Using IR input/output with a Propeller

Bob Belleville

Introduction and Scope

For the cost of a cog and a pin an ordinary 'TV' remote can be easily used to provide a keypad for input to a Prop based system. This works especially well with the Prop because a cog can continuously watch for IR codes and place them in a buffer while other cogs carry on other tasks. Remotes are common and readily available in the surplus market for $2 or less and as 'Universal Remotes' for under $10 at RadioShack or the local drug store for that matter. The interface is a readily available 3 pin IC which uses 5VDC and has a single output line which can be interfaced to the Prop through a resistor (4.7K is fine.) These parts are available from surplus for $1 and from other sources for less than $5. Parallax has a receiver #350-00014 and a kit with a remote #29122.

This document explains how to build and code this interface, identify random remotes you may have handy, record multi-code input, understand some of coding systems currently in use, and build an object to input and buffer button pushes.

This kit includes several files. Some require the tv_terminal object to see what is going on. Others use FullDuplexSerial.spin to access the built-in serial interface on the Demo board or via the PropPlug. Any of the tv_terminal objects can be converted for serial access by simply switching the object name in the OBJ block. The final input/output objects (ir_reader_*.spin and ir_tx_nec.spin) require no other objects.

A resources section at the end of this document gives pointer to a vast amount of web based documentation.

Given the work of this document, building a transmitter to send these signals to other devices is a logical next step. Included are objects to transmit IR codes. The hardware interface here takes a transistor, resistor, and IR led to provide a bright enough IR signal. Here the Prop really shines because it can not only produce the IR codes but also modulate the LED at various frequencies as well.

version history:

late Feb 2007 to mid Mar - development
2007/03/15 - v1.0

file list:

ir_oscope.spin

Used to just see if IR receiver module is connected and operating.

ir_view.spin
: Used to show remote coding via tv_terminal.

ir_view_serial.spin
: As above but more complete using serial_terminal.spin. Auto decodes many formats.

ir_capture.spin
: As above but reports timings in microseconds. Is suitable to gather data on exact remote timing. Does no decoding.

ir_reader_demo.spin
: Used to demonstrate the three reader objects (below.) Can use either tv or serial terminal.

ir_reader_nec.spin
: Used to get input from a NEC remote and place keycodes into a fifo. Can filter device codes.

ir_reader_sony.spin
: Used to get input from a Sony remote and place keycodes into a fifo. Can filter device codes.

ir_reader_rc5.spin
: Used to get input from a RC5 remote and place keycodes into a fifo. Can filter device codes.

ir_tx_nec.spin
: Used to transmit NEC IR codes. Matches the specification of UPD6122 datasheet for timing.

ir_tx_nec_demo.spin
: Demonstrates above objects by sending PC keyboard key through a NEC IR link to the NEC reader above then displays them on the terminal screen.

serial_terminal.spin
: Like SimpleDebug.spin but set up to duplicate the method calls of tv_terminal.spin. It also adds non-blocking input.

readme.pdf
: This document.

readme.abw
crw_1929.jpg
: AbiWord source of this readme and the tv_terminal screen shot.

IR Remote Operation

IR stands for infrared which is light just below the visible spectrum. We can't see it. It turns out that most of our simple point and shoot digital cameras can. This is how to determine if a remote is working. Turn on the camera and watch the LCD display while you point the business end of

any remote at the lens and press any button. You should see a blinking dot when the IR diode is sending information. I haven't tried this with all cameras or remotes but it will work for many. (Oddly enough digital SLR cameras without real time lcd display won't work for this task. Try a time exposure in the dark.)

A remote is just a 'IR flashlight.' It blinks in a specific way to send a code sequence to your TV or other equipment. Remotes send a vast number of different kinds of codes sequences but as we shall see there are only a few common ones and one in particular is very common and easy to decode. In all cases, these codes tells which button has been pressed and specifically which remote is sending the code. This is how you can have a dozen components in your home entertainment system and only one responds to its associated remote.

Technically the IR light emitting diode (LED) in the remote is modulated so that is blinks on and off in the order of 38K times per second. Remotes modulate at various frequencies and this modulation frequence must be matched to the receiver module. In practice none of these devices are all that fussy and all we need to find is a remote and receiver that work together. You will see some remotes that are too far from the receivers center frequency to operate. Receivers near 38K are usually ok with common remotes. Receiver modules filter out this modulation and show only on and off.

Code sequences consists of bursts of these modulate LED flashes separated by periods where the LED is completely off. These on bursts and resting spaces are never shorter than about 0.25millisecond and usually rather longer. So we will forget about modulation for now and consider only the burst as a single event which I'll call a 'mark' and then a resting period which I'll call a 'space'. An IR receiver chip will produce a high or one level for space and a low or zero level for mark. This is what the Prop's input pin will see.

