

PARA BOLUDOS (Arturo)

LA DIRECCION DEL **SLAVE** Y EL **BIT R/W** NO VAN NI MAS NI MENOS QUE EN EL I2CDAT QUE SE TRANSMITE AL PRINCIPIO DE LA COMUNICACIÓN!!!

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| DAT7 | DAT6 | DAT5 | DAT4 | DAT3 | DAT2 | DAT1 | DAT0 |

|  |  |
| --- | --- |
| SLAVE ADRESS(bits 7:1) | R/W(bit 0) |

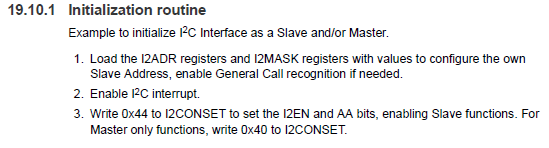
Registros del I2C

|  |
| --- |
| **//direcciones de los registros**  **#define** DIR\_I2C1SCLL ( ( uint32\_t \* ) 0x4005C014UL )  **#define** DIR\_I2C1SCLH ( ( uint32\_t \* ) 0x4005C010UL )  **#define** DIR\_I2C1STAT ( ( uint32\_t \* ) 0x4005C004UL )  **#define** DIR\_I2C1CONSET ( ( uint32\_t \* ) 0x4005C000UL )  **#define** DIR\_I2C1CONCLR ( ( uint32\_t \* ) 0x4005C018UL )  **#define** DIR\_I2C1DAT ( ( uint32\_t \* ) 0x4005C008UL )  **#define** DIR\_I2C1ADR ( ( uint32\_t \* ) 0x4001C020UL )  **#define** DIR\_I2C1MASK ( ( uint32\_t \* ) 0x4001C034UL )  //registros I2C1  **#define** I2C1SCLL DIR\_I2C1SCLL[0]  **#define** I2C1SCLH DIR\_I2C1SCLH[0]  **#define** I2C1STAT DIR\_I2C1STAT[0]  **#define** I2C1CONSET DIR\_I2C1CONSET[0]  **#define** I2C1CONCLR DIR\_I2C1CONCLR[0]  **#define** I2C1DAT DIR\_I2C1DAT[0]  **#define** I2ADR DIR\_I2C1ADR[0]  **#define** I2MASK DIR\_I2C1MASK[0] |

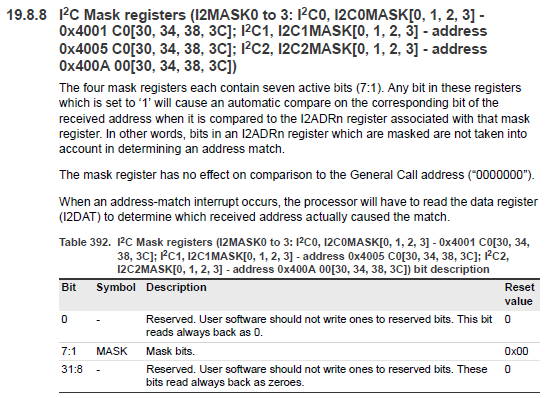
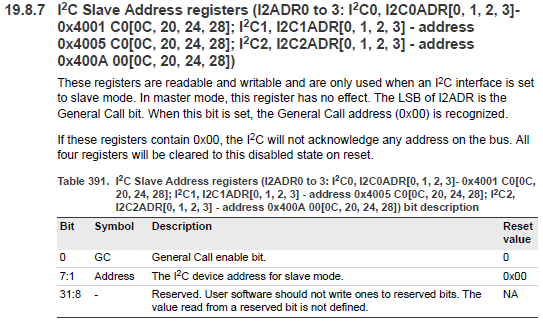
Inicialización del modulo I2C1

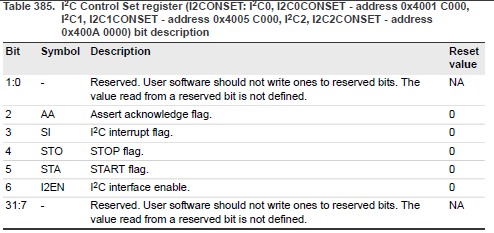
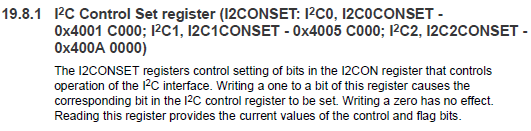
|  |
| --- |
| **void** **InitI2C1**( **void** )  {  //falta darle energía al modulo con el PCONP  //seteo como salida el Serial Data 1  FIO0DIR0 = ENTRADA; //sda1 P0.0  //seteo como salida el Serial Clock 1  FIO0DIR1 = SALIDA; //scl1 P0.1  //configuración de trabajo de SDA y SCL  P00\_PINSEL = PINSEL\_FUNC3;  P01\_PINSEL = PINSEL\_FUNC3;  P00\_PINMODE = 2; //no pull-up, no pull-down resistors  P01\_PINMODE = 2; //no pull-up, no pull-down resistors  P00\_OD = 1; //open drain  P01\_OD = 1; //open drain  //elijo el tiempo de trabajo del modulo, frecuencia de trabajo  PCLKSEL1 &= ~(0x03<<6); //PCLK\_peripheral = CCLK/4  I2C1SCLH = 0x7D; //scl1 100KHz, PCLKi2c1 = clk/4 = 25MHz  I2C1SCLL = 0x7D; // I2C1SCLH + I2C1SCLL = 250 -> 25MHz/250 = 100KHz  //habilito la comunicación I2C ONi2C1 => I2C1CONSET |= 0x40  //habilitas el modulo pero aun no le diste STARTi2C1 => I2C1CONSET |= 0x20  ONi2C1;  //habilito la interrupción en el NVIC I2C1\_IRQn = 11  NVIC\_HabilitarIRQ(*I2C1\_IRQn*);  } |

