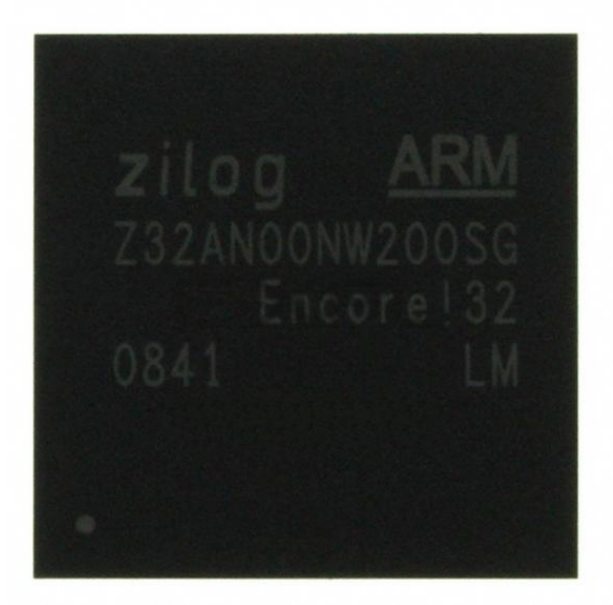


Welcome to E-XFL.COM

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	Obsolete
Core Processor	ARM9®
Core Size	16/32-Bit
Speed	200MHz
Connectivity	EBI/EMI, IrDA, SmartCard, SPI, UART/USART, USB OTG
Peripherals	DMA, LCD, Magnetic Card Reader, POR, PWM, WDT
Number of I/O	76
Program Memory Size	-
Program Memory Type	ROMless
EEPROM Size	-
RAM Size	64K x 8
Voltage - Supply (Vcc/Vdd)	1.71V ~ 3.6V
Data Converters	A/D 4x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	256-LBGA
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/zilog/z32an00nw200sg

Chapter 11: Real-Time Clock (RTC) 93
11.1 Real-Time Clock Time/Counter Registers 93
11.2 RTC Alarm 93
11.3 RTC Wake 93
11.4 RTC Oscillator Source 93
11.5 RTC Battery Backup 93
11.6 RTC Reset 93
11.7 Oscillator External Circuit 94
11.8 RTC Registers (Base → FFFFB000h) 94
11.8.1 Current Time Registers 94
11.8.2 Alarm Registers 96
11.8.3 Control Registers 98
11.9 RTC Locking 99
11.9.1 Address FFFFC00Ch: RTC_APB_STA – RTC APB Status Register 99
11.9.2 Address FFFFC694h: RTC_LCK1 – RTC Lock 1 Register 100
11.9.3 Address FFFFC698h: RTC_LCK2 – RTC Lock 2 Register 100

Chapter 12: Random Number Generator (RNG) 101
12.1 Programming Guide 101
12.2 RNG Registers (Base → FFFFA000h) 101
12.2.1 Offset 000h: RNG_DATA – Random Number Generator Data Register 101
12.2.2 Offset 004h: RNG_CTRL – Random Number Generator Control Register 102

Chapter 13: SHA-1 103
13.1 Programming Guide 103
13.2 SHA-1 Registers (Base → FFFF9000h) 103
13.2.1 Offset 000h: SHA1_H – Hashed Value 103
13.2.2 Offset 014h: SHA1_DATA_IN – Data In 103
13.2.3 Offset 018h: SHA1_CONTROL – SHA-1 Control 104
13.2.4 Offset 01Ch: SHA1_STATUS – SHA-1 Status 104
13.2.5 Offset 020h: SHA1_WH – SHA-1 Initialization Value 104

Chapter 14: Analog-to-Digital Converter (ADC) 105
14.1 Voltage Reference 105
14.2 Clock / Sample Rate 105
14.3 Modes of Operation 105
14.3.1 Continuous Rotating 105
14.3.2 Single Shot 105
14.4 DMA Operation 105
14.5 Registers (Base → FFFF2000h) 106
14.5.1 Offset 000h: ADC_CFG – ADC Configuration Register 106
14.5.2 Offset 004h: ADC_CMD – ADC Command Register 107
14.5.3 Offset 008h: ADC_FIFO – ADC FIFO 107
14.5.4 Offset 00Ch: ADC_INT – ADC Interrupt Register 108
14.5.5 Offset 010h: ADC_STA – ADC Status Register 109

Chapter 15: LCD Interface 110
15.1 Interface Timing 110
15.1.1 Read Cycle 110
15.1.2 Write Cycle 110
15.2 Read and Write Commands 111
15.3 4-bit and 8-bit Operation 111
15.4 DMA Operation 111
15.5 LCD Interface Registers (Base → FFFED000h) 111
15.5.1 Offset 000h: LCD_CFG – LCD Configuration Register 111
15.5.2 Offset 004h: LCD_RD – LCD Read Register 112

6.3 Configuring the Interrupt Controller

Prior to configuring a particular channel, a number of settings must be made:

1. The ARM exception table should be set up. This typically is done by remapping the SRAM to appear at the bottom of memory and placing the following ARM instruction at locations 0x00000018 (IRQ Vector) and 0x0000001C (FIQ Vector) as shown in the example below:
   - LDR PC, [PC,#0xFFFFFF00 - (0x18+8)] ; IRQ Vector (located at A=0x0018)
   - LDR PC, [PC,#0xFFFFFF04 - (0x1C+8)] ; FIQ Vector (located at A=0x001C)
2. INTC_DFLT should be initialized to the address of an error handling ISR. This is to provide a vector for error handling in the case where INTC_IVEC or INTC_FVEC are read when there is no interrupt active. This usually indicates a spurious interrupt or a software error.
3. Set up any or all of the interrupt channels. The following registers and fields are used:
   - INTC_CFGN → IRQ_FIQ, PRI
   - INT_VEC_N → All Bits
   - INTC_EN → Bit "N"
4. The "I" and/or "F" bits of the ARM922T CPSR register should be cleared to enable IRQ and/or FIQ interrupts.

6.4 ISR Invocation

Below is an example of how the CPU, ARM Exception Table, INTC_VECN's, and INTC_IVEC operate together to invoke a Channel 1 ISR upon occurrence of a Channel 1 interrupt. Note the sequence of steps listed at the bottom of the table. FIQ ISR invocation is almost identical, except that the CPU will read the FIQ vector in the ARM exception table and this will cause INTC_FVEC to be read.

