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	Obsolete
Core Processor	eZ8
Core Size	8-Bit
Speed	20MHz
Connectivity	I²C, IrDA, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, DMA, POR, PWM, WDT
Number of I/O	46
Program Memory Size	24KB (24K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	3V ~ 3.6V
Data Converters	A/D 12x10b
Oscillator Type	Internal
Operating Temperature	0°C ~ 70°C (TA)
Mounting Type	Surface Mount
Package / Case	64-LQFP
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/zilog/z8f2402ar020sc00tr

Address Space

Overview

The eZ8 CPU can access three distinct address spaces:

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

These three address spaces are covered briefly in the following subsections. For more detailed information regarding the eZ8 CPU and its address space, refer to the *eZ8 CPU User Manual* available for download at www.zilog.com.

Register File

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

The upper 256 bytes of the 4KB Register File address space are reserved for control of the eZ8 CPU, the on-chip peripherals, and the I/O ports. These registers are located at addresses from F00H to FFFH. Some of the addresses within the 256-byte control register section are reserved (unavailable). Reading from an reserved Register File addresses returns an undefined value. 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 Z8F640x family products contain 2KB to 4KB of on-chip RAM depending upon the device. Reading from Register File addresses outside the available RAM addresses (and not within in the control register address space) returns an undefined value. Writing to these Register File addresses produces no effect. Refer to the **Part Selection Guide** section of the **Introduction** chapter to determine the amount of RAM available for the specific Z8F640x family device.

Table 6. Register File Address Map (Continued)

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page #
FED	Port H Control	PHCTL	00	38
FEE	Port H Input Data	PHIN	XX	42
FEF	Port H Output Data	PHOUT	00	43
Watch-Dog Timer (WDT)				
FF0	Watch-Dog Timer Control	WDTCTL	XXX00000b	75
FF1	Watch-Dog Timer Reload Upper Byte	WDTU	FF	76
FF2	Watch-Dog Timer Reload High Byte	WDTH	FF	76
FF3	Watch-Dog Timer Reload Low Byte	WDTL	FF	76
FF4--FF7	Reserved	—	XX	
Flash Memory Controller				
FF8	Flash Control	FCTL	00	144
FF8	Flash Status	FSTAT	00	145
FF9	Flash Page Select	FPS	00	146
FFA	Flash Programming Frequency High Byte	FFREQH	00	147
FFB	Flash Programming Frequency Low Byte	FFREQL	00	147
eZ8 CPU				
FFC	Flags	—	XX	Refer to the *eZ8 CPU User Manual*
FFD	Register Pointer	RP	XX	
FFE	Stack Pointer High Byte	SPH	XX	
FFF	Stack Pointer Low Byte	SPL	XX	

XX=Undefined

Table 10. Port Availability by Device and Package Type (Continued)

Device	Packages	Port A	Port B	Port C	Port D	Port E	Port F	Port G	Port H
Z8F6401	44-pin	[7:0]	[7:0]	[7:0]	[6:0]	-	-	-	-
Z8F6402	64- and 68-pin	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[7]	[3]	[3:0]
Z8F6403	80-pin	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[3:0]

Architecture

Figure 64 illustrates a simplified block diagram of a GPIO port pin. In this figure, the ability to accommodate alternate functions and variable port current drive strength are not illustrated.

