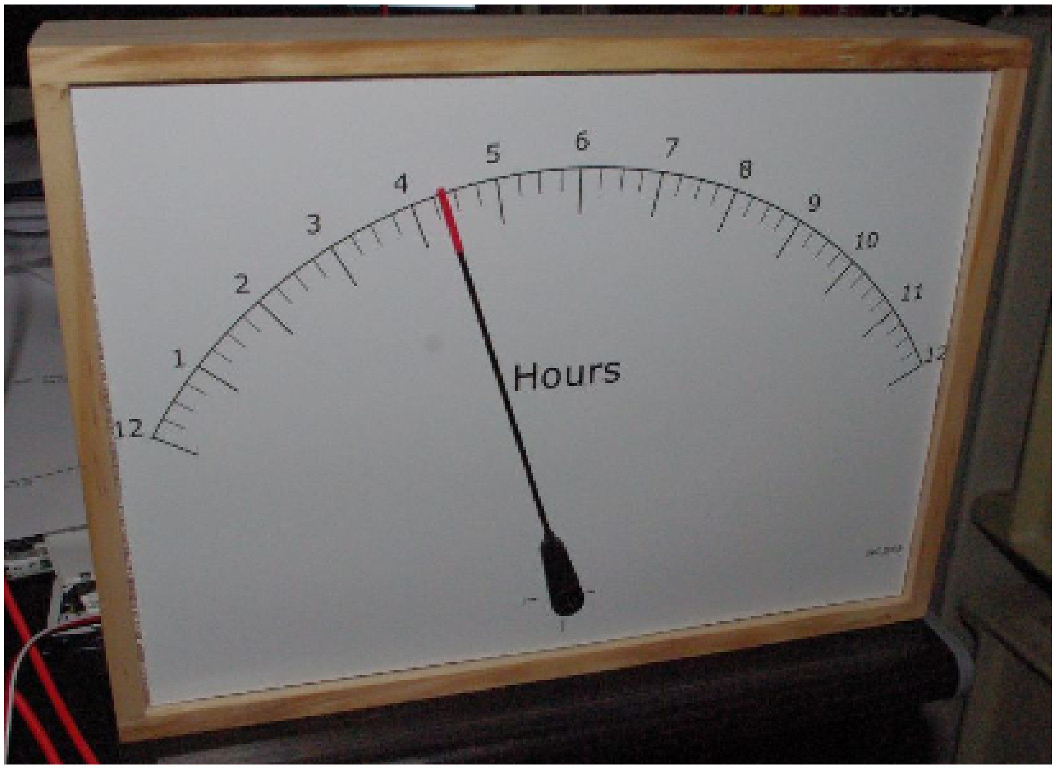
A Clock for Geeks

(0 Votes)

Written by Jon Chandler
Saturday, 19 December 2009 05:46

Clocks abound with all linds of off-beat styles. Here's a clock that appeals to geeks (and also wotks well for stealth clock watching) .



A Geeky Stealth Clock - The Time is about 4:20

The clock is patterned after an analog voltmeter. The meter pointer indicates the time on a meter scale marked in hours, with quarter divisions indicated. The pointer, driven by a servo motor, starts at 12 o'clock on the left, ending at 12 o'clock on the right. The meter pointer travels slowly to the right, indicating the current time. At 12 o'clock, the pointer rapidly transverses from right to left, restarting the process.

Graham has a great explantion of servos in the Proton section. Servos have been a mainstay in RC (radio control) cars and planes and more recently in robotics. Turns out they are pretty simple to use. Servos can be used in two ways; the traditional way (used here) is to turn through about 180 degrees to a known position to operate a control surface, or to turn a wheel for steering. Servos can also be modified for continuous rotation, perhaps for driving a robot. When modified for continuous rotation, the speed and direction is easy to control, but the postion of the servo can't be determined.

