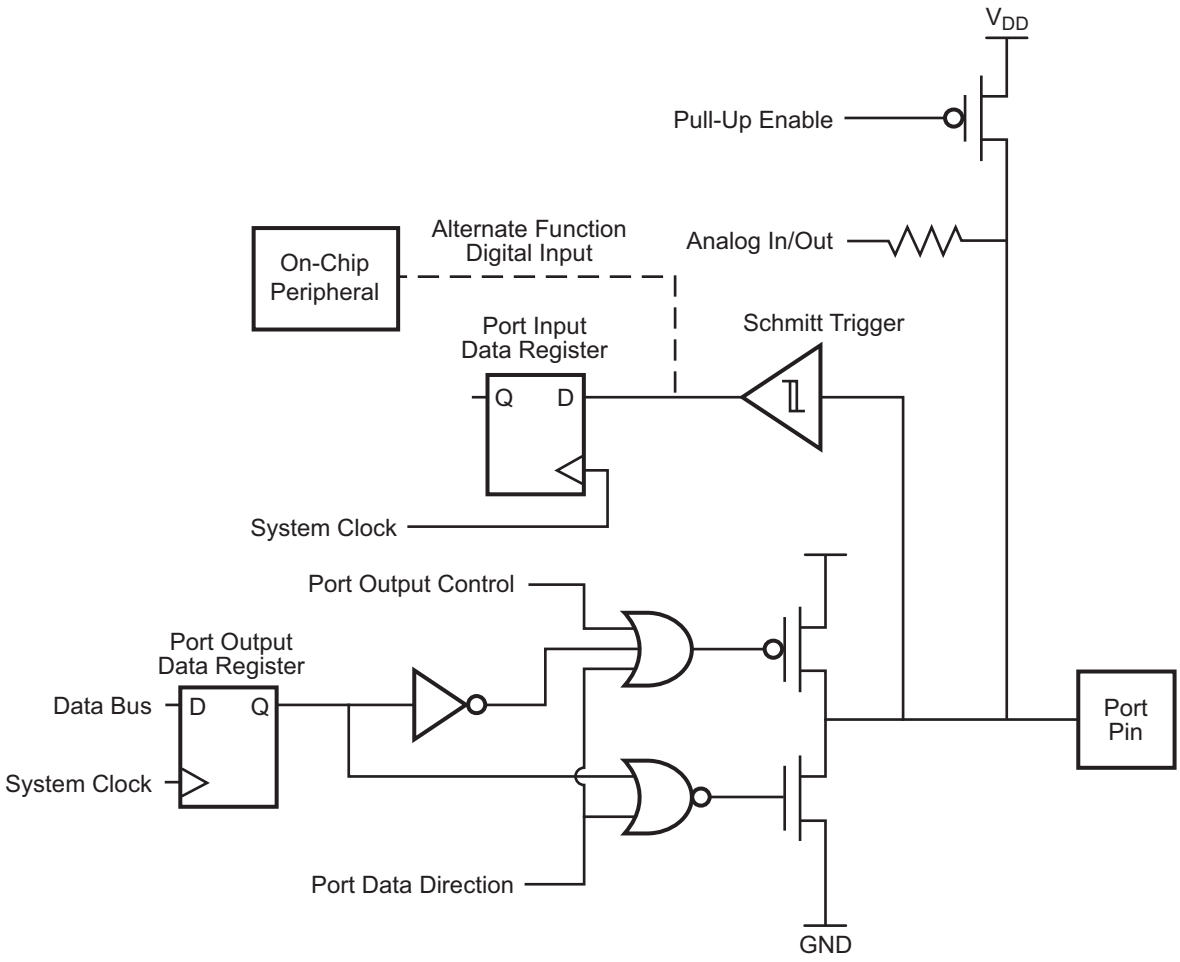


Figure 5. GPIO Port Pin Block Diagram

GPIO Alternate Functions

Many GPIO port pins can be used as both GPIO and to provide access to on-chip peripheral functions, like timers and serial communication devices. The Port A-C Alternate Function subregisters configure these pins for either GPIO or alternate function operation. When a pin is configured for alternate function, control of the port pin direction (input/output) is passed from Port A-C Data Direction registers to the alternate function assigned to this pin. For peripherals with digital input alternate functions (for example, a Timer input T0IN0), selecting the alternate function is not required.

Table 13 lists the alternate functions associated with each port pin and the required alternate function (AF0 and AF1) subregister settings. Enabling an analog alternate function automatically disables the digital Schmitt-trigger input for the associated port input.

Alternate function settings affect the pins direction (output enable) and muxes the output data that come from the peripheral instead of GPIO. The input path coming from the pad to a peripheral is not blocked based on the alternate function setting. The alternate function need not be set for alternate function signals that are used as inputs. The alternate function can be configured as GPIO inputs. When the fault source is not in use, you must disable the input fault source at the peripheral so the fault input does not affect the peripheral.

Table 13. Port Alternate Function Mapping

Port	Pin	Mnemonic	AF0	AF1	Alternate Function Description
Port A	PA7	PA7	0	0	GPIO.
		T0OUT	0	1	Timer 0 output (active Low).
		$\overline{\text{FAULT1}}$	1	0	Fault Input.
		COMPOUT	1	1	Comparator output.
	PA6	PA6	0	0	GPIO.
		SS	0	1	SPI Slave Select.
		CTS	1	0	UART Clear to Send.
		SDA	1	1	I^2C Serial Data.
	PA5	PA5	0	0	GPIO.
		MOSI	0	1	SPI Master Out Slave In.
		TXD	1	0	UART Transmit data.
			1	1	Reserved; do not use.
	PA4	PA4	0	0	GPIO.
		MISO	0	1	SPI Master In Slave Out.
		RXD	1	0	UART Receive data.
			1	1	Reserved; do not use.
	PA3	PA3	0	0	GPIO.
		SCK	0	1	SPI Serial Clock.
		TXDE	1	0	UART Driver Enable.
		SCL	1	1	I^2C serial clock.
	PA2	PA2	0	0	GPIO.
			0	1	Reserved; do not use.
		CINP	1	0	Comparator Positive Input.
			1	1	Reserved; do not use.

Table 13. Port Alternate Function Mapping (Continued)

Port	Pin	Mnemonic	AF0	AF1	Alternate Function Description
Port A	PA1	PA1	0	0	GPIO.
			0	1	Reserved; do not use.
		OPINP and CINN	1	0	Op amp positive input and comparator negative input.
			1	1	Reserved; do not use.
	PA0	PA0	0	0	GPIO.
			0	1	Reserved; do not use.
		OPINN	1	0	Op amp negative input.
			1	1	Reserved; do not use.
Port B	PB7	PB7	0	0	GPIO.
			0	1	Reserved; do not use.
		ANA7	1	0	ADC analog input 7.
			1	1	Reserved; do not use.
	PB6	PB6	0	0	GPIO.
			0	1	Reserved; do not use.
		ANA6	1	0	ADC analog input 6.
			1	1	Reserved; do not use.
	PB5	PB5	0	0	GPIO.
			0	1	Reserved; do not use.
		ANA5	1	0	ADC analog input 5.
			1	1	Reserved; do not use.
	PB4	PB4	0	0	GPIO.
			0	1	Reserved; do not use.
		ANA4	1	0	ADC analog input 4 (also CINN, if CPSEL=1).
			1	1	Reserved; do not use.
	PB3	PB3	0	0	GPIO.
		PB3INT	0	1	GPIO with edge interrupt.
		ANA3 and OPOUT	1	0	ADC analog input 3 and op amp output.
			1	1	Reserved; do not use.

Table 13. Port Alternate Function Mapping (Continued)

Port	Pin	Mnemonic	AF0	AF1	Alternate Function Description
Port B	PB2	PB2	0	0	GPIO/Timer 0 input 2.
		PB2INT	0	1	GPIO/Timer 0 input 2—edge interrupt enabled.
		ANA2	1	0	ADC analog input 2.
		T0IN2	1	1	Timer 0 input 2; dedicated input.
	PB1	PB1	0	0	GPIO/Timer 0 input 1.
		PB1INT	0	1	GPIO/Timer 0 input 1—edge interrupt enabled.
		ANA1	1	0	ADC analog input 1.
			1	1	Reserved; do not use.
	PB0	PB0	0	0	GPIO/Timer 0 input 0.
		PB0INT	0	1	GPIO/Timer 0 input 0—edge interrupt enabled.
		ANA0	1	0	ADC analog input 0.
			1	1	Reserved; do not use.
Port C	PC0	PC0	0	0	GPIO.
			0	1	Reserved; do not use.
		T0OUT	1	0	Timer 0 output.
			1	1	Reserved; do not use.

GPIO Interrupts

Many GPIO port pins can be used as interrupt sources. The Port A[7:0] pins can be configured to generate an interrupt request on either the rising edge or falling edge of the pin input signal. The Port B[3:0] pins can be configured to generate an interrupt request on both the rising and falling edges of the pin input signal. For Port A, the GPIO interrupt edge selection is controlled by the Interrupt Edge Select subregister. Enabling and disabling of the Port Interrupts is handled in the Interrupt Controller. Port B[3:0], with dual edge interrupt capability, is selected by AF1, AF0. For more information, see the Interrupt Controller chapter on page 48.

GPIO Control Register Definitions

Four registers for each Port provide access to GPIO control, input data and output data. Table 14 lists these Port registers. Use the Port A–C Address and Control registers together to provide access to subregisters for Port configuration and control.

Table 14. GPIO Port Registers and Subregisters

Port Register Mnemonic	Port Register Name
PxADDR	Port A–C Address Register—selects subregisters.
PxCTL	Port A–C Control Register—provides access to subregisters.
PxIN	Port A–C Input Data Register.
PxOUT	Port A–C Output Data Register.
Port Subregister Mnemonic	**Port Register Name**
PxDD	Data Direction
PxAF0	Alternate Function 0
PxAF1	Alternate Function 1—Ports A and B only.
PxOC	Output Control (open-drain).
PxHDE	High Drive Enable.
PxSMRE	Stop Mode Recovery Source Enable.
PxPUE	Pull-Up Enable.
PxIRQES	Interrupt Edge Select—Ports A and C only.
IRQPSEL	Interrupt Port Select—Port A only.

Port A-C Address Registers

The Port A–C Address registers, shown in Table 15, select the GPIO port functionality accessible through the Port A–C Control registers. The Port A–C Address and Control registers combine to provide access to all GPIO port control.

Table 15. Port A–C GPIO Address Registers (PxADDR)

Bit	7	6	5	4	3	2	1	0
Field	PADDR[7:0]							
RESET	00H							
R/W	R/W							
Address	FD0H, FD4H, FD8H							

Table 16 lists the Port Control subregisters accessible through the Port A–C Control registers.

Table 16. Port Control Subregisters by Port Address

Port Address	Port Control Subregisters
00h	No function; provides some protection against accidental port reconfiguration.
01h	Data direction.
02h	Alternate Function 0
03h	Output Control (open-drain).
04h	High Drive Enable.
05h	Stop Mode Recovery source enable.
06h	Pull-up enable.
07h	Alternate Function 1—Ports A and Port B only.
08h	Interrupt Edge Select—Port A and Port C only
09h	Port Interrupt Select Register—Port A only
0Ah–FFh	No Function

Port A–C Control Registers

The Port A–C Control registers (PxCTL) set the GPIO port operation. The value in the corresponding Port A–C Address Register determines the control subregisters accessible using the Port A–C Control registers, shown in Table 17. The Port Control registers provide access to all subregisters that configure GPIO port operation.

Table 17. Port A–C Control Registers (P*x*CTL)

Bit	7	6	5	4	3	2	1	0
Field	PCTL							
RESET	00H							
R/W	R/W							
Address	FD1H, FD5H, FD9H							

Port A–C Data Direction Subregisters

The Port A–C Data Direction subregisters, shown in Table 18, are accessed through the Port A–C Control registers by writing 01H to the Port A–C Address registers.

In each of these three data direction subregisters, bits [7:0] control the direction of the associated port pin. Port Alternate Function operation overrides the Data Direction Register setting.

If the value of bit is 0, the direction is output and data in the Port A–C Output Data registers is driven onto the port pin.

If the value of bit is 1, the direction is input. The port pin is sampled, the value is written into the Port A–C Input Data registers and the output driver is tristated.

Table 18. Port A–C Data Direction Subregisters

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	See note.							

Note: If the Port A Address Register contains address 01H, this register is accessible via the Port A and Port C Control Registers.

Bit	Description
[7:0] DDx	**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–C Output Data Register is driven onto the port pin. 1 = Input. The port pin is sampled and the value written into the Port A–C Input Data Register. The output driver is tristated.

Note: x indicates register bits in the range 7–0.

Port A–B Alternate Function 0 Subregisters

The Port A–B Alternate Function subregisters, shown in Table 19, are accessed through Port A–B Control registers by writing 02H to the Port A–B Address registers. The Port A–B Alternate Function subregisters select the alternate functions for the selected pins. To determine the alternate function associated with each port pin, see the GPIO Alternate Functions chapter on page 34.

Caution: You must not enable alternate function for GPIO port pins which do not have an associated alternate function. Failure to follow this guideline may result in unpredictable operation.

Table 19. Port A–B Alternate Function 0 Subregisters

Bit	7	6	5	4	3	2	1	0
Field	AF0_7	AF0_6	AF0_5	AF0_4	AF0_3	AF0_2	AF0_1	AF0_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	See note.							

Note: If the Port A Address Register contains address 02H, this register is accessible via the Port A and Port C Control Registers.

Bit	Description
[7:0] AF0_x	**Port Alternate Function 0 Select** 0 = The alternate function 0 function is not selected. 1 = The alternate function 0 function is selected.

Note: x indicates register bits in the range 7–0.

Port A–C Output Control Subregisters

The Port A–C Output Control subregisters, shown in Table 20, are accessed through the Port A–C Control registers by writing 03H to the Port A–C Address registers. Setting the bits in the Port A–C Output Control subregisters to 1 configures the specified port pins for open-drain operation. These subregisters affect the pins directly, as a result, alternate functions are also affected.

Table 20. Port A–C Output Control Subregisters

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	See note.							

Note: If the Port A Address Register contains address 03H, this register is accessible via the Port A and Port C Control Registers.

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

Note: x indicates register bits in the range 7–0.

Port A–C High Drive Enable Subregisters

The Port A–C High Drive Enable subregisters, shown in Table 21, are accessed through the Port A-C Control registers by writing 04H to the Port A–C Address registers. Setting the bits in the Port A–C High Drive Enable subregisters to 1 configures the specified port pins for high current output drive operation. The Port A–C High Drive Enable subregisters affect the pins directly. As a result, alternate functions are also affected.

Table 21. Port A–C High Drive Enable Subregisters

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	See note.							

Note: If the Port A Address Register contains address 04H, this register is accessible via the Port A and Port C Control Registers.

Bit	Description
[7:0] PHDE	**Port High Drive Enable** 0 = The Port pin is configured for standard output current drive. 1 = The Port pin is configured for high output current drive.

Port A–C Stop Mode Recovery Source Enable Subregisters

The Port A–C Stop Mode Recovery Source Enable subregisters, shown in Table 22, are accessed through the Port A–C Control registers by writing 05H to the Port A-C Address registers. Setting the bits in the Port A–C 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 22. Port A–C STOP Mode Recovery Source Enable Subregisters

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	See note.							

Note: If the Port A Address Register contains address 05H, this register is accessible via the Port A and Port C Control Registers.

Bit	Description
[7:0] PSMREx	**Port STOP Mode Recovery Source Enable** 0 = The Port pin is not configured as a STOP Mode Recovery source. Transitions on this pin during STOP Mode do not initiate 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 STOP Mode Recovery.

Note: x indicates register bits in the range 7–0.

Port A–C Pull-Up Enable Subregisters

The Port A–C Pull-Up Enable subregisters, shown in Table 23, are accessed through the Port A–C Control registers by writing 06H to the Port A–C Address registers. Setting the bits in the Port A–C Pull-Up Enable subregisters enables a weak internal resistive pull-up on the specified port pins.

Table 23. Port A–C Pull-Up Enable Subregisters

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	See note.							

Note: If the Port A Address Register contains address 06H, this register is accessible via the Port A and Port C Control Registers.

Bit	Description
[7:0] PPUEx	**Port Pull-Up Enable** 0 = The weak pull-up on the Port pin is disabled. 1 = The weak pull-up on the Port pin is enabled.

Note: x indicates register bits in the range 7–0.

Port A Interrupt Edge Select Subregister

The Interrupt Edge Select (IRQES) Subregister, shown in Table 24, determines whether an interrupt is generated for the rising edge or falling edge on the selected GPIO Port A input pin.

Table 24. Interrupt Edge Select Subregister (IRQES)

Bit	7	6	5	4	3	2	1	0
Field	Reserved				IES3	IES2	IES1	IES0
RESET	0	0	0	0	0	0	0	0
R/W	R	R	R	R	R/W	R/W	R/W	R/W
Address	See note.							

Note: If the Port A Address Register contains address 08H, this register is accessible via the Port A and Port C Control Registers.

Bit	Description
[3:0] IESx	**Interrupt Edge Select x** 0 = An interrupt request is generated on the falling edge of the PAx input. 1 = An interrupt request is generated on the rising edge of the PAx input, in which x indicates the specific GPIO Port pin number (0 through 7).

Note: x indicates register bits in the range 3–0.

Interrupt Port Select Register

The Interrupt Port Select Subregister, shown in Table 25, is used to select which Port A pins are used as interrupts.

Table 25. Interrupt Port Select Subregister (IRQPS)

Bit	7	6	5	4	3	2	1	0
Field	Reserved				PA73SEL	PA62SEL	PA51SEL	PA40SEL
RESET	0	0	0	0	0	0	0	0
R/W	R	R	R	R	R/W	R/W	R/W	R/W
Address	See note.							

Note: If the Port A Address Register contains address 09H, this register is accessible via the Port A and Port C Control Registers.

Bit	Description
[3:0] PAxxSEL	**Interrupt Port Select x** 0 = An interrupt request is generated on PAx, where x indicates (0 through 3). 1 = An interrupt request is generated on PAx, where x indicates (4 through 7).

Note: x indicates register bits in the range 3–0.

Port B Alternate Function 1 Subregisters

The Port B Alternate Function subregisters, shown in Table 26, are accessed through the Port B Control Register by writing 08H to the Port B Address Register. The Port B Alternate Function subregister selects the alternate functions for the selected pins. To determine the alternate function associated with each port pin, see the GPIO Alternate Functions section on page 34.

Caution: Do not enable alternate function for GPIO port pins which do not have an associated alternate function. Failure to follow this guideline may result in unpredictable operation.

Table 26. Port A–B Alternate Function 1 Subregisters

Bit	7	6	5	4	3	2	1	0
Field	AF1_7	AF1_6	AF1_5	AF1_4	AF1_3	AF1_2	AF1_1	AF1_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	See note.							

Note: If the Port A Address Register contains address 07H, this register is accessible via the Port A and Port C Control Registers.

Bit	Description
[7:0] AF1_x	**Port Alternate Function 1 Select** 0 = The alternate function 1 function is not selected. 1 = The alternate function 1 function is selected.

Note: x indicates register bits in the range 7–0.

Port A-C Input Data Registers

Reading from the Port A–C Input Data registers, shown in Table 27, return the sampled values from the corresponding port pins. The Port A–C Input Data registers are read-only. Sampled data are from the corresponding port pin input.

Table 27. Port A-C Input Data Registers (PxIN)

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	FD2H, FD6H, FDAH							

Note: X = Undefined.

Bit	Description
[7:0] PIN*x*	**Port Input Data** 0 = Input data is a logical 0 (Low). 1 = Input data is a logical 1 (High).

Note: x indicates register bits in the range 7–0.

Port A–C Output Data Registers

The Port A–C Output Data registers, shown in Table 28, write output data to the pins. These bits contain the data to be driven out from 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.

Table 28. Port A-C 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	FD3H, FD7H, FDBH							

Bit	Description
[7:0] POUTx	**Port Output Data x** 0 = Drive is a logical 0 (Low). 1 = Drive is 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.

Note: x indicates register bits in the range 7–0.

Interrupt Controller

The interrupt controller on the Z8FMC16100 Series MCU prioritizes the system exceptions and interrupt requests from the on-chip peripherals and the GPIO port pins. The features of the interrupt controller include:

- Multiple GPIO interrupts
- Interrupts for on-chip peripherals
- Non-maskable system exceptions
- Three levels of individually programmable interrupt priority
- 20 sources of interrupts for the interrupt controller, 9 of the sources can be configured from GPIO pins

System exceptions (SEs) and interrupt requests (IRQs) allow peripheral devices to suspend CPU operation in an orderly manner and force the CPU to start a service routine. Interrupt service routines are involved with the data exchange, 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 for download at www.zilog.com.

Interrupt and System Exception Vector Listing

Table 29 lists the SEs and the interrupts in order of priority. Reset and system exceptions always have priority over interrupts. The system exception and interrupt vectors are 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: Port interrupts are only available in those packages which support the associated port pins.

Table 29. Reset, System Exception and Interrupt Vectors in Order of Priority

Priority	Program Memory Vector Base Address	Programmable Priority?	Interrupt or Trap Source
Reset and System Exceptions			
	0002h	No	Reset
	0004h	No	Watchdog Timer (see the Watchdog Timer section on page 60)
	003AH	No	Primary Oscillator Fail Trap
	003CH	No	Watchdog Timer Oscillator Fail Trap
	0006h	No	Illegal Instruction Trap
Interrupts (maskable)			
Highest	0008h	Yes	PWM Timer
	000Ah	Yes	PWM Fault
	000Ch	Yes	ADC
	000Eh	Yes	Comparator output rising and falling edge
	0010h	Yes	Timer 0
	0012H	Yes	UART 0 receiver
	0014H	Yes	UART 0 transmitter
	0016H	Yes	SPI
	0018H	Yes	I^2C
	001AH	Yes	Reserved
	001CH	Yes	Port C0 with selectable rising or falling input edge
	001EH	Yes	Port B[3:0] rising and fall input edge
	0020h	Yes	Port A7/A3 with selectable rising or falling input edge
	0022H	Yes	Port A6/A2 with selectable rising or falling input edge
	0024H	Yes	Port A5/A1 with selectable rising or falling input edge
	0026H	Yes	Port A4/A0 with selectable rising or falling input edge
Lowest	0028H–0039H	N/A	Reserved

Architecture

Figure 6 displays a block diagram of the Interrupt Controller.

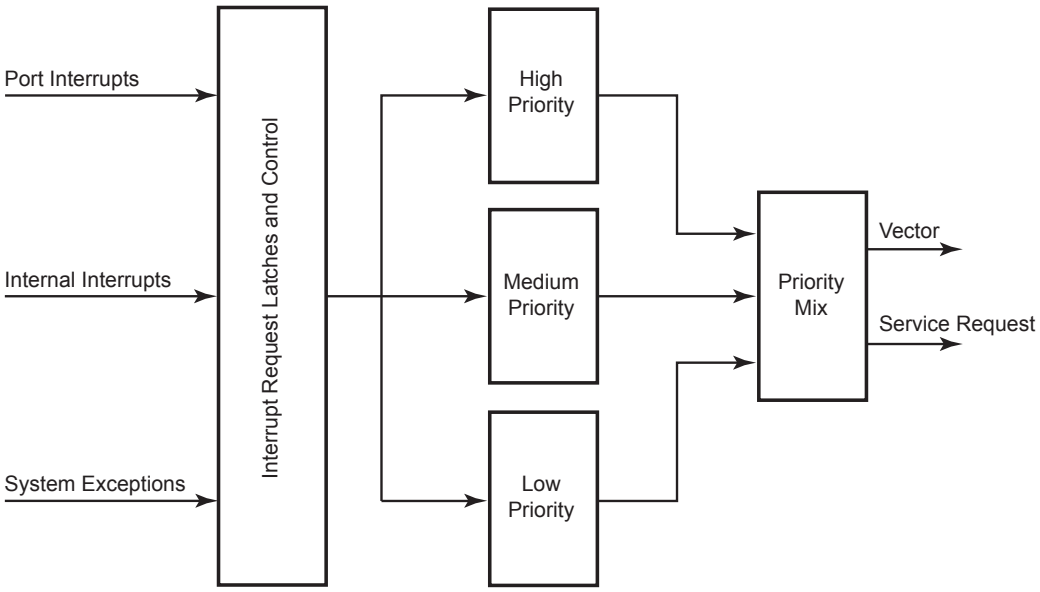


Figure 6. Interrupt Controller Block Diagram

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 a Return from Interrupt (IRET) instruction
- Writing a 1 to the IRQE bit in the Interrupt Control Register

Interrupts are globally disabled by any of the following actions:

- Execution of a Disable Interrupt (DI) instruction
- eZ8 CPU acknowledgement of an interrupt service request from the interrupt controller
- Writing a 0 to the IRQE bit in the Interrupt Control Register
- Reset

- Execution of a Trap instruction
- Illegal Instruction trap

System Exceptions

The Z8FMC16100 Series MCU supports multiple system exceptions. System exceptions are generated for the following events:

- Illegal Instruction trap
- Watchdog Timer interrupt
- Watchdog Timer RC oscillator failure
- Primary oscillator failure

System exceptions, excluding the WDT interrupt, are non-maskable and therefore cannot be disabled by the interrupt controller (setting IRQE to 0 has no effect).

Interrupt Vectors and Priority

Caution: The interrupt controller supports three levels of interrupt priority. Level 3 interrupts are always higher priority than Level 2 interrupts. Level 2 interrupts are always 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 29.

Interrupt Assertion

When an interrupt request occurs, the corresponding bit in the Interrupt Request Register is set. This bit is automatically cleared when the eZ8 CPU vectors to the Interrupt Service Routine (ISR). Writing a 0 to the corresponding bit in the Interrupt Request Register also clears the interrupt request.

If an interrupt is disabled, software polls the appropriate interrupt request register bit and clear the bit directly. The following style of coding to clear bits in the Interrupt Request registers is not recommended. All incoming interrupts that are received between execution of the first LDX command and the last LDX command are lost.

The following code segment is an example of a poor coding style which results in lost interrupt requests:

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

```
Q0, r0
```

To avoid missing interrupts, Zilog® recommends the following style of coding to clear bits in the Interrupt Request 0 Register:

```
ANDX IRQ0, MASK
```

Software Interrupt Assertion

Program code can generate interrupts directly. Writing a 1 to the appropriate bit in the Interrupt Request Register triggers an interrupt (assuming that interrupt is enabled). This bit is automatically cleared when the eZ8 CPU vectors to the ISR.

The following style of coding to generate software interrupts by setting bits in the Interrupt Request registers is not recommended. All incoming interrupts that are received between execution of the first LDX command and the last LDX command are lost.

The following code segment is an example of a poor coding style that results in lost interrupt requests:

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

To avoid missing interrupts, Zilog® recommends the following style of coding to set bits in the Interrupt Request registers:

```
ORX IRQ0, MASK
```

Interrupt Control Register Definitions

The interrupt control registers enable individual interrupts, set interrupt priorities and indicate interrupt requests.

Interrupt Request 0 Register

The Interrupt Request 0 (IRQ0) Register, shown in Table 30, 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 reads the Interrupt Request 0 Register to determine if any interrupt requests are pending.

Table 30. Interrupt Request 0 Register (IRQ0)

Bit	7	6	5	4	3	2	1	0
Field	PWMI	FLTI	ADCI	CMPI	T0I	U0RXI	U0TXI	SPII
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	FC0H							

Bit	Description
[7] PWMI	**PWM Timer Interrupt Request** 0 = No interrupt request is pending for the PWM. 1 = An interrupt request from the PWM is awaiting service.
[6] FLTI	**Fault Interrupt Request** The fault interrupt is generated in the PWM module and originates from the Fault0 pin, Fault1 pin or the Comparator output. An interrupt enable for each of these sources exists in the PWM module. 0 = No Fault interrupt request is pending. 1 = A Fault interrupt request is awaiting service.
[5] ADCI	**ADC Interrupt Request** 0 = No interrupt request is pending for the Analog-to-Digital Converter. 1 = An interrupt request from the Analog-to-Digital Converter is awaiting service.
[4] CMPI	**Comparator Interrupt Request** 0 = No interrupt request is pending for the Comparators. 1 = An interrupt request from the Comparators is awaiting service.
[3] T0I	**Timer 0 Interrupt Request** 0 = No interrupt request is pending for Timer 0. 1 = An interrupt request from Timer 0 is awaiting service.
[2] U0RXI	**UART 0 Receiver Interrupt Request** 0 = No interrupt request is pending for the UART 0 receiver. 1 = An interrupt request from the UART 0 receiver is awaiting service.
[1] U0TXI	**UART 0 Transmitter Interrupt Request** 0 = No interrupt request is pending for the UART 0 transmitter. 1 = An interrupt request from the UART 0 transmitter is awaiting service.
[0] SPII	**SPI Interrupt Request** 0 = No interrupt request is pending for the SPI. 1 = An interrupt request from the SPI is awaiting service.

Interrupt Request 1 Register

The Interrupt Request 1 (IRQ1) Register, shown in Table 31, 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 the interrupts are globally enabled (vectored interrupts), the interrupt controller passes an interrupt request to the eZ8 CPU. If the interrupts are globally disabled (polled interrupts), the eZ8 CPU reads the Interrupt Request 1 Register to determine if any interrupt requests are pending.

Table 31. Interrupt Request 1 Register (IRQ1)

Bit	7	6	5	4	3	2	1	0
Field	I2CI	Reserved	PC0I	PBI	PA73I	PA62I	PA51I	PA40I
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	FC3H							

Bit	Description
[7] I2CI	**I^2C Interrupt Request** 0 = No interrupt request is pending for I2C. 1 = An interrupt request from I2C is awaiting service.
[5] PC0I	**PC0 Interrupt Request** Logic in the Port C GPIO module selects either the rising or falling edge. 0 = No interrupt request is pending for PC0. 1 = An interrupt request from PC0 is awaiting service.
[4] PBI	**PB3–PB0 Interrupt Request** 0 = No interrupt request is pending for any PB3–PB0. 1 = An interrupt request from PB3–PB0 is awaiting service.
[3] PA73I	**PA7 or PA3 Interrupt Request** Logic in the Port A GPIO module selects either PA7 or PA3 and either rising or falling edge. 0 = No interrupt request is pending for PA7 or PA3 1 = An interrupt request from PA7 or PA3 is awaiting service.
[2] PA62I	**PA6 or PA2 Interrupt Request** Logic in the Port A GPIO module selects either PA6 or PA2 and either rising or falling edge. 0 = No interrupt request is pending for PA6 or PA2 1 = An interrupt request from PA6 or PA2 is awaiting service.
[1] PA51I	**PA5 or PA1 Interrupt Request** Logic in the Port A GPIO module selects either PA5 or PA1 and either rising or falling edge. 0 = No interrupt request is pending for PA5 or PA1 1 = An interrupt request from PA5 or PA1 is awaiting service.

Bit	Description (Continued)
[0] PA40I	**PA4 or PA0 Interrupt Request** Logic in the Port A GPIO module selects either PA4 or PA0 and either rising or falling edge. 0 = No interrupt request is pending for PA4 or PA0 1 = An interrupt request from PA4 or PA0 is awaiting service.

IRQ0 Enable High and Low Bit Registers

The IRQ0 Enable High and Low Bit registers, shown in Tables 33 and 34, form a priority-encoded enabling function for interrupts in the Interrupt Request 0 Register. Priority is generated by setting bits in each register. Table 32 indicates the priority control for IRQ0.

Table 32. IRQ0 Enable and Priority Encoding

IRQ0ENH[*x*]	IRQ0ENL[*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 in the range 7–0.			

Table 33. IRQ0 Enable High Bit Register (IRQ0ENH)

Bit	7	6	5	4	3	2	1	0
Field	PWMENH	FLTENH	ADCENH	CMPENH	T0ENH	U0RENH	U0TENH	SPIENH
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	FC1H							

Bit	Description
[7] PWMENH	**PWM Interrupt Request Enable High Bit**
[6] FLTENH	**Fault Interrupt Request Enable High Bit**
[5] ADCENH	**ADC Interrupt Request Enable High Bit**
[4] CMPENH	**Comparator Interrupt Request Enable High Bit**

Bit	Description (Continued)
[3] T0ENH	**Timer 1 Interrupt Request Enable High Bit**
[2] U0RENH	**UART 0 Receive Interrupt Request Enable High Bit**
[1] U0TENH	**UART 0 Transmit Interrupt Request Enable High Bit**
[0] SPIENH	**SPI Interrupt Request Enable High Bit**

Table 34. IRQ0 Enable Low Bit Register (IRQ0ENL)

Bit	7	6	5	4	3	2	1	0
Field	PWMENL	FLTENL	ADCENL	CMPENL	T0ENL	U0RENL	U0TENL	SPIENL
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	FC2H							

Bit	Description
[7] PWMENL	**PWM Interrupt Request Enable Low Bit**
[6] FLTENL	**Fault Interrupt Request Enable Low Bit**
[5] ADCENL	**ADC Interrupt Request Enable Low Bit**
[4] CMPENL	**Comparator Interrupt Request Enable Low Bit**
[3] T0ENL	**Timer 1 Interrupt Request Enable Low Bit**
[2] U0RENL	**UART 0 Receive Interrupt Request Enable Low Bit**
[1] U0TENL	**UART 0 Transmit Interrupt Request Enable Low Bit**
[0] SPIENL	**SPI Interrupt Request Enable Low Bit**

IRQ1 Enable High and Low Bit Registers

The IRQ1 Enable High and Low Bit registers, shown in Tables 36 and 37, form a priority encoded enabling for interrupts in the Interrupt Request 1 Register. Priority is generated by setting bits in each register. Table 35 describes the priority control for IRQ1.

Table 35. 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 in the range 7–0.

Table 36. IRQ1 Enable High Bit Register (IRQ1ENH)

Bit	7	6	5	4	3	2	1	0
Field	I2CENH	Reserved	PC0ENH	PBENH	PA73ENH	PA62ENH	PA51ENH	PA40ENH
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	FC4H							

Bit	Description
[7] I2CENH	**I^2C Interrupt Request Enable High Bit**
[6]	**Reserved** This bit is reserved and must be programmed to 0.
[5] PC0ENH	**Port C0Interrupt Request Enable High Bit**
[4] PBENH	**Port B[3:0] Interrupt Request Enable High Bit**
[3] PA73ENH	**Port A73 Interrupt Request Enable High Bit**
[2] PA62ENH	**Port A62 Interrupt Request Enable High Bit**
[1] PA51ENH	**Port A51 Interrupt Request Enable High Bit**
[0] PA40ENH	**Port A40 Interrupt Request Enable High Bit**

Table 37. IRQ1 Enable Low Bit Register (IRQ1ENL)

Bit	7	6	5	4	3	2	1	0
Field	I2CENL	Reserved	PC0ENL	PBENL	PA73ENL	PA62ENL	PA51ENL	PA40ENL
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	FC5H							

Bit	Description
[7] I2CENL	**I^2C Interrupt Request Enable Low Bit**
[6]	**Reserved** This bit is reserved and must be programmed to 0.
[5] PC0ENL	**Port C0 Interrupt Request Enable Low Bit**
[4] PBENL	**Port B[3:0] Interrupt Request Enable Low Bit**
[3] PA73ENL	**Port A73 Interrupt Request Enable Low Bit**
[2] PA62ENL	**Port A62 Interrupt Request Enable Low Bit**
[1] PA51ENL	**Port A51 Interrupt Request Enable Low Bit**
[0] PA40ENL	**Port A40 Interrupt Request Enable Low Bit**

Interrupt Control Register

The Interrupt Control (IRQCTL) Register, shown in Table 38, contains the Master Enable Bit (IRQE) for all interrupts.

Table 38. 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
R/W	R/W	R	R	R	R	R	R	R
Address	FCFH							

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

Watchdog Timer

The Watchdog Timer (WDT) helps protect against corrupted or unreliable software and other system-level problems which may place the Z8FMC16100 Series MCU into unsuitable operating states. The WDT includes the following features:

- On-chip RC oscillator
- A selectable time-out response: Reset or System Exception
- 16-bit programmable time-out value

Operation

The WDT is a retriggerable one-shot timer that resets or interrupts the Z8FMC16100 Series MCU when the WDT reaches its terminal count. The WDT uses its own dedicated on-chip RC oscillator as its clock source. The WDT has only two modes of operation—ON and OFF. Once enabled, it always counts and must be refreshed to prevent a time-out. An enable can be performed by executing the WDT instruction or by setting the WDT_AO option bit. The WDT_AO bit enables the WDT to operate all the time, even if a WDT instruction has not been executed.

To minimize power consumption, the RC oscillator can be disabled. The RC oscillator is disabled by clearing the WDTEN bit in the Oscillator Control Register. If the RC oscillator is disabled, the WDT will not operate.

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 by the following equation.

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

In this equation, the WDT reload value is assigned by {WDTH[7:0], WDTL[7:0]} and the typical Watchdog Timer RC oscillator frequency is 10 kHz. You must consider system requirements while selecting the time out delay. Table 39 provides information about approximate time-out delays for the default and maximum WDT reload values.

Table 39. Watchdog Timer Approximate Time-Out Delays

WDT Reload Value (Hex)	WDT Reload Value (Decimal)	Approximate Time-Out Delay (with 10 kHz Typical WDT Oscillator Frequency)	
		Typical	Description
0400	1024	102 ms	Reset default value time-out delay.
FFFF	65,536	6.55 s	Maximum time-out delay.

Watchdog Timer Refresh

When first enabled, the WDT is loaded with the value in the Watchdog Timer 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 Watchdog Timer Reload registers. Counting resumes following the reload operation.

When the Z8FMC16100 Series MCU is operating in DEBUG Mode (through the OCD), the WDT is continuously refreshed to prevent spurious WDT time-outs.

Watchdog Timer Time-Out Response

The WDT times out when the counter reaches 0000h. A time-out of the WDT generates either a system exception or a Reset. The WDT_RES option bit determines the time-out response of the WDT. For information about programming the WDT_RES option bit, see the Option Bits chapter on page 221.

WDT System Exception in Normal Operation

If configured to generate a system exception when a time-out occurs, the WDT issues an exception request to the interrupt controller. The eZ8 CPU responds to the request by fetching the System Exception vector and executing code from the vector address. After time-out and system exception generation, the WDT is reloaded automatically and continues counting.

WDT System Exception in STOP Mode

If configured to generate a system exception when a time-out occurs and the Z8FMC16100 Series MCU is in STOP Mode, the WDT automatically initiates a Stop Mode Recovery and generates a system exception request. Both the WDT status bit and the STOP bit in the Reset Status and Control Register are set to 1 following WDT time-out in STOP Mode. For more information, see the Reset and Stop Mode Recovery chapter on page 22.

Following completion of the Stop Mode Recovery the eZ8 CPU responds to the system exception request by fetching the System Exception vector and executing code from the vector address.

WDT Reset in Normal Operation

If configured to generate a Reset when a time-out occurs, the WDT forces the device into the Reset state. The WDT status bit in the Reset Status and Control Register is set to 1. For more information about Reset and the WDT status bit, see the Reset and Stop Mode Recovery chapter on page 22. Following a Reset sequence, the WDT Counter is initialized with its reset value.

WDT Reset in STOP Mode

If enabled in STOP Mode and configured to generate a Reset when a time-out occurs and the device is in STOP Mode, the Watchdog Timer initiates a Stop Mode Recovery. Both the WDT status bit and the STOP bit in the Reset Status and Control Register are set to 1 following WDT time-out in STOP Mode. For more information, see the Reset and Stop Mode Recovery chapter on page 22.

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 spurious 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).

All steps of the Watchdog Timer Reload Unlock sequence must be written in the order listed above. The value in the Watchdog Timer Reload registers is loaded into the counter every time a WDT instruction is executed.

Watchdog Timer Reload High and Low Byte Registers

The Watchdog Timer Reload High and Low Byte (WDTH, WDTL) registers, shown in Tables 40 and 41, form the 16-bit reload value that is loaded into the Watchdog Timer when a WDT instruction executes. The 16-bit reload value is {WDTH[7:0], WDTL[7:0]}.

Write to these registers following the unlock sequence sets the appropriate reload value. Reading from these registers returns the current WDT count value.

Table 40. Watchdog Timer Reload High Byte Register (WDTH)

Bit	7	6	5	4	3	2	1	0
Field	WDTH							
RESET	04H							
R/W	R/W*							
Address	FF2H							
Note: *R/W = A read returns the current WDT count value; a write sets the appropriate reload value.								

Bit	Description
[7:0] WDTH	**WDT Reload High Byte** Most significant byte (MSB), Bits[15:8], of the 16-bit WDT reload value.

Table 41. Watchdog Timer Reload Low Byte Register (WDTL)

Bit	7	6	5	4	3	2	1	0
Field	WDTL							
RESET	00H							
R/W	R/W*							
Address	FF3H							
Note: *R/W = A read returns the current WDT count value; a write sets the appropriate reload value.								

Bit	Description
[7:0] WDTL	**WDT Reload Low** Least significant byte (LSB), Bits[7:0], of the 16-bit WDT reload value.

Pulse Width Modulator

The Z8FMC16100 Series MCU includes a Pulse-Width Modulator (PWM) optimized for Motor Control applications. The PWM features include:

- 6 independent PWM outputs, or 3 complementary PWM output pairs
- Programmable deadband insertion for complementary output pairs
- Edge-aligned or center-aligned PWM signal generation
- PWM OFF state is option-bit-programmable
- PWM outputs driven to OFF state on System Reset
- Asynchronous disabling of PWM outputs on system fault; outputs are forced to the OFF state
- FAULT inputs generate pulse-by-pulse or hard shutdown
- 12-bit reload counter with 1-, 2-, 4-, or 8-bit programmable clock prescaler
- High current source and sink on all PWM outputs
- When outputs are disabled, PWM pairs are used as general-purpose inputs
- ADC synchronized with PWM period
- Narrow pulse suppression with programmable threshold

Architecture

The PWM unit consists of a master timer to generate the modulator time base and six independent compare registers to set the pulse-width modulation for each output. The six outputs are designed to provide control signals for inverter drive circuits. The outputs are grouped into pairs consisting of a High driver and a Low driver output. The output pairs are programmable to operate independently or as complementary signals. In complementary output mode, a programmable dead time is inserted to ensure non-overlapping signal transitions. The master count and compare values feed into modulator logic that generates the proper transitions in the output states. Output polarity and fault/OFF state control logic allows programming of the default OFF states, which forces the outputs to a safe state in the event a fault in the motor drive is detected.

Figure 7 displays the architecture of the PWM modulator.

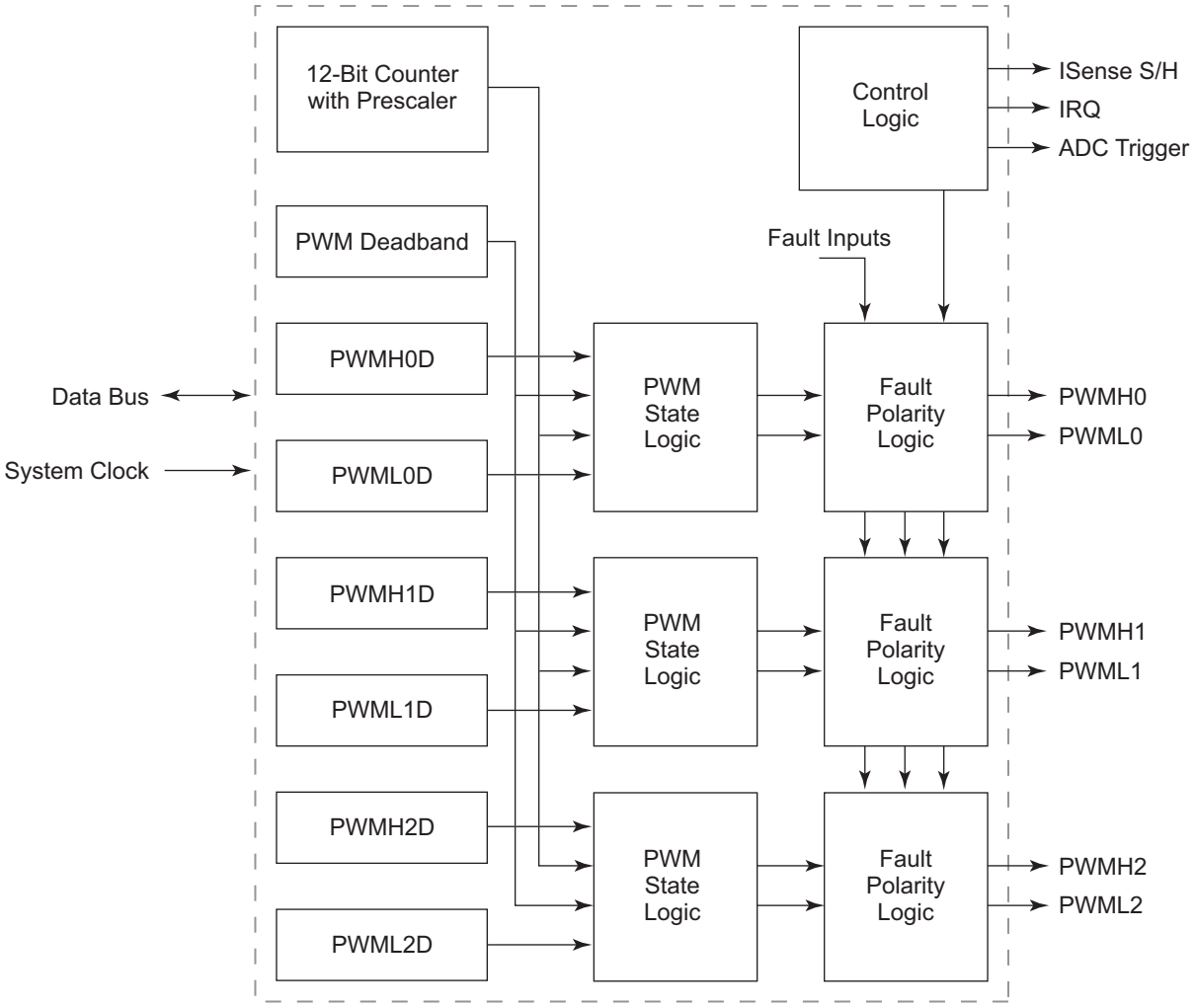


Figure 7. PWM Block Diagram

PWM Option Bits

To protect the configuration of critical PWM parameters, the settings to enable the output channels and the default OFF state are maintained as user option bits. These values are set when the user program code is written to the part; they cannot be changed by software. For more information, see the Option Bits chapter on page 221.

PWM Off State and Output Polarity

The default OFF state and the polarity of the PWM outputs are controlled by the PWMHI and PWMLO option bits. The PWMHI option controls the OFF state and the polarity for the PWM High outputs H0, H1 and H2. The PWMLO option controls the OFF state and the polarity for the Low outputs L0, L1 and L2.

The OFF state is the value programmed in the option bit. For example, programming PWMHI to a 1 sets the OFF state of PWMH0, H1 and H2 to a High logic value and the active state a Low logic value. Conversely, programming PWMHI to a 0 causes the OFF state to be a Low logic value. PWMLO is programmed in a similar manner.

The relative polarity of the PWM channel pairs is controlled by the POL*x* bits in the PWM Control 1 Register (PWMCTL1). These bits do not affect the OFF state programmed by the option bits. Setting these bits inverts the High and Low of the selected channels. The relative channel polarity controls the order in which the signals of a given PWM pair toggle. If a POL*x* bit is reset to zero, the High will first go active at the start of a PWM period. Alternately, if the bit is set, the Low will go active first. A switching of the POL*x* bits is synchronized with the PWM reload event (see the PWM Reload Event section that follows). In complementary mode, the switch is additionally delayed until the end of the programmed deadband time.

PWM Channel Pair Enable

Following a POR, the PWM pins enter a high-impedance state. As the internal reset proceeds, the PWM outputs are forced to the OFF state as determined by the PWMHI and PWMLO OFF state option bits.

The PWM0EN, PWM1EN and PWM2EN option bits enable the PWM0, PWM1 and PWM2 output pairs, respectively. If a PWM channel pair is not enabled, it remains in a high-impedance state after reset and can be used as a general-purpose input.

PWM Reload Event

To prevent erroneous PWM pulse-widths and periods, registers that control the timing of the output are buffered. Buffering causes all of the PWM compare values to update at the same time. In other words, the registers that control the duty cycle and clock source prescaler only take effect upon a PWM reload event. A PWM reload event can be configured to occur at the end of each PWM period, or only every 2, 4, or 8 PWM periods by setting the RELFREQ bits in the PWM Control 1 Register (PWMCTL1). The software must indicate that all new values are ready by setting the READY bit in the PWM Control 0 Register (PWMCTL0) to 1. After this READY bit has been set to 1, the buffered values take effect at the next reload event.

PWM Prescaler

The prescaler allows the PWM clock signal to be decreased by factors of 1, 2, 4, or 8 with respect to the system clock. The PRES[1:0] bit field in the PWM Control 1 Register (PWMCTL1) controls prescaler operation. This 2-bit PRES field is buffered so that the prescale value only changes upon a PWM reload event.

PWM Period and Count Resolution

The PWM counter operates in two modes to allow edge-aligned outputs and center-aligned outputs. Figures 8 and 9 display edge-aligned and center-aligned PWM outputs. The period of the PWM outputs (PERIOD) is determined by which mode the PWM counter is operating. The active time of a PWM output is determined by the programmed duty cycle (PWMDC) and the programmed deadband time (PWMDB).

The sections that follows these two figures describe the PWM timer modes and the registers that control the duty cycle and deadband time.

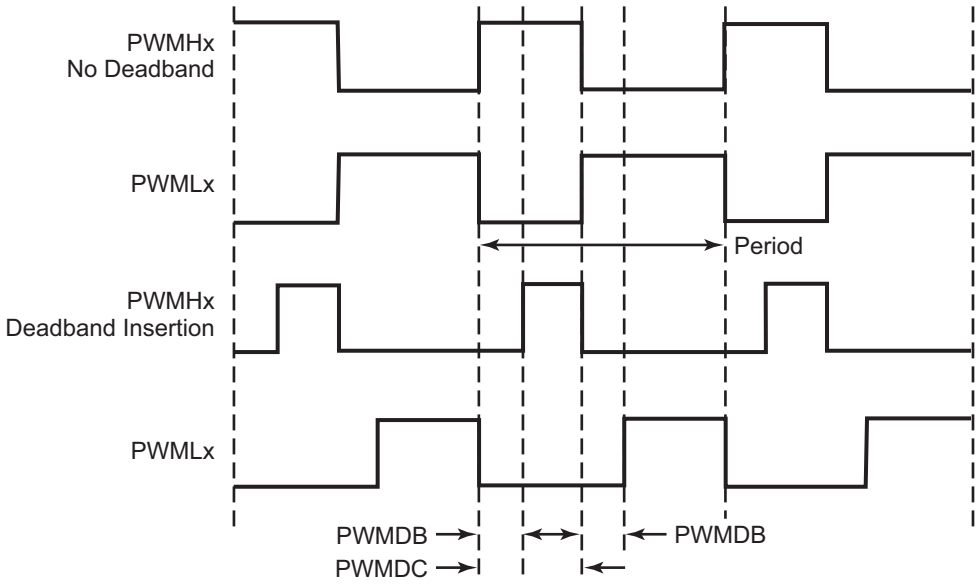


Figure 8. Edge-Aligned PWM Output

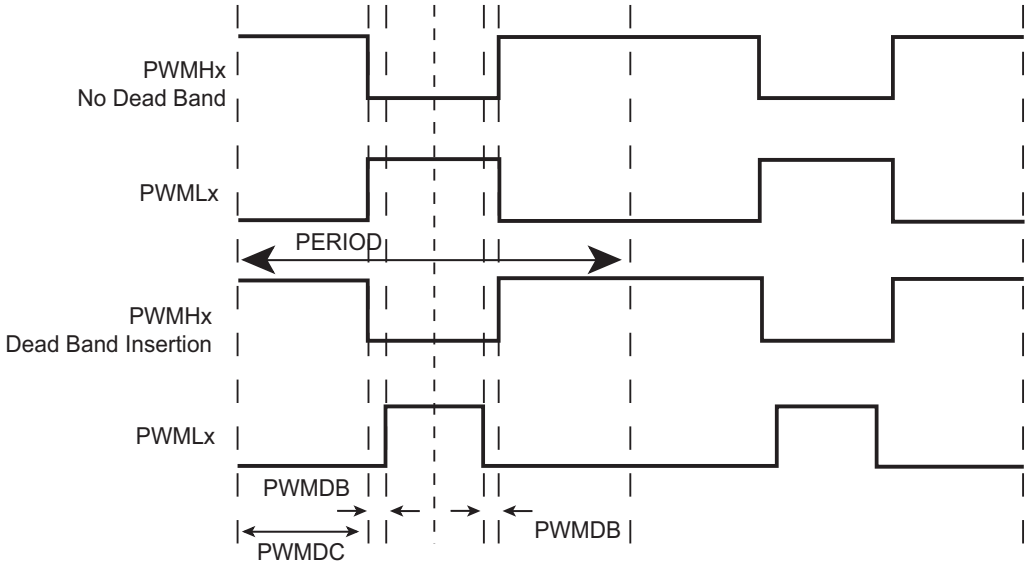


Figure 9. Center-Aligned PWM Output

EDGE-ALIGNED Mode

In EDGE-ALIGNED PWM mode, a 12-bit up counter creates the PWM period with a minimum resolution equal to the PWM clock source period. The counter counts up to the reload value, resets to 000h and then resumes counting.

$$\text{EDGE-ALIGNED PWM Mode Period} = \frac{\text{Prescaler x Reload Value}}{f_{SYSTEMCLK}}$$

CENTER-ALIGNED Mode

In CENTER-ALIGNED PWM mode, a 12-bit up/down counter creates the PWM period with a minimum resolution equal to twice the PWM clock source period. The counter counts up to the reload value and then counts down to 0.

$$\text{CENTER-ALIGNED PWM Mode Period} = \frac{2 \text{ x Prescaler x (Reload Value)}}{f_{SYSTEMCLK}}$$

PWM Duty Cycle Registers

The PWM Duty Cycle registers (PWMH0D, PWML0D, PWMH1D, PWML1D, PWMH2D and PWML2D) contain a 16-bit signed value, in which bit 15 is the sign bit. The Duty Cycle value is compared to the current 12-bit unsigned PWM count value. If the

PWM Duty Cycle value is set less than or equal to 0, the PWM output is deasserted for the full PWM period. If the PWM Duty Cycle value is set to a value greater than the PWM reload value, the PWM output is asserted for the full PWM period.