1. IRQ is set up:
   - The IRQ Vector, INTC_VEC_1 are initialized and the Channel 1 ISR is located as shown above.
   - Channel 1 is configured as an IRQ (as opposed to an FIQ), programmed to have the highest priority, and enabled.
   - A channel 1 interrupt enters the INTC, which is passed on to the CPU by activating IRQ.
2. An active IRQ causes the CPU to execute the IRQ vector instruction at 00000018h:
   - LDR PC, [PC, #0xFFFFFF00-(0x18+8)]
3. LDR instruction causes CPU to read memory location FFFFFF00h (INTC_IVEC).
4. INTC provides the vector of the highest priority active interrupt (00000180h).
5. CPU move the data 00000180h) into PC. Program execution then continues at this address.

6.5 ISR Return from Interrupt

Before returning from an IRQ or FIQ interrupt, the ISR must write to INTC_IEND or INTC_FEND. The data written is not important and is discarded. After this write is done, the final instruction of an IRQ or FIQ ISR must restore the PC and CPSR. When using nested interrupts, the interrupts should be disabled before writing to INTC_IEND.

6.6 Interrupt Nesting

The IRQ processor nests interrupts (preemption of an ISR by a higher priority interrupt). The FIQ processor cannot nest interrupts. Active and enabled interrupt channels with a higher programmed priority always result in an interrupt being passed on to the CPU (via either IRQ or FIQ). To utilize this feature, software must clear the "I" bit of the ARM922T CPSR register within the ISR to re-enable interrupts, thereby "enabling" interrupt nesting. More information can be found in Chapter A2, Programmer's Model of the *ARM Architecture Reference Manual*.

To keep track of the nested interrupts, the IRQ Processor contains a priority stack. Since there are 8 priorities, the stack is 8 deep. A read of INTC_IVEC pushes the stack. A write to INTC_IEND pops the stack. During the time between push and pop, interrupts of the same or lower priority are masked. The programmer's responsibility is to ensure there is a write to INTC_IEND every read of INTC_IVEC.

31:00	WO	0	**Set (SET):** Writes of '1' set the corresponding bit in INTC_SWINT.

6.8.11 Offset 028h: INTC_SWINT_CLR – Software Interrupt Clear Register

Bits	Type	Reset	Description
31:00	WO	0	**Clear (CLR):** Writes of '1' clear the corresponding bit in INTC_SWINT

6.8.12 INTC_VECN – Channel N Vector Register

Offset	Channel	Offset	Channel	Offset	Channel	Offset	Channel
080h	0	0A0h	8	0C0h	16	0E0h	24
084h	1	0A4h	9	0C4h	17	0E4h	25
088h	2	0A8h	10	0C8h	18	0E8h	26
08Ch	3	0ACh	11	0CCh	19	0ECh	27
090h	4	0B0h	12	0D0h	20	0F0h	28
094h	5	0B4h	13	0D4h	21	0F4h	29
098h	6	0B8h	14	0D8h	22	0F8h	30
09Ch	7	0BCh	15	0DCh	23	0FCh	31

There is one register per interrupt channel, as shown in the table above. The bits in each register are described in the table below.

Bits	Type	Reset	Description
31:00	WO	0	**Vector (VEC):** Contains the value to appear in either INTC_IVEC or INTC_FVEC when channel N is the highest priority active interrupt.

Figure 7-12: Sync Burst Flash Configuration (AM29BL802C)

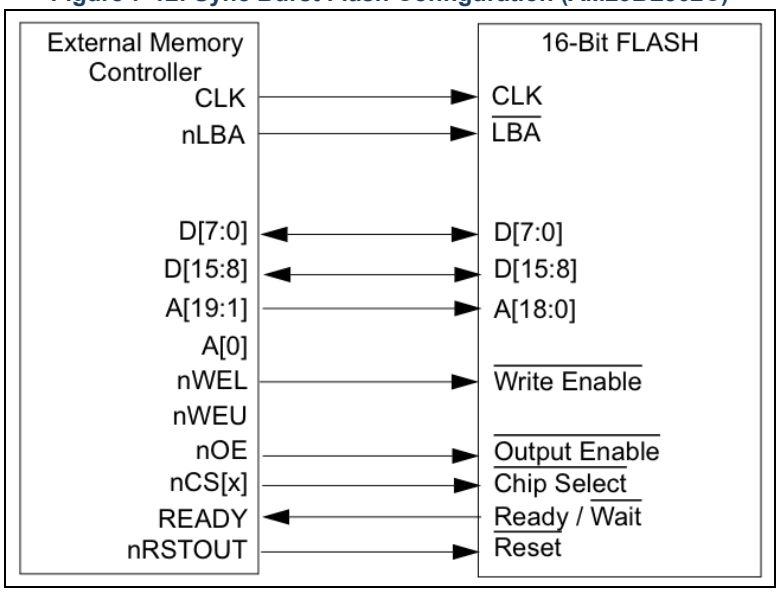
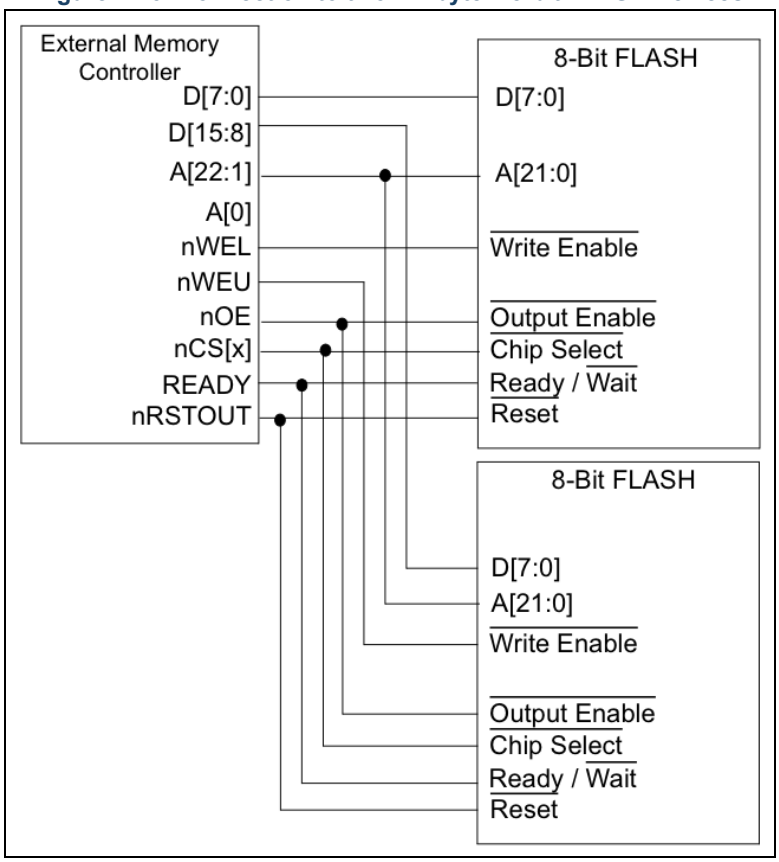


Figure 7-13: Connection to two 4M byte x 8-bit FLASH Devices



8.5 Count-to-Zero Condition

When a channel AHB burst completes, the DMA controller checks to see if DMA_CNTN has been decremented to 0. If so, there are two possible responses:

- If DMA_CRLDN.EN is set, DMA_SRCN, DMA_DESTN, and DMA_CNTN are loaded from the reload registers and the channel remains active using the newly loaded address/count values and the previously programmed configuration values.
- If DMA_STAN.RLOAD is cleared, the channel is disabled and DMA_STAN.EN is cleared to '0'.

