PUMPKIN™
INCORPORATED
750 Naples Street • San Francisco, CA 94112 • (415) 584-6360 • http://www.pumpkininc.com

UM-10
User Manual

GPSRM 1 GPS Receiver Module

TRADEMARKS

The following are Pumpkin trademarks. All other names are the property of their respective owners.

- Pumpkin™and the Pumpkin logo
- Salvo™ and the Salvo logo
- CubeSat Kit™ and the CubeSat Kit logo
- SUPERNOVA™

DISCLAIMER

PUMPKIN RESERVES THE RIGHT TO MAKE ANY CHANGES WITHOUT FURTHER NOTICE TO ANY PRODUCTS HEREIN TO CORRECT ERRORS AND IMPROVE RELIABILITY, FUNCTION, APPEARANCE OR DESIGN. PUMPKIN DOES NOT ASSUME ANY LIABILITY ARISING OUT OF THE APPLICATION OR USE OF ANY PRODUCT OR CIRCUIT DESCRIBED HEREIN; NEITHER DOES IT CONVEY ANY LICENSE UNDER ITS PATENT RIGHTS, NOR THE RIGHTS OF OTHERS.

PUMPKIN INCORPORATED

744 Naples Street
San Francisco, CA 94112 USA
tel: (415) 584-6360
fax: (415) 585-7948

web: http://www.pumpkininc.com/
email: info@pumpkininc.com

web: http://www.cubesatkit.com/
email: info@cubesatkit.com

CHANGELOG

Rev.	Date	Author	Comments
A	20130905	AEK	Initial version.
B	20131025	AEK	Added v0.1.1 updates w/new command syntax, and explicitly called out the need to terminate commands with the ASCII LF character.
C	20131030	AEK	Corrected v0.1.1 syntax for SUP:CLOCk command.
D	20131104	AEK	Updated with v0.2.0 details.
E	20170309	AEK	Substantially updated with v0.11.0 details, CLI, pySCPI, etc.
F	20170314	AEK	Added sections on configuring for space use, OEM719 version and firmware upgrades, etc.
G	20170318	AEK	Added sections on OEM719 re-flash, CLI, updated command lists.
H	20170320	AEK	Added screen captures of GPSRM 1 firmware reprogramming via MPLAB X IPE.
I	20190612	AEK	Updated some sections to reflect transition from OEM615 (now NLA) to OEM719.

Purpose

This manual describes how to use the Pumpkin GPS Receiver Module 1 (GPSRM 1).

Datasheet

A separate datasheet is available for the GPSRM 1, at http://cubesatkit.com/content/datasheet.html. The datasheet should be referenced first, as it covers a wealth of issues related primarily to the hardware of the GPSRM 1.

Introduction

Pumpkin, Inc.'s GPSRM 1 provides CubeSat end-users with a high-performance space-capable GPS receiver integrated into a CubeSat Kit-compatible module, with a versatile interface. The GPSRM 1 is one of several Pumpkin nanosatellite components that include a Supervisor MCU (SupMCU™) that handles telecommands and provides telemetry.

Primary Components

The GPSRM 1 module is a standalone, integrated module for insertion into a system utilizing the CubeSat Kit Bus.

In addition to the GPSRM 1 module, these additional components are included:

- Pumpkin USB Debug adapter w/FPC cable
- Microchip® PICkit programmer
- Pumpkin JFPC-PIC24 adapter with FPC cable
- USB A-to-B cable
- USB A-to-microB cable

The USB Debug Adapter is provided to aid developers when integrating and debugging the GPSRM 1 in their systems. The PICkit is provided to facilitate GPSRM 1 firmware upgrades.

A compatible GPS antenna is not included, but is available separately.

Overview

Status LEDs

Three status-indicating LEDs are present on the GPSRM 1; they are shown in Figure 1 and their functions are described in Table 1, below.

Tip A properly functioning GPSRM 1 will have a blinking green OEM719 status LED; once a GPS lock is established, the GPS Position Valid (PV) LED on the GPSRM 1 will also be ON.

Tip The light pipe in the GPSRM 1's cover protrudes very slightly above the cover, thereby making the OEM719's status LED visible even when integrated into a stack of CubeSat modules.

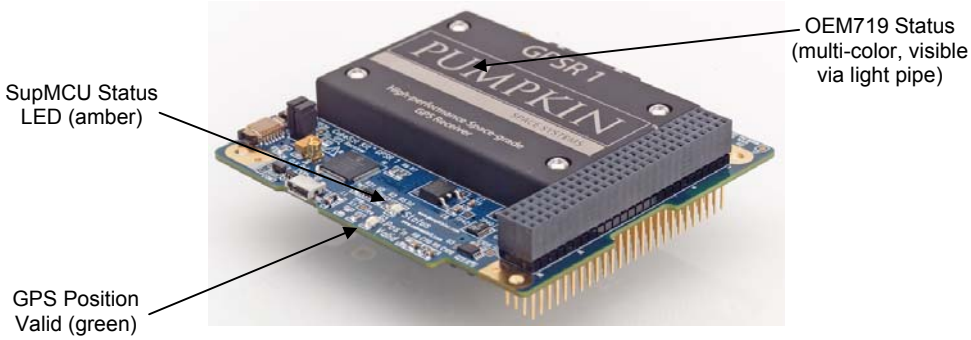


Figure 1: GPSRM 1 status indicator LED locations

Location	Color	Label	Function / Notes
Top of GPSRM 1 cover, visible through light pipe	Red / Yellow / Green	None (OEM719 status)	*Red: OEM719 fault* *Yellow: OEM719 fault* *Green: ON at power-up, blinking when operating normally*
Top of GPSRM 1 PCB	Green	GPS Position Valid (OEM719 PV)	*ON when GPS receiver has a valid lock.* *This LED is only valid when the OEM719 status LED is blinking green.*
Top of GPSRM 1 PCB	Amber	SupMCU Status	*Under control of SupMCU. Defaults to flashing, i.e., SupMCU heartbeat.*

Table 1: GPSRM 1 Status indicators and their functions

Supervisor MCU (SupMCU)

Figure 2: GPSRM 1, showing Supervisor MCU and GPS receiver locations

When the GPSRM 1 is powered, the on-board Supervisor MCU is always on. The supervisor MCU controls a variety of features on the GPSRM 1, including whether or not the NovAtel® GPS receiver is powered on, and whether serial communications can flow between the GPS receiver on the GPSRM 1 and a host on the CubeSat Kit Bus (typically, the C&DH processor).

For test & validation and flight software purposes, the C&DH is normally the processor that communicates with the GPSRM 1's Supervisor MCU via I2C. For standalone GPSRM 1 development work, a standalone I2C Master device[1] is typically used.

GPS Receiver – NovAtel® OEM719

The GPSRM 1 includes a space-grade OEM719[2] multi-channel GPS receiver with COCOM limits unblocked. The OEM719 resides in a sealed RF-shielded and heatsinking aluminum case on the top side of the GPSRM 1 PCB. The OEM719's MCX antenna jack protrudes through a hole in the bottom of the GPSRM 1.

Tip The GPSRM 1 supports all of the different configurations available in the OEM719 family. An L1-only version of the OEM719 is supplied as standard; customers can request other versions of the OEM719 (e.g., L1+L2, GPS+GLONASS, etc.) at additional cost when ordering their GPSRM 1.

There are no outwardly visible differences to indicate which OEM719 is incorporated into a GPSRM 1; the best way to identify

[1] E.g., the Total Phase® Aardvark™. See also pySCPI GUI, below.

[2] The terms "GPS Receiver" and "OEM719" are used interchangeably throughout this manual to refer to the NovAtel OEM719-series GPS receiver contained within the GPSRM 1.

the internal OEM719's capabilities is to connect to it via NovAtel Connect software, and query the receiver version, etc.

Antenna

The GPS receiver has a single MCX jack for connecting to a GPS antenna. The GPS receiver provides power to the antenna's LNA; on the GPSRM 1, this power is provided at or close to +5V.[3]

The GPSRM 1 is provided without an antenna; customer-supplied antenna(s) must conform to the bands and frequencies of the GPS receiver in the customer's GPSRM 1.

PPS Output

The GPSRM 1 routes the GPS receiver's Pulse Per Second (**PPS**) output to its MCX connector **J7** (**PPS Out**). This is a controlled-impedance output and is designed to work with 50Ω cables.

Tip This signal can also be connected to the **PPS** signal on CubeSat Kit Bus connector **H1**, via a zero-Ohm resistor.

The **PPS** signal is valid when the GPS receiver has achieved lock and will appear on **J7** as soon as lock is achieved.

VARF Output

The GPSRM 1 routes the GPS receiver's Variable Frequency (**VARF**) output to its MMCX connector **J8** (**VARF Out**). This output can be enabled via a suitable command to the GPS receiver. **VARF** is phase-locked to **PPS** when the GPS receiver is locked.

Tip **VARF** can be routed to **IO.31** on the CubeSat Kit Bus Connector via a zero-Ohm jumper.

[3] Power to the OEM719 and power to its LNA can be provided separately, without any restrictions on powerup/powerdown phasing. I.e., the OEM719 will not be damaged if LNA power is applied while it is unpowered.

