ORIGINSMART™ API SPECIFICATION

Protocol Specification – software version 04_04_18_09_33

O r i g i n G P S . c o m

1. SCOPE

This document describes the features and specifications of OriginSmart™ Embedded firmware and the application level communication protocol.

2. DISCLAIMER

All trademarks are properties of their respective owners.

Performance characteristics listed in this document do not constitute a warranty or guarantee of product performance. OriginGPS assumes no liability or responsibility for any claims or damages arising out of the use of this document, or from the use of integrated circuits based on this document.

OriginGPS assumes no liability or responsibility for unintentional inaccuracies or omissions in this document.

OriginGPS reserves the right to make changes in its products, specifications and other information at any time without notice.

OriginGPS navigation products are not recommended to use in life saving or life sustaining applications.

3. CONTACT INFORMATION

Support - iot@origingps.com or Online Form

Marketing and sales - marketing@origingps.com

Web – www.origingps.com

4. RELATED DOCUMENTATION

№	DOCUMENT NAME
1	Spider and Hornet - NMEA Protocol Reference Manual
2	OriginIoT Datasheet
3	ORG2101 EVK User Guide

TABLE 1 – RELATED DOCUMENTATION

5. REVISION HISTORY

REVISION	DATE	CHANGE DESCRIPTION

V1.0	April 22, 2017	First release

TABLE 2 – REVISION HISTORY

GLOSSARY

MCU Micro-Controller Unit

6. ABOUT ORIGINIOT

The OriginIoT module ORG210x is an analytic customizable system that collects data from sensors. The data can be transferred to a remote server or cloud platform by the OriginIoT system via wireless cellular communication (GSM or LTE).

The multi-purpose OriginIoT system can accommodate peripheral devices such as sensors or other components via UART, SPI, I^{2}C or GPIO and combines cellular communications module per customers choice, with superior positional accuracy of stand-alone GNSS. Peripheral devices are configured over a web interface, eliminating additional embedded FW efforts. The ease and flexibility of utilizing OriginIoT as a basis for vast array of applications quickens time to market while minimizing the size of your IoT sensor device.

OriginIoT devices enable developers to develop IoT products without writing a single line of embedded code and without RF engineering. A new rapid product cycle is created, dramatically cutting development resources.

7. ABOUT ORIGINGPS

OriginGPS is a world leading designer, manufacturer and supplier of miniature positioning modules, antenna modules, antenna solutions and IoT devices.

OriginGPS develops fully-integrated, miniaturized GPS/GNSS, and integrated IoT solutions for developers. OriginGPS modules introduce unparalleled sensitivity and noise immunity by incorporating Noise Free Zone system (NFZ™) proprietary technology for faster position fix and navigation stability even under challenging satellite signal conditions.

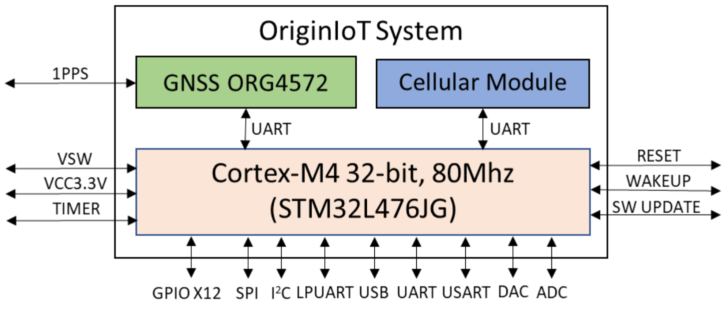
Founded in 2006, OriginGPS is specializing in development of unique technologies that miniaturize RF modules, thereby addressing the market need for smaller wireless solutions. For over a decade, our experts have been developing ultra-sensitive, reliable, high performance modules with the smallest footprint on the market, supporting a range of verticals, such as smart cities, drones, asset tracking, wearables, automotive, and IoT.

8. DESCRIPTION

8.1 SYSTEM VIEW

The OriginIoT system use ST-Micro STM32L476JG MCU as the main platform interfacing with the cellular module. The OriginIoT system provides serial interfaces to control the external devices.