Given all this, many coding methods are possible to represent a series of binary bits. Only three are common:

1. Make the mark a fixed length and make the space short for 0 and long for 1. This is the method used by the so called NEC code which is used by a great number of devices. Inspecting the datasheet for UPD6122 will give the exact format. This is one of many many ICs used in remotes. Many remotes don't use this exact NEC format but still use fixed mark length for bit coding. ir_reader_nec.spin decodes the NEC format and can be modified for other formats.

2. Make the space a fixed length and make the mark short for 0 and long for 1. This method is used by Sony (sometimes) and perhaps others.

ir_view.spin can be adjusted by a compile time constant to decode these signals. ir_reader_sony.spin decode these style codes

3. Vary both mark and space. This is used in the Phillips RC5 scheme and is also call bi-phase coding or Manchester coding. It has the advantage of having the both the 0 and 1 bits taking the same transmission time. They can be easily spotted with the ir_view.spin. In this scheme a 1 bit has a mark in the second half of a bit cell and a space in the first. Zero is just the reverse. Decoding is easy because once you know the bit time you can just sample at fixed time intervals and shift in 0/1. ir_reader_rc5. spin takes this approach.

In addition to the payload of 0/1s, code sequences often have a preamble or start sequence consisting of a long mark and a longish space to alert the receiver that data is about to be sent. This long mark also serves to allow the receivers automatic gain control circuit (AGC) to adjust for both far away remotes and ones closer up. A stop mark/space pair is also common consisting of a normal length mark and a long space.

Often in about 0.1 to 0.05 seconds if a button is still down on the remote, a repeat sequence will be sent to say that the button is still down. In the NEC code case this is just a start sequence followed immediately by the stop sequence --- no data bits. Others repeat the whole thing. There are many variations.

First Step - Confidence Check

Obtain a receiver and wire the ground and +5VDC pins. Connect the output pin through a 4.7K ohm resister to P0 of the Prop. Run the ir_oscope.spin object and watch a stream of "." on the display. Push any button and check that at at least some 0s show up. Without pressing any buttons watch the "."s go by for a while. If no 0 show up the receiver is most likely ok. If random 0s show up often then find another receiver module. This puts us more than half way there. "." is for the space or 1 state of the receiver --- 0s are easy to see mixed with periods.

Second Step - Identify the Remote

Run the ir_view.spin object. Push a button on the remote and try not to get a repeat code. With each press a screen full of numbers should show up.

If the mark count for all the data pairs is about the same you have a type 1 code sequence.

If the space count for all the data pairs is about the same you have a type 2 sequence.

If both mark and space take on different values you have a bi-phase code type 3.

This is the screen shot for a RadioShack 15-2142 4-in-1 universal remote programmed for a NEC TV code 0030 with a click on the "1" button. The receiver is RadioShack 276-640 running at 5VDC. (It is rated to run down to 2.4VDC so it could be run on Prop voltage directly and the resistor eliminated --- but I haven't tested this. What do we see on this screen.

Line 1 '34' is the number of mark/space pairs --- one to start 332/160 32 data pairs and 1 stop pair 21/501. The mark part of each data pair is always 21 or 20 and the space part is either 18 or about 58. In this case the 18s are zero bits and the 58s are 1 bits. This sequence is type 1 above --- fixed mark and short/long space. Next to the bottom line are shown the bits LSB (least significant bit) first which is the bit transmission order. Read those bits backwards gets the bottom line in hex with the MSB (most significant bit) first as is normal for reading numbers.

Read the hex number as 3 parts EE 11 FB04. EE is the 1's complement of 11 and is used for error control. This is the key pressed. That is 11 is the byte code for the "1" button on the remote. You will have to make a list of all the codes you want by using ir_reader_demo.spin object and then build them into your application. FB04 is the device code for the remote itself. This will be the same for all the keys on this remote --- only the key codes

will change. Checking this code in the input object will reject any other remotes used near the Prop module. (You might notice that FB is the complement of 04. Many NEC devices have this convention but not all. They apparently ran out of device codes and went from 8 to 16 bit.) (Some also do not have the key code and complement. The NEC reader can be easily modified to pass any keycode. Check at about line 146.)

These timing numbers are essentially arbitrary --- counts in a Spin loop. According to the NEC spec, 332 should be about 9ms, 160 4.5ms, and 21 0.56ms. Check out the code in this module to see how the loops work. This would give 36.9 counts per ms. The ir receiver is lengthening on time a little bit because of its low pass filter. The length of the spaces isn't quite to spec. This doesn't matter.