Communications – SupMCU

Communications between a CubeSat host and the GPSRM 1's SupMCU is via I2C on the CubeSat Kit Bus. The GPSRM 1 is an I2C slave device by default. The GPSRM 1 accepts SCPI commands and requests for telemetry over I2C at speeds of up to 400kHz via the CubeSat Kit Bus connector **SCL_SYS** and **SDA_SYS** signals.

Communications – GPS receiver

The GPS receiver has multiple communications ports (serial and USB), all of which are simultaneously active.

A host communicates with the GPS receiver in the GPSRM 1 via the GPS receiver's COM1 (serial) port. This port is presented on one of three Tx/Rx pairs that connect to the CubeSat Kit Bus connector; the pair is selected via zero-Ohm jumpers added to the GPSRM 1 during manufacturing as part of an assembly revision (ASSY REV). Once the GPS receiver is powered on and the GPSRM 1's passthrough mode is enabled via SCPI command, a host can establish a direct connection to the GPS receiver via its COM1 port.

The GPS receiver's COM2 port is used by the SupMCU to communicate with the GPS receiver directly. It is not externally available.

The GPS receiver's USB port is available for use with e.g. ground-based GUI software for operating the GPS receiver while connected to a suitable host via USB.

Usage

Using NovAtel® Connect software with the GPSRM 1

The preferred method of connecting NovAtel® Connect software to the GPSRM 1 is via the included USB port. This is also a simple an expedient way to confirm that the GPS receiver inside of the GPRSM 1 is functioning correctly with the chosen antenna.

Tip Pumpkin recommends that you begin your familiarization with the GPSRM 1 by connecting to it via USB and NovAtel's Connect software; no other hardware is required in this configuration.

Figure 3: Micro USB connector to NovAtel® receiver's USB port

Note The USB port **J3** on the GPSRM 1 is a direct USB connection to the NovAtel receiver. There is no connection between the USB port and the Supervisor MCU.

Tip Connecting the GPSRM 1 to an active USB host via its micro-USB connector **J3** results in the GPS receiver being powered up automatically; no SCPI commands to the GPSRM 1 are required. Therefore NovAtel Connect can be used with a GPSRM 1 that is not connected to anything else.

Always observe ESD-safe handling when operating the GPSRM 1!

Connect a USB-to-micro-USB cable between the GPSRM 1 and a laptop running NovAtel's GUI software (NovAtel Connect / CDU or equivalent). Whenever the NovAtel receiver is powered, the

NovAtel software should be able to establish communications with the GPS receiver via one of the laptop's USB ports.

The NovAtel Connect software is useful for a variety of things, including:

- Validating the current operation of the GPS receiver (e.g., current GPS coordinates, how many satellites are in view, what's the PDOP, etc.) along with its connected antenna
- Issuing commands via the command-line interface (e.g. to enable logging on a specified interface)

Note The NovAtel software currently does not support firmware upgrades over USB. Therefore, an alternate configuration is required to perform software updates (typically over COM1).

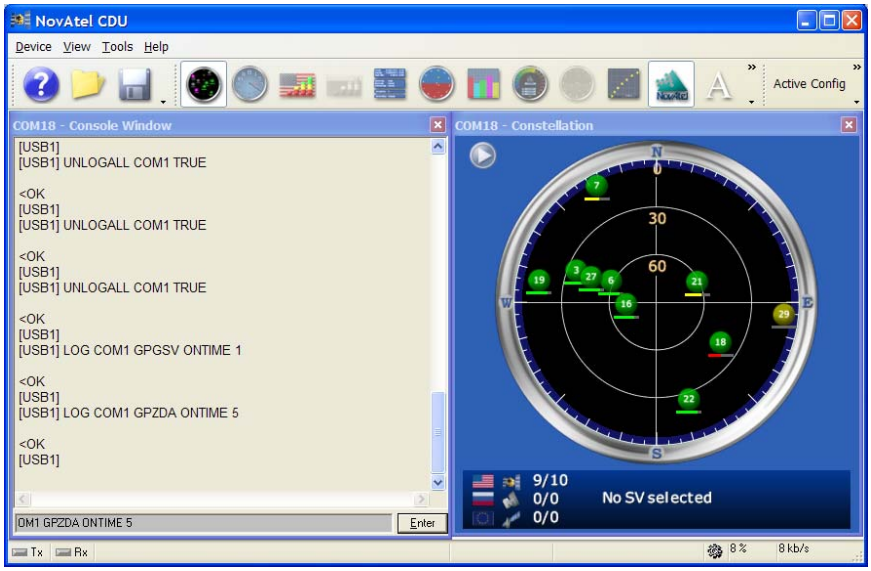


Figure 4: NovAtel CDU GUI software

Installation into CubeSat Kit Bus

Note The antenna connection underneath the GPSRM 1 requires adequate clearance beneath the GPSRM 1 and whatever is below it as part of a CubeSat Kit-compatible module stack. The GPSRM 1 is designed to be mounted directly above a CubeSat Kit Motherboard Module (MB or MBM) hosting a small- or medium-sized Pluggable Processor Module (PPM), e.g. PPM A2 or PPM D1, respectively.

The GPSRM 1 can be located within module stacks of alternate configurations; however, it is up to the user to ascertain whether a particular combination of modules affords the required clearance for the end of the antenna cable underneath the GPSRM 1. Online CAD models of the GPSRM 1 and the connector can be used to assist in this evaluation.

1. Ensure that your CubeSat Kit-compatible system is powered off.
2. Plug your GPS antenna into the underside of the GPSRM 1. The antenna cable must terminate in an MCX right-angle plug; this plugs into a MCX jack on the underside of the GPSRM 1.
3. Remove any other connectors (programming, debug, USB, etc.) from the GPSRM 1.
4. Carefully plug the GPSRM 1 into the CubeSat Kit Bus Connector. Ensure that the pins of the connector are properly aligned.
5. At this point your CubeSat system can be powered up, and the GPSRM 1 will power up as well. The SupMCU will be ready and waiting for SCPI commands.

Powering the GPS Receiver

The GPSRM 1's SupMCU uses minimal power. The circuitry on the GPSRM 1 – along with the SupMCU – conditions and controls power to the OEM719 GPS receiver inside. It manages which power source(s) are routed from the GPSRM 1's USB connector **J3** and CubeSat Kit Bus Connectors **H1** & **H2** to the OEM719's **3V3** and **LNA_PWR** terminals. [4]

Tip The OEM719 inside the GPSRM 1 is insensitive to power sequencing on its **3V3** and **LNA_PWR** terminals; it will not be damaged by any combination of these two power inputs being present or absent. Therefore the GPSRM 1 will not be damaged by, for example, a temporary interruption of a CubeSat's **+5V_SYS** or **VCC_SYS** power.

Note The OEM719 inside the GPSRM 1 requires good, clean +3.3Vdc power to boot and operate properly. This is due in part to the OEM719's large inrush current. The GPSRM 1 is designed with this in mind, and is able to ensure a proper power-up of the OEM719 when sourcing power either from USB or from the CubeSat Kit Bus Connectors.

Power is provided in one of two ways to the GPSRM 1's GPS receiver:

1. When connected to a USB host via its **J3** micro USB connector, the GPSRM 1 regulates +5Vdc USB power down to +3.3Vdc for the OEM719's **3V3** terminal. Additionally, the GPSRM 1 routes +5Vdc USB power to the OEM719's **LNA_PWR** terminal.
2. When installed in a CubeSat module stack that includes an EPS providing +5Vdc on **+5V_SYS** and +3.3Vdc on **VCC_SYS**, the GPSRM 1 can be commanded to power the OEM719 from those two power sources (**VCC_SYS** to **3V3** and **+5V_SYS** to **LNA_PWR**).

Warning Exceeding +5Vdc on **+5V_SYS** or +3.3Vdc on **VCC_SYS** or +5Vdc on **+5V_USB** of the CubeSat Kit Bus Connectors **H1** & **H2** can lead to the immediate destruction of the GPSRM 1 and the OEM719 GPS receiver inside.

[4] These terminals are not directly accessible to the GPSRM 1 user.

Users are *strongly cautioned* to only connect to the GPSRM 1 via standardized connectors, i.e., USB cables and CubeSat-class EPSes that conform to the CubeSat Kit Bus Connector pinout.

Tip Many GPS antennas include a low-noise amplifier (LNA); most require a minimum voltage of around 3Vdc to operate properly. This voltage is provided by the OEM719 via its **LNA_PWR** terminal, which is independent of the OEM719's **3V3** power input. Only a voltage sufficient to power the LNA is required at the **LNA_PWR** terminal; anything more is unnecessary. The +5Vdc that the GPSRM 1 provides to the OEM719 GPS receiver on its **LNA_PWR** terminal is sufficient for a wide range of GPS antenna LNAs.

