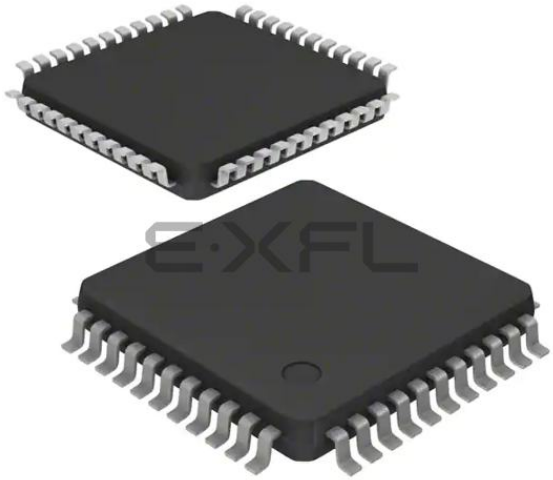

Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	eZ8
Core Size	8-Bit
Speed	20MHz
Connectivity	I²C, IrDA, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, LED, LVD, POR, PWM, Temp Sensor, WDT
Number of I/O	37
Program Memory Size	24KB (24K 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	44-LQFP
Supplier Device Package	44-LQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/zilog/z8f2480an020sg

need to use this on-chip Program RAM to shadow Interrupt Service Routines (ISR). For details, see the PRAM_M section on page 278.

3.2. Program Memory

The eZ8 CPU supports 64KB of Program Memory address space. The F1680 Series MCU contains 8KB to 24KB of on-chip Flash memory in the Program Memory address space, depending on the device.

In addition, the F1680 Series MCU contains up to 1KB of on-chip Program RAM. The Program RAM is mapped in the Program Memory address space beyond the on-chip Flash memory. The Program RAM is entirely under user control and is meant to store interrupt service routines of high-frequency interrupts. Since interrupts bring the CPU out of low-power mode, it is important to ensure that interrupts that occur very often use as low a current as possible. For battery operated systems, Program RAM based handling of high-frequency interrupts provides power savings by keeping the Flash block disabled. Program RAM (PRAM) is optimized for low-current operation and can be easily boot-strapped with interrupt code at power up.

Reading from Program Memory addresses present outside the available Flash memory and PRAM addresses returns FFH. Writing to these unimplemented Program Memory addresses produces no effect. Table 6 describes the Program Memory maps for the F1680 Series MCU.

Table 6. F1680 Series MCU Program Memory Maps

Program Memory Address (Hex)	Function
Z8F2480 Device	
0000–0001	Flash option bits
0002–0003	Reset vector
0004–0005	WDT interrupt vector
0006–0007	Illegal instruction trap
0008–0037	Interrupt vectors*
0038–003D	Oscillator fail traps*
003E–5FFF	Program Flash
E000–E3FF	1KB PRAM

Note: *See Table 36 on page 69 for a list of interrupt vectors and traps.

Table 18. Port Alternate Function Mapping, 28-Pin Parts[1,2]

Port	Pin	Mnemonic	Alternate Function Description	Alternate Function Set Register AFS1
Port A	PA0	T0IN/$\overline{\text{T0OUT}}$	Timer 0 Input/Timer 0 Output Complement	AFS1[0]: 0
		Reserved		AFS1[0]: 1
	PA1	T0OUT	Timer 0 Output	AFS1[1]: 0
		Reserved		AFS1[1]: 1
	PA2	DE0	UART 0 Driver Enable	AFS1[2]: 0
		Reserved		AFS1[2]: 1
	PA3	$\overline{\text{CTS0}}$	UART 0 Clear to Send	AFS1[3]: 0
		Reserved		AFS1[3]: 1
	PA4	RXD0/IRRX0	UART 0/IrDA 0 Receive Data	AFS1[4]: 0
		Reserved		AFS1[4]: 1
	PA5	TXD0/IRTX0	UART 0/IrDA 0 Transmit Data	AFS1[5]: 0
		Reserved		AFS1[5]: 1
	PA6	T1IN/$\overline{\text{T1OUT}}$	Timer 1 Input/Timer 1 Output Complement	AFS1[6]: 0
		SCL	I^2C Serial Clock	AFS1[6]: 1
	PA7	T1OUT	Timer 1 Output	AFS1[7]: 0
		SDA	I^2C Serial Data	AFS1[7]: 1

Notes:
1. Because there are at most two choices of alternate functions for some pins in Ports A and B, the Alternate Function Set Register (AFS2) is implemented but not used to select the function. The alternate function selection must also be enabled, as described in the Port A–E Alternate Function Subregisters section on page 61.
2. Because there is only one alternate function for each Port D and Port E pin, the Alternate Function Set registers are not implemented for Ports D and E. Enabling the alternate function selections automatically enables the associated alternate function, as described in the Port A–E Alternate Function Subregisters section on page 61.

