



μTasker Document

- **USB Demo**

Table of Contents

| | |
|--|----|
| 1. Introduction..... | 3 |
| 2. Configuring the project..... | 3 |
| 3. Test driving the project – enumeration and installation of USB driver | 4 |
| 4. Identifying and changing the virtual COM port via the device manager | 5 |
| 5. Test driving the project – checking the menu and debug outputs..... | 5 |
| 6. Test driving the project – Controlling the Baud Rate | 7 |
| 7. Test driving the project – USB<->RS232 bridge..... | 8 |
| 8. Test driving the project – firmware upgrade via USB | 11 |
| 9. Working with USB and the μTasker simulator..... | 13 |
| 10. Conclusion..... | 17 |

1. Introduction

The μTasker project includes USB device support and a demo showing useful CDC (Communication Device Class) operation. This allows any board with a USB device interface, supported by the μTasker project, to be connected to a PC and controlled from a standard terminal emulator (using virtual COM).

The demo shows a menu driven terminal interface via USB, which can also be used via UART or TELNET (as long as the board also has Ethernet support and the TELNET support is activated in the project). Via the menu, the USB connection can be commanded to operate as an RS232<-> converter (requiring also a UART to be enabled on the board) or for firmware uploads.

This document gives a step-by-step guide for setting up and testing the project on a Windows Vista PC (other Windows operating systems will also work but some screen shots may be a little different). The explanation is generally valid for any used hardware platform which the project supports (the USB user interface is essentially not hardware specific), but any details which may be specific to a particular device will be mentioned as a side note.

As well as demonstrating the use of USB on the target board, the document explains how the μTasker simulator can be used to test the device enumeration and also data exchange via USB, which makes development practical and efficient in avoiding the need for a target and possibly a USB analyser.

2. Configuring the project

To fully test the demo project, a target hardware board is required and the μTasker project should be built with a supported compiler project using the appropriate configuration settings. The generated object code should be programmed to the board and then tests can begin. [*Software demo packs include pre-built objects for certain targets which can be used without requiring the project to be compiled*]

If you need help in performing the required steps, please consult the μTasker tutorial for your particular processor. Also ensure that you have the latest μTasker service pack installed so that you have the latest features available.

For the USB firmware feature to operate completely the “*Bare-minimum*” project build should be used and the “*Bare-minimum*” boot loader also linked to the object file. This is detailed in the appropriate boot-loader documentation, but the standard build can still be used if this operation is not required.

- Check that you have the file
`\Applications\uTaskerV1.3\usb_application.c` in your project since this coordinates the USB interface.
- In the configuration file `config.h` ensure that the project is set up for your particular board and that the USB support is enabled:

```
#define USB_INTERFACE // enable USB driver interface
```
- Since the serial interface is used in the demo also ensure that serial support is enabled (also in `config.h`):

```
#define SERIAL_INTERFACE // enable serial interface driver
```
- In `app_hw_xxxx.h` the UART used can be modified if the default doesn't suit your requirements. For example if you prefer to use UART1 rather than UART0 configure

```
#define DEMO_UART 1
```

Rebuild the project with your chosen compiler and load it to your board.

The RUN LED should blink with a speed of 2.5Hz, indicating that the software is basically operating. If you connect the debug UART to a PC you should be able to enter the demo menu by pressing the ENTER key when the PC's COM port is opened using a terminal emulator (19'600 Baud, 8 bit, no parity). For the following discussion the terminal emulator *Tera Term Pro* is used – this is a good freeware program which operates well with all Windows operating systems. It can be downloaded from the μTasker web site at <http://www.uTasker.com/software/freeware/teratermpro.zip>

3. Test driving the project – enumeration and installation of USB driver

Once the project has been loaded to the board and the board is running you simply need to connect the USB connector to a free USB port on your PC or a USB hub attached to the PC. The USB host (in the PC) will detect that the device has been connected and start the USB *enumeration process* (there is generally a beep from the PC when a device is connected).

The first time that you connect it will need to configure the operation for the defined USB driver. It will also change some settings in the registry on the PC so that on subsequent connections the device can be used immediately. The PC will therefore ask for the driver “*Driver for ‘MyProd’¹ needs to be installed*”. This means that it is looking for an `.INF` file on the PC containing details about what it should do. If the `.INF` file happens to be found it will complete without any intervention, otherwise you will have to give a helping hand and specify where the file is located. First try with “`Search for driver software and install (recommended)`”.

¹Note that ‘MyProd’ is the name of the product that the demo project reports during enumeration. This can be change in `usb_application.c` by changing the UNICODE string `product_str[]`.

Assuming the search is not successful, use the manual option and enter the position where the information can be found. This is in the μTasker project directory at `\Applications\uTaskerV1.3\USB`. It is called `uTaskerVirtualCOM.inf` and contains a reference to the correct *vendor ID* and *product ID* (as detailed during the enumeration process) so that the PC can identify that this is the file that it is looking for. There may be other VirtualCOM `.INF` files in the director and if there is one specially defined for your processor type use it instead since it will contain specific vendor and product IDs from the processor manufacturer specifically for the μTasker project.

In the file there is a reference to the USB driver that is used `usbser.sys`. Since this is already installed on a Windows PC there should not be any more questions. However it will want you to confirm that the ‘non-certified’ driver may be installed... there is no risk involved in confirming this so that the installation can be completed.

Once the installation has terminated Windows will inform that the “`uTasker driven device`” is ready for use. It will also have attached it to a virtual COM port (eg. “`uTasker driven device (COM15)`”). This COM port can however be modified if required, as explained in the following section.

