

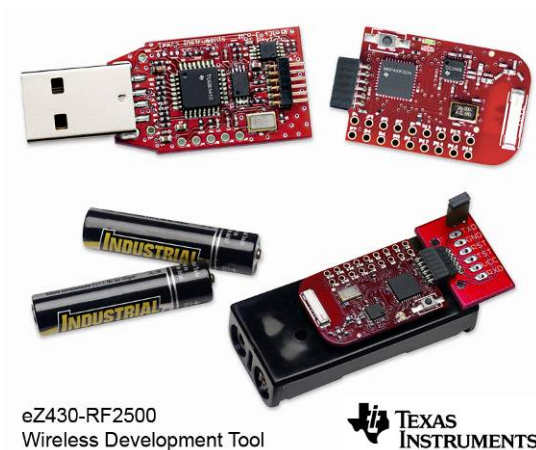


General Radio 1 Day Workshop, Lab Guide

Getting Started: What do you need?

Hardware: *eZ430-RF2500 Evaluation Board*

<http://focus.ti.com/docs/toolsw/folders/print/eZ430-rf2500.html>



eZ430-RF2500
Wireless Development Tool

Software:

1. IAR Embedded Workbench MSP430

<http://supp.iar.com/Download/SW/?item=EW430-EVAL>



2. SmartRF Studio

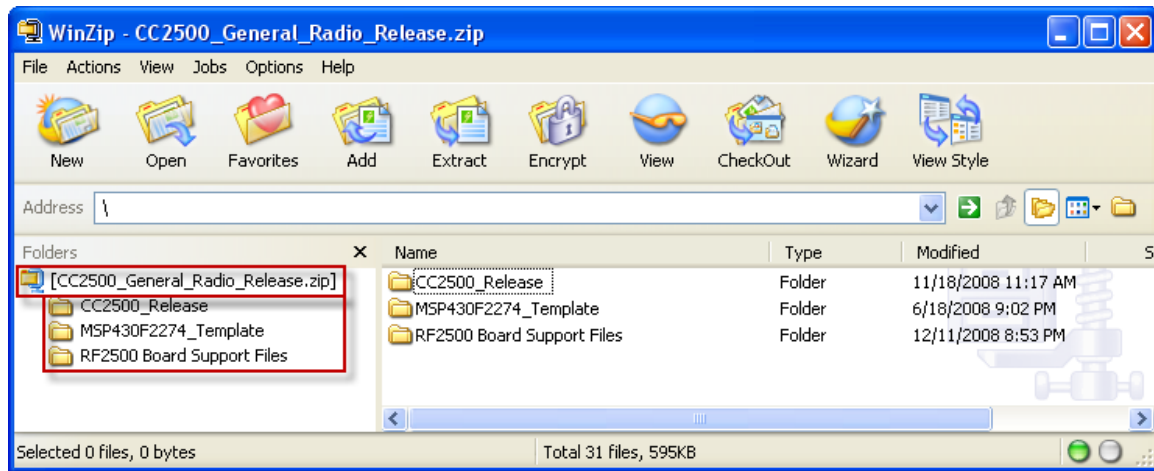
<http://focus.ti.com/docs/toolsw/folders/print/smartrftm-studio.html>

Installation Order

1. Install IAR Studio
2. Install SmartRF Studio
3. Extract CC2500_General_Radio_Release.zip – Use Default Location
4. Plug in eZ430-RF2500 Board – Driver should be found. If it isn't refer to the device driver section further in this document.

Unzipping the Code Files

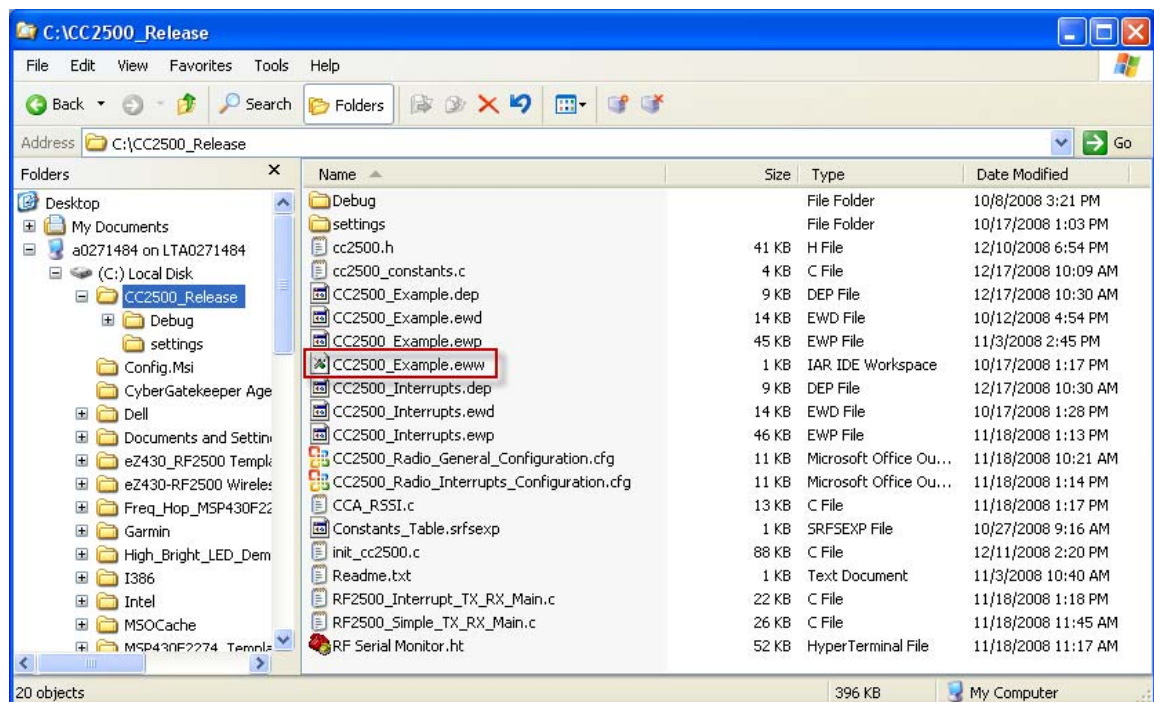
Here is the content of CC2500_General_Radio_Release.zip (Hit View Style Button for this View):



Two file folders are installed (if default C:\ location is used):

1. *CC2500_Release* in the root C directory
2. *MSP430F2274_Template* in the root C Directory
3. *RF2500 Board Support Files* in the root C Directory

Once the .zip file is extracted, an IAR Embedded Workspace is located in the C:\CC2500_Release directory, **CC2500_Example.eww**

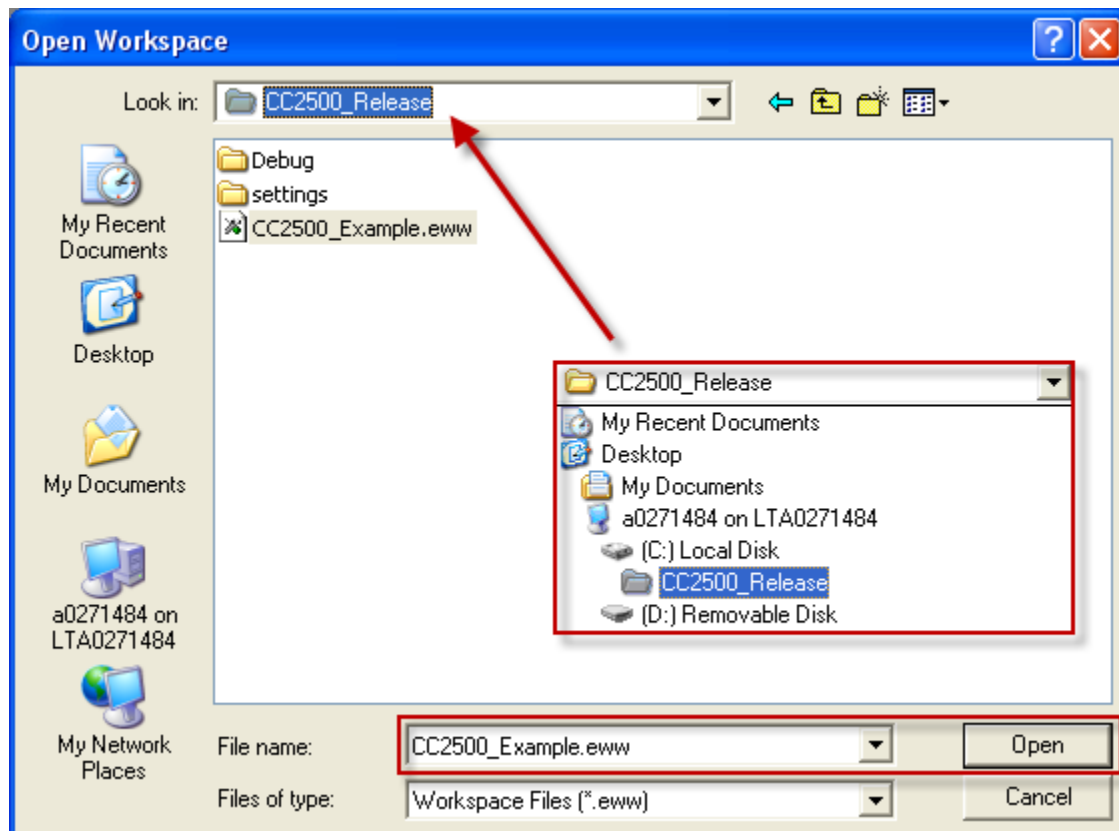
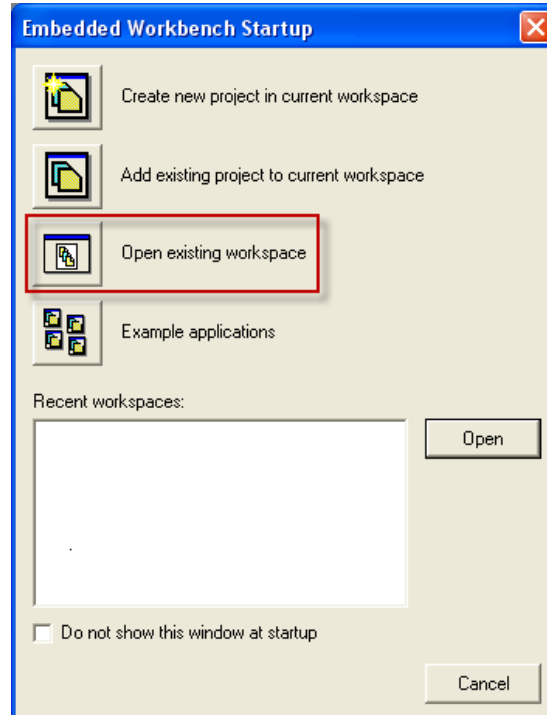




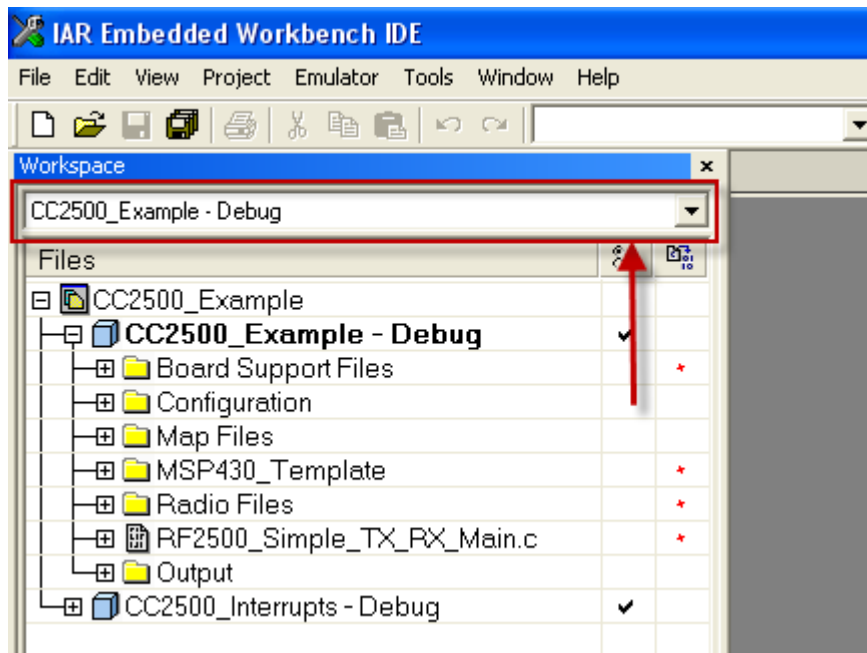
Getting Started with IAR Studio

IAR EW can be found here:

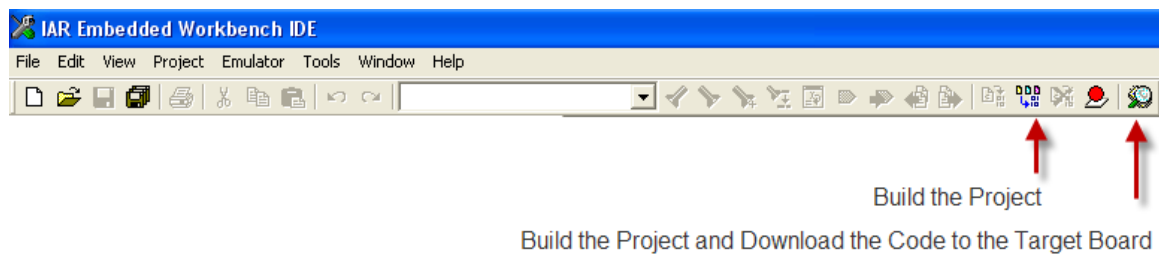
Start->All Programs->IAR Systems->IAR Embedded Workbench for MSP430 V4->IAR Embedded Workbench



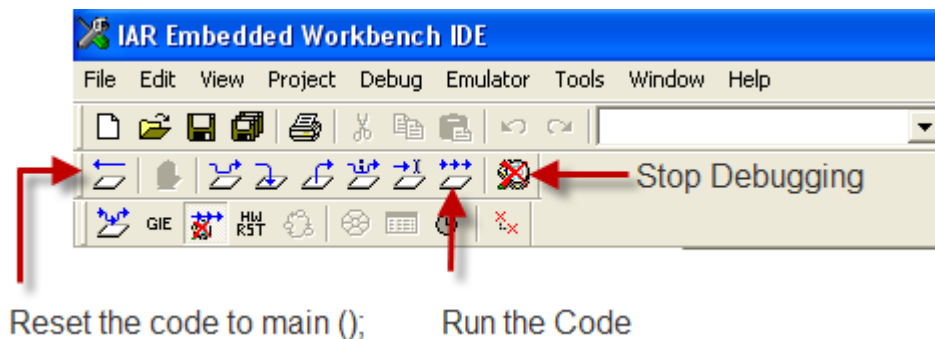
Two projects exist in this workspace. Use the pull down box shown below to change the project.



To build the project or debug the project (downloads code into the board*):



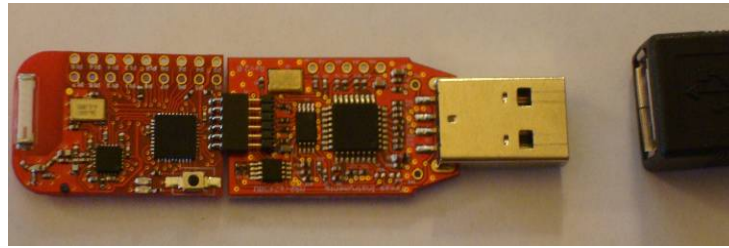
Use these Icons to control your code:



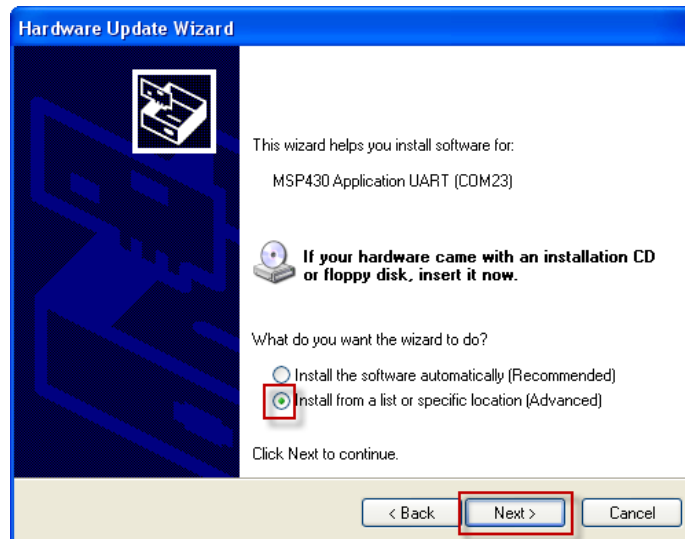
* Board needs to be attached to the FET Tool to Download Code



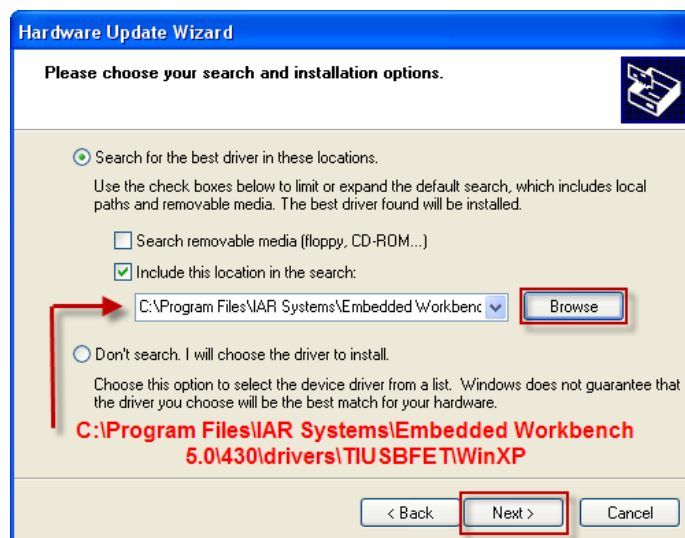
Connecting the eZ430-RF2500 Board



Plug the eZ430-RF2500 into the USB Port. The Windows new device driver wizard appears and should find the driver. If the driver is **NOT FOUND**:

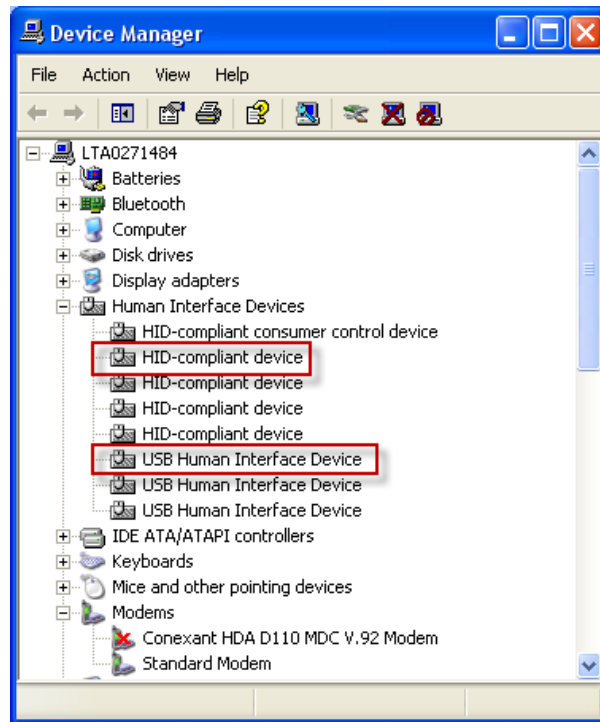


Select install from a specific location, add the path below into the include path.

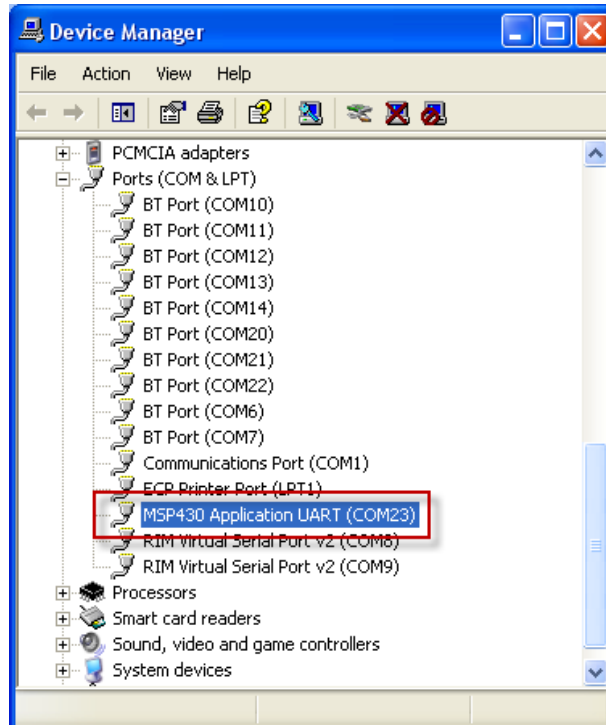


eZ430-RF2500 Board Driver Locations

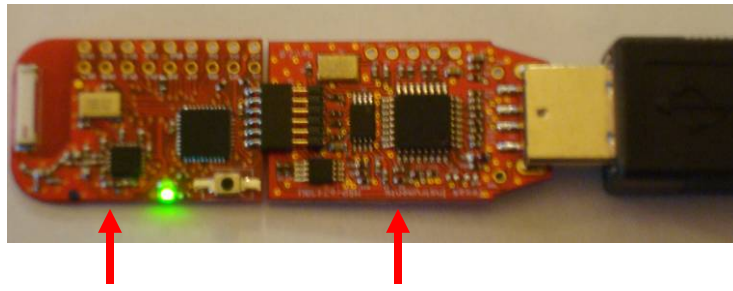
Two USB drivers are installed. The 1st is a Debug Chain Human Interface Device (HID):



The 2nd is a Back Channel UART Com Port:



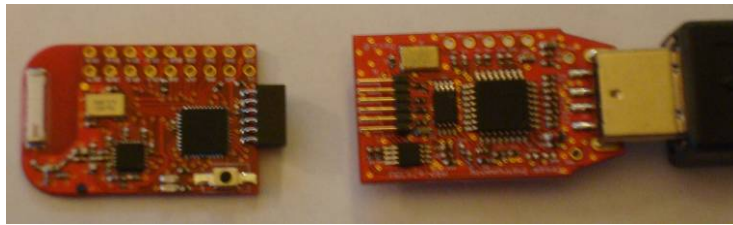
To Perform a HW Reset the eZ430-RF2500



RF2500 Target Board

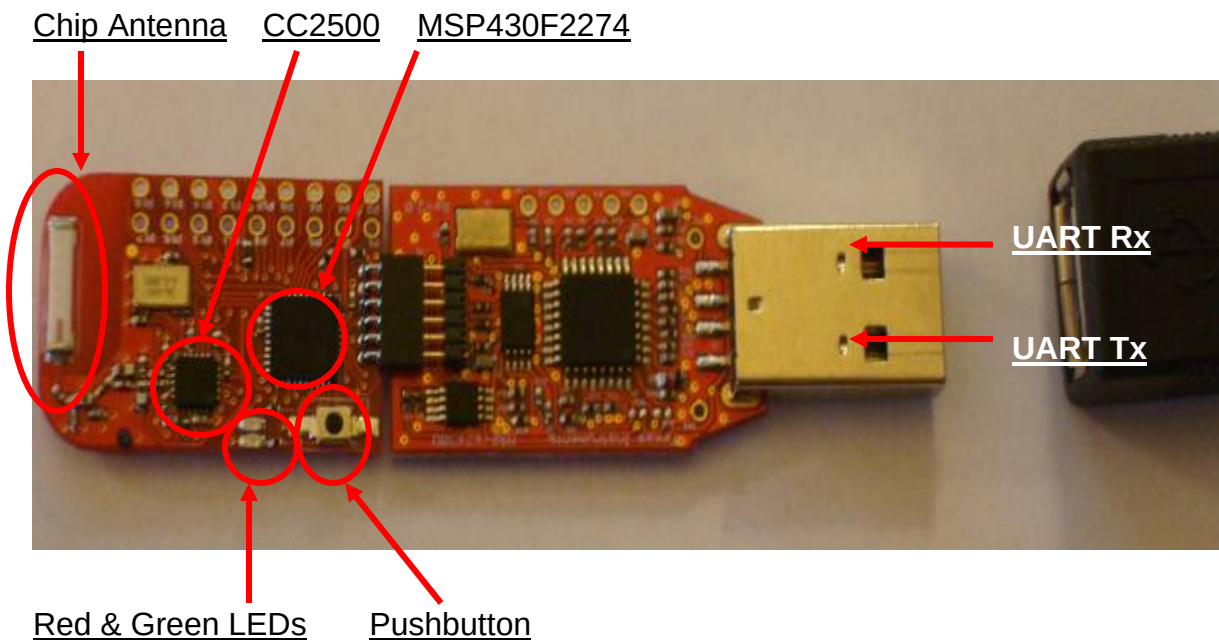
eZ430 Debug & Download

To reset the RF2500 board unplug it from the eZ430 while keeping the eZ430 board plugged into the USB port.