Enabling the GPS Receiver

The ON/OFF state of the GPS receiver Depends on the mode in which the GPSRM 1 is powered:

- When the GPSRM 1 is connected to an active USB host via its micro-USB cable or indirectly via the **+5V_USB** signal on the **H1** CubeSat Kit bus connector, the GPS receiver is ON and cannot be turned off, irrespective of the **GPS:POW** command.
- When no active USB host or **+5V_USB** power is present, then the GPS receiver is ON only when the GPSRM 1 is commanded to turn the GPS receiver ON, via a SCPI command (see

SCPI Commands, below). Otherwise the GPS receiver remains OFF.

Therefore, when a GPSRM 1 is integrated into a CubeSat and no external USB host is connected to the CubeSat or the GPSRM 1, you must issue a SCPI command to turn GPS receiver power ON.

Tip the ability to control power to the GPS receiver inside the GPSRM 1 is in some applications critical towards overall spacecraft power management; this is one of several useful features of the GPSRM 1's SupMCU.

SCPI Commands

The GPRSM 1 functions as an I2C slave in the CubeSat Kit architecture. I2C commands are issued in plain text (i.e., ASCII) form. The command format follows that of and implements a subset of the Standard Commands for Programmable Instruments (SCPI) standard.

Note Commands to the GPSRM 1 must be terminated with an ASCII Line Feed character (**'\n'**, 0x0A, 10 decimal). Command termination can be just the LF character alone, or the ASCII Carriage Return (CR) follow by LF (CR+LF, **'\r\n'**, 0x0D0A).

An I2C master normally issues plaintext commands to the Supervisor MCU. The I2C master can be a C&DH processor, or an external I2C Master controller (e.g., a Total Phase® Aardvark).

The SCPI commands required to enable the GPS receiver are:

```
GPS:POW ON
GPS:PASS ON
```

These two commands to the GPSRM 1's SupMCU will enable power to the GPS receiver, and permit its COM1 serial port to connect to the CubeSat Kit Bus Connector.[5]

See Table 5, below, for a listing of the GPSRM 1's SCPI commands.

[5] The SupMCU automatically disables the GPS receiver's **–RESET** input when it powers the GPS receiver ON. Therefore **GPS:RES OFF** is not required.

USB Debug Adapter

The GPSRM 1 supports a bidirectional USB debugging interface via Pumpkin's USB Debug Adapter. This adapter connects to the GPSRM 1 via a 4-conductor flat printed circuit (FPC) cable. The USB Debug Adapter gets its power from the USB host (i.e., it is configured as a bus-powered USB device).

Tip The GPSRM 1's SupMCU firmware utilizes the USB Debug Adapter to send information messages to and receive debugging commands from a connected terminal. Therefore – especially when getting familiar with the GPSRM 1 and when debugging GPSRM 1-related issues, utilizing the USB Debug Adapter is very useful.

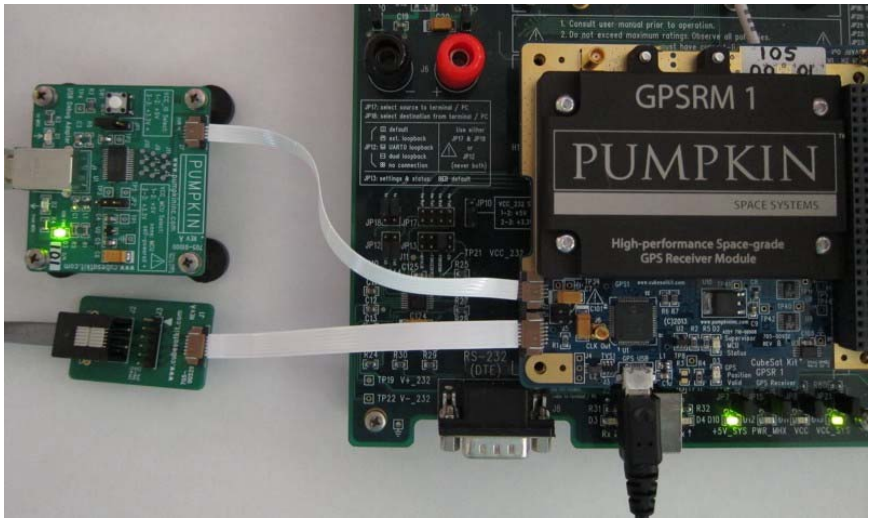


Figure 5: GPSRM 1 on a Development Board, connected to USB Debug Adapter, JFPC-PIC24 and micro-USB cable

To connect the USB Debug Adapter to the GPSRM 1, plug the 4-conductor FPC into both the USB Debug Adapter and **J2** on the GPSRM 1, and then connect the USB Debug Adapter to a host PC. Open a terminal to the USB Debug Adapters's Virtual COM port and configure it for the 115200,N,8,1.

Note The Pumpkin USB Debug Adapter utilizes an FTDI FT232R USB-to-serial converter chip; this chip is supported in nearly every operating system, including Windows, Mac/OS X and Linux. The Pumpkin USB Debug Adapter utilizes FTDI's default PID and VID.

Terminal Interface

The SupMCU on the GPSRM 1 provides a terminal interface via the USB Debug Adapter (above), for status information and a command-line interface (CLI). The terminal interface is used primarily for debugging and health & status verification of the GPSRM 1. The terminal operates at 115200,N,8,1.

With a USB Debug adapter attached and with a terminal window connected and properly configured, you will be presented with the GPSRM 1's splash screen:[6]

[6] Note that in the firmware release shown, the GPSRM 1's SupMCU is running at 29.54MHz.

```
Pumpkin(TM) GPSRM 1 Rev C v0.12.0a, built on Mar 17 2017 at 22:57:42.
Pumpkin(TM) SupMCU Core v0.12.0a, built on Mar 17 2017 at 22:57:42.
init: PIC24EP256MC206 SupMCU w/FRC+PLL oscillator at 29.54MHz.
init: Listening for SCPI commands on I2C address 0x51 (400kHz max SCL) ...
supmcu_flash_init: flash block occupies from 0x200 to 0x9FF
0)00:00:00:00.03 task_supmcu_wdt:       Starting...
0)00:00:00:00.04 task_supmcu_reset:     Starting...
0)00:00:00:00.04 task_supmcu_reset:     Stopped.
0)00:00:00:00.05 task_supmcu_i2c_reset: Starting...
0)00:00:00:00.05 task_supmcu_i2c_reset: Stopped.
0)00:00:00:00.06 task_supmcu_activity:  Starting...
0)00:00:00:00.06 task_supmcu_scpi:      Starting...
0)00:00:00:00.07 task_supmcu_heartbeat: Starting...
0)00:00:00:00.07 task_vinti7:           Starting...
0)00:00:00:00.08 task_vinti7:           Begin Kepler computation.
0)00:00:00:00.10 task_vinti7:           End Kepler, begin Vinti computation.
0)00:00:00:00.31 task_vinti7:           End Vinti computation.
0)00:00:00:00.32 task_vinti7:           t1: initial time (s)
0)00:00:00:00.32 task_vinti7:           t2: time of computed solution (s)
0)00:00:00:00.33 task_vinti7:           r1: input position vector (km)
0)00:00:00:00.34 task_vinti7:           v1: input velocity vector (km/s)
0)00:00:00:00.34 task_vinti7:           r2: output (computed) position vector (km)
0)00:00:00:00.35 task_vinti7:           v2: output (computed) velocity vector (km/s)
0)00:00:00:00.36 task_vinti7:           Given input pos. & vel. vectors at:
                                        t1    =                 0.00 s
0)00:00:00:00.45 task_vinti7:           r1.x =    -18982.9116920829 km
                                        r1.y =    -25047.1371788540 km
                                        r1.z =      -173.0441524398 km
0)00:00:00:00.53 task_vinti7:           xdot =         2.9699000848 km/s
                                        ydot =         0.3299752138 km/s
                                        zdot =         0.2657945052 km/s
0)00:00:00:00.55 task_vinti7:           Vinti7 output pos. & vel. vectors at:
                                        t2    =             86400.00 s
0)00:00:00:00.64 task_vinti7:           r2.x =    -11697.6351992203 km
                                        r2.y =     11186.0199224282 km
                                        r2.z =     -1879.8048359326 km
0)00:00:00:00.72 task_vinti7:           xdot =        -5.6684920156 km/s
                                        ydot =        -0.4031361715 km/s
                                        zdot =        -0.5187717096 km/s
0)00:00:00:00.74 task_vinti7:           Kepler output pos. & vel. vectors at:
                                        t2    =             86400.00 s
0)00:00:00:00.83 task_vinti7:           r2.x =    -11577.1708975164 km
                                        r2.y =     11291.6761629094 km
                                        r2.z =     -1872.8074581634 km
0)00:00:00:00.91 task_vinti7:           xdot =        -5.6747409551 km/s
                                        ydot =        -0.3495039806 km/s
                                        zdot =        -0.5265593348 km/s
0)00:00:00:00.92 task_vinti7:           Stopped.
0)00:00:00:00.93 task_supmcu_led:       Starting...
0)00:00:00:00.93 task_supmcu_led_flash: Starting...
0)00:00:00:00.94 task_nmea:             Starting...
0)00:00:00:00.94 task_supmcu_cli:       Starting...
0)00:00:00:00.95 task_supmcu_cli:       CLI ready for user command(s).

~ >
```

Figure 6: GPSRM 1 debug terminal splash screen

From Figure 6 we see the firmware versions of the GPSRM 1 application and the SupMCU core, the speed at which the SupMCU is operating, the GPSRM 1's I2C address, and other relevant boot-time information.

