u-blox AG
Zürcherstrasse 68
8800 Thalwil
Switzerland
www.u-blox.com

Phone +41 44 722 7444
Fax +41 44 722 7447
info@u-blox.com

Display Data Channel (DDC) Serial Communication Bus

Implementation for u-blox 5

Application Note

Abstract

This document provides u-blox customers with an overview of DDC serial bus architecture, characteristics and design considerations. The information is provided to assist customers in implementing these powerful communications interfaces into their u-blox GPS designs.

your position is our focus

Title	Display Data Channel (DDC)	
Subtitle	Serial Communication Bus	Implementation for u-blox 5
Doc Type	Application Note	Preliminary
Doc Id	GPS.G5-X-08023	

Revision Index	**Date**	**Name**	**Status / Comments**
Initial Version	6/10/2008	TG	

Pb	Products marked with this lead-free symbol on the product label comply with the "Directive 2002/95/EC of the European Parliament and the Council on the Restriction of Use of certain Hazardous Substances in Electrical and Electronic Equipment" (RoHS).
⚠	This is an Electrostatic Sensitive Device (ESD). Observe precautions for handling.

Contents

1 Introduction

The Display Data Channel (DDC) link is a digital serial bus available with all u-blox 5 GPS receiver chipsets and most u-blox 5 modules. To confirm DDC availability with your specific u-blox GPS receiver module, see the product features matrix on the u-blox website (http://www.u-blox.com/products/ublox5_linecard.html).

DDC specifies a 2-wire communication interface based on the industry standard I²C bus. I²C (Inter-Integrated Circuit) is a multi-master serial communications bus developed by Philips that is used to attach low-speed peripherals to a motherboard, embedded system, or cellphone.

☞ For detailed information about the I²C standard, consult the *I²C-bus Specification and User Manual* (http://www.nxp.com/acrobat_download/usermanuals/UM10204_3.pdf).

This document provides an overview of DDC architecture, characteristics and design considerations. The information is designed to assist customers in implementing DDC in u-blox GPS designs.

2 Overview

2.1 Typical Set-up

Two wires, serial data (SDA) and serial clock (SCL), carry information between the devices connected to the bus. These lines are connected to all devices on the DDC. SCL is used to synchronize data transfers and SDA is the data line. Both SCL and SDA lines are "open drain" drivers. This means that DDC devices can only drive them low or leave them open. The pull-up resistor (Rp) pulls the line up to V_{DD} if no DDC device is pulling it down to GND. If the pull-up resistors are missing, the SCL and SDA lines –are undefined and the DDC bus will not work. For most DDC systems the low and high input voltage level thresholds of SDA and SCL depend on V_{DD}. See receiver datasheet for the applicable voltage levels.

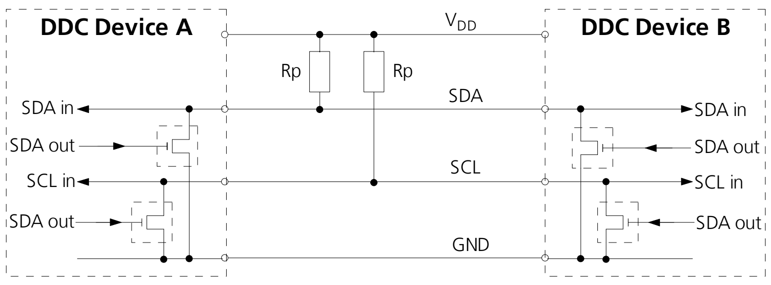


Figure 1: A simple DDC connection

The signal shape and the maximum rate in which data can be transferred over SDA and SCL is limited by the values of Rp and the wire and I/O capacitance (Cp). Long wires and a large number of devices on the bus increase Cp, therefore DDC connections should always be as short as possible. The resistance of the pull-up resistors and the capacitance of the wires should be carefully chosen.

Figure 2 shows a basic block diagram for a DDC interface integrating a u-blox GPS receiver.

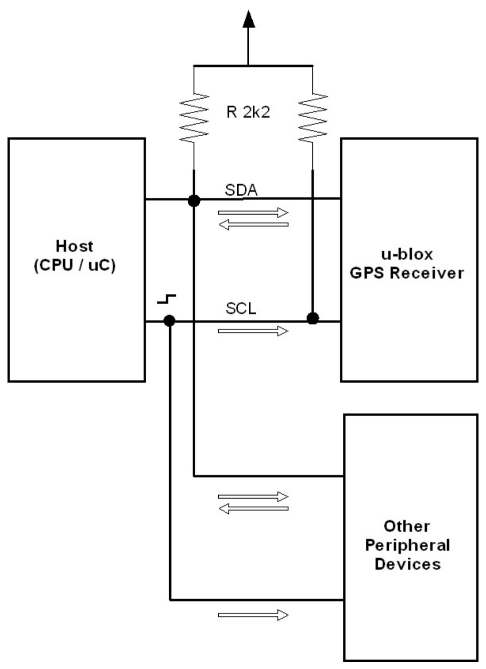


Figure 2: DDC Block Diagram

2.2 Addresses, Roles and Modes

Each device connected to a DDC is identified by a unique 7-bit address (e.g. whether it's a microcontroller, EEPROM or D/A Converter, etc) and can operate as either a transmitter or receiver, depending on the function of the device. The default DDC address for u-blox GPS receivers is set to 0x42. Setting the mode field in the CFG-PRT message for DDC accordingly can change this address.

- The first byte sent is comprised of the address field and R/W bit. Hence the byte seen on the bus 0x42 is shifted by 1 to the left plus R/W bit thus being 0x84 or 0x85 if analyzed by scope or protocol analyzer.

