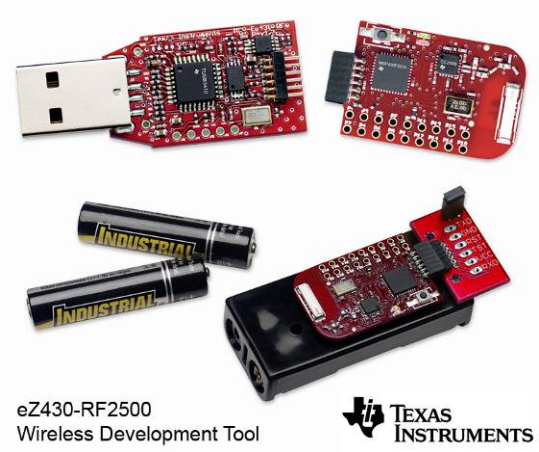
General Radio 1 Day Workshop, Lab Guide

Getting Started: What do you need?

Hardware: *eZ430-RF2500 Evaluation Board*

http://focus.ti.com/docs/toolsw/folders/print/ez430-rf2500.html



Software:

1. *IAR Embedded Workbench MSP430*

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

2. *SmartRF Studio*

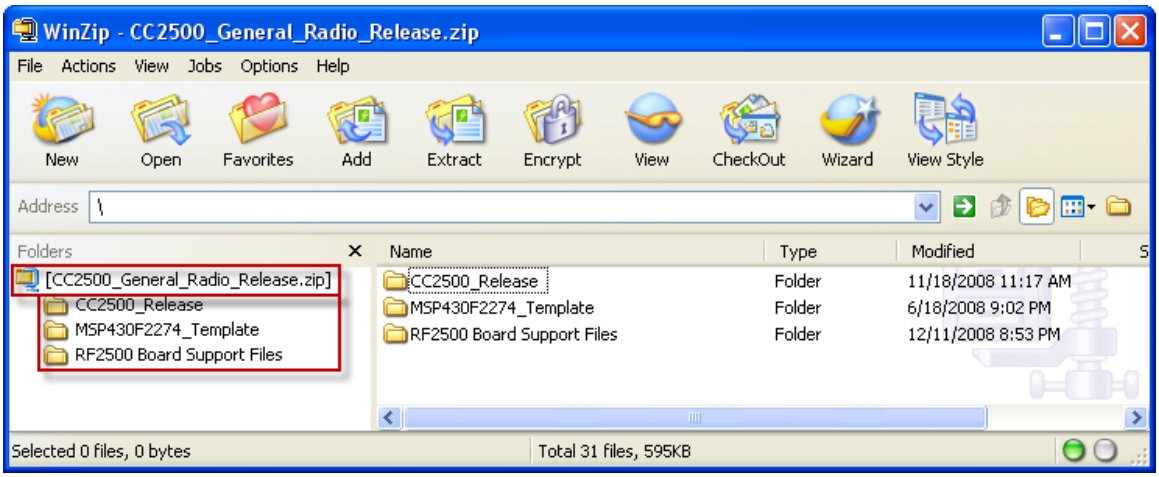
http://focus.ti.com/docs/toolsw/folders/print/smartrftm-studio.html

