@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

@ ECE 372 – PROGRAMMING PROJECT III

@----------------------------------------------------------------------------

@ Project: Using I2C to send data from BBB to LCD

@ This project displays my name on I2C\_LCD ( New Haven 2x20 LCD display)

@----------------------------------------------------------------------------

@ Hai Dang Hoang

@ Reference: Douglas V. Hall and Leela Yadlapalli

@ Winter Term

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

**.text**

**.global** \_start

**\_start:**

@---------------------------------ENABLE CLOCK FOR I2C1-----------------------------@

LDR R0,=0x44E00048 @ Enable Clock for I2C1 (CM\_PER\_I2C1\_CLKCTRL)

MOV R1,#0x2 @ Store 2 to enable

STR R1,[R0]

@--------------------------------CONTROL MODULE I2C1--------------------------------@

LDR R8,=0x44e10958 @ use Pin 17 on P9 for SCL

MOV R9,#0x72 @ Slow slew rate/ Pull Up/ Receiver enable/ Mode 2

STR R9,[R8] @ Store

LDR R8,=0x44e1095C @ use Pin 18 on P9 for SDA

MOV R9,#0x72 @ Slow slew rate/ Pull Up/ Receiver enable/ Mode 2

STR R9,[R8] @ Store

@--------------------------------INITIAL I2C1---------------------------------------@

LDR R0,=0x4802A0A4 @ LOAD I2C\_CON Address

LDR R1,=0x0 @ Disable I2C\_EN by Storing 0

STR R1,[R0]

LDR R0,=0x4802A010 @ LOAD System Configuration Register

MOV R1,#0x0 @ Disable auto Idle

STR R1,[R0]

@ Configure Frequency = 100 KHz

@ Frequency = ICLK / ((SCLL + 7) + (SCLH + 5))

@ ICLK = 48 MHz / I2C\_PSC.PSC

@ Choose PSC = 4 --> ICLK = 48/4 = 12MHz

@ I2C\_SCLL.SCLL = I2C\_SCLH.SCLH ( Technical Reference Manual)

@ Solving SCLL = SCLH = 0x36

@ Setting PSC 3 for divide/4

LDR R0,=0x4802A0B0 @ Load I2C Clock Prescaler Register

MOV R1,#0x3 @ Divide 4

STR R1,[R0]

@ Setting SCLL and SCLH : 0x36

LDR R0,=0x4802A0B4 @ Load I2C\_SCLL Register

MOV R1,#0x36

STR R1,[R0]

LDR R0,=0x4802A0B8 @ Load I2C\_SCLH register

MOV R1,#0x36

STR R1,[R0]

LDR R0,=0x4802A0A8 @ LOAD Own Address

MOV R1,#0x1

STR R1,[R0]

LDR R0,=0x4802A094 @ Load Buffer Configuration Register

MOV R1,#0x0 @ RXTRSH/TXTRSH = 1

STR R1,[R0]

LDR R0,=0x4802A0A4 @ I2C Configuration Register

LDR R1,=0x8000 @ Enable I2C\_EN

STR R1,[R0]

@---------------------------------LCD\_INIT------------------------------------------@

LDR R0,=0x4802A028 @ Load IRQSTATUS Register

LDR R1,=0xFFFF @ Write 1 to clear all

STR R1,[R0]

@ Sending Slave address 0x3C and along with 10 data bytes

MOV R1,#0x3C @ Slave address

LDR R0,=0x4802A0AC @ Load I2C Slave Address Register

STR R1,[R0]

MOV R1,#10 @ 10 Data bytes ( Counter)

LDR R0,=0x4802A098 @ Load Data Counter Register I2C CNT

STR R1,[R0]

LDR R0,=0x4802A0A4 @ LOAD I2C\_CON Address

LDR R1,=0x8603 @ I2C Enable/Master/Transmit/STT/STP

STR R1,[R0]

@ Sending the first packet data: 0x0

MOV R1,#0x0 @ 0x0: Control Byte

BL Transmit\_Data

@ Sending the second byte data: 0x38

MOV R1,#0x38 @ 0x38: Func set

BL Transmit\_Data

@ Sending the third byte data: 0x39

MOV R1,#0x39 @ 0x39 : Func set

BL Transmit\_Data

@ Delay 10ms

BL Delay

@ Sending the fourth byte data: 0x14

MOV R1,#0x14 @ 0x14: Bias set

BL Transmit\_Data

@ Sending the fifth byte data: 0x78

MOV R1,#0x78 @ 0x78: Contrast set

BL Transmit\_Data

@ Sending the sixth byte data: 0x5E

MOV R1,#0x5E @ 0x5E: Power/ICON control/ Contrast set

BL Transmit\_Data

@ Sending the seventh byte data: 0x6D

MOV R1,#0x6D @ 0x6D: Follower control

BL Transmit\_Data

@ Sending the eighth byte data: 0x0C

MOV R1,#0x0C @ 0x0C: Display on

BL Transmit\_Data

@ Sending the ninth byte data: 0x01

MOV R1,#0x01 @ 0x01: Clear Display

BL Transmit\_Data

@ Sending the tenth byte data: 0x06 ( last data byte)

MOV R1,#0x06 @ 0x06: Entry mode set

BL Transmit\_Data

@ Wait access

BL Wait\_access

BL Turn\_off\_ardy

@ Delay 10ms

BL Delay

@--------------------------DISPLAY MY NAME------------------------------------@

LDR R0,=0x4802A028 @ Load IRQSTATUS Register

LDR R1,=0xFFFF @ Write 1 to clear all

STR R1,[R0]

