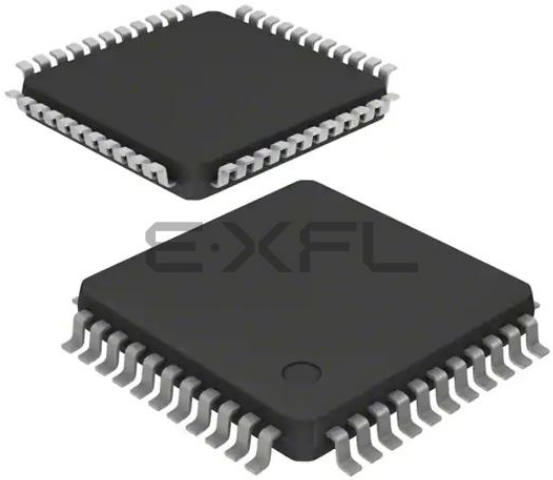

Welcome to E-XFL.COM

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	24MHz
Connectivity	DALI, DMX, I²C, LINbus, SPI, UART/USART, USB
Peripherals	DMA, LVD, POR, PWM, WDT
Number of I/O	36
Program Memory Size	16KB (16K 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 10x12b; D/A 1x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°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/z8f1681an024xk2247

Bit	Description (Continued)
[4] LVD	**Low-Voltage Detection Enable** 0=LVD disabled. 1=LVD enabled (this applies even in Stop Mode).
[3] TEMP	**Temperature Sensor Enable** 0=Temperature Sensor disabled. 1=Temperature Sensor enabled (this applies even in Stop Mode if FRECOV=1).
[2] FRECOV	Fast Recovery 0=Fast Recovery disabled. 1=Fast Recovery enabled. Fast Recovery provides for the shortest Stop-Mode recovery latency at the expense of higher Stop Mode current consumption. See the Reset, Stop-Mode Recovery and Low-Voltage Detection chapter on page 38 to learn more. In addition, this bit must be set for certain peripherals to remain active during Stop Mode as described in this chapter.
[1] COMP0	**Comparator 0 Enable** 0=Comparator 0 is disabled. 1=Comparator 0 is enabled (this applies even in Stop Mode if FRECOV=1).
[0] COMP1	**Comparator 1 Enable** 0=Comparator 1 is disabled). 1=Comparator 1 is enabled (this applies even in Stop Mode if FRECOV=1).

7.10.9. Port A–G Alternate Function Set 1 Subregisters

The Port A–G Alternate Function Set1 Subregister, shown in Table 31, is accessed through the Port A–G Control Register by writing 07h to the Port A–G Address Register. The Alternate Function Set 1 subregisters select the alternate function available at a port pin. Alternate Functions selected by setting or clearing bits of this register are defined in the GPIO Alternate Functions section on page 56.

Table 31. Port A–G Alternate Function Set 1 Subregisters (P*x*AFS1)

Bit	7	6	5	4	3	2	1	0
Field	PAFS17	PAFS16	PAFS15	PAFS14	PAFS13	PAFS12	PAFS11	PAFS10
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	If 07h in Port A–G Address Register, accessible through the Port A–G Control Register							

Note: *x* = A, B, C, D, E, F, G, H, or J.

Bit	Description
[7:0] P*x*AFS1	PAFS1[7:0] – Port Alternate Function Set 1 0: Port Alternate Function selected, as defined in Tables 17 through 21 in the GPIO Alternate Functions section on page 56. 1: Port Alternate Function selected, as defined in Tables 17 through 18 in the GPIO Alternate Functions section on page 56.

Note: Alternate function selection on the port pins must also be enabled as described in the Port A–J Alternate Function Subregisters section on page 88.

Timer(x) Input 1 can also be filtered if the Timer(x–1) Noise Filter is assigned to it by setting NFCON for Timer(x–1). As such, Timer(x–1) Input 0 bypasses the Timer(x–1) Noise Filter, the Timer(x–1) Noise Filter is reassigned to Timer(x) Input 1 and uses the Timer(x) timer clock. When reassigning the Timer 2 Noise Filter, it is connected to Timer 0 Input 1.

Figure 19 shows an configuration example with the Timer0 Noise Filter reassigned to Timer1 (NFCON=1 in the T0NFC Register).