Inicialización del *Master Receiver Mode*



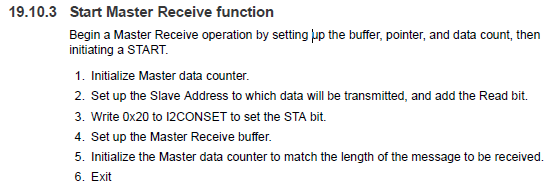
|  |
| --- |
| **//direcciones de MASTER & SLAVE**  **#define** I2CMASTER 0x01  **#define** I2CSLAVE 0x02  I2ADR |= (I2SLAVE<<1); //creo que no hace falta por no ser SLAVE  I2MASK |= (I2CSLAVE<<1);  NVIC\_HabilitarIRQ(*Acá iría el nro. correspondiente a la interrupción*);  I2CONSET &= ~0x40; //no sé si hace falta pero por las dudas  I2CONSET |= 0x40; |



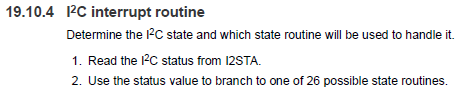
Estados del *Master*

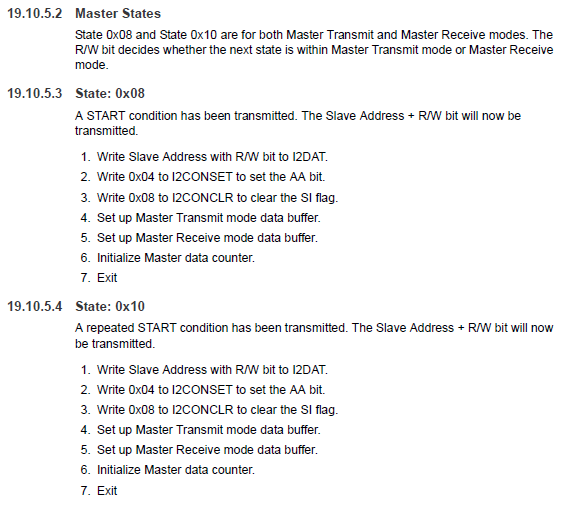
|  |
| --- |
| **#define** I2C\_IDLE 0  **#define** I2C\_STARTED 1  **#define** I2C\_RESTARTED 2  **#define** I2C\_REPEATED\_START 3  **#define** DATA\_ACK 4  **#define** DATA\_NACK 5 |

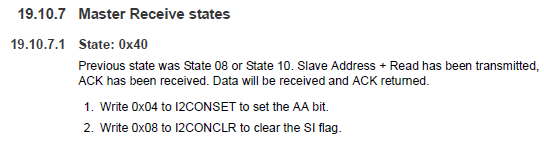


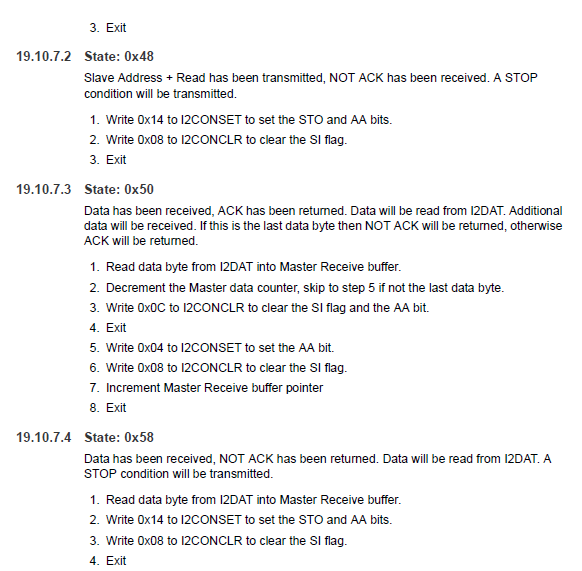
Aclaración: I2DAT mantiene el dato recibido mientras SI = 1 en el CONSET

