FXOS8700CQ Evaluation Kit Description and Usage

Sensors Solutions Division

Jacques Trichet – Sensors Application Engineer

N O V . 1 4 . 2 0 1 4

External Use

Freescale, the Freescale logo, AltiVec, C-5, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C-Ware, the Energy Efficient Solutions logo, Kinetis, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, Qorivva, SafeAssure, the SafeAssure logo, StarCore, Symphony and VortiQa are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. Airfast, BeeKit, BeeStack, CoreNet, Flexis, Layerscape, MagniV, MXC, Platform in a Package, QorIQ Qonverge, QUICC Engine, Ready Play, SMARTMOS, Tower, TurboLink, UMEMS, Vybrid and Xtrinsic are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners. © 2014 Freescale Semiconductor, Inc.

Introduction

- This Document first gives a brief description of the Freescale 6-axis sensor evalkit (reference RD4247FXOS8700) and more specifically the DIP16 daughter board that features the magnetometer+accelerometer combo device
- Then we'll give simple information about the protocol that allows communication with RD4247FXOS8700 evalkit and in particular is used by the PC GUI (i.e. XTRINSIC-ACCEL-MAG-SENSOR-FUSION-DEMO.exe software) to achieve continuous and autonomous raw data streaming
- We will illustrate transaction sequences with examples using a simple Serial Terminal Software to connect and communicate with the "Bridge" MCU through the USB serial port. This "bridge", located on the RD4247FXOS8700 kit intermediate Board, is a MC9S08QE08 device that realizes the USB -actually SCI- to I2C conversion and handles dumb sensors management (initialization and data collection)
- Information provided will allow user to control Freescale 6-axis development kit RD4247FXOS8700 and collect raw data without Sensor Toolbox PC software Environment

Software and Hardware used

- ***RealTerm***

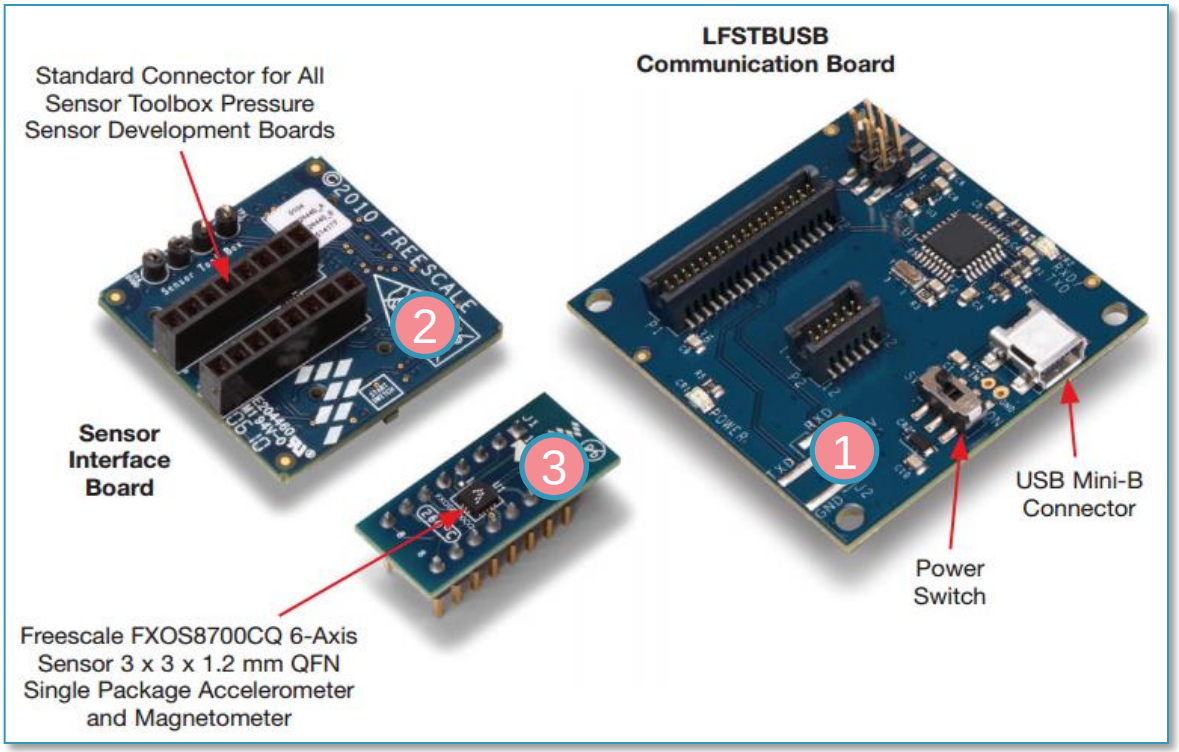
  For demonstration purpose, we will use RealTerm free Serial Terminal Software available at http://realterm.sourceforge.net/index.html#downloads_Download. Note that most Terminal Programs (such as Microsoft Window HyperTerminal) can be used as well.

- ***RD4247FXOS8700 EVK***

  The complete evalkit contains:

  1. LFSTBUSB communication board
  2. Sensor interface board
  3. FXOS8700CQ 6-axis sensor development board

  This hardware (see picture) can be ordered directly at http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=RD4247FXOW8700&tab=Buy_Parametric_Tab&fromSearch=false