The MCU supports the multiple interface types on its available physical pinout as described in **Figure 1: High Level Functional Block Diagram.**



The OriginSmart™ firmware resides in the MCU and at the power, Cold and Warm Restarts, the module will run with the default configuration and will connect to cloud application over cellular module and the Console Terminal over RS-232 port. When the module connected to the end-applications (Cloud Application or Console Terminal) then the module can be configured by the corresponding end-application.

These are the main functions of the OriginSmart™ firmware:

1. Communication link with the end-application
2. Routing the device data to the end-application
3. Retrieve the data from devices upon request from end-application based on interface ID.
4. Deliver the received device data from the end-application to the corresponding device based on interface ID.
5. Stream or log data from devices

Each interface has a unique ID number allowing the end-application to communicate and exchange messages with the corresponding interface (GNSS, Cellular, MCU and external devices). The embedded software configures the interfaces types and parameters and assigns unique ID for each interface for identification purpose as described in OriginIoT interfaces table:

Table 1: OriginIoT Interfaces

Interface Type	**I/O**	**Interface ID**	**Attached Device**	**Connector**	**PIN**
Internal MCU	IO	IF_00	For Internal MCU management	Internal	Internal
2wire UART	IO	IF_01	To interface to Console Terminal	J1	14,16
2,4wire UART	IO	IF_02	Serial interface to cellular module	Internal	Internal
4wire UART	IO	IF_03	Serial interface to GNSS module	Internal	Internal
2wire LPUART	IO	IF_04	Serial interface to external device*	J1	15,17
I2C Master	IO	IF_05	Serial interface to external devices	J1	9,11
GPIO Bitport	O	IF_06	Internal GPIO for cellular reset	Internal	Internal
GPIO Bitport	O	IF_07	Internal GPIO for GNSS reset	Internal	Internal
WAKEUP_1_PIN	I	IF_08	Interrupt interface*	J1	23
WAKEUP_2_PIN	I	IF_09	Interrupt interface*	J1	25
GPIO_IN_1	I	IF_0A	GPIO for external device	J2	6
GPIO_IN_2	I	IF_0B	GPIO for external device	J2	11

GPIO_OUT_1	O	IF_0C	GPIO for external device	J2	7
GPIO_OUT_2	O	IF_0D	GPIO for external device	J2	5
Timer Bitport	O	IF_0F	Timer bitport for external device*	J2	12
DAC Bitport	O	IF_12	Digital to analog convertor for external device*	J2	17
ADC Bitport	I	IF_13	Analog to digital convertor for external device*	J2	23
GPIO Reserved	I/O	IF_14…IF_1F	GPIO for external device*	J2	TBD
2wire UART	I/O	IF_20	Serial interface to external device	J1	18, 22
SPI	I/O	IF_21	Serial interface to external devices*	J1	1, 3, 5
USB 2.0 OTG	I/O	IF_22	Serial interface to external device*	J1	2, 4

* Not supported by OriginSmart™ FW to date, contact iot@origingps.com for details.

10.2 CELLULAR APPLICATION NETWORK

The OriginIoT module will receive private IP address from the mobile operator and will operate as Mobile Originated Application. It will advertise its private IP address and initiate the connection with the End-application.

The application IP address and server information will be stored in NVM. Please refer to OriginIoT EVK datasheet for information on how to change these. Make sure to ask your OriginGPS representative to set the requested network configurations for you in case you are not using an EVK.

A direct IP socket will be established between application and OriginIoT module as described below:

1) Module retrieves the Application IP address from its NVM
2) Module establishes a direct IP socket with the Application
3) Module sends its identification data (MODULE_REGISTER_NOTFY) to the Application. (New message from Module)
4) The Module should keep alive the direct IP sockets which will require one message (MODULE_APP_KEEPALIVE_NOTFY). (One New Message from Module to Application)
5) The Application can send the LoopBack message (MODULE_APP_SET_LB) and the Module echoed back (MODULE_APP_SET_LB_RSP). (New set message from application to the Module, New Response message from the Module to the Application to echoing)
6) Upon Module power cycling and link down, steps 1 to 5 should be repeated.