To build the clock, finding a suitable enclosure is the first step. The servo is about 1.25" tall, so some depth is needed. A deep picture frame might be a good option. I found a wood box used for a smoked salmon gift at the thrift store for this clock. Once the enclosure is selected, making the meter scale is the next step. I used a drawing program to create the scale. Depending on the enclosure, an arc of 90 degrees or 120 degrees works well. I divided the scale into 12 major increments for hours, with 4 minor increments showing quarter hours. I sized my drawing to fit on an 8" x 10" sheet, then printed it at a photo kiosk to get a nice glossy face. After trimming, I attached it to the enclosure with spray mount adhesive. A sample meter face for a 120 degree arc is here. A 90 degree face is here.

In this application, I used a standard type 4310 Futaba servo, which was about $10 at the local hobby store. The smallest min-servo would be suitable for this application as there is no force being transmitted. The pointer is a piece of carbon fiber rod also from the hobby store, although a piece of piano wire would work as well. The rod is attached to a servo arm (instead of the suppled disk) using some heat shrink tubing. A red piece of heat shrink was added for the pointer. The servo was mounted to the back of the enclsoure panel using double-faced tape. Position the hole for the servo carefully at the center of the arc to ensure accuracy.

The picture below shows the enclosure. The back side is used as the meter face.

The Enclosure

Graham's servo description covers the basics nicely. Swordfish doesn't have a specific servo module, but pulsing an output pin high and low with delay commands is fine for this application. The "on" time of the servo over a 20,000 usec period controls the angle of rotation. A 1,500 usec pulse width results in mid-scale rotation. Something around 1,000 usec reaches the extreme rotation to one side, about 2,000 usec is the maximum rotation in the other direction. It is important not to drive the servo too far - it has mechanical stops and something will break if driven too far.

The "calibration" step is about the most difficult part of the project. Write a simple program to pulse the output pin at 1,500 ms. Position the pointer on the servo to indicate approximately 6 o'clock at this position. The servo arm is splined so get it as close as possible but pointing at exactly 6 o'clock isn't required.

Next, pulse the servo at 2,000 usec to determine the end point position. In my case, the servo is made for counterclockwise rotation, so the "maximum" position is near the left end of of the scale. Adjust the pulse width until the pointer lines up exactly with the 12 o'clock position. Since this is a one-time operation, I just "brute forced" it and reloaded the program several times to determine the value. Repeat the process for the other end point. This scaling controls how well the pointer actually corresponds to the time, so get it as close as posible. Depending on the linearity of the servo, this may be all that's required for calibration. See the note on linearity below.