In addition to transmitters and receivers, devices can also be considered as masters or slaves when performing data transfers. A master is the device which initiates a data transfer on the bus and generates the clock signals to permit that transfer. At that time, any device addressed is considered a slave. The DDC-bus is a multi-master bus, i.e. multiple devices are capable of controlling the bus. Such architecture is not permanent and depends on the direction of data transfer at any given point in time. A master device not only allocates the time slots when slaves can respond but also enables and synchronizes designated slaves to physically access the bus by driving the clock. Although multiple nodes can assume the role of a master, only one at any time is permitted to do so. Thus, when one node acts as master, all other nodes act as slaves. Table 1 shows the possible roles and modes for devices connected to a DDC bus.

	Transmit	**Receive**
Master: sends the clock and addresses slaves	Sends data to slave	Receives data from slave
Slave: receives the clock and address	Sends data to master	Receives data from master

Table 1: Possible Roles and Modes of Devices Connected to DDC Bus

The u-blox GPS receiver normally runs in the slave mode. There is an exception when the u-blox GPS receiver becomes a master on the bus and this is when an external EEPROM is attached. In that case, the receiver attempts to establish presence of such a non-volatile memory component by writing and reading from a specific location. If EEPROM is present, the receiver assumes role of a master on the bus and never changes its role to slave until the following start-up (subject to EEPROM presence). This process takes place only once at the start-up, i.e. the receiver's role cannot be changed during the normal operation afterward. This model is an exception and should not be implemented if there are other participants on the bus contending for the bus control (µC / CPU, etc.).

Since the physical layer lacks a handshake mechanism to indicate the data availability, a layer has been inserted between the physical layer and the UBX and NMEA layer. The DDC implements a simple streaming interface that allows for constant data polling, discarding the segments of the data stream that do not belong to a valid UBX or NMEA message. Thus the u-blox GPS receiver returns 0xFF if no data is available. If the polling process is suspended for an extended period of time of 1.5 sec, the receiver temporarily stops writing data to the output buffer to prevent overflowing.

Being a slave on the bus, the u-blox GPS receiver cannot initiate the data transfers on its own. The master node has an exclusive right and responsibility to generate the data clock, therefore the slave nodes need not be configured to use the same baud rate. For the purpose of simplification, if not specified differently, SLAVE denotes the u-blox GPS receiver while MASTER denotes the external device (CPU, µC) controlling the DDC bus by driving the SCL line.

- u-blox GPS receivers support standard mode I^{2}C-bus specification with 7-bit addressing and a data transfer rate up to 100 kbit/s..

2.3 Communication

Unlike USB, USART and SPI interfaces, DDC is not able to communicate in full-duplex mode, i.e. transmission and reception cannot occur simultaneously. In other words, the bus is of a "drop-off" type. With DDC, the actual Baud rate is not fixed to a constant value. Instead, the master device pulses the clock for each transmitted bit. The slave has to follow this speed, which can lead to problems if the slave is not yet ready to send or receive more data. The DDC protocol provides a solution to this in that the slave can hold the SCL line low. This mechanism is referred to as clock stretching.

With clock stretching an addressed slave device may hold the clock line (SCL) low after receiving (or sending) a bit, indicating that it is not yet ready to process more data. The master that is communicating with the slave will attempt to raise the clock to transfer the next bit, but must verify that the clock line was actually raised. If the slave is clock stretching, the clock line will still be low until it is ready to continue communication. This mechanism assures that a slave device can protect itself from being addressed too quickly.

2.3.1 START and STOP Conditions

The DDC standard defines two special conditions for communication. These are the START and STOP conditions. The START and STOP conditions are the only situations where the SDA (data line) is allowed to change while the SCL (clock line) is high. When data is being transferred, SDA must remain stable and not change while SCL is high. The START and STOP conditions mark the beginning and end of a transaction with the slave device.

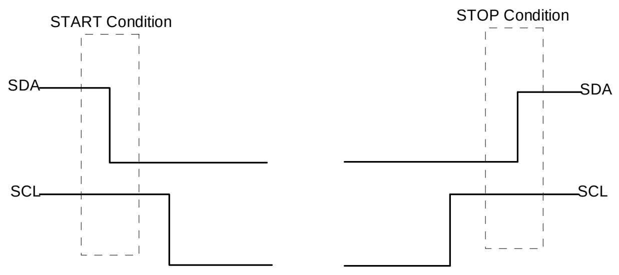


Figure 3: DDC START and STOP Conditions

2.3.2 Data Sequence

Data is transferred in sequences of 8 bits. The bits are placed on the SDA line starting with the MSB (Most Significant Bit). The SCL line is then pulsed high, then low. For every 8 bits transferred, the device receiving the data sends back an acknowledge bit, so there are actually 9 SCL clock pulses to transfer each 8 bit byte of data. If the receiving device sends back a low ACK bit, then it has received the data and is ready to accept another byte. If it sends back a high then it is indicating it cannot accept any further data and the master should terminate the transfer by sending a STOP sequence.

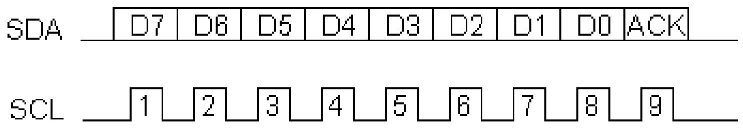


Figure 4: DDC Data Sequence

2.3.3 Communication Process

The following rules and conditions apply while communicating over the bus:

- Data transfer may be initiated only when the bus is not busy
- During data transfer, the data line must remain stable whenever the SCL clock line is high. This scenario, as indicated later, is used in generation of the Start and Stop Conditions.