This will maintain the COM connection to your terminal application while still resetting the RF2500 application.

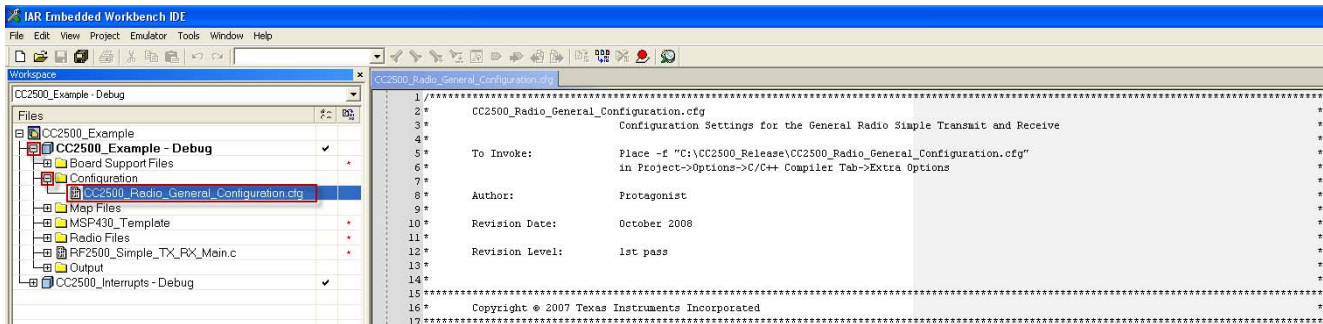
RF2500 Board features:



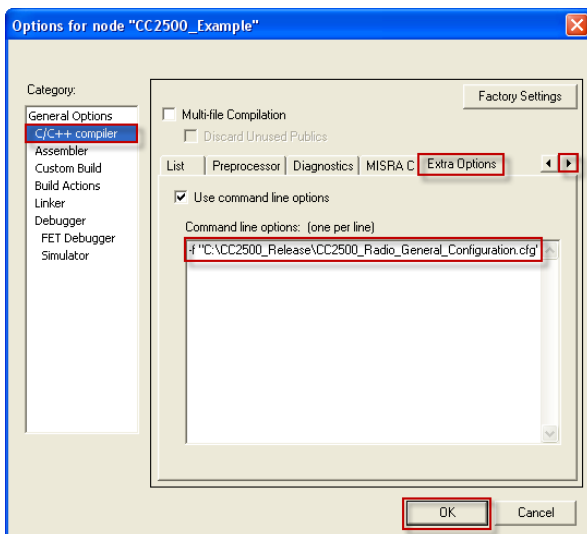
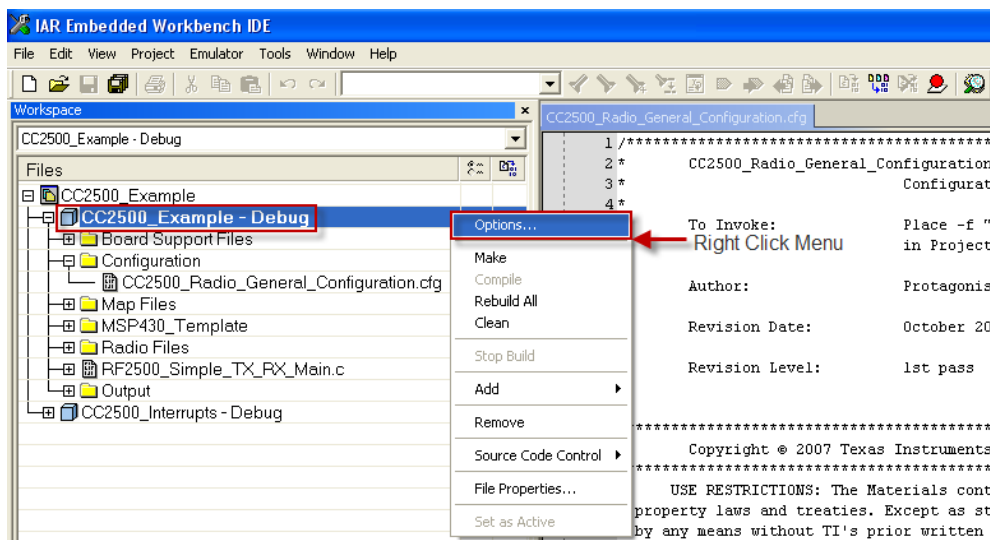


Setting the Configuration File

Under the Configuration Group, a configuration file resides. This file is used to set up the payload and general functions of the radio.



To instantiate the .cfg files into the C program, set the file include in the extra options tab in Project Options. To get the project options, click on the CC2500_Example-Debug and right click to give the menu below, selecting options:



Select the C/C++ compiler category, and hit the right arrow button until the Extra Options Tab appears.

Adding the -f "<configuration file name>" will include that file in the build.

Check the box next to *Use command line options*

If the .cfg file and the .dat files are kept in the My_Design directory (or the same directory that your project is saved in), \$PROJ_DIR\$ can be used instead of the full directory path name.

An overview of the CC2500 Radio General Configuration.cfg

Options:

```
/******
  Use GDO Pin 0 function defined in init_GDO_0
  *****/
-DUSE_GDO_0
1.  Instantiating this code (removing the "/" in front of it) will set up
    the GPIO interrupt related to GDO_0 on the CC2500.  In this code
    example GDO_0 signals a packet transmitted or received.
//-DUSE_GDO_0
2.  Remarking this code removes the GDO_0 interrupt.  In this code
    example, removing the GDO_0 interrupt requires the state machine from
    the CC2500 read on the SPI port is needed to signal a packet
    transmitted or received.

/******
  Use GDO Pin 2 function defined in init_GDO_2
  *****/
-DUSE_GDO_2
1.  Instantiating this code (removing the "/" in front of it) will set up
    the GPIO interrupt related to GDO_2 on the CC2500.  In this code
    example GDO_2 signals a PLL Lock.
//-DUSE_GDO_2
2.  Remarking this code removes the GDO_0 interrupt.  In this code
    example, removing the GDO_2 interrupt requires the polling of the
    CC2500 to signal the PLL is locked.

/******
  Set Radio to embed length byte into the payload, position 1
  *****/
-DVARIABLE_PACKET
1.  Instantiating this code (removing the "/" in front of it) will enable
    the variable packet function in the CC2500 radio.
//-DVARIABLE_PACKET
2.  Remarking this code removes the variable packet function and will
    default to the Smart RF Studio setting if the -DFIXED_PACKET below is
    not instantiated.

/******
  Set Radio to use PKTLEN register to set the payload size, maximum of 64
  to fit in FIFO
  *****/
-DFIXED_PACKET
1.  Instantiating this code (removing the "/" in front of it) will enable
    the Fixed packet function in the CC2500 radio.  The Fixed packet
    length is defined by the PKTLEN register from Smart RF Studio.
//-DFIXED_PACKET
2.  Remarking this code removes the fixed packet function and will
    default to the Smart RF Studio setting if the -DVARIABLE_PACKET above
    is not instantiated.

/******
```

Options Continued:

```

/*****
  Embeds Target Address into the payload, position 2
  *****/
-DALLOW_ADDRESSING
1.  Instantiating this code (removing the "/" in front of it) will enable
    the addressing of the payload. The Address used by the devices is
    defined by the ADDR register in the CC2500.
//-DALLOW_ADDRESSING
2.  Remarking this code removes the addressing function from the CC2500
    payload.

/*****
  Embeds LQI and RSSI Information into the payload, last two positions
  *****/
-DAPPEND_STATUS
1.  Instantiating this code (removing the "/" in front of it) will enable
    the appending of the RSSI and LQI information of the received packet
    onto the last two bytes of the packet.
//-DAPPEND_STATUS
2.  Remarking this code removes the appending of the LQI/RSSI function
    from the CC2500 payload

/*****
  Radio initializes in the IDLE state, this forces the Radio into a
  Transmit or Receive state
  *****/
-DFORCE_TRANSMIT
//-DFORCE_RECEIVE
1.  Instantiating this code (removing "/" in front of it) will force the
    radio in transmit mode at initialization.
//-DFORCE_TRANSMIT
-DFORCE_RECEIVE
2.  Instantiating this code (removing "/" in front of it) will force the
    radio in receive mode at initialization.
//-DFORCE_TRANSMIT
//-DFORCE_RECEIVE
3.  Remarking this code allows transmit or receive to be decided in code.

/*****
  Enables Output to hyperterminal
  *****/
-DTERMINAL
1.  Instantiating this code (removing "/" in front of it) will force the
    UART to output debugging and packet information to be read on a
    terminal application.
//-DTERMINAL
2.  Remarking this code removes the output to the terminal and saves code
    space.
```

Options Continued:

```

/*****
  Enables the CCxxxx Verify Read after Write - Disabling saves code space
*****/
-DVERIFY
1.  Instantiating this code (removing "/" in front of it) will force the
    CC2500 device to read back the code and verify it's contents were
    correctly programmed.

//-DVERIFY
2.  Remarking this code will prevent verification and saves code space.

/*****
  Enables the CCxxxx Debug Info - Disabling saves code space
*****/
-DDEBUG
1.  Instantiating this code (removing "/" in front of it) will force the
    functions to produce debugging code that can help troubleshoot the
    application.

//-DDEBUG
2.  Remarking this code will prevent the debugging and save code space.

/*****
  Change the Radio to Max RF Power
*****/
-DMAX_POWER
1.  Instantiating this code (removing "/" in front of it) will force the
    maximum power output from the radio.

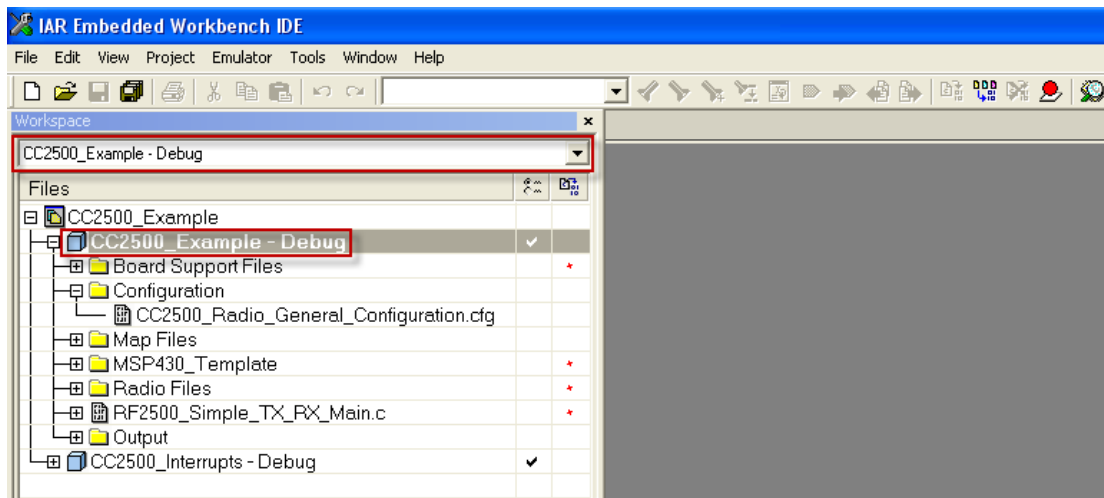
//-DMAX_POWER
2.  Remarking this code will use the power settings from Smart RF Studio.
```



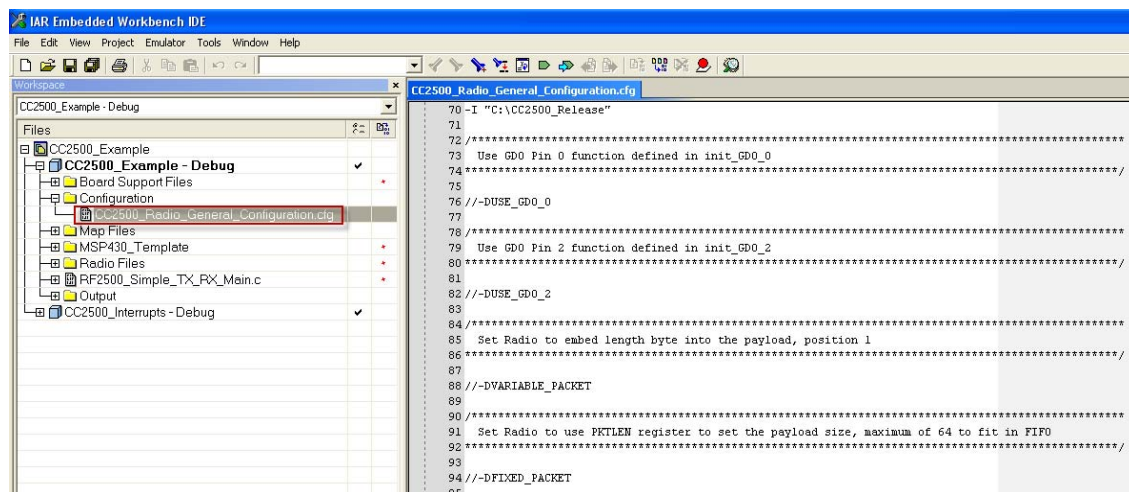
Lab 1: End Device to End Device

Introduction: This lab shows simple point to point communication.

1. Make sure the CC2500_Example – Debug project is selected:

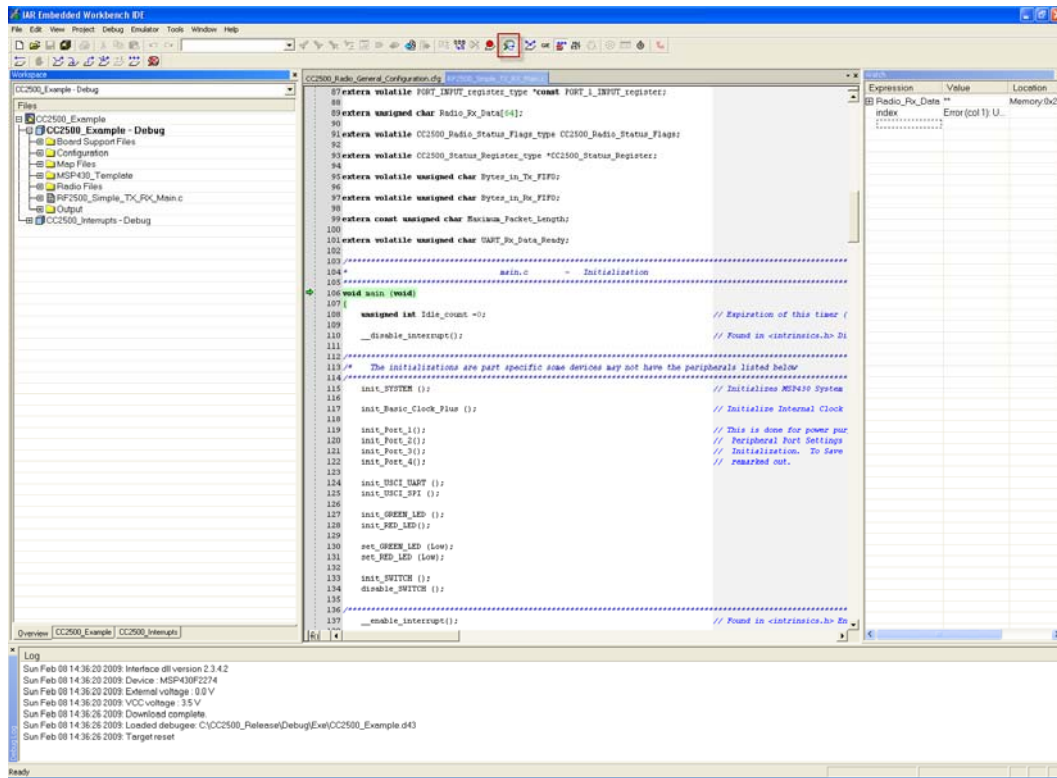


2. Under the CC2500_Example – Debug Project, Open CC2500_Radio_General_Configuration.cfg and set the options accordingly:

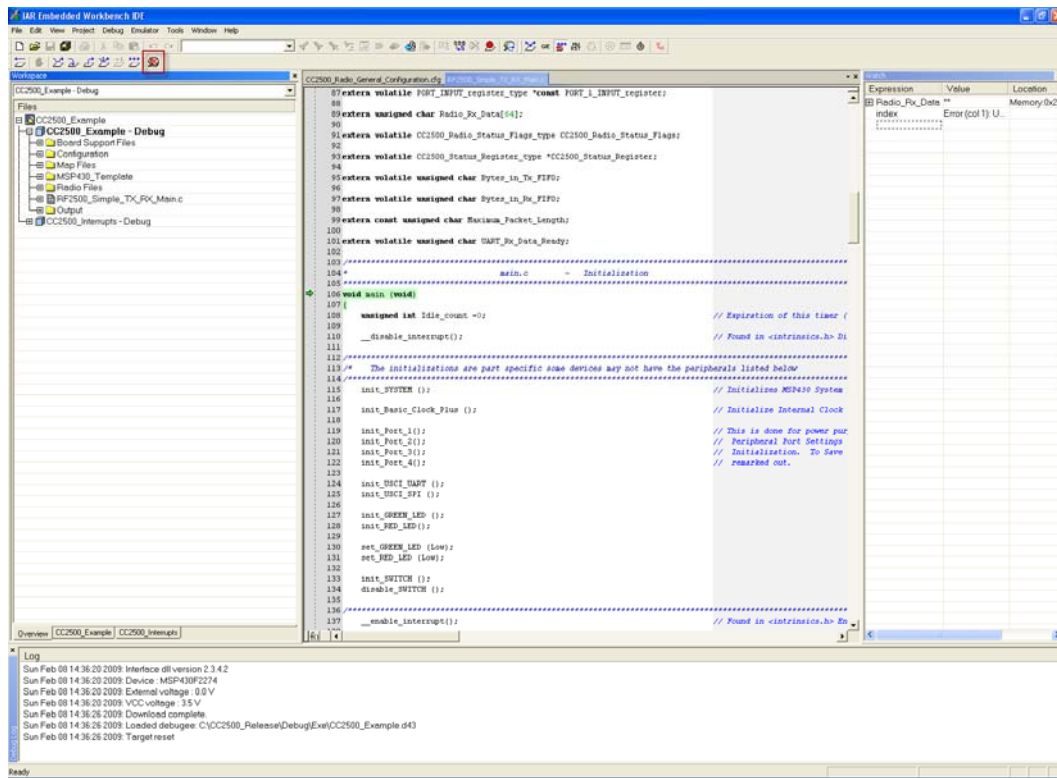


```
// -DUSE_GDO_0
// -DUSE_GDO_2
// -DARIABLE_PACKET
-DFIXED_PACKET
// -DALLOW_ADDRESSING
// -DAPPEND_STATUS
// -DFORCE_TRANSMIT
// -DFORCE_RECEIVE
-DTERMINAL
// -DVERIFY
// -DDEBUG
// -DMAX_POWER
```

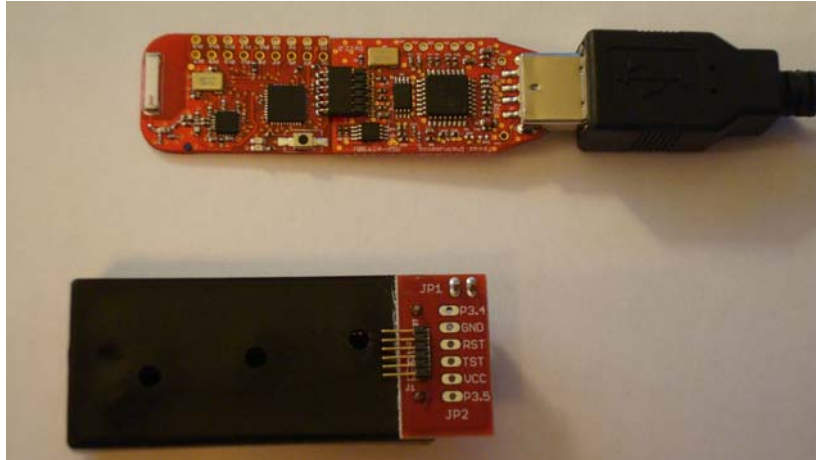
5. Build and Download the code:



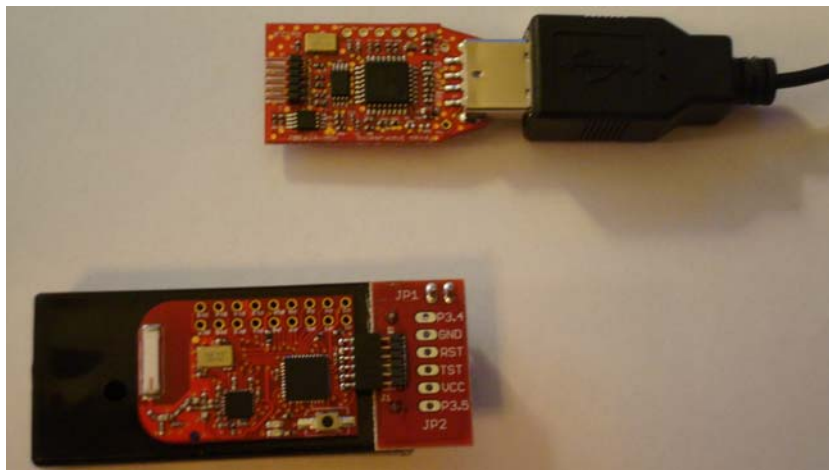
5. Stop Debugging:



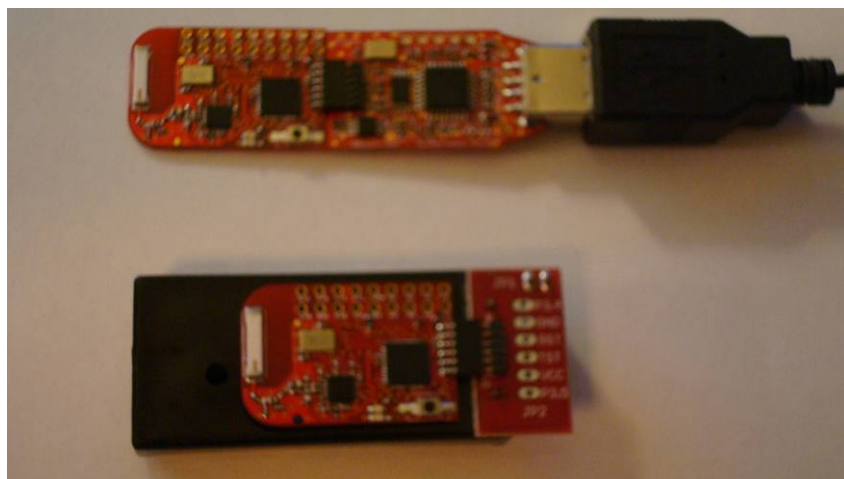
6. Remove the Programmed RF2500 Board from the eZ430:



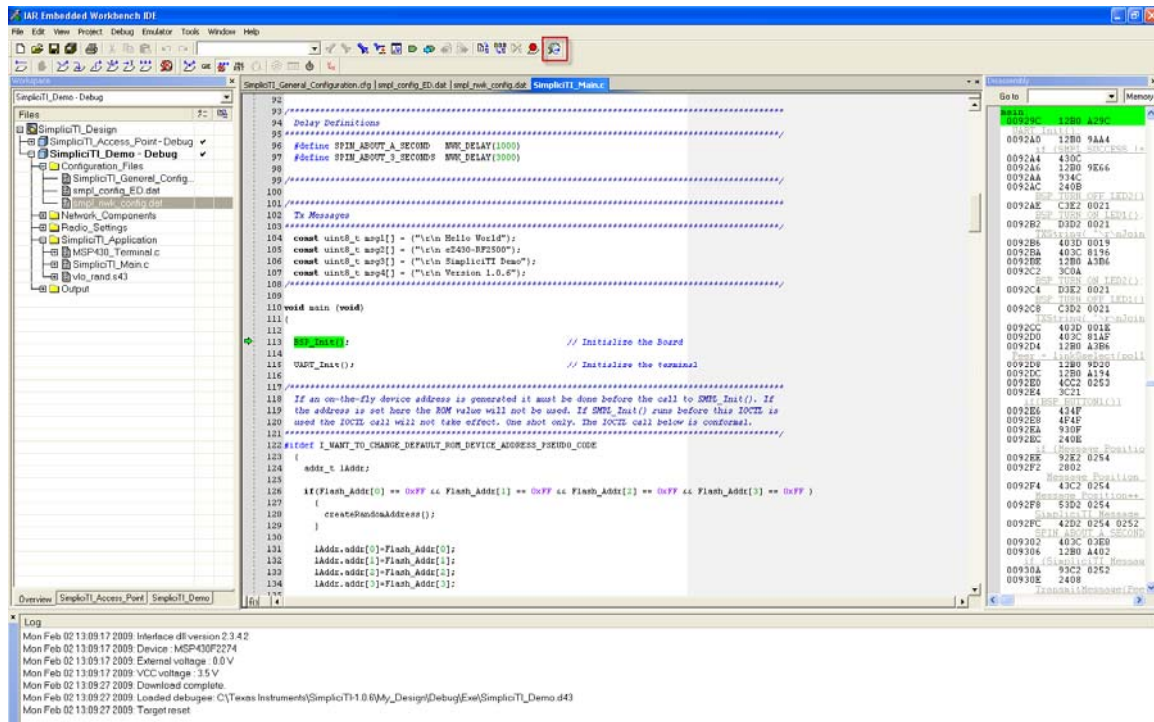
7. Put the Programmed RF2500 Board onto the Battery Board:



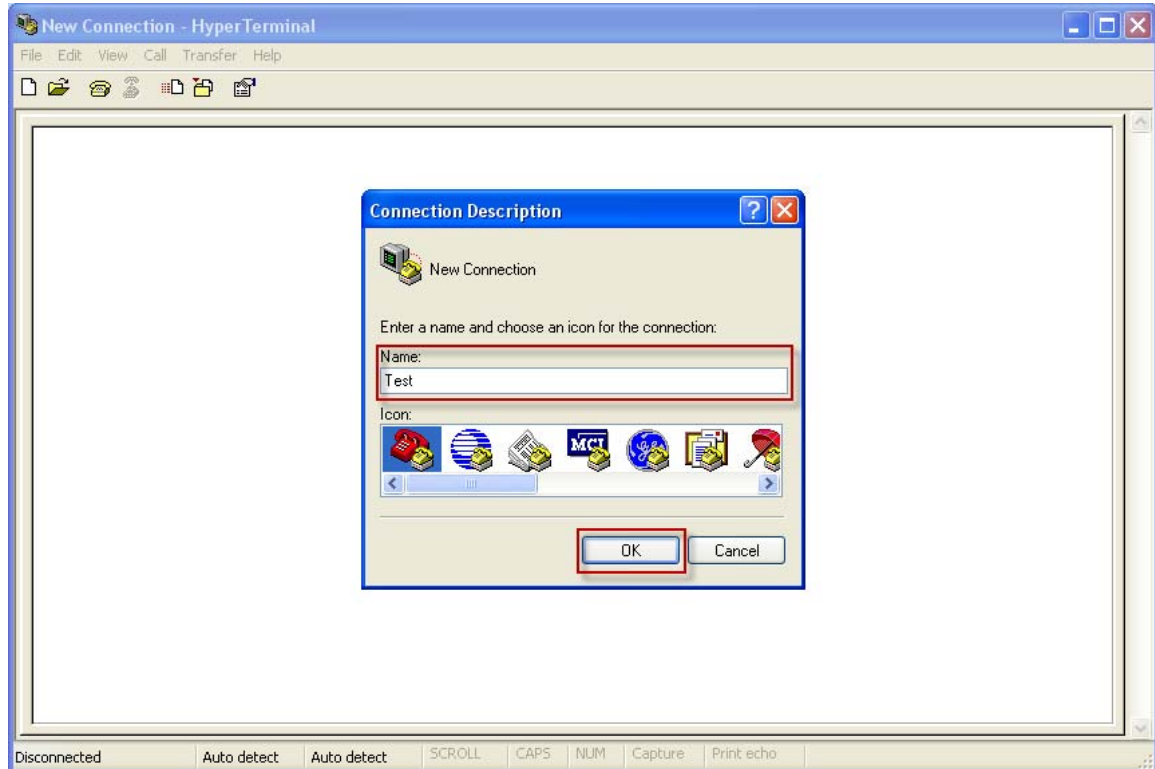
8. Put the un-programmed RF2500 Board onto the eZ430:



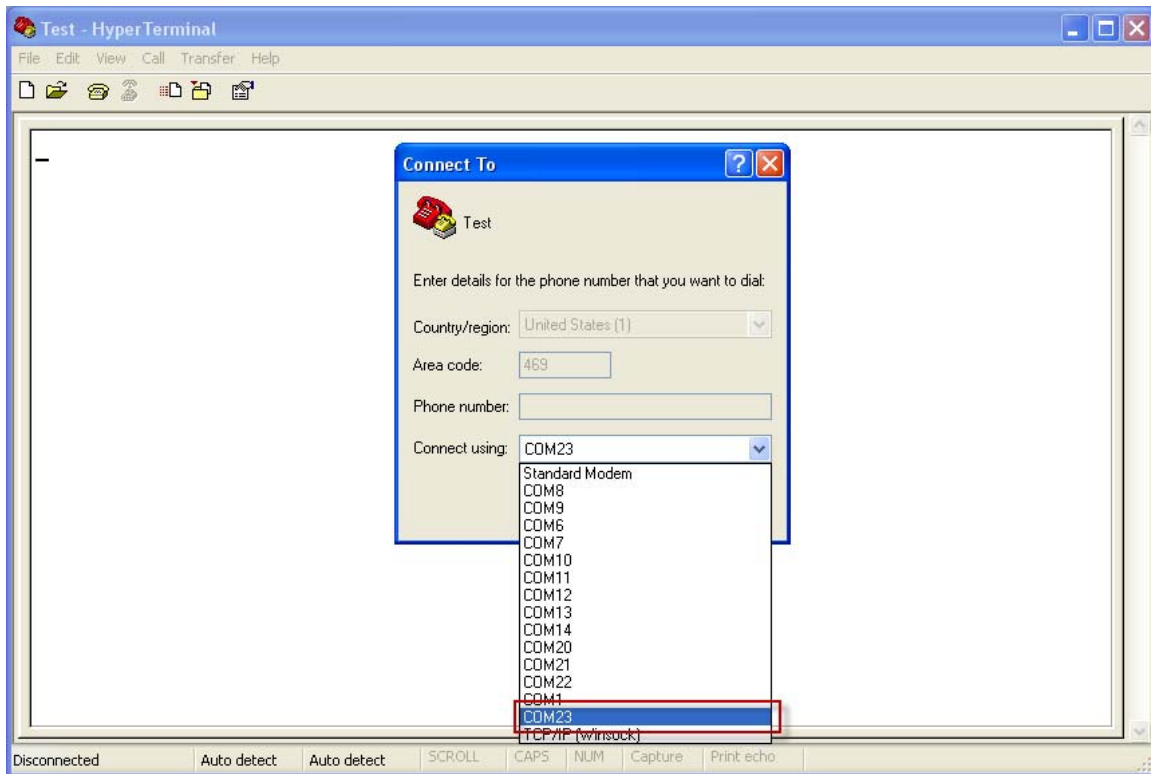
9. Build and Download the code into the un-programmed board:



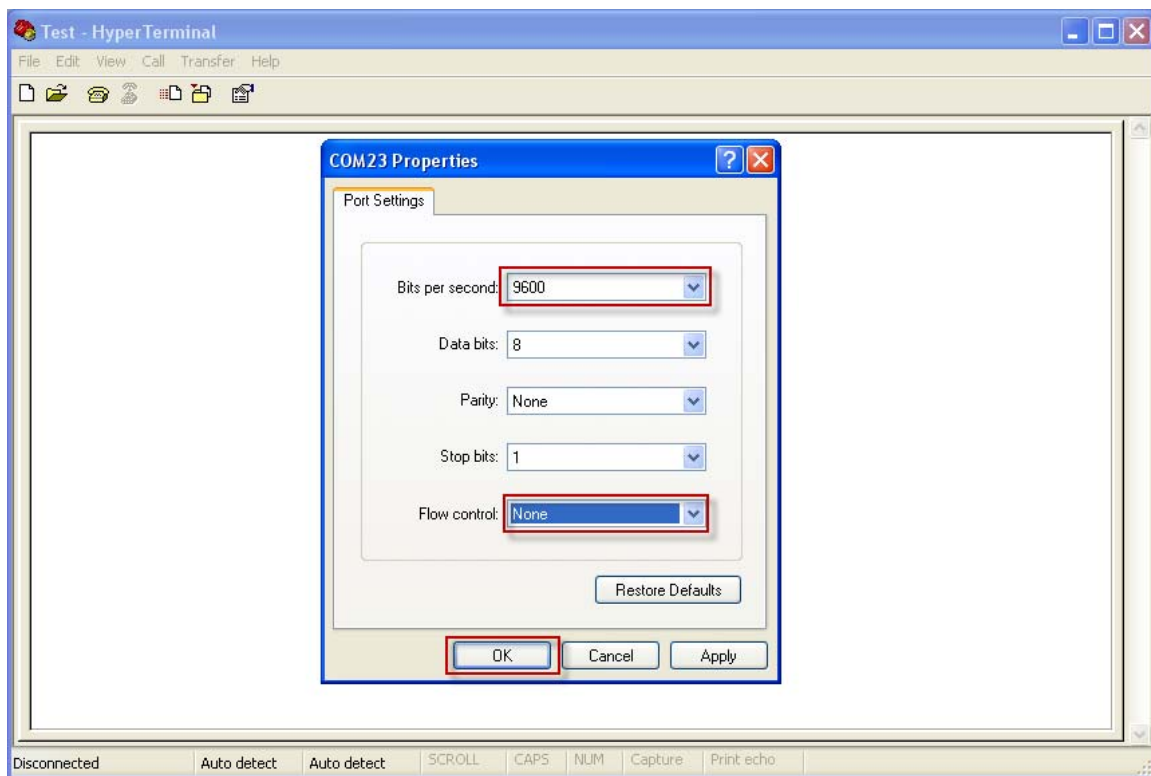
10. Start HyperTerminal:
Start->All Programs->Accessories->Communication->HyperTerminal



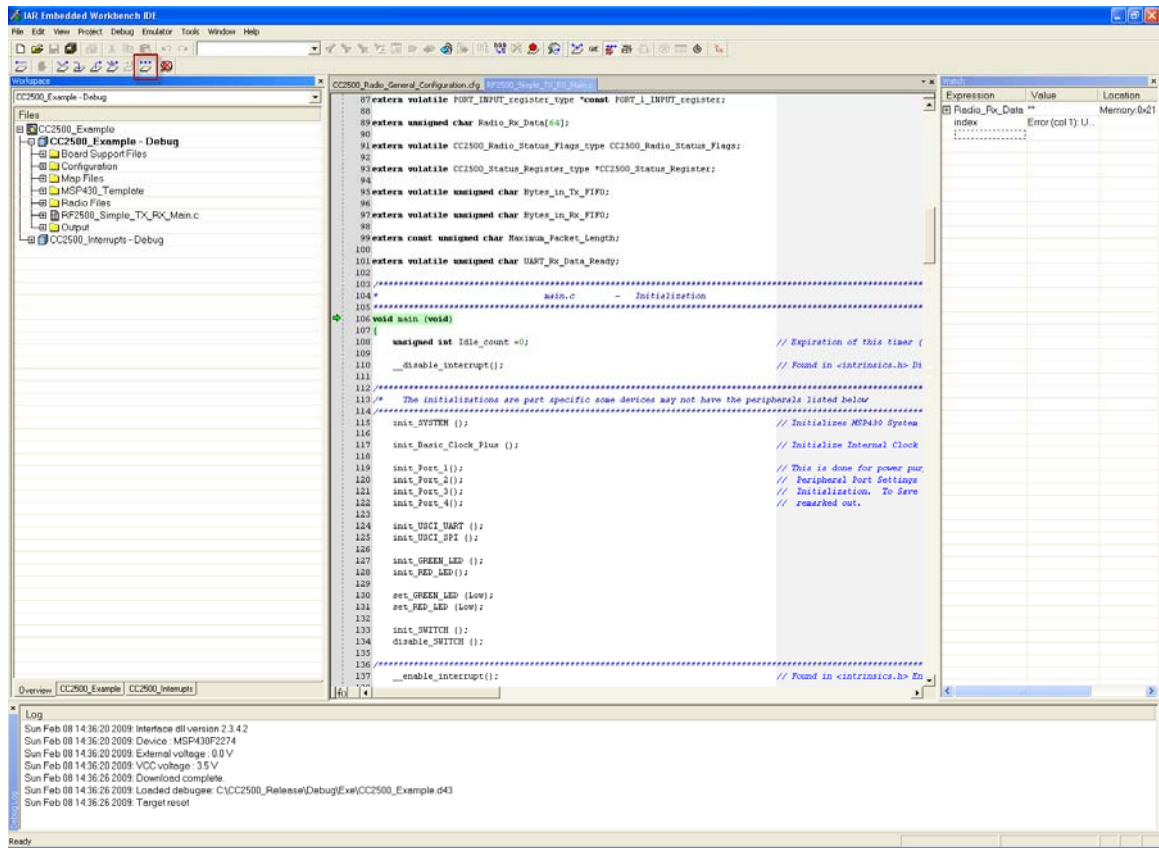
11. Set the Com Port to the MSP430 Application UART, and press OK



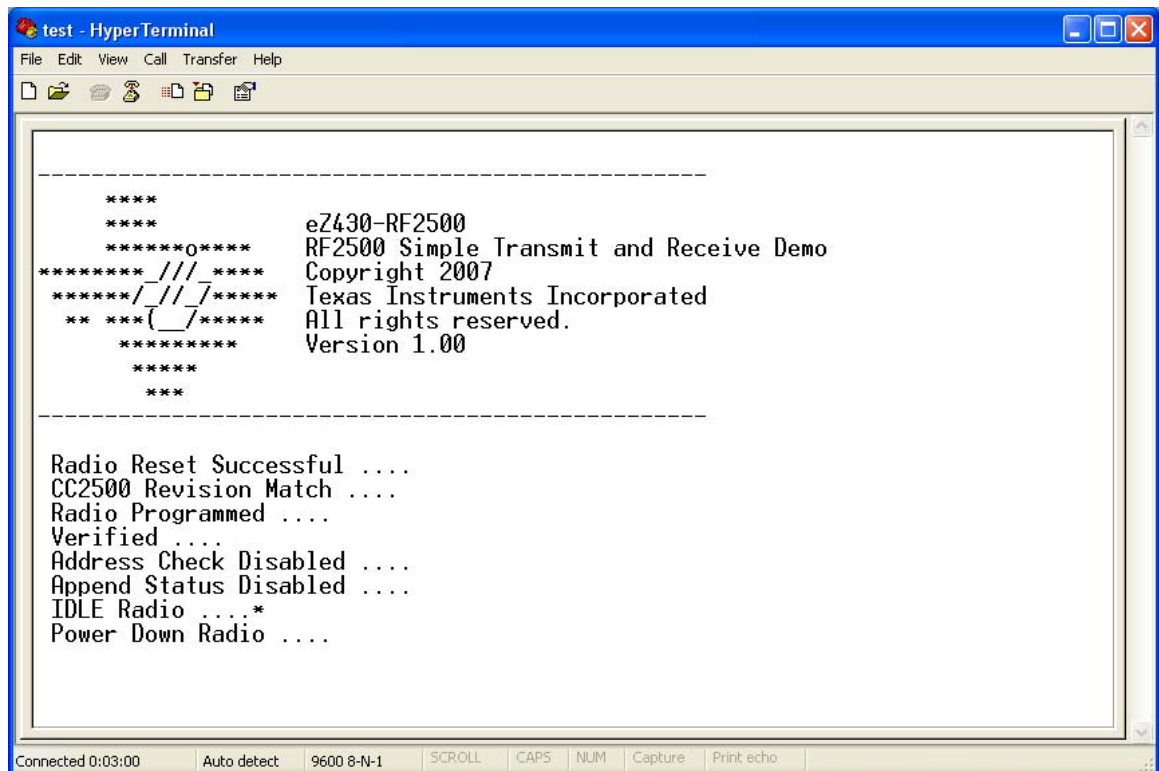
12. Set the Com Port to 9600 Baud, 8N1:



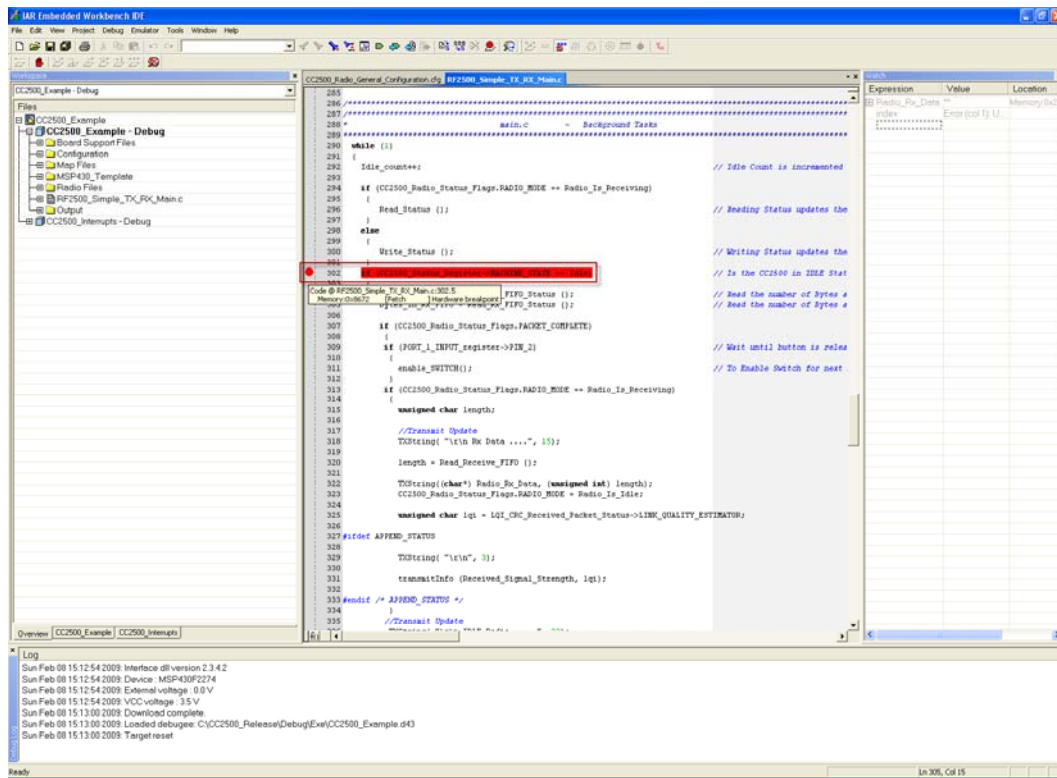
13. Run the Code:



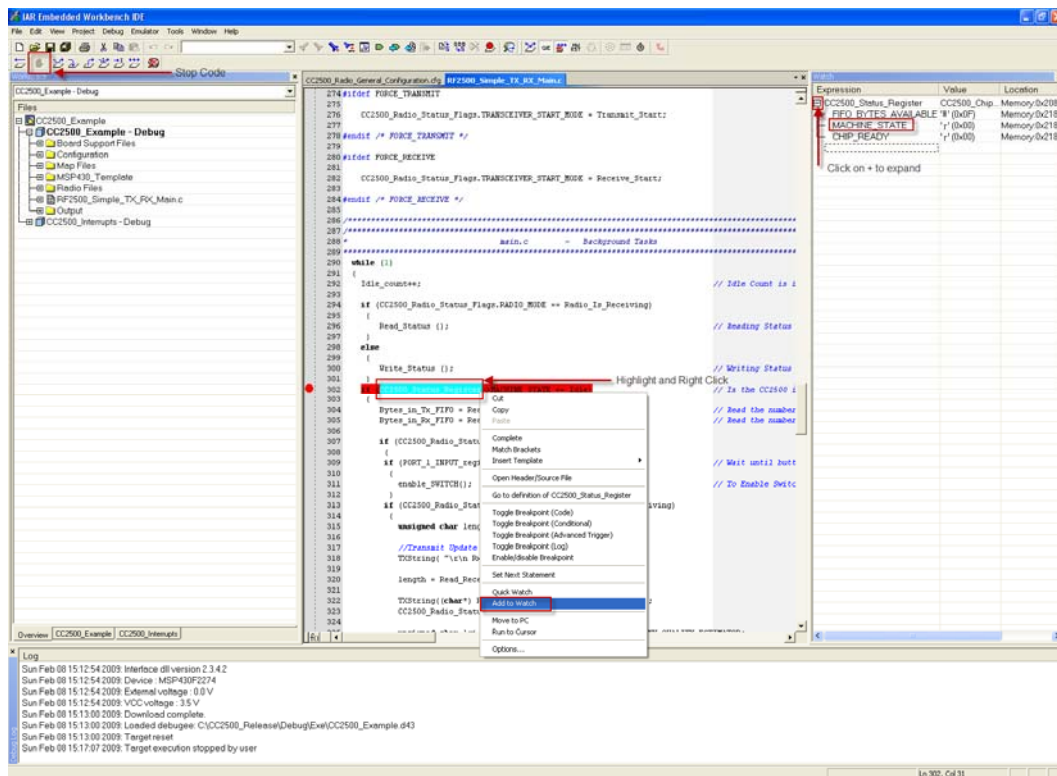
The HyperTerminal Screen should show:



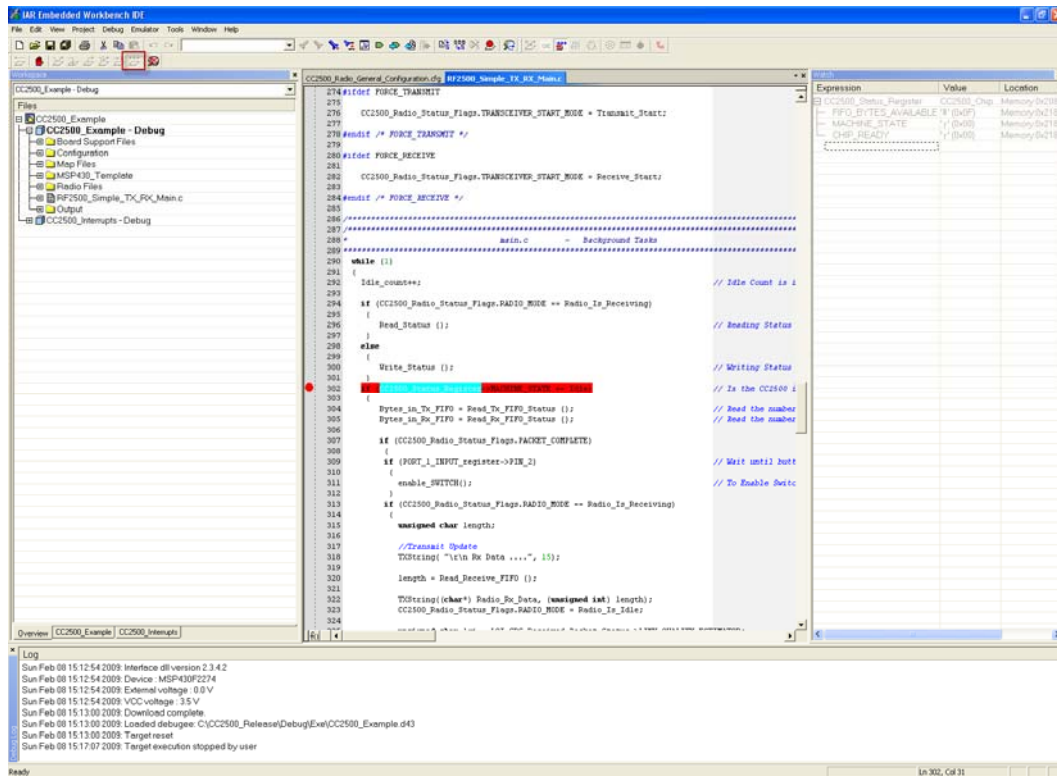
14. Set a breakpoint at line 302:



15. Highlight and watch the variable: CC2500_Status_Register



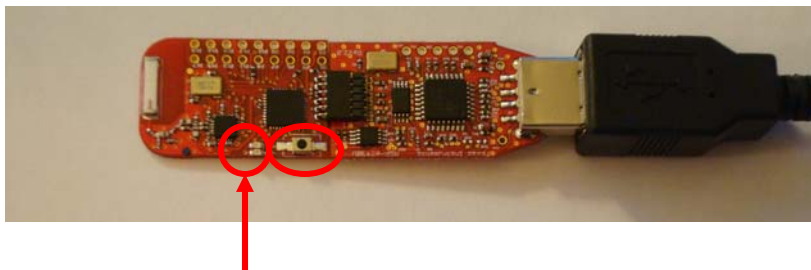
16. Continue the code:



The screenshot shows the IAR Embedded Workbench IDE with the CC2500 radio configuration code. The code is in the main editor window, and the left pane shows the project structure. The code includes a while loop for background tasks and a transmit function. The transmit function is highlighted in red. The right pane shows the expression evaluator with the following data:

Expression	Value	Location
CC2500_Radio_Status_Flags	0x00000000	Memory [0x00]
INFO_BYTES_AVAILABLE	0x00000000	Memory [0x00]
MAC_FRAME_STATE	0x00000000	Memory [0x00]
CHIP_READY	0x00000000	Memory [0x00]

17. Press and hold button to transmit:



Green LED will light (showing the radio is in transmit mode).