SCPI commands via Command-Line Interface (CLI)

All SupMCUs include a command-line interface (CLI) that is accessible only via the debug terminal. *The CLI is currently under development.*

Tip The intent of the CLI is to enable the user a means of issuing SCPI commands locally, without having to communicate with the GPSRM 1 over I2C. Other features may also be added at a later date.

To issue SCPI commands to a GPSRM 1 via the debug terminal, at the prompt type the following three terminal commands (in blue):

```
~> cd SCPI
SCPI > <Enter>
SCPI|0x51 > ?
```

The debug terminal is now ready to accept SCPI commands (e.g., **SUP:RES NOW** or **SUP:LED FLASH**). Type **ls** for a listing of SCPI commands. The I2C address of the SupMCU is displayed in the CLI's prompt.

```
0)00:00:00:00.95 task_supmcu_cli:       CLI ready for user command(s).

~ > cd SCPI
SCPI >
SCPI|0x51 > ?
  Built-in Commands:

  h/help/?        help
  cd ~            Returns to the top or root command directory
  cd ..           Moves the from the current directory up one directory
  cd [dir name]   Enter directory
  ls              List current directory/commands

  Directories are listed in [brackets]; commands are not.
  To execute a command, type the command, then press <Enter>.
  Current directory is shown in prompt.

SCPI|0x51 > ls
SCPI commands:
  SUPervisor:CLOCk
  SUPervisor:COM
  SUPervisor:DEBug
  SUPervisor:I2C:RESet
  SUPervisor:LED
  SUPervisor:RESet
  SUPervisor:SELFtest
  SUPervisor:TELemetry?
  GPS:PASSthrough
  GPS:POWer
  GPS:PROPagate
  GPS:LED
  GPS:LOG
  GPS:LOGPOSition
  GPS:RESet
  GPS:TELemetry?

SCPI|0x51 >
```

Figure 7: GPSRM 1 CLI SCPI command session

pySCPI GUI

Pumpkin's pySCPI is a python-based standalone application that can issue SCPI telecommands to and acquire telemetry over SCPI from connected Pumpkin SupMCUs.

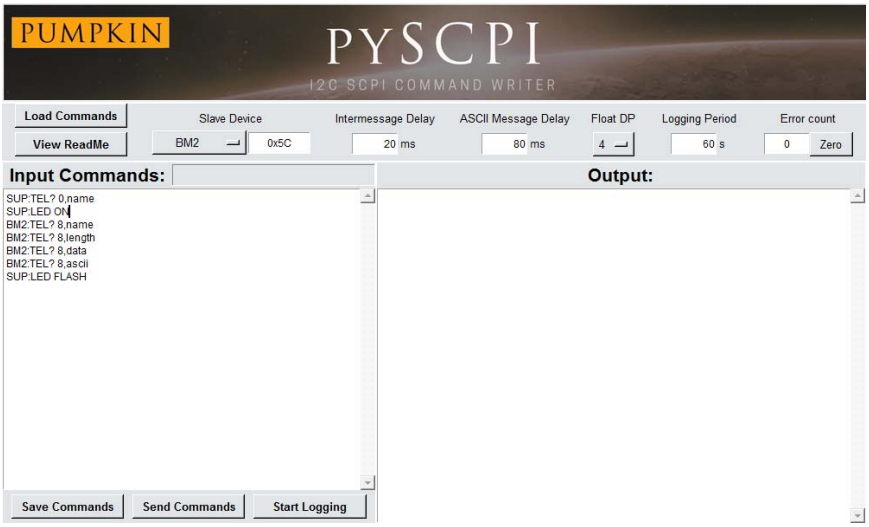


Figure 8: pySCPI at startup

pySCPI works with the Total Phase® Aardvark™ I2C/SPI host adapter; in a typical configuration, the GPSRM 1 is in a CubeSat Kit™-compatible module stack, and the Aardvark is connected to the CubeSat Kit's **SCL_SYS** and **SDA_SYS** I2C bus signals.[7]

pySCPI can issue SCPI commands to the selected SupMCU / I2C address. Commands can be stored to or retrieved from files. The user can define message delays for binary and ASCII telemetry messages. pySCPI can log messages and will report on errors. pySCPI parses binary telemetry data in a human-readable form.

Tip Using pySCPI is a great way to become familiar with SupMCU SCPI telecommands and telemetry requests.

Command Example

Say you want to command the GPSRM 1 to:

- Force its status LED ON
- Enable the CLKOUT function, with a divide-by-512 divider

[7] Pumpkin offers the CubeSat Kit™ Breakout Board Module, with a dedicated 10-pin connector that is compatible with the Total Phase® Aardvark™.

- Power on the GPS receiver
- Enable the GPS receiver's COM1 output to pass out of the GPSRM 1 and onto the CSK bus connector
- Command the GPS receiver to emit an ASCII NMEA message every second with GPS fix data and undulation

Issue the following five SCPI commands to the GPSRM 1's SupMCU:

- **`SUP:LED ON`**
- **`SUP:CLOCK ON,9`**
- **`GPS:POW ON`**
- **`GPS:PASS ON`**
- **`GPS:LOG 1,GGA`**

Tip Each SupMCU-based module has commands and telemetry associated with common SupMCU features, as well as application-specific features (in this case, having to do with the GPS receiver). Hence the GPSRM 1 has SCPI commands with SUP and GPS prefixes.

In pySCPI, select the GPSRM as the (I2C) slave device, type the commands in the Input Commands pane, and press Send Commands. The Output pane will indicate success or failure when sending the commands to the GPSRM 1.

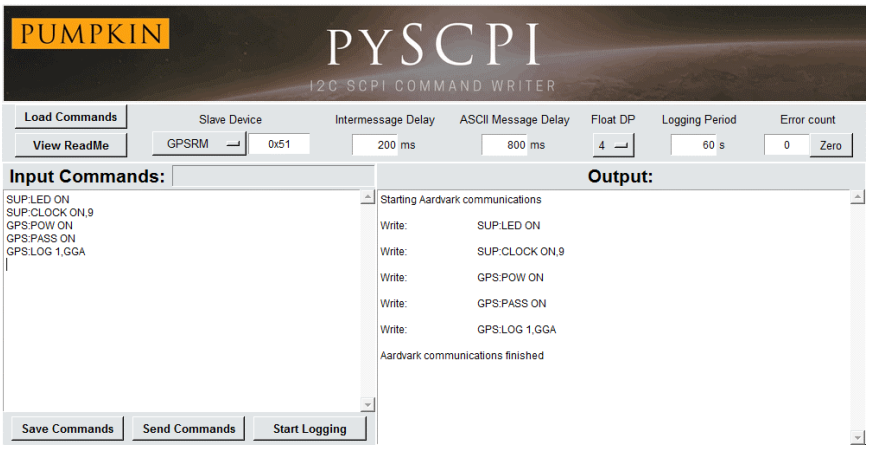


Figure 9: pySCPI output for commands sent to GPSRM 1

The results of these five commands are shown Figure 10, below. Of note are the five corresponding "SCPI command successfully received." messages, as well as some additional information. Not visible in this example is that the GPSRM 1's CLKOUT signal is at 57,522Hz, or approximately the GPSRM 1's internal clock (29.54MHz) divided by 512.

```
0)00:00:00:00.95 task_supmcu_cli:       CLI ready for user command(s).

~ > 0)00:00:00:03.29 task_supmcu_scpi: Accepted "SUP:LED ON" command.
0)00:00:00:03.34 task_supmcu_scpi: Accepted "SUP:CLOCK ON,9" command.
0)00:00:00:03.39 task_supmcu_scpi: Accepted "GPS:POW ON" command.
0)00:00:00:03.45 gpsrm_passthrough_open:        Done.
0)00:00:00:03.45 gpsrm_passthrough_on:  Done.
0)00:00:00:03.46 task_supmcu_scpi: Accepted "GPS:PASS ON" command.
0)00:00:00:03.53 task_supmcu_scpi: Accepted "GPS:LOG 1,GGA" command.
0)00:00:00:03.53 task_nmea:             Starting...
```

Figure 10: GPSRM 1 terminal windows with responses to pySCPI commands from Figure 9

Telemetry Example

You want to know:

- What are the telemetry items available from the SupMCU features of the GPSRM 1?

Guessing there are ten SupMCU-related items, issue the following ten SCPI commands to the GPSRM 1's SupMCU:

- **`SUP:TEL? 0,name`**
- **…**
- **`SUP:TEL? 9,name`**

The results are shown in Figure 9. From it we see that there is a telemetry item 5 that returns the GPSRM 1's elapsed time, in seconds, from when it was powered on. We also see that in this GPSRM 1 firmware release, there are nine SupMCU-specific telemetry items.

Tip All GPSRM 1 telemetry request responses include a timestamp. The timestamp can be used for data logging, etc.