Note also that the virtual COM port which it is attached to may change when the USB cable is connected to a different USB port on the PC. Sometimes the installation procedure is also repeated for different USB ports, but in this case the driver will be automatically found and the installation process is much faster. Often you will not really be sure which virtual COM port the driver is using and so it is advisable to make yourself familiar with the device manager as follows:

4. Identifying and changing the virtual COM port via the device manager

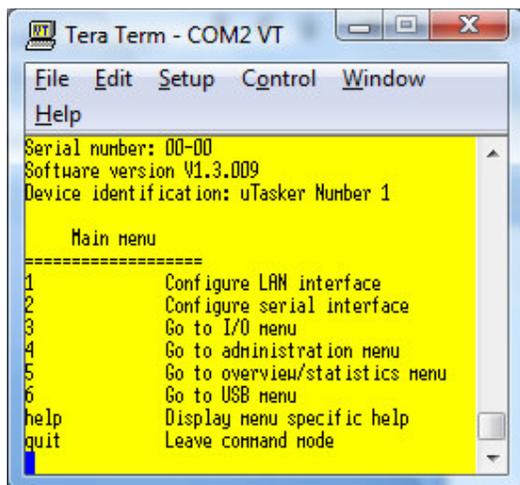
- On the Windows PC start the “Control Panel” and double click on the “Device Manager” (WinXP requires this to first be located in “System → Hardware”).
- A list of devices will be displayed including things like Mice, Monitor, Hard Disk, etc. whereby we are interested in “Connections (LPT & COM)”.
- Expand this entry to see the COM ports and virtual COM ports on the PC. As long as the board is connected the entry “*uTasker driven device (COM15)*” – or other COM number – will be seen. Disconnecting the device causes it to be removed and reconnecting will cause it to be displayed again.
- By double clicking on “*uTasker driven device (COM15)*” its configuration can be viewed and also modified. For example, if the COM port number needs to be changed switch to the “*Connections Settings*” register and open the extended options. Here a different COM number can be selected, which will subsequently be used each time the device is connected to the particular USB port. Note that *Tera Term Pro* can use COM1 to COM16² and so higher numbers need to be adjusted here.

² Note that Tera Term Pro has COM1..COM4 enabled per default but this can be extended by editing the entry `MaxComPort=16` in `TERATERM.INI` in its program directory.

5. Test driving the project – checking the menu and debug outputs

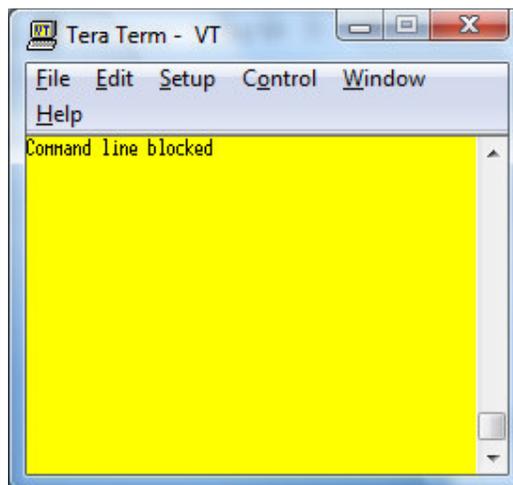
With the board connected to the PC, and its virtual COM displaying in the device manager it is time to start using it to communicate. For this test a second COM port is also connected to the board via the UART debug connector, whereby each of the COM ports (COM2 is assumed for the UART connection and COM 15 is assumed for the USB connection) has its own *Tera Term Pro* terminal emulator session running.

UART session on COM2



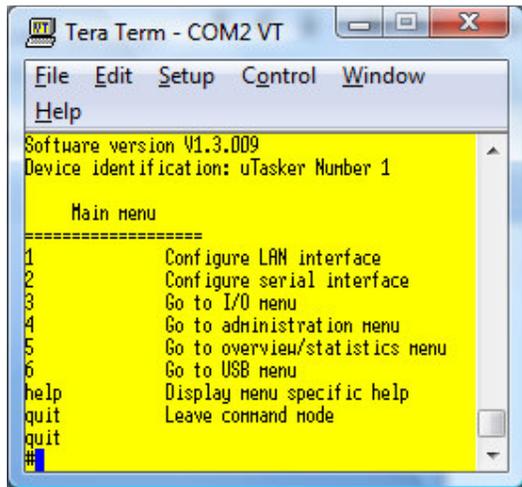
The main menu is displayed when the ENTER key is pressed. All debug outputs are now diverted to the UART port.

USB session on COM 15

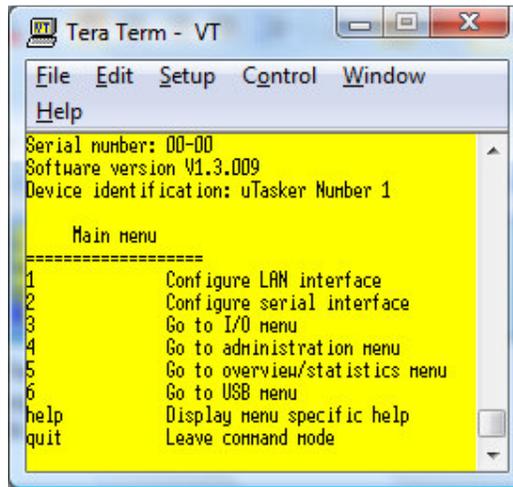


When the ENTER key is pressed at the USB session the message “Command line blocked” informs that the command interface cannot be

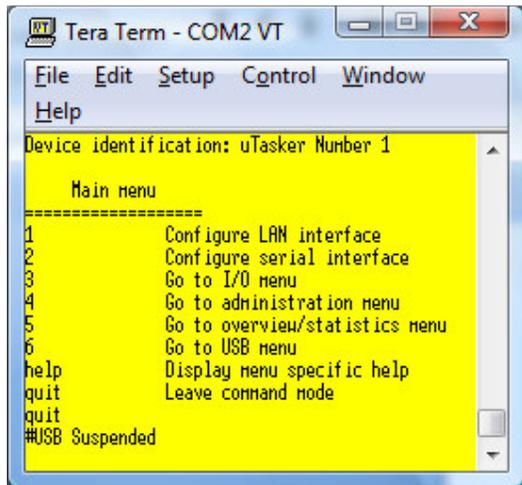
accessed since it is already in use by the UART



After entering quit, the command menu session is terminated at the UART interface...