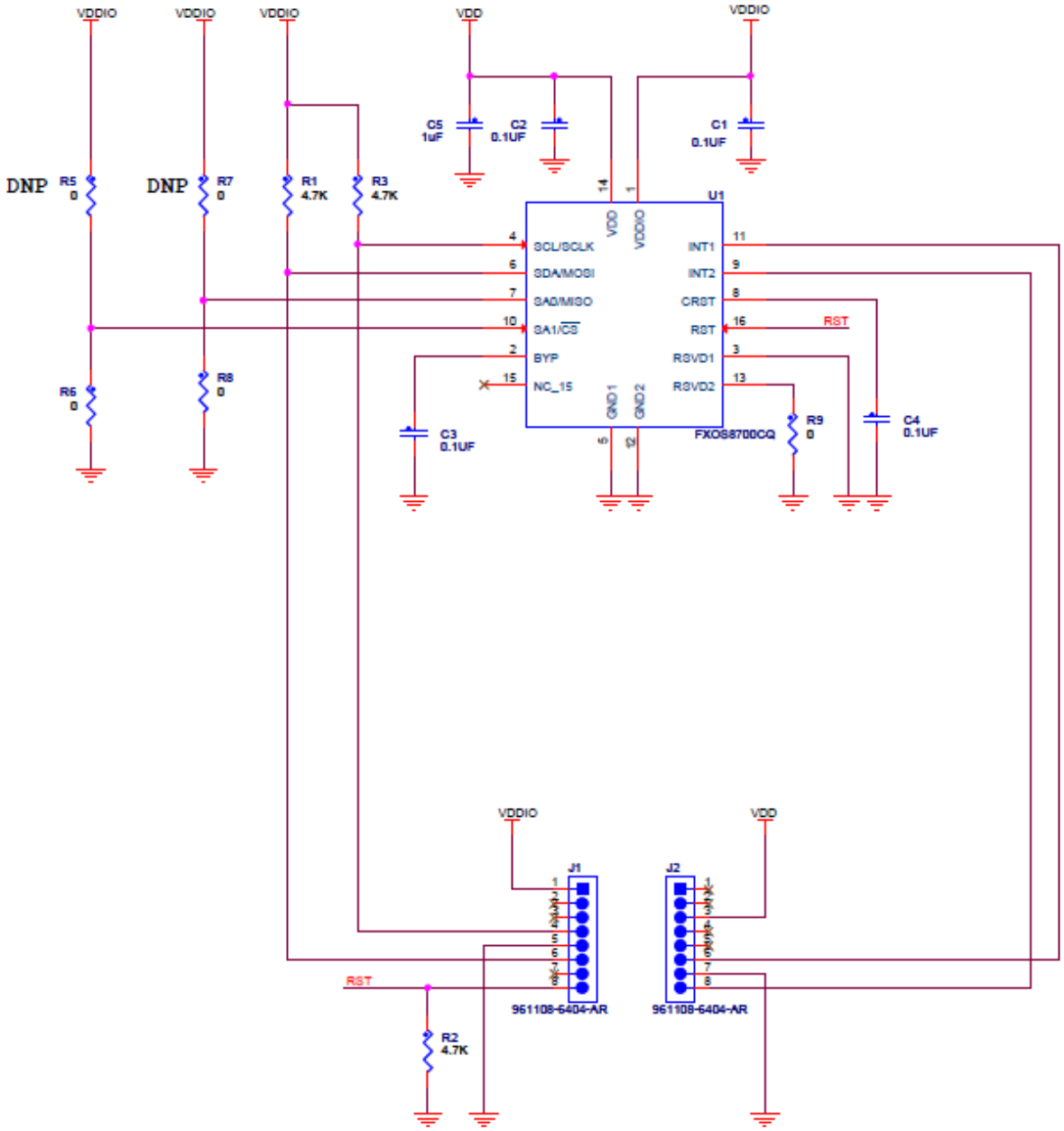


FXOS8700CQ DIP16 Board Description

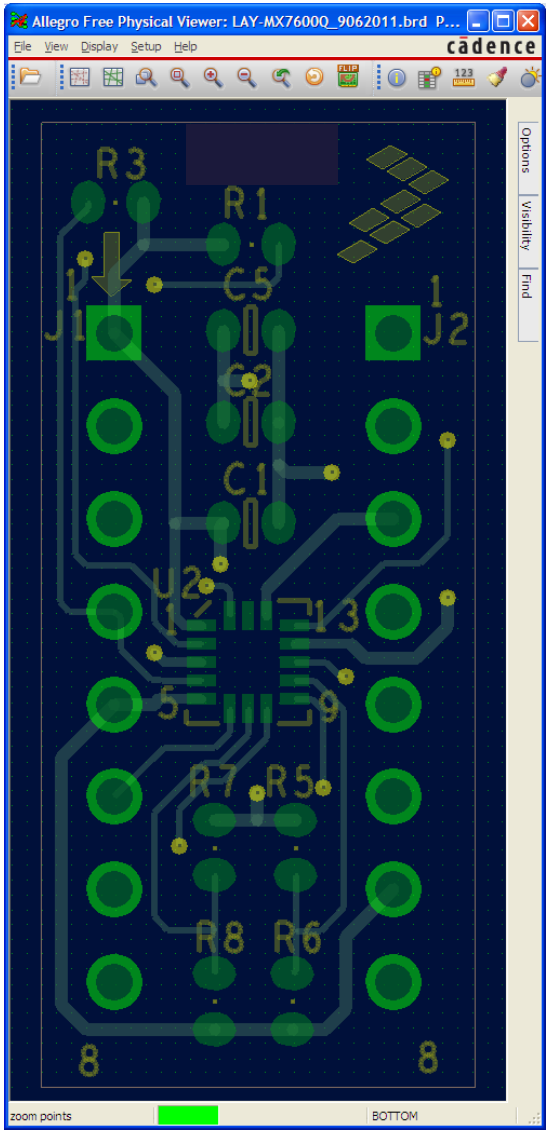
FXOS8700CQ DIP16 Board Schematic

SA0=SA1=0

Device I2C address:
0x4C (pre-production parts)
0x1E (production parts)

