Document	Global Top GPS NMEA over I2C Software Guide				
Author	Steve Chen	**Date**	2016/02/03	**Ver.**	1.2

Global GPS NMEA over I^{2}C Software Guide V 1.2

Steve Chen

Document	Global Top GPS NMEA over I2C Software Guide				
Author	Steve Chen	**Date**	2016/02/03	**Ver.**	1.2

Version History

History			
Date	**Rev.**	**Author**	**Description**
2014/10/01	1.0	Stanly Lin	First Release
2015/07/23	1.1	Steve Chen	Modify 1.slave address 2.Gmm-3301 I²C pin definition
2016/02/03	1.2	Steve Chen	Preface added: announcement of I²C, Change GlobalTop Logo

Document	Global Top GPS NMEA over I2C Software Guide				
Author	Steve Chen	**Date**	2016/02/03	**Ver.**	1.2

0. Preface

Announcement: I²C is now available for the GNSS modules* produced by GlobalTop Technology (programming on software; no hardware modification is needed).

**Modules available for I²C :*

Ivory 3 (GMM-U2P)

Ivory4 (FGPMMOSL3C)

Firefly1(GMM-G3)

Firefly X1(GMM-3301)

Firefly1b(GMM-G3(B))

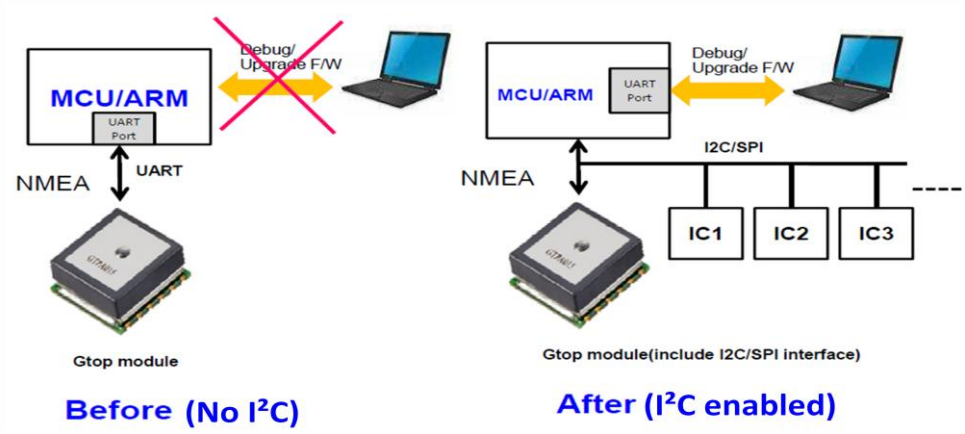
Titan2(GMS-G6)

Titan3(GMS-G9)

When I²C is not enabled, communication between MCU/ARM and chip is in single channel, meaning that the data(NMEA) are transmitted via the UART port, and such configuration makes the UART port no room to perform other tasks such as “Debug” or “Firmware Upgrade” since the port is occupied.

In contrast, when I²C is enabled, data(NMEA) are transmitted via 2 wires; this configuration allows UART port to perform tasks such as “Debug” or “Firmware Upgrade” since UART port is left open to connect to other devices. In addition, the wires can string up modules as many as possible for communication between modules (please refer to Figure 1).

Figure1(without & with I²C):



Flexibility of I²C is the key that makes I²C powerful to various applications.

Here are some significant features and characteristics in I²C:

- It requires only two bus wires
- No strict baud rate requirements
- Straightforward master/slave relationship between all components (GlobalTop modules are configured as slave)
- Each device connected to the bus is assigned by a unique address (software addressable)
- I²C is an absolute multi-master bus providing arbitration and collision detection

However, please note that RTCM will be disabled once I²C is enabled. Please contact us at sales@gtop-tech.com for any question or request regarding to I²C.

Document	Global Top GPS NMEA over I2C Software Guide				
Author	Steve Chen	Date	2016/02/03	Ver.	1.2

1. Description

GlobalTop MT3339/MT3333 GPS module supports both UART and I²C communication peripherals. This software guide instructs how to access GlobalTop module with I²C two-wire peripheral.