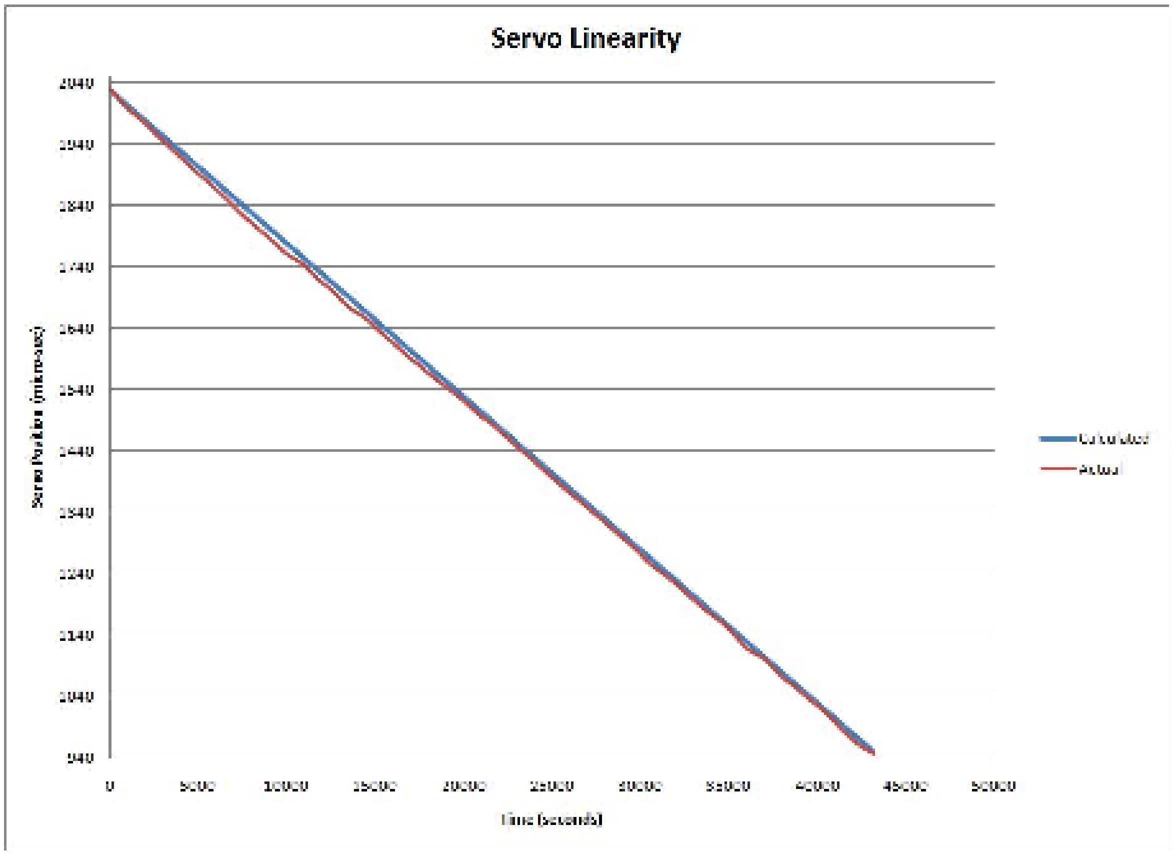
Time-keeping is based on Warren Schroeder's methods on the Swordfish web page, counting clock cycles. I modified Warren's fourth method in the following program. Modiciations were made to use a 20 MHz crystal instead of 8 MHz as used in the example. To acheive decent accuracy, an external crystal must be used; a PIC's internal oscillator will not be accurate enough for good time keeping. For increased accuracy, a real time clock chip could be used. The current time is updated once each second, which is overkill for this project. The servo resolution is approximately 1 minute using the DelayMS() command. The servo position is updated each second through a simple procedure. Since nothing else is happening in this program, no interrupts are needed to control the servo.

There are 42,300 seconds in 12 hours. Seconds are counted from zero starting at 12:00 until 42,300 is reached, when the counter is reset to zero. The fraction of seconds for 12 hours (current count / 43,200) is multiplied by the meter span to determine position. Note that integer math is used here and this fraction is less than 1, so it equates to zero in integer terms - the multiplication by meter span must be done before the division. This is totally unclear.

To power the clock, a 5 volt power supply was salvaged from a cell phone. The servo draws around 10 mA continuously, and peaks at about 100 mA during the brief period the servo is returning from one side to the other.

