

Getting Started Demo for μEZ® GUI

Tutorial 2: RS-232 COM Serial Communications

How to use UART serial communication with the RS-232 COM port on the EXP-1 Expansion Board.

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

NOTE: This tutorial requires the [FDI EXP-1](#) expansion board to be completed.

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:

- | | |
|---|--------|
| 1. Assemble The Hardware and Open the Project Demo | Page 1 |
| 2. Setup The Project Files for The Serial Communication | Page 2 |
| 3. Add The μEZ Library Calls for Serial Communication | Page 4 |
| 4. Run The Program with A Terminal Serial Connection | Page 6 |

Requirements:

Hardware:

- μEZGUI-4088-43WQN [Website](#)
- Segger J-Link Lite Module [Website](#)
- FDI EXP-1 Expansion Board [Website](#)
- RS-232 Serial 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 the FDI EXP-1 Expansion Board.

First get the hardware together. Communication will be made with the μ EZ GUI device using the RS-232 12-volt serial COM port. There is not an RS-232 port already on the μ EZ GUI board. The port that will be used is actually located on an expansion board called the EXP-1 that must be purchased separately. It is strongly suggested that you acquire this board or you will not be able to properly utilize this tutorial guide or later guides. The EXP-1 expansion board adds a lot more hardware than just serial. It also adds Ethernet, USB host and other functionality to your μ EZ GUI device. Though not required, it is a very good investment. An RS-232 serial cable is also needed. This must also be acquired separately. Below is a picture of the hardware needed for this tutorial:



Figure 1:
Required
Hardware

Next, connect the expansion board to the μ EZ GUI so that the on board serial port can be used for serial communication. The expansion board (EXP-1) connects to the μ EZ-GUI with a 50 pin ribbon cable in the primary expansion header (included with the EXP-1).

If you notice the ribbon cable connects upside down to the orientation of the μ EZ GUI device. This is so the expansion board can be mounted under the μ EZ GUI for easy access.



Figure 2:
Attaching
the Ribbon
Cable



Figure 3:
Mounted

Mounting it is not necessary for this tutorial but is something to keep in the back of your mind for later use. When the expansion board has been connected to the μ EZ GUI, connect the RS-232 cable to the expansion board and the computer. If needed, a USB to RS-232 adapter will work fine for this purpose as well.



Figure 4:
Connecting
the RS-232
Cable

Power up the μ EZ GUI with the USB port like normally. The μ EZ GUI will provide power to the expansion board so there is no need for extra power adapters. Also, connect the JTAG module up to the device again so that the code can be tested once you are done making it. 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 μ EZ GUI. A character will be written to the device through a terminal window on the computer and have the μ EZ 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.

