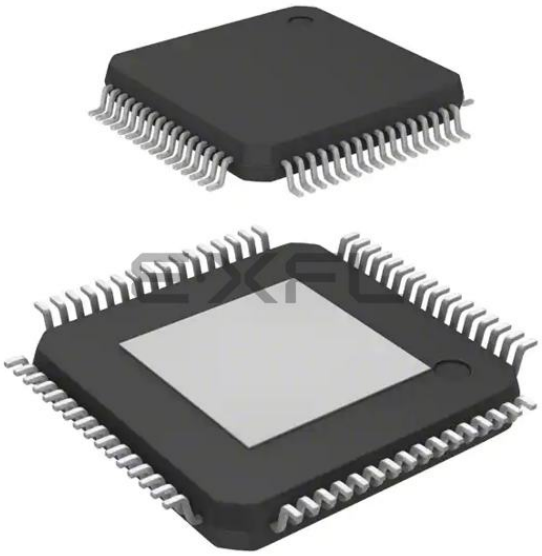


Welcome to E-XFL.COM

Embedded - Microcontrollers - Application Specific: Tailored Solutions for Precision and Performance

Embedded - Microcontrollers - Application Specific represents a category of microcontrollers designed with unique features and capabilities tailored to specific application needs. Unlike general-purpose microcontrollers, application-specific microcontrollers are optimized for particular tasks, offering enhanced performance, efficiency, and functionality to meet the demands of specialized applications.

What Are Embedded - Microcontrollers - Application Specific?

Application-specific microcontrollers are engineered to

Details	
Product Status	Active
Applications	Motor Control
Core Processor	Zneo™
Program Memory Type	FLASH (32kB)
Controller Series	Z16FMC
RAM Size	4K x 8
Interface	I²C, IrDA, LIN, SPI, UART/USART
Number of I/O	46
Voltage - Supply	2.7V ~ 3.6V
Operating Temperature	-40°C ~ 105°C
Mounting Type	Surface Mount
Package / Case	64-LQFP Exposed Pad
Supplier Device Package	64-LQFP
Purchase URL	https://www.e-xfl.com/product-detail/zilog/z16fmc32ag20eg

Master Interrupt Enable

The master interrupt enable bit in the flag register globally enables or disables interrupts. This bit has been moved to the flag register (bit 0). Thus, anytime the register is loaded, it changes the state of the IRQE bit. For the IRET instruction the bit is set based on what has been pushed on the stack.

Interrupts are globally enabled by any of the following actions:

- Execution of an Enable Interrupt (EI) instruction
- Writing 1 to the IRQE bit in the flag register

Interrupts are globally disabled by any of the following actions:

- Execution of a Disable Interrupt (DI) instruction
- CPU acknowledgement of an interrupt service request from the interrupt controller
- Writing 0 to the IRQE bit in the flag register
- Reset
- Execution of a TRAP instruction
- All System Exceptions

Interrupt Vectors and Priority

The interrupt controller supports three levels of interrupt priority. Level 3 is the highest priority, Level 2 is the second highest priority, and Level 1 is the lowest priority. If all the interrupts are enabled with identical interrupt priority (for example, all interrupts enabled as Level 2 interrupts), the interrupt priority is assigned from highest to lowest as specified in Table 27 on page 49. Level 3 interrupts always have higher priority than Level 2 interrupts, which in turn, always have higher priority than Level 1 interrupts. Within each interrupt priority levels (Level 1, Level 2, or Level 3), priority is assigned as specified in Table 27. Reset and System Exceptions have the highest priority.

System Exceptions

System Exceptions are generated for stack overflow, illegal instructions, divide-by-zero, and divide overflow, etc. The System Exceptions are not affected by the IRQE and share a single vector.

Each exception has a bit in the system exception status register. When a system exception occurs it pushes the program counter and the flags on the stack, fetches the system exception vector from 000008H (similar to a IRQ) and the bit associated with that exception is set in the status register. Additional exceptions from the same source are blocked until the status bit of the particular exception is cleared by writing 1 to that status bit. Other types of

$$\text{One-Shot Mode Time-Out Period(s)} = \frac{(\text{Reload Value} - \text{Start Value} + 1) \times \text{Prescale}}{\text{System Clock Frequency (Hz)}}$$

TRIGGERED ONE-SHOT Mode

In TRIGGERED ONE-SHOT mode, the timer operates as follows:

1. The timer is non-active until a trigger is received. The timer trigger is taken from the timer input pin. The TPOL bit in the Timer Control 1 Register selects whether the trigger occurs on the rising edge or the falling edge of the timer input signal.
2. Following the trigger event, the timer counts system clocks up to the 16-bit Reload value stored in the timer reload high and low byte registers.
3. After reaching the Reload value, the timer outputs a pulse on the timer output pin, generates an interrupt and resets the count value in the timer high and low byte registers to `0001H`. The duration of the output pulse is a single system clock. The TPOL bit also sets the polarity of the output pulse.
4. The timer now idles until the next trigger event. Trigger events, which occur while the timer is responding to a previous trigger is ignored.

