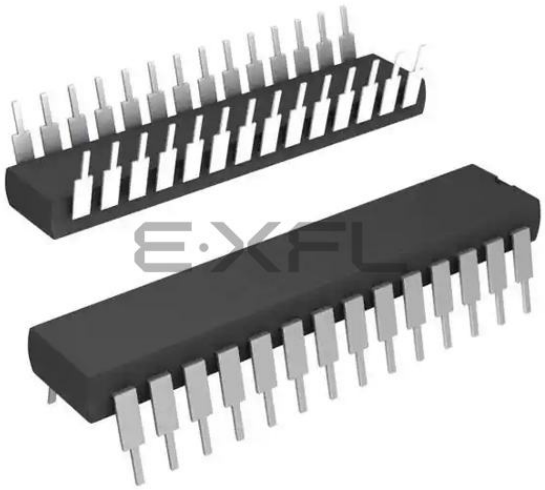


Welcome to E-XFL.COM

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, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	19
Program Memory Size	8KB (8K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 3.6V
Data Converters	-
Oscillator Type	Internal
Operating Temperature	0°C ~ 70°C (TA)
Mounting Type	Through Hole
Package / Case	28-DIP (0.600", 15.24mm)
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/zilog/z8f0812pj020sg

Stop Mode Recovery Using WDT Time-Out

If the WDT times out during STOP Mode, the device undergoes a Stop Mode Recovery sequence. In the WDT Control Register, the WDT and stop bits are set to 1. If the WDT is configured to generate an interrupt upon time-out and the Z8 Encore! XP® F0822 Series device is configured to respond to interrupts, the eZ8 CPU services the WDT interrupt request following the normal Stop Mode Recovery sequence.

Stop Mode Recovery Using a GPIO Port Pin Transition

Each of the GPIO port pins can be configured as a Stop Mode Recovery input source. On any GPIO pin enabled as a STOP Mode Recover 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 WDT Control Register, the stop bit is set to 1.

Caution: In STOP Mode, the GPIO Port Input Data registers (PxIN) are disabled. The Port Input Data registers record the Port transition only if the signal stays on the port pin through the end of the Stop Mode Recovery delay. Therefore, short pulses on the port pin initiates Stop Mode Recovery without being written to the Port Input Data Register or without initiating an interrupt (if enabled for that pin).

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 from High to Low) at timer reload.

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

1. Write to the Timer Control 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 selection 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 only affects the first pass in COUNTER Mode. After the first timer reload in COUNTER Mode, counting always begins at the reset value of 0001H. Generally, in COUNTER Mode the Timer High and Low Byte registers must be written with the value 0001H.
3. Write to the Timer Reload High and Low Byte registers to set the reload value.
4. If required, 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. If using the Timer Output function, configure the associated GPIO port pin for the Timer Output alternate function.
7. Write to the Timer Control Register to enable the timer.

In COUNTER Mode, the number of timer input transitions since the timer start is calculated using the following equation:

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

PWM Mode

In PWM Mode, the timer outputs a Pulse-Width Modulator output signal through a GPIO port pin. The timer input is the system clock. 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

Watchdog Timer Reload Unlock Sequence

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

1. Write 55H to the Watchdog Timer Control Register (WDTCTL).
2. Write AAH to the Watchdog Timer Control Register (WDTCTL).
3. Write the Watchdog Timer Reload Upper Byte Register (WDTU).
4. Write the Watchdog Timer Reload High Byte Register (WDTH).
5. Write the Watchdog Timer Reload Low Byte Register (WDTL).

All three Watchdog Timer Reload registers must be written in this order. There must be no other register writes between each of these operations. If a register write occurs, the lock state machine resets and no further writes occur unless the sequence is restarted. The value in the Watchdog Timer Reload registers is loaded into the counter when the WDT is first enabled and every time a WDT instruction is executed.

Watchdog Timer Control Register Definitions

This section defines the features of the following Watchdog Timer Control registers.

