# Timestamping SuperDARN Pulse Sequence User Guide

Brad Krug

April 2, 2008

## Contents



## 1 Introduction

A method for timestamping the SuperDARN transmission pulses has been achieved using a Symmetricom GPS-PCI-2U card. The desire to timestamp the pulse sequences comes with the launch of the ePOP satellite payload, or more specifically the Radio Receiver Instrument, which will be able to observe HF transmissions from the SuperDARN ground systems. In order to facilitate scientific analysis, these transmissions must be accurately logged.

The GPS PCI card came with a sample application for Windows and a Software Development Kit(SDK) could be purchased for creating custom applications. However, there were no drivers or applications provided for Linux, which is the operating system used by the the SuperDARN cd computers. Therefore, a driver and applications for Linux SuSE 10.3 were developed at the U of S. This document gives a brief description of the GPS-PCI-2U card, demonstrates how to install and use the driver and applications, and provides information on customizing the software.

## 2 The Symmetricom GPS-PCI Card

There were two main requirements for timestamping the SuperDARN pulse that were considered when purchasing a GPS unit:

1. The timestamps had to be accurate to within $\pm 8$ $\mu$s (the timestamping accuracy requirement of the ePOP Radio Receiver Instrument).

2. The device needed to be able to timestamp at a high enough frequency. The current pulse sequences occur roughly every 90 ms and, at a minimum, each sequence would need to be timestamped. The SuperDARN pulse sequence was analyzed on a digital oscilloscope and the pulse sequence was found to be accurate to within $\approx \pm 1.5$ $\mu$s. So technically, only the first pulse in the sequence would need to be timestamped; however, a device that could timestamp every pulse in the sequence would be preferred because:

   (a) it would accommodate the ability to change the sequence length

   (b) it would eliminate the possibility of logging the wrong pulse in the sequence

   (c) anomalous or missing timestamps would be easier to identify

The Symmetricom GPS-PCI-2U card (also called the Symmetricom 560-5908) was chosen because it exceeded both requirements. It has a 3 $\mu$s accuracy with resolution down to the hundreds of ns and, according to the manufacturers' user guide, can timestamp pulses as fast as most computers can read and process them. Testing showed that the timestamps could be logged as close as 2 $\mu$s apart, meaning the entire SuperDARN pulse sequence could be timestamped (the closest two SuperDARN pulses are 1500 $\mu$s apart).

In addition to meeting the two requirements, the GPS-PCI-2U provides the following additional capabilities:

- An onboard clock that can synchronize to a GPS reference, IRIG A or B timecodes or a 1 PPS reference.

- Many possible outputs including 1Hz - 10MHz frequencies and IRIG A or B timecodes.

- A Time capture register to view current time with capabilities of adjusting timezone, daylight savings, and phase delay due to antenna cable.

- Antenna position information (longitude, latitude, altitude).

- The ability to manage interrupts from several sources including the external event interrupt used when timestamping.

# 3 Installation

## 3.1 Test System

The same computer was used to both create and test the software. The test system had the following specs:

- Linux SuSE 10.3 (also successfully compiled on SuSE 10.2)
- Kernel 2.6.22.16-0.1-default (older kernels **may not work**)
- Open PCI slots (only one is needed)
- Disk Space: Total size of software bundle is under 30 MB
- RAM: test system had 512 MB
- gcc and make packages

## 3.2 The Installed Software

The software is bundled as sym560.tar.gz, which contains a main directory called sym560 and subdirectories such as documentation and driver. The software installed is listed below and a more detailed look at each item is found in **Sections 4 - 6**.