2. GlobalTop GPS module I²C specification

MT3339/MT3333 supports fast mode, bit rate up to 400kbit/s.

MT3339/MT3333 supports 7bit address.

MT3339/MT3333 works as slave mode.

MT3339/MT3333 default slave address is 0x10.

Command	Slave ID	R/W	ID+R/W
Read	0010000 (0x10)	1	00100001 (0x21)
Write	0010000 (0x10)	0	00100000 (0x20)

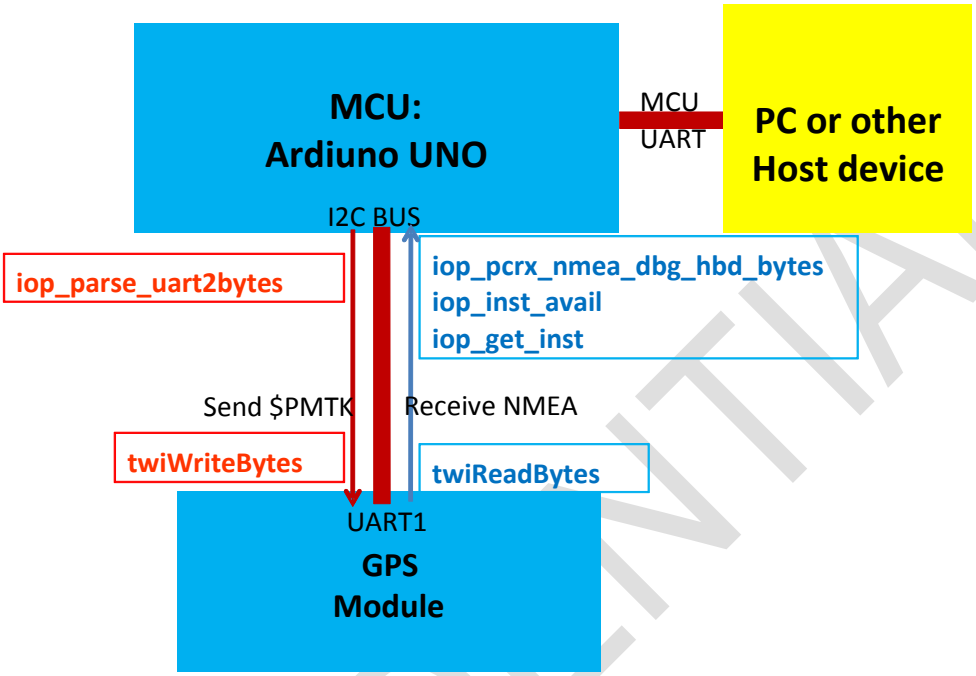
I²C Interface	Description
I²C SDA pin	It outputs GPS information for application.
I²C SCL pin	It is used to receive the clock for I²C application
INT pin	It is used to determine has NMEA or not from I²C buffer of module.

3. Host processor I²C -master Receive NMEA Flow

MT3339/MT3333 I²C TX buffer has 255 bytes in size. Master can read one I²C data packet with maximum 255 bytes at a time. In order to get complete NMEA packet of one second, master needs to read several I²C data packets and then extracts valid NMEA data from the packets. After one I²C data packet is read, set master to sleep 2ms for it to start to receive next I²C data packet, as 2ms is required for MT3339/MT3333 to upload new NMEA data into I²C TX buffer. When entire NMEA packet of one second is read, master will sleep longer in order to wait entire NMEA packet of next second coming.

In this SW guide, Ardiuno UNO EV-board is used as platform MCU.