10.3 COMMUNICATION SPECIFICATION

The OriginSmart™ firmware can interface with the Cloud Application through TCP/IP messaging which is supported by the cellular module. The firmware encapsulates the Header message (Source and destination address of interfaces), the device header and the device raw data to the cellular AT command; and the cellular module will send out the message in TCP/IP format to the cloud application. On the received direction from the cloud application, the GSM module will transport AT based message to the software and based on destination device interface ID, the firmware will route the device data to the corresponding device.

11. Message Format

This section defines and details the OriginSmart™ message format to communicate with devices (Cellular, GNSS, external devices, MCU) and the End-Application.

The Message format includes the Header and the Device Header and device Raw Data segments. The header section includes the destination and source interfaces, message length count, the whether acknowledgement or response is expected. The device header and the raw data segments include data which is device specific.

11.1 BASIC MESSAGE FORMAT

Figure 2 describes the structure of the message interface between the module and the end-application.

Count	SRC IF	DES IF	GET	ExpACK	ExpNumByte	Device Header	Device Raw Data
Count	SRC IF	DES IF	SET	ExpACK	ExpRSP	Device Header	Device Raw Data
Count	SRC IF	DES IF	RSP	ExpACK	Device Header	Device Raw Data	
Count	SRC IF	DES IF	ACK	Device Header	Device Raw Data		
Count	SRC IF	DES IF	NTFY	ExpACK	Device Header	Device Raw Data	
Count	SRC IF	DES IF	Multi GET	ExpACK	Device Header	Device Raw Data	
Count	SRC IF	DES IF	Multi RSP	ExpACK	Device Header	Device Raw Data	

FIGURE 2: ORIGINSMART™ BASIC MESSAGE FORMAT

Table 2: Header Message Format Definition

Byte	Field	Value	Description
1,2	Count	0x0008-0xFFFF	Message Length
3	SRC IF	0x00-0x7F	The interface ID that initiates the message
4	DES IF	0x00-0x7F	The destination interface ID where the message will be delivered
5	MSG Type	0x01	NTFY: Upon detection of event, the device sends Notify message to the End-application. The data details are in Device Raw Data segment.
		0x02	GET: End-application GET message which will send a read request to the corresponding device. The requested reading parameters will be retrieved from the device Raw Data segment.
		0x04	SET: End-application SET message which will send a write request to the device. The writing data parameters will be retrieved from the device Raw Data segment
		0x08	ACK: ACK message by the device to acknowledge the received Get and Set command from the End-application. ACK message by the End-application to acknowledge the received Notify command from the device
		0x10	RSP: RSP message can be initiated by the device to

			response to GET message from the End-application. The raw data will be included in the Device Raw Data segment
		0x14	Multi GET*: Multi GET sends message from End-application to module to start/stop streaming of data from up to three devices on the I2C bus to the end application
		0x16	Multi RSP*: Multi RSP is issued by the module as response from Multi GET command.
6	ExpAck	0x20=NO 0x21=YES	Indicates to other party if ACK message is expected
7	ExpRSP / ExpNumBYTE	0x40=NO 0x41=YES 0x01-0xFF	Indicates to other party if RSP message is expected Only for GET type, the byte 7 is dedicated to set the expected number of bytes the End-application is expecting from the device

* Applicable only for I2C bus (IF_05)

The device header segment will indicate to the OriginSmart™ firmware to follow the proper instructions to exchange messages with the devices. GNSS module is working ex-protocol, see description in section 14.

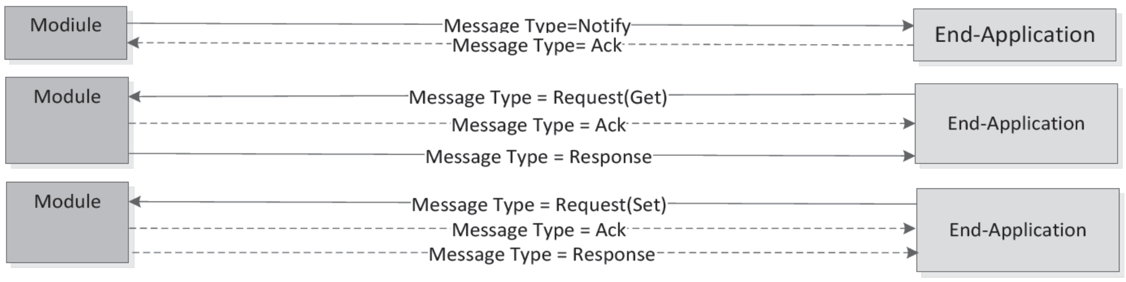


FIGURE 3: HEADER MESSAGE FLOWCHARTS

11.3 UART BASED INTERFACE

The UART/USART/LPUART interface type will not have any Device Header segment and the OriginSmart™ firmware will only process the Interface ID=0 which covers MCU internal messages. For all other UART/LPUART based interfaces, the raw data will be transparently routed to the corresponding devices without further processes.