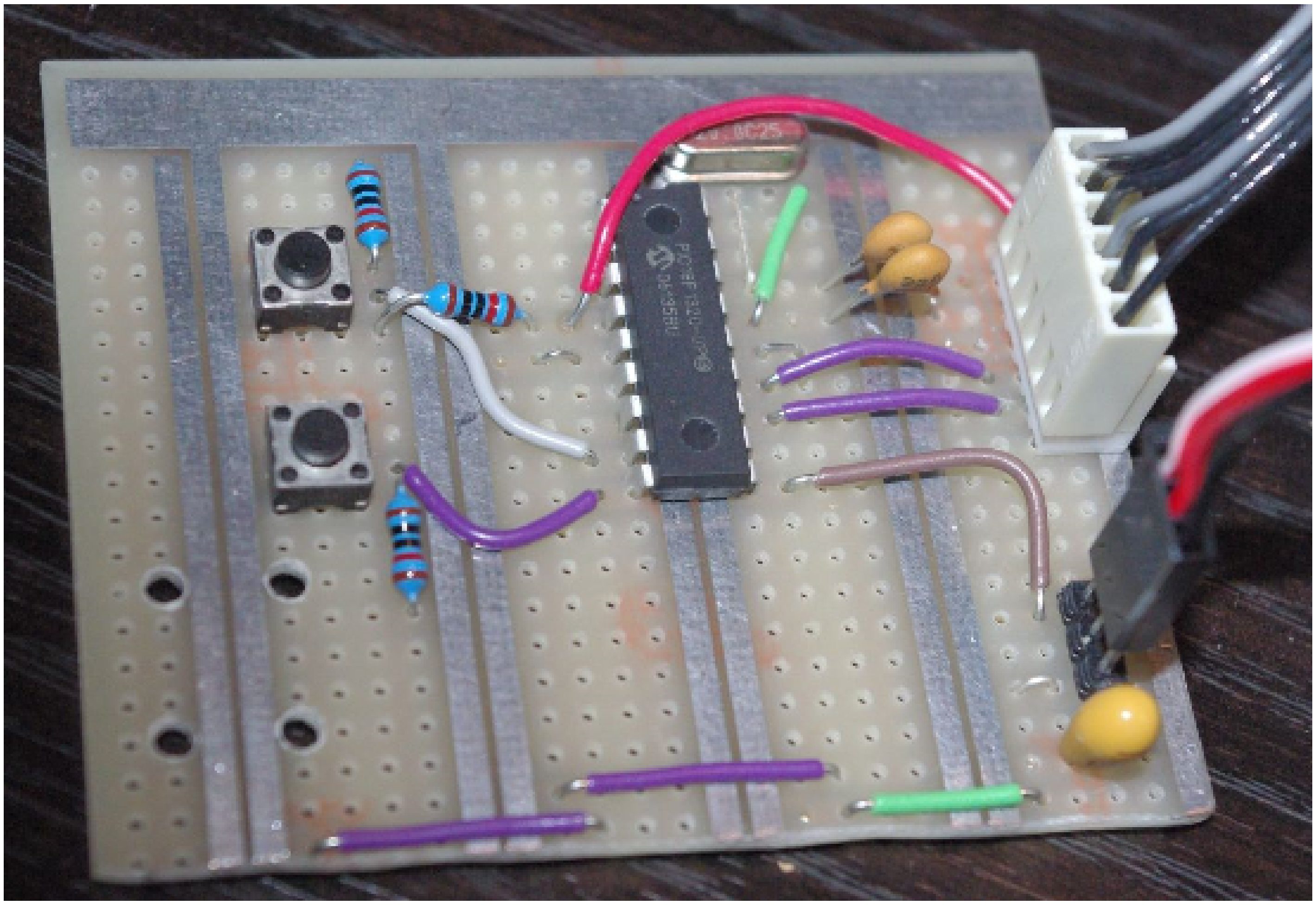
A Note About Linearity

My prototype tests were done using a meter arc with 90 degree span. The results acheived by calibrating the two end points were very good and the clock could be read with good accuracy. For the enclosure shown, I expanded the meter arc to 120 degrees. Turns out the servo is slightly non-linear over this range to the point that time readings were too far off. I made measurements at quarter-hour increments, comparing what looked right visually to calculated values. The results, shown below, aren't far off but it's enough to throw the results off. I used a simple scheme to linearize the data using a CASE SELCT command.



