Application Note:	Getting started with eRIC-SIGFOX
Suitable for:	eRIC-SIGFOX radio module
Date:	July 2017
Version:	1.1
Author:	SG – LPRS

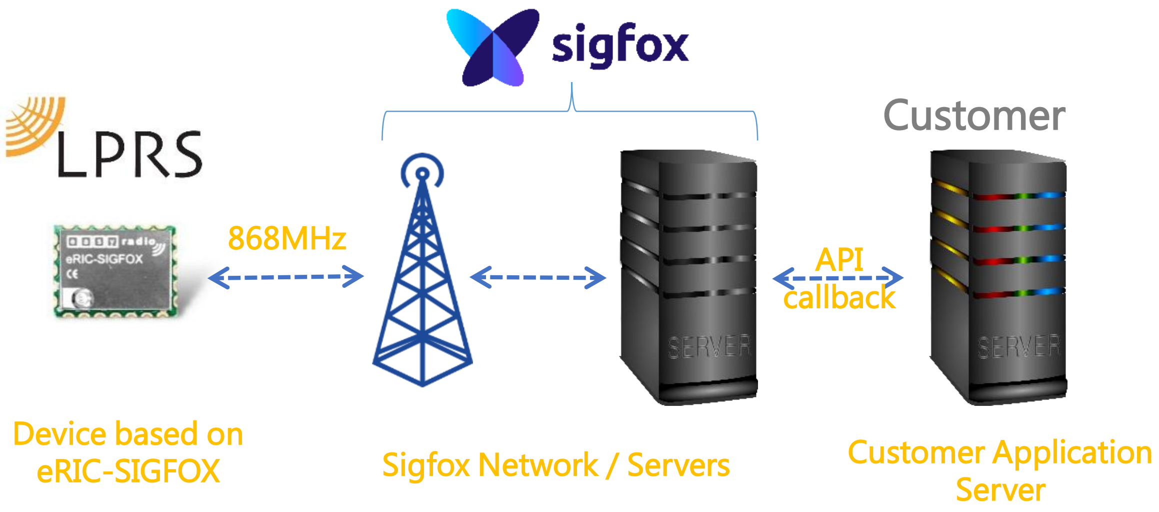
Contents

1. Purpose

eRIC-SIGFOX is an easy to use, simple to control Sigfox AT Command modem module, for allowing device to communicating with the Sigfox world wide radio network for IoT / M2M devices.

This application note aims to help you quickly and easily connect to the Sigfox network and send data.

2. Sigfox Network Overview

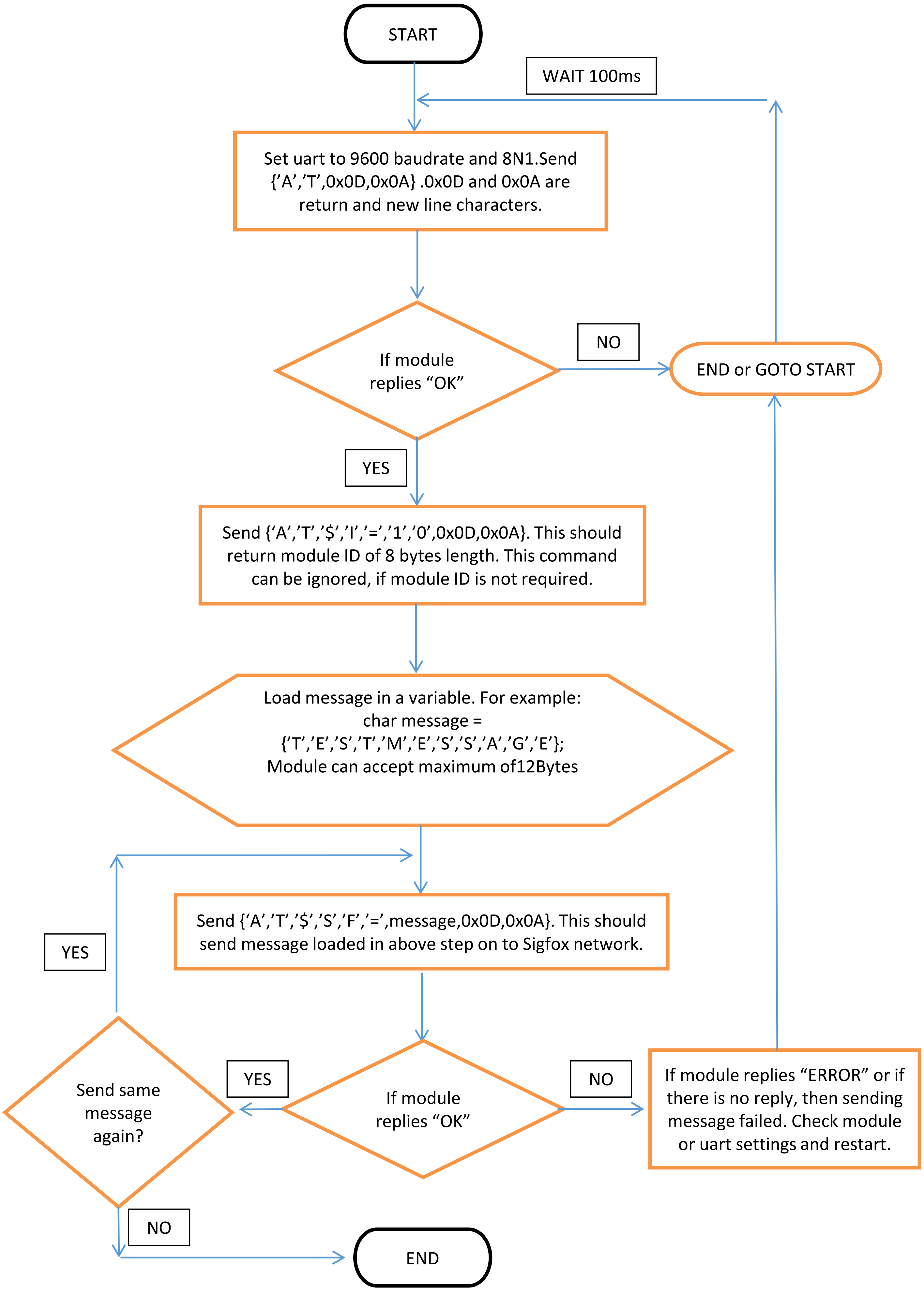


3. Requirements

- eRIC-SIGFOX Radio module (pre-provisioned with LPRS Sigfox Connectivity).
- eR-EVK-SIGFOX Eval Kit (or eRIC-DK board) & related antenna / USB cable.
- PC or MAC with 1 available USB connector.
- Serial terminal program such as Realterm (https://realterm.sourceforge.io/).

OK let's begin…….please turn to page 3 to view the flow diagram which overviews how to send messages to the Sigfox network.