-
- The screenshot displays the IAR Embedded Workbench IDE interface. The main window shows a C source file named `CC2500_Radio_Status_Flags` with the following code:
- ```

274 #define FORCE_TRANSMIT
275
276 CC2500_Radio_Status_Flags.TRANSMITTER_START_MODE = Transmitter_Start;
277
278 #endif /* FORCE_TRANSMIT */
279
280 #define FORCE_RECEIVE
281
282 CC2500_Radio_Status_Flags.RECEIVER_START_MODE = Receiver_Start;
283
284 #endif /* FORCE_RECEIVE */
285
286
287 //=====
288 // Main Loop
289 //=====
290
291 while (1)
292 {
293 // Idle Count is 1
294 if (CC2500_Radio_Status_Flags.RADIO_MODE == Radio_Is_Receiving)
295 {
296 Read_Status ();
297 }
298 else
299 {
300 Write_Status ();
301 }
302 // Is the CC2500 i
303 // Read the number
304 Bytes_in_Tx_FIFO = Read_Tx_FIFO_Status ();
305 Bytes_in_Rx_FIFO = Read_Rx_FIFO_Status ();
306 // Read the number
307 if (CC2500_Radio_Status_Flags.PACKET_COMPLETE)
308 {
309 if (P0RT_I_TNPUT_register->PIN_3)
310 {
311 enable_SWITCH();
312 // To Enable Switc
313 }
314 if (CC2500_Radio_Status_Flags.RADIO_MODE == Radio_Is_Receiving)
315 {
316 unsigned char length;
317 //Transmit Update
318 TXStatus("Txn Rx Data", 10);
319 length = Read_Receive_FIFO ();
320 TXStatus("char", Radio_Px_Data, (unsigned int) length);
321 CC2500_Radio_Status_Flags.RADIO_MODE = Radio_Is_Idle;
322 }
323 }
324 }

```
- The 'Machine State' window on the right shows the following registers:
- | Expression             | Value      | Location     |
|------------------------|------------|--------------|
| CC2500_Status_Register | 0x00000000 | Memory@0x210 |
| FIFO_BYTES_AVAILABLE   | 0x00000000 | Memory@0x210 |
| MACHINE_STATE          | 0x00000000 | Memory@0x210 |
| CHIP_READY             | 0x00000000 | Memory@0x210 |
- The 'Log' window at the bottom shows the following messages:
- ```

Sun Feb 08 15:13:00 2009: Loaded debugge: C:\CC2500_Release\Debug\CC2500_Example.d43
Sun Feb 08 15:13:00 2009: Target reset
Sun Feb 08 15:17:07 2009: Target execution stopped by user
Sun Feb 08 15:26:53 2009: Breakpoint hit: Code @ RF2500_Simple_Tx_Rx_Main.c:302:5
Sun Feb 08 15:29:41 2009: Breakpoint hit: Code @ RF2500_Simple_Tx_Rx_Main.c:302:5
Sun Feb 08 15:29:41 2009: Breakpoint hit: Code @ RF2500_Simple_Tx_Rx_Main.c:302:5
Sun Feb 08 15:29:42 2009: Breakpoint hit: Code @ RF2500_Simple_Tx_Rx_Main.c:302:5
Sun Feb 08 15:29:42 2009: Breakpoint hit: Code @ RF2500_Simple_Tx_Rx_Main.c:302:5
Sun Feb 08 15:29:43 2009: Breakpoint hit: Code @ RF2500_Simple_Tx_Rx_Main.c:302:5

```

test - HyperTerminal

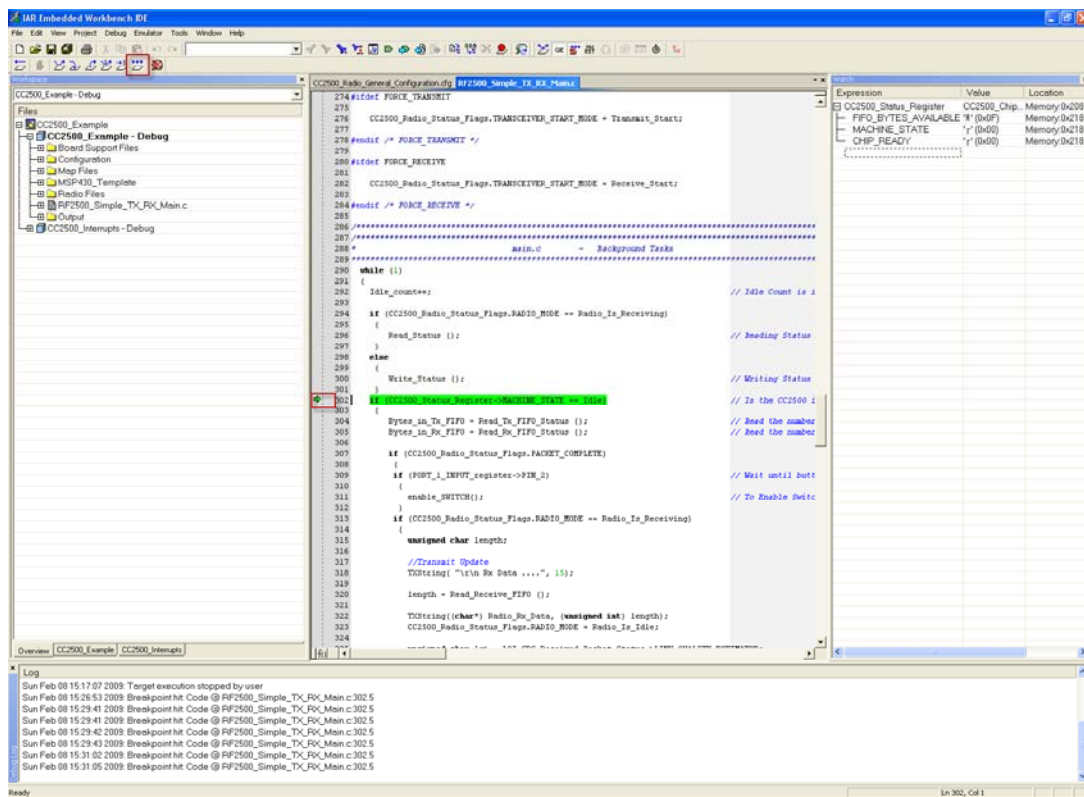
File Edit View Call Transfer Help

*****0***** RF2500 Simple Transmit and Receive Demo
 ***** /// ***** Copyright 2007
 ***** /// ***** Texas Instruments Incorporated
 ** ** { / } ***** All rights reserved.
 ***** Version 1.00

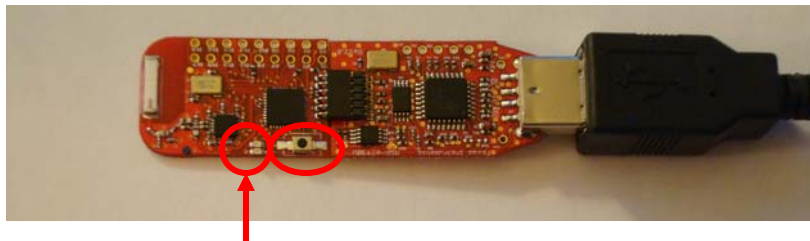
Radio Reset Successful
 CC2500 Revision Match
 Radio Programmed
 Verified
 Address Check Disabled
 Append Status Disabled
 IDLE Radio ...*
 Power Down Radio *
 TX FIFO Loaded ...
 Frequency Synth Enabled ...
 Calibrating
 Frequency Synth On ...
 Transmitting ..*
 IDLE Radio

Connected 0:18:33 Auto detect 9600 8-N-1 SCROLL CAPS NUM Capture Print echo

19. Clear the break point by double clicking on the green arrow on line 302 and then execute the code.

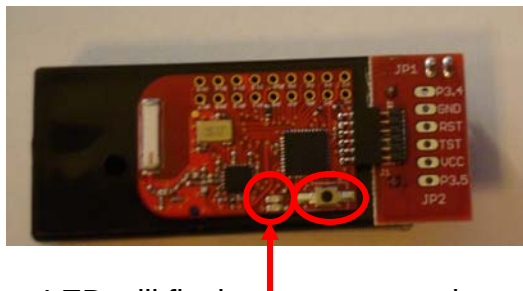


20. Press and release the switch on the eZ430-RF2500 board:



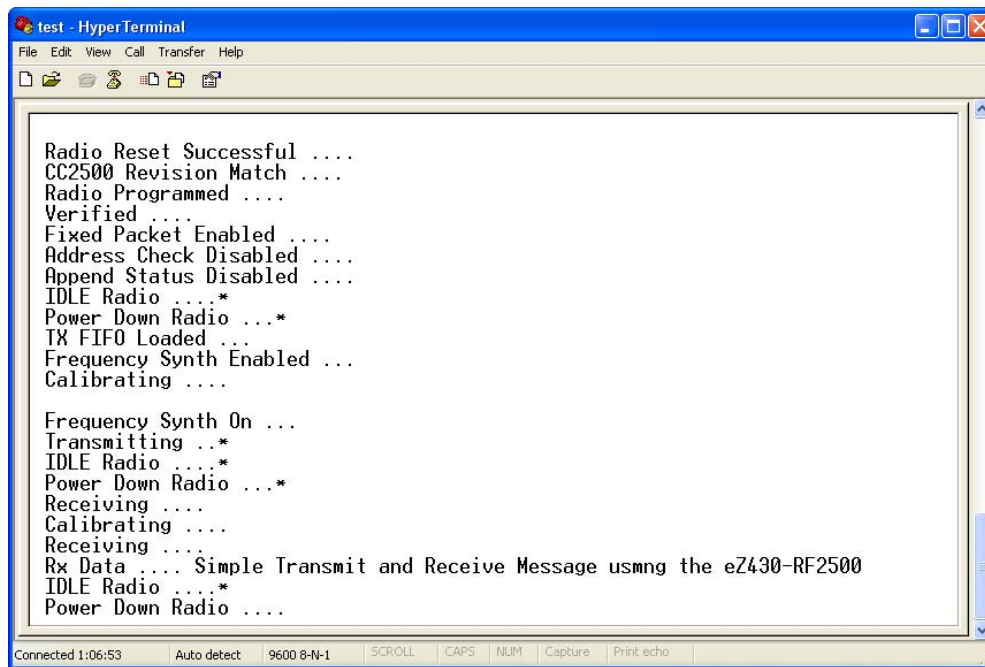
The red LED will light, the radio is in Receive Mode.

21. Press and hold the switch on the Battery Board:



The green LED will flash as a message is transmitted.

The message will appear on the hyperterminal session.



```
test - HyperTerminal
File Edit View Call Transfer Help

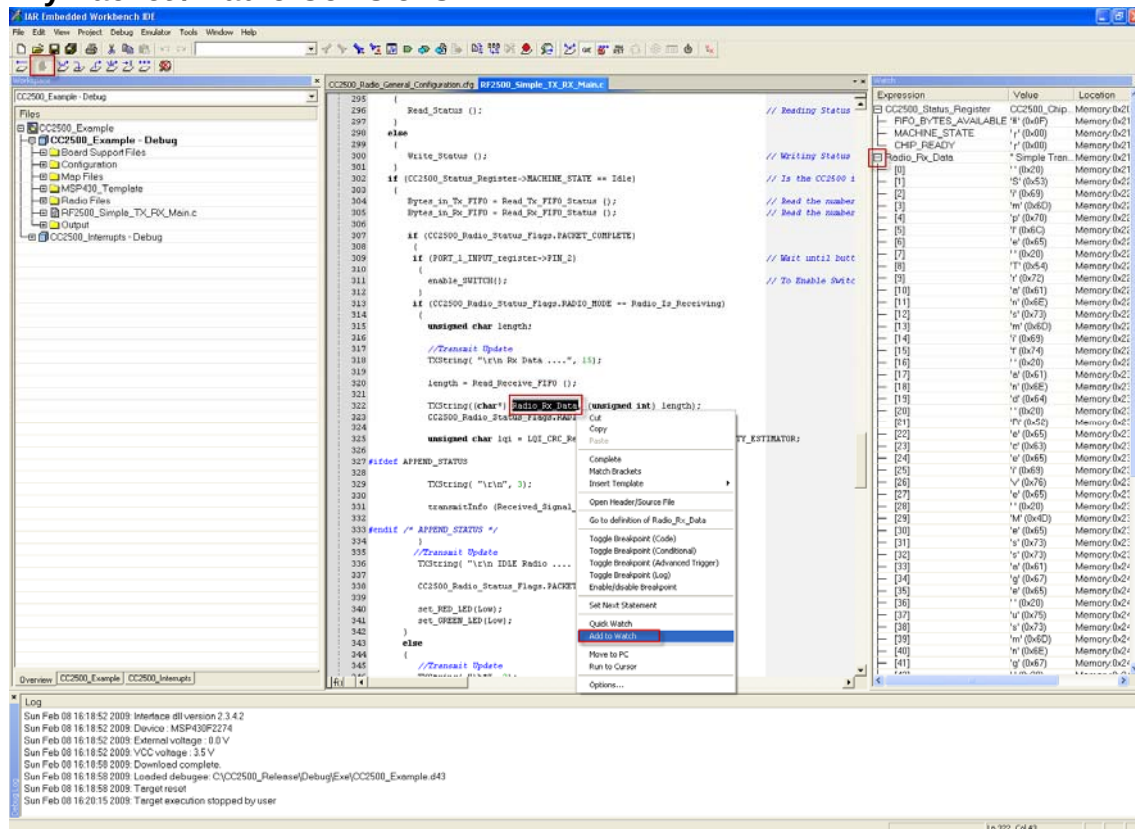
Radio Reset Successful ....
CC2500 Revision Match ....
Radio Programmed ....
Verified ....
Fixed Packet Enabled ....
Address Check Disabled ....
Append Status Disabled ....
IDLE Radio ....*
Power Down Radio ...*
TX FIFO Loaded ...
Frequency Synth Enabled ...
Calibrating ....

Frequency Synth On ...
Transmitting ..*
IDLE Radio ....*
Power Down Radio ...*
Receiving ....
Calibrating ....
Receiving ....
Rx Data .... Simple Transmit and Receive Message using the eZ430-RF2500
IDLE Radio ....*
Power Down Radio ....

Connected 1:06:53  Auto detect  9600 8-N-1  SCROLL  CAPS  NUM  Capture  Print echo
```

22. Stop the debugging session, highlight `Radio_Rx_Data` on line 322, right click on it and add the watch. Click on the + next to the `Radio_Rx_Data` in the watch window:

Any Packet / Radio Collisions??



The screenshot shows the IAR Embedded Workbench IDE. In the main editor, the file `RF2500_Simple_TX_RX_Main.c` is open. Line 322, `Radio Rx Data`, is highlighted, and a right-click context menu is displayed with the option **ADD WATCH** selected. The watch window on the right lists various variables, with `Radio_Rx_Data` highlighted in the first column. The bottom status bar shows the target device as `IA 302, C643`.

