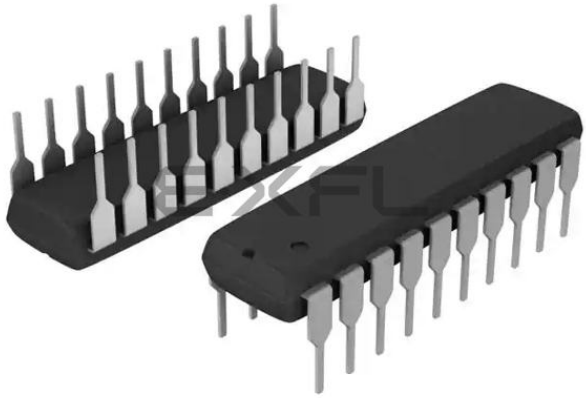


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	eZ8
Core Size	8-Bit
Speed	20MHz
Connectivity	I²C, IrDA, LINbus, UART/USART
Peripherals	Brown-out Detect/Reset, LED, LVD, POR, PWM, Temp Sensor, WDT
Number of I/O	17
Program Memory Size	8KB (8K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	A/D 7x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 105°C (TA)
Mounting Type	Through Hole
Package / Case	20-DIP (0.300", 7.62mm)
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/zilog/z8f0880ph020eg

Chapter 3. Address Space

The eZ8 CPU can access the following three distinct address spaces:

- The Register File contains addresses for general-purpose registers, eZ8 CPU, peripherals and GPIO 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 contain data only

These three address spaces are covered briefly in the following sections. For more details about the eZ8 CPU and its address space, refer to the eZ8 CPU Core User Manual (UM0128), available for download at www.zilog.com.

3.1. Register File

The Register File address space in the Z8 Encore!® MCU is 4KB (4096 bytes). The Register File is composed of two sections: control registers and general-purpose registers. When instructions are executed, registers defined as sources are read and registers defined as destinations are written. 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, on-chip peripherals and the input/output ports. These registers are located at addresses F00H to FFFH. Some of the addresses within the 256 B control register sections are reserved (that is, unavailable). Reading from a reserved Register File address returns an undefined value. Zilog does not recommend writing to the reserved Register File addresses because doing so can produce unpredictable results.

The on-chip Register RAM always begins at address 000H in the Register File address space. The F1680 Series MCU contains 1KB or 2KB of on-chip Register RAM. Reading from Register File addresses outside the available RAM addresses (and not within the control register address space) returns an undefined value. Writing to these Register File addresses produces no effect.

In addition, the F1680 Series MCU contains 1KB of on-chip Program RAM. Normally it is used as Program RAM and is present in the Program Memory address space (see the Program Memory section on page 20). However, it can also be used as additional Register RAM present in the Register File address space 800H–BFFH (1KB Program RAM, 2KB Register RAM), or 400H–7FFH (1KB Program RAM, 1KB Register RAM), if you do not

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.

