ermicroblog

Microcontrollers and Electronics Project Blog

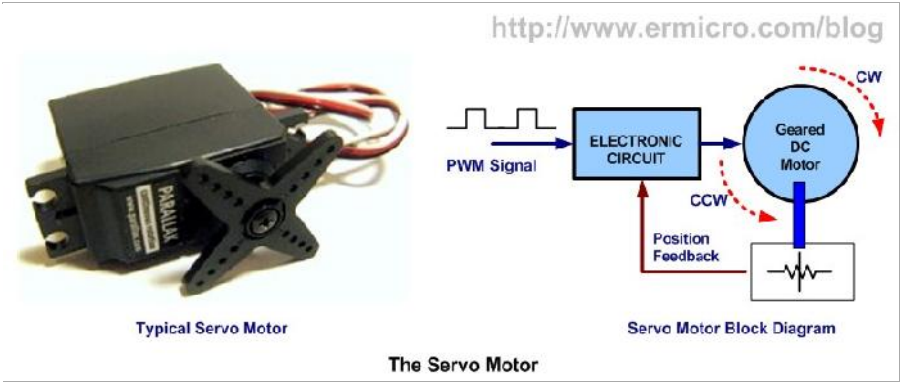
Blog Entry

Basic Servo Motor Controlling with Microchip PIC Microcontroller

February 17, 2009 by rwb, under Robotics.

The servo motor is widely used in model hobbyist such as airplane R/C model for moving the rudder, ailerons, elevators and acceleration control or in the car R/C model for steering and acceleration control. In this tutorial we will learn how to control the servo motor as well as the simple close loop control algorithm for this servo motor.

The servo motor basically is a high quality geared DC motor equipped with electronic circuit for controlling the DC motor rotation direction and position. Currently there are two types of servo motor available on the market, the first one is called **standard servo** and the other one is called **continues servo**; standard servo can rotate to maximum (clockwise or counterclockwise) of 120 to 180 degrees while continues servo can rotate up to 360 degrees in both direction.



The Servo Motor

The Servo Motor

The servo motor use PWM signal for controlling the DC motor; unlike normal PWM usually used in ordinary DC motor; this PWM signal is not use for controlling the rotation speed, instead it is use for controlling the motor direction or position. Most servo motor will work well on 50 Hz of PWM frequency; this mean the PWM signal should have a period of 20ms. The electronic circuit inside the servo motor will response to the PWM signal width; the 0.7ms to 1ms PWM width will make the servo motor to turn clockwise (CW), the 1.7ms to 2ms PWM width will make the servo motor to turn counterclockwise (CCW). For the standard servo the 1.5ms PWM width will turn the servo motor to its center.

Search This Site

Search

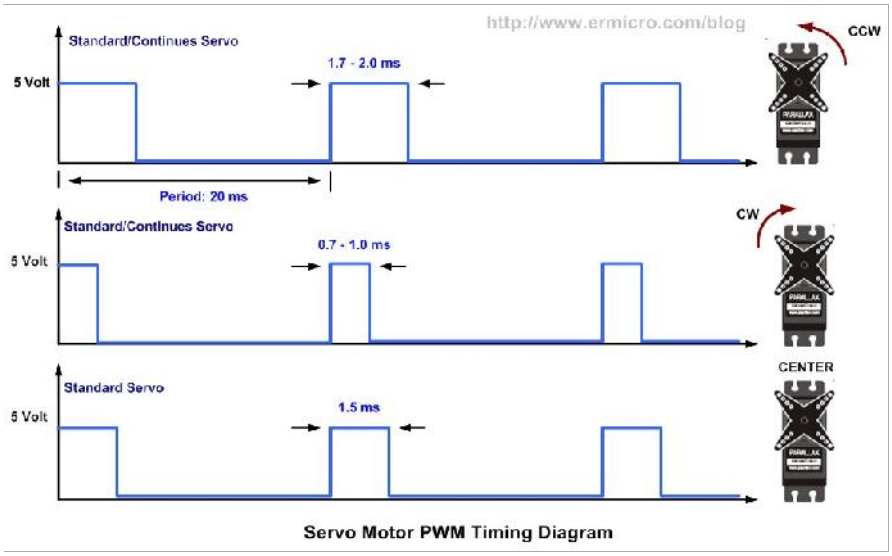
Google™ Custom Search

Future Post

Controlling the Motor is one of interesting topics in the embedded world especially for the robotics enthusiasts, on the next post we will learn the basic of motor electronic circuit as well as how to control it with microcontroller.

Therefore don't miss it, stay tune on this blog !

Subscribe



Servo Motor PWM Timing Diagram

The exact PWM width is depend on the servo motor types and brands; on this tutorial we will use the Parallax Continues Servo which using 1ms and 2ms respectively. The Parallax servo motor consists of three wires colored with White, Red and Black. The Red and Black wires go to the Vcc and Gnd, while the White wire is use to feed the PWM signal from the PIC 16F690 microcontroller I/O port.

Driving the servo motor using PIC 16F690 microcontroller might be simple as you thing at the first time; we just use the PIC PWM peripheral to do the job (you could learn of how to use the PIC PWM peripheral on the article H-Bridge Microchip PIC Microcontroller PWM Motor Controller posted on this blog), but looking at the PIC 16F690 datasheet with the 8 Mhz of internal frequency clock (use in this tutorial) and using maximum prescaler of 16 (TIMER2) the minimum PWM frequency we could achieve can be calculated using this formula:

PWM period = [(PR2 + 1)] x 4 x Tosc x (TMR2 prescaler value) second

Using maximum **PR2** register value of **0xFF** (255 decimal), we will get this result:

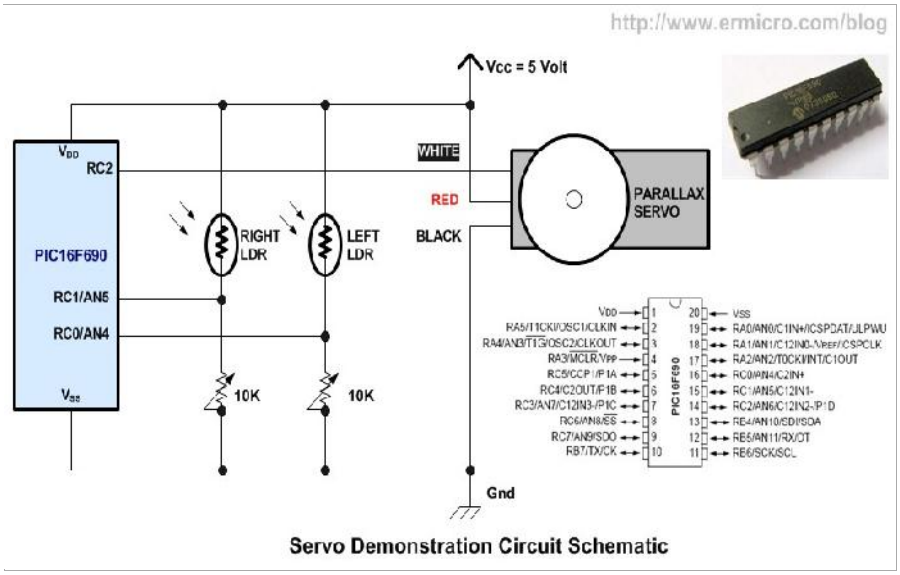
PWM period = (255 + 1) x 4 x (1 / 8000000) x 16= **0.002048 second**
PWM frequency = 1 / PWM period = 1 / 0.002048 = **488.28 Hz**

The 488.28 Hz frequency is still too high from the servo motor working frequency of 50Hz; therefore this leads us to these three methods bellow:

> Keep using the PIC PWM peripheral and lower the operation frequency by setting the **OSCCON** register and **PR2** register until it meets the servo motor frequency requirement. This approach will secrify the program execution speed as we will operate the PIC Microcontroller with the 500 khz clock speed, so we simply not choose it.
>
> Secondly, we create our own PWM function to mimic the PWM signal as follow: turn on the PORT, make some 2 ms delay, turn off the PORT, and make some 18 ms delay and so forth. This approach is what I called a dirty method which is not the efficient way to do it, so we just drop this method.
>
> The third approach is to use the PIC 16F690 microcontroller TIMER0 with the interrupt to actually generate the PWM signal as the TIMER0 have wider prescaler to choose comparing to the TIMER2, but unfortunately the PWM peripheral on the PIC 16F690 only work with TIMER2 not TIMER0. Therefore we will make this TIMER0 as our PWM base generator for driving the servo motor on this tutorial. The principal we learn here could be applied to the other type of PIC Microcontroller or AVR Microcontroller as well.

