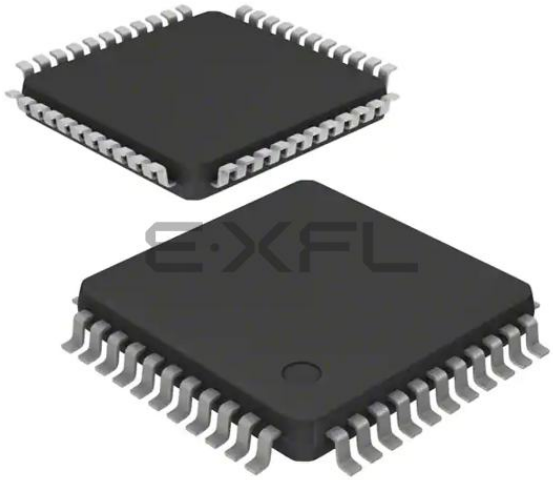

Welcome to E-XFL.COM

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, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, LED, LVD, POR, PWM, Temp Sensor, WDT
Number of I/O	37
Program Memory Size	16KB (16K 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 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 105°C (TA)
Mounting Type	Surface Mount
Package / Case	44-LQFP
Supplier Device Package	44-LQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/zilog/z8f1680an020eg

Table 8. Register File Address Map (Continued)

Address (Hex)	Register Description	Mnemonic	Reset (Hex)[1]	Page #
Analog-to-Digital Converter (ADC)				
F70	ADC Control 0	ADCCTL0	00	189
F71	ADC Raw Data High Byte	ADCRD_H	80	191
F72	ADC Data High Byte	ADCD_H	XX	191
F73	ADC Data Low Bits	ADCD_L	XX	192
F74	ADC Sample Settling Time	ADCSST	FF	193
F75	Sample Time	ADCST	XX	194
F76	ADC Clock Prescale Register	ADCCP	00	195
F77–F7F	Reserved	—	XX	
Low-Power Control				
F80	Power Control 0	PWRCTL0	80	44
F81	Reserved	—	XX	
LED Controller				
F82	LED Drive Enable	LEDEN	00	66
F83	LED Drive Level High Bit	LEDLVLH	00	67
F84	LED Drive Level Low Bit	LEDLVLL	00	67
F85	Reserved	—	XX	
Oscillator Control				
F86	Oscillator Control 0	OSCCTL0	A0	319
F87	Oscillator Control 1	OSCCTL1	00	320
F88–F8F	Reserved			
Comparator 0				
F90	Comparator 0 Control	CMP0	14	257
Comparator 1				
F91	Comparator 1 Control	CMP1	14	258
F92–F9F	Reserved	—	XX	

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.

7.2. Architecture

Figure 9 displays a simplified block diagram of a GPIO port pin and does not illustrate the ability to accommodate alternate functions and variable port current drive strength.