Watchdog Timer Control Register (WDTCTL): see page 74

Watchdog Timer Reload Upper Byte Register (WDTU): see page 75

Watchdog Timer Reload High Byte Register (WDTH): see page 75

Watchdog Timer Reload Low Byte Register (WDTL): see page 76

Watchdog Timer Control Register

The Watchdog Timer Control Register (WDTCTL), shown in Table 48, is a read-only Register that indicates the source of the most recent Reset event, a Stop Mode Recovery event, and a WDT time-out. Reading this register resets the upper four bits to 0.

Writing the 55H, AAH unlock sequence to the Watchdog Timer Control Register (WDTCTL) address unlocks the three Watchdog Timer Reload Byte registers (WDTU, WDTH, and WDTL) to allow changes to the time-out period. These write operations to the WDTCTL address produce no effect on the bits in the WDTCTL. The locking mechanism prevents spurious writes to the Reload registers.

6. Read data from the 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 Step 5 to receive additional data.

Receiving Data Using Interrupt-Driven Method

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

1. Write to the UART Baud Rate High and Low Byte registers to set the required baud rate.
2. Enable the 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 UART Receiver interrupt and set the required priority.
5. Clear the UART Receiver interrupt in the applicable Interrupt Request Register.
6. Write to the UART Control 1 Register to enable MULTIPROCESSOR (9-Bit) Mode functions, if appropriate.
   - Set the Multiprocessor Mode Select (MPEN) to enable MULTIPROCESSOR Mode.
   - Set the Multiprocessor Mode bits, MPMD[1:0], to select the required address matching scheme.
   - Configure the UART to interrupt on received data and errors or errors only (interrupt on errors only is unlikely to be useful for Z8 Encore! XP devices without a DMA block)
7. Write the device address to the Address Compare Register (automatic multiprocessor modes only).
8. Write to the UART Control 0 Register to:
   - Set the receive enable bit (REN) to enable the UART for data reception
   - Enable parity, if required, and if MULTIPROCESSOR Mode is not enabled, and select either even or odd parity.
9. Execute an EI instruction to enable interrupts.

SPI Status Register

The SPI Status Register, shown in Table 66, indicates the current state of the SPI. All bits revert to their reset state if the SPIEN bit in the SPICTL Register equals 0.

Table 66. SPI Status Register (SPISTAT)

Bit	7	6	5	4	3	2	1	0
Field	IRQ	OVR	COL	ABT	Reserved		TXST	SLAS
RESET	0							1
R/W	R/W*				R			
Address	F62H							
Note: *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 multimaster collision (mode fault) has not occurred. 1 = A multimaster 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 is enabled as a Slave, then the following bit settings are true: 0 = $\overline{SS}$ input pin is asserted (Low) 1 = $\overline{SS}$ input is not asserted (High). If SPI is enabled as a Master, this bit is not applicable.

SPI Baud Rate High and Low Byte Registers

The SPI Baud Rate High and Low Byte registers, shown in Tables 69 and 70, combine to form a 16-bit reload value, BRG[15:0], for the SPI Baud Rate Generator. When configured as a general purpose timer, the interrupt interval is calculated using the following equation:

Interrupt Interval (s) = System Clock Period (s) × BRG[15:0]

Table 69. SPI Baud Rate High Byte Register (SPIBRH)

Bit	7	6	5	4	3	2	1	0
Field	BRH							
RESET	1							
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 70. SPI Baud Rate Low Byte Register (SPIBRL)

Bit	7	6	5	4	3	2	1	0
Field	BRL							
RESET	1							
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 Controller

The I²C Controller makes the F0822 Series products bus-compatible with the I²C protocol. The I²C Controller consists of two bidirectional bus lines: a serial data signal (SDA) and a serial clock signal (SCL). Features of the I²C Controller include:

- Transmit and Receive Operation in MASTER Mode
- Maximum data rate of 400 kbit/s
- 7-bit and 10-bit addressing modes for Slaves
- Unrestricted number of data bytes transmitted per transfer

The I²C Controller in the F0822 Series products does not operate in SLAVE Mode.

Architecture

Figure 25 displays the architecture of the I²C Controller.