Instead of just demonstrating the servo motor to rotate clockwise and counterclockwise, I decide to make it more challenging and attractive by putting the LDR (light dependent resistor) as the light sensor to our servo motor and make this servo motor to behave as the light seeking machine; …hey this sound

like we are touching the robotics field; …hmm yes isn't it cool, as we know most of the embedded robotics hobbyist widely use the servo motor for robot's arms, walkers robots, light seeking robot (know also as photovore robot) and many more.

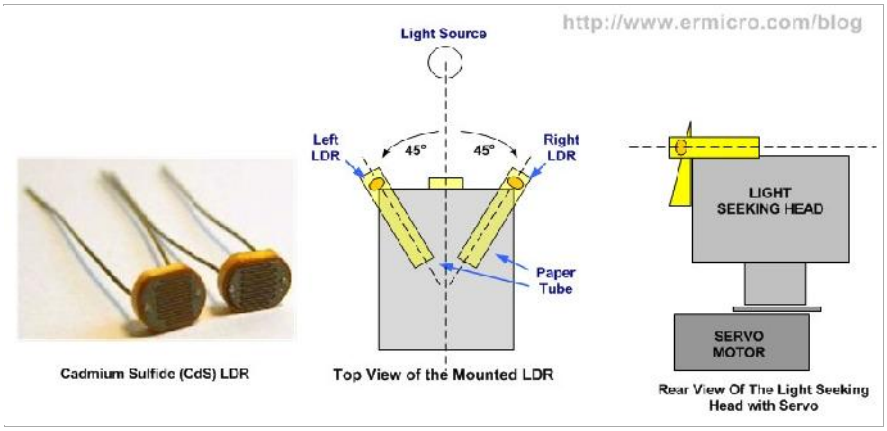


Servo Demonstration Circuit Schematic

The LDR (Light Dependent Resistor)

By just looking at the name, is clear that this is the type of resistor that its resistance depends on the light intensity; it's also called photoresistor, made from the Cadmium Sulfide (CdS) one of the semiconductor material. The LDR will response to light it received, the brightest the light the smaller its resistance and vise versa; on the complete darkness the LDR resistance will become very high (about 150K Ohm; for the LDRs I use in this tutorial)

From the above circuit diagram we connect serially the LDR with the 10K trimport and use it as the voltage divider circuit to the PIC 16F690 analog input **AN4** and **AN5** (you could learn about Basic Resistor Circuit posted in this blog); therefore the variation on the light intensity received by the LDR will result on the variation of the voltage level to the PIC analog input port. Because of the LDR resistance vary widely among the types and brands, therefore we use the trimport and I suggest that you preset it to about 5 K Ohm on the first time and later on you can adjust it as needed.



Cadmium Sulfide (CdS) LDR

Top View of the Mounted LDR

Rear View Of The Light Seeking Head with Servo

The LDR pairs will be functioned as the light sensor that controlled the servo motor position toward the light source; therefore to get the maximum performance we put these two LDRs inside the paper tube in such a way that their position is about 45 degrees away from the light source center (see the top view picture above).

The Light Seeking Head Construction

One of my favorite construction material is to use a thick paper (…yes it's a paper) for fast prototyping as its easy to cut, bend, light weight and quiet strong; our seeking light head project use just a thick paper

Select Category

Blogroll

ermicro shop
ermicroblog Amazon Store
ermicroblog on YouTube
ermicroblog video on Metacafe

Archives

Select Month

► Servo Controller
► Pic Programming
► Arduino Servo

to demonstrate the servo motor concept or what we usually called it as the proof of concept (POC); so let's do the cut and paste; and of course you could experiment with any kind of head forms or faces as you like and these following pictures can be use as the starter.

A. Light Seeking Head Paper Cutting | B. Light Seeking Head Ready to be Painted | C. Final look at the Light Seeking Head

The following is the list of hardware and software used in this tutorial:

1. Thick paper, scissor, glue, duct tape and spray paint for our light seeking head
2. One continues servo motor (in this project I'am using Parallax continues servo)
3. Two LDR
4. Two Trimport 10K
5. PICJazz 16F690 learning board from ermicro (the schema)
6. JazzMate 2576-5V power board, the 5 volt switching power supply
7. Microchip PICKit2 Programmer
8. Microchip MPLAB IDE v8.0 or higher
9. HITEC PICC-Lite PICC-Lite Version 9.60PL1

Now let's take a look at the C code that makes this thing happen:

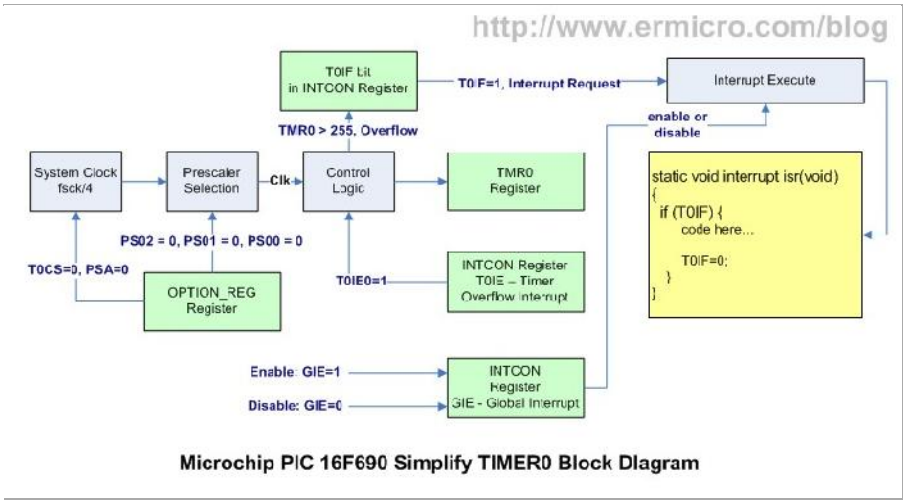
```
// ***************************************************************************
//  File Name    : servo.c
//  Version      : 1.0
//  Description  : Servo Motor Controller
//                 Using TIMER0 for Generating Servo PWM
//  Author       : RWB
//  Target       : PICJazz 16F690 Board
//  Compiler     : HITECT PICC-Lite Version 9.60PL1
//  IDE          : Microchip MPLAB IDE v8.00
//  Programmer   : PICKit2
//  Last Updated : 03 Jan 2009
// ***************************************************************************
#include <pic.h>

/*   PIC Configuration Bit:
**   INTIO     - Using Internal RC No Clock
**   WDTDIS    - Wacthdog Timer Disable
**   PWRTEN    - Power Up Timer Enable
**   MCLREN    - Master Clear Enable
**   UNPROTECT - Code Un-Protect
**   UNPROTECT - Data EEPROM Read Un-Protect
**   BORDIS    - Borwn Out Detect Disable
**   IESODIS   - Internal External Switch Over Mode Disable
**   FCMDIS    - Monitor Clock Fail Safe Disable
*/
__CONFIG(INTIO & WDTDIS & PWRTEN & MCLREN & UNPROTECT \
  & UNPROTECT & BORDIS & IESODIS & FCMDIS);

// Using Internal Clock of 8 Mhz
#define FOSC 8000000L

// Servo definition and variables
#define MAX_VALUE 200
#define CCW_ROTATION MAX_VALUE - 20
#define CW_ROTATION MAX_VALUE - 10
#define STOP_ROTATION MAX_VALUE
#define THRESHOLD_VALUE 50

unsigned char pulse_max=0;
unsigned char pulse_top=0;
unsigned char top_value = 0;

static void interrupt isr(void)
{
```

```
  if(T0IF) {                // TIMER0 Interrupt Flag
    pulse_max++;            // Pulse Max Increment
    pulse_top++;            // Pulse Top Increment

    /* MAX_VALUE=200 turn off the pulse */
    if (pulse_max >= MAX_VALUE) {
      pulse_max=0;
      pulse_top=0;
      RC2=0;                // Turn off RC2
    }

    /* top_value = MAX_VALUE - n, n=10: 10 x 0.1ms = 1.0ms, n=20: 20 x 0.1ms = 2.0ms */
    /* 2ms -> CCW Rotation, 1ms -> CW Rotation */
    if (pulse_top == top_value) {
      RC2=1;                // Turn On RC2
    }

    TMR0 = 156;             // Initial Value for 0.1ms Interrupt
    T0IF = 0;               // Clear TIMER0 interrupt flag
  }
}

void main(void)
{
  unsigned char ldr_left;
  unsigned char ldr_right;
  int ldr_diff;

  OSCCON=0x70;          // Select 8 Mhz internal clock

  /* Initial Port Used */
  TRISC = 0x03;         // Set RC0 and RC1 as input others as Output
  ANSEL = 0x30;         // Set PORT AN4 and AN5 as analog input
  ANSELH = 0x00;        // Set PORT AN8 to AN11 as Digital I/O
  PORTC = 0x00;         // Turn Off all PORTC

  /* Init Servo Pulse */
  pulse_max=0;
  pulse_top=0;
  top_value = MAX_VALUE; // top_value = MAX_VALUE: Servo Motor Stop

  /* Initial ADC */
  ADCON1=0b00110000;    // Select the FRC for 8 Mhz

  /* Init TIMER0: Period: Fosc/4 x Prescaler x TMR0
     0.0005 ms x 2 * 100 = 0.1 ms */

  OPTION = 0b00000000; // 1:2 Prescaller
  TMR0=156;             // Interupt every 0.1 ms
  T0IE = 1;                     // Enable interrupt on TMR0 overflow
  GIE = 1;                      // Global interrupt enable

  for(;;) {
    /* Read the ADC here */
    ADCON0=0b00010001;        // select left justify result. ADC port channel AN4
    GODONE=1;                 // initiate conversion on the channel 4

    while(GODONE) continue;  // Wait for ldr_left conversion done
    ldr_left=ADRESH;          // Read 8 bits MSB, Ignore 2 bits LSB in ADRESL

    ADCON0=0b00010101;        // select left justify result. ADC port channel AN5
    GODONE=1;                 // initiate conversion on the channel 5

    while(GODONE) continue;  // Wait for ldr_right conversion done
    ldr_right=ADRESH;         // Read 8 bits MSB, Ignore 2 bits LSB in ADRESL

    /* Get the different */
    ldr_diff=ldr_left - ldr_right;

    if ((ldr_diff >= -THRESHOLD_VALUE) && (ldr_diff <= THRESHOLD_VALUE)) {
      top_value = MAX_VALUE;     // Stop the Servo Motor
    } else {
      if (ldr_diff > THRESHOLD_VALUE) {
        top_value = CCW_ROTATION;  // Counterclockwise Rotation
      } else {
        top_value = CW_ROTATION;   // Clockwise Rotation
      }
    }
  }
}

/* EOF: servo.c */
```