Count	SRC IF	DES IF	GET	ExpACK	ExpNumByte	Data0	Data1	...	DataN
Count	SRC IF	DES IF	SET	ExpACK	ExpRSP	Data0	Data1	...	DataN
Count	SRC IF	DES IF	RSP	ExpACK	Data0	Data1	...	DataN	
Count	SRC IF	DES IF	ACK	Data0	Data1	...	DataN		
Count	SRC IF	DES IF	NTFY	ExpACK	Data0	Data1	...	DataN	

FIGURE 3: MESSAGE FORMAT FOR UART BASED INTERFACE

The following table describes the commands to control the MCU (DES_IF=0x00)

Table 2: MCU UART Based Message Definition

Byte	Field	Value	Description
Back from Reset Notification			
0,1	BACKFROMRESET_NTFY	0x0002	Notify the End-application when a device comes back from a reset
02	Type	0x00=Warm 0x01=Cold	
Parameter Configuration for GPIO Type Interface			
0,1	IF_PARAM_SET	0x0006	End-application request to set the interface parameters
2	IF_ID	0x00-0xFF	Interface ID
3	IF_TYPE	0x01=GPIO	For GPIO Type
4	IF_DIR	0x00=Input 0x01=Output	Interface Direction
5	IF_OP_Mode	0x00=Assigned 0x01=Pass-through	Assigned: It will be used as additional signal for serial interface. Pass-through: The Value of GPIO will be sent to End-Application.
Parameter Configuration for UART Type Interface			
0,1	IF_PARAM_SET	0x0006	End-application request to set the interface parameters
2	IF_ID	0x00-0xFF	Interface ID
3	IF_TYPE	0x02=UART	For UART Type
4	BaudRate	0x00=1200 0x01=4800 0x02=9600 0x03=19200 0x04=38400 0x05=57600 0x06=115200 0x07=auto-baud Mode 1 0x08=auto-baud Mode 2 0x09=auto-baud Mode 3	Auto baud Mode 1 uses a sync byte of single bit 1. Auto baud Mode 2 uses a sync byte 0x7f. Auto baud Mode 3 uses a sync byte 0x53. Baud rate will default to 57600 if an Autobaud mode is chosen
5	FlowCtrl	0x00=Disable 0x01=RTS 0x02=CTS 0x03=RTS and CTS	
6	StopBit	0x00=1 stop bit 0x01=2 stop bits	
7	Parity	0x00=None 0x01=Even	

		0x02=Odd	
8	Data bits	0x00=7-bit 0x01=8-bit 0x02=9-bit	
9	DataRDY_ID	0x00-0xFF=IF_ID	This will indicate the corresponding GPIO interface. For IF_ID=0x00 will indicate no GPIO is assigned.
A	DataRDY_ACTION	0x00=Internal 0x01=Pass-through	Internal: Upon logic level changes or falling, raising edge on DataRDY_ID, the firmware will read UART data Pass-through: Upon logic level changes or falling, raising edge on DataRDY_ID level to End-application. It will be up to the End-Application to request to read UART data.
B	DataOverRun_ID	0x00-0xFF=IF_ID	This will indicate the corresponding GPIO interface. For IF_ID=0x00 will indicate no GPIO is assigned
C	DataOverRun_ACTION	0x00=internal 0x01=pass-through	Internal: Upon logic level changes or falling, raising edge on DataRDY_ID, the firmware will read UART data Pass-through: Upon logic level changes or falling, raising edge on DataRDY_ID level to End-application. It will be up to the End-Application to request to read UART data.
Parameter Configuration for I2C Type Interface			
0,1	IF_PARAM_SET	0x0006	End-application request to set the interface parameters
2	IF_ID	0x00-0xFF	Interface ID
3	IF_TYPE	0x03=I2C	For I2C Type
4	BitAddrSize	0x00=7-bit 0x01=10-bit	7-bit and 10-bit addressing mode
5	BitRate	0x00=100 kbit/s	Bit rate, standard, fast or