@ Sending Slave address 0x3C and Data bytes

MOV R1,#0x3C @ Slave address

LDR R0,=0x4802A0AC @ Load I2C Slave Address Register

STR R1,[R0]

MOV R2,#11 @ 10 Data bytes ( Counter)

LDR R0,=0x4802A098 @ Load Data Counter Register I2C CNT

STR R1,[R0]

LDR R0,=0x4802A0A4 @ LOAD I2C\_CON Address

LDR R1,=0x8603 @ I2C Enable/Master/Transmit/STT/STP

STR R1,[R0]

@ Sending the first data: 0x80

MOV R1,#0x80 @ 0x80: Control byte

BL Transmit\_Data

@ Sending the second byte data: 0x86

MOV R1,#0x86 @ 0x86: Position

BL Transmit\_Data

@ Sending the third byte data: 0x40

MOV R1,#0x40 @ 0x40: Datasend

BL Transmit\_Data

SUB R2,R2,#3 @ Decrement counter by 3 because of sending 3 data

@( 0x80, 0x86, 0x40)

LDR R0,=MyName @ Load address memory storing my name String

**Display\_Name:** @ Critical Section: sending each ascii character and update pointer.

@ Sending the fourth byte data: 0x48

LDRB R1,[R0],#0x1 @ Get ascii character and update pointer

BL Transmit\_Data @ Transmit data to display

SUBS R2,R2,#0x1 @ Decrement counter

BNE Display\_Name @ If decrement equal zeros --> Finish

**LOOP:**

NOP

B LOOP

@---------END---------------

@ PROCEDURE: Transmit 1 byte data from BBB to I2C\_LCD-------------------------------@

**Transmit\_Data:**

STMFD R13!,{R0,R14} @ Store uses registers on Stack

LDR R0,=0x4802A09C @ Load I2C Data Register

STR R1,[R0] @ Store value to transmitting

@ Wait xrdy

BL Wait\_transmit

BL Turn\_off\_xrdy

LDMFD R13!,{R0,R14} @ Restore values for saved registers

MOV PC, R14 @ Return to mainline

@-----------------------------------------------------------------------------------@

@ PROCEDURE: Transmit status (XRDY) is generated when the CPU needs to put more data

@ in the I2C\_DATA register after the transmitted data has been shifted out on the SDA

@ pin. This procedure: Wait the bit XRDY before transmitting the next data byte

**Wait\_transmit:**

STMFD R13!,{R0-R1,R14} @ Store uses registers on Stack

@ Wait xrdy for transmitting

**Wait\_xrdy:**

LDR R0,=0x4802A024 @ IRQ STATUS RAW

LDR R1,[R0] @ Load value from IRQ\_STATUS RAW

TST R1,#0x10 @ Test Bit 4

BEQ Wait\_xrdy

LDMFD R13!,{R0-R1,R14} @ Restore values for saved registers

MOV PC, R14 @ Return to mainline

@ ----------------------------------------------------------------------------------@

@ PROCEDURE: Wait\_access

@ Registers-ready-for-access (ARDY) is generated by the I2C when the previously

@ programmed address, data, and command have been performed and the status bits have

@ been updated. This bit is used to let the CPU know that the I2C registers are ready

@ for access. This procedure: Wait the bit ARDY before transmitting the next packet

**Wait\_access:**

STMFD R13!,{R0-R1,R14} @ Store uses registers on Stack

@ Wait xardy for accessing

**Wait\_ardy:**

LDR R0,=0x4802A024 @ IRQ STATUS RAW

LDR R1,[R0] @ Load value from IRQ\_STATUS RAW

TST R1,#0x04 @ Test Bit 2

BEQ Wait\_ardy

LDMFD R13!,{R0-R1,R14} @ Restore values for saved registers

MOV PC, R14 @ Return to mainline

@-----------------------------------------------------------------------------------@

@ PROCEDURE: Turn\_off\_xrdy

@ The CPU poll this bit to write the next transmitted data into the I2C\_DATA register

@ This procedure: poll the bit XRDY

**Turn\_off\_xrdy:**

STMFD R13!,{R0-R1,R14} @ Store uses registers on Stack

LDR R0,=0x4802A028 @ Load IRQSTATUS Register

MOV R1,#0x10 @ Write 1 to clear xrdy

STR R1,[R0]

LDMFD R13!,{R0-R1,R14} @ Restore values for saved registers

MOV PC, R14 @ Return to mainline

@-----------------------------------------------------------------------------------@

@ PROCEDURE: Turn\_off\_ardy

@ The CPU poll this bit for stopping access

@ This procedure: poll the bit ARDY

**Turn\_off\_ardy:**

STMFD R13!,{R0-R1,R14} @ Store uses registers on Stack

LDR R0,=0x4802A028 @ Load IRQSTATUS Register

MOV R1,#0x4 @ Write 1 to clear ardy

STR R1,[R0]

LDMFD R13!,{R0-R1,R14} @ Restore values for saved registers

MOV PC, R14 @ Return to mainline

@-----------------------------------------------------------------------------------@

@ PROCEDURE: Delay 10ms

**Delay:**

STMFD R13!,{R0-R1,R14} @ Store uses registers on Stack

LDR R0,=0xFFFF @ Delay 10ms

**wait\_delay:**

SUBS R0,R0,#1

BNE wait\_delay

LDMFD R13!,{R0-R1,R14} @ Restore values for saved registers

MOV PC, R14 @ Return to mainline

@-----------------------------------------------------------------------------------@

@ DATA STRUCTURE

**.data**

**MyName:**

**.ascii** "Hai Dang"

.END