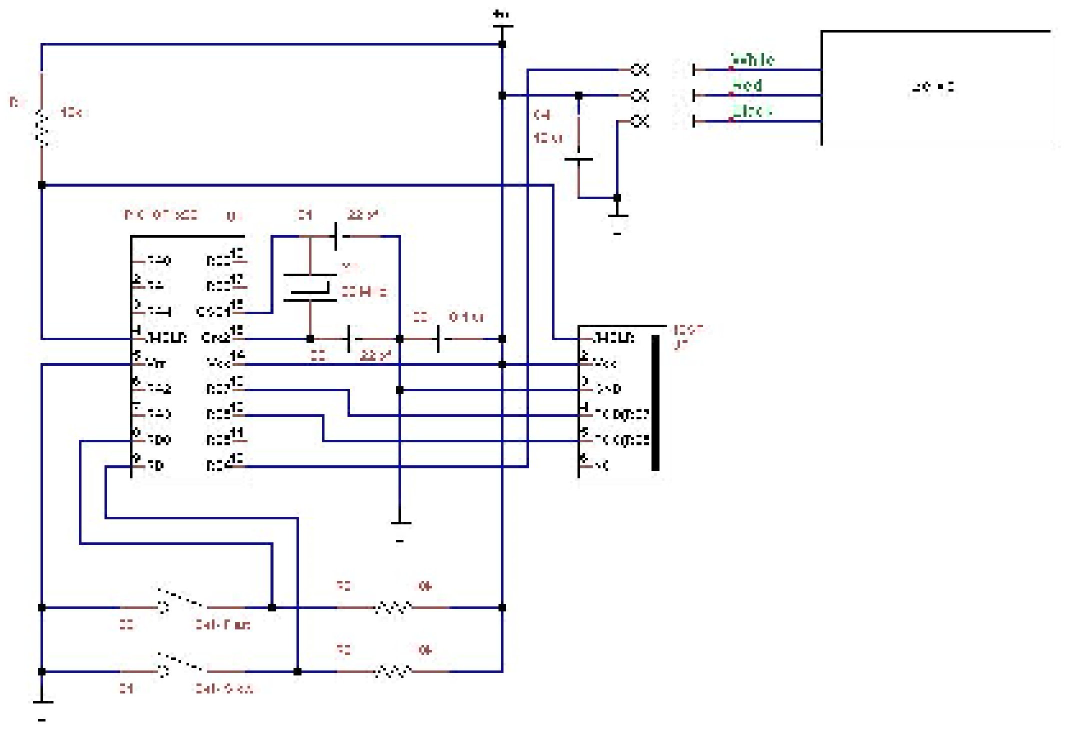
The Circuit

The clock runs on a PIC18F1320 (which I happened to have), hand-assembled on a pref board. A picture of the board and a schematic are shown below. Most PIC 18F series will work for this circuit - a proto-board with an 18F452 was used for code develpment, The two buttons are for setting the time. Because the servo has its own internal electronics, it has 3 connections for control signal, power and ground. No high-current drivers are needed by the circuit.