Communications over a DDC bus basically takes place as follows: The master first transmits a START condition. The START alerts all slave devices on the bus that a transaction is beginning. Next the master transmits the address of the specific slave device, followed by a single bit indicating if it wishes to write to (0) or read from (1) the slave. If the addressed slave device is connected to the bus it responds with an ACK bit (acknowledge) for that address, all the other slave devices on the bus ignore the rest of this transaction and wait for the next START. The master then continues in either transmit or receive mode (according to the read/write bit it sent), and the slave continues in the complementary mode. Having addressed the slave device the master must now send out the internal location or register number inside the slave that it wishes to write to or read from. The slave again responds with an ACK-bit.

In write mode the master transmits the information to the slave to be recorded in the specific registry location. Additional bytes are typically placed in the following registers because the slave automatically increments the internal register address after each byte. When the master has finished writing all data to the slave, it sends a STOP sequence, which completes the transaction. In read mode the master first sends the address sequence to the slave in write mode and follows this with a STOP sequence. The master then sends another START sequence (sometimes called a RESTART) and the DDC address again - this time with the read bit set. The master can then read as many data bytes as desired and terminates the transaction with a STOP sequence.

2.4 Bus Conditions and Events

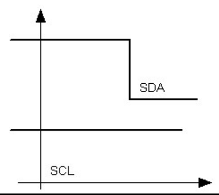
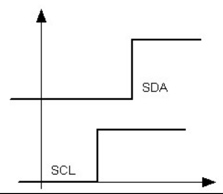
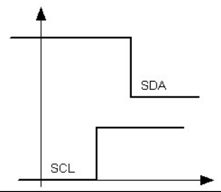
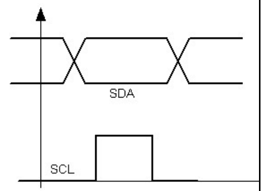
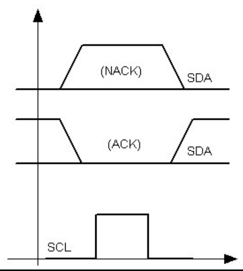
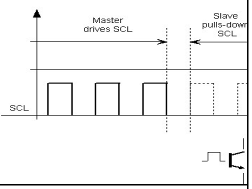
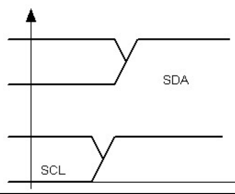
Event or Condition	Description	Signal
Start Condition (Data Transfer Initiation)	A high-to-low SDA transition, while SCL is held high, determines a Start Condition. A bus Idle State is assumed to precede this state. All data transfers are preceded by Start Conditions.	
Stop Condition (Terminate Data Transfer)	A low-to-high SDA transition, while SCL is held high, determines a Stop Condition. All data transfers end with Stop Conditions.	
Repeated Start	A high-to-low SDA transition, while SCL is held high, determines a Repeated Start condition. Wait State is assumed to precede this condition. With this condition bus direction master can change direction of the data flow without loosing control over the SCL line (see the example below).	
Data Valid	After the Start Condition, the state of the SDA line represents valid data when the SDA line is stable for the duration of one full SCL cycle. There is one bit of data per SCL clock.	
ACK or NACK	All data byte transmissions must be Acknowledged (ACK) or Not Acknowledged (NACK) by the recipient. The recipient pulls the SDA line low for an ACK or releases the SDA line for a NACK. The Acknowledge is a one-bit period referenced to one SCL full cycle.	
Clock Stretching	Slave is not ready to accept new data due to a pending interrupt and requires more time to process tasks in the background. Master is suspended until all slaves release the SCL line.	
Bus Idle Condition	This is normal condition of the bus between a Stop and Start Conditions. Both SDA and SCL remain high at those times.	

Table 2: DDC Events and Conditions

3 Enabling DDC Operation

3.1 Port Configuration via UBX Message

u-blox specific configuration details:

☞ For information refer to the *u-blox 5 Protocol Specification* [3].

4 Electrical and Timing Specifications

4.1 Port Configuration via UBX Message

☞ For configuration details consult the receiver specific Data Sheets and the I^2C Standard Specification.

5 Communicating as a Master with the GPS Receiver as Slave

5.1 External Master with the GPS Receiver as Slave

In the following example, a message is read from a u-blox GPS receiver. The external device (CPU, uC,…) acts as master and the GPS receiver as slave. Master device controls the bus and timing of the events by generating the clock. While responding, the GPS Receiver drives the SDA line at determined time slots defined by the SCL (driven by master).

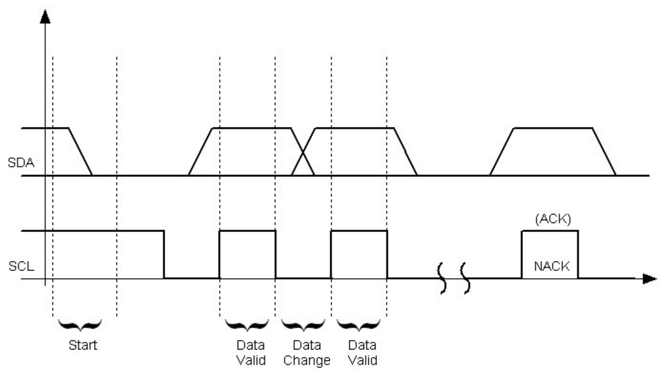


Figure 5: DDC Single Byte Transmission

If using the GPS as slave the DDC clock is generated by the master. The SCL signal should be a clean clock signal. If necessary a Schmitt-Trigger may be used to improve the quality of the signal (TBC).