8.4.7. Interrupt Edge Select Register

The Interrupt Edge Select (IRQES) Register, shown in Table 49, determines whether an interrupt is generated for the rising edge or falling edge on the selected GPIO Port A or Port D input pin.

Table 49. Interrupt Edge Select Register (IRQES)

Bits	7	6	5	4	3	2	1	0
Field	IES7	IES6	IES5	IES4	IES3	IES2	IES1	IES0
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	FCDH							

Bit	Description
[7:0] IES*x*	**Interrupt Edge Select *x*** 0 = An interrupt request is generated on the falling edge of the PA*x* input or PD*x* input. 1 = An interrupt request is generated on the rising edge of the PA*x* input or PD*x* input; *x* indicates the specific GPIO port pin number (0–7).

8.4.8. Shared Interrupt Select Register

The Shared Interrupt Select (IRQSS) Register, shown in Table 50, determines the source of the PAD*x*S interrupts. The Shared Interrupt Select Register selects between Port A and alternate sources for the individual interrupts.

Table 50. Shared Interrupt Select Register (IRQSS)

Bits	7	6	5	4	3	2	1	0
Field	PA7VS	PA6CS	PA5CS	PAD4S	PAD3S	PAD2S	PAD1S	Reserved
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	FCEH							

Bit	Description
[7] PA7VS	**PA7/LVD Selection** 0 = PA7 is used for the interrupt for PA7VS interrupt request. 1 = The LVD is used for the interrupt for PA7VS interrupt request.
[6] PA6CS	**PA6/Comparator 0 Selection** 0 = PA6 is used for the interrupt for PA6CS interrupt request. 1 = The Comparator 0 is used for the interrupt for PA6CS interrupt request.

9.1. Architecture

Figure 11 displays the architecture of the timers.

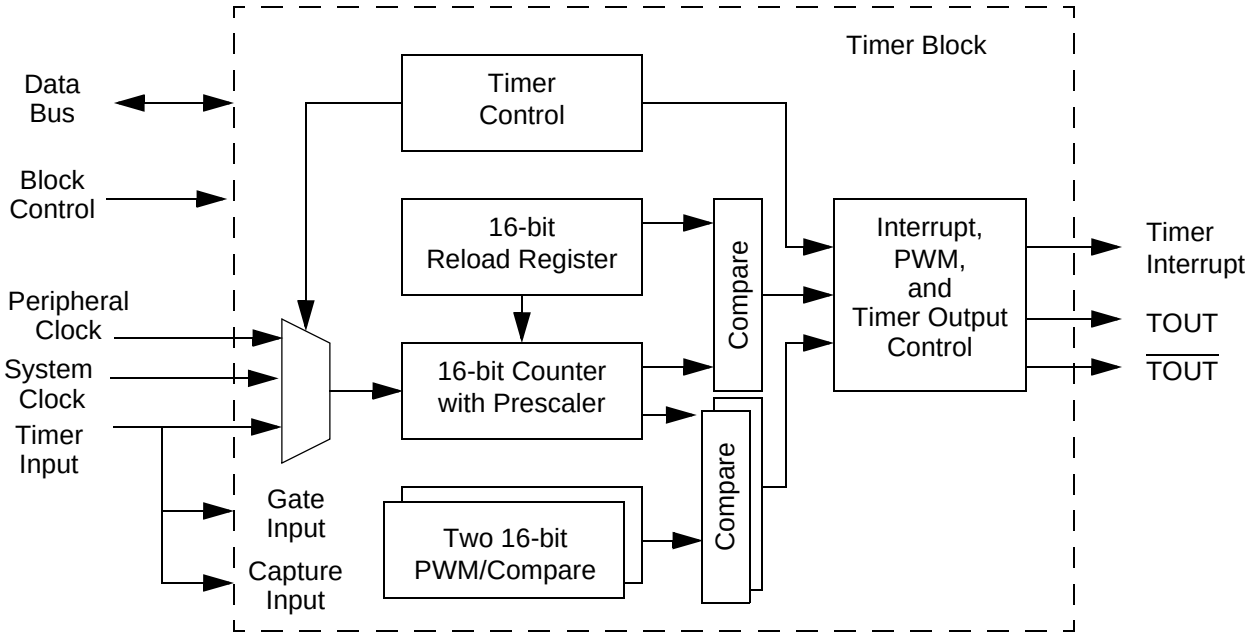

Figure 11. Timer Block Diagram

9.2. Operation

The timers are 16-bit up-counters. Minimum time-out delay is set by loading the value 0001H into the Timer Reload High and Low Byte registers and setting the prescale value to 1. Maximum time-out delay is set by loading the value 0000H into the Timer Reload High and Low Byte registers and setting the prescale value to 128. If the Timer reaches FFFFH, the timer rolls over to 0000H and continues counting.