Independent and Complementary PWM Outputs

The six PWM outputs are configured to operate independently or as three complementary pairs. Operation as six independent PWM channels is enabled by setting the INDEN bit in PWM Control 1 Register (PWMCTL1). The PWMEN bit must be cleared to alter this bit. In independent mode, each PWM output uses its own PWM duty cycle value.

When configured to operate as three complementary pairs, the PWM duty cycle values PWMH0D, PWMH1D and PWMH2D control the modulator output. In complementary output mode, deadband time is also inserted.

The POLx bits in the PWM Control 1 Register (PWMCTL1) select the relative polarity of the High and Low signals. As displayed in Figures 8 and 9, when the POLx bits are cleared to 0, the High PWM output will start in the ON state and transition to the OFF state when the PWM timer count reaches the programmed duty cycle. The Low PWM value starts in the OFF state and transitions to the ON state as the PWM timer count reaches the value in the associated duty cycle register. Setting the POLx causes the High output to start in the OFF state and the Low output to start in the ON state.

Manual Off-State Control of PWM Output Channels

Each PWM output is controlled directly by the modulator logic or set to the OFF state. To manually set the PWM outputs to the OFF state, set the OUTCTL bit and the associated OUTx bits in the PWM Output Control Register (PWMOUT). OFF state control operates individually by channel. For example, suppressing the single output of a pair allows the complementary channel to continue operating. Similarly, if outputs are operating independently, disabling one output channel has no effect on the other PWM outputs.

Deadband Insertion

When PWM outputs are configured to operate as complementary pairs, an 8-bit deadband value can be defined in the PWM Deadband Register (PWMDB). Inserting deadband time causes the modulator to separate the deassertion of one PWM signal from the assertion of its complement. This separation is essential for many motor control applications in that it prevents simultaneous turn-on of the High and Low drive transistors. The deadband counter directly counts system clock cycles and is unaffected by PWM prescaler settings. The width of this deadband is attributed to the number of system clock cycles specified in the PWM Deadband Register (PWMDB). The minimum deadband duration is one system clock and the maximum duration is 255 system clocks. During the deadband period, both PWM outputs of a complementary pair are deasserted. The generation of deadband time

does not alter the PWM period; instead, the deadband time is subtracted from the active time of the PWM outputs. Figures 8 and 9 display the effect of deadband insertion on the PWM output.

Minimum PWM Pulse Width Filter

The PWM modulator is capable of producing pulses as narrow as a single system clock cycle in width. As the response time of external drive circuits can be slower than the period of a system clock, a filter is implemented to enforce a minimum-width pulse on the PWM output pins. All output pulses, whether High or Low, must be at least the minimum number of PWM clock cycles (for more details, see the PWM Prescaler section on page 67) in width as specified in the PWM Minimum Pulse Width Filter (PWMMPF) Register. If the expected pulse width is less than the threshold, the associated PWM output does not change state until the duty cycle value has changed sufficiently to allow pulse generation of an acceptable width. The minimum pulse width filter also accounts for the duty cycle variation caused by the deadband insertion. The PWM output pulse is filtered even if the programmed duty cycle is greater than the threshold, but the pulse width decrease because of deadband insertion causes the pulse to be too narrow. The pulse width filter value is
calculated as:

$$\text{Roundup (PWMMPF)} = \frac{T_{MINPULSEOUT}}{T_{SYSTEMCLOCK} \times PWM_{PRESCALER}}$$

where $T_{MINPULSEOUT}$ is the shortest allowed pulse width on the PWM outputs, in seconds.

The PWM Minimum Pulse Width Filter Register can only be written when the PWMEN bit is cleared. Values written to this register when PWMEN is set will be ignored.

Synchronization of PWM and Analog-to-Digital Converter

The ADC on the Z8FMC16100 Series MCU can be synchronized with the PWM period. Enabling the PWM ADC trigger causes the PWM to generate an ADC conversion signal at the end of each PWM period. Additionally, in center-aligned mode, the PWM generates a trigger at the center of the period. Setting the ADCTRIG bit in the PWM Control 0 Register (PWMCTL0) enables ADC synchronization.

PWM Timer and Fault Interrupts

The PWM generates interrupts to the eZ8 CPU on any of the following events:

PWM Reload. The interrupt is generated at the end of PWM period, when a PWM Register reload occurs (the READY bit is set).

PWM Fault. A fault condition is indicated by asserting any of the $\overline{\text{FAULT}}$ pins or by the assertion of the comparator.

Fault Detection and Protection

The Z8FMC16100 Series MCU contains hardware and software fault controls that allow rapid deassertion of all enabled PWM output signals. A logic Low on an external fault pin ($\overline{\text{FAULT0}}$ or $\overline{\text{FAULT1}}$), or the assertion of the overcurrent comparator, forces the PWM outputs to a predefined OFF state.

Similar deassertion of the PWM outputs can be accomplished in software by writing to the PWMOFF bit in the PWM Control 0 Register. The PWM counter continues to operate while the outputs are deasserted (made inactive) due to one of these fault conditions.

The fault inputs can be individually enabled through the PWM Fault Control Register. If a fault condition is detected and the source is enabled, a fault interrupt is generated. The PWM Fault Status Register (PWMFSTAT) is read to determine which fault source has caused the interrupt.

After a fault has been detected and after the PWM outputs are disabled, modulator control of the PWM outputs can be reenabled either by software, or by deassertion of the $\overline{\text{FAULT}}$ input signal. Selection of either method is made through the PWM Fault Control Register (PWMFCTL). Configuration of the fault modes and reenable methods allows pulse-by-pulse limiting and hard shutdown. When configured in automatic restart mode, the PWM outputs are reengaged at beginning of the next PWM cycle (the master timer value is equal to 0) if all fault signals are deasserted. In a software-controlled restart, all fault inputs must be deasserted and all fault flags cleared.

The fault input pin is Schmitt-triggered. The input signal from the pin and comparators, pass though an analog filter to reject high-frequency noise.

The logic path from the fault sources to the PWM outputs is asynchronous, which ensures that the fault inputs forces the PWM outputs to their OFF state, even if the system clock is stopped.

Note: After Reset the Fault mechanism is active, disable the Fault source for normal port input operation.

PWM Operation in CPU HALT Mode

When the eZ8 CPU is operating in HALT Mode, the PWM continues to operate, if enabled. To minimize the current in HALT Mode, the PWM must be disabled by clearing the PWMEN bit to 0.

PWM Operation in CPU STOP Mode

When the eZ8 CPU is operating in STOP Mode, the PWM is disabled because the system clock has ceased to operate in STOP Mode. The PWM outputs remain in the same state as they were prior to entering STOP Mode. During normal operation, the PWM outputs must be disabled by the software prior to the CPU entering STOP Mode. A fault condition detected in STOP Mode forces the PWM outputs to a predefined OFF state.

Observing the State of PWM Output Channels

The logic value of the PWM outputs can be sampled by reading the PWMIN Register. If a PWM channel pair is disabled (an option bit is not set), the associated PWM outputs are forced to high-impedance and can be used as general-purpose inputs.

PWM High and Low Byte Registers

The PWM High and Low Byte (PWMH and PWML) registers, shown in Tables 42 and 43, contain the current 12-bit PWM count value. Reads from PWMH cause the value in PWML to be stored in a temporary holding register. A read from PWML always returns this temporary register value.

Caution: Writing to the PWM High and Low Byte registers while the PWM is enabled is not recommended. There are no temporary holding registers for Write operations, so simultaneous 12-bit Writes are not possible.

If either the PWM High Register or Low Byte Register 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 42. PWM High Byte Register (PWMH)

Bit	7	6	5	4	3	2	1	0
Field	Reserved				PWMH			
RESET	0H				0H			
R/W	R/W				R/W			
Address	F2CH							

Table 43. PWM Low Byte Register (PWML)

Bit	7	6	5	4	3	2	1	0
Field	PWML							
RESET	00H							
R/W	R/W							
Address	F2DH							

Bit	Description
[7:0] PWML	**PWM High and Low Bytes** These 2 bytes, {PWMH[3:0], PWML[7:0]}, contain the current 12-bit PWM count value.

PWM Reload High and Low Byte Registers

The PWM Reload High and Low Byte (PWMRH and PWMRL) registers, shown in Tables 44 and 45, store a 12-bit reload value, {PWMRH[3:0], PWMRL[7:0]}. The PWM reload value is held in buffer registers. The PWM reload value written to the buffer registers is not used by the PWM generator until the next PWM reload event occurs. Reads from these registers always return the values from the buffer registers.

$$\text{Edge-Aligned PWM Mode Period} = \frac{\text{Prescaler x Reload Value}}{f_{PWMCLK}}$$

$$\text{Center-Aligned PWM Mode Period} = \frac{\text{2 x Prescaler x Reload Value}}{f_{PWMCLK}}$$

Table 44. PWM Reload High Byte Register (PWMRH)

Bit	7	6	5	4	3	2	1	0
Field	Reserved				PWMRH			
RESET	0H				FH			
R/W	R/W				R/W			
Address	F2EH							

Table 45. PWM Reload Low Byte Register (PWMRL)

Bit	7	6	5	4	3	2	1	0
Field	PWMRL							
RESET	FF							
R/W	R/W							
Address	F2FH							

Bit	Description
[7:4]	**Reserved** These bits are reserved and must be programmed to 0000.
[3:0] PWMRH	**PWM Reload Register High and Low** These two bytes form the 12-bit Reload value, {PWMRH[3:0], PWMRL[7:0]}. This value sets the PWM period.

PWM 0–2 Duty Cycle High and Low Byte Registers

The PWM 0–2 H/L Duty Cycle High and Low Byte (PWM*x*DH and PWM*x*DL) registers, shown in Tables 46 and 47, set the duty cycle of the PWM signal. This 14-bit signed value is compared to the PWM count value to determine the PWM output. Reads from these registers always return the values from the temporary holding registers. The PWM duty cycle value is not used by the PWM generator until the next PWM reload event occurs.

$$\text{PWM Duty Cycle} = \frac{\text{PWM Duty Cycle Value}}{\text{PWM Reload Value}}$$

Writing a negative value (DUTYH[7] = 1) forces the PWM to be OFF for the full PWM period. Writing a positive value greater than the 12-bit PWM reload value forces the PWM to be ON for the full PWM period.

Table 46. PWM 0–2 H/L Duty Cycle High Byte Register (PWMH*x*DH, PWML*x*DH)

Bit	7	6	5	4	3	2	1	0
Field	SIGN	Reserved		DUTYH				
RESET	0	00		0_0000				
R/W	R/W	R/W		R/W				
Address	F30H, F32H, F34H, F36H, F38H, F3AH							

Bit	Description
[7] SIGN	**Duty Cycle Sign** 0 = Duty Cycle is a positive two's complement number. 1 = Duty Cycle is a negative two's complement number. Output is forced to the off-state.
[6]	**Reserved** These bits are reserved and must be programmed to 00.
[4:0] DUTYH	**PWM Duty Cycle High and Low Bytes** The two bytes {DUTYH[7:0], DUTYL[7:0]} form a 14-bit signed value. The value is compared to the current 12-bit PWM count.

Table 47. PWM 0–2 H/L Duty Cycle Low Byte Register (PWMH*x*DL, PWML*x*DL)

Bit	7	6	5	4	3	2	1	0
Field	DUTYL							
RESET	00H							
R/W	R/W							
Address	F31H, F33H, F35H, F37H, F39H, F3BH							

Bit	Description
[7:0] DUTYL	**PWM Duty Cycle High and Low Bytes** The two bytes {DUTYH[7:0], DUTYL[7:0]} form a 14-bit signed value. The value is compared to the current 12-bit PWM count.

PWM Control 0 Register

The PWM Control 0 (PWMCTL0) Register, shown in Table 48, controls PWM operation.

Table 48. PWM Control 0 Register (PWMCTL0)

Bit	7	6	5	4	3	2	1	0
Field	PWMOFF	OUTCTL	ALIGN	Reserved	ADCTRIG	Reserved	READY	PWMEN
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	F20H							

Bit	Description
[7] PWMOFF	**Place PWM Outputs in OFF State** 0 = Disable modulator control of PWM pins. Outputs are in predefined off-state. This is not dependent on the reload event. 1 = Reenable modulator control of PWM pins at next PWM reload event.
[6] OUTCTL	**PWM Output Control** 0 = PWM outputs are controlled by the PWM. 1 = PWM outputs selectively disabled (set to off-state) according to values in the OUT*x* bits of the PWMOUT Register.
[5] ALIGN	**PWM Edge Alignment** 0 = PWM outputs are edge aligned. 1 = PWM outputs are center aligned.
[4] Reserved	**Reserved** This bit is reserved and must be programmed to 0.
[3] ADCTRIG	**ADC Trigger Enable** 0 = No ADC trigger pulses. 1 = ADC trigger enabled.
[2] Reserved	**Reserved** This bit is reserved and must be programmed to 0.
[1] READY	**Values Ready for Next Reload Event** 0 = PWM values (prescale, period and duty cycle) are not ready. Do not use values in holding registers at next PWM reload event. 1 = PWM values (prescale, period and duty cycle) are ready. Transfer all values from temporary holding registers to working registers at next PWM reload event.
[0] PWMEN	**PWM Enable** 0 = PWM is disabled and enabled PWM output pins are forced to default off-state. PWM master counter is stopped. Certain control registers may only written in this state. 1 = PWM is enabled and PWM output pins are enabled as outputs.

PWM Control 1 Register

The PWM Control 1 (PWMCTL1) Register, shown in Table 49, controls portions of the PWM operation.

Table 49. PWM Control 1 Register (PWMCTL1)

Bit	7	6	5	4	3	2	1	0
Field	RLFREQ[1:0]		INDEN	POL45	POL23	POL10	PRES[1:0]	
RESET	00		0	0	0	0	00	
R/W	R/W		R/W	R/W	R/W	R/W	R/W	
Address	F21H							

Bit	Description
[7:6] RLFREQ[1:0]	**Reload Event Frequency** This bit field is buffered. Changes to the reload event frequency takes effect at the end of the current PWM period. Reads always return the bit values from the temporary holding register. 00 = PWM reload event occurs at the end of every PWM period. 01 = PWM reload event occurs once every 2 PWM periods. 10 = PWM reload event occurs once every 4 PWM periods. 11 = PWM reload event occurs once every 8 PWM periods.
[5] INDEN	**Independent PWM Mode Enable** 0 = This bit may only be altered when PWMEN (PWMCTL0) cleared; PWM outputs operate as 3 complementary pairs. 1 = PWM outputs operate as 6 independent channels.
[4] POL45	**PWM2 Polarity** 1 = Invert Output polarity for channel pair PWM2. 0 = Noninverted polarity for channel pair PWM2.
[3] POL23	**PWM1 Polarity** 1 = Invert Ouput polarity for channel pair PWM1. 0 = Noninverted polarity for channel pair PWM1.
[2] POL10	**PWM0 Polarity** 1 = Invert Ouput polarity for channel pair PWM0. 0 = Noninverted polarity for channel pair PWM0.
[1:0] PRES	**PWM Prescaler** The prescaler divides down the PWM input clock (either the system clock or the PWMIN external input). This field is buffered. Changes to this field take effect at the next PWM reload event. Reads always return the values from the temporary holding register. 00 = Divide by 1. 01 = Divide by 2. 10 = Divide by 4. 11 = Divide by 8.

PWM Deadband Register

The PWM Deadband (PWMDB) Register, shown in Table 50, stores the 8-bit PWM deadband value. This register determines the number of system clock cycles inserted as dead-time in complementary output mode. The minimum deadband value is 1.

Table 50. PWM DeadBand Register (PWMDB)

Bit	7	6	5	4	3	2	1	0
Field	PWMDB[7:0]							
RESET	01H							
R/W	R/W							
Address	F22H							

Bit	Description
[7:0] PWMDB	**PWM Deadband** Sets the PWM dead band period for which both PWM outputs of a complementary PWM output pair are deasserted.

Note: This register can only be written when PWMEN is cleared.

PWM Minimum Pulse Width Filter

The value in the PWM Minimum Pulse Width Filter (PWMMPF) Register, shown in Table 51, determines the minimum width pulse, either High or Low, which can be generated by the PWM module. The minimum pulse width period is calculated as:

$$T_{MINPULSEOUT} = \frac{(PWMDB+PWMMPF)*PWMPrescale}{FOSC}$$

Caution: A value other than 00H must be written to the PWMMPF Register; otherwise, the PWM output waveform will be distorted.

Table 51. PWM Minimum Pulse Width Filter (PWMMPF)

Bit	7	6	5	4	3	2	1	0
Field	PWMMPF[7:0]							
RESET	00H							
R/W	R/W							
Address	F23H							

Bit	Description
[7:0] PWMMPF	**PWM Minimum Pulse Filter** Sets the minimum allowed output pulse width in PWM clock cycles.

Note: This register can only be written when PWMEN is cleared.

PWM Fault Mask Register

The PWM Fault Mask (PWMFM) Register, shown in Table 52, enables individual fault sources. PWM behavior when an input is asserted is determined by the PWM Fault Control Register (PWMFCTL).

Table 52. PWM Fault Mask Register (PWMFM)

Bit	7	6	5	4	3	2	1	0
Field	Reserved		DBGMSK	Reserved		F1MASK	C0MASK	FMASK
RESET	00		0	00		0	0	0
R/W	R		R/W	R		R/W	R/W	R/W
Address	F24H							

Bit	Description
[7:6]	Must be 0.
[5] DBGMSK	**Debug Entry Fault Mask** 0 = Entering CPU DEBUG Mode generates a PWM fault. 1 = Entering CPU DEBUG Mode does not generate a PWM fault.
[4:3]	**Reserved** These bits are reserved and must be programmed to 00.
[2] F1MASK	**$\overline{\text{Fault1}}$ Fault Mask** 0 = $\overline{\text{Fault1}}$ generates a PWM fault. 1 = $\overline{\text{Fault1}}$ does not generate a PWM fault.
[1] C0MASK	**Comparator Fault Mask** 0 = Comparator generates a PWM fault. 1 = Comparator does not generate a PWM fault.

Bit	Description (Continued)
[0] F0MASK	**Fault Pin Mask** 0 = $\overline{\text{Fault0}}$ pin generates a PWM fault. 1 = Fault0 pin does not generate a PWM fault.
Note: This register can only be written when PWMEN is cleared.	

PWM Fault Status Register

The PWM Fault Status (PWMFSTAT) Register, shown in Table 53, provides status of fault inputs and timer reload. The fault flags indicate which fault source is active. If a fault source is masked the flag in this register will not be set if the source is asserted. The reload flag is set when the timer compare values are updated. Clear flags by writing a 1 to the flag bits. Fault flag bits can only be cleared if the associated fault source has deasserted.

Table 53. PWM Fault Status Register (PWMFSTAT)

Bit	7	6	5	4	3	2	1	0
Field	RLDFlag	Reserved	DBGFLAG	Reserved		F1FLAG	C0FLAG	FFLAG
RESET	U	0	U	00		U	U	U
R/W	R/W1C	R	R/W1C	R		R/W1C	R/W1C	R/W1C
Address	F25H							

Bit	Description
[7] RLDFlag	Reload Flag This bit is set and latched when a PWM timer reload occurs. Writing a 1 to this bit clears the flag.
[6]	**Reserved** This bit is reserved and must be programmed to 0.
[5] DBGFLAG	**Debug Flag** This bit is set and latched when DEBUG Mode is entered. Writing a 1 to this bit clears the flag.
[4:3]	**Reserved** These bits are reserved and must be programmed to 00.
[2] F1FLAG	**$\overline{\text{Fault1}}$ Flag** This bit is set and latched when Fault1 is asserted. Writing a 1 to this bit clears the flag.
[1] C0FLAG	**Comparator 0 Flag** This bit is set and latched when Comparator is asserted. Writing a 1 to this bit clears the flag.
[0] FFLAG	**Fault Flag** This bit is set and latched when the $\overline{\text{FAULT0}}$ input is asserted. Writing a 1 to this bit clears the flag.
Note: For this register, W1C means that you must write a 1 to clear the flag.	

PWM Fault Control Register

The PWM Fault Control (PWMFCTL) Register, shown in Table 54, determines how the PWM recovers from a fault condition. Settings in this register select automatic or software-controlled PWM restart.

Table 54. PWM Fault Control Register (PWMFCTL)

Bit	7	6	5	4	3	2	1	0
Field	Reserved	DBGRST	Fault1INT	Fault1RST	CMP0INT	CMP0RST	Fault0INT	Fault0RST
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	F28H							

Bit	Description
[7] Reserved	**Reserved** This bit is reserved and must be programmed to 0.
[6] DBGRST	**DebugRestart** 0 = Automatic recovery. PWM resumes control of outputs when all fault sources have deasserted and a new PWM period begins. 1 = Software controlled recovery. PWM resumes control of outputs only after all fault sources have deasserted and all fault flags are cleared and a PWM reload occurs.
[5] Fault1INT	**Fault1 Interrupt** 0 = Interrupt on comparator assertion disabled. 1 = Interrupt on comparator assertion enabled.
[4] Fault1RST	**Fault1 Restart** 0 = Automatic recovery. PWM resumes control of outputs when all fault sources have deasserted. 1 = Software controlled recovery. PWM resumes control of outputs only after all fault sources have deasserted and all fault flags are cleared and a PWM reload occurs.
[3] CMP0INT	**Comparator 0 Interrupt** 0 = Interrupt on comparator 0 assertion disabled. 1 = Interrupt on comparator 0 assertion enabled.
[2] CMP0RST	**Comparator 0 Restart** 0 = Automatic recovery. PWM resumes control of outputs when all fault sources have deastered. 1 = Software controlled recovery. PWM resumes control of outputs only after all fault sources have deasserted and all fault flags are cleared and a PWM reload occurs
[1] Fault0INT	**Fault0 Interrupt** 0 = Interrupt on $\overline{\text{Fault0}}$ pin assertion disabled. 1 = Interrupt on $\overline{\text{Fault0}}$ pin assertion enabled.

Note: This register can only be written when PWMEN is cleared.

Bit	Description (Continued)
[0] Fault0RST	**Fault0 Restart** 0 = Automatic recovery. PWM resumes control of outputs when all fault sources have deasstered. 1 = Software-controlled recovery. PWM resumes control of outputs only after all fault sources have deasserted and all fault flags are cleared and a PWM reload occurs

Note: This register can only be written when PWMEN is cleared.

PWM Input Sample Register

PWM pin values are sampled by reading the PWM Input Sample Register, shown in Table 55.

Table 55. PWM Input Sample Register (PWMIN)

Bit	7	6	5	4	3	2	1	0
Field	Reserved	FAULT	IN2H	IN2L	IN1H	IN1L	IN0H	IN0L
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	F26H							

Bit	Description
[7]	**Reserved** This bit is reserved and must be programmed to 0.
[6] FAULT	**Sample Fault0 Pin** 0 = A Low level signal was read on the Fault0 pin. 1 = A High level signal was read on the Fault0 pin.
[5:0] IN2H/IN2L/ IN1H/IN1L/ IN0H/IN0L	**Sample PWM Pins** 0 = A Low level signal was read on the pins. 1 = A High level signal was read on the pins.

PWM Output Control Register

The PWM Output Control (PWMOUT) Register, shown in Table 56, enables modulator control of the six PWM output signals. Output control is enabled by OUTCTL bit in the PWMCTL0 Register. The PWM continues to operate, but has no effect on the disabled PWM pins. If a fault condition is detected, all PWM outputs are forced to their selected OFF state.

Table 56. PWM Output Control Register (PWMOUT)

Bit	7	6	5	4	3	2	1	0
Field	Reserved	Reserved	OUT2L	OUT2H	OUT1L	OUT1H	OUT0L	OUT0H
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	F27H							

Bit	Description
[7:6] Reserved	**Reserved** These bits are reserved and must be programmed to 00.
[5, 3, 1] OUTxL	**PWM L2/L1/L0 Output Configuration** 0 = PWM L2/L1/L0 output signal is enabled and controlled by PWM. 1 = PWM L2/L1/L0 output signal is in low-side off-state.
[4, 2, 0] OUTyH	**PWM H2/H1/H0 Output Configuration** 0 = PWM H2/H1/H0 output signal is enabled and controlled by PWM. 1 = PWM H2/H1/H0 output signal is in high-side off-state.

Note: x indicates low bits in the range 2–0; y indicates high bits in the range 2–0.

Current Sense ADC Trigger Control Register

An ADC trigger is generated when the PWM output signals match the state specified by the Current-Sense ADC Trigger Control Register. The match logic is an AND/OR tree that will resolve to true if based on the register settings. An ADC conversion will be triggered on the rising edge of this signal.

The logic equation for the ADC trigger is:

```
ADCTRIGGER = CSTPOL ^ (
    ( HEN & PWMH0 &PWMH1 & PWMH2) |
    ( LEN & PWML0 &PWML1 & PWML2) |
    ( nHEN & !PWMH0 &!PWMH1 & !PWMH2) |
    ( nLEN & !PWMH0 & !PWMH1 & !PWMH2) )
```

In the preceding equation, the following symbols are defined as:

- The ^ symbol indicates a logical exclusive OR (XOR) function
- The & symbol indicates a logical AND function
- The | symbol indicates a logical OR function
- The ! symbol indicates a logical NOT function

The combinations of polarity, enable and PWM signals allow the logic to generate an ADC-trigger under a wide variety of operating conditions. The HEN, LEN, nHEN and nLEN bits enable a group in the OR logic. The CSTWMx bits allow the level of the PWM output signals to control the equation. If a CSTPWMx bit is cleared, the value of the associated PWMx output always evaluates to a TRUE condition in the equation. Note that these bits do not affect the actual PWM outputs.

Table 57. Current-Sense Trigger Control Register (PWMSHC)

Bit	7	6	5	4	3	2	1	0
Field	CSTPOL	HEN	NHEN	LEN	NLEN	CSTPWM2	CSTPWM1	CSTPWM0
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	F29H							

Bit	Description
[7] CSTPOL	**Sample/Hold Polarity** 0 = Hold when terms are active. 1 = Hold when terms are not active.
[6] HEN	**High Side Active Enable** 0 = Ignore Product of PWMH0, PWMH1 and PWMH2 in Sample/Hold equation 1 = Hold when PWMH0, PWMH1 and PWMH2 are all active.
[5] NHEN	**High Side Inactive Enable** 0 = Ignore Product of $\overline{\text{PWMH0}}$, $\overline{\text{PWMH1}}$ and $\overline{\text{PWMH2}}$ in Sample/Hold equation. 1 = Hold when are all active.
[4] LEN	**Low Side Active Enable** 0 = Ignore Product of PWML0, PWML1 and PWML2 in Sample/Hold equation. 1 = Hold when PWML0, PWML1 and PWML2 are all active.
[3] NLEN	**Low Side Inactive Enable** 0 = Ignore Product of $\overline{\text{PWML0}}$, $\overline{\text{PWML1}}$ and $\overline{\text{PWML2}}$ in Sample/Hold equation. 1 = Hold when $\overline{\text{PWML0}}$, $\overline{\text{PWML1}}$ and $\overline{\text{PWML2}}$ are all active.
[2] CSTPWM2	**PWM Channel2 Sample/Hold Enable** 0 = Channel 2 terms are not used in Sample/Hold equation. 1 = Channel 2 terms are used in Sample/Hold equation.

Bit	Description (Continued)
[1] CSTPWM1	**PWM Channel1 Sample/Hold Enable** 0 = Channel 1 terms are not used in Sample/Hold equation. 1 = Channel 1 terms are used in Sample/Hold equation.
[0] CSTPWM0	**PWM Channel0 Sample/Hold Enable** 0 = Channel 0 terms are not used in Sample/Hold equation. 1 = Channel 0 terms are used in Sample/Hold equation.

General-Purpose Timer

The Z8FMC16100 Series MCU contains one 16-bit reloadable timer used for timing, event counting, or generation of PWM signals.

Features

The features of general-purpose timer include:

- 16-bit reload counter
- Programmable prescaler with prescale values from 1 to 128
- PWM output generation (single or differential)
- Capture and compare capability
- External input pin for event counting, clock gating, or capture signal
- Complementary Timer Output pins
- Timer interrupt

Architecture

Capture and compare capability measures the velocity from a tachometer wheel or reads sensor outputs for rotor position for brushless DC motor commutation. Figure 10 displays the architecture of the timer.

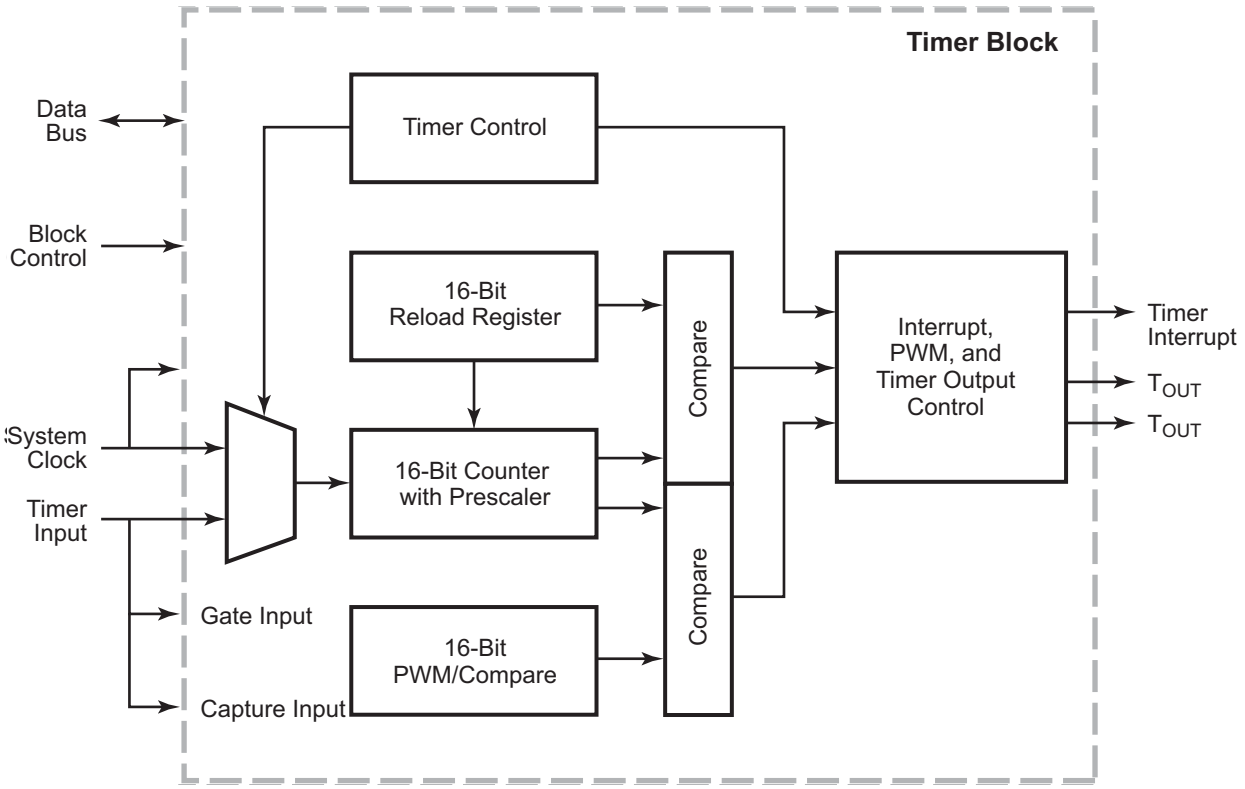


Figure 10. Timer Block Diagram

Operation

The general-purpose timer is a 16-bit up-counter. In normal operation, the timer is initialized to 0001h. After it is enabled, the timer counts up to the value contained in the Reload High and Low Byte registers, then resets to 0001h. The counter either halts or continues, depending on its current mode of operation.

Minimum time-out delay (1 system clock in duration) 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 (2^{16} x 2^{7} system clocks) is set by loading the value 0000h into the Timer Reload High and Low Byte registers and setting the prescale value to 128. When the timer reaches FFFFh, the timer rolls over to 0000h.

If the reload register is set to a value less than the current counter value, the counter continues counting until reaching FFFFh, rolls over to 0000h and continues counting until reaching the reload value, then resets to 0001h.

Timer Operating Modes

The timers can be configured to operate in the following eleven modes:

- ONE-SHOT Mode
- TRIGGERED ONE-SHOT Mode
- CONTINUOUS Mode
- COUNTER Mode
- COMPARATOR COUNTER Mode
- PWM SINGLE OUTPUT Mode
- PWM DUAL OUTPUT Mode
- CAPTURE RESTART Mode
- CAPTURE COMPARE Mode
- COMPARE Mode
- GATED Mode

ONE-SHOT Mode

In ONE-SHOT Mode, the timer counts up to the 16-bit reload value stored in the Timer Reload High and Low Byte registers. The Timer Input is the system clock. After reaching this reload value, the timer generates an interrupt and the count value in the Timer High and Low Byte registers is reset to `0001h`. The timer is automatically disabled and stops counting.

If the Timer Output alternate function is enabled, the Timer Output pin changes state for one system clock cycle (from Low to High, then back to Low if TPOL = 0) at timer reload. If the user chooses, the Timer Output can undergo a permanent state change upon One-Shot time-out, as shown below:

1. Set the TPOL bit in the Timer Control 1 Register to the start value before beginning ONE-SHOT Mode.
2. After starting the timer, set TPOL to the opposite value.

Observe the following procedure to configure a timer for ONE-SHOT Mode and initiate the count:

1. Write to the Timer Control registers to:
   a. Disable the timer.
   b. Configure the timer for ONE-SHOT Mode.
   c. Set the prescale value.

d. If using the Timer Output alternate function, set the initial output level (High or Low) using the TPOL bit.

e. Set the interrupt mode.

2. Write to the Timer High and Low Byte registers to set the starting count value.
3. Write to the Timer Reload High and Low Byte registers to set the reload value.
4. If appropriate, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
5. If using the Timer Output function, configure the associated GPIO port pin for the Timer Output alternate function.
6. Write to the Timer Control 1 Register to enable the timer and initiate counting.

The timer period is calculated by the following equation (Start Value = 1):

$$\text{ONE-SHOT Mode Time-Out Period (s)} = \frac{(\text{Reload Value} - \text{Start Value} + 1) \times \text{Prescaler}}{\text{System Clock Frequency (Hz)}}$$

TRIGGERED ONE-SHOT Mode

In TRIGGERED ONE-SHOT Mode, the timer operates as shown below:

1. The Timer idles until a trigger is received. The timer trigger is taken from the Timer Input pin. 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 signal.
2. Following the trigger event, the timer counts system clocks up to the 16-bit reload value stored in the Timer Reload High and Low Byte registers.
3. Upon reaching the reload value, the timer outputs a pulse on the Timer Output pin, generates an interrupt and resets the count value in the Timer High and Low Byte registers to `0001h`. The duration of the output pulse is a single system clock. The TPOL bit also sets the polarity of the output pulse.
4. The timer idles until the next trigger event. Trigger events that occur while the timer is responding to a previous trigger is ignored.

Observe the following procedure to configure Timer 0 in TRIGGERED ONE-SHOT Mode and initiate operation:

1. Write to the Timer Control registers to:
   a. Disable the timer.
   b. Configure the timer for TRIGGERED ONE-SHOT Mode.

c. Set the prescale value.
d. If using the Timer Output alternate function, set the initial output level (High or Low) via the TPOL bit.
e. Set the interrupt mode.

2. Write to the Timer High and Low Byte registers to set the starting count value.
3. Write to the Timer Reload High and Low Byte registers to set the reload value.
4. If appropriate, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
5. If using the Timer Output function, configure the associated GPIO port pin for the Timer Output alternate function.
6. Write to the Timer Control 1 Register to enable the timer. Counting does not start until the appropriate input transition occurs.

The timer period is calculated by the following equation (Start Value = 1):

$$\text{TRIGGERED ONE-SHOT Mode Time-Out Period (s)} = \frac{(\text{Reload Value} - \text{Start Value} + 1) \times \text{Prescaler}}{\text{System Clock Frequency (Hz)}}$$

Note: The one-shot delay from input trigger to output includes the above-defined time-out period, plus an additional delay of 2–3 system clock cycles, due to the synchronization of the input trigger.

CONTINUOUS Mode

In CONTINUOUS Mode, the timer counts up to the 16-bit reload value stored in the Timer Reload High and Low Byte registers. After 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, if the Timer Output alternate function is enabled, the Timer Output pin changes state (from Low to High or High to Low) after timer reload.

Observe the following procedure to configure a timer for CONTINUOUS Mode and initiate the count:

1. Write to the Timer Control registers to:
   a. Disable the timer.
   b. Configure the timer for CONTINUOUS Mode.
   c. Set the prescale value.

d. If using the Timer Output alternate function, set the initial output level (High or Low) through TPOL.

2. Write to the Timer High and Low Byte registers to set the starting count value (usually 0001h). This setting only affects the first pass in CONTINUOUS Mode. After the first timer reloads in CONTINUOUS Mode, counting begins at the reset value of 0001h.
3. Write to the Timer Reload High and Low Byte registers to set the reload period.
4. If appropriate, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
5. If using the Timer Output function, configure the associated GPIO port pin for the Timer Output alternate function.
6. Write to the Timer Control 1 Register to enable the timer and initiate counting.

The timer period is calculated by the following equation:

$$\text{CONTINUOUS Mode Time-Out Period (s)} = \frac{\text{Reload Value x Prescaler}}{\text{System Clock Frequency (Hz)}}$$

If an initial starting value other than 0001h is loaded into the Timer High and Low Byte registers, use the ONE-SHOT Mode equation to determine the first time-out period.

COUNTER and COMPARATOR COUNTER Modes

In COUNTER Mode, the timer counts input transitions from a GPIO port pin. The Timer Input is taken from the associated GPIO port pin. 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 signal. In COUNTER Mode, the prescaler is disabled.

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

In COMPARATOR COUNTER Mode, the timer counts output transitions from an analog comparator output. Timer 0 takes its input from the output of the comparator. The TPOL bit in the Timer Control 1 Register selects whether the count occurs on the rising edge or the falling edge of the comparator output signal. In COMPARATOR COUNTER Mode, the prescaler is disabled.

Caution: The frequency of the comparator output signal must not exceed one-fourth the system clock frequency.

After 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.

If the Timer Output alternate function is enabled, the Timer Output pin changes state (from Low to High or High to Low) at timer reload.

Observe the following procedure to configure a timer for COUNTER and COMPARATOR
COUNTER modes and initiate the count:

1. Write to the Timer Control registers to:
   a. Disable the timer.
   b. Configure the timer for COUNTER or COMPARATOR COUNTER Mode.
   c. Select either the rising edge or falling edge of the Timer Input or comparator output signal for the count. This choice also sets the initial logic level (High or Low) for the Timer Output alternate function. However, the Timer Output function does not have to be enabled.
2. Write to the Timer High and Low Byte registers to set the starting count value. This setting only affects the first pass in the counter modes. After the first timer reload, counting begins at the reset value of 0001h.
3. Write to the Timer Reload High and Low Byte registers to set the reload value.
4. If appropriate, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
5. Configure the associated GPIO port pin for the Timer Input alternate function (COUNTER Mode).
6. If using the Timer Output function, configure the associated GPIO port pin for the Timer Output alternate function.
7. Write to the Timer Control 1 Register to enable the timer.

PWM SINGLE and DUAL OUTPUT Modes

In PWM SINGLE OUTPUT Mode, the timer outputs a PWM output signal through a GPIO port pin. In PWM DUAL OUTPUT Mode, the timer outputs a PWM output signal and also its complement through two GPIO port pins. The timer first counts up to the 16-bit PWM match value stored in the Timer PWM 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. On 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 Timer Output signal begins with a value equal to TPOL and then transits to TPOL when the timer value matches the PWM value. The Timer Output signal returns to TPOL after the timer reaches the reload value and is reset to 0001h.

In PWM DUAL OUTPUT Mode, the timer also generates a second PWM output signal, Timer Output Complement ($\overline{T_{OUT}}$). A programmable deadband can be configured (PWMD field) to delay (0–128 system clock cycles) the Low to a High (inactive to active) output transitions on these two pins. This configuration ensures a time gap between the deassertion of one PWM output to the assertion of its complement.

Observe the following procedure to configure a timer for either PWM SINGLE or DUAL OUTPUT Mode and initiate PWM operation:

1. Write to the Timer Control registers to:
   a. Disable the timer.
   b. Configure the timer for the selected PWM Mode.
   c. Set the prescale value.
   d. Set the initial logic level (High or Low) and PWM High/Low transition for the Timer Output alternate function with the TPOL bit.
   e. Set the deadband delay (DUAL OUTPUT Mode) with the PWMD field.
2. Write to the Timer High and Low Byte registers to set the starting count value (typically 0001h). The starting count value only affects the first pass in PWM Mode. After the first timer reset in PWM Mode, counting begins at the reset value of 0001h.
3. Write to the PWM High and Low Byte registers to set the PWM value.
4. 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.
5. If appropriate, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
6. Configure the associated GPIO port pin(s) for the Timer Output alternate function.
7. Write to the Timer Control 1 Register to enable the timer and initiate counting.

The PWM period is determined by the following equation:

$$\text{PWM Period (s)} = \frac{\text{Reload Value x Prescaler}}{\text{System Clock Frequency (Hz)}}$$

If an initial starting value other than 0001h is loaded into the Timer High and Low Byte registers, use the ONE-SHOT Mode equation 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 determined by the equation:

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

If TPOL is set to 1, the ratio of the PWM output High time to the total period is determined by the equation:

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

CAPTURE Modes

There are three capture modes that provide slightly different methods for recording the time or time interval between, Timer Input events. These modes are CAPTURE Mode, CAPTURE RESTART Mode and CAPTURE COMPARE Mode. In all three modes, when the appropriate Timer Input transition (capture event) occurs, the timer counter value is captured and stored in the PWM High and Low Byte registers. The TPOL bit in the Timer Control 1 Register determines whether the Capture occurs on a rising edge or a falling edge of the Timer Input signal. The TICONFIG bit determines whether interrupts are generated on capture events, reload events, or both. The INCAP bit in Timer Control 0 Register clears to indicate an interrupt caused by a reload event and sets to indicate the timer interrupt is caused by an input capture event.

There is a delay from the input event to the timer capture of 2–3 system clock cycles, due to internal synchronization logic.

If the Timer Output alternate function is enabled, the Timer Output pin changes state (from Low to High or High to Low) at timer reload. The initial value is determined by the TPOL bit.

CAPTURE Mode. In CAPTURE Mode and after it is enabled, the timer counts continuously and rolls over from FFFFh to 0000h. When the capture event occurs, the timer counter value is captured and stored in the PWM High and Low Byte registers, an interrupt is generated and the timer continues counting. 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.

CAPTURE RESTART Mode. In CAPTURE RESTART Mode, after it is enabled, the timer counts continuously until the capture event occurs or the timer count reaches the 16-bit compare value stored in the Timer Reload High and Low Byte registers. If the capture

event occurs first, the timer counter value is captured and stored in the PWM High and Low Byte registers. An interrupt is generated, the count value in the Timer High and Low Byte registers is reset to 0001h and counting resumes. If no capture event occurs, 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.

CAPTURE/COMPARE Mode. CAPTURE/COMPARE Mode is identical to CAPTURE RESTART Mode, except that counting does not start until the first external Timer Input transition occurs. Every subsequent transition (after the first) of the Timer Input signal captures the current count value. 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. If no capture event occurs, 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.

Observe the following procedure to configure a timer for one of these capture modes and initiate the count:

1. Write to the Timer Control registers to:
   a. Disable the timer.
   b. Configure the timer for the selected capture mode.
   c. Set the prescale value.
   d. Set the capture edge (rising or falling) for the Timer Input.
   e. Configure the timer interrupt to be generated at the input capture event, the reload event, or both, by setting the TICONFIG field.
2. Write to the Timer High and Low Byte registers to set the starting count value (typically 0001h).
3. Write to the Timer Reload High and Low Byte registers to set the reload value.
4. If appropriate, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
5. Configure the associated GPIO port pin for the Timer Input alternate function.
6. Write to the Timer Control 1 Register to enable the timer. In CAPTURE and CAPTURE RESTART modes, the timer begins counting. In CAPTURE COMPARE Mode, the timer does not start counting until the first input transition occurs.

In capture modes, the elapsed time from a timer start to a capture event can be calculated using the following equation (Start Value = 1):

$$\text{Capture Elapsed Time (s)} = \frac{(\text{Capture Value} - \text{Start Value} + 1) \times \text{Prescale}}{\text{System Clock Frequency (Hz)}}$$

COMPARE Mode

In COMPARE Mode, the timer counts up to the 16-bit compare value stored in the Timer Reload High and Low Byte registers. After reaching the compare value, the timer generates an interrupt and counting continues (the timer value is not reset to 0001h). If the Timer Output alternate function is enabled, the Timer Output pin changes state (from Low to High or High to Low).

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

Observe the following procedure to configure a timer for COMPARE Mode and initiate the count:

1. Write to the Timer Control registers to:
   a. Disable the timer.
   b. Configure the timer for COMPARE Mode.
   c. Set the prescale value.
   d. Set the initial logic level (High or Low) for the Timer Output alternate function, if appropriate.
2. Write to the Timer High and Low Byte registers to set the starting count value.
3. Write to the Timer Reload High and Low Byte registers to set the Compare value.
4. If appropriate, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
5. If using the Timer Output function, configure the associated GPIO port pin for the Timer Output alternate function.
6. Write to the Timer Control 1 Register to enable the timer and initiate counting.

The compare time is calculated by the following equation (Start Value = 1):

$$\text{Compare Mode Time (s)} = \frac{(\text{Compare Value} - \text{Start Value} + 1) \times \text{Prescale}}{\text{System Clock Frequency (Hz)}}$$

GATED Mode

In GATED Mode, the timer counts only when the Timer Input signal is in its active state, as determined by the TPOL bit in the Timer Control 1 Register. When the Timer Input signal is active, counting begins. A timer interrupt is generated when the Timer Input signal transits from active to inactive state, a timer reload occurs, or both, depending on TICONFIG[1:0]. To determine if a Timer Input signal deassertion generated the interrupt, read the associated GPIO input value and compare it to the value stored in the TPOL bit.

The timer counts up to the 16-bit reload value stored in the Timer Reload High and Low Byte registers. On 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 continues as long as the Timer Input signal is active. Also, if the Timer Output alternate function is enabled, the Timer Output pin changes state (from Low to High or from High to Low) at timer reload.

Observe the following procedure for configuring a timer for GATED Mode and initiating the count:

1. Write to the Timer Control registers to:
   a. Disable the timer.
   b. Configure the timer for GATED Mode.
   c. Set the prescale value.
   d. Select the active state of the Timer Input through the TPOL bit.
2. Write to the Timer High and Low Byte registers to set the starting count value. This setting only affects the first pass in GATED Mode. After the first timer reset in GATED Mode, counting begins at the reset value of 0001h.
3. Write to the Timer Reload High and Low Byte registers to set the reload value.
4. If appropriate, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
5. Configure the timer interrupt to be generated only at the input deassertion event, the reload event, or both, by setting the TICONFIG field of the Timer Control 0 Register.
6. Configure the associated GPIO port pin for the Timer Input alternate function.
7. Write to the Timer Control 1 Register to enable the timer.
8. The timer counts when the Timer Input is equal to the TPOL bit.

Reading the Timer Count Values

The current count value in the timers can be read while counting (enabled). This Read 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 into a holding register. A subsequent Read from 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 timer is not enabled, a Read from the Timer Low Byte Register returns the actual value in the counter.

Timer 0 High and Low Byte Registers

The Timer 0 High and Low Byte (T0H and T0L) registers, shown in Tables 58 and 59, contain the current 16-bit timer count value. When the timer is enabled, a Read from T0H causes the value in T0L to be stored in a temporary holding register. A Read from T0L always returns this temporary register when the timer is enabled. When the timer is disabled, Reads from the T0L are direct from this temporary register.

Caution: Writing to the Timer High and Low Byte registers while the timer is enabled is not recommended. 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 58. Timer 0 High Byte Register (T0H)

Bit	7	6	5	4	3	2	1	0
Field	TH							
RESET	00H							
R/W	R/W							
Address	F00H							

Table 59. Timer 0 Low Byte Register (T0L)

Bit	7	6	5	4	3	2	1	0
Field	TL							
RESET	01H							
R/W	R/W							
Address	F01H							

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

Timer 0 Reload High and Low Byte Registers

The Timer 0 Reload High and Low Byte (T0RH and T0RL) registers, shown in Tables 60 and 61, store a 16-bit reload value, {TRH[7:0], TRL[7:0]}. Values written to the Timer Reload High Byte Register are stored in a temporary holding register. When a Write to the Timer Reload Low Byte Register occurs, the temporary holding register value is written to the Timer High Byte Register. This operation allows simultaneous updates of the 16-bit timer reload value.

Table 60. Timer 0 Reload High Byte Register (T0RH)

Bit	7	6	5	4	3	2	1	0
Field	TRH							
RESET	FFH							
R/W	R/W							
Address	F02H							

Table 61. Timer 0 Reload Low Byte Register (T0RL)

Bit	7	6	5	4	3	2	1	0
Field	TRL							
RESET	FF							
R/W	R/W							
Address	F03H							

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 sets the maximum count value which initiates a timer reload to 0001H.

Timer 0 PWM High and Low Byte Registers

The Timer 0 PWM High and Low Byte (T0PWMH and T0PWML) registers, shown in Tables 62 and 63, define PWM operations. These registers also store the timer counter values for the Capture modes.

The two bytes {PWMH[7:0], PWML[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 (T0CTL1).

The T0PWMH and T0PWML registers also store the 16-bit captured timer value when operating in CAPTURE or CAPTURE/COMPARE modes.

Table 62. Timer 0 PWM High Byte Register (T0PWMH)

Bit	7	6	5	4	3	2	1	0
Field	PWMH							
RESET	00H							
R/W	R/W							
Address	F04H							

Table 63. Timer 0 PWM Low Byte Register (T0PWML)

Bit	7	6	5	4	3	2	1	0
Field	PWML							
RESET	00H							
R/W	R/W							
Address	F05H							

Bit	Description
[7:0] PWMH, PWML	**PWM High and Low Bytes** These two bytes, {PWMH[7:0], PWML[7:0]}, form a 16-bit value which 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 (T0CTL1). The T0PWMH and T0PWML registers also store the 16-bit captured timer value when operating in CAPTURE or CAPTURE/COMPARE modes.

Timer 0 Control Registers

Two Timer 0 Control registers determine timer configuration (T0CTL0) and operation (T0CTL1).

Timer 0 Control 0 Register

The Timer 0 Control 0 (T0CTL0) Register, together with the Timer 0 Control 1 (T0CTL1) Register, determines the timer configuration and operation. See Table 64.

Table 64. Timer 0 Control 0 Register (T0CTL0)

Bit	7	6	5	4	3	2	1	0
Field	TMODE[3]	TICONFIG		TINSEL	PWMD			INCAP
RESET	0	00		0	000			0
R/W	R/W	R/W		R/W	R/W			R
Address	F06H							

Bit	Description
[7] TMODE[3]	**Timer Mode High Bit** This bit along with TMODE[2:0] field in T0CTL1 Register determines the operating mode of the timer. This is the most significant bit of the Timer Mode selection value. For more details, see the T0CTL1 Register description.
[6:5] TICONFIG	**Timer Interrupt Configuration** This field configures timer interrupt definitions. These bits affect all modes. The effect, per mode, is defined below. ONE SHOT, CONTINUOUS, COUNTER, PWM, COMPARE, DUAL PWM, TRIGGERED ONE-SHOT, COMPARATOR COUNTER: 0x = Timer interrupt occurs on reload. 10 = Timer interrupts are disabled. 11 = Timer Interrupt occurs on reload. GATED: 0x = Timer interrupt occurs on reload or inactive gate edge. 10 = Timer interrupt occurs on inactive gate edge. 11 = Timer interrupt occurs on reload. CAPTURE, CAPTURE/COMPARE, CAPTURE RESTART: 0x = Timer interrupt occurs on reload and capture. 10 = Timer interrupt occurs on capture only. 11 = Timer interrupt occurs on reload only.
[4] TINSEL	**Timer Input Select** 0 = Timer input is the Timer input pin. 1 = Timer input is the comparator output.