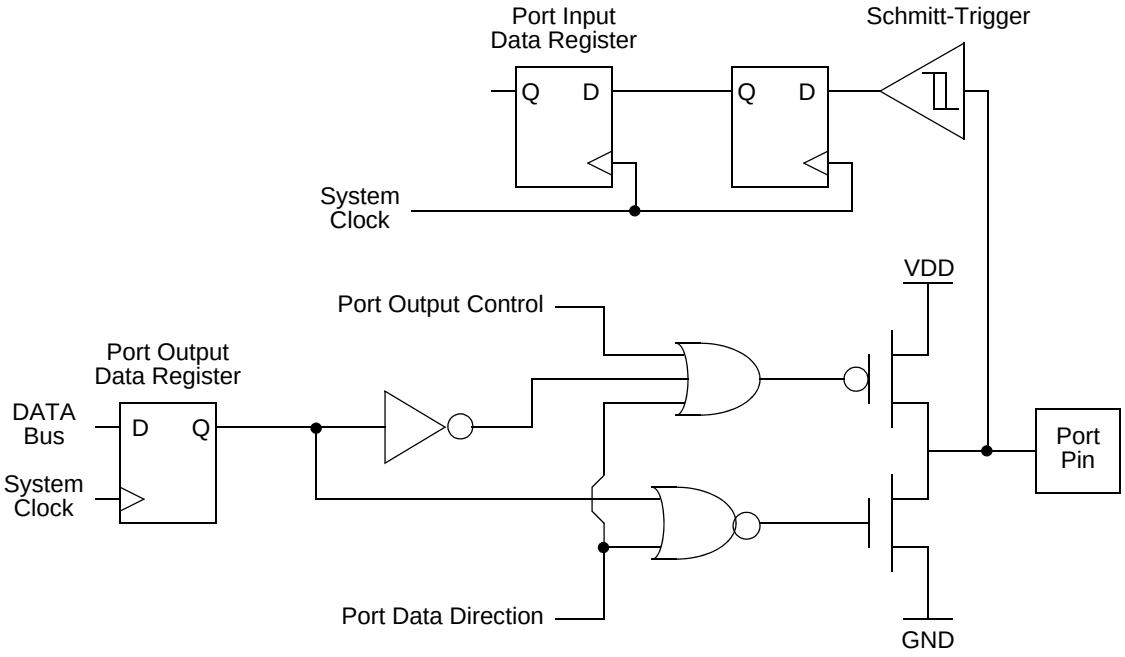

Figure 9. GPIO Port Pin Block Diagram

7.3. GPIO Alternate Functions

Many GPIO port pins are used for GPIO and to access the on-chip peripheral functions like the timers and serial-communication devices. The Port A–E Alternate Function sub-registers configure these pins for either GPIO or alternate function operation. When a pin is configured for alternate function, control of port-pin direction (input/output) is passed from Port A–E Data Direction registers to the alternate functions assigned to this pin. Tables 17 through 19 list the alternate functions possible with each port pin for every package. The alternate function associated at a pin is defined through alternate function sets subregisters AFS1 and AFS2.

The crystal oscillator and the 32 kHz secondary oscillator functionalities are not controlled by the GPIO block. When the crystal oscillator or the 32 kHz secondary oscillator is enabled in the oscillator control block, the GPIO functionality of PA0 and PA1, or PA2 and PA3, is overridden. In such a case, those pins function as input and output for the crystal oscillator.

7.11.7. Port A–E Stop Mode Recovery Source Enable Subregisters

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

Table 27. Port A–E Stop Mode Recovery Source Enable Subregisters (PxSMRE)

Bits	7	6	5	4	3	2	1	0
Field	PSMRE7	PSMRE6	PSMRE5	PSMRE4	PSMRE3	PSMRE2	PSMRE1	PSMRE0
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	If 05H in Port A–E Address Register, accessible through the Port A–E Control Register.							

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

7.11.8. Port A–E Pull-up Enable Subregisters

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

Table 28. Port A–E Pull-Up Enable Subregisters (PxPUE)

Bits	7	6	5	4	3	2	1	0
Field	PPUE7	PPUE6	PPUE5	PPUE4	PPUE3	PPUE2	PPUE1	PPUE0
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	If 06H in Port A–E Address Register, accessible through the Port A–E Control Register.							

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

Table 44. IRQ1 Enable High Bit Register (IRQ1ENH)

Bits	7	6	5	4	3	2	1	0
Field	PA7VENH	PA6C0ENH	PA5C1ENH	PAD4ENH	PAD3ENH	PAD2ENH	PAD1ENH	PA0ENH
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	FC4H							

Bit	Description
[7] PA7VENH	Port A Bit[7] or LVD Interrupt Request Enable High Bit.
[6] PA6C0ENH	Port A Bit[6] or Comparator 0 Interrupt Request Enable High Bit.
[5] PA5C1ENH	Port A Bit[5] or Comparator 1 Interrupt Request Enable High Bit.
[4:1] PAD*x*ENH	Port A or Port D Bit[*x*] (*x*=1, 2, 3, 4) Interrupt Request Enable High Bit.
[0] PA0ENH	Port A Bit[0] Interrupt Request Enable High Bit. See the Shared Interrupt Select Register (IRQSS) on page 82 to determine a selection of either Port A or Port D as the interrupt source.

Table 45. IRQ1 Enable Low Bit Register (IRQ1ENL)

Bits	7	6	5	4	3	2	1	0
Field	PA7VENL	PA6C0ENL	PA5C1ENL	PAD4ENL	PAD3ENL	PAD2ENL	PAD1ENL	PA0ENL
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	FC5H							

Bit	Description
[7] PA7VENL	Port A Bit[7] or LVD Interrupt Request Enable Low Bit.
[6] PA6C0ENL	Port A Bit[6] or Comparator 0 Interrupt Request Enable Low Bit.
[5] PA5C1ENL	Port A Bit[5] or Comparator 1 Interrupt Request Enable Low Bit.
[4:1] PAD*x*ENL	Port A or Port D Bit[*x*] (*x*=1, 2, 3, 4) Interrupt Request Enable Low Bit.
[0] PA0ENL	Port A Bit[0] Interrupt Request Enable Low Bit.