9.2.1. Timer Clock Source

The timer clock source can come from either the peripheral clock or the system clock. Peripheral clock is based on a low frequency/low power 32 kHz secondary oscillator that can be used with external watch crystal. Peripheral clock source is only available for driving Timer and Noise Filter operation. It is not supported for other peripherals.

For timer operation in STOP Mode, peripheral clock must be selected as the clock source. Peripheral clock can be selected as source for both ACTIVE and STOP Mode operation.

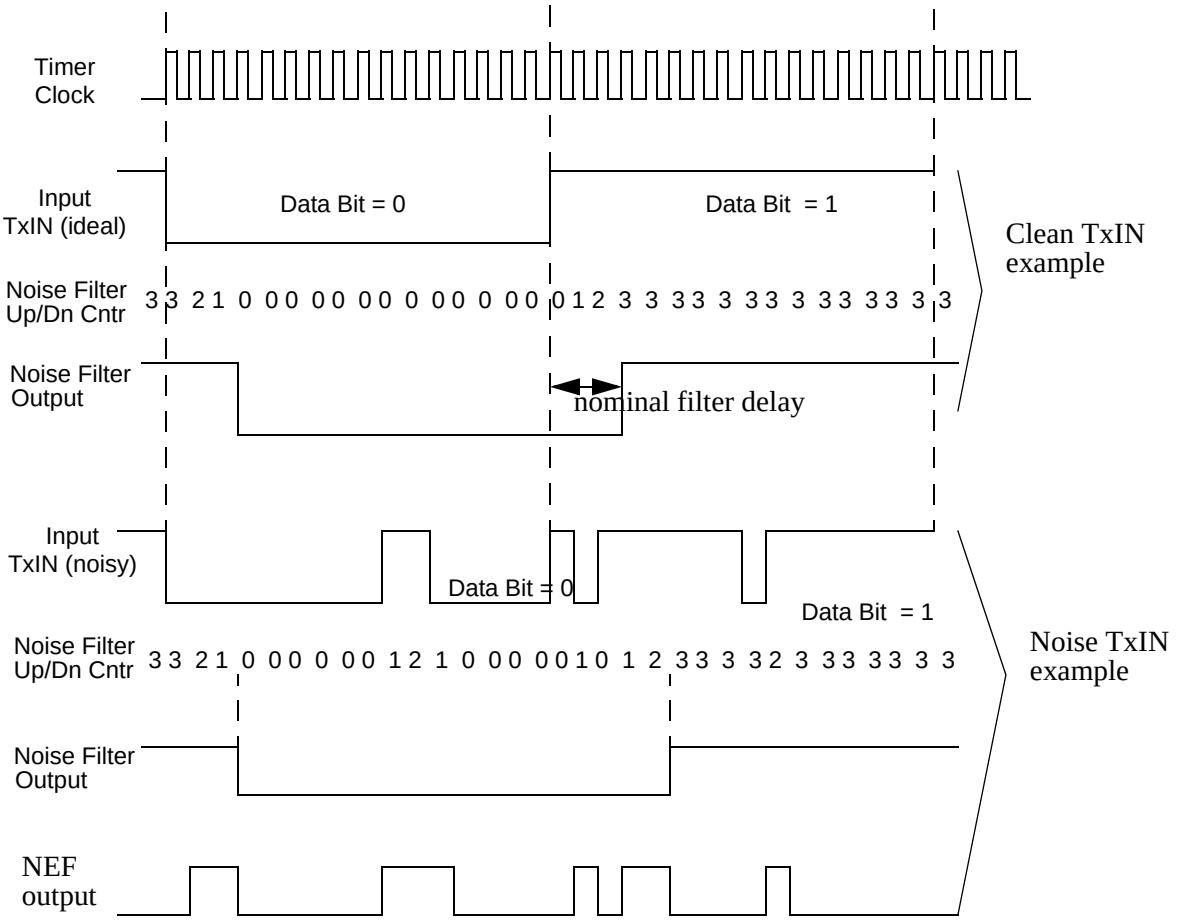

Figure 13. Noise Filter Operation

9.3. Timer Control Register Definitions

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

Timer 0–2 High and Low Byte Registers: see page 109

Timer Reload High and Low Byte Registers: see page 109

Timer 0–2 PWM0 High and Low Byte Registers: see page 110

Timer 0–2 PWM1 High and Low Byte Registers: see page 111

Timer 0–2 Control Registers: see page 112

Timer 0–2 Status Registers: see page 118

Timer 0–2 Noise Filter Control Register: see page 119

Bit	Description (Continued)
[5] CHIEN	**Channel Interrupt Enable** This bit enables generation of channel interrupt. A channel interrupt is generated whenever there is a capture/compare event on the Timer Channel. 0 = Channel interrupt is disabled. 1 = Channel interrupt is enabled.
[4] CHUE	**Channel Update Enable** This bit determines whether writes to the Channel High and Low Byte registers are buffered when TEN = 1. Writes to these registers are not buffered when TEN = 0 regardless of the value of this bit. 0 = Writes to the Channel High and Low Byte registers are buffered when TEN = 1 and only take affect on the next end of cycle count. 1 = Writes to the Channel High and Low Byte registers are not buffered when TEN = 1.
[3]	Reserved; must be 0.
[2:0] CHOP	**Channel Operation Method** This field determines the operating mode of the channel. For a detailed description of the operating modes, see Count Up/Down Mode on page 123. 000 = One-Shot Compare operation. 001 = Continuous Compare operation. 010 = PWM Output operation. 011 = Capture operation. 100 – 111 = Reserved.

12.3. LIN-UART Control Register Definitions

The LIN-UART control registers support the LIN-UART, the associated Infrared Encoder/Decoder and the noise filter. For more information about the infrared operation, see the Infrared Encoder/Decoder section on page 182.

12.3.1. LIN-UART Transmit Data Register

Data bytes written to the LIN-UART Transmit Data Register, shown in Table 83, are shifted out on the TxD pin. The write-only LIN-UART Transmit Data Register shares a Register File address with the read-only LIN-UART Receive Data Register.

Table 83. LIN-UART Transmit Data Register (U0TXD = F40H)

Bit	7	6	5	4	3	2	1	0
Field	TxD							
Reset	X	X	X	X	X	X	X	X
R/W	W	W	W	W	W	W	W	W
Address	F40H, F48H							

