PCR-1000 radio for UN*X GUI Development and an OOP Library

Part I: Protocol

A Ghetto.Org Investigation
PolyWog and Javaman
November 16, 1999

Objective:

Create a shared object (or static) library for facilitation of the Icom PCR-1000 UN*X interface. Create a GUI and command line interface for the Icom PCR-1000 all band, all mode receiver.

PCR-1000 Serial Protocol:

The PCR-1000 begins each command with a set of plain text codes sent across the serial cable at 9600 baud. Each command is terminated with a CR-LF represented by the HEX codes \x0d and \x0a.

Radio Initialization String:
(Equivalent to sending a power on, and `G300` code.)

```
H101\x0d\x0aG300\x0d\x0a
```

Basic command codes:

`G3NN`		- Signal Update
	`00`	- Off (program should poll status)
	`01`	- On (radio sends status automatically when change sent)
	`02`	- Binary Mode (Update off)
	`03`	- Binary Mode (Update on)
`H1NN`		- Power
	`00`	- Off
	`00`	- On
`J40NN`		- Volume
	`00-99`	- Level
`J41NN`		- Squelch
	`00-99`	- Level
`J43NN`		- IF Shift
	`00-FF`	- Level[1]
`J45NN`		- Automagic Gain Control
	`00`	- Off
	`01`	- On
`J46NN`		- Noise Blanking
	`00`	- Off
	`01`	- On
`J47NN`		- RF Attenuator
	`00`	- Off
	`01`	- On
`J50NN`		- VSC
	`00`	- Off

[1] These are the documented HEX values. Something tells me (if its anything like the squelch and volume) that the value ranges are between 00-99.

	01	- On
J51NN		- Tone Squelch
	OO	- Off
	01-33	- Standard tones [67 Hz - 254.1 Hz]
J4ANN		- <UNKNOWN>
	80	- <UNKNOWN>
LD82NN		- Tracking Filter
	00	- set to automagic

Frequency Selection Command:[2]

K0		- Command Initialize
0000000000		- Frequency (freq's must be padded correctly)
	0000050000	- Lower Range (0.050 MHz)
	1300000000	- Upper Range (1300 MHz)
0N		- Mode
	00	- Lower Side Band
	01	- Upper Side Band
	02	- AM
	03	- CW
	04	- <UNKNOWN>
	05	- Narrow FM
	06	- Wideband FM
0N		- Filter
	00	- 3 kHz
	01	- 6 kHz
	02	- 15 kHz
	03	- 50 kHz
	04	- 230 kHz
00		- Mandatory but arbitrary padding at end.

Sample Output for selecting z-100 WHTZ from NYC's Empire State Building.
-Note this was a totally random selection ;^)
To select: 100.3 FM (WFM) 230kHz filter.
Cmd Out: K00100300000060400
Break down: [K0][0100300000][06][04][00]

Radio Queries:

H1\?	- Is the power on? [Reply: H1NN]
I0\?	- Squelch setting?
I1\?	- Signal strength?
I2\?	- Frequency offset?
I3\?	- DTMF Tone?
G4\?	- Firmware revision?
GE\?	- Country/Region?

[2] For the number padding etc, I have used the standard SQL formatting scheme. N = number, 0=any number or padded with zero if blank, 9 = any number padded or not

Standard Radio Reply:[3]

G0NN		- Standard Reply
	00	- *OK* reply
	01	- *Bad* reply
GDNN		- DSP Present
	00	- No
	01	- Yes
GENN		- Country Code(s)
	09	- US
	02	- Euro. /UK
I0NN		- Squelch Status
	04	- Closed
	07	- Open
I1NN		- Signal Strength
	00-FF	- Weak-Strong
I2NN		- Signal Centering
	00-7F	- Low
	80	- Centered
	81-FF	- High
I3NN		- DTMF Tone
	00	- None heard
	1N	- Heard (N= [0-F] where *=E, and #=F)

Radio functions:

G1NN		- Set baud rate
	00	- 300
	01	- 1200
	02	- 2400
	03	- 9600
	04	- 19200
	05	- 38400
G3NN		- Set autoupdate
	00	- Off
	01	- On

Protocol Programming Notes:

The radio will at most times reply with *Ok* or *Bad*. Should it reply with anything else, then it is an ‘‘actual" reply string from the radio and must be parsed. First we should check to see if it was a reply string without garbage characters. Just strip[4] the CRLF chars and see what is left. The easiest way (although not the smartest way) to find out what the reply is for, is to remember what we asked it last, and assume that the reply (if it is not *Ok* or *Bad*) is the reply to our question.

In order to understand the radio, you have to know what to expect. The radio seems to send out ‘\n' even when it has nothing to say. So the best thing to do, when *deciphering* the replies is to make sure that the length read in is greater than one. Because it will always be, at least, one.

Being patient is also something to hold dear. It may take the radio a grueling .5 seconds to respond to a request. However, it shouldn't (on average) take it more than 3 seconds to reply to any

[3] We don't care about \x0d\x0a. Since we know that all replies will be 4 bytes in length, then we should chop off the CRLF by *null*'ing byte MAXLENGTH+1 (or five) ;^)

[4] Use the function strpbrk() sending as the second argument “\x0d\x0a “

one request. So a timer that triggers SIGALRM may be started and set to 5 seconds. If no reply is received within five seconds, you should assume that something is seriously wrong and attempt either bail, or resend the request.

Kudos to: Javaman for redirecting me to the ‘‘original"‘ pcr.c source that was able to open up the protocol barrier. And Devi0us for use of the radio.

Next up: [Part II: The Objects]