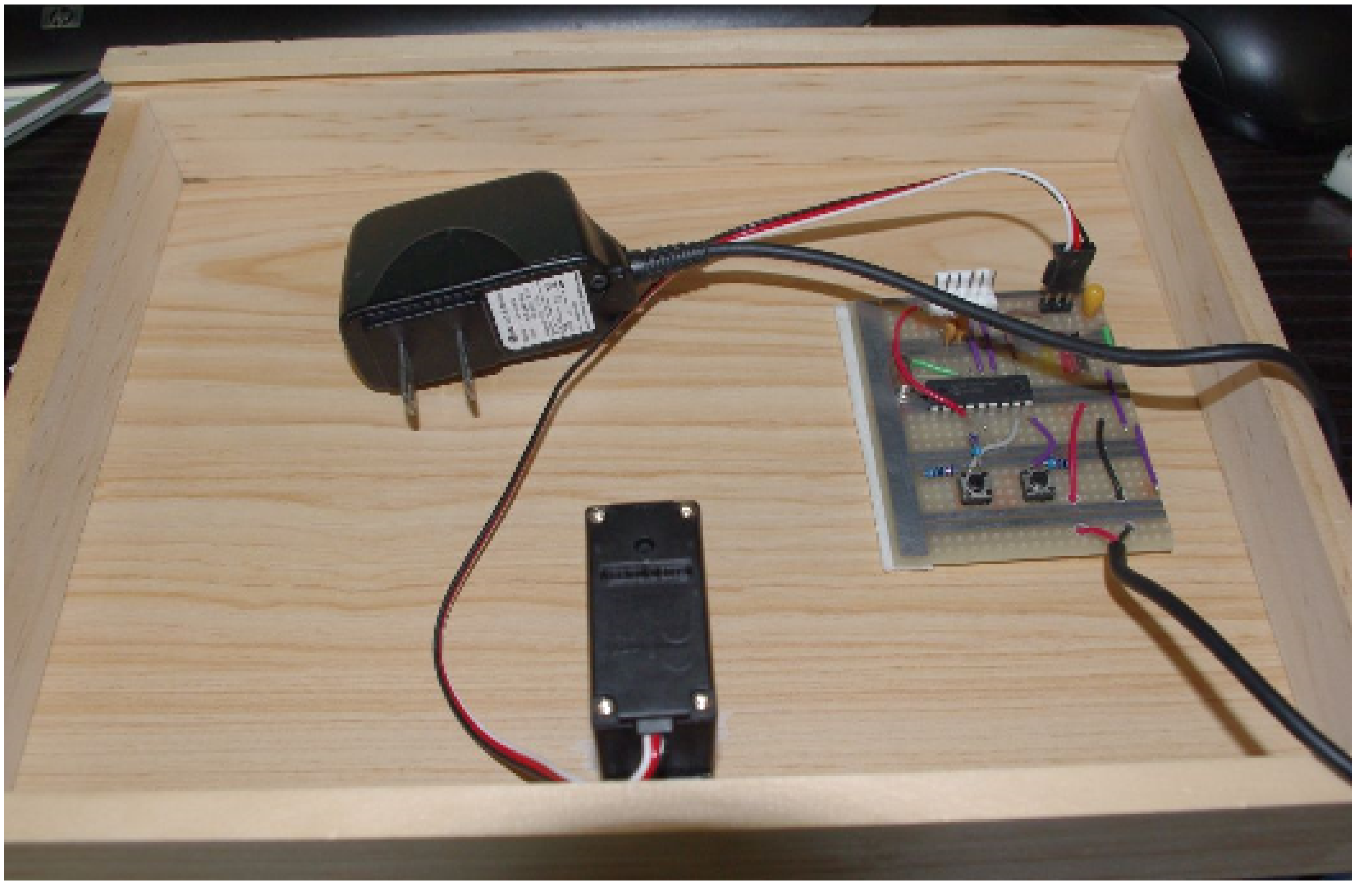


PIC18F1320 on Perfboard

The black 6 condictor wire at the upper right is from the PICKit 2 programmer. The 3 conductor wire is the servo cable. The power supply isn't hooked up yet - power is supplied by the PICKit2.