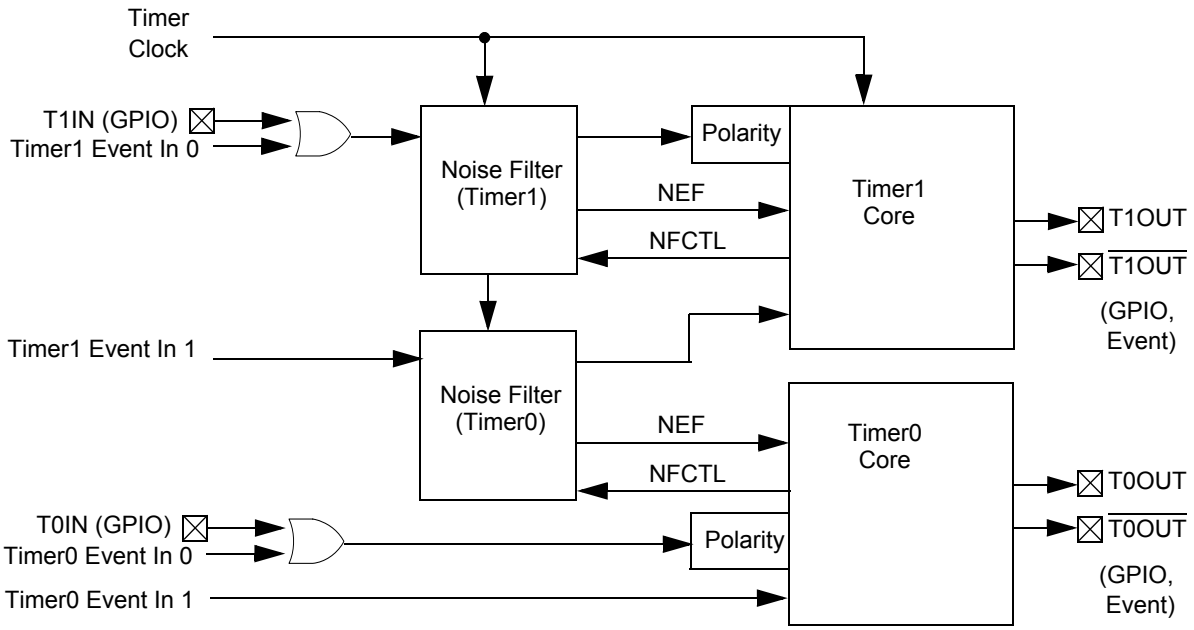

Figure 19. Example with the Timer0 Noise Filter Reassigned to Timer1

Figure 20 shows the operation of the Noise Filter with and without noise. The Noise Filter in this example is a 2-bit up/down counter which saturates at `00` and `11`. A 2-bit counter is described for convenience; the operation of wider counters is similar. The output of the filter switches from 1 to 0 when the counter counts down from `01` to `00` and switches from 0 to 1 when the counter counts up from `10` to `11`. The Noise Filter delays the receive data by three timer clock cycles.

The NEF output signal is checked when the filtered T*x*IN input signal is sampled. The Timer samples the filtered T*x*IN input near the center of the bit time. The NEF signal must be sampled at the same time to detect whether there is noise near the center of the bit time. The presence of noise (NEF=1 at the center of the bit time) does not mean that the sampled data is incorrect; rather, it is intended to be an indicator of the level of noise in the network.

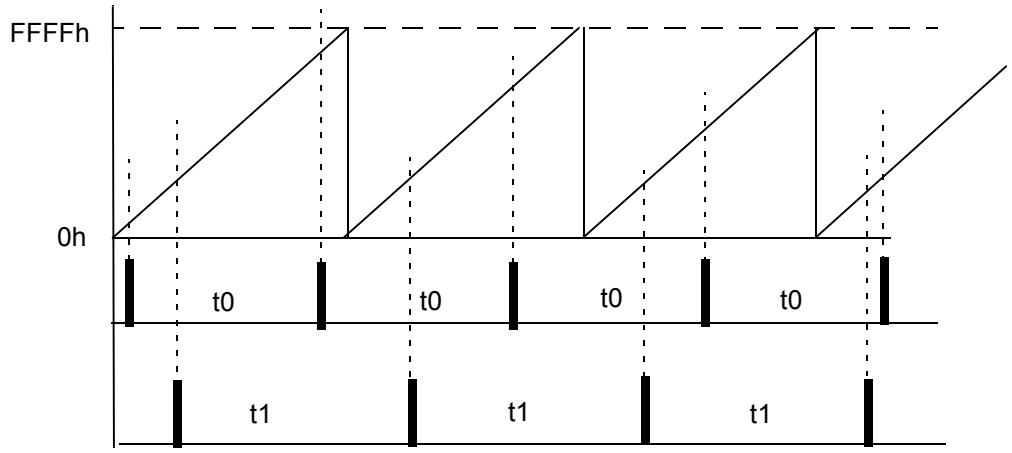

Figure 25. Count Max Mode with Channel Compare

11.7. Multi-Channel Timer Control Register Definitions

11.7.1. Multi-Channel Timer Address Map

Table 89 defines the byte address offsets for the Multi-Channel Timer registers. To save address space, a subaddress is used for the Timer Control 0, Timer Control 1, Channel Status 0, Channel Status 1, Channel-y Control, Channel-y High- and Low-Byte registers. Only the Timer High- and Low-Byte registers and the Reload High- and Low-Byte registers can be directly accessed.

While writing to a subregister, first write its subaddress to the Timer Subaddress Register, then write data to Subregister 0, Subregister 1, or Subregister 2. Reads function the same as writes.

Table 89. Multi-Channel Timer Address Map

Address/ Subaddress	Register/ Subregister Name
Direct Access Register	
FA0	Timer (Counter) High
FA1	Timer (Counter) Low
FA2	Timer Reload High
FA3	Timer Reload Low
FA4	Timer Subaddress
FA5	Subregister 0

Table 97. Multi-Channel Timer Control 1 Register (MCTCTL1)

Bit	7	6	5	4	3	2	1	0
Field	TEN	Reserved	PRES			Reserved	TMODE	
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R	R/W	R/W	R/W	R	R/W	R/W
Address	00h in Subaddress Register, accessible through Subregister 1							

Bit	Description
[7] TEN	**Timer Enable** 0: Timer is disabled and the counter is reset. 1: Timer is enabled to count.
[6]	**Reserved** This bit is reserved and must be programmed to 0.
[5:3] PRES	**Prescale Value** The system clock is divided by the value selected in PRES. The prescaling operation is not applied when the alternate function input pin is selected as the timer clock source. 000: Divide by 1. 001: Divide by 2. 010: Divide by 4. 011: Divide by 8. 100: Divide by 16. 101: Divide by 32. 110: Divide by 64. 111: Divide by 128.
[2]	**Reserved** This bit is reserved and must be programmed to 0.
[1:0] TMODE	**Timer Mode** 00: Count Modulo: the timer counts up to the reload register value, then is reset to 0000h, and counting up resumes. 01: Reserved. 10: Count up/down: the timer counts up to the reload register value, then counts down to 0000h. The count up and count down cycles continue. 11: Reserved.