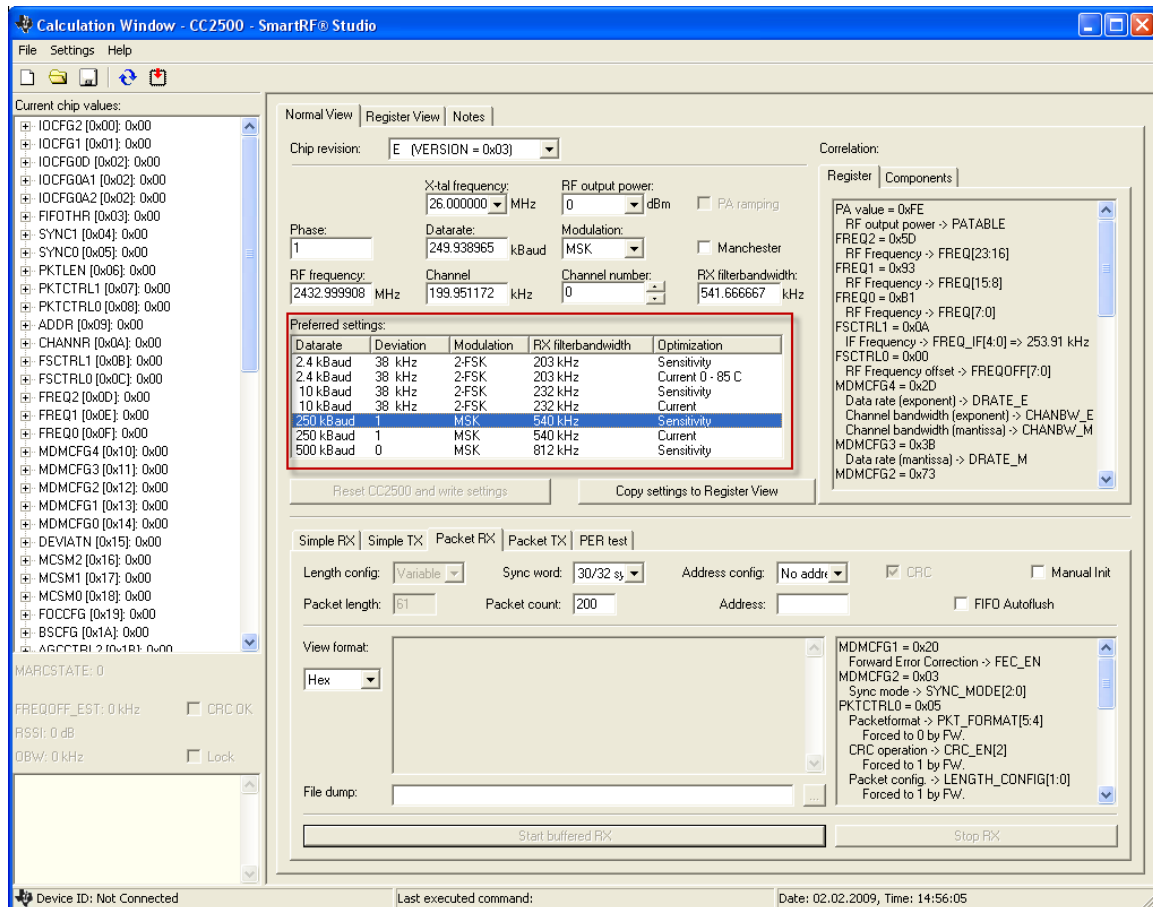


Lab 2: Editing the Radio Settings

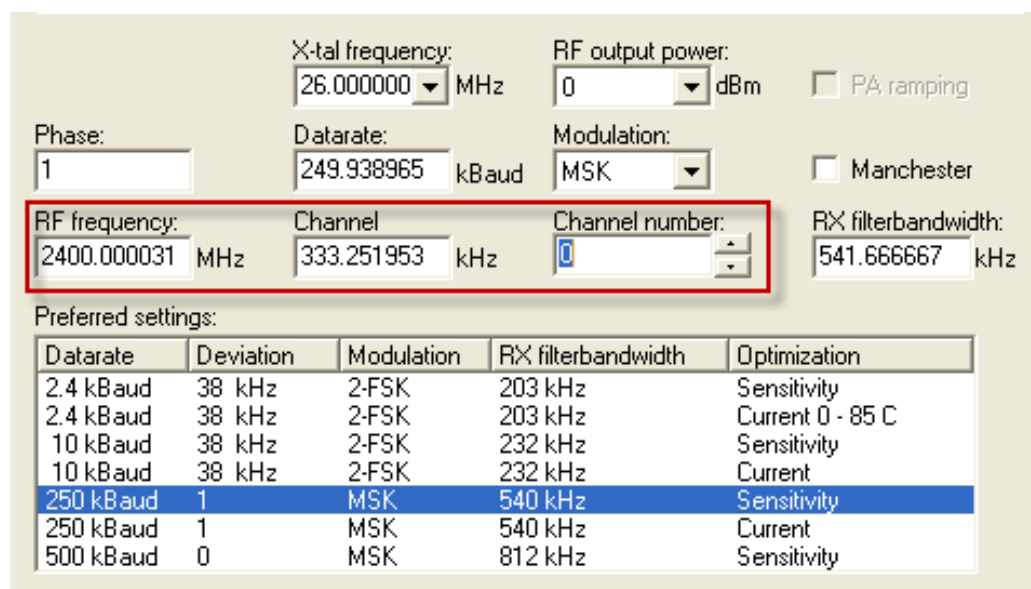
1. With the code stopped, open the cc_2500_constants.c file. Note the Channel and Radio frequency:

```
1 // Chipset
2 // Product = CC2500
3 // Chip version = 2 (CC2500 = 0x02)
4 // Crystal accuracy = 10 ppm
5 // X-tal frequency = 26 MHz
6 // RF output power = 0 dbm
7 // RX filter bandwidth = 232.141857 kHz
8 // Deviation = 50 kHz
9 // Datarate = 9.905599 kbaud
10 // Modulation = (0) 2-FSK
11 // Manchester enable = (0) Manchester disabled
12 // RF Frequency = 2412.496979 MHz
13 // Channel spacing = 244.99995 kHz
14 // Channel number = 10
15 // Optimization = Sensitivity
16 // Sync mode = (0) 16/32 sync word bits detected
17 // Format of RX/TX data = (0) Normal mode, use FIFOs for RX and TX
18 // CRC operation = (1) CRC calculation in TX and CRC check in RX enabled
19 // Forward Error Correction = (0) FEC disabled
20 // Length configuration = (1) Variable length packets, packet length configured by the first byte
21 // Packet length = 255
22 // Frame count = (2) 4 bytes
23 // Spread status = 1
24 // Address check = (0) No address check
25 // FIFO autoflush = 0
26 // Device address = 0
27 // GDO0 signal selection = (4) Asserts when sync word has been sent / received, and de-asserts at
28 // GDO0 signal selection = (4) CHIP_READY
29 // Global Variables
30 //
31 //
32
33 const unsigned char cc_radio_table[10] = {
34 0x00, 0x00, // FCTRL1 - FREQUENCY SYNTHESIZER CONTROL.
35 0x00, 0x00, // FCTRL0 - FREQUENCY SYNTHESIZER CONTROL.
36 0x00, 0x00, // FREQ2 - FREQUENCY CONTROL WORD, HIGH BYTE.
37 0x00, 0x00, // FREQ1 - FREQUENCY CONTROL WORD, MIDDLE BYTE.
38 0x00, 0x00, // FREQ0 - FREQUENCY CONTROL WORD, LOW BYTE.
39 0x10, 0x10, // MCMCFG0 - MCMCFG CONFIGURATION.
40 0x11, 0x11, // MCMCFG1 - MCMCFG CONFIGURATION.
41 0x12, 0x12, // MCMCFG2 - MCMCFG CONFIGURATION.
42 0x13, 0x13, // MCMCFG3 - MCMCFG CONFIGURATION.
43 0x14, 0x14, // MCMCFG4 - MCMCFG CONFIGURATION.
44 0x15, 0x15, // CHANNR - CHANNEL NUMBER.
45 0x16, 0x16, // DEVZLEN - MCMCFG DEVIATION SETTING (WHEN FSK MODULATION IS ENABLED).
46 0x17, 0x17, // FREQ3 - FREQ END RX CONFIGURATION.
47 0x18, 0x18, // FREQ4 - FREQ END TX CONFIGURATION.
48 0x19, 0x19, // MCMCFG5 - MCMCFG CONFIGURATION.
49 0x1A, 0x1A, // MCMCFG6 - MCMCFG CONFIGURATION.
50 0x1B, 0x1B, // MCMCFG7 - MCMCFG CONFIGURATION.
51 0x1C, 0x1C, // MCMCFG8 - MCMCFG CONFIGURATION.
52 0x1D, 0x1D, // MCMCFG9 - MCMCFG CONFIGURATION.
53 0x1E, 0x1E, // MCMCFG10 - MCMCFG CONFIGURATION.
54 0x1F, 0x1F, // MCMCFG11 - MCMCFG CONFIGURATION.
55 0x20, 0x20, // MCMCFG12 - MCMCFG CONFIGURATION.
56 0x21, 0x21, // MCMCFG13 - MCMCFG CONFIGURATION.
57 0x22, 0x22, // MCMCFG14 - MCMCFG CONFIGURATION.
58 0x23, 0x23, // MCMCFG15 - MCMCFG CONFIGURATION.
59 0x24, 0x24, // MCMCFG16 - MCMCFG CONFIGURATION.
60 0x25, 0x25, // MCMCFG17 - MCMCFG CONFIGURATION.
61 0x26, 0x26, // MCMCFG18 - MCMCFG CONFIGURATION.
62 0x27, 0x27, // MCMCFG19 - MCMCFG CONFIGURATION.
63 0x28, 0x28, // MCMCFG20 - MCMCFG CONFIGURATION.
64 0x29, 0x29, // MCMCFG21 - MCMCFG CONFIGURATION.
65 0x2A, 0x2A, // MCMCFG22 - MCMCFG CONFIGURATION.
66 0x2B, 0x2B, // MCMCFG23 - MCMCFG CONFIGURATION.
67 0x2C, 0x2C, // MCMCFG24 - MCMCFG CONFIGURATION.
68 0x2D, 0x2D, // MCMCFG25 - MCMCFG CONFIGURATION.
69 0x2E, 0x2E, // MCMCFG26 - MCMCFG CONFIGURATION.
70 0x2F, 0x2F, // MCMCFG27 - MCMCFG CONFIGURATION.
71 0x30, 0x30, // MCMCFG28 - MCMCFG CONFIGURATION.
72 0x31, 0x31, // MCMCFG29 - MCMCFG CONFIGURATION.
73 0x32, 0x32, // MCMCFG30 - MCMCFG CONFIGURATION.
74 0x33, 0x33, // MCMCFG31 - MCMCFG CONFIGURATION.
75 0x34, 0x34, // MCMCFG32 - MCMCFG CONFIGURATION.
76 0x35, 0x35, // MCMCFG33 - MCMCFG CONFIGURATION.
77 0x36, 0x36, // MCMCFG34 - MCMCFG CONFIGURATION.
78 0x37, 0x37, // MCMCFG35 - MCMCFG CONFIGURATION.
79 0x38, 0x38, // MCMCFG36 - MCMCFG CONFIGURATION.
80 0x39, 0x39, // MCMCFG37 - MCMCFG CONFIGURATION.
81 0x3A, 0x3A, // MCMCFG38 - MCMCFG CONFIGURATION.
82 0x3B, 0x3B, // MCMCFG39 - MCMCFG CONFIGURATION.
83 0x3C, 0x3C, // MCMCFG40 - MCMCFG CONFIGURATION.
84 0x3D, 0x3D, // MCMCFG41 - MCMCFG CONFIGURATION.
85 0x3E, 0x3E, // MCMCFG42 - MCMCFG CONFIGURATION.
86 0x3F, 0x3F, // MCMCFG43 - MCMCFG CONFIGURATION.
87 0x40, 0x40, // MCMCFG44 - MCMCFG CONFIGURATION.
88 0x41, 0x41, // MCMCFG45 - MCMCFG CONFIGURATION.
89 0x42, 0x42, // MCMCFG46 - MCMCFG CONFIGURATION.
90 0x43, 0x43, // MCMCFG47 - MCMCFG CONFIGURATION.
91 0x44, 0x44, // MCMCFG48 - MCMCFG CONFIGURATION.
92 0x45, 0x45, // MCMCFG49 - MCMCFG CONFIGURATION.
93 0x46, 0x46, // MCMCFG50 - MCMCFG CONFIGURATION.
94 0x47, 0x47, // MCMCFG51 - MCMCFG CONFIGURATION.
95 0x48, 0x48, // MCMCFG52 - MCMCFG CONFIGURATION.
96 0x49, 0x49, // MCMCFG53 - MCMCFG CONFIGURATION.
97 0x4A, 0x4A, // MCMCFG54 - MCMCFG CONFIGURATION.
98 0x4B, 0x4B, // MCMCFG55 - MCMCFG CONFIGURATION.
99 0x4C, 0x4C, // MCMCFG56 - MCMCFG CONFIGURATION.
100 0x4D, 0x4D, // MCMCFG57 - MCMCFG CONFIGURATION.
101 0x4E, 0x4E, // MCMCFG58 - MCMCFG CONFIGURATION.
102 0x4F, 0x4F, // MCMCFG59 - MCMCFG CONFIGURATION.
103 0x50, 0x50, // MCMCFG60 - MCMCFG CONFIGURATION.
104 0x51, 0x51, // MCMCFG61 - MCMCFG CONFIGURATION.
105 0x52, 0x52, // MCMCFG62 - MCMCFG CONFIGURATION.
106 0x53, 0x53, // MCMCFG63 - MCMCFG CONFIGURATION.
107 0x54, 0x54, // MCMCFG64 - MCMCFG CONFIGURATION.
108 0x55, 0x55, // MCMCFG65 - MCMCFG CONFIGURATION.
109 0x56, 0x56, // MCMCFG66 - MCMCFG CONFIGURATION.
110 0x57, 0x57, // MCMCFG67 - MCMCFG CONFIGURATION.
111 0x58, 0x58, // MCMCFG68 - MCMCFG CONFIGURATION.
112 0x59, 0x59, // MCMCFG69 - MCMCFG CONFIGURATION.
113 0x5A, 0x5A, // MCMCFG70 - MCMCFG CONFIGURATION.
114 0x5B, 0x5B, // MCMCFG71 - MCMCFG CONFIGURATION.
115 0x5C, 0x5C, // MCMCFG72 - MCMCFG CONFIGURATION.
116 0x5D, 0x5D, // MCMCFG73 - MCMCFG CONFIGURATION.
117 0x5E, 0x5E, // MCMCFG74 - MCMCFG CONFIGURATION.
118 0x5F, 0x5F, // MCMCFG75 - MCMCFG CONFIGURATION.
119 0x60, 0x60, // MCMCFG76 - MCMCFG CONFIGURATION.
120 0x61, 0x61, // MCMCFG77 - MCMCFG CONFIGURATION.
121 0x62, 0x62, // MCMCFG78 - MCMCFG CONFIGURATION.
122 0x63, 0x63, // MCMCFG79 - MCMCFG CONFIGURATION.
123 0x64, 0x64, // MCMCFG80 - MCMCFG CONFIGURATION.
124 0x65, 0x65, // MCMCFG81 - MCMCFG CONFIGURATION.
125 0x66, 0x66, // MCMCFG82 - MCMCFG CONFIGURATION.
126 0x67, 0x67, // MCMCFG83 - MCMCFG CONFIGURATION.
127 0x68, 0x68, // MCMCFG84 - MCMCFG CONFIGURATION.
128 0x69, 0x69, // MCMCFG85 - MCMCFG CONFIGURATION.
129 0x6A, 0x6A, // MCMCFG86 - MCMCFG CONFIGURATION.
130 0x6B, 0x6B, // MCMCFG87 - MCMCFG CONFIGURATION.
131 0x6C, 0x6C, // MCMCFG88 - MCMCFG CONFIGURATION.
132 0x6D, 0x6D, // MCMCFG89 - MCMCFG CONFIGURATION.
133 0x6E, 0x6E, // MCMCFG90 - MCMCFG CONFIGURATION.
134 0x6F, 0x6F, // MCMCFG91 - MCMCFG CONFIGURATION.
135 0x70, 0x70, // MCMCFG92 - MCMCFG CONFIGURATION.
136 0x71, 0x71, // MCMCFG93 - MCMCFG CONFIGURATION.
137 0x72, 0x72, // MCMCFG94 - MCMCFG CONFIGURATION.
138 0x73, 0x73, // MCMCFG95 - MCMCFG CONFIGURATION.
139 0x74, 0x74, // MCMCFG96 - MCMCFG CONFIGURATION.
140 0x75, 0x75, // MCMCFG97 - MCMCFG CONFIGURATION.
141 0x76, 0x76, // MCMCFG98 - MCMCFG CONFIGURATION.
142 0x77, 0x77, // MCMCFG99 - MCMCFG CONFIGURATION.
143 0x78, 0x78, // MCMCFG100 - MCMCFG CONFIGURATION.
144 0x79, 0x79, // MCMCFG101 - MCMCFG CONFIGURATION.
145 0x7A, 0x7A, // MCMCFG102 - MCMCFG CONFIGURATION.
146 0x7B, 0x7B, // MCMCFG103 - MCMCFG CONFIGURATION.
147 0x7C, 0x7C, // MCMCFG104 - MCMCFG CONFIGURATION.
148 0x7D, 0x7D, // MCMCFG105 - MCMCFG CONFIGURATION.
149 0x7E, 0x7E, // MCMCFG106 - MCMCFG CONFIGURATION.
150 0x7F, 0x7F, // MCMCFG107 - MCMCFG CONFIGURATION.
151 0x80, 0x80, // MCMCFG108 - MCMCFG CONFIGURATION.
152 0x81, 0x81, // MCMCFG109 - MCMCFG CONFIGURATION.
153 0x82, 0x82, // MCMCFG110 - MCMCFG CONFIGURATION.
154 0x83, 0x83, // MCMCFG111 - MCMCFG CONFIGURATION.
155 0x84, 0x84, // MCMCFG112 - MCMCFG CONFIGURATION.
156 0x85, 0x85, // MCMCFG113 - MCMCFG CONFIGURATION.
157 0x86, 0x86, // MCMCFG114 - MCMCFG CONFIGURATION.
158 0x87, 0x87, // MCMCFG115 - MCMCFG CONFIGURATION.
159 0x88, 0x88, // MCMCFG116 - MCMCFG CONFIGURATION.
160 0x89, 0x89, // MCMCFG117 - MCMCFG CONFIGURATION.
161 0x8A, 0x8A, // MCMCFG118 - MCMCFG CONFIGURATION.
162 0x8B, 0x8B, // MCMCFG119 - MCMCFG CONFIGURATION.
163 0x8C, 0x8C, // MCMCFG120 - MCMCFG CONFIGURATION.
164 0x8D, 0x8D, // MCMCFG121 - MCMCFG CONFIGURATION.
165 0x8E, 0x8E, // MCMCFG122 - MCMCFG CONFIGURATION.
166 0x8F, 0x8F, // MCMCFG123 - MCMCFG CONFIGURATION.
167 0x90, 0x90, // MCMCFG124 - MCMCFG CONFIGURATION.
168 0x91, 0x91, // MCMCFG125 - MCMCFG CONFIGURATION.
169 0x92, 0x92, // MCMCFG126 - MCMCFG CONFIGURATION.
170 0x93, 0x93, // MCMCFG127 - MCMCFG CONFIGURATION.
171 0x94, 0x94, // MCMCFG128 - MCMCFG CONFIGURATION.
172 0x95, 0x95, // MCMCFG129 - MCMCFG CONFIGURATION.
173 0x96, 0x96, // MCMCFG130 - MCMCFG CONFIGURATION.
174 0x97, 0x97, // MCMCFG131 - MCMCFG CONFIGURATION.
175 0x98, 0x98, // MCMCFG132 - MCMCFG CONFIGURATION.
176 0x99, 0x99, // MCMCFG133 - MCMCFG CONFIGURATION.
177 0x9A, 0x9A, // MCMCFG134 - MCMCFG CONFIGURATION.
178 0x9B, 0x9B, // MCMCFG135 - MCMCFG CONFIGURATION.
179 0x9C, 0x9C, // MCMCFG136 - MCMCFG CONFIGURATION.
180 0x9D, 0x9D, // MCMCFG137 - MCMCFG CONFIGURATION.
181 0x9E, 0x9E, // MCMCFG138 - MCMCFG CONFIGURATION.
182 0x9F, 0x9F, // MCMCFG139 - MCMCFG CONFIGURATION.
183 0xA0, 0xA0, // MCMCFG140 - MCMCFG CONFIGURATION.
184 0xA1, 0xA1, // MCMCFG141 - MCMCFG CONFIGURATION.
185 0xA2, 0xA2, // MCMCFG142 - MCMCFG CONFIGURATION.
186 0xA3, 0xA3, // MCMCFG143 - MCMCFG CONFIGURATION.
187 0xA4, 0xA4, // MCMCFG144 - MCMCFG CONFIGURATION.
188 0xA5, 0xA5, // MCMCFG145 - MCMCFG CONFIGURATION.
189 0xA6, 0xA6, // MCMCFG146 - MCMCFG CONFIGURATION.
190 0xA7, 0xA7, // MCMCFG147 - MCMCFG CONFIGURATION.
191 0xA8, 0xA8, // MCMCFG148 - MCMCFG CONFIGURATION.
192 0xA9, 0xA9, // MCMCFG149 - MCMCFG CONFIGURATION.
193 0xAA, 0xAA, // MCMCFG150 - MCMCFG CONFIGURATION.
194 0xAB, 0xAB, // MCMCFG151 - MCMCFG CONFIGURATION.
195 0xAC, 0xAC, // MCMCFG152 - MCMCFG CONFIGURATION.
196 0xAD, 0xAD, // MCMCFG153 - MCMCFG CONFIGURATION.
197 0xAE, 0xAE, // MCMCFG154 - MCMCFG CONFIGURATION.
198 0xAF, 0xAF, // MCMCFG155 - MCMCFG CONFIGURATION.
199 0xB0, 0xB0, // MCMCFG156 - MCMCFG CONFIGURATION.
200 0xB1, 0xB1, // MCMCFG157 - MCMCFG CONFIGURATION.
201 0xB2, 0xB2, // MCMCFG158 - MCMCFG CONFIGURATION.
202 0xB3, 0xB3, // MCMCFG159 - MCMCFG CONFIGURATION.
203 0xB4, 0xB4, // MCMCFG160 - MCMCFG CONFIGURATION.
204 0xB5, 0xB5, // MCMCFG161 - MCMCFG CONFIGURATION.
205 0xB6, 0xB6, // MCMCFG162 - MCMCFG CONFIGURATION.
206 0xB7, 0xB7, // MCMCFG163 - MCMCFG CONFIGURATION.
207 0xB8, 0xB8, // MCMCFG164 - MCMCFG CONFIGURATION.
208 0xB9, 0xB9, // MCMCFG165 - MCMCFG CONFIGURATION.
209 0xBA, 0xBA, // MCMCFG166 - MCMCFG CONFIGURATION.
210 0xBB, 0xBB, // MCMCFG167 - MCMCFG CONFIGURATION.
211 0xBC, 0xBC, // MCMCFG168 - MCMCFG CONFIGURATION.
212 0xBD, 0xBD, // MCMCFG169 - MCMCFG CONFIGURATION.
213 0xBE, 0xBE, // MCMCFG170 - MCMCFG CONFIGURATION.
214 0xBF, 0xBF, // MCMCFG171 - MCMCFG CONFIGURATION.
215 0xC0, 0xC0, // MCMCFG172 - MCMCFG CONFIGURATION.
216 0xC1, 0xC1, // MCMCFG173 - MCMCFG CONFIGURATION.
217 0xC2, 0xC2, // MCMCFG174 - MCMCFG CONFIGURATION.
218 0xC3, 0xC3, // MCMCFG175 - MCMCFG CONFIGURATION.
219 0xC4, 0xC4, // MCMCFG176 - MCMCFG CONFIGURATION.
220 0xC5, 0xC5, // MCMCFG177 - MCMCFG CONFIGURATION.
221 0xC6, 0xC6, // MCMCFG178 - MCMCFG CONFIGURATION.
222 0xC7, 0xC7, // MCMCFG179 - MCMCFG CONFIGURATION.
223 0xC8, 0xC8, // MCMCFG180 - MCMCFG CONFIGURATION.
224 0xC9, 0xC9, // MCMCFG181 - MCMCFG CONFIGURATION.
225 0xCA, 0xCA, // MCMCFG182 - MCMCFG CONFIGURATION.
226 0xCB, 0xCB, // MCMCFG183 - MCMCFG CONFIGURATION.
227 0xCC, 0xCC, // MCMCFG184 - MCMCFG CONFIGURATION.
228 0xCD, 0xCD, // MCMCFG185 - MCMCFG CONFIGURATION.
229 0xCE, 0xCE, // MCMCFG186 - MCMCFG CONFIGURATION.
230 0xCF, 0xCF, // MCMCFG187 - MCMCFG CONFIGURATION.
231 0xD0, 0xD0, // MCMCFG188 - MCMCFG CONFIGURATION.
232 0xD1, 0xD1, // MCMCFG189 - MCMCFG CONFIGURATION.
233 0xD2, 0xD2, // MCMCFG190 - MCMCFG CONFIGURATION.
234 0xD3, 0xD3, // MCMCFG191 - MCMCFG CONFIGURATION.
235 0xD4, 0xD4, // MCMCFG192 - MCMCFG CONFIGURATION.
236 0xD5, 0xD5, // MCMCFG193 - MCMCFG CONFIGURATION.
237 0xD6, 0xD6, // MCMCFG194 - MCMCFG CONFIGURATION.
238 0xD7, 0xD7, // MCMCFG195 - MCMCFG CONFIGURATION.
239 0xD8, 0xD8, // MCMCFG196 - MCMCFG CONFIGURATION.
240 0xD9, 0xD9, // MCMCFG197 - MCMCFG CONFIGURATION.
241 0xDA, 0xDA, // MCMCFG198 - MCMCFG CONFIGURATION.
242 0xDB, 0xDB, // MCMCFG199 - MCMCFG CONFIGURATION.
243 0xDC, 0xDC, // MCMCFG200 - MCMCFG CONFIGURATION.
244 0xDD, 0xDD, // MCMCFG201 - MCMCFG CONFIGURATION.
245 0xDE, 0xDE, // MCMCFG202 - MCMCFG CONFIGURATION.
246 0xDF, 0xDF, // MCMCFG203 - MCMCFG CONFIGURATION.
247 0xE0, 0xE0, // MCMCFG204 - MCMCFG CONFIGURATION.
248 0xE1, 0xE1, // MCMCFG205 - MCMCFG CONFIGURATION.
249 0xE2, 0xE2, // MCMCFG206 - MCMCFG CONFIGURATION.
250 0xE3, 0xE3, // MCMCFG207 - MCMCFG CONFIGURATION.
251 0xE4, 0xE4, // MCMCFG208 - MCMCFG CONFIGURATION.
252 0xE5, 0xE5, // MCMCFG209 - MCMCFG CONFIGURATION.
253 0xE6, 0xE6, // MCMCFG210 - MCMCFG CONFIGURATION.
254 0xE7, 0xE7, // MCMCFG211 - MCMCFG CONFIGURATION.
255 0xE8, 0xE8, // MCMCFG212 - MCMCFG CONFIGURATION.
256 0xE9, 0xE9, // MCMCFG213 - MCMCFG CONFIGURATION.
257 0xEA, 0xEA, // MCMCFG214 - MCMCFG CONFIGURATION.
258 0xEB, 0xEB, // MCMCFG215 - MCMCFG CONFIGURATION.
259 0xEC, 0xEC, // MCMCFG216 - MCMCFG CONFIGURATION.
260 0xED, 0xED, // MCMCFG217 - MCMCFG CONFIGURATION.
261 0xEE, 0xEE, // MCMCFG218 - MCMCFG CONFIGURATION.
262 0xEF, 0xEF, // MCMCFG219 - MCMCFG CONFIGURATION.
263 0xF0, 0xF0, // MCMCFG220 - MCMCFG CONFIGURATION.
264 0xF1, 0xF1, // MCMCFG221 - MCMCFG CONFIGURATION.
265 0xF2, 0xF2, // MCMCFG222 - MCMCFG CONFIGURATION.
266 0xF3, 0xF3, // MCMCFG223 - MCMCFG CONFIGURATION.
267 0xF4, 0xF4, // MCMCFG224 - MCMCFG CONFIGURATION.
268 0xF5, 0xF5, // MCMCFG225 - MCMCFG CONFIGURATION.
269 0xF6, 0xF6, // MCMCFG226 - MCMCFG CONFIGURATION.
270 0xF7, 0xF7, // MCMCFG227 - MCMCFG CONFIGURATION.
271 0xF8, 0xF8, // MCMCFG228 - MCMCFG CONFIGURATION.
272 0xF9, 0xF9, // MCMCFG229 - MCMCFG CONFIGURATION.
273 0xFA, 0xFA, // MCMCFG230 - MCMCFG CONFIGURATION.
274 0xFB, 0xFB, // MCMCFG231 - MCMCFG CONFIGURATION.
275 0xFC, 0xFC, // MCMCFG232 - MCMCFG CONFIGURATION.
276 0xFD, 0xFD, // MCMCFG233 - MCMCFG CONFIGURATION.
277 0xFE, 0xFE, // MCMCFG234 - MCMCFG CONFIGURATION.
278 0xFF, 0xFF, // MCMCFG235 - MCMCFG CONFIGURATION.
279 0x00, 0x00, // MCMCFG236 - MCMCFG CONFIGURATION.
280 0x01, 0x01, // MCMCFG237 - MCMCFG CONFIGURATION.
281 0x02, 0x02, // MCMCFG238 - MCMCFG CONFIGURATION.
282 0x03, 0x03, // MCMCFG239 - MCMCFG CONFIGURATION.
283 0x04, 0x04, // MCMCFG240 - MCMCFG CONFIGURATION.
284 0x05, 0x05, // MCMCFG241 - MCMCFG CONFIGURATION.
285 0x06, 0x06, // MCMCFG242 - MCMCFG CONFIGURATION.
286 0x07, 0x07, // MCMCFG243 - MCMCFG CONFIGURATION.
287 0x08, 0x08, // MCMCFG244 - MCMCFG CONFIGURATION.
288 0x09, 0x09, // MCMCFG245 - MCMCFG CONFIGURATION.
289 0x0A, 0x0A, // MCMCFG246 - MCMCFG CONFIGURATION.
290 0x0B, 0x0B, // MCMCFG247 - MCMCFG CONFIGURATION.
291 0x0C, 0x0C, // MCMCFG248 - MCMCFG CONFIGURATION.
292 0x0D, 0x0D, // MCMCFG249 - MCMCFG CONFIGURATION.
293 0x0E, 0x0E, // MCMCFG250 - MCMCFG CONFIGURATION.
294 0x0F, 0x0F, // MCMCFG251 - MCMCFG CONFIGURATION.
295 0x10, 0x10, // MCMCFG252 - MCMCFG CONFIGURATION.
296 0x11, 0x11, // MCMCFG253 - MCMCFG CONFIGURATION.
297 0x12, 0x12, // MCMCFG254 - MCMCFG CONFIGURATION.
298 0x13, 0x13, // MCMCFG255 - MCMCFG CONFIGURATION.
299 0x14, 0x14, // MCMCFG256 - MCMCFG CONFIGURATION.
300 0x15, 0x15, // MCMCFG257 - MCMCFG CONFIGURATION.
301 0x16, 0x16, // MCMCFG258 - MCMCFG CONFIGURATION.
302 0x17, 0x17, // MCMCFG259 - MCMCFG CONFIGURATION.
303 0x18, 0x18, // MCMCFG260 - MCMCFG CONFIGURATION.
304 0x19, 0x19, // MCMCFG261 - MCMCFG CONFIGURATION.
305 0x1A, 0x1A, // MCMCFG262 - MCMCFG CONFIGURATION.
306 0x1B, 0x1B, // MCMCFG263 - MCMCFG CONFIGURATION.
307 0x1C, 0x1C, // MCMCFG264 - MCMCFG CONFIGURATION.
308 0x1D, 0x1D, // MCMCFG265 - MCMCFG CONFIGURATION.
309 0x1E, 0x1E, // MCMCFG266 - MCMCFG CONFIGURATION.
310 0x1F, 0x1F, // MCMCFG267 - MCMCFG CONFIGURATION.
311 0x20, 0x20, // MCMCFG268 - MCMCFG CONFIGURATION.
312 0x21, 0x21, // MCMCFG269 - MCMCFG CONFIGURATION.
313 0x22, 0x22, // MCMCFG270 - MCMCFG CONFIGURATION.
314 0x23, 0x23, // MCMCFG271 - MCMCFG CONFIGURATION.
315 0x24, 0x24, // MCMCFG272 - MCMCFG CONFIGURATION.
316 0x25, 0x25, // MCMCFG273 - MCMCFG CONFIGURATION.
317 0x26, 0x26, // MCMCFG274 - MCMCFG CONFIGURATION.
318 0x27, 0x27, // MCMCFG275 - MCMCFG CONFIGURATION.
319 0x28, 0x28, // MCMCFG276 - MCMCFG CONFIGURATION.
320 0x29, 0x29, // MCMCFG277 - MCMCFG CONFIGURATION.
321 0x2A, 0x2A, // MCMCFG278 - MCMCFG CONFIGURATION.
322 0x2B, 0x2B, // MCMCFG279 - MCMCFG CONFIGURATION.
323 0x2C, 0x2C, // MCMCFG280 - MCMCFG CONFIGURATION.
324 0x2D, 0x2D, // MCMCFG281 - MCMCFG CONFIGURATION.
325 0x2E, 0x2E, // MCMCFG282 - MCMCFG CONFIGURATION.
326 0x2F, 0x2F, // MCMCFG283 - MCMCFG CONFIGURATION.
327 0x30, 0x30, // MCMCFG284 - MCMCFG CONFIGURATION.
328 0x31, 0x31, // MCMCFG285 - MCMCFG CONFIGURATION.
329 0x32, 0x32, // MCMCFG286 - MCMCFG CONFIGURATION.
330 0x33, 0x33, // MCMCFG287 - MCMCFG CONFIGURATION.
331 0x34, 0x34, // MCMCFG288 - MCMCFG CONFIGURATION.
332 0x35, 0x35, // MCMCFG289 - MCMCFG CONFIGURATION.
333 0x36, 0x36, // MCMCFG290 - MCMCFG CONFIGURATION.
334 0x37, 0x37, // MCMCFG291 - MCMCFG CONFIGURATION.
335 0x38, 0x38, // MCMCFG292 - MCMCFG CONFIGURATION.
336 0x39, 0x39, // MCMCFG293 - MCMCFG CONFIGURATION.
337 0x3A, 0x3A, // MCMCFG294 - MCMCFG CONFIGURATION.
338 0x3B, 0x3B, // MCMCFG295 - MCMCFG CONFIGURATION.
339 0x3C, 0x3C, // MCMCFG296 - MCMCFG CONFIGURATION.
340 0x3D, 0x3D, // MCMCFG297 - MCMCFG CONFIGURATION.
341 0x3E, 0x3E, // MCMCFG298 - MCMCFG CONFIGURATION.
342 0x3F, 0x3F, // MCMCFG299 - MCMCFG CONFIGURATION.
343 0x40, 0x40, // MCMCFG300 - MCMCFG CONFIGURATION.
344 0x41, 0x41, // MCMCFG301 - MCMCFG CONFIGURATION.
345 0x42, 0x42, // MCMCFG302 - MCMCFG CONFIGURATION.
346 0x43, 0x43, // MCMCFG303 - MCMCFG CONFIGURATION.
347 0x44, 0x44, // MCMCFG304 - MCMCFG CONFIGURATION.
348 0x45, 0x45, // MCMCFG305 - MCMCFG CONFIGURATION.
349 0x46, 0x46, // MCMCFG306 - MCMCFG CONFIGURATION.
350 0x47, 0x47, // MCMCFG307 - MCMCFG CONFIGURATION.
351 0x48, 0x48, // MCMCFG308 - MCMCFG CONFIGURATION.
352 0x49, 0x49, // MCMCFG309 - MCMCFG CONFIGURATION.
353 0x4A, 0x4A, // MCMCFG310 - MCMCFG CONFIGURATION.
354 0x4B, 0x4B, // MCMCFG311 - MCMCFG CONFIGURATION.
355 0x4C, 0x4C, // MCMCFG312 - MCMCFG CONFIGURATION.
356 0x4D, 0x4D, // MCMCFG313 - MCMCFG CONFIGURATION.
357 0x4E, 0x4E, // MCMCFG314 - MCMCFG CONFIGURATION.
358 0x4F, 0x4F, // MCMCFG315 - MCMCFG CONFIGURATION.
359 0x50, 0x50, // MCMCFG316 - MCMCFG CONFIGURATION.
360 0x51, 0x51, // MCMCFG317 - MCMCFG CONFIGURATION.
361 0x52, 0x52, // MCMCFG318 - MCMCFG CONFIGURATION.
362 0x53, 0x53, // MCMCFG319 - MCMCFG CONFIGURATION.
363 0x54, 0x54, // MCMCFG320 - MCMCFG CONFIGURATION.
364 0x55, 0x55, // MCMCFG321 - MCMCFG CONFIGURATION.
365 0x56, 0x56, // MCMCFG322 - MCMCFG CONFIGURATION.
366 0x57, 0x57, // MCMCFG323 - MCMCFG CONFIGURATION.
367 0x58, 0x58, // MCMCFG324 - MCMCFG CONFIGURATION.
368 0x59, 0x59, // MCMCFG325 - MCMCFG CONFIGURATION.
369 0x5A, 0x5A, // MCMCFG326 - MCMCFG CONFIGURATION.
370 0x5B, 0x5B, // MCMCFG327 - MCMCFG CONFIGURATION.
371 0x5C, 0x5C, // MCMCFG328 - MCMCFG CONFIGURATION.
372 0x5D, 0x5D, // MCMCFG329 - MCMCFG CONFIGURATION.
373 0x5E, 0x5E, // MCMCFG330 - MCMCFG CONFIGURATION.
374 0x5F, 0x5F, // MCMCFG331 - MCMCFG CONFIGURATION.
375 0x60, 0x60, // MCMCFG332 - MCMCFG CONFIGURATION.
376 0x61, 0x61, // MCMCFG333 - MCMCFG CONFIGURATION.
377 0x62, 0x62, // MCMCFG334 - MCMCFG CONFIGURATION.
378 0x63, 0x63, // MCMCFG335 - MCMCFG CONFIGURATION.
379 0x64, 0x64, // MCMCFG336 - MCMCFG CONFIGURATION.
380 0x65, 0x65, // MCMCFG337 - MCMCFG CONFIGURATION.
381 0x66, 0x66, // MCMCFG338 - MCMCFG CONFIGURATION.
382 0x67, 0x67, // MCMCFG339 - MCMCFG CONFIGURATION.
383 0x68, 0x68, // MCMCFG340 - MCMCFG CONFIGURATION.
384 0x69, 0x69, // MCMCFG341 - MCMCFG CONFIGURATION.
385 0x6A, 0x6A, // MCMCFG342 - MCMCFG CONFIGURATION.
386 0x6B, 0x6B, // MCMCFG343 - MCMCFG CONFIGURATION.
387 0x6C, 0x6C, // MCMCFG344 - MCMCFG CONFIGURATION.
388 0x6D, 0x6D, // MCMCFG345 - MCMCFG CONFIGURATION.
389 0x6E, 0x6E, // MCMCFG346 - MCMCFG CONFIGURATION.
390 0x6F, 0x6F, // MCMCFG347 - MCMCFG CONFIGURATION.
391 0x70, 0x70, // MCMCFG348 - MCMCFG CONFIGURATION.
392 0x71, 0x71, // MCMCFG349 - MCMCFG CONFIGURATION.
393 0x72, 0x72, // MCMCFG350 - MCMCFG CONFIGURATION.
394 0x73, 0x73, // MCMCFG351 - MCMCFG CONFIGURATION.
395 0x74, 0x74, // MCMCFG352 - MCMCFG CONFIGURATION.
396 0x75, 0x75, // MCMCFG353 - MCMCFG CONFIGURATION.
397 0x76, 0x76, // MCMCFG354 - MCMCFG CONFIGURATION.
398 0x77, 0x77, // MCMCFG355 - MCMCFG CONFIGURATION.
399 0x78, 0x78, // MCMCFG356 - MCMCFG CONFIGURATION.
400 0x79, 0x79, // MCMCFG357 - MCMCFG CONFIGURATION.
401 0x7A, 0x7A, // MCMCFG358 - MCMCFG CONFIGURATION.
402 0x7B, 0x7B, // MCMCFG359 - MCMCFG CONFIGURATION.
403 0x7C, 0x7C, // MCMCFG360 - MCMCFG CONFIGURATION.
404 0x7D, 0x7D, // MCMCFG361 - MCMCFG CONFIGURATION.
405 0x7E, 0x7E, // MCMCFG362 - MCMCFG CONFIGURATION.
406 0x7F, 0x7F, // MCMCFG363 - MCMCFG CONFIGURATION.
407 0x80, 0x80, // MCMCFG364 - MCMCFG CONFIGURATION.
408 0x81, 0x81, // MCMCFG365 - MCMCFG CONFIGURATION.
409 0x82, 0x82, // MCMCFG366 - MCMCFG CONFIGURATION.
410 0x83, 0x83, // MCMCFG367 - MCMCFG CONFIGURATION.
411 0x84, 0x84, // MCMCFG368 - MCMCFG CONFIGURATION.
412 0x85, 0x85, // MCMCFG369 - MCMCFG CONFIGURATION.
413 0x86, 0x86, // MCMCFG370 - MCMCFG CONFIGURATION.
414 0x87, 0x87, // MCMCFG371 - MCMCFG CONFIGURATION.
415 0x88, 0x88, // MCMCFG372 - MCMCFG CONFIGURATION.
416 0x89, 0x89, // MCMCFG373 - MCMCFG CONFIGURATION.
417 0x8A, 0x8A, // MCMCFG374 - MCMCFG CONFIGURATION.
418 0x8B, 0x8B, // MCMCFG375 - MCMCFG CONFIGURATION.
419 0x8C, 0x8C, // MCMCFG376 - MCMCFG CONFIGURATION.
420 0x8D, 0x8D, // MCMCFG377 - MCMCFG CONFIGURATION.
421 0x8E, 0x8E, // MCMCFG378 - MCMCFG CONFIGURATION.
422 0x8F, 0x8F, // MCMCFG379 - MCMCFG CONFIGURATION.
423 0x90, 0x90, // MCMCFG380 - MCMCFG CONFIGURATION.
424 0x91, 0x91, // MCMCFG381 - MCMCFG CONFIGURATION.
425 0x92, 0x92, // MCMCFG382 - MCMCFG CONFIGURATION.
426 0x93, 0x93, // MCMCFG383 - MCMCFG CONFIGURATION.
427 0x94, 0x94, // MCMCFG384 - MCMCFG CONFIGURATION.
428 0x95, 0x95, // MCMCFG385 - MCMCFG CONFIGURATION.
429 0x96, 0x96, // MCMCFG386 - MCMCFG CONFIGURATION.
430 0x97, 0x97, // MCMCFG387 - MCMCFG CONFIGURATION.
431 0x98, 0x98, // MCMCFG388 - MCMCFG CONFIGURATION.
432 0x99, 0x99, // MCMCFG389 - MCMCFG CONFIGURATION.
433 0x9A, 0x9A, // MCMCFG390 - MCMCFG CONFIGURATION.
434 0x9B, 0x9B, // MCMCFG391 - MCMCFG CONFIGURATION.
435 0x9C, 0x9C, // MCMCFG392 - MCMCFG CONFIGURATION.
436 0x9D, 0x9D, // MCMCFG393 - MCMCFG CONFIGURATION.
437 0x9E, 0x9E, // MCMCFG3
```