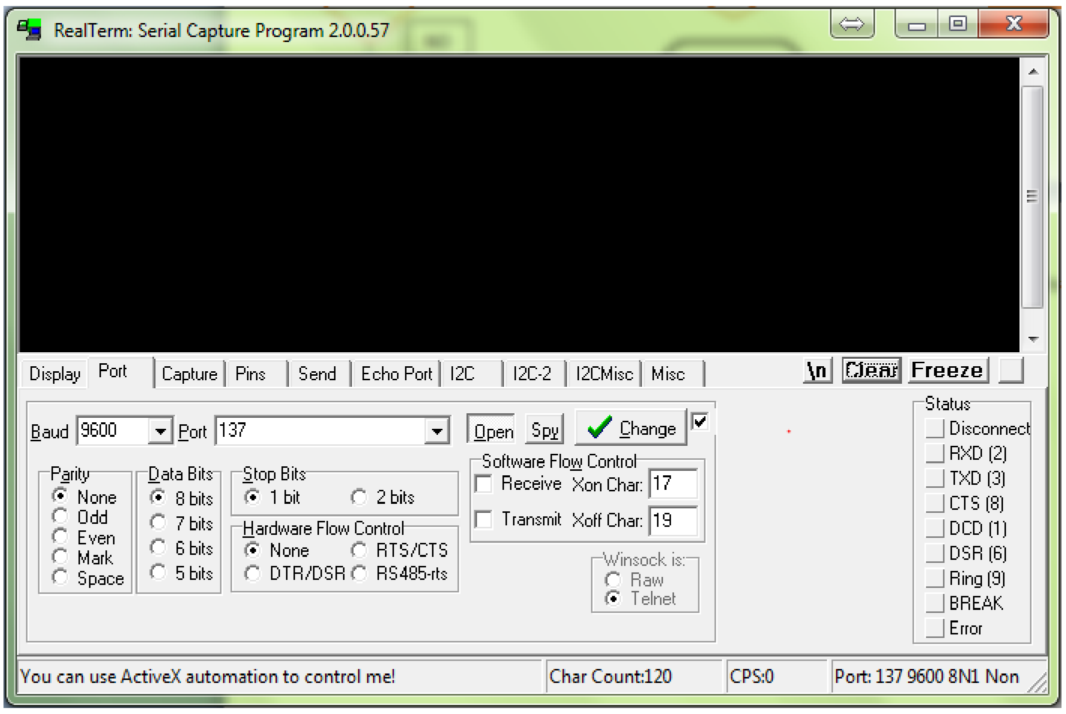
4. Command Flow diagram



START
WAIT 100ms
Set uart to 9600 baudrate and 8N1.Send {'A','T',0x0D,0x0A} .0x0D and 0x0A are return and new line characters.
If module replies "OK"
NO
END or GOTO START
YES
Send {'A','T','$','I','=','1','0',0x0D,0x0A}. This should return module ID of 8 bytes length. This command can be ignored, if module ID is not required.
Load message in a variable. For example: char message = {'T','E','S','T','M','E','S','S','A','G','E'}; Module can accept maximum of12Bytes
YES
Send {'A','T','$','S','F','=',message,0x0D,0x0A}. This should send message loaded in above step on to Sigfox network.
Send same message again?
YES
If module replies "OK"
NO
If module replies "ERROR" or if there is no reply, then sending message failed. Check module or uart settings and restart.
NO
END

5. Using Realterm software with eRIC-SIGFOX

5.1 Open RealTerm and click Port Tab. Choose 9600 Baud with Parity None, 8 Data bits, and 1 Stop bit. The Comm Port for the sigfox module connected in this screenshot is 137. Click Open button.



RealTerm: Serial Capture Program 2.0.0.57

Display | Port | Capture | Pins | Send | Echo Port | I2C | I2C-2 | I2CMisc | Misc | \n | Clear | Freeze

Baud 9600 Port 137 Open Spy Change ✓

Parity: [✓ None] Odd Even Mark Space

Data Bits: [✓ 8 bits] 7 bits 6 bits 5 bits

Stop Bits: [✓ 1 bit] 2 bits

Hardware Flow Control: [✓ None] RTS/CTS DTR/DSR RS485-rts

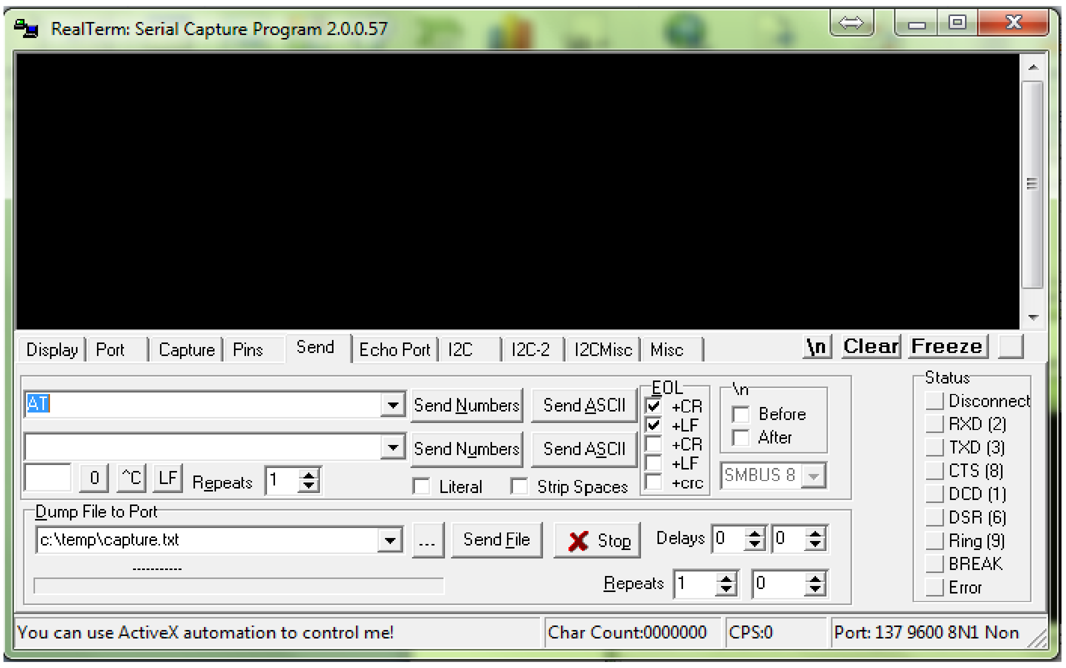
Software Flow Control: Receive Xon Char: 17 Transmit Xoff Char: 19

Winsock is: Raw [✓ Telnet]

Status: Disconnect RXD (2) TXD (3) CTS (8) DCD (1) DSR (6) Ring (9) BREAK Error

You can use ActiveX automation to control me! | Char Count:120 | CPS:0 | Port: 137 9600 8N1 Non

5.2 Click Send tab. Type AT in the first text box with +CR and +LF ticked.



RealTerm: Serial Capture Program 2.0.0.57

Display | Port | Capture | Pins | Send | Echo Port | I2C | I2C-2 | I2CMisc | Misc | \n | Clear | Freeze

