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	23
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	0°C ~ 70°C (TA)
Mounting Type	Surface Mount
Package / Case	28-SSOP (0.209", 5.30mm Width)
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/zilog/z8f1680hj020sg

1.4.3. Non-Volatile Data Storage

Non-Volatile Data Storage (NVDS) is a hybrid hardware/software scheme to implement byte-programmable data memory and is capable of over 100,000 write cycles.

1.4.4. Internal Precision Oscillator

The internal precision oscillator (IPO) is a trimmable clock source which requires no external components. You can select IPO frequency from one of eight frequencies (43.2kHz to 11.0592MHz) and is available with factory-trimmed calibration data.

1.4.5. Crystal Oscillator

The crystal oscillator circuit provides highly accurate clock frequencies using an external crystal, ceramic resonator, or RC network.

1.4.6. Secondary Oscillator

The secondary oscillator is a low-power oscillator, which is optimized for use with a 32kHz watch crystal. It can be used as timer/counter clock source in any mode.

1.4.7. 10-Bit Analog-to-Digital Converter

The Analog-to-Digital Converter (ADC) converts an analog input signal to a 10-bit binary number. The ADC supports up to eight analog input sources multiplexed with GPIO ports.

1.4.8. Low-Power Operational Amplifier

The low-power operational amplifier (LPO) is a general-purpose operational amplifier primarily targeted for current sense applications. The LPO output can be internally routed to the ADC or externally to a pin.

1.4.9. Analog Comparator

The analog comparator compares the signal at an input pin with either an internal programmable voltage reference or a second-input pin. The comparator output is used to either drive an output pin or to generate an interrupt.

Chapter 4. Register Map

Table 8 provides an address map to the register file contained in all Z8 Encore! XP F1680 Series devices. Not all devices and package styles in this product series support the ADC, nor all of the GPIO ports. Therefore, consider the registers for unimplemented peripherals to be reserved.

Table 8. Register File Address Map

Address (Hex)	Register Description	Mnemonic	Reset (Hex)[1]	Page #
General Purpose RAM				
Z8F2480 Device				
000–7FF	General-Purpose Register File RAM	—	XX	
800–EFF	Reserved[2]	—	XX	
Z8F1680 Device				
000–7FF	General-Purpose Register File RAM	—	XX	
800–EFF	Reserved[2]	—	XX	
Z8F0880 Device				
000–3FF	General-Purpose Register File RAM	—	XX	
400–EFF	Reserved[2]	—	XX	
Special Purpose Registers				
Timer 0				
F00	Timer 0 High Byte	T0H	00	109
F01	Timer 0 Low Byte	T0L	01	109
F02	Timer 0 Reload High Byte	T0RH	FF	110
F03	Timer 0 Reload Low Byte	T0RL	FF	110
F04	Timer 0 PWM0 High Byte	T0PWM0H	00	110
F05	Timer 0 PWM0 Low Byte	T0PWM0L	00	111
F06	Timer 0 Control 0	T0CTL0	00	112
F07	Timer 0 Control 1	T0CTL1	00	113

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.

