YILU ZHOU

EVB Pin		Port Bit	Bit Addresses & Labels
1	2	1.	
		2.	
3	4	3.	
		4.	
5	6	5.	
		6.	
7	8	7.	
		8.	
9	10	9.	
		10.	P1.2 — 0x92 CEX2
11	12	11.	P1.3 — 0x93 CEX3
		12.	P1.0 — 0x90 CEX0
13	14	13.	
		14.	P0.6 — 0x86 SDA
15	16	15.	P0.7 — 0x87 SCL
		16.	
17	18	17.	
		18.	
19	20	19.	
		20.	
21	22	21.	
		22.	
23	24	23.	
		24.	
25	26	25.	
		26.	
27	28	27.	
		28.	
29	30	29.	
		30.	
31	32	31.	P3.6 — 0xB6 SS
		32.	
33	34	33.	
		34.	
35	36	35.	
		36.	
37	38	37.	
		38.	
39	40	39.	
		40.	
41 ⟷ 60			

Software Initializations

A) Port I/0

```
P1MDOUT = 0x0D;
P3MDOUT &= 0xBF;
P3 |= ~0xBF;
```

B) Timers

C) Interrupts

```
EA = 1;
EIE1 |= 0x08;
```

D) A/D

E) PCA

```
PCA0MD =0x81;
PCA0CPM2 = 0xC2;
PCA0CN = 0x40;
```

F) XBAR

```
XBR0 = 0x27;
```

G) I2C

```
SMB0CR = 0x93;
ENSMB = 1;
```

Z JIN

EVB Pin			Port Bit	Bit Addresses & Labels
1	2	1.		
		2.		
3	4	3.		
		4.		
5	6	5.		
		6.		
7	8	7.		
		8.		
9	10	9.		
		10.	P1.2	0x92 CEX2
11	12	11.	P1.3	0x93 CEX3
		12.	P1.0	0x90 CEX0
13	14	13.		
		14.	P0.6	0x86 SDA
15	16	15.	P0.7	0x87 SCL
		16.		
17	18	17.		
		18.		
19	20	19.		
		20.		
21	22	21.		
		22.		
23	24	23.		
		24.		
25	26	25.		
		26.		
27	28	27.		
		28.		
29	30	29.		
		30.		
31	32	31.		
		32.	P3.7	0xB7 SS
33	34	33.		
		34.		
35	36	35.		
		36.		
37	38	37.		
		38.		
39	40	39.		
		40.		
41 ⟷	60			

Software Initializations

A) Port I/0

```
P1MDOUT = 0x0D;
P3MDOUT &= 0xBF;
P3 |= ~0xBF;
```

B) Timers

C) Interrupts

```
EA = 1;
EIE1 |= 0x08;
```

D) A/D

E) PCA

```
PCA0MD =0x81;
PCA0CPM2 = 0xC2;
PCA0CN = 0x40;
```

F) XBAR

```
XBR0 = 0x27;
```

G) I2C

```
SMB0CR = 0x93;
ENSMB = 1;
```

YILU ZHOU

compile derivatives
```
#include <studio.h>
#include <i2c.h>
#include <c8051_SDCC.h>
#include <<stdlib.h>
```
declare global variables
```
PCA_COUNTER, PCA_START, READ_COUNTER, DISTANCE,
PW_CENTER, PW_MIN, PW_MAX, PW, PCA_START, PCA_COUNTER,
NEUT_HEIGHT = 45cm.
sbit CF(PCA0 COUNTER OVERFLOW FLAG), SS(slide switch)
```
function prototypes
```
void Port_Init(void);
void SMB_Init(void);
void PCA_Init(void);
void XBR0_Init(void);
void Interrupt_Init(void);
void PCA_ISR(void) __interrupt 9;
void Ping_Ranger(void);
unsigned int Read_Ranger(void);
void Drive_Motor(void);
```
main function
```
declare local variables
    NONE
initialize system, ports and PCA
    Sys_Init();
    putchar(' ');
    Port_Init();
    SMB_Init();
    XBR0_Init();
    Interrupt_Init();
    PCA_Init();
Begin infinite loop
    If slide switch is on
        If 80 ms has passed
            read the ranger using Read_Ranger()
            Set DISTANCE equals the result of Read_Ranger()
            start a ping using Ping_Ranger()
            Reset the READ_COUNTER
            Call Drive_Motor() to control the speed of the motor.
        End if
    End if
    else
        set the PW equals PW_CENTER(stop the motor)
End infinite loop
```
End main function