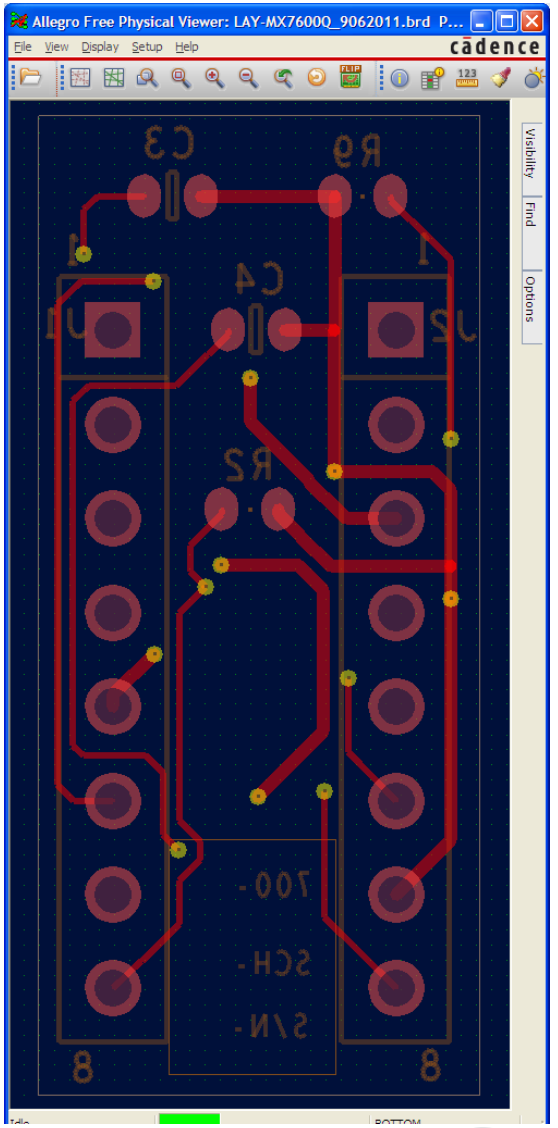


FXOS8700CQ DIP16 Board Layout

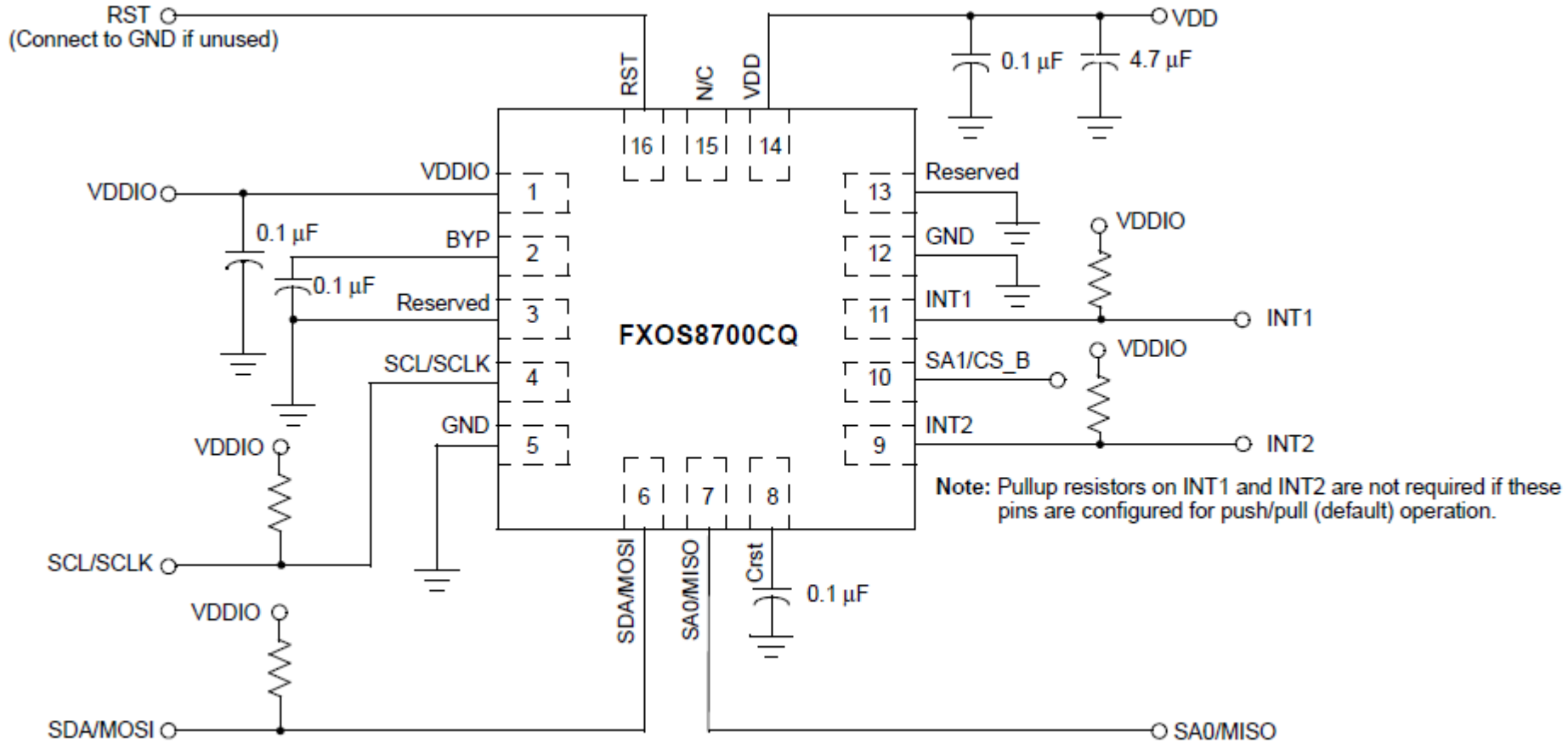


← PCB Top View

PCB Bottom →
(“transparency” view)



FXOS8700CQ typical application (from Datasheet)



Note: Pullup resistors on INT1 and INT2 are not required if these pins are configured for push/pull (default) operation.

Note: Pullup resistors on SCL/SCLK and SDA/MOSI are not required if the device is operated in SPI Interface mode.