Note: W = Write; X = undefined.

Bit	Description
[7:0] TxD	**Transmit Data** LIN–UART transmitter data byte to be shifted out through the TxD pin.

Bit	Description (Continued)
[5] OE	**Receive Data and Autobaud Overrun Error** This bit is set just as in normal UART operation if a receive data overrun error occurs. This bit is also set during LIN Slave autobaud if the BRG counter overflows before the end of the autobaud sequence. This indicates that the receive activity is not an autobaud character or the master baud rate is too slow. The ATB status bit will also be set in this case. This bit is cleared by reading the Receive Data Register. 0 = No autobaud or data overrun error occurred. 1 = An autobaud or data overrun error occurred.
[4] FE	**Framing Error** This bit indicates that a framing error (no stop bit following data reception) is detected. Reading the Receive Data Register clears this bit. 0 = No framing error occurred. 1 = A framing error occurred.
[3] BRKD	**Break Detect** This bit is set in LIN mode if: • It is in Lin Sleep state and a break of at least 4 bit times occurred (Wake-up event) or • It is in Slave Wait Break state and a break of at least 11 bit times occurred (Break event) or • It is in Slave Active state and a break of at least 10 bit times occurs. Reading the Status 0 Register or the Receive Data Register clears this bit. 0 = No LIN break occurred. 1 = LIN break occurred.
[2] TDRE	**Transmitter Data Register Empty** This bit indicates that the Transmit Data Register is empty and ready for additional data. Writing to the Transmit Data Register resets this bit. 0 = Do not write to the Transmit Data Register. 1 = The Transmit Data Register is ready to receive an additional byte for transmission.
[1] TXE	**Transmitter Empty** This bit indicates that the Transmit Shift Register is empty and character transmission is completed. 0 = Data is currently transmitting. 1 = Transmission is complete.
[0] ATB	**LIN Slave Autobaud Complete** This bit is set in LIN SLAVE Mode when an autobaud character is received. If the ABIEN bit is set in the LIN Control Register, then a receive interrupt is generated when this bit is set. Reading the Status 0 Register clears this bit. This bit will be 0 in LIN MASTER Mode.

14.3.2. ADC Raw Data High Byte Register

The ADC Raw Data High Byte Register, shown in Table 102, contains the upper 8 bits of raw data from the ADC output. Access to the ADC Raw Data High Byte register is read-only. This register is used for test only.

Table 102. ADC Raw Data High Byte Register (ADCRD_H)

Bits	7	6	5	4	3	2	1	0
Field	ADCRDH							
Reset	X							
R/W	R							
Address	F71H							

Bit Position	Value (H)	Description
[7:0]	00–FF	**ADC Raw Data High Byte** The data in this register is the raw data coming from the SAR Block. It will change as the conversion is in progress. This register is used for testing only.