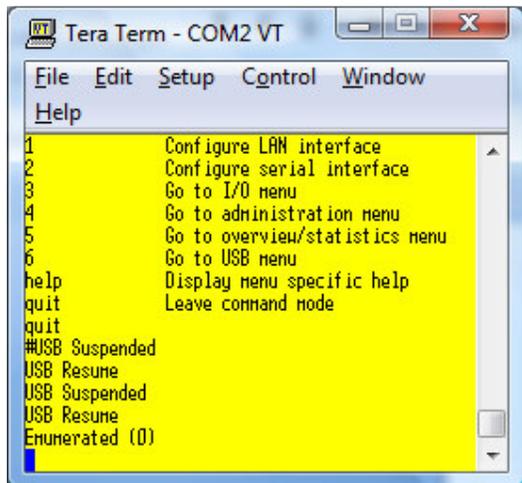


.. and can then be accessed at the USB interface (again by pressing the ENTER key).



When the USB cable is removed, the Virtual COM port is disconnected and the debug output returns to the UART interface, which can be seen due to the debug output #USB Suspended.

The USB Tera Term Pro session should be closed when the USB cable has been removed since the Virtual COM port no longer exists. It should not be started again until the USB connection exists again.

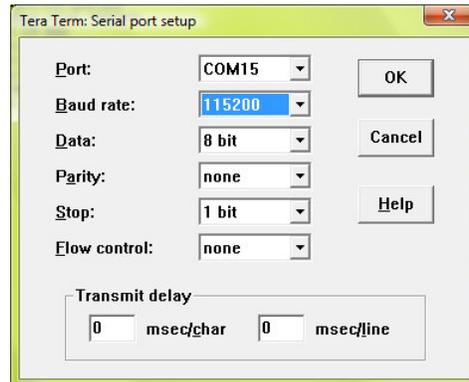


The USB cable has been reconnected (a beep from the PC is normally also heard). The debug output also display the state changes involved, completing with the enumeration (where 0 indicates that the configuration number 0 was activated).

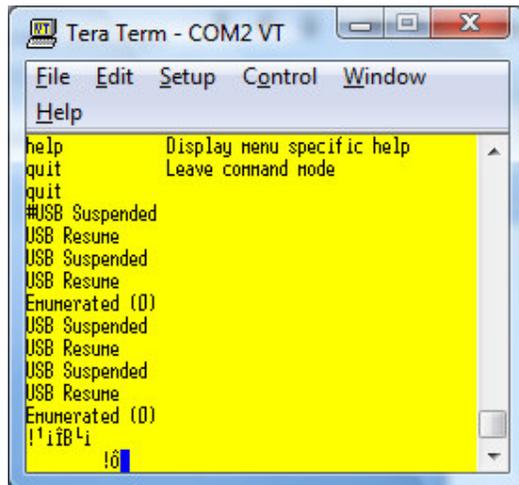
6. Test driving the project – Controlling the Baud Rate

With the USB connection re-established Tera Term Pro can be started again and attached to the Virtual COM. Although the Virtual COM connection is USB based it still has settings for BAUD, defaulting to 19'600 to be compatible with UART settings.

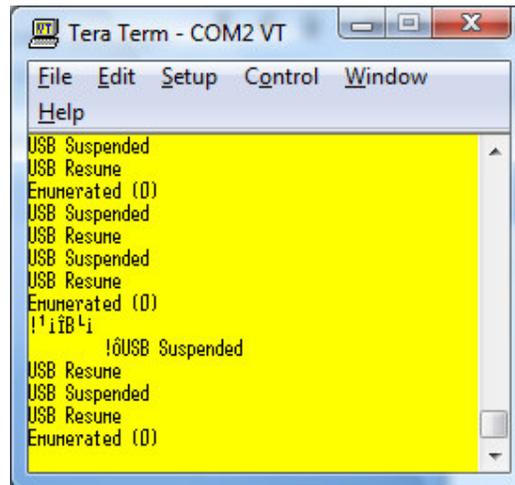
Open the menu `Setup | Serial Port...` in the USB terminal program and change the speed to 115200.



This speed setting doesn't really have relevance for the USB physical connection but the Virtual COM driver informs the application on the board of the new settings and this in turn configures the setting of the UART channel. Therefore the debug UART will now be operating with a speed of 115200 Baud and its *Tera Term Pro* setting also need to be adjusted to suit so that it will display data correctly. To following shows the debug outputs when the USB connection is removed and reconnected again when the speed is not modified:



The last messages are not readable since they are being received at a faster Baud rate than the program is configured to use.