8.4.6. IRQ2 Enable High and Low Bit Registers

Table 46 describes the priority control for IRQ2. The IRQ2 Enable High and Low Bit registers, shown in Tables 47 and 48 form a priority-encoded enabling for interrupts in the Interrupt Request 2 Register. Priority is generated by setting bits in each register.

Table 46. IRQ2 Enable and Priority Encoding

IRQ2ENH[*x*]	IRQ2ENL[*x*]	Priority	Description
0	0	Disabled	Disabled
0	1	Level 1	Low
1	0	Level 2	Nominal
1	1	Level 3	High

Note: An *x* indicates the register bits from 0–7.

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

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

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

9.2.3.4. COUNTER Mode

In COUNTER Mode, the timer counts input transitions from a GPIO port pin. The timer input is taken from the GPIO port pin Timer Input alternate function. 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 timer clock frequency.

Upon reaching the reload value stored in the Timer Reload High and Low Byte registers, the timer generates an interrupt, the count value in the Timer High and Low Byte registers is reset to 0001H and counting resumes. Also, 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 steps to configure a timer for COUNTER Mode and initiate the count:

1. Write to the Timer Control 1 Register to:
   - Disable the timer.
   - Configure the timer for COUNTER Mode.
   - Select either the rising edge or falling edge of the Timer Input signal for the count. This also sets the initial logic level (High or Low) for the Timer Output Alternate Function. However, the Timer Output function is not required to be enabled.
2. Write to the Timer Control 2 Register to choose the timer clock source.

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.

Chapter 12. 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 and providing LIN protocol support. The LIN-UART is a superset of the standard Z8 Encore!® UART, providing all its standard features, LIN protocol support and a digital noise filter.

LIN-UART includes the following features:

- 8-bit asynchronous data transfer
- Selectable even- and odd-parity generation and checking
- Option of 1 or 2 stop bits
- Selectable MULTIPROCESSOR (9-bit) Mode with three configurable interrupt schemes
- Separate transmit and receive interrupts
- Framing, parity, overrun and break detection
- 16-bit baud rate generator (BRG) which can function as a general purpose timer with interrupt
- Driver Enable output for external bus transceivers
- LIN protocol support for both MASTER and SLAVE modes:
  - Break generation and detection
  - Selectable Slave Autobaud
  - Check Tx versus Rx data when sending
- Configuring digital-noise filter on Receive Data line

12.1. LIN-UART Architecture

The LIN-UART consists of three primary functional blocks: transmitter, receiver and baud-rate generator. The LIN-UART's transmitter and receiver function independently but use the same baud rate and data format. The basic UART operation is enhanced by the Noise Filter and IrDA blocks. Figure 19 displays the LIN-UART architecture.

The window remains open until the count again reaches 8 (in other words, 24 baud clock periods since the previous pulse is detected), giving the endec a sampling window of minus 4 baud rate clocks to plus 8 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 the next falling edge. As each falling edge is detected, the endec clock counter is reset, resynchronizing the endec to the incoming signal, allowing the endec to tolerate jitter and baud rate errors in the incoming datastream. Resynchronizing the endec does not alter the operation of the UART, which ultimately receives the data. The UART is only synchronized to the incoming data stream when a start bit is received.

13.3. Infrared Encoder/Decoder Control Register Definitions

All infrared endec configuration and status information is set by the UART control registers as defined beginning on page 163.

Caution: To prevent spurious signals during IrDA data transmission, set the IREN bit in the UART Control 1 Register to 1 to enable the infrared encoder/decoder before enabling the GPIO Port alternate function for the corresponding pin of UART. See Tables 17 through 19 on pages 49–54 for details.

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

Slave 10-Bit Address Recognition Mode. If IRM = 0 during the address phase and the controller is configured for MASTER/SLAVE or SLAVE 10-bit address mode, the hardware detects a match to the 10-bit slave address defined in the I2CMODE and I2CSLVAD registers and generates the slave address match interrupt (the SAM bit = 1 in the I2CISTAT Register). The I^2C controller automatically responds during the Acknowledge phase with the value in the NAK bit of the I2CCTL Register.