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 using the SDA signal. After the first bit is sent, the transmit interrupt is asserted.
14. If more bytes remain to be sent, return to Step 9.
15. Software responds by setting the stop bit of the I2C Control Register (or start bit to initiate a new transaction). In the STOP case, software clears the TXI bit of the I2C Control Register at the same time.
16. The I2C Controller completes transmission of the data on the SDA signal.
17. The slave can either Acknowledge or Not Acknowledge the last byte. Because either the stop or start bit is already set, the NCKI interrupt does not occur.
18. The I2C Controller sends the stop (or restart) condition to the I2C bus. The stop or start bit is cleared.

Address-Only Transaction with a 10-Bit Address

In situations in which software must determine if a slave with a 10-bit address is responding without sending or receiving data, a transaction is performed which only consists of an address phase. Figure 28 displays this *address only* transaction to determine if a slave with a 10-bit address will acknowledge.

As an example, this transaction is used after a write has been executed to an EEPROM to determine when the EEPROM completes its internal write operation and is again responding to I2C transactions. If the slave does not acknowledge, the transaction is repeated until the slave is able to acknowledge.

S	Slave Address 1st Seven Bits	W = 0	A/A	Slave Address 2nd Byte	A/A

Figure 28. 10-Bit Address Only Transaction Format

Observe the following procedure for an address-only transaction to a 10-bit addressed slave:

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 (TDRE = 1).
4. Software responds to the TDRE interrupt by writing the first slave address byte. The least-significant bit must be 0 for the write operation.
5. Software asserts the start bit of the I2C Control Register.

Analog-to-Digital Converter

The Analog-to-Digital Converter (ADC) converts an analog input signal to a 10-bit binary number. The features of the sigma-delta ADC include:

- Five analog input sources are multiplexed with GPIO ports
- Interrupt upon conversion complete
- Internal voltage reference generator

The ADC is available only in the Z8F0822, Z8F0821, Z8F0422, Z8F0421, Z8R0822, Z8R0821, Z8R0422 and Z8R0421 devices.

Architecture

Figure 32 displays the three major functional blocks (converter, analog multiplexer and voltage reference generator) of the ADC. The ADC converts an analog input signal to its digital representation. The five-input analog multiplexer selects one of the five analog input sources. The ADC requires an input reference voltage for the conversion. The voltage reference for the conversion can be input through the external V_{REF} pin or generated internally by the voltage reference generator.

Flash Memory

The products in the Z8 Encore! XP® F0822 Series feature either 8KB (8192) or 4KB (4096) bytes of Flash memory with Read/Write/Erase capability. Flash memory is programmed and erased in-circuit by either user code or through the OCD.

The Flash memory array is arranged in 512-byte per page. The 512-byte page is the minimum Flash block size that can be erased. Flash memory is divided into eight sectors which is protected from programming and erase operations on a per sector basis.

Table 81 describes the Flash memory configuration for each device in the Z8F082xfamily. Table 82 lists the sector address ranges. Figure 33 displays the Flash memory arrangement.

Table 81. Flash Memory Configurations

Part Number	Flash Size	Number of Pages	Flash Memory Addresses	Sector Size	Number of Sectors	Pages per Sector
Z8F08xx	8KB (8192)	16	0000H–1FFFH	1KB (1024)	8	2
Z8F04xx	4KB (4096)	8	0000H–0FFFH	0.5KB (512)	8	1

Table 82. Flash Memory Sector Addresses

	Flash Sector Address Ranges	
Sector Number	Z8F04xx	Z8F08xx
0	0000H–01FFH	0000H–03FFH
1	0200H–03FFH	0400H–07FFH
2	0400H–05FFH	0800H–0BFFH
3	0600H–07FFH	0C00H–0FFFH
4	0800H–09FFH	1000H–13FFH
5	0A00H–0BFFH	1400H–17FFH
6	0C00H–0DFFH	1800H–1BFFH
7	0E00H–0FFFH	1C00H–1FFFH

Page Select Register

The Page Select (FPS) Register, shown in Table 86, selects the Flash memory page 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 provided by the PAGE field are erased to FFH.

The Page Select Register shares its Register File address with the Flash Sector Protect Register. The Page Select Register cannot be accessed when the Flash Sector Protect Register is enabled.