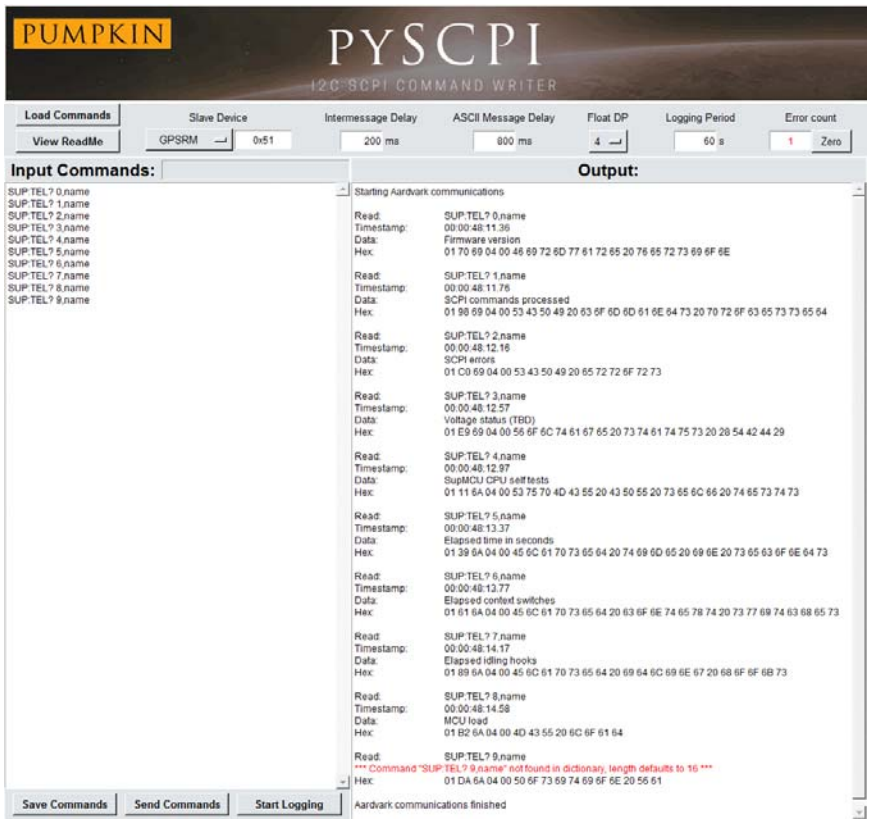


Figure 11: pySCPI output from requesting names of first ten SupMCU telemetries from GPSRM 1

Tip GPS-specific telemetry is requested via GPS:TEL? n,aaaa SCPI commands.

Diving further into the GPSRM 1, you may want to better understand the GPSRM 1's telemetry with ID=5. From within pySCPI, issue the four related commands that cover the telemetry with ID=5:

- **`SUP:TEL? 5,name`**
- **`SUP:TEL? 5,length`**
- **`SUP:TEL? 5,data`**
- **`SUP:TEL? 5,ascii`**

The result of these four SCPI commands is shown in Figure 10. Several interesting features of the telemetry are apparent:

- The **`ascii`** telemetry flavor is presented over I2C entirely in human-readable form, with the ready bit and timestamp preceding the data

- The other three telemetry flavors (**`name`**, **`length`** and **`data`**) are presented in binary form over I2C, and displayed in hex format within pySCPI's Output pane
- The first five bytes of binary telemetry include a ready bit (**`0x01`**) and a 4-byte timestamp (**`0x0004892C`**, **`0x00048954`**, **`0x0004897C`** in this example)
- You can see the progress of time in the increasing values of the timestamps associated with the telemetry requests

Figure 12: pySCPI output for all fields for GPSRM 1's SupMCU telemetry item 5

Additionally, for the SupMCU telemetry with ID=5, we see:

- The telemetry contains just a single item, a count
- The telemetry is 21 bytes long in total;[8] pySCPI reads the correct number of bytes for each telemetry command:

`0x01`:	1-byte status including ready bit
`0x0004897C`:	4-byte (**`uint32_t`**) timestamp
`0x0000000000000B9E`:	8-byte (**`uint64_t`**) counter value
(not shown):	8-byte checksum (not yet implemented)
total length of telemetry:	*21 bytes*

[8] As reported by the **`SUP:TEL? 5,length`** command. The data is presented in little-endian format; i.e. the counter in this example is 0x0B9E, or 2974.

Configuring the GPS Receiver for Space Use

In order to function properly in a LEO environment, the GPS receiver must be (re-)configured with adjustments that are implemented via the commands shown in Table 2.[9]

OEM6 Command	Argument(s)	Notes
SETTROPOMODEL	**none**	*Removes the tropospheric model, as LEO satellites are operating above this atmospheric layer.*
ECUTOFF **SBASECUTOFF** **GLOECUTOFF** **GALECUTOFF** **QZSSECUTOFF**	**-30**	*Allows tracking of GNSS Satellites to 30 degrees below the horizon, as LEO satellites are 400-800km above the Earth and will see satellites below the horizontal.* *Note that only the ECUTOFF commands for systems of interest need to be input.*
DOPPLERWINDOW	**all user 37000**	*Allows acquisition and reacquisition of GNSS satellites at LEO velocities. 37000 is appropriate for LEO orbits.* *Note that the Doppler window is a +/- in Hz; i.e., 37000 as input results in a search window from -37000Hz to +37000Hz.*
POSTIMEOUT	**10**	*Allows time to compute PVT at LEO velocities.*
APPROXPOSTIMEOUT	**5**	*Allows time to compute initial approximate position as part of the PVT algorithm at LEO velocities.*

Table 2: Summary of OEM6 commands for GPS receiver, for use in LEO orbits

Note It is up to you to verify that the GPS receiver has received these commands without error.

Note Once these settings have been applied to the GPS receiver, it may no longer achieve lock within the time period specified for terrestrial lock. This is a side-effect of these commands.

Hidden / Undocumented OEM6 Commands

The **DOPPLERWINDOW** and **APPROXPOSTIMEOUT** commands are not documented in the OEM6 family documentation. Users who wish to command the GPS receiver in binary mode should refer to the command references for these two commands at the end of this

[9] As of OEM719 firmware release 6.400. See the OEM6™ Family Firmware Reference Manual, available at http:www.novatel.com.

document, and to Table 3, below. If commanding the GPS receiver via ASCII commands, refer to Table 2, above.

OEM6 Command	AssignSystemEnum	DopplerWindowModeEnum
DOPPLERWINDOW	3 = ALL 99 = GPS 100 = SBAS	0 = AUTO 1 = USER

Table 3: DOPPLERWINDOW binary command enums

Note Each of the above-listed enums is 4 bytes long.

GPS Receiver Firmware

Identifying / Verifying COCOM Unblocking

In order to function in a space environment at speeds in excess of 515m/s, the GPS receiver must be running special high-speed software. This software "unblocks" the COCOM[10] limits under which many GPS receivers – including the OEM719 unit inside the GPSRM 1 – operate by default. These velocity limits are a byproduct of United States arms control regulations, and affect many manufacturers of GPS receivers. The limits can be removed as part of an application to the United States Directorate of Defense Trade Controls (DDTC)[11] or United States Commerce Department; fees apply.

Note All GPSRM 1s are provided with COCOM-unblocked OEM719 receivers from the factory.

NovAtel appends the letter 'H' (for "high-speed") to both the model and the software version of OEM719-series GPS receivers that are configured to work in space. You can verify that your GPS receiver is configured for space use by utilizing NovAtel Connect software to read the Receiver Version, as shown in Figure 13, below.

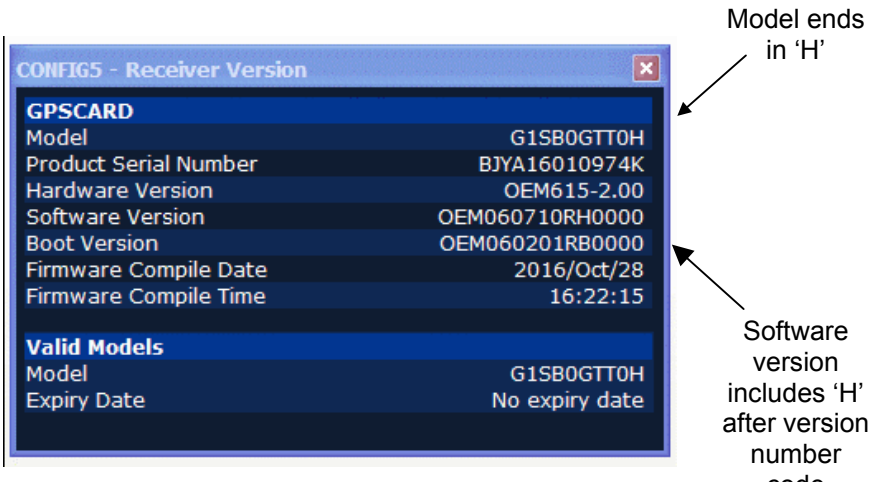


Figure 13: Sample Receiver Version window in NovAtel Connect software; COCOM-unblocked OEM719

[10] Coordinating Committee for Multilateral Export Controls

[11] Pumpkin is registered with the DDTC. Applications to purchase GPSRM 1s from non-US persons include an export license application, managed by Pumpkin on behalf of the (non-US) end-user.