14.3.3. ADC Data High Byte Register

The ADC Data High Byte Register, shown in Table 103, contains the upper eight bits of the ADC output. Access to the ADC Data High Byte Register is read-only. Reading the ADC Data High Byte Register latches data in the ADC Low Bits Register.

Table 103. ADC Data High Byte Register (ADCD_H)

Bits	7	6	5	4	3	2	1	0
Field	ADCDH							
Reset	X							
R/W	R							
Address	F72H							

Bit Position	Value (H)	Description
[7:0]	00–FF	**ADC High Byte** The last conversion output is held in the data registers until the next ADC conversion has completed.

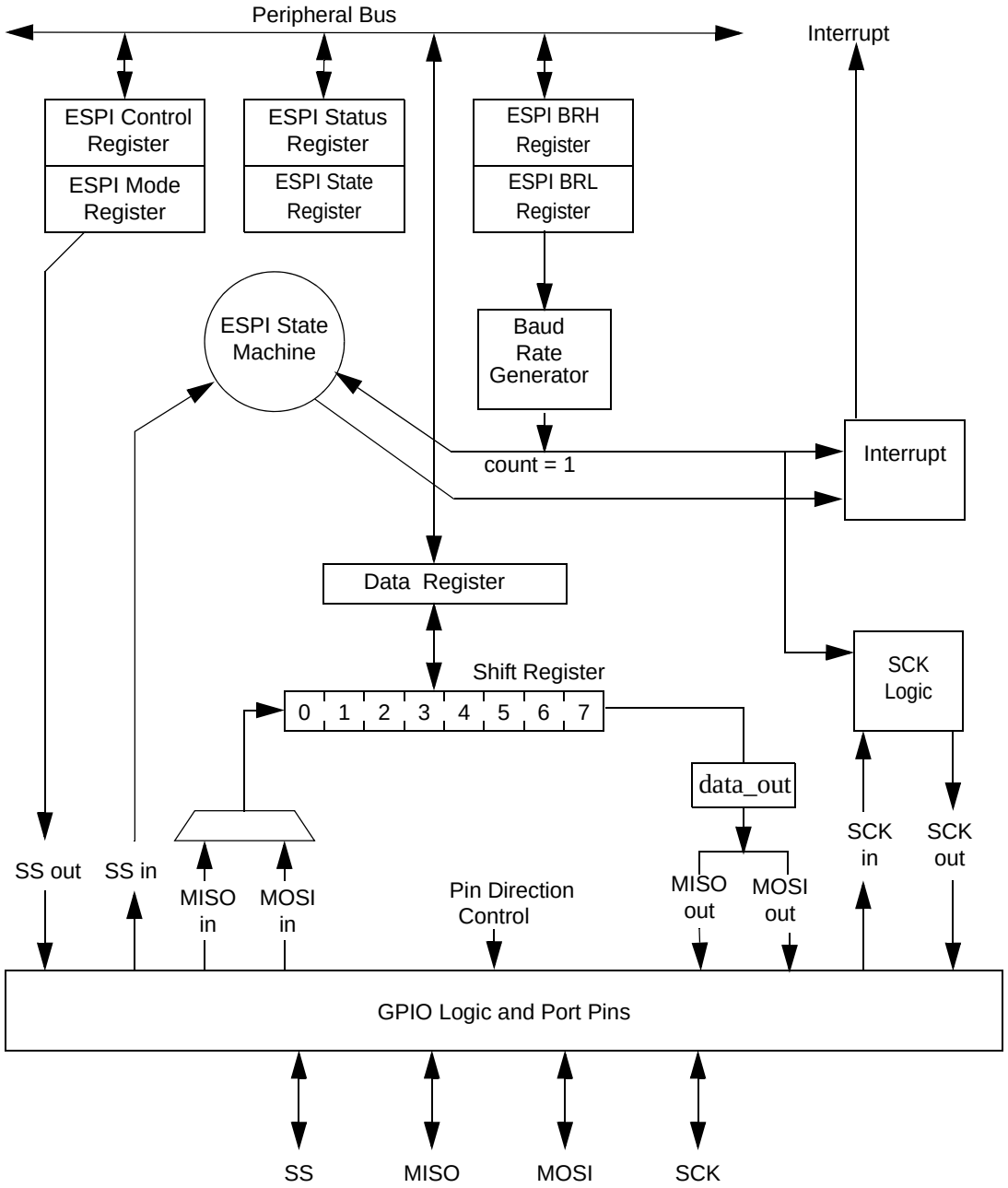

Figure 33. ESPI Block Diagram

17.2.3. Start and Stop Conditions

The Master generates the start and stop conditions to start or end a transaction. To start a transaction, the I^2C controller generates a start condition by pulling the SDA signal Low while SCL is High. To complete a transaction, the I^2C controller generates a stop condition by creating a Low-to-High transition of the SDA signal while the SCL signal is High. These start and stop events occur when the start and stop bits in the I^2C Control Register are written by software to begin or end a transaction. Any byte transfer currently under way including the Acknowledge phase finishes before the start or stop condition occurs.