Bit	Description (Continued)
[3:1] PWMD	**PWM Delay Value** This field is a programmable delay to control the number of additional system clock cycles following a PWM or Reload compare before the Timer Output or the Timer Output Complement is switched to the active state. This field ensures a time gap between the deassertion of one PWM output to the assertion of its complement. 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] INCAP	**Input Capture Event** 0 = Previous timer interrupt is not a result of a Timer Input Capture Event. 1 = Previous timer interrupt is a result of a Timer Input Capture Event.

Timer 0 Control 1 Register

The Timer 0 Control 1 (T0CTL1) Register, shown in Table 65, enables/disables the timer, sets the prescaler value and determines the timer operating mode.

Table 65. Timer 0 Control 1 Register (T0CTL1)

Bit	7	6	5	4	3	2	1	0
Field	TEN	TPOL	PRES			TMODE		
RESET	0	0	000			000		
R/W	R/W	R/W	R/W			R/W		
Address	F07H							

Bit	Description
[7] TEN	**Timer Enable** 0 = Timer is disabled. 1 = Timer is enabled.

Bit	Description (Continued)
[6] TPOL	**Timer Input/Output Polarity** This bit is a function of the current operating mode of the timer. It determines the polarity of the input and/or output signal. When the timer is disabled, the Timer Output signal is set to the value of this bit. **ONE-SHOT Mode** If the timer is enabled the Timer Output signal pulses (changes state) for one system clock cycle after timer Reload. **CONTINUOUS Mode** If the timer is enabled the Timer Output signal is complemented after timer Reload. COUNTER Mode—If the timer is enabled the Timer Output signal is complemented after timer reload. 0 = Count occurs on the rising edge of the Timer Input signal. 1 = Count occurs on the falling edge of the Timer Input signal. **PWM SINGLE OUTPUT Mode** When enabled, the Timer Output is forced to $\overline{\text{TPOL}}$ after PWM count match and forced back to TPOL after Reload. **CAPTURE Mode** If the timer is enabled, the Timer Output signal is complemented after timer Reload. 0 = Count is captured on the rising edge of the Timer Input signal. 1 = Count is captured on the falling edge of the Timer Input signal. **COMPARE Mode** The Timer Output signal is complemented after timer Reload. **GATED Mode** The Timer Output signal is complemented after timer Reload. 0 = Timer counts when the Timer Input signal is High and interrupts are generated on the falling edge of the Timer Input. 1 = Timer counts when the Timer Input signal is Low and interrupts are generated on the rising edge of the Timer Input. **CAPTURE/COMPARE Mode** If the timer is enabled, the Timer Output signal is complemented after timer Reload. 0 = Counting starts on the first rising edge of the Timer Input signal. The current count is captured on subsequent rising edges of the Timer Input signal. 1 = Counting starts on the first falling edge of the Timer Input signal. The current count is captured on subsequent falling edges of the Timer Input signal.

Bit	Description (Continued)
[6] TPOL (cont'd)	**PWM DUAL OUTPUT Mode** If enabled, the Timer Output is set=$\overline{\text{TPOL}}$ after PWM match and set=TPOL after Reload. If enabled the Timer Output Complement takes on the opposite value of the Timer Output. The PWMD field in the T0CTL1 Register determines an optional added delay on the assertion (Low to High) transition of both Timer Output and the Timer Output Complement for deadband generation. **CAPTURE RESTART Mode** If the timer is enabled the Timer Output signal is complemented after timer Reload. 0 = Count is captured on the rising edge of the Timer Input signal. 1 = Count is captured on the falling edge of the Timer Input signal. **COMPARATOR COUNTER Mode** If the timer is enabled the Timer Output signal is complemented after timer Reload. 0 = Count is captured on the rising edge of the Timer Input signal. 1 = Count is captured on the falling edge of the Timer Input signal. **TRIGGERED ONE-SHOT Mode** If the timer is enabled the Timer Output signal is complemented after timer Reload. 0 = The timer triggers on a Low to High transition on the input. 1 = The timer triggers on a High to Low transition on the input.

Bit	Description (Continued)
[5:3] PRES	**Prescale** The timer input clock is divided by 2^{PRES}, where 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 T0CTL0 Register determines the operating mode of the timer. TMODE[3:0] selects from 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 = COMPARATOR COUNTER Mode. 1011 = TRIGGERED ONE-SHOT Mode.

LIN-UART

The Local Interconnect Network Universal Asynchronous Receiver/Transmitter (LIN-UART) is a full-duplex communication channel capable of handling asynchronous data transfers in standard UART applications as well as providing LIN protocol support.

Features of the LIN-UART include:

- 8-bit asynchronous data transfer
- Selectable even- and odd-parity generation and checking
- Option of one or two 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 may 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 versus Rx data when sending
- Configurable digital noise filter on Receive Data line

Architecture

The LIN-UART consists of three primary functional blocks: transmitter, receiver and BRG. The LIN-UART's transmitter and receiver function independently, but employ the same baud rate and data format. The basic UART operation is enhanced by the Noise Filter and IrDA blocks. Figure 11 displays the LIN-UART architecture.

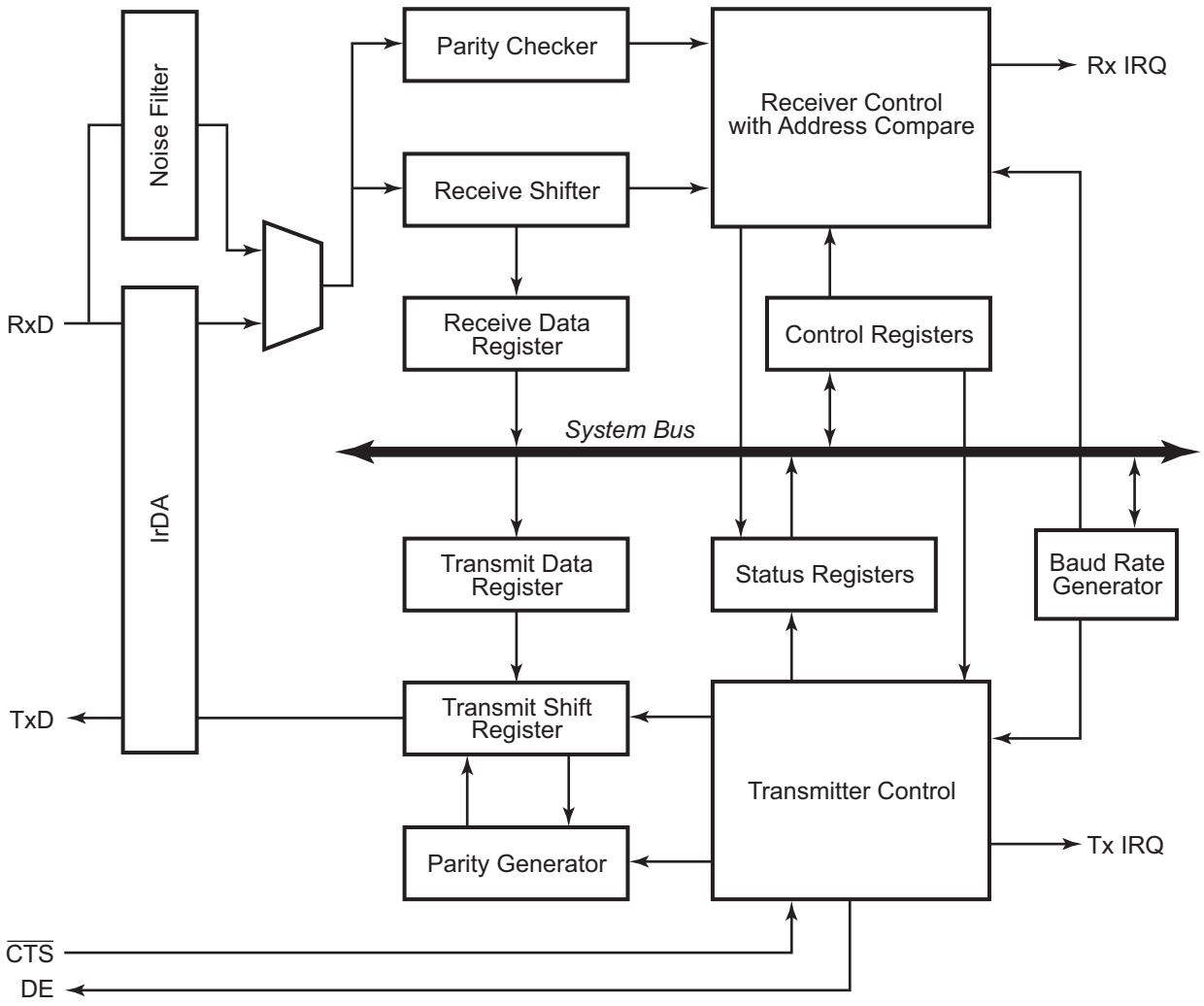


Figure 11. LIN-UART Block Diagram

Data Format for Standard UART Modes

The LIN-UART always transmits and receives data in an 8-bit data format, 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 1 or 2 active High stop bits. Figures 12 and 13 display the asynchronous data format employed by the LIN-UART without parity and with parity, respectively.

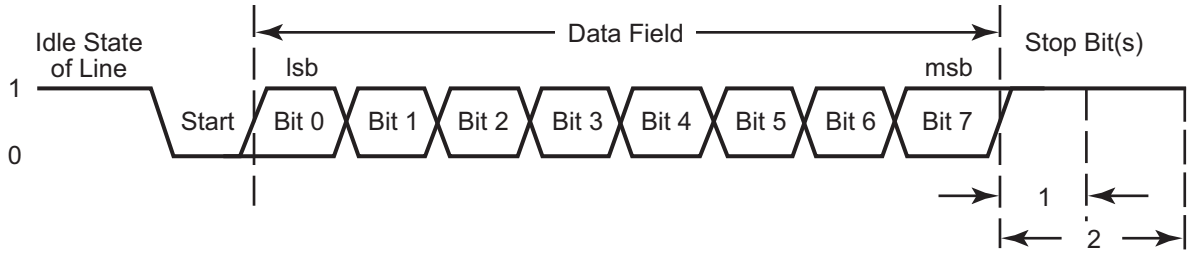


Figure 12. LIN-UART Asynchronous Data Format without Parity

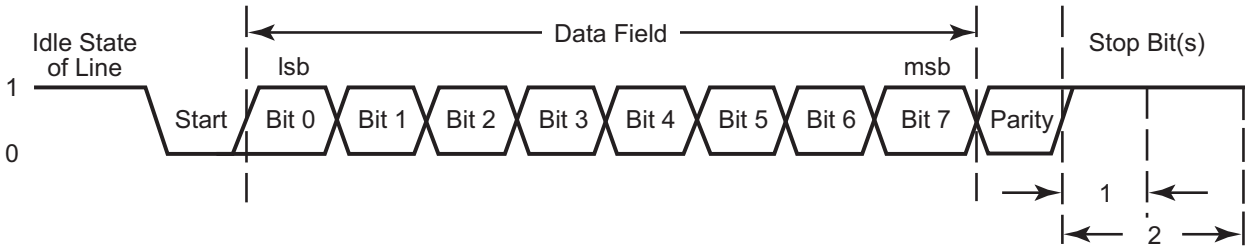


Figure 13. LIN-UART Asynchronous Data Format with Parity

Transmitting Data using Polled Method

Observe the following procedure to transmit data using the polled operating method:

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

   Set the MULTIPROCESSOR Mode Select (MPEN) to enable MULTIPROCESSOR Mode.
4. Write to the LIN-UART Control 0 Register to:
   a. Set the transmit enable bit (TEN) to enable the LIN-UART for data transmission.
   b. If parity is appropriate and MULTIPROCESSOR Mode is not enabled, set the parity enable bit (PEN) 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.

5. Check the TDRE bit in the LIN-UART Status 0 Register to determine if the Transmit Data Register is empty (indicated by a 1). If empty, continue to Step 6. 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.
6. If in MULTIPROCESSOR Mode, write the LIN-UART 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.
7. Write the data byte to the LIN-UART Transmit Data Register. The transmitter automatically transfers the data to the Transmit Shift Register and transmits the data.
8. If appropriate and if MULTIPROCESSOR Mode is enabled, make any changes to the Multiprocessor Bit Transmitter (MPBT) value.
9. To transmit additional bytes, return to Step 4.

Transmitting Data using Interrupt-Driven Method

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

1. Write to the LIN-UART Baud Rate High and Low Byte registers to set the appropriate baud rate.
2. Enable the LIN-UART 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 LIN-UART Transmitter interrupt and set the appropriate priority.
5. If MULTIPROCESSOR Mode is appropriate, write to the LIN-UART Control 1 Register to enable Multiprocessor (9-bit) mode functions.

   Set the MULTIPROCESSOR Mode Select (MPEN) to enable MULTIPROCESSOR Mode.
6. Write to the LIN-UART Control 0 Register to:
   a. Set the transmit enable bit (TEN) to enable the LIN-UART for data transmission
   b. If MULTIPROCESSOR Mode is not enabled, 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.

7. Execute an EI instruction to enable interrupts.

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

1. If in MULTIPROCESSOR Mode, write the LIN-UART 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.
2. Write the data byte to the LIN-UART Transmit Data Register. The transmitter automatically transfers the data to the Transmit Shift Register and transmits the data.
3. Execute 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 software to write the data to the Transmit Data Register instead of invoking the interrupt-service routine.

Receiving Data using Polled Method

Observe the following procedure to configure the LIN-UART for polled data reception:

1. Write to the LIN-UART Baud Rate High and Low Byte registers to set the appropriate baud rate.
2. Enable the LIN-UART pin functions by configuring the associated GPIO port pins for alternate function operation.
3. Write to the LIN-UART Control 1 Register to enable MULTIPROCESSOR Mode functions, if appropriate.
4. Write to the LIN-UART Control 0 Register to:
   a. Set the receive enable bit (REN) to enable the LIN-UART for data reception.
   b. If MULTIPROCESSOR Mode is not enabled, enable parity, if appropriate and select either even or odd parity.

5. Check the RDA bit in the LIN-UART 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 <CrossRef>Step 6. If the Receive Data Register is empty (indicated by a 0), continue to monitor the RDA bit awaiting reception of the valid data.
6. Read data from the LIN-UART 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 <CrossRef>Step 5 to receive additional data.

Receiving Data using Interrupt-Driven Method

The LIN-UART Receiver interrupt indicates the availability of new data (as well as error conditions).

Observe the following procedure to configure the LIN-UART receiver for interrupt-driven operation:

1. Write to the LIN-UART Baud Rate High and Low Byte registers to set the appropriate baud rate.
2. Enable the LIN-UART 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 LIN-UART Receiver interrupt and set the appropriate priority.
5. Clear the LIN-UART Receiver interrupt in the applicable Interrupt Request Register.
6. Write to the LIN-UART Control 1 Register to enable MULTIPROCESSOR (9-bit) Mode functions, if appropriate.
   a. Set the MULTIPROCESSOR Mode Select (MPEN) to enable MULTIPROCESSOR Mode.
   b. Set MULTIPROCESSOR Mode bits MPMD[1:0] to select the appropriate address matching scheme.
   c. Configure the LIN-UART to interrupt on received data and errors or errors only (interrupt on errors only is unlikely to be useful for Z8FMC16100 Series MCU devices without a DMA block).
7. Write the device address to the Address Compare Register (automatic multiprocessor modes only).
8. Write to the LIN-UART Control 0 Register to:

a. Set the receive enable bit (REN) to enable the LIN-UART for data reception.
b. If MULTIPROCESSOR Mode is not enabled, enable parity, if appropriate and select either even or odd parity.

9. Execute an EI instruction to enable interrupts.

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

1. Check the LIN-UART Status 0 Register to determine the source of the interrupt-error, break, or received data.
2. If the interrupt was due to data available, read the data from the LIN-UART 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 interrupt-service routine and await more data.

Clear To Send Operation

The Clear To Send ($\overline{\text{CTS}}$) pin, if enabled by the CTSE bit of the LIN-UART 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 beginning any new character transmission. To delay transmission of the next data character, an external receiver must deassert $\overline{\text{CTS}}$ 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 $\overline{\text{CTS}}$ deasserts in the middle of a character transmission, the current character is sent completely.

External Driver Enable

The LIN-UART provides a Driver Enable (DE) signal for off-chip bus transceivers. This feature reduces the software overhead associated with 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 that envelopes the entire transmitted data frame including parity and Stop bits as displayed in Figure 14. The Driver Enable signal asserts when a byte is written to the LIN-UART 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 allows a setup time to enable the transceiver.

The Driver Enable signal deasserts one system clock period after the last STOP bit is transmitted. This one 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 LIN-UART Control Register 1 sets the polarity of the Driver Enable signal.

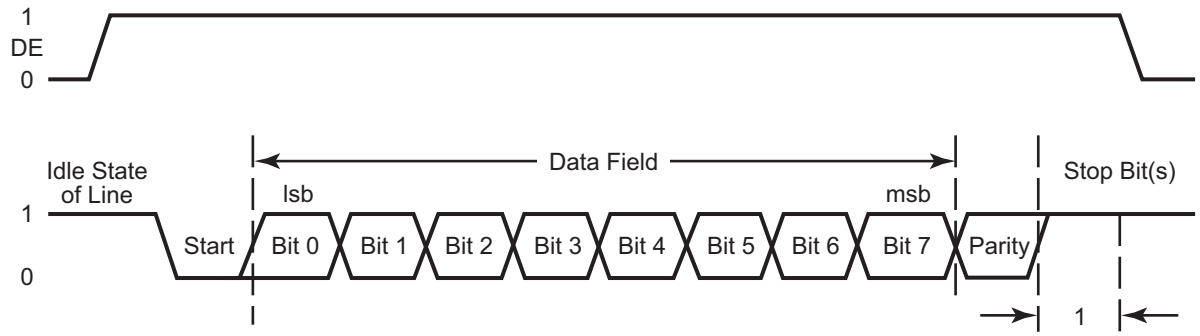


Figure 14. LIN-UART Driver Enable Signal Timing (shown with 1 Stop Bit and Parity)

The Driver Enable to START bit set-up time is calculated as follows:

$$\frac{1}{\text{Baud Rate (Hz)}} \leq \text{DE to Start Bit Setup Time(s)} \leq \frac{2}{\text{Baud Rate (Hz)}}$$

LIN-UART Special Modes

The special modes of the LIN-UART are:

- MULTIPROCESSOR Mode
- LIN Protocol Mode

The LIN-UART features a common control Register (Control 0) that contains a unique register address and several mode-specific control registers (Multiprocessor Control, Noise Filter Control and LIN 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.

MULTIPROCESSOR Mode

The LIN-UART 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 bit (MP) is transmitted immediately following the 8 bits of data and immediately preceding the STOP bit(s) as displayed in Figure 15.

The character format is shown below.

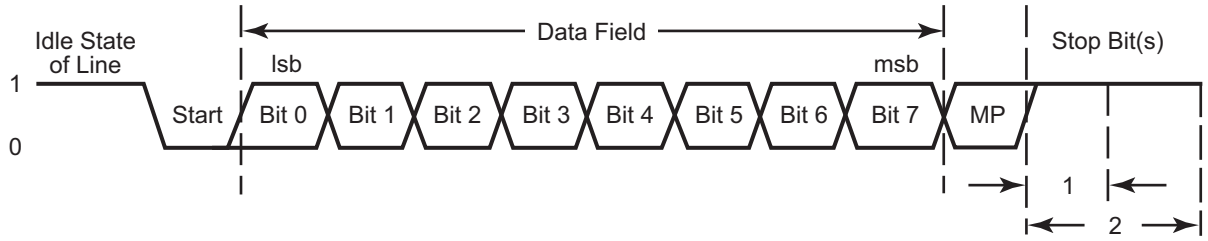


Figure 15. LIN-UART Asynchronous MULTIPROCESSOR Mode Data Format

In MULTIPROCESSOR (9-bit) Mode, the Parity bit location (9th bit) becomes the MULTIPROCESSOR control bit. The LIN-UART Control 1 and Status 1 registers provide MULTIPROCESSOR (9-bit) Mode control and status information. If an automatic address matching scheme is enabled, the LIN-UART Address Compare Register holds the network address of the device.

MULTIPROCESSOR Mode Receive Interrupts

When MULTIPROCESSOR (9-bit) Mode is enabled, the LIN-UART processes only frames addressed to it. The determination of whether a frame of data is addressed to the LIN-UART 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, as it does not need to access the LIN-UART when it receives data directed to other devices on the multinode network. The following 3 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 LIN-UART Control 1 Register. For all MULTIPROCESSOR modes, bit MPEN of the LIN-UART 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 LIN-UART 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 LIN-UART Status 1 Register for each incoming byte. If MPRX=1, a new frame has begun. If the address of this new frame is different from the LIN-UART's address, then MPMD[0] must be set to 1 by software, causing the LIN-UART interrupts to go inactive until the next address byte. If the new frame's address matches the LIN-UART's, then the data in the new frame is processed.

The second scheme is enabled by setting MPMD[1:0] to 10B and writing the LIN-UART's address into the LIN-UART Address Compare Register. This mode introduces more hardware control, interrupting only on frames that match the LIN-UART's address. When an incoming address byte does not match the LIN-UART'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 LIN-UART Status 1 Register. When the next address byte occurs, the hardware compares it to the LIN-UART's address. If there is a match, the interrupt occurs and the NEWFRM bit is set for the first byte of the new frame. If there is no match, the LIN-UART 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 LIN-UART's address into the LIN-UART 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.

LIN Protocol Mode

The LIN protocol as supported by the LIN-UART 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 LIN-UART 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 LIN-UART asserts the PLE bit (physical layer error bit in Status 0 Register). The *message frame* time-out aspect of the protocol is left to software, requiring the use of an additional general purpose timer. The LIN Mode of the LIN-UART 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 through 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.

- Parity Error (PE bit in 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 LIN-UART is transmitting. This applies to both Master and Slave operating modes.
- The Break Detect interrupt (BRKD bit in Status 0 Register) indicates when a Break is detected by the slave (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 Status Register.
- The Break Detect interrupt (BRKD bit in Status 0 Register) indicates when a Wake-up message has been received if the LIN-UART is in LinSleep state.
- In LIN SLAVE Mode, if the BRG counter overflows while measuring the autobaud period (Start bit to beginning of bit 7 of 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 erroroccurs. The OE bit is also set if a data overrun error occurs.

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 no more than ±1.5%. These accuracy requirements must include effects such as voltage and temperature drift during operation.

Before sending or 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.

In order to autobaud with the required accuracy, the LIN slave system clock must be at least 100 times the baud rate.

LIN Mode Initialization and Operation

The LIN protocol mode is selected by setting either the LMST (LIN Master) or LSLV (LIN Slave) and optionally (for LIN slave) the Autobaud Enable (ABEN) bits in the LIN Control Register. To access the LIN Control Register, the MSEL (Mode Select) field of the

LIN-UART Mode Select/Status Register must be = 010B. The LIN-UART Control 0 Register must be initialized with TEN = 1, REN = 1, all other bits = 0.

In addition to the LMST, LSLV and ABEN bits in the LIN Control Register, a LinState[1:0] field exists that 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.

The Noise Filter may also need to be enabled and configured when interfacing to a LIN bus.

LIN MASTER Mode Operation

LIN MASTER Mode is selected by setting LMST = 1, LSLV = 0, ABEN = 0, LinState[1:0] = 11B. If the LIN bus protocol indicates the bus is required go into the LIN sleep state, the LinState[1:0] bits must be set = 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 LIN-UART starts the Break once the SBRK bit is set and any character transmission currently underway has completed. The SBRK bit is deasserted by hardware once the break is completed.

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 after 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 is signalled by the receive data interrupt (RDA bit is 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 has completed.

LIN Sleep Mode

While the LIN bus is in the sleep state, the CPU can be in either low power STOP Mode, in HALT Mode, or in normal operational state. Any device on the LIN bus may 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 5 ms. Transmitting a 00h character is one way to transmit the wake-up message.

If the CPU is in STOP Mode, the LIN-UART 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 may preclude making an accurate measurement of the Wake-up message duration.

If the CPU is in HALT or operational mode, the LIN-UART (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 may be obtained by reading the RxBreakLength field in the Mode Status Register. After a Wake-up message has been detected, the LIN-UART can be placed (by software) into either 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 LIN-UART is disabled, the Wake-up message can be 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 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.

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 LIN-UART detects the Break and generates an interrupt to the CPU. The duration of the Break is observable in the RxBreakLength field of the Mode Status Register. A Break of less than 11 bit times in duration does not generate a break interrupt when the LIN-UART is in Wait for Break state. If the Break duration exceeds 15 bit times, the RxBreakLength field contains the value Fh.

Following the Break the LIN-UART 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 standard. 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 minimum 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 LIN-UART hardware transits to the *Active* state where the Identifier character is received and the characters of the *Response* section of the message are sent or received. The Slave remains in the *Active* state until a Break is received or software forces a state change. When it is in Active State (autobaud has completed), a Break of 10 or more bit times is recognized and will cause a transition to the Autobaud state.

If the Identifier character indicates that this slave device is not participating in the message, software can set the LinState[1:0] = 01b (Wait for Break State) to ignore the rest of the message. No further receive interrupts will occur until the next Break.

LIN-UART Interrupts

The LIN-UART features separate interrupts for the transmitter and receiver. In addition, when the LIN-UART primary functionality is disabled, the BRG can also function as a basic timer with interrupt capability.

Transmitter Interrupts

The transmitter generates a single interrupt when the Transmit Data Register Empty bit (TDRE) is set to 1. This 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 the first bit of a character out. At this point, the Transmit Data Register may be written with the next character to send. This provides 7 bit periods of latency to load the Transmit Data Register before the Transmit Shift Register completes shifting the current character. Writing to the LIN-UART Transmit Data Register clears the TDRE bit to 0.

Receiver Interrupts

The receiver generates an interrupt when any of the following occurs:

- A data byte is received and is available in the LIN-UART Receive Data Register. This interrupt can be disabled independent of the other receiver interrupt sources through the RDAIRQ bit (this feature is useful in devices which support DMA). The received data interrupt occurs after the receive character is 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 (physical layer error in LIN Mode)

LIN-UART Overrun Errors

When an overrun error condition occurs the LIN-UART prevents overwriting of the valid data currently in the Receive Data Register. The Break Detect and Overrun status bits are not displayed until 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 may not contain valid data and must be ignored. The BRKD bit indicates if the overrun is caused due to a break condition on the line. After reading the status byte indicating an overrun error, the Receive Data Register must be read again to clear the error bits in the LIN-UART Status 0 Register.

In LIN Mode, an Overrun Error is signaled 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.

LIN-UART Data-Handling and Error-Handling Procedure

Figure 16 displays the recommended procedure for use in LIN-UART receiver interrupt service routines.

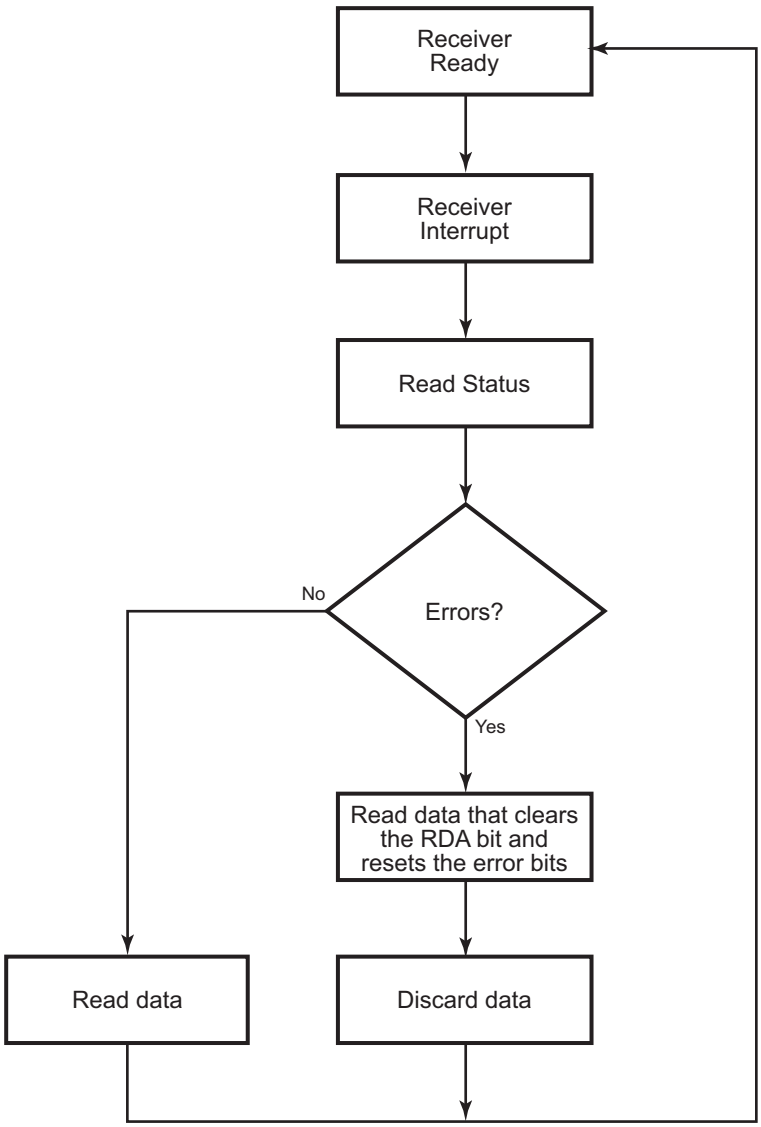


Figure 16. LIN-UART Receiver Interrupt Service Routine Flow

Baud Rate Generator Interrupts

If the BRGCTL bit of the Multiprocessor Control Register (LIN-UART Control 1 Register with MSEL = 000b) is set and the REN bit of the Control 0 Register is 0, the LIN-UART Receiver interrupt asserts when the LIN-UART Baud Rate Generator reloads. This allows the BRG to function as an additional counter if the LIN-UART receiver functionality is not employed. The transmitter is enabled in this mode.

LIN-UART Baud Rate Generator

The LIN-UART 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 LIN-UART 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 LIN-UART. The LIN-UART data rate is calculated using the following equation for normal UART operation:

$$\text{UART Data Rate (bps)} = \frac{\text{System Clock Frequency (Hz)}}{16 \times \text{UART Baud Rate Divisor Value}}$$

The LIN-UART data rate is calculated using the following equation for LIN Mode UART operation:

$$\text{UART Data Rate (bps)} = \frac{\text{System Clock Frequency (Hz)}}{\text{UART Baud Rate Divisor Value}}$$

When the LIN-UART is disabled, the Baud Rate Generator can function as a basic 16-bit timer with interrupt on time-out. To configure the Baud Rate Generator as a timer with interrupt on time-out, complete the following procedure:

1. Disable the LIN-UART receiver by clearing the REN bit in the LIN-UART Control 0 Register to 0 (TEN bit may be asserted, transmit activity may occur).
2. Load the appropriate 16-bit count value into the LIN-UART 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 LIN-UART Control 1 Register to 1.

Noise Filter

A noise filter circuit is included which filters noise on a digital input signal (such as UART Receive Data) before the data is sampled by the block. This is likely to be a requirement for protocols with 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 widths range from 4 bits to 11 bits.

- The digital filter output features hysteresis.
- Provides an active low *Saturated State* output (FiltSatB) used as an indication of the presence of noise.

Architecture

Figure 17 displays how the noise filter is integrated with the LIN-UART for use on a LIN network.

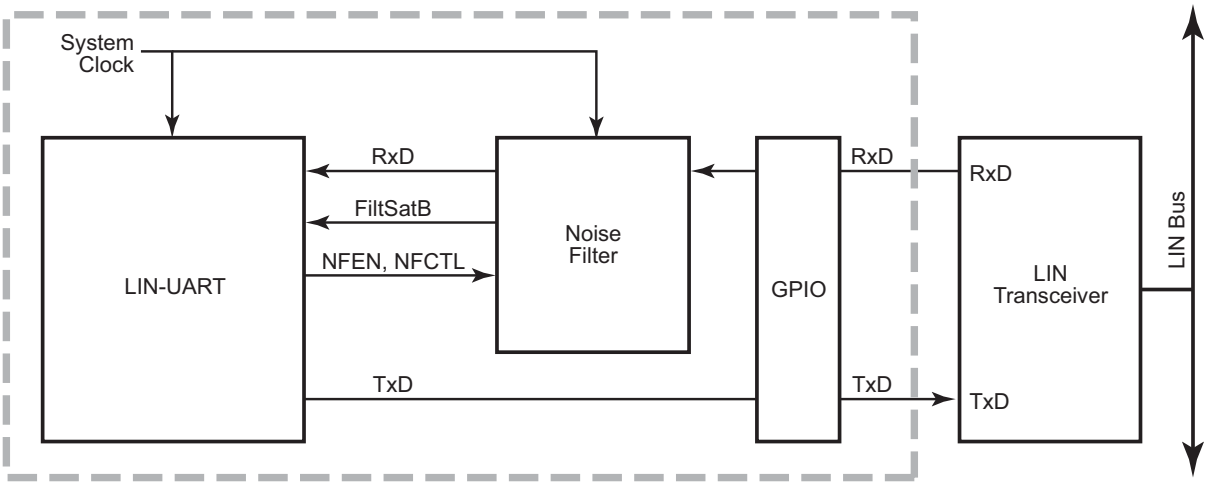


Figure 17. Noise Filter System Block Diagram

Operation

Figure 18 displays 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` and switches from 0 to 1 when the counter counts up from `10b` to `11b`. The noise filter delays the receive data by three System Clock cycles.

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 center of bit time) does not mean the sampled data is incorrect, just that the filter is not in its *saturated* state of all 1s or all 0s. If FiltSatB = 1, when RxD is sampled during a receive character, 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.

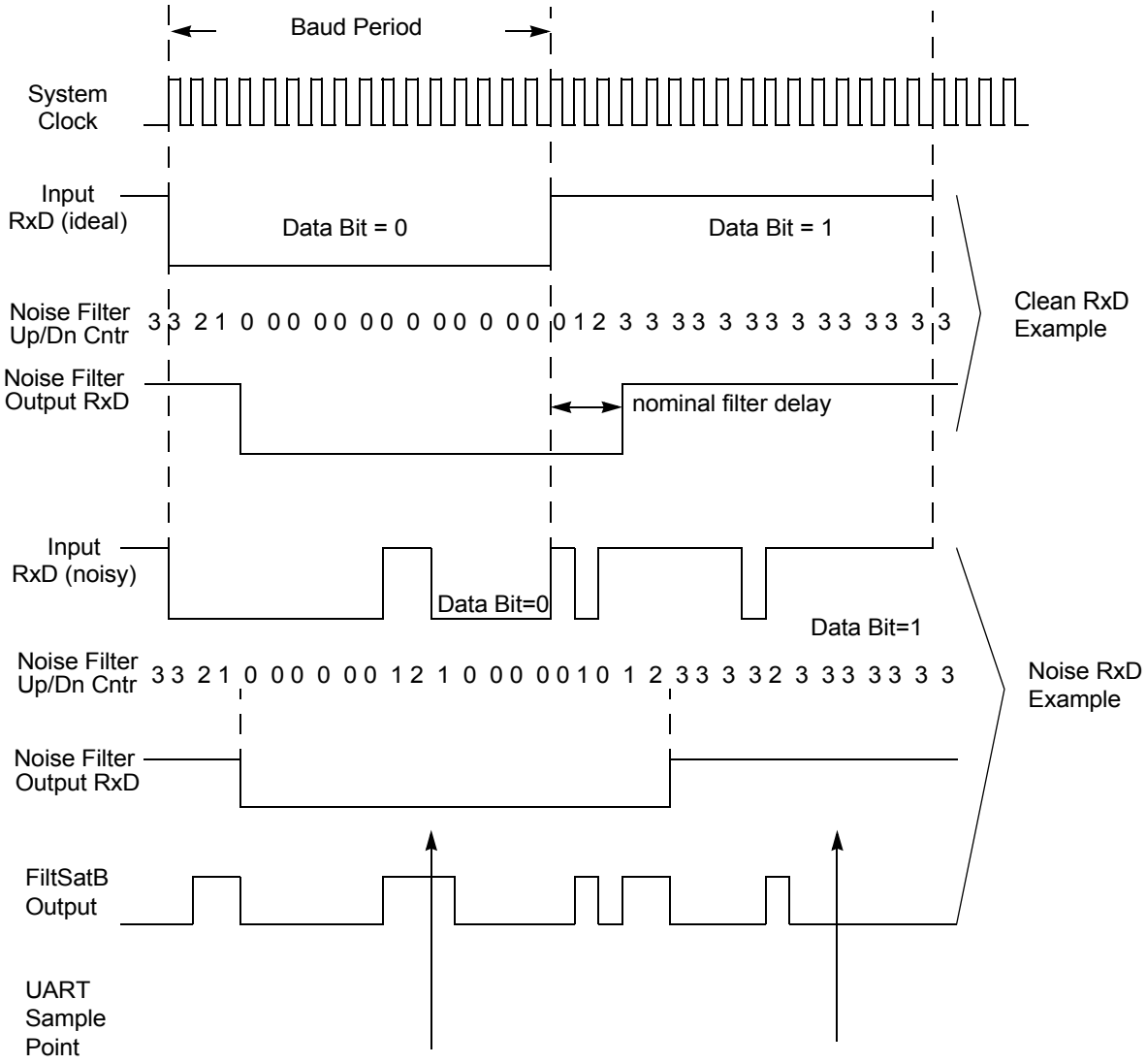


Figure 18. Noise Filter Operation

LIN-UART Control Register Definitions

The LIN-UART control registers support the LIN-UART, the associated Infrared Encoder/Decoder and the noise filter. For more information about the infrared operation, see the Infrared Encoder/Decoder chapter on page 142.

LIN-UART Transmit Data Register

Data bytes written to the LIN-UART Transmit Data Register, shown in Table 66, are shifted out on the TxD pin. The Write-only LIN-UART Transmit Data Register shares a Register File address with the read-only LIN-UART Receive Data Register.

Table 66. LIN-UART Transmit Data Register (U0TXD)

Bit	7	6	5	4	3	2	1	0
Field	TXD							
RESET	X							
R/W	W							
Address	F40H							

Bit	Description
[7:0] TXD	**Transmit Data** LIN-UART transmitter data byte to be shifted out through the TXD pin.

LIN-UART Receive Data Register

Data bytes received through the RxD pin are stored in the LIN-UART Receive Data Register, shown in Table 67. The read-only LIN-UART Receive Data Register shares a Register File address with the Write-only LIN-UART Transmit Data Register.

Table 67. LIN-UART Receive Data Register (U0RXD)

Bit	7	6	5	4	3	2	1	0
Field	RXD							
RESET	X							
R/W	R							
Address	F40H							

Bit	Description
[7:0] RXD	**Receive Data** LIN-UART receiver data byte from the RXD pin.

LIN-UART Status 0 Register

The LIN-UART Status 0 Register identifies the current LIN-UART operating configuration and status. Table 68 describes the Status 0 Register for Standard UART Mode. Table 69 describes the Status 0 Register for LIN Mode.

Table 68. LIN-UART Status 0 Register – Standard UART Mode (U0STAT0)

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							

Bit	Description
[7] RDA	**Receive Data Available** This bit indicates that the LIN-UART Receive Data Register has received data. Reading the LIN-UART Receive Data Register clears this bit.
[6] PE	**Parity Error** This bit indicates that a parity error has occurred. Reading the Receive Data Register clears this bit.
[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 has not been read. Reading the Receive Data Register clears this bit.
[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.**
[3] BRKD	**Break Detect** This bit indicates that a break has occurred. If the data bits, parity/multiprocessor bit and STOP bit(s) are all zeros then this bit is set to 1. Reading the Receive Data Register clears this bit.
[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.
[1] TXE	**Transmitter Empty** This bit indicates that the Transmit Shift Register is empty and character transmission is finished.
[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 loopback mode. CTS only affects transmission if the CTSE bit = 1.

Table 69. LIN-UART Status 0 Register, LIN Mode (U0STAT0)

Bit	7	6	5	4	3	2	1	0
Field	RDA	PLE	ABOE	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							

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.
[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 caused 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.
[5] ABOE	**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, indicating the receive activity was 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.
[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.
[3] BRKD	**Break Detect** This bit is set in LIN Mode if (a) in LinSleep state and a break of at least 4 bit times occurred (Wake-up event) or (b) in Slave Wait Break state and a break of at least 11 bit times occurred (Break event) or (c) 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.
[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.
[1] TXE	**Transmitter Empty** This bit indicates that the Transmit Shift Register is empty and character transmission is finished.
[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.

LIN-UART Mode Select and Status Register

The LIN-UART Mode Select and Status Register, shown in Table 70, contains mode select and status bits. A more detailed discussion of each bit follows the table.

Table 70. LIN-UART Mode Select and Status Register (U0MDSTAT)

Bit	7	6	5	4	3	2	1	0
Field	MSEL			Mode Status				
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R	R	R	R	R
Address	F44H							

Bit	Description
[7:5] MSEL	**Mode Select** This R/W 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 ModeStatus field when reading this register. 000 = Multiprocessor and normal UART control/status. 001 = Noise Filter control/status. 010 = LIN Protocol control/status. 011–110: Reserved. 111 = LIN-UART Hardware Revision (allows hardware revision to be read in the Mode Status field).
[4:0] Mode Status	**Mode Status** This read-only field returns status corresponding to the mode selected by MSEL as follows: 000: Multiprocessor and normal UART mode status = {NE, 0, 0, NEWFRM, MPRX} 001: Noise Filter status = {NE, 0,0,0,0}. 010: LIN Mode status = {NE, RxBreakLength[3:0]}. 011–110: reserved = {0, 0, 0, 0, 0}. 111: LIN-UART hardware revision.

MULTIPROCESSOR Mode Status field (MSEL = 000B)

Noise Event (NE). This bit is asserted if digital noise is detected on the receive data line while 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 may be in extreme cases), just that one or more of the noise filter data samples near the center of the bit time did not match the average data value.

New Frame (NEWFRM). Status bit denoting the start of a new frame. Reading the LIN-UART 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 LIN-UART Receive Data Register resets this bit to 0.

Digital Noise Filter Mode Status Field (MSEL = 001B)

NE—Noise Event. This bit is asserted if digital noise is detected on the receive data line while 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 may be in extreme cases), just that one or more of the noise filter data samples near the center of the bit time did not match the average data value.

LIN Mode Status Field (MSEL = 010B)

Noise Event (NE). This bit is asserted if some noise level is detected on the receive data line while the data is sampled (center of bit time). If this bit is set, it does not indicate that the receive data is corrupt (though it may be in extreme cases), just that one or more of the 16x data samples near the center of the bit time did not match the average data value.

RxBreakLength. LIN Mode received break length. This field may be read following a break (LIN WAKE-UP or BREAK) so software can determine the measured duration of the break. If the break exceeds 15 bit times the value saturates at 1111B.

Hardware Revision Mode Status Field (MSEL = 111B)

This field indicates the hardware revision of the LIN-UART block.
00_xxx = LIN UART hardware revision.
01_xxx = Reserved.
10_xxx = Reserved.
11_xxx = Reserved.

LIN-UART Control 0 Register

The LIN-UART Control 0 Register, shown in Table 71, configures the basic properties of the LIN-UART's transmit and receive operations. A more detailed discussion of each bit follows the table.

Table 71. LIN-UART Control 0 Register (U0CTL0)

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							

Bit	Description
[7] TEN	**Transmit Enable** This bit enables or disables the transmitter. The enable is also controlled by the $\overline{CTS}$ signal and the CTSE bit. If the $\overline{CTS}$ 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	**CTS Enable** 0 = The $\overline{CTS}$ signal has no effect on the transmitter. 1 = The LIN-UART recognizes the $\overline{CTS}$ signal as an enable control for the transmitter.
[4] PEN	**Parity Enable** This bit enables or disables parity. 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 transmitted and expected on all received data. 1 = Odd parity is transmitted and expected on all received data.

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 finished sending data before setting this bit. In standard UART mode, the duration of the break is determined by how long 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. 0 = No break is sent. 1 = The output of the transmitter is 0.
[1] STOP	**Stop Bit Select** 0 = The transmitter sends one stop bit. 1 = The transmitter sends two stop bits.
[0] LBEN	**Loop Back Enable** 0 = Normal operation. 1 = All transmitted data is looped back to the receiver within the IrDA module.

LIN-UART Control 1 Registers

Multiple registers, shown in Tables 72 through 74, are accessible by a single bus address. The register selected is determined by the Mode Select (MSEL) field. These registers provide additional control over LIN-UART operation.

Multiprocessor Control Register

When MSEL = 000b, the Multiprocessor Control Register, shown in Table 72, provides control for UART MULTIPROCESSOR Mode, IrDA Mode, Baud Rate Timer Mode, as well as other features which apply to multiple modes.

Table 72. MultiProcessor Control Register (U0CTL1 with MSEL = 000b)

Bit	7	6	5	4	3	2	1	0
Field	MPMD[1]	MPEN	MPMD[0]	MPBT	DEPOL	BRGCTL	$\overline{\text{RDAIRQ}}$	IREN
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 with MSEL = 000b							

Bit	Description
[7,5] MPMD[1], MPMD[0]	**MULTIPROCESSOR Mode** If MULTIPROCESSOR (9-bit) Mode is enabled, the following IRQ events can occur: 00 = The LIN-UART generates an interrupt request on all received bytes (data and address). 01 = The LIN-UART generates an interrupt request only on received address bytes. 10 = The LIN-UART 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 LIN-UART 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 (9-bit) 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 0 in the multiprocessor bit location of the data stream (9th bit). 1 = Send 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 LIN-UART behavior depending on whether the LIN-UART receiver is enabled (REN = 1 in the LIN-UART Control 0 Register). When the LIN-UART receiver is not enabled, this bit determines whether the Baud Rate Generator issues interrupts. 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 1. Reads from the Baud Rate High and Low Byte registers return the current BRG count value. When the LIN-UART receiver is enabled, this bit allows reads from the Baud Rate registers to return the BRG count value instead of the reload value. 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 Enable** 0 = Received data and receiver errors generates an interrupt request to Interrupt Controller. 1 = Received data does not generate an interrupt request to the Interrupt Controller. Only receiver errors generate an interrupt request.
[0] IREN	**Infrared Encoder/Decoder Enable** 0 = Infrared Encoder/Decoder is disabled. LIN-UART operates normally. 1 = Infrared Encoder/Decoder is enabled. The LIN-UART transmits and receives data through the Infrared Encoder/Decoder.

Noise Filter Control Register

When MSEL = 001b, the Noise Filter Control Register, shown in Table 73, provides control for the digital noise filter.

Table 73. Noise Filter Control Register (U0CTL1 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 with MSEL = 001b							

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 the more delay that is introduced by the filter and the wider the noise event that 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.

LIN Control Register

When MSEL = 010b, the LIN Control Register provides control for the LIN Mode of operation.

Table 74. LIN Control Register (U0CTL1 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 with MSEL = 010b							

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 follows Autobaud, if in LIN SLAVE Mode and ABEN = 1. When the Autobaud character is received, a receive interrupt is generated and the ATB bit is set in the Status 0 Register. There is no receive data associated with this interrupt. The Baud Reload registers will be updated by hardware with the new bit period value.

Bit	Description (Continued)
[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 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 may 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 may be set).
[2:0] TxBreakLength	**Transmit Break Length** 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.

LIN-UART Address Compare Register

The LIN-UART Address Compare Register stores the multinode network address of the LIN-UART. When the MPMD[1] bit of the LIN-UART Control Register 0 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. See Table 75.

Table 75. LIN-UART Address Compare Register (U0ADDR)

Bit	7	6	5	4	3	2	1	0
Field	COMP_ADDR							
RESET	00H							
R/W	R/W							
Address	F45H							

Bit	Description
[7:0] COMP_ADDR	**Compare Address** This 8-bit value is compared to the incoming address bytes.

LIN-UART Baud Rate High and Low Byte Registers

The LIN-UART Baud Rate High and Low Byte registers, Tables 77 and 76, combine to create a 16-bit baud rate divisor value (BRG[15:0]) which sets the data transmission rate (baud rate) of the LIN-UART.

Table 76. LIN-UART Baud Rate Low Byte Register (U0BRL)

Bit	7	6	5	4	3	2	1	0
Field	BRL							
RESET	FFH							
R/W	R/W							
Address	F47H							

Table 77. LIN-UART Baud Rate High Byte Register (U0BRH)

Bit	7	6	5	4	3	2	1	0
Field	BRH							
RESET	FFH							
R/W	R/W							
Address	F46H							

Note: The UART must be disabled when updating the Baud Rate registers because the high and low registers must be written independently.

The LIN-UART data rate is calculated using the following equation for standard UART operation:

$$\text{UART Data Rate (bps)} = \frac{\text{System Clock Frequency (Hz)}}{16 \times \text{UART Baud Rate Divisor Value}}$$

The LIN-UART data rate is calculated using the following equation for LIN Mode UART operation:

$$\text{UART Data Rate (bps)} = \frac{\text{System Clock Frequency (Hz)}}{\text{UART Baud Rate Divisor Value}}$$

For a given LIN-UART data rate, the integer baud rate divisor value is calculated using the following equation for standard UART operation:

$$\text{UART Baud Rate Divisor Value} = \text{Round}\left(\frac{\text{System Clock Frequency (Hz)}}{16 \times \text{UART Data Rate (bits/s)}}\right)$$

For a given LIN-UART data rate, the integer baud rate divisor value is calculated using the following equation for LIN Mode UART operation:

$$\text{UART Baud Rate Divisor Value s} = \text{Round}\left(\frac{\text{System Clock Frequency (Hz)}}{\text{UART Data Rate (bits/s)}}\right)$$

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 LIN-UART baud rate error must never exceed 5 percent. Table 78 through Table 82 provide error data for popular baud rates and commonly-used crystal oscillator frequencies for normal UART modes of operation.

Table 78. LIN-UART 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	2.399	–0.03
115.2	11	113.64	–1.19	1.20	1042	1.199	–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	0.299	–0.01
19.2	65	19.23	0.16				

Table 79. LIN-UART 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	208.33	–16.67	2.40	260	2.40	–0.03
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	0.16				

Table 80. LIN-UART Baud Rates, 5.5296 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	36	9.60	0.00
625.0	N/A	N/A	N/A	4.80	72	4.80	0.00
250.0	1	345.6	38.24	2.40	144	2.40	0.00
115.2	3	115.2	0.00	1.20	288	1.20	0.00
57.6	6	57.6	0.00	0.60	576	0.60	0.00
38.4	9	38.4	0.00	0.30	1152	0.30	0.00
19.2	18	19.2	0.00				

Table 81. LIN-UART Baud Rates, 3.579545 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	23	9.73	1.32
625.0	N/A	N/A	N/A	4.80	47	4.76	–0.83
250.0	1	223.72	–10.51	2.40	93	2.41	0.23
115.2	2	111.9	–2.90	1.20	186	1.20	0.23
57.6	4	55.9	–2.90	0.60	373	0.60	–0.04
38.4	6	37.3	–2.90	0.30	746	0.30	–0.04
19.2	12	18.6	–2.90				

Table 82. LIN-UART Baud Rates, 1.8432 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	12	9.60	0.00
625.0	N/A	N/A	N/A	4.80	24	4.80	0.00
250.0	N/A	N/A	N/A	2.40	48	2.40	0.00
115.2	1	115.2	0.00	1.20	96	1.20	0.00
57.6	2	57.6	0.00	0.60	192	0.60	0.00
38.4	3	38.4	0.00	0.30	384	0.30	0.00
19.2	6	19.2	0.00				

Infrared Encoder/Decoder

The Z8FMC16100 Series MCU contains two fully-functional, high-performance UART to Infrared Encoder/Decoders (endecs). Each infrared endec is integrated with an on-chip UART to allow easy communication between the Z8FMC16100 Series MCU and IrDA Physical Layer Specification, version1.3-compliant infrared transceivers. Infrared communication provides secure, reliable, low-cost, point-to-point communication between PCs, PDAs, cell phones, printers and other infrared enabled devices.

Architecture

Figure 19 displays the architecture of the infrared endec.

