Reel Time

Contents

Program Code

```
' {$STAMP BS2}
' {$PBASIC 2.5}

'Reel TimeClock.
'11/03/08 - Error logic in case reels do not recalibrate
'10/12/08 - Tweaks to tighten and recalibrate code
'10/06/08 - Tighten rows 1,4 just on Recalibrate
' 9/28/08 - Force row-level recalibration when each row "loops"
' 9/21/08 - Fixes and Chime
' 9/19/08 - PowerSave mode
' 9/16/08 - LED code for all four rows
' 9/08/08 - LED emittor/detector logic; Recalibration and Tightening code (1st time worked!)
' 9/02/08 - Row calculations
' 8/29/08 - Got motors to turn according to target/current positions
' 8/26/08 - Bit logic for motor control
' 8/21/08 - Basic motor logic, fleshed out more button and time logic
' 8/13/08 - Time routines and button logic
' 8/07/08 - Basic setup and testing
' David Henshaw, 2008

GOSUB Declarations      'And Setup
GOSUB ForceRecalibrate  'In case of power failure, force all rows to recalibrate themselves

'Main code starts here
Main:
DO
  GOSUB Time_Check      'What time is it?

  IF mins <> oldMins OR hrs <> oldHrs THEN

    'Used later to see if it's time to party
    minscounter = minscounter + 1 'can let it loop after 255 - never need to set back to 0

    DO
      'Find out if we need to move the reels?
      GOSUB Determine_Target_Reel_Positions

      'If any of the reels are moving backwards, force a recalibrate and then loop back above to
reset reel positions
      RecalibrateFlag = NO

      FOR index2 = 0 TO 3
        IF Target_Reel_Position(index2) < Current_Reel_Position(index2) THEN
          RecalibrateFlag = YES
          RecalibrateReel(index2) = YES
        ENDIF
      NEXT

      IF RecalibrateFlag = Yes THEN GOSUB Recalibrate

    LOOP UNTIL RecalibrateFlag = NO

    GOSUB Move_Reels
    SEROUT 0, Baudmode, ["?g?y1?x00?lSELECT = menu"]

    'Is it top of the hour? Time to chime
    IF mins=0 THEN
      HIGH 15         'Ding
      PAUSE 100
      LOW 15          'Dong
```

```
    ENDIF

  ENDIF

  'Store values for comparison next time round
  oldhrs=hrs
  oldMins=mins

  'If Select button is pushed, enter menu system
  IF btnSelect = Pushed THEN GOTO Menu
  'If PowerSave button is pushed, enter Power Save
  IF btnPowerSave = Pushed AND PrevbtnPowerSave = NotPushed THEN GOTO PowerSave
  PrevbtnPowerSave = btnPowerSave

LOOP


'-----------------------------------------
Time_Check:

PAUSE 100
oldSecs=secs
GOSUB get_time           'Get the current time from the real-time clock

'Display the time on the LCD - just need to do once per second
IF secs <> oldSecs THEN   GOSUB show_time

RETURN


'-----------------------------------------
PowerSave:

GOSUB ForceRecalibrate
SEROUT 0, Baudmode, ["?y1?x00Power Save"]
PAUSE 1000

PowerLoop:

DO
  GOSUB Time_Check
LOOP UNTIL btnPowerSave = Pushed

DO
  PAUSE 5
LOOP UNTIL btnPowerSave= NotPushed

oldmins=$99 'Force redisplay of reels

GOTO Main


'-----------------------------------------
ForceRecalibrate:

FOR index = 0 TO 3
    RecalibrateReel(index) = YES 'Force each reel to recalibrate
NEXT

'keep on rolling...

'-----------------------------------------
Recalibrate:

'Move rows to position 0 and keep going until triggered by tape
```

```
'Set row position to 0
'Tighten rows 2 and 3

FOR index = 0 TO 3
  IF RecalibrateReel(index) = YES THEN
    Current_Reel_Position(index)=15 '15 is the max allowed
    Target_Reel_Position(index)=0   'Want to force to the beginning
    'SEROUT 0, Baudmode, ["?y2?x00Recalibrate"]
  ENDIF
NEXT

GOSUB Move_Reels

RETURN


'----------------------------------------
Pulse_Motors:

PULSOUT Latch_595, 5 'Latch outputs