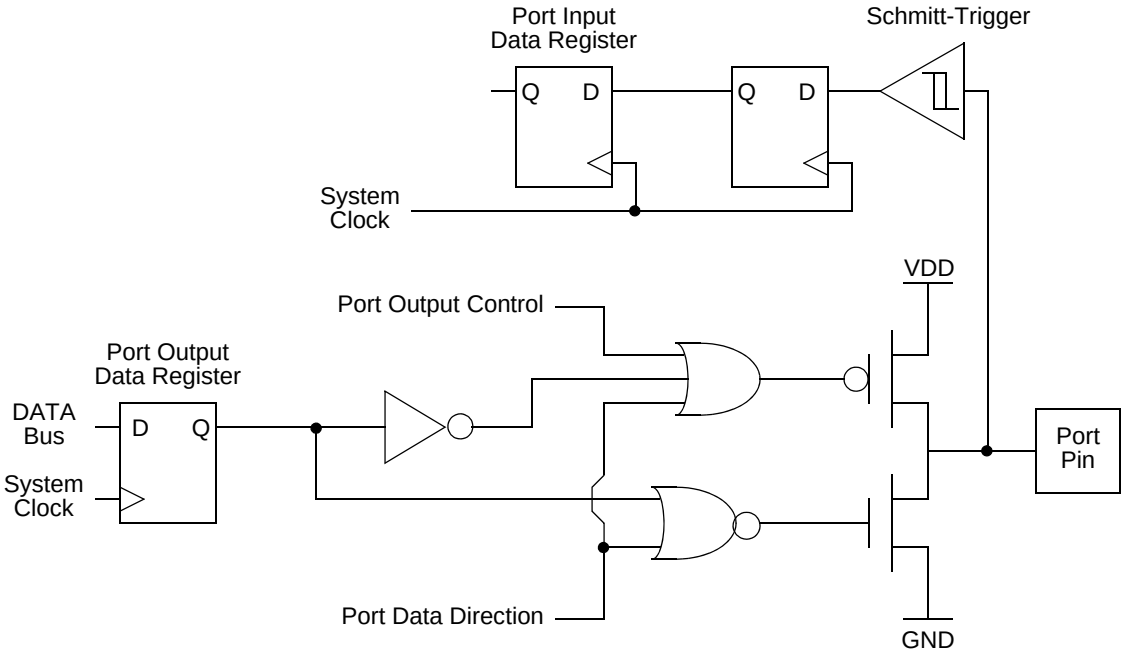


Figure 9. GPIO Port Pin Block Diagram

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.

7.4. Direct LED Drive

The Port C pins provide a current synchronized output capable of driving an LED without requiring an external resistor. The output synchronizes current at programmable levels of 3mA, 7mA, 13mA and 20mA. This mode is enabled through the Alternate Function sub-register AFS1 and is programmable through the LED control registers. For proper function, the LED anode must be connected to V_{DD} and the cathode to the GPIO pin.

Using all Port C pins in LED drive mode with maximum current can result in excessive total current. For the maximum total current for the applicable package, see the Electrical Characteristics chapter on page 349.

7.5. Shared Reset Pin

On all the devices, the Port D0 pin shares function with a bidirectional reset pin. Unlike all other I/O pins, this pin does not default to GPIO pin on power-up. This pin acts as a bidirectional input/open-drain output reset with an internal pull-up until user software reconfigures it as GPIO PD0. The Port D0 pin is output-only when in GPIO Mode, and must be configured as an output. PD0 supports the High Drive feature but not the Stop Mode Recovery feature.

7.6. Crystal Oscillator Override

For systems using the crystal oscillator, PA0 and PA1 is used to connect the crystal. When the main crystal oscillator is enabled (see the Oscillator Control1 Register section on page 320), the GPIO settings are overridden and PA0 and PA1 is disabled.

7.7. 32kHz Secondary Oscillator Override

For systems using a 32kHz secondary oscillator, PA2 and PA3 is used to connect a watch crystal. When the 32kHz secondary oscillator is enabled (see the Oscillator Control1 Register section on page 320), the GPIO settings are overridden and PA2 and PA3 is disabled.

7.8. 5V Tolerance

All GPIO pins, including those that share functionality with an ADC, crystal or comparator signals are 5V-tolerant and can handle inputs higher than V_{DD} even with the pull-ups enabled.

System clock is only for operation in ACTIVE and HALT modes. System clock is software selectable in Oscillator Control Module as external high-frequency crystal or internal precision oscillator. The TCLKS field in the Timer Control 2 Register selects the timer clock source.

! Caution: When the timer is operating on a peripheral clock, the timer clock is asynchronous to the CPU clock. To ensure error-free operation, disable the timer before modifying its operation (also include changing the timer clock source). Therefore, any write to the timer control registers cannot be performed when the timer is enabled and a peripheral clock is used.

When the timer uses a peripheral clock and the timer is enabled, any read from TxH or TxL is not recommended, because the results can be unpredictable. Disable the timer first, then read it. If the timer works in the CAPTURE, CAPTURE/COMPARE, CAPTURE RESTART or DEMODULATION modes, any read from TxPWM0H, TxPWM0L, TxPWM1H, TxPWM1L or TxSTAT must be performed after a capture interrupt occurs; otherwise, results can be unpredictable. The INPCAP bit of the Timer Control 0 Register is the same as these PWM registers. When the timer uses the main clock, you can write/read all timer registers at any time.

9.2.2. Low-Power Modes

Timers can operate in both HALT Mode and STOP Mode.

9.2.2.1. Operation in HALT Mode

When the eZ8 CPU enters HALT Mode, the timer will continue to operate if enabled. To minimize current in HALT Mode, the timer can be disabled by clearing the TEN control bit. The noise filter, if enabled, will also continue to operate in HALT Mode and rejects any noise on the timer input pin.

9.2.2.2. Operation in STOP Mode

When the eZ8 CPU enters STOP Mode, the timer continues to operate if enabled and peripheral clock is chosen as the clock source. In STOP Mode, the timer interrupt (if enabled) automatically initiates a Stop Mode Recovery and generates an interrupt request. In the Reset Status Register, the stop bit is set to 1. Also, timer interrupt request bit in Interrupt Request 0 register is set. Following completion of the Stop Mode Recovery, if interrupts are enabled, the CPU responds to the interrupt request by fetching the timer interrupt vector. The noise filter, if enabled, will also continue to operate in STOP Mode and rejects any noise on the timer input pin.

