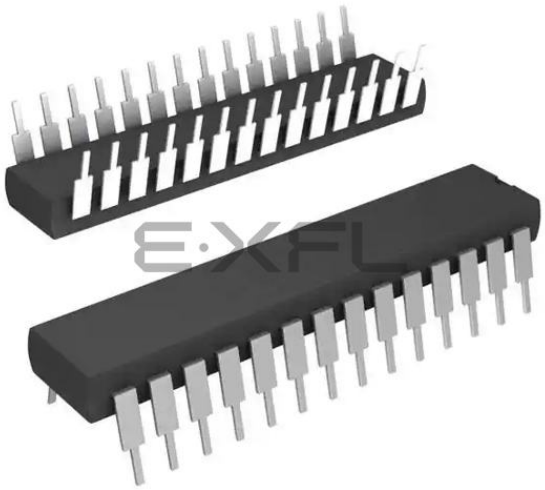

Welcome to E-XFL.COM

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	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 8x10b
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/z8f0880pj020sg

List of Figures

1.4. An Overview of the eZ8 CPU and its Peripherals

Zilog's eZ8 CPU, latest 8-bit CPU meets the continuing demand for faster and more code-efficient microcontrollers. It executes a superset of the original Z8® instruction set. The eZ8 CPU features include:

- Direct register-to-register architecture allows each register to function as an accumulator, improving execution time and decreasing the required program memory
- Software stack allows greater depth in subroutine calls and interrupts more than hardware stacks
- Compatible with existing Z8 code
- Expanded internal Register File allows access up to 4KB
- New instructions improve execution efficiency for code developed using higher-level programming languages including C
- Pipelined instruction fetch and execution
- New instructions for improved performance including BIT, BSWAP, BTJ, CPC, LDC, LDCI, LEA, MULT and SRL
- New instructions support 12-bit linear addressing of the register file
- Up to 10 MIPS operation
- C-Compiler friendly
- 2 to 9 clock cycles per instruction

For more details about eZ8 CPU, refer to the eZ8 CPU Core User Manual (UM0128), available for download at www.zilog.com.

1.4.1. General-Purpose Input/Output

The F1680 MCU features 17 to 37 port pins (Ports A–E) for general purpose input/output (GPIO) pins. The number of GPIO pins available is a function of package. Each pin is individually programmable.

1.4.2. Flash Controller

The Flash Controller is used to program and erase Flash memory. The Flash Controller supports protection against accidental program and erasure.

writing this bit does not clear it. The LVD circuit can also generate an interrupt when enabled (see the Interrupt Vectors and Priority section on page 71). The LVD is not latched, so enabling the interrupt is the only way to guarantee detection of a transient low-voltage event.

The LVD circuit is either enabled or disabled by the Power Control Register bit 4. For more details, see the Power Control Register Definitions section on page 44.

5.5. Reset Register Definitions

The following sections define the Reset registers.

5.5.0.1. Reset Status Register

The Reset Status (RSTSTAT) Register, shown in Table 12, is a read-only register that indicates the source of the most recent Reset event, Stop Mode Recovery event and/or WDT time-out. Reading this register resets the upper 4 bits to 0.

This register shares its address with the Reset Status Register, which is write-only.

Table 12. Reset Status Register (RSTSTAT)

Bits	7	6	5	4	3	2	1	0
Field	POR/VBO	STOP	WDT	EXT	Reserved			LVD
Reset	See Table 13			0	0	0	0	0
R/W	R	R	R	R	R	R	R	R
Address	FF0H							

Bit	Description
[7] POR/VBO	**Power-On initiated VBO Reset or general VBO Reset Indicator** If this bit is set to 1, a POR or VBO Reset event occurs. This bit is reset to 0, if a WDT time-out or Stop Mode Recovery occurs. This bit is also reset to 0 when the register is read.
[6] STOP	**Stop Mode Recovery Indicator** If this bit is set to 1, a Stop Mode Recovery occurs. If the STOP and WDT bits are both set to 1, the Stop Mode Recovery occurs because of a WDT time-out. If the stop bit is 1 and the WDT bit is 0, the Stop Mode Recovery is not caused by a WDT time-out. This bit is reset by Power-On Reset or WDT time-out that occurred while not in STOP Mode. Reading this register also resets this bit.
[5] WDT	**Watchdog Timer time-out Indicator** If this bit is set to 1, a WDT time-out occurs. A POR resets this pin. A Stop Mode Recovery from a change in an input pin also resets this bit. Reading this register resets this bit. This Read must occur before clearing the WDT interrupt.