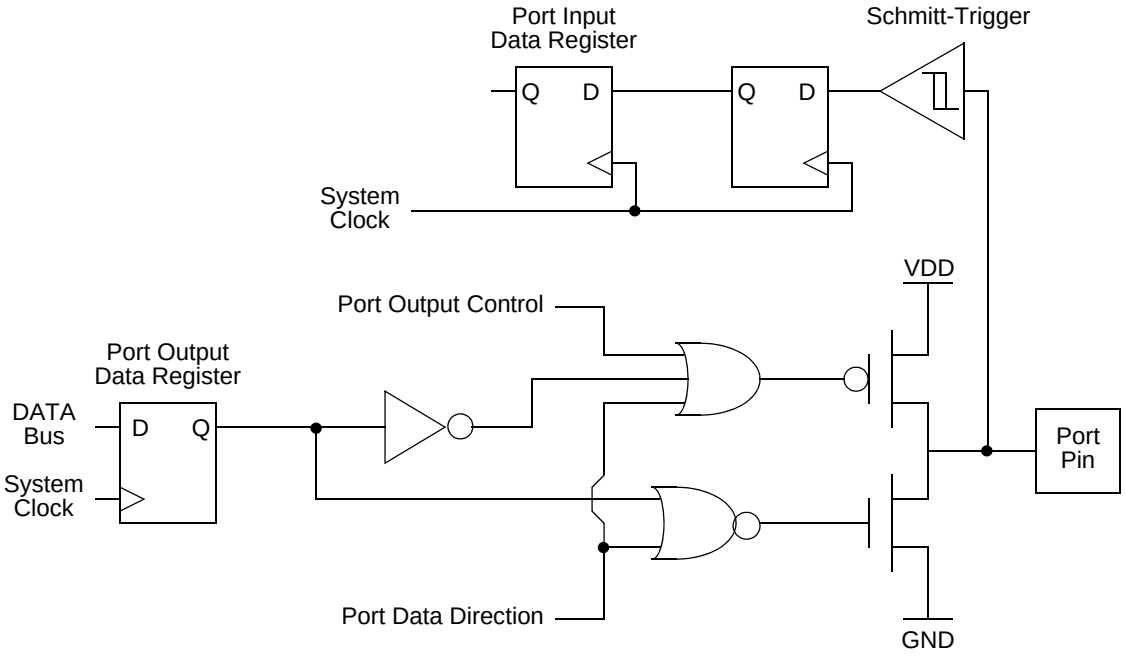

Figure 9. GPIO Port Pin Block Diagram

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.

7.10. GPIO Interrupts

Many of the GPIO port pins can be used as interrupt sources. Some port pins can be configured to generate an interrupt request on either the rising edge or falling edge of the pin-input signal. Other port-pin interrupt sources generate an interrupt when any edge occurs (both rising and falling). For more details about interrupts using the GPIO pins, see the Interrupt Controller chapter on page 68.

7.11. GPIO Control Register Definitions

Four registers for each port provide access to GPIO control, input data and output data. Table 20 lists these port registers. Use Port A–E Address and Control registers together to provide access to subregisters for port configuration and control.

Table 20. GPIO Port Registers and Subregisters

Port Register Mnemonic	Port Register Name
P*x*ADDR	Port A–E Address Register (Selects subregisters)
P*x*CTL	Port A–E Control Register (Provides access to subregisters)
P*x*IN	Port A–E Input Data Register
P*x*OUT	Port A–E Output Data Register
Port Subregister Mnemonic	**Port Register Name**
P*x*DD	Data Direction
P*x*AF	Alternate Function
P*x*OC	Output Control (Open-Drain)
P*x*HDE	High Drive Enable
P*x*SMRE	Stop Mode Recovery Source Enable
P*x*PUE	Pull-up Enable
PxAFS1	Alternate Function Set 1
PxAFS2	Alternate Function Set 2

8.4.2. Interrupt Request 1 Register

The Interrupt Request 1 (IRQ1) Register, shown in Table 38, 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 a 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 38. Interrupt Request 1 Register (IRQ1)

Bits	7	6	5	4	3	2	1	0
Field	PA7VI	PA6CI	PA5CI	PAD4I	PAD3I	PAD2I	PAD1I	PA0I
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	FC3H							

Bit	Description
[7] PA7VI	**Port A7 or LVD Interrupt Request** 0 = No interrupt request is pending for GPIO Port A7 or LVD. 1 = An interrupt request from GPIO Port A7 or LVD.
[6] PA6CI	**Port A6 or Comparator 0 Interrupt Request** 0 = No interrupt request is pending for GPIO Port A6 or Comparator 0. 1 = An interrupt request from GPIO Port A6 or Comparator 0.
[5] PA5CI	**Port A5 or Comparator 1 Interrupt Request** 0 = No interrupt request is pending for GPIO Port A5 or Comparator 1. 1 = An interrupt request from GPIO Port A5 or Comparator 1.
[4:1] PADxI	**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; *x* indicates the specific GPIO port pin number (1–4).
[0] PA0I	**Port A Pin 0 Interrupt Request** 0 = No interrupt request is pending for GPIO Port A0. 1 = An interrupt request from GPIO Port A0 is awaiting service. For interrupt source select description, see the Shared Interrupt Select Register section on page 82.