Tip The OEM719 receiver in Figure 11 above is running firmware version 6.710H. The 'R' before the 'H' (for high-speed) indicates that this is a fully tested, release version of the software.

Alternate custom firmware specially constructed by NovAtel for particular end-users may have a different suffix after the version number, e.g., "Snn" for the nth special software version created for a class of end-users.

OEM719 Model Encoding

The OEM719 GPS receiver model for the base configuration of the GPSRM 1 is G1SB0GTT0H; the alphanumeric code that explains the GPS receiver's configuration is:

- G = GPS only
- 1 = L1 only
- S = SBAS enabled
- B = generates corrections
- G = GLIDE
- TT = 20Hz position and measurements
- H = High speed (COCOM unblocked)

As an option, OEM719s with additional features (e.g., L1+L2, GPS + GLONASS, etc.) are available for use in the GPSRM 1. Please contact Pumpkin for fir further information.

Tip More information on the OEM6 family configurations, options and nomenclature can be found in NovAtel's documentation.

Firmware Upgrades

There are two distinct firmwares in a GPSRM 1; the SupMCU's firmware (from Pumpkin), and the GPS receiver's firmware (from NovAtel).

Tip In most end-user applications, there will never be a need to update the GPS receiver firmware.

GPSRM 1 Firmware

Pumpkin is continuously developing the GPSRM 1's firmware. Users can update their GPSRM 1 firmware in the field by using the supplied Microchip PICkit™ or alternative Microchip programmer, and the supplied Pumpkin JFPC-PIC24 programming adapter.

Pumpkin provides **`.hex`** firmware upgrade files for the GPSRM 1.

Tip Pumpkin SupMCU release firmware names have '**`r`**' suffixes; interim releases have '**`i`**' suffixes.

To reprogram the GPSRM 1, follow the steps outlined below:

1. Install Microchip's MPLAB X® Integrated Programming Environment (IPE). This is a standalone GUI application that provides a simple means of (re-)programming the PIC24EP256MC206 16-bit MCU that serves as the GPSRM 1's SupMCU.[12]
2. Connect the laptop running MPLAB X IPE to the PICkit™, to the JFPC-PIC24, to the GPSRM 1. See Figure 5, above.
3. Launch the IPE. Under Tool, select the programming tool you have available (typically a PICkit™) and click on Connect. Once successfully connected to a powered GPSRM 1, the IPE's Output window will emit a "Target device PIC24EP256MC206 found." message. See Figure 14 for a successful connection when using a RealICE™ in-circuit emulator.

[12] The MPLAB X IPE is typically installed along with the MPLAB X IDE.

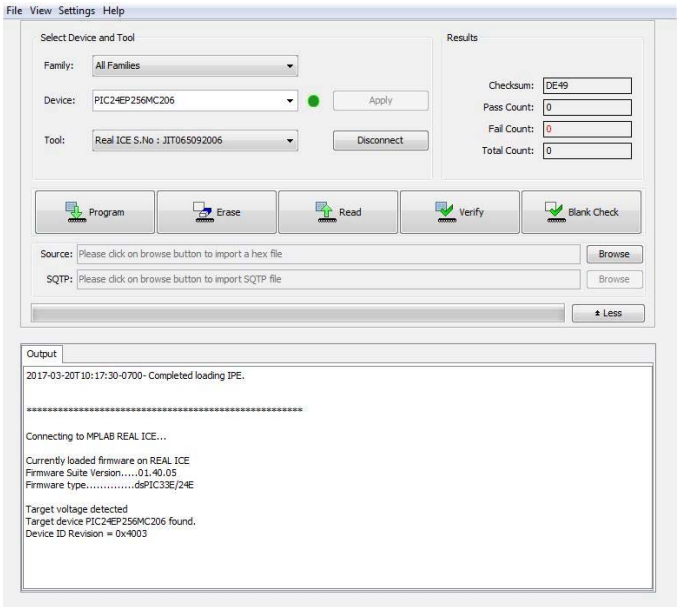


Figure 14: MPLAB IPE successfully connected to GPSRM 1

4. Browse to the GPSRM 1 firmware **.hex** file and select it.
5. Click on the Program button and the IPE will (re-)program the GPSRM 1's SupMCU with the selected firmware. A successful reprogramming with an interim release is shown in Figure 15.

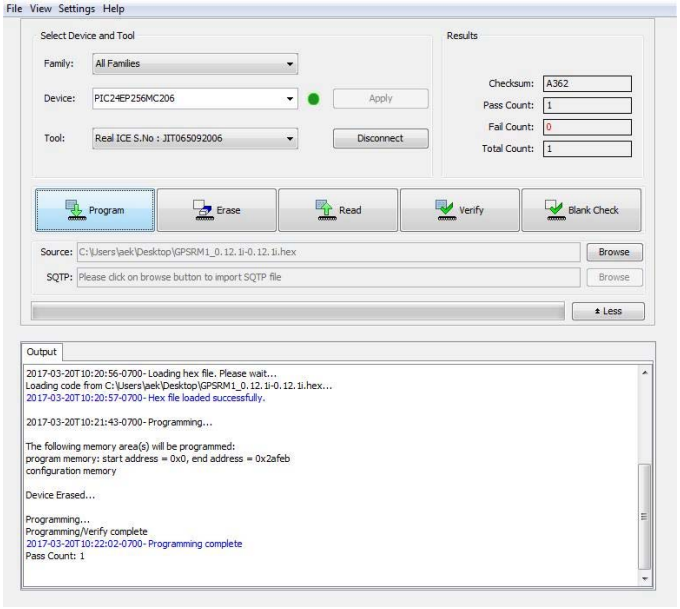


Figure 15: MPLAB IPE after successfully reprogramming GPSRM 1

6. Disconnect the JFPC-PIC24 from the GPSRM 1.

Tip A standalone (or unpowered) GPSRM 1 can be powered by a Pumpkin USB Debug Adapter; install jumper **JP2** to connect pins 2&3. This provides +3.3V to the SupMCU on the GPSRM 1, when the USB Debug Adapter is connected via the GPSRM 1's **J2**.

A software development kit (SDK) is available for the GPSRM 1 at additional cost. Users who wish to customize the GPSRM 1 firmware may want to acquire this SDK.

OEM719 Firmware

The firmware inside the OEM719 GPS receiver inside the GPSRM 1 can be upgraded without disassembly. The OEM719's Product Serial Number (PSN, see Figure 13, above) must be provided to NovAtel; NovAtel then provides a firmware upgrade file and model authorization code. Both H-model firmware and an H-model authorization code are required to maintain the COCOM-unblocked functionality that was delivered with your GPSRM 1.

To update the OEM719 firmware, follow these steps:

1. Acquire new H-model OEM719 firmware (a **.hex** file) and authorization code specific to your OEM719's PSN. Install and launch the NovAtel WinLoad.exe software on your laptop. Load the **.hex** file into WinLoad.exe.
2. Connect the GPSRM 1 to your laptop in a manner that connects the GSPRM 1's passthrough signals to a COM port on your laptop. The Pumpkin CubeSat Kit Test Board G can be used, as it presents one of a selected three pairs of Tx/Rx lines (**IO.[5,4]**, **IO.[17,16]** or **IO.[33,32]**) to an RS-232 driver with a DB-9 male connector; with a null-modem cable to a laptop serial port or serial-to-USB converter, this can successfully connect to WinLoad.exe.
3. Connect the GPSRM 1 to a Pumpkin USB Debug Adapter with its **JP2**. Power up the OEM719; the simplest way to do this is to power the GPSRM 1 via its USB connector.
4. Immediately press Query Card in WinLoad.exe, and *within 30s* via the GPSRM 1's debug terminal, issue the **GPS:POW ON** and **GPS:PASS ON** commands. This will turn the OEM719 GPS receiver on and make it visible to WinLoad.exe, while the latter is querying the OEM719 for its bootloader.

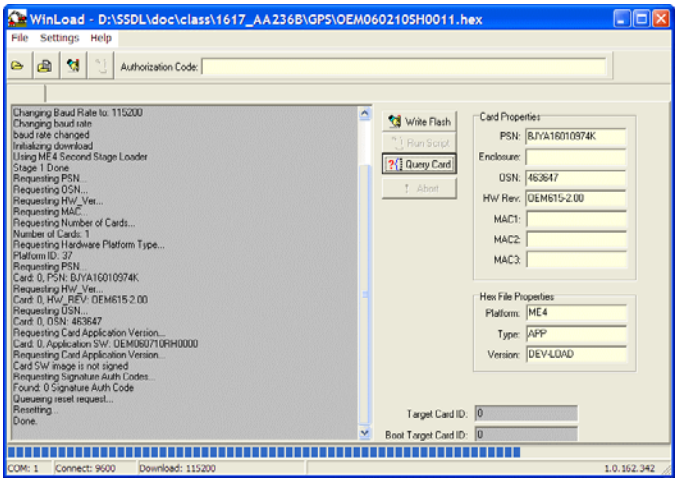


Figure 16: NovAtel WinLoad.exe splash screen after successfully querying the OEM719

5. In WinLoad.exe, choose Write Flash. You will be prompted for the authorization code – enter it in the dialog window and press OK. WinLoad.exe will now reflash the OEM719 inside the GPSRM 1.