AT | Send Numbers | Send ASCII

EOL: [✓ +CR] [✓ +LF] +CR +LF +crc

\n: Before After

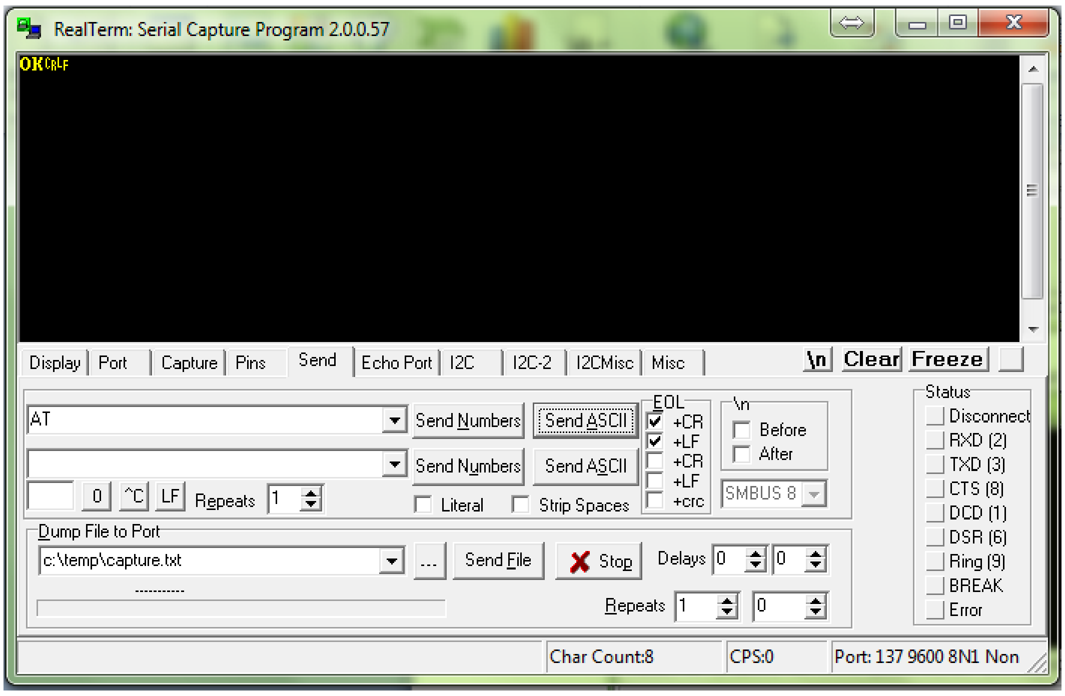
0 ^C LF Repeats 1 | Literal | Strip Spaces | SMBUS 8

Dump File to Port: c:\temp\capture.txt | ... | Send File | Stop | Delays 0 0 | Repeats 1 0

Status: Disconnect RXD (2) TXD (3) CTS (8) DCD (1) DSR (6) Ring (9) BREAK Error

You can use ActiveX automation to control me! | Char Count:0000000 | CPS:0 | Port: 137 9600 8N1 Non

5.3 Click Send ASCII button and wait for OK to be displayed on black screen.



```
OKCRLF
```

AT

EOL: [✓ +CR] [✓ +LF] +CR +LF +crc

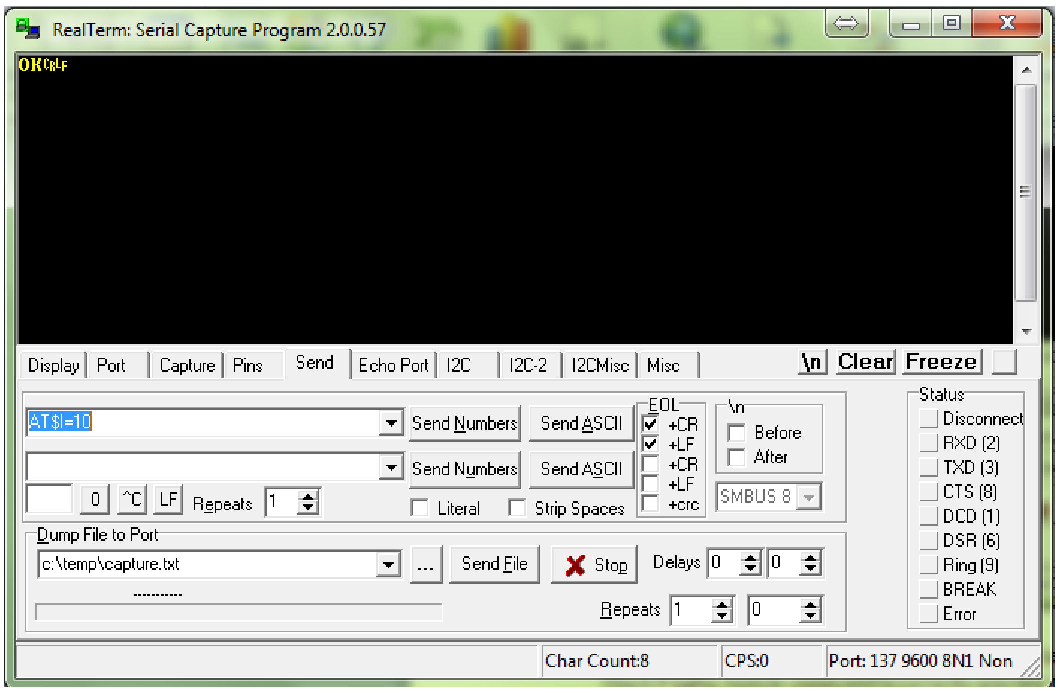
Repeats 1

c:\temp\capture.txt

Char Count:8 | CPS:0 | Port: 137 9600 8N1 Non

5.4 If there is no OK displayed on black screen, check Baud setting, display settings of Realterm. Check if Sigfox module Comm port is correctly selected. Try again from start.

