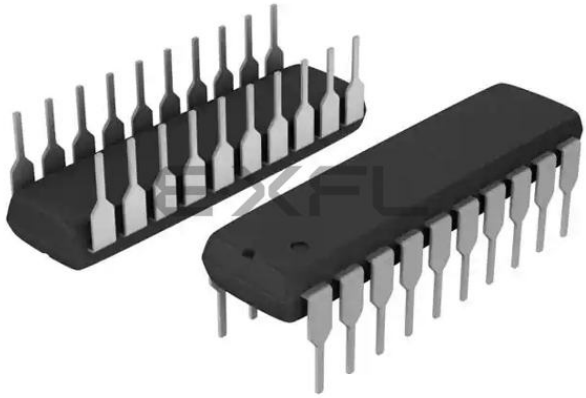

Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	eZ8
Core Size	8-Bit
Speed	20MHz
Connectivity	I²C, IrDA, LINbus, UART/USART
Peripherals	Brown-out Detect/Reset, LED, LVD, POR, PWM, Temp Sensor, WDT
Number of I/O	17
Program Memory Size	24KB (24K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	3K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	A/D 7x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 105°C (TA)
Mounting Type	Through Hole
Package / Case	20-DIP (0.300", 7.62mm)
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/zilog/z8f2480ph020eg

need to use this on-chip Program RAM to shadow Interrupt Service Routines (ISR). For details, see the PRAM_M section on page 278.

3.2. Program Memory

The eZ8 CPU supports 64KB of Program Memory address space. The F1680 Series MCU contains 8KB to 24KB of on-chip Flash memory in the Program Memory address space, depending on the device.

In addition, the F1680 Series MCU contains up to 1KB of on-chip Program RAM. The Program RAM is mapped in the Program Memory address space beyond the on-chip Flash memory. The Program RAM is entirely under user control and is meant to store interrupt service routines of high-frequency interrupts. Since interrupts bring the CPU out of low-power mode, it is important to ensure that interrupts that occur very often use as low a current as possible. For battery operated systems, Program RAM based handling of high-frequency interrupts provides power savings by keeping the Flash block disabled. Program RAM (PRAM) is optimized for low-current operation and can be easily boot-strapped with interrupt code at power up.

Reading from Program Memory addresses present outside the available Flash memory and PRAM addresses returns FFH. Writing to these unimplemented Program Memory addresses produces no effect. Table 6 describes the Program Memory maps for the F1680 Series MCU.

Table 6. F1680 Series MCU Program Memory Maps

Program Memory Address (Hex)	Function
Z8F2480 Device	
0000–0001	Flash option bits
0002–0003	Reset vector
0004–0005	WDT interrupt vector
0006–0007	Illegal instruction trap
0008–0037	Interrupt vectors*
0038–003D	Oscillator fail traps*
003E–5FFF	Program Flash
E000–E3FF	1KB PRAM

Note: *See Table 36 on page 69 for a list of interrupt vectors and traps.

Table 8. Register File Address Map (Continued)