Observe the following steps to configure timer 0 in TRIGGERED ONE-SHOT mode and initiate operation:

1. Write to the timer control registers to:
   - Disable the timer
   - Configure the timer for TRIGGERED ONE-SHOT mode
   - Set the prescale value
   - Set the initial output level (High or Low) via the TPOL bit for the timer output alternate function
   - Set the INTERRUPT mode
2. Write to the timer high and low byte registers to set the starting count value.
3. Write to the timer reload high and low byte registers to set the reload value.
4. Enable the timer interrupt, if required and set the timer interrupt priority by writing to the relevant interrupt registers.
5. When using the timer output function, configure the associated GPIO port pin for the timer output alternate function.
6. Write to the Timer Control 1 Register to enable the timer. Counting does not start until the appropriate input transition occurs.

The timer period is calculated by the following equation (Start Value = 1):

Table 83. LIN-UART Baud Rates (Continued)

5.5296 MHz System Clock			
Desired Rate (kHz)	**BRG Divisor (Decimal)**	**Actual Rate (kHz)**	**Error (%)**
1250.0	N/A	N/A	N/A
625.0	N/A	N/A	N/A
250.0	1	345.6	38.24
115.2	3	115.2	0.00
57.6	6	57.6	0.00
38.4	9	38.4	0.00
19.2	18	19.2	0.00
9.60	36	9.60	0.00
4.80	72	4.80	0.00
2.40	144	2.40	0.00
1.20	288	1.20	0.00
0.60	576	0.60	0.00
0.30	1152	0.30	0.00

1.8432 MHz System Clock			
Desired Rate (kHz)	**BRG Divisor (Decimal)**	**Actual Rate (kHz)**	**Error (%)**
1250.0	N/A	N/A	N/A
625.0	N/A	N/A	N/A
250.0	N/A	N/A	N/A
115.2	1	115.2	0.00
57.6	2	57.6	0.00
38.4	3	38.4	0.00
19.2	6	19.2	0.00
9.60	12	9.60	0.00
4.80	24	4.80	0.00
2.40	48	2.40	0.00
1.20	96	1.20	0.00
0.60	192	0.60	0.00
0.30	384	0.30	0.00

3.579545 MHz System Clock			
Desired Rate (kHz)	**BRG Divisor (Decimal)**	**Actual Rate (kHz)**	**Error (%)**
1250.0	N/A	N/A	N/A
625.0	N/A	N/A	N/A
250.0	1	223.72	–10.51
115.2	2	111.9	–2.90
57.6	4	55.9	–2.90
38.4	6	37.3	–2.90
19.2	12	18.6	–2.90
9.60	23	9.73	1.32
4.80	47	4.76	–0.83
2.40	93	2.41	0.23
1.20	186	1.20	0.23
0.60	373	0.60	–0.04
0.30	746	0.30	–0.04

Receiving IrDA Data

Data received from the infrared transceiver via the **IR_RXD** signal through the RXD pin is decoded by the infrared endec and passed to the UART. The UART's baud rate clock is used by the infrared endec to generate the demodulated signal (**RXD**) that drives the UART. Each UART/Infrared data bit is 16-clocks wide. Figure 24 displays data reception. When the infrared endec is enabled, the UART's RXD signal is internal to the Z16FMC products when the IR_RXD signal is received through the RXD pin.