17.2.6.2. General Call and Start Byte Address Recognition

If GCE = 1 and IRM = 0 during the address phase and the controller is configured for MASTER/SLAVE or SLAVE in either 7- or 10-bit address modes, the hardware detects a match to the General Call Address or the start byte and generates the slave address match interrupt. A General Call Address is a 7-bit address of all 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^2C controller automatically responds during the address acknowledge phase with the value in the NAK bit of the I2CCTL Register. If the software is set to process the data bytes associated with the GCA bit, the IRM bit can optionally be set following the SAM interrupt to allow the software to examine each received data byte before deciding to set or clear the NAK bit. A start byte will not be acknowledged—a requirement of the I^2C specification.

17.2.6.3. Software Address Recognition

To disable hardware address recognition, the IRM bit must be set to 1 prior to the reception of the address byte(s). When IRM = 1, each received byte generates a receive interrupt (RDRF = 1 in the I2CISTAT Register). The software must examine each byte and determine whether to set or clear the NAK bit. The slave holds SCL Low during the Acknowledge phase until the software responds by writing to the I2CCTL Register. The value written to the NAK bit is used by the controller to drive the I^2C bus, then releasing the SCL. The SAM and GCA bits are not set when IRM = 1 during the address phase, but the RD bit is updated based on the first address byte.

17.2.6.4. Slave Transaction Diagrams

In the following transaction diagrams, the shaded regions indicate data transferred from the Master to the Slave and the unshaded regions indicate the data transferred from the Slave to the Master. The transaction field labels are defined as follows:

S Start
W Write
A Acknowledge
$\overline{A}$ Not Acknowledge
P Stop

17.2.6.5. Slave Receive Transaction with 7-Bit Address

The data transfer format for writing data from a Master to a Slave in 7-bit address mode is displayed in Figure 47. The procedure that follows describes the I^2C Master/Slave Controller operating as a slave in 7-bit addressing mode and receiving data from the bus master.

S	Slave Address	W=0	A	Data	A	Data	A	Data	A/$\overline{A}$	P/S

Figure 47. Data Transfer Format—Slave Receive Transaction with 7-Bit Address

1. The software configures the controller for operation as a slave in 7-bit addressing mode, as follows:
   a. Initialize the MODE field in the I^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 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.

d. Set IEN = 1 in the I^2C Control Register. Set NAK = 0 in the I^2C Control Register.

2. The Master initiates a transfer by sending the address byte. The SLAVE Mode I^2C controller finds an address match and detects that the R/$\overline{W}$ bit = 1 (read by the master from the slave). The I^2C controller acknowledges, indicating that it is ready to accept the transaction. The SAM bit in the I2CISTAT Register is set to 1, causing an interrupt. The RD bit is set to 1, indicating a Read from the slave.
3. The software responds to the interrupt by reading the I2CISTAT Register, thereby clearing the SAM bit. Because RD = 1, the software responds by loading the first data byte into the I2CDATA Register. The software sets the TXI bit in the I2CCTL 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 Step 5 to Step 7 until the final byte has been transferred. If the software has not yet loaded the next data byte when the master brings SCL Low to transfer the most significant data bit, the slave I^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^2C 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.

17.2.6.8. Slave Transmit Transaction With 10-Bit Address

The data transfer format for a master reading data from a slave with 10-bit addressing is displayed in Figure 50. The following procedure describes the I^2C Master/Slave Controller operating as a slave in 10-bit addressing mode, transmitting data to the bus master.

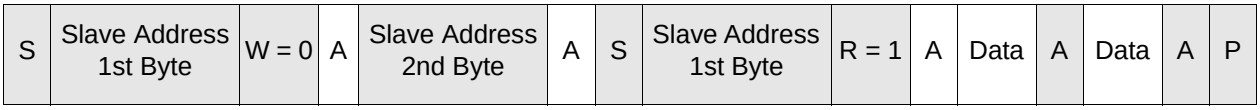

Figure 50. Data Transfer Format—Slave Transmit Transaction with 10-Bit Address

1. The software configures the controller for operation as a slave in 10-bit addressing mode.
   a. Initialize the MODE field in the I^2C Mode Register for either SLAVE ONLY mode or MASTER/SLAVE Mode with 10-bit addressing.
   b. Optionally set the GCE bit.
   c. Initialize the SLA[7:0] bits in the I2CSLVAD Register and SLA[9:8] in the I^2C MODE Register.
   d. Set IEN = 1 and NAK = 0 in the I^2C Control Register.
2. The Master initiates a transfer by sending the first address byte. The SLAVE Mode I^2C controller recognizes the start of a 10-bit address with a match to SLA[9:8] and detects R/W bit = 0 (a Write from the master to the slave). The I^2C controller acknowledges indicating it is available to accept the transaction.
3. The Master sends the second address byte. The SLAVE Mode I^2C 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^2C controller acknowledges on the I^2C bus, indicating it is available to accept the data.
4. The software responds to the slave address match interrupt by reading the I2CISTAT Register, which clears the SAM bit. Because the RD bit = 0, no further action is required.
5. The Master sees the Acknowledge and sends a restart instruction, followed by the first address byte with R/W set to 1. The SLAVE Mode I^2C controller recognizes the restart instruction followed by the first address byte with a match to SLA[9:8] and detects R/W = 1 (the master reads from the slave). The slave I^2C 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.