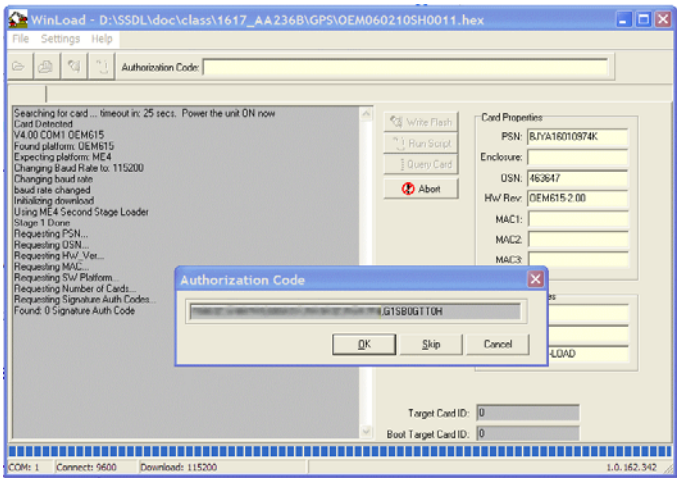


Figure 17: Dialog window for OEM719 authorization code

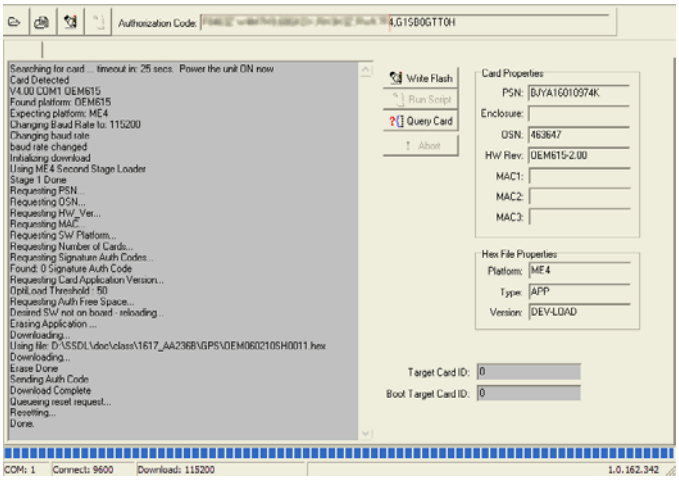
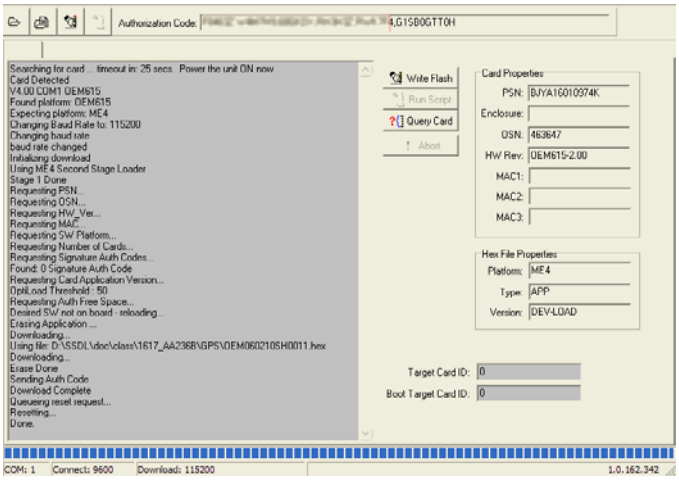


Figure 18: WinLoad.exe after successfully reflashing OEM719 with custom software

6. Verify that the new firmware is in place, via WinLoad.exe or NovAtel Connect's Receiver Version window.



Tip Pumpkin's CubeSat Kit™ Test Board G provides an RS-232 port with DB-9 connector for use with the GPSRM 1; it can connect to any of the three serial port pairs (**`IO.[5,4]`**, **`IO.[17,16]`** or **`IO.[33,32]`**) that the GPSRM 1 uses to connect to the CubeSat Kit Bus Connector.

Assembly Revisions

A variety of assembly revisions are available for the GPSRM 1. Please refer to the GPSRM 1 datasheet for more information.

Note Assembly revisions are applied to the GPSRM 1 at the factory. Competent end-users may be able to change some of the assembly revisions in the field; this typically involves desoldering one or more zero-Ohm jumpers, and (re-)soldering them in alternate locations.

Troubleshooting

Difficulties Turning On

The OEM719's green status LED must be blinking at roughly 2Hz for proper GPSRM 1 operation. A status LED that remains red or red+yellow five seconds after the application of power to the GPSRM 1 indicates that the OEM719 is having difficulty turning on.

Most problems associated with the GPS receiver powering on are ultimately traced to a CubeSat power system with insufficient 3Vdc power.

To debug a power-related problem, we suggest:

1. Try running the GPSRM 1 in standalone mode (see Using NovAtel® Connect software with the GPSRM 1). In this configuration, the GPSRM 1 gets all its power from the USB host, and should achieve GPS lock (with a connected antenna) within 50s (no almanac or ephemeris and no approximate position or time). Success in this configuration indicates that the GPSRM 1 is functioning properly.
2. Next, integrate the GPSRM 1 into a CubeSat module stack, with a CubeSat-class EPS that provides +5Vdc and +3.3Vdc on **+5V_SYS** and **VCC_SYS**, respectively. With an active USB host connected via **J3**, the OEM719's status LED should be green and blinking. Issue the SCPI commands **GPS:PASS ON** and **GPS:LOG 1,GGA**, and observe the NMEA ASCII string being sent out on the configured serial output pin of the GPSRM 1 (**IO.5**, **IO.17** or **IO.33** on the CubeSat Kit Bus Connectors). This establishes that the OEM719 is alive (and powered via USB), that its is accepting commands from the SupMCU, and that the passthrough function is working to route the OEM719's COM1 port to the CubeSat Kit Bus Connectors.
3. Repeat step 2 above, but issue the **GPS:POW ON** command first. If this configuration fails, then the power system you are using to supply **+5V_SYS** and **VCC_SYS** to the CubeSat Kit Bus Connectors is inadequate and should be examined.

Tip GPSRM 1 test points **TP41** (OEM719 **LNA_PWR**) and **TP42** (OEM719 **3V3**) can be used to ascertain if the OEM719 is receiving adequate power. Typical values are 4.7Vdc at **TP41** and 3.3Vdc at **TP42**. Voltages will be slightly lower when operating from USB power.

Tip When debugging what appear to be power-related problems with the GPSRM 1, it's helpful to listen for the sound/alert that a laptop will make when the OEM719 is connected and disconnected via USB. These audible connect/disconnect alerts will occur whenever the OEM719 inside the GPSRM 1 is power-cycled, either intentionally or inadvertently. Listening for these sounds can alert you to power supply problems that might not be obvious.

Firmware Releases

0.12.x

Many internal updates and command-processing improvements. New commands (e.g., **GPS:LED**) implemented. CLI improvements and fixes.

0.11.x

Updates including changes to telemetry and internal speeds.

0.2.0 – 0.1.0

Firmware v0.2.0 fixes I2C issues and permits operation at 100kHz and 400kHz I2C clock speeds.

Note I2C masters that drive the GPSRM 1 can expect to see (SCL) clock stretching when commanding the GPSRM 1.

Firmware v0.1.1 is the first release that conforms to the command syntax of the Standard Commands for Programmable Instruments (SCPI) standard. This release also enhances the Status LED's capabilities.

Firmware v0.1.0 is the initial release. It permits basic manipulation of critical features of the Supervisor MCU. The GPSRM 1 receives commands but does not provide any readback functionality (i.e., there are commands, but no telemetry).

Firmware Features / Specifications

Item	Value / Feature	Notes
Initial state on power-up	• OEM719 OFF • Supervisor MCU CLK Out off • COM1 passthrough disabled • SupMCU status LED flashes	No commands are sent to OEM719 at startup.
Operating speed	29.54MHz	To support orbit propagator
I2C address	0x51	Defined in firmware
I2C bus speeds supported	400kHz	
J1 (program/debug)	Supervisor MCU in-circuit debugging port	Connect to ICD or PICkit® via Pumpkin JFPC-PIC24
J2 (terminal window)	115200,N,8,1	Connect to terminal via Pumpkin USB Debug Adapter
J3 (USB)	Active whenever OEM719 is running	
J4	Do not use	
J5	Do not use	Do not place jumpers here.
J6 (**CLK Out**)	Supervisor MCU's REFCLK	MMCX jack. See command set.
J7 (**PPS Out**)	Valid whenever GPS has lock	MCX jack. Requires 50Ω termination, o/wise rings considerably.
J8 (**VARF Out**)	Controlled via OEM719 **FREQUENCYOUT** command	MMCX jack.
D3 (**GPS Position Valid LED**)	Active when GPS has lock.	May also be lit when GPS is starting / has not yet locked (!).
D2 (**SupMCU Status LED**)	Command-dependent.	See command set.