Address (Hex)	Register Description	Mnemonic	Reset (Hex)[1]	Page #
F17	Timer 2 Control 1	T2CTL1	00	113
F28	Timer 2 PWM1 High Byte	T2PWM1H	00	111
F29	Timer 2 PWM1 Low Byte	T2PWM1L	00	111
F2A	Timer 2 Control 2	T2CTL2	00	117
F2B	Timer 2 Status	T2STA	00	118
F2E	Timer 2 Noise Filter Control	T2NFC	00	119
F2F–F3F	Reserved	—	XX	
LIN UART 0				
F40	LIN UART0 Transmit Data	U0TXD	XX	163
	LIN UART0 Receive Data	U0RXD	XX	164
F41	LIN UART0 Status 0—Standard UART Mode	U0STAT0	0000011Xb	165
	LIN UART0 Status 0—LIN Mode	U0STAT0	00000110b	166
F42	LIN UART0 Control 0	U0CTL0	00	170
F43	LIN UART0 Control 1—Multiprocessor Control	U0CTL1	00	172
	LIN UART0 Control 1—Noise Filter Control	U0CTL1	00	174
	LIN UART0 Control 1—LIN Control	U0CTL1	00	175
F44	LIN UART0 Mode Select and Status	U0MDSTAT	00	168
F45	UART0 Address Compare	U0ADDR	00	177
F46	UART0 Baud Rate High Byte	U0BRH	FF	177
F47	UART0 Baud Rate Low Byte	U0BRL	FF	178
LIN UART 1				
F48	LIN UART1 Transmit Data	U1TXD	XX	163
	LIN UART1 Receive Data	U1RXD	XX	164
F49	LIN UART1 Status 0—Standard UART Mode	U1STAT0	0000011Xb	165
	LIN UART1 Status 0—LIN Mode	U1STAT0	00000110b	166
F4A	LIN UART1 Control 0	U1CTL0	00	170

Notes:
1. XX=Undefined.
2. The Reserved space can be configured as General-Purpose Register File RAM depending on the user option bits (see the User Option Bits chapter on page 277) and the on-chip PRAM size (see the Ordering Information chapter on page 372). If the PRAM is programmed as General-Purpose Register File RAM on Reserved space, the starting address always begins immediately after the end of General-Purpose Register File RAM.

Table 19. Port Alternate Function Mapping, 40-/44-Pin Parts[1,2]

Port	Pin	Mnemonic	Alternate Function Description	Alternate Function Set Register AFS1
Port A	PA0	T0IN/$\overline{\text{T0OUT}}$	Timer 0 Input/Timer 0 Output Complement	AFS1[0]: 0
		Reserved		AFS1[0]: 1
	PA1	T0OUT	Timer 0 Output	AFS1[1]: 0
		Reserved		AFS1[1]: 1
	PA2	DE0	UART 0 Driver Enable	AFS1[2]: 0
		Reserved		AFS1[2]: 1
	PA3	$\overline{\text{CTS0}}$	UART 0 Clear to Send	AFS1[3]: 0
		Reserved		AFS1[3]: 1
	PA4	RXD0/IRRX0	UART 0/IrDA 0 Receive Data	AFS1[4]: 0
				AFS1[4]: 1
	PA5	TXD0/IRTX0	UART 0/IrDA 0 Transmit Data	AFS1[5]: 0
				AFS1[5]: 1
	PA6	T1IN/$\overline{\text{T1OUT}}$	Timer 1 Input/Timer 1 Output Complement	AFS1[6]: 0
		Reserved		AFS1[6]: 1
	PA7	T1OUT	Timer 1 Output	AFS1[7]: 0
		Reserved		AFS1[7]: 1

Notes:
1. Because there are at most two choices of alternate functions for some pins in Ports A–C, the Alternate Function Set Register (AFS2) is implemented but not used to select the function. The alternate function selection must also be enabled, as described in the Port A–E Alternate Function Subregisters section on page 61.
2. Because there is only one alternate function for each Port D and Port E pin, the Alternate Function Set registers are not implemented for Ports D and E. Enabling the alternate function selections automatically enables the associated alternate function, as described in the Port A–E Alternate Function Subregisters section on page 61.
3. This timer function is only available in the 44-pin package; its alternate functions are reserved in the 40-pin package.

Table 19. Port Alternate Function Mapping, 40-/44-Pin Parts[1,2] (Continued)