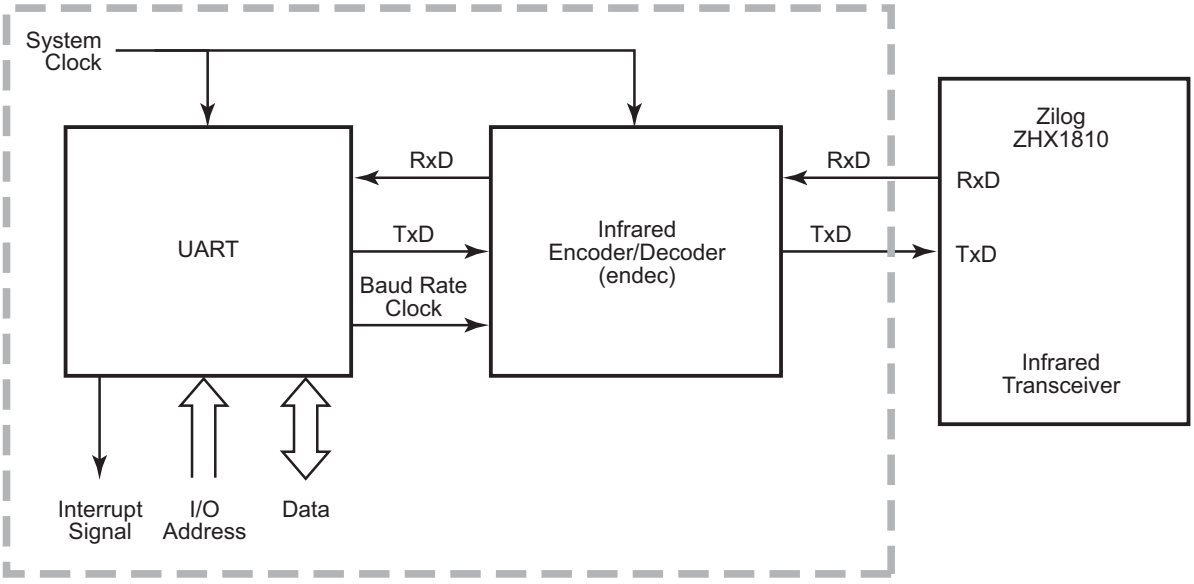


Figure 19. Infrared Data Communication System Block Diagram

Operation

When the infrared endec is enabled, the transmitted data from the associated on-chip UART is encoded as digital signals in accordance with IrDA standard and output to the infrared transceiver using the TXD pin. The data received from the infrared transceiver is passed to the infrared endec using the RXD pin, decoded by the infrared endec and passed

to the UART. Communication is half-duplex, which means simultaneous data transmission and reception is not allowed.

The baud rate is set by the UARTs Baud Rate Generator and supports IrDA standard baud rates from 9600 baud to 115.2KBaud. Higher baud rates are possible, but do not meet IrDA specifications. The UART must be enabled to use the infrared endec. The infrared endec data rate is calculated using the following equation:

$$\text{Infrared Data Rate (bps)} = \frac{\text{System Clock Frequency (Hz)}}{\text{16 x UART Baud Rate Divisor Value}}$$

Transmitting IrDA Data

The data to be transmitted using the infrared transceiver is first sent to the UART. The UART's transmit signal (TXD) and baud rate clock are used by the IrDA to generate the modulation signal (IR_TXD) which drives the infrared transceiver. Each UART/Infrared data bit is 16-clocks wide. If the data to be transmitted is 1, the IR_TXD signal remains Low for the full 16-clock period. If the data to be transmitted is 0, a 3-clock high pulse is output following a 7-clock low period. After the 3-clock high pulse, a 6-clock low pulse is output to complete the full 16-clock data period. Figure 20 displays the IrDA data transmission. When the infrared endec is enabled, the UART's TXD signal is internal to the Z8FMC16100 Series MCU while the IR_TXD signal is output through the TXD pin.

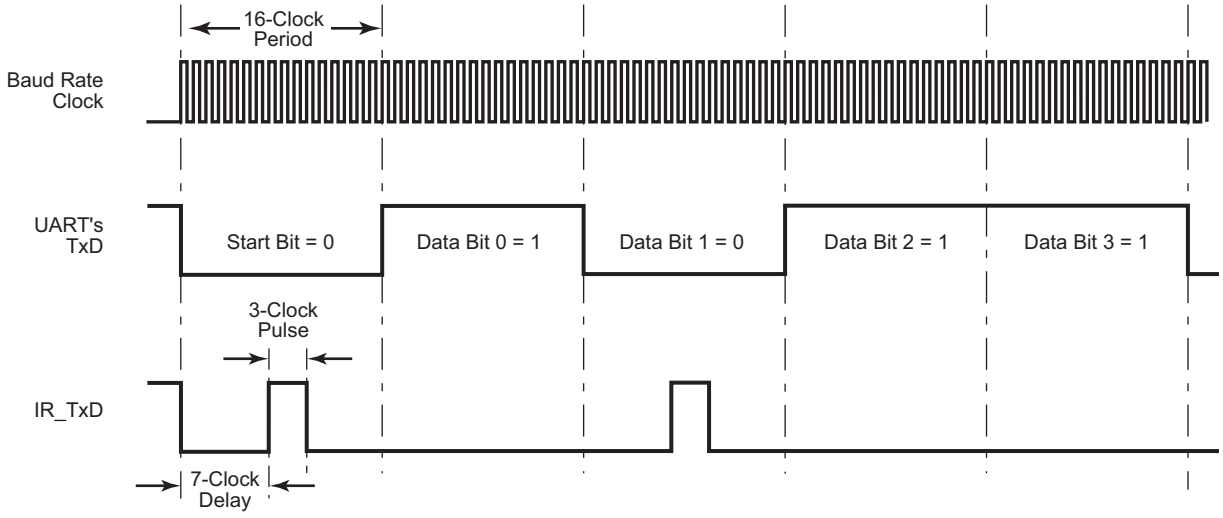


Figure 20. Infrared Data Transmission

Receiving IrDA Data

Data received from the infrared transceiver via the IR_RXD signal through the RXD pin is decoded by the infrared endec and passed to the UART. The UART's baud rate clock is used by the infrared endec to generate the demodulated signal (RXD) which drives the UART. Each UART/Infrared data bit is 16-clocks wide. Figure 21 displays the data reception. When the infrared endec is enabled, the UART's RXD signal is internal to the Z8FMC16100 Series MCU while the IR_RXD signal is received through the RXD pin.

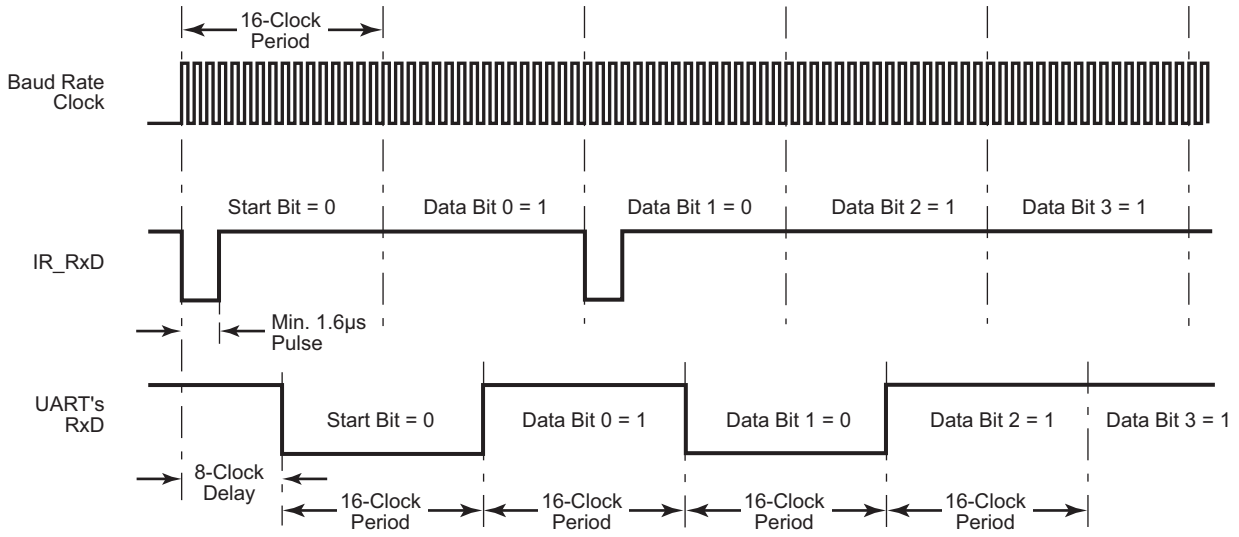


Figure 21. Infrared Data Reception

Caution: The system clock frequency must be at least 1.0MHz to ensure proper reception of the 1.6µs minimum-width pulses allowed by the IrDA standard.

Endec Receiver Synchronization

The IrDA receiver uses a local baud rate clock counter (0 to 15 clock periods) to generate an input stream for the UART and to create a sampling window for detection of incoming pulses. The generated UART input (UART RXD) is delayed by 8 baud rate clock periods with respect to the incoming IrDA data stream. When a falling edge in the input data stream is detected, the endec counter is reset. When the count reaches a value of 8, the UART RXD value is updated to reflect the value of the decoded data. When the count reaches 12 baud clock periods, the sampling window for the next incoming pulse opens. The window remains open until the count again reaches 8 (or in other words 24 baud clock

periods since the previous pulse was detected) giving the endec a sampling window of minus four baud rate clocks to plus eight baud rate clocks around the expected time of an incoming pulse. If an incoming pulse is detected inside this window this process is repeated. If the incoming data is a logical 1 (no pulse), the endec returns to the initial state and waits for next falling edge. As each falling edge is detected, the endec clock counter is reset, resynchronizing the endec to the incoming signal. This allows the endec to tolerate jitter and baud rate errors in the incoming data stream. Resynchronizing the endec does not alter the operation of the UART, that ultimately receives the data. The UART is only synchronized to the incoming data stream when a START bit is received.

Infrared Encoder/Decoder Control Register Definitions

All infrared endec configuration and status information is set by the UART control registers as defined in the LIN-UART Control Register Definitions section on page 124.

Caution: To prevent spurious signals during IrDA data transmission, set the IREN bit in the UARTx Control 1 Register to 1 to enable the Infrared Encoder/Decoder before enabling the GPIO Port alternate function for the corresponding pin.

Serial Peripheral Interface

The Serial Peripheral Interface (SPI) is a synchronous interface allowing several SPI-type devices, such as EEPROMs, to be interconnected. The SPI features include:

- Full-duplex, synchronous, character-oriented communication
- Four-wire interface
- Data transfer rates up to a maximum of one-half the system clock frequency
- Error detection
- Dedicated Baud Rate Generator

Architecture

The SPI can be configured as either a master (in single or multimaster systems) or a slave as displayed in Figures 22 through 24.

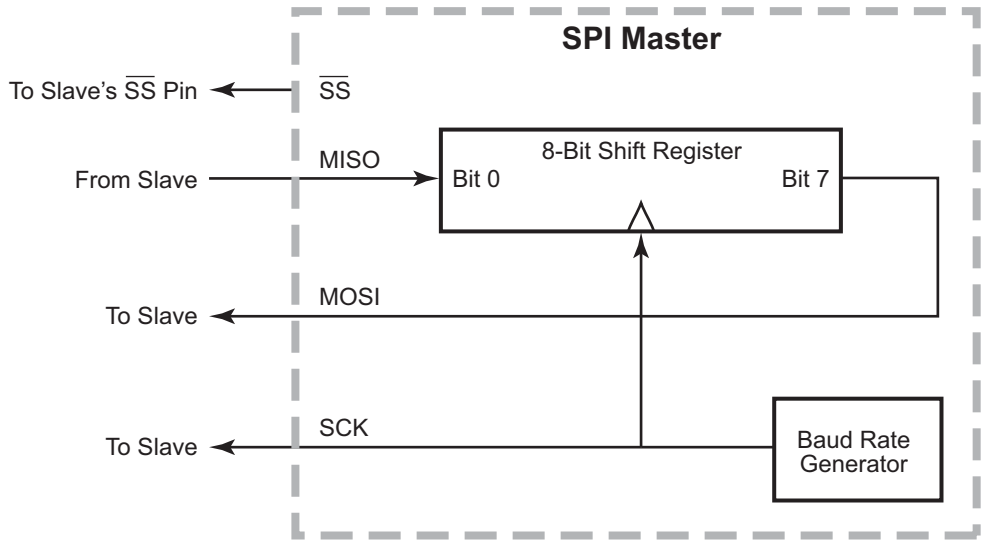


Figure 22. SPI Configured as a Master in a Single Master, Single Slave System

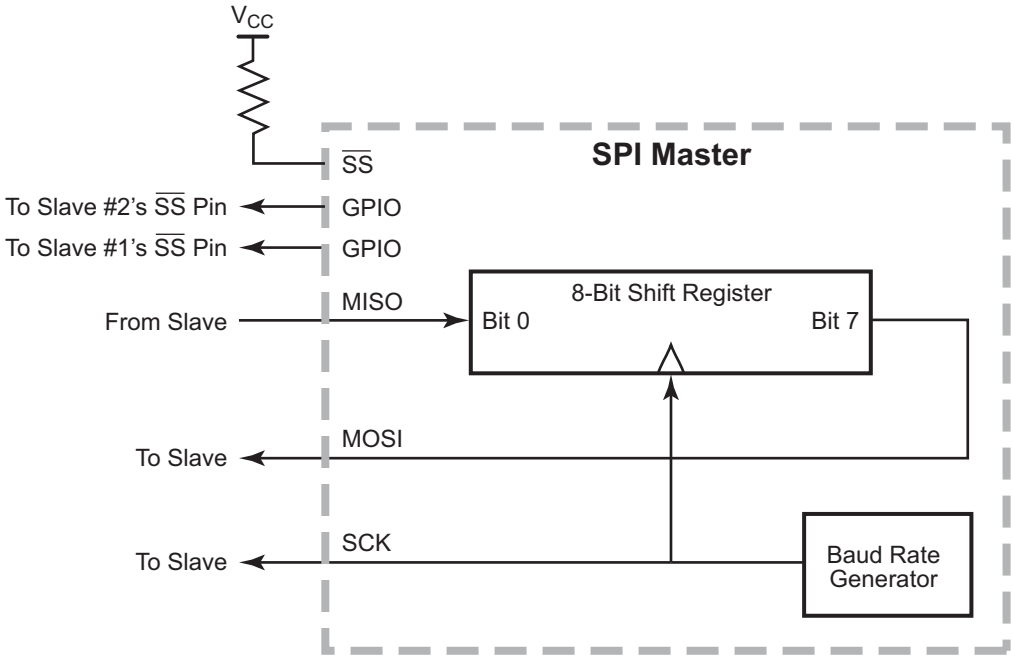


Figure 23. SPI Configured as a Master in a Single Master, Multiple Slave System

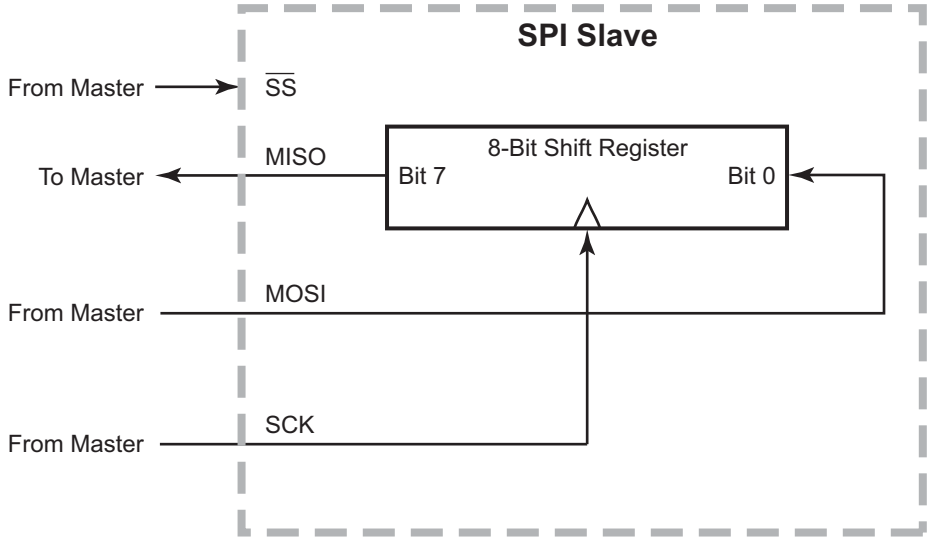


Figure 24. SPI Configured as a Slave

Operation

The SPI is a full-duplex, synchronous, character-oriented channel that supports a four-wire interface (serial clock, transmit, receive and slave select). The SPI block consists of a transmit/receive shift register, a Baud Rate (clock) Generator and a control unit.

During an SPI transfer, data is sent and received simultaneously by both the master and the slave SPI devices. Separate signals are required for data and the serial clock. When an SPI transfer occurs, a multibit (typically 8-bit) character is shifted out one data pin and an multibit character is simultaneously shifted in on a second data pin. An 8-bit shift register in the master and another 8-bit shift register in the slave are connected as a circular buffer. The SPI shift register is single-buffered in the transmit and receive directions. New data to be transmitted cannot be written into the shift register until the previous transmission is complete and receive data (if valid) has been read.

SPI Signals

The four basic SPI signals are:

- Master-In/Slave-Out (MISO)
- Master-Out/Slave-In (MOSI)
- Serial Clock (SCK)
- Slave Select ($\overline{SS}$)

Each signal is described in both MASTER and SLAVE modes.

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. It is one of the two lines that transfer serial data, with the most significant bit sent first. The MISO pin of a slave device is placed in a high-impedance state if the slave is not selected. When the SPI is not enabled, this signal is in a high-impedance state.

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. It is one of the two lines that transfer serial data, with the most significant bit sent first. When the SPI is not enabled, this signal is in a high-impedance state.

Serial Clock

The Serial Clock (SCK) synchronizes data movement both in and out of the device through its MOSI and MISO pins. In MASTER Mode, the SPI's Baud Rate Generator creates the serial clock. The master drives the serial clock through its own serial clock (SCK) pin to the slave's SCK pin. When the SPI is configured as a slave, the SCK pin is an input and the clock signal from the master synchronizes the data transfer between the master and slave devices. These slave devices ignore the SCK signal unless the $\overline{SS}$ pin is asserted. When configured as a slave, the SPI block requires a minimum SCK period of greater than or equal to 8 times the system (X_{IN}) clock period.

The master and slave are each capable of exchanging a character of data during a sequence of NUMBITS clock cycles (refer to the NUMBITS field in the SPIMODE Register). In both master and slave SPI devices, data is shifted on one edge of the SCK and is sampled on the opposite edge, where data is stable. Edge polarity is determined by the SPI phase and polarity control.

Slave Select

The active Low Slave Select ($\overline{SS}$) input signal selects a slave SPI device. $\overline{SS}$ must be Low prior to all data communication to and from the slave device. $\overline{SS}$ must remain Low for the full duration of each character transferred. The $\overline{SS}$ signal may stay Low during the transfer of multiple characters, or may deassert between each character.

When the SPI is configured as the only master in an SPI system, the $\overline{SS}$ pin can be set as either an input or an output. For communication between the Z8FMC16100 Series MCU's SPI master and external slave devices, the $\overline{SS}$ signal, as an output, can assert the $\overline{SS}$ input pin on one of the slave devices. Other GPIO output pins can also be employed to select external SPI slave devices.

When the SPI is configured as one master in a multimaster SPI system, the $\overline{SS}$ pin should be set as an input. The $\overline{SS}$ input signal on the master must be High. If the $\overline{SS}$ signal goes Low (indicating that another master is driving the SPI bus), a collision error flag is set in the SPI Status Register.

SPI Clock Phase and Polarity Control

The SPI supports four combinations of serial clock phase and polarity using two bits in the SPI Control Register. The clock polarity bit, CLKPOL, selects an active High or active Low clock and has no effect on the transfer format. Table 83 lists the SPI Clock Phase and Polarity Operation parameters. The clock phase bit, PHASE, selects one of two fundamentally different transfer formats. For proper data transmission, clock phase and polarity must be identical for the SPI master and the SPI slave. The master always places data on the MOSI line a half-cycle before the receive clock edge (SCK signal) for the slave to latch the data.

Table 83. SPI Clock Phase and Clock Polarity 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

Transfer Format Phase Equals Zero

Figure 25 displays the timing diagram for an SPI transfer in which PHASE is cleared to 0. The two SCK waveforms show polarity with CLKPOL reset to 0 and with CLKPOL set to 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.

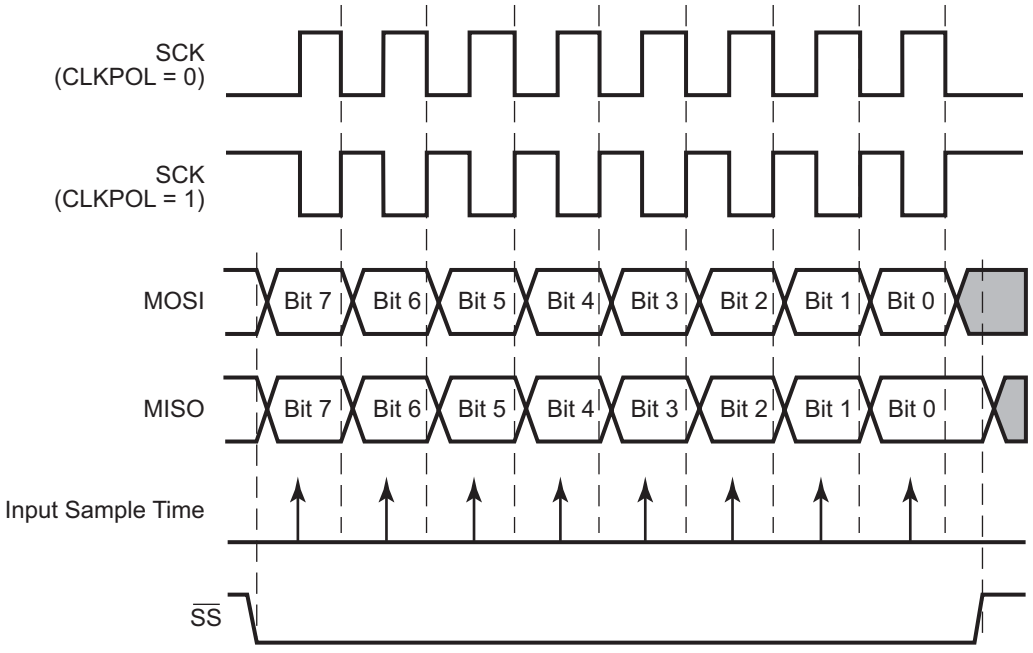


Figure 25. SPI Timing When Phase is 0

Transfer Format Phase Equals One

Figure 26 displays the timing diagram for an SPI transfer in which PHASE is 1. Two waveforms are depicted for SCK, one for CLKPOL reset to 0 and another for CLKPOL set to 1.

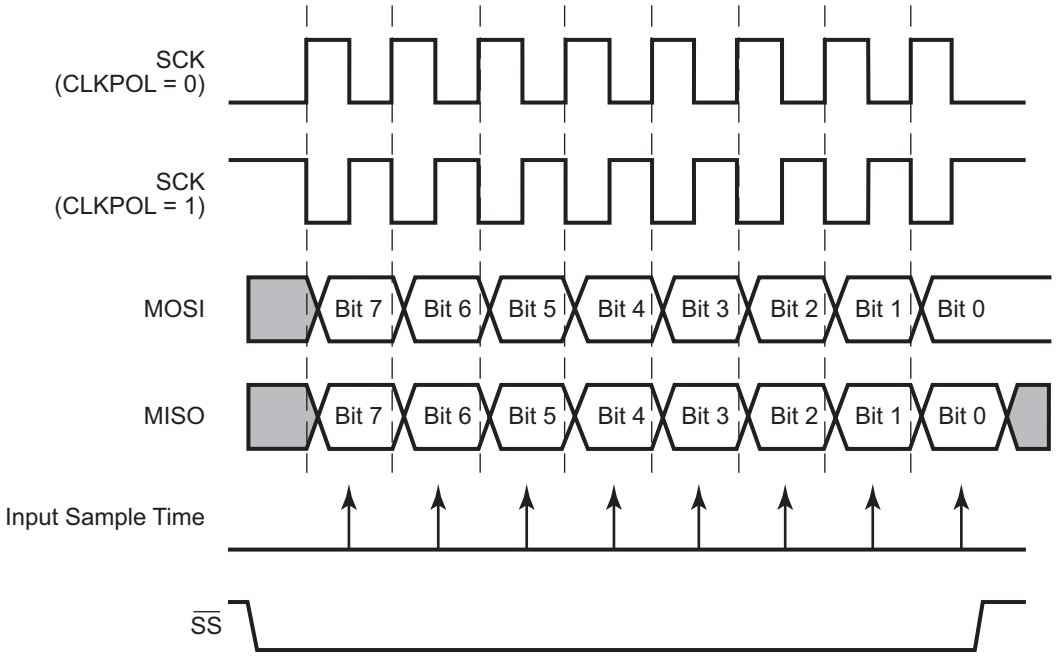


Figure 26. SPI Timing When Phase is 1

Multimaster Operation

In a multimaster 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 then be configured in OPEN-DRAIN mode to prevent bus contention. At any time, only one SPI device is configured as the master and all other SPI devices on the bus are configured as slaves. The master enables a single slave by asserting the $\overline{SS}$ pin on that slave only. Then, the single master drives data out its SCK and MOSI pins to the SCK and MOSI pins on the slaves (including those which are not enabled). The enabled slave drives data out its MISO pin to the MISO master pin.

For a master device operating in a multimaster system, if the $\overline{SS}$ pin is configured as an input and is driven Low by another master, the COL bit is set to 1 in the SPI Status Register. The COL bit indicates the occurrence of a multimaster collision (mode fault error condition).

Slave Operation

The SPI block is configured for SLAVE Mode operation by setting the SPIEN bit to 1 and the MMEN bit to 0 in the SPICTL Register and setting the SSIO bit to 0 in the SPIMODE Register. The IRQE, PHASE, CLKPOL and WOR bits in the SPICTL Register and the NUMBITS field in the SPIMODE Register must be set to be consistent with the other SPI devices. The STR bit in the SPICTL Register can be used, if appropriate, to force a start-up interrupt. The BIRQ bit in the SPICTL Register and the SSV bit in the SPIMODE Register are not used in SLAVE Mode. The SPI baud rate generator is not used in SLAVE Mode; therefore, the SPIBRH and SPIBRL registers do not require initialization.

If the slave contains data to send to the master, the data should be written to the SPIDAT Register before the transaction starts (first edge of SCK when $\overline{SS}$ is asserted). If the SPIDAT Register is not written prior to the slave transaction, the MISO pin outputs the value that is currently in the SPIDAT Register.

Due to the delay resulting from synchronization of the SPI input signals to the internal system clock, the maximum SPICLK baud rate that can be supported in SLAVE Mode is the system clock frequency (X_{IN}) divided by 8. This rate is controlled by the SPI master.

Error Detection

The SPI contains error detection logic that supports SPI communication protocols and recognizes when communication errors have occurred. The SPI Status Register indicates when a data transmission error has been detected.

Overrun

An overrun error (write collision) indicates that a Write to the SPI Data Register was attempted while a data transfer is in progress (in either MASTER or SLAVE modes). An overrun sets the OVR bit in the SPI Status Register to 1. Writing a 1 to OVR clears this error flag. The SPI Data Register is not altered when a Write occurs while a data transfer is in progress.

Mode Fault

A mode fault indicates when more than one master is trying to communicate at the same time (a multimaster collision). The mode fault is detected when the enabled master's $\overline{SS}$ pin is asserted. A mode fault sets the COL bit in the SPI Status Register to 1. Writing a 1 to COL clears this error flag.

SLAVE Mode Abort

In the SLAVE Mode, if the $\overline{SS}$ pin deasserts before all bits in a character are transferred, the transaction aborts. When this condition occurs, the ABT bit is set in the SPISTAT Register as well as the IRQ bit (which indicates that the transaction is complete). The next

time $\overline{SS}$ asserts, the MISO pin outputs SPIDAT[7], regardless of where the previous transaction suspended. Writing a 1 to ABT clears this error flag.

SPI Interrupts

When SPI interrupts are enabled, the SPI generates an interrupt after character transmission/reception is completed in both MASTER and SLAVE modes. A character can be defined to be 1–8 bits by the NUMBITS field in the SPI Mode Register. In SLAVE Mode, it is not necessary for $\overline{SS}$ to deassert between characters to generate an interrupt. The SPI in SLAVE Mode can also generate an interrupt if the $\overline{SS}$ signal deasserts prior to transfer of all the bits in a character (see description of slave abort error above). Writing a 1 to the IRQ bit in the SPI Status Register clears the pending SPI interrupt request. The IRQ bit must be cleared to 0 by the interrupt service routine to generate future interrupts. To start the transfer process, an SPI interrupt can be forced by software to write a 1 to the STR bit in the SPICTL Register.

If the SPI is disabled, an SPI interrupt can be generated by a Baud Rate Generator time-out. This timer function must be enabled by setting the BIRQ bit in the SPICTL Register. This Baud Rate Generator time-out does not set the IRQ bit in the SPISTAT Register, just the SPI interrupt bit in the interrupt controller.

SPI Baud Rate Generator

In SPI 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 from the system clock. The SPI Baud Rate High and Low Byte registers combine to form a 16-bit reload value, BRG[15:0], for the SPI Baud Rate Generator. The SPI baud rate is calculated using the following equation:

$$\text{SPI Baud Rate (bps)} = \frac{\text{System Clock Frequency (Hz)}}{2 \times \text{BRG[15:0]}}$$

Minimum baud rate is obtained by setting BRG[15:0] to 0000h for a clock divisor value of (2 x 65536 = 131072).

When the SPI is disabled, the Baud Rate Generator can function as a basic 16-bit timer with an interrupt upon time-out. To configure the Baud Rate Generator as a timer with an interrupt upon time-out, complete the following procedure:

1. Disable the SPI by clearing the SPIEN bit in the SPI Control Register to 0.
2. Load the appropriate 16-bit count value into the SPI Baud Rate High and Low Byte registers.

3. Enable the Baud Rate Generator timer function and the associated interrupt by setting the BIRQ bit in the SPI Control Register to 1.

SPI Data Register

The SPI Data Register, shown in Table 84, stores both the outgoing (transmit) data and the incoming (receive) data. Reads from the SPI Data Register always return the current contents of the 8-bit shift register. Data is shifted out starting with bit 7. The last bit received resides in bit position 0.

With the SPI configured as a master, writing a data byte to this register initiates data transmission. With the SPI configured as a slave, writing a data byte to this register loads the shift register in preparation for the next data transfer with the external master. In either MASTER or SLAVE Mode, if a transmission is already in progress, Writes to this register are ignored and the overrun error flag, OVR, is set in the SPI Status Register.

When character length is less than 8 bits (as set by the NUMBITS field in the SPI Mode Register), the transmit character must be left-justified in the SPI Data Register. A received character of less than 8 bits is right-justified (the final bit received is in bit position 0). For example, if the SPI is configured for 4-bit characters, the transmit characters must be written to SPIDATA[7:4] and the received characters are read from SPIDATA[3:0].

Table 84. SPI Data Register (SPIDATA)

Bit	7	6	5	4	3	2	1	0
Field	DATA							
RESET	X							
R/W	R/W							
Address	F60H							

Bit	Description
[7:0] DATA	**Data** Transmit and/or receive data.

SPI Control Register

The SPI Control Register, shown in Table 85, configures the SPI for transmit and receive operations.

Table 85. SPI Control Register (SPICTL)

Bit	7	6	5	4	3	2	1	0
Field	IRQE	STR	BIRQ	PHASE	CLKPOL	WOR	MMEN	SPIEN
RESET	00H							
R/W	R/W							
Address	F61H							

Bit	Description
[7] IRQE	**Interrupt Request Enable** 0 = SPI interrupts are disabled. No interrupt requests are sent to the Interrupt Controller. 1 = SPI interrupts are enabled. Interrupt requests are sent to the Interrupt Controller.
[6] STR	**Start an SPI Interrupt Request** 0 = No effect. 1 = Setting this bit to 1 also sets the IRQ bit in the SPI Status Register to 1. Setting this bit forces the SPI to send an interrupt request to the Interrupt Control. This bit can be used by software for a function similar to transmit buffer empty in a UART. Writing a 1 to the IRQ bit in the SPI Status Register clears this bit to 0.
[5] BIRQ	**BRG Timer Interrupt Request** If the SPI is enabled, this bit has no effect. If the SPI is disabled: 0 = The Baud Rate Generator timer function is disabled. 1 = The Baud Rate Generator timer function and time-out interrupt are enabled.
[4] PHASE	**Phase Select** Sets the phase relationship of the data to the clock. For more details about operation of the PHASE bit, see SPI Clock Phase and Polarity Control section.
[3] CLKPOL	**Clock Polarity** 0 = SCK idles Low (0). 1 = SCK idle High (1).
[2] WOR	**Wire-OR (Open-Drain) Mode Enabled** 0 = SPI signal pins not configured for open-drain. 1 = All four SPI signal pins (SCK, $\overline{SS}$, MISO, MOSI) configured for open-drain function. This setting is typically used for multi-master and/or multi-slave configurations.
[1] MMEN	**SPI MASTER Mode Enable** 0 = SPI configured in SLAVE Mode. 1 = SPI configured in MASTER Mode.
[0] SPIEN	**SPI Enable** 0 = SPI disabled. 1 = SPI enabled.

SPI Status Register

The SPI Status Register, shown in Table 86, indicates the current state of the SPI. All bits revert to their reset state if the SPIEN bit in the SPICTL Register = 0.

Table 86. SPI Status Register (SPISTAT)

Bit	7	6	5	4	3	2	1	0
Field	IRQ	OVR	COL	ABT	Reserved		TXST	SLAS
RESET	0	0	0	0	0	0	0	1
R/W	R/W*				R			
Address	F62H							
R/W* = Read access. Write a 1 to clear the bit to 0.								

Bit	Description
[7] IRQ	**Interrupt Request** If SPIEN = 1, this bit is set if the STR bit in the SPICTL Register is set, or upon completion of an SPI master or slave transaction. This bit does not set if SPIEN = 0 and the SPI Baud Rate Generator is used as a timer to generate the SPI interrupt. 0 = No SPI interrupt request pending. 1 = SPI interrupt request is pending.
[6] OVR	**Overrun** 0 = An overrun error has not occurred. 1 = An overrun error has been detected.
[5] COL	**Collision** 0 = A multi-master collision (mode fault) has not occurred. 1 = A multi-master collision (mode fault) has been detected.
[4] ABT	**SLAVE Mode Transaction Abort** This bit is set if the SPI is configured in SLAVE Mode, a transaction is occurring and $\overline{SS}$ deasserts before all bits of a character have been transferred as defined by the NUMBITS field of the SPIMODE Register. The IRQ bit also sets, indicating the transaction has completed. 0 = A SLAVE Mode transaction abort has not occurred. 1 = A SLAVE Mode transaction abort has been detected.
[3:2]	**Reserved** These bits are reserved and must be programmed to 00.
[1] TXST	**Transmit Status** 0 = No data transmission currently in progress. 1 = Data transmission currently in progress.
[0] SLAS	**Slave Select** If SPI enabled as a Slave, 0 = $\overline{SS}$ input pin is asserted (Low) 1 = $\overline{SS}$ input is not asserted (High). If SPI enabled as a Master, this bit is not applicable.

SPI Mode Register

The SPI Mode Register, shown in Table 87, configures the character bit width and the direction and value of the $\overline{SS}$ pin.

Table 87. SPI Mode Register (SPIMODE)

Bit	7	6	5	4	3	2	1	0
Field	Reserved		DIAG	NUMBITS[2:0]			SSIO	SSV
RESET	00H							
R/W	R		R/W					
Address	F63H							

Bit	Description
[7:6]	**Reserved** These bits are reserved and must be programmed to 00.
[5] DIAG	**Diagnostic Mode Control bit** This bit is for SPI diagnostics. Setting this bit allows the Baud Rate Generator value to be read using the SPIBRH and SPIBRL register locations. 0 = Reading SPIBRH, SPIBRL returns the value in the SPIBRH and SPIBRL registers 1 = Reading SPIBRH returns bits [15:8] of the SPI Baud Rate Generator; and reading SPIBRL returns bits [7:0] of the SPI Baud Rate Counter. The Baud Rate Counter High and Low byte values are not buffered. **Caution:** Exercise caution if reading the values while the BRG is counting.
[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. Refer to the SPI Data Register description for information about valid bit positions when the character length is less than 8-bits. 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** 0 = $\overline{SS}$ pin configured as an input. 1 = $\overline{SS}$ pin configured as an output (MASTER Mode only).
[0] SSV	**Slave Select Value** If $\overline{SS}$IO = 1 and SPI configured as a Master: 0 = $\overline{SS}$ pin driven Low (0). 1 = $\overline{SS}$ pin driven High (1). This bit has no effect if SSIO = 0 or if SPI is configured as a Slave.

SPI Diagnostic State Register

The SPI Diagnostic State Register, shown in Table 88, provides observability of internal state. It is a read-only register used for SPI diagnostics. More detail about each bit follows the table.

Table 88. SPI Diagnostic State Register (SPIDST)

Bit	7	6	5	4	3	2	1	0
Field	SCKEN	TCKEN	SPISTATE					
RESET	00H							
R/W	R							
Address	F64H							

Bit	Description
[7] SCKEN	**Shift Clock Enable** 0 = The internal Shift Clock Enable signal is deasserted. 1 = The internal Shift Clock Enable signal is asserted (shift register is updates on next system clock).
[6] TCKEN	**Transmit Clock Enable** 0 = The internal Transmit Clock Enable signal is deasserted. 1 = The internal Transmit Clock Enable signal is asserted. When this is asserted the serial data out is updated on the next system clock (MOSI or MISO).
[5:0] SPISTATE	**SPI State Machine** Defines the current state of the internal SPI State Machine.

SPI Baud Rate High and Low Byte Registers

The SPI Baud Rate High and Low Byte registers, shown in Tables 89 and 90, combine to form a 16-bit reload value (BRG[15:0]) for the SPI Baud Rate Generator. The SPI baud rate is calculated using the following equation:

$$\text{SPI Baud Rate (bps)} = \frac{\text{System Clock Frequency (Hz)}}{2 \times \text{BRG[15:0]}}$$

Minimum baud rate is obtained by setting BRG[15:0] to 0000h for a clock divisor value of (2 x 65536 = 131072).

Table 89. SPI Baud Rate High Byte Register (SPIBRH)

Bit	7	6	5	4	3	2	1	0
Field	BRH							
RESET	FFH							
R/W	R/W							
Address	F66H							

Bit	Description
[7:0] BRH	**SPI Baud Rate High Byte** Most significant byte (BRG[15:8]) of the SPI Baud Rate Generator's reload value.

Table 90. SPI Baud Rate Low Byte Register (SPIBRL)

Bit	7	6	5	4	3	2	1	0
Field	BRL							
RESET	FFH							
R/W	R/W							
Address	F67H							

Bit	Description
[7:0] BRL	**SPI Baud Rate Low Byte** Least significant byte (BRG[7:0]) of the SPI Baud Rate Generator's reload value.

I²C Master/Slave Controller

The I²C Master/Slave Controller ensures that the Z8FMC16100 Series MCU devices are bus-compatible with the I²C protocol. The I²C bus consists of the serial data signal (SDA) and a serial clock signal (SCL) bidirectional lines. The features of I²C controller include:

- Operates in MASTER/SLAVE or SLAVE ONLY modes
- Supports arbitration in a multimaster environment (MASTER/SLAVE modes)
- 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 I²C controller is disabled

Architecture

Figure 27 displays the architecture of the I²C controller.

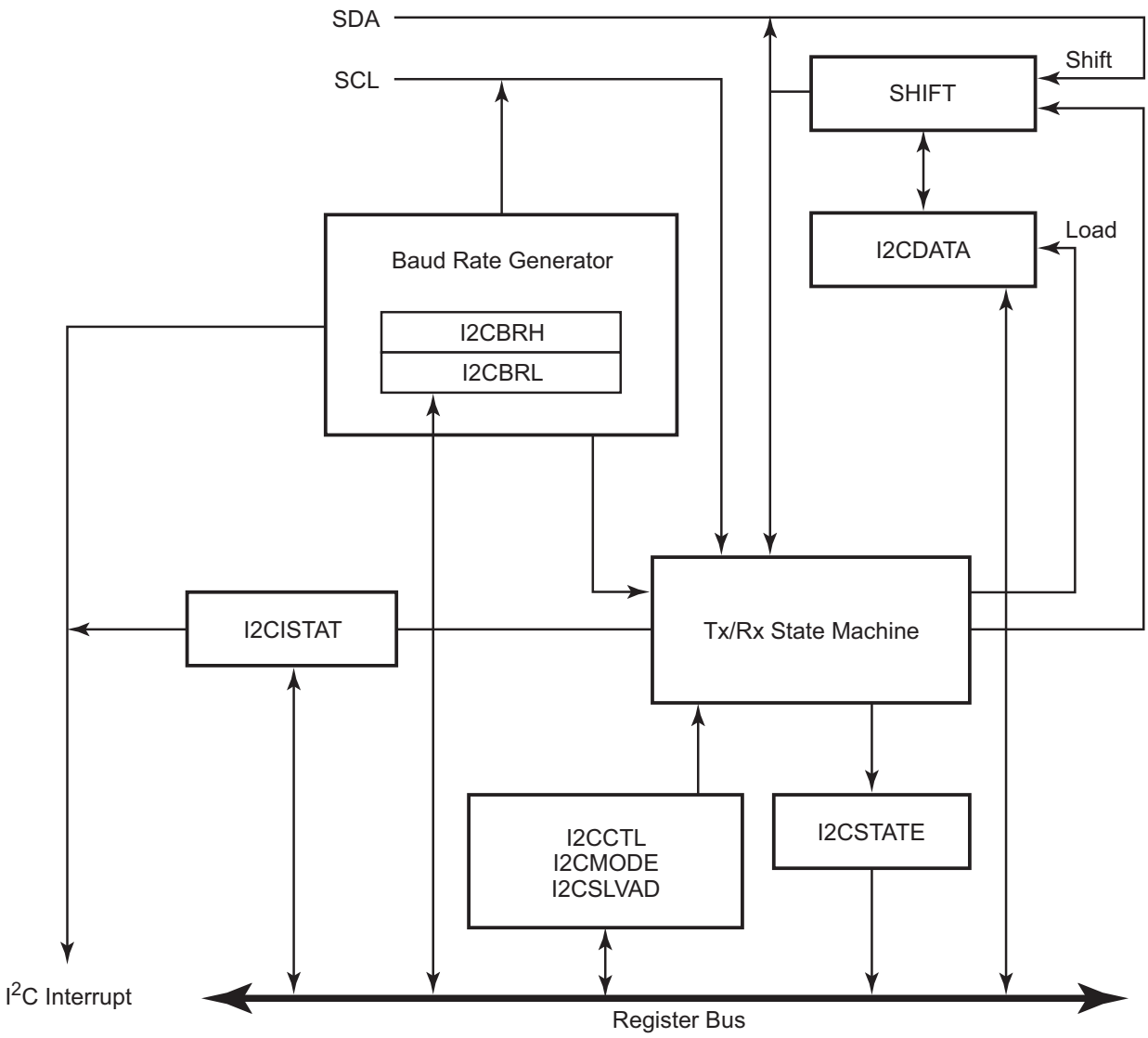


Figure 27. I²C Controller Block Diagram

I²C Master/Slave Controller Registers

Table 91 summarizes the I²C master/slave controller's software-accessible registers.

Table 91. I²C Master/Slave Controller Registers

Name	Abbreviation	Description
I²C Data	I2CDATA	Transmit/Receive Data Register.
I²C Interrupt Status	I2CISTAT	Interrupt Status Register.
I²C Control	I2CCTL	Control Register—basic control functions.
I²C Baud Rate High	I2CBRH	High byte of baud rate generator initialization value.
I²C Baud Rate Low	I2CBRL	Low byte of baud rate generator initialization value.
I²C State	I2CSTATE	State Register.
I²C Mode	I2CMODE	Selects MASTER or SLAVE modes, 7-bit or 10-bit addressing; configure address recognition, defines slave address bits [9:8].
I²C Slave Address	I2CSLVAD	Defines slave address bits [7:0].

Comparison with the MASTER Mode Only I²C Controller

Porting code written for the MASTER ONLY I²C controller found on other Z8FMC16100 Series MCUs to the I²C Master/Slave Controller is straightforward. The I2CDATA, I2CCTL, I2CBRH and I2CBRL Register definitions have not changed. The following bullets highlight the differences between these two designs:

- The Status (I2CSTATE) Register from the MASTER ONLY I²C controller is split into the Interrupt Status (I2CISTAT) Register and the State (I2CSTATE) Register because more interrupt sources are available. The ACK, 10B, TAS (now called AS) and DSS (now called DS) bits, formerly part of the Status Register, are now part of the State Register.
- The I2CSTATE Register was called the Diagnostic State (I2CDST) Register in the MASTER-Mode-only version. The I2CDST Register provided diagnostic information. The I2CSTATE Register contains status and state information that may be useful to software in an operational mode.
- The I2CMODE Register was called the Diagnostic Control (I2CDIAG) Register in the MASTER-Mode-only version. The I2CMODE Register provides control for the SLAVE modes of operation, as well as the most significant two bits of the 10-bit slave address.
- The I2CSLVAD Register is added to provide programming capabilities for the slave address.
- The ACKV bit in the I2CSTATE Register enables the master to check the Acknowledge from the slave before sending the next byte.

- Support for multimaster environments—if arbitration is lost when operating as a master, the ARBLST bit in the I2CISTAT Register is set and the mode automatically switches to SLAVE Mode.

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 several masters on the bus, with arbitration. In a multimaster environment, the controller switches from MASTER to SLAVE MODE upon 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.

SDA and SCL Signals

The I^2C circuit sends all addresses, data and acknowledges signals over the SDA line, 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 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 transaction. All data is transferred in bytes; there is no limit to the amount of data transferred in one operation. When transmitting address, data, or an Acknowledge, the SDA signal changes in the middle of the Low period of SCL**.** When receiving 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 less than a system clock period in width will be rejected. This filter should be enabled when running in I^2C FAST Mode (400 Kbps) and can also be used at lower data rates.

I^2C Interrupts

The I^2C controller contains multiple interrupt sources that are combined into one interrupt request signal to the interrupt controller. If the I^2C controller is enabled, the source of the

interrupt is determined by which bits are set in the I2CISTAT Register. If the I^2C Controller is disabled, the BRG controller can be 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 clears automatically when software reads the register or performs another task, such as reading/writing the data register.

Transmit Interrupts

Transmit interrupts (TDRE bit = 1 in I2CISTAT) 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^2C Control Register
- The I^2C 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^2C Data Register always clears the TRDE bit to 0.

Receive Interrupts

Receive interrupts (RDRF bit = 1 in I2CISTAT) occur when a byte of data has been received by the I^2C controller. The RDRF bit is cleared by reading from the I^2C Data Register. If the RDRF interrupt is not serviced prior to the completion of the next Receive byte, the I^2C 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 for data bytes following a slave address that did 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.

Slave Address Match Interrupts

Slave address match interrupts (SAM bit = 1 in I2CISTAT) occur when the I^2C controller is in SLAVE MODE and an address is received that 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 through the MODE[1:0] field of the I^{2}C Mode Register for 7-bit slave addressing, the most significant 7 bits of the first byte of the 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/W} and the second byte is compared against SLA[7:0].

Arbitration Lost Interrupts

Arbitration Lost interrupts (ARBLST bit = 1 in I2CISTAT) occur when the I^{2}C controller is in MASTER Mode and loses arbitration (outputs a 1 on SDA and receives a 0 on SDA). The I^{2}C controller switches to SLAVE MODE when this instance occurs. This bit clears automatically when the I2CISTAT Register is read.

Stop/Restart Interrupts

A Stop/Restart event interrupt (SPRS bit = 1 in I2CISTAT) occurs when the I^{2}C 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^{2}C State Register indicates whether the bit was 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 cleared automatically when the I2CISTAT Register is read. The Stop/Restart interrupt only occurs on a selected (address match) slave.

Not Acknowledge Interrupts

Not Acknowledge interrupts (NCKI bit = 1 in I2CISTAT) occur in MASTER Mode when a Not Acknowledge is received or sent by the I^{2}C controller and the START or STOP bit is not set in the I^{2}C 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^{2}C 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.

General Purpose Timer Interrupt from Baud Rate Generator

If the I^{2}C controller is disabled (IEN bit in the I2CCTL Register = 0) and the BIRQ 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^{2}C controller to be used as a general-purpose timer when the I^{2}C controller is disabled.

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.

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- 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 (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 can control 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.

Master Transactions

The following sections describe master read and write transactions to both 7-bit and 10-bit slaves.

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 can occur 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.

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 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).

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

Master Write Transaction with a 7-Bit Address

Figure 28 displays 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 28. Data Transfer Format—Master Write Transaction with a 7-Bit Address

The procedure for a master transmit operation to a 7-bit addressed slave is as follows:

1. The software initializes the MODE field in the I^2C Mode Register for MASTER/SLAVE Mode with either a 7- 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^2C Control Register.
2. The software asserts the TXI bit of the I^2C Control Register to enable transmit interrupts.
3. The I^2C interrupt asserts, because the I^2C 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^2C Data Register.
5. The software sets the START bit of the I^2C Control Register.
6. The I^2C controller sends a START condition to the I^2C slave.
7. The I^2C controller loads the I^2C Shift Register with the contents of the I^2C 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^2C Data Register.
10. The I^2C controller shifts the remainder of the address and the Write bit out via the SDA signal.
11. The I^2C slave sends an Acknowledge (by pulling the SDA signal Low) during the next High period of SCL. The I^2C controller sets the ACK bit in the I^2C Status Register.

    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 and clearing the TXI bit. The I^2C 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^2C controller loads the contents of the I^2C Shift Register with the contents of the I^2C Data Register.
13. The I^2C 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^2C 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^2C Control Register.
17. The I^2C controller completes transmission of the data on the SDA signal.
18. The I^2C controller sends a STOP condition to the I^2C bus.

Note: If the slave terminates the transaction early by responding with a Not Acknowledge during the transfer, the I^2C 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^2C controller hardware automatically flushes transmit data in this not acknowledge case.

Master Write Transaction with a 10-Bit Address

Figure 29 displays 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 29. Data Transfer Format—Master Write Transaction with a 10-Bit Address

The first seven 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 Read/Write control bit (which is cleared to 0). The transmit operation is performed in the same manner as 7-bit addressing.

The procedure for a master transmit operation to a 10-bit addressed slave is as follows:

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 ensuing 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, refer to 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.

Master Read Transaction with a 7-Bit Address

Figure 30 displays the data transfer format for a read operation to a 7-bit addressed slave.

S	Slave Address	R = 1	A	Data	A	Data	A	P/S

Figure 30. Data Transfer Format—Master Read Transaction with a 7-Bit Address

The procedure for a master Read operation to a 7-bit addressed slave is as follows:

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^{2}C controller sends the address and Read bit out via the SDA signal.
7. The I^{2}C 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^{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.
8. The I^{2}C controller shifts in the first byte of data from the I^{2}C slave on the SDA signal.
9. The I^{2}C controller asserts the receive interrupt.
10. 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.
11. The I^{2}C controller sends a Not Acknowledge to the I^{2}C slave if the next byte is the final byte; otherwise, it sends an Acknowledge.
12. If there are more bytes to transfer, the I^{2}C controller returns to Step 7.
13. A NAK interrupt (NCKI bit in I2CISTAT) is generated by the I^{2}C controller.
14. The software responds by setting the STOP bit of the I^{2}C Control Register.
15. A STOP condition is sent to the I^{2}C slave.

Master Read Transaction with a 10-Bit Address

Figure 31 displays 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	A	P

Figure 31. Data Transfer Format—Master Read Transaction with a 10-Bit Address

The first seven 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.

The data transfer procedure for a Read operation to a 10-bit addressed slave is as follows:

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 and clearing the TXI 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 I2C controller shifts out the next eight bits of the address. After the first bit shifts, the I2C controller generates a transmit interrupt.
12. The software responds by setting the START bit of the I2C 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 I2C Data Register.
14. To read only one byte, the software responds by setting the NAK bit of the I2C Control Register.
15. After the I2C controller shifts out the address bits listed in Step 9 (the second address transfer), the I2C 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 I2C controller sets the NCKI bit in the I2C Status Register, sets the ACKV bit and clears the ACK bit in the I2C State Register. The software responds to the Not Acknowledge interrupt by setting the STOP bit and clearing the TXI bit. The I2C 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 I2C controller sends a repeated START condition.
17. The I2C controller loads the I2C Shift Register with the contents of the I2C Data Register (the third address transfer).
18. The I2C controller sends 11110b, followed by the two most-significant bits of the slave read address and a 1 (Read).
19. The I2C slave sends an Acknowledge by pulling the SDA signal Low during the next High period of SCL.
20. The I2C controller shifts in a byte of data from the slave.
21. The I2C controller asserts the Receive interrupt.
22. The software responds by reading the I2C Data Register. If the next data byte is to be the final byte, the software must set the NAK bit of the I2C Control Register.
23. The I2C controller sends an Acknowledge or Not Acknowledge to the I2C slave, based on the value of the NAK bit.
24. If there are more bytes to transfer, the I2C controller returns to Step 18.
25. The I2C controller generates a NAK interrupt (the NCKI bit in the I2CISTAT Register).
26. The software responds by setting the STOP bit of the I2C Control Register.
27. A STOP condition is sent to the I2C slave.

