
Embedded Environment Visualizer Library Handbook

by Hakan Sarac

Contents

1	Introduction	3
2	Matlab Routines	3
2.1	DataReceiver.m	3
2.1.1	Options	3
2.2	RawDataHandler.m	3
3	MultipleFloatDataSender library	4
3.1	Adding library to a CCS project	4
3.2	The content of the MultipleFloatDataSender library	5
3.2.1	void InitializeSciaRegisters(float fSciBaudRate)	5
3.2.2	void SciaBackgroundTask(void)	6
3.2.3	int SciSendMultipleFloatWithTag(float *FloatArrayToBeSent, uint16_t ui16NumberOfFloats)	6
3.2.4	int SciaUartSend_NoInterrupt(char *BuffWriteArray, unsigned int lengthOfData);	7
4	Using the EEVL library	7
5	Bugs and Warnings	7

1 Introduction

This document explains how to use the Embedded Environment Visualizer Library (EEVL). EEVL consists of three parts: two matlab routines and a library. These parts are called:

- DataReceiver.m
- RawDataHandler.m
- MultipleFloatDataSender library

Each of these items are explained in their relevant section.

The aim of this library is to be able to collect, visualize and save data from an embedded environment, which is most of the time a trouble.

2 Matlab Routines

2.1 DataReceiver.m

This routine is the main routine that collects, saves, plots the received data on the computer.

2.1.1 Options

This routine has several options, which are:

- **EnableSaving** : This enables/disables the saving of the received data. The received data is tagged with date and hour information.
- **ProcessRawDataThresholdInBytes** : When this routine received **ProcessRawDataThresholdInBytes** bytes of data, the received data is processed plotted/saved.
- **EnablePlotting** : This enables/disables the plotting of the received data. The received data is being plotted with time axis.
- **DataSampleRate (Hz)**: This variable is used for determining the time axis of the plotting. It should match the data sending rate of the DSP.
- **TheSerialChannelDevice** : The serial channel port seen by the computer is stated in this variable. The port value can be determined from the device manager. For the example given in Figure 1, TheSerialChannelDevice should be set as 'COM5'.

2.2 RawDataHandler.m

The saved data by DataReceiver.m routine can be handled using this routine.

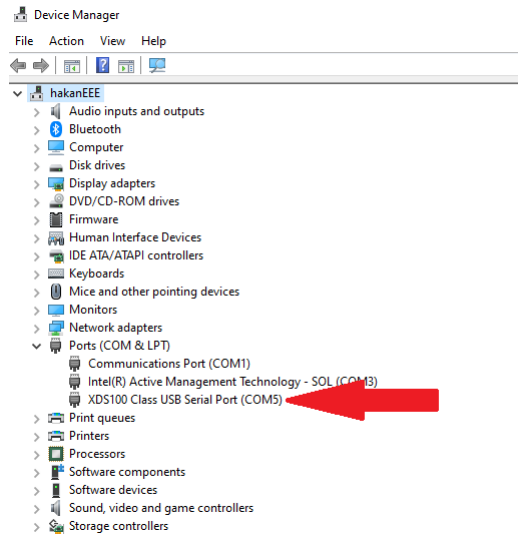


Figure 1: The serial port on the Device Manager screen

3 MultipleFloatDataSender library

This library is the critical part of the EEVL. For the embedded environment, all the necessary functions and/or variables are defined in this library. In order to add the library to a CCS project, there are three steps to be done.

3.1 Adding library to a CCS project

1. Add the MultipleFloatDataSender.lib file to **Build -> C2000 Linker -> File Search Path**, as shown in Figure 2.
2. Add the library project path to the included path, as show in Figure 3
3. Include the library header "**MultipleFloatDataSender.h**" to the project, as shown in Figure 4

