Hardwired Forth

Presentation for Forth Day, November 15th, 2014

Contents

Introduction

One of my retirement occupations is building small microprocessor-based electronic modules that are used for my own projects and for products sold in small quantities on eBay. Many of these products and projects are described at the BOBZ Products site that is hosted at http://www.kiblerelectronics.com

All of the modules are programmed in MyForth. MyForth, a GForth-based development environment for 8051 microprocessors, is also described at the above site (the kibler site).

Many of the modules are commanded via a serial port, either manually with a terminal or with a serial programming script. These serial-enabled modules are the focus of my presentation.

Later, I will show you the latest example of one such product, a radio frequency link module that transmits data in a variety of modes between a Master Station and one or more remote clients.

But, first, I will introduce a concept that I have found useful in thinking about how to build systems that can be configured for a variety of applications without having to worry about building circuit boards and programming microprocessors.

Forth in Hardware

This concept, "Hardwired Forth", is based upon the time-tested method of developing Forth software. This should not be confused with the concept of "Haywired Forth", which results in a tangled mess of confusing Forth Words.

The basic idea of Hardwired Forth is perhaps best illustrated with an example that is familiar to most of you: the washing machine example application documented by Forth, Incorporated.

This example shows how complex and flexible applications can be built using simple functional modules (Words) that are well-tested and are designed to implement functions related to a particular application. In the washing machine example, typical Words are "rinse", "spin" and "fill", each performing a function common to washing machines.

I have found the classic Forth software development methodology useful in designing hardware modules that perform a range of functions initiated by commands sent to a serial interpreter on a microprocessor. Thus, each serial command performs a function that can be roughly considered a Forth application Word, but implemented in hardware. For example, a washing machine controller might respond to a "spin" command by turning on the motor controller for the tub.

In the case of the radio frequency link module, RAFL, it executes command Words such as "adr", which is used to set an address. Other command Words, such as "remote" and "master", set the context for interpreting commands, much like a Forth vocabulary. Thus, in the "master" mode, the "adr" command sets the attentive address, which is the Remote being addressed. However, in the "remote" mode, "adr" sets the address of a Remote.

To build more complex systems, a control processor such as a PC, Raspberry Pi, Beaglebone or Arduino can execute commands under control of a scripting language such as GForth, Python, a Bash shell script or Perl. Alternatively, a microprocessor-based control processor can issue a series of serial Words from a hard-coded script. The kibler site given above describes how this is done with examples in Bash, Python and Perl scripts. These examples are in the Application Notes section of the BOBZ hosted site.

For more complex or expanded functionality, multiple hardwired Forth products can be cobbled together with multiple serial ports and command scripts. For example, one of our products, DACS, is a small data acquisition system that controls several high-power outputs, digital inputs and analog inputs (not to mention temperature monitoring, so I won't). Several DACs can be attached to a control processor such as a Raspberry Pi, using USB to serial adapters. The control processor can also control multiple DACS modules attached to RAFL remotes via a RAFL Master Station.

Thus, complex hardware applications can be developed using only an inexpensive control processor with multiple serial ports and scripts. No microprossor programming is required: application development requires only a scripting environment, not a microprocessor development environment.

Development

Our development environment for the RAFL product may be of some interest and serves to illustrate some of the possibilities discussed above.

To develop the RAFL product we needed the following:

- **RAFL Modules** -- At least two RAFL development modules: one configured as Master and one configured as a Remote.
- **A DACS module** -- This is attached to the Remote and tests the Remote's ability to receive commands from the Master, transmit them to the DACS and send DACS responses back to the Master.
- **A Programmer** -- This is shared between the RAFL modules and manually switched between them during development. With multiple programmers attached to multiple serial ports, programming of multiple Target systems can be controlled entirely from a command line.
- **Serial Ports** -- Multiple serial ports are needed for querying the Master, monitoring the Remote's serial I/O and programming the RAFL modules.

RAFL Modules

You can see what a RAFL module looks like by contacting me after the presentation for a show and tell. There are also photos of the RAFL and DACS modules on the web site, as well as a photo of the development hardware.

Here are some RAFL features that may be of interest:

- **Configuration** -- All RAFL modules are capable of being configured either as a Master or as a Remote: the software is identical. Configuration is performed by setting up parameters interactively, through interpreter commands, using a terminal program. The resulting configuration can be saved to flash memory with a command. After configuration, each RAFL automatically starts up as either a Master or as a Remote.
- **Local I/O** -- In addition to a managed serial I/O port, each RAFL module has local I/O. Local I/O includes a 12-bit analog input, a temperature sensor, a transistor-isolated output and four digital inputs
- **Modes** -- The RAFL Master Station can operate in several modes, including loopback, terminal, data and broadcast. In the loopback mode, data sent to a Remote is re-sent from the Remote's input message buffer. The broadcast mode allows messages to be sent to all Remotes simultaneously: responses are put in a buffer at each Remote for later query from the Master. Data and terminal modes allow data transmission in an ASCII terminal mode or in a fixed-length data mode (all codes, including hex 00 can be transmitted).

DACS Module

The DACS module connects to the serial port of RAFL Remote. The Remote can send commands received from the Master to the DACS and transmit the DACS response back to the Master. A USB to serial adapter bridged across the RAFL/DACS serial connection allows monitoring of the transaction between the DACS and the Remote.

Programmer

The programmer is a small microprocessor-based module programmed to translate serial programming commands to the two-wire C2 programming protocol used by Silicon Labs chips. This is described in the documentation for the MIDE product (MyForth IDE) at the kibler web site.

The programmer is itself one of the hardwired Forth modules of which I speak. In the development environment, it is controlled via a GForth script running on a Raspberry Pi. This script is initiated via a SSH link from the Linux development PC. Applications are developed in MyForth which runs on the PC. Once an application is compiled, the binary programming file is sent via SCP to the Raspberry Pi.

Serial Ports

In addition to the programmer port, a serial port is needed for commanding the RAFL Master and another port is needed for monitoring the serial commands sent to the DACS module by the RAFL Remote.

These two serial ports are implemented with USB to serial adapters attached (through a hub) to the Raspberry Pi. Two SSH links to the Pi are used to bring up minicom terminal sessions for each adapter. Note that there are no serial adapters required for the Linux development PC: all development can be performed remotely via a network connection to the Pi (presently, a wireless connection).

Revision	Date	Description
2	12Nov14	Final Version
1	10Nov14	Initial Release

email for BOBZ product support