PIC Microcontroller TIMER0 Peripheral

The heart of the servo motor PWM pulse is rely on the PIC TIMER0 peripheral; this TIMER0 together with the interrupt service will be used as our servo motor base PWM pulse generator. The TIMER0 peripheral is actually the 8-bit counter that always increase it count base on the clock pulse supply to it. The PIC microcontroller TIMER0 counter use the **TMR0** register to hold the counted number, each of the clock pulse will increase the **TMR0** register value by 1 until it reach its maximum value of 255 and start all over again from 0 (overflow); and when this happening the PIC TIMER0 peripheral will rise the interrupt signal. The interrupt signal raised by the PIC TIMER0 peripheral will be interpreted by the microcontroller as this following explanation:

1. Stop whatever you are doing right now
2. Save the current execution address for later used in memory; this area of memory is known as the stack area
3. Jump to the assigned TIMER0 interrupt service address
4. Start execute the code in the TIMER0 interrupt service address
5. Return back to your last task by loading the last execution address in the stack area and continue execute the code from it.



Microchip PIC 16F690 Simplify TIMER0 Block Diagram

The principal we use here is to set the **TMR0** register to overflow every 0.1 ms and set our own counter variable (**pulse_max**) to count up to maximum 200; this will give us the constant 20 ms period which is the same as 50 Hz frequency required by our servo motor.

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
INTCON	GIE	PEIE	T0IE	INTE	RABIE	T0IF	INTF	RABIF	0000 0000	0000 0000
OPTION_REG	RABPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111
TMR0	Timer0 Module Register								xxxx xxxx	uuuu uuuu
TRISA	—	—	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	--11 1111	--11 1111

Legend: – = Unimplemented locations, read as '0', u = unchanged, x = unknown. Shaded cells are not used by the Timer0 module.

Prescaler Rate Select bits (PS2, PS1, PS0) in OPTION_REG:

Bit Value: 000 – 1:2 Prescale; 001 – 1:4 Prescale; 010 – 1:8 Prescale; 011 – 1:16 Prescale; 100 – 1:32 Prescale; 101- 1:64 Prescale
110 – 1:128 Prescale; 111 – 1:256 Prescale

Summary of Registers Associated with PIC 16F690 TIMER0

The TIMER0 period could be calculated using this formula bellow:

TIMER0 period = [(TMR0 + 1)] x 4 x Tosc x (TIMER0 prescaler value) second

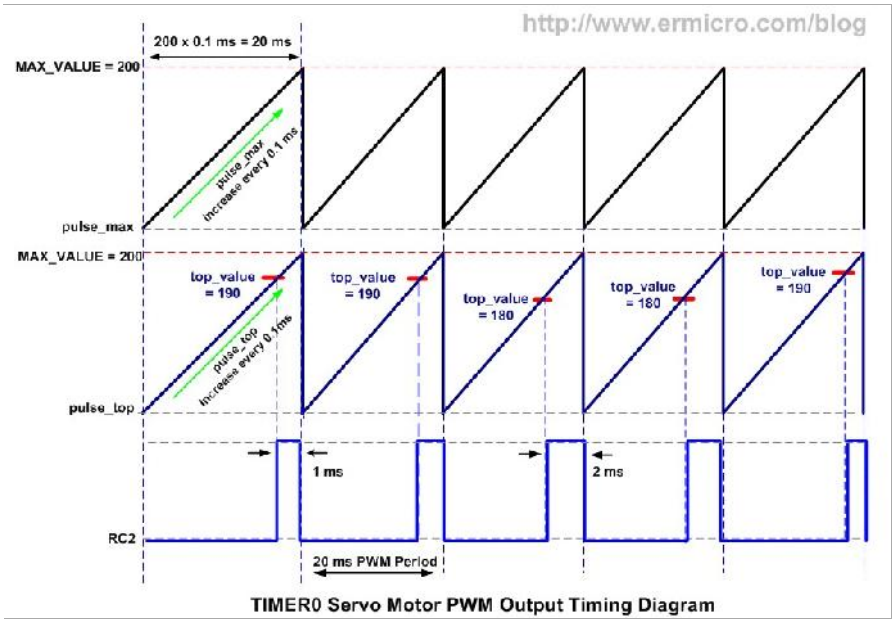
By selecting the TIMER0 prescaler of 2; **PS2**=0, **PS1=0** and **PS0=0** bits in **OPTION_REG** register and initial the **TMR0** register value to **156** (99 more counts to reach its maximum value of 255) with the system frequency clock of 8 Mhz, the PIC microcontroller TIMER0 overflow period can be calculated as follow:

TIMER0 period = [((255 - 156) + 1)] x 4 x 1/8000000 x 2 = 0.0001 second = **0.1 ms**

The following C code is used to initialize the PIC 16F690 TIMER0 peripheral:

```
/* Init TIMER0: Period: Fosc/4 x Prescaler x TMR0
   0.0005 ms x 2 * 100 = 0.1 ms */

OPTION = 0b00000000; // 1:2 Prescaller
TMR0 = 156;          // Interupt every 0.1 ms
T0IE = 1;            // Enable interrupt on TMR0 overflow
GIE = 1;             // Global interrupt enable
```

In order to generated the PWM pulse, we need to have two separate variable counter here; one is the **pulse_max** variable used for resetting the **RC2** port (logical "**0**") which connected to the servo motor and secondly is the **pulse_top** variable used to set the **RC2** port; the **pulse_top** variable will be compared to the **top_value** variable; and if it equal than we will set the **RC2** port (logical "**1**").