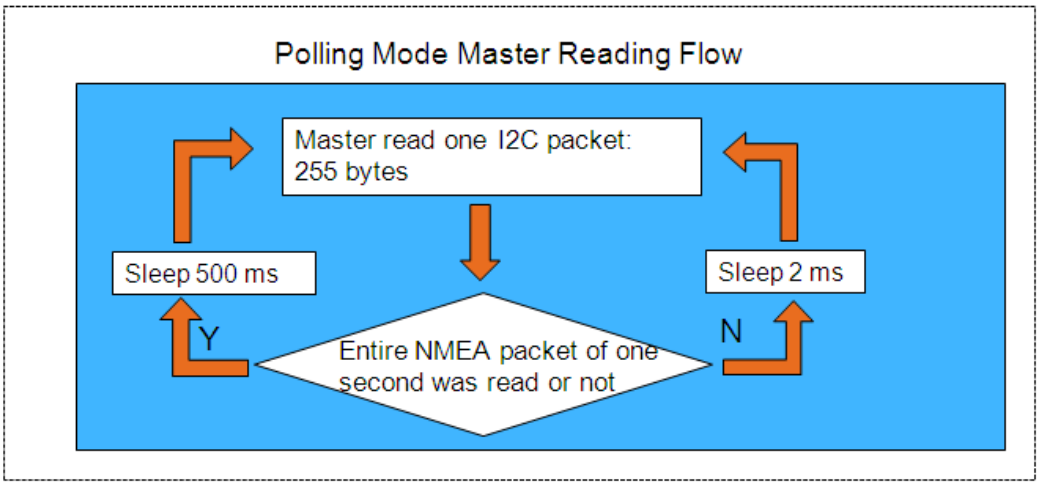
The hardware connection and software architecture are simply shown in the following chart:



MTK has two modes for NMEA to be read via I²C, and they are **Polling mode** and **Interrupt mode**.

1. Polling mode

In polling mode master reads entire NMEA packet of one second by repeatedly reading each polling time interval. This time interval can be configured according to GPS fix interval. It should be less than GPS fix interval.

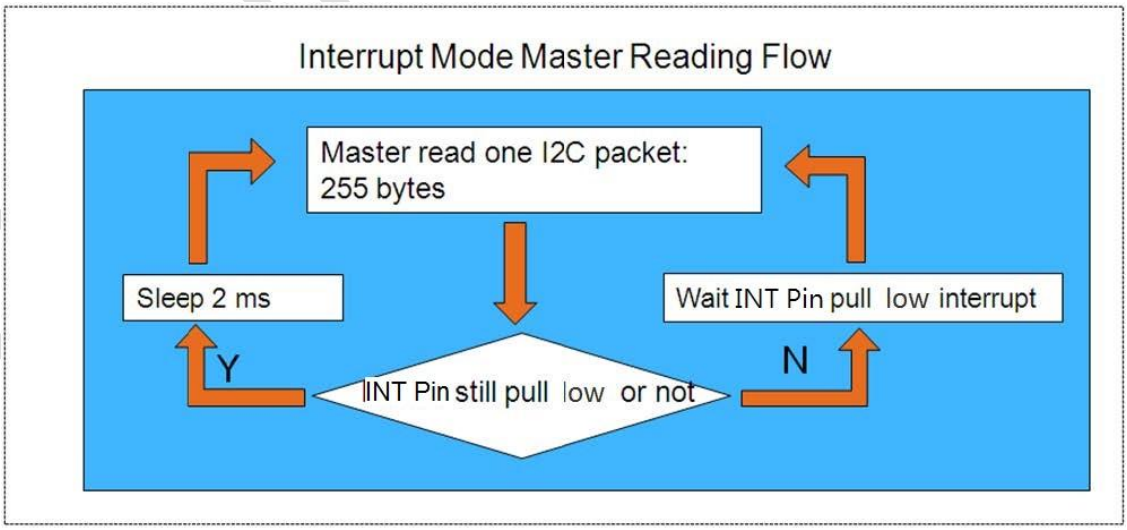


Note: the figure above assumes GPS fix interval to be 1 second, so polling time interval is set to 500ms.

2. Interrupt Mode

This mode is used when the module supports interrupt Pin. When NMEA data is ready and uploaded into I²C buffer, the interrupt Pin will pull high. After an NMEA packet of one second is completely read, the interrupt Pin pull low.

