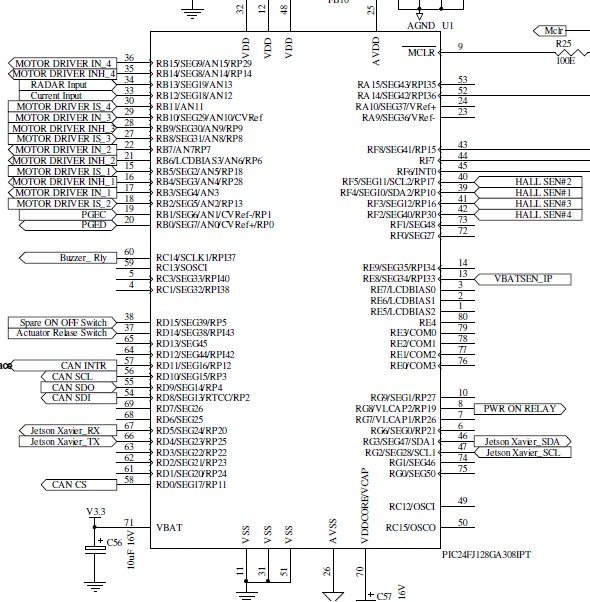
**Autonomous Tractor Safety System**

**Objective 8.0** To demonstrate I2C in ATSS PCB board.

****

**I2C1CON** I2C1 CONTROL REGISTER: 0x8200

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| I2CEN | - | I2CSIDL | SCLREL | IPMIEN | A10M | DISSLW | SMEN |
| I2C ENABLE |  | I2C1 IN IDLE MODE | SCL1 RELEASE CTRL | INTELLIGENT PLATFORM MGMT INTF EN | ADDRESS BIT | SLEW RATE CTRL | SMBUS I/P LVLS |
| 1 |  | 0 | 0 | 0 | 0 | 1 | 0 |
| ENABLED |  | CONTINUE | RESET | DISABLED | SEVEN | DISABLED | DISABLE THRESHOLDS |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| GCEN | STREN | ACKDT | ACKEN | RCEN | PEN | RSEN | SEN |
| GENERAL CALL EN | CLK STRETCH | ACK BIT | ACK EN | RX EN | STOP EN | REPEATED START EN | START EN |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| DISABLED | DISABLED | SEND ACK | ACK NOT IN PROGRESS | RX NOT IN PROGRESS | STOP NOT IN PROGRESS | REPEATED START NOT IN PROGRESS | START NOT IN PROGRESS |

**I2C1STAT** I2C1 STATUS REGISTER: 0x0000

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ACKSTAT | TRSTAT | - | - | - | BCL | GCSTAT | ADD10 |
| ACK STATUS | TX STATUS |  |  |  | MASTER BUS COLLISION DETECT | GEN CALL RX | 10 BIT ADDR MATCH |
| 0 | 0 |  |  |  | 0 | 0 | 0 |
| ACK WAS DETECTED LAST | MASTER TX NOT IN PROGRESS |  |  |  | NONE | NOT RX | NOT MATCHED |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| IWCOL | I2COV | D/A | P | S | R/W | RBF | TBF |
| WRITE COLLISION DETECT | OVERFLOW DETECT | RX DATA OR ADDR | STOP BIT | START BIT | READ/  WRITE | RX BUFFER FULL | TX BUFFER FULL |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| NONE | NONE | ADDR | NOT DETECTED | NOT DETECTED | WRITE TO SLAVE | EMPTY | EMPTY |