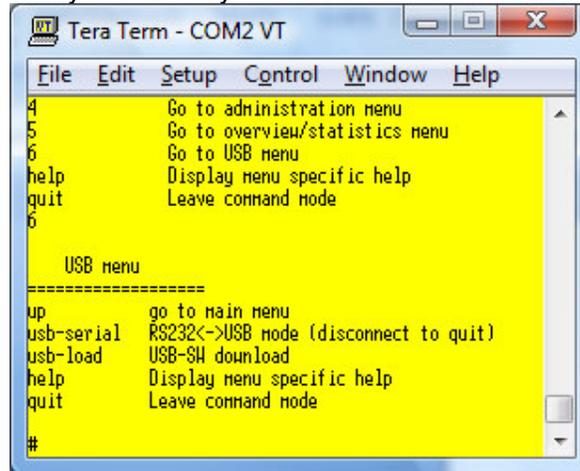
After adjusting the Baud setting in Tera Term Pro to match the new rate, subsequent disconnections and connection is then correct, as is further communication with the command menu

The control of the Baud rate via the Virtual COM setting is not required for the menu mode of operation but it is important when using the board as a USB<->RS232 bridge since the terminal emulator program needs to be able to control setting of the RS232 interface which is acting as a bridge extension to it. This mode is tested in the following section.

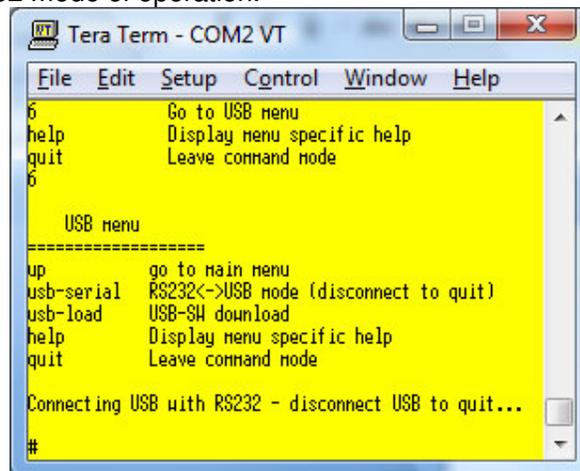
7. Test driving the project – USB<->RS232 bridge

With the USB connection re-established Tera Term Pro can be started again and attached to the Virtual COM. *Note that Tera Term Pro must be closed before the USB connection is established and it will set the UART baud rate to its default setting again when starting.*

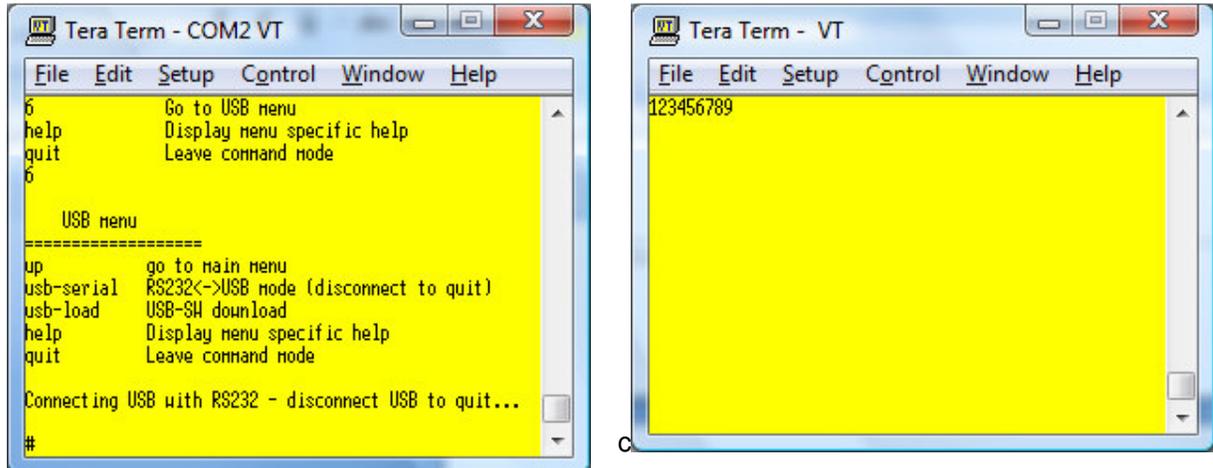
- In either *Tera Term Pro* emulator (UART or USB) the command menu can be started (by pressing the ENTER key).
- Enter 6, followed by ENTER key to enter the USB sub-menu.



- Finally enter the command `usb-serial` (followed by the ENTER key) to enter the USB<->RS232 mode of operation.



As noted when entering this mode, it is exited only by disconnecting the USB cable. Once in this mode all data received from the USB link on the board will be transmitted to the RS232 output and all data received on the RS232 input is transferred to the USB link. This means that the target board is now performing the function of a USB to RS232 adapter.



Typing in the numbers 1 2 3 4 5 6 7 8 9 in the UART program window...

... causes them to appear in the USB program windows. The reverse is also true – typing into the USB program windows causes the characters to appear in the UART program window.

To test file transfers, including file transfers in both directions at the same time, *Tera Term Pro* supports the function `File | File Send...`. By sending large ASCII files they will appear in the opposite windows. (Note that may be useful to configure `Setup | Terminal...` so that new lines are sent and received as CR + LF at one of the terminal emulators so that the received data doesn't remain on one single line in the output window. SREC files are useful for this test.

Another file transfer method supported by *Tera Term Pro* can be found under `File | Transfer..` where well known file transfer protocols like XMODEM can be used. This has the added advantages that it can send binary files and any data errors during the transfer will be recognised and counted.