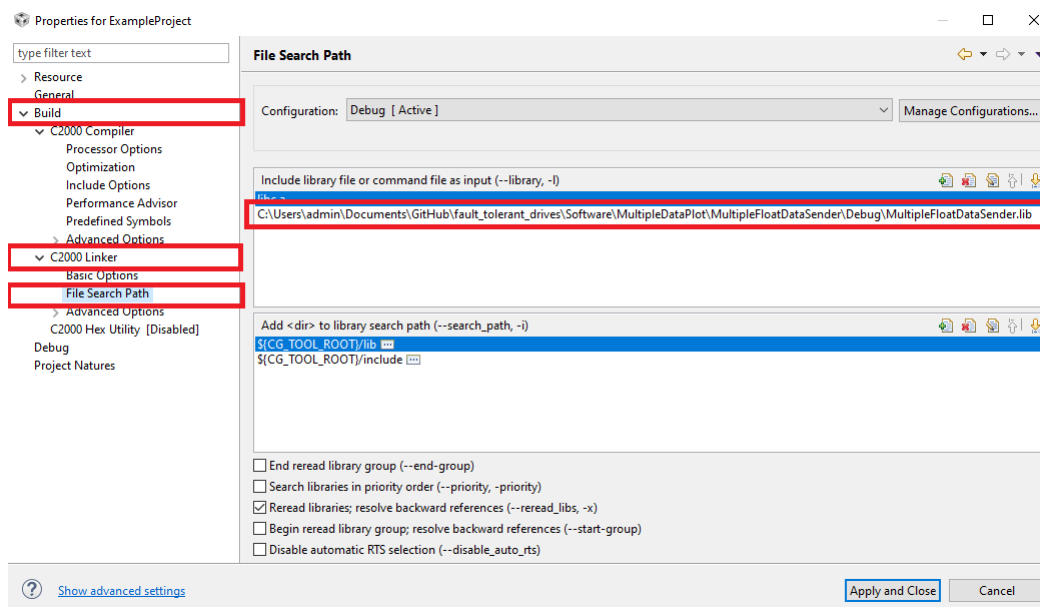


Figure 2: Linking the library file

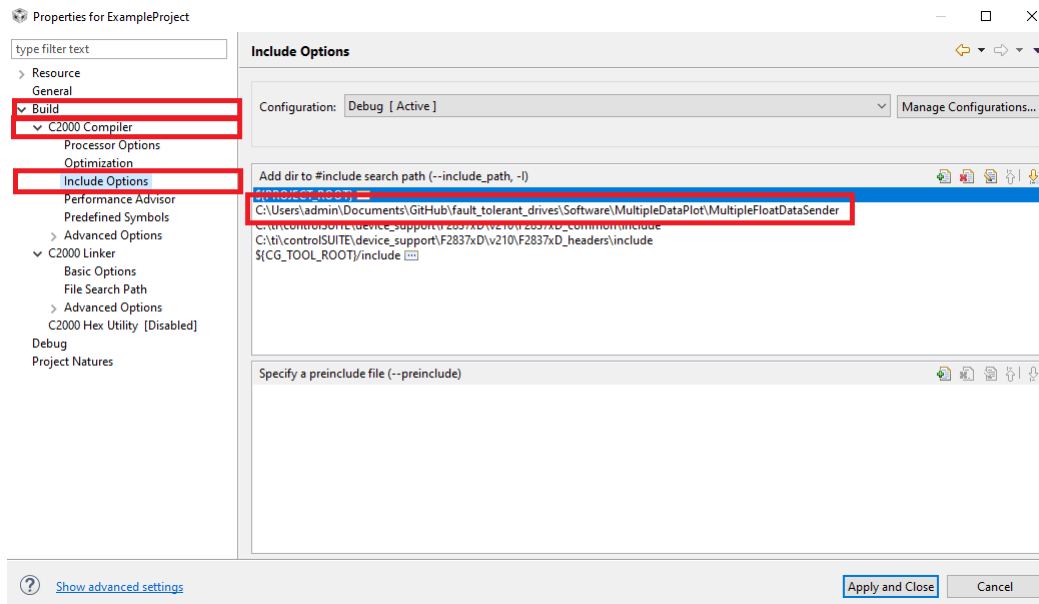


Figure 3: Addition of library path to the project

```

1#include <F2837xD_Device.h>
2#include <F2837xD_Examples.h>
3#include <string.h>
4#include "MultipleFloatDataSender.h"
5
6__interrupt void cpu_timer0_isr(void); /*prototype of the ISR functions*/
7__interrupt void cpu_timer1_isr(void); /*prototype of the ISR functions*/
8__interrupt void cpu_timer2_isr(void); /*prototype of the ISR functions*/
9
10
11
12
13
14
15/**
16 * main.c
17 */
18
19float fSynchronousData = 5.0;
20unsigned char SendData = 0;
21float fFloatToBeSent[6] = {2,2,2,2,2,2};
22#define SENDFREQ 2500
23#define PI 3.14159
24
25int main(void)
26{
27
28    InitSysCtrl(); /*initialize the peripheral clocks*/
29
30    EALLOW;
31    ClkCfgRegs.PERCLKDIVSEL.bit.EPMCLKDIV = 0; // EPM Clock Divide Select: /1 of PLL
32    EDIS;

```

Figure 4: Addition of library path to the project

3.2 The content of the MultipleFloatDataSender library

For initializations and operation, the library has several functions. Each of these functions and an example usage is provided in this section

3.2.1 void InitializeSciaRegisters(float fSciBaudRate)

3.2.1.1 Description

InitializeSciaRegisters() attempts to initialize the Scia registers for the library operation. The baudrate of the channel can be set by the **fSciBaudRate** variable.

The GPIOs for the Scia are NOT initialized. The developer should initialize the preferred Scia GPIOs by him/herself.

3.2.1.2 Return value

This function has no return value.

3.2.1.3 Cautions

This function does not initialize the GPIOs for SCIA. The user must initialize the GPIOs by him/herself.

For **F28379D LaunchPad**, GPIO42 and GPIO43 must be set to Scia module. For **F28379D controlCARD**, GPIO28 and GPIO29 must be set to Scia module. If you are using some external JTAG probe, the scia GPIOs that are connected to the FTDI chip must be initialized.