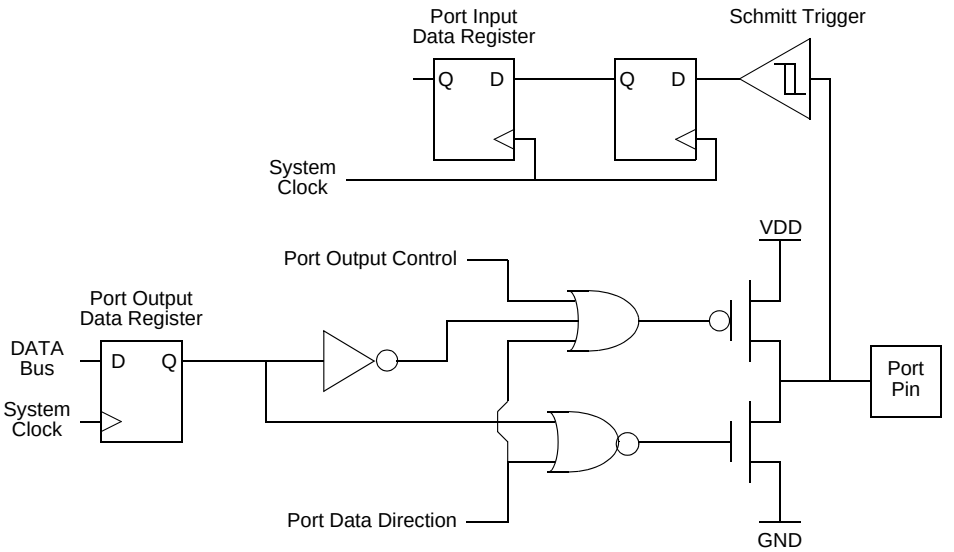

Figure 64. GPIO Port Pin Block Diagram

GPIO Alternate Functions

Many of the GPIO port pins can be used as both general-purpose I/O and to provide access to on-chip peripheral functions such as the timers and serial communication devices. The Port A-H Alternate Function sub-registers configure these pins for either general-purpose I/O or alternate function operation. When a pin is configured for alternate function, control

Architecture

Figure 65 illustrates a block diagram of the interrupt controller.

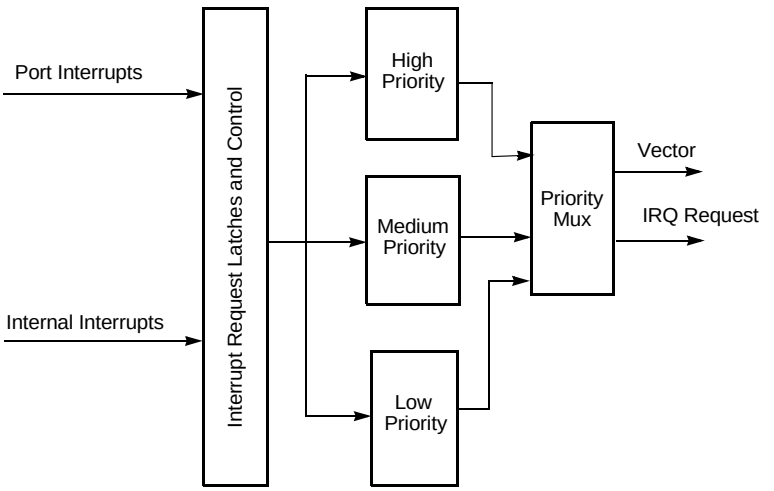

Figure 65. Interrupt Controller Block Diagram

Operation

Master Interrupt Enable

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

Interrupts are globally enabled by any of the following actions:

- Execution of an EI (Enable Interrupt) instruction
- Execution of an IRET (Return from Interrupt) 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 DI (Disable Interrupt) 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

I^2CI— I^2C Interrupt Request
0 = No interrupt request is pending for the I^2C.
1 = An interrupt request from the I^2C is awaiting service.

SPII—SPI Interrupt Request
0 = No interrupt request is pending for the SPI.
1 = An interrupt request from the SPI is awaiting service.

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.

Interrupt Request 1 Register

The Interrupt Request 1 (IRQ1) register (Table 24) stores interrupt requests for both vectored and polled interrupts. When a request is presented to the interrupt controller, the corresponding bit in the IRQ1 register becomes 1. If interrupts are globally enabled (vectored interrupts), the interrupt controller passes an interrupt request to the eZ8 CPU. If interrupts are globally disabled (polled interrupts), the eZ8 CPU can read the Interrupt Request 1 register to determine if any interrupt requests are pending.

