January 2000 – Interfacing an X10 Firecracker

I often think that my wife is an official Santa's helper. She really gets into Christmas. We had two trees this year as well as lights in the yard, reindeers, and a nativity scene. Each year my contribution to this is to wire up the lighting systems with X10 controllers. These are the little modules that communicate via the AC lines in your house. With these modules I can turn the lights on and off with a remote control that also plugs into the wall. You don't have to go outside on a cold night to unplug the lights (not that it gets too cold here in Houston).

My wife is pretty much anti-technology (opposites attract after all). But this year I put something together that even impressed her. I t started when I finally decided to get a Firecracker kit from http://www.x10.com. They provide these kits for "free" (you have to pay $5.90 for shipping – *Update: this promotion is over but you can still get the Firecracker, just not at this low price*). Here's what you get:

An X-10 wireless RF transceiver

An ordinary X-10 module

A wireless RF remote that sends commands to the transceiver

A small DB9 device that plugs into a serial port (the Firecracker)

Forget the Firecracker for a minute (we'll get back to it). The way this works is you push buttons on the RF remote and the transceiver picks them up and retransmits them over your house wiring to regular X10 modules. You can turn devices on and off or dim lights (if you have a lamp module). The transceiver is also an X10 module so with the basic kit you can control 2 devices. You can find X10 modules at Radio Shack and many hardware stores. If you can order them on the Web they are usually less expensive, but either way they are widely available.

So what's the Firecracker for? This tiny module plugs into a 9-pin RS232 port. The module has a pass through connector so you can share the port with some other RS232 device like a modem. The Firecracker sends RF commands just like the remote control. The free software provided gives you a remote on your PC screen that works just like the regular remote.

Who wants a PC running all the time to run your Christmas lights? Not me. This calls out for a BASIC Stamp! Of course the BASIC Stamp 2 can control X10 using a $30 or so transceiver that plugs into the wall. Expensive! Wired to the wall! Why do that when you can spring $6 and run your X10 wirelessly from a BASIC Stamp.

The only problem is that the Firecracker uses an odd protocol. Once you figure it out, though, it is easy to put all the logic in subroutines. That's what is in the code below.

Just a disclaimer: I have nothing to do with www.x10.com at all. Once you buy something from them they will inundate you with e-mail. You can go to their Web site and fill in a form to move to their "low volume" mailing list which is much better, or you can have yourself

removed all together. I wouldn't want to do that; they have some nice stuff. But I also didn't want a piece of e-mail from them every single day!

Under the Hood

It is simple to connect the Firecracker to the BASIC Stamp. I took a female DB9 connector and wired it like this:

DB9 pin	BASIC Stamp connection
4	P0
5	Ground
7	P1

That's it; just 3 wires. You could even adapt this for the BASIC Stamp 1 if you wanted to do so.

Using the code is simple. First call **resetfirecracker** to get everything to a known state. Then you can set the **house** variable and **unit** variable to address the device you want to command. To talk to house A, for example, set **house** to 0. The **unit** is a number from 0 to 15. You also set **cmd** to what you want to do (0=off, 1=on, 2=bright, 3=dim). Then call **sendcmd** and you are done. The Firecracker gets its power from the BASIC Stamp pins. The bright and dim commands apply to the last device turned on, so you also set **unit** to 0 when sending these commands.

The example program below uses switches wired to P8, P9, P10, and P11. The switches connect between the pins and ground. Each pin also has a pull up resistor (say 10K or 22K) so the pin reads high when the switch is open and low when you close the switch.

The example program turns off all the devices so they are in a known state. Then when you press a button it toggles the state of the corresponding device. Of course, you could write as fancy a program as you wanted to take the time to write. For example, you could use a light sensor to turn the lights on and off automatically. Or light them according to a timer—use your imagination.

If you want to know more about the Firecracker protocol there is a nice article about it at: http://eagle.cc.ukans.edu/~mturvey/firecracker.html (*Update: this link appears dead; try www.geocities.com/ido_bartana/Firecracker_protocol.htm instead)*. You'll also find links there to more software for Windows, Linux, and some pictures of the device.

The Code

```
' Firecracker Interface (Al Williams)
' http://www.al-williams.com/awce
' Wire Ground (DB9 pin 5)
' DTR (DB9 pin 4)
' RTS (DB9 pin 7)
```

```
rts con 1
dtr con 0

' The Firecracker header (2 bytes)
h1 con $D5
h2 con $AA

' the Firecracker footer (1 byte)
foot con $AD

' byte to send
byt var byte
tmp var byte
i var nib
x var nib
state var bit(4)
house var byte '0=A 1=B...
unit var nib  ' 0-15  (must be 0 for bright/dim)
cmd var nib  ' 0=off 1=on 2=bright 3=dim

gosub resetfirecracker

' Sample program
main:
' all off
  for x=0 to 3
    house=0
    unit=x
    cmd=0
    gosub sendcmd
    pause 20
    state(x)=0
  next

mainloop:
  if in11=0 then c0  ' look for buttons on p8-11
  if in10=0 then c1
  if in9=0 then c2
  if in8=0 then c3
  goto mainloop

c0:
  x=0
  goto ccmd
c1:
  x=1
  goto ccmd
c2:
  x=2
  goto ccmd
c3:
```

```
  x=3
ccmd:
  unit=x
  cmd=state(x)+1
  if cmd=1 then gocmd
  cmd=0
gocmd:
   gosub sendcmd
   pause 250
   state(x)=cmd
   goto mainloop

' End of example program
' Send command
sendcmd:
  byt=h1
  gosub sendbyte
  byt=h2
  gosub sendbyte
  read housetbl+house,byt
  if unit<9 then lowunit
  byt=byt+4
lowunit:
  gosub sendbyte
  byt=$20
  if cmd=0 then addunit
  byt=0
  if cmd=1 then addunit
  byt=$88
  if cmd=2 then nounit
  byt=$98
  if cmd=3 then nounit
' huh???

addunit:
  read unittbl+(unit//8),tmp
  byt=byt+tmp
nounit:
  gosub sendbyte
  byt=foot
  gosub sendbyte
return
' Send 1 raw byte
sendbyte:
  debug hex byt," "
  for i=0 to 7
   if byt & $80 = $80 then xmit1
   gosub send0
nextbit:
   pause 1
   byt=byt*2
```

```
   next
   return

' Send a 1
xmit1:
  gosub send1
  goto nextbit


' Send bits (0 or 1)
send0:
  low rts
  pause 1
  high rts
  return

send1:
  low dtr
  pause 1
  high dtr
  return

' Always reset firecracker first
resetfirecracker:
low rts
low dtr
pause 50  ' reset
high rts
high dtr
pause 50
return
' Data for house and unit codes
housetbl data $60,$70,$40,$50,$80,$90,$A0,$B0,$E0,$F0,$C0,$D0
          data $00,$10,$20,$30
unittbl data 0,$10,$8,$18,$40,$50,$48,$58
```