8.6 Chaining Buffers

The reload registers can be used for chaining buffers together. This allows the DMA to continue to service a request without immediate processing from the CPU. When a count-to-zero condition occurs with DMA_CRLDN.EN set to '1', the channel remains active, the DMA_SRCN, DMA_DESTN, and DMA_CNTN is loaded with values from the reload registers, and the DMA operation continues with the new DMA buffer. To prevent improper operation, program the address before setting DMA_CRLDN.EN. The following fields affect buffer chaining.

- DMA_SRLDN → All bits
- DMA_DRLDN → All bits
- DMA_CRLDN → EN, COUNT

8.7 DMA Interrupts

The following registers and fields control DMA interrupts.

- DMA_CTRL → IENx
- DMA_ISTAT → PENDx
- DMA_CFGN → CTZ_IEN, STA_IEN
- DMA_STAN → EN, IPEND, CTZ, RLOAD, BUS_ERR, TO

8.8 Channel Time-outs

Each channel can be configured to generate an interrupt when its associated request line is inactive. An example of this feature is to determine an idle UART receive channel. The time-out mechanism consists of a single global 24-bit pre-scalar and per-channel programmable 8-bit timers. The global pre-scalar has 3 taps: a divide by 256, a divide by 64k, or a divide by 16M.

Each channel's 8-bit timer increments using a tap chosen by DMA_CFGN.PS. DMA_CFGN.TO selects how high the timer must count before generating an interrupt. The timer is reset whenever any of the following conditions occurs:

- The DMA request line programmed for the channel is activated.
- The channel is disabled for any reason (DMA_STAN.EN is zero).

Any timer can be disabled by clearing DMA_CFGN.PS to '00'. When all 8 channels are configured, the global pre-scalar is disabled. Normally, the timer starts counting as soon as the channel is enabled and DMA_CFGN.PS is non-zero. But if DMA_CFGN.WAIT is set, the timer starts counting only after the first DMA request is received from the peripheral. The time-out period can be calculated using the following equation: $T_{TIMEOUT} = T_{HCLK} \times N_{PS} \times N_{TO}$

With hclk frequency of 90 MHz and PS=10b and TO=100b, the time-out period is:

$$T_{TIMEOUT} = \frac{65,536 \times 64}{90\ MHz} = 46.6ms$$

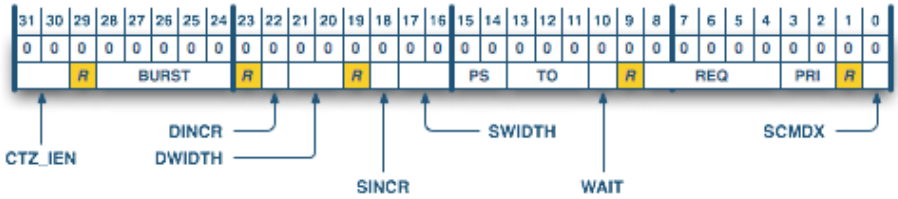
The following registers and fields control channel time-outs.

- DMA_CTRL → IENx
- DMA_CFGN → PS, TO, WAIT
- DMA_STAN → TO

8.12.2 Per-Channel Registers

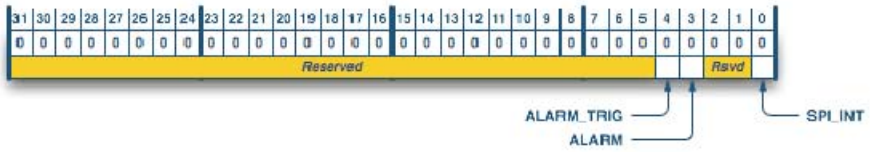
Offsets								Register
Ch0	Ch1	Ch2	Ch3	Ch4	Ch4	Ch6	Ch7	
100h	120h	140h	160h	180h	1A0h	1C0h	1E0h	DMA_CFGN
104h	124h	144h	164h	184h	1A4h	1C4h	1E4h	DMA_STAN
108h	128h	148h	168h	188h	1A8h	1C8h	1E8h	DMA_SRCN
10Ch	12Ch	14Ch	16Ch	18Ch	1ACh	1CCh	1ECh	DMA_DESTN
110h	130h	150h	170h	190h	1B0h	1D0h	1F0h	DMA_CNTN
114h	134h	154h	174h	194h	1B4h	1D4h	1F4h	DMA_SRLDN
118h	138h	158h	178h	198h	1B8h	1D8h	1F8h	DMA_DRLDN
11Ch	13Ch	15Ch	17Ch	19Ch	1BCh	1DCh	1FCh	DMA_CRLDN

8.12.2.1 DMA_CFGn – DMA Channel "n" Config Register



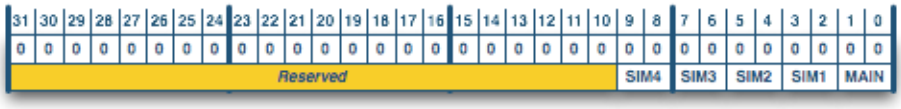
Bits	Type	Reset	Description
31	RW	0	**Count-to-Zero Interrupt Enable (CTZ_IEN):** When set, the IPEND goes active whenever a Count-to-Zero event occurs.
30	RW	0	**Status Interrupt Enable (STA_IEN):** When set, IPEND will go active whenever DMA_STAN.EN transitions from 1 to 0.
29	*RO*	*0*	*Reserved*
28:24	RW	00000	**Burst Size (BURST):** The number of bytes to be transferred into and out of the DMA FIFO in handling a single burst. • 00000: 1 byte • 00001: 2 bytes • ... • 11111: 32 bytes
23	*RO*	*0*	*Reserved*
22	RW	0	**Destination Increment Enable (DINCR):** When set, enables incrementing of DMA_DESTN on every AHB transaction. Forced to 0 for DMA transmit to peripherals.
21:20	RW	00	**Destination Width (DWIDTH):** Indicates the width of the each AHB transaction to the destination peripheral or memory. • 00: byte • 01: half-word • 10: word • 11: reserved
19	*RO*	*0*	*Reserved*
18	RW	0	**Source Increment Enable (SINCR):** When set, enables incrementing of DMA_SRCN upon every AHB transaction. Forced to 0 for DMA receive from peripherals.

