Getting Started Demo for µEZ® GUI

Tutorial 3: TTL UART Serial Communications

How to use serial communication with the alternate TTL COMM port.

NOTE: Tutorial 2 and 3 are identical in procedure but use different communication ports.

Summary:

Tutorial 1 focused on how to get the demo project running on the µEZ GUI device using the project creator. Now that there is a working project and it is able to be flashed onto the device and run some external functionality can be added to it. In this tutorial you are going to talk with the µEZ GUI device via the serial communication ports (not the USB port). This will demonstrate how to add code to the project to use more hardware as well as how to initiate serial communication to and from the µEZ GUI device.

Index:

Parts:

Requirements:

Hardware:

- µEZGUI-4088-43WQN Website
- Segger J-Link Lite Module Website
- USB to Serial TTL Cable Website
- 2x USB A to USB Mini Cable Website

Software:

- IAR Embedded Workbench Website
- Putty Website
- µEZ GUI Project Creator (Previously Used) Website
- µEZ GUI Online Library Website

Part 1) Assemble The Hardware and Open the Project Demo

Step 1:

- Collect the required hardware.
- Acquire a USB to 3.3v TTL serial cable.

First get the hardware together. In this tutorial communication with the device will use the alternate COMM port on the board. To do that a special USB to TTL serial cable is needed which has an internal adapter to convert the signal. Below is a picture of the cable needed for this tutorial:

Figure 1: *Required Hardware*

Most cable adapters come with 6 pins broken out, but only 3 are needed. The alternate COMM port on the side of the µEZ GUI uses a Hirose DF13 connector.

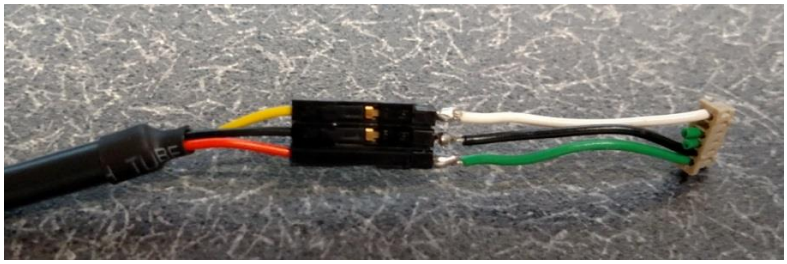

Figure 2: *USB TO TTL UART Connector*

Step 2:

- Connect USB Rx to µEZ GUI TX (Yellow to pin 1).
- Connect USB Tx to µEZ GUI Rx (Orange to pin 6).
- Connect Ground to Ground (Black to pins 4 or 5).

Connect the USB Rx to the µEZ GUI TX (Yellow to Pin 1), USB Tx to µEZ GUI Rx (Orange to Pin 6), and Ground to Ground (Black to Pins 4 or 5). A triangular arrow marks pin 1 on the back of the µEZ GUI. These images have been included to visually assist you:

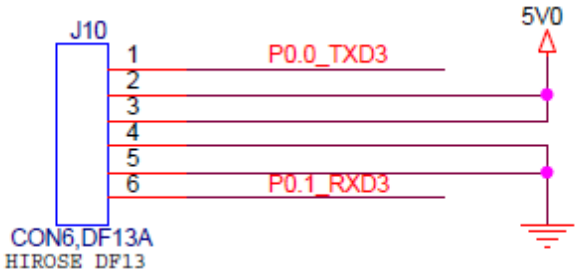

Figure 3: *TTL to UART Pinout*

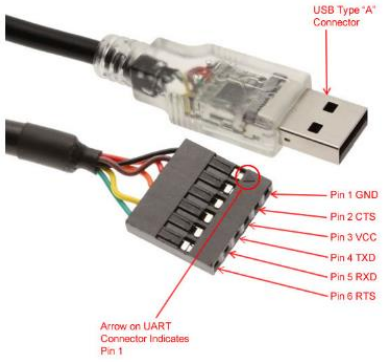

Figure 4: *USB to Serial Pinout*

In the pictures, the appropriate female compliment to the male Hirose DF13 connector on the µEZ GUI board is used. However, you do not need to use this connector, and if confident, can individually pin out the connection to the board.