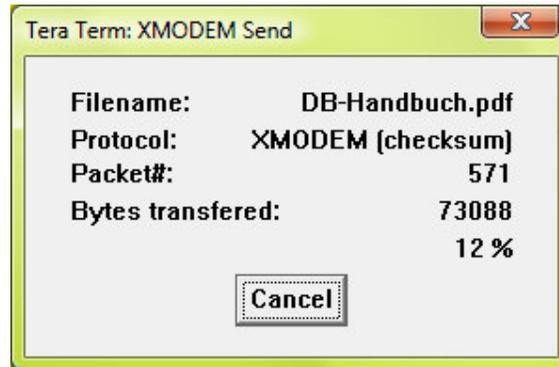
To test an XMODEM transfer using *Tera Term Pro* the following procedure can be followed:

- Ensure that the RS232 interface is not configured for XON/XOFF operation. Since binary data will be sent the XON/XOFF control characters will probably occur in the normal data stream and so would cause problems. To disable XON/XOFF use the command menu's sub-menu 2 (Configure serial interface) and the command "set_flow NONE". This setting can also be saved if required by subsequently issuing the command "save"
- Chose the transfer direction and in the transmitting terminal window select `File | Transfer | XMODEM | Send...` and select a file to be transmitted but don't confirm it just yet*.
- In the receive terminal windows select `File | Transfer | XMODEM | Receive...` and give a name for the file to be received. Confirm this and the terminal will wait for the file reception.

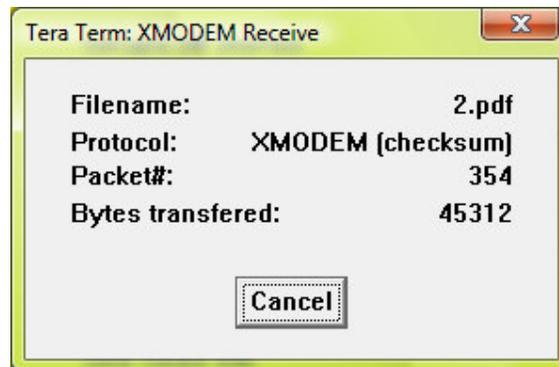
When configuring the files for transmission and reception ensure that the same options are used for transmission and reception (checksum, CRC etc.).

- In the transmitting terminal window confirm the file to be sent and the transmission will begin.

**Note that it is possible to immediately confirm the transmission file, rather than waiting for the receiver to be set up. In this case the transmitter will be sending probes to the receiver and this waiting for the receiver to respond in order to start. However there is a limit to the time that the transmitter waits before cancelling the transfer.*



The *Tera Term Pro* XMODEM transmission dialog showing the transfer progress during transmission



The *Tera Term Pro* XMODEM reception dialog showing the reception during the transfer. Note that the receiver doesn't know the length of the file being received until the transfer terminates

Since the transfer protocol includes handshaking it can also tolerate transmission errors, which can be tested by temporarily disconnecting the RS232 cable. This will result in an unsuccessful block transmission and, after a timeout, will be repeated and the transfer will be continued.

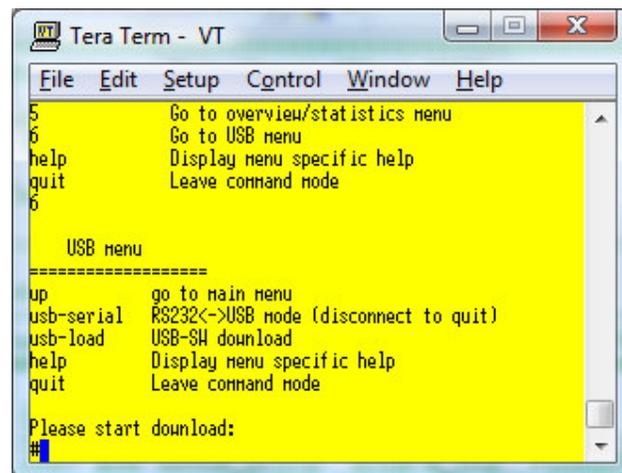
8. Test driving the project – firmware upgrade via USB

In order for this test to operate completely, the loaded project should be built with the “*Bare-Minimum*” boot loader target option and the boot loader should also be installed. After the file has been transferred it will then be used as replacement for the original code and started. If the boot loader is not installed the upload transfer will still take place but the new code will not actually be executed.

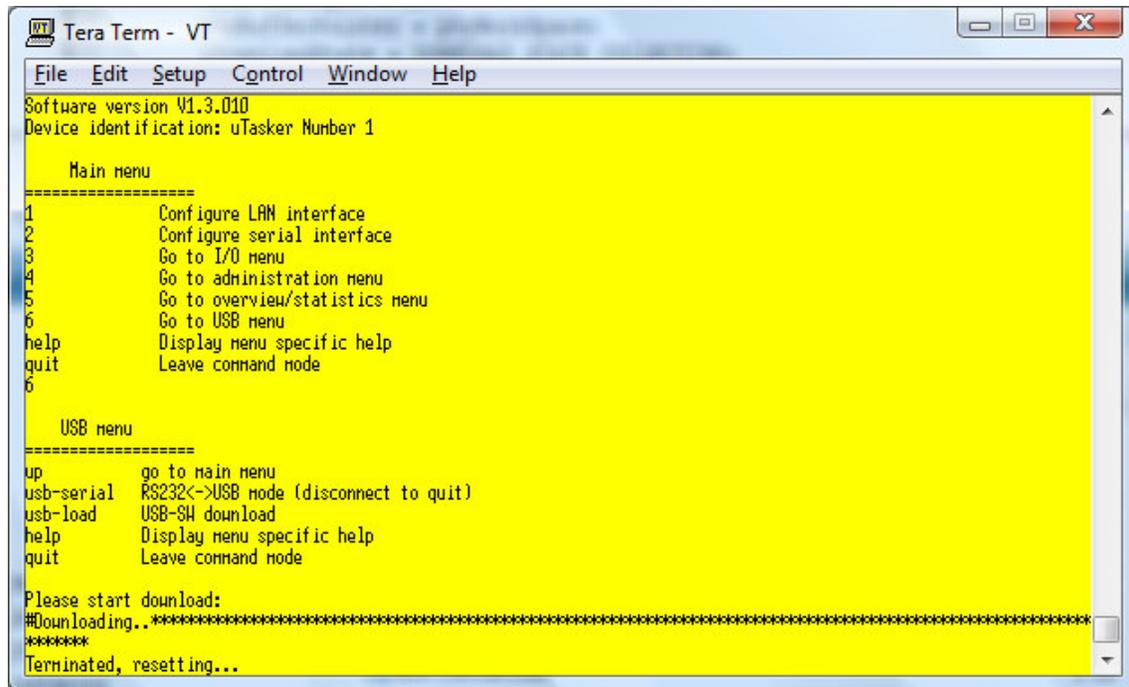
A simple verification that the new file is really being executed after the upload is to check the version number displayed when connecting to the command menu as depicted in previous sections. As long as the new software has a different software revision number it is simple to verify the success of the operation; the software version can be changed in `application.h`:

```
#define SOFTWARE_VERSION "V1.3.009"
```