Table 24. Interrupt Request 1 Register (IRQ1)

BITS	7	6	5	4	3	2	1	0
FIELD	PAD7I	PAD6I	PAD5I	PAD4I	PAD3I	PAD2I	PAD1I	PAD0I
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
ADDR	FC3H							

PAD*x*I—Port A or Port D Pin *x* Interrupt Request
0 = No interrupt request is pending for GPIO Port A or Port D pin *x*.
1 = An interrupt request from GPIO Port A or Port D pin *x* is awaiting service.

where *x* indicates the specific GPIO Port pin number (0 through 7). For each pin, only 1 of either Port A or Port D can be enabled for interrupts at any one time. Port selection (A or D) is determined by the values in the Interrupt Port Select Register.

Interrupt Control Register

The Interrupt Control (IRQCTL) register (Table 37) contains the master enable bit for all interrupts.

Table 37. Interrupt Control Register (IRQCTL)

BITS	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
ADDR	FCFH							

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, eZ8 CPU acknowledgement of an interrupt request, or Reset.
0 = Interrupts are disabled.
1 = Interrupts are enabled.

Reserved
These bits must be 0.

– Configure the timer for Gated mode.
– Set the prescale value.

2. Write to the Timer High and Low Byte registers to set the starting count value. This only affects the first pass in Gated mode. After the first timer reset in Gated mode, counting always begins at the reset value of 0001H.
3. Write to the Timer Reload High and Low Byte registers to set the Reload value.
4. If desired, 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 register to enable the timer.
7. Assert the Timer Input signal to initiate the counting.

Capture/Compare Mode

In Capture/Compare mode, the timer begins counting on the *first* external Timer Input transition. The desired transition (rising edge or falling edge) is set by the TPOL bit in the Timer Control Register. The timer input is the system clock.

Every subsequent desired transition (after the first) of the Timer Input signal captures the current count value. The Capture value is written to the Timer PWM High and Low Byte Registers. When the Capture event occurs, an interrupt is generated, the count value in the Timer High and Low Byte registers is reset to 0001H, and counting resumes.

If no Capture event occurs, the timer counts up to the 16-bit Compare value stored in the Timer Reload High and Low Byte registers. Upon reaching the Compare value, the timer generates an interrupt, the count value in the Timer High and Low Byte registers is reset to 0001H and counting resumes.

The steps for configuring a timer for Capture/Compare mode and initiating the count are as follows:

1. Write to the Timer Control register to:
   – Disable the timer
   – Configure the timer for Capture/Compare mode.
   – Set the prescale value.
   – Set the Capture edge (rising or falling) for the Timer Input.
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 Compare value.
4. If desired, 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 register to enable the timer.
7. Counting begins on the first appropriate transition of the Timer Input signal. No interrupt is generated by this first edge.

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

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

Reading the Timer Count Values

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

Timer Output Signal Operation

Timer Output is a GPIO Port pin alternate function. Generally, the Timer Output is toggled every time the counter is reloaded.

Timer Control Register Definitions

Timers 0–2 are available in all packages. Timer 3 is available only in the 64-, 68- and 80-pin packages.

Timer 0-3 High and Low Byte Registers

The Timer 0-3 High and Low Byte (TxH and TxL) registers (Tables 38 and 39) contain the current 16-bit timer count value. When the timer is enabled, a read from TxH causes the value in TxL to be stored in a temporary holding register. A read from TMRL always returns this temporary register when the timers are enabled. When the timer is disabled, reads from the TMRL reads the register directly.

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, so simultaneous 16-bit writes are not possible. If either the Timer High or Low Byte registers are

Watch-Dog Timer

Overview

The Watch-Dog Timer (WDT) helps protect against corrupt or unreliable software, power faults, and other system-level problems which may place the Z8 Encore!® into unsuitable operating states. The Watch-Dog Timer includes the following features:

- On-chip RC oscillator
- A selectable time-out response: Short Reset or interrupt
- 24-bit programmable time-out value

Operation

The Watch-Dog Timer (WDT) is a retriggerable one-shot timer that resets or interrupts the Z8F640x family device when the WDT reaches its terminal count. The Watch-Dog Timer uses its own dedicated on-chip RC oscillator as its clock source. The Watch-Dog Timer 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 Watch-Dog Timer to operate all the time, even if a WDT instruction has not been executed.