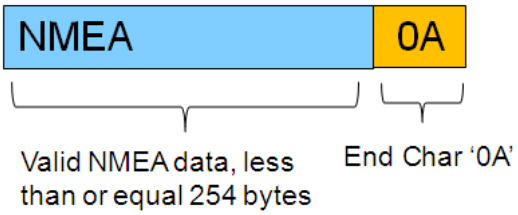
Note: Interrupt Pin can also be used in polling mode to determine slave whether has NMEA or not.



3.1 I²C data packet format in slave buffer

I²C data packet in slave buffer has 254 valid NMEA bytes at most and one end char <LF>, so maser will read 255 bytes for an I²C data packet at one time. When slave I²C TX buffer is empty, master will read one I²C data packet (255 bytes; all data are garbage bytes). The method to process garbage bytes will be introduced in following section.

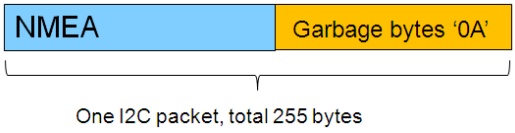
Packet format in slave I²C buffer:



3.2 Three type of I²C packet that master read from slave

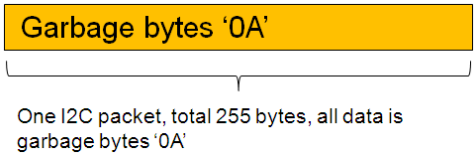
1. When slave buffer has already saved some data, master will read one I²C packet (255 bytes) from slave; it includes some valid data in the header of packet and some garbage bytes in the end of packet.

I²C packet format:



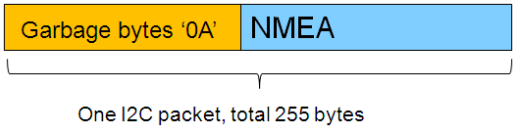
2. When slave buffer is empty, master will read one I²C packet (255 bytes) from slave. All data in packet are garbage bytes.

I²C packet format:



3. If MT3339/MT3333 I²C buffer is empty, master will start to read one I²C packet (it will read garbage bytes in the beginning). Even the reading procedure is not finished, MT3339/MT3333 will keep uploading new data into I²C buffer, and master will only read valid NMEA data bytes.

I²C packet format:



3.3 How to extract valid NMEA data from many I²C packets

As above section described, valid NMEA data need to be extracted from many I²C packets. MTK provides sample code for extracting valid NMEA data. It will be introduced in next section.

Note: When extracting NMEA data from I²C packets, all '0A' Characters will be discarded. There will have three situations: (1) the end char of one I²C packet '0A'. (2) Garbage bytes ('0A' under normal circumstances). (3)The end char of NMEA sentence '0A', discarding it doesn't affect to parse NMEA sentence.

4 MT3339/MT3333 Write PMTK Command via I²C bus

User can input PMTK command via I²C bus. Since the capacity for MT3339/MT3333 I²C RX buffer is 255 bytes, one I²C packet that master inputs must be less than 255 bytes, and the time interval of two input I²C packets cannot be less than 10 milliseconds because it takes slave 10 milliseconds to process input data.

5 Sample Code to Receive NMEA and Send PMTK

MTK provides API and sample code for receiving NMEA data and sending PMTK.