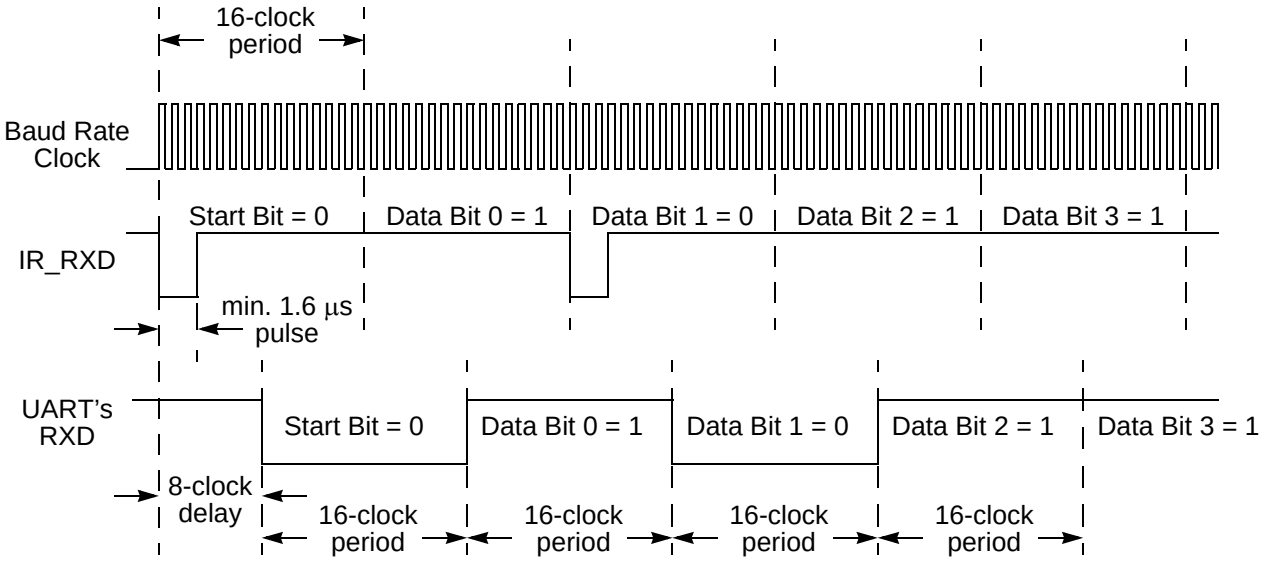


Figure 24. Infrared Data Reception

Caution: The system clock frequency must be at least 1.0 MHz to ensure proper reception of the 1.6µs minimum width pulses allowed by the IrDA standard.

Endec Receiver Synchronization

The IrDA receiver uses a local baud rate clock counter (0 to 15 clock periods) to generate an input stream for the UART and to create a sampling window for detection of incoming pulses. The generated UART input (UART RXD) is delayed by 8 baud rate clock periods with respect to the incoming IrDA data stream. When a falling edge in the input data stream is detected, the Endec counter is reset. When the count reaches a value of 8, the UART RXD value is updated to reflect the value of the decoded data.

is to set the TEOF bit of the transmit data command register when the final TDRE interrupt or DMA request is being serviced (set TEOF before or simultaneously with writing the final data byte). When the final bit of the final character is transmitted, the hardware will automatically deassert the SSV and TEOF bits. The second method is for software to directly clear the SSV bit after the transaction completes. If software clears the SSV bit directly, it is not necessary for software to also set the TEOF bit on the final transmit byte. After writing the final transmit byte, the end of the transaction is detected by waiting for the final RDRF interrupt or monitoring the TFST bit in the ESPI Status Register.

The transmit underrun and receive overrun errors do not occur in an SPI mode master. If the RDRF and TDRE requests have not been serviced before the current byte transfer completes, SCLK is paused until the data register is read and written. The transmit underrun and receive overrun errors will occur in a slave if the slave's software/DMA does not keep up with the master data rate. If a transmit underrun occurs in SLAVE mode, the shift register in the slave is loaded with all 1s.

In the SPI mode, the SCK is active only for the data transfer with one SCK period per bit transferred. If the SPI bus has multiple slaves, the slave select lines to all or one of the slaves must be controlled independently by software using GPIO pins.

Figure 28 displays multiple character transfer in SPI mode. Note that while character 'n' is being transferred using the shift register, software/DMA responds to the receive request for character n-1 and the transmit request for character n+1.

Multi-Master SPI Operation

In a multi-master SPI system, all SCK pins are tied together, all MOSI pins are tied together and all MISO pins are tied together. All SPI pins must be configured in open-drain mode to prevent bus contention. At any time, only one SPI device is configured as the master and all other devices on the bus are configured as slaves. The master asserts the $\overline{SS}$ pin on the selected slave. Then, the active master drives the clock and transmit data on the SCK and MOSI pins to the SCK and MOSI pins on the slave (including those slaves which are not enabled). The enabled slave drives data out its MISO pin to the MISO master pin.

When the ESPI is configured as a master in a multi-master SPI system, the $\overline{SS}$ pin must be configured as an input. The $\overline{SS}$ input signal on a device configured as a master must remain High. If the $\overline{SS}$ signal on the active master goes Low (indicating another master is accessing this device as a slave), a collision error flag is set in the ESPI status register. The slave select outputs on a master in a multi-master system must come from GPIO pins.

SPI Slave Operation

The ESPI block is configured for SLAVE mode operation by setting the MMEN bit = 0 in the ESPICTL Register and setting the `SSIO` bit = `0` in the ESPIMODE Register. The SSMD field of the ESPI mode register is set to 00 for SPI protocol mode. The `Phase`, `CLKPOL` and `WOR` bits in the ESPICTL Register and the NUMBITS field in the ESPIMODE Register must be set to be consistent with the other SPI devices. Typically for an SPI slave SSPO = 0.

If the slave has data to send to the master, the data must be written to the data register before the transaction starts (first edge of SCK when $\overline{SS}$ is asserted). If the data register is not written prior to the slave transaction, the MISO pin outputs all 1s.

Due to the delay resulting from synchronization of the $\overline{SS}$ and SCK input signals to the internal system clock, the maximum SCK baud rate which is supported in SLAVE mode is the system clock frequency divided by 8. This rate is controlled by the SPI master.

Figure 32 illustrates the ESPI configuration in SPI SLAVE mode.

ESPI Mode Register

The ESPI Mode Register (see Table 92) configures the character bit width and mode of the ESPI IO pins.