Bit	Description (Continued)
[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.

On-Chip Debugger. Writing an invalid value or an invalid sequence returns the Flash Controller to its locked state. The write-only Flash Control Register shares its Register File address with the read-only Flash Status Register.

Table 134. Flash Control Register (FCTL)

Bits	7	6	5	4	3	2	1	0
Field	FCMD							
Reset	0	0	0	0	0	0	0	0
R/W	W	W	W	W	W	W	W	W
Address	FF8H							

Bit	Description
[7:0] FCMD	**Flash Command** 73H = First unlock command. 8CH = Second unlock command. 95H = Page Erase command (must be third command in sequence to initiate Page Erase). 63H = Mass Erase command (must be third command in sequence to initiate Mass Erase). 5EH = Enable Flash Sector Protect Register Access

20.3.2. Flash Status Register

The Flash Status register (Table 135) 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 135. Flash Status Register (FSTAT)

Bits	7	6	5	4	3	2	1	0
Field	Program_status		FSTAT					
Reset	0	0	0	0	0	0	0	0
R/W	R	R	R	R	R	R	R	R
Address	FF8H							

Bit	Description
[7:6] Program_ status	**Indicate the fail or success after Flash Write/Erase** 00 = Success. 10 = Success. 11 = Fail due to low power. 01 = Reserved.

Bit	Description
[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 set up as a counter. 1 = OCDCNTR generates a 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 exits 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; must be 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 On-Chip Debugger is not reset. This bit is automatically cleared to 0 when the reset finishes. 0 = No effect. 1 = Reset the device.

secutive. It is possible to write to or read from other registers within the unlocking/locking operation.

When selecting a new clock source, the primary oscillator failure detection circuitry and the Watchdog Timer oscillator failure circuitry must be disabled. If POFEN and WOFEN are not disabled prior to a clock switch-over, it is possible to generate an interrupt for a failure of either oscillator. The Failure detection circuitry can be enabled anytime after a successful write of SCKSEL in the Oscillator Control Register.

The internal precision oscillator is enabled by default. If the user code changes to a different oscillator, it may be appropriate to disable the IPO for power savings. Disabling the IPO does not occur automatically.

24.1.2. Clock Failure Detection and Recovery

Clock failure detection and recovery are provided for the primary oscillator. Clock failure detection is provided for the Watchdog Timer oscillator.

24.1.2.1. Primary Oscillator Failure

The Z8 Encore! XP F1680 Series devices can generate nonmaskable interrupt-like events when the primary oscillator fails. To maintain system function in this situation, the clock failure recovery circuitry automatically forces the Watchdog Timer oscillator to drive the system clock. The Watchdog Timer oscillator must be enabled to allow the recovery. Although this oscillator runs at a much slower speed than the original system clock, the CPU continues to operate allowing execution of a clock failure vector and software routines that either remedy the oscillator failure or issue a failure alert. This automatic switch-over is not available, if the Watchdog Timer is the primary oscillator. It is also unavailable if the Watchdog Timer oscillator is disabled, though it is not necessary to enable the Watchdog Timer reset function outlined in the Watchdog Timer chapter on page 140.

The primary oscillator failure detection circuitry asserts if the system clock frequency drops below 1 kHz ±50%. If an external signal is selected as the system oscillator, it is possible that a very slow but nonfailing clock can generate a failure condition. Under these conditions, do not enable the primary oscillator failure circuitry (i.e., clear the POFEN bit).

24.1.2.2. Watchdog Timer Failure

In the event of a Watchdog Timer oscillator failure, a similar nonmaskable interrupt-like event is issued. This event does not trigger an attendant clock switch-over, but alerts the CPU of the failure. After a Watchdog Timer failure, it is no longer possible to detect a primary oscillator failure. The failure detection circuitry does not function if the Watchdog Timer is used as the primary oscillator or if the Watchdog Timer oscillator has been disabled. For either of these cases, it is necessary to disable the detection circuitry by clearing the WDFEN bit of the OSCCTL0 Register.

