CamTimer

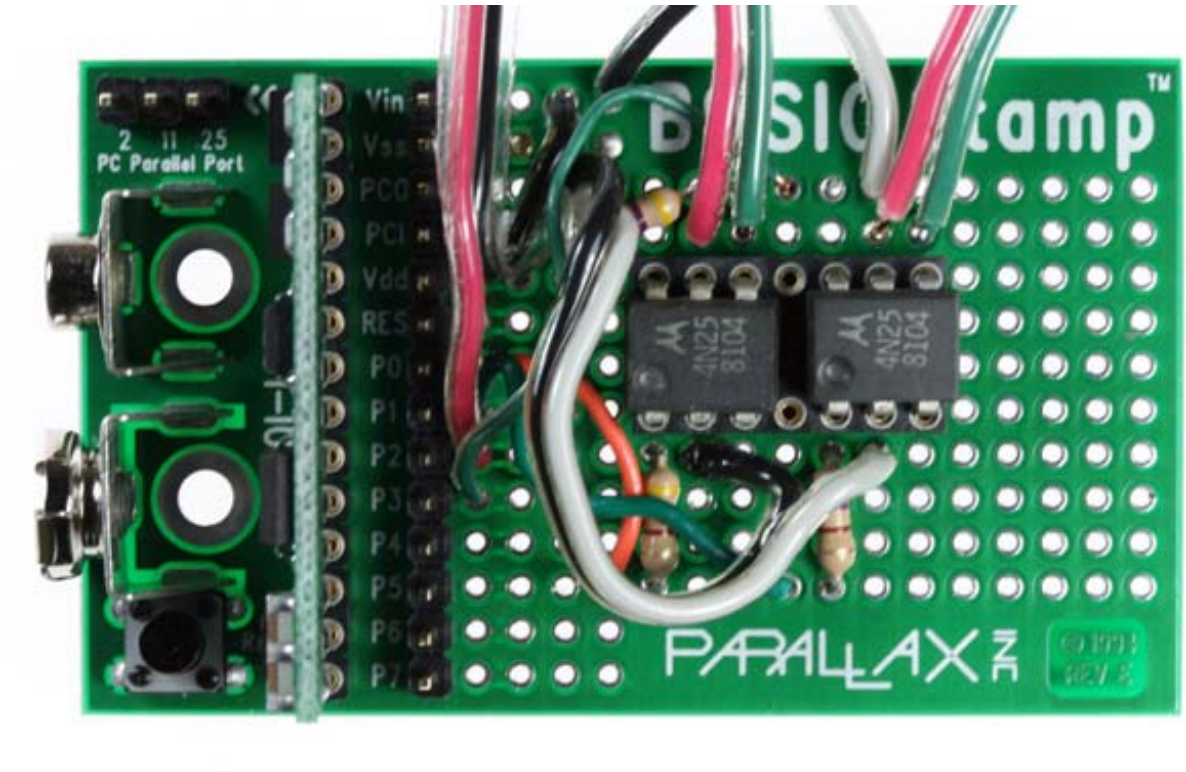

Figure 1. The complete CamTimer on a BS1 carrier board.

This project started with the idea of a creating a systematic sequence of photographs; of a sunset or maybe a sequence of shots showing a rose opening. Maybe it was the images of those cool time-lapse movies of a bean sprout coming up out of the soil that I saw as a kid. Since I enjoy putting together little electronic projects and I have a little-used but extensive junk box, I nourished the idea. The result is the little device described here – my first Stamp project.

Figure 2. The completed project in a black box case.

The CamTimer will repeatedly operate my Canon Digital Rebel XTi at a preprogrammed time interval. The device is based on the Basic Stamp 1 (BS1) and uses the carrier board. It runs on batteries, contains only a few components and is very easy to build. The program is pretty straightforward and there is still a lot of memory left for modifications.

Figure 3. Remote cable for connection to the camera.

The camera has a jack that accommodates a remote switch box that you can purchase or build. It is through that jack that the CamTimer will control the camera. You will need to make a cable with a 2.5 mm stereo plug. That cable will terminate in three wires. One wire is the ground; the other two will operate the autofocus (AF) and shutter, respectively, when connected to the cable ground wire. You will need to do your own testing so that you can correctly identify the leads.

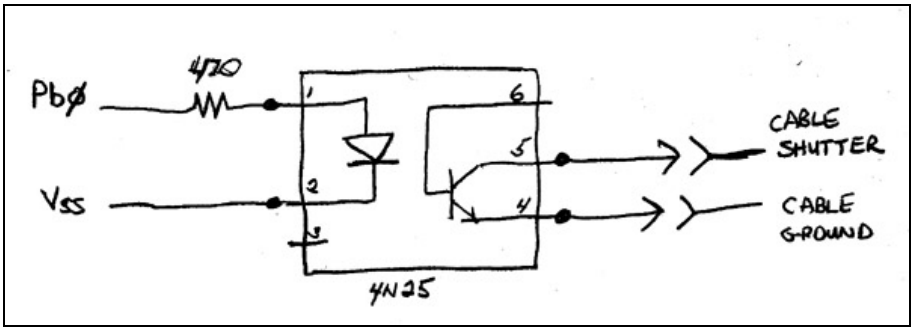

Figure 4. The optoisolator control circuit.