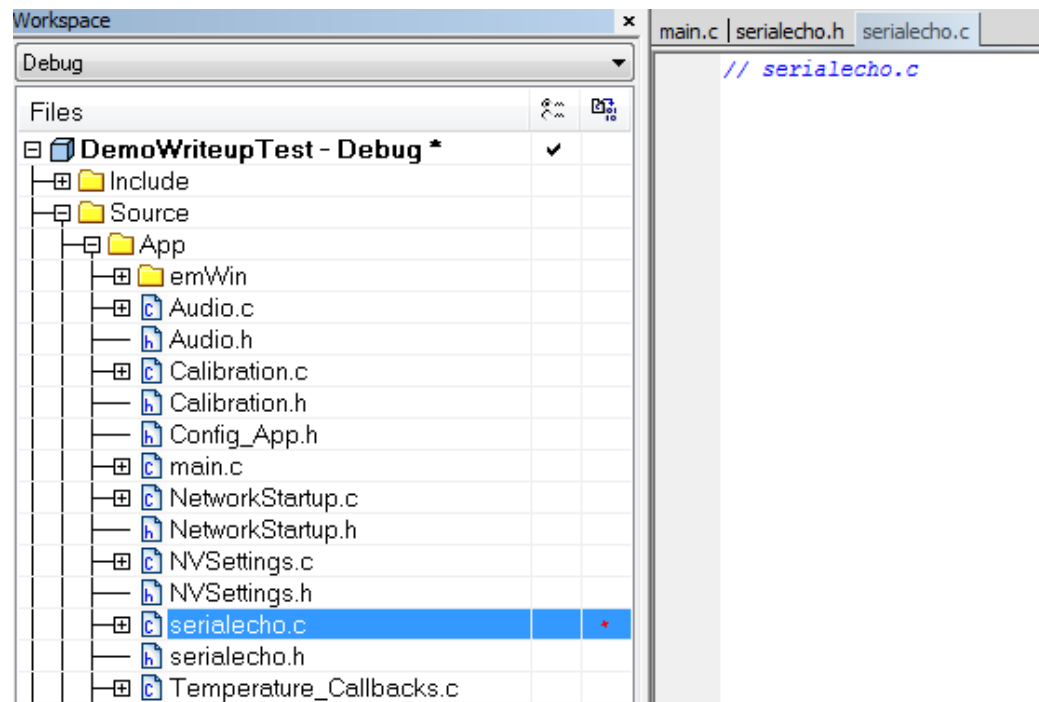


Figure 5:
IAR
Workspace

More setup is needed so that the functions will be able to run. μ EZ 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 μ EZ 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 μ EZ. 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 μ EZ device is powered on and loaded.

Part 3) Add The μ EZ 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 μ EZ 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 μ EZ library functions. It is located [here](#). When on the page go to the *Modules* tab at the top and navigate to *μ EZStream*. You will now be on this page:



UEZStream

uEZ Stream Interface More...

Functions

T_uezError	UEZStreamClose	(T_uezDevice aDevice)
T_uezError	UEZStreamControl	(T_uezDevice aDevice, TUInt32 aControl, void *aControlData)
T_uezError	UEZStreamFlush	(T_uezDevice aDevice)
T_uezError	UEZStreamOpen	(const char *const aName, T_uezDevice *aDevice)
T_uezError	UEZStreamRead	(T_uezDevice aDevice, void *aData, TUInt32 aNumBytes, TUInt32 *aNumBytesRead, TUInt32 aTimeout)
T_uezError	UEZStreamWrite	(T_uezDevice aDevice, void *aData, TUInt32 aNumBytes, TUInt32 *aNumBytesWritten, TUInt32 aTimeout)

Figure 6:
Doxygen
 μ EZ Library
for
UEZStream

These are the functions that we are going to use to create the serial communication with the μ EZ 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 uart0;`” 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[1];`” for the buffer. Only one index is needed for the input character. Now, open the communication stream by using the line “`UEZStreamOpen("Console", &uart0);`”. Now the function should look like this:

```
TUInt32 StartSerialEcho(T_uezTask aMyTask, void *aParams)
{
    T_uezDevice uart0;
    TUInt8 receiveCOM[1];

    UEZStreamOpen("Console", &uart0);
```

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 “`UEZStreamFlush(uart0);`” 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( uart0,
               (void*) receiveCOM,
               sizeof(receiveCOM),
               NULL,
               UEZ_TIMEOUT_INFINITE);
```


Now copy the write command exactly as shown:

```
UEZStreamWrite( uart0,
                (void*)receiveCOM,
                sizeof(receiveCOM),
                NULL,
                UEZ_TIMEOUT_INFINITE);
```

Notice that the arguments and parameters are identical for both. The parameters tell the μ EZ 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 uart0;
    TUInt8 receiveCOM[1];

    UEZStreamOpen("Console", &uart0);

    while(1)
    {
        UEZStreamFlush(uart0);

        UEZStreamRead( uart0,
                      (void*) receiveCOM,
                      sizeof(receiveCOM),
                      NULL,
                      UEZ_TIMEOUT_INFINITE);

        UEZStreamWrite( uart0,
                       (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 [\$\mu\$ EZ 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:

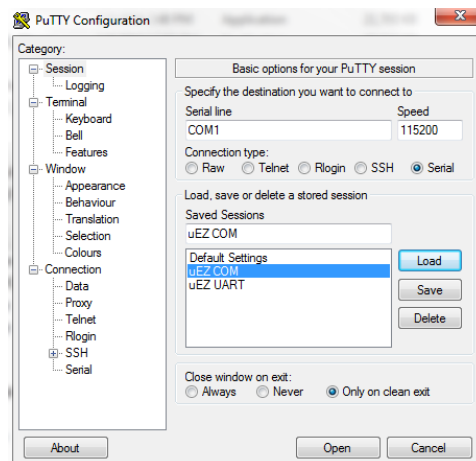


Figure 7:
Putty
Config.
Main
Screen

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 COM1. Next, go to Connection->Serial and make sure everything matches as follows:

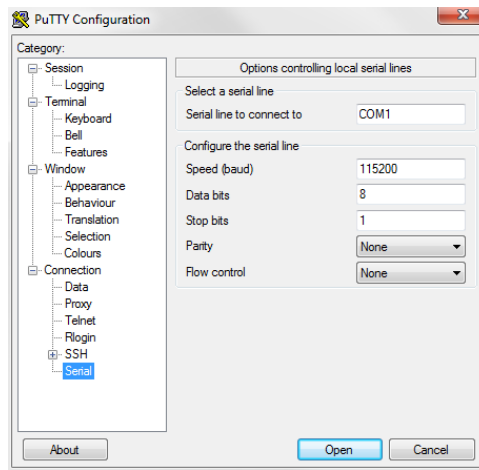


Figure 8:
Putty
Config.
Serial
Screen

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.

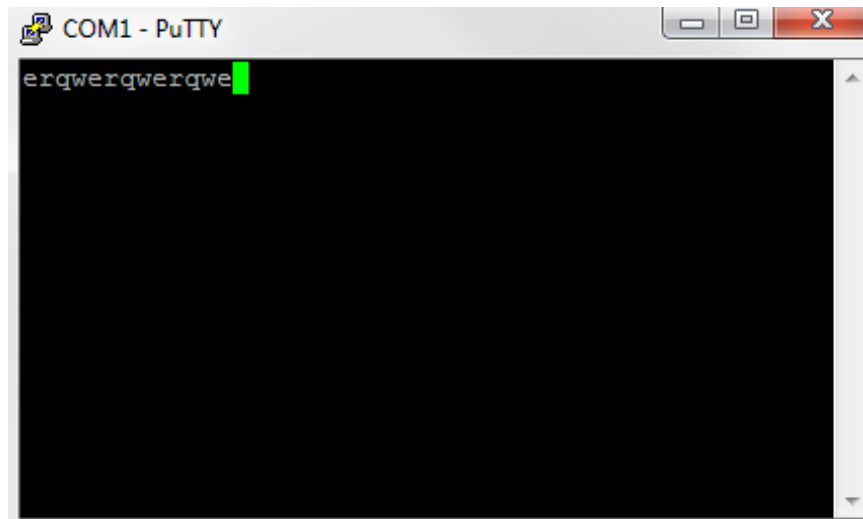


Figure 9:
RS232 Port
Terminal
Output

Congratulations! You have now successfully implemented serial communication on your μ EZ GUI device. You can now communicate to your μ EZ GUI device using the RS-232 serial communications ports available to you. This can be very useful in issuing commands as well as debugging code later on, or you can just have some fun with it like I like to do. With the steps we learned in this tutorial we 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:

- | | |
|--|--------|
| 5. Assemble The Hardware and Open Your Project Demo | Page 2 |
| 6. Setup The Project Files for The Serial Communication | Page 3 |
| 7. Add The μ EZ Library Calls for Serial Communication | Page 6 |
| 8. Run The Program with A Terminal Serial Connection | Page 9 |

Figures:

- | | |
|---|---------|
| 1. Required Hardware | Page 2 |
| 2. Attaching The Ribbon Cable | Page 2 |
| 3. Mounted | Page 3 |
| 4. Connecting The RS-232 Serial Cable | Page 3 |
| 5. IAR Workspace | Page 4 |
| 6. Doxygen μ EZ Library for UEZStream | Page 6 |
| 7. Putty Configuration Main Screen | Page 9 |
| 8. Putty Configuration Serial Screen | Page 10 |
| 9. COM Port Terminal Output | Page 10 |

Hardware:

- μ EZGUI-4088-43WQN [Website](#)
- Segger J-Link Lite Module [Website](#)
- FDI EXP-1 Expansion Board [Website](#)
- RS-232 Serial Cable [Website](#)
- 2x USB A to USB Mini Cable [Website](#)

Software:

- IAR Embedded Workbench [Website](#)
- Putty [Website](#)
- μ EZ GUI Project Creator (Previously Used) [Website](#)
- μ EZ GUI Online Library [Website](#)