Pulse = ~ Pulse      'Flip the pulse flag
PAUSE 50
RETURN


'----------------------------------------
Determine_Target_Reel_Positions:

'Reel 1 (Top): General message e.g. "It's almost"
'Reel 2 (2nd): Minutes indicator e.g. "Quarter past"
'Reel 3 (3rd): Hour e.g. "Midnight", "2 o'clock"
'Reel 4 (Bottom): AM/PM/etc indicator e.g. "In the afternoon"

'Reel 1
SELECT mins DIG 0 'Just the last digit
  CASE 3,4,8,9
    index = 1
  CASE 1,2,6,7
    index = 3
  CASE ELSE
    index = 2
ENDSELECT

Target_Reel_Position(0) = index


'Reel 2
Target_Reel_Position(1) = ((mins + 2) / 5)
IF mins < 3 THEN Target_Reel_Position(1) = 12 'Forces "exactly" for the first few minutes of the
hour

'Reel 3
'Stores hrs in index3 and setup correct hours phrase
index3 = hrs
IF mins > 32 THEN index3 = index3 + 1 'Want to say it's coming up to a new hour
IF index3 > 23 THEN index3 = 0 ' Reset in case it's almost midnight

SELECT index3
  CASE 0
    ' "midnight"
    index=12
  CASE 12
    ' "midday"
    index=13
  CASE ELSE
```

```
    ' hrs = index position for 1 thru 11 and 13 thru 23
    index =  index3
    IF index > 11 THEN index = index - 12
ENDSELECT

Target_Reel_Position(2) = index


'Reel 4
SELECT index3
  CASE 0 TO 6
    ' "Early morning"
    index = 1
  CASE 7 TO 11
    ' "In the morning"
    index = 2
  CASE 12 TO 17
    ' "In the afternoon"
    index = 3
  CASE 18 TO 21
    ' "In the evening"
    index = 4
  CASE ELSE
    ' "At night"
    index = 5
ENDSELECT

Target_Reel_Position(3) = index

'RANDOM "TIME TO PARTY" message
'If evening or night and minscount between 100 and 102 (inclusive)
'Three minutes allows for all the reels to move into place
SELECT minscounter
  CASE 100 TO 103
    SELECT index3
      CASE 18 TO 23
        Target_Reel_Position(0) = 4    'It's time
        Target_Reel_Position(1) = 13   'to party
        Target_Reel_Position(2) = 14   '[blank]
        Target_Reel_Position(3) = 0    'quote
        'DEBUG "@TP"
    ENDSELECT
ENDSELECT

RETURN

'----------------------------------------
Move_Reels:

'Print target reel positions
DEBUG CR, DEC2 hrs, ":", DEC2 mins
SEROUT 0, Baudmode, ["?y3?x00"]
FOR index = 0 TO 3
  DEBUG " ", DEC2 Target_Reel_Position(index)
  SEROUT 0, Baudmode, [DEC2 Target_Reel_Position(index), "."]
NEXT


'Beginning of loop
DO

  'Reset flags
  FOR index1 = 0 TO 3
    Enab_Flag(index1)      = 1     'Do not enable (move) row
    Direction_Flag(index1) = 0     'Default direction
```

```
    RecalibrateRow(index1) = 0     'Reset recalibrate flag
  NEXT

  Keep_Turning = NO

  'Some reels have to move more than others, and in different directions
  'Calculate for each row...
    FOR index2 = 0 TO 3
      'Does this row need to change?

      IF Target_Reel_Position(index2) <> Current_Reel_Position(index2) THEN

        'Yes, this row needs to change.
        Keep_Turning = YES 'Flag used later to see if we need to keep turning

        'Enab bit must = 0 to allow motors to change
        Enab_Flag(index2) = 0


        'Next, which direction?
        IF Target_Reel_Position(index2) > Current_Reel_Position(index2) THEN
          Current_Reel_Position(index2) = Current_Reel_Position(index2) + 1
           RecalibrateRow(index2)=1      'Tighten row at end of this move (only when going
forward)
        ELSE
          Current_Reel_Position(index2) = Current_Reel_Position(index2) - 1
           'Presume direction bit must = "1" - already defaults to 0
          Direction_flag(index2) = 1
        ENDIF

      ENDIF
    NEXT

  IF Keep_Turning = YES THEN 'Begin to activate motors

    SEROUT 0, Baudmode, ["?y1?x00?lMoving..."]