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									5, 4 **LDWX** ER2,ER1							
F																

Figure 68. Second Op Code Map after 1FH

Table 189. DC Characteristics (Continued)

		T_A = 0°C to +70°C T_A = –40°C to +105°C				
Symbol	**Parameter**	**Min**	**Typ**	**Max**	**Units**	**Conditions**
V_{IH2}	High Level Input Voltage	0.7*V_{DD}	–	V_{DD}+0.3	V	Ports B and C (Analog)
V_{OL1}	Low Level Output Voltage	–	–	0.4	V	I_{OL} = 2 mA; V_{DD} = 3.0V High Output Drive disabled.
V_{OH1}	High Level Output Voltage	V_{DD}–0.5	–	–	V	I_{OH} = –2 mA; V_{DD} = 3.0V High Output Drive disabled.
V_{OL2}	Low Level Output Voltage	–	–	0.6	V	I_{OL} = 20 mA; V_{DD} = 3.3V High Output Drive enabled.
V_{OH2}	High Level Output Voltage	V_{DD}–0.5	–	–	V	I_{OH} = –20 mA; V_{DD} = 3.3V High Output Drive enabled.
I_{IL}	Input Leakage Current	–5	–	+5	µA	V_{DD} = 3.6V; V_{IN} = V_{DD} or V_{SS}[1]
I_{TL}	Tristate Leakage Current	–5	–	+5	µA	V_{DD} = 3.6V
I_{LED}	Controlled LED Current Drive	1.5	3	4.5	mA	{AFS2,AFS1} = {0,0}, V_{DD} = 3.3V
		2.8	7	10.5	mA	{AFS2,AFS1} = {0,1}, V_{DD} = 3.3V
		7.8	13	19.5	mA	{AFS2,AFS1} = {1,0}, V_{DD} = 3.3V
		12	20	30	mA	{AFS2,AFS1} = {1,1}, V_{DD} = 3.3V
C_{PAD}	GPIO Port Pad Capacitance	–	8.0[2]	–	pF	TBD
C_{XIN}	XIN Pad Capacitance	–	8.0[2]	–	pF	TBD
C_{XOUT}	XOUT Pad Capacitance	–	9.5[2]	–	pF	TBD
I_{PU}	Weak Pull-up Current	30	100	350	µA	V_{DD} = 3.0V–3.6V

Notes:
1. This condition excludes all pins that have on-chip pull-ups, when driven Low.
2. These values are provided for design guidance only and are not tested in production.

Table 196. Analog-to-Digital Converter Electrical Characteristics and Timing

		T_A = 0°C to +70°C T_A = –40°C to +105°C				
Symbol	**Parameter**	**Min**	**Typ**	**Max**	**Units**	**Conditions**
N	Resolution	–	10	–	Bit	
INL	Integral Nonlinearity	–5		5	LSB	
DNL	Differential Nonlinearity	–1		4	LSB	
	Gain Error		15		LSB	
	Offset Error	–15		15	LSB	PDIP package
		–9		9	LSB	Other packages
I_{DD}ADC	ADC Active Current	–	–	2.5	mA	
I_{DDQ}ADC	ADC Quiescent Current	–	5	–	nA	
V_{INT_REF}	Internal Reference Voltage	–	1.6	–	V	REFEN=1, INTREF_SEL=0. See Table 101 on page 189.
		–	AVDD	–	V	REFEN=1, INTREF_SEL=1. See Table 101 on page 189.
V_{EXT_REF}	External Reference Voltage	1.6	–	90% AV_{DD}	V	REFEN=0. See Table 101 on page 189.
V_{INANA}	Analog Input Range	0	–	1.6	V	Internal reference = 1.6V
		0	–	90% AV_{DD}	V	External reference or use AVDD as internal reference
C_{IN}	Analog Input Load	–	–	5	pF	
T_S	Sample Time	1.8	–	–	µs	
T_H	Hold Time	0.5	–	–	µs	
T_{CONV}	Conversion Time	–	13	–	clock cycles	
GBW_{IN}	Input Bandwidth	–	200	–	kHz	
T_{WAKE}	Wake-up Time	–	–	10	µs	External reference
		–	–	10	µs	Internal reference
f_{ADC_CLK}	Maximum Frequency of adc_clk	–	–	5	MHz	V_{DD} = 2.7V to 3.6V
		–	–	2.5	MHz	V_{DD} = 1.8V to 2.7V