Chapter 12. Watchdog Timer

The Watchdog Timer (WDT) function helps protect against corrupted or unreliable software and other system-level problems that can place the F6482 Series MCU into unsuitable operating states. The WDT includes the following features:

- Clocked by the Watchdog Timer Oscillator (WTO)
- A selectable time-out response: System Reset or Interrupt
- 16-bit programmable time-out value

12.1. Operation

The WDT is a retriggerable one-shot timer that resets or interrupts the F6482 Series MCU when the WDT reaches its terminal count. The WDT uses the Watchdog Timer Oscillator (WTO) as its clock source. The WDT has only two modes of operation: ON and OFF. After it is enabled, the WDT always counts and must be retriggered to prevent a time-out. An enable can be performed by executing the WDT instruction or by writing the WDT_AO option bit to 0. When 0, The WDT_AO bit enables the WDT to operate continuously, even if a WDT instruction has not been executed.

The WDT is a 16-bit reloadable downcounter that uses two 8-bit registers in the eZ8 CPU register space to set the reload value. The nominal WDT time-out period is calculated using the following equation:

$$\text{WDT Time–Out Period (ms)} = \frac{\text{WDT Reload Value}}{10}$$

In the above equation, the WDT reload value is computed using {WDTH[7:0], WDTL[7:0]} and the typical Watchdog Timer RC Oscillator frequency is 10 kHz. Users must consider system requirements when selecting the time-out delay. Table 103 indicates the approximate time-out delays for the default and maximum WDT reload values.

Table 103. Watchdog Timer Approximate Time-Out Delays

WDT Reload Value (Hex)	WDT Reload Value (Decimal)	Approximate Time-Out Delay (with 10 kHz Typical WDT Oscillator Frequency)	
		Typical	Description
0400	1024	102 ms	Reset default value time-out delay.
FFFF	65,536	6.55 s	Maximum time-out delay.

13.4. Operation

13.4.1. Calendar Mode Operation

In Calendar Mode (MODE=0), The Real-Time Clock maintains time by keeping count of seconds, minutes, hours, day-of-the-week, day-of-the-month, month and year. The current time is kept in a 24-hour format. The format for all count and alarm registers is selectable between binary and binary-coded decimal (BCD) operations. The calendar operation maintains the correct day-of-the-month. Compensation for leap year must be performed by software.

13.4.2. Counter Mode Operation

In Counter Mode (MODE=1), four of the RTC counter registers are cascaded to form a 32-bit counter. An alarm can be configured by loading the RTC alarm registers with the appropriate alarm values and by selecting which counter bytes are enabled for matching in the RTC Alarm Control Register. The counter registers that are utilized in Counter Mode are: RTC_SEC (Byte 3, MSB), RTC_MIN (Byte 2), RTC_HRS (Byte 1), and RTC_DOM (Byte 0, LSB). The corresponding alarm registers that are utilized in Counter Mode are: RTC_ASEC (Byte 3, MSB), RTC_AMIN (Byte 2), RTC_AHRS (Byte 1), and RTC_ADOM (Byte 0, LSB).

13.4.3. Real-Time Clock Alarm

The clock is programmed to generate an alarm condition when the current count matches the alarm set-point registers. In Calendar Mode (MODE=0), alarm registers are available for seconds, minutes, hours, day-of-the-week, and day-of-the-month. In Counter Mode (MODE=1), alarms are available for each of the 4 bytes that comprise the 32-bit counter. Each alarm is independently enabled. To generate an alarm condition, the current time must match all enabled alarm values. For example, if the day-of-the-week and hour alarms are both enabled, the alarm only occurs at a specified hour on a specified day. The alarm triggers an interrupt if configured to do so in the Interrupt Controller. The alarm status, ALARM, is set if the alarm condition is currently met.

Alarm value registers and alarm control registers are written at any time. Alarm conditions are generated when the count value matches the alarm value. The comparison of alarm and count values occurs whenever the RTC count increments. With automatically configured prescaling (FREQ_SEL) of 32,768 kHz, 50 Hz, or 60 Hz, the RTC count increments one time every second. The RTC is also forced to perform a comparison at any time by writing a 1 to the RTC_LOCK bit (the RTC_LOCK bit is not required to be changed to a 0 first).

13.4.4. Real-Time Clock Source Selection

The RTC can be driven by four possible clock sources:

PCLK (CLK_SEL=00). Selecting the 32.768 kHz clock source (FREQ_SEL=00) automatically configures the clock prescaler for this frequency.

14.1.14. UART-LDD and DMA Support

The UART-LDD will assert DMA RX request whenever receive data is available (RDA=1) and will deassert DMA RX request whenever the Receive Data Register is read by the DMA or software. When using DMA, it can be desirable to clear RDAIRQ so that interrupts occur on receive errors but not upon receive data.

The UART-LDD will assert DMA TX request whenever the Transmit Data Register is empty (TDRE=1) and will deassert DMA TX request whenever the Transmit Data Register is written by the DMA or software.