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

9.3.5.2. Timer 0–2 Control 1 Register

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

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

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

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

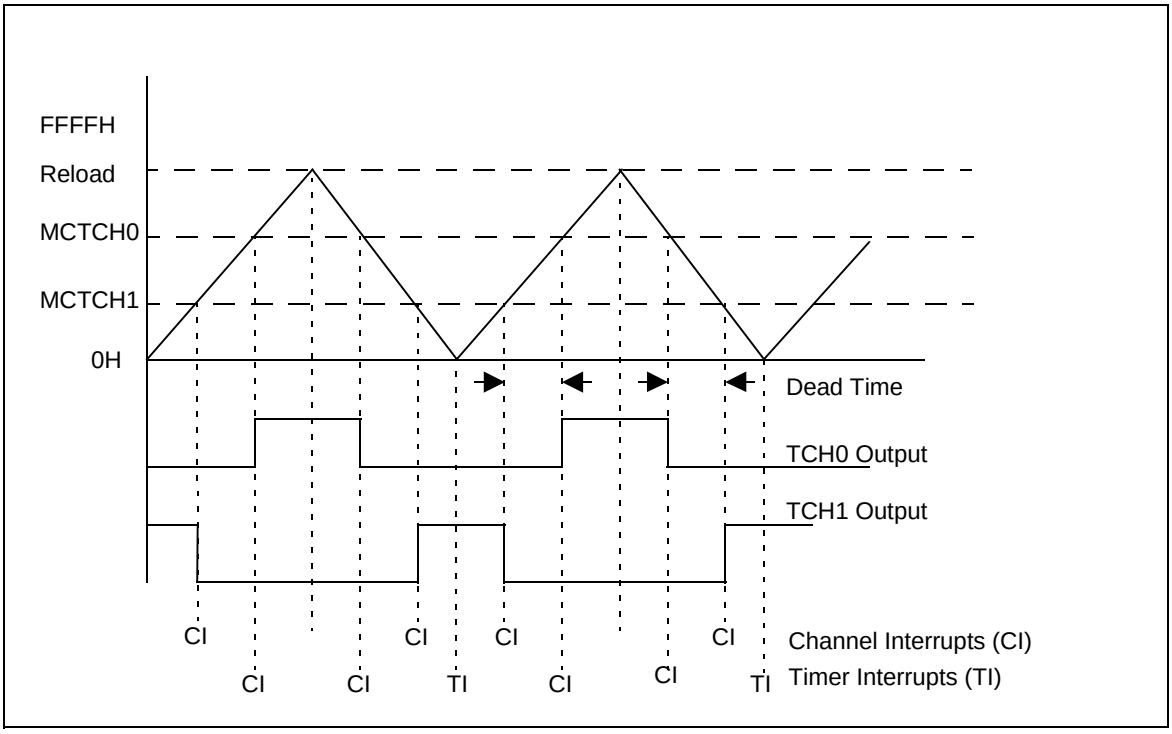


Figure 17. Count Up/Down Mode with PWM Channel Outputs and Deadband

10.6.2. Multiple Timer Intervals Generation

Figure 18 shows a timing diagram featuring two constant time intervals, T0 and T1. The timer is in Count Modulo Mode with reload = FFFFH. Channels 0 and 1 are set up for CONTINUOUS COMPARE operation. After every channel compare interrupt, the channel Capture/Compare registers are updated in the interrupt service routine by adding a constant equal to the time interval required. This operation requires that the Channel Update Enable (CHUE) bit must be set in channels 0 and 1 so that writes to the Capture/ Compare registers take affect immediately.

interrupts to go inactive until the next address byte. If the new frame's address matches the LIN-UART's, then the data in the new frame is also processed.

The second scheme is enabled by setting MPMD[1:0] to 10B and writing the LIN-UART's address into the LIN-UART Address Compare Register. This mode introduces more hardware control, interrupting only on frames that match the LIN-UART's address. When an incoming address byte does not match the LIN-UART's address, it is ignored. All successive data bytes in this frame are also ignored. When a matching address byte occurs, an interrupt is issued and further interrupts occur on each successive data byte. The first data byte in the frame has NEWFRM=1 in the LIN-UART Status 1 Register. When the next address byte occurs, the hardware compares it to the LIN-UART's address. If there is a match, the interrupt occurs and the NEWFRM bit is set to the first byte of the new frame. If there is no match, the LIN-UART ignores all incoming bytes until the next address match.

The third scheme is enabled by setting MPMD[1:0] to `11B` and by writing the LIN-UART's address into the LIN-UART Address Compare Register. This mode is identical to the second scheme, except that there are no interrupts on address bytes. The first data byte of each frame remains accompanied by a NEWFRM assertion.