TIMER0 Servo Motor PWM Output Timing Diagram

From the timing diagram above the **pulse_max** variable is used to hold our own PWM counter and the **pulse_max** value increase every 0.1ms (**TMR0** overflow). By the time it reaches the maximum value of 200 than we will reset the **RC2** port together with the **pulse_max** variable and the **pulse_top** variable. The **pulse_top** value also increase every 0.1 ms and its value constantly compared to the **top_value** variable value; when it equal we will set the **RC2** port (logical "**1**"); for example by setting the **top_value** variable to 190, means when the **pulse_top** variable reach 190, the **RC2** port will be set and when **pulse_max** reach 200 the **RC2** port will be reset, this will make our servo motor to turn clockwise. Again by setting the **top_value** variable to 180 the servo motor will turn counterclockwise as it receive the 2 ms of PWM signal. Bellow is the code inside the TIMER0 interrupt service routine:

```
if(T0IF) {                // TIMER0 Interrupt Flag
  pulse_max++;            // Pulse Max Increment
  pulse_top++;            // Pulse Top Increment

/* MAX_VALUE=200 turn off the pulse */
  if (pulse_max >= MAX_VALUE) {
    pulse_max=0;
    pulse_top=0;
    RC2=0;                // Turn off RC2
  }

  /* top_value = MAX_VALUE - n, n=10: 10 x 0.1ms = 1.0ms, n=20: 20 x 0.1ms = 2.0ms   */
  /* 2ms -> CCW Rotation, 1ms -> CW Rotation */
  if (pulse_top == top_value) {
    RC2=1;               // Turn On RC2
  }
  TMR0 = 156;             // Initial Value for 0.1ms Interrupt
  T0IF = 0;               // Clear TIMER0 interrupt flag
}
```

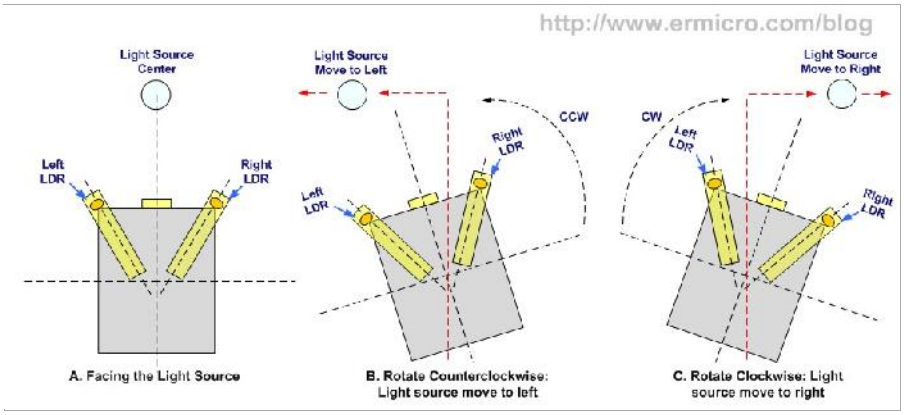
Because the TIMER0 interrupt flag bit (**T0IF**) in the **INTCON** register is not automatically reset to **0**, therefore we have to manually reset it inside the TIMER0 interrupt function.

As you see by using this method the program code for controlling the servo motor will be much easier now as we only assign the correct value to the **top_value** variable and let the PIC microcontroller TIMER0 peripheral do the job by supplying the required PWM signal to the servo motor as it shown on this following C code example:

```
#define MAX_VALUE 200
#define CCW_ROTATION MAX_VALUE - 20
#define CW_ROTATION MAX_VALUE - 10
#define STOP_ROTATION MAX_VALUE

top_value = MAX_VALUE;     // Stop the Servo Motor
top_value = CCW_ROTATION;  // Counterclockwise Rotation
top_value = CW_ROTATION;   // Clockwise Rotation
```

The Light Seeking Sensor

As I mention before that instead of just making our servo motor to rotate clockwise (CW) or counterclockwise (CCW); we will use it to position our paper head toward the light source. This can be achieve by using the pair of LDRs to detect the light source and base on the light intensity information received by the LDR pair; we make the servo motor to rotate in such away that our head paper will always facing the light source as seen on this following picture

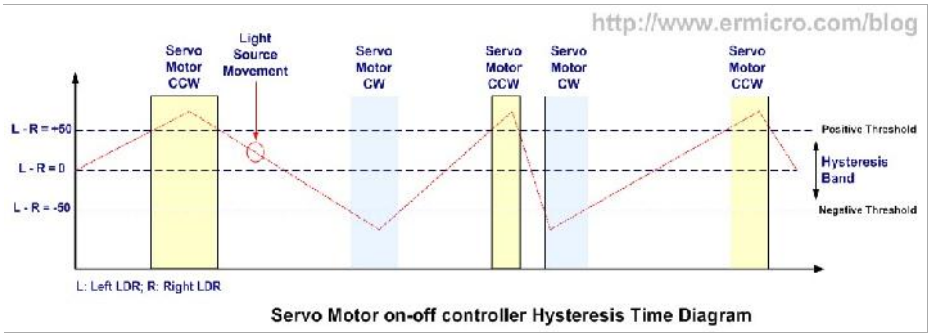


The LDR pairs will constantly give the light source position feedback to the servo motor so it can always turn our paper head toward the light source; this is known as the **close loop control** which is one of the most important topics in the embedded system.

The method of controlling the paper head we use here is called the "**on-off controller**" or "**bang-bang controller**"; this is the simplest method of controlling usually found in the heater, air-condition and refrigerator for controlling the temperature or in the line follower robot and summo wrestling robot. The more advanced controlling method is called the "**PID (Proportional, Integral, Derivative) controller**" which is used such as in motor speed controlling, however we will not discuss this controlling method on this tutorial, but once you understand and implement the principal of the basic close loop control presented here, later on it will be easier for you to learn this PID controlling method.

The algorithm we use here is base on the differential value returned by the left LDR and the right LDR; the positive result will rotate the servo motor counterclockwise and the negative result will rotate the servo motor clockwise. The servo motor will keep rotating until the different result is zero; which mean the paper head is facing the light source.

To make the servo motor rotate smoothly we use the threshold value or known as the **hysteresis band** in the on-off controller method, this mean if the different result is within the hysteresis band the servo motor will always stop and if the different result is outside the hysteresis band the servo motor will start to rotate counterclockwise or clockwise as shown in this following time diagram:



Servo Motor on-off controller Hysteresis Time Diagram

The following is the C code for implementing the on-off controller method with the hysteresis band:

```
#define THRESHOLD_VALUE 50

/* Get the different */
ldr_diff=ldr_left - ldr_right;

if ((ldr_diff >= -THRESHOLD_VALUE) && (ldr_diff <= THRESHOLD_VALUE)) {
  top_value = MAX_VALUE;     // Stop the Servo Motor
} else {
  if (ldr_diff > THRESHOLD_VALUE) {
    topvalue=CCW_ROTATION;  // Counterclockwise Rotation
  } else {
    topvalue=CW_ROTATION;   // Clockwise Rotation
  }
}
```

The LDRs value is read by the PIC ADC peripheral through the analog input port channel 4 (**AN4**) and channel 5 (**AN5**); when the left LDR received more light compare to the right LDR the voltage level input on the **AN4** port will be more higher and the different result in ldr_diff variable will be positive; when the right LDR received more light than the voltage level input on the AN5 port will be more higher and the different result in ldr_diff variable will be negative. The following is the C code used to read the LDR's voltage level:

```
/* Read the ADC here */
ADCON0=0b00010001;       // select left justify result. ADC port channel AN4
GODONE=1;                    // initiate conversion on the channel 4

while(GODONE) continue;  // Wait for ldr_left conversion done
ldr_left=ADRESH;         // Read 8 bits MSB, Ignore 2 bits LSB in ADRESL

ADCON0=0b00010101;       // select left justify result. ADC port channel AN5
GODONE=1;                    // initiate conversion on the channel 5

while(GODONE) continue;  // Wait for ldr_right conversion done
ldr_right=ADRESH;        // Read 8 bits MSB, Ignore 2 bits LSB in ADRESL
```

To learn more about the PIC ADC (analog to digital conversion) peripheral you could read the PIC Analog to Digital Converter C Programming posted in this blog.