Port	Pin	Mnemonic	Alternate Function Description	Alternate Function Set Register AFS1
Port D	PD0	$\overline{\text{RESET}}$	External Reset	N/A
	PD1	C1INN	Comparator 1 Input (N)	
	PD2	C1INP	Comparator 1 Input (P)	
	PD3	$\overline{\text{CTS1}}$/C1OUT	UART 1 Clear to Send or Comparator 1 Output	
	PD4	RXD1/IRRX1	UART 1/IrDA 1 Receive Data	
	PD5	TXD1/IRTX1	UART 1/IrDA 1 Transmit Data	
	PD6	DE1	UART 1 Driver Enable	
	PD7	C0OUT	Comparator 0 Output	
Port E	PE0	T4IN[3]		N/A
		Reserved		
	PE1	SCL	I^2C Serial Clock	
		Reserved		
	PE2	SDA	I^2C Serial Data	
		Reserved		
	PE3	T4CHA[3]		
		Reserved		
	PE4	T4CHB[3]		
		Reserved		
	PE5	T4CHC[3]		
		Reserved		
	PE6	T4CHD[3]		
		Reserved		

Notes:
1. Because there are at most two choices of alternate functions for some pins in Ports A–C, the Alternate Function Set Register (AFS2) is implemented but not used to select the function. The alternate function selection must also be enabled, as described in the Port A–E Alternate Function Subregisters section on page 61.
2. Because there is only one alternate function for each Port D and Port E pin, the Alternate Function Set registers are not implemented for Ports D and E. Enabling the alternate function selections automatically enables the associated alternate function, as described in the Port A–E Alternate Function Subregisters section on page 61.
3. This timer function is only available in the 44-pin package; its alternate functions are reserved in the 40-pin package.

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

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

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

Example 2. A good coding style that avoids lost interrupt requests:

```
ANDX IRQ0, MASK
```

8.3.4. Software Interrupt Assertion

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

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

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

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

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

Example 4. A good coding style that avoids lost interrupt requests:

```
ORX IRQ0, MASK
```

8.4. Interrupt Control Register Definitions

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

Table 54 provides an example initialization sequence for configuring Timer 0 in DEMODULATION Mode and initiating operation.

Table 54. DEMODULATION Mode Initialization Example

Register	Value	Comment
T0CTL0	C0H	TMODE[3:0] = 1100B selects DEMODULATION Mode. TICONFIG[1:0] = 10B enables interrupt only on Capture events. CSC = 0 selects the Timer Input from the GPIO pin. PWMD[2:0] = 000B has no effect. INPCAP = 0 has no effect. TEN = 0 disables the timer. PRES[2:0] = 000B sets prescaler to divide by 1. TPOLHI,TPOL = 10 enables trigger and Capture on both rising and falling edges of Timer Input. TCLKS = 1 enables 32kHz peripheral clock as timer clock source
T0CTL1	04H	
T0CTL2	11H	
T0H	00H	Timer starting value = 0001H.
T0L	01H	
T0RH	ABH	Timer reload value = ABCDH
T0RL	CDH	
T0PWM0H	00H	Initial PWM0 value = 0000H
T0PWM0L	00H	
T0PWM1H	00H	Initial PWM1 value = 0000H
T0PWM1H	00H	
T0NFC	C0H	NFEN = 1 enables noise filter NFCTL = 100B enables 8-bit up/down counter
PAADDR	02H	Selects Port A Alternate Function control register.
PACTL[1:0]	11B	PACTL[0] enables Timer 0 Input alternate function. PACTL[1] enables Timer 0 Output alternate function.
IRQ0ENH[5]	0B	Disables the Timer 0 interrupt.
IRQ0ENL[5]	0B	
T0CTL1	84H	TEN = 1 enables the timer. All other bits remain in their appropriate settings.

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

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

Bit	Description (Continued)
[3:1] PWMD	**PWM Delay Value** This field is a programmable delay to control the number of timer clock cycles time delay before the Timer Output and the Timer Output Complement is forced to their active state. 000 = No delay 001 = 2 cycles delay 010 = 4 cycles delay 011 = 8 cycles delay 100 = 16 cycles delay 101 = 32 cycles delay 110 = 64 cycles delay 111 = 128 cycles delay
[0] INPCAP	Input Capture Event This bit indicates if the last timer interrupt is due to a Timer Input Capture Event. 0 = Previous timer interrupt is not a result of Timer Input Capture Event. 1 = Previous timer interrupt is a result of Timer Input Capture Event.