    'Do not tighten Rows 1 and 4 - they do not end up with much slack
    IF RecalibrateFlag = NO THEN
      RecalibrateRow(0)=0
      RecalibrateRow(3)=0
    ENDIF

    'Loop that turns the motors
    FOR index2 = 1 TO length

      'Have we reached the start of each row yet?
      IF Row1Begin = YES AND PrevRow1Begin= NO THEN
        Enab_Flag(0) = 1             'Do not move this row any more
        Current_Reel_Position(0) = 0 'We are at position 0
        ReachedBeginning(0) = YES    'We have seen the black marker
      ENDIF
      IF Row2Begin = YES AND PrevRow2Begin= NO THEN
        Enab_Flag(1) = 1
        Current_Reel_Position(1) = 0
        ReachedBeginning(1) = YES
      ENDIF
      IF Row3Begin = YES AND PrevRow3Begin= NO THEN
        Enab_Flag(2) = 1
        Current_Reel_Position(2) = 0
        ReachedBeginning(2) = YES
      ENDIF
      IF Row4Begin = YES AND PrevRow4Begin= NO THEN
        Enab_Flag(3) = 1
        Current_Reel_Position(3) = 0
```

```
        ReachedBeginning(3) = YES
      ENDIF

      PrevRow1Begin = Row1Begin
      PrevRow2Begin = Row2Begin
      PrevRow3Begin = Row3Begin
      PrevRow4Begin = Row4Begin

      'Prep one pulse string (a list of bits) to the 595 shift registers
      SHIFTOUT DataOut_595, Clock_595, MSBFIRST,
          [Enab_Flag(3)\1, Direction_Flag(3)\1, Pulse\1,
           Enab_Flag(3)\1, Direction_Flag(3)\1, Pulse\1,
           Enab_Flag(2)\1, Direction_Flag(2)\1, Pulse\1,
           Enab_Flag(2)\1, Direction_Flag(2)\1, Pulse\1,
           Enab_Flag(1)\1, Direction_Flag(1)\1, Pulse\1,
           Enab_Flag(1)\1, Direction_Flag(1)\1, Pulse\1,
           Enab_Flag(0)\1, Direction_Flag(0)\1, Pulse\1,
           Enab_Flag(0)\1, Direction_Flag(0)\1, Pulse\1]

      'Send the pulse string
      GOSUB Pulse_Motors

    NEXT


    'Do we need to tighten any rows?
    TightenFlag = NO

    FOR index1 = 0 TO 3
      IF RecalibrateRow(index1) = 1 THEN TightenFlag = YES
    NEXT

    IF TightenFlag = YES THEN

    'Tighten the rows IF we've moved forward
    'Uses ~ (NOT) because 0=turn, 1=do not turn ... opposite of the recalibrate flag from earlier
     FOR index1= 1 TO tighten

        'Tighten reels in opposite directions
         SHIFTOUT DataOut_595, Clock_595, MSBFIRST,
               [~RecalibrateRow(3)\1, 0\1, Pulse\1,
                ~RecalibrateRow(3)\1, 1\1, Pulse\1,
                ~RecalibrateRow(2)\1, 0\1, Pulse\1,
                ~RecalibrateRow(2)\1, 1\1, Pulse\1,
                ~RecalibrateRow(1)\1, 0\1, Pulse\1,
                ~RecalibrateRow(1)\1, 1\1, Pulse\1,
                ~RecalibrateRow(0)\1, 0\1, Pulse\1,
                ~RecalibrateRow(0)\1, 1\1, Pulse\1]

      GOSUB Pulse_Motors

     NEXT

    ENDIF

  ENDIF

LOOP UNTIL Keep_Turning = NO

'Disable_Motors:
'Set enab to HIGH so motor does not overheat
'old:
[1\1,0\1,0\1,1\1,0\1,0\1,1\1,0\1,0\1,1\1,0\1,0\1,1\1,0\1,0\1,1\1,0\1,0\1,1\1,0\1,0\1,1\1,0\1,0\1,1\1,0\1,0\1]
'actual bits for 595: 10010010 01001001 00100100 = 146,73,36
SHIFTOUT DataOut_595, Clock_595, MSBFIRST, [146, 73, 36]
```

```
GOSUB Pulse_Motors

'Clear the "Recalibrate" line
'SEROUT 0, Baudmode, ["?y2?l"]

'Check to see if we didn't see the beginning of reel marker
ReelError = NO

FOR index1 = 0 TO 3
  'DEBUG CR,"@RE ", DEC2 index1, " ", DEC2 RecalibrateReel(index1), " ", DEC2
ReachedBeginning(index1)
  IF RecalibrateReel(index1) = YES AND ReachedBeginning(index1) = NO THEN ReelError = YES
  RecalibrateReel(index1) = NO  'Reset
NEXT

IF ReelError = YES THEN
    'Print Error Message
    SEROUT 0, Baudmode, ["?y1?x00ERROR - CALL ARTIST"]