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

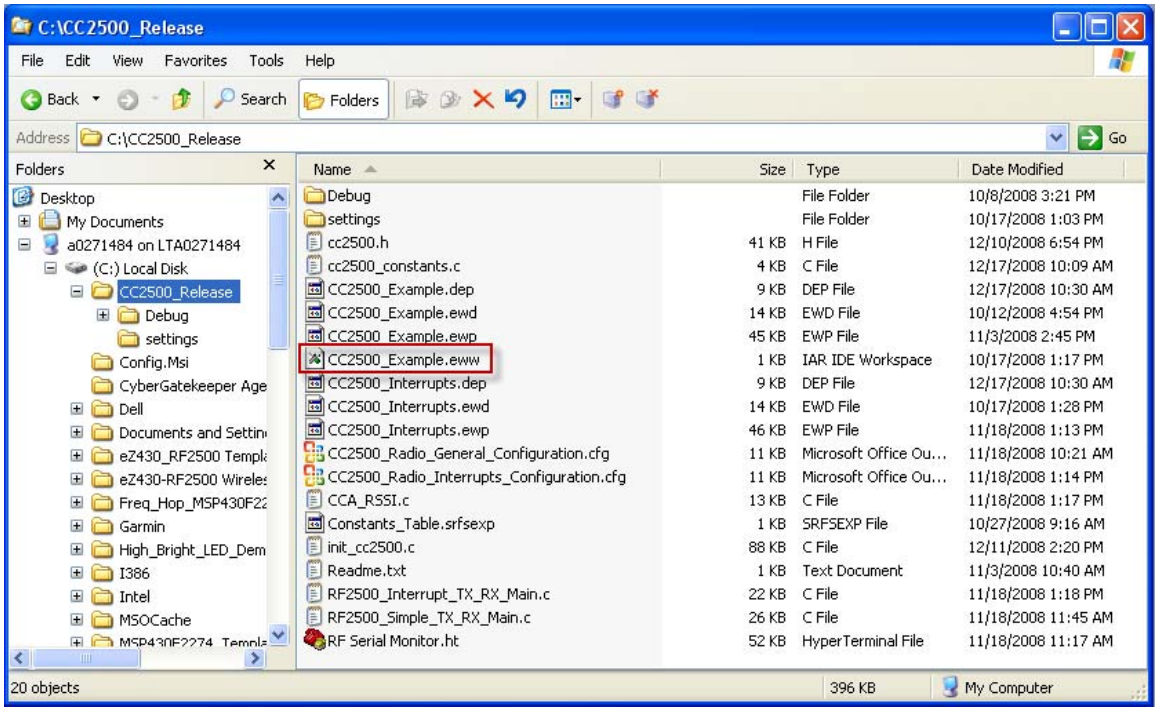
Here is the content of CC2500_General_Radio_Release.zip (Hit View Style Button for this View):



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

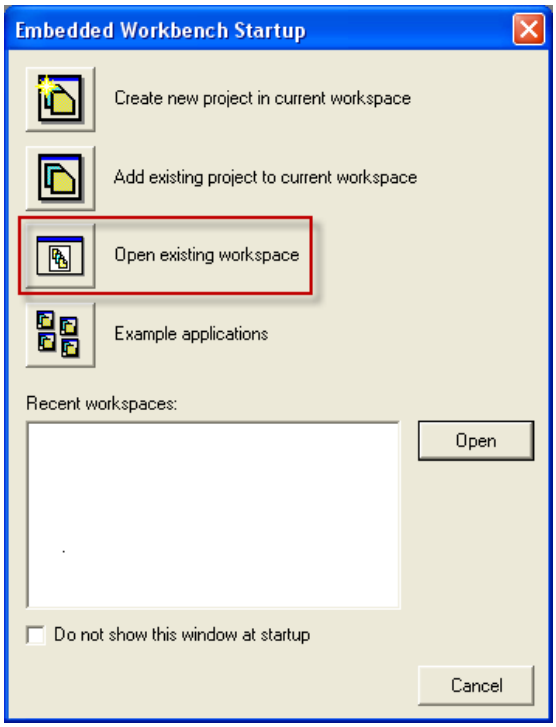
Once the .zip file is extracted, an IAR Embedded Workspace is located in the C:\CC2500_Release directory, ***CC2500_Example.eww***