3.2.2 void SciaBackgroundTask(void)

3.2.2.1 Description

SciaBackgroundTask() governs the sending of the data through SCI interface. To be able to send the data packets through SCI, this function should be called periodically. It is advised to call this function outside of ISRs.

3.2.2.2 Return value

This function has no return value.

3.2.2.3 Cautions

This function should be called **at least** once every 1 milisecond, can be called more frequent as well. It is advised to call this function outside of the ISRs. An example call is given in [this link](#).

3.2.3 int SciSendMultipleFloatWithTheTag(float *FloatArrayToBeSent, uint16_t ui16NumberOfFloats)

3.2.3.1 Description

SciSendMultipleFloatWithTheTag() is the function that is used for sending the data to the computer.

When called, this function sends **ui16NumberOfFloats** number of floats inside the **FloatArrayToBeSent** array. An example usage of this function can be found in [this link](#).

3.2.3.2 Return value

On success, this function returns the number of bytes that is sent through Scia. This function return -1 if the data buffer of the Scia is full.

3.2.3.3 Cautions

To send 6 float numbers with 921600 baudrate value, the maximum call rate of the function should not exceed 3kHz.

The reason is that a package of data consists of 4 bytes of tag data and 4 bytes for each float number, therefore each data package consists of 28 bytes. To send 28 bytes of data, 280 bits are sent through the Scia (8 bit data, 1 start bit, 1 stop bit). All these values results in the maximum data rate stated in (1).

$$MaximumPackageRate = \frac{BaudRateValue(bits/sec)}{BitsPerPackage} = \frac{921600}{280} = 3300 \text{ packages/sec} \quad (1)$$

3.2.4 int SciaUartSend_NoInterrupt(char *BuffWriteArray, unsigned int lengthOfData);

3.2.4.1 Description

This function sends **lengthOfData** bytes of data starting from the address stated with **BuffWriteArray** pointer.

For the normal operation of the MultipleFloatDataSender library, SciaUartSend_NoInterrupt function is not used.

3.2.4.2 Return value

On success, this function returns the number of bytes that is sent through Scia.

3.2.4.3 Cautions

For the normal operation of the MultipleFloatDataSender library, SciaUartSend_NoInterrupt function is not used.

4 Using the EEVL library

Before starting to use the EEVL library make sure that the followings are done.

1. The options of DataReceiver.m (stated in 2.1.1) has been properly set.
2. The MultipleFloatDataSender library should be initialized properly, as explained in 3.1.
3. The corresponding gpios of the Scia must be initialized. For **F28379D LaunchPad**, GPIO42 and GPIO43 must be set to Scia module. For **F28379D controlCARD**, GPIO28 and GPIO29 must be set to Scia module.
4. Scia registers are initialized properly, as explained in 3.2.1
5. SciaBackgroundTask is called at least 1kHz rate, as stated in 3.2.2
6. SciSendMultipleFloatWithTheTag function is called at a rate of 3kHz (max), as stated in 3.2.3.

Overall, [an example CCS project](#) and recommended settings for the [DataReceiver.m](#) are provided as hyperlinks.

5 Bugs and Warnings

1. During operation DataReceiver.m might give deficient data error as shown in Figure 5. If you get this error too much (around once a second) it is recommended to reduce the value of **ProcessRawDataThresholdInBytes** inside the DataReceiver.m routine.
2. If the serial channel has already been opened, you might get the error given in Figure 6. You should call "fclose(instrfind)" command in MATLAB to close, if the device is opened with MATLAB and not closed. (If the device opened by some other program, then that program must be closed.)

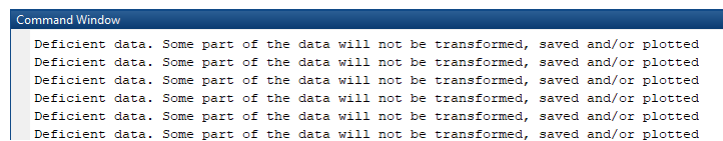


Figure 5: Deficient data error

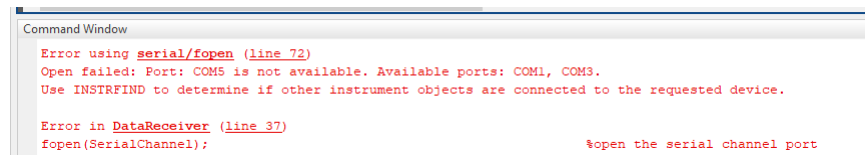
A screenshot of a 'Command Window' interface. The window has a title bar and a light gray background. It displays two error messages in red text. The first message is an 'Error using serial/fopen (line 72)' with a sub-message: 'Open failed: Port: COM5 is not available. Available ports: COM1, COM3. Use INSTRFIND to determine if other instrument objects are connected to the requested device.' The second message is an 'Error in DataReceiver (line 37)' showing the code 'fopen(SerialChannel);' followed by a red comment '%%open the serial channel port'.

Figure 6: The serial channel device already open error

References