To use the firmware upload feature, connect to the command menu via the USB terminal and enter the USB sub-menu. In the USB sub-menu execute the command `usb-load`



In the *Tera Term Pro* menu `File | Send File...` ensure that the option “*Binary*” is active and select the new software image to be loaded. This must be in the correct format otherwise it will not be accepted by the software on the target board and an error message will appear. By using the µTasker converter `uTaskerConvert.exe` in conjunction with the original binary image the correct format can be generated – see the corresponding µTasker document for full details. The following screen shot shows a successful download taking place and subsequently terminating:



```
Tera Term - VT
File Edit Setup Control Window Help
Software version V1.3.010
Device identification: uTasker Number 1

Main menu
-----
1      Configure LAN interface
2      Configure serial interface
3      Go to I/O menu
4      Go to administration menu
5      Go to overview/statistics menu
6      Go to USB menu
help   Display menu specific help
quit   Leave command node
6

USB menu
-----
up      go to main menu
usb-serial RS232<->USB mode (disconnect to quit)
usb-load USB-SH download
help   Display menu specific help
quit   Leave command node

Please start download:
#Downloading..*****
*****
Terminated, resetting...
```

After each 512byte block of code is written to FLASH a star '*' is displayed. Once the end of the file has been received and the last code data has been successfully saved to FLASH the text "Terminated, resetting..." is displayed.

After the file has been completely received and the code successfully programmed the board will reset and start again with the new code operating. The actual programming process after the reset can take a few seconds (depending on processor type) during which any LED activity on the board will stop. The reset will also cause the USB connection to be temporarily lost and so *Tera Term Pro* needs also to be restarted. *Ensure that it is closed before the USB connection re-enumerates and is opened again after the enumeration has taken place!*

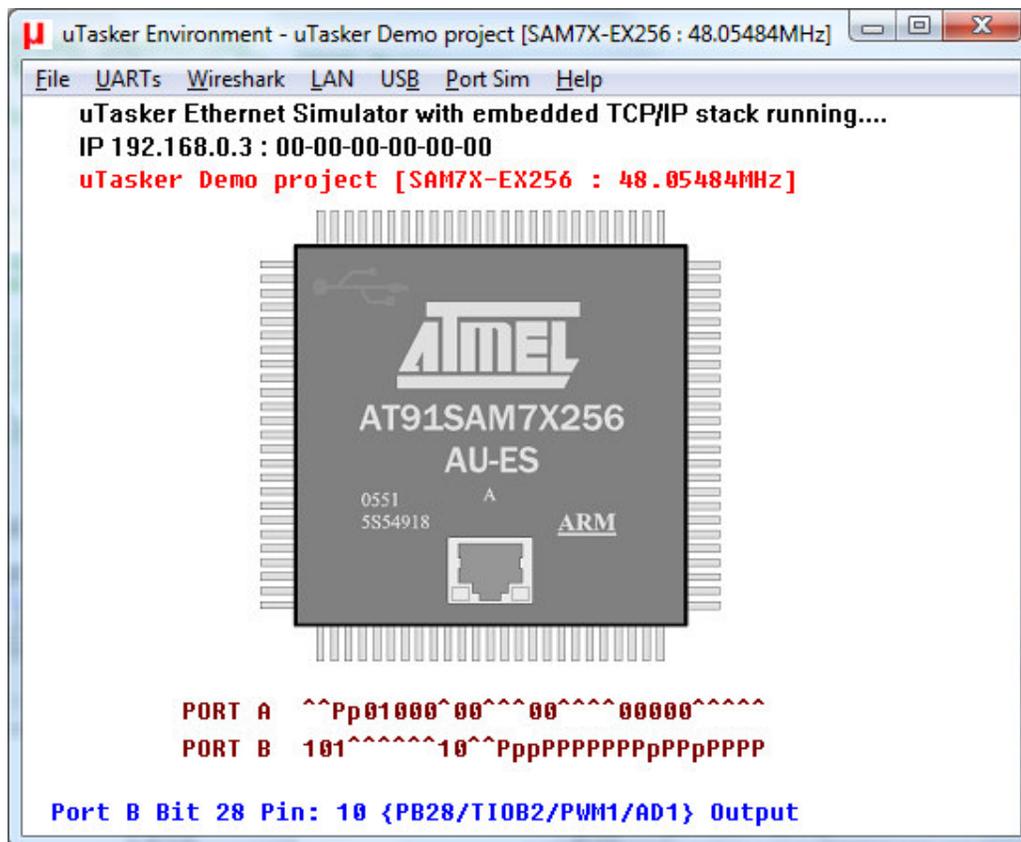
9. Working with USB and the μTasker simulator

One of the main aims of the μTasker project is to make development of embedded software more efficient and easier than when all work needs to be performed on the target. Its simulator allows code to be developed and tested on a PC, where code operation in respect to peripherals and interrupts is also representative, resulting in code which has a high probability of working as required on real hardware after its much more efficient development in the simulator environment.