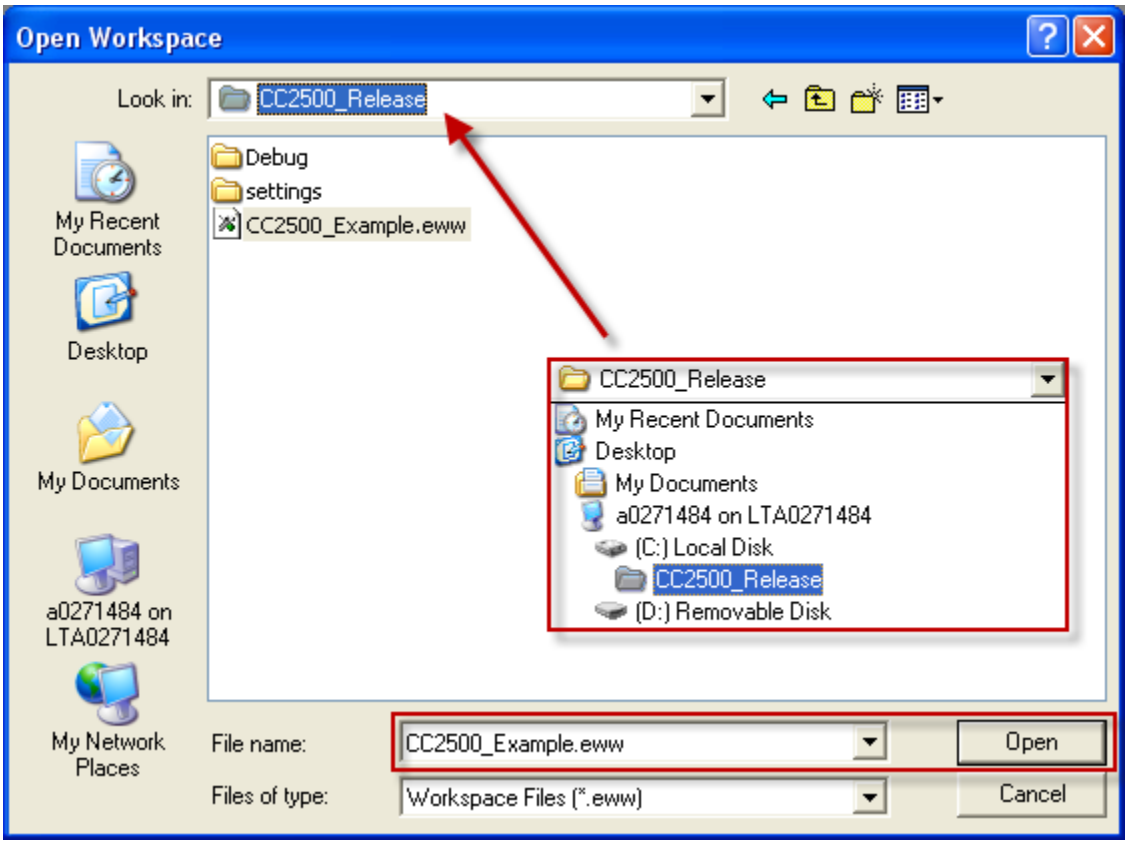


Getting Started with IAR Studio

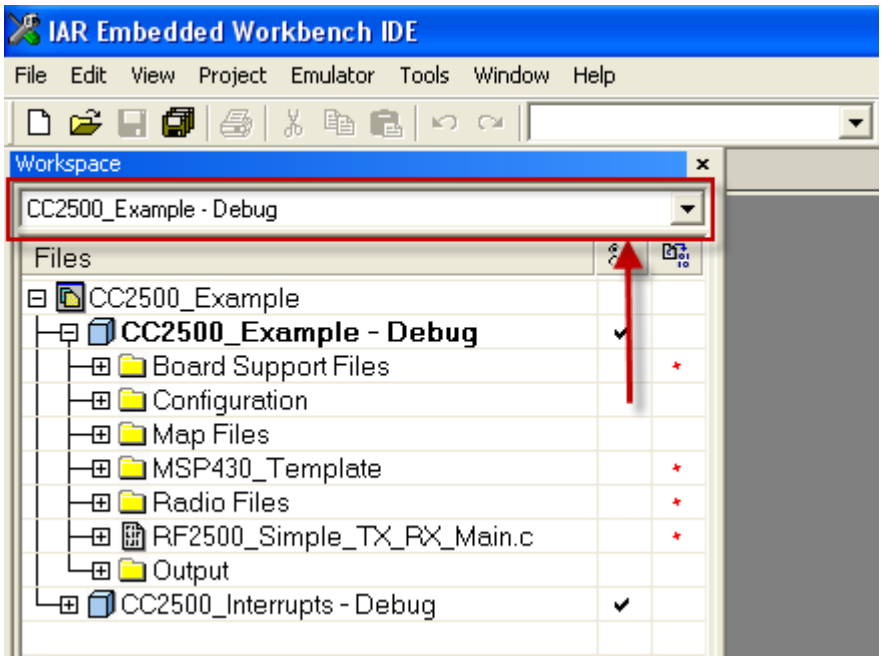
IAR EW can be found here:

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

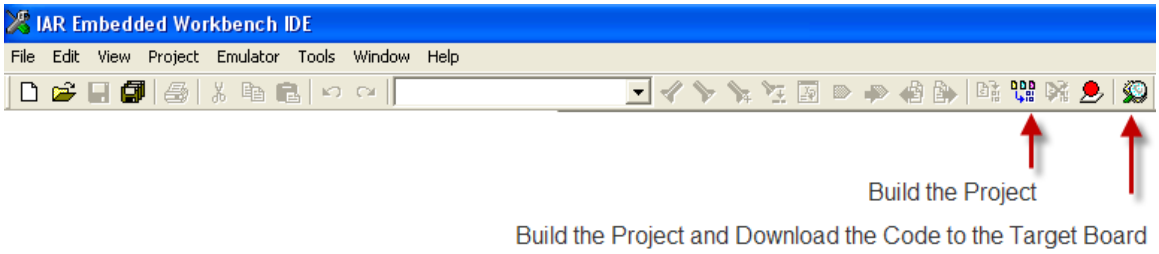




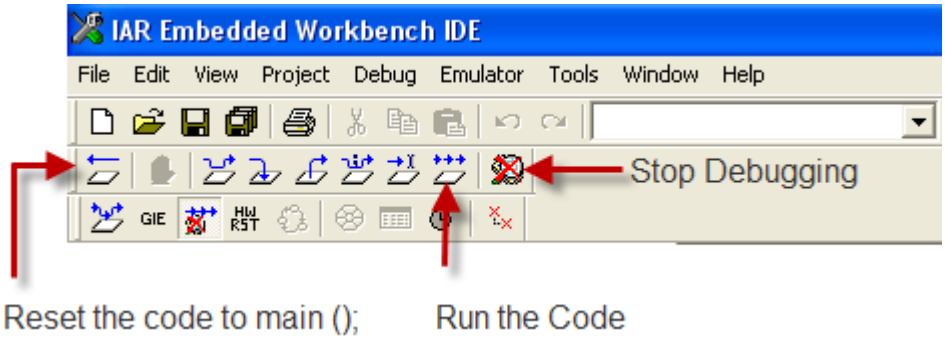
Two projects exist in this workspace. Use the pull down box shown below to change the project.



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

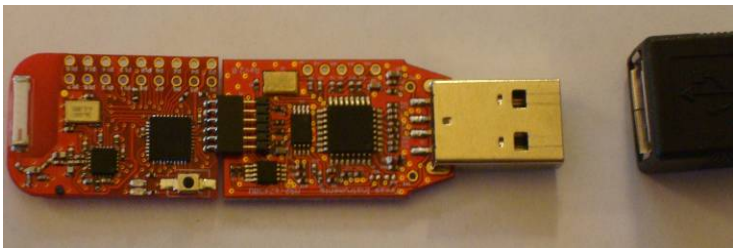


Use these Icons to control your code:

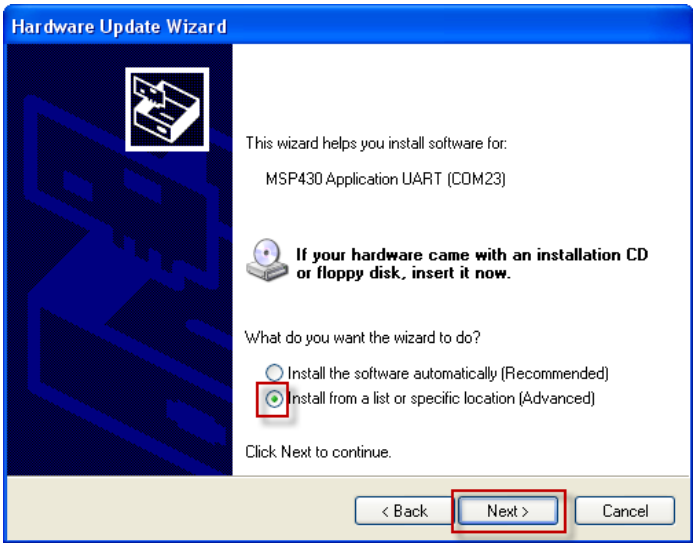


* Board needs to be attached to the FET Tool to Download Code

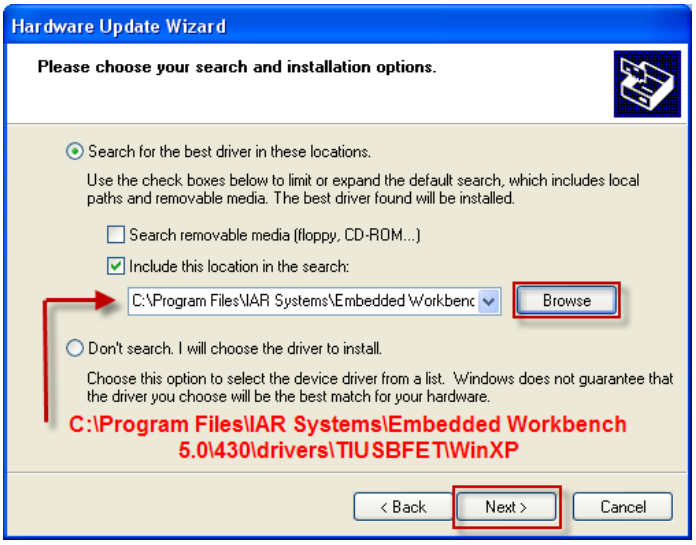
Connecting the eZ430-RF2500 Board



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:**

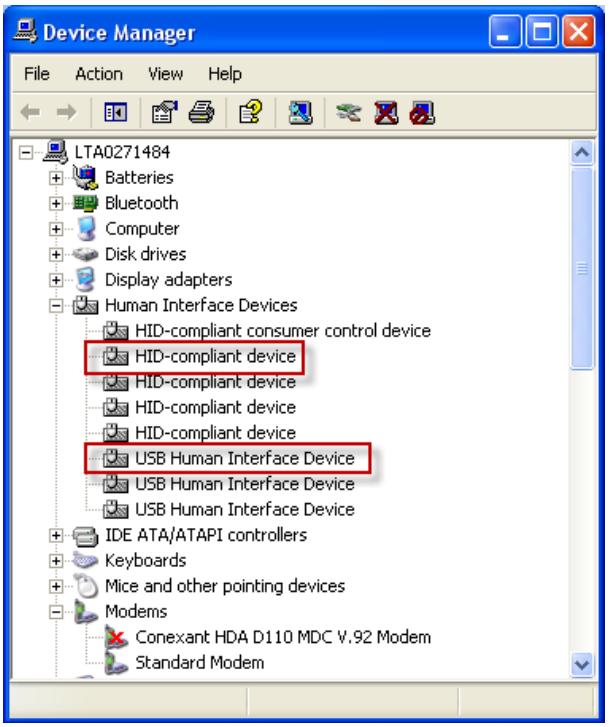


Select install from a specific location, add the path below into the include path.

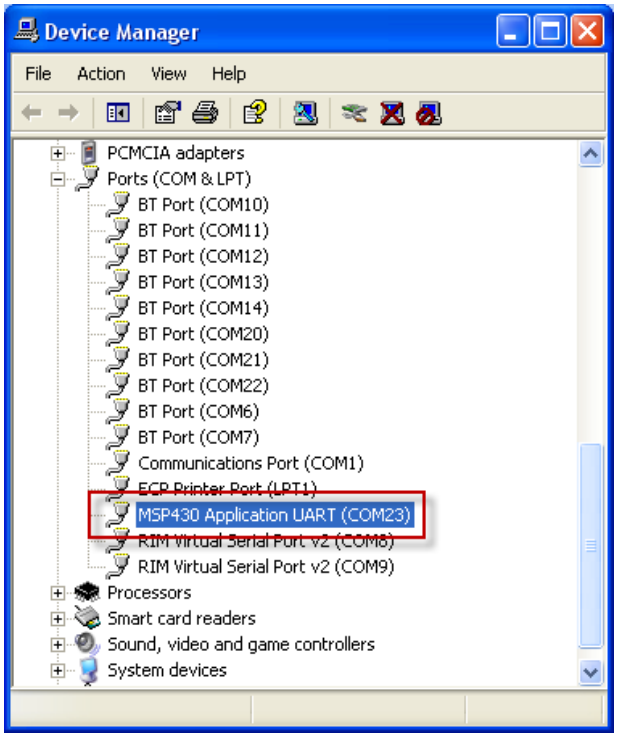


eZ430-RF2500 Board Driver Locations

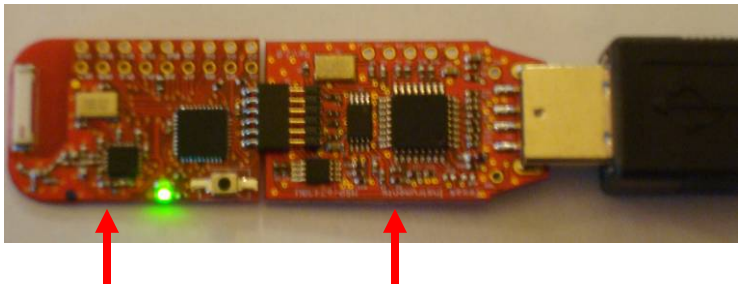
Two USB drivers are installed. The 1st is a Debug Chain Human Interface Device (HID):



The 2nd is a Back Channel UART Com Port:

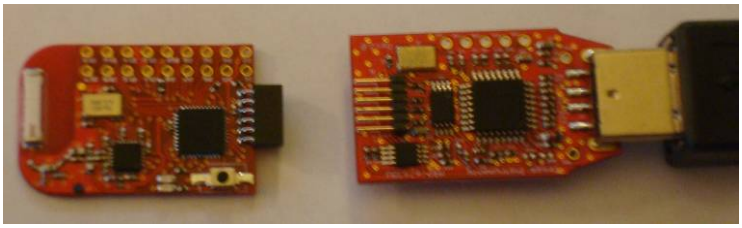


To Perform a HW Reset the eZ430-RF2500



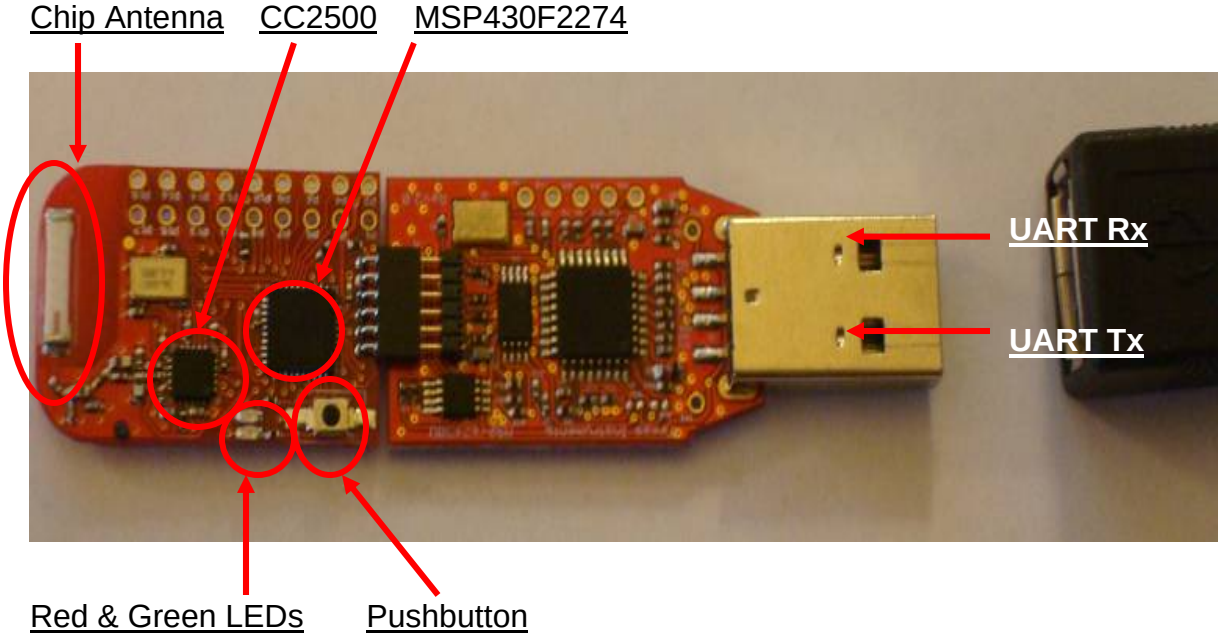
RF2500 Target Board eZ430 Debug & Download

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



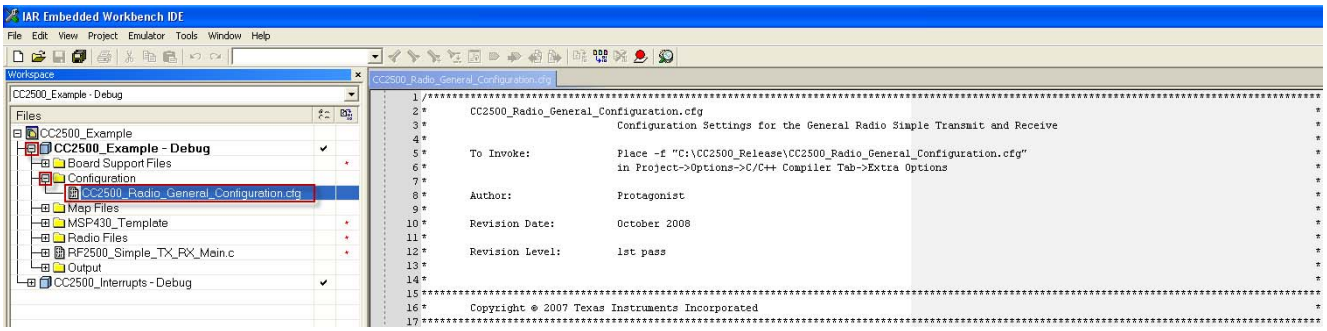
This will maintain the COM connection to your terminal application while still resetting the RF2500 application.