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

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

Bit	Description (Continued)
[6] TPOL	**Timer Input/Output Polarity** Operation of this field is a function of the current operating modes of the timer. **ONE-SHOT Mode** When the timer is disabled, the Timer Output signal is set to the value of this bit. When the timer is enabled, the Timer Output signal is complemented upon timer reload. **CONTINUOUS Mode** When the timer is disabled, the Timer Output signal is set to the value of this bit. When the timer is enabled, the Timer Output signal is complemented upon timer reload. **COUNTER Mode** When the timer is disabled, the Timer Output signal is set to the value of this bit. When the timer is enabled, the Timer Output signal is complemented upon timer reload. 0 = Count occurs on the rising edge of the Timer Input signal. 1 = Count occurs on the falling edge of the Timer Input signal. **PWM SINGLE OUTPUT Mode** 0 = Timer Output is forced Low (0) when the timer is disabled. When enabled, the Timer Output is forced High (1) on PWM count match and forced Low (0) on Reload. 1 = Timer Output is forced High (1) when the timer is disabled. When enabled, the Timer Output is forced Low (0) on PWM count match and forced High (1) on Reload. **CAPTURE Mode** 0 = Count is captured on the rising edge of the Timer Input signal. 1 = Count is captured on the falling edge of the Timer Input signal. **COMPARE Mode** When the timer is disabled, the Timer Output signal is set to the value of this bit. When the timer is enabled, the Timer Output signal is complemented on timer reload. **GATED Mode** 0 = Timer counts when the Timer Input signal is High (1) and interrupts are generated on the falling edge of the Timer Input. 1 = Timer counts when the Timer Input signal is Low (0) and interrupts are generated on the rising edge of the Timer Input. **CAPTURE/COMPARE Mode** 0 = Counting is started on the first rising edge of the Timer Input signal. The current count is captured on subsequent rising edges of the Timer Input signal. 1 = Counting is started on the first falling edge of the Timer Input signal. The current count is captured on subsequent falling edges of the Timer Input signal.

Table 72. Multi-Channel Timer Subaddress Register (MCTSA)

Bit	7	6	5	4	3	2	1	0
Field	MCTSA							
Reset	X	X	X	X	X	X	X	X
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	FA4H							

10.7.5. Multi-Channel Timer Subregister x (0, 1, or 2)

The Multi-Channel Timer subregisters 0, 1 or 2 store the 8-bit data write to subregister or 8-bit data read from subregister. The Multi-Channel Timer Subaddress Register selects the subregister to be written to or read from.

Table 73. Multi-Channel Timer Subregister x (MCTSRx)

Bit	7	6	5	4	3	2	1	0
Field	MCTSRx							
Reset	X	X	X	X	X	X	X	X
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	FA5H, FA6H, FA7H							

10.7.6. Multi-Channel Timer Control 0, Control 1 Registers

The Multi-Channel Timer Control registers (MCTCTL0, MCTCTL1) control Multi-Channel Timer operation. Writes to the PRES field of the MCTCTL1 Register are buffered when TEN = 1 and will not take effect until the next end of the cycle count occurs.

Table 74. Multi-Channel Timer Control 0 Register (MCTCTL0)

Bit	7	6	5	4	3	2	1	0
Field	TCTST	CHST	TCIEN	Reserved	Reserved	TCLKS		
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R	R/W	R	R	R/W	R/W	R/W
Address	See note.							
Note: If a 00H is in the Subaddress Register, it is accessible through Subregister 0.								