9.3.5.2. Timer 0–2 Control 1 Register

The Timer 0–2 Control 1 (TxCTL1) registers enable and disable the timers, set the prescaler value and determine the timer operating mode. See Table 64.

Table 64. Timer 0–2 Control 1 Register (TxCTL1)

Bit	7	6	5	4	3	2	1	0
Field	TEN	TPOL	PRES			TMODE		
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F07H, F0FH, F17H							

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

Bit	Description (Continued)
[5:3] PRES	**Prescale Value** The timer input clock is divided by 2PRES, 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 the TxCTL0 Register, determines the operating mode of the timer. TMODE[3:0] selects among the following modes: 0000 = ONE-SHOT Mode 0001 = CONTINUOUS Mode 0010 = COUNTER Mode 0011 = PWM SINGLE OUTPUT Mode 0100 = CAPTURE Mode 0101 = COMPARE Mode 0110 = GATED Mode 0111 = CAPTURE/COMPARE Mode 1000 = PWM DUAL OUTPUT Mode 1001 = CAPTURE RESTART Mode 1010 = COMPARATOR COUNTER Mode 1011 = TRIGGERED ONE-SHOT Mode 1100 = DEMODULATION Mode

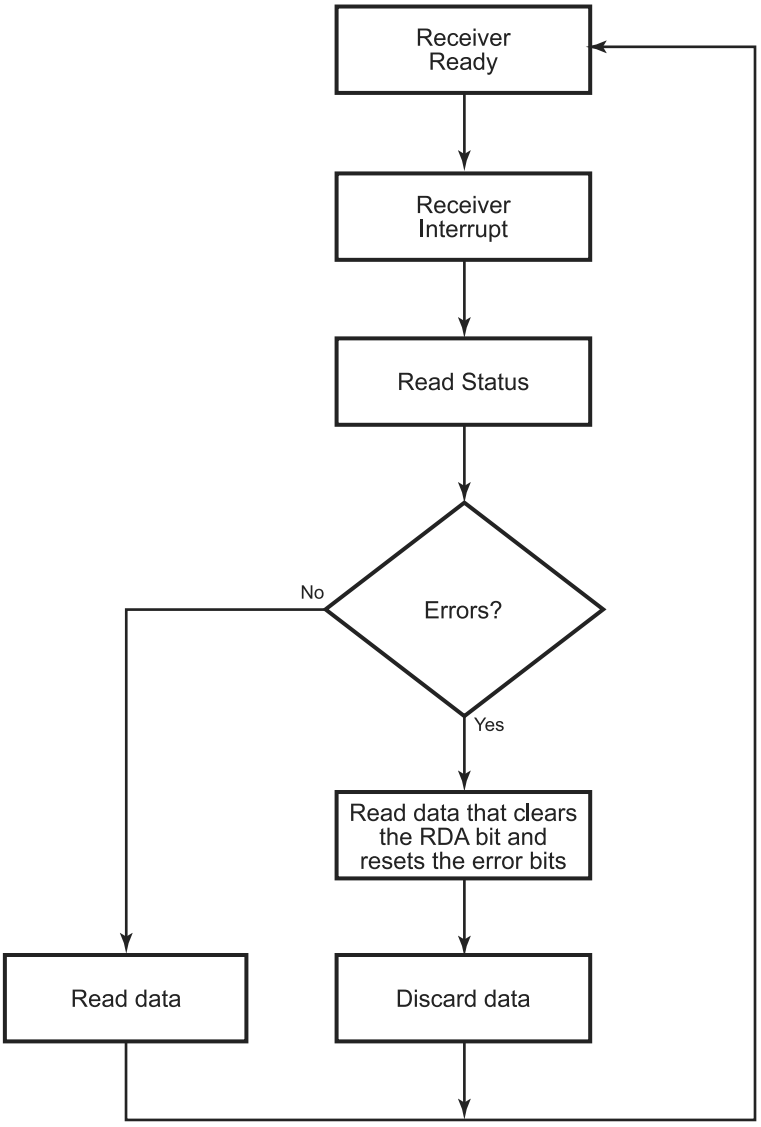

Figure 24. LIN-UART Receiver Interrupt Service Routine Flow

12.1.11.5. 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 action allows the Baud Rate Generator to function as an additional counter, if the LIN-UART receiver functionality is not employed. The transmitter can be enabled in this mode.

