Driving Servo Motor with EAB

Introduction

A servo system is defined as the drive, motor, and feedback device that allow precise control of position, velocity, or torque using feed-back loops. Examples of servo motors include motors used in machine tools and automation robots. In order to drive a servo motor, Pulse Width Modulation (PWM) signals is required. The Embedded Application Board is used to generate the PWM Signals for driving the Servo Motor.

➢ Servo motors operate at 50 Hz or have a PWM period of 20mSec.

➢ Servo motor is controlled by high time of PWM Signal.

➢ The High time should be between 0.8mSec to 2.2mSec.

➢ The high time controls the angular position of servo arm. For e.g.:

✓ 0.8mSec Left most position (around +60 degree)

✓ 1.5mSec Centre position (0 degree)

✓ 2.2msec Right most position (around -60 degree)

Components

The Components required for Controlling Servo Motor with EAB are…

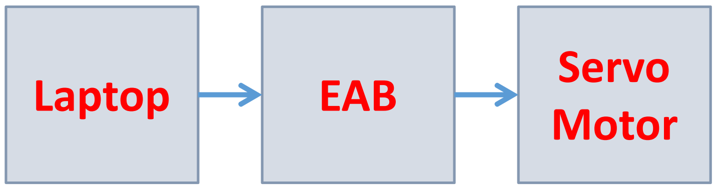
➢EAB

➢Servo Motor

➢ USB Cable

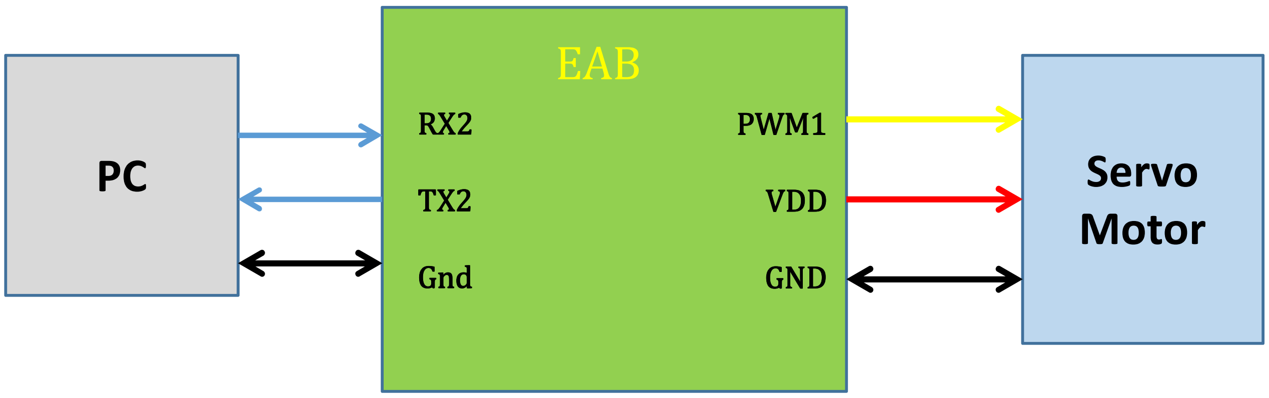
Block Diagram

Block level representation of the different blocks of the Servo Motor Control is shown bellow...