3. Select a Preferred Setting to start with:



4. Set your base frequency and Channel Bandwidth:



5. And set your target channel:

X-tal frequency: 26.000000 MHz
 RF output power: 0 dBm ☐ PA ramping
 Phase: 1 Datarate: 249.938965 kBaud Modulation: MSK ☐ Manchester
 RF frequency: 2419.995148 MHz Channel: 333.251953 kHz Channel number: 60 RX filterbandwidth: 541.666667 kHz

Preferred settings:

Datarate	Deviation	Modulation	RX filterbandwidth	Optimization
2.4 kBaud	38 kHz	2-FSK	203 kHz	Sensitivity
2.4 kBaud	38 kHz	2-FSK	203 kHz	Current 0 - 85 C
10 kBaud	38 kHz	2-FSK	232 kHz	Sensitivity
10 kBaud	38 kHz	2-FSK	232 kHz	Current
250 kBaud	1	MSK	540 kHz	Sensitivity
250 kBaud	1	MSK	540 kHz	Current
500 kBaud	0	MSK	812 kHz	Sensitivity

6. Export your Code:

Calculation Window - CC2500 - SmartRF® Studio

File Settings Help

- Reset configuration...
- Open configuration...
- Save configuration...
- Load CC2500 state...
- Save CC2500 state...
- Export CC2500 registers...
- Import CC2500 registers...
- Export CC2500 code...
- Close

Chip revision: E (VERSION = 0x03)
 X-tal frequency: 26.000000 MHz RF output power: 0 dBm ☐ PA ramping
 Phase: 1 Datarate: 249.938965 kBaud Modulation: MSK ☐ Manchester
 RF frequency: 2419.995148 MHz Channel: 333.251953 kHz Channel number: 60 RX filterbandwidth: 541.666667 kHz

Preferred settings:

Datarate	Deviation	Modulation	RX filterbandwidth	Optimization
2.4 kBaud	38 kHz	2-FSK	203 kHz	Sensitivity
2.4 kBaud	38 kHz	2-FSK	203 kHz	Current 0 - 85 C
10 kBaud	38 kHz	2-FSK	232 kHz	Sensitivity
10 kBaud	38 kHz	2-FSK	232 kHz	Current
250 kBaud	1	MSK	540 kHz	Sensitivity
250 kBaud	1	MSK	540 kHz	Current
500 kBaud	0	MSK	812 kHz	Sensitivity

Correlation:

Register Components

PA value = 0xFE
 RF output power -> PATABLE
 FREQ2 = 0x5C
 RF Frequency -> FREQ[23:16]
 FREQ1 = 0x4E
 RF Frequency -> FREQ[15:8]
 FREQ0 = 0xC5
 RF Frequency -> FREQ[7:0]
 FSCTRL1 = 0x0A
 IF Frequency -> FREQ_IF[4:0] => 253.91 kHz
 FSCTRL0 = 0x00
 RF Frequency offset -> FREQOFF[7:0]
 MDMCFG4 = 0x2D
 Data rate (exponent) -> DRATE_E
 Channel bandwidth (exponent) -> CHANBW_E
 Channel bandwidth (mantissa) -> CHANBW_M
 MDMCFG3 = 0x3B
 Data rate (mantissa) -> DRATE_M
 MDMCFG2 = 0x73

Reset CC2500 and write settings Copy settings to Register View

Simple RX Simple TX Packet RX Packet TX PER test

Length config: Variable Sync word: 30/32 sy Address config: No addr ☒ CRC ☐ Manual Init

Packet length: 61 Packet count: 200 Address: ☐ FIFO Autoflush

View format: Hex

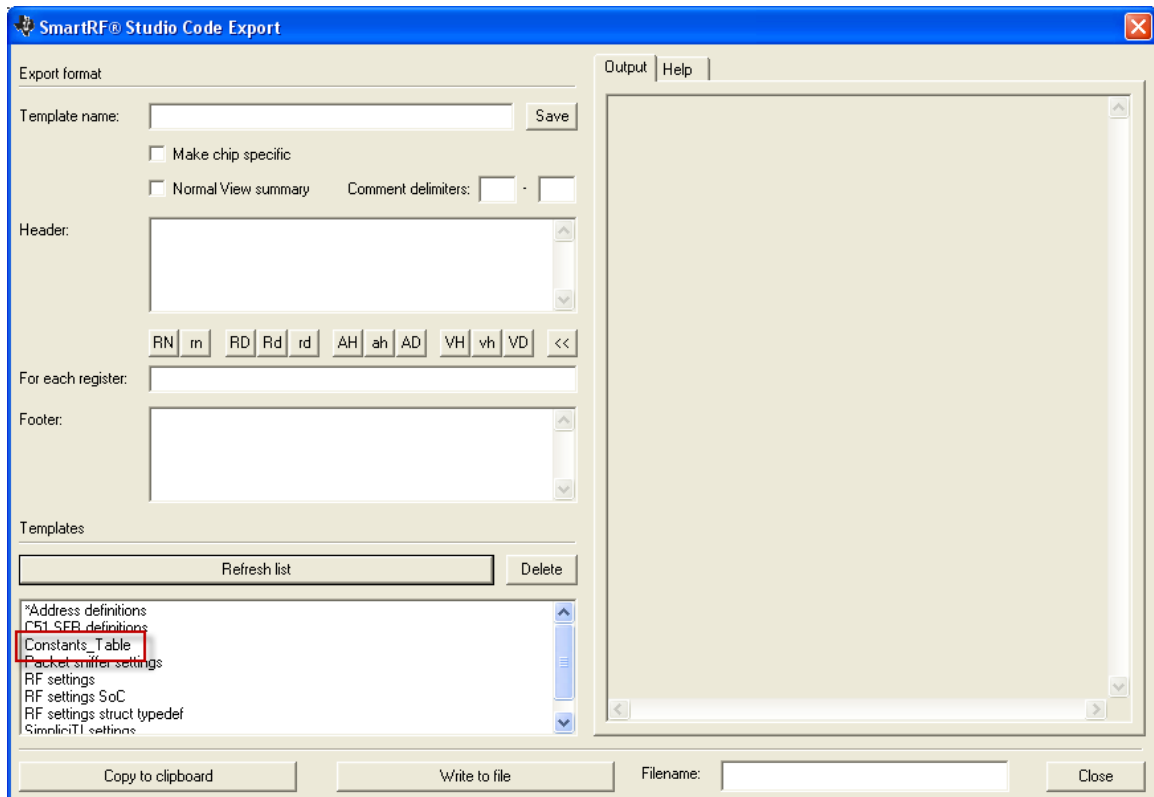
File dump:

Start buffered RX Stop RX

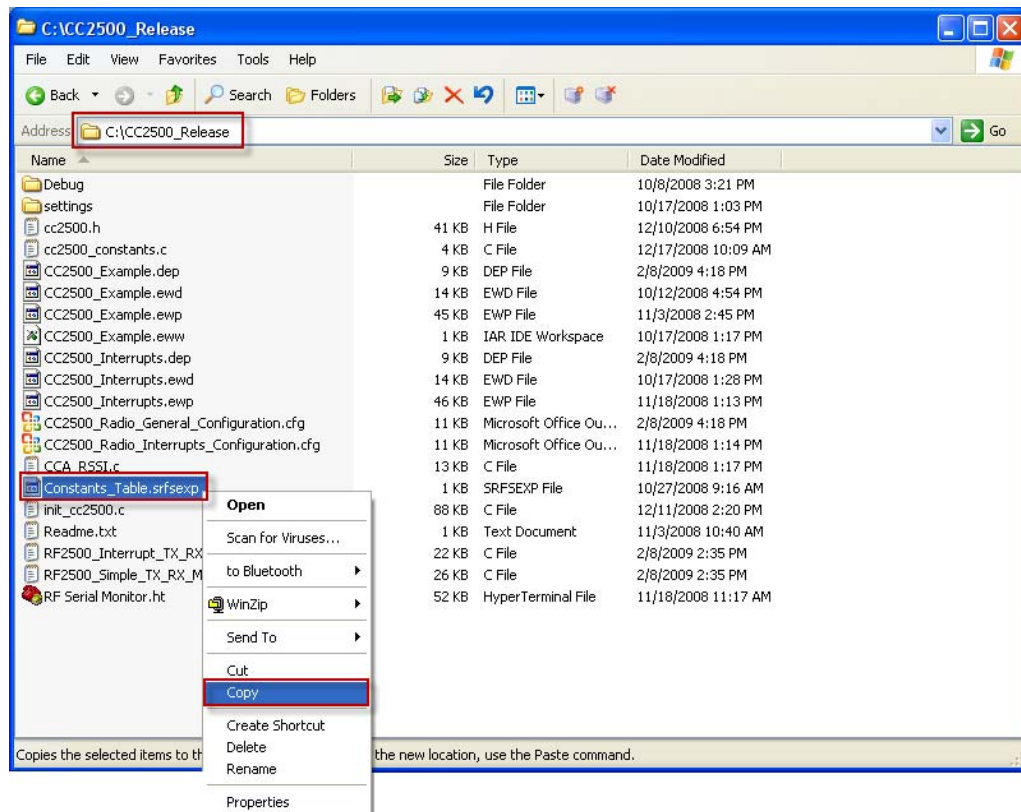
MDMCFG1 = 0x20
 Forward Error Correction -> FEC_EN
 MDMCFG2 = 0x03
 Sync mode -> SYNC_MODE[2:0]
 PKTCTRL0 = 0x05
 Packet format -> PKT_FORMAT[5:4]
 Forced to 0 by Pw/
 CRC operation -> CRC_EN[2]
 Forced to 1 by Pw/
 Packet config -> LENGTH_CONFIG[1:0]
 Forced to 1 by Pw/

Device ID: Not Connected Last executed command: Date: 02.02.2009, Time: 15:01:42

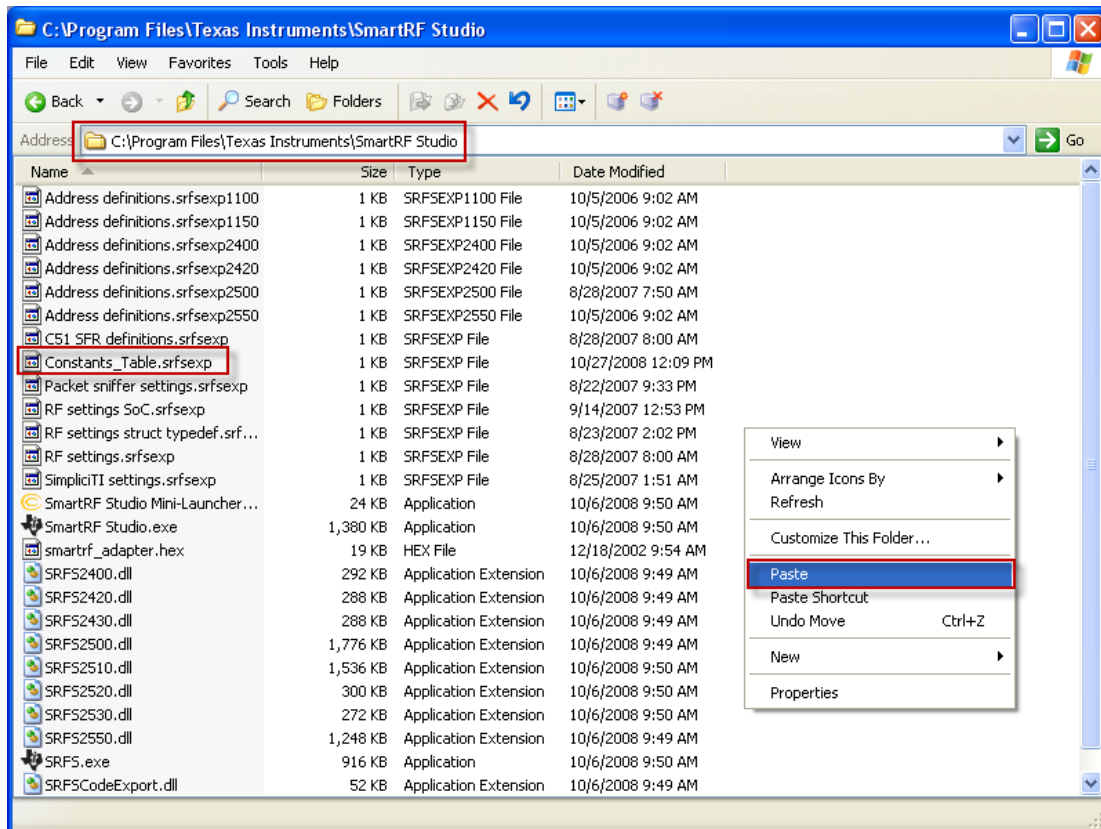
7. Importing a Settings Template



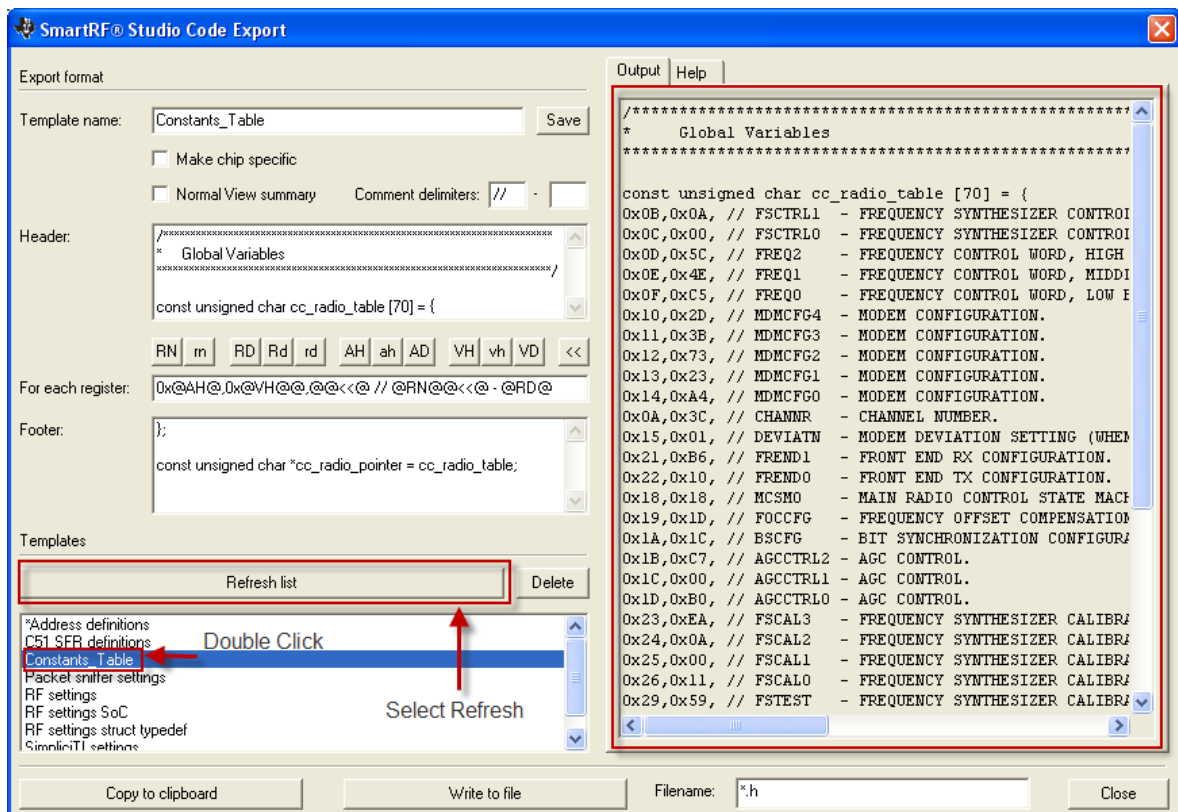
8. Copy Constants_Table.srfsexp in the CC2500_Release Directory:



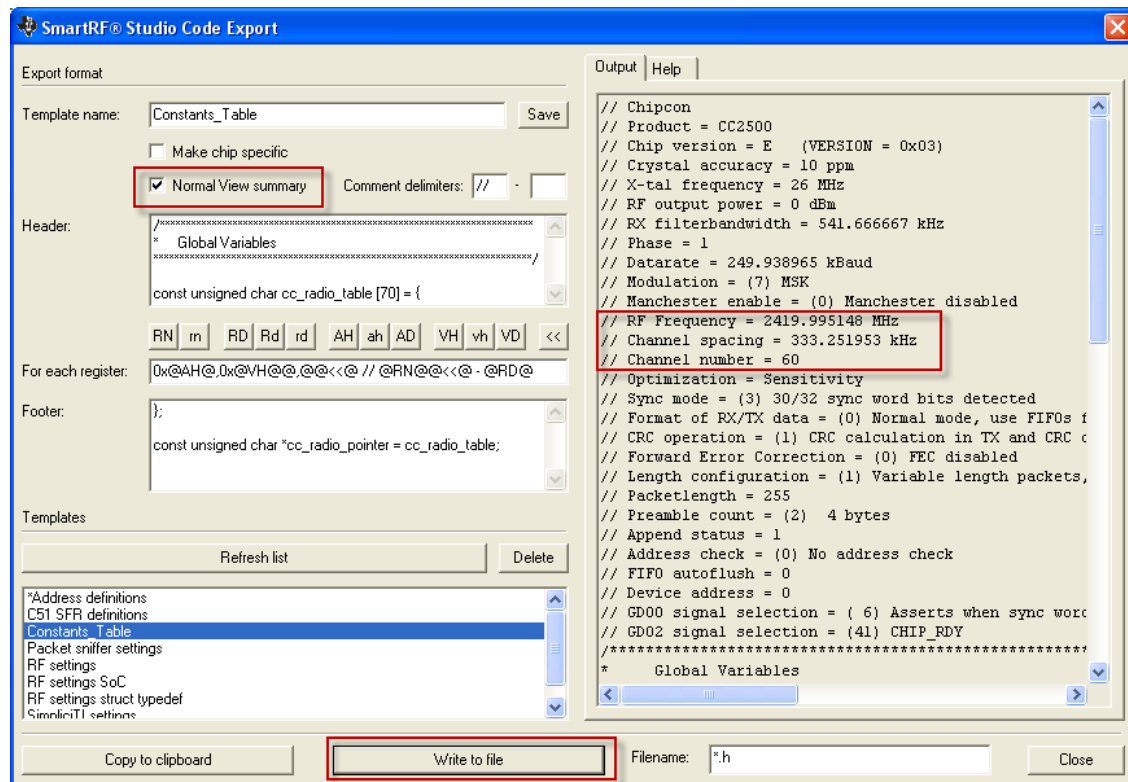
9. And paste it in the SmartRF Studio Directory:



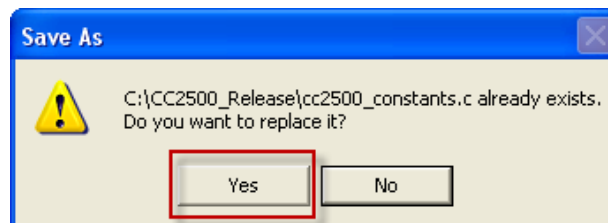
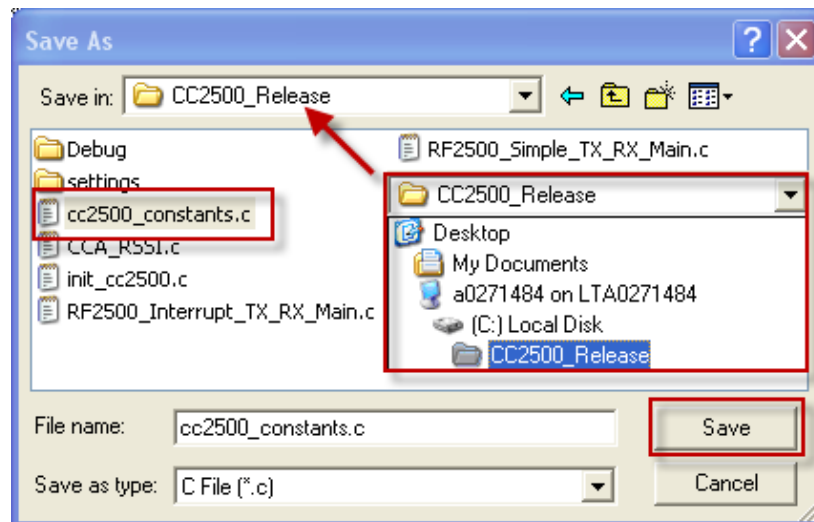
10. Refresh the list for the Constants_Table to appear. Double click on it to use the template to generate the code:



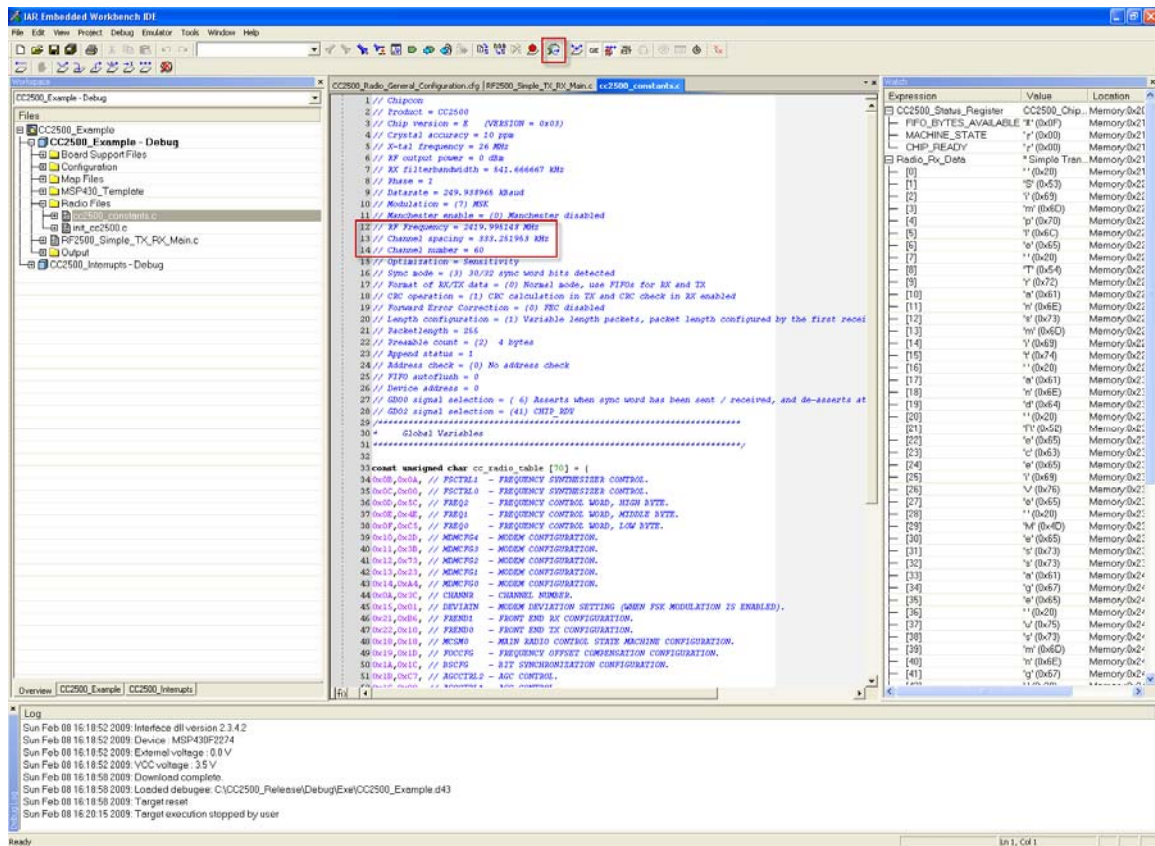
11. Select Normal View summary to show the radio settings. Then select Write to file to update the code:



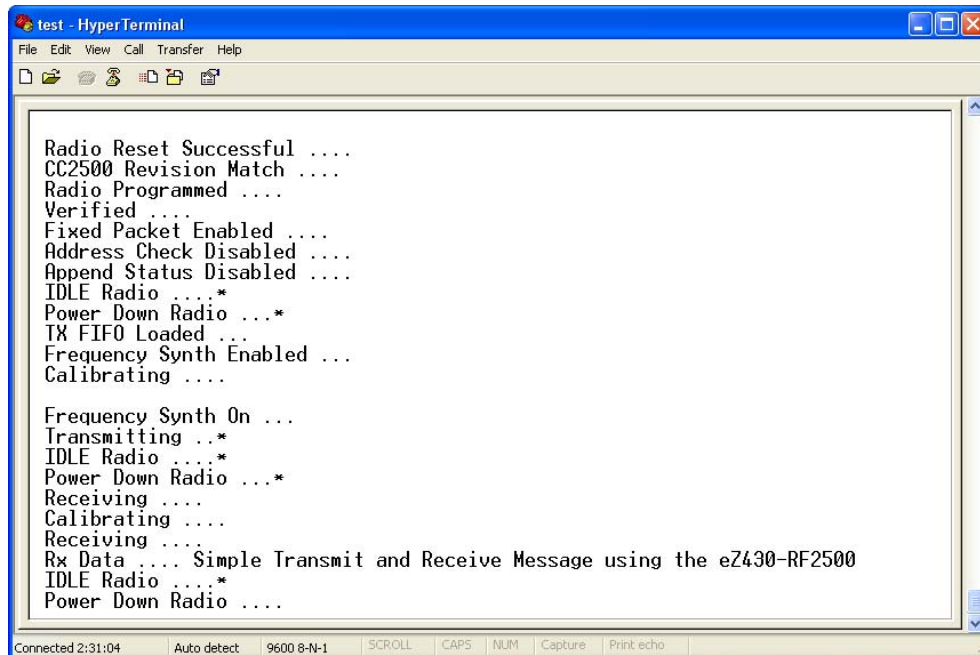
12. Overwrite cc2500_constants.c:



Was the file overwritten?



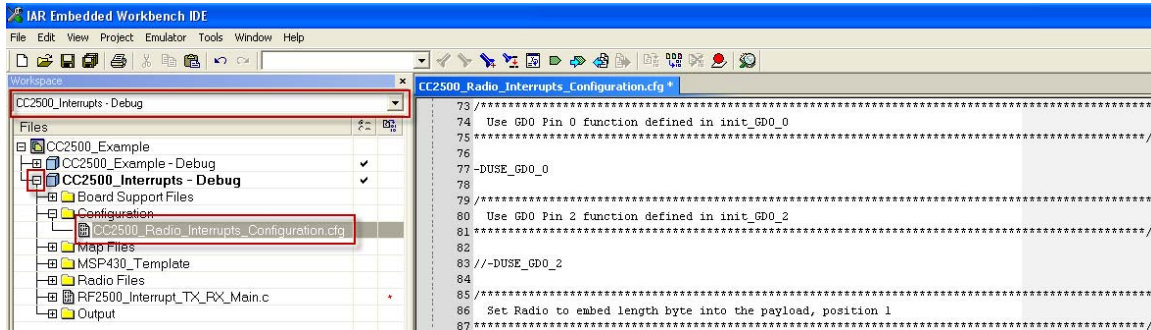
13. Rebuild both devices. The functionality of Lab 1 should occur but with less interference from your neighbor.





Lab 3: Eliminating the State Machine

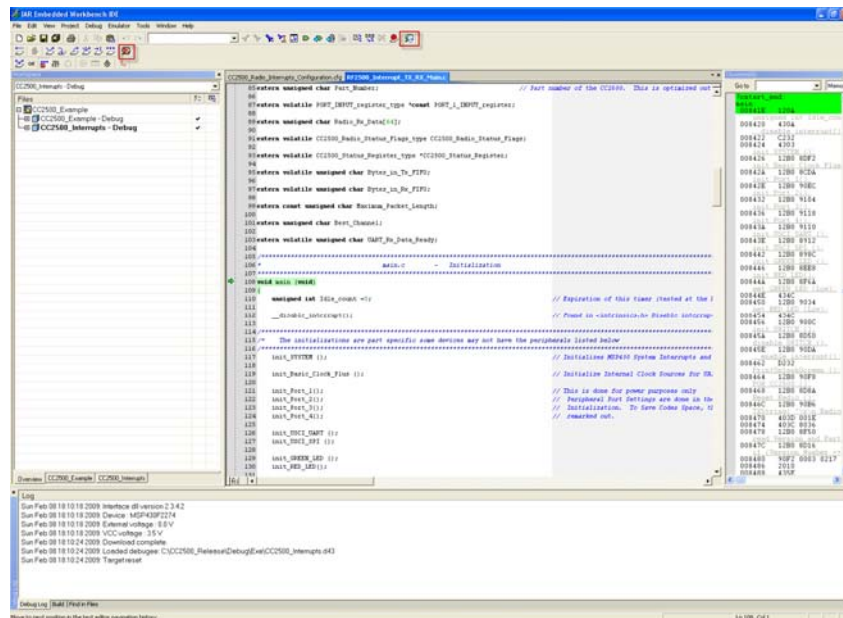
1. Select the CC2500_Interrupts – Debug Project and open the CC2500_Radio_Interrupts_Configurations.cfg file:



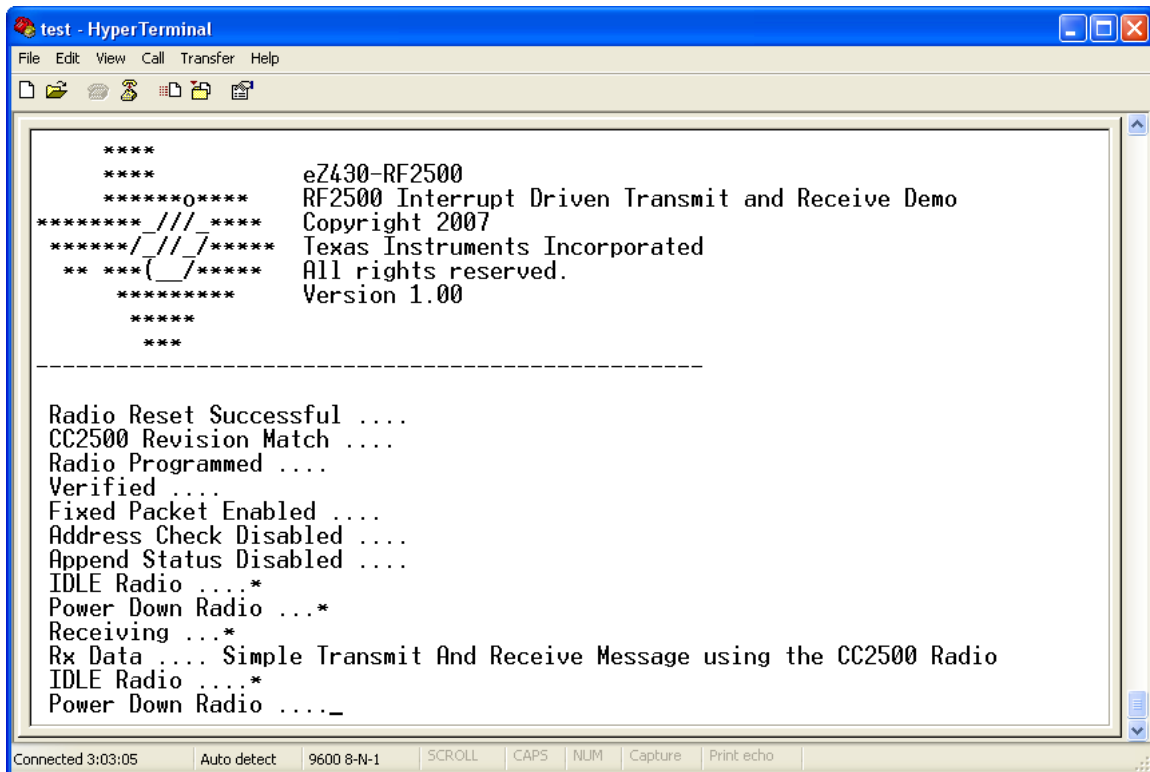
2. Set in CC2500_Radio_Interrupts_Configurations.cfg:

```
-DUSE_GDO_0
// -DUSE_GDO_2
// -DVARIBLE_PACKET
-DFIXED_PACKET
// -DALLOW_ADDRESSING
// -DAPPEND_STATUS
// -DFORCE_TRANSMIT
// -DFORCE_RECEIVE
-DTERMINAL
// -DVERIFY
// -DDEBUG
// -DMAX_POWER
```

3. Build & Download the Code into both devices and run the steps in Lab 1:



Output to the hyperterminal:



```
****
****
*****O*****
*****///*****
*****//*****
** ***(//*****
*****
*****
*****
*****

eZ430-RF2500
RF2500 Interrupt Driven Transmit and Receive Demo
Copyright 2007
Texas Instruments Incorporated
All rights reserved.
Version 1.00

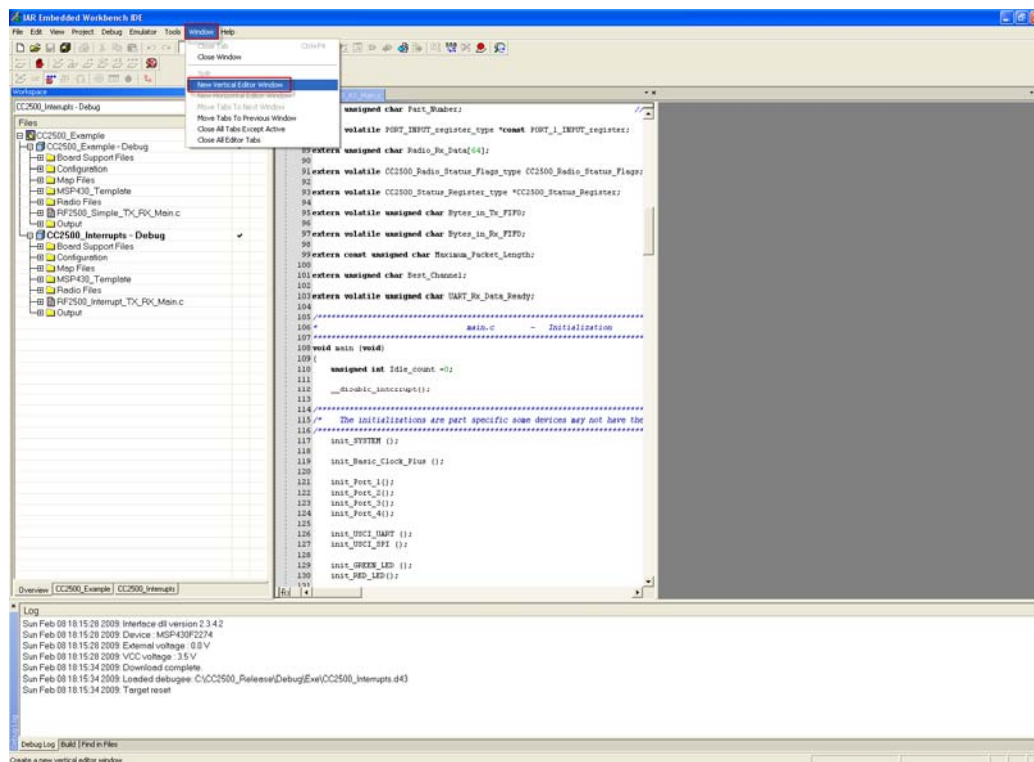
-----

Radio Reset Successful ....
CC2500 Revision Match ....
Radio Programmed ....
Verified ....
Fixed Packet Enabled ....
Address Check Disabled ....
Append Status Disabled ....
IDLE Radio ....*
Power Down Radio ...*
Receiving ...*
Rx Data .... Simple Transmit And Receive Message using the CC2500 Radio
IDLE Radio ....*
Power Down Radio ...._

Connected 3:03:05    Auto detect    9600 8-N-1    SCROLL    CAPS    NUM    Capture    Print echo
```

What changed?

4. Add a vertical window:



5. Compare the Interrupt Main versus the Simple Main:

```
344 set_RED_LED(Low);
345 set_GREEN_LED(Low);
346
347 else
348 {
349 //Transmit Update
350 TXString("Tx TX FIFO Loaded ....", 21);
351
352 //Transmit Update
353 TXString("Tx TX FIFO Loaded ....", 21);
354
355 if (CC2500_Radio_Status_Flags.TRANSMITTER_START_MODE == Transmitter_Start)
356 {
357 //Transmit Update
358 TXString("Tx TX Frequency Synch Enabled ....", 30);
359
360 //Transmit Update
361 TXString("Tx TX Frequency Synch Enabled ....", 30);
362
363 Enable_Radio_Tx (1);
364
365 TXString("Tx TX Transmitting ....", 19);
366
367 CC2500_Radio_Status_Flags.TRANSMITTER_START_MODE = Transmitter_Stop;
368
369 //Transmit Update
370 TXString("Tx TX Transmitting ....", 19);
371
372 if (GDO_0 is used to signal successful packet tran
373 CC2500_Radio_Status_Flags.PACKET_COMPLETE = Set;
374
375 #endif /* USE_GDO_0 */
376
377 CC2500_Radio_Status_Flags.RADIO_MODE = Radio_Is_Transmitting;
378
379 Idle_Count = 0;
380
381 if (CC2500_Radio_Status_Flags.TRANSMITTER_START_MODE == Receive_Start)
382 {
383 //Transmit Update
384 TXString("Tx TX Receiving ....", 17);
385
386 Enable_Radio_Rx (1);
387
388 CC2500_Radio_Status_Flags.TRANSMITTER_START_MODE = Transmitter_Stop;
389
390 //Transmit Update
391 TXString("Tx TX Receiving ....", 17);
392
393 CC2500_Radio_Status_Flags.PACKET_COMPLETE = Set;
394
395 #endif /* USE_GDO_0 */
396
397 CC2500_Radio_Status_Flags.RADIO_MODE = Radio_Is_Receiving;
398
399 Idle_Count = 0;
400
401 if (Idle_Count >= 0x0032)
402 {
403 //Transmit Update
404 TXString("Tx TX Power Down Radio ....", 25);
405
406 }
407
408 }
409
410 }
411
412 }
413
414 }
415
416 }
417
418 }
419
420 }
421
422 }
423
424 }
425
426 }
427
428 }
429
430 }
431
432 }
433
434 }
435
436 }
437
438 }
439
440 }
441
442 }
443
444 }
445
446 }
447
448 }
449
450 }
451
452 }
```