Table 92. ESPI Mode Register (ESPIMODE)

Bits	7	6	5	4	3	2	1	0
Field	SSMD			NUMBITS[2:0]			SSIO	SSPO
RESET	000			000			0	0
R/W	R/W			R/W			R/W	R/W
ADDR	FF_E263H							

Bits	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 the Slave Select section on page 152. **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 or DMA must 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 with which the transmit data register is written with the initial byte. At the end of a frame (after the final RDRF event), SSV is deasserted by software. Alternatively, SSV is automatically deasserted by hardware if the TEOF bit in the Transmit Data Command register is set when the final transmit byte is loaded. In SPI 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 a remote loop back (echo) function. **010 = I^2S Mode** In this mode, the value from SSV will be output by the Master on the $\overline{SS}$ pin 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/DMA continue to provide data. The I^2S protocol (Inter IC Sound) 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) will trigger the start of a transaction on the next SCK cycle.

I²C Master/Slave Controller Registers

Table 98 summarizes the I²C Master/Slave Controller software-accessible registers.

Table 98. I²C Master/Slave Controller Registers

Name	Abbreviation	Description
I²C Data	I2CDATA	Transmit/Receive Data Register
I²C Interrupt Status	I2CISTAT	Interrupt Status Register
I²C Control	I2CCTL	Control Register – basic control functions
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. Configures address recognition, Defines Slave Address bits [9:8].
I²C Slave Address	I2CSLVAD	Defines Slave Address bits [7:0]

Comparison with Master Mode only I²C Controller

Porting code written for the Master-only I²C Controller found on other Z8 Encore!® parts to the I²C Master/Slave Controller is straightforward. The I2CDATA, I2CCTL, I2CBRH and I2CBRL register definitions are not changed. The difference between the Master-Only I²C Controller and the I²C Master/Slave Controller designs is explained below.

- The Status register (I2CSTATE) from the Master-only I²C Controller is split into the Interrupt Status (I2CISTAT) register and the State (I2CSTATE) register because there are more interrupt sources. The ACK, 10B, TAS (now called AS) and DSS (now called DS) bits formerly in the Status Register are moved to the State Register.
- The I2CSTATE register is called as Diagnostic State (I2CDST) register in the Master Only mode version. The I2CDST register provides diagnostic information. The I2CSTATE register contains status and state information that are useful to software in operational mode.
- The I2CMODE register is called as Diagnostic Control (I2CDIAG) register in the MASTER ONLY mode version. The I2CMODE register provides control for SLAVE modes of operation as well as the most significant two bits of the 10-bit slave address.
- The I2CSLVAD register is added for programming the slave address.
- The ACKV bit in the I2CSTATE Register enables the Master to verify the Acknowledge from the Slave before sending the next byte.
- Support for multi-master environments. If arbitration is lost when operating as a Master, the ARBLST bit in the I2CISTAT Register is set and the mode automatically switches to Slave mode.

watermark DMA interrupt is used to notify software when the N–1st byte has been received.

- Configure the selected DMA channel for I^2C receive. The IEOB bit must be set in the DMACTL register for the final buffer to be transferred. Typically one buffer will be defined with a transfer length of N where N bytes are expected to be received from the master. The watermark is set to 1 by writing a 0x01 to DMAxLAR[23:16].
- The I^2C interrupt must be enabled in the interrupt controller to alert software of any I^2C error conditions.
- The I^2C Master/Slave must be configured as defined in the sections above describing Slave mode transactions. The TXI bit in the I2CCTL register must be cleared.
- When the SAM interrupt occurs, set the DMAIF bit in the I2CMODE register.
- The DMA transfers the data to memory as it is received from the master.
- When the first DMA interrupt occurs indicating that the (N–1)st byte is received, the NAK bit must be set in the I2CCTL register.
- When the second DMA interrupt occurs, it indicates that the Nth byte is received. A Stop I^2C interrupt occurs (SPRS bit set in the I2CSTAT register) when the master issues the STOP (or RESTART) condition.
- Clear the DMAIF bit in the I2CMODE register.

Slave Read Transaction with Data DMA

In this transaction the I^2C Master/Slave operates as a slave, sending data to the master.

- Configure the selected DMA channel for I^2C transmit. The IEOB bit must be set in the DMACTL register for the final buffer to be transferred. Typically a single buffer with a transfer length of N is defined.
- The I^2C interrupt must be enabled in the interrupt controller to alert software of any I^2C error conditions. A Not Acknowledge interrupt occurs on the final byte transferred.
- The I^2C Master/Slave must be configured as defined in the sections above describing Slave mode transactions. The TXI bit in the I2CCTL register must be cleared.
- When the SAM interrupt occurs, set the DMAIF bit in the I2CMODE register.
- The DMA transfers the data to be transmitted to the master.
- When the DMA interrupt occurs, the final byte is being transferred to the master. The master must send a Not Acknowledge for this final byte, setting the NCKI bit in the I2CSTAT register and generating the I^2C interrupt. A Stop or Restart interrupt (SPRS bit set in I2CSTAT register) follows.
- Clear the DMAIF bit in the I2CMODE register.