Table 86. Page Select Register (FPS)

Bit	7	6	5	4	3	2	1	0
Field	INFO_EN	PAGE						
RESET	0							
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 Flash 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. Flash memory address[15:9] = PAGE[6:0].

Flash Sector Protect Register

The Flash Sector Protect Register, shown in Table 87, protects Flash memory sectors from being programmed or erased from user code. The Flash Sector Protect Register shares its Register File address with the 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). To determine the appropriate Flash memory sector address range and sector number for your F0822 Series product, please refer to Table 82 on page 143.

If breakpoints 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 is still enabled to service DMA and interrupt requests.

The loop on a BRK instruction can be used to service interrupts in the background. For interrupts to be serviced in the background, there cannot be any breakpoints in the ISR. Otherwise, the CPU stops on the breakpoint in the interrupt routine. For interrupts to be serviced in the background, interrupts must also be enabled. Debugging software should not automatically enable interrupts when using this feature, because interrupts are typically disabled during critical sections of code where interrupts should not occur (such as adjusting the stack pointer or modifying shared data).

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 it is looping on, software should not set the DBGMODE bit of the OCDCTL Register. The CPU can have vectored to and be in the middle of an ISR when this bit gets set. Instead, software must clear the BRKLP bit. This allows the CPU to finish the ISR it is in and return the BRK instruction. When the CPU returns to the BRK instruction it was previously looping on, it automatically sets the DBGMODE bit and enter DEBUG Mode.

Software should also note that the majority of the OCD commands are still disabled when the eZ8 CPU is looping on a BRK instruction. The eZ8 CPU must be stopped and the part must be in DEBUG Mode before these commands can be issued.

Breakpoints 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 Breakpoint, write 00H to the appropriate address, overwriting the current instruction. To remove a Breakpoint, the corresponding page of Flash memory must be erased and reprogrammed with the original data.

OCDCNTR Register

The OCD contains a multipurpose 16-bit counter register. It can be used for the following:

- Count system clock cycles between breakpoints
- Generate a BRK when it counts down to zero
- Generate a BRK when its value matches the Program Counter

When configured as a counter, the OCDCNTR Register starts counting when the OCD 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 breakpoints.

or generate a BRK when its value matches the Program Counter. Because this register is really 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 (04H). The Write OCD Control Register command writes the data that follows to the OCDCTL Register. When the Read Protect option bit is enabled, the DBGMODE bit (OCDCTL[7]) can only be set to 1, it cannot be cleared to 0 and the only method of putting the device back into normal operating mode is to reset the device.

```
DBG ← 04H
DBG ← OCDCTL[7:0]
```

Read OCD Control Register (05H). The Read OCD Control Register command reads the value of the OCDCTL Register.

```
DBG ← 05H
DBG → OCDCTL[7:0]
```

Write Program Counter (06H). The Write Program Counter command writes the data that follows to the eZ8 CPU's Program Counter. If the device is not in DEBUG Mode or if the Read Protect option bit is enabled, the Program Counter values are discarded.

```
DBG ← 06H
DBG ← ProgramCounter[15:8]
DBG ← ProgramCounter[7:0]
```