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.

14.2.4. Internal Voltage Reference Generator

The Internal Voltage Reference Generator provides the voltage, VR2, for the RBUF. VR2 is 1.6V.

14.2.5. Calibration and Compensation

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

14.3. ADC Control Register Definitions

The registers that control analog-to-digital conversion functions are defined in this section.

14.3.1. ADC Control Register 0

The ADC Control Register 0, shown in Table 101, initiates the A/D conversion and provides ADC status information.

Table 101. ADC Control Register 0 (ADCCTL0)

Bits	7	6	5	4	3	2	1	0
Field	START	INTREF_SEL	REFEN	ADCEN	ANAIN[3:0]			
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 Position	Value (H)	Description
[7] START	0	**ADC Start/Busy** Writing a 0 has no effect. Reading a 0 indicates the ADC is available to begin a conversion.
	1	Writing a 1 starts a conversion. Reading a 1 indicates that a conversion is currently in progress.
[6] INTREF_SEL	0	Select 1.6 V as internal reference.
	1	Select AVDD as internal reference.

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 the not acknowledge case.

17.2.5.5. Master Write Transaction with a 10-Bit Address

Figure 44 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 44. Data Transfer Format—Master Write Transaction with a 10-Bit Address

14. The software responds by writing the data to be written out to the I^2C Control Register.
15. The I^2C controller shifts out the remainder of the second byte of the slave address (or ensuring data bytes, if looping) via the SDA signal.
16. The I^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, see the second paragraph of Step 11.
17. The I^2C 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^2C Control Register.
20. The I^2C controller completes transmission of the data on the SDA signal.
21. The I^2C controller sends a stop condition to the I^2C bus.

Note: If the slave responds with a Not Acknowledge during the transfer, the I^2C controller asserts the NCKI bit, sets the ACKV bit, clears the ACK bit in the I^2C 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.

17.2.5.6. Master Read Transaction with a 7-Bit Address

Figure 45 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 45. Data Transfer Format—Master Read Transaction with a 7-Bit Address

Observe the following steps for a Master Read operation to a 7-bit addressed slave:

1. The software initializes the MODE field in the I^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 the I^2C 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^2C Control Register.

17.3.2. I²C Interrupt Status Register

The read-only I²C Interrupt Status Register, shown in Table 120, indicates the cause of any current I²C interrupt and provides status of the I²C controller. When an interrupt occurs, one or more of the TDRE, RDRF, SAM, ARBLST, SPRS or NCKI bits is set. The GCA and RD bits do not generate an interrupt but rather provide status associated with the SAM bit interrupt.

Table 120. I²C Interrupt Status Register (I2CISTAT = F51H)

Bits	7	6	5	4	3	2	1	0
Field	TDRE	RDRF	SAM	GCA	RD	ARBLST	SPRS	NCKI
Reset	1	0	0	0	0	0	0	0
R/W	R	R	R	R	R	R	R	R
Address	F51H							