The Watch-Dog Timer is a 24-bit reloadable downcounter that uses three 8-bit registers in the eZ8 CPU register space to set the reload value. The nominal WDT time-out period is given by the following equation:

$$\text{WDT Time-out Period (ms)} = \frac{\text{WDT Reload Value}}{50}$$

where the WDT reload value is the decimal value of the 24-bit value given by {WDTU[7:0], WDTH[7:0], WDTL[7:0]} and the typical Watch-Dog Timer RC oscillator frequency is 50kHz. The Watch-Dog Timer cannot be refreshed once it reaches 000002H. The WDT Reload Value must not be set to values below 000004H. Table 45 provides

UART

Overview

The Universal Asynchronous Receiver/Transmitter (UART) is a full-duplex communication channel capable of handling asynchronous data transfers. The Z8F640x family device contains two fully independent UARTs. The UART uses a single 8-bit data mode with selectable parity. Features of the UART include:

- 8-bit asynchronous data transfer
- Selectable even- and odd-parity generation and checking
- Option of one or two Stop bits
- Separate transmit and receive interrupts
- Framing, parity, overrun and break detection
- Separate transmit and receive enables
- Selectable 9-bit multiprocessor (9-bit) mode
- 16-bit Baud Rate Generator (BRG)

Architecture

The UART consists of three primary functional blocks: transmitter, receiver, and baud rate generator. The UART's transmitter and receiver function independently, but employ the same baud rate and data format. Figure 67 illustrates the UART architecture.

6. Write to the UART Control 0 register to:
   – Set the receive enable bit (REN) to enable the UART for data reception
   – Enable parity, if desired, and select either even or odd parity.

The UART and DMA are now configured for data reception and automatic data transfer to the Register File. When a valid data byte is received by the UART the following occurs:

7. The UART notifies the DMA Controller that a data byte is available in the UART Receive Data register.
8. The DMA Controller requests control of the system bus from the eZ8 CPU.
9. The eZ8 CPU acknowledges the bus request.
10. The DMA Controller transfers the data from the UART Receive Data register to another location in RAM and then return bus control back to the eZ8 CPU.

The UART and DMA can continue to transfer incoming data bytes without eZ8 CPU intervention. When a UART error is detected, the UART Receiver interrupt is generated. The associated interrupt service routine (ISR) should perform the following:

11. Check the UART Status 0 register to determine the source of the UART error or break condition and then respond appropriately.

Multiprocessor (9-bit) mode

The UART has a Multiprocessor mode that uses an extra (9th) bit for selective communication when a number of processors share a common UART bus. In Multiprocessor (9-bit) 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 illustrated in Figure 70. The character format is:

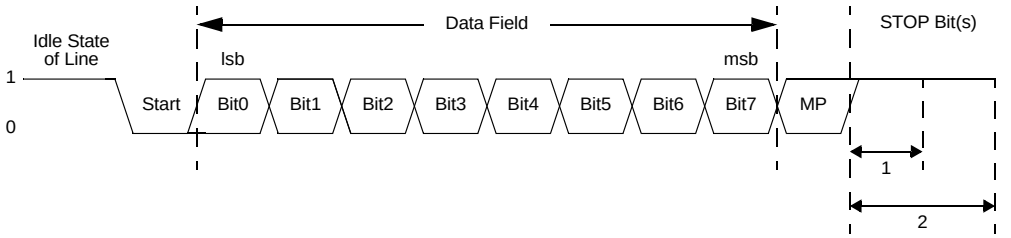

Figure 70. UART Asynchronous Multiprocessor (9-bit) Mode Data Format

In Multiprocessor (9-bit) mode, parity is not an option as the Parity bit location (9th bit) becomes the Multiprocessor control bit. The UART Control 1 and Status 1 registers provide multiprocessor (9-bit) mode control and status information.