14.1.15. UART-LDD Baud Rate Generator

The UART-LDD Baud Rate Generator creates a lower frequency baud rate clock for data transmission. The input to the Baud Rate Generator is the system clock. The UART-LDD Baud Rate High and Low Byte registers combine to create a 16-bit baud rate divisor value (BRG[15:0]) that sets the data-transmission rate (baud rate) of the UART-LDD. The UART-LDD data rate for normal UART operation and DMX operation is calculated using the following equation:

$$\text{UART Data Rate (bits/s)} = \frac{\text{System Clock Frequency (Hz)}}{16 \times \text{UART Baud Rate Divisor Value}}$$

The UART-LDD data rate for LIN Mode UART operation is calculated using the following equation:

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

The UART-LDD data rate for DALI Mode operation is calculated using the following equation:

$$\text{UART Data Rate (bits/s)} = \frac{\text{System Clock Frequency (Hz)}}{32 \times \text{UART Baud Rate Divisor Value}}$$

When the UART-LDD is disabled, the BRG functions as a basic 16-bit timer with interrupt on time-out. To configure the BRG as a timer with interrupt on time-out, follow the procedure below:

1. Disable the UART-LDD receiver by clearing the REN bit in the UART-LDD Control 0 Register to 0 (i.e., the TEN bit can be asserted; transmit activity can occur).
2. Load the appropriate 16-bit count value into the UART-LDD Baud Rate High and Low Byte registers.

Bit	Description (Continued)
[3] STRTPOL	**Start Bit Polarity** 0: Start bit is a logic 0. 1: Start bit is a logic 1. STRTPOL is typically set for DALI.
[2] BITORD	**Bit Order** 0: Standard UART bit order with the LSB (TxD[0]/RxD[0]) transmitted/received first. 1: DALI bit order with the MSB (TxD[7]/RxD[7]) transmitted/received first.
[1] CLSNE	**Collision Detection Enable** 0: Collision detection is disabled. 1: Collision detection is enabled. CLSNE should be set only for DALI master transmissions. If a collision occurs while CLSNE is set, CLSN will be set in the UART-LDD Status Register and an interrupt will be generated.
[0] PARTRXE	**Partial Byte Reception Enable** PARTRXE has an effect only when receiving. 0: Partial bytes are not loaded into RXDATA. 1: Partial bytes are loaded into RXDATA if a partial byte has been received upon receiving a stop bit.

13.3.10. DMX Control Registers

When MSEL=101b, the DMX Control 0–1 registers, shown in Table 134, provide control for the DMX Mode of operation.

Table 134. DMX Control 0-1 Registers (U*x*CTL1 with MSEL=101b)

Bit	7	6	5	4	3	2	1	0
Field	DMXMST	DMXSLV	DMXSIRQ		Reserved	AUTOBRK	WFBRK	COMP_ADDR[8]
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R	R/W	R/W	R/W
Address	U0CTL1 @ F43h, U1CTL1 @ F4Bh							

Note: R/W=read/write; *x* = 0,1.

Bit	Description
[7] DMXMST	**DMX Master Mode** 0: DMX Master Mode not selected. 1: DMX Master Mode selected.
[6] DMXSLV	**DMX Slave Mode** 0: DMX Slave Mode not selected. 1: DMX Slave Mode selected.

15.3.3.2. Inter-IC Sound (I^2S) Mode

This mode is selected by setting the SSMD field of the Mode Register to 010. The PHASE and CLKPOL bits of the Control Register must be cleared to 0 and the ESPIEN1 bit must be set=1 (ESPIEN0 can be either 1 or 0). If the ESPI is being used to both send and receive I^2S data, the TDRE interrupt should be serviced before the RDRF interrupt. Figure 41 shows I^2S Mode, with $\overline{SS}$ alternating between consecutive frames. Each audio frame consists of a fixed number of bits, typically a multiple of 8 bits such as 16.

The SSV indicates whether the corresponding bytes are left or right channel data. The SSV value must be updated when servicing the TDRE interrupt/request for the first byte in a left or write channel frame. This update can be made by performing a byte write to update SSV, followed by a byte Write to the Data Register. The $\overline{SS}$ signal will lead the data by one SCK period.

A DMA can be used to transfer audio data, but software must toggle the SSV bit in sync with the first data byte of each left/right channel word.

A transaction is terminated when the master has no more data to transmit. After the last bit is transferred, SCK will stop, and $\overline{SS}$ and SSV will return to their default states. A transmit underrun error will occur at this point.