functions
```
void Port_Init()
    set P3.6 to digital input, set P3.6 to high impedance.
    set output pin for CEX0, CEX2, and CEX3 in push-pull mode.
End Port_Init()

void SMB_Init()
```

```
   Set SCL to 100khz
   Enable SMBus
End SMB_Init()

void XBR0_Init()
   configure the crossbar the same as the the same as Lab3.1 (0x27)
End XBR0_Init()

void Interrupt_Init()
   Enable general Interrupt
   Enable PCA overflow interrupts
End Interrupt_Init()

void PCA_Init()
   Enable SYSCLK/12 and enable interrupts
   Enable PCA COUNTER
End PCA_Init()

void PCA_ISR() __interrupt 9
   Increment PCA_COUNTER to count the number of overflows
   if PCA_COUNTER is larger than 3
      Increment the READ_COUNTER
      Set PCA_COUNTER to 0
   End if
   If PCA interrupt flag is set
      Clear the overflow flag
      Set PCA0 to PCA_START
   End if
   handle other PCA interrupt sources
End PCA_ISR() __interrupt 9

void Ping_Ranger()
   write 0x51 to reg 0 of the ranger
   write one byte of data to reg 0 at addr
End Ping_Ranger()

unsigned int Read_Ranger()
   define local variables
      Data[2]: to store the data from ranger
      range: to store the calculated distance
      addr: the address of the ranger
   read two bytes, starting at reg 2 of the ranger
   calculate the distance from the Data
   return the range
End Read_Ranger()

void Drive_Motor()
   int ERROR;
   ERROR = NEUT_HEIGHT - DISTANCE
   Let PW equals k*(ERROR) + PW_CENTER
   if DISTANCE is larger than 80cm
      Set PW to PW_MIN, full reverse
   End if
   else if DISTANCE is smaller or equal to 10cm
```

```
    Set PW to PW_MAX, full forward
  End if
  update speed command
    update lo byte of CCM 2
    update hi byte of CCM 2
  print PW
End Drive_Motor()
```

YILU ZHOU

```
compile directives
   #include <studio.h>
   #include <c8051_SDCC.h>
   #include <stdlib.h>
   #include <I2C.h>
declare global variables
   unsigned int PW_CENTER, PW_LEFT, PW_RIGHT, SERVO_PW,
   ReadCompass, PCA_START, h_count, new_heading,
   SS, Desired_Heading, error, actual_heading.
   sbit CF (PCA 0 COUNTER OVERFLOW FLAG)
function prototypes

   void Port_Init(void);

   void SMB_Init(void);
   void PCA_Init(void);
   void XBR0_Init(void);
   void Interrupt_Init(void);
   void PCA_ISR (void) __interrupt 9;

main function
   initialize system, ports and PCA
      Sys_Init();
      putchar(' ');
      SMB_Init();
      XBR0_Init();
      Interrupt_Init();
      PCA_Init();
      Port_Init();
   print beginning message.
   wait for 1 sec
   start while (1) loop
      if ss is on and 40 ms has passed
         actual_heading = ReadCompass();
         error = Desired_Heading- actual_heading
         if error larger than 1800
            error-=3600
         else if error smaller than -1800
            error+=3600
         SERVO_PW = errors * Kps + PW_CENTER // 5/12 = 750/1800
         if SERVO_PW > PW_RIGHT, limit it to PW_RIGHT
         if SERVO_PW < PW_LEFT, limit it to PW_LEFT
         print SERVO_PW
         print desired_heading
         print actual_heading
         update Servo command
      else
         SERVO_PW = PW_CENTER
      update lo byte of CCM 0
      update hi byte of CCM 0
   End while (1)loop
End main function
```

Functions

Z JIN

```
void Port_Init()
    set output pin for CEX0, CEX2, and CEX3 in push-pull mode.
    set p3.7 to digital input.
End Port_Init()

void SMB_Init()
    set SCL to 100KHz
    Enable SMBus
End SMB_Init()

void XBR0_Init()
    configure the crossbar as directed in the labor manual.
End XBR0_Init()

void Interrupt_Init()
    Enable general interrupt
    Enable PCA overflow interrupts
End Interrupt_Init()

void PCA_Init()
    Enable SYSCLK/12 and enable interrupt.
    Enable CCM0 16bit PWM
    Enable PCA counter
End PCA_Init()

void PCA_ISR() __interrupt 9
    Increment PCA_COUNTER to count the number of overflows
    If PCA interrupt flag is set
        Clear the overflow flag
        Set PCA0 to PCA_START
        if two overflow is done
            set h_count to 0
            set new_heading to 1
    handle other PCA interrupt sources
    increment count
End PCA_ISR() __interrupt 9

unsigned int ReadCompass()
    unsigned char addr to 0xC0 for compass's address
    unsigned char Data[2] that is an array with length of 2
    unsigned int heading for returning degrees between 0 and 3599
    read two byte starting at reg 2
    set heading equals the combine of two values from reg 2
    return heading in thenths degrees
End ReadCompass.
```