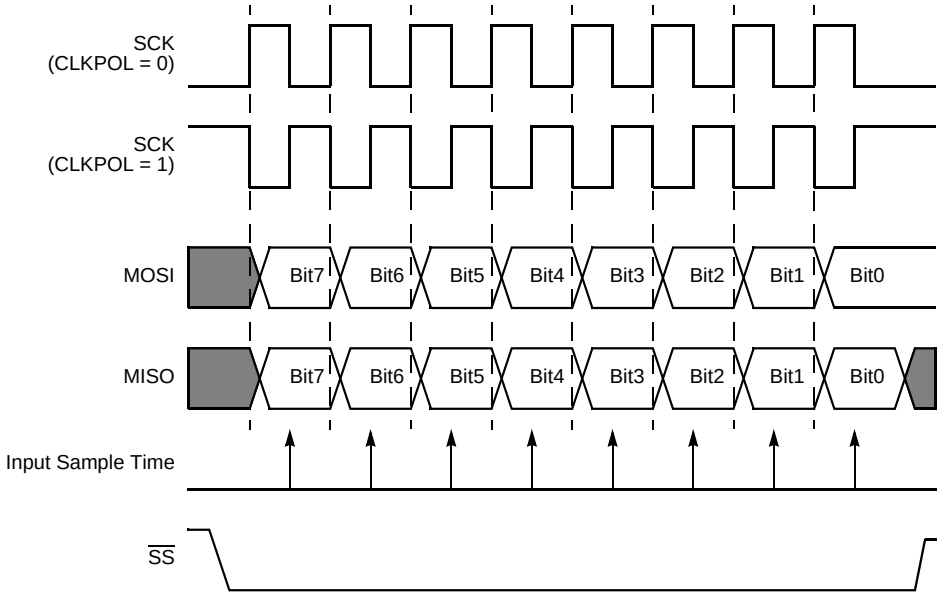

Figure 78. SPI Timing When PHASE is 1

Multi-Master Operation

In a multi-master SPI system, all SCK pins are tied together, all MOSI pins are tied together and all MISO pins are tied together. All SPI pins must then be configured in open-drain mode to prevent bus contention. At any one 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 multi-master 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 multi-master collision (mode fault error condition).

2. The I2C Controller waits for the slave to send an Acknowledge (by pulling the SDA signal Low). If the slave pulls the SDA signal High (Not-Acknowledge), the I2C Controller sends a Stop signal.
3. If the slave needs to service an interrupt, it pulls the SCL signal Low, which halts I2C operation.
4. If there is no other data in the I2C Data register or the STOP bit in the I2C Control register is set by software, then the Stop signal is sent.

Figure 79 illustrates the data transfer format for a 7-bit addressed slave. Shaded regions indicate data transferred from the I2C Controller to slaves and unshaded regions indicate data transferred from the slaves to the I2C Controller.

Figure 79. 7-Bit Addressed Slave Data Transfer Format

The data transfer format for a transmit operation on a 7-bit addressed slave is as follows:

1. Software asserts the IEN bit in the I2C Control register.
2. Software asserts the TXI bit of the I2C Control register to enable Transmit interrupts.
3. The I2C interrupt asserts, because the I2C Data register is empty
4. Software responds to the TDRE bit by writing a 7-bit slave address followed by a 0 (write) to the I2C Data register.
5. Software asserts the START bit of the I2C Control register.
6. The I2C Controller sends the START condition to the I2C slave.
7. The I2C Controller loads the I2C Shift register with the contents of the I2C Data register.
8. After one bit of address has been shifted out by the SDA signal, the Transmit interrupt is asserted.
9. Software responds by writing the contents of the data into the I2C Data register.
10. The I2C Controller shifts the rest of the address and write bit out by the SDA signal.
11. The I2C slave sends an acknowledge (by pulling the SDA signal low) during the next high period of SCL. The I2C Controller sets the ACK bit in the I2C Status register.
12. The I2C Controller loads the contents of the I2C Shift register with the contents of the I2C Data register.
13. The I2C Controller shifts the data out of via the SDA signal. After the first bit is sent, the Transmit interrupt is asserted.

Table 74. DMA*x* Start/Current Address Low Byte Register (DMA*x*START)

BITS	7	6	5	4	3	2	1	0
FIELD	DMA_START							
RESET	X	X	X	X	X	X	X	X
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	FB3H, FHBH							

DMA_START—DMA*x* Start/Current Address Low
These bits, with the four lower bits of the DMA*x*_H register, form the 12-bit Start/Current address. The full 12-bit address is given by {DMA_START_H[3:0], DMA_START[7:0]}.

