

30 September 2008

A snapshot ZGUS radio library, built on SimpliciTI, with a sample radio modem source code application for the eZ430-RF2500 kit is available from <http://www.zgus.com/rf/zgusrf>.

The library is for Code Composer Essentials 3 development suite and works with the free version of Code Composer Essentials 3.

The radio libraries package ([http://www.zgus.com/rf/zgusrf/radio\\_libraries.zip](http://www.zgus.com/rf/zgusrf/radio_libraries.zip)) consists of two compiled libraries (zgusrf/zgusrf.lib by ZGUS and SimpliciTI/SimpliciTI.lib ported from TI code) and a small source code radio modem application in relay\_uart\_rf/ that allows two UART ports to communicate transparently by radio without any hardware or inbuilt software handshaking. The UARTS do not have to run at the same BAUD rates.

The radio modem library purpose is solely to enable demonstration of use of a forthcoming application by radio in as simple and cheaply a manner as practical. Since it is unlikely there will be further development and documentation before use of the ZGUS radio modem library in its first sample application, the library is being released now. The radio modem library facilities and buffering capabilities have been designed to fit specific applications and may not suit your applications. The fact is the library works nicely for its intended purpose.

**While support is only applicable for use in sample ZGUS applications, anyone with questions arising from this posting is welcome to post them to the ZGUS support forum at <http://groups.yahoo.com/group/zgus>.**

### **Following are some specific usage notes, including current limitations.**

Currently the ZGUS uart and radio initialisation library call **txrx\_init** will only accept parameters for UART baud rates 2400 or 9600. These parameters have nothing to do with the radio baud rates. To try other UART baud rates then the UART baud rate will need to be adjusted AFTER a call to **txrx\_init**. There are two other txrx\_init parameters, one is the SimpliciTI radio device address, the other is a timeout parameter in milliseconds used for both radio and UART purposes. 100 is a safe value.

One side must call with **rf\_open(RFOPEN\_NOBLOCK)** for a SMPL\_Link no blocking call and the other side with **rf\_open(RFOPEN\_BLOCK)** for a SMPL\_LinkListen blocking call that is repeated until SMPL\_SUCCESS results. There is currently little use of power saving. To compile for blocking use **#define LINKLISTEN** in zgus\\_\_sgetapps.h (with **//#define LINKTO** commented out). To compile for non blocking use **#define LINKTO** in zgus\\_\_sgetapps.h (with **//#define LINKLISTEN** commented out). These macros automatically ensure different SimpliciTI radio device addresses are used.

After **txrx\_init** and **rf\_open**, a call to **relay\_rf()** (which does not return) will implement transparent transfer of data between the UART and radio in both directions.

On a practical usage note, the eZ430 MSP430 Application UART causes CPU usage to jump very high after the first character has been sent, so it is recommended to use another USB-UART device with a PC. The problem occurred with both HyperTerm and with Hidprobe with the eZ430 MSP430 Application UART. The problem did not occur with HyperTerm or with Hidprobe when the PC was connected to a FTDI chip instead (using a DLP-2232M board) instead of the eZ430 Application UART.

Hidprobe PC software for UART use is available from <http://www.zgus.com/hid>

The basic radio parameters are the same as for the port of SimpliciTI-1.0.6 to Code Composer Essentials for the '2 simple End Devices with bi-di' application for the ez430-RF2500 board. This port is also available from <http://www.zgus.com/rf> in source code form. To change hard coded radio register values then facilities provided by the SimpliciTI library can be used. The compiled library has a number of minor differences with the source code version. The main one is that TimerA is kept running and overflows through continuous counting without use of TACCR0.

**A requirement of any application that uses zgusrf.lib is to implement a timer tick that keeps a variable defined in the application as `volatile mstime_t mstime_isr=0;` updated.**

This variable is declared as an extern in zgusrf.lib in order to force a user to pay attention to its importance by causing an error if the variable is not defined at the application level. However the user is allowed the user to implement update of the variable whatever way they choose. Please see txrx\_sw.c in the application for an example. In fact easy update of this variable is the reason for the modification to SimpliciTI so as TimerA is not stopped and overflows through continuous counting. A function in zgusrf.lib that directly uses mstime\_isr is mstime\_t mstime(void).

The ZGUS library buffers characters received from the UART to a size of four times the size of the data payload transferred by radio (64 bytes). The data payload per radio transferred is currently hard coded to 16 resulting in a hard coded MAX\_APP\_PAYLOAD that is set to 17. If received UART characters have not been removed and placed in other buffers in time then data to be transmitted will be overwritten by later data from the UART. The ZGUS library requires all radio data packets to be acknowledged. If there is a long run of failed acknowledgments then a reset is forced automatically. There is currently no facility to retransmit an unacknowledged packet.

**Pumping data into the UART in bursts of no more than 63 bytes, rather than with long gaps between characters sent, will result in more efficient use of the radio channel.**

The easiest way to test radio modem is to use the eZ430 Application UART with one radio board (say non blocking open) and use the battery board on the other radio board (say blocking open) and short P3.4 to P3.5 on the battery board.

If the end opened as **RFOPEN\_BLOCK** end loses power then after re-powering the other end will reset after a number of consecutive failed acknowledgments which will force a fresh link.

If the end opened as **RFOPEN\_NOBLOCK** end loses power then after re-powering communication will re-establish nearly immediately.

SimpliciTI.lib uses the PORT2 interrupt. Zgusrf.lib uses UART TX interrupt and a UART RX interrupt to transfer buffered data. The sample application use the TIMERA1 (overflow flag) interrupt to provide a required timer tick

There are functions in the ZGUS radio library that are for other applications and have no relevance for radio modem use.

The compiled SimpliciTI.lib hard codes smpl\_nwk\_config.dat (converted for include file use) parameters to

```
#define MAX_HOPS 3
#define MAX_HOPS_FROM_AP 1
#define MAX_APP_PAYLOAD 17
#define DEFAULT_LINK_TOKEN 0x01020304
#define DEFAULT_JOIN_TOKEN 0x05060708
```

The compiled SimpliciTI.lib hard codes smpl\_config.dat (converted for include file use) parameters to

```
#define NUM_CONNECTIONS 2
#define SIZE_INFRAME_Q 2
#define SIZE_OUTFRAME_Q 2
#define END_DEVICE
```

John Heenan, Auscyber Pty Ltd

Not including SimpliciTI, Copyright 2008 Auscyber Pty Ltd. All rights reserved.

ZGUS is a trading name of Auscyber Pty Ltd

YOU ACKNOWLEDGE AND AGREE THAT THE SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY, TITLE, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL AUSCYBER PTY LTD OR ITS LICENSORS BE LIABLE OR OBLIGATED UNDER CONTRACT, NEGLIGENCE, STRICT LIABILITY, CONTRIBUTION, BREACH OF WARRANTY, OR OTHER LEGAL EQUITABLE THEORY ANY DIRECT OR INDIRECT DAMAGES OR EXPENSES INCLUDING BUT NOT LIMITED TO ANY INCIDENTAL, SPECIAL, INDIRECT, PUNITIVE OR CONSEQUENTIAL DAMAGES, LOST PROFITS OR LOST DATA, COST OF PROCUREMENT OF SUBSTITUTE GOODS, TECHNOLOGY, SERVICES, OR ANY CLAIMS BY THIRD PARTIES (INCLUDING BUT NOT LIMITED TO ANY DEFENCE THEREOF), OR OTHER SIMILAR COSTS.

Should you have any questions regarding your right to use this Software, contact Auscyber Pty Ltd T/A ZGUS at [www.zgus.com](http://www.zgus.com).