5.5 Clear first text box and type AT$I=10 with +CR and +LF ticked.



```
OKCRLF
```

AT$I=10

EOL: [✓ +CR] [✓ +LF] +CR +LF +crc

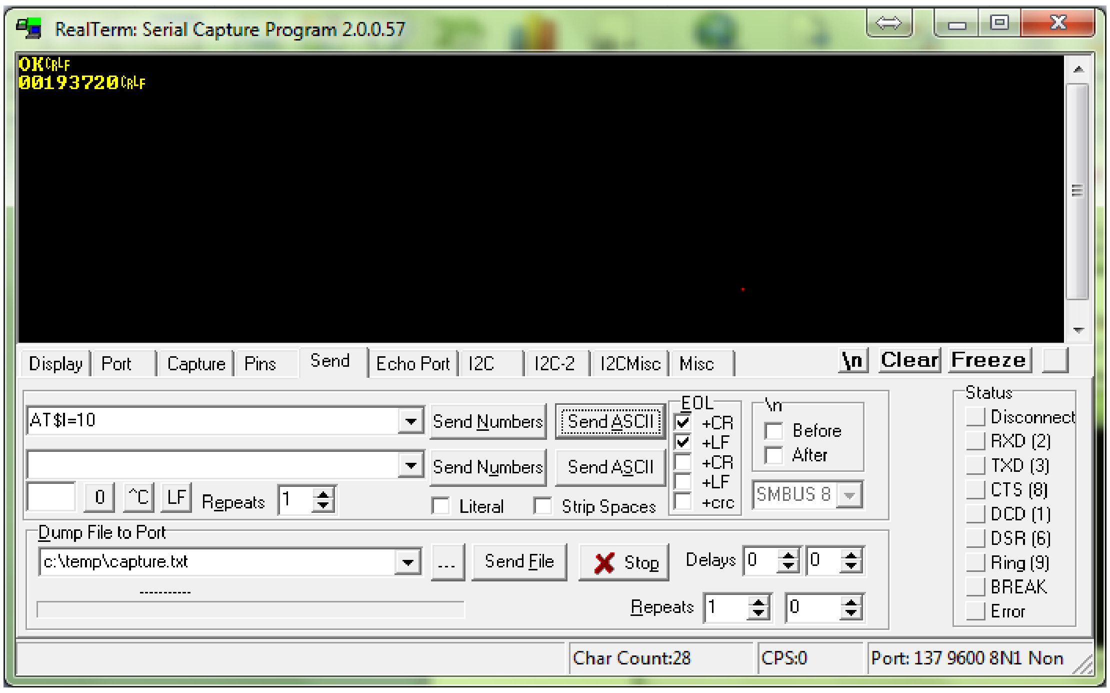
Repeats 1

c:\temp\capture.txt

Char Count:8 | CPS:0 | Port: 137 9600 8N1 Non

www.lprs.co.uk

5.6 Click Send ASCII button. The module ID should be displayed on black screen.



RealTerm: Serial Capture Program 2.0.0.57

```
OKCRLF
00193720CRLF
```

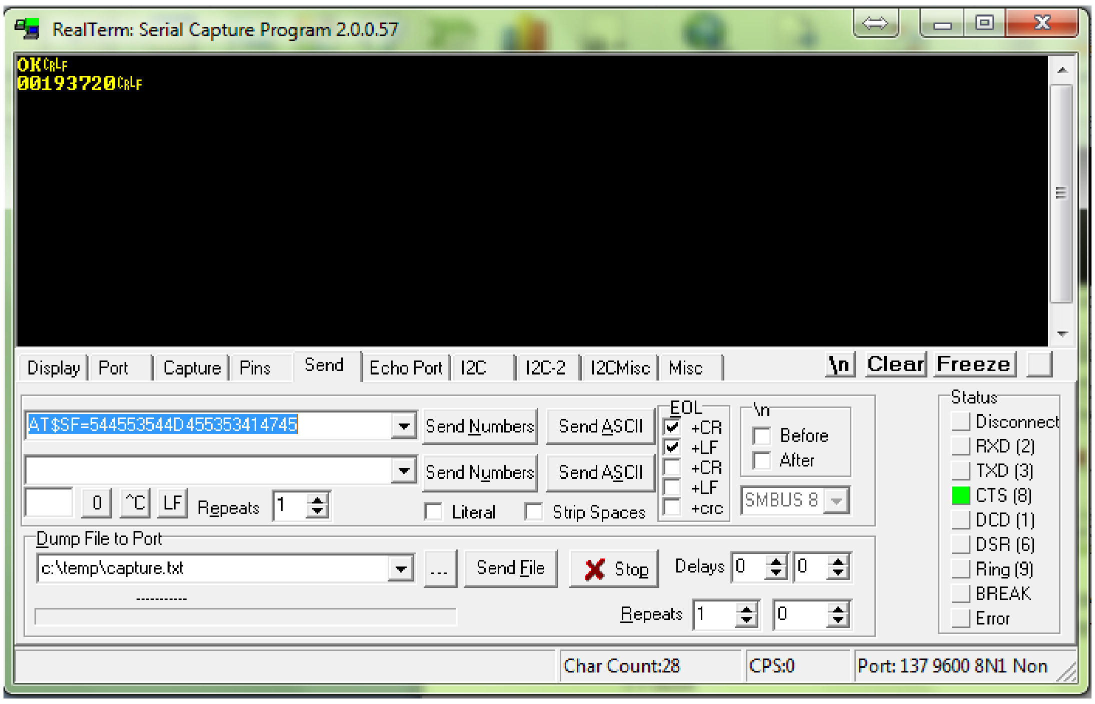
AT$I=10

EOL: [✓ +CR] [✓ +LF]

Char Count:28 | CPS:0 | Port: 137 9600 8N1 Non

5.7 Clear first text box and type AT$SF=544553544D455353414745 with +CR and +LF ticked. This is the command to send a message and the message (TESTMESSAGE) here is shown as Hex value in ascii.

T	= 0x54
E	= 0x45
S	= 0x53
T	= 0x54
M	= 0x4D
E	= 0x45
S	= 0x53
S	= 0x53
A	=0x41
G	= 0x47
E	= 0x45



RealTerm: Serial Capture Program 2.0.0.57

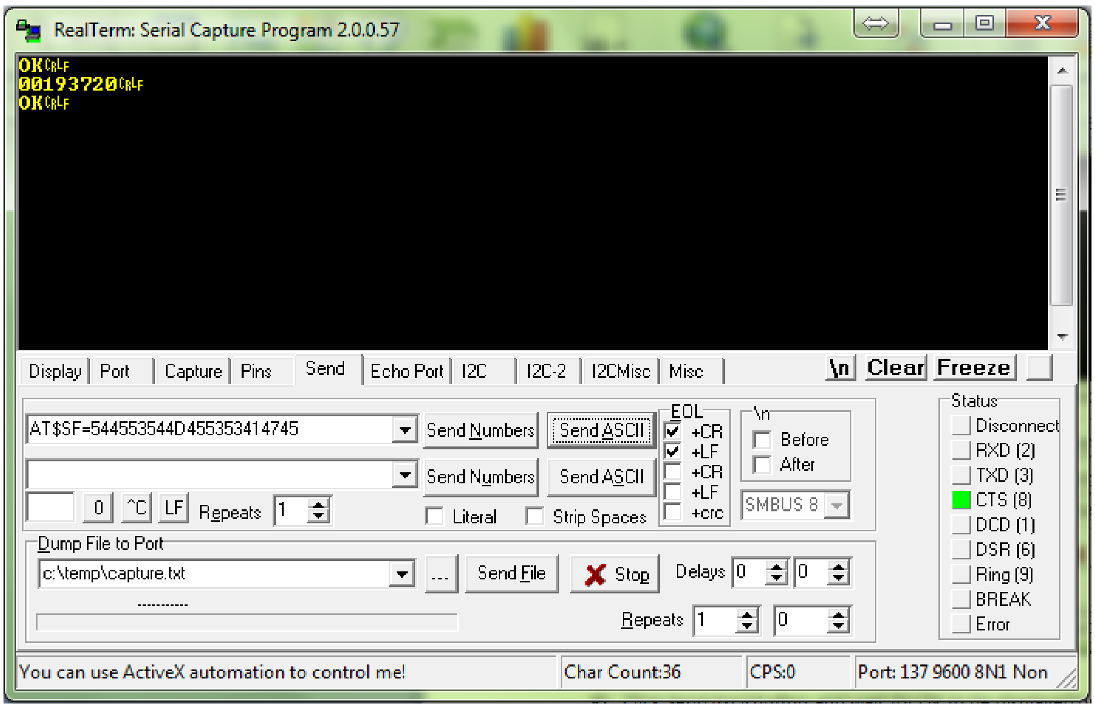
```
OK CRLF
00193720 CRLF
```

AT$SF=544553544D455353414745

c:\temp\capture.txt

Char Count:28 | CPS:0 | Port: 137 9600 8N1 Non

5.8 Click Send Ascii button and wait for OK to be displayed on black screen. It will take 10-15 seconds to get reply OK, back from module. If the module replies OK, that means the message has been delivered onto sigfox network.