$$\text{PWM Period (s)} = \frac{\text{Reload Value} \times \text{Prescale}}{\text{Timer Clock Frequency (Hz)}}$$

If TPOL is set to 0, the ratio of the PWM output High time to the total period is calculated using the following equation:

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

If TPOL is set to 1, the ratio of the PWM output High time to the total period is calculated using the following equation:

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

9.2.3.7. PWM DUAL Output Mode

In PWM DUAL OUTPUT Mode, the timer outputs a Pulse Width Modulator output signal and also its complement through two GPIO port pins. The timer first counts up to the 16-bit PWM match value stored in the Timer PWM0 High and Low Byte registers. When the timer count value matches the PWM value, the Timer Outputs (TOUT and $\overline{\text{TOUT}}$) toggle. The timer continues counting until it reaches the reload value stored in the Timer Reload High and Low Byte registers. Upon reaching the reload value, the timer generates an interrupt, the count value in the Timer High and Low Byte registers is reset to `0001H` and TOUT and $\overline{\text{TOUT}}$ toggles again and counting resumes.

If the TPOL bit in the Timer Control 1 Register is set to 1, the Timer Output signal begins as High (1) and then transitions to Low (0) when the timer value matches the PWM value. The Timer Output signal returns to High (1) after the timer reaches the reload value and is reset to `0001H`.

If the TPOL bit in the Timer Control 1 Register is set to 0, the Timer Output signal begins as Low (0) and then transitions to High (1) when the timer value matches the PWM value. The Timer Output signal returns to Low (0) after the timer reaches the reload value and is reset to `0001H`.

The timer also generates a second PWM output signal, Timer Output Complement ($\overline{\text{TOUT}}$). $\overline{\text{TOUT}}$ is the complement of the Timer Output PWM signal (TOUT). A programmable deadband delay can be configured to set a time delay (0 to 128 timer clock cycles) when one PWM output transitions from High to Low and the other PWM output

Observe the following steps to configure a timer for CAPTURE RESTART Mode and initiate the count:

1. Write to the Timer Control 1 Register to:
   - Disable the timer
   - Configure the timer for CAPTURE RESTART Mode. Setting the mode also involves writing to TMODE[3] bit in the TxCTL0 Register
   - Set the prescale value
   - Set the Capture edge (rising or falling) for the Timer Input
2. Write to the Timer Control 2 Register to choose the timer clock source.
3. Write to the Timer Control 0 Register to set the timer interrupt configuration field TICONFIG.
4. Write to the Timer High and Low Byte registers to set the starting count value (typically `0001H`).
5. Write to the Timer Reload High and Low Byte registers to set the reload value.
6. Clear the Timer PWM High and Low Byte registers to `0000H`. This allows user software to determine if interrupts are generated by either a Capture Event or a Reload. If the PWM High and Low Byte registers still contain `0000H` after the interrupt, then the interrupt is generated by a Reload.
7. If required, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers. By default, the timer interrupt will be generated for both input capture and reload events. If required, configure the timer interrupt to be generated only at the Input Capture event or the reload event by setting TICONFIG field of the Timer Control 0 Register.
8. Configure the associated GPIO port pin for the Timer Input alternate function.
9. Write to the Timer Control 1 Register to enable the timer and initiate counting.

In CAPTURE Mode, the elapsed time from Timer start to Capture event can be calculated using the following equation:

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

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

generated the interrupt, read the associated GPIO input value and compare to the value stored in the TPOL bit.

The timer counts up to the 16-bit reload value stored in the Timer Reload High and Low Byte registers. The timer input is the timer clock. When reaching the reload value, the timer generates an interrupt, the count value in the Timer High and Low Byte registers is reset to 0001H and counting resumes (assuming the Timer Input signal is still asserted). 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 reset.