12.1.10. LIN Protocol Mode

The Local Interconnect Network (LIN) protocol as supported by the LIN-UART module is defined in rev 2.0 of the LIN Specification Package. The LIN protocol specification covers all aspects of transferring information between LIN Master and Slave devices using *message frames* including error detection and recovery, SLEEP Mode and wake-up from SLEEP Mode. The LIN-UART hardware in LIN mode provides character transfers to support the LIN protocol including break transmission and detection, wake-up transmission and detection and slave autobauding. Part of the error detection of the LIN protocol is for both master and slave devices to monitor their receive data when transmitting. If the receive and transmit data streams do not match, the LIN-UART asserts the PLE bit (physical layer error bit in Status 0 Register). The *message frame* time-out aspect of the protocol depends on software requiring the use of an additional general purpose timer. The LIN mode of the LIN-UART does not provide any hardware support for computing/verifying the checksum field or verifying the contents of the identifier field. These fields are treated as data and are not interpreted by hardware. The checksum calculation/verification can easily be implemented in software via the ADC (Add with Carry) instruction.

The LIN bus contains a single Master and one or more Slaves. The LIN master is responsible for transmitting the message frame header which consists of the Break, Synch and Identifier fields. Either the master or one of the slaves transmits the associated *response* section of the message which consists of data characters followed by a checksum character.

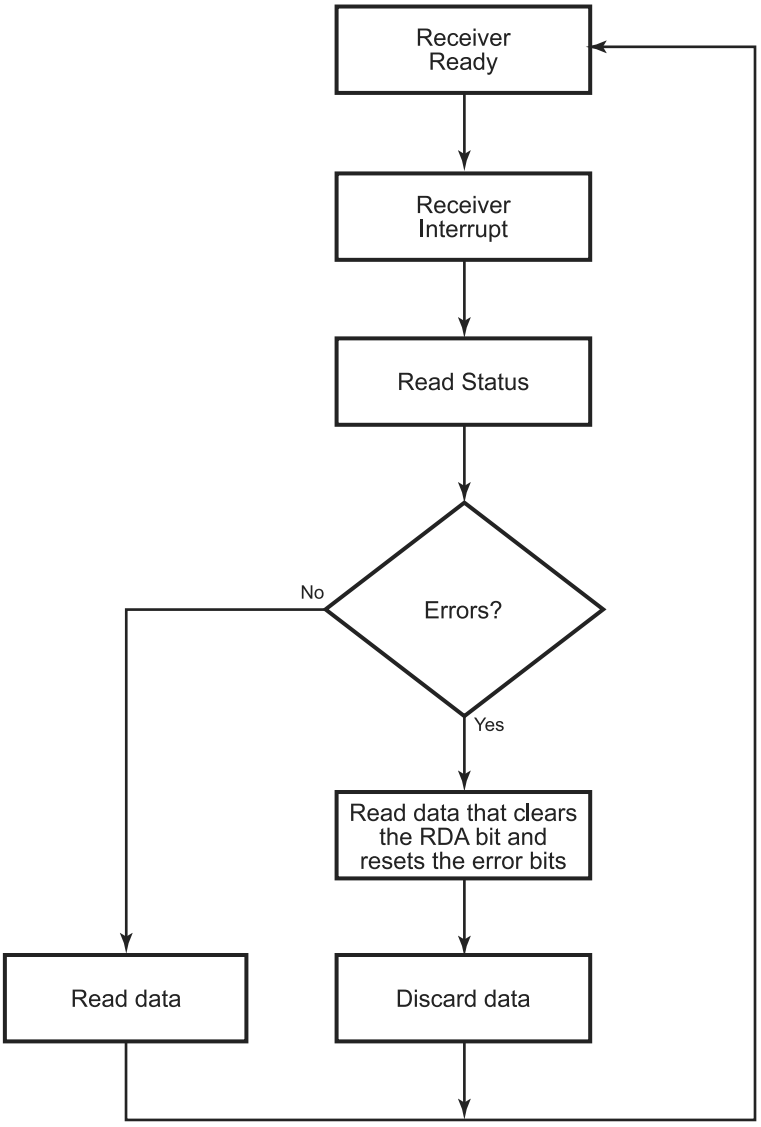


Figure 24. LIN-UART Receiver Interrupt Service Routine Flow

12.1.11.5. Baud Rate Generator Interrupts

If the BRGCTL bit of the Multiprocessor Control Register (LIN-UART Control 1 Register with MSEL = 000b) is set and the REN bit of the Control 0 Register is 0. The LIN-UART Receiver interrupt asserts when the LIN-UART Baud Rate Generator reloads. This action allows the Baud Rate Generator to function as an additional counter, if the LIN-UART receiver functionality is not employed. The transmitter can be enabled in this mode.

