# Masterclock PCIe Application Notes for Linux

(March 2018)

## Building Software Package

1. Unpack the archive.

2. Open the ‘makefile’ and edit the line ‘CFG_FILE’ as needed to reflect the desired location of the software package configuration file.

3. Edit the ‘gpsmcr.conf’ file and adjust at least the LogFile parameter to indicate where the application-specific log file should be located. For time code reader cards, be sure to set the two time code reader configuration items in ‘gpsmcr.conf’.

4. Run ‘make’ to build the software package. The following files are produced:

‘gpsmcr’ - daemon service program

‘gmtool’ - utility to display operational status and demonstrate API features

5. Open the ‘gpsmcr.conf’ file and set the LogFile and SocketFile parameters relative to the needs of your installation.

6. Change other configuration file options if needed.

## Usage

‘gpsmcr’ is a Linux daemon program and requires super-user privileges to support certain features such as making adjustments to the system real-time clock. The program can be started from the shell command-line or from a system start-up script.

‘gpsmcr’ has the following command-line options to support optional features:

*-h*

Show command-line options and brief description.

*-nolog*

Turns off logging to the application-specific log file. Important messages will still be sent to the system log.

*-nofork*

Prevents the program from forking and running in the background. Useful for diagnostics and testing new installations.

*-nosync*

Run but do not apply any time synchronization adjustments to the system real-time clock. Useful for diagnostics or situations where access to the API set is the only desired feature.

# Configuration File

Various parameters of the software operation can be configured through the 'gpsmcr.conf' configuration file. The options mentioned in Building Software Package will need to be set for each unique installation.

Additional options and a description of their function can be found in the configuration file itself. Most options can be left at the default settings.

For time code reader cards (PCIe-TCR) be sure to appropriately set the two time code reader configuration items in 'gpsmcr.conf'.

The *JamThreshold* option can be set to zero (0) to disable all jam synchronization of system time to reference. This may be useful in certain environments where any jam syncing may be detrimental to operations. Note that correcting large system / reference time discrepancies may take much longer when this option is disabled.

# GMTOOL

'gmtool' is a diagnostic program which can be used to monitor the operation of the 'gpsmcr' daemon program. Basic information such as system time, reference time, and time synchronization status are displayed. The program also serves as a demonstration of the API features.

# API Message Set

GPSMCR for Linux package provides an ASCII-based "API" message set for exchanging information with PCIe-series of cards. The API could be useful for determining operational status of the card and software, custom time synchronization applications, etc.

To use the API from custom application, open an IPC (inter-process communication) socket to the GPSMCR communication socket. The location is defined in configuration file option 'SocketFile'.

Note that the GPSMCR daemon program must be running to provide the API services.

**Message A:**

Daemon and card version information.

Send: A

Receive (e.g.): A<1,0,0,1,2,0,12,14,07,04,25,2016,Y>

| Field # | Description |
|---|---|
| 1 | daemon program version (major) |
| 2 | daemon program version (minor) |
| 3 | daemon program version (revision) |
| 4 | PC card firmware version (major) |
| 5 | PC card firmware version (minor) |
| 6 | PC card firmware version (revision) |
| 7,8,9 | daemon start-up time: hour, minute, and second |
| 10,11,12 | daemon start-up time: month, day, and year |
| 13 | 'Y' when installed card is PCIe-GPS |

**Message B:**

Current reference status information.

Send: B

Receive (e.g.): B<0,1,2,3,10>

| Field # | Description |
|---|---|
| 1 | real-time clock status (one of time reference status) |
| 2 | GPS status (one of time reference status) |
| 3 | time code status (one of time reference status) |
| 4 | high-precision oscillator status |
| 5 | high-precision oscillator stabilization achievement level |

Time reference lock statuses are one of:

0 - lock invalid (never locked)
1 - lock not present (had been locked, but lock is lost)
2 - lock lost (one-time notification)
3 - lock restored (one-time notification)
4 - lock valid

**Message C:**

Current reference time. All time provided is relative to top of the herein-referenced second. Use a second capture of the high-performance system time to compute a delta from last top-of-second.