10.3.1.2 Offset 004h: SC_IMASK – Interrupt Mask



Bits	Type	Reset	Description
31:05	*RO*	*0*	*Reserved*
04	RW	0	**Alarm Trigger Mask (ALARM_TRIG):** When set, mask the interrupt when there is a transition from High to Low on the SC_nALARM input.
03	RW	0	**Alarm Mask (ALARM):** Masks SC_nALARM when this bit is set.
02:01	*RO*	*0*	*Reserved*
00	RW	0	**SPI Interrupt Mask (SPI_INT):** When set, mask the interrupt from SPI.

10.3.1.3 Offset 008h: SC_VCC_CFG – Smart Cards VCC Configuration

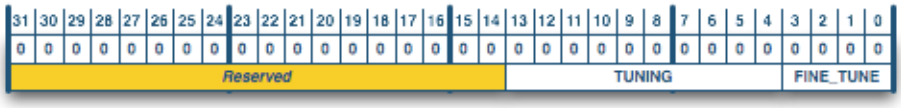


Bits	Type	Reset	Description
31:10	*RO*	*0*	*Reserved*
09:08	RW	00	**SC4 VCC configuration (SIM4):** Configures VCC voltage • 00 = 0 V • 01 = 1.8 V • 10 = 3 V • 11 = 5 V
07:06	RW	00	**SIM3 VCC configuration (SIM3):** Configures VCC voltage. Same encodings as SIM4.
05:04	RW	00	**SIM2 VCC configuration (SIM2):** Configures VCC voltage. Same encodings as SIM4.
03:02	RW	00	**SIM1 VCC configuration (SIM1):** Configures VCC voltage. Same encodings as SIM4.
01:00	RW	00	**Main VCC configuration (MAIN):** Configures VCC voltage. Same encodings as SIM4.

10.3.2.5 Offset 0Ch: SC_LCR – Line Control

Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07	RW	0	**Data Convention for RX Data (CNV):** When set, inverse convention. When cleared, normal convention.
06	RW	0	**Repetitions Interrupt (REP_INT):** When set, enables interrupt when the number of repetitions is reached.
05	RW	0	**Sticky Parity (STKY_PAR):** When set, sticky parity. When cleared, non-sticky parity.
04	RW	0	**Parity Select (PAR_SEL):** When set, even parity. When cleared, odd parity.
03	RW	0	**Parity Enable (PAR_EN):** When set, parity bit is generated or checked between the last data word bit and Stop bit of the serial data.
02	RW	0	**Stop Bits (STOP_BITS):** When set, enables the guard time (stop bit). The minimum is 1 stop bit before sending a new character. The receiver checks the first Stop bit only, regardless of the number of Stop bits selected. When cleared, one stop bit only.
01:00	RW	0	**Character Length (CHAR_LEN):** Specifies number of bits transmitted and received in each serial character. • 00: 5 characters • 01: 6 characters • 10: 7 characters • 11: 8 characters

10.3.2.6 Offset 10h: SC_BRR – Baud Rate Register



This holds the factor for dividing the Smart Card clock to produce an ETU timing reference. The formula is:

$$1\,ETU = 10 \times BRR_{TUNING} + BRR_{FINE\_TUNE}$$

Bits	Type	Reset	Description
31:14	*RO*	*0*	*Reserved*
13:04	RW	000h	**Tuning (TUNING):** Tuning value for ETU timing reference.
3:0	RW	0h	**Fine Tune (FINE_TUNE):** Fine tune adder for ETU reference value 0h to 9h.

10.3.2.9 Offset 1Ch: SC_GTIME – Guard Time

Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07:00	RW	00h	**Guard Time (GUARD_TIME):** Provides the Guard Time value in ETUs. • 0x00: 11 ETU (1 stop bit) • 0x01: 12 ETU • 0x02: 13 ETU • ... • 0xFF: 266 ETU (256 stop bits)

10.3.2.10 Offset 20h: SC_NUM_REP – Number of Repetition

Bits	Type	Reset	Description
31:04	*RO*	*0*	*Reserved*
03:00	RW	0h	**Number of repeated Parity Errors allowed in TX (RX?) Register (NUM_REP):** Holds the maximum number of repetitions in case of receive parity errors. • 0x0: 0 repetitions • 0x1: 1 repetitions • ... • 0xF: 15 repetitions

10.3.2.11 Offset 24h: SC_REV_DLY – Reverse Delay

Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07:00	RW	00h	**Number of Delay (DELAY_TIME):** • 0x00: 0 delay • 0x01: 1 delay • ... • 0xFF: 255 delay

11.8.1.3 Offset 008h: RTC_HRS – Current Hours

Bits	Type	Reset	Description
31:05	*RO*	*0*	*Reserved*
04:00	RW	Undef	**Value (VAL):** Stores the current hours count. Maximum value is 23.

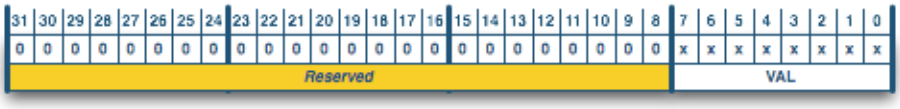
11.8.1.4 Offset 00Ch: RTC_DOM – Current Day-of-the-Month

Bits	Type	Reset	Description
31:05	*RO*	*0*	*Reserved*
04:00	RW	Undef	**Value (VAL):** Stores the current day-of-the-month count, including leap year calculations. The counter does not correctly exclude leap days for the years 1900, 2100, i.e. every 200th year. 1 = 1st day, 2 = 2nd day, etc.

11.8.1.5 Offset 010h: RTC_MON – Current Month

Bits	Type	Reset	Description
31:04	*RO*	*0*	*Reserved*
03:00	RW	Undef	**Value (VAL):** Stores the current month count. 0 = N/A. 1 = January, 12 = December

11.8.1.6 Offset 014h: RTC_YR – Current Year



Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07:00	RW	Undef	**Value (VAL):** Stores the current year count. The current year is 1900+YEAR. Valid for 1900–2155. Example: 0x65 = 2001

Chapter 13: SHA-1

The SHA-1 module provides a message digest for given text of almost any length per the SHA-1 protocol.