|  |
| --- |
| //defino variables globales del main  //I2C  **#define** BUFSIZE 0x20  **#define** WRITE 0  **#define** READ 1  **volatile** uint32\_t I2CMasterState = I2C\_IDLE;  **volatile** uint32\_t I2CSlaveState = I2C\_IDLE;  **volatile** uint8\_t I2CMasterBuffer[BUFSIZE];  **volatile** uint8\_t I2CSlaveBuffer[BUFSIZE];  **volatile** uint32\_t I2CCount = 0;  **volatile** uint32\_t I2CReadLength;  **volatile** uint32\_t I2CWriteLength;  //no entiendo las inicializaciones del master counter, etc.  //no entiendo donde PUTA está el R/W bit |









Seteo de registros de acuerdo a códigos leídos del I2CSTAT (acompañar cada acción viendo las ultimas tablas del PDF de I2C recomendable)

Código sacado de prueba KIT\_INFOTRONIX para el I2C1, que por suerte vamos a requerir esas salidas *Serial Data y Serial Clock*.

|  |
| --- |
| **switch**( I2C1STAT )  {  //State: 0x00 Bus Error. Enter not addressed Slave mode and release bus.  **case** 0x00 :  //no se envio ninguna direccion y el buss esta libre  I2C1CONSET=0x14;  I2C1CONCLR=0x08;  **break**;  //State: 0x08  //A START condition has been transmitted. The Slave Address + R/W bit will now be transmitted.  **case** 0x08:  // se envio la condicion de START. COMIENZA LA COMUNICACION  //cargar\_direccion();  I2C1DAT=0xA0; //1010 A2 A1 A0 R/W  I2C1CONSET=0x04;  I2C1CONCLR=0X08;  **break**;  **case** 0x10:  //idem anterior, solo que indica que se envio de nuevo una condicion de START  //cargar\_direccion();  I2C1DAT=0xA0;  I2C1CONSET=0x04;  I2C1CONCLR=0X08;  **break**;  //State: 0x18  //Previous state was State 0x08 or State 0x10, Slave Address + Write has been transmitted,  //ACK has been received. The first data byte will be transmitted.  //1. Load I2DAT with first data byte from Master Transmit buffer.  //2. Write 0x04 to I2CONSET to set the AA bit.  //3. Write 0x08 to I2CONCLR to clear the SI flag.  //4. Increment Master Transmit buffer pointer.  //5. Exit  **case** 0x18:  I2C1DAT=0x7f;  I2C1CONSET=0x04;  I2C1CONCLR=0x08;  **break**;  //State: 0x28  //Data has been transmitted, ACK has been received. If the transmitted data was the last  //data byte then transmit a STOP condition, otherwise transmit the next data byte.  //1. Decrement the Master data counter, skip to step 5 if not the last data byte.  //2. Write 0x14 to I2CONSET to set the STO and AA bits.  //3. Write 0x08 to I2CONCLR to clear the SI flag.  //4. Exit  //5. Load I2DAT with next data byte from Master Transmit buffer.  //6. Write 0x04 to I2CONSET to set the AA bit.  //7. Write 0x08 to I2CONCLR to clear the SI flag.  //8. Increment Master Transmit buffer pointer  //9. Exit  **case** 0x28:  I2C1CONSET=0x14;  I2C1CONCLR=0x08;  //estado\_temp=ESCRITO;  **break**;  //State: 0x40 Previous state was State 08 or State 10. Slave Address + Read has been transmitted,  //ACK has been received. Data will be received and ACK returned.  **case** 0x40 :  I2C1CONSET=0x04;  I2C1CONCLR=0x08;  **break**;  //State:0x48 Slave Address+Read has been transmitted NOT ACK has been received A STOP condition will be transmitted  **case** 0x48:  I2C1CONSET=0x14;  I2C1CONCLR=0x08;  **break**;  // State: 0x50  //Data has been received, ACK has been returned. Data will be read from I2DAT. Additional  //data will be received. If this is the last data byte then NOT ACK will be returned, otherwise ACK will be returned.  //1. Read data byte from I2DAT into Master Receive buffer.  //2. Decrement the Master data counter, skip to step 5 if not the last data byte.  //3. Write 0x0C to I2CONCLR to clear the SI flag and the AA bit.  //4. Exit  //5. Write 0x04 to I2CONSET to set the AA bit.  //6. Write 0x08 to I2CONCLR to clear the SI flag.  //7. Increment Master Receive buffer pointer  //8. Exit  **case** 0x50:  I2C1CONCLR=0x0C;  I2C1CONSET=0x20; //envio una nueva condicion de start una vez que recibi el dato de esa direccion  **break**; |