Inside the C Code

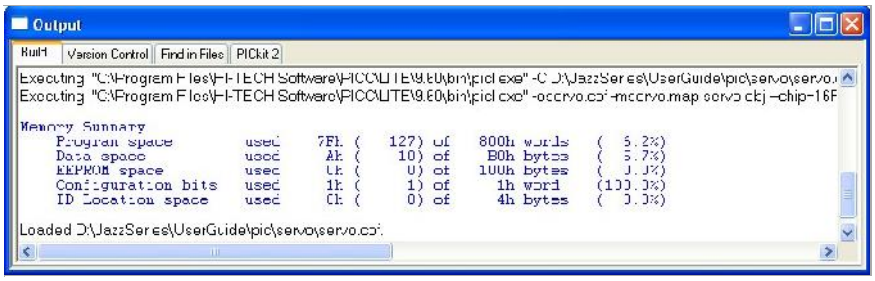
This program is start by initializing the PIC port used in this tutorial and continue with the PIC ADC peripheral clock selection in the **ADCON1** register for using the 8 Mhz internal clock and the last is to initial the PIC TIMER0 peripheral. After the initialization process the program enter the endless loop where we read the voltage level returned by the left LDR and the right LDR and do the "**bang-bang controller**" algorithm.

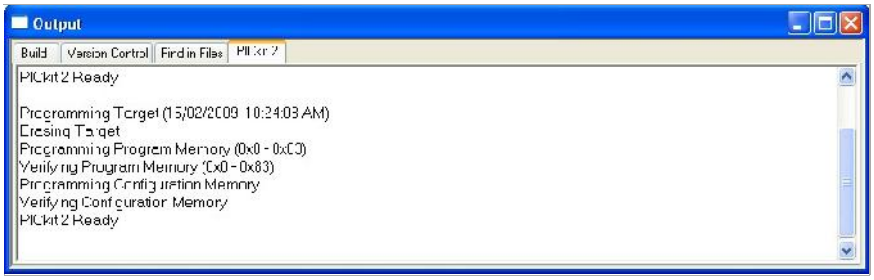
```
/* Initial Port Used */
TRISC = 0x03;        // Set RC0 and RC1 as input others as Output
ANSEL = 0x30;        // Set PORT AN4 and AN5 as analog input
ANSELH = 0x00;       // Set PORT AN8 to AN11 as Digital I/O
PORTC = 0x00;        // Turn Off all PORTC
```

It's important to remember that we have set the port to the analog input mode first before we use it for reading the analog signal.

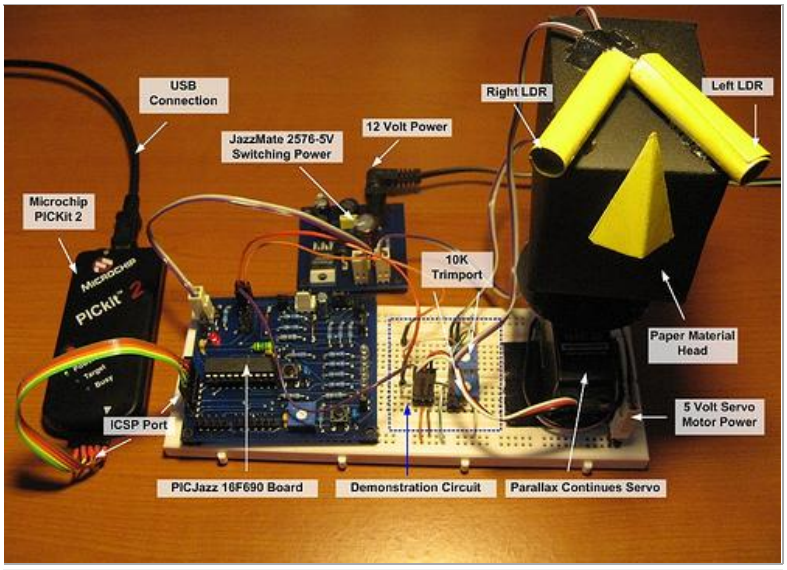
Downloading the Code

After compiling and simulating your code hook up your PICKit2 programmer to the PICJazz 16F690 board ICSP port turn the PICJazz 16F690 power. From the MPLAB IDE menu select **Programmer** -> **Select Programmer** -> **Pickit2** it will automatically configure the connection and display it on the PICkit2 tab Output windows:





Now you are ready to down load the code from MPLAB IDE menu select **Programmer** -> **Program**; this will down load the HEX code into the PICJazz 16F690 board:



Now just relax and enjoy your light seeking head in action:

Bookmarks and Share

Related Posts

- Building BRAM your first Autonomous Mobile Robot using Microchip PIC Microcontroller – Part 1
- H-Bridge Microchip PIC Microcontroller PWM Motor Controller
- Behavior Based Artificial Intelligent Mobile Robot with Sharp GP2D120 Distance Measuring Sensor – BRAM Part 2
- Seven Segment Display Thermometer with PIC Microcontroller
- PIC Analog to Digital Converter C Programming

106 Responses to "Basic Servo Motor Controlling with Microchip PIC Microcontroller"

30.04.09 **#1**
Comment by **ajak**.

Hi Ronald, if i am right this line of code should be change..

unsigned char topvalue=0;

should be like this

unsigned char top_value=0;

can you recheck it? so you can edit it..

thanks..

30.04.09 **#2**
Comment by **ajak**.

Your blog is very nice and very informative.. Keep up the

good job.. Thanks.

Goodluck..

30.04.09 **#3**
Comment by **rwb**.

Thank you ajak for your correction, you are right all the topvalue should be changed with top_value !

30.05.10 **#4**
Comment by **avinash**.

Hi,

How can I know the exact timing requirement of my Servo. I have Futaba S3003 Servo but I am unable to locate the datasheet.

And its web page does not has that Info.

Please help me soon

31.05.10 **#5**
Comment by **rwb**.

For CCW use 1.7 – 2.0 ms, CW 0.7 – 1.0 ms and center is about 1.5 ms. "The exact PWM width is depend on the servo motor types and brands". Try to experiment with your own servo (Futaba S3003) to get the required value, most of the servo is manufactured with wide tolerant to these values; therefore you could start with the value shown on this project.

31.05.10 **#6**
Comment by **avinash**.

Thanks!

But where I can get exact values. Don't the manufacturer provide this data anywhere ?

31.05.10 **#7**
Comment by **rwb**.

Yes they do; for the parallax servo I used in this project they have the data sheet complete with the example code using parallax basic stamp. The keyword here is to "experiment" with your servo.

31.05.10 **#8**
Comment by **avinash**.

Hi,
I experimented with my Futaba S3003 servo and the results were

0.4 ms for 0 degree (may be some call it -90)
2.3 ms for 180 degree

does it look ok?

31.05.10 **#9**
Comment by **avinash**.

Maximum angle the servo is turning is slight more than 180 degrees is that OK ?

31.05.10 **#10**
Comment by **rwb**.

For me as long as the servo works it's ok despite the value you used (remember in electronics world there are no such an absolute value). You could try to gently calibrate the servo by adjusting the servo trimport if it move more than 180 degrees.

07.07.10 **#11**
Comment by **riverdan**.

OK, I highlighted and copy the c code, paste it in notepad ++ ,Save it as c code, project wized it but when i try to compile it i get errors. rwb sent me the code for the room therm and it worked fine. i had the same thing with it. would someone send me the servo.c file. I have Microsoft Visual Studio 2008 riverdan@riverdan.com

Thanks All

07.07.10 **#12**
Comment by **rwb**.