Table 106. I2CSTATE_H

State Encoding	State Name	State Description
0000	Idle	I^2C bus is idle or I^2C Controller is disabled.
0001	Slave Start	I^2C Controller has received a start condition.
0010	Slave Bystander	Address did not match – ignore remainder of transaction.
0011	Slave Wait	Waiting for STOP or RESTART condition after sending a Not Acknowledge instruction.
0100	Master Stop2	Master completing STOP condition (SCL = 1, SDA = 1).
0101	Master Start/Restart	Master mode sending START condition (SCL = 1, SDA = 0).
0110	Master Stop1	Master initiating STOP condition (SCL = 1, SDA = 0).
0111	Master Wait	Master received a Not Acknowledge instruction, waiting for software to assert STOP or START control bits.
1000	Slave Transmit Data	Nine substates, one for each data bit and one for the acknowledge.
1001	Slave Receive Data	Nine substates, one for each data bit and one for the acknowledge.
1010	Slave Receive Addr1	Slave Receiving first address byte (7 and 10 bit addressing) Nine substates, one for each address bit and one for the acknowledge.
1011	Slave Receive Addr2	Slave Receiving second address byte (10 bit addressing) Nine substates, one for each address bit and one for the acknowledge.
1100	Master Transmit Data	Nine substates, one for each data bit and one for the acknowledge.
1101	Master Receive Data	Nine substates, one for each data bit and one for the acknowledge.
1110	Master Transmit Addr1	Master sending first address byte (7- and 10-bit addressing) Nine substates, one for each address bit and one for the acknowledge.
1111	Master Transmit Addr2	Master sending second address byte (10-bit addressing) Nine substates, one for each address bit and one for the acknowledge.

DMA Description

The DMA is used to off load the processor from doing repetitive tasks. DMA transfers data from one memory address to another memory address. Because all peripherals are mapped in memory, the DMA transfers data to or from peripherals.

The DMA transfers data from the source address to the destination address. This requires a read and/or write cycle that is generated by the DMA controller. Each DMA transfer requires a minimum of two system clock cycles to execute.

The DMA operates in direct or linked list mode. Direct mode and Linked List mode are almost the same. In Direct mode the software loads the DMA channel registers directly. In linked list mode the DMA loads its registers from memory.

DMA Register Description

Each DMA channel consists of 16-bit control register, a 16-bit transfer length register, a 24-bit destination address register, a 24-bit source address register and a 24-bit list address register (see Figure 46).

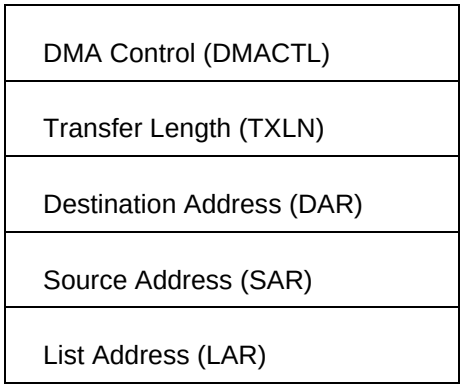


Figure 46. DMA Channel Registers

Buffers

A buffer is an allocation of contiguous memory bytes. Buffers are allocated by software to be used by the DMA. The DMA transfers data to or from buffers. A typical application would be to send data to serial channels such as I^2C, UART and SPI. The data to be sent is placed in a buffer by software.

Frames

A frame is a single buffer or a collection of buffers. Frame boundaries spans multiple buffers.

signals are Acknowledge (ACK), Command Valid (CMDVLD), End Of Frame (EOF-SYNC) and Read Status (RDSTAT). The two 4-bit busses are Command Bus (CMDBUS) and Stat Bus (STATBUS).

A DMA transfer is initiated with the Request (REQ). When the DMA is servicing a Request from a peripheral it will assert its acknowledge signal (ACK) to let the peripheral know that a transfer is in progress. When the first byte of the transfer is written the CMD-VLD is asserted and the command bits are placed on the CMDBUS. The peripheral needs to latch the command from the bus when it sees this combination of signals.

If the EOF bit is set on the current buffer, and when the TXLN decrements to zero, the EOFSYNC signal is asserted on the final data transfer to the peripheral to signal that it is the final byte in the frame.

After receiving the EOFSYNC signal the peripheral need to assert the Request EOF signal to the DMA to let the DMA know that the descriptor is closed. This could be immediately or at some later time if the data transferred still needs to be processed. For peripherals, which do not support a Request EOF, the EOFSYNC is tied to Request EOF to terminate the transfer.

After the Request EOF is asserted the DMA closes the descriptor. The DMA asserts the ACK and RDSTAT signal, if the descriptor EOF bit is set. The peripheral, if it has status, places it on the STATBUS. This status is then placed in the descriptor and DMA status bits when it is closed.

