**21.3.15 How to Program I2C**

**21.3.15.1 Module Configuration Before Enabling the Module**

1. Program the prescaler to obtain an approximately 12-MHz I2C module clock (I2C\_PSC = x; this value is to be calculated and is dependent on the System clock frequency).

2. Program the I2C clock to obtain 100 Kbps or 400 Kbps (SCLL = x and SCLH = x; these values are to be calculated and are dependent on the System clock frequency).

3. Configure its own address (I2C\_OA = x) - only in case of I2C operating mode (F/S mode).

4. Take the I2C module out of reset (I2C\_CON:I2C\_EN = 1).

**21.3.15.2 Initialization Procedure**

1. Configure the I2C mode register (I2C\_CON) bits.

2. Enable interrupt masks (I2C\_IRQENABLE\_SET), if using interrupt for transmit/receive data.

3. Enable the DMA (I2C\_BUF and I2C\_DMA/RX/TX/ENABLE\_SET) and program the DMA controller) - only in case of I2C operating mode (F/S mode), if using DMA for transmit/receive data.

**21.3.15.3 Configure Slave Address and DATA Counter Registers**

In master mode, configure the slave address (I2C\_SA = x) and the number of byte associated with the transfer (I2C\_CNT = x).

**21.3.15.4 Initiate a Transfer**

Poll the bus busy (BB) bit in the I2C status register (I2C\_IRQSTATUS\_RAW). If it is cleared to 0 (bus not busy), configure START/STOP (I2C\_CON: STT / I2C\_CON: STP condition to initiate a transfer) - only in case of I2C operating mode (F/S mode).

**21.3.15.5 Receive Data**

Poll the receive data ready interrupt flag bit (RRDY) in the I2C status register (I2C\_IRQSTATUS\_RAW), use the RRDY interrupt (I2C\_IRQENABLE\_SET.RRDY\_IE set) or use the DMA RX (I2C\_BUF.RDMA\_EN set together with I2C\_DMARXENABLE\_SET) to read the received data in the data receive register (I2C\_DATA). Use draining feature (I2C\_IRQSTATUS\_RAW.RDR enabled by I2C\_IRQENABLE\_SET.RDR\_IE)) if the transfer length is not equal with FIFO threshold.

**21.3.15.6 Transmit Data**

Poll the transmit data ready interrupt flag bit (XRDY) in the I2C status register (I2C\_IRQSTATUS\_RAW), use the XRDY interrupt (I2C\_IRQENABLE\_SET.XRDY\_IE set) or use the DMA TX (I2C\_BUF.XDMA\_EN) set together with I2C\_DMATXENABLE\_SET) to write data into the data transmit register (I2C\_DATA). Use draining feature (I2C\_IRQSTATUS\_RAW.XDR enabled by I2C\_IRQENABLE\_SET.XDR\_IE)) if the transfer length is not equal with FIFO threshold.

**21.3.16 I2C Behavior During Emulation**

To configure the I2C to stop during emulation suspend events (for example, debugger breakpoints), set up the I2C and the Debug Subsystem:

1. Set I2C\_SYSTEST.FREE=0. This will allow the Suspend\_Control signal from the Debug Subsystem (Chapter 27) to stop and start the I2C. Note that if FREE=1, the Suspend\_Control signal is ignored and the I2C is free running regardless of any debug suspend event. This FREE bit gives local control from a module perspective to gate the suspend signal coming from the Debug Subsystem.

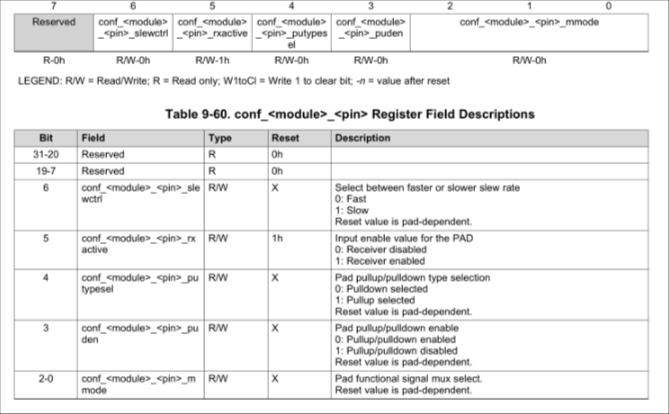
2. Set the appropriate xxx\_Suspend\_Control register = 0x9, as described in Section 27.1.1.1, Debug Suspend Support for Peripherals. Choose the register appropriate to the peripheral you want to suspend during a suspend event.

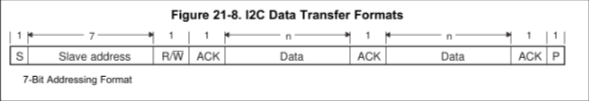
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| T | r | a | v | i | s | \*space\* |
| 0x54 | 0x72 | 0x61 | 0x76 | 0x69 | 0x73 | 0x20 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| H | e | r | m | a | n | t |
| 0x48 | 0x65 | 0x72 | 0x6D | 0x61 | 0x6E | 0x74 |

Pin 17 and 18 moved to Mode 2 for 12C1\_SCL and I2C1\_SDA, respectively.

conf\_spi0\_d1 at offset 0x958 for pin 17, conf\_spi0\_cs0 at offset 0x95C. Base at 0x44E10000 for control module registers. Write 0x2 (0x6A, 0110 1010) to both of these registers to switch their modes.





tLOW, tHIGH = 1/((100 Kbps) \* 2) = 5us iCLK = 1/(12MHz) = 83.3 ns

SCLL = tLOW/ICLK – 7 = 53 = 0x35

SCLH = tHIGH/ICLK – 5 = 55 = 0x37

Interrupt number 71

|  |  |  |
| --- | --- | --- |
| I2C1 Base Address: 0x4802A000 | | |
|  |  |  |
| I2C\_IRQSTATUS\_RAW | 0x24 |  |
| I2C\_IRQENABLE | 0x2C | Write 0x7FFF to enable all interrupts |
| I2C\_CNT | 0x98 | Write 0x0A for sending the instructions, write 0x11 for sending the message |
| I2C\_DATA | 0x9C | Load character or instruction to be sent |
| I2C\_OA | 0xA8 | Write 0x00 |
| I2C\_CON | 0xA4 | Write 0x8601 to start, write 0x8602 to stop |
| I2C\_SA | 0xAC | Write 0x3C (slave address) |
| I2C\_PSC | 0xB0 | Write 0x03, divide 48MHz by 4 to get 12MHz |
| I2C\_SCLL | 0xB4 | Write 0x35, set 100kbps |
| I2C\_SCLH | 0xB8 | Write 0x37, set 100kbps |
| I2C\_SYSC | 0x10 | Write 0x02 to reset |

|  |  |
| --- | --- |
| I2C1 CLOCK CONTROL --- CM\_PER base address at 0x44E00000 | |
| CM\_PER\_I2C1\_CLKCTRL (0x48) | Write 0x2 to enable |