Schematic Diagram

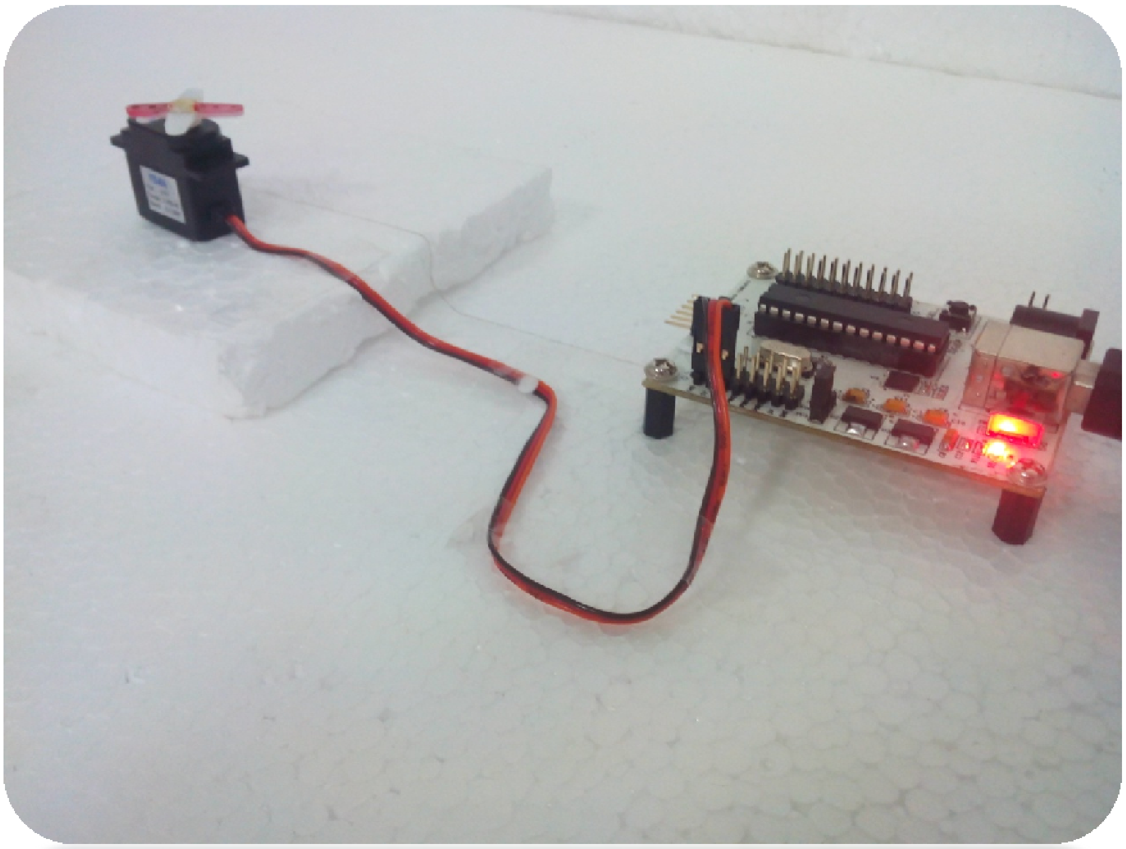
The Schematic diagram illustrates the circuit connections for designing the application.