Send: C

Receive (e.g.): C<1485211753,1485211722,652673>

| Field # | Description |
|---|---|
| 1 | reference (card) time (seconds since 00:00:00 UTC January 1, 1970) |
| 2 | high-performance system time capture, seconds (since 00:00:00 UTC January 1, 1970) |
| 3 | high-performance system time capture, nanoseconds |

**Message D:**

System time synchronization status.

Send: D

Receive (e.g.): D<0,1485211722>

| Field # | Description |
|---|---|
| 1 | Current synchronization operation mode |
| 2 | Last measured reference/system time delta (nanoseconds) |

**Message E:**

Reports the contents of the NMEA GPS string $GPGGA.

Send: E

Receive (e.g.): E<123519,4807.038,N,01131.000,E,1,08,0.9,545.4,M,46.9,M,,*47>

| Field | Description |
|---|---|
| 123519 | Fix taken at 12:35:19 UTC |
| 4807.038,N | Latitude 48 deg 07.038' N |
| 01131.000,E | Longitude 11 deg 31.000' E |
| 1 | Fix quality, one of:<br>0 = invalid<br>1 = GPS fix (SPS)<br>2 = DGPS fix<br>3 = PPS fix<br>4 = Real Time Kinematic<br>5 = Float RTK<br>6 = estimated (dead reckoning) (2.3 feature)<br>7 = Manual input mode<br>8 = Simulation mode |

| | |
|---|---|
| 08 | Number of satellites being tracked |
| 0.9 | Horizontal dilution of position |
| 545.4,M | Altitude, Meters, above mean sea level |
| 46.9,M | Height of geoid (mean sea level) above WGS84 ellipsoid |
| (empty field) | time in seconds since last DGPS update |
| (empty field) | DGPS station ID number |
| *47 | checksum of NMEA string |

**Message F:**

Set time on PCIe-xxx card.

| | |
|---|---|
| Send (e.g.): | F<12,24,2016,4,15,20,500> |
| Receive: | Y (acknowledge)<br>Z<reason code> (negative acknowledge - error) |

| Field | Description |
|---|---|
| 12 | month |
| 24 | day of month |
| 2016 | year |
| 4 | hour |
| 15 | minutes |
| 20 | seconds |
| 500 | milliseconds |

Reason codes for negative acknowledge, one of:

1 = command not recognized
2 = bad parameter(s)
3 = too many parameter(s)
4 = not enough parameter(s)

# Troubleshooting Tips for Card Installation

You may normally expect for the GPSMCR software to automatically detect the PCIe-xxx's serial ports. If the card is not automatically detected, or not detected correctly, some additional serial configuration on the Linux system may be necessary.

**Serial Devices**

Linux manages serial configuration through device files located in the filesystem under /dev.

Normally, you might see:

/dev/ttyS0

..
/dev/ttySx

Note that your flavor of Linux/Un*x may use slightly different nomenclature to refer to serial devices. Consult your operating system documentation.

From this point, instructions will refer to installing the PCIe-GPS/MCR (hereinafter PCIe) in a Linux 2.6 kernel operating system.

Some Linux systems will detect, initialize, and establish serial devices for the first four physical serial ports that are found. This presents a configuration challenge because the PCIe-GPS/MCR itself contains four independent serial ports. It is possible for Linux to find other serial ports in the system first resulting in only some (or even none) of the PCIe serial ports being detected. Some manual configuration will be required to resolve this.

One method to deal with this situation is to disable all hardware serial interfaces other than the PCIe card. This may involve disabling all on-motherboard serial ports in the computer's BIOS, removing add-in serial interface boards, or both. This may not be desirable if there is an expectation that these serial port(s) may be used in the future.