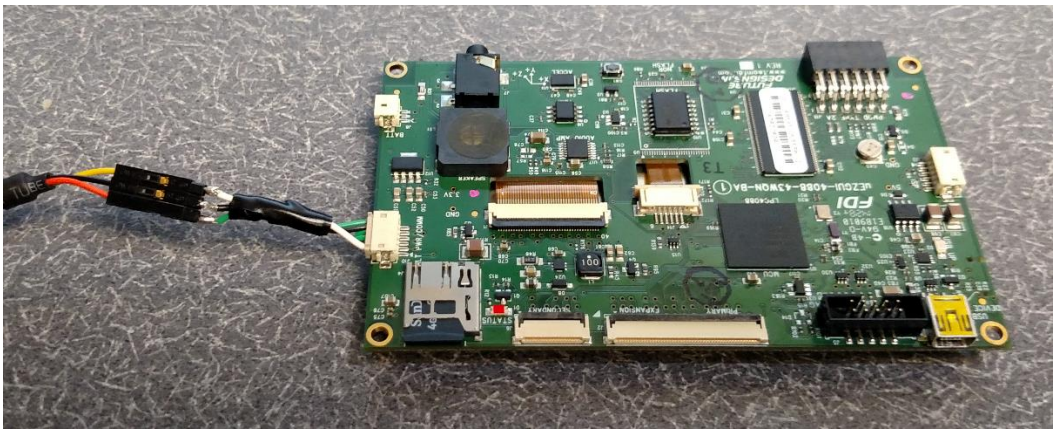

Figure 5: *Connecting to the TTL UART*

Step 3:

- Open IAR Embedded Workbench.
- Open the demo project from tutorial 1.

Once the µEZ GUI is connected to the computer via the alternate COMM port and the USB to Serial cable it is time to add the code. Now go to IAR Embedded Workbench and open the demo project that was created in tutorial 1. Code will be added for the serial communication functionality to it and have it run on its own in the background.

Part 2) Setup The Project Files for The Serial Communication

A simple echo test will be created to demonstrate the functionality of serial communication on the µEZ GUI. You are going to write a character to the device through a terminal window on the computer and have the µEZ GUI write that character back in the same window.

Step 4:

- Create a header file named *serialecho.h*
- Create a source file named *serialecho.c*
- Add these files to the demo project.

First, create a header and a source file in the project that will hold the code for the serial communication to run. Go into the project and make two new files and name them *serialecho.h* and *serialecho.c*. Do not put anything in them yet, just save them in the source folder of the project. Now add the files to the actual project. In the Files window of the Workspace dialog, right-click somewhere in the window and click *Add Files…* Navigate to the source folder of the project where the files were saved and click each of them to be added. Now the files are part of the project.

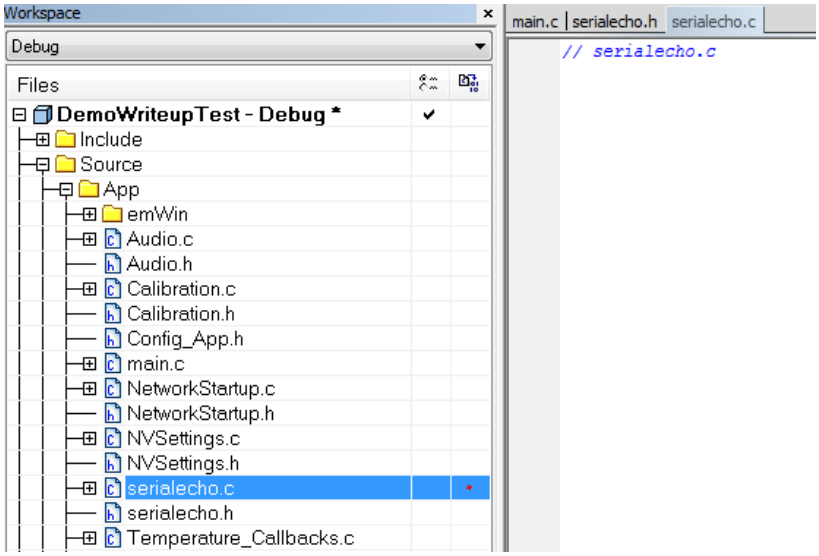

Figure 6: *IAR Workspace*

More setup is needed so that the functions will be able to run. µEZ GUI schedules functions by running tasks parallel to each other. For the communication, setup a task to run the function so that it can always be running in the background.