Read Program Counter (07H). 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 (08H). The Write Register command writes data to the Register File. Data can be written 1-256 bytes at a time (256 bytes can be 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 Flash Control 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
```

Bit	Description (Continued)
[4] BRKLOOP	**Breakpoint Loop** This bit determines what action the OCD takes when a BRK instruction is decoded if breakpoints are enabled (BRKEN is 1). If this bit is 0, then the DBGMODE bit is automatically set to 1 and the OCD enter DEBUG Mode. If BRKLOOP is set to 1, then 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 Z8 Encore! XP® F0822 Series device. The device 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 Z8 Encore! XP® F0822 Series device.

DC Characteristics

Table 98 lists the DC characteristics of the Z8 Encore! XP® F0822 Series products. All voltages are referenced to V_{SS}, the primary system ground.

Table 98. DC Characteristics

		T_A = –40°C to 105°C				
Symbol	**Parameter**	**Minimum**	**Typical**	**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*V_{DD}	–	5.5	V	Ports A and C pins when their programmable pull-ups are disabled.
V_{IH2}	High Level Input Voltage	0.7*V_{DD}	–	V_{DD}+0.3	V	Port B pins. Ports A and C pins when their programmable pull-ups are enabled.
V_{IH3}	High Level Input Voltage	0.8*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.
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 = –40°C to +70°C
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 = –40°C to +70°C
V_{OL3}	Low Level Output Voltage High Drive	–	–	0.6	V	I_{OL} = 15mA; V_{DD} = 3.3V High Output Drive enabled; T_A = +70°C to +105°C
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 = +70°C to +105°C
V_{RAM}	RAM Data Retention	0.7	–	–	V	
I_{IL}	Input Leakage Current	–5	–	+5	µA	V_{DD} = 3.6V; V_{IN} = V_{DD} or V_{SS}[1]

Table 122. 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 123. 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 Auto-Increment Addresses
LDE	dst, src	Load External Data to/from Data Memory
LDEI	dst, src	Load External Data to/from Data Memory and Auto-Increment 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

Op Code Maps

A description of the Op Code map data and the abbreviations are provided in Figure 57 and Table 128. Figures 58 and 59 provide information about each of the eZ8 CPU instructions.

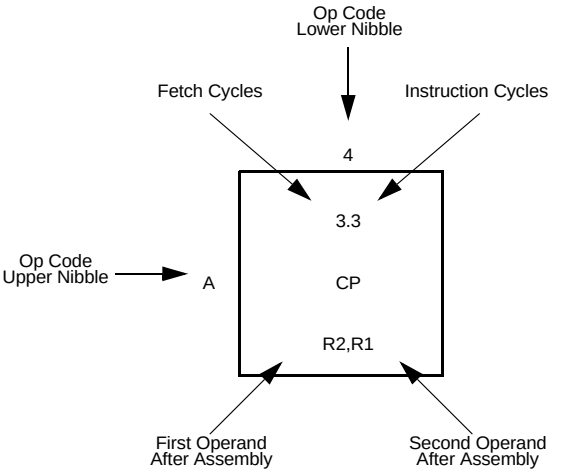