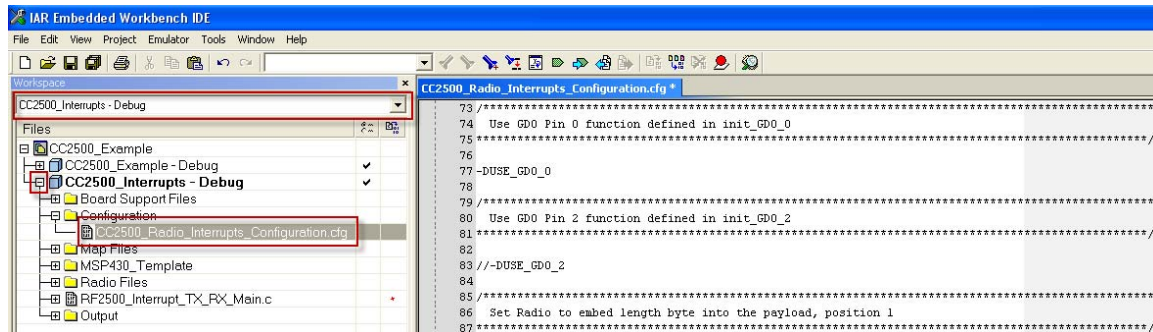
Interrupt Service Routine Location:

```
523 // _bic_SR_register_on_exit (0x10);
524 // Pull CPU out of sleep mode on RETI
525 // Debounce Pin
526 // Clear ISR Flag
527
528 if (PORT_2_INTERRUPT_FLAG_register->PIN_5)
529 {
530 // _bic_SR_register_on_exit (0x10);
531 // Pull CPU out of sleep mode on RETI
532 // Debounce Pin
533 // Clear ISR Flag
534
535 if (PORT_2_INTERRUPT_FLAG_register->PIN_6)
536 {
537 // _bic_SR_register_on_exit (0x10);
538 // Pull CPU out of sleep mode on RETI
539
540 PORT_2_INTERRUPT_EDGE_SELECT_register->PIN_6 = Low_to_High;
541 CC2500_Radio_Status_Flags.PACKET_COMPLETE = Set;
542 Synch_Used_Transmitted_Received = Clear;
543
544 if (PORT_2_INTERRUPT_FLAG_register->PIN_6 == High)
545 {
546 PORT_2_INTERRUPT_EDGE_SELECT_register->PIN_6 = High_to_Low;
547 Synch_Used_Transmitted_Received = Set;
548
549 }
550
551 for (unsigned int i=0; i<=0192; i++)
552 PORT_2_INTERRUPT_FLAG_register->PIN_6 = Clear;
553
554 if (PORT_2_INTERRUPT_FLAG_register->PIN_7)
555 {
556 static unsigned int transmission_count = 0;
557 // _bic_SR_register_on_exit (0x10);
558 // Pull CPU out of sleep mode on RETI
559
560 CC2500_Radio_Status_Flags.FIL_LOCKED = Set;
561
562 transmission_count++;
563
564 for (unsigned int i=0; i<=0192; i++)
565 PORT_2_INTERRUPT_FLAG_register->PIN_7 = Clear;
566
567 }
568 }
```




Lab 4: Adding Radio Information

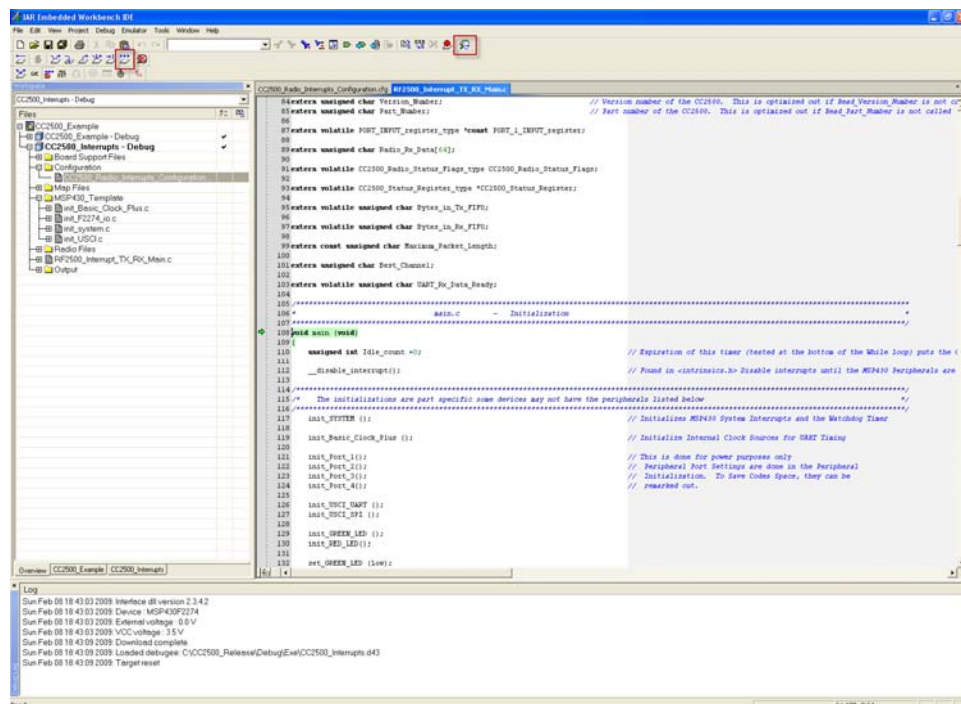
1. Select the CC2500_Interrupts – Debug Project and open the CC2500_Radio_Interrupts_Configurations.cfg file:



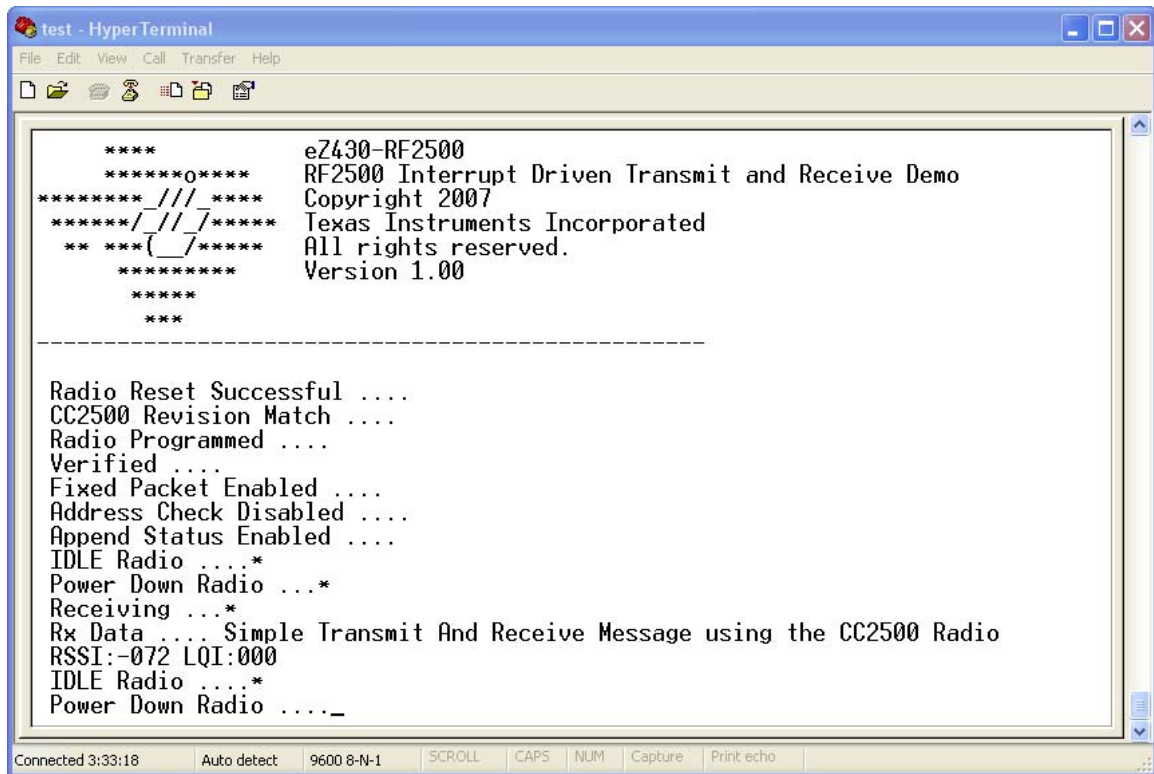
2. Set in CC2500_Radio_Interrupts_Configurations.cfg:

```
-DUSE_GDO_0
//-DUSE_GDO_2
//-DVARIBLE_PACKET
-DFIXED_PACKET
//-DALLOW_ADDRESSING
-DAPPEND_STATUS
//-DFORCE_TRANSMIT
//-DFORCE_RECEIVE
-DTERMINAL
//-DVERIFY
//-DDEBUG
//-DMAX_POWER
```

3. Rebuild both boards and execute the code as in Lab 1:



Output to Hyperterminal:



```
test - HyperTerminal
File Edit View Call Transfer Help

****
*****0****
*****_///_****
*****/_///_****
** ***(_/****
*****
*****
***

-----

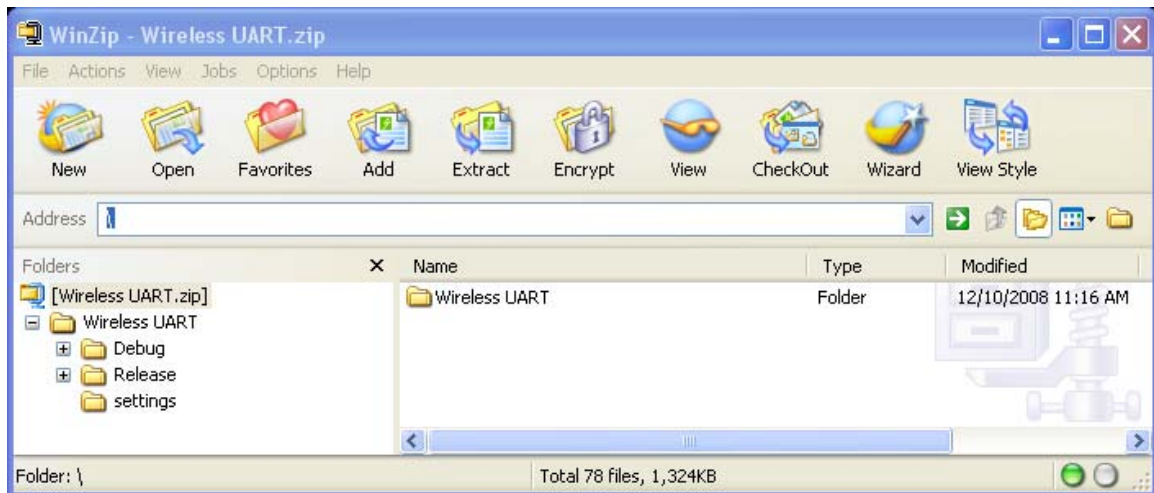
Radio Reset Successful ....
CC2500 Revision Match ....
Radio Programmed ....
Verified ....
Fixed Packet Enabled ....
Address Check Disabled ....
Append Status Enabled ....
IDLE Radio ....*
Power Down Radio ...*
Receiving ...*
Rx Data .... Simple Transmit And Receive Message using the CC2500 Radio
RSSI:-072 LQI:000
IDLE Radio ....*
Power Down Radio ....._

Connected 3:33:18   Auto detect   9600 8-N-1   SCROLL   CAPS   NUM   Capture   Print echo
```

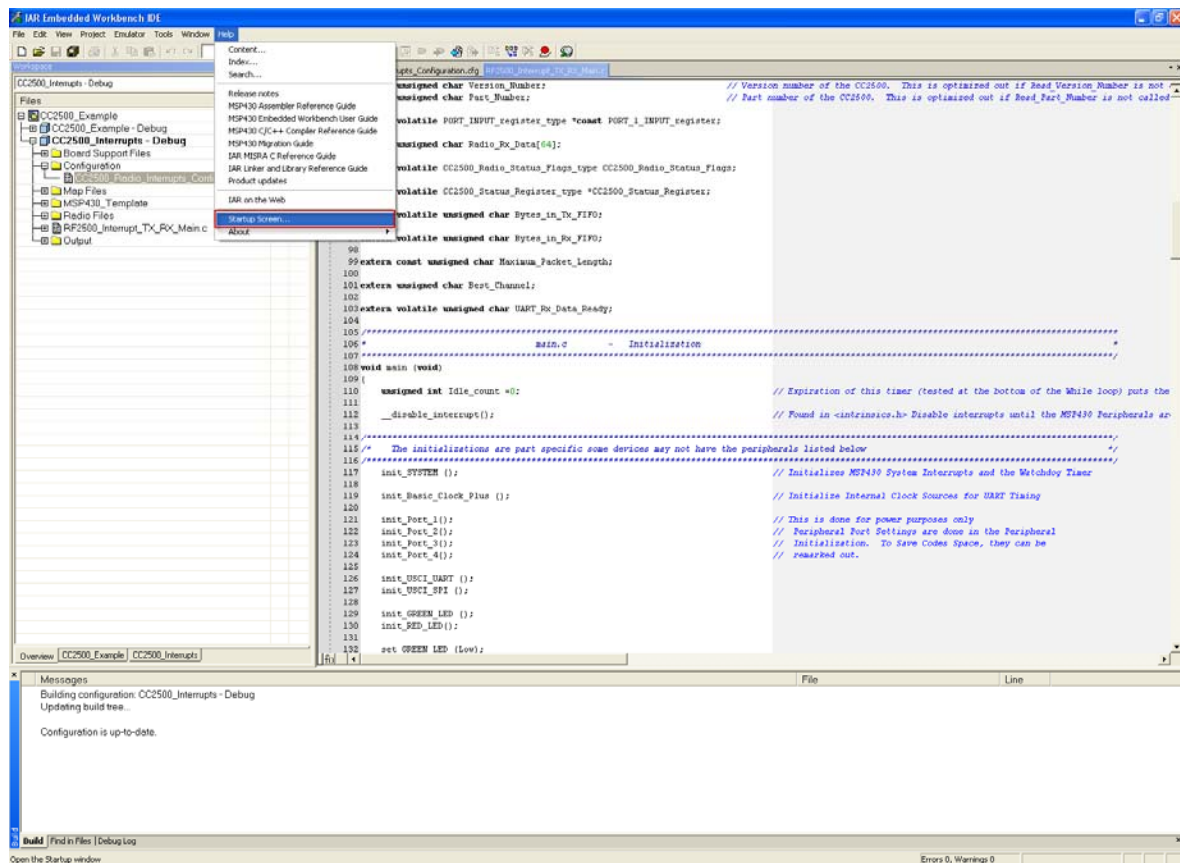



Lab 5: Wireless UART

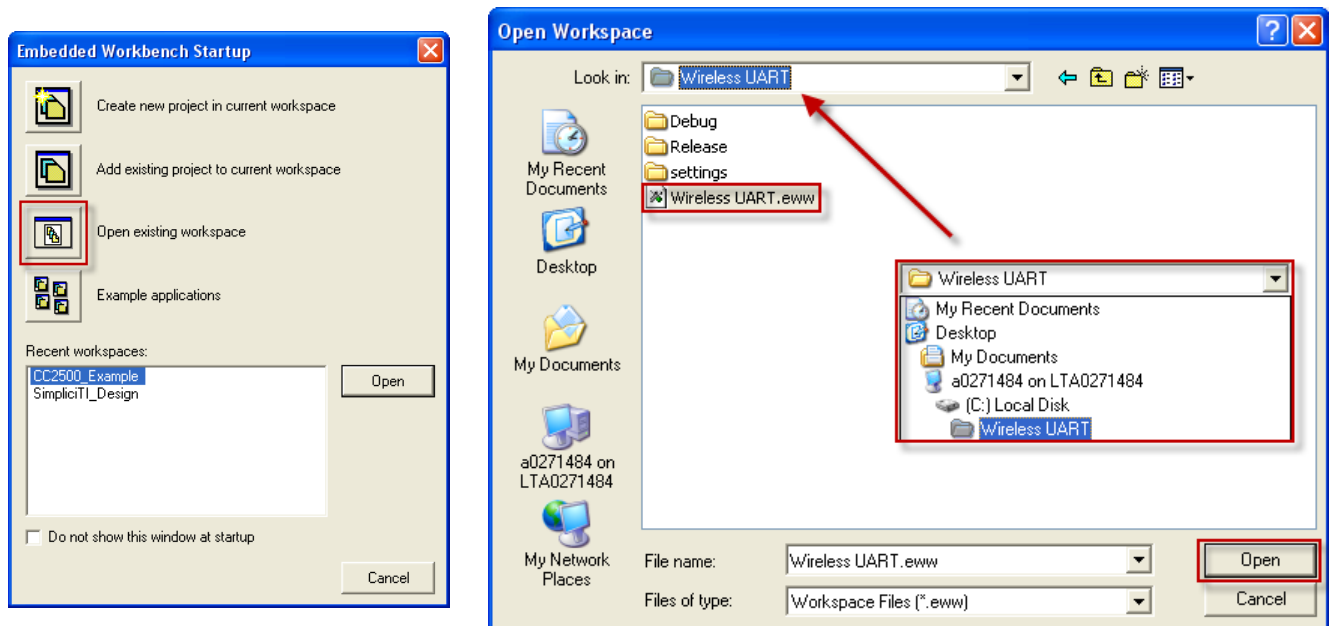
1. Unzip Wireless UART.zip



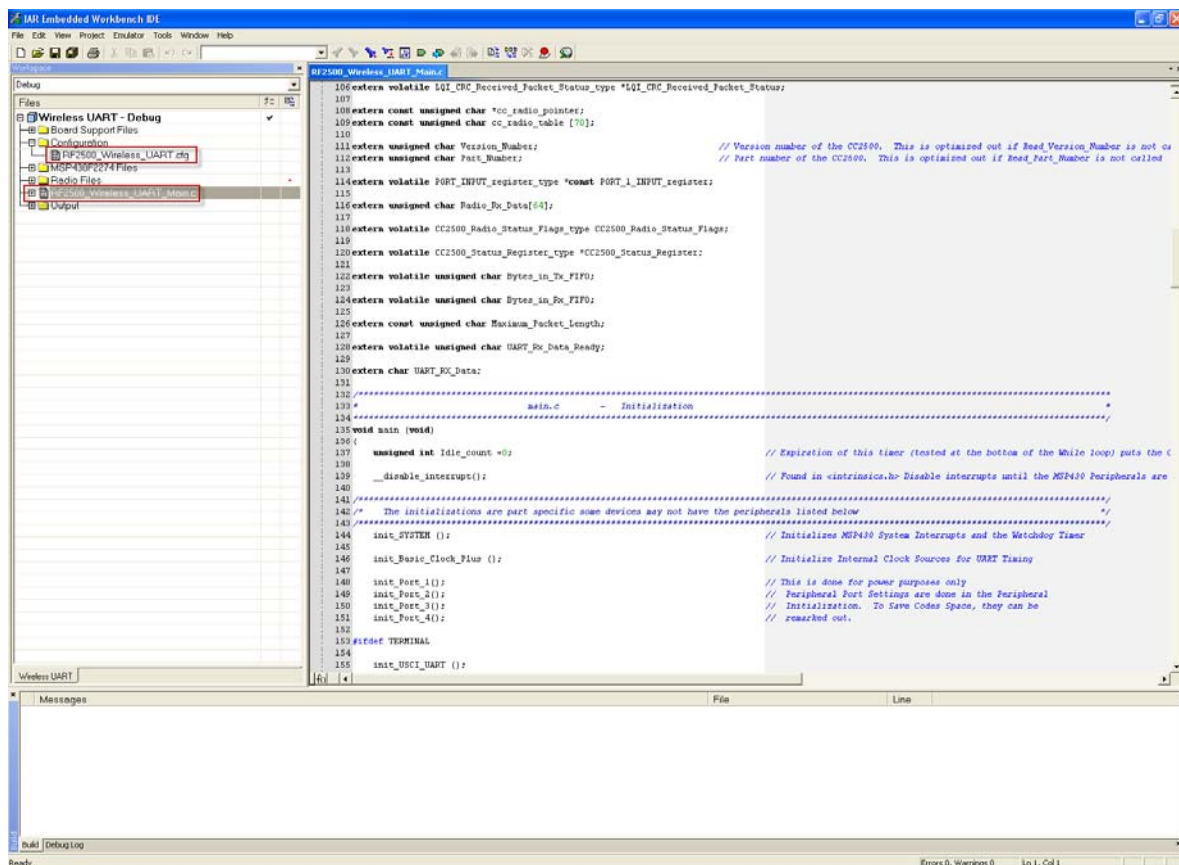
2. To start the New Workspace Wizard, make sure to Stop Debugging, then go to Help->Startup Screen



3. Open an existing workspace:



The Wireless UART is the General Radio project with a dedicated Main file and specific configuration file.



The Configuration is set as such:

-DUSE_GDO_0
// -DUSE_GDO_2

-DVARIABLE_PACKET
// -DFIXED_PACKET

-DALLOW_ADDRESSING

// -DAPPEND_STATUS

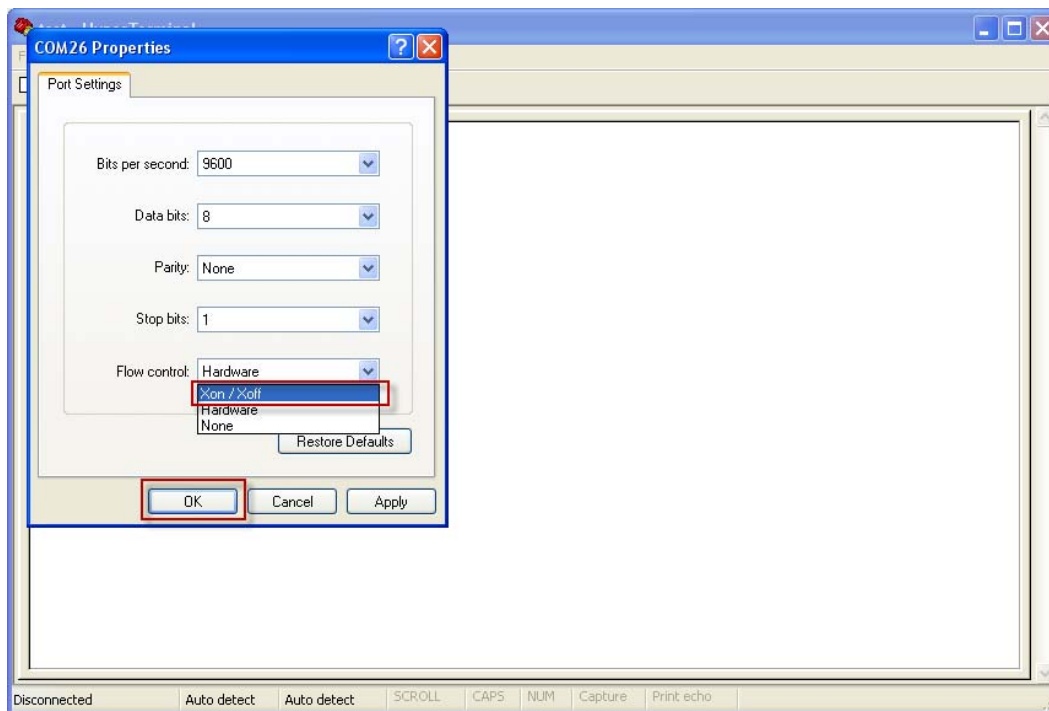
// -DFORCE_TRANSMIT
// -DFORCE_RECEIVE

-DTERMINAL

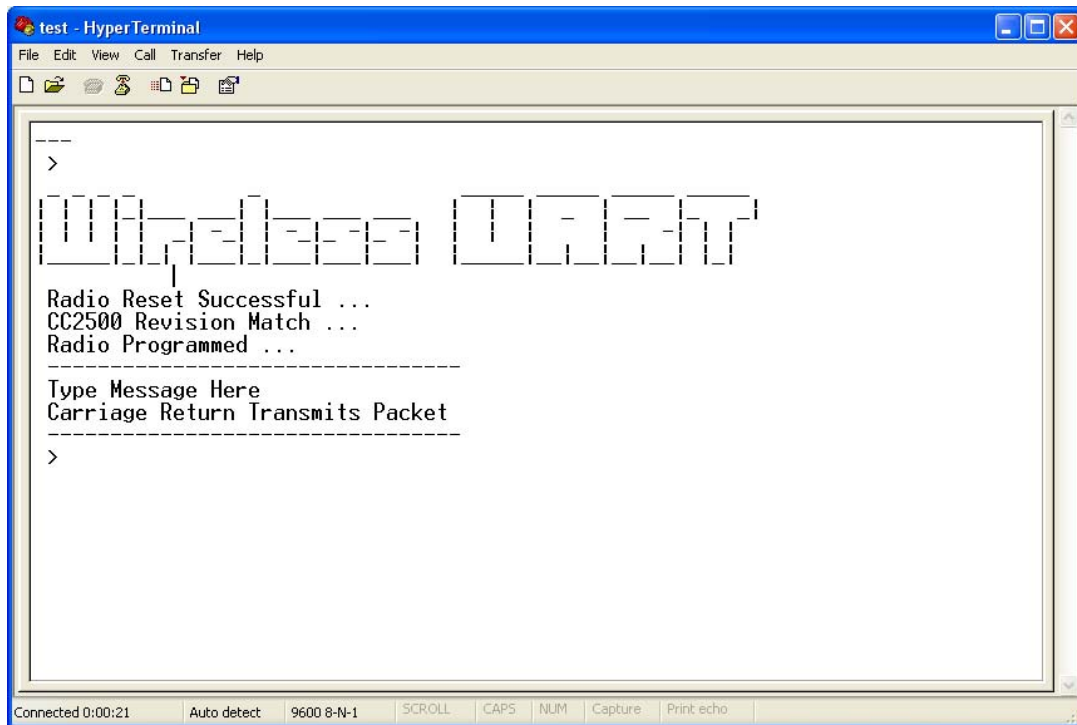
// -DVERIFY
// -DDEBUG

-DMAX_POWER

4. This lab needs 2 eZ430-RF2500 boards, both set up on HyperTerminal. The HyperTerminal session will need software flow control:



HyperTerminal Welcome Screen:



Typing a message on one Terminal application followed by a Carriage Return will send the message.