If a peripheral needs to close a descriptor because of an error or the end of a packet is reached then it asserts it is Request EOF. If the transfer length is not zero, then the DMA will set the EOF bit, close the descriptor and generate an interrupt.

Buffer Closure

A DMA buffer closure is requested in two ways. The first is when the transfer length reaches zero. The second is when the DMA receives a request End Of Frame from the peripheral. When either of these cases occur, the DMA begins closure of the buffer.

Loop Mode Closure

If the LOOP bit is set then the current buffer descriptor is not modified. The DMAxLAR increments or a new LAR value is fetched from the descriptor.

EOF Closure

The DMAxEN bit is reset to 0. If the EOF bit is set, the CMDSTAT field is set with the status data from the peripheral. If the channel is in linked list mode then the DMAxCTL word is written back to the CONTROL word of the descriptor. The DMAxLAR increments or is loaded with new LAR data from the descriptor if the TXFR bit is set.

Normal Closure

The DMAxEN bit is reset to 0. If the channel is in linked list mode then the DMAxCTL word is written back to the CONTROL word of the descriptor. The DMAxLAR increments or is loaded with new LAR data from the descriptor if the TXFR bit is set.

DMA Modes

Each DMA channel operates in two modes, direct and linked list. Both modes use the DMA channel registers. The only difference is in how they are loaded. In direct mode, the DMA channel registers are directly loaded by software. When the transfer is complete, the DMA stops. In linked list mode the DMA will load its own registers from a descriptor list which is pointed to by the DMAxLAR register. It then loads the next descriptor in the list and continue executing.

The descriptor Control/Status field and address bytes have the same format as the control and address registers in the DMA.

Direct Mode

Direct mode only uses the registers in the DMA for operation. The software writes these register directly to set up and enable the DMA. Direct mode is entered by directly setting the DMAxEN bit in the DMAxCTL0 register. Figure 47 displays the DMA registers and how they point to the buffers allocated in memory.

5. Fetch the TXLN length from the descriptor and place it in the DMAxTXLN register in the DMA channel.
6. After the reads have been completed, the DMA starts looking for requests and transfer data until the transfer length reaches zero or the DMA receives a Request EOF signal.
7. When the DMA receives the Request EOF signal, it performs the following operations based upon the LOOP and EOF bit:
   - 00: The DMA writes the descriptor Control/Status word with the DMAxEN bit reset to 0.
   - 01: The DMA requests status from the peripheral. It then writes the descriptor Control/Status word with the DMAxEN bit reset to 0 and the status returned from the peripheral. The DMA then writes the TXLN length to the descriptor.
   - 1X: The DMA does not modify the descriptor.
8. If the HALT bit is set the DMA closes the current buffer but does not fetch the next descriptor.
9. After a new DMAxLAR address has been updated, the DMA goes back to step 2 above and fetches the control/status byte.

DMA Priority

The DMA priority is based upon the final channel serviced. After a channel is serviced it becomes the lowest-priority channel. Table 121 lists the DMA priority.

Table 121. DMA Priority

Last Channel Serviced	DMA Priority
DMA0	DMA1 (Highest) DMA2 DMA3 DMA0 (Lowest)
DMA1	DMA2 (Highest) DMA3 DMA0 DMA1 (Lowest)
DMA2	DMA3 (Highest) DMA0 DMA1 DMA2 (Lowest)
DMA3	DMA 0 (Highest) DMA 1 DMA 2 DMA 3 (Lowest)

Bits	7	6	5	4	3	2	1	0
Field	CHANSTATE				REQSEL			
RESET	0	0	0	0	0	0	0	0
R/W	R	R	R	R	R/W	R/W	R/W	R/W
ADDR	FFE400H, FFE401H, FFE402H, FFE403H							

Bits	Description
[7:4] CHANSTATE	**Channel State** 0000 = DMA Off 0001 = Direct Mode, Waiting for End Of Frame signal 0010 = Linked List Mode, Waiting for End Of Frame signal 0011 = Reserved 0100 = Direct Mode, First byte transfer, send command 0101 = Linked List Mode, First byte transfer, send command 0110 = Direct Mode, Transfer of buffer in progress 0111 = Linked List Mode, Transfer of buffer in progress 1000 = Direct Mode, Close Descriptor 1001 = Linked List Mode, New List 1010 = Linked List Mode, Close Descriptor 1011–1111 = Reserved

DMA Control Register

The DMA Control Register enables and controls DMA transfers (see Table 124).