    'Go to PowerSave logic (press button to "reset" the reels)
    GOTO PowerLoop
ENDIF

RETURN


'----------------------------------------
Menu:

'The "Select" Button has been pressed
index=0 'reset this counter

DO

  PAUSE 750

  IF btnSelect = Pushed THEN index = index + 1
  IF index > 2 THEN index = 0

  SELECT index
    CASE 0
        SEROUT 0, Baudmode, ["?y1?x00Hours   (+/-)"]
        IF btnForward = Pushed  THEN hrs = hrs + 1
        IF btnBackward = Pushed THEN hrs = hrs - 1
    CASE 1
        SEROUT 0, Baudmode, ["?y1?x00Minutes (+/-)"]
        IF btnForward = Pushed  THEN mins = mins + 1
        IF btnBackward = Pushed THEN mins = mins - 1
    CASE 2
        SEROUT 0, Baudmode, ["?y1?x00?lExit (+)"]
        IF btnForward = Pushed  THEN GOTO Main
  ENDSELECT

  GOSUB Set_Time
  GOSUB Get_Time
  GOSUB Show_Time

LOOP

'----------------------------------------
RTC_Out: ' send ioByte to reg in DS1302
  HIGH CS1302
  SHIFTOUT DataIO, Clock, LSBFIRST, [reg, ioByte]
  LOW CS1302
RETURN
```

```
'----------------------------------------
RTC_In: ' read ioByte from reg in DS1302
  HIGH CS1302
  SHIFTOUT DataIO, Clock, LSBFIRST, [reg]
  SHIFTIN DataIO, Clock, LSBPRE, [ioByte]
  LOW CS1302
RETURN

'----------------------------------------
Get_Time: ' read data with burst mode
  HIGH CS1302
  SHIFTOUT DataIO, Clock, LSBFIRST, [RdBurst]
  SHIFTIN DataIO, Clock, LSBPRE, [secs, mins, hrs, work, work, work]
  LOW CS1302

  'Convert time in BCD to Decimal
  hrs = (hrs10 * 10) + hrs01
  mins = (mins10 * 10) + mins01
  secs = (secs10 * 10) + secs01
RETURN

'----------------------------------------
Show_Time:
  'Send to LCD
  SEROUT 0, Baudmode, ["?y0?x00Time: ", DEC2 hrs, ":", DEC2 mins, ":", DEC2 secs]
RETURN

'----------------------------------------
Set_Time: ' write data with burst mode

'Need to manage rollover from 60 secs and mins to 0 and vice versa
IF hrs = 255 THEN hrs = 23
IF hrs > 23 THEN hrs = 0
IF mins = 255 THEN mins = 59
IF mins > 59 THEN mins = 0

'CONVERT DECimal to BCD
'bcdbyte = (binary DIG 1) << 4 + (binary DIG 0)

hrs = (hrs DIG 1) << 4 + (hrs DIG 0)
mins = (mins DIG 1) << 4 + (mins DIG 0)
secs = (secs DIG 1) << 4 + (secs DIG 0)

HIGH CS1302
SHIFTOUT DataIO, Clock, LSBFIRST, [WrBurst]
SHIFTOUT DataIO, Clock, LSBFIRST, [secs, mins, hrs, 0, 0, 0, 0, 0]
LOW CS1302

RETURN

'----------------------------------------
Declarations:
'Pin    5432109876543210
DIRS = %1000001110001111' Pins: 0 = In, 1 = Out (Read Right to Left!)
'Pin 0 = LCD
'Pin 1,2,3 = DS1302
'Pin 4,5,6 = Buttons (In)
'Pin 7,8,9 = 74HC595 clock, data, latch (out)
'Pin 10,11,12,13 = LED detector Row 1,2,3,4 (In)
'Pin 14 = Button (In)
'Pin 15 = Chime

' -----------------------------------------------------------------------------
' I/O Definitions
' -----------------------------------------------------------------------------
```

```
DataIO CON 2      ' DS1302.6
Clock CON 1       ' DS1302.7
CS1302 CON 3      ' DS1302.5
Clock_595 CON 7   ' shift clock (74HC595.11)
DataOut_595 CON 8 ' serial data out (74HC595.14)
Latch_595 CON 9   ' output latch (74HC595.12)

' -----------------------------------------------------------------------------
' Constants
' -----------------------------------------------------------------------------
WrSecs CON $80 ' write seconds
RdSecs CON $81 ' read seconds
WrMins CON $82 ' write minutes
RdMins CON $83 ' read minutes
WrHrs CON $84 ' write hours
RdHrs CON $85 ' read hours
CWPr CON $8E ' write protect register
WPr1 CON $80 ' set write protect
WPr0 CON $00 ' clear write protect
WrBurst CON $BE ' write burst of data
RdBurst CON $BF ' read burst of data