13.1 Programming Guide

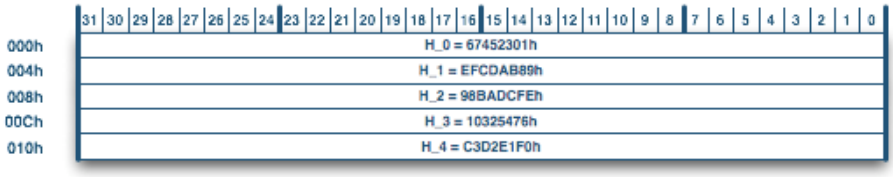
To generate a hashed signature value, execute the following actions:

1. If the existing hash value is acceptable, set SHA1_CONTROL.INIT to '1'. Alternatively, write SHA1_WH with a hash value, and set SHA1_CONTROL.INIT and SHA1_CONTROL.INIT_SEL to '1'.
2. Write a value to SHA1_DATA_IN.DATA. Sixteen writes to W_IN must be performed to provide data to the SHA-1. DMA can be used to provide these values.
3. Wait for 65 clocks and read SHA1_STATUS.VALID. Once this bit is set, SHA1_H contains the output. SHA1_STATUS.VALID remains set until all 5 registers are read.

13.2 SHA-1 Registers (Base → FFFF9000h)

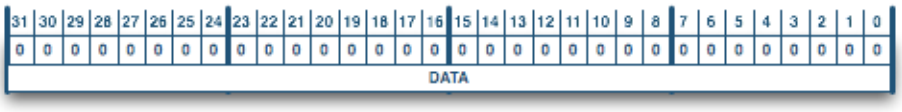
Address	Register	Description
000h – 010h	SHA1_H	Hashed value 0-4 from SHA-1
014h	SHA1_DATA_IN	Data IN register
018h	SHA1_CONTROL	SHA-1 Control
01Ch	SHA1_STATUS	SHA-1 Status Control
020h – 030h	SHA1_WH	Initial hash value 0-4

13.2.1 Offset 000h: SHA1_H – Hashed Value



Bits	Type	Reset	Description
159:127	RO	C3D2E1F0h	**H-4 (H_4):** This value is valid when STATUS.VALID is '1'.
127:96	RO	10325476h	**H-3 (H_3):** This value is valid when STATUS.VALID is '1'.
95:64	RO	98BADCFEh	**H-2 (H_2):** This value is valid when STATUS.VALID is '1'.
63:32	RO	EFCDAB89h	**H-1 (H_1):** This value is valid when STATUS.VALID is '1'.
31:00	RO	67452301h	**H-0 (H_0):** This value is valid when STATUS.VALID is '1'.

13.2.2 Offset 014h: SHA1_DATA_IN – Data In



Bits	Type	Reset	Description
31:00	WO	00000000h	**Data (DATA):** 32-bit value for SHA-1.

14.5 Registers (Base → FFFF2000h)

Offset	Register	Description
000h	ADC_CFG	ADC Configuration Register
004h	ADC_CMD	ADC Command Register
008h	ADC_FIFO	ADC FIFO Register
00Ch	ADC_INT	ADC Interrupt Register
010h	ADC_STA	ADC Status Register

14.5.1 Offset 000h: ADC_CFG – ADC Configuration Register

Bits	Type	Reset	Description
31:29	RW	000	**H-Channel Select (HCH_SEL):** • 000: N/A • 001: ADC_IN[1] pin • 010: ADC_IN[2] pin • 011: ADC_IN[3] pin • 100: ADC_IN[4] pin • 101 - 111: N/A
28:26	RW	000	**G-Channel Select (GCH_SEL):** Same encoding as HCH_SEL
25:23	RW	000	**F-Channel Select (FCH_SEL):** Same encoding as HCH_SEL
22:20	RW	000	**E-Channel Select (ECH_SEL):** Same encoding as HCH_SEL
19:17	RW	000	**D-Channel Select (DCH_SEL):** Same encoding as HCH_SEL
16:14	RW	000	**C-Channel Select (CCH_SEL):** Same encoding as HCH_SEL
13:11	RW	000	**B-Channel Select (BCH_SEL):** Same encoding as HCH_SEL
10:08	RW	000	**A-Channel Select (ACH_SEL):** Same encoding as HCH_SEL
07:00	RW	00h	**ADC Clock Divider (ADC_CLK_DIV):** Value used to derive the ADC clock from the ADC hclk. This clock is driven into the analog section of the ADC and must be configured to ensure that the divided clock is less than 600 kHz. The sample rate is 1/12 of the divided clock frequency. • 0: No clock to ADC (analog portion) • 1: ADC Divided Clock = hclk / 2 • 2: ADC Divided Clock = hclk / 3 • … • 255: ADC Divided Clock = hclk / 256

16.2.1.6 Compare Mode

In Compare mode, the timer counts up to the 16-bit maximum Compare value stored in the Timer Compare register. The timer input is the hclk. Upon reaching the Compare value, the timer generates an interrupt and counting continues (the timer value is not reset to 0001H). Also, if the Timer Output function is enabled (in the GPIO), the Timer Output pin changes state (from Low to High or from High to Low) upon Compare. When the Timer reaches FFFFh, the timer rolls over to 0000H and continues counting.

The steps for configuring a timer for Compare mode and initiating the count are as follows:

1. Clear TxCTL.TEN, Write TxCTL.TMODE to “101”, and set TxCTL.PRES. If using the timer output function, set the initial output level via TPOL:
2. Write to the Timer register to set the starting count value.
3. Write to the Timer Compare register to set the Compare value.
4. If desired, enable the timer interrupt (via the Interrupt Mask Register in the Interrupt Controller) and set the timer interrupt priority (by writing to the appropriate Configuration Table Register in the Interrupt Controller herein).
5. If using the Timer Output function, configure the associated GPIO port pin for the Timer Output function. (See chapter covering GPIO herein.)
6. Write to the Timer Control register to enable the timer and initiate counting (TEN = “1”).

In Compare mode, the hclk always provides the timer input. The Compare time is given by the following equation:

$$Compre\ Mode\ Time(s) = \frac{(Compare\ Value - Start\ Value) \times Prescale}{System\ Clock\ Frequency\ (Hz)}$$

16.2.1.7 Gated Mode

In Gated mode, the timer counts only when the Timer Input signal is in its active state (asserted), as determined by the TPOL bit in the Timer Control register. When the Timer Input signal is asserted, counting begins. A timer interrupt is generated when the Timer Input signal is de-asserted or a Timer Compare occurs.

The timer counts up to the 16-bit Compare value stored in the Timer Compare register. The timer input is the hclk. When reaching the Compare value, the timer generates an interrupt, the count value in the Timer register is reset to 0001H and counting resumes (assuming the Timer Input signal is still asserted). Also, if the Timer Output function is enabled (in the GPIO), the Timer Output pin changes state (from Low to High or from High to Low) at Timer Compare.