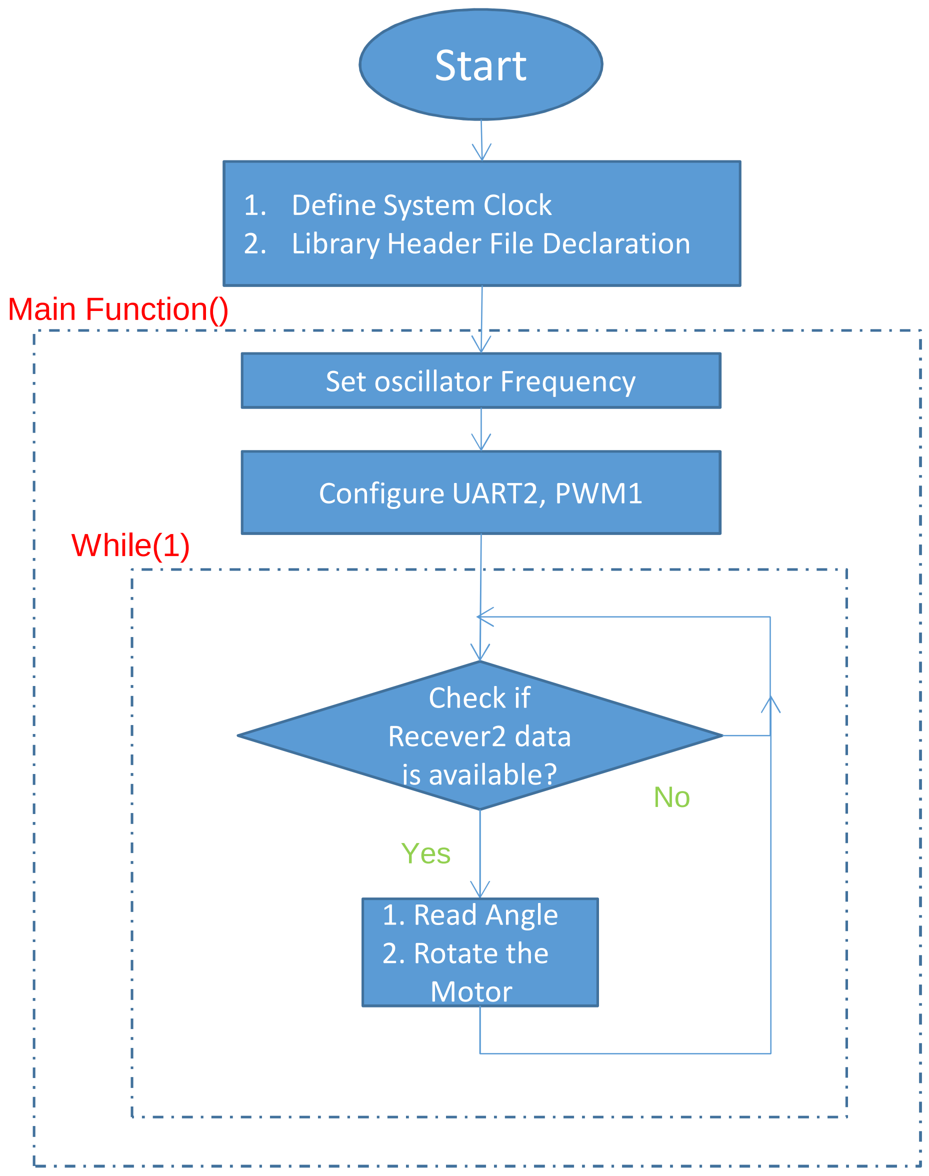
Connection Description

The Servo Motor has 3 pins viz. PWM Signal pin, VDD and Ground pin. These pins are connected to the respective PWM, Ground and VDD pin of Embedded Application Board. For driving the Servo Motor, the operating voltage of Embedded Application Board should be set to 5V. The PWM1 of Embedded Application Board is used for connecting with PWM Signal pin of Servo Motor. It is advise not to rotate the servo motor manually, which in turn can damage the Servo Motor.