Step 5:

- #include *uEZ.h* and *uEZStream.h*
- Create a function prototype for "StartSerialEcho"

First, add two includes to the .h file, *uEZ.h* and *uEZStream.h*. These are the include files necessary for the µEZ library to run the code for UART, which will be seen later on. After that, make a function prototype to call with the task. Create the function prototype shown below:

```
TUInt32 StartSerialEcho(T_uezTask aMyTask, void *aParams);
```

The TUInt32 is the type needed to be called by a task to retain an error message.

NOTE: The parameters won't be used in the function but need to be included as shown to be called properly as a task.

Step 6:

- #include *serialecho.h*
- Put a while(1) loop inside the function.

Next setup the .c file. Include the *serialecho.h* file at the top. Now create a space for the function definition. Put a *while(1)* loop inside so that the task runs continually and doesn't exit, which will end communication.

When finished, the files will look like this:

```
// serialecho.h

#include <uEZ.h>
#include <uEZStream.h>

TUInt32 StartSerialEcho(T_uezTask aMyTask, void *aParams);
```

```
// serialecho.c

#include "serialecho.h"

TUInt32 StartSerialEcho(T_uezTask aMyTask, void *aParams)
{
    while(1)
    {

    }
}
```

Step 7:

- Navigate to *MainTask* in *main.c*
- Insert the UEZTaskCreate function call for StartSerialEcho.

The task must be started when the device starts so that it can run in the background. To do this, add a small amount of code to the *main.c* file. It is listed in the file window on the left. Open it and navigate down to the function called *MainTask*. This is the function that sets up the device tasks and functions when it is powered on. Just below the first *printf* statement insert this line of code:

```
UEZTaskCreate(StartSerialEcho, "Echo", 96, (void *)0, UEZ_PRIORITY_NORMAL, 0);
```

NOTE: Do not worry about the parameter values. They only need to be adjusted for advanced tasks which will be covered later on.

This is how an independent task is created in µEZ. Much like a thread, it will create the task using the *StartSerialEcho* function that was created earlier and will call the task "*Echo*". Remember to include the *serialecho.h* file in *main.c*. The program should be able to compile and run now. Though it isn't able to be seen, the task that was just created will now be running in the background when the µEZ device is powered on and loaded.

Part 3) Add The µEZ Library Calls for Serial Communication

Now, add the code of the function to make the task work. To do this you need to understand and use some function calls from the µEZ library. They are already setup to provide the needed functionality for the task. Conveniently, there is a Doxygen system on the FDI website which has an outline of all the different µEZ library functions. It is located here. When on the page go to the *Modules* tab at the top and navigate to *µEZStream*. You will now be on this page:

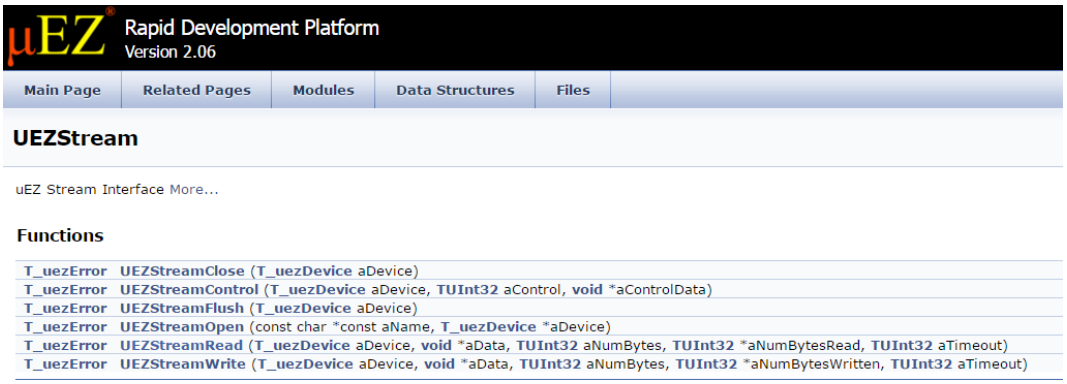

Figure 7: *Doxygen µEZ Library for UEZStream*

These are the functions that we are going to use to create the serial communication with the µEZ GUI device.