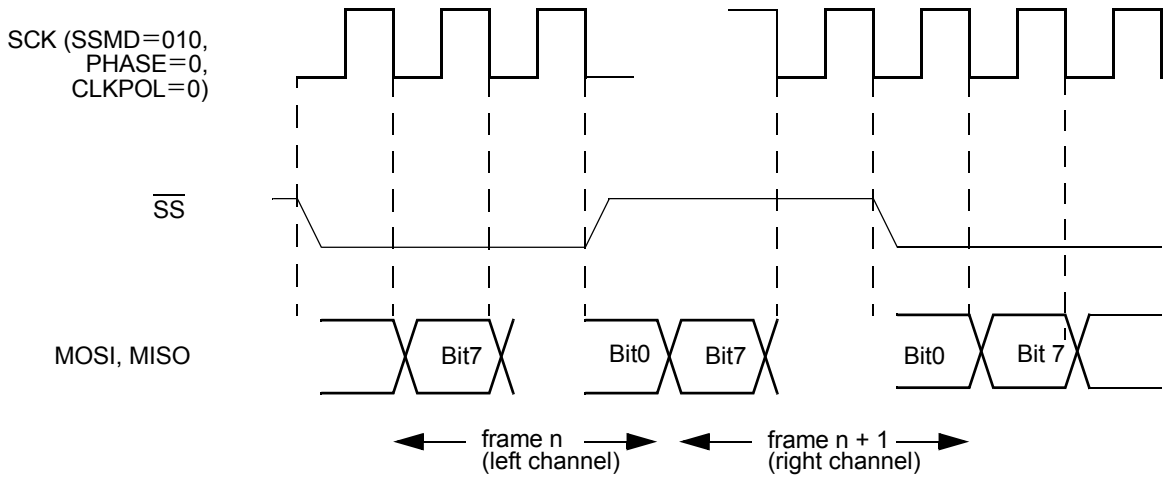

Figure 41. I^2S Mode (SSMD=010), Multiple Frames

15.3.4. SPI Protocol Configuration

This section describes how to configure the ESPI block for the SPI protocol. In the SPI protocol, the master sources the SCK and asserts Slave Select signals to one or more slaves. The Slave Select signals are typically active Low.

Slave 10-Bit Address Recognition Mode. If IRM=0 during the address phase and the controller is configured for MASTER/SLAVE or SLAVE 10-BIT ADDRESS Mode, the hardware detects a match to the 10-bit slave address defined in the I2CMODE and I2CSLVAD registers and generates the slave address match interrupt (the SAM bit=1 in the I2CISTAT Register). The I^2C controller automatically responds during the Acknowledge phase with the value in the NAK bit of the I2CCTL Register.

16.2.6.2. General Call and Start Byte Address Recognition

If GCE=1 and IRM=0 during the address phase and the controller is configured for master/slave or slave in either 7- or 10-bit address modes, the hardware detects a match to the General Call Address or the start byte and generates the slave address match interrupt. A General Call Address is a 7-bit address of all zeroes, with the R/$\overline{W}$ bit=0. A start byte is a 7-bit address of all zeroes, with the R/$\overline{W}$ bit=1. The SAM and GCA bits are set in the I2CISTAT Register. The RD bit in the I2CISTAT Register distinguishes a General Call Address from a start byte which is cleared to 0 for a General Call Address). For a General Call Address, the I^2C controller automatically responds during the address acknowledge phase with the value in the NAK bit of the I2CCTL Register. If the software is set to process the data bytes associated with the GCA bit, the IRM bit can optionally be set following the SAM interrupt to allow the software to examine each received data byte before deciding to set or clear the NAK bit. A start byte will not be acknowledged – a requirement of the I^2C specification.

16.2.6.3. Software Address Recognition

To disable hardware address recognition, the IRM bit must be set to 1 prior to the reception of the address byte(s). When IRM=1, each received byte generates a receive interrupt (RDRF=1 in the I2CISTAT Register). The software must examine each byte and determine whether to set or clear the NAK bit. The slave holds SCL Low during the acknowledge phase until the software responds by writing to the I2CCTL Register. The value written to the NAK bit is used by the controller to drive the I^2C bus, then releasing the SCL. The SAM and GCA bits are not set when IRM=1 during the address phase, but the RD bit is updated based on the first address byte.

16.2.6.4. Slave Transaction Diagrams

In the following transaction diagrams, the shaded regions indicate data transferred from the master to the slave and the unshaded regions indicate the data transferred from the slave to the master. The transaction field labels are defined as follows:

S	Start
W	Write
A	Acknowledge
A	Not Acknowledge
P	Stop

16.2.6.5. Slave Receive Transaction with 7-Bit Address

The data transfer format for writing data from a master to a slave in 7-bit address mode is shown in Figure 50. The procedure that follows describes the I²C Master/Slave Controller operating as a slave in 7-bit addressing mode and receiving data from the bus master.

S	Slave Address	W=0	A	Data	A	Data	A	Data	A/$\overline{A}$	P/S

Figure 50. Data Transfer Format, Slave Receive Transaction with 7-Bit Address

1. The software configures the controller for operation as a slave in 7-bit addressing mode, as follows:
   a. Initialize the MODE field in the I²C Mode Register for either SLAVE ONLY Mode or MASTER/SLAVE Mode with 7-bit addressing.
   b. Optionally set the GCE bit.
   c. Initialize the SLA[6:0] bits in the I²C Slave Address Register.
   d. Set IEN=1 in the I²C Control Register. Set NAK=0 in the I²C Control Register.

2. The bus master initiates a transfer, sending the address byte. In SLAVE Mode, the I²C controller recognizes its own address and detects that R/$\overline{W}$ bit=0 (written from the master to the slave). The I²C controller acknowledges, indicating it is available to accept the transaction. The SAM bit in the I2CISTAT Register is set to 1, causing an interrupt. The RD bit in the I2CISTAT Register is cleared to 0, indicating a write to the slave. The I²C controller holds the SCL signal Low, waiting for the software to load the first data byte.
3. The software responds to the interrupt by reading the I2CISTAT Register (which clears the SAM bit). After seeing the SAM bit to 1, the software checks the RD bit. Because RD=0, no immediate action is required until the first byte of data is received. If software is only able to accept a single byte, it sets the NAK bit in the I2CCTL Register at this time.
4. The master detects the Acknowledge and sends the byte of data.
5. The I²C controller receives the data byte and responds with an Acknowledge or a Not Acknowledge, depending on the state of the NAK bit in the I2CCTL Register. The I²C controller generates the receive data interrupt by setting the RDRF bit in the I2CISTAT Register.
6. The software responds by reading the I2CISTAT Register, finding the RDRF bit=1 and reading the I2CDATA Register clearing the RDRF bit. If software can accept only one more data byte, it sets the NAK bit in the I2CCTL Register.
7. The master and slave loops through Step 4 to Step 6 until the master detects a Not Acknowledge instruction or runs out of data to send.