**Code Prg09.c** /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Source code file: Prg09.c

Author, version, date: PSS ver 1.0 01.11.24

Program function: I2C

Simulation: PIC24FJ128GA308 MCU, MPLAB X IDE ver 6.05, XC16 ver 2.10

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*

Read and Write 24LC16 Serial EEPROM (SEE) using I2C

Fscl = (Fcy / (I2C1BRG + 1 + (Fcy / 10000000))), Fcy = Fosc / 2 = 16 MHz, Fscl =

100 KHz, I2C1BRG = 157.4 = approx. 157 or 0x9D

\*/

#include <xc.h>

#include <libpic30.h>

//#include <i2c.h>

//#include <timer.h>

#define BUS\_FRQ 100000L // 100kHz

#define READ\_CMD 1

#define WRITE\_CMD 0

int i**,** r**;**

void initSEE **(**void**);**

int addressSEE **(**long add**);**

int iReadSEE **(**long address**);**// NB:Addr must be an even value between 0x0000 & 0x7FE

void iWriteSEE **(**long address**,** int data**);**

main **(**void**)**

**{**

initSEE**();**

**for(** i **=** 0**;** i **<** 1024**;** i**++)** // Test filling all the 16-bit integers

**{**

iWriteSEE **(**i **<<** 1**,** i**);**

r **=** iReadSEE **(**i **<<** 1**);**

**if** **(**r **!=** i**)** **break;**

**}**

**while(** 1**)**

**{**

asm**(**"nop"**);**

asm**(**"nop"**);**

asm**(**"nop"**);**

**}**

**}**

void initSEE**(** void**)** // Configure I2C for 7 bit address mode 100kHz

**{**

// OpenI2C1 (I2C\_ON | I2C\_IDLE\_CON | I2C\_7BIT\_ADD | I2C\_STR\_EN

// | I2C\_GCALL\_DIS | I2C\_SM\_DIS | I2C\_IPMI\_DIS,

// (FCY /(2\*BUS\_FRQ))-1);

I2C1CON = 0x8200;

I2C1STAT = 0x0000;

IdleI2C1**();**

// OpenTimer1(T1\_ON | T1\_SOURCE\_INT | T1\_PS\_1\_256, -1);

TMR1 **=** 0**;**

**while(**TMR1 **<** 100**);**

**}**

int addressSEE**(** long add**)** // Send the address selection command, repeat if SEE busy

**{**

int cmd**;**

cmd **=** 0xA0 **|** **((**add **>>** 7**)** **&** 0xE**);** // Form SEE command + address MSB(3)

**while(** 1**)** // WRITE() the Address MSB, try send command and repeat until ACK is

**{ //**  received, while waiting for ACK

StartI2C1**();**

IdleI2C1**();**

MasterWriteI2C1**(**cmd **+** WRITE\_CMD**);** // Send command and address MSB(3)

IdleI2C1**();**

**if** **(**I2C1STATbits**.**ACKSTAT **==** 0**)** **break;**

StopI2C1**();**

IdleI2C1**();**

**}**

MasterWriteI2C1**(**add**);** // Send address LSB

IdleI2C1**();**

**return** cmd**;**

**}**

int iReadSEE**(**long add**)** // Random access read command sequence

**{**

int cmd**,** r**;**

cmd **=** addressSEE**(**add**);** // Select address

StopI2C1**();**

IdleI2C1**();**

StartI2C1**();** // Read command

IdleI2C1**();**

MasterWriteI2C1**(**cmd **+** READ\_CMD**);**

IdleI2C1**();**

r **=** MasterReadI2C1**();** // Stream data in (will continue until NACK is sent)

AckI2C1**();**

IdleI2C1**();**

r **|=** **(**MasterReadI2C1**()** **<<** 8**);**

NotAckI2C1**();** // Terminate read sequence (send NACK then STOP)

IdleI2C1**();**

StopI2C1**();**

IdleI2C1**();**

**return** r**;**

**}**

void iWriteSEE**(**long add**,** int v**)** // SEE write command sequence

**{**

int cmd**;**

cmd **=** addressSEE**(**add**);** // Select address

MasterWriteI2C1**(**v **&** 0xFF**);** // Stream data out

IdleI2C1**();**

MasterWriteI2C1**(**v **>>** 8**);**

IdleI2C1**();**

StopI2C1**();** // Terminate the command sequence

IdleI2C1**();**

**}**