Bits	15	14	13	12	11	10	9	8
Field	DMAxEN	LOOP	TXSIZE		DSTCTL		SRCCTL	
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
ADDR	FFE410H, FFE420H, FFE430H, FFE440H							

Bits	7	6	5	4	3	2	1	0
Field	IEOB	TXFR	EOF	HALT	CMDSTAT			
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
ADDR	FFE411H, FFE421H, FFE431H, FFE441H							

Bits	Description
15 DMAxEN	**DMA X Enable** If this bit is written directly then normal mode is executed. If this bit is read in from a descriptor then linked list mode is executed. 0 = DMA is disabled. 1 = DMA is enabled.
14 LOOP	**LOOP Mode** 0 = Descriptor is modified when the buffer is closed. 1 = Descriptor is not modified when buffer is closed.
[13:12] TXSIZE	**Transfer Size** 00 = Byte 01 = Word 10 = Quad 11 = Reserved
[11:10] DSTCTL	**Destination Control Register** 00 = Destination address does not change 01 = Destination address increments 10 = Destination address decrements 11 = Reserved
[9:8] SRCCTL	**Source Control Register** 00 = Source address does not change 01 = Source address increments 10 = Source address decrements 11 = Reserved

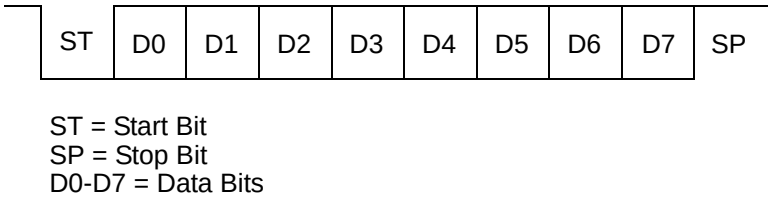


Figure 53. OCD Serial Data Format

Each bit time is of same length. The bit period is set by the baud rate generator.

When the transmitter sends a character, it first sends a Low start bit. The transmitter then waits one bit time. After the start bit is sent, the transmitter sends the next data bit. The transmitter sends each data bit in turn, waiting one full bit time before sending the next data bit. After the final data bit is sent, the transmitter sends a high stop bit for one bit time.

The receiver looks for the falling edge of the start bit. After the receiver sees the start bit is Low, it waits one half bit time and samples the middle of the start bit. If the middle of the start bit is High, the receiver considers this as a false start bit. The receiver ignores a false start bit and searches for another falling edge. If the middle of the start bit is Low, the receiver considers the start bit valid. The receiver will wait a full bit time from the middle of the start bit to sample the next data bit. The next data bit is sampled in the middle of the bit period. The receiver repeats this operation for each data bit, waiting one full bit time to between sampling each data bit.

After the receiver has sampled the final data bit, it waits one full bit time and sample the middle of the stop bit. If the stop bit is Low, the receiver detects a framing error.

If the stop bit is High, the data was correctly framed between a start and stop bit. After the receiver samples the middle of the stop bit, it begins searching for another start bit. The receiver does not wait for the full stop bit to be received before searching for the next start bit, in effect correcting for any bit skew due to error between the transmit and receive baud rate clocks.

Baud Rate Generator

The baud rate generator (BRG) is used to generate a bit clock for transmit and receive operations. The BRG reload register is automatically configured by the auto-baud detector, or it is written by software.

The value in the BRG reload register is calculated as:

Status Register

The Status Register (DBGSTAT), shown in Table 144, contains information about the state of the UART.

Table 144. Status Register (DBGSTAT)

Bits	7	6	5	4	3	2	1	0
Field	RDRF	RXOV	RXFE	RXBRK	TDRE	TXCOL	RXBUSY	TXBUSY
RESET	0	0	0	0	1	0	0	0
R/W	R/W1C	R/W1C	R/W1C	R/W1C	R/W1S	R/W1C	R	R
ADDR	FF_E085							

Bits	Description
7 RDRF	**Receive Data Register Full** This bit reflects the status of the Receive Data Register. When data is written to the Receive Data Register, or data is transferred from the shift register to the Receive Data Register, this bit is set to 1. When the Receive Data Register is read, this bit is cleared to zero. This bit is also cleared to zero by writing a one to this bit. 0 = Receive Data Register is empty. 1 = Receive Data Register is full.
6 RXOV	**Receive Overrun** This bit is set when a Receive Overrun occurs. A Receive Overrun occurs when there is data in the Receive Data Register and another byte is written to this register. 0 = Receive Overrun has not occurred 1 = Receive Overrun has occurred.
5 RXFE	**Receive Framing Error** This bit is set when a Receive Framing error has been detected. This bit is cleared by writing a one to this bit. 0 = No Framing Error detected. 1 = Receive Framing Error detected.
4 RXBRK	**Receive Break Detect** This bit is set when a Break condition has been detected. This occurs when 10 or more bits received are Low. This bit is cleared by writing a one to this bit. 0 = No Break detected. 1 = Break detected.
3 TDRE	**Transmit Data Register Empty** This bit reflects the status of the Transmit Data Register. When the Transmit Data Register is written, this bit is cleared to zero. When data from the transmit data register is read or transferred to the transmit shift register, this bit is set to 1. This bit is written to one to abort the transmission of data being held in the transmit data register. 0 = Transmit Data Register is full. 1 = Transmit Data Register is empty.