DMA*x* End Address Low Byte Register

The DMA*x* End Address Low Byte register, in conjunction with the DMA*x*_H register, forms a 12-bit End Address.

Table 75. DMA*x* End Address Low Byte Register (DMA*x*END)

BITS	7	6	5	4	3	2	1	0
FIELD	DMA_END							
RESET	X	X	X	X	X	X	X	X
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	FB4H, FBCH							

DMA_END—DMA*x* End Address Low
These bits, with the four upper bits of the DMA*x*_H register, form a 12-bit address. This address is the ending location of the DMA*x* transfer. The full 12-bit address is given by {DMA_END_H[3:0], DMA_END[7:0]}.

DMA_ADC Address Register

The DMA_ADC Address register points to a block of the Register File to store ADC conversion values as illustrated in Table 76. This register contains the seven most-significant bits of the 12-bit Register File addresses. The five least-significant bits are calculated from the ADC Analog Input number (5-bit base address is equal to twice the ADC Analog Input number). The 10-bit ADC conversion data is stored as two bytes with the most significant byte of the ADC data stored at the even numbered Register File address.

General Purpose I/O Port Input Data Sample Timing

Figure 93 illustrates timing of the GPIO Port input sampling. The input value on a GPIO Port pin is sampled on the rising edge of the system clock. The Port value is then available to the eZ8 CPU on the second rising clock edge following the change of the Port value.

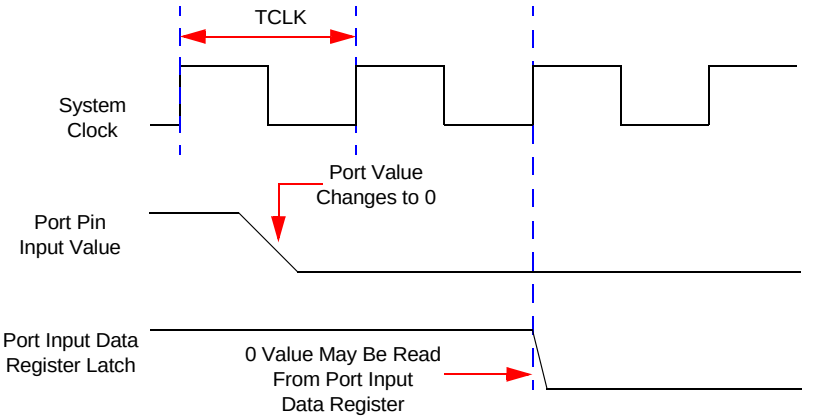

Figure 93. Port Input Sample Timing

Table 107. GPIO Port Input Timing

		Delay (ns)	
Parameter	**Abbreviation**	**Minimum**	**Maximum**
T_{S_PORT}	Port Input Transition to XIN Rise Setup Time (Not pictured)	5	–
T_{H_PORT}	XIN Rise to Port Input Transition Hold Time (Not pictured)	5	–
T_{SMR}	GPIO Port Pin Pulse Width to Insure Stop Mode Recovery (for GPIO Port Pins enabled as SMR sources)	1µs	

Opcode Maps

Figures 101 and 102 provide information on each of the eZ8 CPU instructions. A description of the opcode map data and the abbreviations are provided in Figure 100 and Table 127.