For a given LIN-UART data rate, the integer baud rate divisor value is calculated using the following equation for LIN mode UART operation:

$$\text{UART Baud Rate Divisor Value (BRG)} = \text{Round}\left(\frac{\text{System Clock Frequency (Hz)}}{\text{UART Data Rate (bits/s)}}\right)$$

The baud rate error relative to the appropriate baud rate is calculated using the following equation:

$$\text{UART Baud Rate Error (\%)} = 100 \times \left(\frac{\text{Actual Data Rate} - \text{Desired Data Rate}}{\text{Desired Data Rate}}\right)$$

For reliable communication, the LIN-UART baud rate error must never exceed 5 percent. Tables 96 through 100 provide error data for popular baud rates and commonly-used crystal oscillator frequencies for normal UART modes of operation.

Table 96. LIN-UART Baud Rates, 20.0 MHz System Clock

Applicable Rate (kHz)	BRG Divisor (Decimal)	Actual Rate (kHz)	Error (%)	Applicable Rate (kHz)	BRG Divisor (Decimal)	Actual Rate (kHz)	Error (%)
1250.0	1	1250.0	0.00	9.60	130	9.62	0.16
625.0	2	625.0	0.00	4.80	260	4.81	0.16
250.0	5	250.0	0.00	2.40	521	2.399	–0.03
115.2	11	113.64	–1.19	1.20	1042	1.199	–0.03
57.6	22	56.82	–1.36	0.60	2083	0.60	0.02
38.4	33	37.88	–1.36	0.30	4167	0.299	–0.01
19.2	65	19.23	0.16				

Table 97. LIN-UART Baud Rates, 10.0 MHz System Clock

Applicable Rate (kHz)	BRG Divisor (Decimal)	Actual Rate (kHz)	Error (%)	Applicable Rate (kHz)	BRG Divisor (Decimal)	Actual Rate (kHz)	Error (%)
1250.0	N/A	N/A	N/A	9.60	65	9.62	0.16
625.0	1	625.0	0.00	4.80	130	4.81	0.16
250.0	3	208.33	–16.67	2.40	260	2.40	–0.03
115.2	5	125.0	8.51	1.20	521	1.20	–0.03
57.6	11	56.8	–1.36	0.60	1042	0.60	–0.03

16.4.4. ESPI Mode Register

The ESPI Mode Register, shown in Table 112, configures the character bit width and mode of the ESPI I/O pins.

Table 112. ESPI Mode Register (ESPIMODE)

Bits	7	6	5	4	3	2	1	0
Field	SSMD			NUMBITS[2:0]			SSIO	SSPO
Reset	000			0	0	0	0	0
R/W	R/W			R/W	R/W	R/W	R/W	R/W
Address	F63H							

Bit	Description
[7:5] SSMD	**Slave Select Mode** This field selects the behavior of $\overline{SS}$ as a framing signal. For a detailed description of these modes, see Slave Select on page 200. **000 = SPI Mode** When SSIO = 1, the $\overline{SS}$ pin is driven directly from the SSV bit in the Transmit Data Command Register. The Master software should set SSV (or a GPIO output if the $\overline{SS}$ pin is not connected to the appropriate Slave) to the asserted state prior to or on the same clock cycle that the Transmit Data Register is written with the initial byte. At the end of a frame (after the last RDRNE event), SSV will be automatically deasserted by hardware. In this mode, SCK is active only for data transfer (one clock cycle per bit transferred). **001 = Loopback Mode** When ESPI is configured as Master (MMEN = 1), the outputs are deasserted and data is looped from Shift Register Out to Shift Register In. When ESPI is configured as a Slave (MMEN = 0) and $\overline{SS}$ in asserts, MISO (Slave output) is tied to MOSI (Slave input) to provide an asynchronous remote loop back (echo) function. **010 = I2S Mode (Synchronous Framing with SSV)** In this mode, the value from SSV will be output by the Master on the $\overline{SS}$ pin with one SCK period before the data and will remain in that state until the start of the next frame. Typically this mode is used to send back to back frames with $\overline{SS}$ alternating on each frame. A frame boundary is indicated in the Master when SSV changes. A frame boundary is detected in the Slave by $\overline{SS}$ changing state. The $\overline{SS}$ framing signal will lead the frame by one SCK period. In this mode SCK will run continuously, starting with the initial $\overline{SS}$ assertion. Frames will run back-to-back as long as software continues to provide data. An example of this mode is the I^2S protocol (Inter IC Sound) which is used to carry left and right channel audio data with the $\overline{SS}$ signal indicating which channel is being sent. In SLAVE Mode, the change in state of $\overline{SS}$ (Low to High or High to Low) triggers the start of a transaction on the next SCK cycle.