		0x01=400 kbit/s 0x02=1000 kbit/s	fast plus
6	RegType	0x01=BYTE 0x02=WORD 0x03=LWORD	Register type
6+size (1 or 2 bytes)	Device Address	Byte or word based on bitAddrSize	
7+ size (1 or 2 bytes)	Data bits	0x00=7-bit 0x01=8-bit 0x02=9-bit	
8+ size (1 or 2 bytes)	DataRDY_ID	0x00-0xFF=IF_ID	This will indicate the corresponding GPIO interface. For IF_ID=0x00 will indicate no GPIO is assigned.
9+ size (1 or 2 bytes)	DataRDY_ACTION	0x00=Internal 0x01=Pass-through	Internal: Upon logic level changes or falling, raising edge on DataRDY_ID, the firmware will read UART data Pass-through: Upon logic level changes or falling, raising edge on DataRDY_ID level to End-application. It will be up to the End-Application to request to read UART data.
A+ size (1 or 2 bytes)	DataOverRun_ID	0x00-0xFF=IF_ID	This will indicate the corresponding GPIO interface. For IF_ID=0x00 will indicate no GPIO is assigned
B+ size (1 or 2 bytes)	DataOverRun_ACTION	0x00=internal 0x01=pass-through	Internal: Upon logic level changes or falling, raising edge on DataRDY_ID, the firmware will read UART data Pass-through: Upon logic level changes or falling, raising edge on DataRDY_ID level to End-application. It will be up to the End-Application to request to read UART data.
Interface Parameter Configuration Response			
0,1	IF_PARAM_SET_RSP	0x0007	Response from MCU to

			the IF_PARAM_SET based on IF_ID
2	IF_ID	0x00-0xFF	Interface ID
3	IF_TYPE	0x01=GPIO 0x02=UART 0x03=I2C	
4	State	0x00=Fail 0x01=Completed	Confirm that the IF_PARAM_SET has been successfully executed.
Module Registration Notification			
0,1	MODULE_REGISTER_NOTFY	0x0060	OriginIoT sends a Notify message to register at server.
2…15	IMEI	14 Bytes	IMEI number of Module
16	Value	0x00=Unregister 0x01=Register	
Keep Alive Notification			
0,1	MODULE_APP_KEEPALIVE_NOTFY	0x0063	OriginIoT modules sends keepalive message to Application
Loopback Test			
0,1	MODULE_APP_SET_LB	0x0064	Application request to echo back from module
2…	Data		
Loopback Test Response			
0,1	MODULE_APP_SET_LB_RSP	0x0065	Response from Module to MODULE_APP_SET_LB message
2…	Data		

The following table describes the commands to control the devices (GNSS, external devices)

Table 4: Device UART Based Message Definition

Byte	**Field**	**Value**	**Description**
0	Data0	0x00-0xFF	Device raw data
01…0N	Data1…DataN	0x00-0xFF	Device raw data

11.4 I2C BASED INTERFACE

The I2C based interface type contains the device header and raw date segments. OriginSmart™ FW exchanges the raw data transparently with the devices based on the device header segment.

Count	SRC IF	DES IF	GET	ExpACK	ExpNumByte	Dev_Add	Reg Start

Count	SRC IF	DES IF	SET	ExpACK	ExpRSP	Dev_Add	Reg Start	NumBytes	Data0	Data1	…	DataN

Count	SRC IF	DES IF	RSP	ExpACK	Dev_Add	Reg Start	NumBytes	Data0	Data1	…	DataN

Count	SRC IF	DES IF	ACK	Dev_Add	Reg Start

Count	SRC IF	DES IF	NTFY	ExpACK	Dev_Add	Reg Start	NumBytes	Data0	Data1	…	DataN

Count	SRC IF	DES IF	Multi GET	ExpACK	Enable	NumDevices	Rate	NumSamples	Dev_Add0	Reg Start0	NumBytes0	…	Dev_AddN	Reg StartN	NumBytesN