FXOS8700CQ DIP16 Board J1 Connector

DIP16 Board Pin Number / Name	FXOS8700CQ Pin Number / Name	Description
J1-1 / VDDIO	1 / VDDIO	Digital IOs Supply Rail 1.62 to 3.6V
J1-4 / SCL	4 / SCL	I²C Slave Data Line Pulled-up to VDDIO by R3
J1-5 / GND	5,12 / GND1,2	Main Ground
J1-6 / SDA	6 / SDA	I²C Slave Data Line Pulled-up to VDDIO by R1
J1-8 / RST	16 / RST	Pulled to GND by R2
J1-2,3,7 / NC	N/A	Unused

FXOS8700CQ DIP16 Board J2 Connector

DIP16 Board Pin Number / Name	FXOS8700CQ Pin Number / Name	Description
J2-3 / VDD	14 / VDD	Main Supply 1.95 to 3.6V
J2-6 / INT1	11 / INT1	First Interrupt Line
J2-7 / GND	5,12 / GND1,2	Main Ground
J2-8 / INT2	9 / INT2	Second Interrupt Line
J2-1,2,4,5 / NC	N/A	Unused

RD4247FXOS8700 Communication Protocol Details and Examples

Bridge Protocol Basic Commands Description

The syntax of commands used to access the sensor registers through the bridge is shown below

- **Device Write Register**
  - Command sent from PC to MCU
    - Byte 0 : SPACE - message header (0x20)
    - Byte 1 : ‘m’ - character command for device Write Reg
    - Byte 2 : Slave I2C address - slave I2C address of the device
    - Byte 3 : Size - number of consecutive bytes to write
    - Byte 4 : Register Addr - device first register address to write into
    - Byte 5 : Data x size - data to be written (size x bytes)
    - Byte n : CR - terminating character (0x0D)
  - Answer sent from MCU to PC
    - Byte 0 : SPACE - message header (0x20)
    - Byte 1 : ‘M’ - character answer for device Write Reg
    - Byte 2 : CR - terminating character (0x0D)

- **Device Read Register**
  - Command sent from PC to MCU
    - Byte 0 : SPACE - message header (0x20)
    - Byte 1 : ‘l’ (lower case L) - character command for device Read Reg
    - Byte 2 : Slave I2C address - slave I2C address of the device
    - Byte 3 : Size - number of consecutive bytes to read. Maximum 20.
    - Byte 4 : Register Addr - device first register address to read from
    - Byte 5 : CR - terminating character (0x0D)
  - Answer sent from MCU to PC
    - Byte 0 : SPACE - message header (0x20)
    - Byte 1 : ‘L’ - character answer for device Read Reg
    - Byte 2: Size - number of bytes read (should be the same as in command)
    - Byte 3: Register Addr - device first register address read (should be the same as in command)
    - Byte 4: Data x size - consecutive data read (size x bytes)
    - Byte n: CR - terminating character (0x0D)

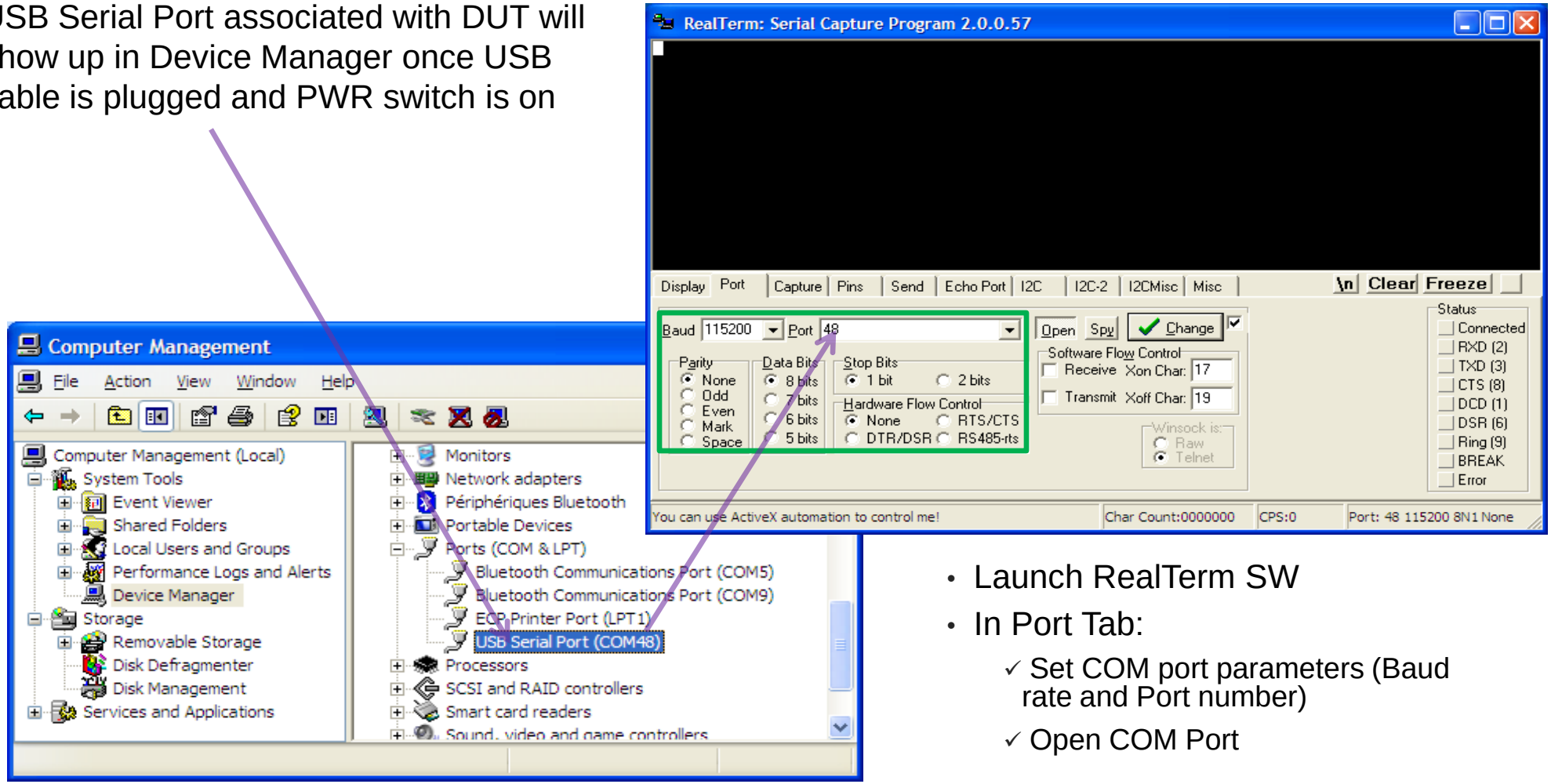
Bridge Protocol Basic Commands Description