- When the second DMA interrupt occurs, it indicates that the Nth byte has been received. Set the stop bit in the I2CCTL Register; this stop bit is polled by software to determine when the transaction is actually completed.
- Clear the DMAIF bit in the I2CMODE Register.

16.2.7.3. Slave Write Transaction with Data DMA

In a transaction in which the I^2C master/slave operates as a slave that receives data written by a master, the software must set the NAK bit after the (N–1)st byte has been received or during the reception of the last byte. As in the Master Read transaction described previously, the watermark DMA interrupt is used to notify software when the (N–1)st byte has been received.

- Configure the selected DMA channel for I^2C receive. The IEOB bit must be set in the DMAxCTL0 Register for the last 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 setting WMCNT to 0001 in the DMAxCNTH Register.
- 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 a previous section 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.

16.2.7.4. 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 DMAxCTL0 Register for the last 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 last 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.

Table 166. Determining USB Endpoint Buffer Memory Allocation with Only Endpoints 0, 1 and 2 Used

Subregister	Subregister Value	Endpoint Buffer Size (Bytes)
USBO1ADDR	OUT_EP0_SIZE ÷ 2	OUT_EP0_SIZE = 2 * USBO1ADDR
USBO2ADDR	USBO1ADDR + (OUT_EP1_SIZE ÷ 2)	OUT_EP1_SIZE = 2 * (USBO2ADDR –USBO1ADDR)
USBO3ADDR	00h	OUT_EP2_SIZE = 2 * ((2 * USBISTADDR) –USBO2ADDR)
USBISTADDR	(USBO2ADDR + (OUT_EP2_SIZE ÷ 2)) ÷ 2	OUT_EP3_SIZE = n/a
USBI1ADDR	IN_EP0_SIZE ÷ 2	IN_EP0_SIZE = 2 * USBI1ADDR
USBI2ADDR	USBI1ADDR + (IN_EP1_SIZE ÷ 2)	IN_EP1_SIZE = 2 * (USBI2ADDR–USBI1ADDR)
USBI3ADDR	00h	IN_EP2_SIZE = 2 * ((8 * USBISPADDR) –USBI2ADDR)
USBISPADDR	((USBI2ADDR + (IN_EP2_SIZE ÷ 2)) ÷ 8) + (USBISTADDR ÷ 4)	IN_EP3_SIZE = n/a

17.2.4. USB Control Transfers Using Endpoint 0

A control transfer consists of two or three stages:

- Setup stage
- Data stage (optional)
- Status stage

The following describes control write and control read transfers, in addition to the associated status and interrupt bit.

17.2.4.1. Control Write

In the Setup stage of a control transfer, after receiving a Setup token, the USB Module sets the HSNAK bit in the USB Endpoint 0 Control and Status Subregister (USBEP0CS) and the SUTOKIRQ bit in the USB Protocol Interrupt Request Subregister (USBIRQ). A USB interrupt is generated if the SUTOKIEN bit is set in the USB Protocol Interrupt Enable Subregister (USBIEN). Subsequently, if an 8-byte data packet is received correctly, the USB Module sets the SUDAVIRQ bit in the USBIRQ Subregister and a USB interrupt is generated if the SUDAVIEN bit is set in the USBIEN Subregister. The 8-byte data packet can be accessed from the USB Setup Buffer Byte 0–7 subregisters (USBSUx), as described in the Setup Buffer section on page 347.

The Data stage of a control transfer consists of one or more OUT bulk-like transactions. After each correct OUT packet is received during the Data stage, the USB Module sets the OUT0IRQ bit in the USB OUT Interrupt Request Subregister (USBOUTIRQ), and a USB interrupt is generated if the OUT0IEN is set in the USB OUT Interrupt Enable Subregister (USBOUTIEN). The OUT0BC Subregister contains the number of data bytes received in the last OUT transaction. Software should service the interrupt request, then prepare the endpoint for the next transaction by reloading the OUT0BC Subregister with any value, which results in hardware setting the OUTBUSY bit in the USBEP0CS Subregister. Until this preparation task is performed, the USB Controller will NAK subsequent data packets.

The Status stage of a control transfer is the final operation in the sequence. Software should clear the HSNAK bit (by writing a 1 to it) to instruct the USB Module to ACK the Status stage. The USB Module sends the STALL handshake when both HSNAK and STALL bits are set. Prior to the Status stage, after the last successful transaction in the Data stage when all expected bytes of the transfer have been received or sent by the USB Module and a STALL handshake is to be sent for any additional data stage tokens, DSTALL is typically set by software. When DSTALL is set, the USB Module will send a STALL handshake if additional data stage tokens are sent and, if this transmission occurs, the STALL bit will be automatically set so that the USB Module will send a STALL handshake in the Status stage. If DSTALL is set, a token that indicates a transition to the status stage (e.g., an OUT token for an IN endpoint) will not cause a STALL handshake.