17.2.4. Software Control of I^2C Transactions

The I^2C controller is configured via the I^2C Control and I^2C Mode registers. The MODE[1:0] field of the I^2C Mode Register allows the configuration of the I^2C controller for MASTER/SLAVE or SLAVE ONLY mode and configures the slave for 7-bit or 10-bit addressing recognition.

MASTER/SLAVE Mode can be used for:

- MASTER ONLY operation in a Single Master/One or More Slave I^2C system
- MASTER/SLAVE in a Multimaster/multislave I^2C system
- SLAVE ONLY operation in an I^2C system

In SLAVE ONLY mode, the start bit of the I^2C Control Register is ignored (software cannot initiate a master transaction by accident) and operation to SLAVE ONLY Mode is restricted thereby preventing accidental operation in MASTER Mode. The software controls I^2C transactions by enabling the I^2C controller interrupt in the interrupt controller or by polling the I^2C Status Register.

To use interrupts, the I^2C interrupt must be enabled in the interrupt controller and followed by executing an EI instruction. The TXI bit in the I^2C Control Register must be set to enable transmit interrupts. An I^2C interrupt service routine then checks the I^2C Status Register to determine the cause of the interrupt.

To control transactions by polling, the TDRE, RDRF, SAM, ARBLST, SPRS and NCKI interrupt bits in the I^2C Status Register should be polled. The TDRE bit asserts regardless of the state of the TXI bit.

17.2.5. Master Transactions

The following sections describe Master Read and Write transactions to both 7-bit and 10-bit slaves.

14. The software responds by writing the data to be written out to the I^2C Control Register.
15. The I^2C controller shifts out the remainder of the second byte of the slave address (or ensuring data bytes, if looping) via the SDA signal.
16. The I^2C slave sends an Acknowledge by pulling the SDA signal Low during the next High period of SCL. The I^2C controller sets the ACK bit in the I^2C Status Register. If the slave does not acknowledge, see the second paragraph of Step 11.
17. The I^2C controller shifts the data out by the SDA signal. After the first bit is sent, the transmit interrupt asserts.
18. If more bytes remain to be sent, return to Step 14.
19. The software responds by asserting the stop bit of the I^2C Control Register.
20. The I^2C controller completes transmission of the data on the SDA signal.
21. The I^2C controller sends a stop condition to the I^2C bus.

Note: If the slave responds with a Not Acknowledge during the transfer, the I^2C controller asserts the NCKI bit, sets the ACKV bit, clears the ACK bit in the I^2C State Register and halts. The software terminates the transaction by setting either the stop bit (end transaction) or the start bit (end this transaction, start a new one). The Transmit Data Register is flushed automatically.

17.2.5.6. Master Read Transaction with a 7-Bit Address

Figure 45 displays the data transfer format for a Read operation to a 7-bit addressed slave.

S	Slave Address	R = 1	A	Data	A	Data	A	P/S

Figure 45. Data Transfer Format—Master Read Transaction with a 7-Bit Address

Observe the following steps for a Master Read operation to a 7-bit addressed slave:

1. The software initializes the MODE field in the I^2C Mode Register for MASTER/SLAVE Mode with 7- or 10-bit addressing (the I^2C bus protocol allows the mixing of slave address types). The MODE field selects the address width for this mode when addressed as a slave (but not for the remote slave). The software asserts the IEN bit in the I^2C Control Register.
2. The software writes the I^2C Data Register with a 7-bit slave address, plus the Read bit (which is set to 1).
3. The software asserts the start bit of the I^2C Control Register.

17.3.2. I²C Interrupt Status Register

The read-only I²C Interrupt Status Register, shown in Table 120, indicates the cause of any current I²C interrupt and provides status of the I²C controller. When an interrupt occurs, one or more of the TDRE, RDRF, SAM, ARBLST, SPRS or NCKI bits is set. The GCA and RD bits do not generate an interrupt but rather provide status associated with the SAM bit interrupt.