This slide provides the syntax of some useful commands allowing to check the serial link, query Bridge FW revision, initiate and close data streaming

- **Handshake**
  - Command sent from PC to MCU
    - Byte 0 : SPACE - message header (0x20)
    - Byte 1 : ‘h’ - character command for Handshake
    - Byte n : CR - terminating character (0x0D)
  - Answer sent from MCU to PC
    - Byte 0 : SPACE - message header (0x20)
    - Byte 1 : ‘H’ - character answer for Handshake
    - Byte 2 : CR - terminating character (0x0D)
- **Bridge FW information**
  - Command sent from PC to MCU
    - Byte 0 : SPACE - message header (0x20)
    - Byte 1 : ‘i’ (lower case I) - character command for bridge FW information
    - Byte 2 : CR - terminating character (0x0D)
  - Answer sent from MCU to PC
    - Byte 0 : SPACE - message header (0x20)
    - Byte 1 : ‘I’ (uppercase I) - character answer for bridge FW information
    - Byte 2-9 - bridge FW information details
    - Byte 10 : CR - terminating character (0x0D)
- **Start/Stop 6DOF data streaming**
  - Command sent from PC to MCU
    - Byte 0 : SPACE - message header (0x20)
    - Byte 1 : ‘d’ (lower case D) - character command for streaming 6-axis device Data
    - Byte 2 : 0x - streaming flag (01 to Start, 00 to Stop)
    - Byte 3 : CR - terminating character (0x0D)
  - Answer sent from MCU to PC (See examples in next slides)

Configure and Open Serial Com Port

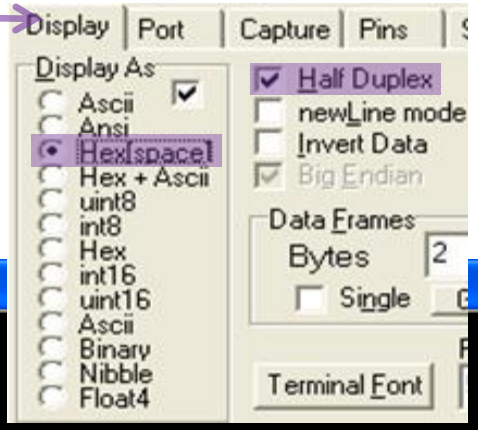
USB Serial Port associated with DUT will show up in Device Manager once USB cable is plugged and PWR switch is on



- Launch RealTerm SW
- In Port Tab:
  - ✓ Set COM port parameters (Baud rate and Port number)
  - ✓ Open COM Port

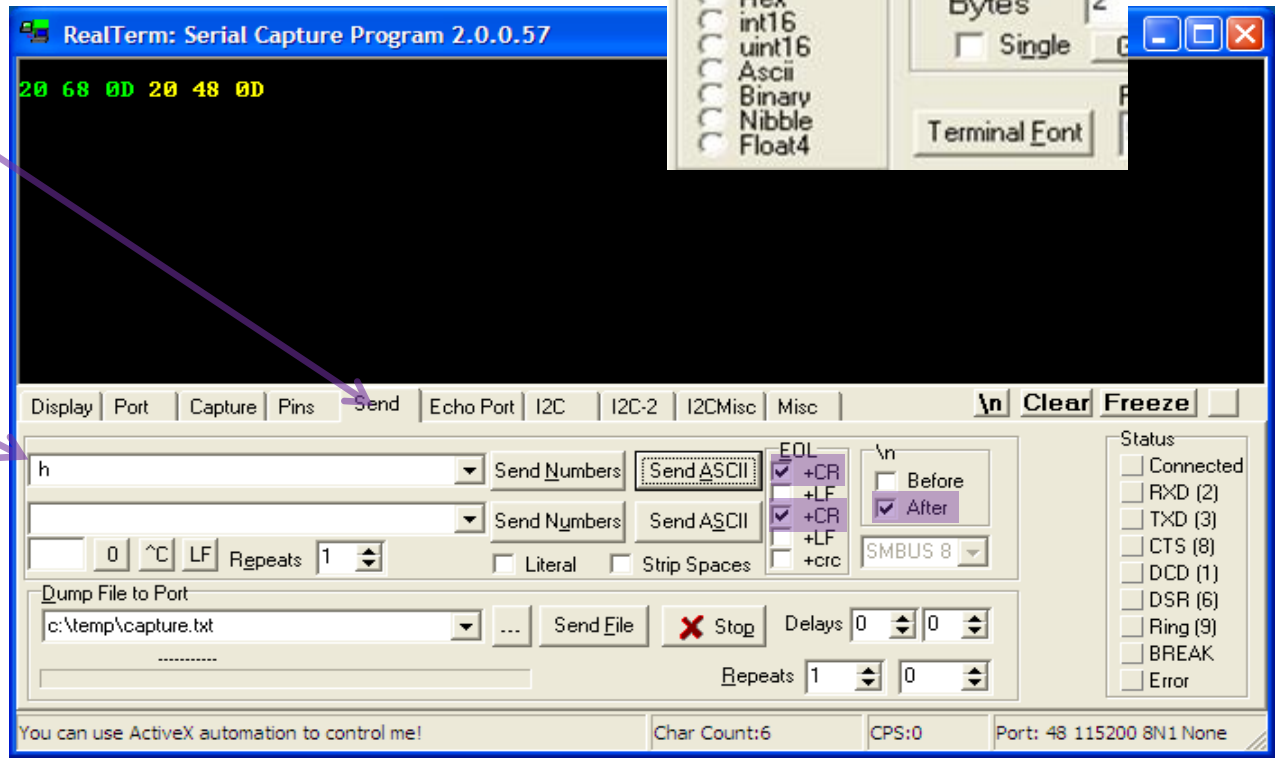
Check Serial Port Communication (Handshake)

- Suggested RealTerm Display tab settings:
  - tick Half Duplex to show outgoing characters in green
  - Display As Hex[space] to see characters hexadecimal code