1. The driver module **sym560_driver.ko** is compiled into the **sym560/driver** directory (along with some intermediate files).
2. The **sym560** startup script is created and copied to the **/etc/init.d** directory. This script is used to load and unload the sym560_driver.ko module.
3. The user application **sym560_cmdline** is compiled into the **sym560/userapp/app** directory and copied to the **/usr/bin** directory. This application has multiple uses from fetching the GPS-PCI card time to timestamping external events both manually and automatically.
4. The application **event_cap** is compiled and copied to **/usr/bin**. This program is called by the main sym560_cmdline program and should not be manually executed.
5. The **stopstamp.bash** script is copied to **/usr/bin**. This script can be called by cron to stop automated timestamping.
6. The **findpulse.pl** perl script is copied to **/usr/bin**. This script will take plain text timestamp files created by the sym560_cmdline program and try to group the timestamps into pulse sequences.

.

### 3.3 How to Install

The short version:

1. Extract the software: tar -xzvf sym560.tar.gz
2. As user run: make (or make all)
3. As root run: make install
4. Yast→System→System Services: enable sym560

The more detailed version:

1. Extract the software contents with the command tar -xzvf sym560.tar.gz.
2. As a **regular user** enter the main directory (the one containing the Makefile, and the documentation, driver, and userapp directories) and enter: **make**. This will:
   (a) Compile the sym560_driver.ko module in the driver directory (other intermediate files will also appear).
   (b) Compile the sym560_cmdline user application in the userapp/app/ directory (again, some intermediate files will also be created).
   (c) Create the sym560 script with the appropriate path to the driver module. This script should be identical to the sym560.org file EXCEPT that it should be executable, and the MODULE variable will have been changed from 'PATHTOMODULE' to the actual path to the .ko module.

   If successful the output should end with the line:

   ```
   chmod 0755 /home/krug/SuperDARN/EPOP/SymmetricomPCI/sym560/driver/sym560
   ```

3. Next, switch to **root** and type: **make install**. This will:
   (a) Copy the applications and scripts to /usr/bin/ (this can be changed by altering the BINDIR variable in the Makefile).[1]
   (b) Copy the sym560 script to the directory /etc/init.d/. This is where scripts that are to be executed on startup should be placed.
4. The final step is to ensure that the sym560_driver.ko module is automatically loaded on startup. Go to **YaST→System→System Services (Runlevel)**. Select **sym560** and click **enable**.

### 3.4 How to Uninstall

To uninstall, enter the main directory and (as **root**) type: **make clean**. The output should be similar to that shown below.

```
cd ./userapp/app/; rm -f *.o *~ sym560_cmdline event_cap
cd ./driver/; rm -rf *.o *~ core .depend .*.cmd *.ko *.mod.c .tmp_versions sym560
rm -f /usr/bin/stopstamp.bash /usr/bin/sym560_cmdline /usr/bin/event_cap
/usr/bin/findpulse.pl /etc/init.d/sym560
```

[1] Certain applications look to /usr/bin and will also need to be updated. For instance, sym560_cmdline executes the event_cap process and the findpulses.pl script, which it expects to find in /usr/bin.

# 4 The Driver

In order to communicate with the GPS-PCI device in Linux a driver needed to written. The driver was made specifically for the Symmetricom 560-5908 and as such provides more specific functionality than most linux drivers. The driver is used to open, close, read, and write to the device but also has several built in IO commands.[2]

The driver was written to produce output to the **/var/log/messages** log file. This file can be viewed (as root) in "realtime" by opening it with **less** and then typing 'Shift-f'. The realtime update can be halted with the 'Ctrl-c' key combo, followed by 'q' to quit less. After following the installation procedure of **Section 3.3**, the driver module can be manually loaded and unloaded by typing (as root):

```
/etc/init.d/sym560 start (or 'stop' to unload)
```

When the module is loaded the following message should be seen in the log file:

```
kernel: ************************************************
kernel: Probing the Symmetricom 560-590U (sym560) PCI card
kernel:
kernel: ************************************************
kernel: Vendor ID= 10b5, Device ID= 9050, Revision = 1
kernel:
kernel: Sym560 Memory Specs:
kernel:     Base Address Register 2 = 3959422976
kernel:     Length                  = 512 bytes
kernel:     Base Address Register 0 = 3967811584
kernel:     Length                  = 128 bytes
kernel:     Mapped Virutal Address  = f8838000
kernel:     Mapped LCR VirtAddress  = f9326000
kernel:
kernel: IRQ NUM = 17
kernel: Sym560 registered as:
kernel:     Major ID = 251
kernel:     Minor ID = 0
kernel:
kernel: Valid ioctl command are as follows:
kernel: event capture : 8008f800
kernel: simpletest : f801
kernel: Check Signal: f802
kernel: Check INTCSR: f803
kernel: sym560 driver was successfully registered
```

During the development of the software, the driver was made to print additional information to the log file. This included a message everytime the device was read or written to.