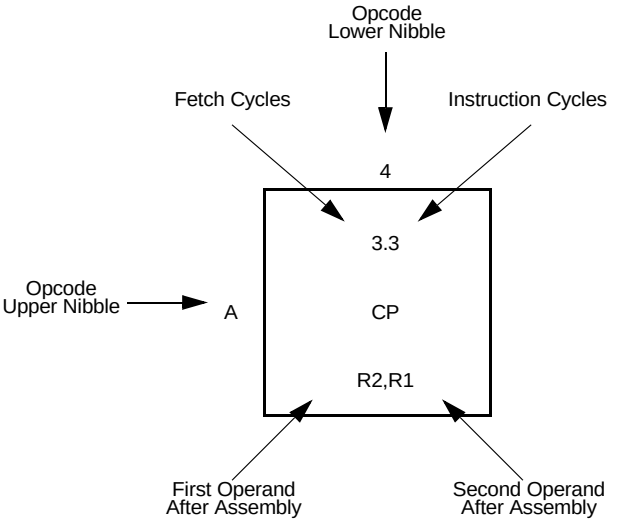

Figure 100. Opcode Map Cell Description

Ordering Information

Table 128. Ordering Information

Part	Flash KB (Bytes)	RAM KB (Bytes)	Max. Speed (MHz)	Temp (^{0}C)	Voltage (V)	Package	Part Number
Z8 Encore!® with 16KB Flash, Standard Temperature							
Z8 Encore!®	16 (16,384)	2 (2048)	20	0 to +70	3.0 - 3.6	PDIP-40	Z8F1601PM020SC
Z8 Encore!®	16 (16,384)	2 (2048)	20	0 to +70	3.0 - 3.6	LQFP-44	Z8F1601AN020SC
Z8 Encore!®	16 (16,384)	2 (2048)	20	0 to +70	3.0 - 3.6	PLCC-44	Z8F1601VN020SC
Z8 Encore!®	16 (16,384)	2 (2048)	20	0 to +70	3.0 - 3.6	LQFP-64	Z8F1602AR020SC
Z8 Encore!®	16 (16,384)	2 (2048)	20	0 to +70	3.0 - 3.6	PLCC-68	Z8F1602VS020SC
Z8 Encore!® with 24KB Flash, Standard Temperature							
Z8 Encore!®	24 (24,576)	2 (2048)	20	0 to +70	3.0 - 3.6	PDIP-40	Z8F2401PM020SC
Z8 Encore!®	24 (24,576)	2 (2048)	20	0 to +70	3.0 - 3.6	LQFP-44	Z8F2401AN020SC
Z8 Encore!®	24 (24,576)	2 (2048)	20	0 to +70	3.0 - 3.6	PLCC-44	Z8F2401VN020SC
Z8 Encore!®	24 (24,576)	2 (2048)	20	0 to +70	3.0 - 3.6	LQFP-64	Z8F2402AR020SC
Z8 Encore!®	24 (24,576)	2 (2048)	20	0 to +70	3.0 - 3.6	PLCC-68	Z8F2402VS020SC
Z8 Encore!® with 32KB Flash, Standard Temperature							
Z8 Encore!®	32 (32,768)	2 (2048)	20	0 to +70	3.0 - 3.6	PDIP-40	Z8F3201PM020SC
Z8 Encore!®	32 (32,768)	2 (2048)	20	0 to +70	3.0 - 3.6	LQFP-44	Z8F3201AN020SC
Z8 Encore!®	32 (32,768)	2 (2048)	20	0 to +70	3.0 - 3.6	PLCC-44	Z8F3201VN020SC
Z8 Encore!®	32 (32,768)	2 (2048)	20	0 to +70	3.0 - 3.6	LQFP-64	Z8F3202AR020SC
Z8 Encore!®	32 (32,768)	2 (2048)	20	0 to +70	3.0 - 3.6	PLCC-68	Z8F3202VS020SC
Z8 Encore!®with 48KB Flash, Standard Temperature							
Z8 Encore!®	48 (49,152)	4 (4096)	20	0 to +70	3.0 - 3.6	PDIP-40	Z8F4801PM020SC
Z8 Encore!®	48 (49,152)	4 (4096)	20	0 to +70	3.0 - 3.6	LQFP-44	Z8F4801AN020SC
Z8 Encore!®	48 (49,152)	4 (4096)	20	0 to +70	3.0 - 3.6	PLCC-44	Z8F4801VN020SC
Z8 Encore!®	48 (49,152)	4 (4096)	20	0 to +70	3.0 - 3.6	LQFP-64	Z8F4802AR020SC
Z8 Encore!®	48 (49,152)	4 (4096)	20	0 to +70	3.0 - 3.6	PLCC-68	Z8F4802VS020SC
Z8 Encore!®	48 (49,152)	4 (4096)	20	0 to +70	3.0 - 3.6	QFP-80	Z8F4803FT020SC

P

Q

R