A control write transfer example is shown in Figure 56.

19.6. Event System Register Definitions

Four register addresses provide access to the Event System subregisters that control source and destination selection for each Event System channel. Table 219 lists these Event System registers and subregisters.

Table 219. Event System Registers and Subregisters

Event System Source Selection Registers and Subregisters		
Register Mnemonic	**Address**	**Register Name**
ESSSA	F98h	Event System Source Subaddress Register
ESSSD	F99h	Event System Source Subdata Register
Subregister Mnemonic	**Subregister Address***	**Subregister Name**
ESCHxSRC	0–7h	Event System Channel 0–7 Source subregisters

Event System Destination Selection Registers and Subregisters		
Register Mnemonic	**Address**	**Register Name**
ESDSA	F9Ah	Event System Destination Subaddress Register
ESDSD	F9Bh	Event System Destination Subdata Register
Subregister Mnemonic	**Subregister Address***	**Subregister Name**
ESCHDSTx	0–3Fh	Event System Destination 0–3F Channel subregisters

Note: *The ESSSA Register contains the ESCHxSRC Subregister address; the ESDSA Register contains the ESCHDSTx Subregister address.

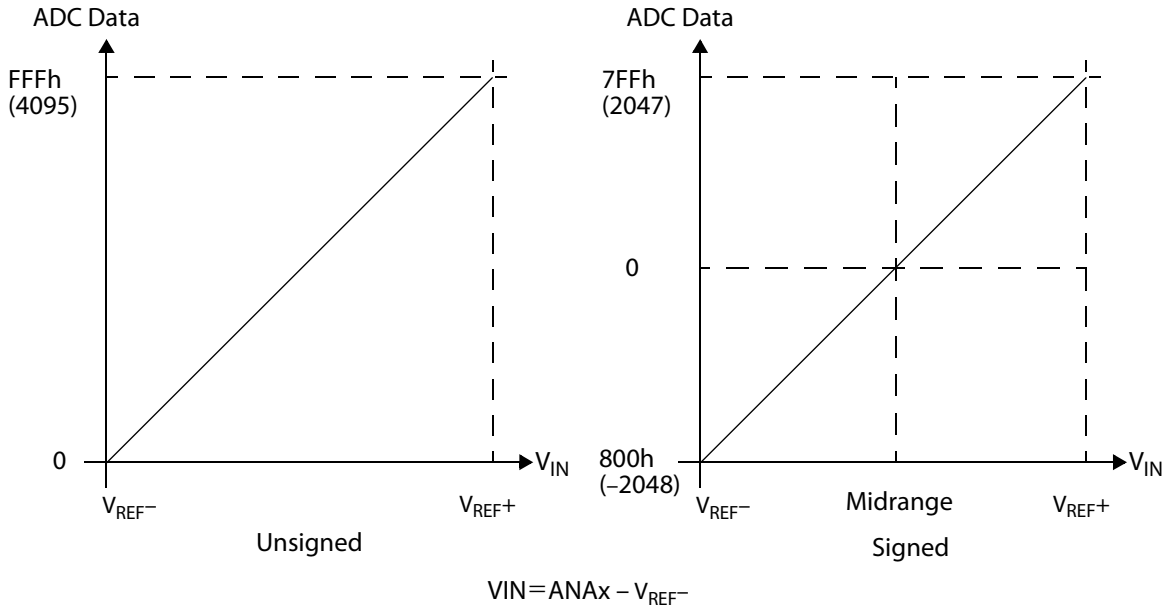

Figure 71. ADC Data (12-Bit) vs. Input Voltage for Single-Ended Input Modes

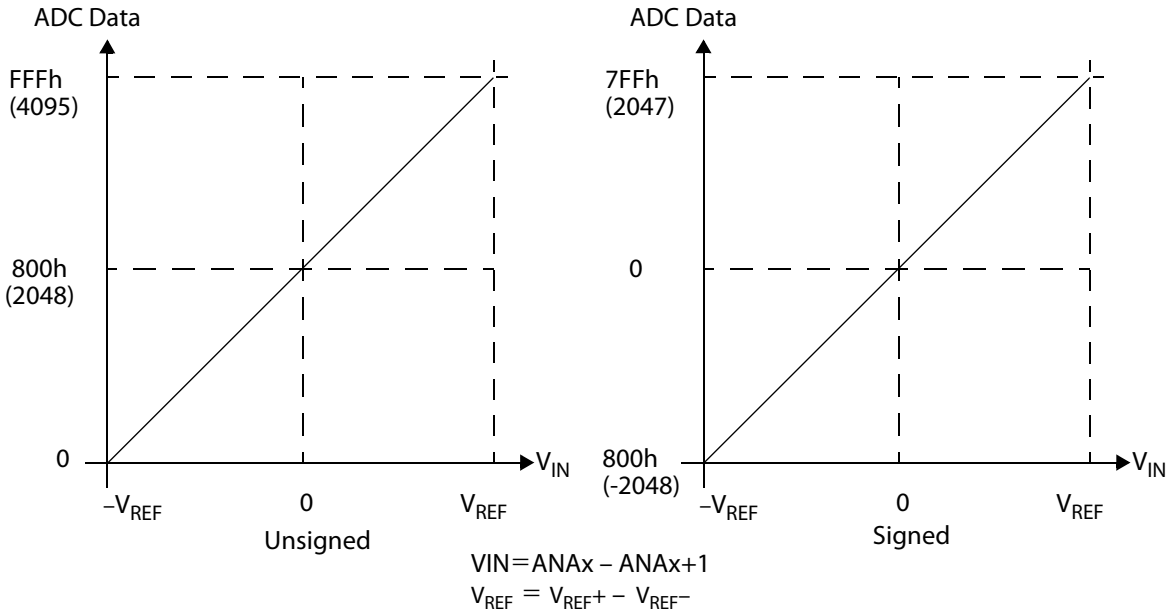

Figure 72. ADC Data (12-Bit) vs. Input Voltage for Balanced Differential Input Mode