Try to cut and paste again, its seem the variable top_value is wrongly being interpreted by the blog's editor to topvalue. When I intended to put top_value=VALUE it always interpreted as topvalue=VALUE, but if I write top_value = VALUE (i.e. put a space between the equal sign) it seem its correctly put the right one.

25.10.10 **#13**
Comment by **akhutwad**.

I am unable to find PIC16F690 here in Pune. But I am having PIC16F877A and PIC18F452 with me. I tried compiling same program with them,but encounter lots of error. what changes need to be done to work with mention processors. I am new to mc programming, tried changing few things but not working with demo setup.

thanks in advance.

Ankush

25.10.10 **#14**
Comment by **rwb**.

You need to change the configuration bit and the OSCCON register when you use PIC16F877A since the PIC16F877A microcontroller has no internal oscillator clock like PIC16F690.

25.10.10 **#15**
Comment by **akhutwad**.

Please find below code edited by me, original program is written by me. I just tried to modify to work for PIC16F877a.

I wanted to use RA0 & RA1 as input channel and RD7 as output the motor. Please please please have some time and verify the code once (code truncated!)

25.10.10 **#16**
Comment by **rwb**.

If you want to submit a large code, you could email directly to ermicroblog@gmail.com. When you submit your code through this comment it will contain many HTML specified code, therefore your code will be mixed with the HTML code which make your code hard to be read.

Anyway seeing your code it seem you are trying to use "delay_us" and "for()" loop to control your servo. This means you have to measure precisely the servo required timing (PWM). My suggestion is to try the method presented on this project to control your servo.

26.10.10 **#17**
Comment by **akhutwad**.

please advice me what all changes need to be done to work above program as it on pic16f877a. I am new to the programming, please suggest changes. i tried various options, but it did not work. I am not sure whether pic is faulty or not. I tried another program with stepper motor earlier on same PIC and it work there. I want to use servo instead of stepper so I need to test this program.

26.10.10 **#18**
Comment by **rwb**.

The program did not work has many reasons, since you used your own hardware and code. Driving stepper motor is totally different compared to servo motor. Therefore I only can suggest that you learn and understand the principle of how to drive the servo motor presented in this project and try to apply it.

26.10.10 **#19**
Comment by **akhutwad**.

will this logic work for high torque servo motor too? do do we need any signal booster circuit to get more torque?

26.10.10 **#20**
Comment by **rwb**.

Yes, the logic should work on any standard servo motor, but you still need to read the servo motor specification to make sure it could be driven with standard servo PWM signal.

30.10.10 **#21**
Comment by **akhutwad**.

what is the pin layout for PICKit3 icsp programmer for PIC16F690

30.10.10 **#22**
Comment by **rwb**.

The standard Microchip PICKit3 ICSP for PIC16F690: Pin 1: MCLR/Vpp -> RA3 (4),Pin 2:Vdd -> Vdd (1),Pin 3: GND -> Vss (20),Pin 4:PGD/ICSPDAT -> RA0 (19),Pin 5: PGC/ICSPCLK -> RA1 (18),and Pin 6:LVP -> No Connection. For more information you should read the PICKit3 user's guide and PIC16F690 datasheet.

31.10.10 **#23**
Comment by **akhutwad**.

what need to be add to the program, if no light source detected. e.g if there is no light or in complete dark condition i want motor be reset to 0

31.10.10 **#24**
Comment by **rwb**.

In this project when both LDR return equal value (ADC), the servo will stop (the ADC value within the threshold value). Here you could add the program logic to check whether they both in dark condition and then give command to reset the servo (move to center). Again you should understand how the program's work in order to adapt it to your need.

01.11.10 **#25**
Comment by **akhutwad**.

what should be the top_value to reset the motor position.

01.11.10 **#26**
Comment by **rwb**.

The standard servo center position (reset) pulse width is about 1.5 ms. Therefore top_value = MAX_VALUE – 15, where MAX_VALUE = 200. For detail of how to get this value, please study the article above.

10.11.10 **#27**
Comment by **shubash123**.

video is really promising, and reading all the above comments. can this be work for solar tracker. how to move motor for particular angle with pwm signal and reset it back to zero position at the end of day.

10.11.10 **#28**
Comment by **rwb**.

Yes, it could be used for solar tracker, you have to experiment with the PWM timing until it reach to the needed angle (for precision angle you could use a stepper motor). At the end of day you could reset the servo facing to the rising sun on the next day.

12.11.10 **#29**
Comment by **shubash123**.

Thanks for the reply
I m still confuse about motor movement. how motor will move if light source will move only one direction in daytime. how the difference will calculate. what will be the value of ldr

in that condition. as per the above program motor initialize in central position and then as per the light move it goes clockwise or anticlockwise but how to start from zero position

12.11.10 **#30**

Comment by **rwb**.

You should read the above article especially in the "Light Seeking Sensor" part. See the picture and explanation of how the LDR is mounted so they could be functioned as the light tracking sensor. On this project there is no default (zero) position, the servo simply stay on its previous position when dark (no light). Therefore you need to add this feature in the C program code.

13.11.10 **#31**

Comment by **shubash123**.

Hi, I tried this code and circuit on test board, but motor moves faster then visible in above video. are you using different code for video? how motor moves slow here in video? is Trimport doing any trick?

13.11.10 **#32**

Comment by **rwb**.

Yes, the code presented here is the same as the code run in the above video. Thing that you should know when you supply the continues PWM signal to the DC servo motor, then the rotation speed is depend on the servo mechanical (i.e. geared motor inside) not on the PWM signal. The way we control the DC servo motor rotation is by using the required PWM signal for some period of time. Trimport and LDR are used to provide analog input (voltage divider) to the microcontroller.

19.12.10 **#33**

Comment by **saad770**.

please tell me why are u using PICJazz 20PIN Board
.we can use any pic burner?
one thing more servo demoanstration cct is the only cct that we have to implement?

inshort what is the significance of PICJazz 20PIN Board?

thanks

19.12.10 **#34**

Comment by **rwb**.

Of course you could use any development board available on the market, the reason I use the PICJazz 20PIN development board because is specially designed to be used on 8-bit 20 pins Microchip microcontroller PIC16/18 as well as the PICAXE microcontroller which is base on Microchip PIC16/18 microcontroller. The PICJazz 20PIN board is suitable for most beginners and hobbyists who want to make project base on the Microchip 20 pins Microcontroller and the board also use standard through hole components for easy repair. You could find more information about this board here.

On this project I use PIC16F690 TIMER0 to control the servo

movement.

20.12.10 **#35**
Comment by **saad770**.

really sory to disturb u again.
my compiler is giving so much errors i have pasted the same code as given above.here's the errors

Executing: "C:\Program Files\Microchip\MPASM Suite\MPASMWIN.exe" /q /p16F690 "Untitled.asm" /l"Untitled.lst" /e"Untitled.err" /d__DEBUG=1
Error[108] C:\USERS\MY PC\DOCUMENTS\DOWNLOADS \COMPRESSED\UNTITLED.ASM 1 : Illegal character (/)
…[error messages is truncated]…
please help me out.
thanks

20.12.10 **#36**
Comment by **rwb**.

This is a C program (servo.c) not an Assembler program, you need to use the Microchip HITECT PICC C Compiler and configure the Microchip MPLAB IDE to use the HITECT PICC C compiler instead of Microchip PIC Assembler (MPASM) compiler in order to compile this project successfully!

21.12.10 **#37**
Comment by **saad770**.

thanks a lot
actually im new in this thats y dont know.
thanks again

15.01.11 **#38**
Comment by **ayush**.

to get more precision on motor control i decreased the pulse from 1ms to .01ms by changing TMR0 value to 9 and changing max_value to 2000 and top_value to 1900 or 1800 but motor is not working properly so plz tell wats wrong with above method or recommend a different method to improve precision.

16.01.11 **#39**
Comment by **rwb**.