The steps for configuring a timer for Gated mode and initiating the count are as follows:

1. Clear TxCTL.TEN, Write TxCTL.TMODE to “110”, and set TxCTL.PRES:
2. Write to the Timer register to set the starting count value. This only affects the first pass in Gated mode. After the first timer reset in Gated mode, counting always begins at the reset value of 0001H.
3. Write to the Timer Compare register to set the Compare value.
4. If desired, enable the timer interrupt (via the Interrupt Mask Register in the Interrupt Controller) and set the timer interrupt priority (by writing to the appropriate Configuration Table Register in the Interrupt Controller herein).
5. Configure the associated GPIO port pin for the Timer Input function.

If using the Timer Output function, configure the associated GPIO port pin for the Timer Output function. (See chapter covering GPIO herein.)

6. Write to the Timer Control register to enable the timer (TEN = “1”).
7. Assert the Timer Input signal to initiate the counting (High if TPOL = “0”; Low if TPOL = “1”).

16.3.2.3 Counter/Cascade Mode

In this mode, the timer counts time outs on the cascade input that comes from the previous timer. Timers 4 through 8 are sequentially tied from cascade output to cascade input and Timer 8's cascade output ties to Timer 4 cascade input. Any number of timers can be put in the cascade chain and the first timer can be any one of the 5 timers. The first timer in the chain can be set up to be in One-Shot, Continuous or Compare mode to have an output occur for the next cascade input. The rest of the timers in the chain must be in Counter/Cascade mode. In this mode, the pre-scalar is ignored.

For the timers using this mode in the chain, upon reaching TxR, the timer generates an interrupt, the count value in Tx is reset to 00000001h and counting resumes. To use the full 32-bits for the timer, 00000000h should be written to Tx. The steps for configuring a timer for Counter mode and initiating the count are as follows:

1. Clear TxCTL.TEN and write TxCTL.TMODE to "010".
2. Write Tx to set the starting count value. This affects the first pass. After reaching TxR, counting begins at the reset value of 00000001h.
3. Write TxR.
4. If desired, enable the timer interrupt (via the Interrupt Mask Register in the Interrupt Controller) and set the timer interrupt priority (by writing to the appropriate Configuration Table Register in the Interrupt Controller herein).
5. Set TxCTL.TEN.

The number of Cascade Input transitions since the timer start is given by the following equation:

$$Counter\ Mode\ Timer\ Input\ Transitions = Current\ Count\ Value - Start\ Value$$

16.3.2.4 Compare Mode

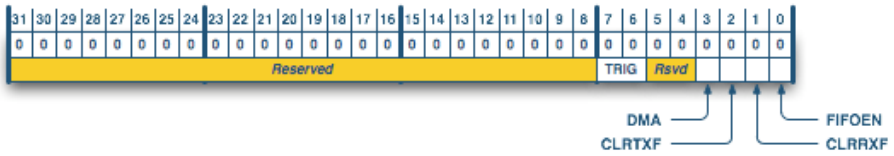
In this mode, Tx counts to TxR via hclk. Upon reaching TxR, an interrupt is generated, Tx is reset to 00000001h, and counting resumes. When the Timer reaches FFFFFFFFh, the timer rolls to 00000000h and continues counting. The steps for configuring a timer for this mode are:

1. Clear TxCTL.TEN and write TxCTL.TMODE to "101", and write TxCTL.PRES.
2. Write Tx and TxR.
3. If desired, enable the timer interrupt (via the Interrupt Mask Register in the Interrupt Controller) and set the timer interrupt priority (by writing to the appropriate Configuration Table Register in the Interrupt Controller herein).
4. Set TxCTL.TEN

The Compare time is given by the following equation:

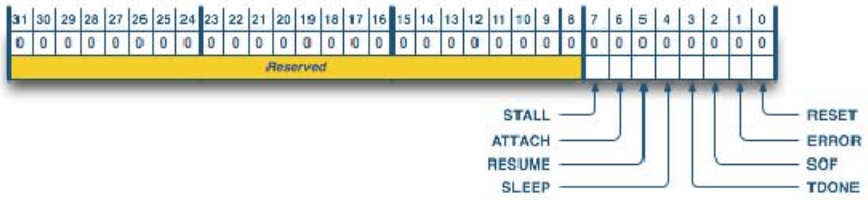
$$Compare\ Mode\ Time\ (s) = \frac{(Compare\ Value - Start\ Value) \times Prescale}{System\ Clock\ Frequency\ (Hz)}$$

17.5.2.5 Offset 008h: UARTx_FCR – UART FIFO Control Register



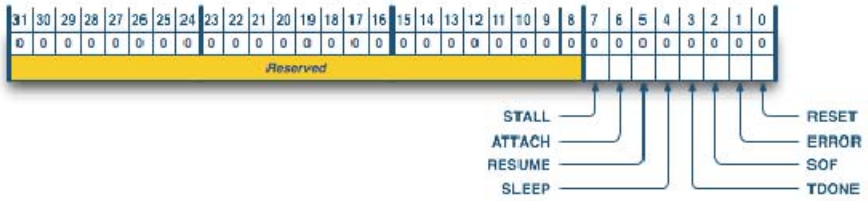
Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07:06	WO	00	**Trigger Level (TRIG):** See below: • 00 = Receive data interrupt/DMA request is generated when there is 1 byte in the receive FIFO. Transmit DMA request is generated when there is 1 or fewer bytes in the transmit FIFO. Valid only if FIFO is enabled. • 01 = Receive data interrupt/DMA request is generated when there are 4 bytes in the FIFO. Transmit DMA request is generated when there are 4 or fewer bytes in the transmit FIFO. Valid only if FIFO is enabled. • 10 = Receive data interrupt/DMA request is generated when there are 8 bytes in the FIFO. Transmit DMA request is generated when there are 8 or fewer bytes in the transmit FIFO. Valid only if FIFO is enabled. • 11 = Receive data interrupt/DMA request is generated when there are 14 bytes in the FIFO. Transmit DMA request is generated when there are 14 or fewer bytes in the transmit FIFO. Valid only if FIFO is enabled.
05:04	*RO*	*0*	*Reserved*
03	WO	0	**DMA Enable (DMA):** When set, DMA is enabled.
02	WO	0	**Clear Transmit FIFO (CLRTXF):** Writes of '1' clear the transmit FIFO and resets the transmit FIFO pointers. Valid only if the FIFOEN is set. Writes of '0' have no effect.
01	WO	0	**Clear Receive FIFO (CLRRXF):** Writes of '1' clear the receive FIFO, clear the receive error FIFO, and resets the receive FIFO pointers. Valid only if FIFOEN is set. Writes of '0' have no effect.
00	WO	0	**FIFO Enable (FIFOEN):** 0 = Transmit and receive FIFOs are disabled. Transmit and receive buffers are only 1 byte deep. 1 = Transmit and receive FIFOs are enabled. The receive FIFO will not be enabled during Multi-drop Mode (UARTx_MCR.MDM is set).