Note: *Any PWM pin can be used for Controlling a Servo Motor.*



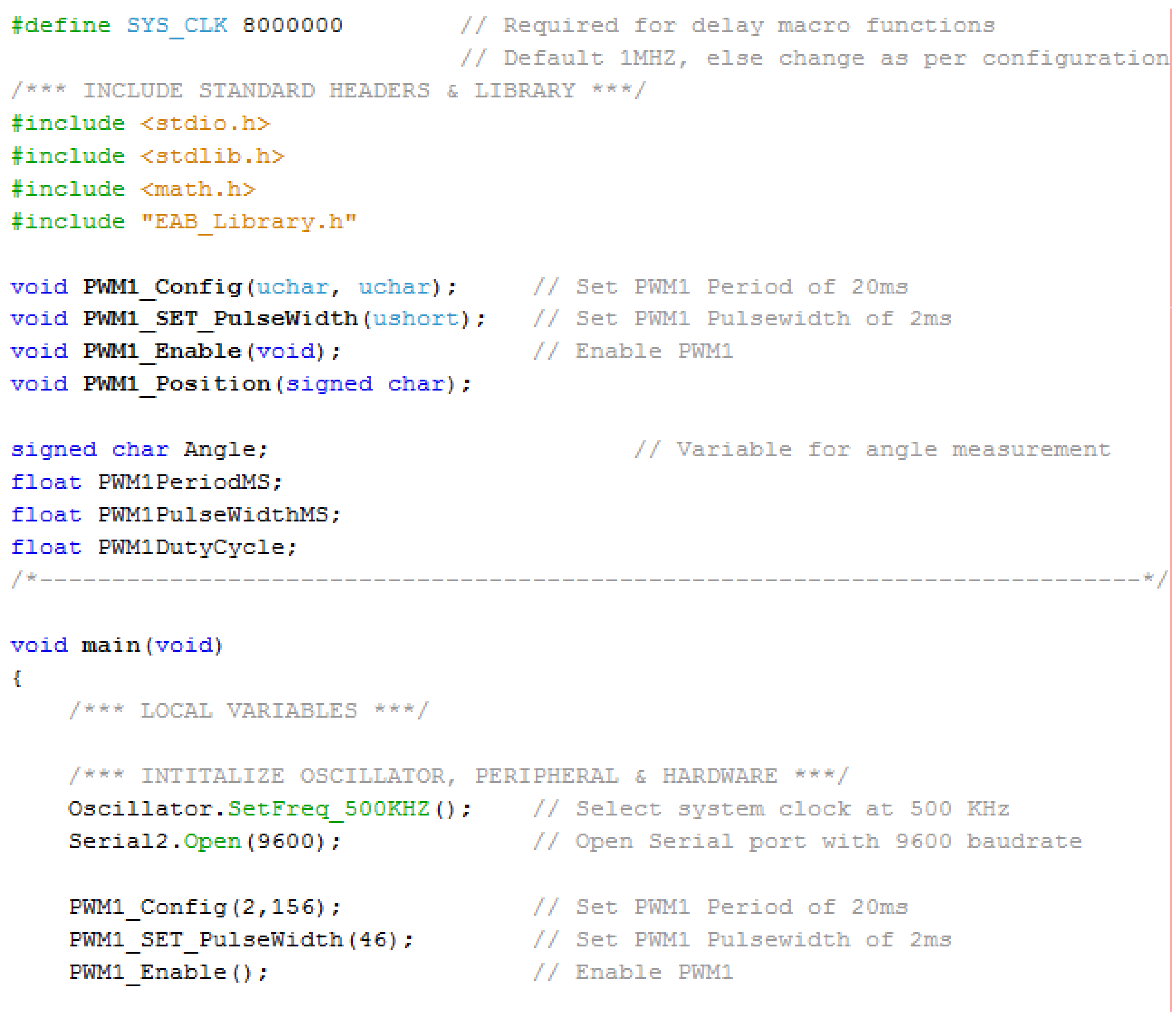
Code Flow Chart



Source Code

The Source code shown below is the firmware to be flashed in the microcontroller of the Embedded Application Board. The Source code is commented for better understanding of the user.

