PC Control of MTH Engines by Radio Connection to the TIU

Mark DiVecchio

To me, MTH makes the best engines and rolling stock for my railroads of choice, the Pittsburgh & Lake Erie Railroad and the Aliquippa & Southern Railroad (where my grandfather worked). MTH has many many engines and dozen of pieces of rolling stock for these railroads. My layout uses only MTH's DCS.

DCS has been out for almost 15 years. I've been waiting for a way to control my layout using my PC. I've always used Windows PC's so this effort was done originally using XP and more recently Windows 7 and Windows 10.

A few years ago, Mike Hewett presented a PC interface to the tethered mode of operation of DCS. He showed how to sniff out the RS-232 packets running between the Remote and the TIU, how to save those packets and how to later transmit those packets to the TIU from the PC. Mike's methodology was to record the packets sent by the Remote when each key on the Remote was pressed. Without regard to the contents of the packets, he saved them in a file. Then, later, his PC program could read up those saved packets and send them to the TIU. He created a very nice touch screen interface and he could run his DCS trains from his PC. I contacted Mike back then and he sent me copies of his program and I was able to build up his interface to the TIU, sniff out the needed packets and I had a way to control my trains from my PC.

This worked up to a point. When I added a new engine number, I had to run the packet sniffer again and pick up the packets needed for the new engine number. Mike's program only captured a subset of the many, many types of commands that could be sent to the TIU. Mike did not read back responses from the TIU or process any of those returned packets. I was looking for something more. I needed to understand the protocol between the Remote and the TIU.

With a lot of effort, I was able to understand almost all of the communications between the Remote and TIU. I am now able to create packets to control the DCS engines. The packets are complete with correct addressing, command syntax and CRC.

I figured this out by examination of the packets that I could sniff using Mike's original RS-232 wired interface design and the port settings that he found. Without Mike's insights into the RS-232 data stream, I don't think that I would have been able to get a foothold into this protocol.

So again, I figured this out just by looking at the RS-232 stream over the tether cable. No code disassembly, no logic analyzers, no opening up of Remotes or TIU's.

Once I had the RTC program working over the Remote to TIU tether cable, I wanted to develop a radio based version. More searching on the Internet turned up the information needed. I found the frequencies used: the Remote transmitted on 916.5 MHz and the TIU transmitted on 905.8 MHz. I was pretty sure that the protocol used was the same as used over the tether cable (it was).

A posting on a blog led me to believe that the radio chip used was a Texas Instruments

TRF6901. On the TI web page, the TRF6901 was listed as "NRND" - TI speak for "Not Recommended for New Designs". I downloaded the data sheet and saw that the chip supported FSK (Frequency Shift Keying) and OOK (On-Off Keying). So the Remote and TIU probably spoke either FSK or OOK. Another big hint was the comment on the TI TRF6901 web page that stated: "Replaced by CC1101".

I learned that OOK, which stands for "On-Off Keying" is a simple bit serial protocol where the transmitter is just turned on and off based on the "1" or "0" value of the bit being transmitted. Kind of like Morse Code or what we hams call "CW" - Continuous Wave - transmission.

Since the TRF6901 was obsolete, maybe the CC1101 (http://www.ti.com/product/cc1101) could be used. The data sheet said that it could do OOK. It was worth investigating. Now did anyone make a board using the CC1101? I found the panStamp web site (http://www.panstamp.com/) and product called the "panStamp AVR". Exactly what I needed, an *Atmega328p* MCU and CC1101 on a small circuit board along with a carrier board containing a USB port for a PC. I've used the *Atmega328p* MCU (it is the basis of the Arduino) in other projects so I was familiar with it. Easily covers the two frequencies I needed to cover. Its a very low power draw solution. I ordered a few (they're cheap).

The *Atmega328p* MCU is programmed using the standard Arduino IDE and board files from panStamp. I learned how to program the CC1101 from scratch. I searched the Internet and got little bits of help. I found a program called SmartRF Studio 7 (http://www.ti.com/tool/smartrftm-studio) from TI which helps set the values needed for the CC1101 configuration registers. Months of digging through the CC1101 data sheet and test code for the *Atmega328p* MCU finally led me to a working receiver. Then a working transmitter. I called my program "RTCModem". I could communicate between the PC and the TIU with my Remote Train Control (http://www.silogic.com/trains/RTC_Running.html) program over the radio. This page has many videos showing the program in action.

Epilogue

My purpose in what I did was to develop a new way to run my trains. I'm not competing with anyone else, especially MTH since they don't have a product for PC control that they sell or give away. I'm not selling anything that I've done. Everything that I have done, I am freely distributing under the GNU Public Licenses.

The DCS patent does not talk about the protocol between the Remote and TIU. I've learned from my analysis that the protocol is an industry standard RLL(0,1) which is just a fancy way of saying how often the RS-232 signal changes from 1 to 0. Part of the encoding is actually patented and not by MTH - #5,625,644 issued in 1997. The next part uses Morton or Z-Curve encoding. You can Google that. Third are the commands themselves. They are in ASCII and are based on an Application Note published by Microchip Technology in 2002, look for AN759 (http://ww1.microchip.com/downloads/en/AppNotes/00759b.pdf). Another document, AN752 (http://ww1.microchip.com/downloads/en/AppNotes/00752a.pdf), describes the MCRF4XX CRC protocol used. These public documents describe the protocol completely, it was just adapted to model train control. All of these documents were found on the Internet.

Writing a program that encompasses every aspect of train operation is not something one person can do alone. Maybe because of that, PC control will never be practicable. It is not

possible for me to single-handedly write a program that will satisfy everyone. But I can probably do enough to let enterprising and interested hobbyists have the ability to operate their trains in a different way.

I wrote my program, RTC, using Borland C++ Builder because that is the software that I used years ago when I was still working. There are probably other better environments which would better handle the real-time control aspects (like asynchronous RS-232 characters arriving and Critical Section protection). C++ Builder is not really good at that. When I started writing the program, I had no idea what the protocol was. I had to write my program to handle many different protocols, of which all except one turned out to be dead ends.

For details on the previous steps look at this document
(http://www.silogic.com/trains/RTC/Remote%20Control%20%28radio%29.pdf).

Getting the RTC Program

If you have already seen my web page and watched the videos and just want to download the RTC program, Click Here
(http://www.silogic.com/trains/RTC_Running.html#RTC_Download).

You can get a detailed description of the serial protocol to the TIU here
(http://www.silogic.com/trains/RTC/Serial%20Data%20Stream%20Decoding.pdf).
And a description of the command set command set here
(http://www.silogic.com/trains/RTC/Remote%20to%20TIU%20Command%20Set.pdf).