To computerize the camera AF and shutter functions, I used a simple optoisolator circuit. If you have never used one of these, they are very handy way of control the outside world while maintaining some isolation. On one side of the integrated circuit (IC) is an infra-red LED. This is connected to the BS1's port bit 0 with a 470 ohm resistor and, of course the other end of the LED is connected to the BS1 ground. This is exactly the method you typically use to control visible LEDs. The other side of the IC is an NPN transistor. We use that transistor as a switch to connect the shutter operate end of the cable to the ground end of the cable. Now, turning on the BS1's port bit 0 will operate the LED which will provide a base current to the NPN transistor which will make the switch connection and operate the shutter. The exact same design is duplicated for the AF function using an additional optoisolator and the BS1's port bit 1.

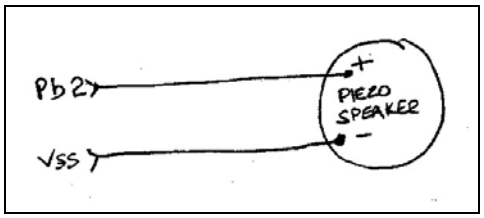

Figure 5. The piezo speaker hookup.

A third output bit is used to operate a piezoelectric speaker. The one from Parallax works great for me. Here we connect the BS1's port bit 2 to the + terminal on the speaker and the Vss to the – terminal. The speaker is used to announce power up (one beep), a start button press (two beeps) and gives a warning signal (three beeps) before operating the camera.

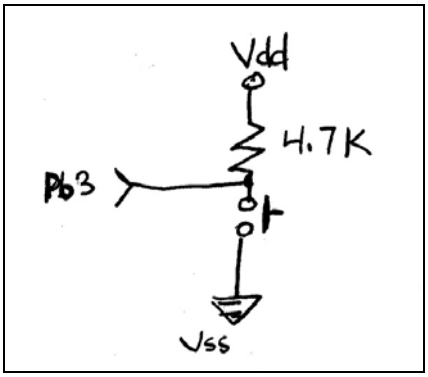

Figure 6. The pull-up input circuit for the start switch.

The start button is a simple momentary switch. A pull-up resistor is connected to Vdd with the other side connected to the BS1's port bit 3. The momentary switch connects to Vss, so the bit will be read high when not pressed and low when pressed.

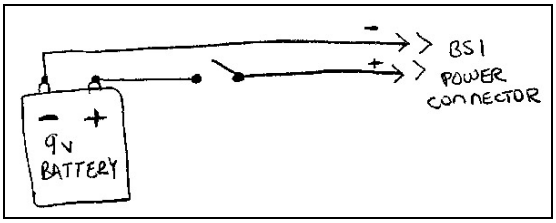

Figure 7. Power switch hookup.

Finally, a single pole toggle switch is wired in series with the + line of the battery power. Flipping the switch powers up the unit and makes it easy to save battery life.

Operating the CamTimer is simple and straightforward. You program a delay value using the Parallax Editor / Development software and configure a few software switches then connect the unit to the camera and you are all set. When you want to use a different delay you simply program another value.

Figure 8. Front view of the CamTimer showing the switches.

Looking at the code, you can see that two pairs of delay values can be used - Sdelay1 and Sdelay2 or Ldelay1 and Ldelay2. Pairs of delay values are used to increase the total possible delay. They run sequentially, so the program will delay for the amount specified by the first delay value and then delay for the amount of the second delay value. Sdelay uses the PAUSE command and stays at full power. Ldelay uses the SLEEP command and goes into low-power mode while continuing to count. Sdelay times in milliseconds and offers the best accuracy of the two but has a maximum delay of about 2.2 minutes (setting both values to 65535). Ldelay times in seconds and has a maximum count of over 36 hours (setting both values to 65535).

Figure 9. The assembled and wired circuit.

There's a few other software switches to set. The binary variable 'dtype', tells the program which of the two delay types you want to use (0-Sdelay and 1=Ldelay). The binary variable 'repeat' determines whether the camera will be operated once (repeat=0) or will recycle and keep operating after every delay cycle (repeat=1). The binary variable 'silence' will determine whether the warning beeps before the shutter operates are used (0= no warning beeps and 1= warning beeps). Once the delay pairs and these three variables are set, the CamTimer is ready to work.

If you use this scheme for taking time lapse photographs over extended periods of time, make sure you have the settings on the camera such that the least amount of power is used (e.g., don't display preview pics). I am anticipating that a fully charged camera battery and a fresh 9 volt for the BS1 will last for an extended amount of photographing – but I will find out as I use the device.

The code listing is below and it should be pretty easy to follow. In total, only half the code space is used up and there are a few spare port bits and variables. So, there's plenty of room for additions and modifications.

I will try to post some pics that turn out well.