19.8.9 Offset 080h: USB_ISTAT – Interrupt Status Register



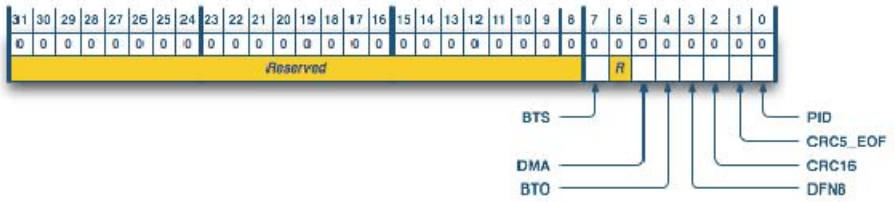
Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07	RW1C	0	**Stall (STALL):** In device mode, set when a STALL ACK sent by the controller. In host mode, set when controller detects a STALL ACK during handshake of a USB transaction.
06	RW1C	0	**Attach (ATTACH):** Only valid in host mode. Set when there have been no transitions on USB for 2.5µs and current bus state is not SE0, indicating attachment of a USB peripheral.
05	RW	0	**Resume (RESUME):** Set when a K-state is observed on D+/D- for 2.5usec.
04	RW	0	**Sleep (SLEEP):** This bit is set if the controller has detected a constant idle on the bus for 3 ms. The sleep timer is reset by activity on the bus.
03	RW1C	0	**Token Done (TDONE):** Set when the current token is complete. Clearing this bit causes USB_STAT to be cleared or the STAT holding register to be loaded into USB_STAT.
02	RW	0	**Start of Frame Token (SOF):** In device mode, this bit is set if the controller has received an SOF token. In host mode this bit is set when the SOF threshold is reached.
01	RW	0	**Error (ERROR):** Set when any error condition in USB_ESTAT occurred.
00	RW	0	**Reset (RESET):** Set when the controller detects a USB reset for 2.5 µs. It is not asserted again until the USB reset condition has been removed, and then re-asserted.

19.8.10 Offset 084h: USB_IEN – Interrupt Enable Register



Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07	RW	0	**Stall (STALL):** Enables USB_ISTAT.STALL to generate an interrupt.
06	RW	0	**Attach (ATTACH):** Enables USB_ISTAT.ATTACH to generate an interrupt.
05	RW	0	**Resume (RESUME):** Enables USB_ISTAT.RESUME to generate an interrupt.
04	RW	0	**Sleep (SLEEP):** Enables USB_ISTAT.SLEEP to generate an interrupt.
03	RW	0	**Token Done (TDONE):** Enabled USB_ISTAT.TDONE to generate an interrupt.
02	RW	0	**Start of Frame Token (SOF):** Enables USB_ISTAT.SOF to generate an interrupt.
01	RW	0	**Error (ERROR):** Enables USB_ISTAT.ERROR to generate an interrupt.
00	RW	0	**Reset (RESET):** Enables USB_ISTAT.RESET to generate an interrupt.

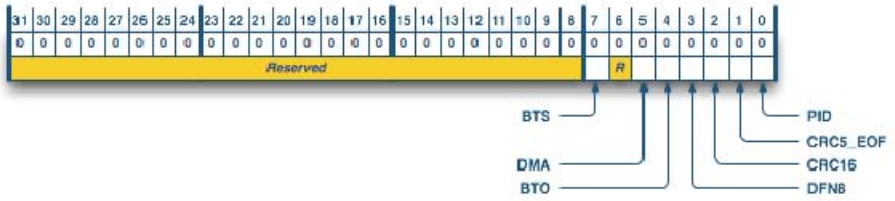
19.8.11 Offset 088h: USB_ESTAT – Error Interrupt Status Register



Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07	RW	0	**Bit Stuff (BTS):** When set, a bit stuff error detected, and the packet rejected.
06	*RO*	*0*	*Reserved*
05	RW	0	**DMA (DMA):** Occurs if there is a transmit data underflow, a receive data overflow, or if a data packet is larger than the buffer allocated in the BDT. In this last case data is truncated as it is put into memory.
04	RW	0	**Turnaround Timeout (BTO):** Set if more that 16-bit times are counted from the previous EOP before a transition from IDLE.
03	RW	0	**DFN8:** When set, the data field received was not 8-bits.
02	RW	0	**CRC16 (CRC16):** When set, the data packet was rejected due to a CRC16 error.
01	RW	0	**CRC5/EOF:** In device mode, indicates a CRC5 error in the token packet. In host mode, the controller is transmitting or receiving data and the SOF counter has expired.
00	RW	0	**PID (PID):** The PID check field failed.

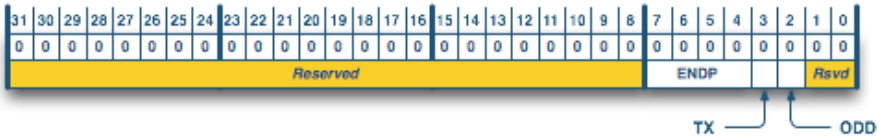
19.8.12 Offset 08Ch: USB_EEN – Error Interrupt Enable Register

Setting any of these bits enables the respective error interrupt source USB_ESTAT.



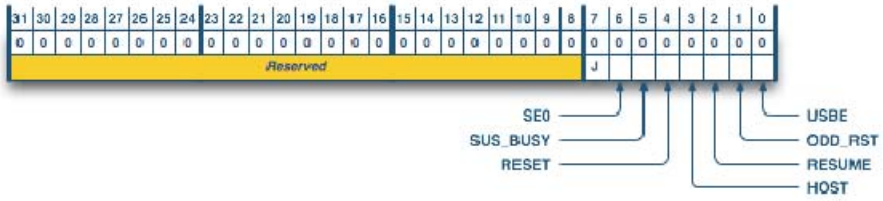
Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07	RW	0	**BTS Error (BTS):** Enables USB_ESTAT.BTS to generate an interrupt.
06	*RO*	*0*	*Reserved*
05	RW	0	**DMA (DMA):** Enables USB_ESTAT.DMA to generate an interrupt.
04	RW	0	**Turnaround Timeout (BTO):** Enables USB_ESTAT.BTO to generate an interrupt.
03	RW	0	**DFN8 (DFN8):** Enables USB_ESTAT.DFN8 to generate an interrupt.
02	RW	0	**CRC16 (CRC16):** Enables USB_ESTAT.CRC16 to generate an interrupt.
01	RW	0	**CRC5/EOF (CRC5_EOF):** Enables USB_ESTAT.CRC5_EOF to generate an interrupt.
00	RW	0	**PID (PID):** Enables USB_ESTAT.PID to generate an interrupt.