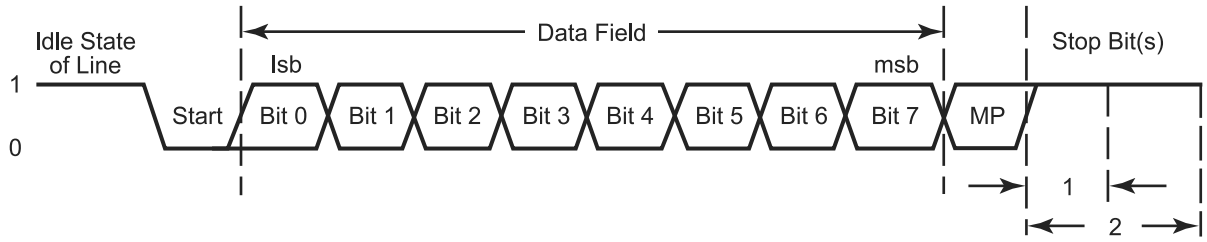

Figure 23. LIN-UART Asynchronous MULTIPROCESSOR Mode Data Format

In MULTIPROCESSOR (9-bit) Mode, the Parity bit location (9^{th} bit) becomes the Multiprocessor control bit. The LIN-UART Control 1 and Status 1 registers provide MULTIPROCESSOR (9-bit) Mode control and status information. If an automatic address matching scheme is enabled, the LIN-UART Address Compare register holds the network address of the device.

12.1.9.1. MULTIPROCESSOR Mode Receive Interrupts

When MULTIPROCESSOR (9-bit) Mode is enabled, the LIN-UART processes only frames addressed to it. To determine whether a frame of data is addressed to the LIN-UART can be made in hardware, software or a combination of the two, depending on the multiprocessor configuration bits. In general, the address compare feature reduces the load on the CPU, because it is not required to access the LIN-UART when it receives data directed to other devices on the multinode network. The following three MULTIPROCESSOR Modes are available in hardware:

- Interrupt on all address bytes
- Interrupt on matched address bytes and correctly framed data bytes
- Interrupt only on correctly framed data bytes

These modes are selected with MPMD[1:0] in the LIN-UART Control 1 Register. For all MULTIPROCESSOR Modes, bit MPEN of the LIN-UART Control 1 Register must be set to 1.

The first scheme is enabled by writing `01b` to MPMD[1:0]. In this mode, all incoming address bytes cause an interrupt, while data bytes never cause an interrupt. The interrupt service routine checks the address byte which triggered the interrupt. If it matches the LIN-UART address, the software clears MPMD[0]. At this point, each new incoming byte interrupts the CPU. The software determines the end of the frame and checks for it by reading the MPRX bit of the LIN-UART Status 1 Register for each incoming byte. If MPRX = 1, a new frame begins. If the address of this new frame is different from the LIN-UART's address, then MPMD[0] must be set to 1 by software, causing the LIN-UART

UART Control 0 Register must be initialized with TEN = 1, REN = 1 and all other bits = 0.

In addition to the LMST, LSLV and ABEN bits in the LIN Control Register, a LinState[1:0] field exists which defines the current state of the LIN logic. This field is initially set by software. In the LIN SLAVE Mode, the LinState field is updated by hardware as the slave moves through the Wait For Break, AutoBaud and Active states.

12.1.10.3. LIN MASTER Mode Operation

LIN MASTER Mode is selected by setting LMST = 1, LSLV = 0, ABEN = 0 and LinState[1:0] = 11B. If the LIN bus protocol indicates the bus is required go into the LIN SLEEP state, the LinState[1:0] bits must be set to 00B by software.

The break is the first part of the message frame transmitted by the master, consisting of at least 13 bit periods of logical zero on the LIN bus. During initialization of the LIN master, the duration (in bit times) of the break is written to the TxBreakLength field of the LIN Control Register. The transmission of the break is performed by setting the SBRK bit in the Control 0 Register. The LIN-UART starts the break after the SBRK bit is set and any character transmission currently underway has completed. The SBRK bit is deasserted by hardware until the break is completed.

If it is necessary to generate a break longer than 15 bit times, the SBRK bit can be used in normal UART mode where software times the duration of the break.

The Synch character is transmitted by writing a 55H to the Transmit Data Register (TDRE must = 1 before writing). The Synch character is not transmitted by the hardware until the break is complete.

The identifier character is transmitted by writing the appropriate value to the Transmit Data Register (TDRE must = 1 before writing).