Zeroes are about 18 counts and ones are 58. A threshold of 38 should be used to safely reject noise. ((58-18)/2+18=38) That is, space counts below 38 should be taken to be zero and above 38 taken to be 1 for best noise rejection.

ir_view will give counts for sequence types other than NEC but the decoding into 0/1 will have to be modified if you want to use the Sony code or bi-phase. Both NEC type1 and Sony type 2 are implemented by a compile time constant. Long sequences are prevented from scrolling off the screen and data is lost. Try making the constant _esl shorter to separate first key from repeat which is most likely what is causing the long input. There should be a longish space count somewhere on the screen. This is actually an inter code space. I've seen these as short at 16ms.

The stop pair is something of a problem --- the space part actually continues until another button is pressed or the repeat sequence starts. The viewer's loop just counts to _esl and shows the result so far. Watch this as you may run into some strange remotes.

A far better program for identifying remotes is ir_view_serial.spin. This object uses the serial port to remove the problem of information running off the top of the screen. It also allow the capture of data to a text file. This object attempts to automatically identify and decode any remote. This isn't actually possible so be carefully as it will get things wrong sometimes.

Use the instructions in the object to run it. Below is a sample run with some comments added on the right:

---------- begin capture data ----------
total samples: 68

334 165

21 18 20 19 20 18 21 59
21 18 21 18 20 19 20 18
21 59 21 60 20 60 20 18
21 60 20 60 20 60 21 59
21 59 21 18 21 59 21 18 typical NEC code
20 60 21 18 20 19 20 18 with 2 repeats
21 18 20 60 21 18 20 60
20 19 20 60 20 60 21 59
21 1448
variable space code: EA15F708(32)

331 82
21 3558
repeat (?)

331 82
21 100001
repeat (?)

total samples: 172

312 154
19 57 19 56 19 56 19 56
19 18 19 56 19 17 19 57 a JVC remote & 2 repeats
19 17 19 56 19 17 19 18
19 56 19 56 19 18 18 57
19 678
variable space code: 0000B2AF(16)
19 56 19 57 19 56 19 56
19 17 19 57 19 17 19 56 repeats don't have start
19 18 18 57 19 17 19 17 burst
19 57 19 56 19 17 19 57
19 678
variable space code: 0000B2AF(16)
19 56 19 57 19 56 19 56
19 17 20 56 19 17 19 56
19 18 19 56 19 17 19 17
20 56 19 56 19 17 19 57
19 10001
variable space code: 0000B2AF(16)

total samples: 48
34 30 66 31 32 30 32 31 an RC5 code and 1 repeat
32 30 33 30 32 31 32 30 (toggle bit same)
33 30 32 31 32 62 66 3328
rc5(14bit) or rc5x(15bit): 0000C008(14)
33 30 66 30 32 31 32 31

```
32 30   32 31   32 31   32 30
32 31   32 30   33 62   65 10001
rc5(14bit) or rc5x(15bit): 0000C008(14)

total samples:  48
34 30   33 30   66 30   33 30   second press same key RC5
32 31   32 30   33 30   32 31   (note Toggle bit toggled C-->E)
32 30   32 31   32 62   66 3329
rc5(14bit) or rc5x(15bit): 0000E008(14)
32 30   32 31   65 31   32 31
32 30   33 30   32 31   32 30
33 30   32 31   32 62   66 10001
rc5(14bit) or rc5x(15bit): 0000E008(14)

total samples:  126

89 20
44 20   44 19   44 20   22 19   a Sony code with repeats
22 20   22 19   22 20   22 19
44 20   22 19   44 20   43 20
44 20   22 19   22 20   43 20
22 20   22 19   44 20   22 485
variable mark code: 00049D07(20)

87 20
44 20   44 20   43 20   22 20
22 19   22 19   22 20   22 19
44 20   22 19   44 20   44 19
44 20   22 19   22 20   44 19
22 20   22 19   44 20   22 485
variable mark code: 00049D07(20)

88 19
44 20   44 20   43 20   22 20
22 19   22 19   23 19   22 19
44 20   22 19   44 20   44 19
44 20   22 19   23 19   44 19
23 19   22 19   44 20   22 10001
variable mark code: 00049D07(20)

---------- end capture file ----------
```

Object ir_capture.spin produces accurate mark space timings in microseconds. Runs much like above. Times out in about 5 seconds if there is no more ir input and dumps the sample buffer to the terminal. Here is an example capture file. It is coded in the first column to make processing by another program easier.