5.2 Full Read / Write Access

Slave does not provide any write access except for writing UBX messages (and NMEA messages) to the receiver, such as configuration or aiding data. The following figure depicts a typical RD / WR cycle:

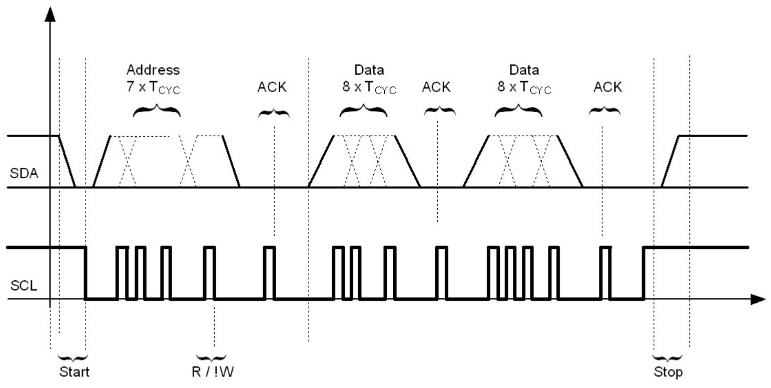


Figure 6: DDC R/ !W Cycle

START Message: Each message is initiated with a “START” condition and terminated with a “STOP” condition. The number of data bytes transferred between the START and STOP conditions is determined by master. As defined by the system protocol, the bytes of the message may have special meaning, such as “device address byte” or “data byte”.

Address Slave: The first byte is the device address byte. The slave's address is set to 0x42 by default but can be changed using CFG-PRT. This field contains a device address and R/W bit. R/W = 0 indicates master’s desire to write to and R/W = 1 desire to read from slave.

Slave Acknowledge: Slave must output an ACK / NACK after reception of each byte whether it is ready or not to receive it. Master must generate one extra SCL cycle to obtain an ACK / NACK from slave. The exception is if master is receiving data from slave. Then it must indicate the end of the transfer by not generating an ACK/NACK after the last byte is received. Please also note that all decisions about auto-increment or decrement (in the slave) of previously accessed memory locations are made internally and are not a subject of the DDC specification.

Master Transmit: The following bytes, sent by master to slave (in case of writing) or slave to master (in case of reading), are data bytes containing the UBX message. Slave must ACK each data byte. There must be anywhere from 2,…,N bytes read from the GPS receiver.

STOP Message: Master generates a STOP Condition to terminate the message. The bus returns to Idle State.

5.3 Register Map

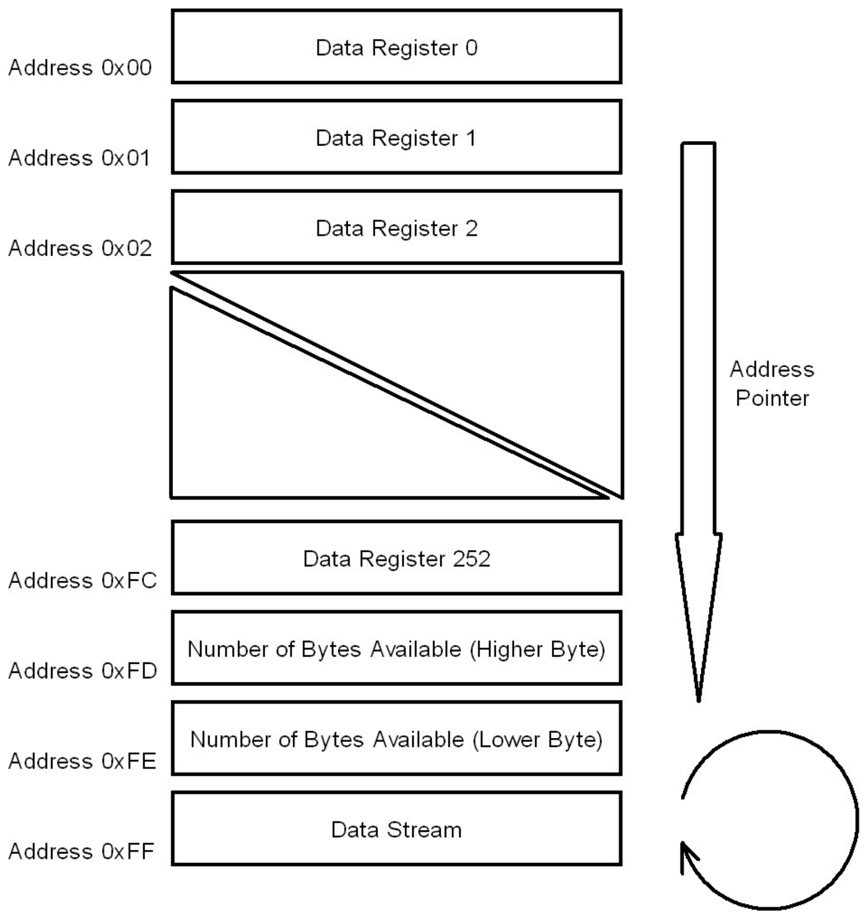


Figure 7: Register Map

To allow both polled access to the full message stream and quick access to the key data, the register layout depicted in Figure 7 is provided. The data registers 0 to 252, at addresses 0x00 to 0xFC, each 1 byte in size are reserved for future use. The registers located at 0xFD and 0xFE contain the currently available number of bytes in the message stream that can be read. The message stream is located at 0xFF. Subsequent reads from 0xFF return the messages in the transmit buffer, on the byte by byte basis. If the number of byte-read cycles exceeds the number of available bytes, the payload is padded using the value 0xFF.

☞ Registers 0x00 to 0xFC are reserved for future use and will be defined in later firmware releases. Do not use them, as they don't provide any meaningful data!