If the master is sending the *response* portion of the message, these data and checksum characters are written to the Transmit Data Register when the TDRE bit asserts. If the Transmit Data Register is written after TDRE asserts, but before TXE asserts, the hardware inserts one or two stop bits between each character as determined by the stop bit in the Control 0 Register. Additional idle time occurs between characters, if TXE asserts before the next character is written.

If the selected slave is sending the response portion of the frame to the master, each receive byte will be signalled by the receive data interrupt (RDA bit will be set in the Status 0 Register). If the selected slave is sending the response to a different slave, the master can ignore the response characters by deasserting the REN bit in the Control 0 Register until the frame time slot is completed.

12.1.10.4. LIN SLEEP Mode

While the LIN bus is in the *sleep* state, the CPU can either be in low power STOP Mode, in HALT Mode, or in normal operational state. Any device on the LIN bus can issue a

12.3.3. LIN-UART Status 0 Register

The LIN-UART Status 0 Register identifies the current LIN-UART operating configuration and status. Table 85 describes the Status 0 Register for standard UART mode. Table 86 describes the Status 0 Register for LIN mode.

Table 85. LIN-UART Status 0 Register—Standard UART Mode (U0STAT0 = F41H)

Bit	7	6	5	4	3	2	1	0
Field	RDA	PE	OE	FE	BRKD	TDRE	TXE	CTS
Reset	0	0	0	0	0	1	1	X
R/W	R	R	R	R	R	R	R	R
Address	F41H, F49H							

Note: R = Read; X = undefined.

Bit	Description
[7] RDA	**Receive Data Available** This bit indicates that the LIN-UART Receive Data Register has received data. Reading the LIN-UART Receive Data Register clears this bit. 0 = The LIN-UART Receive Data Register is empty. 1 = There is a byte in the LIN-UART Receive Data Register.
[6] PE	**Parity Error** This bit indicates that a parity error has occurred. Reading the Receive Data Register clears this bit. 0 = No parity error occurred. 1 = A parity error occurred.
[5] OE	**Overrun Error** This bit indicates that an overrun error has occurred. An overrun occurs when new data is received and the Receive Data Register is not read. Reading the Receive Data Register clears this bit. 0 = No overrun error occurred. 1 = An overrun error occurred.
[4] FE	**Framing Error** This bit indicates that a framing error (no stop bit following data reception) was 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 indicates that a break occurred. If the data bits, parity/multiprocessor bit and stop bit(s) are all zeros then this bit is set to 1. Reading the Receive Data Register clears this bit. 0 = No break occurred. 1 = A break occurred.

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.

12.3.5. LIN-UART Control 0 Register

The LIN-UART Control 0 Register, shown in Table 89, configures the basic properties of LIN-UART's transmit and receive operations. A more detailed discussion of each bit follows the table.

Table 89. LIN-UART Control 0 Register (U0CTL0 = F42H)

Bit	7	6	5	4	3	2	1	0
Field	TEN	REN	CTSE	PEN	PSEL	SBRK	STOP	LBEN
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	F42H, F4AH							

Note: R/W = Read/Write.

Bit	Description
[7] TEN	**Transmit Enable** This bit enables or disables the transmitter. The enable is also controlled by the $\overline{CTS}$ signal and the CTSE bit. If the $\overline{CTS}$ signal is Low and the CTSE bit is 1, the transmitter is enabled. 0 = Transmitter disabled. 1 = Transmitter enabled.
[6] REN	**Receive Enable** This bit enables or disables the receiver. 0 = Receiver disabled. 1 = Receiver enabled.
[5] CTSE	**Clear To Send Enable** 0 = The $\overline{CTS}$ signal has no effect on the transmitter. 1 = The LIN-UART recognizes the $\overline{CTS}$ signal as an enable control for the transmitter.
[4] PEN	**Parity Enable** This bit enables or disables parity. Even or odd is determined by the PSEL bit. 0 = Parity is disabled. This bit is overridden by the MPEN bit. 1 = The transmitter sends data with an additional parity bit and the receiver receives an additional parity bit.
[3] PSEL	**Parity Select** 0 = Even parity is sent as an additional parity bit for the transmitter/receiver. 1 = Odd parity is sent as an additional parity bit for the transmitter/receiver.