You could not change the 1ms to 0.01ms. The servo motor is designed to work with 1ms pulse period in order to rotate clockwise (please read the articles for more explanation). Therefore to make the servo motor rotate at specified angle, you need to apply this 1ms pulse to the servo at specified period of time, means the servo rotation angle is depend on how long you apply this 1ms pulse to the servo motor.

25.01.11 **#40**
Comment by **pichaha**.

Hello I am a new member in your site…your articles are great…ok in response to your previous reply…how about if I want to stop at a specific angle for some time??

25.01.11 **#41**
Comment by **rwb**.

Just stop supply the required pulse to the servo motor after it get to the specific angle (top_value = MAX_VALUE) and call the delay function.

26.01.11 **#42**
Comment by **pichaha**.

Thanks for the reply…I have two questions here…the first one…if I stop the motor(from your reply it mean keep providing 0v to it right), won't the motor turn back to its initial position which is what other electrical devices will normally function?…secondly if I want to go back to the initial position, all I need is just appplying 1.5ms. pulse width right? Sorry, I am new in servo, thansk a lot

26.01.11 **#43**
Comment by **rwb**.

The servo will stop at its current position when you stop supply a pulse and yes it will go back to center when you apply the 1.5ms pulse.

26.01.11 **#44**
Comment by **pichaha**.

Finally I have some understading about servo…this will definitely provide great help to my projects…thanks a lot and best regards.

09.02.11 **#45**
Comment by **pichaha**.

Hi,
I have bought a new servo…and I have tried to test the servo motor using your timer0 idea…but the result coming out was very weird…my expected outcome is to see the servo keep on turning from clockwise to cuonter clockwise and vice versa…can I send my source code (very simple servo testing codes)to you and you help me rectify where is going wrong? Thanks a lot.

09.02.11 **#46**
Comment by **rwb**.

Driving servo is simply supplying the right PWM pulse, you could use the Microchip PICKit2 programmer as a logic analyzer to see if your program generate the right pulse to the servo motor. For more information you could read "PIC18 Pulse Width Modulation (PWM) DC Motor Speed Controller with the RPM Counter Project" article.

10.02.11 **#47**
Comment by **pichaha**.

Currently I don't have pickit 2, I am using other programmer and mcu(not 16f690) for testing, but my target chip will be 16f690. Is there any other method?
Actually, I connected the pwm signal and power lines wrongly when I first using the servo but I discovered immediately and

unplug them (only a few seconds). Will this bring any damage to my servo motor?

10.02.11 **#48**
Comment by **pichaha**.

I have tried to measure my pwm signal using osciloscope, I acquired 44.4Hz for my frequency and 22.5ms for my period…Not the exact 50Hz and 20ms, is it the reason why it can't work properly? So, if I want to turn cw or ccw, do i need to set exactly the value of 1ms or 2ms…if the value less or greater will it work? Sorry for the inconvenience, I never touch this topic before…
I am using 20MHz external crystal oscillator and I have it connected to CLKIN and CLKOUT and it is in parallel with two disc capacitor connected to ground. Thanks a lot.

10.02.11 **#49**
Comment by **rwb**.

The servo motor typically is designed with quite wide tolerance to the PWM frequency, therefore you should experiment with it.

11.02.11 **#50**
Comment by **pichaha**.

OK but I am afraid if the pulsewidth is overrange it will damage my servo. The PWM at min angle for my servo is 0.5ms and max angle is 2.35ms. So, 22ms or 44Hz is unable to drive servo right?

11.02.11 **#51**
Comment by **unni.0p**.

is the Processor fan of computer a servo motor?
can i use that for a servo motor?

11.02.11 **#52**
Comment by **rwb**.

@pichaha: As long as the supply power (i.e. polarity and voltage) is within the servo motor specification than your servo motor will be fine. The wrong PWM frequency only effecting the servo motor rotation.

@unni.0p: No its not a servo motor, it usually called brushless DC motor. The servo motor used in this project is a combination of geared DC motor with the electronic motor control circuit and position feedback mechanism inside.

12.02.11 **#53**
Comment by **pichaha**.

Hi,
I would like to ask, can the servo motor be driven using 5V voltage generated from 7805 voltage regulator??
If i am using 9V battery as the regulator input. Thanks a lot.

13.02.11 **#54**
Comment by **rwb**.

Yes, as long as you use the 5 volt servo motor specification

15.02.11 **#55**
Comment by **pichaha**.

Hi,
My servo is working already, the motor malfunction was caused by the power shortage of the supply. But I have met another question here…I want to let my servo to turn in clockwise direction, so I let the pwm signal to has 1.0ms pulse width, after that I put a delay loop to make it turn in a certain degree(I use push button to trigger), then I stop the motor, there are still other instructions after that. My problem is the rotation works only when my delay loop is long enough. When my delay loop is short, it sometime works and sometime not works. So how if i want my servo to move just a small ditance? Any suggestion? Thanks a lot.

16.02.11 **#56**
Comment by **rwb**.

Experiment, improve your delay and read button program routine

16.02.11 **#57**
Comment by **pichaha**.

What do you mean by improve my delay, do you mean prolong it? What if I just want my servo to move for a short distance(small degree)? Thanks a lot

16.02.11 **#58**
Comment by **rwb**.

In order to rotate the servo motor clockwise at certain degree you need to supply the 1 ms PWM at certain period of time. That's what I mean that you have to "experiment" with it (i.e. "delay loop"), because you need to know the require period before stop the PWM. The read button routine also could effecting this behavior (bounced effect).

27.02.11 **#59**
Comment by **Eric88**.

To: rwb

can you provide me a basic servo motor assembler code controlling with Microchip PIC Microcontroller ??
thanks…

27.02.11 **#60**
Comment by **rwb**.

No, I don't have the assembler code for this project.

05.03.11 **#61**
Comment by **milanivic**.

can you tell me, why output mesage error:
- undefined identifier "OPTION"
- undefined identifier "GODONE"

thanks

05.03.11 **#62**
Comment by **rwb**.

Make sure you install and configure the Microchip MPLAB IDE and Microchip HITECT PICC C compiler correctly.

06.03.11 **#63**
Comment by **milanivic**.

Thenks, I instal MPLAB IDE 8.63 and everythings working. Many thanks for yuor help!

12.03.11 **#64**
Comment by **Secky**.

Hi,

What a great and detail article you have, learn a lot from it. thank you.=)

Just wondering can I use this neat method to control more servos by creating another variable pulse_top2 and top_value2 and add in the following code?

```
if (pulse_top2 == top_value2) {
RC3=1; // Turn On RC3
}
```

13.03.11 **#65**
Comment by **rwb**.

Yes you could experiment with it and make some adjustment to the TIMER0 counter register (TMR0) initial value for the timing.

25.09.11 **#66**
Comment by **nerv**.

can i use a pic16f684 for this project

26.09.11 **#67**
Comment by **rwb**.

Yes you could try to use the 14-Pins Microchip PIC16F684 microcontroller. Just remember that PIC16F684 doesn't have the ANSELH register.

30.11.11 **#68**
Comment by **BF1Quang**.

Hi rwb, I'm Quang and I'm doing a project relating to RC servo, I use 6 RC servos to make a robot arm with 5 degrees of freedom! I can control the RC Servo now, I'm using 3 types of RC Servo (Hitec, Futaba and TowerPro). Unfortunately, I got problems with the power of the Servo and the ATmega, I intend to use 6V battery to power the RC Servo but it's maybe excess the limit voltage for the MCU. I'm really need your advice to solve that. What should I do, and can you give me a Schematic to wire the servo and the MCU. I mean if the

MCU and the Servo have the same power supply, they can operate smoothly, but when I'd separated them I got problems (The Servo didn't work!!!). I'm looking forward to your guide! Thank you very much!!

02.12.11 **#69**
Comment by **rwb**.

You could use two separate power supply i.e. 5 volt for the Microcontroller and 6 volt for the servo as long as they both share the same ground. The other option, you could use a diode (e.g. 1N4001/2) to reduce the 6 volt to about 5.3 volt for the Microcontroller. You could read more information about this subject on this following article:

Diode for reducing the voltage output

07.12.11 **#70**
Comment by **BF1Quang**.