5.4 Random Read Access

Random read operations allow *master* to access any register in a random manner. To perform this type of operation, observe the following sequence:

- The register address to be read from must be indicated in the WRITE request to the receiver. Following the START CONDITION from the master, the 7-bit device address including the RW bit (logic LOW for WR access) is clocked onto the bus by *master* transmitter.
- *Slave* answers with an ACK to indicate that it is responsible for the given address.
- Next, the 8-bit address of the register to be read must be written to the bus.

- Following the *slave's* ACK, without generating the STOP CONDITION, *master* again triggers a START CONDITION and writes the device address, but this time the RW bit is logic HI to initiate READ ACCESS.
- Now, *master* can read 1 to N bytes from the *slave*, generating a NACK and a STOP CONDITION after the last byte is read.
- After every byte is read, the internal address counter is incremented by one, saturating at 0xFF. This saturation implies that the address pointer is frozen continuously pointing to the last register thus enabling the raw message stream to be read again.

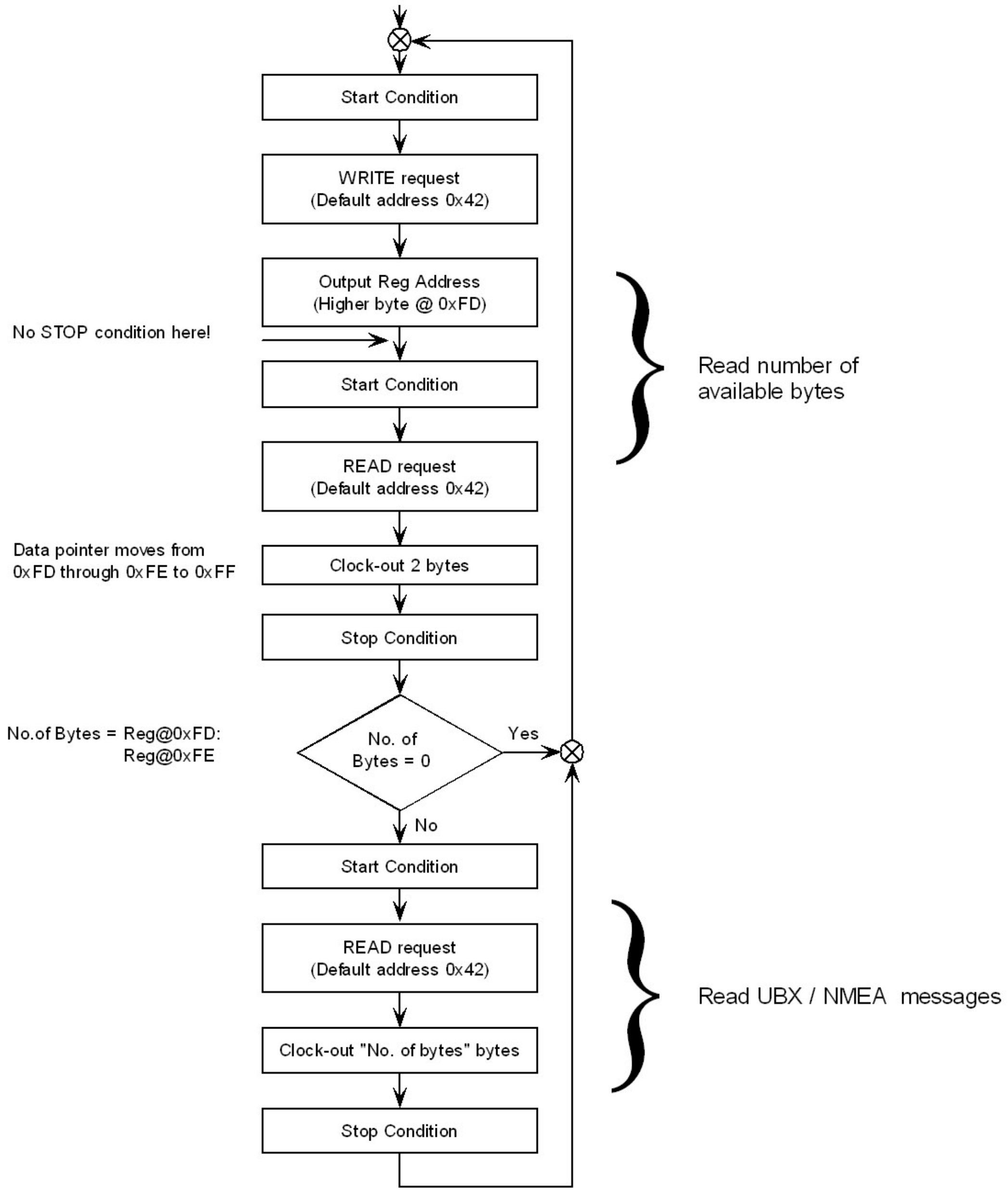

Start Condition
WRITE request (Default address 0x42)
Output Reg Address (Higher byte @ 0xFD)
No STOP condition here!
Start Condition
READ request (Default address 0x42)
Data pointer moves from 0xFD through 0xFE to 0xFF
Clock-out 2 bytes
Stop Condition
Read number of available bytes
No.of Bytes = Reg@0xFD: Reg@0xFE
No. of Bytes = 0
Yes
No
Start Condition
READ request (Default address 0x42)
Clock-out "No. of bytes" bytes
Stop Condition
Read UBX / NMEA messages

Figure 8: Read Access

5.5 DDC Current Address Read Access

Slave contains an address counter that maintains the address of the last register accessed, internally incremented by one. Therefore, if the previous read access was to address n (n is any legal address), the next current address read operation would access data from address n+1. Upon receipt of the device address with the RW bit set to one, *slave* issues an ACK and *master* can read 1 to N bytes, generating a NACK and a STOP Condition after the last byte is read. To allow direct access to streaming data, the internal address counter is initialized by default to 0xFF, i.e. current address reads (without a preceding random read access) returns the raw message stream. The address counter can be set to another address at any point in time using a random read access.