[2]Two important IO commands, which may be needed for creating custom applications (discussed in Section 7), are:

1. event capture: timestamps external events
2. INTCSR: enables interrupts on the PCI card

These log messages were suppressed in order to ensure the timestamping interrupts would occur as quickly as possible; however they can be turned back on (see **Section 7**).
**WARNING:** The driver has not been written to address "race conditions" (ex: two process trying to write to the same register at the same time). This could happen if two users were logged in and both running the sym560_cmdline application, or if a user tried running the sym560_cmdline application while automatic timestamping was taking place.

# 5 The Application: sym560_cmdline

The application **sym560_cmdline** can be run both automatically and manually. It is recommended that you use this application manually to test the device, before running in auto mode.

## 5.1 Manual operation

The application can be run on the command line in manual mode by typing sym560_cmdline without any arguments. In manual mode, the application provides the following:

1. Information such as time, position and antenna status. This information can be found under the **Main Menu**.
2. Timestamping external event capabilities in **Event Capture Mode**.
3. Control of the BNC output frequency in **Rate Generator Mode**.

When run, the user will first be prompted with the question:

```
Initialize GPS (y/n)?:
```

If the GPS antenna is attached choose 'y' and the device will attempt to synchronize its onboard clock with the GPS reference. This may take some time but the operation will time out after 5 minutes if unsuccessful.

### 5.1.1 Main Menu

When the program is started up (and after the initialize gps prompt) the following menu will appear:

```
*****************************************
Main Command Menu:

    antenna, a : View antenna status
     signal, s : View satellite signal strength
   position, p : View antenna position
       time, t : View onboard time
       init, i : Initialize GPS synchronized generator

      event, e : Event capture mode
  generator, g : Rate generator mode
```

```
       help, h : Prints this menu
       quit, q : Quit program
*****************************************
```

**antenna/a:** Check antenna port for shorts or open loads. If it is reported as being BOTH short and open see section XX: Troubleshooting.

**signal/s:** Checks the signal strength of up to 6 locked satellites. A signal strength greater than 5.0 is considered good according to the manufacturers manual. The SV number is just a satellite ID.

**position/p:** Displays the antenna longitude, latitude, and altitude.

**time/t:** Displays the time from the year to the hundreds of nanoseconds.

**init/i:** Will attempt to synchronize the onboard clock with the GPS reference. It will attempt for a maximum of 5 minutes.

### 5.1.2 Event Capture Mode

Event Capture Mode has a submenu (shown below) that allows you to set up and timestamp events.

```
*****************************************
Event Capture Command Menu:

      setup, v : View event setup
     source, o : Choose event source
      start, s : Start event capture
       data, d : View timestamp data

       help, h : Prints this menu
       back, b : Back to main menu
       quit, q : Quit program
*****************************************
```

**setup/v:** Display the currently selected event and edge that will be timestamped.

**source/o:** Provides the following menu to choose the event source that will trigger a timestamp:

```
          Rising Edge
       1) External Event
       2) Rate Synthesizer
       3) Rate Generator
       4) Time Compare

       Falling Edge
```

```
5) External Event
6) Rate Synthesizer
7) Rate Generator
8) Time Compare
```

It is recommended that only the External Event (falling or rising edge) be chosen. The other events have not been implemented but are provided to show the capabilities of the GPS-PCI card.

**start/s:** Start timestamping events. To stop the timestamping you will be prompted to press any key followed by enter. Once the timestamping has been stopped, the user will then be asked to enter the name of a file to store the plain text timestamps.