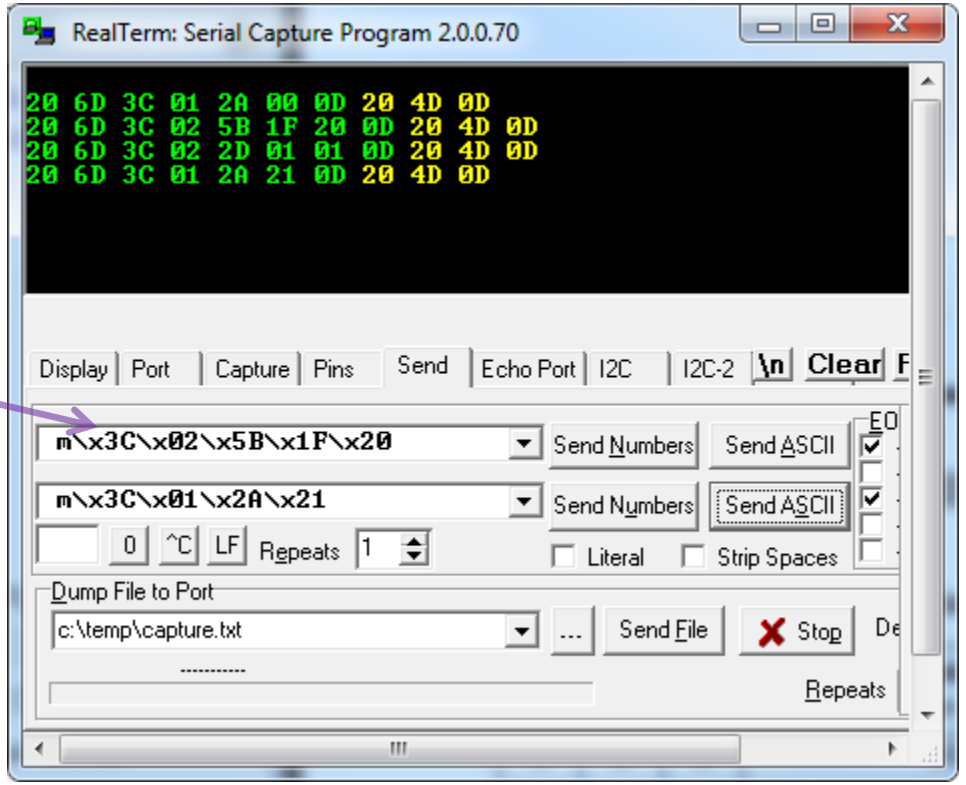
- In RealTerm Send tab:
  - tick EOL +CR to automatically add a carriage return to sent data
  - tick \n After to automatically change to a new line
  - **type "<space>h" in field**

    (you can use backslash with Hexadecimal code "\xHH" to send any value, i.e. **\x20\x68** sends the same string)
  - Hit **Send ASCII** button (<CR> is automatically added as EOL)
- The DUT will echo "<space>H<CR>" as confirmed by the yellow codes "20 48 0D"



DUT Initialization for Data Streaming

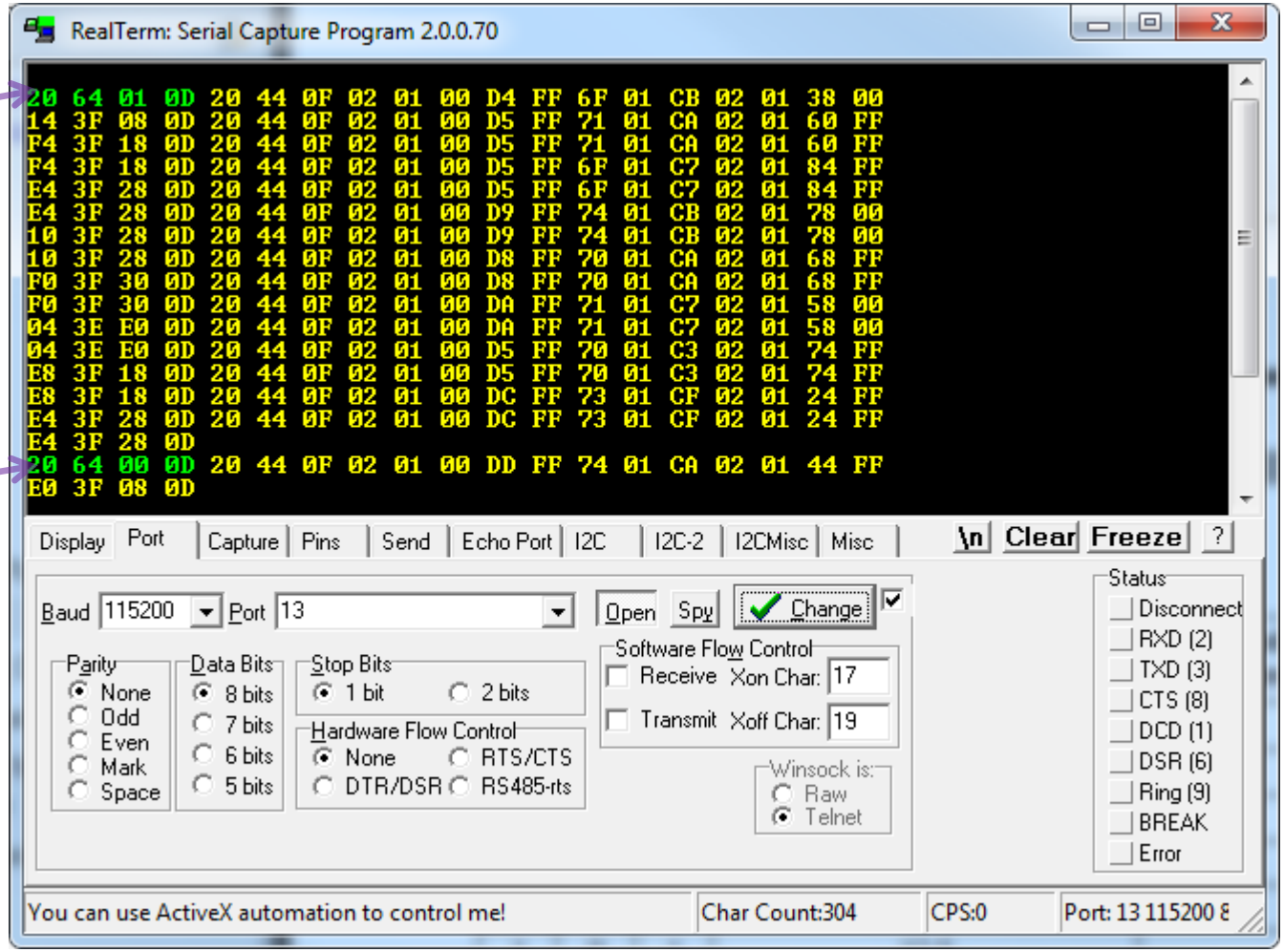
- In order to prepare the device for data streaming, a specific Configuration Sequence must be sent to the DUT after power-up:
  - This commands/answers sequence is reflected in the RealTerm log window below