Table 4: GPSRM 1 firmware features & specifications

Command Set – 0.12.x

The SCPI command set for is listed below. All earlier versions are deprecated – customers should upgrade their GPSRM 1s to the latest / last versions.

Command	Action
SUPervisor :CLOCk {OFF| ON,<divider>}	Force Supervisor MCU's **REFCLK** to be inactive (**OFF**) or active (**ON**, **divider**) on CLK Out (**J6**) output. The **REFCLK** output is the SupMCU's internal clock divided by (2^**divider)**. Allowable values for divider are 0 to 15. The frequency of the resulting CLK Out is a function of the GPSRM 1's base clock frequency (see splash screen).
SUPervisor :LED {OFF|ON|FLASh}	Specify the behavior of the SupMCU's Status LED: **OFF**: LED is always off **ON**: LED is always on **FLASh**: LED flashes (SupMCU heartbeat)
SUPervisor :RESet NOW	Force SupMCU restart.
SUPervisor :SELF {START|STOP| REStart}	Run SupMCU self-tests.
SUPervisor :TEL? n,{ASCII|DATA| LENGTH|NAME}	Request SupMCU telemetry.
GPS :PASSthrough {OFF|ON}	Force OEM719's COM1 to be disconnected from the CSK Bus (**OFF**) or to pass through on-board isolators to the CSK Bus (**ON**).
GPS :POWer {OFF|ON}	Disable or enable power to the OEM719. Has no effect when GPSRM 1 is connected to a live USB host.
GPS :LED {SUP|GPS|PASS}	Specify the behavior of the SupMCU's Status LED: **SUP**: follows SUP:LED command **GPS**: LED is on when GPS receiver is on **PASS**: LED is on when passthrough is on
GPS :LOG {OFF|GGA|GSA |GSV|RMC|VTG}	Configure the OEM719 to log the requested GPxxx message to its COM1 port once per second. The Supervisor MCU issues the following command to the OEM719's COM2 port: **LOG COM1 GPxxx ONTIME 1** or **UNLOGALL COM1 TRUE**
GPS :RESet {OFF|ON}	Force GPS receiver's **–RESET** input high (**OFF**) or low (**ON**).
GPS :TEL? n,{ASCII|DATA| LENGTH|NAME}	Request GPS telemetry (*not* GPS receiver telemetry).

Table 5: GPSRM 1 firmware v0.12.x SCPI commands

APPROXPOSTIMEOUT

Set the approximate position timeout of the receiver

Message Details

Message Type: COMMAND
MessageID: 1513
Direction Type: INOUT
Hidden: Y
Access string: MSG_ACCESS_ALL

Message Parameters

#	Name	Description	Format	Bytes	Offset
1	ulMyTimeoutSec	Timeout in seconds	ULONG	4	0

DOPPLERWINDOW

Change the behavior of the doppler search for automatic channel assignments in the receiver.

Message Details

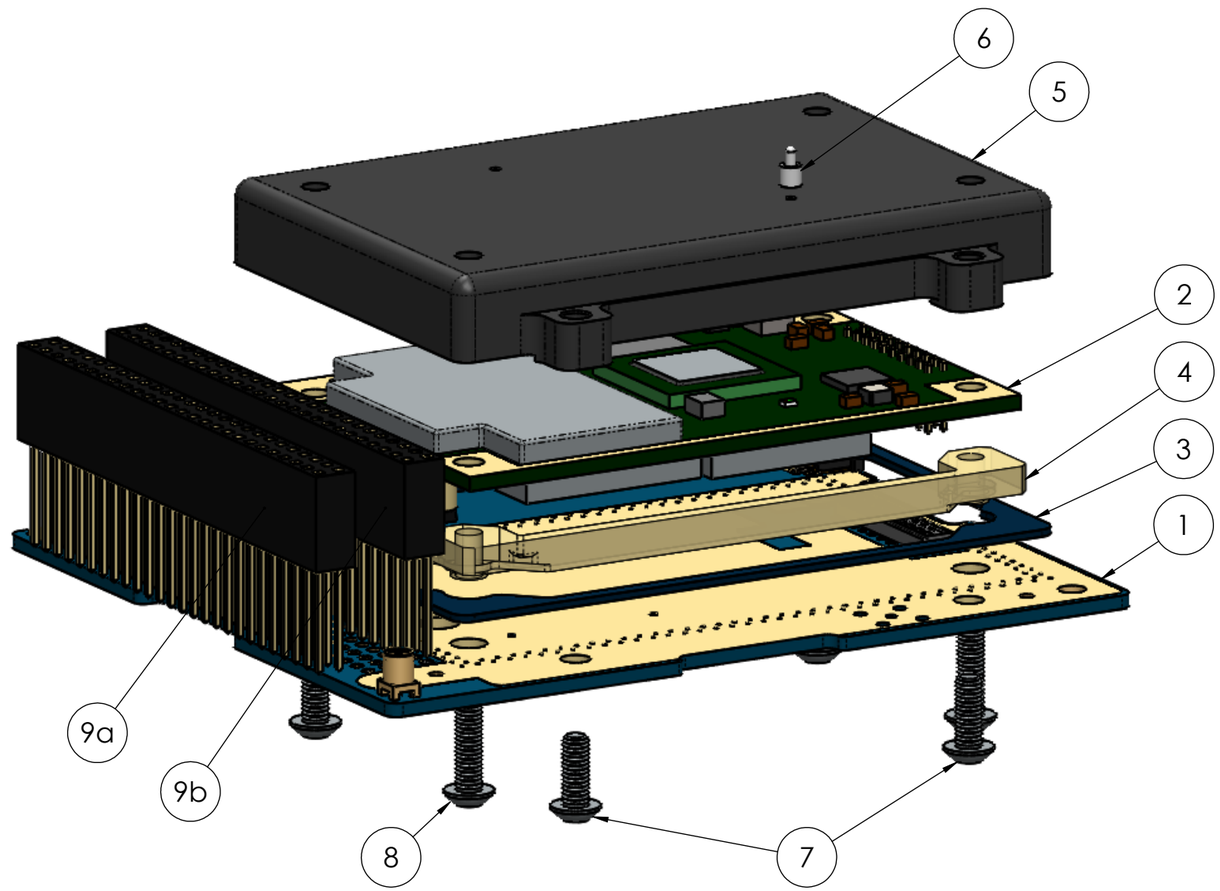
Message Type: COMMAND
MessageID: 1512
Direction Type: INOUT
Hidden: Y
Access string: MSG_ACCESS_ALL

Message Parameters

#	Name	Description	Format	Bytes	Offset
1	eMySystem	Satellite system to change settings for.	AssignSystemEnum	4	0
2	eMySetting	Doppler window mode.	DopplerWindowModeEnum	4	4
3	ulMyDopplerWindow	Doppler window to search in HZ. This is a +/- value.	ULONG	4	8
4	lMyCenterFrequency	Reserved for future use: center frequency of search in Hz. (Default = 0)	LONG	4	12

REVISION HISTORY

REV	LOC	DESCRIPTION	DATE	APP'VD



ITEM NO.	DESCRIPTION	PART NUMBER	MANUFACTURER	QTY
1	GPSRM 1 PCB	705-00932	PUMPKIN	1
2	DUAL-FREQUENCY GNSS RECEIVER w/ MCX JACK CONNECTOR	OEM719	NOVATEL	1
3	2-PIECE GPSRM EMI GASKET	703-00896	PUMPKIN	1
4	GPSRM RAIL	703-01826	PUMPKIN	2
5	GPSRM COVER w/ HELICOILS	710-01039	PUMPKIN	1
6	LIGHT PIPE, GPSRM	703-01132	PUMPKIN	1
7	M3x8mm BUTTON HEAD SCREW	92095A181	MCMASTER CARR	2
8	M3x12mm BUTTON HEAD SCREW	92095A183	MCMASTER CARR	4
9a	.100" PC/104 STACKTHROUGH	ESQ-126-39-G-D	SAMTEC	2
9b	.100" PC/104 NON-STACKTHROUG (NOT SHOWN)	ESQ-126-37-G-D	SAMTEC	2

WEIGHT: 79g

PUMPKIN INCORPORATED

750 Naples - San Francisco - CA 94112
(415) 584-6360 ph - (415) 585-7948 fax
info@pumpkininc.com

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES ARE:
FRACTIONS DECIMALS ANGLES
.X = ± .030
.XX = ± .010 ± 0° 30"
.XXX = ± .005

CAD GENERATED DRAWING, DO NOT MANUALLY UPDATE

APPROVALS	DATE
DRAWN *JMM*	12/20/17
CHECKED	
QUAL ENG	

DO NOT SCALE DRAWING

TITLE
GPSRM 1 Exploded View & Parts List

SIZE B | DWG. *710-00908* | REV D1
SCALE 3 : 2 | SHEET 1 OF 1