The Schematic



The Assembled Clock

Program Listing

```
{
*************************************************************************
*  Name    : Servo_Clock.bas                                            *
*  Author  : Jon Chandler                                               *
*  Notice  : Copyright (c) 2010 Jon Chandler                            *
*          : All Rights Reserved                                        *
*  Date    : 1/4/2010                                                   *
*  Version : 1.0                                                        *
*  Notes   : Servo-driven geeky clock with Real Time Clock              *
*          : fucntions based on Warren Schroeder's methods              *
*          : using PR2 Free Running Timer                               *
* http://www.sfcompiler.co.uk/wiki/pmwiki.php?n=SwordfishUser.SoftRTC *
*                                                                       *
* Presented to: http://digital-diy.com/  by Jon Chandler                *
*                                                                       *
* Timekeeping depends on a 20 MHz crystal                               *
*************************************************************************
}

Device = 18F1320
'Clock = 8
Clock = 20
Include "utils.bas"

Include ("utils.bas")


Dim Set_Fast As PORTB.0  'push buttons for setting time
Dim Set_Slow As PORTB.1

Dim Servo As PORTB.4


{
   For One Second Update:

   8MHz Fosc = 2MHz internal clock = 0.5us per cycle (timer count)
   Use 16-bit Timer1, No Prescaler
   Set CCPR1 = 50000; Timer1 resets on match every 50000 counts = 25000us
   Each Timer1 reset requires 1 cycle compensation... so set CCPR1 = 49999
   40 interrupts x 25000us each = 1 second
}

{
   For One Second Update: 20 MHz

   20MHz Fosc = 5MHz internal clock = 0.2us per cycle (timer count)
   Use 16-bit Timer1, No Prescaler
   Set CCPR1 = 50000; Timer1 resets on match every 50000 counts = 10000us
   Each Timer1 reset requires 1 cycle compensation... so set CCPR1 = 49999
   100 interrupts x 10000us each = 1 second
}


Dim C1 As Word Absolute $0FBE             ' CCPR1L + CCPR1H
Dim Int_Counter As LongWord
Dim update As Boolean
Dim secs,mins,hrs As LongWord
Dim Posit As Word

Interrupt RTC()
   Dec(Int_Counter)
   If Int_Counter = 0 Then
      Int_Counter = 100                     ' each interrupt = 10000us x 100 int's = 1 second
      update = true
   End If
   PIR1.2 = 0                             ' clear CCP1 interrupt flag
End Interrupt

Sub Clock24()
 Dim clk As String
    Inc(secs)
    If secs =  43200 Then                     ' check each tally for rollover
       secs = 0

    End If
    update = false

   Posit = 2029  - secs*(2029-950)/43200

   'Note: For the servo used, maxium rotation was counter-clockwise.  A time
   'of 2039 usec corresponds to the zero position.  A time of 950 usec corresponds
   'to the maximum position.  This must be determined by experimentation for each
   'servo used.
   '43,200 = number of seconds in 12 hours.

   'linearize output - Note: this may not be needed and depends on the sefvo used

   Select Posit

        Case > 2010
            Posit = Posit

        Case > 1960
            Posit = Posit - 7

        Case > 1870
            Posit = Posit -12

        Case > 1660
            Posit = Posit -18

        Case > 1560
            Posit = Posit -12

        Case > 1500
            Posit = Posit -9

        Case > 1180
            Posit = Posit -6

        Case > 0
            Posit = Posit -4

     End Select

End Sub

Sub Initialize() 'timekeeping routine
   ADCON1 = 15
   secs = 0
   mins = 0
   hrs = 0
   Int_Counter = 100
   update = false
   INTCON = 192                              ' enable GIE & PEIE
   T1CON = 0                                 ' no prescaler timer OFF
   TMR1H = 0                                 ' clear TMR1
   TMR1L = 0
   CCP1CON = 11                              ' enable special trigger event
   C1 = 49999                                ' set match value
   PIE1.2 = 1                                ' enable CCP1 interrupt
   PIR1.2 = 0                                ' clear CCP1 interrupt flag
   T1CON.0 = 1                               ' Timer1 ON
   Enable(RTC)                               ' enable jump to RTC ISR
End Sub

Sub settime()
    Clock24

    End Sub


     Posit = 0


   Initialize

      secs = 0 'initialize to 12:00.


   SetAllDigital

   While 1=1



    If update = true Then
         Clock24                             ' update 24H Clock output
    End If

    High(PORTB.4)          'create servo pulse corresponding to position and
    DelayUS(Posit)         'repeatedly send approximately every 200 milliseconds
    Low (PORTB.4)
    DelayUS(20000-Posit)

    'clock set procedure
    If Set_Fast = 0 Then   'increment seconds at 10x and free run
        secs = secs +9
        update = true
    EndIf

    If set_slow = 0 Then    'free run - i.e., don't wait for timer interrupt
        update = true
    EndIf



   Wend

```

Comments (0)	Add New	Delete All

Write comment

Your Contact Details:

Name:	Jon Chandler
Email:	automatic

Comment:

Title:

UBBCode: B I U

Message:

Send

Last Updated (Monday, 04 January 2010 00:00)