WrRam CON $C0 ' RAM address control
RdRam CON $C1
Yes CON 1
No CON 0
Pushed CON 1
NotPushed CON 0
length CON 136 'how long to move reels
tighten CON 28 'tried 16, 32


' -----------------------------------------------------------------------------
' Variables
' -----------------------------------------------------------------------------
index VAR Byte ' general counters
index1 VAR Byte
index2 VAR Byte
index3 VAR Byte
minscounter VAR Byte 'Used to count so we can display "Time to party" message

'DS1302 variables
  reg VAR Byte ' DS1302 address to read/write
  ioByte VAR Byte ' data to/from DS1302
  secs VAR Byte ' seconds
  secs01 VAR secs.LOWNIB
  secs10 VAR secs.HIGHNIB
  mins VAR Byte ' minutes
  mins01 VAR mins.LOWNIB
  mins10 VAR mins.HIGHNIB
  hrs VAR Byte ' hours
  hrs01 VAR hrs.LOWNIB
  hrs10 VAR hrs.HIGHNIB
  work VAR Byte ' work variable for display output

'Arrays to store current and next/target logical positions of each reel
Current_Reel_Position VAR Nib(4)
Target_Reel_Position VAR Nib(4)
Enab_Flag VAR Bit(4)
Direction_Flag VAR Bit(4)
RecalibrateRow VAR Bit(4) 'Used to figure out if a particular row needs to get tightened (not
recalibrated)
TightenFlag VAR Bit 'Used to figure out if we need to bother tightening
ReachedBeginning VAR Bit(4) 'Used to flag when reel has returned to beginning of roll
RecalibrateReel VAR Bit(4) 'Used to figure out if a particular row needs to get recalibrated
Pulse VAR Bit
```

```
Keep_Turning VAR Bit
ReelError VAR Bit 'Used to signal if there is an error

oldSecs VAR Byte ' previous seconds value
oldhrs VAR Byte ' previous hours value
oldMins VAR Byte ' previous minutes value

'Buttons and switches correspond to following bits (i.e. pins) on the Basic Stamp
  btnSelect VAR IN4
  btnForward VAR IN5
  btnBackward VAR IN6
  Row1Begin VAR IN10
  Row2Begin VAR IN11
  Row3Begin VAR IN12
  Row4Begin VAR IN13
  btnPowerSave VAR IN14

'To support detecting beginning of each row
  PrevRow1Begin VAR Bit
  PrevRow2Begin VAR Bit
  PrevRow3Begin VAR Bit
  PrevRow4Begin VAR Bit
  PrevbtnPowerSave VAR Bit
  RecalibrateFlag VAR Bit


'Roll into Setup:

'----------------------------------------
Setup:

LCDSetup:

  BaudMode CON 84   ' 9600 baud, non-inverted

   HIGH 0  ' this is the steady state for non-inverted
   PAUSE 2000  ' Allow to stabilize

'   SEROUT 0, Baudmode, ["?G420"] ' configure LCD geometry
'   PAUSE 200  ' pause to allow LCD EEPROM to program

'   SEROUT 0, Baudmode, ["?B40"] ' set backlight to 40 hex
'   PAUSE 200    ' pause to allow LCD EEPROM to program

ClockSetup:

  reg = CWPr ' clear write protect register
  ioByte = WPr0

  LOW Latch_595 ' make output and keep low

  GOSUB RTC_Out
  hrs = 01 ' preset time
  mins= 03 ' preset minutes
  secs= 05
  GOSUB Set_Time

RETURN
```

Creative Commons License Deed

Attribution-Noncommercial-Share Alike 3.0 Unported

You are free:

- to Share — to copy, distribute and transmit the work
- to Remix — to adapt the work

Under the following conditions:

- Attribution. You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).
- Noncommercial. You may not use this work for commercial purposes.
- Share Alike. If you alter, transform, or build upon this work, you may distribute the resulting work only under the same or similar license to this one.

For any reuse or distribution, you must make clear to others the license terms of this work. The best way to do this is with a link to this web page.

Any of the above conditions can be waived if you get permission from the copyright holder.

Nothing in this license impairs or restricts the author's moral rights.