Table 118. I²C Master/Slave Controller Registers (Continued)

Name	Abbreviation	Description
I²C Baud Rate High	I2CBRH	High byte of baud rate generator initialization value.
I²C Baud Rate Low	I2CBRL	Low byte of baud rate generator initialization value.
I²C State	I2CSTATE	State register.
I²C Mode	I2CMODE	Selects MASTER or SLAVE modes, 7-bit or 10-bit addressing; configure address recognition, define slave address bits [9:8].
I²C Slave Address	I2CSLVAD	Defines slave address bits [7:0].

17.2. Operation

The I²C Master/Slave Controller operates in MASTER/SLAVE Mode, SLAVE ONLY Mode, or with master arbitration. In MASTER/SLAVE Mode, it can be used as the only Master on the bus or as one of the several masters on the bus, with arbitration. In a Multi-Master environment, the controller switches from MASTER to SLAVE Mode on losing arbitration.

Though slave operation is fully supported in MASTER/SLAVE Mode, if a device is intended to operate only as a slave, then SLAVE ONLY mode can be selected. In SLAVE ONLY mode, the device will not initiate a transaction, even if the software inadvertently sets the start bit.

17.2.1. SDA and SCL Signals

The I²C circuit sends all addresses, Data and Acknowledge signals over the SDA line, with most-significant bit first. SCL is the clock for the I²C bus. When the SDA and SCL pin alternate functions are selected for their respective GPIO ports, the pins are automatically configured for open-drain operation.

The Master is responsible for driving the SCL clock signal. During the Low period of the clock, a slave can hold the SCL signal Low to suspend the transaction if it is not ready to proceed. The Master releases the clock at the end of the Low period and notices that the clock remains Low instead of returning to a High level. When the slave releases the clock, the I²C master continues the transaction. All data is transferred in bytes; there is no limit to the amount of data transferred in one operation. When transmitting address, data, or an Acknowledge, the SDA signal changes in the middle of the Low period of SCL. When receiving address, Data, or an Acknowledge; the SDA signal is sampled in the middle of the High period of SCL.

A low-pass digital filter can be applied to the SDA and SCL receive signals by setting the Filter Enable (FILTEN) bit in the I²C Control Register. When the filter is enabled, any glitch that is less than a system clock period in width will be rejected. This filter should be

In these two equations, TEMPCALH and TEMPCALL are a pair of Flash option bits containing the calibration data. For more details, see the discussion of TEMPCALH and TEMPCALL in the Flash Option Bits chapter on page 276.

Note: The equations above are temporary test results of the Z8F1680 MCU, version A. The coefficient in the formula may change according to results from tests of version B.

19.1.1. Calibration

The temperature sensor undergoes calibration during the manufacturing process and is maximally accurate only at 30°C. Accuracy decreases as measured temperatures move further from the calibration point.

Because this sensor is an on-chip sensor, Zilog recommends that the user accounts for the difference between ambient and die temperatures when inferring ambient temperature conditions.