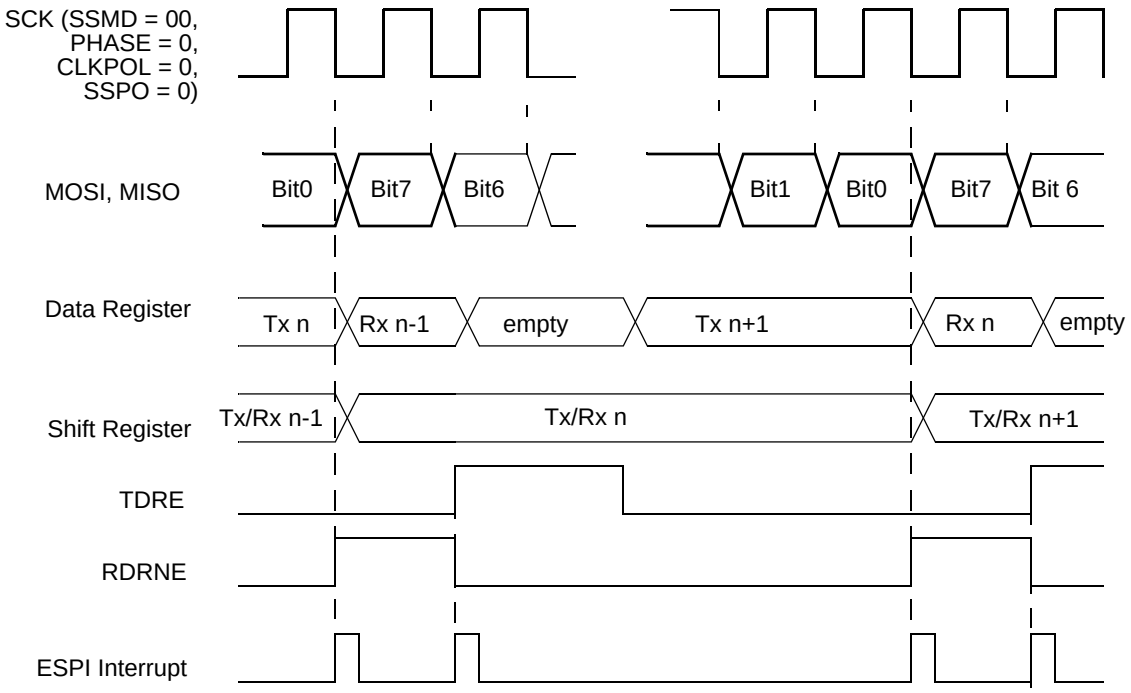

Figure 36. SPI Mode (SSMD = 00)

16.3.3.2. Synchronous Frame Sync Pulse Mode

This mode is selected by setting the SSMD field of the Mode Register to 10. This mode is typically used for continuous transfer of fixed length frames where the frames are delineated by a pulse of duration one SCK period. The SSV bit in the ESPI Transmit Data Command register does not control the $\overline{SS}$ pin directly in this mode. SSV must be set before or in sync with the first transmit data byte being written. The $\overline{SS}$ signal will assert 1 SCK cycle before the first data bit and will stop after 1 SCK period. SCK is active from the initial assertion of $\overline{SS}$ until the transaction end due to lack of transmit data.

The transaction is terminated by the Master when it no longer has data to send. If TDRE=1 at the end of a character, the $\overline{SS}$ output will remain detached and SCK stops after the last bit is transferred. The TUND bit (transmit underrun) will assert in this case. After the transaction has completed, hardware will clear the SSV bit. Figure 37 displays a frame with synchronous frame sync pulse mode.

16.4.5. ESPI Status Register

The ESPI Status Register, shown in Table 113, indicates the current state of the ESPI. All bits revert to their Reset state if the ESPI is disabled.

Table 113. ESPI Status Register (ESPISTAT)

Bits	7	6	5	4	3	2	1	0
Field	TDRE	TUND	COL	ABT	ROVR	RDRNE	TFST	SLAS
Reset	1	0	0	0	0	0	0	1
R/W	R	R/W*	R/W*	R/W*	R/W*	R	R	R
Address	F64H							

Note: R/W* = Read access. Write a 1 to clear the bit to 0.