Step 8:

- Define the Variables
- Open the UEZStream

The first thing needed is to define the variables that are needed to make this happen. The first variable needed defines the device itself. Put "`T_uEZDevice uart3;`" above the while loop in the function. Next, a character buffer is needed to receive and send back the character. On the next line put "`TUInt8 receiveCOM[2];`" for the buffer. Only one index is needed for the input character. Now, open the communication stream by using the line "`uEZStreamOpen("UART3", &uart3);`". Now the function should look like this:

```
TUInt32 StartSerialEcho(T_uezTask aMyTask, void *aParams)
{
    T_uezDevice uart3;
    TUInt8 receiveCOM[1];

    UEZStreamOpen("UART3", &uart3);
```

One thing that is easily missed is that UART3 is not already initialized like UART0, therefore a line of code must be added to initialize it. Above the *uEZStreamOpen* function call, add this line:

```
UEZPlatform_FullDuplex_UART3_Require(2048,2048);
```

That will initialize the device's UART3 port and allow it to start running.

Step 9:

- Flush the stream buffer.
- Create the read command.
- Create the write command.

Now, go into the while loop and add the actual send and receive functionality. The first thing needed to be done at the start of every serial loop is to flush the stream so there isn't anything leftover to throw out in the communication buffer. Put the line "`µEZStreamFlush(uart3);`" as the first line inside the while loop. Now, include the receive and send functions which are called *UEZStreamRead* and *UEZStreamWrite*.

Copy the read function exactly as shown as the next part of the function:

```
UEZStreamRead(  uart3,
                (void*) receiveCOM,
                sizeof(receiveCOM),
                NULL,
                UEZ_TIMEOUT_INFINITE);
```

Now copy the write command exactly as shown:

```
UEZStreamWrite( uart3,
                (void*)receiveCOM,
                sizeof(receiveCOM),
                NULL,
                UEZ_TIMEOUT_INFINITE);
```

Notice that the arguments and parameters are identical for both. The parameters tell the µEZ GUI to listen for one character until it is received and then put it in the *receiveCOM* buffer. It will then write one character from the buffer (the size of the buffer itself) back to the terminal window.

Step 10:

- Close the *while(1)* loop.
- Insert a *return 0* at the end.

Now, close out the while loop and put a return 0 at the end. It will never be reached, but because it is a value returning function that can be called as a task, something has to be set to return.

After everything is done the function should look like this:

```
TUInt32 StartSerialEcho(T_uezTask aMyTask, void *aParams)
{
    T_uezDevice uart3;
    TUInt8 receiveCOM[1];

    UEZPlatform_FullDuplex_UART3_Require(2048,2048);
    UEZStreamOpen("UART3", &uart3);

    while(1)
    {
        UEZStreamFlush(uart3);

        UEZStreamRead(  uart3,
                        (void*) receiveCOM,
                        sizeof(receiveCOM),
                        NULL,
                        UEZ_TIMEOUT_INFINITE);

        UEZStreamWrite( uart3,
                        (void*)receiveCOM,
                        sizeof(receiveCOM),
                        NULL,
                        UEZ_TIMEOUT_INFINITE);
    }

    return 0;
}
```

NOTE: This is a bare bones version of the code that does not include error checking.

This version of this code does not include any formatting or error checking that would be vitally important in a real implementation. Examples on how to implement formatting or error checking can be found online in the µEZ libraries and in an already made inclusive demo project which has a more detailed example of this code. At this point you should make sure the project compiles and correct any errors that may be encountered.

Step 11:

- Compile the project.
- Correct any compilation errors.

Part 4) Run The Program with A Terminal Serial Connection

Step 12:

- Download PuTTY.
- Run PuTTY and configure for serial.
- Open the terminal window.

The program is now ready to run but an interface needs to be setup to use with the serial communication. An easy to use and free program with serial console functionality is the widely used *PuTTY*. It can be downloaded here.

Once downloaded, run the exe file and change the initial configuration screen to mirror this:

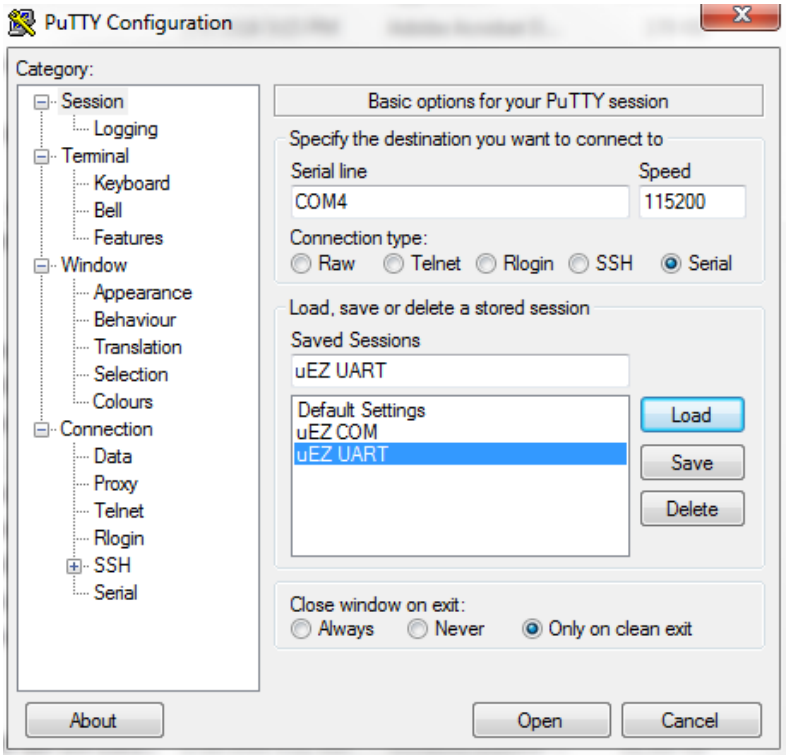

Figure 8: *Putty Config. Main Screen*

The COM number needs to be set to whatever COM port the serial cable is connected to on your computer (found in device manager), and is not necessarily COM4. Next, go to Connection->Serial and make sure everything matches as follows:

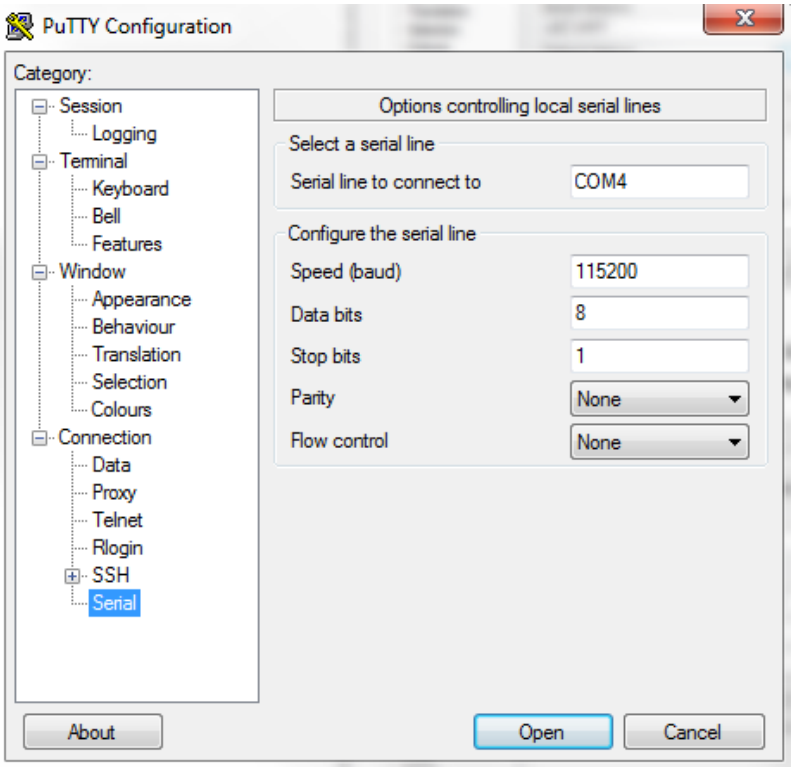

Figure 9: *Putty Config. Serial Screen*

Now, click open and a console window will appear. If the program has been uploaded and is running on the computer, then you will be able to type characters into the terminal window and have them typed back to you from the device.

Figure 10: *TTL Port Terminal Output*

Congratulations! You have now successfully implemented serial communication on the µEZ GUI device with UART3. You can now communicate to the µEZ GUI device through the alternate TTL serial communication port that is available. This can be very useful in issuing commands as well as debugging code later on. With the steps learned in this tutorial you will be able to add more device functionality more easily in the coming tutorials. Please reference our downloadable inclusive demo project for examples of more in depth coding related to this task and others.

Enjoy!

Reference & Information:

Parts:

Figures:

Hardware:

- µEZGUI-4088-43WQN Website
- Segger J-Link Lite Module Website
- USB to Serial TTL Cable Website
- 2x USB A to USB Mini Cable Website

Software:

- IAR Embedded Workbench Website
- Putty Website
- µEZ GUI Project Creator (Previously Used) Website
- µEZ GUI Online Library Website