Observe the following steps to configure a timer for GATED Mode and initiate the count:

1. Write to the Timer Control 1 Register to:
   - Disable the timer
   - Configure the timer for GATED Mode
   - Set the prescale value
2. Write to the Timer Control 2 Register to choose the timer clock source.
3. Write to the Timer Control 0 Register to set the timer interrupt configuration field TICONFIG.
4. Write to the Timer High and Low Byte registers to set the starting count value. This value 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.
5. Write to the Timer Reload High and Low Byte registers to set the reload value.
6. If required, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers. By default, the timer interrupt will be generated for both input deassertion and reload events. If required, configure the timer interrupt to be generated only at the Input Deassertion event or the Reload event by setting TICONFIG field of the Timer Control 0 Register.
7. Configure the associated GPIO port pin for the Timer Input alternate function.
8. Write to the Timer Control 1 Register to enable the timer.
9. Assert the Timer Input signal to initiate the counting.

9.2.4. 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.

9.2.5. Timer Output Signal Operation

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

9.2.6. Timer Noise Filter

A Noise Filter circuit is included which filters noise on a Timer Input signal before the data is sampled by the block.

The Noise Filter has the following features:

- Synchronizes the receive input data to the Timer Clock
- NFEN (Noise Filter Enable) input selects whether the Noise Filter is bypassed (NFEN=0) or included (NFEN=1) in the receive data path
- NFCTL (Noise Filter Control) input selects the width of the up/down saturating counter digital filter. The available widths range from 4 bits to 11 bits
- The digital filter output has hysteresis
- Provides an active Low *saturated state* output (FiltSatB) which is used as an indication of the presence of noise
- Available for operation in STOP Mode

9.2.7. Architecture

Figure 12 displays how the Noise Filter is integrated with the Timer.

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

12.3.8. LIN Control Register

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

Table 92. LIN Control Register (U0CTL1 = F43H with MSEL = 010b)

Bit	7	6	5	4	3	2	1	0
Field	LMST	LSLV	ABEN	ABIEN	LinState[1:0]		TxBreakLength	
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F43H, F4BH							

Note: R/W = Read/Write.

Bit Position	Value	Description
[7] LMST	**LIN MASTER Mode**	
	0	LIN MASTER Mode not selected.
	1	LIN MASTER Mode selected (if MPEN, PEN, LSLV = 0).
[6] LSLV	**LIN SLAVE Mode**	
	0	LIN SLAVE Mode not selected.
	1	LIN SLAVE Mode selected (if MPEN, PEN, LMST = 0).
[5] ABEN	**Autobaud Enable**	
	0	Autobaud not enabled.
	1	Autobaud enabled, if in LIN SLAVE Mode.
[4] ABIEN	**Autobaud Interrupt Enable**	
	0	Interrupt following autobaud does not occur.
	1	Interrupt following autobaud enabled, if in LIN SLAVE Mode. When the autobaud character is received, a receive interrupt is generated and the ATB bit is set in the Status0 Register.

```
nop      ; wait for output to settle
ldx IRQ0,#0 ; clear any spurious interrupts pending
ei
```

18.2. Comparator Control Register Definitions

This section defines the features of the following Comparator Control registers.

Comparator 0 Control Register: see page 257

Comparator 1 Control Register: see page 258

18.2.1. Comparator 0 Control Register

The Comparator 0 Control Register (CMP0), shown in Table 130, configures the Comparator 0 inputs and sets the value of the internal voltage reference.

Table 130. Comparator 0 Control Register (CMP0)

Bits	7	6	5	4	3	2	1	0
Field	INPSEL	INNSEL	REFLVL				TIMTRG	
Reset	0	0	0	1	0	1	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F90H							

Bit	Description
[7] INPSEL	**Signal Select for Positive Input** 0 = GPIO pin used as positive comparator 0 input. 1 = Temperature sensor used as positive comparator 0 input.
[6] INNSEL	**Signal Select for Negative Input** 0 = Internal reference disabled, GPIO pin used as negative comparator 0 input. 1 = Internal reference enabled as negative comparator 0 input.