6 Communicating to a Slave with the GPS Receiver as Master

Only the second DDC interface (SDA2 / SCL2) is available for communication to peripherals or host. Pins SDA2 and SCL2 do have internal pull-ups. If capacitive bus load is large, additional external pull-ups may be needed in order to reduce the pull-up resistance.

Table 3 lists the maximum total pull-up resistor values for the DDC interface. The pull-up resistors integrated in the pads of the baseband-IC can simply be ignored for high capacitive loads. However for small loads, e.g. if just connecting to an external EEPROM, these built-in pull-ups are sufficient.

Load Capacitance	Pull-Up Resistor Value R20, R21
50 pF	18 kΩ
100 pF	9 kΩ
250 pF	4 kΩ

Table 3: Pull-Up Resistor Values for DDC Interface

Serial I^{2}C memory can be connected to the DDC interface. It will automatically be recognized by firmware. The memory address must be set to 0b1010000 and its size is fixed to 4 kB.

External memory is only supported if CFG_PIN is set to 0, i.e. configuration pins are disabled.

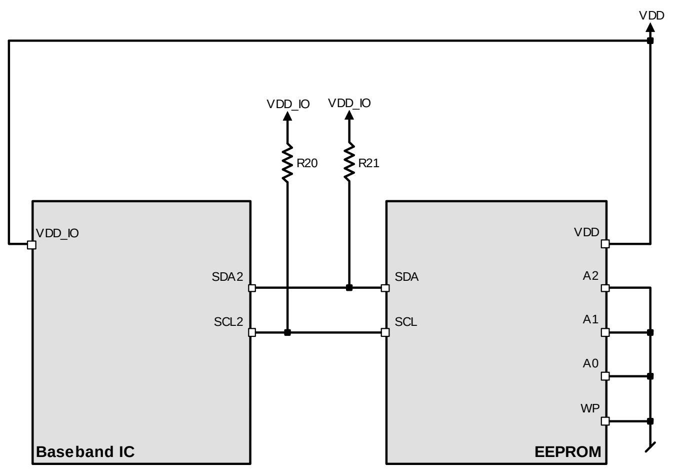


Figure 9: Connecting external Serial I2C Memory

7 Design Tips and FAQ

Data Availability

Slave starts outputting the data to the DDC buffer once all requirements are met:

1. The START CONDITION is present followed by the recognized receiver's address
2. The messages to be output are prepared at the end of the epoch

Slave, however, does not prepare messages in advance unless condition 1) is met. As a consequence of this concept the number of bytes available is initially 0 and it has to be polled continuously until it reflects the real number of available bytes. It is only then that reading can take place. If reading takes place prior to the indication that there are some bytes pending for reading slave outputs 0xFF by default.

Timeout

After addressing *slave, master* has a given amount of time to interrogate the receiver for the number of available bytes as well as to perform message reading itself. If there is no activity from the *master's* side for a period of time longer than the specified Timeout *slave* considers the connection broken. In that case, any attempt to read out the output buffer (located at the address 0xFF) using just the clock signal results in default 0xFF reading. In order to reestablish connection *master* has to regenerate the START condition including the *slave's* address.

The Timeout period is 1 sec for the firmware revisions 4.01 and earlier and 1.5 sec for the revisions 5.0 and later.

Minimum Connection

Beside usual GND link among the network participants, no other pins but SCL and SDA are required to achieve communication. The recipient selection is achieved using different addresses rather than generating CHIP SELECT or SLAVE SELECT signals to open the receivers gates as is the case with the SPI bus.

Transmitted Messages

The DDC port behaves as any other u-blox 5 communication interface so in order to transmit certain type of messages they must be enabled. No special rules apply to the DDC link.

DDC Port Configuration

If there are unknown problems and the communication link with the receiver cannot be established, alternative ports (such as USB or UART) could be used to access and/or change the DDC settings. For that reason the demo designs should support alternative communication links as well. This further helps with troubleshooting of the DDC link, which is the most complex link to implement form the system design point of view.

Data Pointer

The Data Pointer is a common source of problems while trying to establish connection with the u-blox receiver via DDC link. As indicated above in Random Read Access, the data pointer grows up with every successive READ procedure. When the end of the DDC stack is reached (address 0xFF) the Data Pointer does not wrap around but rather keeps pointing to the very end location. If at any point in time the Data Pointer points to the reserved area (addresses from 0x00 to 0xFC) the receiver reading will always return the default value 0xFF. For that reason it is advised that prior to any reading either of the following steps be performed:

- The number of available bytes is always read before reading the stream of UBX / NMEA messages. This requires presetting of the Data Pointer to 0xFC thus ensuring correct positioning prior to the reading process.