It is not possible to map a USB device in the PC to operate as an extension of the simulated device (as is possible for UART and Ethernet interfaces), mainly due to the fact that the PC has only HOST hardware. But, the simulator includes an enumeration sequence emulator which verifies the enumeration code and allows the software project to respond to enumeration and suspend sequences. It also includes the ability to play back USB endpoint data sequences which allows efficient development and testing of USB interfaces and associated applications.

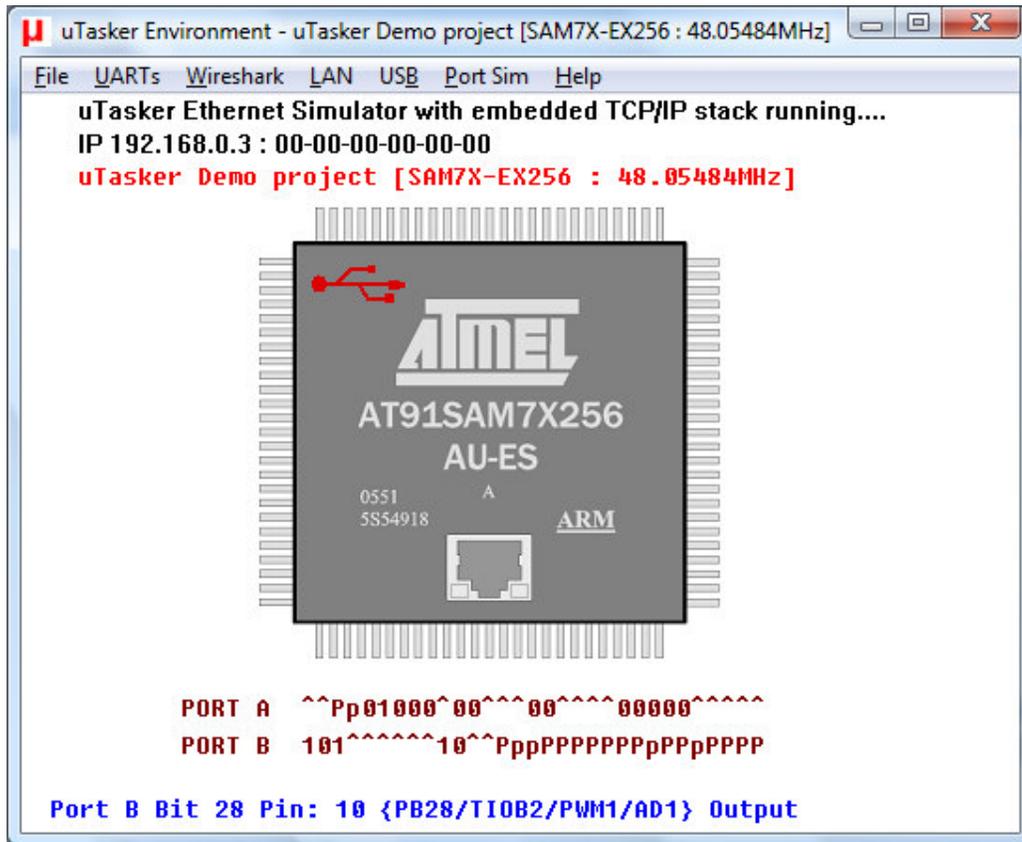
This section describes how the project which has been verified on the target board in previous sections of this document can also be tested with the μTasker simulator and how further development could be performed by extending the method. In fact the original project was initially developed in this environment and only final testing (and some possible optimisations) performed on the target.

- Rather than cross-compile the project for the target it is opened and compiled in the VisualStudio environment. When executed it looks something like this, where the actual target processor will depend on the project target.



Example of the simulator running on the simulated Olimex SAM7X-EX256 board

- In order to work with the USB interface it must be connected to a USB HOST and enumerated. To do this in the simulator use the USB menu and select the command “enumeration”.



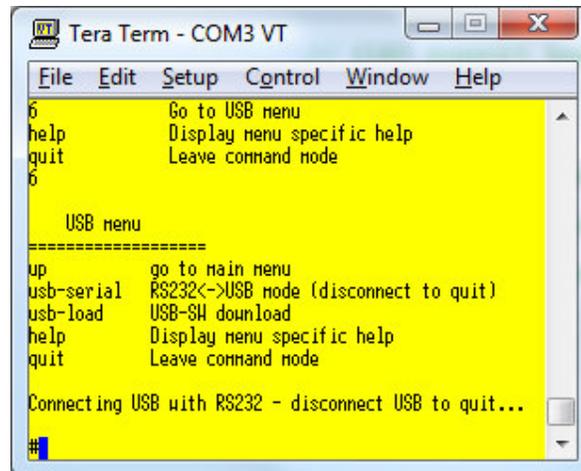
Simulator displaying that the USB interface has completed enumeration (notice that the USB sign is now displayed as red)

During the enumeration sequence a standard sequence of messages was passed to the USB endpoint 0 interface (involving passing them through the interrupt routines involved and verifying the data received from the embedded USB driver). By placing breakpoints in the associated interrupt and driver routines each stage can be analysed if required.

Only once the interface has been successfully enumerated can subsequent communication take place.