Enjoy.

```
'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
' CamTimer  05-01-08 RFG
' Basic Stamp 1 <-> Digital Rebel XTi
' Timer Camera Operation
'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
' directives Basic Stamp Editor 2.4 from Parallax
' {$STAMP BS1}
' {$PORT COM2}
'
'
' Output Bits
' bit 0 = Optoisolator for shutter
' bit 1 = Optoisolator for autofocus
' bit 2 = Piezo speaker
```

```
'
' Input bits
' bit 3 = Start switch
'
' bit 4, 5, 6, 7 are unused
'
' Dedicated variables
' b0 - bit status values for:
'   Bit1 = dtype (delay type - either normal Sdelay or sleep Ldelay)
'   Bit1 = repeat (switch for single or continuous operation)
'   Bit2 = silence (turns shutter warning beeps on/off)
'
'   W1 = Sdelay1
'   W2 = Sdelay2
'   W3 = Ldelay1
'   W4 = Ldelay2
'
' B1, B10 and B11 are free
' ----------------------------------------
' Change the Delay and status variables here
' ----------------------------------------
'
' Use *either* Sdelay or Ldelay
'
' Sdelay will time in milliseconds with no sleep mode so no low-power
' mode while timing. We will start counting in Sdelay2 (if non-zero0
' and then do Sdelay 1 (set Sdelay2 to zero if it's not used). For delays
' of 1-65535(65.5 seconds) just use Sdelay1 - using Sdelay2 increase the maximum
' time period to 131 seconds (2 x 65.5)
SYMBOL  Sdelay1 = 0
SYMBOL  Sdelay2 = 0
'
' Ldelay1 and Ldelay2 work like Sdelay1 and Sdelay2 but Ldelay times in units
' of 2.304 seconds with some particulars. The value is rounded up to the
' nearest multiple of 2.304. So a value of 10 will delay for 11.52 seconds.
' A delay of 65535 is just over 18 hours and if Sdelay2 is also used, we
' can have a maximum interval of over 36 hours. Ldelay will put the Basic Stamp
' into low power sleep mode and save drastically on battery life
SYMBOL  Ldelay1 = 3600
SYMBOL  Ldelay2 = 0
'
' The last three switches follow and they are binary
SYMBOL  dtype = 1   ' Set to 0 if Sdelay is used or 1 if Ldelay is used
'
SYMBOL  repeat = 1  ' Set to 1 if delay recycles else set to 0 for single use
'
SYMBOL  silence = 0 ' Set to 1 to NOT issue warning tones else set to 0

' ----------------------------------------
'
' The values above will use a delay of 1 hour (3600 seconds[Ldelay1=3600, Ldelay2=0]),
' using the Ldelay pairs (dtype=1) which uses the SLEEP function. The camera will operate
```

```
' after every delay cycle (repeat=1) and will produce audible beeps before operating
' (silence=0)
'
' Start here
'
Start:
' announce the power up with a single beep
GOSUB beep1

'Fill the critical variables with the user-defined setup values
LET W1 = Sdelay1
LET W2 = Sdelay2
LET W3 = Ldelay1
LET W4 = Ldelay2

LET BIT0 = dtype
LET BIT1 = repeat
LET BIT2 = silence

' On power-up all port bits are input so,
' set direction register and initial states
' for our 3 output bits
LOW 0   ' Opto 1 - shutter
LOW 1   ' Opto 2 - autofocus
LOW 2   ' Piezo speaker

' wait for the start switch
wait:
  IF PIN3=1 THEN wait
'
' announce that we got a start press with two beeps
GOSUB beep2
'
' Figure out the delay or cycle time
cycle:
' are we using pause or sleep?
IF BIT0=1 THEN useLD

' we are using sdelay and the pause command
IF W2=0 THEN useSD1
' there are two delays
PAUSE W2
useSD1:
PAUSE W1
' operate the shutter

GOTO ShOp
' we're using ldelay and the sleep command
useLD:

IF W4=0 THEN useLD1
SLEEP W4
```

```
useLD1:
SLEEP W3
' fall through to operate the AF and shutter
ShOp:

' unless silence is on, do the a three-beep warning
IF BIT2=1 THEN ShOp1
GOSUB beep3
'
' Here we take the pic. It is probably not necessary to operate
' the AF as the shutter operate will do that but I included it
' because some of my reading was somewhat confusing. Also, I
' delay 0.5 sec before resetting the output bit - this is just
' a guess that works fine for me - it's probably a bit long
ShOp1:
' operate the autofocus
HIGH 1
PAUSE 500
LOW 1

' operate the shutter
HIGH 0
PAUSE 500
LOW 0
'
' if we are not repeating then just stop
IF BIT1 = 0 THEN finish

GOTO cycle

finish:

END


' beep subroutines
Beep3:
SOUND 2,(111,18)
SOUND 2,(0,10)
Beep2:
SOUND 2,(111,18)
SOUND 2,(0,10)
Beep1:
SOUND 2,(111,18)
SOUND 2,(0,10)
RETURN
'-----------------------------------
' fine
'-----------------------------------
```