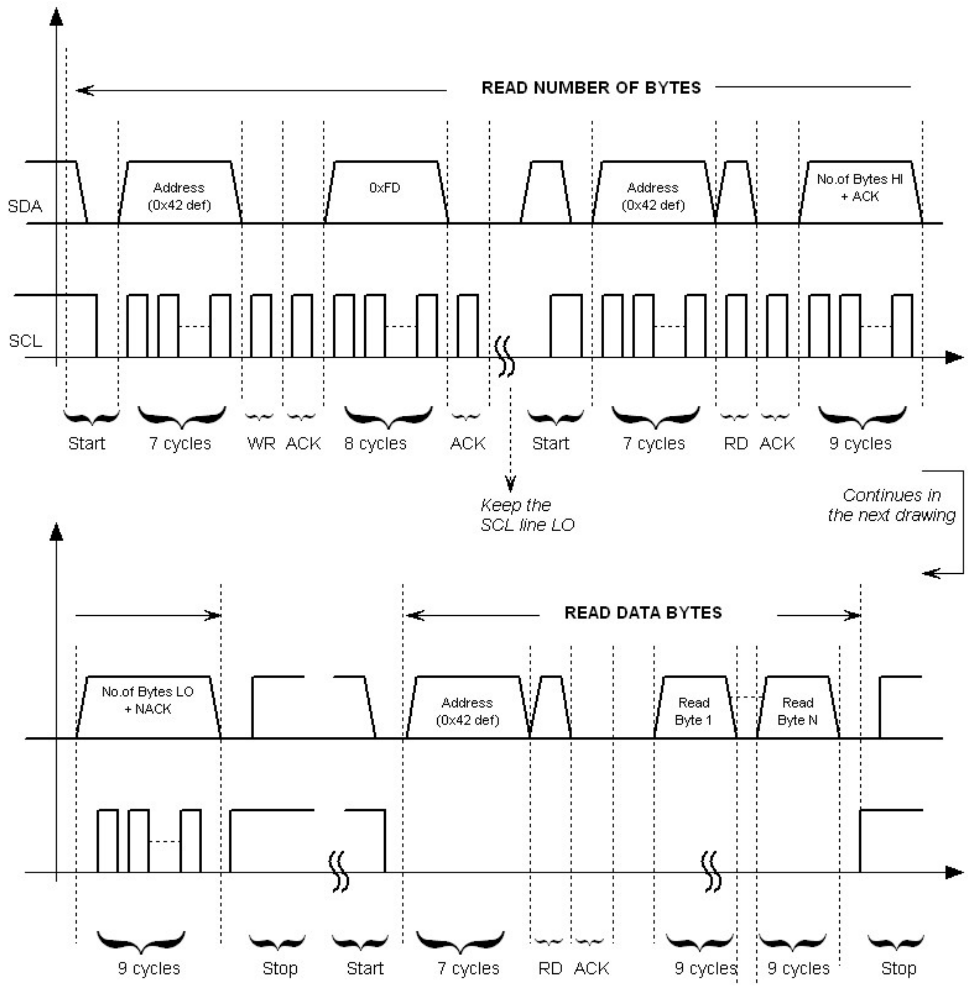


Figure 10: Full Reading (with No. of Bytes)

- The Data Pointer is simply preset to point to the location 0xFF. In that case there is no information about the number of bytes available and *master* would have to continuously read from *slave* to ensure that there is no buffer overrun. As a consequence of this approach, *master* would have to read the data at a higher rate than the *slave's* update cycle. This further results in default readings 0xFF in between adjacent packages (due to the excessive reading) that would have to be removed by the *master's* parsing routine. The figure X7 depicts the data stream reading assuming that the Data Pointer is preset.

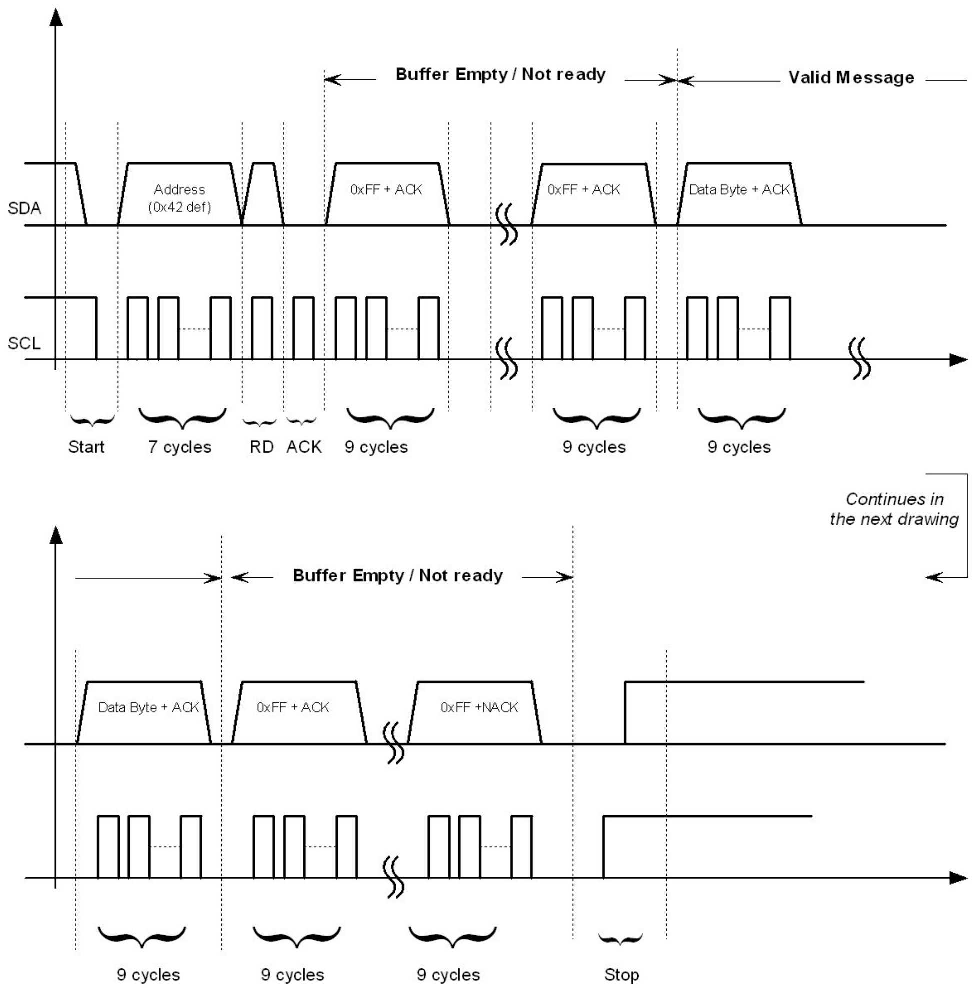


Figure 11: Full Reading (without No. of Bytes)