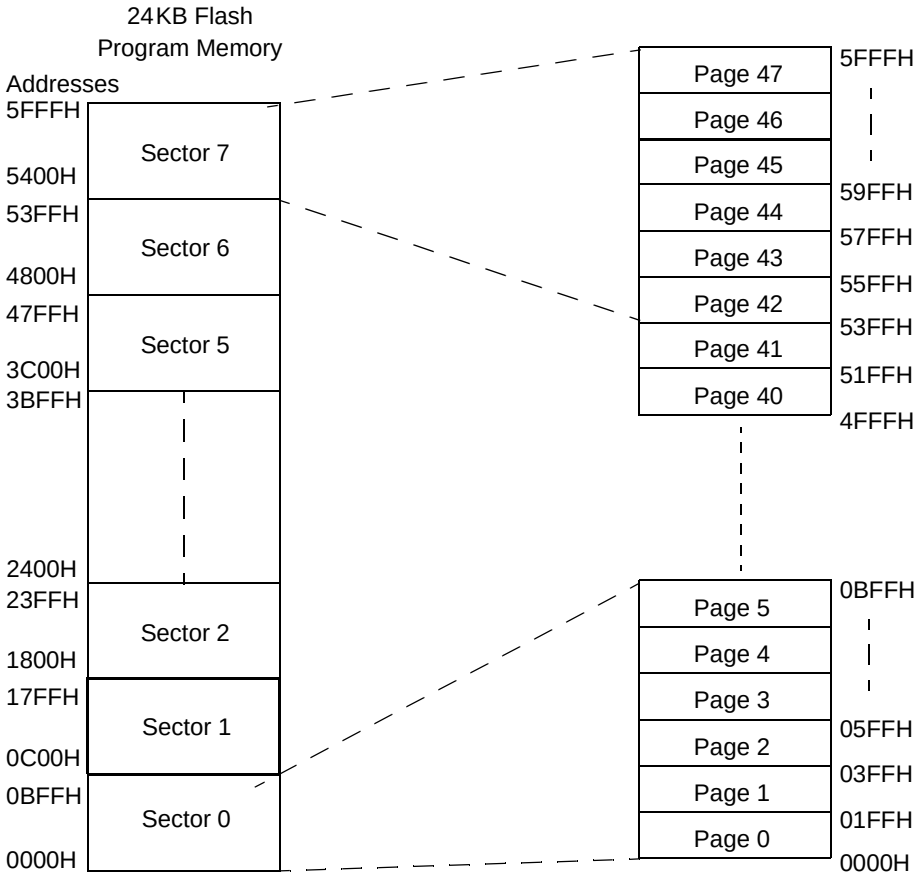


Figure 53. 24KB Flash Memory Arrangement

20.2. Operation

The Flash Controller programs and erases Flash memory. The Flash Controller provides the proper Flash controls and timing for byte programming, Page Erase and Mass Erase of Flash memory.

The Flash Controller contains several protection mechanisms to prevent accidental programming or erasure. These mechanisms operate on the page, sector and full-memory levels.

The Flow Chart in Figure 54 displays basic Flash Controller operation. The sections that follow provide details about the various operations (Lock, Unlock, Byte Programming, Page Protect, Page Unprotect, Page Select Page Erase and Mass Erase) shown in Figure 54.

20.2.8. Flash Controller Behavior in Debug Mode

The following changes in behavior of the Flash Controller occur when the Flash Controller is accessed using the On-Chip Debugger:

- The Flash Write Protect option bit is ignored
- The Flash Sector Protect register is ignored for programming and erase operations
- Programming operations are not limited to the page selected in the Page Select Register
- Bits in the Flash Sector Protect register can be written to 1 or 0
- The second write of the Page Select register to unlock the Flash Controller is not necessary
- The Page Select register can be written when the Flash Controller is unlocked
- The Mass Erase command is enabled through the Flash Control Register

! Caution: For security reasons, the Flash controller allows only a single page to be opened for write/erase. When writing multiple Flash pages, the Flash controller must go through the unlock sequence again to select another page.

20.3. Flash Control Register Definitions

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

Flash Control Register: see page 271

Flash Status Register: see page 272

Flash Page Select Register: see page 273

Flash Sector Protect Register: see page 274

Flash Frequency High and Low Byte Registers: see page 274

20.3.1. Flash Control Register

The Flash Controller must be unlocked using the Flash Control Register (see Table 134) before programming or erasing Flash memory. The Flash Controller is unlocked by writing to the Flash Page Select Register, then 73H 8CH, sequentially, to the Flash Control Register, and finally again to the Flash Page Select Register with the same value as the previous write. When the Flash Controller is unlocked, Mass Erase or Page Erase can be initiated by writing the appropriate command to the FCTL. Erase applies only to the active page selected in the Flash Page Select Register. Mass Erase is enabled only through the

Table 150. LVD_Trim Values (Continued)

	LVD Threshold (V)			
LVD_TRIM	**Minimum**	**Typical**	**Maximum**	**Description**
01101		2.55		
01110		2.50		
01111		2.45		
10000		2.40		
10001		2.35		
10010		2.30		
10011		2.25		
10100		2.20		
10101		2.15		
10110		2.10		
10111		2.05		
11000		2.00		
11001		1.95		
11010		1.90		
11011		1.85		
11100		1.80		
11101		1.75		
11110		1.70		
11111		1.65		Minimum LVD threshold, default on Reset.

21.2.5. Zilog Calibration Option Bits

This section describes the calibration of the temperature sensor's Low and High bytes.

21.2.5.1. Temperature Sensor Calibration High and Low Byte Registers

Tables 157 and 158 present the Temperature Sensor Calibration High and Low Byte registers.

Table 157. Temperature Sensor Calibration High Byte at FE60H (TEMPCALH)

Bits	7	6	5	4	3	2	1	0
Field	TEMPCALH							
Reset	U	U	U	U	U	U	U	U
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	Information Page Memory FE60H							

Note: U = Unchanged by Reset. R/W = Read/Write.

Bit	Description
[7:0] TEMPCALH	**Temperature Sensor Calibration High Byte** Bits [7:3] of this register are not used. Bit 2 indicates whether the calibration data is added to or subtracted from the measured ADC data. If bit 2 is 0, the calibration data is added; if bit 2 is 1, the calibration data is subtracted. Bits 1 and 0 are the High two bits of the 10-bit calibration data.