Chapter 19. Temperature Sensor

The on-chip Temperature Sensor allows you to measure temperature on the die to an accuracy of roughly ±7°C over a range of –40°C to +105°C. Over a reduced range, the accuracy is significantly better. This block is a moderately accurate temperature sensor for low-power applications where high accuracy is not required. Uncalibrated accuracy is significantly worse, therefore the temperature sensor is not recommended for untrimmed use:

- On-chip temperature sensor
- ±7°C full-range accuracy for calibrated version
- ±1.5°C accuracy over the range of 20°C to 30°C
- Flash recalibration capability

19.1. Operation

The on-chip temperature sensor is a Proportional To Absolute Temperature (PTAT) topology which provides for zero-point calibration. A pair of Flash option bytes contain the calibration data. The temperature sensor can be disabled by a bit in the Power Control Register 0 (see page 44) to reduce power consumption.

The temperature sensor can be directly read by the ADC to determine the absolute value of its output. The temperature sensor output is also available as an input to the comparator for threshold type measurement determination. The accuracy of the sensor when used with the comparator is substantially less than when measured by the ADC. Maximum accuracy can be obtained by customer retrimming the sensor using an external reference and a high-precision external reference in the target application.

During normal operation, the die undergoes heating that will cause a mismatch between the ambient temperature and that measured by the sensor. For best results, the XP device should be placed into STOP Mode for sufficient time such that the die and ambient temperatures converge (this time will be dependent on the thermal design of the system). The temperature sensor should be measured immediately after recovery from STOP Mode.

The following two equations define the relationship between the ADC reading and the die temperature. In each equation, T is the temperature in degrees Celsius, and ADC is the 10-bit compensated ADC value.

Equation #1. If bit 2 of TEMPCALH calibration option byte is 0, then:

T = (25/128) * (ADC + {TEMPCALH_bit1, TEMPCALH_bit0, TEMPCALL}) – 77

Equation #2. If bit 2 of TEMPCALH calibration option byte is 1, then:

T = (25/128) * (ADC – {TEMPCALH_bit1, TEMPCALH_bit0, TEMPCALL}) –77

The Flash Sector Protect Register can be configured to prevent sectors from being programmed or erased. After a sector is protected, it cannot be unprotected by user code. The Flash Sector Protect Register is cleared after reset and any previously written protection values is lost. User code must write this register in their initialization routine if they want to enable sector protection.

The Flash Sector Protect Register shares its Register File address with the Page Select Register. The Flash Sector Protect Register is accessed by writing the Flash Control Register with 5EH. After the Flash Sector Protect Register is selected, it can be accessed at the Page Select Register address. When user code writes the Flash Sector Protect Register, bits can only be set to 1. Thus, sectors can be protected, but not unprotected, via register write operations. Writing a value other than 5EH to the Flash Control Register deselects the Flash Sector Protect Register and reenables access to the Page Select Register. code:

1. Write 00H to the Flash Control Register to reset the Flash Controller.
2. Write 5EH to the Flash Control Register to select the Flash Sector Protect Register.
3. Read and/or write the Flash Sector Protect Register which is now at Register File address FF9H.
4. Write 00H to the Flash Control Register to return the Flash Controller to its reset state.

The Sector Protect Register is initialized to 0 on Reset, putting each sector into an unprotected state. When a bit in the Sector Protect Register is written to 1, the corresponding sector can no longer be written or erased. After a bit of the Sector Protect Register has been set, it can not be cleared except by a System Reset.

20.2.4. Byte Programming

Flash memory is enabled for byte programming on the active page after unlocking the Flash Controller. Erase the address(es) to be programmed using either the Page Erase or Mass Erase command prior to performing byte programming. An erased Flash byte contains all 1s (FFH). The programming operation can only be used to change bits from 1 to 0. To change a Flash bit (or multiple bits) from 0 to 1 requires execution of either the Page Erase or Mass Erase command.

Byte programming can be accomplished using the On-Chip Debugger's Write Memory command or eZ8 CPU execution of the LDC or LDCI instructions. For a description of the LDC and LDCI instructions, refer to the eZ8 CPU Core User Manual (UM0128), available for download at www.zilog.com. While the Flash Controller programs the contents of Flash memory, the eZ8 CPU idles but the system clock and on-chip peripherals continue to operate.

After a byte is written, the page remains unlocked, allowing for subsequent writes to other bytes on the same page. To exit programming mode and lock Flash memory, write any value to the Flash Control Register except the Mass Erase or Page Erase commands.

J