Bit	Description
[7] TDRE	**Transmit Data Register Empty** 0 = Transmit Data Register is full or ESPI is disabled. 1 = Transmit Data Register is empty. A write to the ESPI (Transmit) Data Register clears this bit.
[6] TUND	**Transmit Underrun** 0 = A Transmit Underrun error has not occurred. 1 = A Transmit Underrun error has occurred.
[5] COL	**Collision** 0 = A multi-Master collision (mode fault) has not occurred. 1 = A multi-Master collision (mode fault) has occurred.
[4] ABT	**SLAVE Mode Transaction Abort** This bit is set if the ESPI 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 ESPIMODE register. This bit can also be set in SLAVE Mode by an SCK monitor time-out (MMEN = 0, BRGCTL = 1). 0 = A SLAVE Mode transaction abort has not occurred. 1 = A SLAVE Mode transaction abort has occurred.
[3] ROVR	**Receive Overrun** 0 = A Receive Overrun error has not occurred. 1 = A Receive Overrun error has occurred.
[2] RDRNE	**Receive Data Register Not Empty** 0 = Receive Data Register is empty. 1 = Receive Data Register is not empty.

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 `11110b`, followed by the two most-significant address bits and a 0 (write) to the I^2C Data Register.
3. The software asserts the start bit of the I^2C Control Register.
4. The I^2C controller sends a start condition.
5. The I^2C controller loads the I^2C Shift Register with the contents of the I^2C Data Register.
6. After the first bit has been shifted out, a transmit interrupt is asserted.
7. The software responds by writing the least significant eight bits of address to the I^2C Data Register.
8. The I^2C controller completes shifting of the first address byte.
9. The I^2C slave sends an Acknowledge by pulling the SDA signal Low during the next High period of SCL.

   If the slave does not acknowledge the address byte, the I^2C controller sets the NCKI bit in the I^2C Status Register, sets the ACKV bit and clears the ACK bit in the I^2C State Register. The software responds to the Not Acknowledge interrupt by setting the stop bit and clearing the TXI bit. The I^2C controller flushes the Transmit Data Register, sends the stop condition on the bus and clears the stop and NCKI bits. The transaction is complete and the following steps can be ignored.
10. The I^2C controller loads the I^2C Shift Register with the contents of the I^2C Data Register (the lower byte of the 10-bit address).
11. The I^2C controller shifts out the next eight bits of the address. After the first bit shifts, the I^2C controller generates a transmit interrupt.
12. The software responds by setting the start bit of the I^2C Control Register to generate a repeated start condition.
13. The software writes `11110b`, followed by the 2-bit slave address and a 1 (Read) to the I^2C Data Register.
14. If the user chooses to read only one byte, the software responds by setting the NAK bit of the I^2C Control Register.
15. After the I^2C controller shifts out the address bits listed in Step 9 (the second address transfer), the I^2C slave sends an Acknowledge by pulling the SDA signal Low during the next High period of SCL.

    If the slave does not acknowledge the address byte, the I^2C controller sets the NCKI bit in the I^2C Status Register, sets the ACKV bit and clears the ACK bit in the I^2C