Table 120. I²C Interrupt Status Register (I2CISTAT = F51H)

Bits	7	6	5	4	3	2	1	0
Field	TDRE	RDRF	SAM	GCA	RD	ARBLST	SPRS	NCKI
Reset	1	0	0	0	0	0	0	0
R/W	R	R	R	R	R	R	R	R
Address	F51H							

Bit	Description
[7] TDRE	**Transmit Data Register Empty** When the I²C controller is enabled, this bit is 1 when the I²C Data Register is empty. When set, this bit causes the I²C controller to generate an interrupt, except when the I²C controller is shifting in data during the reception of a byte or when shifting an address and the RD bit is set. This bit clears by writing to the I2CDATA Register.
[6] RDRF	**Receive Data Register Full** This bit is set = 1 when the I²C controller is enabled and the I²C controller has received a byte of data. When asserted, this bit causes the I²C controller to generate an interrupt. This bit clears by reading the I2CDATA Register.
[5] SAM	**Slave Address Match** This bit is set = 1 if the I²C controller is enabled in SLAVE Mode and an address is received that matches the unique slave address or General Call Address (if enabled by the GCE bit in the I²C Mode Register). In 10-bit addressing mode, this bit is not set until a match is achieved on both address bytes. When this bit is set, the RD and GCA bits are also valid. This bit clears by reading the I2CISTAT Register.
[4] GCA	**General Call Address** This bit is set in SLAVE Mode when the General Call Address or Start byte is recognized (in either 7 or 10 bit SLAVE Mode). The GCE bit in the I²C Mode Register must be set to enable recognition of the General Call Address and Start byte. This bit clears when IEN = 0 and is updated following the first address byte of each SLAVE Mode transaction. A General Call Address is distinguished from a Start byte by the value of the RD bit (RD = 0 for General Call Address, 1 for Start byte).
[3] RD	**Read** This bit indicates the direction of transfer of the data. It is set when the Master is reading data from the Slave. This bit matches the least-significant bit of the address byte after the start condition occurs (for both MASTER and SLAVE modes). This bit clears when IEN = 0 and is updated following the first address byte of each transaction.

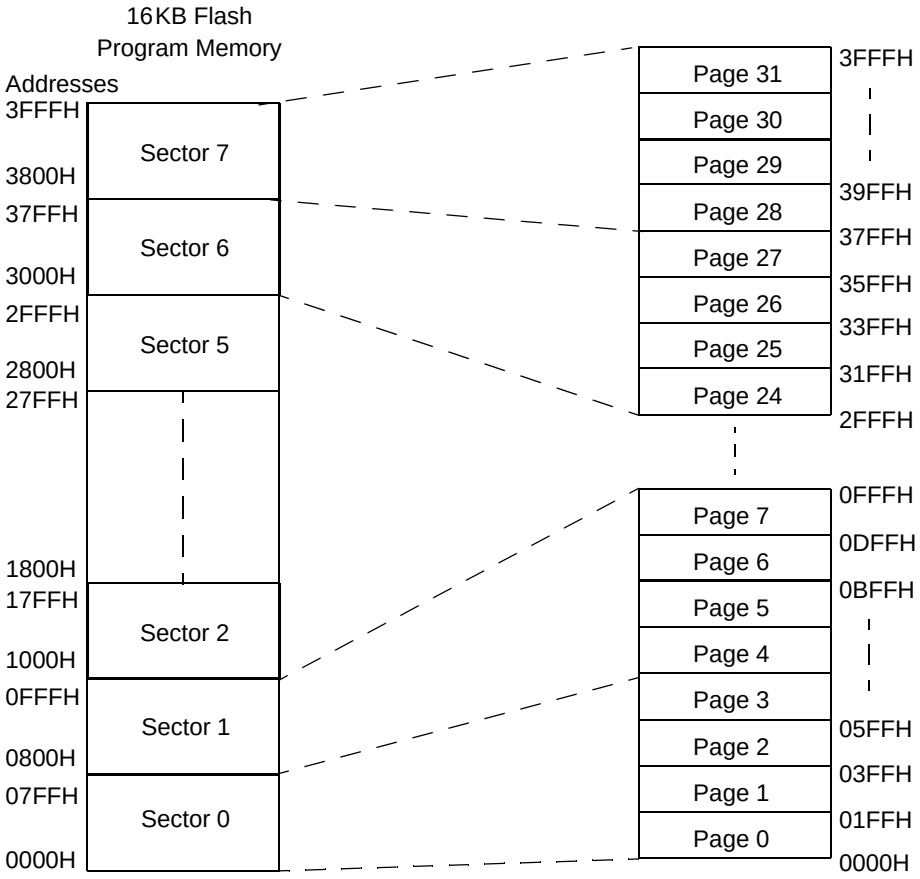

Figure 52. 16KB Flash Memory Arrangement

Read Baud Reload Register (1BH). The Read Baud Reload Register command returns the current value in the Baud Reload Register.

```
DBG ← 1BH
DBG → BAUD[15:8]
DBG → BAUD[7:0]
```