RealTerm: Serial Capture Program 2.0.0.57

```
OK CRLF
00193720 CRLF
OK CRLF
```

AT$SF=544553544D455353414745

c:\temp\capture.txt

You can use ActiveX automation to control me! | Char Count:36 | CPS:0 | Port: 137 9600 8N1 Non

5.9 If the module replies, ERROR: parse error, check if the message format is not Hex. Or if the module replies ERR_SEND_FRAME_DATA_LENGTH, check if the length of the message doesn't exceed 12bytes.

5.10 If the sigfox module is licensed and registered on sigfox network, the delivered messaged can be viewed on backend sigfox portal.

6. Microchip PIC24HJ32GP302 MCU Code example of flowchart.

```
_FGS(GWRP_OFF&GCP_OFF);
_FOSCSEL(FNOSC_FRCPLL & IESO_ON);  //8Mhz
_FOSC(POSCMD_NONE & OSCIOFNC_ON &IOL1WAY_OFF & FCKSM_CSECME);
_FWDT(WINDIS_OFF & FWDTEN_OFF); //watch dog timer off
_FICD(ICS_PGD3 & JTAGEN_OFF);

#define FOSC  clock //8000000LL  // clock-frequecy in Hz with suffix LL (64-bit-long), eg.
32000000LL for 32MHz
#define FCY       (FOSC)  // MCU is running at FCY MIPS
#define delay_us(x) __delay32(((x*FCY)/1000000L)) // delays x us
 #define delay_ms(x) __delay32(((x*FCY)/1000L))  // delays x ms
 #define delay_s(x) __delay32(((x*FCY)/1L))  // delays x s

const long BaudValues[] = {1200,2400,4800,9600,19200,38400,31250,76800,115200} ;
const int UxBRG_Values[] = {416,207,103,51,25,12,15,5,16}; //(8000000/(16*baudrate))-1
//;115200 also works in 8mhz ,just needs high baud rate select
                                    //with forumal (8000000/(4*baudrate))-1 for 115200 at 8mhz

void Timer2_Init(); //V1.2
volatile unsigned long Uart_timeout;
volatile unsigned long Uart_WaitforData_Timeout;

int main(void)
{
        _PLLPOST = 3;  //to make 8Mhz
        _PLLPRE = 1;   //to make 8Mhz

        _TUN = 10;   //to make fine adjustments near to 8Mhz
        while(OSCCONbits.LOCK!=1) {};              //Wait for Oscillator


        _LPOSCEN = 0; //Disable secondary oscillator _which is on RA4 and RB4
```

//SET 9600 Baud and 8N1 Uart

```
_U2RXR0 = 0;     //RP2(00010) assigned to RPINR19 (U2RX)
                        _U2RXR1 = 1;
                        _U2RXR2 = 0;
                        _U2RXR3 = 0;
                        _U2RXR4 = 0;

            _U2CTSR0 = 1;   //RP3(00011) assigned to RPINR19 (U2CTS)
                        _U2CTSR1 = 1;
                        _U2CTSR2 = 0;
                        _U2CTSR3 = 0;
                        _U2CTSR4 = 0;

                        _RP1R0 = 1;   //RP1 as U2TX output which it should be 00101 or
_RP1R = 5
            _RP1R1 = 0;
            _RP1R2 = 1;
            _RP1R3 = 0;
            _RP1R4 = 0;

            _RP0R0 = 0; //RP0 as U2RTS which should be 00110 or _RP0R = 6
            _RP0R1 = 1;
            _RP0R2 = 1;
            _RP0R3 = 0;
            _RP0R4 = 0;

        _UARTEN = 0;  //UARTx is enabled; all UARTx pins are controlled by UARTx as
        defined by UEN<1:0>
        _UTXEN = 0;   //Transmit is enabled, UxTX pin is controlled by UARTx


        _USIDL = 0; //Continues module operation in Idle mode
        _UEN1 = 0;  //UxTX, UxRX, UxCTS and UxRTS pins are enabled and used
        _UEN0 = 0; //UxTX, UxRX, UxCTS and UxRTS pins are enabled and used
        _BRGH = 0; //BRG generates 16 clocks per bit period
        _PDSEL0 = 0;  //8-bit data, no parity
        _PDSEL1 = 0;  //8-bit data, no parity
        _STSEL = 0;   //One Stop bit
        _RTSMD = 0;
        _IREN = 0; //IRDA is dsabled

        U1BRG = UxBRG_Values[0];
```

```
_UTXISEL0 = 0;   //Interrupt when a character is transferred to the Transmit Shift Register
(TSR) and as a result, the
                    //transmit buffer becomes empty
            _UTXISEL1 = 1;   //Interrupt when a character is transferred to the Transmit Shift
Register (TSR) and as a result, the
                    //transmit buffer becomes empty
            _URXISEL0 = 0;   //Interrupt is set when any character is received and transferred from
the RSR to the receive buffer;
                     //receive buffer has one or more characters
            _URXISEL1 = 0;   //Interrupt is set when any character is received and transferred from
the RSR to the receive buffer;
                    //receive buffer has one or more characters


            _UARTEN = 1;  //UARTx is enabled; all UARTx pins are controlled by UARTx as defined
by UEN<1:0>
            _UTXEN = 1;   //Transmit is enabled, UxTX pin is controlled by UARTx
```

//Send AT command

```
While(1)
{
char AT[] = {'A','T',0x0D,0x0A};
i=0;
while(i<4)
{
              while(U2STAbits.UTXBF == 1);
            U2TXREG = AT[i++];  //send one byte of data into Tx reg. Send AT\r  to module to find
module
}
                                    while(U2STAbits.TRMT == 0);//Transmission is in progress... wait

                                    i=0;
                                    Uart_WaitforData_Timeout = 0;
//                                  _T2IE = 1;
                                    while(!(U2STAbits.URXDA)) //wait for reply from module
{
                                                        if(Uart_WaitforData_Timeout>Uart_timeout)
                                                         break;
```

```
}
                              i = 0;
                              while((U2STAbits.URXDA))
                              {
                                             Data[i++] = U2RXREG;//Send received data to PC
                                Uart_WaitforData_Timeout = 0;
                                while(!(U2STAbits.URXDA))
                                {

                                 if(Uart_WaitforData_Timeout>Uart_timeout)
                                             break;
                   }
                              }
                              if(Data[0]== 'O' && Data[1]=='K')
                              {
                                Break; //It is Sigfox
                              }
}
        Timer2_Stop();//V1.4
       Timer2_Reset();//V1.4
```