Bit	Description
[7] TDRE	**Transmit Data Register Empty** When the I²C controller is enabled, this bit is 1 when the I²C Data Register is empty. When set, this bit causes the I²C controller to generate an interrupt, except when the I²C controller is shifting in data during the reception of a byte or when shifting an address and the RD bit is set. This bit clears by writing to the I2CDATA Register.
[6] RDRF	**Receive Data Register Full** This bit is set = 1 when the I²C controller is enabled and the I²C controller has received a byte of data. When asserted, this bit causes the I²C controller to generate an interrupt. This bit clears by reading the I2CDATA Register.
[5] SAM	**Slave Address Match** This bit is set = 1 if the I²C controller is enabled in SLAVE Mode and an address is received that matches the unique slave address or General Call Address (if enabled by the GCE bit in the I²C Mode Register). In 10-bit addressing mode, this bit is not set until a match is achieved on both address bytes. When this bit is set, the RD and GCA bits are also valid. This bit clears by reading the I2CISTAT Register.
[4] GCA	**General Call Address** This bit is set in SLAVE Mode when the General Call Address or Start byte is recognized (in either 7 or 10 bit SLAVE Mode). The GCE bit in the I²C Mode Register must be set to enable recognition of the General Call Address and Start byte. This bit clears when IEN = 0 and is updated following the first address byte of each SLAVE Mode transaction. A General Call Address is distinguished from a Start byte by the value of the RD bit (RD = 0 for General Call Address, 1 for Start byte).
[3] RD	**Read** This bit indicates the direction of transfer of the data. It is set when the Master is reading data from the Slave. This bit matches the least-significant bit of the address byte after the start condition occurs (for both MASTER and SLAVE modes). This bit clears when IEN = 0 and is updated following the first address byte of each transaction.

Table 126. 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	Nine substates, one for each data bit and one for the Acknowledge.
1001	Slave Receive Data	Nine substates, one for each data bit and one for the Acknowledge.
1010	Slave Receive Addr1	Slave receiving first address byte (7- and 10-bit addressing) Nine substates, one for each address bit and one for the Acknowledge.
1011	Slave Receive Addr2	Slave Receiving second address byte (10-bit addressing) nine substates, one for each address bit and one for the Acknowledge.
1100	Master Transmit Data	Nine substates, one for each data bit and one for the Acknowledge.
1101	Master Receive Data	Nine substates, one for each data bit and one for the Acknowledge.
1110	Master Transmit Addr1	Master sending first address byte (7- and 10-bit addressing) nine substates, one for each address bit and one for the Acknowledge.
1111	Master Transmit Addr2	Master sending second address byte (10-bit addressing) nine substates, one for each address bit and one for the Acknowledge.

Table 127. 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.

17.3.7. I²C Slave Address Register

The I²C Slave Address Register, shown in Table 129, provides control over the lower order address bits used in 7 and 10 bit slave address recognition.

Table 129. I²C Slave Address Register (I2CSLVAD = 57H)

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

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

```
DBG ← Size[7:0]
DBG → 1-65536 data bytes
```

Read Program Memory CRC (0EH). 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 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 Program memory.

```
DBG ← 0EH
DBG → CRC[15:8]
DBG → CRC[7:0]
```

Step Instruction (10H). The Step Instruction command steps one assembly instruction at the current Program Counter (PC) 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 (11H). The Stuff Instruction command steps one assembly instruction and allows specification of the first byte of the instruction. The remaining 0-4 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 breakpoint. If the device is not in DEBUG mode or the Read Protect option bit is enabled, the OCD ignores this command.

```
DBG ← 11H
DBG ← opcode[7:0]
```

Execute Instruction (12H). The Execute Instruction command allows sending an entire instruction to be executed to the eZ8 CPU. This command can also step over breakpoints. 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 opcode
```

Write Line Control Register (18H). The Write Line Control Register command writes the data that follows to the Line Control Register.

```
DBG ← 18H
DBG ← LCR[7:0]
```

Read Line Control Register (19H). The Read Line Control Register command returns the current value in the Line Control Register.