- Since the demo also requires a UART interface, we need to connect one to *Tera Term Pro*. Although it is also possible to map to a real COM port on the PC and use a cross-over cable, a virtual solution is preferred using the freeware program *com0com*, which can be obtained from <http://com0com.sourceforge.net/>. This installs a virtual loop back between two virtual COM ports and so avoids the need for any cables. It will be assumed that this has been configured to add virtual COM3 and virtual COM4 to the PC so *Tera Term Pro* can be connected to COM3 and the simulator can map the debug UART to COM4. To map the UART to COM4 simply ensure that the following entry is correct in `app_hw_xxxx.h`:

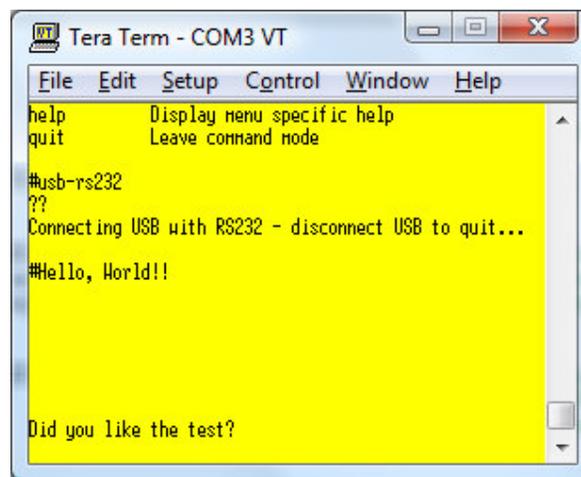
```
#define SERIAL_PORT_0 '4' // map debug UART0 to COM 3
```



Tera Term Pro connected to virtual COM3 (com0com loop back to COM4, where COM4 is mapped to the debug UART of the project running in the µTasker simulator). Display after entering the USB menu and commanding the USB-RS232 mode.

- The USB endpoint 1 is used to receive data so, in order to test reception, USB endpoint 1 data is required. In the simulator open the menu `Port Sim | Open Script` and chose the simulation file `ep1_data_test.sim`.

This will play a predefined sequence of USB endpoint 1 data, which will be received by the USB application and passed on to the UART output. The result is the following display at the *Tera Term Pro* output window.



The content of the script is very simple but defined the sequence, the data and the timing to cause the data to scroll as seen.

```
// USB endpoint 1 data reception
+0 USB-1 = "Hello, World!!" 0d 0a
+100 USB-1 = 0a
+100 USB-1 = 0a
+100 USB-1 = 0a
+100 USB-1 = 0a
+200 USB-1 = 0a
+500 USB-1 = 0a
+1000 USB-1 = 0a
+1000 USB-1 = "Did you like the test?" 0d 0a
// End of test
```

The test can be repeated by commanding `Port Sim | Repeat last script`. When the simulator is terminated the last script setting is saved so that repeats can be quickly performed by using the short cut rather than having to reselect the file again.

- As final test with the simulator, type in some text into the *Tera Term Pro* window. This is being received by the application (from UART input) and being passed to the USB interface. For example, type in the alphabet from a to z.

Take a look in the simulation directory `\Applications\TaskerV1.3\Simulator` and you will see two files called `USB_Endpoint_00.txt` and `USB_Endpoint_02.txt` which have been recording endpoint transmission activity. The project is sending on endpoint 2 and opening this file shows that the following data has been transmitted:

```
[DATA1] 0x61
[DATA0] 0x62
[DATA1] 0x63
[DATA0] 0x64
[DATA1] 0x65
[DATA0] 0x66
[DATA1] 0x67
[DATA0] 0x68
[DATA1] 0x69
[DATA0] 0x6A
[DATA1] 0x6B
[DATA0] 0x6C
[DATA1] 0x6D
[DATA0] 0x6E
[DATA1] 0x6F
[DATA0] 0x70
[DATA1] 0x71
[DATA0] 0x72
[DATA1] 0x73
[DATA0] 0x74
[DATA1] 0x75
[DATA0] 0x76
[DATA1] 0x77
[DATA0] 0x78
[DATA1] 0x79
[DATA0] 0x7A
```

These 26 DATA1/0 messages are the letters of the alphabet (lower case) each being transmitted separately corresponding to the input characters as received by the UART and passed on immediately to the USB interface.

The routine responsible for passing the received data to the USB interface can be found in `usb_application.c`:

```
// Called to transmit data from defined USB endpoint (serial input uses this)
//
extern QUEUE_TRANSFER fnSendToUSB(unsigned char *ptrData, QUEUE_TRANSFER Length)
{
    return (fnWrite(USBPortID_comms, ptrData, Length)); // send to endpoint 2
}
```

If a break point is set in this routine the write process can be stepped through, including the handling of the USB interrupts involved. Changes to handling procedures can be made and tested using representative sequences to ensure that the new software basically performs as required before final tests are carried out on the target.

10. Conclusion

The purpose of this document was to explain how the μTasker USB demo can be tested on target hardware, showing CDC (Communication Device Class) support to enable terminal interaction, USB<->RS232 bridging and firmware upload.

The step-by-step description should enable even new users with little experience with USB to successfully complete and understand the process.

As well as target testing, the validation and analysis of the operation was shown to be possible using the μTasker simulator, giving a method enabling further development and testing in an efficient environment without the need for target operation and debugging.

A major advantage of using the CDC class and Virtual COM is the fact that existing PC programs (such as the terminal emulator used for the tests) can be used without it knowing whether the COM port that it is using is a real COM port or not. Existing PC programs based on COM communication can usually also use USB (as virtual COM port) with no modifications required at the PC side. The connection can however operate faster due to the higher USB communication data rate. Other USB classes would require the PC software to be modified to suit the chosen USB mode, requiring extra knowledge of the techniques required and extra work involved in order to achieve the equivalent state of operation.

Modifications:

- V0.0 14.12.2008 First draft.
- V.0.1 28.12.2008 Additional notes about SW upload and testing with file transfer.