Thank you so much, it's really really helpful, I used 7805! ^^

28.12.11 **#71**
Comment by **nerv**.

hye,

i have tried running the program using mplab ide, it works but when i use proteus to do the simulation it also works but there are around 6752 warning for this part :
while(GODONE) continue; // Wait for ldr_left conversion done
0×0057 – ADC conversion started before wait time has expired following previous conversion or channel change.
0×0062 – ADC conversion started before wait time has expired following previous conversion or channel change.

what can be done?or this this ok once i construct the real thing..

29.12.11 **#72**
Comment by **rwb**.

Yes, just construct the real project.

05.01.12 **#73**
Comment by **nerv**.

hye,

why i can't burn the code into my 16f690,im using pickit 2..

this error pop out :
PK2Error0027: Failed verify (Address = 0×0 – Expected Value 0×3000 – Value Read 0x3FFF)

tq.

06.01.12 **#74**
Comment by **rwb**.

Make sure you install the Microchip MPLAB IDE and connect your Microchip PICKit2 to the PIC16F690 microcontroller

Correctly.

10.01.12 **#75**
Comment by **nerv**.

i already construct my circuit,the problem is when i point the light to each of the ldr i can hear and see the motor moving CCW & CW but with very very little movement and i can hear it like moving foward and reverse when i point the light to the LDR.Why is this?

TQ..

10.01.12 **#76**
Comment by **nerv**.

and my white pin which is the signal is only around 0.5 volt, where else the requirement is 3.3v right?why is this please help me..tq.

10.01.12 **#77**
Comment by **rwb**.

Please check your servo operating voltage specification, make sure you recheck and use the same schematic/circuit as shown on the project.

10.01.12 **#78**
Comment by **nerv**.

i am using the same servo as this project..and the same circuit.my rc2 is giving 0.5v, and is this the problem, because this servo needs 3.2 to more right for the rc2 port.and 1 more thing can i share the voltage for the servo with other's like the LDR.

11.01.12 **#79**
Comment by **rwb**.

Remember the RC2 port output is a PWM (Pulse Width Modulation) signal, therefore you need to used an oscilloscope when you want to examine its output. Using DVM (Digital Volt Meter) only give you an average value not the pulse HIGH value. Yes you could share the voltage for the servo as shown on the schematic. Again recheck your circuit.

29.02.12 **#80**
Comment by **gechihoney**.

Hi your tutorials are really good.
I need to control the speed of a stepper motor, at the moment I'm using delays to simply vary the amount of time the windings of my motor are energised. But I want to know how to use PWM signals to control the motor instead. Pls help

01.03.12 **#81**
Comment by **rwb**.

The stepper motor used different method of rotating the rotor. You could use the TIMER to control the speed instead of the delay. For more information about using the stepper motor you could read this following article:

Using Transistor as a Switch

26.04.12 **#82**
Comment by **john**.

hye,
i try to use your program in my pic16f887 but it does not worked at all. Can you give me some suggestion or modification that can be made to fit to this pic16f887. please help me. sorry for my bad english.

01.05.12 **#83**
Comment by **rwb**.

You could read the comment #14 above.

09.06.12 **#84**
Comment by **pradeep_K_B**.

hi,can i control 2 or more servos using one 16f690???

11.06.12 **#85**
Comment by **rwb**.

Yes, you can and of course you need to modify this program.

12.06.12 **#86**
Comment by **fariskhan101**.

where do i change the configuration bit and the OSCCON for pic 16f8877a and change it into?thank you i really need help.im new to this…

12.06.12 **#87**
Comment by **fariskhan101**.

undefined identifier “OSCCON”
undefined identifier “ANSEL”
undefined identifier “ANSELH”
how can i solve this if i wanna use if for 16f877a

12.06.12 **#88**
Comment by **rwb**.

You need to change the configuration bit and the OSCCON register when you use PIC16F877A since the PIC16F877A microcontroller has no internal oscillator clock like PIC16F690. The PIC16F877A also don’t have the ANSEL and ANSELH registers.

13.06.12 **#89**
Comment by **fariskhan101**.

IF im using pic16f886,but i want to use a 4mhz external cyrstal how do i do so?

17.06.12 **#90**
Comment by **rwb**.

The default OSCCON setup is 4 MHz or you could use this

following statement:

OSCCON=0×60; // Select 4 Mhz internal clock (default)

For more information please read the Microchip PIC16F886 Microcontroller datasheet

18.06.12 **#91**
Comment by **fariskhan101**.

thank you,i read it but i cant really understand.so by changing this,im i still able to move this servo motor and achieve a signal if i use the oscilator?

20.06.12 **#92**
Comment by **rwb**.

Yes, you need to modify the TIMER0 timing in order to achieve the same signal result. Please read the article and the Microchip PIC16F886 microcontroller datasheet for more information.

12.08.12 **#93**
Comment by **iwestbury**.

Thank you very much. I was looking for a servo interface to a PIC. This is an excellent article and simply the best explanation I have found of creating a PWM pulse with the hardware timers.

Thanks again, it was extremely useful.

07.09.12 **#94**
Comment by **alex**.

Thank you very much, your tutorial is super good. However I have a problem when I implemented your "code" in MikroC. I got an error, for some reason MicroC doesn't recognize the following expression:

GODONE=1;

Could you help me solve this problem?
Thank you so much

07.09.12 **#95**
Comment by **rwb**.

In MikroC you should use GO_DONE_bit instead of GODONE in Microchip HI-TECH C. Or simply use the MikroC ADC_Read() ADC library to read the ADC value.

08.09.12 **#96**
Comment by **alex**.

Hi rwd,

Thank you very much, you are very helpful. Quick question, by any chance do you have any tutorials on ultrasound in arrays configuration?

thank you!!

08.09.12 **#97**
Comment by **rwb**.

No, I don't have it.

20.09.12 **#98**
Comment by **alex**.

Hello rwb
Question: I am attempting to implement your project, but I am getting the following error.

Error [192] C:\Documents and Settings\ale\Desktop \FIU\Senior\Codigos\infra-II\Infra.c; 92.1 undefined identifier "OPTION"
Error [192] C:\Documents and Settings\ale\Desktop \FIU\Senior\Codigos\infra-II\Infra.c; 100.1 undefined identifier "GODONE".

I am using MPLab IDEv 8.87, any recommendation!

20.09.12 **#99**
Comment by **rwb**.

Check your MPLAB IDE installation and make sure you select the proper Language Toolsuite (e.g. HI-TECH Universal Toolsuite), from MPLAB IDE menu Project -> Select Language Toolsuite.

22.09.12 **#100**
Comment by **alex**.

Thanks for your reply. i just added the following line of code "#define _LEGACY_HEADERS", and compiled without problem. Now I have another question, since I my board is an easyPIC board I am using the Mikroprog to download the " .HEX"(created in MPLab) file to the MICRO. It should work fine rigth?

22.09.12 **#101**
Comment by **rwb**.

Yes you could use the mikroProg to load the HEX file produced by the Microchip MPLAB.

23.10.12 **#102**
Comment by **mirza_sd**.

hey, please can you post a detail schematic of the circuit and the whole design. where i can find Jazzmate

25.10.12 **#103**
Comment by **rwb**.

The schematic is shown in the project. You could Contact the sales@ermicro.com regarding the JazzMate board.

30.10.12 **#104**
Comment by **mirza_sd**.

Hi, Instead of cardboard paper.

I was wondering if we install a PV panel of 4 to 6 kgs, to sense the sunlight.

would code be different from what is written and motor specification?

30.10.12 **#105**
Comment by **rwb**.

For that kind of load, you need to use a big torque servo motor for moving the PV Panel. About the code you need to check the servo motor control signal pulse specification, usually it use a similar control signal pulse as described in this project.

13.03.13 **#106**
Comment by **uscahmad**.

Hello rwb
Question: I am using pic18f4550 and Iam using MikroC pro and my simulation software is Proteus I have problem with the interrupt_routine the pic send me errors //stack overflow is forcing device reset //the code I am used is :
void interrupt_routine() org 0×04
{
…code truncated…
}
could you please help me to solve this problem ?
Thanks very much

Leave a Comment

You must be logged in to post a comment.