Write Test Mode Register (F0H). The Write Test Mode Register command writes the data that follows to the Test Mode register.

```
DBG ← F0H
DBG ← TESTMODE[7:0]
```

Read Test Mode Register (F1H). The Read Test Mode Register command returns the current value in the Test Mode register.

```
DBG ← F1H
DBG → TESTMODE[7:0]
```

Write Option Bit Registers (F2H). The Write Option Bit Registers command is used to write to the option bit configuration registers. The option bit configuration registers store the device configuration and are loaded from Flash every time the Z8 Encore! XP F1680 Series is reset. The registers may be temporarily written using this OCD command to test peripherals without having to program the Flash Information Area and resetting the Z8 Encore! XP F1680 Series. The ZilogUserSel bit selects between Zilog option bits (1) and user option bits (0).

```
DBG ← F2H
DBG ← {ZilogUserSel,1'b0,OptAddr[4:0]}
DBG ← OptData[7:0]
```

Read Option Bit Registers (F3H). The Read Option Bit Registers command is used to read the option bit registers that store the device configuration that is read out of flash when the Z8 Encore! XP F1680 Series is reset. The ZilogUserSel bit selects between reading Zilog option bits (1) or user option bits (0).

```
DBG ← F1H
DBG ← {ZilogUserSel,1'b0,OptAddr[4:0]}
DBG → OptData[7:0]
```

23.4. On-Chip Debugger Control Register Definitions

This section defines the features of the following On-Chip Debugger Control registers.

OCD Control Register: see page 310

OCD Status Register: see page 312

Line Control Register: see page 313

Baud Reload Register: see page 314

Tables 178 through 185 contain the instructions belonging to each group and the number of operands required for each instruction. Some instructions appear in more than one table as these instructions are to be considered as a subset of more than one category. Within these tables, the source operand is identified as src, the destination operand is dst and a condition code is cc.

Table 178. Arithmetic Instructions

Mnemonic	Operands	Instruction
ADC	dst, src	Add with Carry
ADCX	dst, src	Add with Carry using Extended Addressing
ADD	dst, src	Add
ADDX	dst, src	Add using Extended Addressing
CP	dst, src	Compare
CPC	dst, src	Compare with Carry
CPCX	dst, src	Compare with Carry using Extended Addressing
CPX	dst, src	Compare using Extended Addressing
DA	dst	Decimal Adjust
DEC	dst	Decrement
DECW	dst	Decrement Word
INC	dst	Increment
INCW	dst	Increment Word
MULT	dst	Multiply
SBC	dst, src	Subtract with Carry
SBCX	dst, src	Subtract with Carry using Extended Addressing
SUB	dst, src	Subtract
SUBX	dst, src	Subtract using Extended Addressing

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.

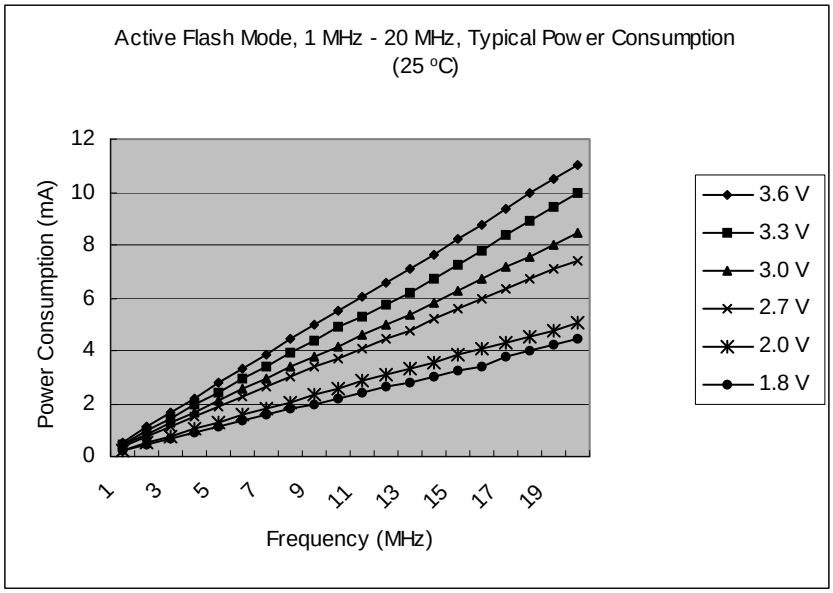

Figure 69. Typical Active Flash Mode Supply Current (1–20MHz)