//Send TESTMESSAGE every 15minutes

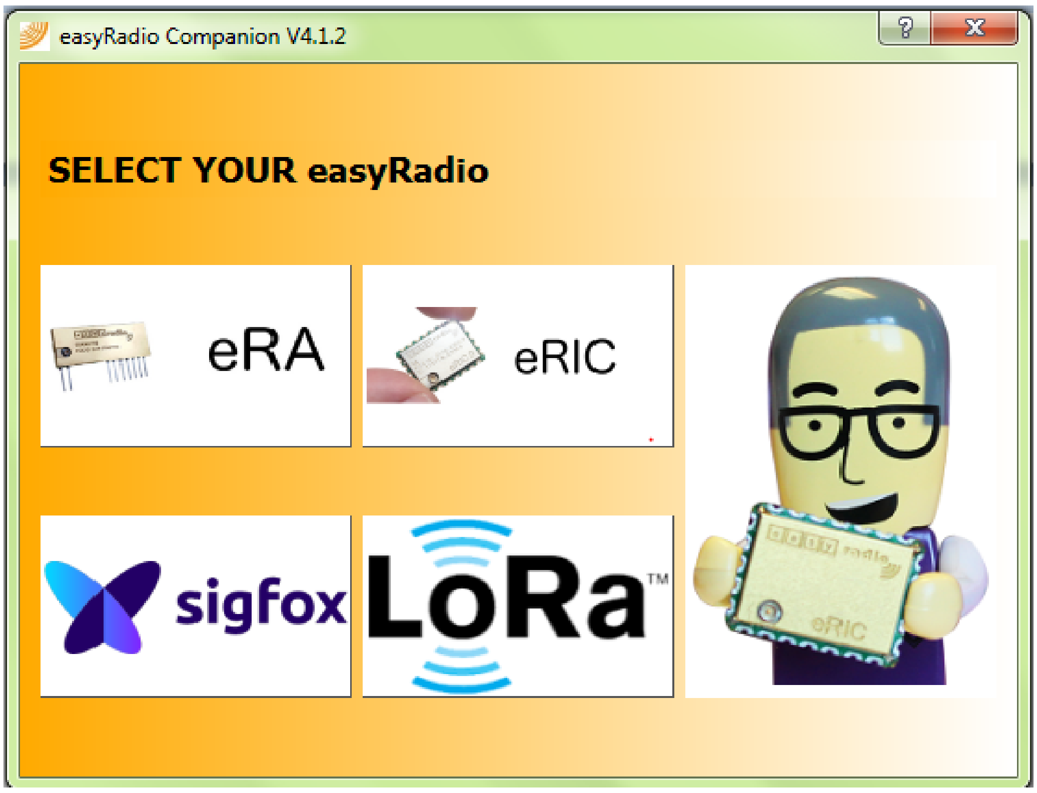
```
while(1)
{
char SigfoxCommand[] = {'A','T','$','S','F','='
,’5’,’4’,’4’,’5’,’5’,’3’,’5’,’4’,’4’,’D’,’4’,’5’,’5’,’3’,’5’,’3’,’4’,’1’,’4’,’7’,’4’,’5’};
 j = 0;
 while(j<28)
 {
  while(U2STAbits.UTXBF == 1);
 U2TXREG = SigfoxCommand[j++];  //send received data from GPS into Module Tx reg
}
 while(U2STAbits.UTXBF == 1);
 U2TXREG = 0x0D;//Send cr lf
 while(U2STAbits.UTXBF == 1);
 U2TXREG = 0x0A;
 while(U2STAbits.TRMT == 0);//Transmission is in progress... wait
 delay_s(900);//900seconds delay or 15mins
}
}
```

```
void Timer2_Init()
{
 T2CONbits.TCKPS1 = 0;//1;
 T2CONbits.TCKPS0 = 0;//Dont divide 8000000//1; //Divide by
256;8000000/256 = 31250
 T2CONbits.TCS = 0; //Internal clock


  PR2 = 1000; //1000cycles changed to 1000 from 100 in V1.4
  Timer2_Reset();//TMR2 = 0;
  Timer2_ClearFlag();//_T2IF = 0;
  Timer2_Start();//T2CONbits.TON = 1;
  Timer2_InterruptEnable();//_T2IE = 1;

}

void __attribute__((interrupt, auto_psv)) _T2Interrupt()
{
        Timer2_ClearFlag();//THis needs to be here important
        Uart_WaitforData_Timeout++; //For uart increment
}
```

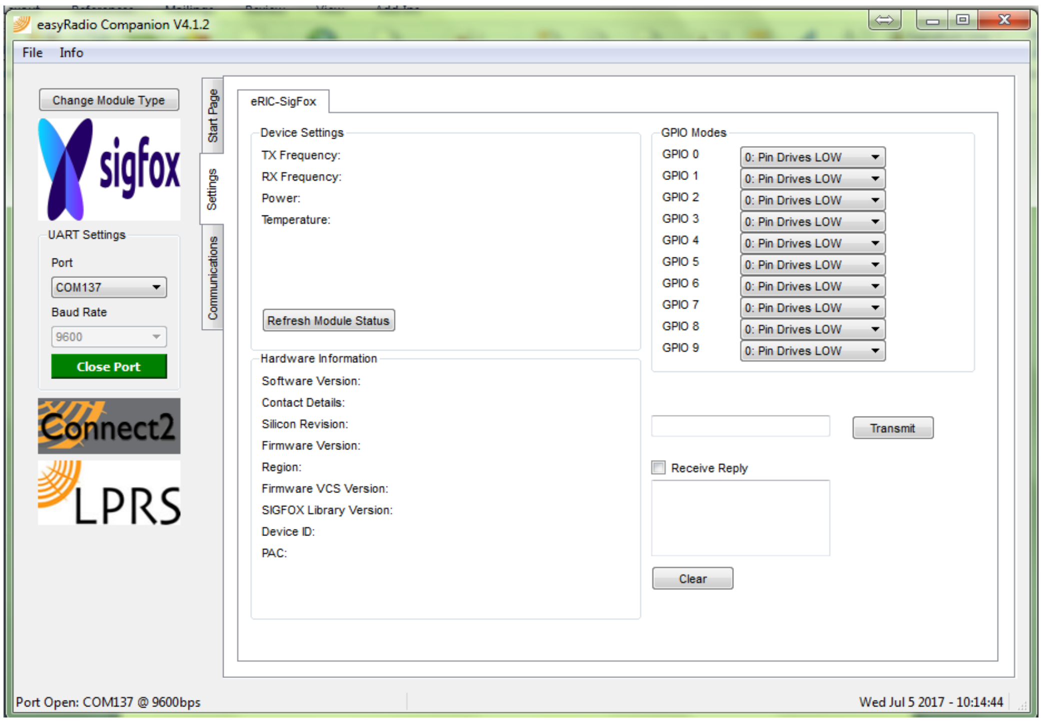
********Code example finish*********

7. Using LPRS's easyRadio Companion software to send data.

7.1 Open latest easyRadio companion software and click Sigfox Icon.



7.2 Select port and click Open Port button.



easyRadio Companion V4.1.2

File Info

Change Module Type

UART Settings

Port: COM137

Baud Rate: 9600

Close Port

Start Page | Settings | Communications

eRIC-SigFox

Device Settings

TX Frequency:
RX Frequency:
Power:
Temperature:

Refresh Module Status

Hardware Information

Software Version:
Contact Details:
Silicon Revision:
Firmware Version:
Region:
Firmware VCS Version:
SIGFOX Library Version:
Device ID:
PAC:

GPIO Modes

GPIO 0	0: Pin Drives LOW
GPIO 1	0: Pin Drives LOW
GPIO 2	0: Pin Drives LOW
GPIO 3	0: Pin Drives LOW
GPIO 4	0: Pin Drives LOW
GPIO 5	0: Pin Drives LOW
GPIO 6	0: Pin Drives LOW
GPIO 7	0: Pin Drives LOW
GPIO 8	0: Pin Drives LOW
GPIO 9	0: Pin Drives LOW

Transmit

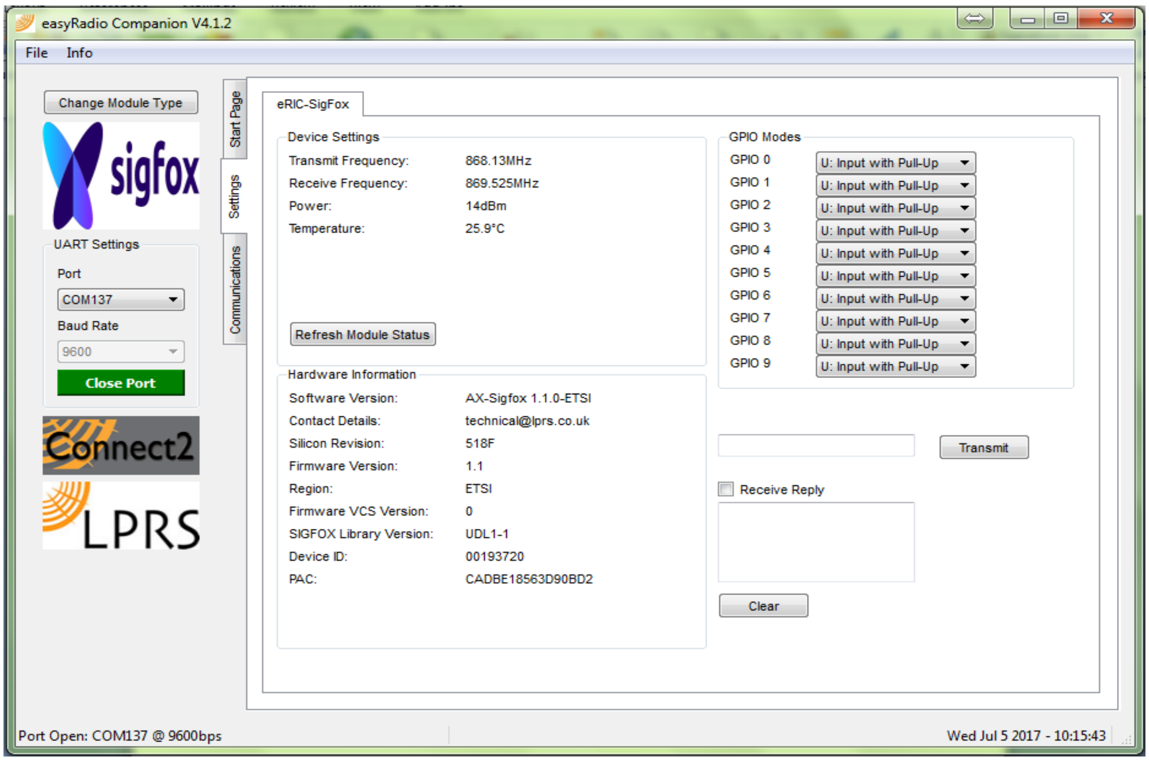
Receive Reply

Clear

Port Open: COM137 @ 9600bps

Wed Jul 5 2017 - 10:14:44

7.3 Click Refresh Module Status. Sigfox module details will be displayed.



easyRadio Companion V4.1.2

File Info

Change Module Type

UART Settings

Port: COM137

Baud Rate: 9600

Close Port

eRIC-SigFox

Device Settings

Transmit Frequency:	868.13MHz
Receive Frequency:	869.525MHz
Power:	14dBm
Temperature:	25.9°C

Refresh Module Status

Hardware Information

Software Version:	AX-Sigfox 1.1.0-ETSI
Contact Details:	technical@lprs.co.uk
Silicon Revision:	518F
Firmware Version:	1.1
Region:	ETSI
Firmware VCS Version:	0
SIGFOX Library Version:	UDL1-1
Device ID:	00193720
PAC:	CADBE18563D90BD2

GPIO Modes

GPIO 0	U: Input with Pull-Up
GPIO 1	U: Input with Pull-Up
GPIO 2	U: Input with Pull-Up
GPIO 3	U: Input with Pull-Up
GPIO 4	U: Input with Pull-Up
GPIO 5	U: Input with Pull-Up
GPIO 6	U: Input with Pull-Up
GPIO 7	U: Input with Pull-Up
GPIO 8	U: Input with Pull-Up
GPIO 9	U: Input with Pull-Up