Slave Transactions

The following sections describe Read and Write transactions to the I^{2}C controller configured for 7-BIT and 10-BIT SLAVE modes.

Slave Address Recognition

The following slave address recognition options are supported.

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^{2}C controller automatically responds during the Acknowledge phase with the value in the NAK bit of the I2CCTL Register.

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 0's with the R/$\overline{W}$ bit = 0. A START byte is a 7-bit address of all 0's 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^{2}C 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^{2}C specification.

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^{2}C 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.

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

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 32. The procedure that follows describes the I^2C Master/Slave Controller operating as a slave in 7-BIT ADDRESS 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 32. Data Transfer Format—Slave Receive Transaction with 7-Bit Address

1. The software configures the controller for operation as a slave in 7-BIT ADDRESS 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 bus master initiates a transfer, sending the address byte. In SLAVE Mode, the I^2C controller recognizes its own address and detects that the R/$\overline{W}$ bit = 0 (written from the master to the slave). The I^2C 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^2C 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^2C controller receives the data 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.
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 steps 4 to 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^2C 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.

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 33. The procedure that follows describes the I^2C Master/Slave Controller operating as a slave in 10-BIT ADDRESS 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 33. Data Transfer Format—Slave Receive Transaction with 10-Bit Address

1. The software configures the controller for operation as a slave in 10-BIT ADDRESS 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^2C 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 the 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^{2}C 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^{2}C 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 steps 5 to 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^{2}C 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.

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 37. The procedure that follows describes the I^{2}C Master/Slave Controller operating as a slave in 7-BIT ADDRESS Mode and transmitting data to the bus master.

S	Slave Address	R = 1	A	Data	A	Data	A	P/S

Figure 34. Data Transfer Format—Slave Transmit Transaction with 7-bit Address

1. The software configures the controller for operation as a slave in 7-BIT ADDRESS 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, 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 Register to enable transmit interrupts. When the master initiates the data transfer, the I^2C 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^2C 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 Not Acknowledge instruction if this byte is the final data byte).
8. The bus cycles through steps 5 to 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^2C controller holds SCL Low until the data Register has been written. When a Not Acknowledge instruction is received by the slave, the I^2C 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 (set SPRS bit in I2CISTAT Register).
12. The software responds to the STOP/RESTART interrupt by reading the I2CISTAT Register, which clears the SPRS bit.

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 35. The following procedure describes the I^{2}C Master/Slave Controller operating as a slave in 10-BIT ADDRESS 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	A	P

Figure 35. Data Transfer Format—Slave Transmit Transaction with 10-Bit Address

1. The software configures the controller for operation as a slave in 10-BIT ADDRESS 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 I2CMODE Register.
   d. Set IEN = 1, NAK = 0 in the I^{2}C Control Register.
2. The master initiates a transfer, 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 the 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 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 the R/$\overline{W}$ set to 1. The SLAVE Mode I^{2}C controller recognizes the RESTART instruction followed by the first address byte with a match to SLA[9:8] and detects the R/$\overline{W}$ = 1 (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^2C controller acknowledges on the bus.

6. The software responds to the interrupt by reading the I2CISTAT Register, 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^2C 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^2C 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^2C 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 steps 7 to 10 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^2C controller holds SCL Low until the data register is written.

    When a Not Acknowledge is received by the slave, the I^2C 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 I^2C controller asserts the STOP/RESTART interrupt (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.

I^2C Data Register

The I^2C Data Register, shown in Table 92, contains the data that is to be loaded into the Shift Register to transmit onto the I^2C bus. This register also contains data that is loaded from the Shift Register after it is received from the I^2C bus. The I^2C 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 I^{2}C controller is in SLAVE MODE and data is being received).

Table 92. I^{2}C Data Register (I2CDATA)

Bit	7	6	5	4	3	2	1	0
Field	DATA							
RESET	0							
R/W	R/W							
Address	F50H							

I^{2}C Interrupt Status Register

The read-only I^{2}C Interrupt Status Register, shown in Table 93, indicates the cause of any current I^{2}C interrupt and provides status of the I^{2}C 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 rather provide status associated with the SAM bit interrupt.

Table 93. I^{2}C Interrupt Status Register (I2CISTAT)

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 I^{2}C Controller is enabled, this bit is 1 when the I^{2}C Data Register is empty. When set, this bit causes the I^{2}C Controller to generate an interrupt, except when the I^{2}C 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 I^{2}C Controller is enabled and the I^{2}C Controller has received a byte of data. When asserted, this bit causes the I^{2}C 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 I^{2}C Controller is enabled in SLAVE Mode and an address is received which matches the unique Slave address or General Call Address (if enabled by the GCE bit in the I^{2}C Mode Register). In 10-BIT ADDRESS 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.

Bit	Description (Continued)
[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 I^2C 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.
[2] ARBLST	**Arbitration Lost** This bit is set when the I^2C Controller is enabled in MASTER Mode and loses arbitration (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^2C 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 (Master reading data from Slave), indicating the Master is finished reading. A STOP or RESTART condition follows. In SLAVE Mode this bit clears when the I2CISTAT Register is read.

I²C Control Register

The I²C Control Register, shown in Table 94, enables and configures I²C operation.

Table 94. I²C Control Register (I2CCTL)

Bit	7	6	5	4	3	2	1	0
Field	IEN	START	STOP	BIRQ	TXI	NAK	FLUSH	FILTEN
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W*	R/W*	R/W	R/W	R/W1	R/W	R/W
Address	F52H							

Note: R/W = This bit may be set (write 1) but not cleared.

Bit	Description
[7] IEN	**I²C Enable** This bit enables the I²C Controller.
[6] START	**Send Start Condition** When set, this bit causes the I²C Controller (when configured as the Master) to send the Start condition. Once asserted, it is cleared by the I²C 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, the START condition is sent if there is data in the I2CDATA or I2CSHIFT Register. If there is no data in one of these registers, the I²C Controller waits until data is loaded. If this bit is set while the I²C 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 also 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 I²C Controller (when configured as the Master) to send the STOP condition after the byte in the I²C Shift Register has completed transmission or after a byte has been received in a receive operation. When set, this bit is reset by the I²C 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 STOP is set while a SLAVE Mode transaction is underway, the STOP bit will be cleared by hardware.
[4] BIRQ	**Baud Rate Generator Interrupt Request** This bit is ignored when the I²C Controller is enabled. If this bit is set = 1 when the I²C 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 I²C Data Register is empty.

Bit	Description (Continued)
[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^2C Data Register and sets the TDRE bit to 1. This bit allows flushing of the I^2C Data Register when an NAK condition is received after the next data byte has been written to the I^2C Data Register. Reading this bit always returns 0.
[0] FILTEN	**I^2C 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^2C 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.

I^2C Baud Rate High and Low Byte Registers

The I^2C Baud Rate High and Low Byte registers, shown in Tables 95 and 96, combine to form a 16-bit reload value, BRG[15:0], for the I^2C Baud Rate Generator. The I^2C baud rate is calculated using the following equation:

$$I^2C\ Baud\ Rate\ (bps) = \frac{System\ Clock\ Frequency\ (Hz)}{4 \times BRG[15:0]}$$

Note: In the preceding equation, if BRG = 0000h, use 10000h.

Table 95. I^2C Baud Rate High Byte Register (I2CBRH)

Bit	7	6	5	4	3	2	1	0
Field	BRH							
RESET	FFH							
R/W	R/W							
Address	F53H							

Bit	Description
[7] BRH	**I^2C Baud Rate High Byte** Most significant byte, BRG[15:8], of the I^2C Baud Rate Generator's reload value.

Note: If the DIAG bit in the I²C Mode Register is set to 1, a read of the I2CBRH Register returns the current value of the I²C Baud Rate Counter[15:8].

Table 96. I²C Baud Rate Low Byte Register (I2CBRL)

Bit	7	6	5	4	3	2	1	0
Field	BRL							
RESET	FFH							
R/W	R/W							
Address	F54H							

Bit	Description
[7] BRL	**I²C Baud Rate Low Byte** 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].

I²C State Register

The read-only I²C State Register, shown in Table 97, 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 the I²C bus. When the DIAG bit of the I²C Mode Register is set, this register returns the value of the I²C Controller state machine, which is shown in Table 98.

Table 97. I²C 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	X	X
R/W	R	R	R	R	R	R	R	R
Address	F55H							

Bit	Description
[7] ACKV	
[6]	
[5]	
[4]	
[3]	
[2]	
[1]	
[0]	

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, 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.

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.

AS—Address State
This bit is active High while the address is being transferred on the I^2C bus.

DS—Data State
This bit is active High while the data is being transferred on the I^2C bus.

10B—This bit indicates whether a 10 or 7-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 once the address has been sent.

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

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 I2C bus can be observed via the GPIO Input Register.

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 98. I^{2}C 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]	
[6]	
[5]	
[4]	
[3]	
[2]	
[1]	
[0]	

I2CSTATE_H—I^{2}C State
This field defines the current state of the I^{2}C Controller. It is the most significant nibble of the internal state machine. Table 99 defines the states for this field.

I2CSTATE_L—Least significant nibble of the I^2C state machine. This field defines the substates for the states defined by I2CSTATE_H. Table 100 defines the values for this field.

Table 99. I2CSTATE_H

State Encoding	State Name	State Description
0000	Idle	I^2C bus is idle or I^2C controller is disabled.
0001	Slave Start	I^2C 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	9 substates, one for each data bit and one for the Acknowledge.
1001	Slave Receive Data	9 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) 9 substates, one for each address bit and one for the Acknowledge.
1011	Slave Receive Addr2	Slave Receiving second address byte (10-bit addressing) 9 substates, one for each address bit and one for the Acknowledge.
1100	Master Transmit Data	9 substates, one for each data bit and one for the Acknowledge.
1101	Master Receive Data	9 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) 9 substates, one for each address bit and one for the Acknowledge.
1111	Master Transmit Addr2	Master sending second address byte (10-bit addressing) 9 substates, one for each address bit and one for the Acknowledge.

Table 100. 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.
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

I²C Mode Register

The I²C Mode Register, shown in Table 101, provides control over master versus slave operating mode, slave address and diagnostic modes.

Table 101. I²C Mode Register (I2CMODE)

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	R/W		R/W	R/W	R/W		R/W
Address	F56H							

Bit	Description
[7]	
[6]	

Bit	Description (Continued)
[5]	
[4]	
[3]	
[2]	
[1]	
[0]	

MODE—Selects the I^2C 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

IRM—Interactive Receive Mode
Valid in SLAVE Mode when software needs to 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.

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^2C Status Register indicates whether the address matched the General Call Address/START byte or not. Following the General Call Address byte, software may set the IRM bit that allows software to examine the following data byte(s) before acknowledging.

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.

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^2C 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.

I^2C Slave Address Register

The I^2C Slave Address Register (Table 102) provides control over the lower order address bits used in 7- and 10-bit slave address recognition.

Table 102. I^2C Slave Address Register (I2CSLVAD)

Bit	7	6	5	4	3	2	1	0
Field	SLA[7:0]							
RESET	00H							
R/W	R/W							
Address	F57H							

Bit	Description
[7]	
[6]	
[5]	
[4]	
[3]	
[2]	
[1]	
[0]	

SLA[7:0] — **Slave Address Bits 7-0**. Initialize with the appropriate Slave address value. When using 7 bit Slave addressing, SLA[9:7] are ignored.

Comparator and Operational Amplifier

The Z8FMC16100 Series MCU devices feature a general-purpose comparator and an operational amplifier. The comparator is a moderate speed (200 ns propagation delay) device that is designed for a maximum input offset of 5 mV. The comparator is used to compare two analog input signals. General-purpose input pins (CINP and CINN) provide the comparator inputs. The output is available as an interrupt source.

The operational amplifier is a two-input, one-output operational amplifier with a open loop gain of 10,000 (80 dB). One general-purpose input pin (OPINP) provides a non-inverting amplifier input, while another general-purpose input pin (OPINN) provides the inverting amplifier input. The output is available at the output pin (OPOUT).

The key operating characteristics of the operational amplifier are:

- Input common-mode range from GND (0.0 V) to V_{DD} – 1 V
- Input offset voltage less than 15 mV
- Output voltage swing from GND + 0.1 V to V_{DD} – 0.1 V
- Input bias current less than 1 nA
- Operating the operational amplifier open loop (no feedback) effectively provides another on-chip comparator (if appropriate)

Comparator Operation

The comparator output reflects the relationship between the non-inverting input and the inverting (reference) input. If the voltage on the non-inverting input is higher than the voltage on the inverting input, the comparator output is at a High state. If the voltage on the non-inverting input is lower than the voltage on the inverting input, the comparator output is at a Low state.

To operate, the comparator must be enabled by setting the CMPEN bit in the Comparator and Op Amp Register to 1. In addition the CINP and CINN comparator input alternate functions must be enabled on their respective GPIO pins. For more information, see the GPIO Alternate Functions section on page 34.

The comparator does not automatically power-down. To reduce operating current when not in use, the comparator is disabled by clearing the CMPEN bit to 0.

Operational Amplifier Operation

To operate, the operational amplifier must be enabled by setting the OPEN bit in the Comparator and Op Amp Register to 1. In addition, the OPINP, OPINN and OPOUT alternate functions must be enabled on their respective GP I/O pins. For more information, see the GPIO Alternate Functions section on page 34.

The logical value of the operational amplifier output (OPOUT) can be read from the Port 3 data input register if both the operational amplifier and input pin Schmitt trigger are enabled. For more information, see the GPIO Alternate Functions section on page 34. The operational amplifier can also generate an interrupt through the GPIO Port B3 input interrupt, if enabled.

The output of the operational amplifier is also connected to an analog input (ANA3) of the Analog-to-Digital Converter (ADC) multiplexer.

The operational amplifier does not automatically power-down. To reduce operating current when not in use, the operational amplifier is disabled by clearing the OPEN bit in the Comparator and Op Amp Register to 0. When the operational amplifier is disabled, the output is high-impedance. Operation of operational amplifier in unity gain mode may require external compensation.

Interrupts

The comparator generates an interrupt on any change in the logic output value (from 0 to 1 and 1 to 0). For information about enabling and prioritization of the comparator interrupt, see the Interrupt Controller chapter on page 48.

Comparator and Op Amp Control Register

The Comparator and Op Amp Control (CMPOPC) Register, shown in Table 103, enables the comparator and operational amplifier and provides access to the comparator output.

Table 103. Comparator and Op Amp Control Register (CMPOPC)

Bit	7	6	5	4	3	2	1	0
Field	OPEN	Reserved		CPSEL	CMPIRQ	CMPIV	CMPOUT	CMPEN
RESET	0	00		0	0	0	X	0
R/W	R/W	R/W		R/W	R/W	R/W	R	R/W
Address	F90H							

Bit	Description
[7] OPEN	**Operational Amplifier Disable** 0 = Operational amplifier is disabled. 1 = Operational amplifier is enabled.
[6:5]	**Reserved** These bits are reserved and must be programmed to 00.
[4] CPSEL	**Comparator Input Select** 0 = Comparator input is PA1 1 = Comparator input is PB4
[3] CMPIRQ	**Comparator Interrupt Edge Select** 0 = Interrupt Request on Comparator Rising Edge 1 = Interrupt Request on Comparator Falling Edge
[2] CMPIV	**PWM Fault Comparator Polarity** 0 = PWM Fault is active when CP- > CP+ 1 = PWM Fault is active when CP+ > CP-
[1] CMPOUT	**Comparator Output Value** 0 = Comparator output is logical 0. 1 = Comparator output is logical 1.
[0] CMPEN	**Comparator Enable** 0 = Comparator is disabled. 1 = Comparator is enabled.

Analog-to-Digital Converter

The Z8FMC16100 Series MCU includes an eight-channel Analog-to-Digital Converter (ADC). The ADC converts an analog input signal to a 10-bit binary number.

The features of the successive-approximation ADC include:

- Eight analog input sources multiplexed with general-purpose I/O ports
- Fast conversion time, less than 5µs
- Programmable timing controls
- Interrupt upon conversion complete
- Internal voltage reference generator
- Internal reference voltage available externally
- Ability to supply external reference voltage
- Timer count capture on every ADC conversion

Architecture

Figure 36 displays the ADC architecture, which consists of an 8-input multiplexer, sample-and-hold amplifier and 10-bit successive-approximation 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.

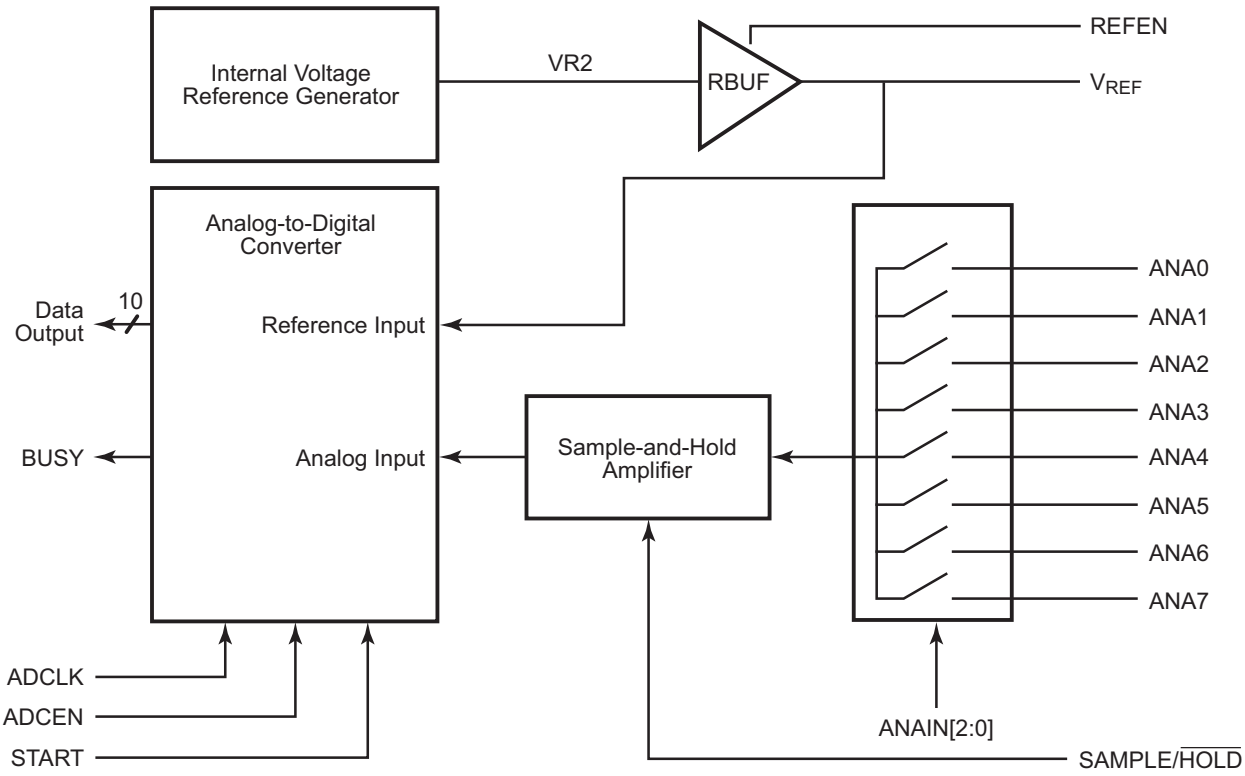


Figure 36. Analog-to-Digital Converter Block Diagram

Operation

The ADC converts an analog input ($ANA_{X)}$ to a 10-bit digital representation. The equation for calculating the digital value is shown below:

$$\text{ADC Output} = 1024 \times (ANA_x \div V_{REF})$$

Assuming zero gain and offset errors, any voltage outside the ADC input limits of AV_{SS} and V_{REF} returns all 0s or 1s, respectively.

A new conversion is initiated by either software write to the ADC Control Register's START bit or by PWM trigger. For information about the PWM trigger, see the Synchronization of PWM and Analog-to-Digital Converter section on page 70.

To avoid disrupting a conversion already in progress, the START bit can be read to indicate ADC operation status (busy or available).

Caution: Starting a new conversion while another conversion is in progress will stop the conversion in progress and result in the new conversion to not complete.

ADC Timing

Each ADC measurement consists of 3 phases:

1. Input sampling (programmable, minimum of 1.0µs).
2. Sample-and-hold amplifier settling (programmable, minimum of 0.5µs).
3. Conversion is 13 ADCLK cycles.

Figure 37 displays the control and flow of an ADC conversion.

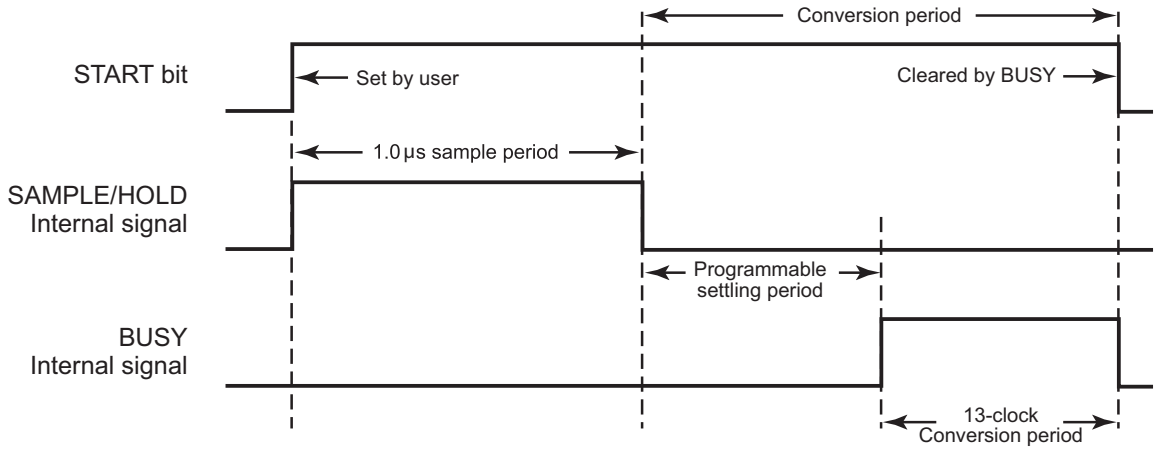


Figure 37. ADC Timing Diagram

Figure 38 displays the timing of an ADC conversion period.

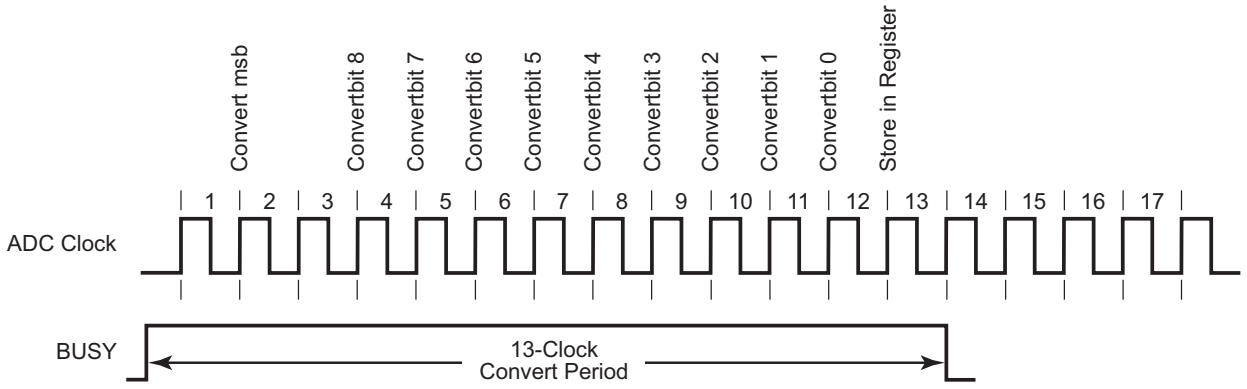


Figure 38. ADC Convert Timing

ADC Interrupt

The ADC can generate an interrupt request when a conversion is completed. An interrupt request that is pending when the ADC is disabled is not automatically cleared.

ADC Timer 0 Capture

The Timer 0 count is captured for every ADC conversion. The capture of the Timer 0 count occurs after the programmed sample time is complete for every conversion and stored in the ADC Timer Capture Register (ADCTCAP).

Reference Buffer

The reference buffer (RBUF) supplies the reference voltage for the ADC. When enabled, the internal voltage reference generator supplies the ADC and the voltage is available on the V_{REF} pin. When RBUF is disabled, the ADC must have the reference voltage supplied externally through the V_{REF} pin. RBUF is controlled by the REFEN bit in the ADC Control Register.

Internal Voltage Reference Generator

The Internal Voltage Reference Generator provides the voltage (VR2) for the RBUF. This VR2 value is 2 V.

Calibration and Compensation

You can perform calibration and store the values into Flash, or the user code can perform a manual offset calibration. There is no provision for manual gain calibration.

ADC Control Register 0

The ADC Control Register 0 (ADCCTL0), shown in Table 104, initiates the analog-to-digital conversion and provides ADC status information.

Table 104. ADC Control Register 0 (ADCCTL0)

Bit	7	6	5	4	3	2	1	0
Field	START	Reserved	REFEN	ADCEN	Reserved	ANAIN[2:0]		
RESET	0	0	0	0	0	0	0	0
R/W	R/W1	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F70H							

Bit	Description
[7] START	**ADC Start/Busy** 0 = Writing to 0 has no effect; reading a 0 indicates that the ADC is available to begin a conversion. 1 = Writing to 1 starts a conversion; reading a 1 indicates that a conversion is currently in progress.
[6]	**Reserved** This bit is reserved and must be programmed to 0.
[5] REFEN	**Reference Enable** 0 = Internal reference voltage is disabled allowing an external reference voltage to be used by the ADC. 1 = Internal reference voltage for the ADC is enabled. The internal reference voltage can be measured on the V_{REF} pin.
[4] ADCEN	**ADC Enable** 0 = ADC is disabled for low power operation. 1 = ADC is enabled for normal use.
[3]	**Reserved** This bit is reserved and must be programmed to 0.

Bit	Description (Continued)
[2:0] ANAIN	**Analog Input Select** 000 = ANA0 input is selected for analog-to-digital conversion. 001 = ANA1 input is selected for analog-to-digital conversion. 010 = ANA2 input is selected for analog-to-digital conversion. 011 = ANA3 input is selected for analog-to-digital conversion. 100 = ANA4 input is selected for analog-to-digital conversion. 101 = ANA5 input is selected for analog-to-digital conversion. 110 = ANA6 input is selected for analog-to-digital conversion. 111 = ANA7 input is selected for analog-to-digital conversion.

ADC Raw Data High Byte Register

The ADC Data Raw High Byte Register, shown in Table 105, contains the upper eight bits of raw data from the ADC output. Access to the ADC Raw Data High Byte Register is read-only. This register is used for test only.

Table 105. ADC Raw Data High Byte Register (ADCRD_H)

Bit	7	6	5	4	3	2	1	0
Field	ADCRDH							
RESET	X							
R/W	R							
Address	F71H							

Note: X = Undefined.

Bit	Description
[7:0] ADCRDH	**ADC Raw Data High Byte** 00H–FFH = The data in this register is the raw data coming from the SAR Block. It will change as the conversion is in progress. This register is used for testing only.

ADC Data High Byte Register

The ADC Data High Byte Register, shown in Table 106, contains the upper eight bits of the ADC output. Access to the ADC Data High Byte Register is read-only. Reading the ADC Data High Byte Register latches data in the ADC Low Bits Register.

Table 106. ADC Data High Byte Register (ADCD_H)

Bit	7	6	5	4	3	2	1	0
Field	ADCDH							
RESET	X							
R/W	R							
Address	F72H							

Note: X = Undefined.

Bit	Description
[7:0] ADCDH	**ADC High Byte** 00H–FFH = The last conversion output is held in the data registers until the next ADC conversion has completed.

ADC Data Low Bits Register

The ADC Data Low Bits Register, shown in Table 107, contain the lower bits of the ADC output as well as an overflow status bit. Access to the ADC Data Low Bits Register is read-only. Reading the ADC Data High Byte Register latches data in the ADC Low Bits Register.

Table 107. ADC Data Low Bits Register (ADCD_L)

Bit	7	6	5	4	3	2	1	0
Field	ADCDL		Reserved					
RESET	X		X					
R/W	R		R					
Address	F73H							

Note: X = Undefined.

Bit	Description
[7:6] ADCDL	**ADC Low Bits** 00–11b = These bits are the 2 least significant bits of the 10-bit ADC output. These bits are undefined after a Reset. The low bits are latched into this register whenever the ADC Data High Byte Register is read.
[5:0]	**Reserved** These bits are reserved and must be programmed to 000000.

Sample Settling Time Register

The Sample Settling Time Register, shown in Table 108, is used to program the length of time from the SAMPLE/$\overline{\text{HOLD}}$ signal to the START signal, when the conversion can begin. The number of clock cycles required for settling varies from system to system depending on the system clock period used. You must program this register to contain the number of clocks required to meet a 0.5µs minimum settling time.

Table 108. Sample and Settling Time (ADCSST)

Bit	7	6	5	4	3	2	1	0
Field	Reserved			SST				
RESET	0			1	1	1	1	1
R/W	R			R/W				
Address	F74H							

Bit	Description
[7:5]	**Reserved** These bits are reserved and must be programmed to 000.
[4:0] SST	**Sample Settling Time** 0H–FH = The number of system clock periods to meet a 0.5µs minimum.

Sample Time Register

The Sample Time Register, shown in Table 109, is used to program the length of active time for the sample after a conversion has begun by setting the START bit in the ADC Control Register or initiated by the PWM. The number of system clock cycles required for sample time varies from system to system, depending on the clock period used. You must program this register to contain the number of system clocks required to meet a 1µs minimum sample time.

Table 109. Sample/Hold Time (ADCST)

Bit	7	6	5	4	3	2	1	0
Field	Reserved		ST					
RESET	0		1	1	1	1	1	1
R/W	R/W		R/W					
Address	F75H							

Bit	Description
[7:6]	**Reserved** These bits are reserved and must be programmed to 00.
[5:0] ST	**Sample/Hold Time** 0H–FH = The number of system clock periods to meet a 1µs minimum.

ADC Clock Prescale Register

The ADC Clock Prescale Register, shown in Table 110, is used to provide a divided system clock to the ADC. When this register is programmed with 0h, the System Clock is used for the ADC Clock.

Table 110. ADC Clock Prescale Register (ADCCP)

Bit	7	6	5	4	3	2	1	0
Field	Reserved				DIV16	DIV8	DIV4	DIV2
RESET	0				0	0	0	0
R/W	R/W							
Address	F76H							

Bit	Description
[7:4]	**Reserved** These bits are reserved and must be programmed to 0000.
[3] DIV16	**Divide By 16** 0 = Clock is not divided. 1 = System Clock is divided by 16 for ADC Clock.
[2] DIV8	**Divide By 8** 0 = Clock is not divided. 1 = System Clock is divided by 8 for ADC Clock.
[1] DIV4	**Divide By 4** 0 = Clock is not divided. 1 = System Clock is divided by 4 for ADC Clock.
[0] DIV2	**Divide By 2** 0 = Clock is not divided. 1 = System Clock is divided by 2 for ADC Clock.

ADC Timer Capture High Byte Register

The High byte of the ADC Timer Capture Register, shown in Table 111, contains the upper eight bits of the ADC Timer 0 count. Access to the ADC Timer Capture High Byte Register is read-only.

Table 111. ADC Timer Capture High Byte Register (ADCTCAP_H)

Bit	7	6	5	4	3	2	1	0
Field	ADCTCAPH							
RESET	X							
R/W	R							
Address	F08H							

Bit	Description
[7:0] ADCTCAPH	**ADC Timer Capture Count High Byte** 00H–FFH = The timer count is held in the data registers until the next ADC conversion is started.

ADC Timer Capture Low Byte Register

The low byte of the ADC Timer Capture Low Byte Register, shown in Table 112, contains the lower eight bits of the ADC Timer 0 count. Access to the ADC Timer Capture Low Byte Register is read-only.

Table 112. ADC Timer Capture Low Byte Register (ADCTCAP_L)

Bit	7	6	5	4	3	2	1	0
Field	ADCTCAPL							
RESET	X							
R/W	R							
Address	F09H							

Bit	Description
[7:0] ADCTCAPL	**ADC Timer Capture Count Low Byte** 00H–FFH = The timer count is held in the data registers until the next ADC conversion is started.

Program Memory

The Z8FMC16100 Series MCU products feature up to 16KB (16,384 bytes) of nonvolatile Flash Memory with read/write/erase capability. Flash Memory can be programmed and erased in-circuit by either user code or through the On-Chip Debugger (OCD).

The Flash Memory array is arranged in 512-byte pages; 512 bytes is the minimum Flash block size that can be erased. Flash Memory is also divided into 8 sectors which can be protected from programming and erase operations on a per-sector basis.

Table 113 lists the Flash Memory configuration for each device in the Z8FMC16100 Series of MCUs. Table 114 lists the sector address ranges, and Figure 39 displays the Flash Memory arrangement.

Table 113. Flash Memory Configurations

Part Number	Flash Size	Number of Pages	Program Memory Addresses	Sector Size	Number of Sectors	Pages per Sector
Z8FMC16	16K (16,384)	32	0000h–3FFFh	2K (2048)	8	4
Z8FMC08	8K (8,192)	16	0000h–1FFFh	1K (1024)	8	2
Z8FMC04	4K (4,096)	8	0000h–0FFFh	512	8	1

Table 114. Flash Memory Sector Addresses

	Flash Sector Address Ranges		
Sector Number	Z8FMC16	Z8FMC08	Z8FMC04
0	0000h–07FFh	0000h–03FFh	0000h–01FFh
1	0800h–0FFFh	0400h–07FFh	0200h–03FFh
2	1000h–17FFh	0800h–0BFFh	0400h–05FFh
3	1800h–1FFFh	0C00h–0FFFh	0600h–07FFh
4	2000h–27FFh	1000h–13FFh	0800h–09FFh
5	2800h–2FFFh	1400h–17FFh	0A00h–0BFFh
6	3000h–37FFh	1800h–1BFFh	0C00h–0DFFh
7	3800h–3FFFh	1C00h–1FFFh	0E00h–0FFFh

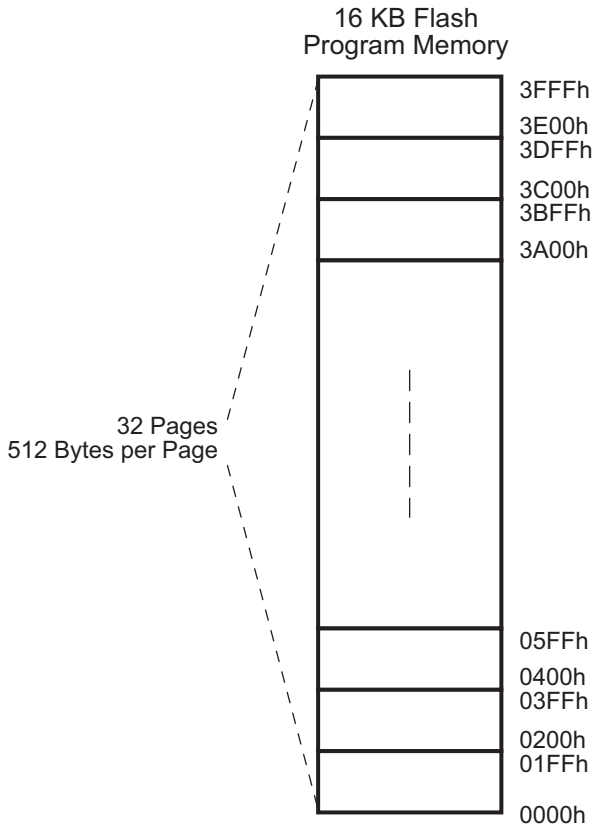


Figure 39. Flash Memory Arrangement

Information Area

Table 115 describes the Z8FMC16100 Series MCU's information area. This 512 byte information area is accessed by setting bit 7 of the Page Select Register to 1. When access is enabled, the information area is mapped into Program Memory and overlays the 512 bytes at addresses FE00h to FFFFh. When information area access is enabled, LDC instructions return data from the information area. CPU instruction fetches always arrive from Program Memory regardless of the information area access bit. Access to the information area is read-only.

Table 115. Z8FMC16100 Series MCU Information Area Map

Program Memory Address (Hex)	Function
FE00h–FE3Fh	Reserved.
FE40h–FE53h	**Part Number**: 20-character ASCII alphanumeric code, left-justified and filled with zeroes.
FE54h–FFFFh	Reserved.

Operation

The Flash Controller provides the appropriate signals and timing for the Byte Programming, Page Erase and Mass Erase operations performed in Flash Memory. The Flash Controller contains a protection mechanism, via the Flash Control Register (FCTL), to prevent accidental programming or erasure. The following sections provide details about the Lock, Unlock, Sector Protect, Byte Programming, Page Erase and Mass Erase operations.

- Timing Using Flash Frequency Registers
- Flash Read Protection
- Flash Write/Erase Protection
- Byte Programming
- Page Erase
- Mass Erase
- Flash Controller Bypass
- Flash Controller Behavior in Debug Mode

Timing Using Flash Frequency Registers

Before performing a program or erase operation on Flash Memory, the programmer must first configure the Flash Frequency High and Low Byte registers. The Flash Frequency registers allow programming and erasure of Flash Memory with system clock frequencies ranging from 32 kHz through 20 MHz (valid range is limited to device operating frequencies).

The Flash Frequency High and Low Byte registers combine to form a 16-bit value, FFREQ, to control the timing of Flash program and erase operations. The 16-bit Flash Frequency value must contain the system clock frequency in kHz. This value is calculated using the following equation:

$$FFREQ[15:0] = \frac{\text{System Clock Frequency (Hz)}}{1000}$$

Caution: Flash programming and erasure are not supported for system clock frequencies below 32 kHz, above 20 MHz, or outside of the device's operating frequency range. The Flash Frequency High and Low Byte registers must be loaded with the correct value to ensure proper Flash programming and erase operations.

Flash Read Protection

The user code within Flash Memory can be protected from external access. Programming the Flash Read Protect option bit prevents the reading of user code by the OCD or by using the Flash Controller Bypass mode. For more information, see the Option Bits chapter on page 221 and the On-Chip Debugger chapter on page 235.

Flash Write/Erase Protection

The Z8FMC16100 Series MCU device provides several levels of protection against accidental program and erasure of the contents of Flash Memory. This protection is provided by the following three features, each of which is described in this section:

- Flash Controller Unlock Mechanism
- Flash Sector Protection
- Flash Write Protection Option Bit

Flash Controller Unlock Mechanism

Upon Reset, the Flash Controller locks to prevent accidental program or erasure of Flash Memory. To program or erase Flash Memory, the Flash Controller must be unlocked. After unlocking the Flash Controller, Flash Memory can be programmed or erased. Any value written by user code to the Flash Control Register or the Page Select Register out of sequence locks the Flash Controller.

Observe the following procedure to unlock the Flash Controller from user code:

1. Write `00h` to the Flash Control Register to reset the Flash Controller.
2. Write the page to be programmed or erased to the Page Select Register.
3. Write the first unlock command `73h` to the Flash Control Register.
4. Write the second unlock command `8Ch` to the Flash Control Register.

5. Rewrite the page written in Step 2 to the Page Select Register.

Flash Sector Protection

The Flash Sector Protect Register can be configured to prevent sectors from being programmed or erased. After a sector is protected, it cannot be unprotected by user code. The Flash Sector Protect Register is cleared after reset and any previously written protection values are lost. User code must write this register in their initialization routine to enable sector protection.

The Flash Sector Protect Register shares its Register File address with the Page Select Register. The Flash Sector Protect Register is accessed by writing the Flash Control Register with 5EH. After the Flash Sector Protect Register is selected, it can be accessed at the Page Select Register address. When user code writes the Flash Sector Protect Register, bits can only be set to 1. Therefore, sectors can be protected, but not unprotected, through register Write operations. The Flash Sector Protect Register is deselected by writing any value to the Flash Control Register.

Observe the following procedure to setup the Flash Sector Protect Register from user code:

1. Write 00h to the Flash Control Register to reset the Flash Controller.
2. Write 5Eh to the Flash Control Register to select the Flash Sector Protect Register.
3. Read and/or write the Flash Sector Protect Register, which now resides at Register File address FF9h.
4. Write 00h to the Flash Control Register to return the Flash Controller to its reset state.

Flash Write Protection Option Bit

The Flash Write Protect option bit can be enabled to block all program and erase operations from user code. For more information, see the Option Bits chapter on page 221.

Byte Programming

When the Flash Controller is unlocked, Writes to Program Memory from user code programs a byte into the Flash if the address is located in the unlocked page. 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 a Page Erase or Mass Erase operation.

Byte programming can be performed using the eZ8 CPU's LDC or LDCI instructions. For a description of the LDC and LDCI instructions, refer to the eZ8 CPU Core User Manual (UM0128), which is available for download at www.zilog.com.

When the Flash Controller programs Flash Memory, the eZ8 CPU idles, but system clock and on-chip peripherals continue to operate. Interrupts which occur when a programming

operation is in progress are serviced after the programming operation is complete. To exit programming mode and lock the Flash Controller, write 00h to the Flash Control Register.

User code cannot program Flash Memory on a page which lies in a protected sector. When user code writes memory locations, only addresses located in the unlocked page are programmed. Memory Writes outside of the unlocked page are ignored.

Caution: Each memory location must not be programmed more than twice before an erase is required.

Observe the following procedure to program Flash Memory from user code:

1. Write 00h to the Flash Control Register to reset the Flash Controller.
2. Write the page of memory to be programmed to the Page Select Register.
3. Write the first unlock command 73h to the Flash Control Register.
4. Write the second unlock command 8Ch to the Flash Control Register.
5. Rewrite the page written in Step 2 to the Page Select Register.
6. Write Program Memory using LDC or LDCI instructions to program Flash.
7. Repeat Step 6 to program additional memory locations on the same page.
8. Write 00h to the Flash Control Register to lock the Flash Controller.

Page Erase

Flash Memory can be erased one page (512 bytes) at a time. Page-erasing Flash Memory sets all bytes in that page to the value FFh. The Page Select Register identifies the page to be erased. While the Flash Controller executes the Page Erase operation, the eZ8 CPU idles, but the system clock and on-chip peripherals continue to operate. The eZ8 CPU resumes operation after the Page Erase operation completes. Interrupts which occur when the Page Erase operation is in progress are serviced after the Page Erase operation is complete. When the Page Erase operation is complete, the Flash Controller returns to its locked state. Only pages located in unprotected sectors can be erased.

Observe the following procedure to perform a Page Erase operation:

1. Write 00h to the Flash Control Register to reset the Flash Controller.
2. Write the page to be erased to the Page Select Register.
3. Write the first unlock command (73h) to the Flash Control Register.
4. Write the second unlock command 8Ch to the Flash Control Register.
5. Rewrite the page written in Step 2 to the Page Select Register.

6. Write the Page Erase command (95h) to the Flash Control Register.

Mass Erase

The Flash Memory cannot be mass-erased by user code.

Flash Controller Bypass

The Flash Controller can be bypassed and the control signals for Flash Memory are brought out to the GPIO pins. Bypassing the Flash Controller allows faster programming algorithms by controlling the Flash programming signals directly.

Flash Controller Bypass is recommended for gang-programming applications and large-volume customers who do not require in-circuit programming of Flash Memory.

For more information about bypassing the Flash controller, refer to the Third Party Flash Programming Support for Z8 Encore! MCUs Application Note, which is available for download on www.zilog.com.

Flash Controller Behavior in Debug Mode

The following changes in behavior of the Flash Controller occur when the Flash Controller is accessed using the OCD:

- The Flash Write Protect option bit is ignored
- The Flash Sector Protect Register is ignored for programming and erase operations
- Programming operations are not limited to the page selected in the Page Select Register
- Bits in the Flash Sector 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
- The Page Select Register can be written when the Flash Controller is unlocked
- The Mass Erase command is enabled through the Flash Control Register
- The page erase and programming operations are disabled if the Memory Read Protect option is enabled

Flash Control Register Definitions

The Flash Control Register, shown in Table 116, unlocks the Flash Controller for programming and erase operations or to select the Flash Sector Protect Register.

The write-only Flash Control Register shares its Register File address with the read-only Flash Status Register.

Table 116. Flash Control Register (FCTL)

Bit	7	6	5	4	3	2	1	0
Field	FCMD							
RESET	00H							
R/W	W							
Address	FF8H							

Bit	Description
[7:0] FCMD	**Flash Command** 73H = First unlock command. 8CH = Second unlock command. 95H = Page erase command. 63H = Mass erase command. 5EH = Flash Sector Protect Register select. All other commands or any command out of sequence locks the Flash Controller.

Flash Status Register

The Flash Status Register, shown in Table 117, indicates the current state of the Flash Controller. This register can be read at any time. The read-only Flash Status Register shares its Register File address with the Write-only Flash Control Register.

Table 117. Flash Status Register (FSTAT)

Bit	7	6	5	4	3	2	1	0
Field	Reserved		FSTAT					
RESET	00B		00_0000B					
R/W	R		R					
Address	FF8H							

Bit	Description
[7:6]	**Reserved** These bits are reserved and must be programmed to 00.
[5:0] FSTAT	**Flash Controller Status** 00_0000 = Flash Controller locked. 00_0001 = First unlock command received. 00_0010 = Second unlock command received. 00_0011 = Flash Controller unlocked. 00_0100 = Flash Sector Protect Register selected. 0_1xxx = Program operation in progress. 01_0xxx = Page erase operation in progress. 10_0xxx = Mass erase operation in progress.

Flash Page Select Register

The Flash Page Select (FPS) Register, shown in Table 118, selects one of the 32 available Flash Memory pages to be erased or programmed. Each Flash Page contains 512-bytes of Flash Memory. During a Page Erase operation, all Flash Memory locations with the 7 most significant bits of the address assigned by the PAGE field are erased to FFh.

The Flash Page Select Register shares its Register File address with the Flash Sector Protect Register. The Flash Page Select Register cannot be accessed when the Flash Sector Protect Register is enabled.

Table 118. Flash Page Select Register (FPS)

Bit	7	6	5	4	3	2	1	0
Field	INFO_EN	PAGE						
RESET	0	000_0000B						
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 at addresses FE00H through FFFFH.
[6:0] PAGE	**Page Select** This 7-bit field selects the Flash Memory page for Programming and Page Erase operations. Program Memory Address[15:9] = PAGE[6:0].

Flash Sector Protect Register

The Flash Sector Protect Register, shown in Table 119, protects Flash Memory sectors from being programmed or erased from user code. The Flash Sector Protect Register shares its Register File address with the Flash Page Select Register. The Flash Sector Protect Register can be accessed only after writing the Flash Control Register with 5Eh.

User code can only write bits in this register to 1 (bits cannot be cleared to 0 by user code).

Table 119. Flash Sector Protect Register (FPROT)

Bit	7	6	5	4	3	2	1	0
Field	SECT7	SECT6	SECT5	SECT4	SECT3	SECT2	SECT1	SECT0
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	FF9H							

Note: *R/W = Register is accessible for Read operations and can only be written to 1 via user code.

Bit	Description
[7:0] SECT*n*	**Sector Protect** 0 = Sector *n* can be programmed or erased from user code. 1 = Sector *n* is protected and cannot be programmed or erased from user code.

Flash Frequency High and Low Byte Registers

The Flash Frequency High and Low Byte registers, shown in Table 120, combine to form a 16-bit value (FFREQ) to control timing for Flash program and erase operations. The 16-bit Flash Frequency registers must be written with the system clock frequency in kHz for Program and Erase operations.

Calculate the Flash Frequency value using the following equation:

$$FFREQ[15:0] = \{FFREQH[7:0],FFREQL[7:0]\} = \frac{\text{System Clock Frequency}}{1000}$$

Caution: Flash programming and erasure is not supported for system clock frequencies below 32 kHz, above 20 MHz, or outside the valid operating frequency range for the device. The Flash Frequency High and Low Byte registers must be loaded with the correct values to ensure proper program and erase times.

Table 120. Flash Frequency High and Low Byte Registers (FFREQH, FFREQL)

Bit	7	6	5	4	3	2	1	0
Field	FFREQH							
RESET	00H							
R/W	R/W							
Address	FFAH							
Bit	7	6	5	4	3	2	1	0
Field	FFREQL							
RESET	00H							
R/W	R/W							
Address	FFBH							

Bit	Description
[7:0] FFREQH, FFREQL	**Flash Frequency High and Low Bytes** These 2 bytes, {FFREQH[7:0], FFREQL[7:0]}, contain the 16-bit Flash Frequency value.

Option Bits

Option bits allow user configuration of certain aspects of the Z8FMC16100 Series MCU operation. The feature configuration data is stored in program memory and read during Reset.

The features available for control using the option bits include:

- Watchdog Timer (WDT) time-out selection of interrupt or Reset
- Watchdog Timer enabled at Reset
- Code protection by preventing external read access of program memory
- Ability to prevent accidental programming and erasure of program memory
- Voltage Brown-Out (VBO) can be disabled during STOP Mode to reduce power consumption
- External oscillator mode selection
- Selectable PWM OFF state, output polarity, fault state and Reset state
- Disable PWM output pairs, enabling them as inputs
- $\overline{\text{RESET}}$/Fault0 pin function selection
- Low power clock divide mode selection

Option Bit Types

Two types of option bits (user option and trim option bits) allow configuration of certain aspects of Z8FMC16100 Series MCU operation; each is described in this section.

- User option bits
- Trim option bits

User Option Bits

The user option bits are contained in the first two bytes of program memory. As these locations contain application-specific device configuration, you can alter these bytes by programming Flash Memory.

Note: The information in these bytes is lost when Page 0 of program memory is erased.

Trim Option Bits

The trim option bits are contained in the information page of Flash Memory. These bits are factory-programmed values required to optimize the operation of on-board analog circuitry and cannot be altered by the user. Program memory can be erased without endangering these values.

Caution: It is possible to alter the working values of these bits by accessing the Trim Bit Address and Data registers, but these working values are lost after a Reset.

There are 32 trim addresses. To read or write these values, the user code must first write a value between 00h and 1Fh into the Trim Bit Address Register. Writing the Trim Bit Data Register changes the working value of the target trim data. Reading the Trim Bit Data Register returns the working value of the target trim data.

User Option Bit Configuration By Reset

Each time the user option bits are programmed or erased, the device must be Reset for the change to take place.

Option Bit Address Space

The first two bytes of program memory at addresses 0000h, shown in Table 121, and 0001h, shown in Table 122, are reserved for the user option bits.

Table 121. User Option Bits at Program Memory Address 0000H

Bit	7	6	5	4	3	2	1	0
Field	WDT_RES	WDT_AO	OSC_SEL[1:0]		VBO_AO	RP	Reserved	FWP
RESET	U	U	U		U	U	U	U
R/W	R/W	R/W	R/W		R/W	R/W	R/W	R/W
Address	Program Memory 0000H							

Note: U = Unchanged by Reset; 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 Reset.

Bit	Description (Continued)
[6] WDT_AO	**Watchdog Timer Always On** 0 = Watchdog Timer is automatically enabled. 1 = Watchdog Timer is enabled on execution of the WDT instruction.
[5:4] OSC_SEL	**External Oscillator Mode Selection** 00 = Reserved. 01 = Minimum power for use with very low frequency crystals (32kHz to 1.0MHz). 10 = Medium power for use with medium frequency crystals or ceramic resonators (0.5MHz to 10.0MHz). 11 = Maximum power for use with high frequency crystals (8.0MHz to 20.0MHz).
[3] VBO_AO	**Voltage Brown-Out Always On** 0 = Voltage Brown-Out Protection is disabled in STOP Mode to reduce total power consumption. 1 = Voltage Brown-Out Protection is always enabled.
[2] RP	**Read Protect** 0 = External access to User program code is disabled. 1 = User program code is accessible.
[1]	**Reserved** This bit is reserved and must be programmed to 1.
[0] FWP	**Flash Write Protect** 0 = Programming, Page Erase and Mass Erase using User Code is disabled. 1 = Programming, Page Erase and Mass Erase are enabled for all of Flash Program Memory.

The bits described in Table 122 define PWM behavior. The High and Low default off state (the output polarity) is also defined here. The off state is used by the PWM output control and PWM Fault logic. PWM output pairs can be disabled and used as high-impedance input pins. The RESET/Fault0 pin function is also selectable.

Table 122. Options Bits at Program Memory Address 0001H

Bit	7	6	5	4	3	2	1	0
Field	FLTSEL	LPDEN	Reserved	PWM2EN	PWM1EN	PWM0EN	PWMHI	PWMLO
RESET	U	U	U	U	U	U	U	U
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	Program Memory 0001H							

Note: U = Unchanged by Reset; R/W = Read/Write.

Bit	Description
[7] FLTSEL	**$\overline{\text{RESET}}$/$\overline{\text{Fault0}}$ Select** 0 = $\overline{\text{RESET}}$/$\overline{\text{Fault0}}$ pin is configured as a $\overline{\text{Fault0}}$ input. 1 = $\overline{\text{RESET}}$/$\overline{\text{Fault0}}$ pin is configured as $\overline{\text{RESET}}$ input.