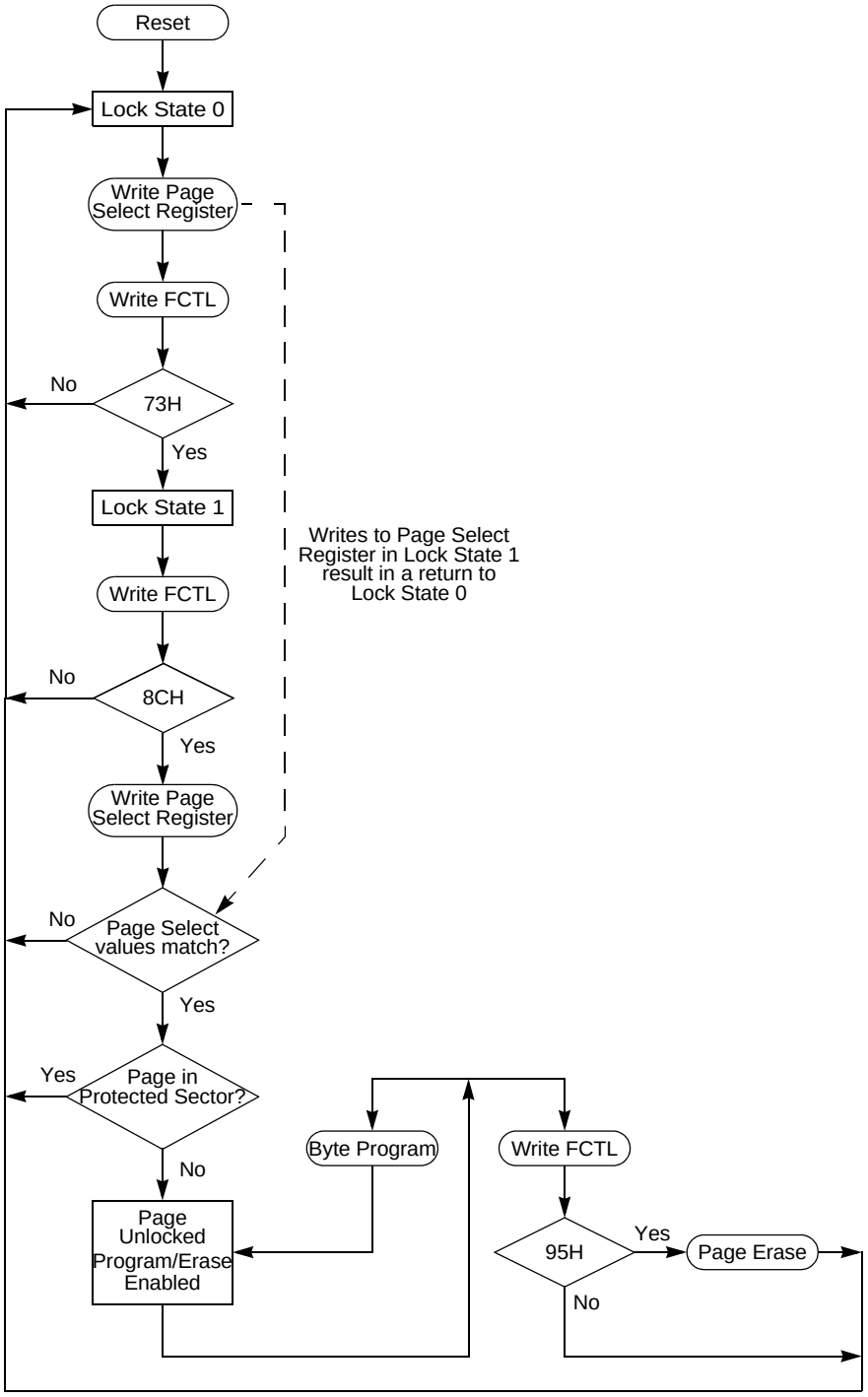

Figure 54. Flowchart: Flash Controller Operation

! **Caution:** The byte at each Flash memory address cannot be programmed (any bits written to 0) more than twice before an erase cycle occurs.

20.2.5. Page Erase

The Flash memory can be erased one page (512 bytes) at a time. Page-erasing Flash memory sets all bytes in that page to the value FFH. The Flash Page Select register identifies the page to be erased. Only a page residing in an unprotected sector can be erased. With the Flash Controller unlocked, writing the value 95h to the Flash Control Register initiates the Page Erase operation on the active page. While the Flash Controller executes the Page Erase operation, the eZ8 CPU idles but the system clock and on-chip peripherals continue to operate. The eZ8 CPU resumes operation after the Page Erase operation completes. If the Page Erase operation is performed using the OCD, poll the Flash Status register to determine when the Page Erase operation is complete. When the Page Erase is complete, the Flash Controller returns to its locked state.

20.2.6. Mass Erase

Flash memory can also be mass-erased using the Flash Controller, but only by using the On-Chip Debugger. Mass-erasing Flash memory sets all bytes to the value FFH. With the Flash Controller unlocked, writing the value 63H to the Flash Control Register initiates the Mass Erase operation. While the Flash Controller executes the Mass Erase operation, the eZ8 CPU idles but the system clock and on-chip peripherals continue to operate. Using the On-Chip Debugger, poll the Flash Status register to determine when the Mass Erase operation is complete. When the Mass Erase is complete, the Flash Controller returns to its locked state.