- Use backslash with Hexadecimal code “\xHH” to send any value. The string associated to 2nd command is shown here
- DUT will acknowledge each “Register Write” command by issuing “<space>M<CR>” answer i.e. “20 4D 0D”

Data Streaming Start/Stop Commands

- Start data Streaming
  - This command will trigger the transfer of raw measured data as19 bytes data packets transmitted continuously at 25Hz rate
- Stop data Streaming
  - Streaming will stop after ongoing packet transfer is complete

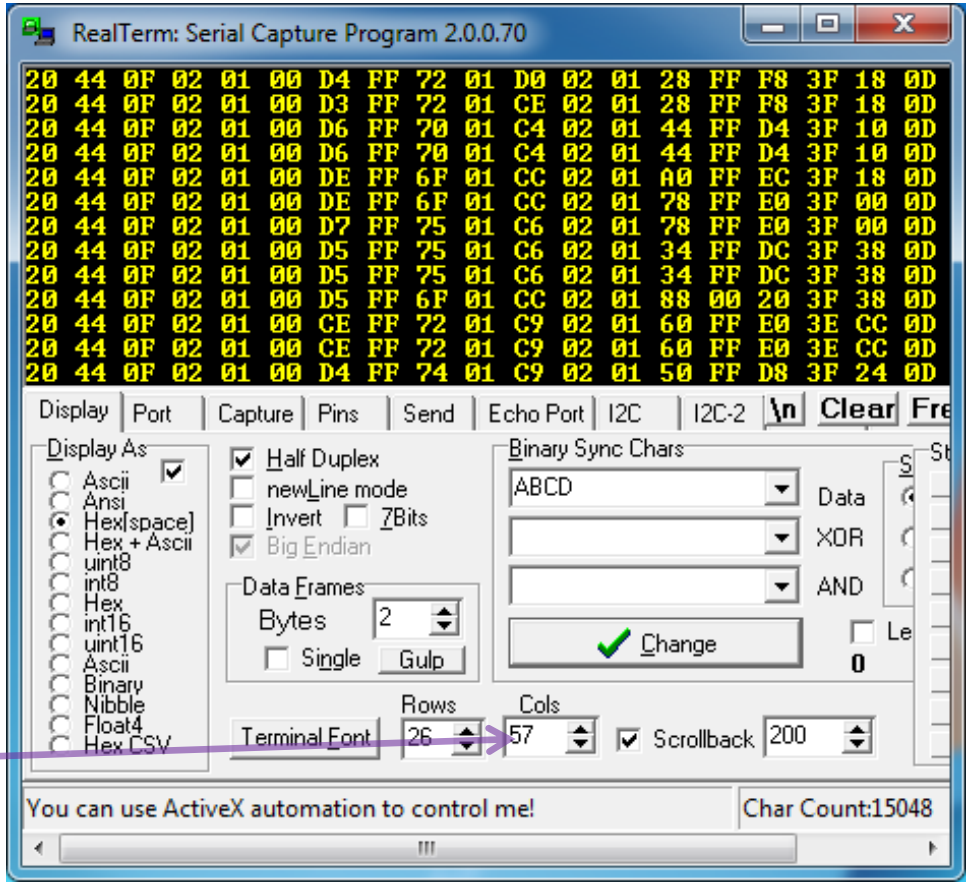


6DOF Data Packet Format used for Data streaming

20 44 0F 02 01 00 D5 FF 75 01 C6 02 01 34 FF DC 3F 38 0D

Header - 1 - Magnetometer X,Y,Z data - 2 - Accelerometer X,Y,Z data - <CR>

- A data packet has a fixed length of 19 bytes. Magnetometer data are transmitted 1st, followed by Accelerometer data
- the 2 sensors X,Y,Z data are coded as 2's complement 16bits signed integers (left justified if sensor resolution is less than 16 bits)
- last char of data packet (i.e. <CR>) shall not be used as sync byte as it could also be found in sensors data fields. Header is more appropriate as sync pattern. Anyway as packet size and format is invariant, a sync pattern is even not necessary
- Trick: In RealTerm Display tab, select 57 as Columns number to align data packets