Bit	Description (Continued)
[6] LPDEN	**Low Power Divide Mode Enable** 0 = See the Oscillator Control chapter on page 227. Low Power Divide mode is enabled. 1 = Low Power Divide mode is disabled.
[5]	**Reserved** This bit is reserved and must be programmed to 1.
[4] PWM2EN	**PWM Output Pair PWM2 Enable** 0 = PWM2 outputs are enabled and controlled by PWM logic. 1 = PWM2 outputs are always high-impedance.
[3] PWM1EN	**PWM Output Pair PWM1 Enable** 0 = PWM1 outputs are enabled and controlled by PWM logic. 1 = PWM1 outputs are always high-impedance.
[2] PWM0EN	**PWM Output Pair PWM0 Enable** 0 = PWM0 outputs are enabled and controlled by PWM logic. 1 = PWM0 outputs are always high-impedance.
[1] PWMHI	**PWM High Side (PWM outputs 0H,1H, 2H) Default Off State** 0 = PWM High-side inactive state is Low, active state is High. 1 = PWM High-side inactive state is High, active state is Low.
[0] PWMLO	**PWM Low Side (PWM outputs 0L,1L, 2L) Default Off State** 0 = PWM Low-side inactive state is Low, active state is High. 1 = PWM Low-side inactive state is High, active state is Low.

Trim Bit Address Register

The Trim Bit Address Register, shown in Table 123, contains the target address for access to the trim option bits.

Table 123. Trim Bit Address Register (TRMADR)

Bit	7	6	5	4	3	2	1	0
Field	TRMADR							
RESET	00H							
R/W	R/W							
Address	FF6H							

Bit	Description
[7:0] TRMADR	**00–1FH = Trim Bit Address Register**

Trim Bit Data Register

The Trim Bit Data Register, shown in Table 124, contains the read or write data for access to addresses in the Trim Bit Address Register.

Table 124. Trim Bit Data Register (TRMDR)

Bit	7	6	5	4	3	2	1	0
Field	TRMDR							
RESET	00H							
R/W	R/W							
Address	FF7H							

Bit	Description
[7:0} TRMDR	**00–FFH = Trim Bit Data Register**

Trim Bit Address 0001H

The IPO Trim Bit Address Register at address 0001H, shown in Table 125, is loaded from the Flash Information Area following reset or Stop Mode Recovery. Write to this register allows user frequency adjustment of the IPO. Writing to this register does not effect the Flash Memory contents.

Table 125. IPO Trim Option Bits at 0001H (IPO_TRIM)

Bit	7	6	5	4	3	2	1	0
Field	TEMP_TRIM						IPO_TRIM[9:8]	
RESET	U							
R/W	R/W							
Address	Information Page Memory 0021H							

Note: U = Unchanged by Reset; R/W = Read/Write.

Bit	Description
[7:2] TEMP_TRIM	**Internal Precision Oscillator Trim Bits** 00–3FH = Used for temperature compensation.
[1:0] IPO_TRIM	**Internal Precision Oscillator Trim Byte** 00–11H = Trimming for Internal Precision Oscillator frequency adjustment; used with IPO Trim1 options bits as bits [9:8].

Trim Bit Address 0002H

IPO Trim Bit Address Register at address 0002H, shown in Table 126, is loaded from the Flash Information Area following reset or Stop Mode Recovery. Writing to this register allows user frequency adjustment of the IPO. Writing to this register does not effect the Flash Memory contents.

Table 126. IPO Trim1 Option Bits at 0002H (IPO_TRIM1)

Bit	7	6	5	4	3	2	1	0
Field	IPO_TRIM[7:0]							
RESET	U							
R/W	R/W							
Address	Information Page Memory 0022H							

Note: U = Unchanged by Reset; R/W = Read/Write.

Bit	Description
[7:0] IPO_TRIM	**Internal Precision Oscillator Trim Byte** 00–FFH = Trimming byte for Internal Precision Oscillator frequency adjustment. Use with IPO trim bits [9:8] in the IPO Trim Bit Address Register at address 0001H Register.

Trim Bit Address 0003H

V_{REF}_TRIM. The trim values for the V_{REF} circuit used by the Analog-to-Digital Converter (ADC) contains factory-trimmed values for V_{REF} (ADC Reference Voltage). These values are used to set the V_{REF} voltage to meet a specified tolerance, the format of which is to be determined in the future.

Oscillator Control

The Z8FMC16100 Series MCU uses three possible clocking schemes, each user-selectable:

- Trimmable internal precision oscillator
- On-chip oscillator using off-chip crystal/resonator or external clock driver
- On-chip low precision Watchdog Timer oscillator

In addition, Z8FMC16100 Series MCUs contain clock failure detection and recovery circuitry, allowing continued operation despite any potential failure of the primary oscillator.

The on-chip system clock frequency can be reduced through a clock divider allowing reduced dynamic power dissipation. Flash Memory can be powered down during portions of the clock period when running slower than 10 MHz.

Operation

This section explains the logic used to select the system clock, divide down the system clock and handle oscillator failures. For the description of specific operation of each oscillator, see the Watchdog Timer chapter on page 60, the On-Chip Oscillator chapter on page 232 and the Internal Precision Oscillator chapter on page 234.

System Clock Selection

The oscillator control block selects from the available clocks. Table 127 details each clock source and its usage.

Table 127. Oscillator Configuration and Selection

Clock Source	Characteristics	Required Setup
Internal Precision Oscillator	• 5.5296 MHz • High precision possible when trimmed • No external components required	• This is the reset default
External Crystal/ Resonator, External Clock Drive	• 0 MHz to 20 MHz • Very high accuracy (dependent on crystal/resonator or external source) • Requires external components	• Configure Option Bits for correct external oscillator mode • Unlock and write Oscillator Control Register (OSCCTL) to enable external oscillator • Wait for 50 ms stabilization time • Unlock and write Oscillator Control Register (OSCCTL) to select external oscillator
Internal Watchdog Timer Oscillator	• 10 kHz, nominal • Low accuracy • No external components required • Low power consumption	• Unlock and write Oscillator Control Register (OSCCTL) to enable and select Internal WDT oscillator

Unintentional accesses to the Oscillator Control Register (OSCCTL) can stop the chip by switching to a nonfunctioning oscillator. Accidental alteration of the OSCCTL Register is prevented by a locking/unlocking scheme. To write the register, unlock it by two writes to the OSCCTL Register with the values E7H followed by 18H. A third write to the OSCCTL Register then changes the value of the register and returns the register to a locked state. Any other sequence of oscillator control register writes has no effect. The values written to unlock the register must be ordered correctly, but need not be consecutive. It is possible to access other registers within the locking/unlocking operation.

Clock Selection Following System Reset

The Internal Precision Oscillator (IPO) is selected following a System Reset. Startup code after the System Reset may change the system clock source by unlocking and configuring the OSCCTL Register. If the LPDEN bit in Program Memory Address 0001H is 0, Flash Low Power Mode is enabled during reset. When Flash Low Power mode is enabled during reset, the FLPEN bit in the Oscillator Control Register (OSCCTL) is set and the DIV field of the OSCDIV Register is set to 08h.

Clock Failure Detection and Recovery for Primary Oscillator

The Z8FMC16100 Series MCU generates a System Exception, when a failure of the primary oscillator occurs, if the POFEN bit is set in the OSCCTL Register. To maintain system function in this situation, the clock failure recovery circuitry automatically forces

the Watchdog Timer (WDT) oscillator to drive the system clock. Although this oscillator runs at a much lower frequency 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 switch-over is not available if the WDT is the primary oscillator.

If the system clock frequency drops below 1 kHz ±50%, the primary oscillator failure detection circuitry asserts. For operating frequencies below 2 kHz, do not enable the clock failure circuitry (POFEN must be deasserted in the OSCCTL Register).

Clock Failure Detection and Recovery for WDT Oscillator

In the event of a WDT oscillator failure, a System Exception is issued if the WDFEN bit of the OSCCTL Register is set. This event does not trigger an attendant clock switch-over, but alerts the CPU of the failure. After a WDT failure, it is no longer possible to detect a primary oscillator failure.

The WDT oscillator failure detection circuit counts system clocks while looking for a WDT clock. The logic counts 8000 system clock cycles before determining that a failure occurred. The system clock rate determines the speed at which the WDT failure can be detected. A very slow system clock results in very slow detection times. If the WDT is the primary oscillator or if the Watchdog Timer oscillator is disabled, deassert the WDFEN bit of the OSCCTL Register.

Oscillator Control Register

The Oscillator Control Register (OSCCTL) enables/disables the various oscillator circuits, enables/disables the failure detection/recovery circuitry, actively powers down the flash and selects the primary oscillator, which becomes the system clock.

The Oscillator Control Register must be unlocked before writing. Writing the two-step sequence E7H followed by 18H to the Oscillator Control Register address unlocks it. The register locks after completion of a register write to the OSCCTL.

Table 128. Oscillator Control Register (OSCCTL)

Bit	7	6	5	4	3	2	1	0
Field	INTEN	XTLEN	WDTEN	POFEN	WDFEN	FLPEN	SCKSEL	
RESET	1	0	1	0	0	0*	00	
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Address	F86H							

Note: *The reset value is 1 if the option bit LPDEN is 0.

Bit	Description
[7] INTEN	**Internal Precision Oscillator Enable** 0 = Internal precision oscillator is disabled. 1 = Internal precision oscillator is enabled.
[6] XTLEN	**Crystal Oscillator Enable** 0 = Crystal oscillator is disabled. 1 = Crystal oscillator is enabled.
[5] WDTEN	**Watchdog Timer Oscillator Enable** 0 = Watchdog Timer oscillator is disabled. 1 = Watchdog Timer oscillator is enabled.
[4] POFEN	**Primary Oscillator Failure Detection Enable** 0 = Failure detection and recovery of primary oscillator is disabled. This bit is cleared automatically if a primary oscillator failure is detected. 1 = Failure detection and recovery of primary oscillator is enabled.
[3] WDFEN	**Watchdog Timer Oscillator Failure Detection Enable** 0 = Failure detection of Watchdog Timer oscillator is disabled.This bit is cleared automatically if a Watchdog Timer oscillator failure is detected. 1 = Failure detection of Watchdog Timer oscillator is enabled.
[2] FLPEN	**Flash Low Power Mode Enable** 0 = Flash Low Power Mode is disabled. 1 = Flash Low Power Mode is enabled. The Flash is powered down during idle periods of the clock and powered up during Flash reads. This bit should only be set if the frequency of the primary oscillator source is 8MHz or lower. The reset value of this bit is controlled by the LPDEN option bit during reset.
[1:0] SCKSEL	**System Clock Oscillator Select** 00 = Internal precision oscillator functions as system clock at 5.6MHz. 01 = Crystal oscillator or external clock driver functions as system clock . 10 = Reserved. 11 = Watchdog Timer oscillator functions as system clock.

Oscillator Divide Register

The Oscillator Divide Register (OSCDIV) provides the value that divides the system clock. The Oscillator Divide Register must be unlocked before writing. Writing the two-step sequence E7h, followed by 18h, to the Oscillator Control Register address unlocks the register. The register locks after completion of a register write to the OSCDIV.

Table 129. Oscillator Divide Register (OSCDIV)

Bit	7	6	5	4	3	2	1	0
Field	DIV							
RESET	00H*							
R/W	R/W							
Address	F87H							

Note: *The reset value is 08H if the option bit LPDEN is 0.

Bit	Description
[7:0] DIV	**Oscillator Divide** In the range 00H–FFH, the 00H address determines that the divider is disabled. All other entries represent divide values for scaling the system clock.

On-Chip Oscillator

The products in the Z8FMC16100 Series MCU features an on-chip oscillator for use with external crystals with frequencies ranging from 32 kHz to 20 MHz. In addition, the oscillator supports ceramic resonators with oscillation frequencies up to 20 MHz. This oscillator generates the primary system clock for the internal eZ8 CPU and the majority of the on-chip peripherals. Alternatively, the X_{IN} input pin can also accept a CMOS-level clock input signal (32 kHz to 20 MHz). If an external clock generator is used, the X_{OUT} pin must remain unconnected.

When configured for use with crystal oscillators or external clock drivers, the frequency of the signal on the X_{IN} input pin determines the frequency of the system clock (that is, no internal clock divider).

Crystal Oscillator Operation

Figure 40 displays a recommended configuration for connection with an external fundamental-mode, parallel-resonant crystal operating at 20 MHz. Recommended 20 MHz crystal specifications are provided in Table 130. The printed circuit board layout must add no more than 4 pF of stray capacitance to either the X_{IN} or X_{OUT} pins. If oscillation does not occur, reduce the values of capacitors C1 and C2 to decrease loading.

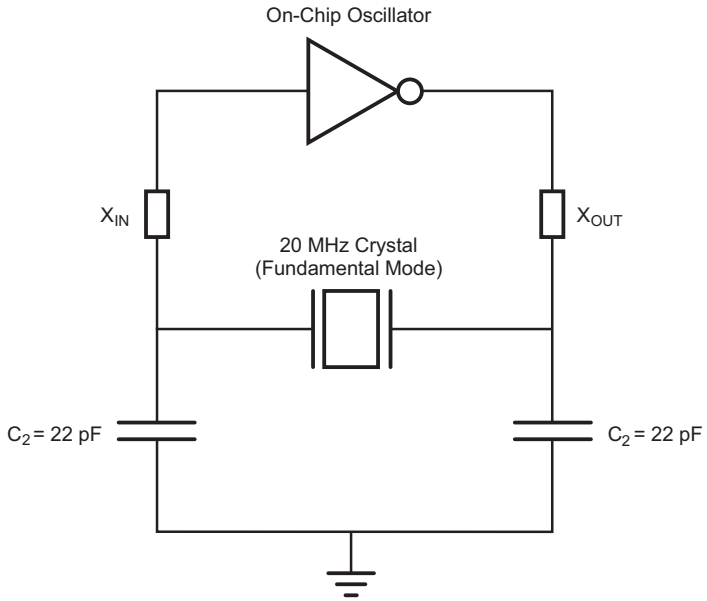


Figure 40. Recommended 20 MHz Crystal Oscillator Configuration

Table 130. Recommended Crystal Oscillator Specifications (20MHz Operation)

Parameter	Value	Units	Comments
Frequency	20	MHz	
Resonance	Parallel		
Mode	Fundamental		
Series Resistance (R_S)	25	Ohm	Maximum
Load Capacitance (C_L)	20	pF	Maximum
Shunt Capacitance (C_0)	7	pF	Maximum
Drive Level	1	mW	Minimum

Internal Precision Oscillator

The Internal Precision Oscillator (IPO) is designed for use without external components. The IPO comes factory trimmed a ±4% frequency accuracy over the operating temperature and supply voltage range of the device.

The features of the IPO include:

- On-chip RC oscillator that does not require external components
- Trimmed to ±4% accuracy
- Target output frequency of 5.5296 MHz
- Trimming possible through Flash option bits with user override
- IPO can eliminate crystals or ceramic resonators in applications where high timing accuracy is not required

Operation

The internal oscillator is an RC relaxation oscillator which has its sensitivity to power supply variation minimized. By using ratio tracking thresholds, the effect of power supply voltage is cancelled out. The dominant source of oscillator error is the absolute variance of chip-level fabricated components, such as capacitors. Two 8-bit trimming registers, incorporated into the design, allow compensation of absolute variation of oscillator frequency. Once calibrated, the oscillator frequency is relatively stable and does not require subsequent calibration.

By default, the oscillator is configured through the Flash Option bits. However, the user code can override these trim values as described in the Trim Option Bits section on page 222.

On-Chip Debugger

The Z8FMC16100 Series MCU device includes an integrated on-chip debugger (OCD) that provides advanced debugging features, including:

- Reading and writing of the Register File
- Reading and writing of program and data memory
- Setting of break points
- Execution of eZ8 CPU instructions

Architecture

The OCD consists of four primary functional blocks: Transmitter, Receiver, Autobaud Generator and Debug Controller. Figure 41 displays the architecture of the OCD.

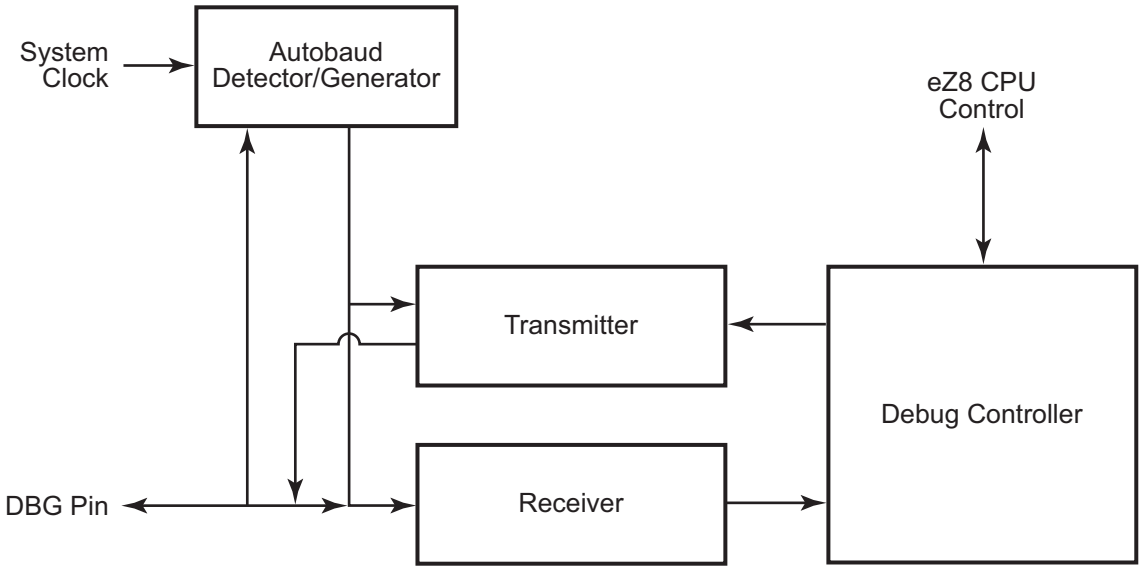


Figure 41. On-Chip Debugger Block Diagram

OCD Interface

The 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 Z8FMC16100 Series MCU device to the serial port of a host PC using minimal external hardware. Two different methods for connecting the DBG pin to an RS-232 interface are displayed in Figures 42 and 43.

Caution: For operation of the Z8FMC16100 Series MCU, power pins (V_{DD} and AV_{DD}) must be supplied with power and ground pins (V_{SS} and AV_{SS}) must be properly grounded. The DBG pin must always be connected to V_{DD} through an external pull-up resistor.

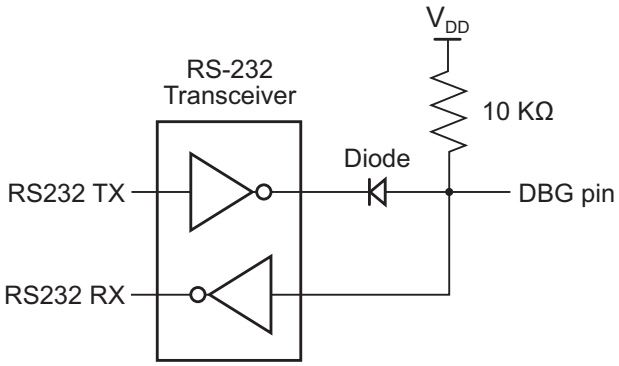


Figure 42. Interfacing the On-Chip Debugger's DBG Pin with an RS-232 Interface, #1 of 2

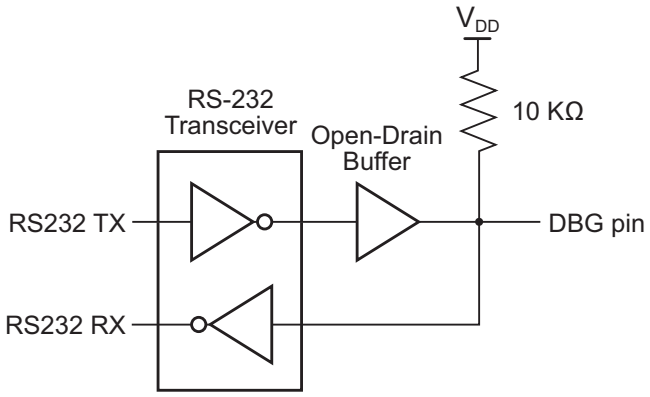


Figure 43. Interfacing the On-Chip Debugger's DBG Pin with an RS-232 Interface, #2 of 2

Debug Mode

The operating characteristics of the Z8FMC16100 Series MCU device in DEBUG Mode are:

- The eZ8 CPU fetch unit stops, idling the eZ8 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)

Entering DEBUG Mode

The device enters DEBUG Mode following any of the following operations:

- Writing the DBGMODE bit in the OCD Control Register to 1 using the OCD interface
- eZ8 CPU execution of a BRK (break point) instruction (when enabled)
- Match of PC to OCDCNTR Register (when enabled)
- OCDCNTR Register decrements to `0000h` (when enabled)
- The DBG pin is Low when the device exits Reset

Exiting DEBUG Mode

The device exits DEBUG Mode following any of the following operations:

- Clearing the DBGMODE bit in the OCD Control Register to 0
- Power-On Reset
- Voltage Brown-Out reset
- Asserting the $\overline{\text{RESET}}$ pin Low to initiate a Reset
- Driving the DBG pin Low while the device is in STOP Mode initiates a System Reset

OCD Data Format

The 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 44.

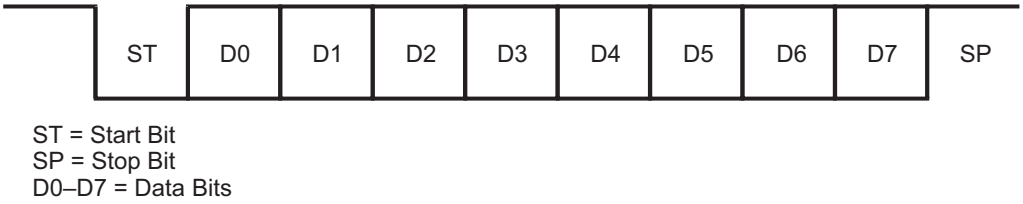


Figure 44. OCD Data Format

OCD Autobaud Detector/Generator

To run over a range of baud rates (bps) with various system clock frequencies, the OCD has an Autobaud 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. The character 80h contains eight continuous bits Low (one START bit plus 7 data bits). The Autobaud Detector measures this period and sets the OCD Baud Rate Generator accordingly.

The Autobaud Detector/Generator is clocked by the system clock. The minimum baud rate is the system clock frequency divided by 512. The maximum recommended baud rate is the system clock frequency divided by 8. Table 131 lists minimum and recommended maximum baud rates for sample crystal frequencies.

Table 131. OCD Baud-Rate Limits

System Clock Frequency	Maximum Asynchronous Baud Rate (bps)	Minimum Baud Rate (bps)
20.0 MHz	2.5 M	39.1 K
1.0 MHz	125 K	1960
32.768 kHz	4096	64

If the OCD receives a Serial Break (ten or more continuous bits Low), the Autobaud Detector/Generator resets. The Autobaud Detector/Generator can then be reconfigured by sending 80h. If the Autobaud Detector overflows while measuring the Autobaud character, the Autobaud Detector will remain reset.

OCD Serial Errors

The OCD detects any of the following error conditions on the DBG pin:

- Serial Break (a minimum of ten continuous bits Low)
- Framing Error (received STOP bit is Low)
- Transmit Collision (OCD and host simultaneous transmission detected by the OCD)

When the OCD detects one of these errors, it aborts any command currently in progress, transmits a Serial Break 4096 system clock cycles long back to the host and resets the Autobaud Detector/Generator. A Framing Error or Transmit Collision can be caused by the host sending a Serial Break to the OCD. Due to the open-drain nature of the interface, returning a Serial Break back to the host only extends the length of the Serial Break , if the host releases the Serial Break early.

The host transmits a Serial Break on the DBG pin when first connecting to the Z8FMC16100 Series MCU device or when recovering from an error. A Serial Break from the host resets the Autobaud Generator/Detector but does not reset the OCD Control Register. A Serial Break leaves the device in DEBUG Mode if that is the current mode. The OCD is held in Reset until the end of the Serial Break when the DBG pin returns High. Due to the open-drain nature of the DBG pin, the host can send a Serial Break to the OCD even if the OCD is transmitting a character.

Automatic Reset

The Z8FMC16100 Series MCU devices have the capability to switch clock sources during operation. If the Autobaud is set and the clock source is switched, the Autobaud value

becomes invalid. A new Autobaud value must be configured with the new clock frequency.

The oscillator control logic has clock switch detection. If a clock switch is detected and the Autobaud is set, the device automatically sends a Serial Break for 4096 clocks. This resets the Autobaud and indicate to the host that a new Autobaud character must be sent.

Break Points

Execution break points are generated using the BRK instruction (op code 00h). When the eZ8 CPU decodes a BRK instruction, it signals the OCD. If break points are enabled, the OCD idles the eZ8 CPU and enters DEBUG Mode. If break points are not enabled, the OCD ignores the BRK signal and the BRK instruction operates as a NOP instruction.

If break points are enabled, the OCD can be configured to automatically enter DEBUG Mode, or to loop on the break instruction. If the OCD is configured to loop on the BRK instruction, then the CPU remains able to service DMA and interrupt requests.

The loop on BRK instruction can service interrupts in the background. For interrupts to be serviced in the background, there cannot be any break points in the interrupt service routine. Otherwise, the CPU stops on the break point in the interrupt routine. For interrupts to be serviced in the background, interrupts must be enabled. Debugging software does not automatically enable interrupts when using this feature. Interrupts are typically disabled during critical sections of code where interrupts do not occur (such as adjusting the stack pointer or modifying shared data).

Host software can poll the IDLE bit of the OCDSTAT Register to determine if the OCD is looping on a BRK instruction. When software wants to stop the CPU on the BRK instruction on which it is looping, software must not set the DBGMODE bit of the OCDCTL Register. The CPU may have vectored to an interrupt service routine. Instead, software clears the BRKLP bit. This allows the CPU to finish the interrupt service routine it may be in and return to the BRK instruction. When the CPU returns to the BRK instruction on which it was previously looping, it automatically sets the DBGMODE bit and enters DEBUG Mode.

The majority of the OCD commands remain disabled when the eZ8 CPU is looping on a BRK instruction. The eZ8 CPU must be in DEBUG Mode before these commands are issued.

Break Points in Flash Memory

The BRK instruction is op code 00h, which corresponds to the fully programmed state of a byte in Flash Memory. To implement a break point, write 00h to the appropriate address, overwriting the current instruction. To remove a break point, erase the corresponding page of Flash Memory and reprogram with the original data.

OCDCNTR Register

The OCD contains a multipurpose 16-bit counter register which can be used for the following events:

- Count system clock cycles between breakpoints
- Generate a BRK when it counts down to 0
- Generate a BRK when its value matches the Program Counter

When configured as a counter, the OCDCNTR Register starts counting when the on-chip debugger leaves DEBUG Mode and stops counting when it enters DEBUG Mode again or when it reaches the maximum count of FFFFh. The OCDCNTR Register automatically resets itself to 0000h when the OCD exits DEBUG Mode, if it is configured to count clock cycles between break points.

If the OCDCNTR Register is configured to generate a BRK when it counts down to zero, it will not be reset when the CPU starts running. Once the OCD exits DEBUG Mode, the counter starts counting down toward zero. If the OCD enters DEBUG Mode before the OCDCNTR Register counts down to zero, the OCDCNTR stops counting.

If the OCDCNTR Register is configured to generate a BRK when the program counter matches the OCDCNTR Register, the OCDCNTR Register will not be reset when the CPU resumes executing and it will not be decremented when the CPU is running. A BRK is generated when the program counter matches the value in the OCDCNTR Register before executing the instruction at the location of the program counter.

Caution: The OCDCNTR Register is used by many OCD commands. It counts the number of bytes for the register and memory read/write commands. It retains the residual value when generating the CRC. If the OCDCNTR is used to generate a BRK, its value must be written as a final step before leaving DEBUG Mode.

As this register is overwritten by various OCD commands, it must only be used to generate temporary break points, such as stepping over CALL instructions or running to a specific instruction and stopping.

When the OCDCNTR Register is read, it returns the inverse of the data in this register. The OCDCNTR Register is only decremented when counting. The mode where it counts the number of clock cycles in between execution is achieved by counting down from its maximum count. When the OCDCNTR Register is read, the counter appears to be counted up as its value is inverted. The value in this register is always inverted when it is read. If this register is used as a hardware break point, the value read from this register will be inverse of the data actually in the register.

On-Chip Debugger Commands

The host communicates to the OCD by sending OCD commands using the DBG interface. During normal operation, only a subset of the OCD commands are available. In Debug mode, all OCD commands are available unless the user code is protected by programming the Read Protect option bit (RP). The Read Protect option bit prevents the code in memory from being read out of the Z8FMC16100 Series MCU device. When this option is enabled, several OCD commands are disabled. Table 132 lists a summary of the OCD commands. Each OCD command is described in detail in the list following Table 132. Table 132 indicates the commands that operate when the device is not in DEBUG Mode (normal operation) and those commands that are disabled by programming the Read Protect option bit.

Table 132. On-Chip Debugger Commands

Debug Command	Command Byte	Enabled When Not In Debug Mode?	Disabled by Read Protect Option Bit
Read Revision	00h	Yes	—
Write OCD Counter Register	01h	—	—
Read OCD Status Register	02h	Yes	—
Read OCD Counter Register	03h	—	—
Write OCD Control Register	04h	Yes	—
Read OCD Control Register	05h	Yes	—
Write Program Counter	06h	—	Disabled
Read Program Counter	07h	—	Disabled
Write Register	08h	—	Writes to on-chip peripheral registers are enabled. Writes to the on-chip RAM are disabled.
Read Register	09h	—	Reads from on-chip peripheral registers are enabled. Reads from the on-chip RAM are disabled.
Write Program Memory	0Ah	—	Disabled
Read Program Memory	0Bh	—	Disabled
Write Data Memory	0Ch	—	Disabled
Read Data Memory	0Dh	—	Disabled
Read Program Memory CRC	0Eh	—	—

Note: Unlisted command byte values are reserved.

Table 132. On-Chip Debugger Commands (Continued)

Debug Command	Command Byte	Enabled When Not In Debug Mode?	Disabled by Read Protect Option Bit
Reserved	0Fh	—	—
Step Instruction	10h	—	Disabled
Stuff Instruction	11H	—	Disabled
Execute Instruction	12H	—	Disabled
Read Baud Reload Register	1BH	—	—

Note: Unlisted command byte values are reserved.

In the following list of OCD Commands, data and commands sent from the host to the OCD are identified by 'DBG ← Command/Data'. Data sent from the OCD back to the host is identified by 'DBG → Data'

Read Revision. The Read Revision command returns the revision identifier.

```
DBG ← 00h
DBG → REVID[15:8] (Major revision number)
DBG → REVID[7:0]   (Minor revision number)
```

Write OCD Counter Register. The Write OCD Counter Register command writes the data that follows to the OCDCNTR Register. If the device is not in DEBUG Mode, the data is discarded.

```
DBG ← 01h
DBG ← OCDCNTR[15:8]
DBG ← OCDCNTR[7:0]
```

Read OCD Status Register. The Read OCD Status Register command reads the OCDSTAT Register.

```
DBG ← 02h
DBG → OCDSTAT[7:0]
```

Read OCD Counter Register. The OCD Counter Register can be used to count system clock cycles in between break points, generate a BRK when it counts down to 0, or generate a BRK when its value matches the Program Counter. As this register is a down counter, the returned value is inverted, when this register is read so the returned result appears to be an up counter. If the device is not in DEBUG Mode, this command returns FFFFh.

```
DBG ← 03h
DBG → ~OCDCNTR[15:8]
DBG → ~OCDCNTR[7:0]
```

Write OCD Control Register. The Write OCD Control Register command writes the data that follows to the OCDCTL Register.

```
DBG ← 04h
DBG ← OCDCTL[7:0]
```

Read OCD Control Register. The Read OCD Control Register command reads the value of the OCDCTL Register.

```
DBG ← 05h
DBG → OCDCTL[7:0]
```

Write Program Counter. The Write Program Counter command writes the data that follows to the eZ8 CPU's Program Counter (PC). If the device is not in DEBUG Mode or if the Read Protect option bit is enabled, the PC values are discarded.

```
DBG ← 06h
DBG ← ProgramCounter[15:8]
DBG ← ProgramCounter[7:0]
```

Read Program Counter. The Read Program Counter command reads the value in the eZ8 CPU's Program Counter. If the device is not in DEBUG Mode or if the Read Protect option bit is enabled, this command returns FFFFh.

```
DBG ← 07h
DBG → ProgramCounter[15:8]
DBG → ProgramCounter[7:0]
```

Write Register. The Write Register command writes data to the Register File. Data can be written 1–256 bytes at a time (256 bytes are written by setting size to zero). If the device is not in DEBUG Mode, the address and data values are discarded. If the Read Protect option bit is enabled, then only writes to the on-chip peripheral registers are allowed and all other register write data values are discarded.

```
DBG ← 08h
DBG ← {4'h0,Register Address[11:8]}
DBG ← Register Address[7:0]
DBG ← Size[7:0]
DBG ← 1-256 data bytes
```

Read Register. The Read Register command reads data from the Register File. Data can be read 1–256 bytes at a time (256 bytes can be read by setting size to zero). If the device is not in DEBUG Mode or if the Read Protect option bit is enabled and on-chip RAM is being read from, this command returns FFh for all the data values.

```
DBG ← 09h
DBG ← {4'h0,Register Address[11:8]
DBG ← Register Address[7:0]
```

```
DBG ← Size[7:0]
DBG → 1-256 data bytes
```

Write Program Memory. The Write Program Memory command writes data to Program Memory. This command is equivalent to the LDC and LDCI instructions. Data can be written 1–65536 bytes at a time (65536 bytes can be written by setting size to 0). The on-chip Flash controller must be written to and unlocked for the programming operation to occur. If the Flash controller is not unlocked, the data is discarded. If the device is not in DEBUG Mode or if the Read Protect option bit is enabled, the data is discarded.

```
DBG ← 0Ah
DBG ← Program Memory Address[15:8]
DBG ← Program Memory Address[7:0]
DBG ← Size[15:8]
DBG ← Size[7:0]
DBG ← 1-65536 data bytes
```

Read Program Memory. The Read Program Memory command reads data from Program Memory. This command is equivalent to the LDC and LDCI instructions. Data can be read 1 to 65536 bytes at a time (65536 bytes can be read by setting size to 0). If the device is not in DEBUG Mode or if the Read Protect option bit is enabled, this command returns FFh for the data.

```
DBG ← 0Bh
DBG ← Program Memory Address[15:8]
DBG ← Program Memory Address[7:0]
DBG ← Size[15:8]
DBG ← Size[7:0]
DBG → 1-65536 data bytes
```

Write Data Memory. The Write Data Memory command writes data to Data Memory. This command is equivalent to the LDE and LDEI instructions. Data can be written 1 to 65536 bytes at a time (65536 bytes can be written by setting size to 0). If the device is not in DEBUG Mode or if the Read Protect option bit is enabled, the data is discarded.

```
DBG ← 0Ch
DBG ← Data Memory Address[15:8]
DBG ← Data Memory Address[7:0]
DBG ← Size[15:8]
DBG ← Size[7:0]
DBG ← 1-65536 data bytes
```

Read Data Memory. The Read Data Memory command reads from Data Memory. This command is equivalent to the LDE and LDEI instructions. Data can be read 1 to 65536 bytes at a time (65536 bytes can be read by setting size to 0). If the device is not in DEBUG Mode, this command returns FFh for the data.

```
DBG ← 0Dh
DBG ← Data Memory Address[15:8]
DBG ← Data Memory Address[7:0]
DBG ← Size[15:8]
DBG ← Size[7:0]
DBG → 1-65536 data bytes
```

Read Program Memory CRC. The Read Program Memory CRC command computes and returns the CRC (cyclic redundancy check) of Program Memory using the 16-bit CRC-CCITT polynomial ($x^{16} + x^{12} + x^5 + 1$). The CRC is preset to all 1s. The least-significant bit of the data is shifted through the polynomial first. The CRC is inverted when it is transmitted. If the device is not in DEBUG Mode, this command returns FFFFh for the CRC value. Unlike most other OCD Read commands, there is a delay from issuing of the command until the OCD returns the data. The OCD reads the Program Memory, calculates the CRC value and returns the result. The delay is a function of the Program Memory size and is approximately equal to the system clock period multiplied by the number of bytes in the Program Memory.

```
DBG ← 0Eh
DBG → CRC[15:8]
DBG → CRC[7:0]
```

Step Instruction. The Step Instruction command steps one assembly instruction at the current Program Counter location. If the device is not in DEBUG Mode or the Read Protect option bit is enabled, the OCD ignores this command.

```
DBG ← 10h
```

Stuff Instruction. The Stuff Instruction command steps one assembly instruction and allows specification of the first byte of the instruction. The remaining 0 to 4 bytes of the instruction are read from Program Memory. This command is useful for stepping over instructions where the first byte of the instruction has been overwritten by a break point. If the device is not in DEBUG Mode or the Read Protect option bit is enabled, the OCD ignores this command.

```
DBG ← 11h
DBG ← op code[7:0]
```

Execute Instruction. The Execute Instruction command allows sending an entire instruction to be executed to the eZ8 CPU. This command can also step over break points. The number of bytes to send for the instruction depends on the op code. If the device is not in DEBUG Mode or the Read Protect option bit is enabled, the OCD ignores this command.

```
DBG ← 12h
DBG ← 1-5 byte op code
```

Read Baud Reload Register. The Read Baud Reload Register command returns the current value in the Baud Reload Register.

```
DBG ← 1Bh
DBG → BAUD[15:8]
DBG → BAUD[7:0]
```

OCD Control Register

The OCD Control Register, shown in Table 133, controls the state of the on-chip debugger. This register enters or exits DEBUG Mode and enables the BRK instruction. It can also reset the Z8FMC16100 Series MCU.

A *reset and stop* function can be achieved by writing 81H to this register. A *reset and go* function is achieved by writing 41H to this register. If the device is in DEBUG Mode, a *run* function is implemented by writing 40h to this register.

A more detailed description of each bit follows Table 133.

Table 133. OCD Control Register (OCDCTL)

Bit	7	6	5	4	3	2	1	0
Field	DBGMODE	BRKEN	DBGACK	BRKLOOP	BRKPC	BRKZRO	Reserved	RST
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W

Bit	Description
[7] DBGMODE	**Debug Mode** Setting this bit to 1 causes the device to enter DEBUG Mode. When in DEBUG Mode, the eZ8 CPU stops fetching new instructions. Clearing this bit causes the eZ8 CPU to resume execution. This bit is automatically set when a BRK instruction is decoded and Breakpoints are enabled. 0 = The device is running (operating in NORMAL Mode). 1 = The device is in DEBUG Mode.
[6] BRKEN	**Breakpoint Enable** This bit controls the behavior of the BRK instruction (op code 00H). By default, Breakpoints are disabled and the BRK instruction behaves like a NOP. If this bit is set to 1 and a BRK instruction is decoded, the OCD takes action dependent upon the BRKLOOP bit. 0 = BRK instruction is disabled. 1 = BRK instruction is enabled.
[5] DBGACK	**Debug Acknowledge** This bit enables the debug acknowledge feature. If this bit is set to 1, then the OCD sends a Debug Acknowledge character (FFH) to the host when a Breakpoint occurs. This bit automatically clears itself when an acknowledge character is sent. 0 = Debug Acknowledge is disabled. 1 = Debug Acknowledge is enabled.

Bit	Description (Continued)
[4] BRKLOOP	**Breakpoint Loop** This bit determines what action the OCD takes when a BRK instruction is decoded and breakpoints are enabled (BRKEN is 1). If this bit is 0, the DBGMODE bit is automatically set to 1 and the OCD enters DEBUG Mode. If BRKLOOP is set to 1, the eZ8 CPU loops on the BRK instruction. 0 = BRK instruction sets DBGMODE to 1. 1 = eZ8 CPU loops on BRK instruction.
[3] BRKPC	**Break When PC == OCDCNTR** If this bit is set to 1, then the OCDCNTR Register is used as a hardware breakpoint. When the program counter matches the value in the OCDCNTR Register, DBGMODE is automatically set to 1. If this bit is set, the OCDCNTR Register does not count when the CPU is running. 0 = OCDCNTR is setup as counter. 1 = OCDCNTR generates hardware break when PC == OCDCNTR.
[2] BRKZRO	**Break When OCDCNTR ==** 0000H If this bit is set, then the OCD automatically sets the DBGMODE bit when the OCDCNTR Register counts down to 0000H. If this bit is set, the OCDCNTR Register is not reset when the part leaves DEBUG Mode. 0 = OCD does not generate BRK when OCDCNTR decrements to 0000H. 1 = OCD sets DBGMODE to 1 when OCDCNTR decrements to 0000H.
[1]	**Reserved** This bit is reserved and must be programmed to 0.
[0] RST	**Reset** Setting this bit to 1 resets the device. The controller goes through a normal POR sequence with the exception that the OCD is not reset. This bit is automatically cleared to 0 when the reset finishes. 0 = No effect. 1 = Reset the device.

OCD Status Register

The OCD Status Register, shown in Table 134, reports status information about the current state of the debugger and the system.

Table 134. OCD Status Register (OCDSTAT)

Bit	7	6	5	4	3	2	1	0
Field	IDLE	HALT	RPEN	Reserved				
RESET	0	0	0	0				
R/W	R	R	R	R				

Bit	Description
[7] IDLE	**CPU Idle** This bit is set if the part is in DEBUG Mode (DBGMODE is 1) or if a BRK instruction has occurred since the last time OCDCTL was written. This can be used to determine if the CPU is running or if it is idle. 0 = The eZ8 CPU is running. 1 = The eZ8 CPU is either stopped or looping on a BRK instruction.
[6] HALT	**HALT Mode** 0 = The device is not in HALT Mode. 1 = The device is in HALT Mode.
[5] RPEN	**Read Protect Option Bit Enabled** 0 = The Read Protect option bit is disabled (Flash option bit is 1). 1 = The Read Protect option bit is enabled (Flash option bit is 0), disabling many OCD commands.
[4:0]	**Reserved** These bits are reserved and must be programmed to 00000.

Baud Reload Register

The Baud Reload Register, shown in Table 135, contains the measured autobaud value.

Table 135. Baud Reload Register

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	Reserved				RELOAD											
RESET	0H				000H											
R/W	R				R											

Bit	Description
[15:12]	**Reserved** These bits are reserved and must be programmed to 0000.
[11:0] RELOAD	**Baud Reload Value** This value is the measured autobaud value. This value is calculated using the following formula: $\text{RELOAD} = \frac{\text{SYSCLK}}{\text{BAUDRATE}} \times 8$

Electrical Characteristics

The electrical characteristics of the Z8FMC16100 Series MCUs are described in this chapter.

Absolute Maximum Ratings

The ratings listed in Table 136 are stress ratings only. Operation of the device at any condition outside as indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability. For improved reliability, unused inputs must be tied to one of the supply voltages (V_{DD} or V_{SS}).

Caution: Stresses greater than those listed in Table 136 may cause permanent damage to the device.

Table 136. Absolute Maximum Ratings

Parameter	Minimum	Maximum	Units
Ambient temperature under bias	–40	+105	ºC
Storage temperature	–65	+150	ºC
Voltage on PA3, PA4, PA5, PA6 and PA7 pins with respect to V_{SS}[1]	–0.3	+5.5	V
Voltage on all of Port B, PA0, PA1, PA2, PC0, $\overline{\text{RESET}}$, DBG, X_{IN}, V_{DD} and AV_{DD} pins with respect to V_{SS}	–0.3	+3.6	V
Maximum current on input and/or inactive output pin	–5	+5	µA
Maximum output current from digital active output pin	–25	+25	mA
Maximum output current from digital active output pin	–5	+5	mA
32-pin LQFP Package Maximum Ratings at –40ºC to 70ºC			
Total power dissipation		811	mW
Maximum current into V_{DD} or out of V_{SS}		225	mA
32-pin LQFP Package Maximum Ratings at 70ºC to 105ºC			
Total power dissipation		295	mW
Maximum current into V_{DD} or out of V_{SS}		82	mA

Note: *This condition excludes all pins that have on-chip pull-ups, when driven Low.

Table 136. Absolute Maximum Ratings (Continued)

Parameter	Minimum	Maximum	Units
32-pin QFN Package Maximum Ratings at –40ºC to 70ºC			
Total power dissipation		1580	mW
Maximum current into V_{DD} or out of V_{SS}		439	mA
32-pin QFN Package Maximum Ratings at 70ºC to 105ºC			
Total power dissipation		575	mW
Maximum current into V_{DD} or out of V_{SS}		160	mA

Note: *This condition excludes all pins that have on-chip pull-ups, when driven Low.

DC Characteristics

Table 137 lists the DC characteristics of the Z8FMC16100 Series MCU products. All voltages are referenced to V_{SS}, the primary system ground.

Table 137. DC Characteristics

		T_A = –40ºC to 105ºC				
Symbol	**Parameter**	**Minimum**	**Typical[1]**	**Maximum**	**Units**	**Conditions**
V_{DD}	Supply Voltage	2.7	—	3.6	V	
V_{IL1}	Low Level Input Voltage	–0.3	—	0.3*V_{DD}	V	For all input pins except $\overline{\text{RESET}}$, DBG and X_{IN}.
V_{IL2}	Low Level Input Voltage	–0.3	—	0.2*V_{DD}	V	For $\overline{\text{RESET}}$, DBG and X_{IN}.
V_{IH1}	High Level Input Voltage	0.7 x V_{DD}	—	5.5	V	PA3, PA4, PA5, PA6 and PA7 pins when their programmable pull-ups are disabled.
V_{IH2}	High Level Input Voltage	0.7 x V_{DD}	—	V_{DD}+0.3	V	PA3, PA4, PA5, PA6 and PA7 pins when their programmable pull-ups are enabled. All of Port B, PA0, PA1, PA2 and PC0 pins.
V_{IH3}	High Level Input Voltage	0.8 x V_{DD}	—	V_{DD}+0.3	V	$\overline{\text{RESET}}$, DBG and X_{IN} pins.
V_{OL1}	Low Level Output Voltage	—	—	0.4	V	I_{OL} = 2 mA; V_{DD} = 3.0V; High Output Drive disabled.

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.

Table 137. DC Characteristics (Continued)

		T_A = –40ºC to 105ºC				
Symbol	**Parameter**	**Minimum**	**Typical[1]**	**Maximum**	**Units**	**Conditions**
V_{OH1}	High Level Output Voltage	2.4	—	—	V	I_{OH} = –2 mA; V_{DD} = 3.0V; High Output Drive disabled.
V_{OL2}	Low Level Output Voltage High Drive	—	—	0.6	V	I_{OL} = 20 mA; V_{DD} = 3.3V; High Output Drive enabled; T_A = –40ºC to +70ºC.
V_{OH2}	High Level Output Voltage High Drive	2.4	—	—	V	I_{OH} = –20 mA; V_{DD} = 3.3V; High Output Drive enabled; T_A = –40ºC to +70ºC
V_{OL3}	Low Level Output Voltage High Drive	—	—	0.6	V	I_{OL} = 15 mA; V_{DD} = 3.3V; High Output Drive enabled; T_A = +70ºC to +105ºC.
V_{OH3}	High Level Output Voltage High Drive	2.4	—	—	V	I_{OH} = 15 mA; V_{DD} = 3.3V High Output Drive enabled; T_A = +70ºC to +105ºC.
V_{RAM}	RAM Data Retention	0.7	—	—	V	—
I_{IL}	Input Leakage Current	–5	—	+5	µA	V_{DD} = 3.3V; V_{IN} = V_{DD} or V_{SS}[2].
I_{TL}	Tri-State Leakage Current	–5	—	+5	µA	V_{DD} = 3.3V.
I_{PU1}	Weak Pull-up Current	9	20	50	µA	V_{DD} = 2.7–3.6V. T_A = 0ºC to +70ºC.

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.

Table 137. DC Characteristics (Continued)

		T_A = –40ºC to 105ºC				
Symbol	**Parameter**	**Minimum**	**Typical[1]**	**Maximum**	**Units**	**Conditions**
I_{PU2}	Weak Pull-up Current	7	20	75	µA	V_{DD} = 2.7–3.6V. T_A = –40ºC to +105ºC.
Idd ACTIVE	ACTIVE mode device current		22	38	mA	Max.: 20MHz CLK, max temp., max V_{DD} Typ.: 25ºC, V_{DD} nom., 20MHz
Idd HALT	HALT Mode device current		8	12	mA	Max.: 20MHz CLK, max temp., max V_{DD} Typ.: 25ºC, V_{DD} nom., 20MHz
Idd STOP 1	Chip leakage current in STOP Mode		2	15	µA	Max.: VBO disabled, WDT enabled, max V_{DD}, max temp. Typ.: 25ºC, V_{DD} nom.
Idd STOP 2	Chip leakage current in STOP Mode		<1	15	µA	Max.: VBO and WDT disabled, max V_{DD}, max temp. Typ.: 25ºC, V_{DD} nom.

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.

Figure 45 displays the active mode current consumption while operating at 25ºC and 3.3V, plotted opposite the system clock frequency. All GPIO pins are configured as outputs and driven High.

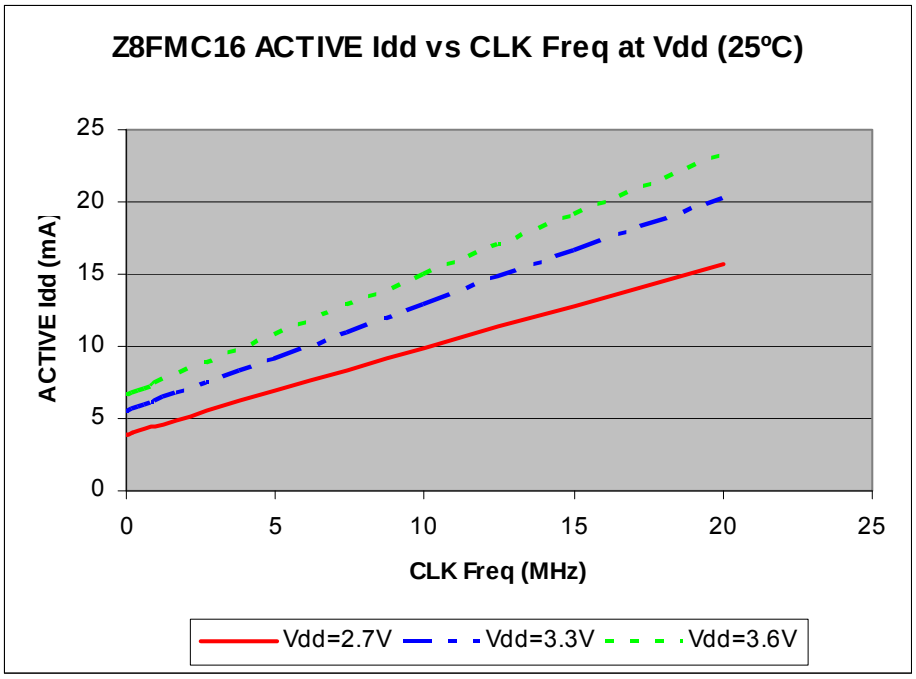


Figure 45. ACTIVE Mode Current Consumption as a Function of Clock Frequency with V_{DD} as Parameter at 25ºC

Figure 46 displays the current consumption in HALT Mode while operating at 25ºC plotted opposite the system clock frequency. All GPIO pins are configured as outputs and driven High.

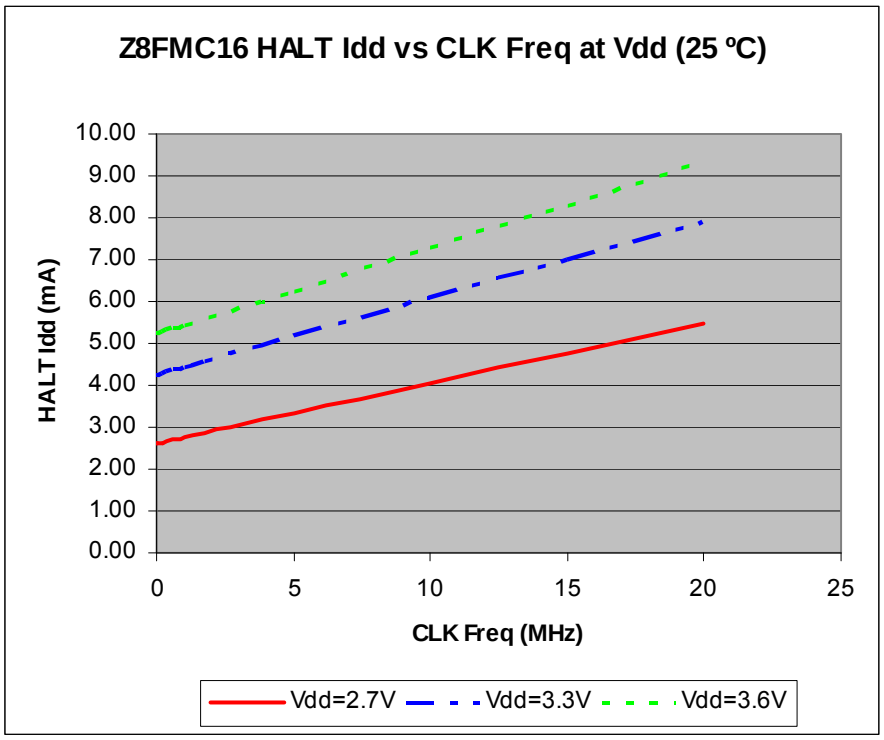


Figure 46. HALT Mode Current Consumption as a Function of Clock Frequency with V_{DD} as a Parameter at 25ºC

Figure 47 displays the STOP Mode supply current plotted opposite the ambient temperature and V_{DD} level, with respect to all of the peripherals being disabled. All GPIO pins are configured as outputs and driven High.