A better method to deal this situation is to increase the number of serial ports that the Linux operating system will recognize by default. This can be done by providing configuration changes to the kernel module at boot time (usually via 'grub' or 'lilo'). To the bootloader add kernel option '8250.nr_uarts=X", where X is the desired new maximum to support. (In cases where the serial driver is compiled directly into the kernel, the option is 'nr_uarts=X').

Theoretically, 'X' would be set to the total number of physical serial ports currently installed in the system including the four associated with the PCIe card. When is doubt, '8' would be a good starting point. Note that the PCIe software will have to examine each of these serial ports to auto-detect the location of the card and its features. Therefore, avoid setting this number unnecessarily high.

For example, in RedHat Enterprise Linux 7:

1. Edit the file '/etc/default/grub'. To the line GRUB_CMDLINE_LINUX add the necessary parameter, for example: 8250.nr_uarts=5
2. Save the file.
3. Update the grub bootloader options so that the new change is permanent, run: grub2-mkconfig -o "$(readlink -e /etc/grub2.cfg)"
4. Restart the system for the changes to take effect.

After making the configuration changes and rebooting the system you should see at least four, and usually more, ttySx devices in /dev.

**UART Clock Frequency Settings**

There may be an additional run-time configuration change to make to the PCIe serial ports. The crystal (clock) on the PCIe board is not the default frequency used for most serial UARTs. Some Linux systems are unaware of this. To accommodate this, a change to the serial driver's 'baud_base' parameter is needed for each PCIe port.

Procedure:

(1) Identify the serial devices (/dev/ttySx) associated with the PCIe.

Use the 'lspci' tool to determine the BIOS-assigned configuration for the PCIe. We are specifically concerned with the IRQ and I/O port address space assigned and will examine these values to determine which /dev/ttySx (if any) are assigned to our card.

Dump the output of 'lspci -v' to a file. Examine the contents looking for the "Pericom Semiconductor Device 7954." The data below this PCI item will identify the IRQ (interrupt) assigned, and also the beginning of the I/O address space (example 0xd000) and the address space length which will be 64 bytes for the 4-port Pericom chip. All Pericom ports will share the assigned IRQ (example 17).

Now we will examine the configuration for the existing serial devices. For example, in /dev we may have:

/dev/ttyS0
/dev/ttyS1
/dev/ttyS2
/dev/ttyS3
/dev/ttyS4

For each device, use the 'setserial' tool to view the device's assigned configuration. For example 'setserial /dev/ttyS0' might produce the following output:

/dev/ttyS0, Line 0, UART: 16550A, Port: 0x03f8, IRQ: 4

We can tell that the IRQ shown above (4) is not the same IRQ assigned to our Pericom device (17). The I/O port address space is also not within the I/O port space mapped to the PCIe card. Therefore, we know that this serial port is physically located elsewhere in the computer - most probably on the computer's motherboard.

Continue the 'setserial' procedure for each serial device in /dev until we have identified which (if any) serial devices have been initialized on the PCIe board. For example, a 'setserial' examination of ttyS1, ttyS2, ttyS3, and ttyS4 might produce the following results:

/dev/ttyS1, Line 1, UART: 16650, Port: 0xd000, IRQ: 17

/dev/ttyS2, Line 2, UART: 16650, Port: 0xd008, IRQ: 17

/dev/ttyS3, Line 3, UART: 16650, Port: 0xd010, IRQ: 17

/dev/ttyS4, Line 4, UART: 16650, Port: 0xd018, IRQ: 17

Examining the IRQ and I/O port address space information above shows that all of these devices are referring to resources assigned to the PCIe card.

2) Using the 'setserial' tool, change the 'base_baud' configuration on each serial device associated with the PCIe card. Using the above example:

setserial /dev/ttyS1 baud_base 921600

setserial /dev/ttyS2 baud_base 921600

setserial /dev/ttyS3 baud_base 921600

setserial /dev/ttyS4 baud_base 921600

The serial ports should now be configured properly and ready for use. GPSMCR application will auto-detect the PCIe card at run-time.