The Data Pointer can be modified only in the following two ways:

A valid WRITE command is issued (containing the address to be written to)

A valid READ command is issued. The Data Pointer is incremented as many times as there are byte-read cycles present on the bus. The Data Pointer does not wrap around 0xFF value.

No other combination of START or STOP conditions can reset or preset the Data Pointer, i.e. in all other cases the Data Pointer remains intact.

Data Availability

Slave starts outputting the data to the DDC buffer once all requirements are met:

The START CONDITION is present followed by the recognized receiver's address

The messages to be output are prepared at the end of the epoch.

8 Troubleshooting

- *Is there a stable supply voltage Vcc?* Often, external I2C devices (like I2C masters or monitors) must be provided with Vcc.
- *Are appropriate termination resistances attached between SDA, SCL and Vcc?* The voltage level on SDA and SCL must be Vcc as long as the bus is idle and drop near GND if shorted to GND. [Note: Very few I2C masters exist which drive SCL high and low, i.e. the SCL line is not open-drain. In this case, a termination resistor is not needed and SCL cannot be pulled low. These masters will not work together with other masters (as they have no multi-master support) and may not be used with devices which stretch SCL during transfers.]
- *Are SDA and SCL mixed up?* This may accidentally happen e.g. when connecting I2C buses with cables or connectors.
- *Do all I2C devices support the I2C supply voltage used on the bus?*
- *Do all I2C devices support the maximum SCL clock rate used on the bus?*
- If more than one I2C master is connected to the bus: *do all masters provide multi-master support?*
- *Are the high and low level voltages on SDA and SCL correct during I2C transfers?* The I2C standard defines the low level threshold with 0.3 Vcc, the high level threshold with 0.7 Vcc. Modifying the termination resistance Rp, the serial resistors Rs or lowering the SCL clock rate could help here.
- *Are there spikes or noise on SDA, SCL or even Vcc?* They may result from interferences from other components or because the capacitances Cp and/or Cc are too high. The effects can often be reduced by using shorter interconnections.

Related Documents

[1] u-blox' GPS Dictionary, Doc No GPS-X-00001

[2] GNSS Compendium, Doc No GPS-X-02007

[3] u-blox 5 Protocol Specification, Docu. No GPS.G5-X-07003

☞ For regular updates to u-blox documentation and to receive product change notifications please register on our homepage.

Contact

For further info, please contact us:

Headquarters

u-blox AG
Zuercherstrasse 68
CH-8800 Thalwil
Switzerland

Phone:	+41 44 722 74 44
Fax:	+41 44 722 74 47
E-mail:	info@u-blox.com

www.u-blox.com

Offices

North, Central and South America

u-blox America, Inc.
1902 Campus Commons Drive
Suite 310
Reston, VA 20191
USA

Phone:	+1 (703) 483 3180
Fax:	+1 (703) 483 3179
E-mail:	info_us@u-blox.com

Regional Office West Coast:

Phone:	+1 (703) 483 3184
Fax:	+1 (703) 483-3179
E-mail:	info_us@u-blox.com

Technical Support:

Phone:	+1 (703) 483 3185
E-mail:	support_us@u-blox.com

Europe, Middle East, Africa

u-blox AG
Zuercherstrasse 68
CH-8800 Thalwil
Switzerland

Phone:	+41 44 722 74 44
Fax:	+41 44 722 74 47
E-mail:	info@u-blox.com

Technical Support:

Phone:	+41 44 722 74 44
E-mail:	info@u-blox.com

Asia, Australia, Pacific

u-blox Singapore Pte. Ltd.
435 Orchard Road
#17-01, Wisma Atria,
Singapore 238877

Phone:	+65 6734 3811
Fax:	+65 6736 1533
E-mail:	info_ap@u-blox.com
Support:	support_ap@u-blox.com

Regional Office China:

Room 716-718
No. 65 Fuxing Road
Beijing, 100036, China

Phone:	+86 10 68 133 545
Fax:	+86 10 68 217 890
E-mail:	info_cn@u-blox.com
Support:	support_cn@u-blox.com

Regional Office Japan:

22F Shibuya Mark City West,
1-12-1 Dogenzaka Shibuya-ku
Tokyo 150-0043 Japan

Phone:	+81 3 4360 5343
Fax:	+81 3 4360 5301
E-mail:	info_jp@u-blox.com
Support:	+81 3 4360 5344
	support_jp@u-blox.com

Regional Office Korea:

Room 501, Gyeong Hui Building 109-18,
Samseong-Dong,
GangNam-Gu,
Seoul, Korea 135-090

Phone:	+82 2 542 0861
Fax:	+82 2 542 0862
E-mail:	info_kr@u-blox.com
Support:	support_kr@u-blox.com

Regional Office Taiwan:

Room 305
3F, #181, ZouTze Street
Neihu Dis.
Taipei, Taiwan

Phone:	+886 2 2657 1090
Fax:	+886 2 2657 1097
E-mail:	info_tw@u-blox.com
Support:	support_tw@u-blox.com