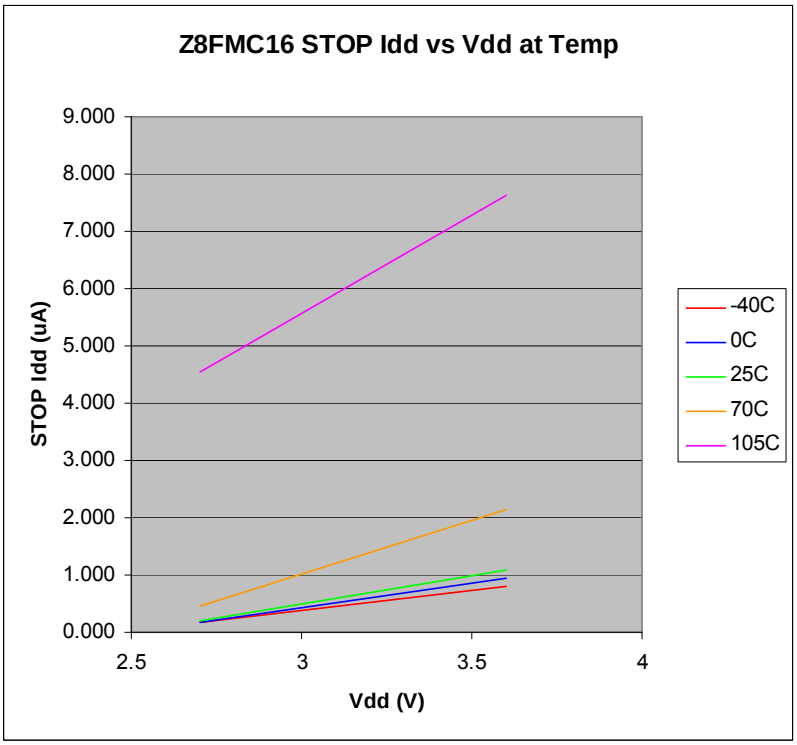


Figure 47. STOP Mode Current Consumption as a Function of Temperature with V_{DD} as a Parameter, All Peripherals Disabled

AC Characteristics

The section provides information about the AC characteristics and timing. All AC timing information assumes a standard load of 50pF on all outputs. Data in the typical column is from characterization at 3.3V and 25ºC. These values are provided for design guidance only and are not tested in production.

Table 138. AC Characteristics

		V_{DD} = 2.7–3.6V T_A = –40ºC to 105ºC			
Symbol	**Parameter**	**Minimum**	**Maximum**	**Units**	**Conditions**
F_{SYSCLK}	System clock frequency	—	20.0	MHz	
F_{XTAL}	Crystal oscillator frequency	0.032768	20.0	MHz	System clock frequencies below the crystal oscillator minimum require an external clock driver.
T_{XIN}	System clock period	50	—	ns	$T_{CLK} = 1 \div F_{SYSCLK}$.
T_{XINH}	System clock high time	20	30	ns	T_{CLK} = 50 ns.
T_{XINL}	System clock low time	20	30	ns	T_{CLK} = 50 ns.

On-Chip Peripheral AC and DC Electrical Characteristics

Table 139 provides electrical characteristics and timing information for the POR and VBO circuits.

Table 139. POR and VBO Electrical Characteristics and Timing

		T_A = –40ºC to 105ºC				
Symbol	**Parameter**	**Minimum**	**Typical***	**Maximum**	**Units**	**Conditions**
V_{POR}	Power-On Reset voltage threshold	2.20	2.45	2.70	V	$V_{DD} = V_{POR}$.
V_{VBO}	Voltage Brown-Out reset voltage threshold	2.15	2.40	2.65	V	$V_{DD} = V_{VBO}$.
	$V_{POR}–V_{VBO}$		50	100	mV	
	Starting V_{DD} voltage to ensure valid Power-On Reset	—	V_{SS}	—	V	
T_{ANA}	Power-On Reset analog delay	—	50	—	ms	$V_{DD} > V_{POR}$; T_{POR} Digital Reset delay follows T_{ANA}.
T_{POR}	Power-On Reset digital delay	—	5.0	—	ms	50 WDT Oscillator cycles (10 kHz) + 16 System Clock cycles (20 MHz).
T_{VBO}	Voltage Brown-Out pulse rejection period	—	10	—	ms	$V_{DD} < V_{VBO}$ to generate a Reset.
T_{RAMP}	Time for V_{DD} to transition from V_{SS} to V_{POR} to ensure valid Reset	0.10	—	100	ms	
I_{CC}	Supply current		500		µA	V_{DD} = 3.3 V.

Note: *Data in the Typical column is based on characterization at 3.3 V and 25ºC. These values are provided for design guidance only and are not tested in production.

Table 140 provides electrical characteristics and timing information for the External RC Oscillator.

Table 140. External RC Oscillator Electrical Characteristics and Timing

		T_A = –40ºC to 105ºC				
Symbol	**Parameter**	**Minimum**	**Typical***	**Maximum**	**Units**	**Conditions**
V_{DD}	Operating voltage range	2.70	—	—	V	
R_{EXT}	External resistance from X_{IN} to V_{DD}	40	45	200	KΩ	
C_{EXT}	External capacitance from X_{IN} to V_{SS}	0	20	1000	pF	
F_{OSC}	External RC oscillation frequency	—	—	4	MHz	

Note: *When using the external RC oscillator mode, the oscillator may stop oscillating if the power supply drops below 2.7V, but before the power supply drops to the Voltage Brown-Out threshold. The oscillator resumes oscillation as soon as the supply voltage exceeds 2.7V.

Table 141 provides electrical characteristics and timing information for the Internal Precision Oscillator.

Table 141. Internal Precision Oscillator Electrical Characteristics and Timing

		T_A = –40ºC to 105ºC				
Symbol	**Parameter**	**Minimum**	**Typical**	**Maximum**	**Units**	**Conditions**
F_{Of}	Output frequency*	5.308	5.5296	5.75	MHz	4% limits
		5.391	5.5296	5.668	MHz	Tighter limits are 2.5%; these are applicable at 3.3V and 0ºC-70ºC and are based on characterization data. These are provided for design guidance ONLY. These limits ARE NOT tested in production.
I_{CC}	Supply current		1.6		mA	V_{DD} = 3.3V.

Note: *The frequency is factory programmed.

Table 142 provides electrical characteristics and timing information for the Watchdog Timer.

Table 142. Watchdog Timer Electrical Characteristics and Timing

		V_{DD} = 2.7–3.6V T_A = –40ºC to 105ºC				
Symbol	**Parameter**	**Minimum**	**Typical**	**Maximum**	**Units**	**Conditions**
F_{OSC}	Output frequency	5	10	20	kHz	V_{DD} = 3.3V.
I_{CC}	Supply current		2		µA	V_{DD} = 3.3V.

Table 143 provides electrical characteristics and timing information for the Reset and Stop Mode Recovery functions.

Table 143. Reset and Stop Mode Recovery Pin Timing

		T_A = –40ºC to 105ºC				
Symbol	**Parameter**	**Minimum**	**Typical**	**Maximum**	**Units**	**Conditions**
T_{RESET}	$\overline{\text{RESET}}$ pin assertion to initiate a System Reset.	4	—	—	T_{CLK}	Not in STOP Mode. T_{CLK} = System Clock period.
T_{SMR}	Stop Mode Recovery pin pulse rejection period	10	20	40	ns	RESET, DBG and GPIO pins configured as SMR sources.

Table 144 lists the Flash Memory electrical characteristics and timing.

Table 144. Flash Memory Electrical Characteristics and Timing

	V_{DD} = 2.7 to 3.6V T_A = –40°C to 105°C				
Parameter	**Minimum**	**Typical**	**Maximum**	**Units**	**Notes**
Flash Byte Read Time	50	—	—	ns	
Flash Byte Program Time	20	—	40	µs	
Flash Page Erase Time	10	—	—	ms	
Flash Mass Erase Time	200	—	—	ms	
Writes to Single Address Before Next Erase	—	—	2		

Note: *This parameter is only an issue when bypassing the Flash Controller.

Table 144. Flash Memory Electrical Characteristics and Timing (Continued)

	V_{DD} = 2.7 to 3.6V T_A = –40°C to 105°C				
Parameter	**Minimum**	**Typical**	**Maximum**	**Units**	**Notes**
Flash Row Program Time	—	—	8	ms	Cumulative program time for single row cannot exceed limit before next erase*
Data Retention	100	—	—	years	25°C
Endurance	10,000	—	—	cycles	Program/erase cycles

Note: *This parameter is only an issue when bypassing the Flash Controller.

Table 145 provides electrical characteristics and timing information for the ADC and illustrates the input frequency response of the ADC.

Table 145. Analog-to-Digital Converter Electrical Characteristics and Timing

		T_A = –40ºC to 105ºC				
Symbol	**Parameter**	**Minimum**	**Typical**	**Maximum**	**Units**	**Conditions**
	Resolution	10	—		bits	External V_{REF} = 2.0 V.
	Throughput conversion	13			CLKs	ADC clock cycles.
	ADCCLK frequency			20	MHz	
DNL	Differential nonlinearity for 8-bit use	–0.99		1.25	LSB	20 MHz sys clock with ADC clock divided by 4
INL	Integral nonlinearity for 8-bit use	–1.25		1.25	LSB	20 MHz sys clock with ADC clock divided by 4
DNL	Differential nonlinearity for 10-bit		2		LSB	20 MHz sys clock with ADC clock divided by 4
INL	Integral nonlinearity for 10-bit		2		LSB	20 MHz sys clock with ADC clock divided by 4
	Offset error	–30		30	mV	
	Gain error	–25		25	LSB	
V_{REF}	On-chip voltage reference	1.9	2	2.1	V	
	Analog input voltage range	0		V_{REF}	V	
	Analog input current			500	nA	
	Reference input current		2.0		mA	Worst-case code

Table 145. Analog-to-Digital Converter Electrical Characteristics and Timing (Continued)

		T_A = –40ºC to 105ºC				
Symbol	**Parameter**	**Minimum**	**Typical**	**Maximum**	**Units**	**Conditions**
V_{REF}	External V_{REF} voltage			AVDD -1.0V	V	
	Analog input capacitance			15	pF	
AV_{DD}	Operation supply voltage	2.7		3.6	V	
	Operating current, AV_{DD}		9.0		mA	At 20MHz ADC clock
	Power-down current		< 1		µA	

Table 146 provides electrical characteristics and timing information for the on-chip comparator.

Table 146. Comparator Electrical Characteristics

		= 2.7–3.6V T_A = –40ºC to 105ºC				
Symbol	**Parameter**	**Minimum**	**Typical**	**Maximum**	**Units**	**Conditions**
V_{COFF}	Input offset	—	5	15	mV	V_{DD} = 3.3V; V_{IN} = V_{DD} ÷ 2.
T_{CPROP}	Propagation delay	—	200		ns	V_{COMM} mode =1V V_{DIFF}=100 mV
I_B	Input bias current			1	µA	
CMVR	Common-mode voltage range	–0.3		V_{DD}–1	V	
I_{CC}	Supply current		55		µA	V_{DD} = 3.6V, 25ºC
T_{wup}	Wake up time from off state			5	µs	CINP = 0.9V CINN= 1.0V

Table 147 provides electrical characteristics and timing information for the on-chip Operational Amplifier.

Table 147. Operational Amplifier Electrical Characteristics

		= 2.7–3.6V T_A = –40ºC to 105ºC				
Symbol	**Parameter**	**Minimum**	**Typical**	**Maximum**	**Units**	**Conditions**
V_{OS}	Input offset		5	15	mV	V_{DD} =3.3 V; V_{CM} = V_{DD} ÷ 2.
TC_{VOS}	Input offset average drift		1		µV/C	
I_B	Input bias current		TBD		uA	
I_{OS}	Input offset current		TBD		uA	
CMVR	Common Mode Voltage Range	–0.3		V_{DD} – 1	V	
V_{OL}	Output Low			0.1	V	I_{SINK} = 100 µA.
V_{OH}	Output High	V_{DD} – 1			V	I_{SOURCE} = 100 µA.
CMRR	Common Mode Rejection Ratio		70		dB	0 < V_{CM} < 1.4 V; T_A = 25ºC.
PSRR	Power Supply Rejection Ratio		80		dB	V_{DD} = 2.7 V – 3.6 V; T_A = 25ºC.
A_{VOL}	Voltage gain		80		dB	
SR+	Slew rate while rising		12		V/us	R_{LOAD} = 33 K; C_{LOAD} = 50 pF; A_{VCL} = 1, V_{IN} = 0.7 V to 1.7 V.
SR–	Slew rate while falling		16		V/us	R_{LOAD} = 33 K; C_{LOAD} = 50 pF; A_{VCL} = 1, V_{IN} = 1.7 V to 0.7 V.
GBW	Gain-Bandwidth Product	5			MHz	
FM	Phase Margin		50		degree	
I_S	Supply Current			1	mA	V_{DD} = 3.6 V; V_{OUT} = V_{DD} ÷ 2.
T_{WUP}	Wake-up time from off state			20	us	

General-Purpose Input/Output Port Input Data Sample Timing

Figure 48 displays the timing of the GPIO Port input sampling. Table 148 lists the GPIO port input timing.

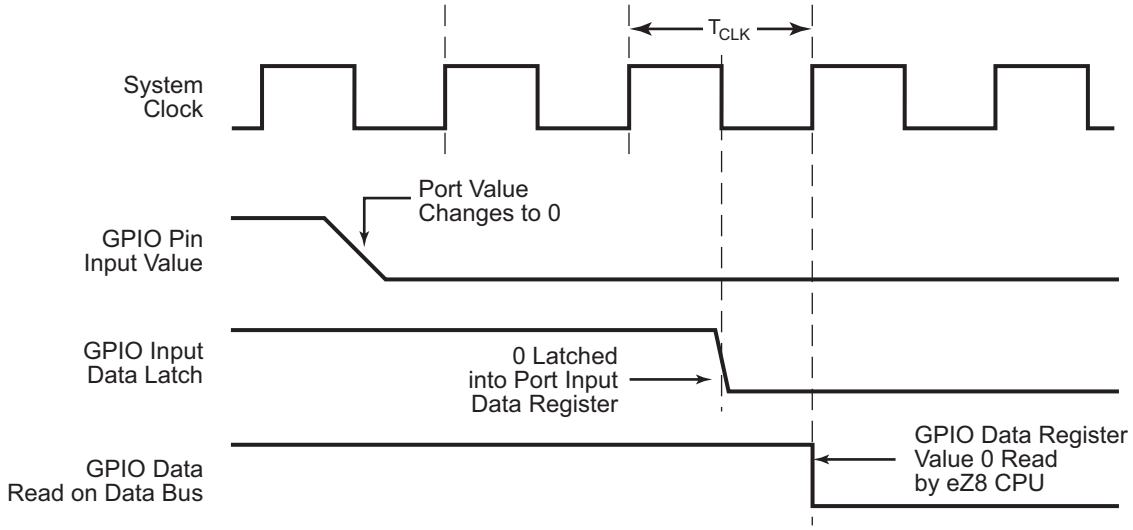


Figure 48. Port Input Sample Timing

Table 148 and Table 149 provide timing information for the GPIO Port inputs and outputs.

Table 148. GPIO Port Input Timing

		Delay (ns)	
Parameter	**Abbreviation**	**Min**	**Max**
T_{S_PORT}	Port input transition to X_{IN} fall setup time (not pictured)	5	—
T_{H_PORT}	X_{IN} fall to port input transition hold time (not pictured).	5	—
T_{SMR}	GPIO port pin pulse width to ensure Stop Mode Recovery (for GPIO port pins enabled as SMR sources).	1 µs	

General-Purpose Input/Output Port Output Timing

Figure 49 and Table 149 provide timing information for the GPIO port pins.

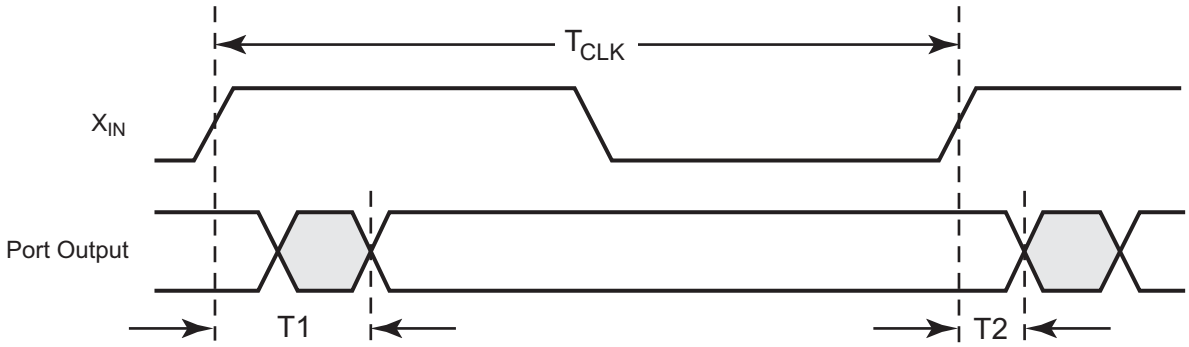


Figure 49. GPIO Port Output Timing

Table 149. GPIO Port Output Timing

		Delay (ns)	
Parameter	**Abbreviation**	**Minimum**	**Maximum**
T_1	X_{IN} Rise to Port Output Valid Delay	—	15
T_2	X_{IN} Rise to Port Output Hold Time	2	—

On-Chip Debugger Timing

Figure 50 and Table 150 provide timing information for the DBG pin. The DBG pin timing specifications assume a 4 µs maximum rise and fall time.

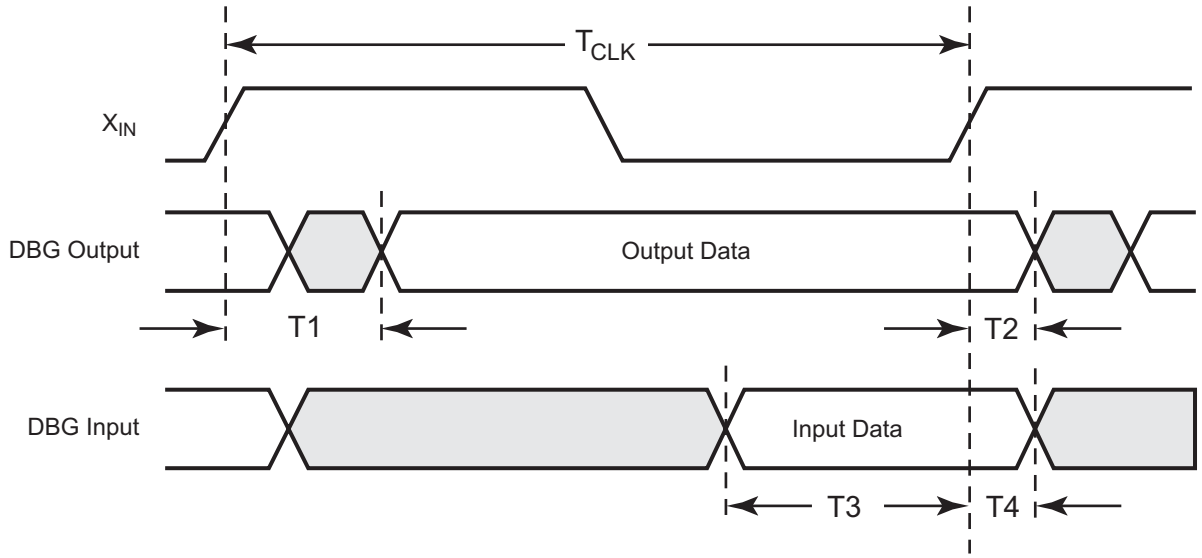


Figure 50. On-Chip Debugger Timing

Table 150. On-Chip Debugger Timing

		Delay (ns)	
Parameter	**Abbreviation**	**Minimum**	**Maximum**
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	10	—
T_4	DBG to X_{IN} Rise Input Hold Time	5	—
	DBG Frequency		System Clock ÷ 4

UART Timing

Figure 51 and Table 151 provide timing information for UART pins for the case where the Clear To Send input pin ($\overline{CTS}$) is used for flow control. In this example, it is assumed that the Driver Enable polarity is configured to be Active Low and is represented here by $\overline{DE}$. The $\overline{CTS}$ to $\overline{DE}$ assertion delay (T1) assumes the UART Transmit Data Register is loaded with data prior to $\overline{CTS}$ assertion.

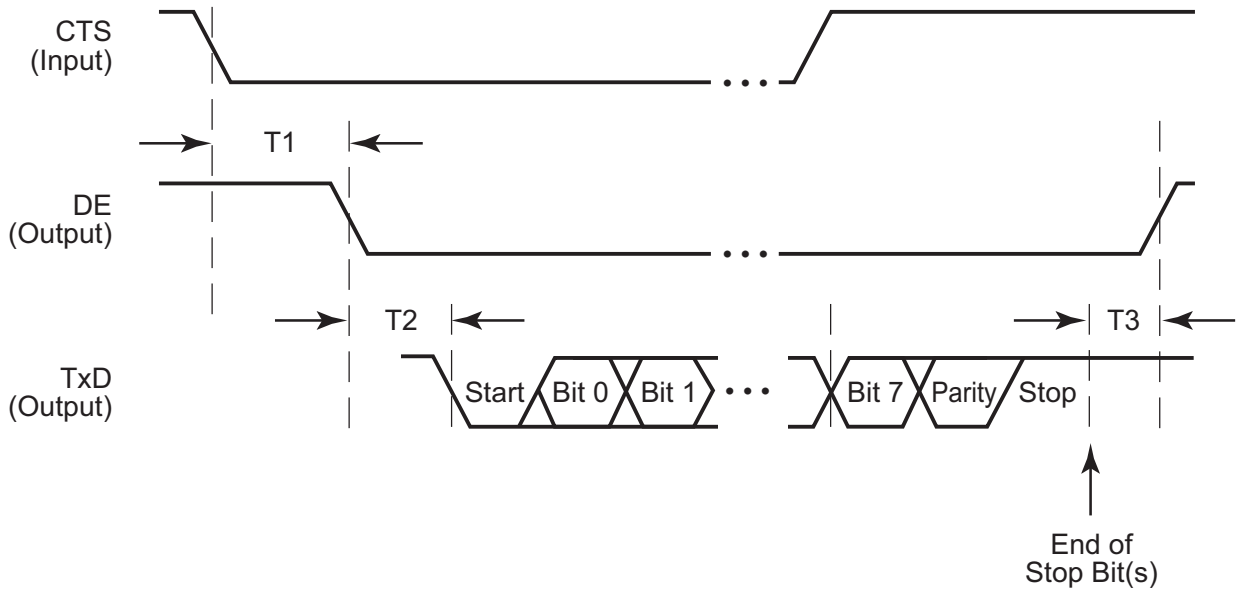


Figure 51. UART Timing with $\overline{CTS}$

Table 151. UART Timing with $\overline{CTS}$

		Delay (ns)	
Parameter	**Abbreviation**	**Minimum**	**Maximum**
T_1	$\overline{CTS}$ fall to $\overline{DE}$ assertion delay	2 x X_{IN} period	2 x X_{IN} period + 1 bit period
T_2	$\overline{DE}$ assertion to TxD falling edge (start) delay	1 bit period	1 bit period + 1 x X_{IN} period
T_3	End of stop bit (s) to $\overline{DE}$ deassertion delay	1 x X_{IN} period	2 x X_{IN} period

Figure 52 and Table 152 provide timing information for UART pins for the case where the Clear To Send input signal ($\overline{CTS}$) is not used for flow control. In this example, it is assumed that the Driver Enable polarity is configured to be Active Low and is represented here by $\overline{DE}$. $\overline{DE}$ asserts after the UART Transmit Data Register is written. $\overline{DE}$ remains asserted for multiple characters as long as the Transmit Data Register is written with the next character before the current character has completed.

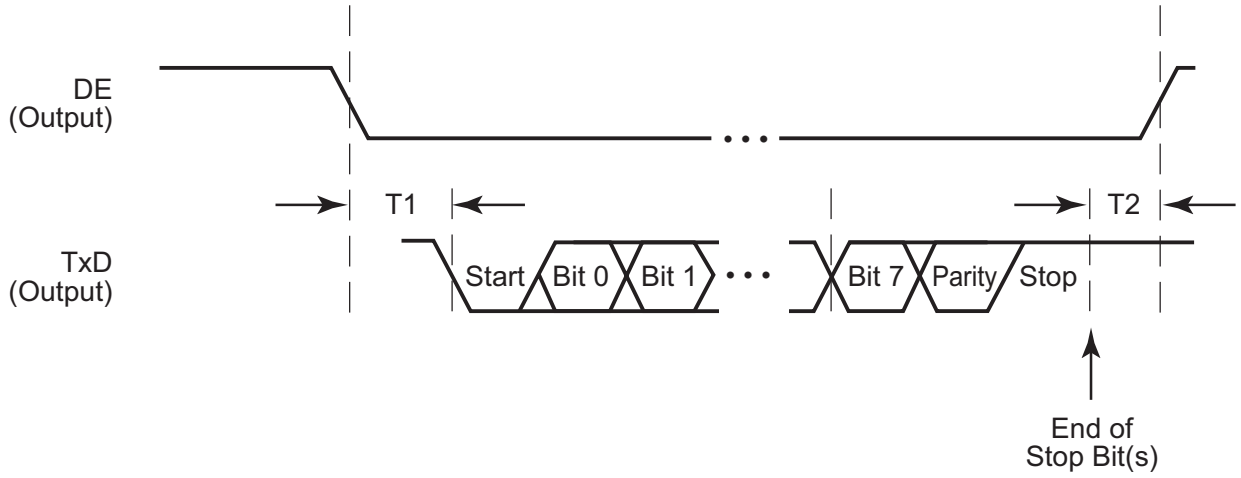


Figure 52. UART Timing without $\overline{CTS}$

Table 152. UART Timing without $\overline{CTS}$

		Delay (ns)	
Parameter	**Abbreviation**	**Minimum**	**Maximum**
T_1	$\overline{DE}$ assertion to TxD falling edge (start) delay	1 bit period	1 bit period + 1 x X_{IN} period
T_2	End of stop bit (s) to $\overline{DE}$ deassertion delay	1 x X_{IN} period	2 x X_{IN} period

eZ8 CPU Instruction Set

This chapter describes how to use the eZ8 CPU.

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 can contain labels, operations, operands and comments.

Labels can be 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, the pseudo-ops are interpreted as directives that control or assist the assembly process. The source program is processed (assembled) by the assembler to obtain a machine language program called the object code. The object code is executed by the eZ8 CPU. An example segment of an assembly language program is detailed in the code below:

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.
```

```
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.
```

Assembly Language Syntax

For proper instruction execution, eZ8 CPU assembly language syntax requires that the operands be written as ‘destination, source’. After assembly, the object code usually has the operands in the order ’source, destination’, but ordering is op code-dependent. The following instruction examples illustrate the format of some basic assembly instructions and the resulting object code produced by the assembler. This binary format must be followed for manual program coding or for those who intend to implement their 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:

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 in working registers 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:

Assembly Language Code	ADD	43H,	R8	(ADD dst, src)
Object Code	04	E8	43	(OPC src, dst)

Note: The size of the register file varies depending upon device type. The register file is 512 bytes for the Z8FMC16100 Series MCU.

eZ8 CPU Instruction Notation

In the eZ8 CPU Instruction Summary and Description sections, the operands, condition codes, status flags and address modes are represented by a notational shorthand described in Table 153.

Table 153. Notational Shorthand

Notation	Description	Operand	Range
b	Bit	b	b represents a value from 0 to 7 (000b to 111b).
cc	Condition Code	—	See Condition Codes overview in the refer to 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 the+127 to –128 range.

Table 154 contains additional symbols that are used throughout the Instruction Summary and Instruction Set Description sections.

Table 154. 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,

dst ← dst + src

The above code segment indicates that source data is added to the destination data, and that the result is stored in the destination location.

Condition Codes

The C, Z, S and V flags control the operation of the conditional jump (JP cc and JR cc) instructions. Sixteen frequently-useful functions of the flag settings are encoded in a 4-bit field called the condition code (cc), which forms Bits 7:4 of the conditional jump instructions. Table 155 lists the condition codes. Some binary condition codes are created using more than one assembly code mnemonic. The result of the flag test operation decides whether the conditional jump is executed.

Table 155. Condition Codes

Binary	Hex	Assembly Mnemonic	Definition	Flag Test Operation
0000	0	F	Always False	—
0001	1	LT	Less Than	(S XOR V) = 1
0010	2	LE	Less Than or Equal	(Z or (S XOR V)) = 1
0011	3	ULE	Unsigned Less Than or Equal	(C OR Z) = 1
0100	4	OV	Overflow	V = 1
0101	5	MI	Minus	S = 1
0110	6	Z	Zero	Z = 1
0110	6	EQ	Equal	Z = 1
0111	7	C	Carry	C = 1
0111	7	ULT	Unsigned Less Than	C = 1
1000	8	T (or blank)	Always True	—
1001	9	GE	Greater Than or Equal	(S XOR V) = 0
1010	A	GT	Greater Than	(Z OR (S XOR V)) = 0
1011	B	UGT	Unsigned Greater Than	(C = 0 AND Z = 0) = 1
1100	C	NOV	No Overflow	V = 0
1101	D	PL	Plus	S = 0
1110	E	NZ	Non-Zero	Z = 0
1110	E	NE	Not Equal	Z = 0
1111	F	NC	No Carry	C = 0
1111	F	UGE	Unsigned Greater Than or Equal	C = 0

eZ8 CPU Instruction Classes

The 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 156 through 163 contain the instructions belonging to each group and the number of operands required for each instruction. Some instructions appear in more than one table as these instruction are 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 a condition code is *cc*.

Table 156. 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 157. 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

Table 157. Bit Manipulation Instructions (Continued)

Mnemonic	Operands	Instruction
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 158. 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 159. CPU Control Instructions

Mnemonic	Operands	Instruction
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
STOP	—	STOP Mode
WDT	—	Watchdog Timer Refresh

Table 160. 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 161. 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 162. Program Control Instructions

Mnemonic	Operands	Instruction
ATM	—	Atomic
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 163. 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

eZ8 CPU Instruction Summary

Table 164 lists 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 164. eZ8 CPU Instruction Summary

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
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

Notes: Flags Notation symbols legend:
* = Value is a function of the result of the operation.
– = Unaffected.
X = Undefined.
0 = Reset to 0.
1 = Set to 1.

Table 164. 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 + 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	Atomic execution			2F	—	—	—	—	—	—	1	1
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
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

Notes: Flags Notation symbols legend:
* = Value is a function of the result of the operation.
– = Unaffected.
X = Undefined.
0 = Reset to 0.
1 = Set to 1.

Table 164. 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
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	—	*	*	*	v	—	2	2
		IR		31							2	3
DECW dst	dst ← dst – 1	RR		80	—	*	*	*	—	—	2	5
		IRR		81							2	6
DI	IRQE ← 0			8F	—	—	—	—	—	—	1	2
DJNZ dst, RA	dst ← dst – 1 if dst ≠ 0 PC ← PC + X	r		0A–FA	—	—	—	—	—	—	2	3
EI	IRQE ← 1			9F	—	—	—	—	—	—	1	2
HALT	HALT Mode			7F	—	—	—	—	—	—	1	2

Notes: Flags Notation symbols legend:
* = Value is a function of the result of the operation.
– = Unaffected.
X = Undefined.
0 = Reset to 0.
1 = Set to 1.

Table 164. 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
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 IRQE ← 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
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

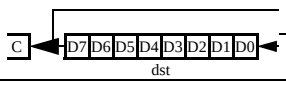
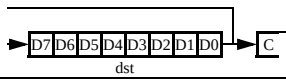
Notes: Flags Notation symbols legend:
* = Value is a function of the result of the operation.
– = Unaffected.
X = Undefined.
0 = Reset to 0.
1 = Set to 1.

Table 164. 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
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	1F E8	—	—	—	—	—	—	5	4
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

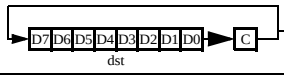
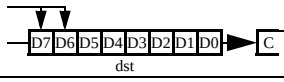
Notes: Flags Notation symbols legend:
* = Value is a function of the result of the operation.
– = Unaffected.
X = Undefined.
0 = Reset to 0.
1 = Set to 1.

Table 164. 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
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
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		1F 70							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

Notes: Flags Notation symbols legend:
* = Value is a function of the result of the operation.
– = Unaffected.
X = Undefined.
0 = Reset to 0.
1 = Set to 1.

Table 164. 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
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
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

Notes: Flags Notation symbols legend:
* = Value is a function of the result of the operation.
– = Unaffected.
X = Undefined.
0 = Reset to 0.
1 = Set to 1.

Table 164. 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
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
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		Vec-tor	F2	—	—	—	—	—	—	2	6
WDT				5F	—	—	—	—	—	—	1	2

Notes: Flags Notation symbols legend:
* = Value is a function of the result of the operation.
– = Unaffected.
X = Undefined.
0 = Reset to 0.
1 = Set to 1.

Table 164. 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
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

Notes: Flags Notation symbols legend:
* = Value is a function of the result of the operation.
– = Unaffected.
X = Undefined.
0 = Reset to 0.
1 = Set to 1.

Flags Register

The Flags Register contains the status information regarding the most recent arithmetic, logical, bit manipulation, or rotate and shift operation. The Flags Register contains six bits of status information that are set or cleared by CPU operations. Four of the bits (C, V, Z and S) are tested for use with conditional jump instructions. Two flags (H and D) cannot be tested and are used for Binary-Coded Decimal (BCD) arithmetic.

The two remaining bits, User Flags (F1 and F2), are available as general-purpose status bits. User Flags are unaffected by arithmetic operations and must be set or cleared by instructions. The User Flags cannot be used with conditional Jumps. They are undefined at initial power-up and are unaffected by Reset. Figure 53 displays the flags and their bit positions in the Flags Register.

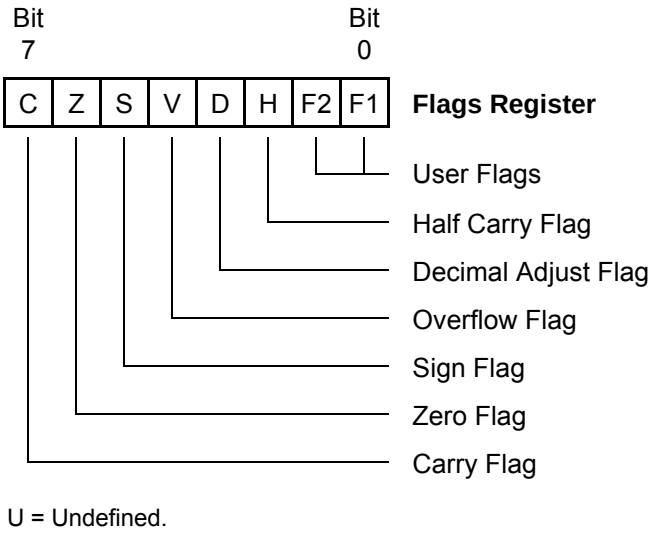


Figure 53. Flags Register

Interrupts, the software trap (TRAP) instruction and illegal instruction traps write the value of the Flags Register to the stack. Executing an Interrupt Return (IRET) instruction restores the value saved on the stack into the Flags Register.

Op Code Maps

Figure 54 and Table 165 provide descriptions of the op code map data and the abbreviations. Figures 55 and Figure 56 provide information about each of the eZ8 CPU instructions.

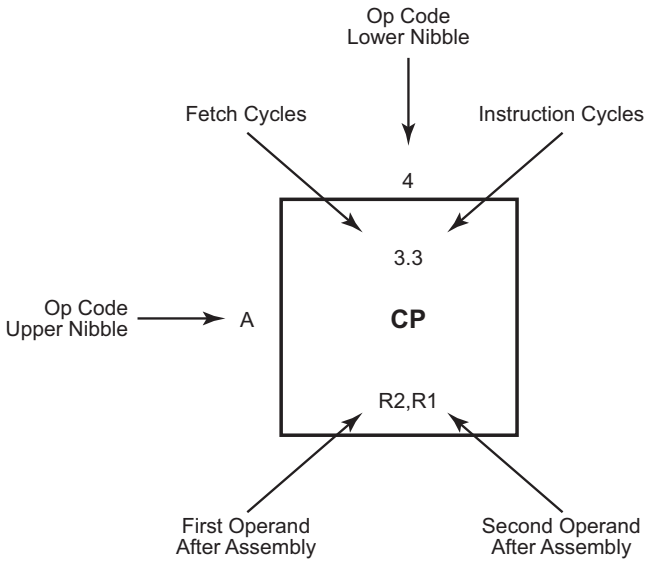


Figure 54. Op Code Map Cell Description

Table 165. 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

Table 165. Op Code Map Abbreviations (Continued)

Abbreviation	Description	Abbreviation	Description
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

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						1.1 **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	2.8 **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**[3] r1,rr2,X	3.4 **LDX**[3] rr1,r2,X						1.2 **DI**
9	2.2 **RL** R1	2.3 **RL** IR1	2.5 **LDE** r2,Irr1	2.8 **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**[3] 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	2.8 **LDCI** Ir1,Irr2	2.3 **JP**[2] IRR1	2.8 **LDC** Ir1,Irr2		3.4 **LD** r1,r2,X	3.3 **PUSHX**[3] ER2							1.2 **RCF**
D	2.2 **SRA** R1	2.3 **SRA** IR1	2.5 **LDC** r2,Irr1	2.8 **LDCI** Ir2,Irr1	2.6 **CALL**[2] IRR1	2.2 **BSWAP** R1	3.3 **CALL** DA	3.4 **LD** r2,r1,X	3.3 **POPX**[3] 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	3.3 **LD** R1,IM	3.4 **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	2.9 **MULT** RR1	3.3 **LD** R2,IR1	3.3 **BTJ** p,b,r1,X	3.4 **BTJ** p,b,Ir1,X								

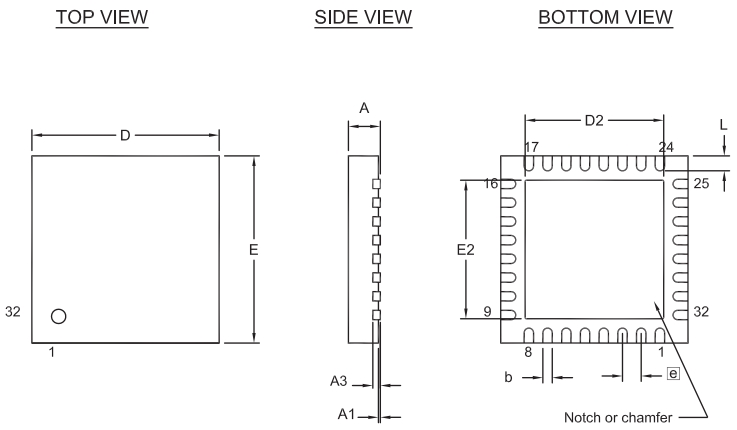
Figure 55. First Op Code Map

Upper Nibble (Hex) \ Lower 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									4.2 **LDWX** ER2,ER1							
F																

Figure 56. Second Op Code Map after 1Fh

Packaging

Figure 57 displays the 32-pin quad flat no lead (QFN) package available for the Z8FMC16100 Series MCU.

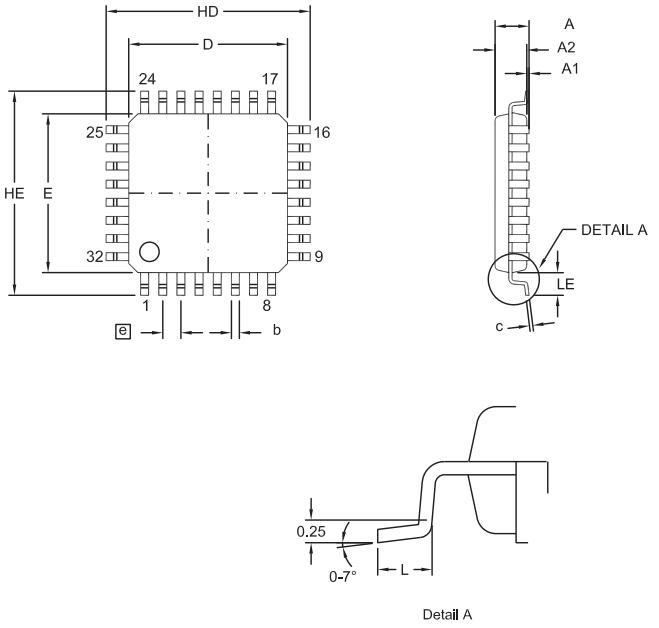


SYMBOL	MILLIMETER		INCH	
	MIN	MAX	MIN	MAX
A	0.80	1.00	0.031	0.039
A1	-	0.05	-	0.002
⚠D A3	0.20 REF		0.008 REF	
b	0.18	0.30	0.007	0.012
D	4.95	5.05	0.195	0.199
E	4.95	5.05	0.195	0.199
⚠D D2	3.00	3.70	0.118	0.146
⚠D E2	3.00	3.70	0.118	0.146
e	0.50 BSC		0.020 BSC	
L	0.35	0.45	0.014	0.018

1. CONTROLLING DIMENSIONS : mm
2. MAX. COPLANARITY : 0.08 mm / 0.003"

Figure 57. 32-Pin Quad Flat No Lead Package

Figure 58 displays the 32-pin low quad flat package (LQFP) available for the Z8FMC16100 Series MCU.



SYMBOL	MILLIMETER		INCH	
	MIN	MAX	MIN	MAX
A	1.40	1.60	0.055	0.063
A1	0.05	0.15	0.002	0.006
A2	1.35	1.45	0.053	0.057
b	0.30	0.45	0.012	0.018
c	0.09	0.20	0.004	0.008
HD	8.75	9.25	0.344	0.364
D	6.90	7.10	0.272	0.280
HE	8.75	9.25	0.344	0.364
E	6.90	7.10	0.272	0.280
e	0.80 BSC		0.031 BSC	
L	0.45	0.75	0.018	0.030
LE	1.00 REF		0.039 REF	

1. CONTROLLING DIMENSIONS : mm
2. MAX. COPLANARITY : .10mm / 0.004"

Figure 58. 32-Pin Low Quad Flat Package

Ordering Information

Table 166 identifies the basic features available for each device within the Z8FMC16100 Series MCU product line. Table 167 provides ordering information for these products by part number. See the Part Number Description section on page 297 for a description of a part number's unique identifying attributes.

Table 166. Z8FMC16100 Series Part Selection Guide

Product Feature	Z8FMC16100	Z8FMC08100	Z8FMC04100
Flash (KB)	16	8	4
SRAM (B)	512	512	512
General-Purpose I/O	17	17	17
Motor Control PWM Channels	6	6	6
ADC Inputs	8	8	8
Operational Amplifier	Yes	Yes	Yes
Comparator	Yes	Yes	Yes
16-bit Standard Timers w/ Capture, Compare, PWM	Yes	Yes	Yes
UART with support for LIN and IrDA	Yes	Yes	Yes
I^2C or SPI Controller	Yes	Yes	Yes
Watchdog Timer	Yes	Yes	Yes
5.5296 MHz Internal Precision Oscillator	Yes	Yes	Yes

Each of the parts listed in Table 167 is shown in a lead-free package. The use of lead-free packaging adheres to a socially responsible environmental standard. To order the standard plastic (lead-soldered) package, contact Zilog Customer Support.

Table 167. Ordering Information for the Z8FMC16100 Series Products*

Part Number	Flash KB (Bytes)	SRAM (Bytes)	GPIO	Max. Speed (MHz)	I^2C/SPI	Trimmed IPO	Package	Temp (ºC)
Z8FMC16100 with 16KB Flash and 512 B SRAM								
Z8FMC16100QKSG	16 (16,384)	512	17	20	I^2C/SPI	Y	QFN-32	0 to +70
Z8FMC16100QKEG								–40 to +105
Z8FMC16100AKSG	16 (16,384)	512	17	20	I^2C/SPI	Y	LQFP-32	0 to +70
Z8FMC16100AKEG								–40 to +105

Note: *Factory-programmed versions of the devices in this table are available upon request from Zilog.

Table 167. Ordering Information for the Z8FMC16100 Series Products* (Continued)

Part Number	Flash KB (Bytes)	SRAM (Bytes)	GPIO	Max. Speed (MHz)	I^2C/SPI	Trimmed IPO	Package	Temp (ºC)
Z8FMC08100 with 8KB Flash and 512 B SRAM								
Z8FMC08100QKSG	8 (8,192)	512	17	20	I^2C/SPI	Y	QFN-32	0 to +70
Z8FMC08100QKEG								–40 to +105
Z8FMC08100AKSG	8 (8,192)	512	17	20	I^2C/SPI	Y	LQFP-32	0 to +70
Z8FMC08100AKEG								–40 to +105
Z8FMC04100 with 4KB Flash and 512 B SRAM								
Z8FMC04100QKSG	4 (4,096)	512	17	20	I^2C/SPI	Y	QFN-32	0 to +70
Z8FMC04100QKEG								–40 to +105
Z8FMC04100AKSG	4 (4,096)	512	17	20	I^2C/SPI	Y	LQFP-32	0 to +70
Z8FMC04100AKEG								–40 to +105
Z8FMC16100 Series Development Kit								
Z8FMC160100KITG	Z8FMC16100 Series Development Kit							
Z8FMC161000ZEM	Z8FMC161000ZEM In-Circuit Emulator Development Tool							
ZUSBOPTSC01ZAC	USB Opto-isolated Smart Cable Accessory Kit							

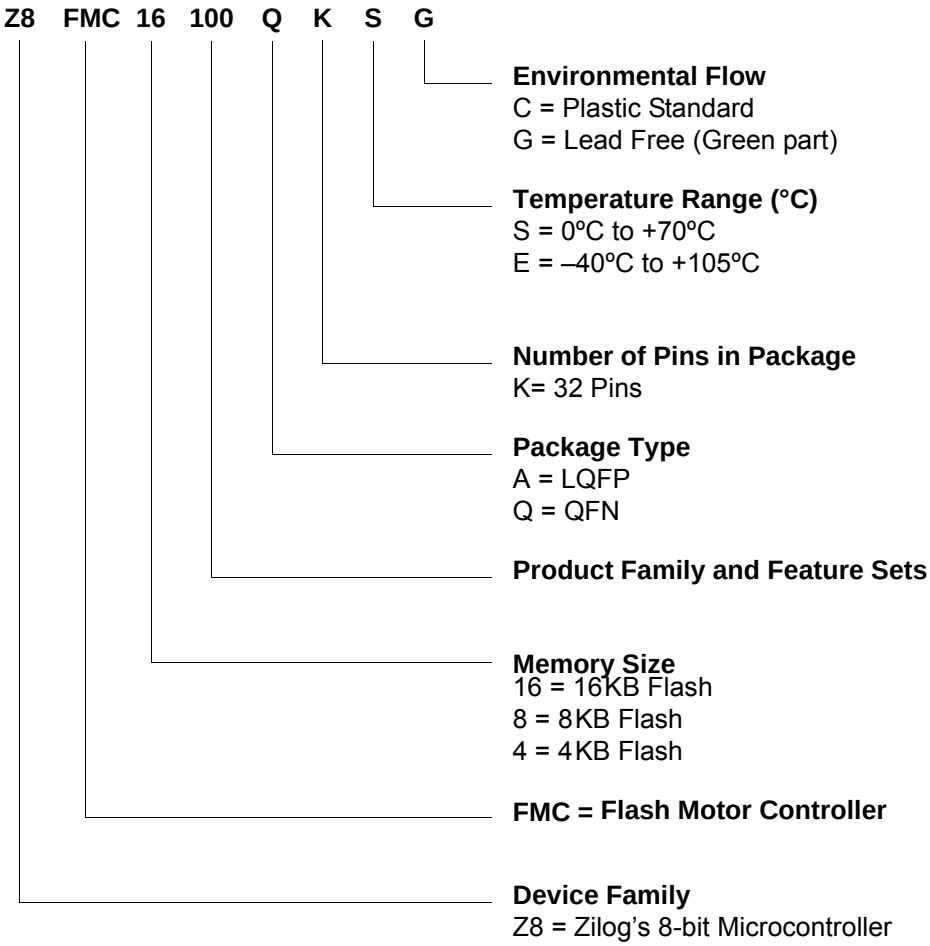
Note: *Factory-programmed versions of the devices in this table are available upon request from Zilog.

For complete details about Z8FMC16100 Series Flash MCU, development tools and downloadable software, visit www.zilog.com.

Part Number Description

Zilog part numbers consist of a number of components, as indicated in the following example.

Example. Part number Z8FMC16100QKSG is a 16-bit Flash Motor Controller with 16KB Program Memory in a QFN package with 32 pins, operating within a 0ºC to +70ºC temperature range and built using lead-free solder.



Appendix A. Register Tables

For the reader's convenience, this appendix lists all Z8FMC16100 Series registers numerically by hexadecimal address.

General Purpose RAM

In the F083A Series, the 000–EFF hexadecimal address range is partitioned for general-purpose random access memory, as follows.

Hex Addresses: 000–1FF

This address range is reserved for general-purpose register file RAM. For more details, see the Register File section on page 13.

Hex Addresses: 200–EFF

This address range is reserved.

Timer 0

For more information about these Timer Control registers, see the General-Purpose Timer chapter on page 86.

Hex Address: F00

Table 168. Timer 0 High Byte Register (T0H)

Bit	7	6	5	4	3	2	1	0
Field	TH							
RESET	00H							
R/W	R/W							
Address	F00H							

Hex Address: F01

Table 169. Timer 0 Low Byte Register (T0L)

Bit	7	6	5	4	3	2	1	0
Field	TL							
RESET	01H							
R/W	R/W							
Address	F01H							

Hex Address: F02

Table 170. Timer 0 Reload High Byte Register (T0RH)

Bit	7	6	5	4	3	2	1	0
Field	TRH							
RESET	FFH							
R/W	R/W							
Address	F02H							

Hex Address: F03

Table 171. Timer 0 Reload Low Byte Register (T0RL)

Bit	7	6	5	4	3	2	1	0
Field	TRL							
RESET	FF							
R/W	R/W							
Address	F03H							

Hex Address: F04

Table 172. Timer 0 PWM High Byte Register (T0PWMH)

Bit	7	6	5	4	3	2	1	0
Field	PWMH							
RESET	00H							
R/W	R/W							
Address	F04H							

Hex Address: F05

Table 173. Timer 0 PWM Low Byte Register (T0PWML)

Bit	7	6	5	4	3	2	1	0
Field	PWML							
RESET	00H							
R/W	R/W							
Address	F05H							

Hex Address: F06

Table 174. Timer 0 Control 0 Register (T0CTL0)

Bit	7	6	5	4	3	2	1	0
Field	TMODE[3]	TICONFIG		TINSEL	PWMD			INCAP
RESET	0	00		0	000			0
R/W	R/W	R/W		R/W	R/W			R
Address	F06H							

Bit	Description
[7] TMODE[3]	**Timer Mode High Bit** This bit along with TMODE[2:0] field in T0CTL1 Register determines the operating mode of the timer. This is the most significant bit of the Timer Mode selection value. For more details, see the T0CTL1 Register description.
[6:5] TICONFIG	**Timer Interrupt Configuration** This field configures timer interrupt definitions. These bits affect all modes. The effect, per mode, is defined below. ONE SHOT, CONTINUOUS, COUNTER, PWM, COMPARE, DUAL PWM, TRIGGERED ONE-SHOT, COMPARATOR COUNTER: 0x = Timer interrupt occurs on reload. 10 = Timer interrupts are disabled. 11 = Timer Interrupt occurs on reload. GATED: 0x = Timer interrupt occurs on reload or inactive gate edge. 10 = Timer interrupt occurs on inactive gate edge. 11 = Timer interrupt occurs on reload. CAPTURE, CAPTURE/COMPARE, CAPTURE RESTART: 0x = Timer interrupt occurs on reload and capture. 10 = Timer interrupt occurs on capture only. 11 = Timer interrupt occurs on reload only.

Bit	Description (Continued)
[4] TINSEL	**Timer Input Select** 0 = Timer input is the Timer input pin. 1 = Timer input is the comparator output.
[3:1] PWMD	**PWM Delay Value** This field is a programmable delay to control the number of additional system clock cycles following a PWM or Reload compare before the Timer Output or the Timer Output Complement is switched to the active state. This field ensures a time gap between the deassertion of one PWM output to the assertion of its complement. 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] INCAP	**Input Capture Event** 0 = Previous timer interrupt is not a result of a Timer Input Capture Event 1 = Previous timer interrupt is a result of a Timer Input Capture Event.

Hex Address: F07

Table 175. Timer 0 Control 1 Register (T0CTL1)

Bit	7	6	5	4	3	2	1	0
Field	TEN	TPOL	PRES			TMODE		
RESET	0	0	000			000		
R/W	R/W	R/W	R/W			R/W		
Address	F07H							

Bit	Description
[7] TEN	**Timer Enable** 0 = Timer is disabled. 1 = Timer is enabled.

Bit	Description (Continued)
[6] TPOL	**Timer Input/Output Polarity** This bit is a function of the current operating mode of the timer. It determines the polarity of the input and/or output signal. When the timer is disabled, the Timer Output signal is set to the value of this bit. **ONE-SHOT Mode** If the timer is enabled the Timer Output signal pulses (changes state) for one system clock cycle after timer Reload. **CONTINUOUS Mode** If the timer is enabled the Timer Output signal is complemented after timer Reload. COUNTER Mode—If the timer is enabled the Timer Output signal is complemented after timer reload. 0 = Count occurs on the rising edge of the Timer Input signal. 1 = Count occurs on the falling edge of the Timer Input signal. **PWM SINGLE OUTPUT Mode** When enabled, the Timer Output is forced to $\overline{\text{TPOL}}$ after PWM count match and forced back to TPOL after Reload. **CAPTURE Mode** If the timer is enabled, the Timer Output signal is complemented after timer Reload. 0 = Count is captured on the rising edge of the Timer Input signal. 1 = Count is captured on the falling edge of the Timer Input signal. **COMPARE Mode** The Timer Output signal is complemented after timer Reload. **GATED Mode** The Timer Output signal is complemented after timer Reload. 0 = Timer counts when the Timer Input signal is High and interrupts are generated on the falling edge of the Timer Input. 1 = Timer counts when the Timer Input signal is Low and interrupts are generated on the rising edge of the Timer Input. **CAPTURE/COMPARE Mode** If the timer is enabled, the Timer Output signal is complemented after timer Reload. 0 = Counting starts on the first rising edge of the Timer Input signal. The current count is captured on subsequent rising edges of the Timer Input signal. 1 = Counting starts on the first falling edge of the Timer Input signal. The current count is captured on subsequent falling edges of the Timer Input signal.

Bit	Description (Continued)
[6] TPOL (cont'd)	**PWM DUAL OUTPUT Mode** If enabled, the Timer Output is set=$\overline{\text{TPOL}}$ after PWM match and set=TPOL after Reload. If enabled the Timer Output Complement takes on the opposite value of the Timer Output. The PWMD field in the T0CTL1 Register determines an optional added delay on the assertion (Low to High) transition of both Timer Output and the Timer Output Complement for deadband generation. **CAPTURE RESTART Mode** If the timer is enabled the Timer Output signal is complemented after timer Reload. 0 = Count is captured on the rising edge of the Timer Input signal. 1 = Count is captured on the falling edge of the Timer Input signal. **COMPARATOR COUNTER Mode** If the timer is enabled the Timer Output signal is complemented after timer Reload. 0 = Count is captured on the rising edge of the Timer Input signal. 1 = Count is captured on the falling edge of the Timer Input signal. **TRIGGERED ONE-SHOT Mode** If the timer is enabled the Timer Output signal is complemented after timer Reload. 0 = The timer triggers on a Low to High transition on the input. 1 = The timer triggers on a High to Low transition on the input.

Bit	Description (Continued)
[5:3] PRES	**Prescale** The timer input clock is divided by 2^{PRES}, where 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 T0CTL0 Register determines the operating mode of the timer. TMODE[3:0] selects from 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 = COMPARATOR COUNTER Mode. 1011 = TRIGGERED ONE-SHOT Mode.

Hex Address: F08

Table 176. ADC Timer Capture High Byte Register (ADCTCAP_H)

Bit	7	6	5	4	3	2	1	0
Field	ADCTCAPH							
RESET	X							
R/W	R							
Address	F08H							

Bit	Description
[7:0] ADCTCAPH	**ADC Timer Capture Count High Byte** 00H–FFH = The timer count is held in the data registers until the next ADC conversion is started.

Hex Address: F09

Table 177. ADC Timer Capture Low Byte Register (ADCTCAP_L)

Bit	7	6	5	4	3	2	1	0
Field	ADCTCAPL							
RESET	X							
R/W	R							
Address	F09H							

Bit	Description
[7:0] ADCTCAPL	**ADC Timer Capture Count Low Byte** 00H–FFH = The timer count is held in the data registers until the next ADC conversion is started.

Hex Address: F0A–F1F

This address range is reserved.

Pulse Width Modulator

For more information about these PWM registers, see the Pulse Width Modulator chapter on page 64.

Hex Address: F20

Table 178. PWM Control 0 Register (PWMCTL0)

Bit	7	6	5	4	3	2	1	0
Field	PWMOFF	OUTCTL	ALIGN	Reserved	ADCTRIG	Reserved	READY	PWMEN
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	F20H							

Bit	Description
[7] PWMOFF	**Place PWM Outputs in OFF State** 0 = Disable modulator control of PWM pins. Outputs are in predefined off-state. This is not dependent on the reload event. 1 = Reenable modulator control of PWM pins at next PWM reload event.

Bit	Description (Continued)
[6] OUTCTL	**PWM Output Control** 0 = PWM outputs are controlled by the PWM. 1 = PWM outputs selectively disabled (set to off-state) according to values in the OUT*x* bits of the PWMOUT Register.
[5] ALIGN	**PWM Edge Alignment** 0 = PWM outputs are edge aligned. 1 = PWM outputs are center aligned.
[4] Reserved	**Reserved** This bit is reserved and must be programmed to 0.
[3] ADCTRIG	**ADC Trigger Enable** 0 = No ADC trigger pulses. 1 = ADC trigger enabled.
[2] Reserved	**Reserved** This bit is reserved and must be programmed to 0.
[1] READY	**Values Ready for Next Reload Event** 0 = PWM values (prescale, period and duty cycle) are not ready. Do not use values in holding registers at next PWM reload event. 1 = PWM values (prescale, period and duty cycle) are ready. Transfer all values from temporary holding registers to working registers at next PWM reload event.
[0] PWMEN	**PWM Enable** 0 = PWM is disabled and enabled PWM output pins are forced to default off-state. PWM master counter is stopped. Certain control registers may only written in this state. 1 = PWM is enabled and PWM output pins are enabled as outputs.

Hex Address: F21

Table 179. PWM Control 1 Register (PWMCTL1)

Bit	7	6	5	4	3	2	1	0
Field	RLFREQ[1:0]		INDEN	POL45	POL23	POL10	PRES[1:0]	
RESET	00		0	0	0	0	00	
R/W	R/W		R/W	R/W	R/W	R/W	R/W	
Address	F21H							

Bit	Description
[7:6] RLFREQ[1:0]	**Reload Event Frequency** This bit field is buffered. Changes to the reload event frequency takes effect at the end of the current PWM period. Reads always return the bit values from the temporary holding register. 00 = PWM reload event occurs at the end of every PWM period. 01 = PWM reload event occurs once every 2 PWM periods. 10 = PWM reload event occurs once every 4 PWM periods. 11 = PWM reload event occurs once every 8 PWM periods.
[5] INDEN	**Independent PWM Mode Enable** 0 = This bit may only be altered when PWMEN (PWMCTL0) cleared; PWM outputs operate as 3 complementary pairs. 1 = PWM outputs operate as 6 independent channels.
[4] POL45	**PWM2 Polarity** 1 = Invert Output polarity for channel pair PWM2. 0 = Noninverted polarity for channel pair PWM2.
[3] POL23	**PWM1 Polarity** 1 = Invert Output polarity for channel pair PWM1. 0 = Noninverted polarity for channel pair PWM1.
[2] POL10	**PWM0 Polarity** 1 = Invert Output polarity for channel pair PWM0. 0 = Noninverted polarity for channel pair PWM0.
[1:0] PRES	**PWM Prescaler** The prescaler divides down the PWM input clock (either the system clock or the PWMIN external input). This field is buffered. Changes to this field take effect at the next PWM reload event. Reads always return the values from the temporary holding register. 00 = Divide by 1. 01 = Divide by 2. 10 = Divide by 4. 11 = Divide by 8.

Hex Address: F22

Table 180. PWM DeadBand Register (PWMDB)

Bit	7	6	5	4	3	2	1	0
Field	PWMDB[7:0]							
RESET	01H							
R/W	R/W							
Address	F22H							

Bit	Description
[7:0] PWMDB	**PWM Deadband** Sets the PWM dead band period for which both PWM outputs of a complementary PWM output pair are deasserted.

Note: This register can only be written when PWMEN is cleared.

Hex Address: F23

Table 181. PWM Minimum Pulse Width Filter (PWMMPF)

Bit	7	6	5	4	3	2	1	0
Field	PWMMPF[7:0]							
RESET	00H							
R/W	R/W							
Address	F23H							

Bit	Description
[7:0] PWMMPF	**PWM Minimum Pulse Filter** Sets the minimum allowed output pulse width in PWM clock cycles.

Note: This register can only be written when PWMEN is cleared.

Hex Address: F24

Table 182. PWM Fault Mask Register (PWMFM)

Bit	7	6	5	4	3	2	1	0
Field	Reserved		DBGMSK	Reserved		F1MASK	C0MASK	FMASK
RESET	00		0	00		0	0	0
R/W	R		R/W	R		R/W	R/W	R/W
Address	F24H							

Bit	Description
[7:6]	Must be 0.
[5] DBGMSK	**Debug Entry Fault Mask** 0 = Entering CPU DEBUG Mode generates a PWM fault. 1 = Entering CPU DEBUG Mode does not generate a PWM fault.
[4:3]	**Reserved** These bits are reserved and must be programmed to 00.
[2] F1MASK	**$\overline{\text{Fault1}}$ Fault Mask** 0 = $\overline{\text{Fault1}}$ generates a PWM fault. 1 = $\overline{\text{Fault1}}$ does not generate a PWM fault.
[1] C0MASK	**Comparator Fault Mask** 0 = Comparator generates a PWM fault. 1 = Comparator does not generate a PWM fault.
[0] F0MASK	**Fault Pin Mask** 0 = $\overline{\text{Fault0}}$ pin generates a PWM fault. 1 = $\overline{\text{Fault0}}$ pin does not generate a PWM fault.
Note: This register can only be written when PWMEN is cleared.	

Hex Address: F25

Table 183. PWM Fault Status Register (PWMFSTAT)

Bit	7	6	5	4	3	2	1	0
Field	RLDFlag	Reserved	DBGFLAG	Reserved		F1FLAG	C0FLAG	FFLAG
RESET	U	0	U	00		U	U	U
R/W	R/W1C	R	R/W1C	R		R/W1C	R/W1C	R/W1C
Address	F25H							

Bit	Description
[7] RLDFlag	Reload Flag This bit is set and latched when a PWM timer reload occurs. Writing a 1 to this bit clears the flag.
[6]	**Reserved** This bit is reserved and must be programmed to 0.
[5] DBGFLAG	**Debug Flag** This bit is set and latched when DEBUG Mode is entered. Writing a 1 to this bit clears the flag.
[4:3]	**Reserved** These bits are reserved and must be programmed to 00.
[2] F1FLAG	**$\overline{\text{Fault1}}$ Flag** This bit is set and latched when Fault1 is asserted. Writing a 1 to this bit clears the flag.
[1] C0FLAG	**Comparator 0 Flag** This bit is set and latched when Comparator is asserted. Writing a 1 to this bit clears the flag.
[0] FFLAG	**Fault Flag** This bit is set and latched when the $\overline{\text{FAULT0}}$ input is asserted. Writing a 1 to this bit clears the flag.

Note: For this register, W1C means that you must write a 1 to clear the flag.

Hex Address: F26

Table 184. PWM Input Sample Register (PWMIN)

Bit	7	6	5	4	3	2	1	0
Field	Reserved	FAULT	IN2H	IN2L	IN1H	IN1L	IN0H	IN0L
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	F26H							

Bit	Description
[7]	**Reserved** This bit is reserved and must be programmed to 0.
[6] FAULT	**Sample $\overline{\text{Fault0}}$ Pin** 0 = A Low level signal was read on the $\overline{\text{Fault0}}$ pin. 1 = A High level signal was read on the $\overline{\text{Fault0}}$ pin.
[5:0] IN2H/IN2L/ IN1H/IN1L/ IN0H/IN0L	**Sample PWM Pins** 0 = A Low level signal was read on the pins. 1 = A High level signal was read on the pins.

Hex Address: F27

Table 185. PWM Output Control Register (PWMOUT)

Bit	7	6	5	4	3	2	1	0
Field	Reserved	Reserved	OUT2L	OUT2H	OUT1L	OUT1H	OUT0L	OUT0H
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	F27H							

Bit	Description
[7:6] Reserved	**Reserved** These bits are reserved and must be programmed to 00.
[5, 3, 1] OUTxL	**PWM L2/L1/L0 Output Configuration** 0 = PWM L2/L1/L0 output signal is enabled and controlled by PWM. 1 = PWM L2/L1/L0 output signal is in low-side off-state.
[4, 2, 0] OUTyH	**PWM H2/H1/H0 Output Configuration** 0 = PWM H2/H1/H0 output signal is enabled and controlled by PWM. 1 = PWM H2/H1/H0 output signal is in high-side off-state.

Note: x indicates low bits in the range 2–0; y indicates high bits in the range 2–0.

Hex Address: F28

Table 186. PWM Fault Control Register (PWMFCTL)

Bit	7	6	5	4	3	2	1	0
Field	Reserved	DBGRST	Fault1INT	Fault1RST	CMP0INT	CMP0RST	Fault0INT	Fault0RST
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	F28H							

Bit	Description
[7] Reserved	**Reserved** This bit is reserved and must be programmed to 0.
[6] DBGRST	**DebugRestart** 0 = Automatic recovery. PWM resumes control of outputs when all fault sources have deasserted and a new PWM period begins. 1 = Software controlled recovery. PWM resumes control of outputs only after all fault sources have deasserted and all fault flags are cleared and a PWM reload occurs.
[5] Fault1INT	**$\overline{\text{Fault1}}$ Interrupt** 0 = Interrupt on comparator assertion disabled. 1 = Interrupt on comparator assertion enabled.
[4] Fault1RST	**$\overline{\text{Fault1}}$ Restart** 0 = Automatic recovery. PWM resumes control of outputs when all fault sources have deasserted. 1 = Software controlled recovery. PWM resumes control of outputs only after all fault sources have deasserted and all fault flags are cleared and a PWM reload occurs.
[3] CMP0INT	**Comparator 0 Interrupt** 0 = Interrupt on comparator 0 assertion disabled. 1 = Interrupt on comparator 0 assertion enabled.
[2] CMP0RST	**Comparator 0 Restart** 0 = Automatic recovery. PWM resumes control of outputs when all fault sources have deastered. 1 = Software controlled recovery. PWM resumes control of outputs only after all fault sources have deasserted and all fault flags are cleared and a PWM reload occurs
[1] Fault0INT	**$\overline{\text{Fault0}}$ Interrupt** 0 = Interrupt on $\overline{\text{Fault0}}$ pin assertion disabled. 1 = Interrupt on $\overline{\text{Fault0}}$ pin assertion enabled.
[0] Fault0RST	**$\overline{\text{Fault0}}$ Restart** 0 = Automatic recovery. PWM resumes control of outputs when all fault sources have deastered. 1 = Software-controlled recovery. PWM resumes control of outputs only after all fault sources have deasserted and all fault flags are cleared and a PWM reload occurs

Note: This register can only be written when PWMEN is cleared.

Hex Address: F29

Table 187. Current-Sense Trigger Control Register (PWMSHC)

Bit	7	6	5	4	3	2	1	0
Field	CSTPOL	HEN	NHEN	LEN	NLEN	CSTPWM2	CSTPWM1	CSTPWM0
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	F29H							

Bit	Description
[7] CSTPOL	**Sample/Hold Polarity** 0 = Hold when terms are active. 1 = Hold when terms are not active.
[6] HEN	**High Side Active Enable** 0 = Ignore Product of PWMH0, PWMH1 and PWMH2 in Sample/Hold equation 1 = Hold when PWMH0, PWMH1 and PWMH2 are all active.
[5] NHEN	**High Side Inactive Enable** 0 = Ignore Product of $\overline{\text{PWMH0}}$, $\overline{\text{PWMH1}}$ and $\overline{\text{PWMH2}}$ in Sample/Hold equation. 1 = Hold when are all active.
[4] LEN	**Low Side Active Enable** 0 = Ignore Product of PWML0, PWML1 and PWML2 in Sample/Hold equation. 1 = Hold when PWML0, PWML1 and PWML2 are all active.
[3] NLEN	**Low Side Inactive Enable** 0 = Ignore Product of $\overline{\text{PWML0}}$, $\overline{\text{PWML1}}$ and $\overline{\text{PWML2}}$ in Sample/Hold equation. 1 = Hold when $\overline{\text{PWML0}}$, $\overline{\text{PWML1}}$ and $\overline{\text{PWML2}}$ are all active.
[2] CSTPWM2	**PWM Channel2 Sample/Hold Enable** 0 = Channel 2 terms are not used in Sample/Hold equation. 1 = Channel 2 terms are used in Sample/Hold equation.
[1] CSTPWM1	**PWM Channel1 Sample/Hold Enable** 0 = Channel 1 terms are not used in Sample/Hold equation. 1 = Channel 1 terms are used in Sample/Hold equation.
[0] CSTPWM0	**PWM Channel0 Sample/Hold Enable** 0 = Channel 0 terms are not used in Sample/Hold equation. 1 = Channel 0 terms are used in Sample/Hold equation.

Hex Address: F2A–F2B

This address range is reserved.

Hex Address: F2C

Table 188. PWM High Byte Register (PWMH)

Bit	7	6	5	4	3	2	1	0
Field	Reserved				PWMH			
RESET	0H				0H			
R/W	R/W				R/W			
Address	F2CH							

Hex Address: F2D

Table 189. PWM Low Byte Register (PWML)

Bit	7	6	5	4	3	2	1	0
Field	PWML							
RESET	01H							
R/W	R/W							
Address	F2DH							

Hex Address: F2E

Table 190. PWM Reload High Byte Register (PWMRH)

Bit	7	6	5	4	3	2	1	0
Field	Reserved				PWMRH			
RESET	0H				FH			
R/W	R/W				R/W			
Address	F2EH							

Hex Address: F2F

Table 191. PWM Reload Low Byte Register (PWMRL)

Bit	7	6	5	4	3	2	1	0
Field	PWMRL							
RESET	FF							
R/W	R/W							
Address	F2FH							

Hex Address: F30

Table 192. PWM 0–2 H/L Duty Cycle High Byte Register (PWMH*x*DH, PWML*x*DH)

Bit	7	6	5	4	3	2	1	0
Field	SIGN	Reserved		DUTYH				
RESET	0	00		0_0000				
R/W	R/W	R/W		R/W				
Address	F30H, F32H, F34H, F36H, F38H, F3AH							

Hex Address: F31

Table 193. PWM 0–2 H/L Duty Cycle Low Byte Register (PWMH*x*DL, PWML*x*DL)

Bit	7	6	5	4	3	2	1	0
Field	DUTYL							
RESET	00H							
R/W	R/W							
Address	F31H, F33H, F35H, F37H, F39H, F3BH							

Bit	Description
[7:0] DUTYL	**PWM Duty Cycle High and Low Bytes** The two bytes {DUTYH[7:0], DUTYL[7:0]} form a 14-bit signed value. The value is compared to the current 12-bit PWM count.

Hex Address: F32

Table 194. PWM 0–2 H/L Duty Cycle High Byte Register (PWMH*x*DH, PWML*x*DH)

Bit	7	6	5	4	3	2	1	0
Field	SIGN	Reserved		DUTYH				
RESET	0	00		0_0000				
R/W	R/W	R/W		R/W				
Address	F30H, F32H, F34H, F36H, F38H, F3AH							

Hex Address: F33

Table 195. PWM 0–2 H/L Duty Cycle Low Byte Register (PWMH*x*DL, PWML*x*DL)

Bit	7	6	5	4	3	2	1	0
Field	DUTYL							
RESET	00H							
R/W	R/W							
Address	F31H, F33H, F35H, F37H, F39H, F3BH							

Bit	Description
[7:0] DUTYL	**PWM Duty Cycle High and Low Bytes** The two bytes {DUTYH[7:0], DUTYL[7:0]} form a 14-bit signed value. The value is compared to the current 12-bit PWM count.

Hex Address: F34

Table 196. PWM 0–2 H/L Duty Cycle High Byte Register (PWMH*x*DH, PWML*x*DH)

Bit	7	6	5	4	3	2	1	0
Field	SIGN	Reserved		DUTYH				
RESET	0	00		0_0000				
R/W	R/W	R/W		R/W				
Address	F30H, F32H, F34H, F36H, F38H, F3AH							

Hex Address: F35

Table 197. PWM 0–2 H/L Duty Cycle Low Byte Register (PWMH*x*DL, PWML*x*DL)

Bit	7	6	5	4	3	2	1	0
Field	DUTYL							
RESET	00H							
R/W	R/W							
Address	F31H, F33H, F35H, F37H, F39H, F3BH							

Bit	Description
[7:0] DUTYL	**PWM Duty Cycle High and Low Bytes** The two bytes {DUTYH[7:0], DUTYL[7:0]} form a 14-bit signed value. The value is compared to the current 12-bit PWM count.

Hex Address: F36

Table 198. PWM 0–2 H/L Duty Cycle High Byte Register (PWMH*x*DH, PWML*x*DH)

Bit	7	6	5	4	3	2	1	0
Field	SIGN	Reserved		DUTYH				
RESET	0	00		0_0000				
R/W	R/W	R/W		R/W				
Address	F30H, F32H, F34H, F36H, F38H, F3AH							

Hex Address: F37

Table 199. PWM 0–2 H/L Duty Cycle Low Byte Register (PWMH*x*DL, PWML*x*DL)

Bit	7	6	5	4	3	2	1	0
Field	DUTYL							
RESET	00H							
R/W	R/W							
Address	F31H, F33H, F35H, F37H, F39H, F3BH							

Bit	Description
[7:0] DUTYL	**PWM Duty Cycle High and Low Bytes** The two bytes {DUTYH[7:0], DUTYL[7:0]} form a 14-bit signed value. The value is compared to the current 12-bit PWM count.

Hex Address: F38

Table 200. PWM 0–2 H/L Duty Cycle High Byte Register (PWMH*x*DH, PWML*x*DH)

Bit	7	6	5	4	3	2	1	0
Field	SIGN	Reserved		DUTYH				
RESET	0	00		0_0000				
R/W	R/W	R/W		R/W				
Address	F30H, F32H, F34H, F36H, F38H, F3AH							

Hex Address: F39

Table 201. PWM 0–2 H/L Duty Cycle Low Byte Register (PWMH*x*DL, PWML*x*DL)

Bit	7	6	5	4	3	2	1	0
Field	DUTYL							
RESET	00H							
R/W	R/W							
Address	F31H, F33H, F35H, F37H, F39H, F3BH							

Bit	Description
[7:0] DUTYL	**PWM Duty Cycle High and Low Bytes** The two bytes {DUTYH[7:0], DUTYL[7:0]} form a 14-bit signed value. The value is compared to the current 12-bit PWM count.

Hex Address: F3A

Table 202. PWM 0–2 H/L Duty Cycle High Byte Register (PWMH*x*DH, PWML*x*DH)

Bit	7	6	5	4	3	2	1	0
Field	SIGN	Reserved		DUTYH				
RESET	0	00		0_0000				
R/W	R/W	R/W		R/W				
Address	F30H, F32H, F34H, F36H, F38H, F3AH							

Hex Address: F3B

Table 203. PWM 0–2 H/L Duty Cycle Low Byte Register (PWMH*x*DL, PWML*x*DL)

Bit	7	6	5	4	3	2	1	0
Field	DUTYL							
RESET	00H							
R/W	R/W							
Address	F31H, F33H, F35H, F37H, F39H, F3BH							

Bit	Description
[7:0] DUTYL	**PWM Duty Cycle High and Low Bytes** The two bytes {DUTYH[7:0], DUTYL[7:0]} form a 14-bit signed value. The value is compared to the current 12-bit PWM count.

Hex Addresses: F3C–F3F

This address range is reserved.

LIN-UART

For more information about these LIN-UART registers, see the LIN-UART chapter on page 106.

Hex Address: F40

Table 204. LIN-UART Transmit Data Register (U0TXD)

Bit	7	6	5	4	3	2	1	0
Field	TXD							
RESET	X							
R/W	W							
Address	F40H							

Table 205. LIN-UART Receive Data Register (U0RXD)

Bit	7	6	5	4	3	2	1	0
Field	RXD							
RESET	X							
R/W	R							
Address	F40H							

Hex Address: F41

Table 206. LIN-UART Status 0 Register, Standard UART Mode (U0STAT0)

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							

Hex Address: F42

Table 207. LIN-UART Control 0 Register (U0CTL0)

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							

Hex Address: F43

Table 208. MultiProcessor Control Register (U0CTL1 with MSEL = 000b)

Bit	7	6	5	4	3	2	1	0
Field	MPMD[1]	MPEN	MPMD[0]	MPBT	DEPOL	BRGCTL	RDAIRQ	IREN
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 with MSEL = 000b							

Hex Address: F44

Table 209. LIN-UART Mode Select and Status Register (U0MDSTAT)

Bit	7	6	5	4	3	2	1	0
Field	MSEL			Mode Status				
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R	R	R	R	R
Address	F44H							

Hex Address: F45

Table 210. LIN-UART Address Compare Register (U0ADDR)

Bit	7	6	5	4	3	2	1	0
Field	COMP_ADDR							
RESET	00H							
R/W	R/W							
Address	F45H							

Hex Address: F46

Table 211. LIN-UART Address Compare Register (U0ADDR)

Bit	7	6	5	4	3	2	1	0
Field	COMP_ADDR							
RESET	00H							
R/W	R/W							
Address	F45H							

Hex Address: F47

Table 212. LIN-UART Baud Rate Low Byte Register (U0BRL)

Bit	7	6	5	4	3	2	1	0
Field	BRL							
RESET	FFH							
R/W	R/W							
Address	F47H							

Hex Addresses: F48–F5F

This address range is reserved.

I²C

For more information about these I²C registers, see the I2C Master/Slave Controller chapter on page 160.

Hex Address: F50

Table 213. I²C Data Register (I2CDATA)

Bit	7	6	5	4	3	2	1	0
Field	DATA							
RESET	0							
R/W	R/W							
Address	F50H							

Hex Address: F51

Table 214. I²C Interrupt Status Register (I2CISTAT)

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							

Hex Address: F52

Table 215. I²C Control Register (I2CCTL)

Bit	7	6	5	4	3	2	1	0
Field	IEN	START	STOP	BIRQ	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	R/W	R/W
Address	F52H							

Hex Address: F53

Table 216. I²C Baud Rate High Byte Register (I2CBRH)

Bit	7	6	5	4	3	2	1	0
Field	BRH							
RESET	FFH							
R/W	R/W							
Address	F53H							

Hex Address: F54

Table 217. I²C Baud Rate Low Byte Register (I2CBRL)

Bit	7	6	5	4	3	2	1	0
Field	BRL							
RESET	FFH							
R/W	R/W							
Address	F54H							

Hex Address: F55

Table 218. I²C 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	X	X
R/W	R	R	R	R	R	R	R	R
Address	F55H							

Table 219. 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							

Hex Address: F56

Table 220. I²C Mode Register (I2CMODE)

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	R/W		R/W	R/W	R/W		R/W
Address	F56H							

Hex Address: F57

Table 221. I²C Slave Address Register (I2CSLVAD)

Bit	7	6	5	4	3	2	1	0
Field	SLA[7:0]							
RESET	00H							
R/W	R/W							
Address	F57H							

Hex Addresses: F58–F5F

This address range is reserved.

SPI

For more information about these SPI registers, see the Serial Peripheral Interface chapter on page 146.

Hex Address: F60

Table 222. SPI Data Register (SPIDATA)

Bit	7	6	5	4	3	2	1	0
Field	DATA							
RESET	X							
R/W	R/W							
Address	F60H							

Hex Address: F61

Table 223. SPI Control Register (SPICTL)

Bit	7	6	5	4	3	2	1	0
Field	IRQE	STR	BIRQ	PHASE	CLKPOL	WOR	MMEN	SPIEN
RESET	00H							
R/W	R/W							
Address	F61H							

Hex Address: F62

Table 224. SPI Status Register (SPISTAT)

Bit	7	6	5	4	3	2	1	0
Field	IRQ	OVR	COL	ABT	Reserved		TXST	SLAS
RESET	0	0	0	0	0	0	0	1
R/W	R/W*				R			
Address	F62H							

Note: *R/W = Read access; write a 1 to clear the bit to 0.

Hex Address: F63

Table 225. SPI Mode Register (SPIMODE)

Bit	7	6	5	4	3	2	1	0
Field	Reserved		DIAG	NUMBITS[2:0]			SSIO	SSV
RESET	00H							
R/W	R		R/W					
Address	F63H							

Hex Address: F64

Table 226. SPI Diagnostic State Register (SPIDST)

Bit	7	6	5	4	3	2	1	0
Field	SCKEN	TCKEN	SPISTATE					
RESET	00H							
R/W	R							
Address	F64H							

Hex Address: F65

This address is reserved.

Hex Address: F66

Table 227. SPI Baud Rate High Byte Register (SPIBRH)

Bit	7	6	5	4	3	2	1	0
Field	BRH							
RESET	FFH							
R/W	R/W							
Address	F66H							

Hex Address: F67

Table 228. SPI Baud Rate Low Byte Register (SPIBRL)

Bit	7	6	5	4	3	2	1	0
Field	BRL							
RESET	FFH							
R/W	R/W							
Address	F67H							

Hex Addresses: F68–F6F

This address range is reserved.

Analog-to-Digital Converter

For more information about these ADC registers, see the Analog-to-Digital Converter chapter on page 197.

Hex Address: F70

Table 229. ADC Control Register 0 (ADCCTL0)

Bit	7	6	5	4	3	2	1	0
Field	START	Reserved	REFEN	ADCEN	Reserved	ANAIN[2:0]		
RESET	0	0	0	0	0	0	0	0
R/W	R/W1	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F70H							

Bit	Description
[7] START	**ADC Start/Busy** 0 = Writing to 0 has no effect; reading a 0 indicates that the ADC is available to begin a conversion. 1 = Writing to 1 starts a conversion; reading a 1 indicates that a conversion is currently in progress.
[6]	**Reserved** This bit is reserved and must be programmed to 0.

Bit	Description (Continued)
[5] REFEN	**Reference Enable** 0 = Internal reference voltage is disabled allowing an external reference voltage to be used by the ADC. 1 = Internal reference voltage for the ADC is enabled. The internal reference voltage can be measured on the V_{REF} pin.
[4] ADCEN	**ADC Enable** 0 = ADC is disabled for low power operation. 1 = ADC is enabled for normal use.
[3]	**Reserved** This bit is reserved and must be programmed to 0.
[2:0] ANAIN	**Analog Input Select** 000 = ANA0 input is selected for analog-to-digital conversion. 001 = ANA1 input is selected for analog-to-digital conversion. 010 = ANA2 input is selected for analog-to-digital conversion. 011 = ANA3 input is selected for analog-to-digital conversion. 100 = ANA4 input is selected for analog-to-digital conversion. 101 = ANA5 input is selected for analog-to-digital conversion. 110 = ANA6 input is selected for analog-to-digital conversion. 111 = ANA7 input is selected for analog-to-digital conversion.

Hex Address: F71

Table 230. ADC Raw Data High Byte Register (ADCRD_H)

Bit	7	6	5	4	3	2	1	0
Field	ADCRDH							
RESET	X							
R/W	R							
Address	F71H							

Note: X = Undefined.

Bit	Description
[7:0] ADCRDH	**ADC Raw Data High Byte** 00H–FFH = The data in this register is the raw data coming from the SAR Block. It will change as the conversion is in progress. This register is used for testing only.

Hex Address: F72

Table 231. ADC Data High Byte Register (ADCD_H)

Bit	7	6	5	4	3	2	1	0
Field	ADCDH							
RESET	X							
R/W	R							
Address	F72H							
Note: X = Undefined.								

Bit	Description
[7:0] ADCDH	**ADC High Byte** 00H–FFH = The last conversion output is held in the data registers until the next ADC conversion has completed.

Hex Address: F73

Table 232. ADC Data Low Bits Register (ADCD_L)

Bit	7	6	5	4	3	2	1	0
Field	ADCDL		Reserved					
RESET	X		X					
R/W	R		R					
Address	F73H							
Note: X = Undefined.								

Bit	Description
[7:6] ADCDL	**ADC Low Bits** 00–11b = These bits are the 2 least significant bits of the 10-bit ADC output. These bits are undefined after a Reset. The low bits are latched into this register whenever the ADC Data High Byte Register is read.
[5:0]	**Reserved** These bits are reserved and must be programmed to 000000.

Hex Address: F74

Table 233. Sample and Settling Time (ADCSST)

Bit	7	6	5	4	3	2	1	0
Field	Reserved			SST				
RESET	0			1	1	1	1	1
R/W	R			R/W				
Address	F74H							

Bit	Description
[7:5]	**Reserved** These bits are reserved and must be programmed to 000.
[4:0] SST	**Sample Settling Time** 0H–FH = The number of system clock periods to meet a 0.5µs minimum.

Hex Address: F75

Table 234. Sample/Hold Time (ADCST)

Bit	7	6	5	4	3	2	1	0
Field	Reserved		ST					
RESET	0		1	1	1	1	1	1
R/W	R/W		R/W					
Address	F75H							

Bit	Description
[7:6]	**Reserved** These bits are reserved and must be programmed to 00.
[5:0] ST	**Sample/Hold Time** 0H–FH = The number of system clock periods to meet a 1µs minimum.

Hex Address: F76

Table 235. ADC Clock Prescale Register (ADCCP)

Bit	7	6	5	4	3	2	1	0
Field	Reserved				DIV16	DIV8	DIV4	DIV2
RESET	0				0	0	0	0
R/W	R/W							
Address	F76H							

Bit	Description
[7:4]	**Reserved** These bits are reserved and must be programmed to 0000.
[3] DIV16	**Divide By 16** 0 = Clock is not divided. 1 = System Clock is divided by 16 for ADC Clock.
[2] DIV8	**Divide By 8** 0 = Clock is not divided. 1 = System Clock is divided by 8 for ADC Clock.
[1] DIV4	**Divide By 4** 0 = Clock is not divided. 1 = System Clock is divided by 4 for ADC Clock.
[0] DIV2	**Divide By 2** 0 = Clock is not divided. 1 = System Clock is divided by 2 for ADC Clock.

Hex Addresses: F77–F85

This address range is reserved.

Oscillator Control

For more information about these Oscillator Control registers, see the Oscillator Control chapter on page 227.

Hex Address: F86

Table 236. Oscillator Control Register (OSCCTL)

Bit	7	6	5	4	3	2	1	0
Field	INTEN	XTLEN	WDTEN	POFEN	WDFEN	FLPEN	SCKSEL	
RESET	1	0	1	0	0	0*	00	
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Address	F86H							
Note: *The reset value is 1 if the option bit LPDEN is 0.								

Bit	Description
[7] INTEN	**Internal Precision Oscillator Enable** 0 = Internal precision oscillator is disabled. 1 = Internal precision oscillator is enabled.
[6] XTLEN	**Crystal Oscillator Enable** 0 = Crystal oscillator is disabled. 1 = Crystal oscillator is enabled.
[5] WDTEN	**Watchdog Timer Oscillator Enable** 0 = Watchdog Timer oscillator is disabled. 1 = Watchdog Timer oscillator is enabled.
[4] POFEN	**Primary Oscillator Failure Detection Enable** 0 = Failure detection and recovery of primary oscillator is disabled. This bit is cleared automatically if a primary oscillator failure is detected. 1 = Failure detection and recovery of primary oscillator is enabled.
[3] WDFEN	**Watchdog Timer Oscillator Failure Detection Enable** 0 = Failure detection of Watchdog Timer oscillator is disabled.This bit is cleared automatically if a Watchdog Timer oscillator failure is detected. 1 = Failure detection of Watchdog Timer oscillator is enabled.
[2] FLPEN	**Flash Low Power Mode Enable** 0 = Flash Low Power Mode is disabled. 1 = Flash Low Power Mode is enabled. The Flash is powered down during idle periods of the clock and powered up during Flash reads. This bit should only be set if the frequency of the primary oscillator source is 8MHz or lower. The reset value of this bit is controlled by the LPDEN option bit during reset.
[1:0] SCKSEL	**System Clock Oscillator Select** 00 = Internal precision oscillator functions as system clock at 5.6MHz. 01 = Crystal oscillator or external clock driver functions as system clock . 10 = Reserved. 11 = Watchdog Timer oscillator functions as system clock.

Hex Address: F87

Table 237. Oscillator Divide Register (OSCDIV)

Bit	7	6	5	4	3	2	1	0
Field	DIV							
RESET	00H*							
R/W	R/W							
Address	F87H							

Note: *The reset value is 08H if the option bit LPDEN is 0.

Bit	Description
[7:0] DIV	**Oscillator Divide** In the range 00H–FFH, the 00H address determines that the divider is disabled. All other entries represent divide values for scaling the system clock.

Trim Control

For more information about these Trim Control registers, see the Option Bits chapter on page 221.

Hex Addresses: F88–F8F

This address range is reserved.

Comparator and Op Amp

For more information about these Comparator and Op Amp registers, see the Comparator and Operational Amplifier chapter on page 194.

Hex Address: F90

Table 238. Comparator and Op Amp Control Register (CMPOPC)

Bit	7	6	5	4	3	2	1	0
Field	OPEN	Reserved		CPSEL	CMPIRQ	CMPIV	CMPOUT	CMPEN
RESET	0	00		0	0	0	X	0
R/W	R/W	R/W		R/W	R/W	R/W	R	R/W
Address	F90H							

Bit	Description
[7] OPEN	**Operational Amplifier Disable** 0 = Operational amplifier is disabled. 1 = Operational amplifier is enabled.
[6:5]	**Reserved** These bits are reserved and must be programmed to 00.
[4] CPSEL	**Comparator Input Select** 0 = Comparator input is PA1 1 = Comparator input is PB4
[3] CMPIRQ	**Comparator Interrupt Edge Select** 0 = Interrupt Request on Comparator Rising Edge 1 = Interrupt Request on Comparator Falling Edge
[2] CMPIV	**PWM Fault Comparator Polarity** 0 = PWM Fault is active when CP- > CP+ 1 = PWM Fault is active when CP+ > CP-
[1] CMPOUT	**Comparator Output Value** 0 = Comparator output is logical 0. 1 = Comparator output is logical 1.
[0] CMPEN	**Comparator Enable** 0 = Comparator is disabled. 1 = Comparator is enabled.

Hex Addresses: F91–FBF

This address range is reserved.

Interrupt Controller

For more information about these Oscillator Control registers, see the Interrupt Controller chapter on page 48.

Hex Address: FC0

Table 239. Interrupt Request 0 Register (IRQ0)

Bit	7	6	5	4	3	2	1	0
Field	PWMI	FLTI	ADCI	CMPI	T0I	U0RXI	U0TXI	SPII
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	FC0H							

Bit	Description
[7] PWMI	**PWM Timer Interrupt Request** 0 = No interrupt request is pending for the PWM. 1 = An interrupt request from the PWM is awaiting service.
[6] FLTI	**Fault Interrupt Request** The fault interrupt is generated in the PWM module and originates from the Fault0 pin, Fault1 pin or the Comparator output. An interrupt enable for each of these sources exists in the PWM module. 0 = No Fault interrupt request is pending. 1 = A Fault interrupt request is awaiting service.
[5] ADCI	**ADC Interrupt Request** 0 = No interrupt request is pending for the Analog-to-Digital Converter. 1 = An interrupt request from the Analog-to-Digital Converter is awaiting service.
[4] CMPI	**Comparator Interrupt Request** 0 = No interrupt request is pending for the Comparators. 1 = An interrupt request from the Comparators is awaiting service.
[3] T0I	**Timer 0 Interrupt Request** 0 = No interrupt request is pending for Timer 0. 1 = An interrupt request from Timer 0 is awaiting service.
[2] U0RXI	**UART 0 Receiver Interrupt Request** 0 = No interrupt request is pending for the UART 0 receiver. 1 = An interrupt request from the UART 0 receiver is awaiting service.
[1] U0TXI	**UART 0 Transmitter Interrupt Request** 0 = No interrupt request is pending for the UART 0 transmitter. 1 = An interrupt request from the UART 0 transmitter is awaiting service.
[0] SPII	**SPI Interrupt Request** 0 = No interrupt request is pending for the SPI. 1 = An interrupt request from the SPI is awaiting service.

Hex Address: FC1

Table 240. IRQ0 Enable High Bit Register (IRQ0ENH)

Bit	7	6	5	4	3	2	1	0
Field	PWMENH	FLTENH	ADCENH	CMPENH	T0ENH	U0RENH	U0TENH	SPIENH
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	FC1H							

Hex Address: FC2

Table 241. IRQ0 Enable Low Bit Register (IRQ0ENL)

Bit	7	6	5	4	3	2	1	0
Field	PWMENL	FLTENL	ADCENL	CMPENL	T0ENL	U0RENL	U0TENL	SPIENL
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	FC2H							

Hex Address: FC3

Table 242. Interrupt Request 1 Register (IRQ1)

Bit	7	6	5	4	3	2	1	0
Field	I2CI	Reserved	PC0I	PBI	PA73I	PA62I	PA51I	PA40I
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	FC3H							

Bit	Description
[7] I2CI	**I^2C Interrupt Request** 0 = No interrupt request is pending for I2C. 1 = An interrupt request from I2C is awaiting service.
[5] PC0I	**PC0 Interrupt Request** Logic in the Port C GPIO module selects either the rising or falling edge. 0 = No interrupt request is pending for PC0. 1 = An interrupt request from PC0 is awaiting service.

Bit	Description (Continued)
[4] PBI	**PB3–PB0 Interrupt Request** 0 = No interrupt request is pending for any PB3–PB0. 1 = An interrupt request from PB3–PB0 is awaiting service.
[3] PA73I	**PA7 or PA3 Interrupt Request** Logic in the Port A GPIO module selects either PA7 or PA3 and either rising or falling edge. 0 = No interrupt request is pending for PA7 or PA3 1 = An interrupt request from PA7 or PA3 is awaiting service.
[2] PA62I	**PA6 or PA2 Interrupt Request** Logic in the Port A GPIO module selects either PA6 or PA2 and either rising or falling edge. 0 = No interrupt request is pending for PA6 or PA2 1 = An interrupt request from PA6 or PA2 is awaiting service.
[1] PA51I	**PA5 or PA1 Interrupt Request** Logic in the Port A GPIO module selects either PA5 or PA1 and either rising or falling edge. 0 = No interrupt request is pending for PA5 or PA1 1 = An interrupt request from PA5 or PA1 is awaiting service.
[0] PA40I	**PA4 or PA0 Interrupt Request** Logic in the Port A GPIO module selects either PA4 or PA0 and either rising or falling edge. 0 = No interrupt request is pending for PA4 or PA0 1 = An interrupt request from PA4 or PA0 is awaiting service.

Hex Address: FC4

Table 243. IRQ1 Enable High Bit Register (IRQ1ENH)

Bit	7	6	5	4	3	2	1	0
Field	I2CENH	Reserved	PC0ENH	PBENH	PA73ENH	PA62ENH	PA51ENH	PA40ENH
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	FC4H							

Bit	Description
[7] I2CENH	**I^2C Interrupt Request Enable High Bit**
[6]	**Reserved** This bit is reserved and must be programmed to 0.
[5] PC0ENH	**Port C0Interrupt Request Enable High Bit**
[4] PBENH	**Port B[3:0] Interrupt Request Enable High Bit**

Bit	Description (Continued)
[3] PA73ENH	**Port A73 Interrupt Request Enable High Bit**
[2] PA62ENH	**Port A62 Interrupt Request Enable High Bit**
[1] PA51ENH	**Port A51 Interrupt Request Enable High Bit**
[0] PA40ENH	**Port A40 Interrupt Request Enable High Bit**

Hex Address: FC5

Table 244. IRQ1 Enable Low Bit Register (IRQ1ENL)

Bit	7	6	5	4	3	2	1	0
Field	I2CENL	Reserved	PC0ENL	PBENL	PA73ENL	PA62ENL	PA51ENL	PA40ENL
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	FC5H							

Bit	Description
[7] I2CENL	**I^2C Interrupt Request Enable Low Bit**
[6]	**Reserved** This bit is reserved and must be programmed to 0.
[5] PC0ENL	**Port C0 Interrupt Request Enable Low Bit**
[4] PBENL	**Port B[3:0] Interrupt Request Enable Low Bit**
[3] PA73ENL	**Port A73 Interrupt Request Enable Low Bit**
[2] PA62ENL	**Port A62 Interrupt Request Enable Low Bit**
[1] PA51ENL	**Port A51 Interrupt Request Enable Low Bit**
[0] PA40ENL	**Port A40 Interrupt Request Enable Low Bit**

Hex Addresses: FC9–FCE

This address range is reserved.

Hex Address: FCF

Table 245. 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
R/W	R/W	R	R	R	R	R	R	R
Address	FCFH							

GPIO Port A

For more information about these Oscillator Control registers, see the General-Purpose Input/Output chapter on page 33.

Hex Address: FD0

Table 246. Port A–C 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							
Address	FD0H, FD4H, FD8H							

Hex Address: FD1

Table 247. Port A–C Control Registers (P*x*CTL)

Bit	7	6	5	4	3	2	1	0
Field	PCTL							
RESET	00H							
R/W	R/W							
Address	FD1H, FD5H, FD9H							

Hex Address: FD2

Table 248. Port A-C Input Data Registers (PxIN)

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	FD2H, FD6H, FDAH							

Note: X = Undefined.

Bit	Description
[7:0] PIN*x*	**Port Input Data** 0 = Input data is a logical 0 (Low). 1 = Input data is a logical 1 (High).

Note: x indicates register bits in the range 7–0.

Hex Address: FD3

Table 249. Port A-C 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	FD3H, FD7H, FDBH							

Bit	Description
[7:0] POUTx	**Port Output Data x** 0 = Drive is a logical 0 (Low). 1 = Drive is 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.

Note: x indicates register bits in the range 7–0.

GPIO Port B

For more information about these Oscillator Control registers, see the General-Purpose Input/Output chapter on page 33.

Hex Address: FD4

Table 250. Port A–C 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							
Address	FD0H, FD4H, FD8H							

Hex Address: FD5

Table 251. Port A–C Control Registers (P*x*CTL)

Bit	7	6	5	4	3	2	1	0
Field	PCTL							
RESET	00H							
R/W	R/W							
Address	FD1H, FD5H, FD9H							

Hex Address: FD6

Table 252. Port A-C Input Data Registers (PxIN)

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	FD2H, FD6H, FDAH							

Note: X = Undefined.

Bit	Description
[7:0] PIN*x*	**Port Input Data** 0 = Input data is a logical 0 (Low). 1 = Input data is a logical 1 (High).

Note: x indicates register bits in the range 7–0.

Hex Address: FD7

Table 253. Port A-C 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	FD3H, FD7H, FDBH							

Bit	Description
[7:0] POUTx	**Port Output Data x** 0 = Drive is a logical 0 (Low). 1 = Drive is 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.

Note: x indicates register bits in the range 7–0.

GPIO Port C

For more information about these Oscillator Control registers, see the General-Purpose Input/Output chapter on page 33.

Hex Address: FD8

Table 254. Port A–C 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							
Address	FD0H, FD4H, FD8H							

Hex Address: FD9

Table 255. Port A–C Control Registers (P*x*CTL)

Bit	7	6	5	4	3	2	1	0
Field	PCTL							
RESET	00H							
R/W	R/W							
Address	FD1H, FD5H, FD9H							

Hex Address: FDA

Table 256. Port A-C Input Data Registers (PxIN)

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	FD2H, FD6H, FDAH							

Note: X = Undefined.

Bit	Description
[7:0] PIN*x*	**Port Input Data** 0 = Input data is a logical 0 (Low). 1 = Input data is a logical 1 (High).

Note: x indicates register bits in the range 7–0.

Hex Address: FDB

Table 257. Port A-C 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	FD3H, FD7H, FDBH							

Bit	Description
[7:0] POUTx	**Port Output Data x** 0 = Drive is a logical 0 (Low). 1 = Drive is 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.

Note: x indicates register bits in the range 7–0.

Reset and Watchdog Timer

For more information about these Reset and Watchdog Timer registers, see the Reset and Stop Mode Recovery chapter on page 22 and the the Watchdog Timer chapter on page 60.

Hex Address: FF0

Table 258. Reset Status and Control Register (RSTSTAT)

Bit	7	6	5	4	3	2	1	0
Field	POR	STOP	WDT	EXT	FLT	Reserved		FLTSEL
RESET	See Table 10.							0
R/W	R	R	R	R	R	R		R/W
Address	FF0H							

Hex Address: FF1

This address is reserved.

Hex Address: FF2

Table 259. Watchdog Timer Reload High Byte Register (WDTH)

Bit	7	6	5	4	3	2	1	0
Field	WDTH							
RESET	04H							
R/W	R/W*							
Address	FF2H							
Note: *R/W = Read returns the current WDT count value; write sets the appropriate reload value.								

Hex Address: FF3

Table 260. Watchdog Timer Reload Low Byte Register (WDTL)

Bit	7	6	5	4	3	2	1	0
Field	WDTL							
RESET	00H							
R/W	R/W*							
Address	FF3H							
Note: *R/W = Read returns the current WDT count value; write sets the appropriate reload value.								

Hex Addresses: FF4–FF5

This address range is reserved.

Hex Address: FF6

Table 261. Trim Bit Address Register (TRMADR)

Bit	7	6	5	4	3	2	1	0
Field	TRMADR							
RESET	00H							
R/W	R/W							
Address	FF6H							

Bit	Description
[7:0] TRMADR	**00–1FH = Trim Bit Address Register**

Hex Address: FF7

Table 262. Trim Bit Data Register (TRMDR)

Bit	7	6	5	4	3	2	1	0
Field	TRMDR							
RESET	00H							
R/W	R/W							
Address	FF7H							

Bit	Description
[7:0] TRMDR	**00–FFH = Trim Bit Data Register**

Flash Memory Controller

For more information about these Flash Memory Control registers, see the Program Memory chapter on page 209.

Hex Address: FF8

Table 263. Flash Control Register (FCTL)

Bit	7	6	5	4	3	2	1	0
Field	FCMD							
RESET	00H							
R/W	W							
Address	FF8H							

Bit	Description
[7:0] FCMD	**Flash Command** 73H = First unlock command. 8CH = Second unlock command. 95H = Page erase command. 63H = Mass erase command. 5EH = Flash Sector Protect Register select. All other commands or any command out of sequence locks the Flash Controller.

Table 264. Flash Status Register (FSTAT)

Bit	7	6	5	4	3	2	1	0
Field	Reserved		FSTAT					
RESET	00B		00_0000B					
R/W	R		R					
Address	FF8H							

Bit	Description
[7:6]	**Reserved** These bits are reserved and must be programmed to 00.
[5:0] FSTAT	**Flash Controller Status** 00_0000 = Flash Controller locked. 00_0001 = First unlock command received. 00_0010 = Second unlock command received. 00_0011 = Flash Controller unlocked. 00_0100 = Flash Sector Protect Register selected. 0_1xxx = Program operation in progress. 01_0xxx = Page erase operation in progress. 10_0xxx = Mass erase operation in progress.

Hex Address: FF9

Table 265. Flash Sector Protect Register (FPROT)

Bit	7	6	5	4	3	2	1	0
Field	SECT7	SECT6	SECT5	SECT4	SECT3	SECT2	SECT1	SECT0
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	FF9H							

Note: *R/W = Register is accessible for Read operations and can only be written to 1 via user code.

Bit	Description
[7:0] SECT*n*	**Sector Protect** 0 = Sector *n* can be programmed or erased from user code. 1 = Sector *n* is protected and cannot be programmed or erased from user code.

Hex Address: FFA

Table 266. Flash Frequency High Byte Register (FFREQH)

Bit	7	6	5	4	3	2	1	0
Field	FFREQH							
RESET	00H							
R/W	R/W							
Address	FFAH							

Hex Address: FFB

Table 267. Flash Frequency Low Byte Register (FFREQL)

Bit	7	6	5	4	3	2	1	0
Field	FFREQL							
RESET	00H							
R/W	R/W							
Address	FFBH							

eZ8 CPU

Refer to the eZ8 CPU Core User Manual (UM0128), which is available for download at www.zilog.com.

Index

Numerics

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W

X

Z

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 or consider participating in the Zilog Forum at http://zilog.com/forum.

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.