Refer to the EAB User Guide and the EAB Programming Guide for more details on how to Flash(burn) program(Source Code) in the microcontroller of Embedded Application Board.



```
#define SYS_CLK 8000000        // Required for delay macro functions
                               // Default 1MHZ, else change as per configuration
/*** INCLUDE STANDARD HEADERS & LIBRARY ***/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "EAB_Library.h"

void PWM1_Config(uchar, uchar);      // Set PWM1 Period of 20ms
void PWM1_SET_PulseWidth(ushort);    // Set PWM1 Pulsewidth of 2ms
void PWM1_Enable(void);              // Enable PWM1
void PWM1_Position(signed char);

signed char Angle;                          // Variable for angle measurement
float PWM1PeriodMS;
float PWM1PulseWidthMS;
float PWM1DutyCycle;
/*----------------------------------------------------------------------------*/

void main(void)
{
    /*** LOCAL VARIABLES ***/

    /*** INTITALIZE OSCILLATOR, PERIPHERAL & HARDWARE ***/
    Oscillator.SetFreq_500KHZ();     // Select system clock at 500 KHz
    Serial2.Open(9600);              // Open Serial port with 9600 baudrate

    PWM1_Config(2,156);              // Set PWM1 Period of 20ms
    PWM1_SET_PulseWidth(46);         // Set PWM1 Pulsewidth of 2ms
    PWM1_Enable();                   // Enable PWM1
    /*** PLACE THE REPETITIVE TASKS IN THIS LOOP ***/
    while(1)
    {
        if(Serial2_RxFlag)              // Check if UART2 Buffer is high
        {
            Serial2_RxFlag=0;           // Clear the receiver flag
            Angle= Serial2.ReadByte();  // Put the receive value to variable
            PWM1_Position(Angle);       // Call function
        }
    }
}
/*---------------------------------------------------------------------------*/
/*** Function for converting angle value to pulsewidth of PWM Signals ***/
void PWM1_Position(signed char Angle1 )
{
    float x, x0, y, y0, c;
    uint K;
    float  tosc = 1.0/500000;
    y=Angle1;
    x0=0.0015;
    y0=0;
    c=-0.00001; //slope = (x1-x0)/(y1-y0)
    if(y>60)// condition for input angle
    {
      y=60;
    }
    if(y<-60)
    {
      y=-60;
    }
    // Calculate Duty cycle in millisecond from angle
    x=x0+(y-y0)*c;
     //Calculate the Value of Duty cycle register from Time
    K= x/( tosc* 16);
    PWM1_SET_PulseWidth(K);
}
void PWM1_Config(uchar v_PreScaler_u8,uchar v_PeriodNumber_u8)
{
    signed long v_FreqOsc_s32;
    uchar a_Prescaler_u8[]={1,4,16};
    v_FreqOsc_s32 = 500000;             //FreqOsc

    PinDigitalIn(PWM1);                 // Disable the PWM1 pin output driver
    CCPTMRS0bits.C1TSEL = 0x01;         // Select Timer 4 for PWM use
    CCP1CONbits.CCP1M = 0b1100;         // CCP1 configured in PWM mode

    T4CONbits.T4CKPS = v_PreScaler_u8;
    PR4 = v_PeriodNumber_u8;
    T4CONbits.TMR4ON = 1;               //Enable Timer 1

    PWM1PeriodMS = (4*1000.0*a_Prescaler_u8[v_PreScaler_u8]*v_PeriodNumber_u8/
            ((float)v_FreqOsc_s32));
}
/*---------------------------------------------------------------------------*/
```

```
void PWM1_SET_PulseWidth(ushort v_WidthNumber_u16)
{
    float v_FreqOsc_s32;
    ushort v_WidthThreshold_u16 = PR4;
    uchar a_Prescaler_u8[]={1,4,16};
    uchar v_PreScaler_u8 = T4CONbits.T4CKPS;
    v_FreqOsc_s32 = 500000;

    v_WidthThreshold_u16 = 4*v_WidthThreshold_u16 + 1;

    if(v_WidthNumber_u16>v_WidthThreshold_u16)     //Duty cycle cant be more than 100%
        v_WidthNumber_u16 = v_WidthThreshold_u16;  //Setting 100% duty cycle

    CCP1CONbits.DC1B = (uchar)(v_WidthNumber_u16 & 0x0003); //LSB for pulse width1

    CCPR1L = (uchar)((v_WidthNumber_u16 & 0x03FC)>>2);      //MSB for the pulse width

    PWM1PulseWidthMS = (1000.0*v_WidthNumber_u16*a_Prescaler_u8[v_PreScaler_u8])/
            ((float)v_FreqOsc_s32);
    PWM1DutyCycle = (PWM1PulseWidthMS * 100)/PWM1PeriodMS;
}
/*----------------------------------------------------------------------------*/

void PWM1_Enable(void)
{
    PIR5bits.TMR4IF=0;
    while(!PIR5bits.TMR4IF);
    PIR5bits.TMR4IF=0;
    PinDigitalOut(PWM1);                    // Set PWM1 pin direction as output
}
/*----------------------------------------------------------------------------*/
```

Output

After the code is programmed into the microcontroller, connect the servo motor with proper pin connection. Open the terminal program HTerm in Laptop/PC. Provide the desired angle of rotation in HTerm and hit enter. For 0 to 60 degree angle rotation, user can enter any value between 0 to 60. But, for -1 to -60 degree, user need to enter value from 196 to 255. The value 196 indicates -60 degree and 255 indicates -1 degree. So any value between 128 to 256 can be entered for 0 to -60 angle of rotation. The reason the user need to do so as it is not possible to enter negative value in the terminal program Hterm.

Application

Various Application can be built using Servo Motor. Some of the Applications are:

➢ Controlled Robotic Motion

➢ Security Gate Systems

➢ Automated Manufacturing

➢ RC Airplanes

➢ Robotic Arms

,