Table 158. Temperature Sensor Calibration Low Byte at FE61H (TEMPCALL)

Bits	7	6	5	4	3	2	1	0
Field	TEMPCALL							
Reset	U	U	U	U	U	U	U	U
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	Information Page Memory FE61H							

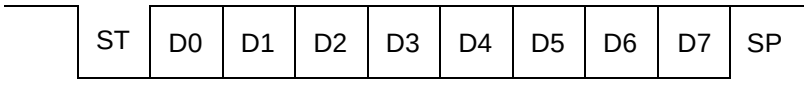
Note: U = Unchanged by Reset. R/W = Read/Write.

Bit	Description
[7:0] TEMPCALL	**Temperature Sensor Calibration Low Byte** TEMPCALL is the low eight bits of 10-bit calibration data. The entire 10-bit calibration data field is {TEMPCALH[1:0], TEMPCALL}.

- Voltage Brown-Out reset
- Asserting the $\overline{\text{RESET}}$ pin Low to initiate a Reset
- Driving the DBG pin Low when the device is in STOP Mode initiates a System Reset

23.2.2. OCD Data Format

The On-Chip Debugger (OCD) interface uses the asynchronous data format defined for RS-232. Each character is transmitted as 1 start bit, 8 data bits (least-significant bit first) and 1 stop bit (see Figure 59).



ST = Start bit
SP = Stop bit
D0–D7 = Data bits

Figure 59. OCD Data Format

23.2.3. OCD Autobaud Detector/Generator

To run over a range of baud rates (bits per second) with various system clock frequencies, the On-Chip Debugger has an Autobaud Detector/Generator. After a reset, the OCD is idle until it receives data. The OCD requires that the first character sent from the host is the character 80H. The character 80H contains eight continuous bits Low (one start bit plus 7 data bits). The Autobaud Detector measures this period and sets the OCD Baud Rate Generator accordingly.

The Autobaud Detector/Generator is clocked by the system clock. The minimum baud rate is the system clock frequency divided by 512. If the data can be synchronized with the system clock, the autobaud generator can run as high as the system clock frequency (1 clock/bit). The maximum recommended baud rate is the system clock frequency divided by 8. Table 162 lists the minimum and recommended maximum baud rates for sample crystal frequencies.

Figures 69 through 72 display the typical current consumption at voltages of 1.8V, 2.0V, 2.7V, 3.0V, 3.3V and 3.6V, respectively, versus different system clock frequencies while operating at a temperature of 25ºC.

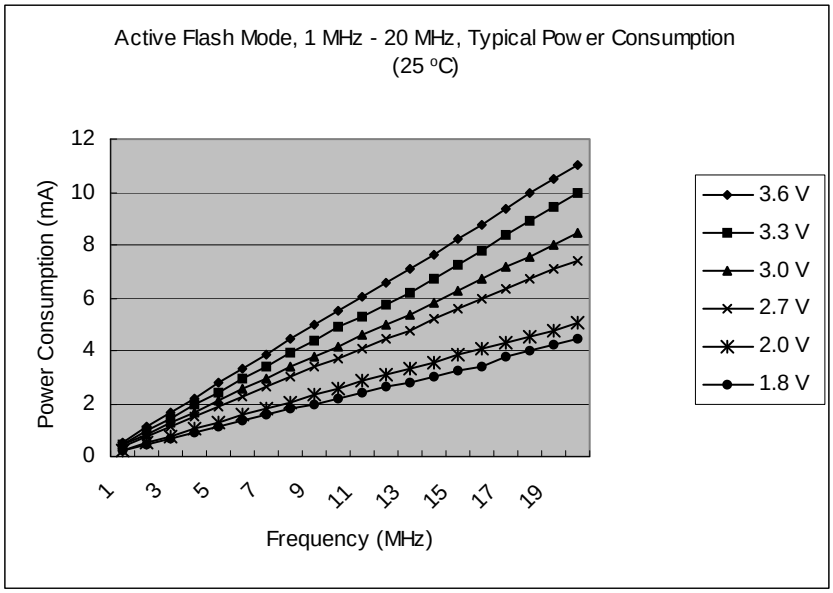


Figure 69. Typical Active Flash Mode Supply Current (1–20MHz)

Table 210. Package and Pin Count Description

	Pin Count			
Package	**20**	**28**	**40**	**44**
SOIC	√	√		
SSOP	√	√		
PDIP	√	√	√	
LQFP				√
QFN				√

31.1.0.1. Precharacterization Product

The product represented by this document is newly introduced and Zilog® has not completed the full characterization of the product. The document states what Zilog knows about this product at this time, but additional features or nonconformance with some aspects of the document might be found, either by Zilog or its customers in the course of further application and characterization work. In addition, Zilog cautions that delivery might be uncertain at times, because of start-up yield issues.

Index