Table 157. POR and VBO Electrical Characteristics and Timing

		T_A = –40°C to 105°C				
Symbol	**Parameter**	**Min**	**Typ[1]**	**Max**	**Units**	**Conditions**
V_{POR}	Power-on reset voltage threshold	2.20	2.45	2.70	V	$V_{DD} = V_{POR}$
V_{VBO}	Voltage Brownout reset voltage threshold	2.15	2.40	2.65	V	$V_{DD} = V_{VBO}$
	$V_{POR}–V_{VBO}$		50	100	mV	
	Starting V_{DD} voltage to ensure valid POR	–	V_{SS}	–	V	
T_{ANA}	Power-on reset analog delay	–	50	–	ms	$V_{DD} > V_{POR}$; T_{POR} Digital Reset delay follows T_{ANA}
T_{POR}	Power-on reset digital delay	—	12	—	µs	66 IPO cycles
T_{VBO}	Voltage Brownout pulse rejection period	—	10	—	ms	$V_{DD} < V_{VBO}$ to generate a Reset
T_{RAMP}	Time for V_{DD} to transition from V_{SS} to V_{POR} to ensure valid Reset	0.10	—	100	ms	
I_{CC}	Supply current		500		µA	V_{DD} = 3.3V

Note:
1. Data in the typical column is from characterization at 3.3 V and 0°C. These values are provided for design guidance only and are not tested in production.

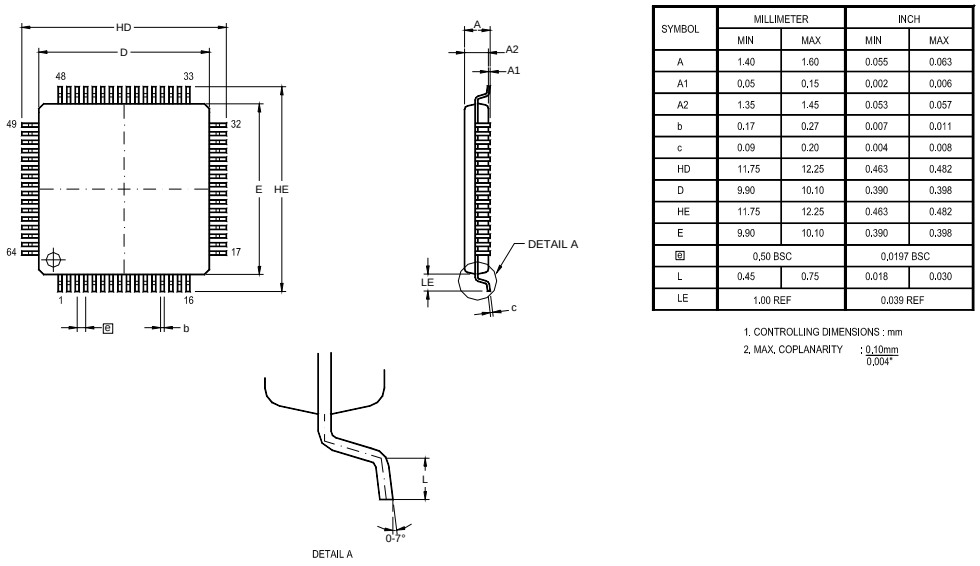
Table 158 lists the Reset and Stop Mode Recovery pin timing.

Table 158. Reset and Stop Mode Recovery Pin Timing

		T_A = –40°C to 105°C				
Symbol	**Parameter**	**Min**	**Typ**	**Max**	**Units**	**Conditions**
T_{RESET}	$\overline{\text{RESET}}$ pin assertion to initiate a System Reset	4	–	–	T_{CLK}	Not in STOP Mode. T_{CLK} = System Clock period.
T_{SMR}	Stop Mode Recovery pin Pulse Rejection Period	10	20	40	ns	RESET, DBG and GPIO pins configured as SMR sources.

Packaging

Figure 70 displays the 64-pin low-profile quad flat package (LQFP) available for the Z16FMC devices.



SYMBOL	MILLIMETER		INCH	
	MIN	MAX	MIN	MAX
A	1.40	1.60	0.055	0.063
A1	0,05	0,15	0,002	0,006
A2	1.35	1.45	0.053	0.057
b	0.17	0.27	0.007	0.011
c	0.09	0.20	0.004	0.008
HD	11.75	12.25	0.463	0.482
D	9.90	10.10	0.390	0.398
HE	11.75	12.25	0.463	0.482
E	9.90	10.10	0.390	0.398
e	0,50 BSC		0,0197 BSC	
L	0.45	0.75	0.018	0.030
LE	1.00 REF		0.039 REF	

1. CONTROLLING DIMENSIONS : mm
2. MAX. COPLANARITY : 0,10mm / 0,004"

Figure 70. 64-Pin Low-Profile Quad Flat Package (LQFP)

Ordering Information

Table 172 identifies the basic features and package styles available for each device within the Z16FMC product line.