5.1 Providing API for Receiving Queue

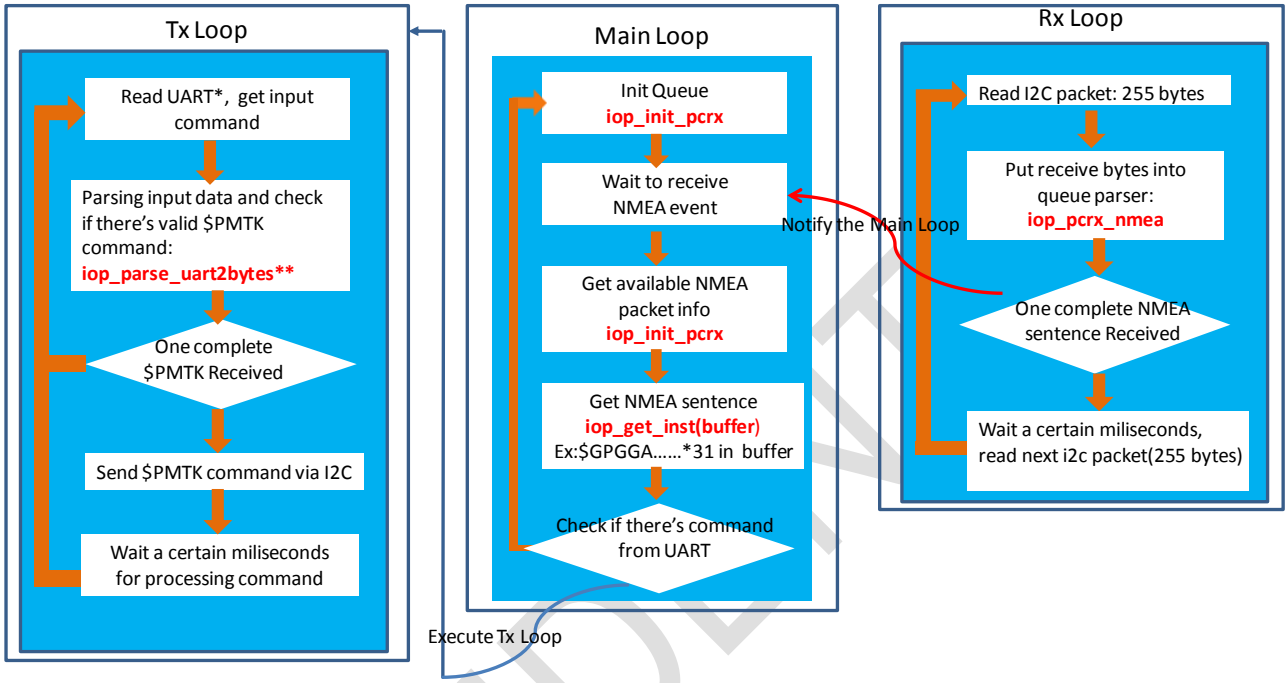
NMEA and MTK Debug data will be extracted by Queue received from many I²C packets. Queue will also discard garbage bytes and valid data automatically.

Document	Global Top GPS NMEA over I2C Software Guide				
Author	Steve Chen	**Date**	2016/02/03	**Ver.**	1.2

Function Name	Description	Prototype and Parameter
iop_init_pcrx	Initialize receive queue	
iop_inst_avail	Get available NMEA sentence information.	BOOL iop_inst_avail(short *inst_id, short *dat_idx, short *dat_siz) inst_id - NMEA sentence type dat_idx - start data index in queue dat_siz - NMEA sentence size
iop_get_inst	Get NMEA sentence data from queue buffer	void iop_get_inst(short idx, short size, void *data) idx - start data index in queue size - NMEA sentence size data - data buffer used to save NMEA sentence
iop_pcrx_nmea	Process I²C packets, get valid NMEA data and discard garbage bytes.	void iop_pcrx_nmea(unsigned char data)
iop_pcrx_nmea_dbg_hbd_bytes	Process I²C packets, get valid NMEA data and debug log code.	void iop_pcrx_nmea_dbg_hbd_bytes(unsigned char aData[], int i4NumByte) aData[] - the array keeps all data read form I²C packets. i4NumByte - size of the array keeps all data read form I²C packets

5.2 Firmware Flow of Receiving Queue

Workflow of receiving Queue is as following:



Note:

*:The item "UART" here means the MCU Serial Port connecting to PC or other Host. User can get NMEA and input $PMTK commands through this interface.

**:This function is platform dependent. It is used for MCU to receive UART input data. For different platform, users must implement this function by themselves. The $PMTK command is ranging from 8 to 64bytes in length. It is required to have enough buffer size in firmware to keep these data bytes.