Figure 57. Op Code Map Cell Description

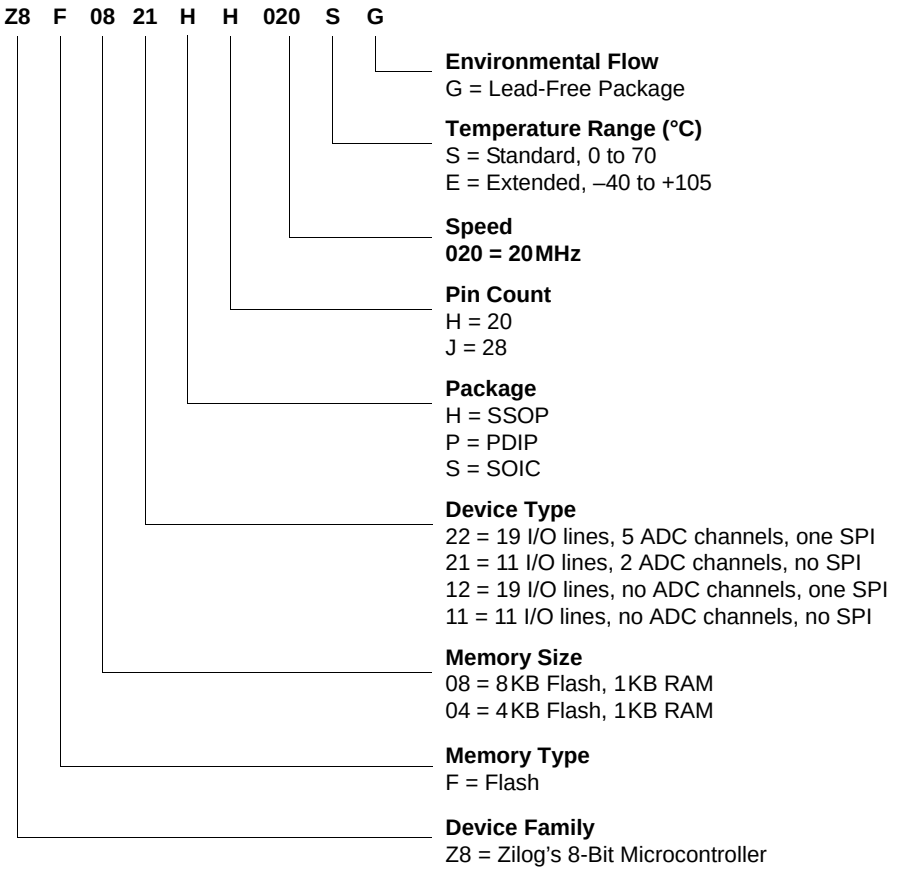
Table 128. Op Code Map Abbreviations

Abbreviation	Description	Abbreviation	Description
b	Bit position	IRR	Indirect register pair
cc	Condition code	p	Polarity (0 or 1)
X	8-bit signed index or displacement	r	4-bit working register
DA	Destination address	R	8-bit register
ER	Extended addressing register	r1, R1, Ir1, Irr1, IR1, rr1, RR1, IRR1, ER1	Destination address
IM	Immediate data value	r2, R2, Ir2, Irr2, IR2, rr2, RR2, IRR2, ER2	Source address
Ir	Indirect working register	RA	Relative
IR	Indirect register	rr	Working register pair
Irr	Indirect working register pair	RR	Register pair

Part Number Suffix Designations

Zilog part numbers consist of a number of components, as indicated in the following example.

Example. Part number Z8F0821HH020SG is an 8-bit Flash Motor Controller with 8KB of Program Memory, equipped with 11 I/O lines and 2 ADC channels in a 20-pin SSOP package, operating within a 0ºC to +70ºC temperature range and built using lead-free solder.



Hex Address: F72

Table 170. ADC Data High Byte Register (ADCD_H)

Bit	7	6	5	4	3	2	1	0
Field	ADCD_H							
RESET	X							
R/W	R							
Address	F72H							

Hex Address: F73

Table 171. ADC Data Low Bits Register (ADCD_L)

Bit	7	6	5	4	3	2	1	0
Field	ADCD_L		Reserved					
RESET	X							
R/W	R							
Address	F73H							

Hex Addresses: F74–FBF

This address range is reserved.

Interrupt Request Control Registers

For more information about the IRQ registers, see the Interrupt Control Register Definitions section on page 45.

Hex Address: FC0

Table 172. Interrupt Request 0 Register (IRQ0)

Bit	7	6	5	4	3	2	1	0
Field	Reserved	T1I	T0I	U0RXI	U0TXI	I2CI	SPII	ADCI
RESET	0							
R/W	R/W							
Address	FC0H							

Hex Address: FC5

Table 177. IRQ1 Enable Low Bit Register (IRQ1ENL)

Bit	7	6	5	4	3	2	1	0
Field	PA7ENL	PA6ENL	PA5ENL	PA4ENL	PA3ENL	PA2ENL	PA1ENL	PA0ENL
RESET	0							
R/W	R/W							
Address	FC5H							

Hex Address: FC6

Table 178. Interrupt Request 2 Register (IRQ2)

Bit	7	6	5	4	3	2	1	0
Field	Reserved				PC3I	PC2I	PC1I	PC0I
RESET	0							
R/W	R/W							
Address	FC6H							

Hex Address: FC7

Table 179. IRQ2 Enable High Bit Register (IRQ2ENH)

Bit	7	6	5	4	3	2	1	0
Field	Reserved				C3ENH	C2ENH	C1ENH	C0ENH
RESET	0							
R/W	R/W							
Address	FC7H							

Hex Address: FC8

Table 180. IRQ2 Enable Low Bit Register (IRQ2ENL)

Bit	7	6	5	4	3	2	1	0
Field	Reserved				C3ENL	C2ENL	C1ENL	C0ENL
RESET	0							
R/W	R/W							
Address	FC8H							