Count	SRC IF	DES IF	Multi RSP	ExpACK	Timer	Data0	Data1	…	DataN

FIGURE 4: MESSAGE FORMAT FOR I2C BASED INTERFACE

Table 5: Device I2C Based Message Definition

Size (Bytes)	Field	Value	Description
Based on bitAddrSize	Dev_Add	0x00-0xFF	Device slave address
Size based on Reg Type	Reg Start		I2C Register address. The first register address to be read/written
1	NumBytes	0x01-0xFF	Number of bytes of data to be written.
1	Data0… DataN	0x00-0xFF	Device raw data
1	Enable	0x00=Disable 0x10-0x12=Enable,	Disable: stops Multi_Get streaming. If Disable=0x00 is used, the remainder of the message fields are not required

		choose timer	Enable: start Multi_Get streaming Timers (0,1,2) – the system internally assigns 3 timers that the user can work with. One Multi_GET message can collect data from up to 3 devices on the I2C bus at the same data rate. If different data rates are needed the user should send multiple Multi_GET messages using different Timers. Once a certain timer is chosen, it cannot be used for another message.
1	NumDevices	0x01-0x03	Number of device in the I2C bus that participate in Multi_GET message.
2	Rate	0x0001-0x00FF	Data rate in milliseconds. 0x01=one sample per 1 millisecond. 0xFF=one sample per 256 milliseconds. Note: data rates are highly dependent on the computation load of the system, therefore higher rates are not guaranteed.
2	NumSamples	0x0000=Continuous 0x0001-0xFFFF	Number of samples to be collected and sent.

11.5 GPIO BASED INTERFACE

The GPIO interface type will not have any Device Header segment and the OriginSmarT™ firmware only processes the GPIO based interface ID and the raw data is transparently routed to the corresponding GPIO ports.

Count	SRC IF	DES IF	GET	ExpACK	ExpNumByte	Data0

Count	SRC IF	DES IF	SET	ExpACK	ExpRSP	Data0

Count	SRC IF	DES IF	RSP	ExpACK	Data0

Count	SRC IF	DES IF	ACK	Data0

Count	SRC IF	DES IF	NTFY	ExpACK	Data0

FIGURE 5: MESSAGE FORMAT FOR GPIO BASED INTERFACE

Byte	Field	Value	Description
0	Data0	(0x00, 0x01)	GPIO Data

12. NON-VOLTATILE DATA

12.1 HEADER NVM DATA FIELD

The NVM stores interface parameter settings as shown in table2 (IF_PARAM_SET). The user does not have access to the NVM otherwise. SW, HW release versions, unique ID and part Numbers are stored in NVM for internal usage.

12.2 DEFAULT CONFIGURATIONS

At OriginIoT™ power up, the firmware will apply the default configuration of the MCU interfaces from ID=0x00 to ID=22 as described in table 1, and enable the corresponding interface drivers, queueing and message processing.

These are the default configurations:

- ID=0x00 MCU – No default settings
- ID=0x01 2 Wire UART Serial Debug Console – Baud rate: 460800, Data Bits: 8, Parity: None, Stop Bits: 1, Hardware Flow Control: Off
- ID=0x02 4 Wire UART Cellular Interface – Baud rate: 230400, Data Bits: 8, Parity: None, Stop Bits: 1, Hardware Flow Control: RTS CTS
- ID=0x03 4 Wire UART GNSS Interface – Baud rate: 19200, Data Bits: 8, Parity: None, Stop Bits:1, Hardware Flow Control: None
- ID=0x04 LPUART 2 Wire Interface - Baud rate: 9600, Data Bits: 8, Parity: None, Stop Bits:1, Hardware Flow Control: None
- ID=0x05 I2C Interface – 7 bit addressing mode
- ID=0x06-0x07 Cellular and GNSS reset – Both set as high-level outputs
- ID=0x0A-0x0D User Defined GPIO Pins – 0A and 0B are set as inputs and 0C and 0D are set as outputs.
- ID=0x20 User Defined UART - Baud rate: 4800, Data Bits: 8, Parity: None, Stop Bits:1, Hardware Flow Control: None

12.3 DYNAMIC CONFIGURATIONS

- MCU – the End-Application can set at any time the specific MCU configuration through the messaging format and definitions that are described in Table 3. These dynamic data is stored into the NVM for reapplying configuration upon restart cases.
- GNSS – The-application will be able to configure the GNSS module at any time by sending the GNSS NMEA commands via messaging format. See section 14 for details.
- External Devices – the End-application is able to configure the external devices at any time by sending the corresponding external device commands.