Transmit

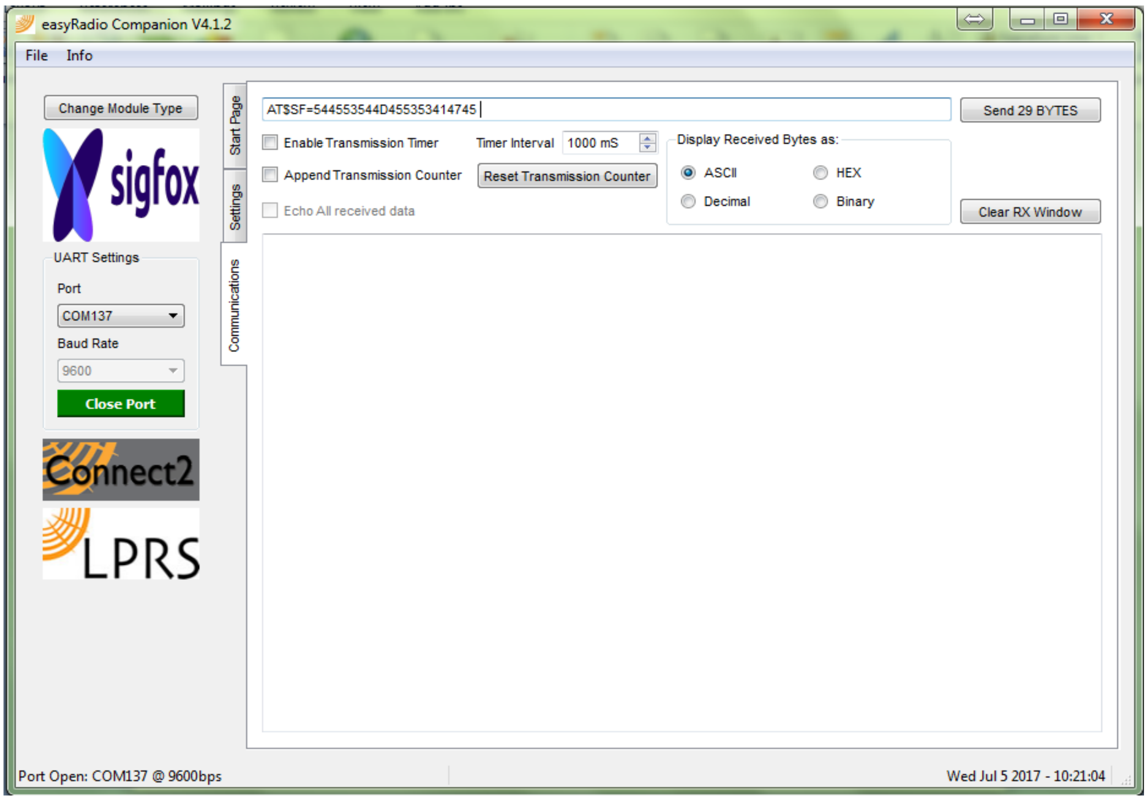
Receive Reply

Clear

Port Open: COM137 @ 9600bps   Wed Jul 5 2017 - 10:15:43

7.4 Click communications tab. Clear first text box and type:
AT$SF=544553544D455353414745 <u>*with*</u> a space at the end.

Note: This message payload is 22bytes in ASCII and 11bytes in HEX....just fitting within the 12byte maximum payload for Sigfox.



easyRadio Companion V4.1.2

File Info

Change Module Type

UART Settings

Port: COM137

Baud Rate: 9600

Close Port

```
AT$SF=544553544D455353414745 
```

Send 29 BYTES

Enable Transmission Timer   Timer Interval 1000 mS

Append Transmission Counter   Reset Transmission Counter

Echo All received data

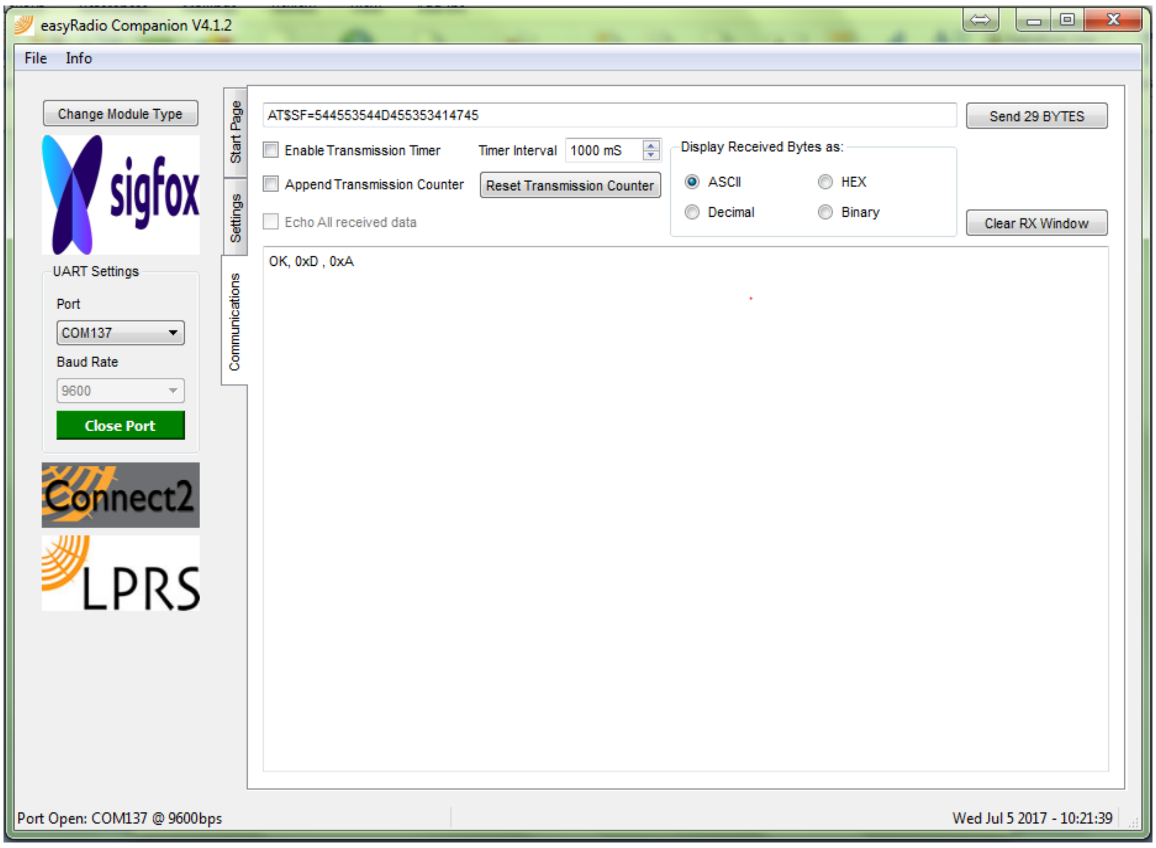
Display Received Bytes as: [✓ ASCII] HEX Decimal Binary

Clear RX Window

Port Open: COM137 @ 9600bps   Wed Jul 5 2017 - 10:21:04

7.5 Click Send 29BYTES button. After 10-15 seconds, the module will reply OK. The sigfox message is delivered to sigfox network.

Note: 29BYTES is the size of issuing the whole AT command + payload to the eRIC module in ASCII, the Sigfox payload is 11 bytes.



```
ATSSF=544553544D455353414745
```

Port: COM137
Baud Rate: 9600
[ ] Enable Transmission Timer — Timer Interval: 1000 mS
[ ] Append Transmission Counter
[ ] Echo All received data
Display Received Bytes as: [✓ ASCII] HEX Decimal Binary

```
OK, 0xD , 0xA
```

Port Open: COM137 @ 9600bps — Wed Jul 5 2017 - 10:21:39

8. Conclusion

Congratulations, you have just sent some data using the Sigfox network, happy developing!

If you have any questions or problems operating this demo / the module etc, please do not hesitate to contact the LPRS team or your local distributor for assistance, we welcome the opportunity to help.