```
DBG ← 19H
DBG → LCR[7:0]
```

23.4.3. Line Control Register

The Line Control Register, shown in Table 166, is used to configure the output driver characteristics during transmission. This register is only used in high-speed implementations.

Table 166. OCD Line Control Register (OCDLCR)

Bit	7	6	5	4	3	2	1	0
Field	Reset		NBTX	NBEN	TXFC	TXDH	TXD	TXHD
Reset	00		0	0	0	0	0	0
R/W	R		R/W	R/W	R/W	R/W	R/W	R/W

Bit	Description
[7:6] Reset	**Reset**
[5] NBTX	**Nine Bit Transmit** This control bit sets the polarity of the ninth bit when nine bit mode is enabled. 0 = Ninth bit is zero. 1 = Ninth bit is one.
[4] NBEN	**Nine Bit Enable** This control bit enables nine-bit mode; it is useful when transmit flow control using remote start bit is enabled to detect valid characters. 0 = Nine Bit mode disabled. 1 = Nine Bit mode enabled.
[3] TXFC	**Transmit Flow Control** 0 = Transmit Flow Control disabled. 1 = Transmit Flow Control using remote start bit.
[2] TXDH	**Transmit Drive High** 0 = Pin is not driven High during 0 to 1 transitions. 1 = Pin is driven High during 0 to 1 transitions.
[1] TXD	**Transmit Drive** 0 = Pin is only driven Low during transmission (Open-Drain). 1 = Pin is always driven during transmission.
[0] TXHD	**Transmit High Drive Strength** 0 = Pin output driver is Low drive strength. 1 = Pin output driver is High drive strength.

Table 201. Low Voltage Detect Electrical Characteristics (Continued)

		T_A = 0°C to +70°C T_A = –40°C to +105°C V_{DD} = 1.8 to 3.6 V				
Symbol	**Parameter**	**Min**	**Typ**	**Max**	**Units**	**Conditions**
V_{TH_PRO}	Detected Source Voltage for Flash Protection	2.4	2.5	2.6	V	
T_{DELAY}	Delay from source voltage falling lower than V_{TP} to I_{VD_OUT} output logic High	50	1000	–	ns	

Note: [1] V_{TP} is a user-set threshold voltage to be detected.

Table 202. Crystal Oscillator Characteristics

		T_A = 0°C to +70°C T_A = –40°C to +105°C							
		V_{DD} = 2.7 to 3.6V			V_{DD} = 1.8 to 2.7V				
Symbol	**Parameter**	**Min**	**Typ**	**Max**	**Min**	**Typ**	**Max**	**Units**	**Conditions**
I_{DD}XTAL	Crystal Oscillator Active Supply Current	–	–	500	–	–	300	µA	
I_{DDQ}XTAL	Crystal Oscillator Quiescent Current	–	5	–	–	5	–	nA	
S_{CLK}	Clk_out State in Crystal Disable	1	1	1	1	1	1		
F_{XTAL}	External Crystal Oscillator Frequency	1	–	20	1	–	20	MHz	See Figure 74.
T_{SET}	Startup Time After Enable	–	10,000	30,000	–	10,000	30,000	Cycle	
	Clk_out Duty Cycle	40	50	60	40	50	60	%	
	Clk_out Jitter	–	1	–	–	1	–	%	

Table 203. Low Power 32 kHz Secondary Oscillator Characteristics

		T_A = 0°C to +70°C T_A = –40°C to +105°C							
		V_{DD} = 2.7 to 3.6 V			V_{DD} = 1.8 to 2.7 V				
Symbol	**Parameter**	**Min**	**Typ**	**Max**	**Min**	**Typ**	**Max**	**Units**	**Conditions**
$I_{DD}XTAL2$	32 kHz Secondary Oscillator Active Current	–	–	20	–	–	10	µA	
$I_{DDQ}XTAL2$	32 kHz Secondary Oscillator Quiescent Current	–	5	–	–	5	–	nA	
S_{CLK}	Clk_out State in Crystal Disable	1	1	1	1	1	1		
F_{XTAL2}	External Crystal Oscillator Frequency	–	32.768	–	–	32.768	–	kHz	
T_{SET}	Startup Time After Enable	–	400	1000	–	400	1000	mS	
	Clk_out Duty Cycle	40	50	60	40	50	60	%	
	Clk_out Jitter	–	1	–	–	1	–	%	

H

I