20.2.7. Flash Controller Bypass

The Flash Controller can be bypassed and the control signals for Flash memory are brought out to the GPIO pins. Bypassing the Flash Controller allows faster row programming algorithms by controlling these Flash programming signals directly.

Row programming is recommended for gang programming applications and large-volume customers who do not require in-circuit initial programming of Flash memory. Mass Erase and Page Erase operations are also supported when the Flash Controller is bypassed.

For more information about bypassing the Flash Controller, please contact Zilog Technical Support.

21.2.4.7. Trim Bit Address 0007H

In the Trim Option Bits Register at address 0007H and shown in Table 155, all bits are reserved.

Table 155. Trim Option Bits at 0007H (TFilter0)

Bits	7	6	5	4	3	2	1	0
Field	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
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 0027H							

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

21.2.4.8. Trim Bit Address 0008H

In the Trim Option Bits Register at address 0008H and shown in Table 156, all bits are reserved.

Table 156. Trim Option Bits at 0008H (TFilter1)

Bits	7	6	5	4	3	2	1	0
Field	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
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 0028H							

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

Modes section on page 42) and configured for a threshold voltage of 2.4V or greater (see the Trim Bit Address Space section on page 282).

A System Reset that occurs during a write operation (such as a pin reset or watchdog timer reset) also perturbs the byte currently being written. All other bytes in the array remain unperturbed.

22.2.4. Optimizing NVDS Memory Usage for Execution Speed

As listed in Table 161, the NVDS read time varies drastically, this discrepancy being a trade-off for minimizing the frequency of writes that require post-write page erases. The NVDS read time of address N is a function of the number of writes to addresses other than N since the most recent write to address N, as well as the number of writes since the most recent page erase. Neglecting effects caused by page erases and results caused by the initial condition in which the NVDS is blank, a rule of thumb is that every write since the most recent page erase causes read times of unwritten addresses to increase by 0.8µs up to a maximum of 258µs.

Table 161. NVDS Read Time

Operation	Minimum Latency (µs)	Maximum Latency (µs)
Read	71	258
Write	126	136
Illegal Read	6	6
Illegal Write	7	7

If NVDS read performance is critical to your software architecture, you can optimize your code for speed by using either of the methods listed below.

- Periodically refresh all addresses that are used; the most useful method. The optimal use of NVDS, in terms of speed, is to rotate the writes evenly among all planned addresses, bringing all reads closer to the minimum read time. Because the minimum read time is much less than the write time, however, actual speed benefits are not always realized.
- Use as few unique addresses as possible to optimize the impact of refreshing.

23.4.2. OCD Status Register

The OCD Status Register, shown in Table 165, reports status information about the current state of the debugger and the system.

Table 165. OCD Status Register (OCDSTAT)

Bit	7	6	5	4	3	2	1	0
Field	IDLE	HALT	RPEN	Reserved				
Reset	0	0	0	0				
R/W	R	R	R	R				

Bit	Description
[7] IDLE	**CPU Idle** This bit is set if the part is in Debug mode (DBGMODE is 1) or if a BRK instruction has occurred since the last time OCDCTL was written. This can be used to determine if the CPU is running or if it is idle. 0 = The eZ8 CPU is running. 1 = The eZ8 CPU is either stopped or looping on a BRK instruction.
[6] HALT	**HALT Mode** 0 = The device is not in HALT Mode. 1 = The device is in HALT Mode.
[5] RPEN	**Read Protect Option Bit Enable** 0 = The Read Protect option bit is disabled (Flash option bit is 1). 1 = The Read Protect option bit is enabled (Flash option bit is 0), disabling many OCD commands.
[4:0]	Reserved; must be 0.

Table 181. CPU Control Instructions (Continued)

Mnemonic	Operands	Instruction
STOP	—	STOP Mode
WDT	—	Watchdog Timer Refresh

Table 182. 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

Table 183. Logical Instructions

Mnemonic	Operands	Instruction
AND	dst, src	Logical AND
ANDX	dst, src	Logical AND using Extended Addressing
COM	dst	Complement
OR	dst, src	Logical OR
ORX	dst, src	Logical OR using Extended Addressing
XOR	dst, src	Logical Exclusive OR
XORX	dst, src	Logical Exclusive OR using Extended Addressing