**data/d:** Prompts for the name of a file and will display the contents. This option was provided as a quick way to check timestamping trials.

### 5.1.3 Rate Generator Mode

The Rate Generator Mode was created so that an output reference frequency could be specified and enabled.

```
*****************************************
Rate Generator Command Menu:

      setup, v : View generator setup
       rate, r : Choose rate
     enable, e : Enable Rate Generator

       help, h : Prints this menu
       back, b : Back to main menu
       quit, q : Quit program
*****************************************
```

**setup/v:** Displays the current status of the rate generator including the frequency, whether or not it is turned on.[3]

**rate/r:** Set the rate. A variety of options are available from 'Disabled' (the generator is still considered 'on' but just not outputing any frequency) to 10 MHz.

**enable/e:** Ensures the generator is running (it most likely already will be).

[3]When in synchronized generator mode the generator is always considered on. The card is in synchronized generator mode when the onboard clock is being synchronized to some reference such as a GPS reference.

## 5.2 Automated Operation

The sym560_cmdline program also has automated timestamping functionality. This mode was created specifically for timestamping the SuperDARN pulse sequence. To start the program in automated timestamping mode run the command with the 'auto' argument as follows:

```
sym560_cmdline auto
```

This will cause the program to syncronize the onboard clock to the GPS reference, and then immediately begin timestamping external events.

The automated timestamping is stopped by running the **stopstamp.bash** script. Once stopped, the timestamp data is saved into two different files:

1. **timestampdata.txt**, which is the raw text timestamps with the format:

```
YEAR = 2008
 DAY = 050
TIME = 19:28 UTC
 SEC = 30.4556572

YEAR = 2008
 DAY = 050
TIME = 19:28 UTC
 SEC = 30.4609130

YEAR = 2008
 DAY = 050
TIME = 19:28 UTC
 SEC = 30.4698876
```

2. **pulses_YYYY_DDD.txt**, which is the condensed version of the timestamps. This output looks as follows:

```
PULSES TIMESTAMPED FOR
  YEAR: 2008
  DAY: 050

N 19:28:30.4556572

N 19:28:30.4609130

N 19:28:30.4698876
```

   Note that the above timestamps are not SuperDARN pulse sequences and are thus automatically labeled as non sequence pulses (N). The condensed output is created by processing the raw text timestamps using the findpulse.pl script described in the next section. It is likely that both output formats will not be needed and therefore one may be deleted, either manually or by changing the sym560_cmdline application itself to delete one of the files.

The following is an example of how to set up the automatic timestamping of the SuperDARN pulse sequence for a specific daily time. The example time period used is from 3:00 to 3:30.

1. Follow the installation procedure from **Section 3.3** if you haven't already done so.

2. Create a cron file with the following entries:

```
00 3 * * * cd /path/to/timestampdir/ ; sym560_cmdline auto
30 3 * * * stopstamp.bash
```

   In linux a cron job can be added by running the command **crontab -e**. This opens up the crontab in vim for editing.

# 6 The Sequence Identifier Script: findpulse.pl

The final piece of software created was the perl script: **findpulse.pl**. It was made to condense the timestamp text files and make them easier to read. This script takes the raw text timestamps created when capturing events in either manual or auto mode and searches them for a pattern of pulses, or more specifically a SuperDARN pulse sequence. The sequence is defined in the script as an array containing the seperation time in ms between pulses as follows:

```
my @psep = (21.0, 12.0 , 3.0, 4.5, 6.0, 16.5, 1.5);
```

The date (year and day) of the timestamps is written once at the top of the file. If pulse sequences are found the resulting output will look like:

```
16:29:53.1965083
16:29:53.2175084
16:29:53.2295090
16:29:53.2325082
16:29:53.2370087
16:29:53.2430086
16:29:53.2595083
16:29:53.2610088

16:29:53.2930680
16:29:53.3140679
16:29:53.3260681
16:29:53.3290681
16:29:53.3335675
16:29:53.3395674
16:29:53.3560676
16:29:53.3575676
```

If timestamps are not found to be part of a sequence then they will be labeled with an N. If a sequence is detected but certain pulses were not timestamped (this does happen, although not very often), then the missing pulses are labeled as MISSING.