19.8.13 Offset 090h: USB_STAT – USB Status Register



Bits	Type	Reset	Description
31:08	RO	0	*Reserved*
07:04	RO	0	**Endpoint (ENDP):** Encodes the endpoint address that received or transmitted the previous token. This allows the CPU to determine which BDT entry was updated by the last USB transaction.
03	RO	0	**Transmit (TX):** When set, indicates the last BDT updated was for a transmit. When cleared, the last transaction was a receive data transfer.
02	RO	0	**Odd (ODD):** When set, last buffer descriptor update was in the odd bank of the BDT.
01:00	RO	0	*Reserved*

19.8.14 Offset 094h: USB_CTRL – USB Control Register



Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07	RW	0	**J-State (J):** USB differential receiver JSTATE. Polarity is affected by USB_ADDR.LS.
06	RW	0	**Single Ended Zero (SE0):** Live USB Single Ended Zero signal.
05	RW	0	**Transmit Suspend (device) / Token Busy (host) (SUS_BSY):** When set as a device, the controller has disabled packet transmission and reception. Clearing this bit allows the controller to continue token processing. This bit is set when a Setup Token is received. As a host, the controller is executing a USB token and no more token commands must be written.
04	RW	0	**Reset (RESET):** When set, the controller performs USB reset signaling. Only valid in host mode. Software must set this bit to '1' for the required amount of time and then clear it back to '0'. For more information on RESET signaling see the USB specification.
03	RW	0	**Host Enable (HOST):** When set, enables the controller to operate as a host.
02	RW	0	**Resume (RESUME):** When set, the controller executes resume signaling. Software must set this bit for the required amount of time and then clear it to 0. If HOST is set the controller appends a Low Speed End of Packet to the resume signaling when RESUME is cleared.
01	RW	0	**Even/Odd Reset (ODD_RST):** Setting this clears all BDT ODD ping/pong bits.
00	RW	0	**USB Enable (USBE):** Setting this bit enables the controller and resets all ODD bits to the BDT. If HOST is set, SOFs are generated by the controller.

20.4 Using Output

A GPIO pin can always be used as an input. The state of the pin can be sampled through GPIO_IN even when configured to operate as a primary function (GPIO_EN = ‘0’) or when configured as an output (GPIO_OE = ‘1’).

To be used as an output, GPIO_EN must be set. Once set, the pin can be forced to drive a value on the pin by setting GPIO_OE. With GPIO_OE=1, setting GPIO_OUT to 0 drive a ‘0’, while GPIO_OUT=1 drives a ‘1’. The pin can be tri-stated by clearing GPIO_OE. GPIO_EN, GPIO_OE, and GPIO_OUT can be modified either by writing to them directly or by writing to the associated SET and CLR registers. This allows bits to be changed without a Read-Modify-Write operation.

20.5 Configuring Interrupts

Below is the suggested sequence for the initial configuration of GPIO interrupts. This can be done for any number of GPIO bits. The GPIO Interrupt function can be used independent of the GPIO_EN and GPIO_OE settings.

1. Disable interrupts by clearing GPIO_IEN. This prevents the Interrupt Pending Flop from being set but does not mask an already pending interrupt in the flop. See JK-flop labeled Interrupt Pending Flop in Figure 20-1. GPIO_IEN can also be cleared by writing to GPIO_IEN_CLR.
2. Clear any pending interrupts by writing to GPIO_ICLR. This clears the Interrupt Pending Flop.
3. Write to GPIO_IMOD to select level/edge mode.
4. Write to GPIO_IPOL to select either rising/falling trigger level.
5. Write to GPIO_ISEL to select either A/B channel. This provides two priority levels within the interrupt controller.
6. Write to GPIO_IEN to re-enable interrupts. Once set, the Interrupt Pending Flop can be set and interrupts are active and ready. GPIO_IEN can also be set by writing to GPIO_IEN_SET.
7. Configure the Interrupt Controller as given section Chapter 6:. You may want to do this last step prior to setting the GPIO_IEN bits.

20.6 Handling Interrupts

The recommended sequence of GPIO interrupt handling is provided below:

1. Based upon GPIO_ISEL, read either GPIO_IAST or GPIO_IBST to determine the source(s).
2. Perform application specific interrupt tasks associated with the bit(s).
3. Clear the Interrupt Pending Flop by writing to GPIO_ICLR. This also clears and re-arms the rising and falling edge trigger flops.
4. Signal and End-of-Interrupt to the interrupt controller. See section Chapter 6: for more details
5. Return from the ISR.

20.7 Wake Function

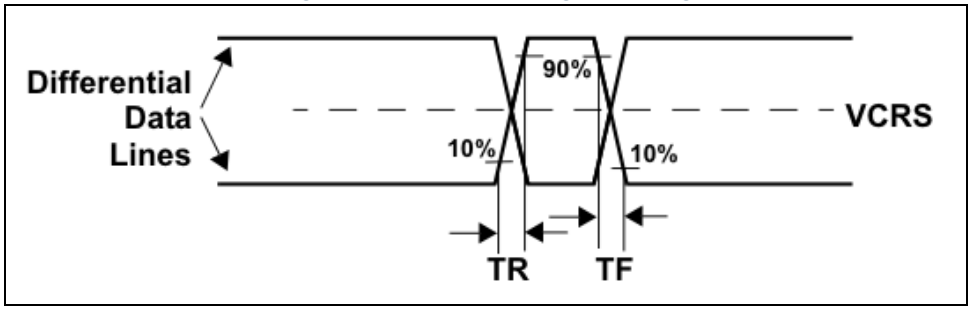
Below are suggested steps to enable wake. Like interrupts, wake is independent of GPIO_EN and GPIO_OE:

1. Set the polarity (rising or falling edge) by writing to GPIO_IPOL. The Wake function relies upon rising and falling edge detectors to detect and latch the transition on the GPIO pin. These flops operate asynchronously and do not require the GPIO clock to be active.
2. Clear the edge detector flops by writing to the GPIO_ICLR register.
3. Activate the GPIO Wake function by writing to the GPIO_WKEN or GPIO_WKEN_SET register.
4. Configure the PMU to wake upon GPIO per section Chapter 3:.

21.6 USB Electrical and Timing

The Z32AN Series conforms to the Universal Serial Bus 2.0 standard specification.

Figure 21-3: USB Data Signal Timing



Symbol	Parameter	T_A = –40°C to 85°C		Units	Conditions
		Min	Max		
I_{CCUSB}	Supply Currents in STOP with USB Suspend	—	—	µA	No I/O switching, all blocks and oscillator disabled. USB Transceivers configured to wake with activity.
t_R	Time for data to rise from 10% to 90% of rail values	4.0	20.0	ns	USB Full speed
t_F	Time for data to fall from 90% to 10% of rail values	4.0	20.0	ns	USB Full speed