Bit	Description (Continued)
RESOLUT = 0	
[7:4]	Reserved: these bits must be programmed to 0000.
[3:0]	0–F: The 4 MSBs of the last conversion result are compared against the 4 LSBs of this data register.
RESOLUT = 1	
[7:6]	Reserved: these bits must be programmed to 00.
[5:0]	00–3F: The 6 MSBs of the last conversion result are compared against the 6 LSBs of this data register.

21.3.11. ADC Window Upper Threshold Low Register

The ADC Window Upper Threshold Low Register, shown in Table 245, contains the unsigned LSBs of the ADC window upper threshold. This register is used in conjunction with ADCUWINH to define the ADC window upper threshold.

Table 245. ADC Window Upper Threshold Low Register (ADCUWINL)

Bits	7	6	5	4	3	2	1	0
Field	UWINL							
Reset	FFh							
R/W	R/W							
Address	F7Ah							

Bit	Description
UWINL	**ADC Window Upper Threshold Low** The data in this register is unsigned. Interrupt is asserted if the ADC result is greater than the value of UWINH and UWINL.
RESOLUT = x	
[7:0]	00–FF: The 8 LSBs of the last conversion result are compared against this data register.

Chapter 24. Comparators and Reference System

The F6482 Series devices feature a reference system and two identical general-purpose, rail-to-rail comparators, each of which compares two analog input signals with four speed-vs.-power settings and three hysteresis options. A 4-to-1 input multiplexer exists on each comparator positive input and each comparator negative input. Multiplexing can be configured such that a GPIO (C0INP/C1INP) pin provides a positive comparator input and/or a GPIO (C0INN/C1INN) provides a negative input. The output of each comparator is available as an interrupt source and can be routed to an external pin using the GPIO multiplex, as well as to the Event System.

Features for each comparator include:

- Positive input selections offering a GPIO, a temperature sensor and op amp outputs AMPAOUT and AMPBOUT
- Negative input selections offering a GPIO, fixed internal reference levels, a programmable internal reference, and the DAC
- Output can be an interrupt source
- Output can drive an external pin and/or be an Event System source
- Operation in Stop Mode
- Power-vs.-speed control with four available settings
- Hysteresis control with three available settings
- Window detection: signal above window, signal inside window, signal below window
- Additional output in the form of a logical OR of each comparator output, is an Event System source, is useful for window detection signaling

Features of the Reference System are as follows:

- Reference Generator:
  - Fixed reference voltages including: the bandgap voltage, $AV_{DD}/2$, 0.75V, 1.0V, and 1.25V, that are available to certain internal functions
  - VBIAS with four selectable levels (2.5V, 2.0V, 1.5V, 1.25V) that is available as an internal positive voltage reference for the ADC and as a GPIO alternate function to provide a low-power external reference
  - DAC internal positive voltage reference with four selectable levels (2.5V, 2.0V, 1.5V, 1.25V)
- Two programmable references provide 32 taps (steps), with the highest tap selectable as either VBIAS or AV_{DD}

33.4.15. Universal Serial Bus

Table 345 presents electrical and timing data for the F6482 Series' Universal Serial Bus (USB) function.

Table 345. USB Electrical Characteristics

		T_A=–40°C to +85°C V_{DD}=3.0V to 3.6V				
Symbol	**Parameter**	**Min**	**Typ***	**Max**	**Units**	**Conditions**
V_{IL}	Low Level Input Voltage			0.8	V	
V_{IH}	High Level Input Voltage	2.0			V	
V_{CM}	Common Mode Range	0.8		2.5	V	
V_{DI}	Differential Input Sensitivity	0.2			V	
V_{OL}	Low Level Output Voltage	0		0.3	V	1.5kΩ to 3.6V and 27Ω external series resistor.
V_{OH}	High Level Output Voltage	2.8		V_{DD}	V	15kΩ to VSS and 27Ω external series resistor.
R_{DR}	D+, D- Driver Output Resistance	28		44	Ω	Including 27Ω external series resistor.
R_{PUUSB}	Internal Pull-up Resistor	900		1575	Ω	Idle bus state.
		1425		3090	Ω	Receiving.

Note: *Data in the Typical column is from characterization at 3.0V and 25°C. These values are provided for design guidance only, and are not tested in production.

33.4.16. Internal Precision Oscillator

Table 346 presents electrical and timing data for the F6482 Series' Internal Precision Oscillator (IPO) function.

Table 346. IPO Electrical Characteristics

		V_{DD}=1.8V to 3.6V T_A=–40°C to +85°C				
Symbol	**Parameter**	**Min**	**Typ***	**Max**	**Units**	**Conditions**
I_{DD}IPO	IPO Active Supply Current		1.3		µA	

Note: *Data in the Typical column is from characterization at 3.0V and 25°C. These values are provided for design guidance only and are not tested in production.