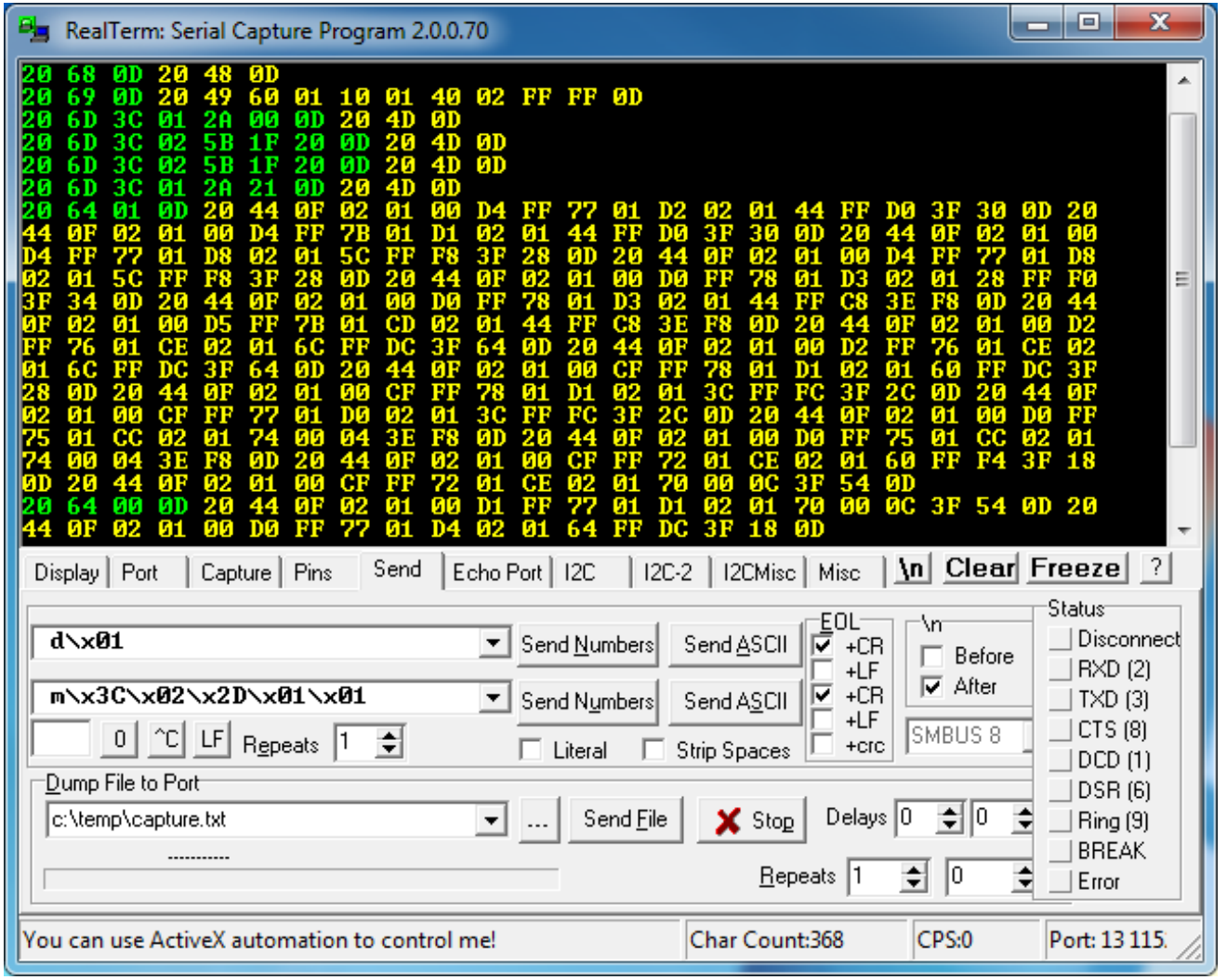


Summary of Communication Sequence for Data Streaming

Step #	Command	Answer	Purpose
1	20 68 0D	20 48 0D	Handshake (**to verify if communication is ok**)
2	20 69 0D	20 49 60 01 10 01 40 02 FF FF 0D	Get Bridge Rev Info (**optional**)
3	20 6D **3C** 01 2A 00 0D	20 4D 0D	I2C Addr.= x**3C** (7-bits Addr.= x1E) Write 00 in Register 2A to set device in Standby Mode
4	20 6D 3C 02 5B 1F 20 0D	20 4D 0D	Select Max OSR and Hybrid Mode with Auto-Increment (Write 1F and 20 in Registers 5B & 5C)
5	20 6D 3C 02 2D 01 01 0D	20 4D 0D	Configure Data-ready interrupt to INT1 pin (Registers 2D & 2E)
6	20 6D 3C 01 2A 21 0D	20 4D 0D	Select active mode with Output data rate of 25Hz (Hybrid Mode)
7	20 64 01 0D	20 44 0F 02 01 FC 70 FE 0E 04 85 02 0D 44 E6 1C 32 F0 0D 20 44 0F 02 01 FC 70 FE 0E 04 85 02 0D 44 E6 1C 32 30 0D	Start data streaming
8	20 64 00 0D	No specific answer	Stop data streaming

Complete Communication Log for Data Streaming

- Note that during data streaming, the bridge MCU is collecting the 6-axis measurement from FXOS8700 device according to INT1 interrupt signal (data ready event). Consequently the communication on the I2C interface is kept minimal



Communication Sequence Summary

- Note that Magnetometer raw data can have significant offset that needs to be calibrated (Hard Iron)

Command		Answer from DUT	
20 68 0D	handshake (optional)	20 48 0D	
20 6D 3C 01 2A 00 0D	Standby Mode	20 4D 0D	
20 6D 3C 02 5B 1F 20 0D	Initalize DUT 1/2	20 4D 0D	
20 6D 3C 02 2D 01 01 0D	Initalize DUT 2/2	20 4D 0D	
20 6D 3C 01 2A 21 0D	Active Mode	20 4D 0D	
20 64 01 0D	Start data Streaming	20 44 0F 02 01 00 D5 FF 75 01 C6 02 01 34 FF DC 3F 38 0D	
		Mag. X Y Z data + Accel. X Y Z data	
	19 bytes data packet received continuously at 25Hz rate		
20 64 00 0D	Stop data Streaming		
data	**decimal**	**value**	**unit**
Mag X	213	21.30	µT (scale= 0.1µT/LSB)
Mag Y	-139	-13.90	µT
Mag Z	454	45.40	µT
Accel X	308	18.80	mg (scale= 2g/2^15/LSB)
Accel Y	-36	-2.20	mg
Accel Z	16184	987.79	mg

Conclusion

- Freescale 6-axis Sensor development kit (RD4247FXOS8700) can easily be configured by sending a few commands via a serial terminal Software
- Once Data streaming is configured and triggered, DUT continuously provides 6DOF data packets to the Host PC through the COM port. Extraction of the 6 measurement values out of a data packet is quite straightforward
- Instructions provided in the document can be applied to more sophisticated programs that are able to use a serial port object such as Matlab script or Microsoft Visual Studio project. This enables automatic collection of the 6DOF raw measurement as well as real time data processing and plotting


freescale™