This script can be run either manually or automatically. When run manually from the command line the script will prompt for an input file and an output file. To run the script automatially it must be run with two arguments as follows:

```
findpulse.pl auto inputfile.txt
```

The output file is automatically named **pulses_YYYY_DDD.txt** where the year and day are taken from the first timestamp in the file. When **sym560_cmdline auto** is run, the findpulse.pl is the last process to be executed.

# 7 Customizing the Software

This section is for those who wish to modify the software to meet specific needs or two fix any bugs found. The software has all been internally documented so this section is just meant to give direction on how to start modifying the code.

## 7.1 Modifying the Driver

Getting started:

- The file **sym560_driver.c** is the source code for the driver.
- In Linux there are two distinct divisions: user space and kernel space. This driver was written in kernel space (along with most other drivers) and thus uses slightly different libraries and functions then most user applications and programs. For instance, printk is used to print messages in kernel space, as opposed to printf in user space. Also, kernel space and user space have access to different memory regions.
- The driver was written in the C programming language.
- The book *Linux Device Drivers* by Rubini et al, was essential reading when creating the driver. There are free online pdf versions and it is highly recommended reading. Most of the coding principles used for this driver come from that book.

Some important functions:

- **sym560_probe** initializes the device if it's found when the module is loaded.
- **sym560_remove** is called when the module is unloaded and undoes everything performed by the probe function.
- **sym560_llseek** is used to change the offset for reading and writing to specific locations.
- **sym560_read** and **sym560_write** are used to read from, or write to, the device.
- **sym560_event_handler** is an interrupt handler called if interrupts are enabled.
- **sym560_ioctl** is used to handle specific commands such as capturing events, or enabling interrupts.

### 7.2 Modifying the application

Getting started:

- The source code can be found in the userapp/app/ directory.
- The file **sym560_cmdline.c** is the main function which processes the user input (or starts automatic event capture mode).
- The file **sym560_functions.c** is where the majority of the source code is found. The code has been modularized into many functions each performing a specific task. This file also has a header file containing global definitions, includes, and function declarations. Some important functions include:
  - **read_pci** and **read_pci_verbose**.
  - **write_pci** and **write_pci_verbose**. The two verbose commands can be used for debugging purposes. They will print out in hex and binary what has been read from, or written to the device.
- **event_cap.c** is the source code for the process used for nothing other than to continuously timestamp events.
- **Chapter 3: Operation** of the Symmetricom GPS-PCI card user manual provides all the information on the GPS card functionality needed to program applications for the card.

## 8 Known Issues

- If the gps antenna is unplugged, either at the pci card or at the antenna, while the computer is on, then the Hardware Status Register hangs in a state indicating both a short and an open load. In this state the card will not lock on to a GPS signal. The only known way to clear this state is to power cycle the computer.
- The driver has not been written (for the most part) to protect against race conditions. A race condition occurs when two processes are trying to access the device at the same time, for example if two processes try to write to the same register at the same time. To avoid race conditions never run more than one instance of the sym560_cmdline application (or any custom built applications) at the same time. There is a chapter in *Linux Device Drivers* that deals with handling race conditions.
- If the findpulse.pl script is used on a timestamp raw text file that has a captured pulse sequences similar to the one being detected but not identical then the resulting pulses_YYYY_DDD.txt output file could indicate many missing pulses and flag others as N (not part of a sequence). **Section 6** describes how to change the sequence that the findpulse.pl script will look for.
- A lot of the GPS card functionality has not be implemented in the application. However, the documented code along with the Symmetricom User Manual can be used to create/modify applications with additional functionality.

- The driver may not compile or run on any system running a lower kernel. In the kernel update from SuSE 10.2 to 10.3, there were some minor changes made to the driver. For instance, one of the header files was deprecated and replaced by another, and some macros were also replaced by others.