---------- start capture file ----------
any key to begin
any key to begin
any key to begin
any key to begin
any key to begin
ir_capture
type any note and press return to sample

>sasem nec remote 1 key and repeats
1,92
2,9023,4473
2,615,505
2,616,504
2,616,504
2,615,1611
2,615,505
2,615,504
2,615,504
2,615,491
2,615,1624
2,616,1624
2,615,1624
2,616,491
2,616,1624
2,615,1624
2,616,1624
2,616,1611
2,615,1624
2,616,504
2,615,505
2,615,491
2,616,504
2,615,504
2,616,504
2,616,491
2,616,504
2,616,1624
2,615,1624
2,615,1611
2,616,1623
2,615,1624
2,615,1624
2,615,1611
2,615,40161
3

2,9023,2220
2,618,96060
3
2,9023,2220
2,618,24022658
3

>JVC remote 'enter'
1,274
2,8446,4165
2,574,1530
2,573,1530
2,573,1530
2,574,1530
2,575,477
2,573,1531
2,575,477
2,574,1530
2,574,478
2,574,1530
2,574,477
2,573,478
2,574,1530
2,575,1529
2,575,477
2,575,1529
2,576,18319
3
2,578,1525
2,577,1527
2,577,1526
2,578,1526
2,577,474
2,577,1527
2,577,474
2,577,1527
2,578,474
2,577,1526
2,578,474
2,577,475
2,577,1527
2,578,1526
2,579,473
2,577,1527
2,579,18315
3
2,587,1517

2,587,1517
2,587,1517
2,587,1517
2,586,465
2,586,1518
2,587,465
2,586,1518
2,586,465
2,586,1518
2,587,465
2,586,466
2,585,1519
2,585,1519
2,586,466
2,585,1519
2,585,-10767139
3

---------- end capture input ----------

A IR-Reader Object

Armed with this information we can build a reader object. Included in this kit are ir_reader_(nec/sony/rc5).spin and ir_reader_demo.spin. Tune the reader to the remote you wish to use and run the demo. As keys are pressed they are shown on the monitor (either tv or serial.) ir_reader_*.spin has a fifo to buffer key codes as they arrive. The demo shows how to setup for and dequeue key codes.

Setting up this object requires a pin number where the receiver module will be wired. deviceID is the 16bit number for a given control. You can use ir_view.spin (etc.) to find out what number you have. If you give a 0 then any NEC 32 bit coded remote will be accepted but be warned that the key codes will overlap. A repeat delay tells how many repeated codes to skip before buffering more copies of the last valid code. This helps to give time to get your finger off the button if you want only one but add repeats if you hold longer. A possible modification of the object would be to set the high bit of repeated key codes so that the calling application can respond correctly. This is done with a flag bit during initialization. Few or no remotes actually have 128 or more keys to the codes are usually 127 or below.

Transmitting IR codes

Spin isn't fast enough to actually modulate a LED at these frequencies but assembler is plenty fast. Included is a transmitter object which sends NEC codes by essentially implementing upd6122 chip. Object ir_tx_nec_demo.

spin shows how to use this module. Building a general purpose 'universal' transmitter has to be left for later and hopefully another programmer.

Resources

Sometime in the '90s(?) Juergen Putzger produced a simple text file to describe IR codes. I've had it in my notes for many years. As of 2007 it is on line at:

http://www.ee.washington.edu/circuit_archive/text/ir_decode.txt

For Linux there is a program to use IR to control PCs. This has a lot of good information on receiver modules etc. Check out the 'home-brew' links.

http://www.lirc.org/

There is a version of the program for window call winLirc. Which can be found using Google.

To understand the coding of RC5 Wikipedia has an entry on Manchester coding which was used in the original coax Ethernet.

http://en.wikipedia.org/wiki/Manchester_encoding

More here by Guy Kuo.

http://www.hifi-remote.com/infrared/IR-bi-phase.shtml

This base site takes up modifying a particular kind of remote and is most interesting. This is more for the NEC type:

http://www.hifi-remote.com/infrared/IR-PWM.shtml

Other somewhat related info at:

http://www.hifi-remote.com/ofa/

Phillips makes high end programmable remotes for home theater applications called Pronto. You can't get far on the web before you run into 'Pronto Hex Format' which a code for describing any kind of IR sequence. Start here:

http://www.remotecentral.com/index.html

Details of the format and a good description of both the NEC and Sony formats as they relate to Pronto Hex Format is here:

http://www.remotecentral.com/features/irdisp1.htm

A Google search on "universal remote" will give much more. This is a hot topic now (2007.)

sbproject has by far the best description of ir formats in use. It is a 'Rosetta Stone' of protocols:

http://www.sbprojects.com/knowledge/ir/rc6.htm

EndNotes

V1.0

These were my first assembly objects and nearly my first Spin programs. You will see a good deal of variation in style as I got the hang of this environment.

I'm pretty sure the ir_readers_*.spin don't need the lock. But I have to study more carefully to be sure my code meets the single producer/single consumer model. Anyway it shows how they are done. The transmitter is clearly ok.

I thought the code in ir_capture.spin was cool. It allows accurate timing in one cog and a deadman timeout with less time accuracy in another. This is such a cool machine.

Everything has been tested and operated when I made up the .zip file. No doubt there are errors. Let me know on the forum. I'll help as I can. Right now I pretty tired of IR remotes and ready to move on.