13. RESTART STRATEGY

13.1 Power on Reset

When the OriginIoT™ module (MCU) comes back from the power-on-reset, the OriginIoT™ software will apply the default and dynamic MCU configuration data and it will initiate the Cold Restart on GNSS and Cellular modules by writing the dedicated MCU GPIO pins that are connected to GNSS and Cellular activation and reset pins. When the connection is re-established with the end-application the OriginIoT™ the firmware will notify the end-application about the restarted device. It will be up to the end-application to send the dynamic configuration for the module.

13.2 Hardware Reset Pin

Hardware reset pin behaves as described in section 13.1

14. GNSS FUNCTIONALITY

Despite being assembled on the module itself, GNSS module should be regarded as external device connected to 4 Wire UART ID=03. As so, its operation should be according to the message format described in this document. Please refer to **Spider and Hornet - NMEA Protocol Reference Manual** document to get acquainted with the relevant NMEA messages for this module.

From NMEA messages one should strip $ sign and checksum data (checksum enable byte should be set to enable). OriginSmart™ FW adds this sign and checksum data and sends the complete message to the GNSS module. All NMEA messages should be converted from string to HEX before they are sent to the module.

GNSS is configured to start-up and run immediately after module start-up. Its default configuration is sending RMS type message once per second and transfer it via cellular module to the end-application.

15. SAMPLE MESSAGES

15.2 External Accelerometer ADXL345

- IF_PARAM_SET for Accelerometer ADXL345 on I2C bus address 3a

00 11 80 00 04 20 41 00 06 05 03 00 00 01 3a 00 00 00 00

00 11	80	00	04	20	41	00 06 05	03 00 00 01 3a 00 00 00 00
count	src	dest	type	ack	rsp	dev h	dev d

This message has to be sent one, the data is saved at NVM and needs to be restored only after upload of new firmware.

Response: 00 0B 00 80 10 20 00 07 05 03 01

- ADXL345 set data format, writing the data 01, to register 31

00 0b 80 05 04 20 41 3a 31 01 01

00 0b	80	05	04	20	41	3a	31 01 01
count	src	dest	type	ack	rsp	dev h	dev d

This message data is not stored in NVM and should be sent again after each time the module is powered up.

Response: 00 09 05 80 10 20 3A 31 01

- Set ADXL to measure mode by sending data 28 to register 2d

00 0b 80 05 04 20 41 3a 2d 01 28

00 0b	80	05	04	20	41	3a	2d 01 28
count	src	dest	type	ack	rsp	dev h	dev d

This message data is not stored in NVM and should be sent again after each time the module is powered up.

Response: 00 09 05 80 10 20 3A 2D 01

- Read from ADXL one time, read 6 bytes starting at register 32

00 09 80 05 02 20 06 3a 32

00 09	80	05	02	20	06	3a	32
count	src	dest	type	ack	n bytes	dev h	dev d

Response: 00 0F 05 80 10 20 3A 32 06 00 00 00 00 7B 00

The response message includes raw data of X,Y,Z axis in two bytes complement (11 bits). The format can be changed by altering the 31 register byte as shown above. One can see that there are zeroes in the X and Y axis (since the module was stationary on the table when sampled) and Z equal to approximately 1g to represent gravity.

13.3 Multi GET

All the devices that participate in the Multi_GET message should be initialized by set messages as shown in the example above. Only I2C devices can participate in MultiGet messages:

- Start streaming of accelerometer data at 256ms framerate

```
00 0e 80 05 14 20 10 01 ff 00 00 ff 3a 32 06
```

Responses:

```
00 0F 05 80 16 20 00 06 00 F9 FF 00 00 7C 00 0A
```

Last 6 bytes are changing as you move the module

- Stop streaming of accelerometer data

```
00 07 80 05 14 20 00
```