RF2500 Board features:

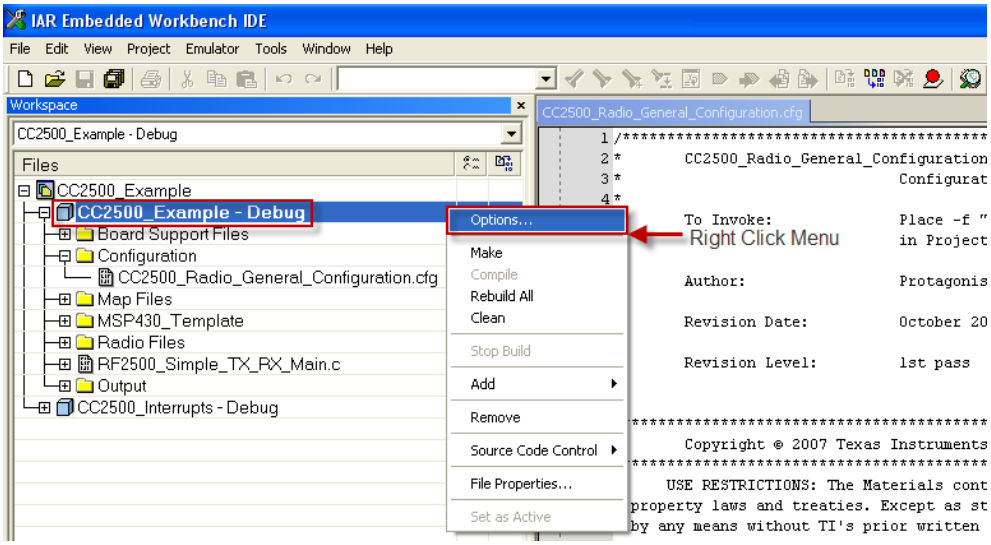


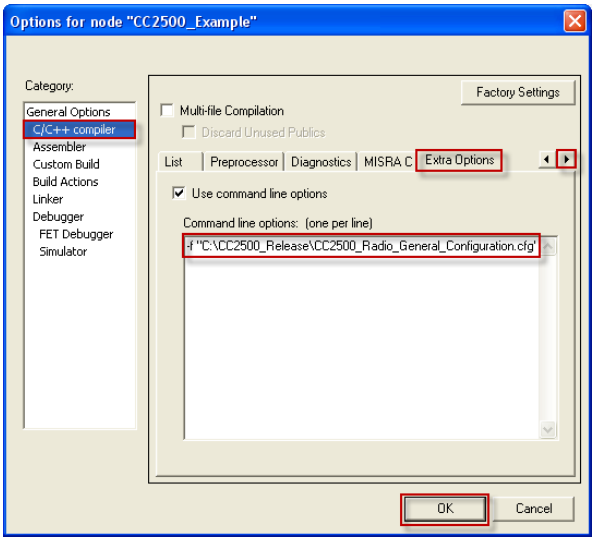
Setting the Configuration File

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_O 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.
```

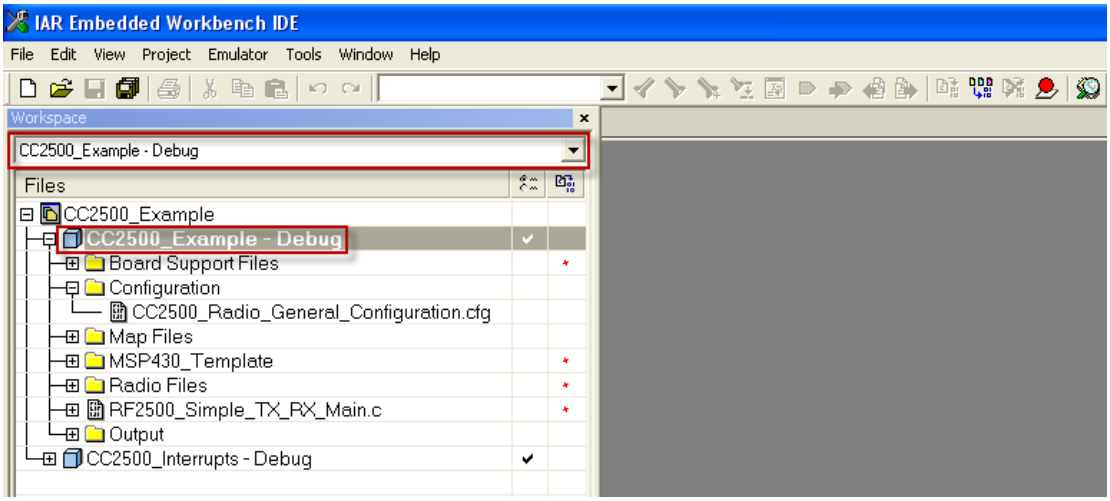
Options Continued:

```
/*****************************************************************************
  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.
```

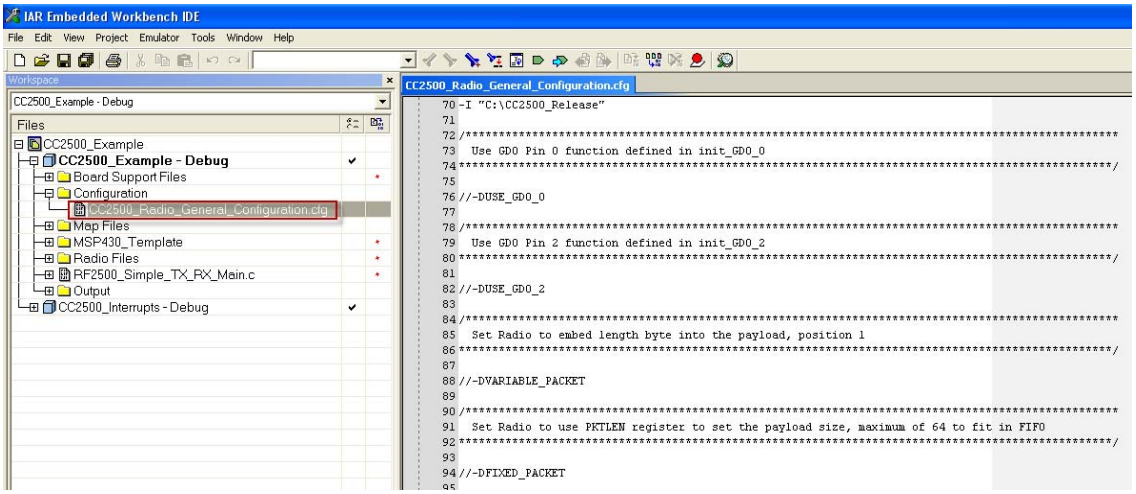
Lab 1: End Device to End Device

Introduction: This lab shows simple point to point communication.

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

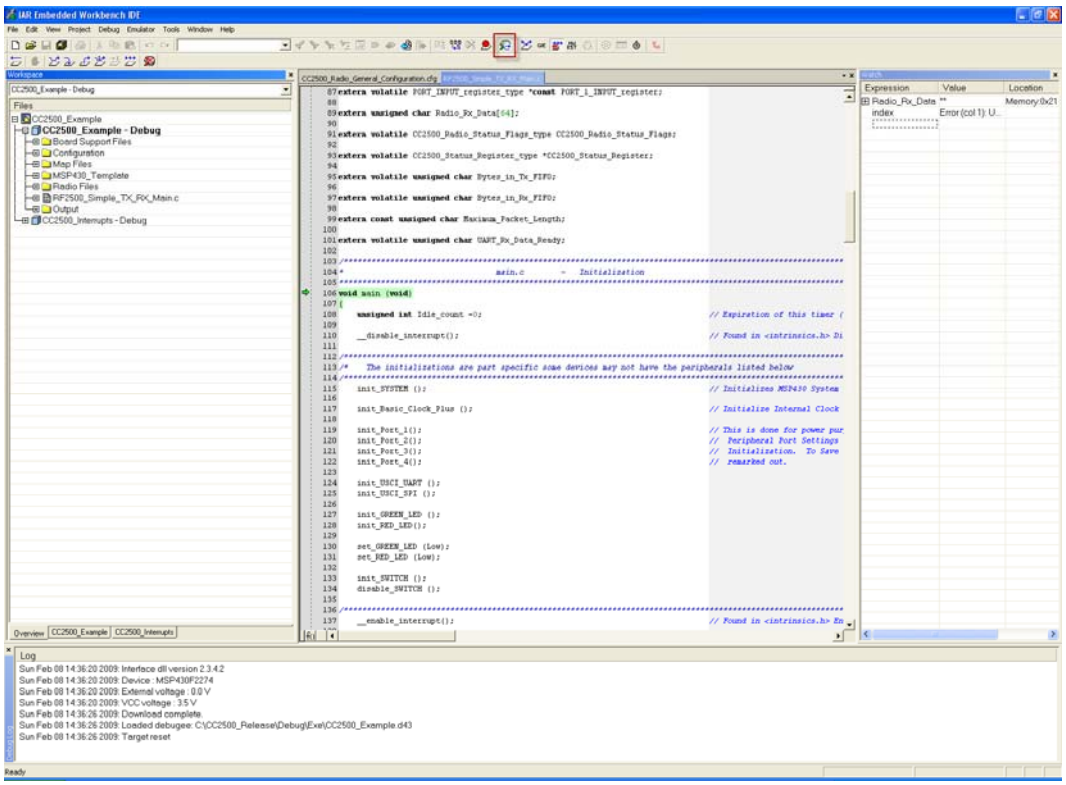


2. Under the CC2500_Example – Debug Project, Open CC2500_Radio_General_Configuration.cfg and set the options accordingly:

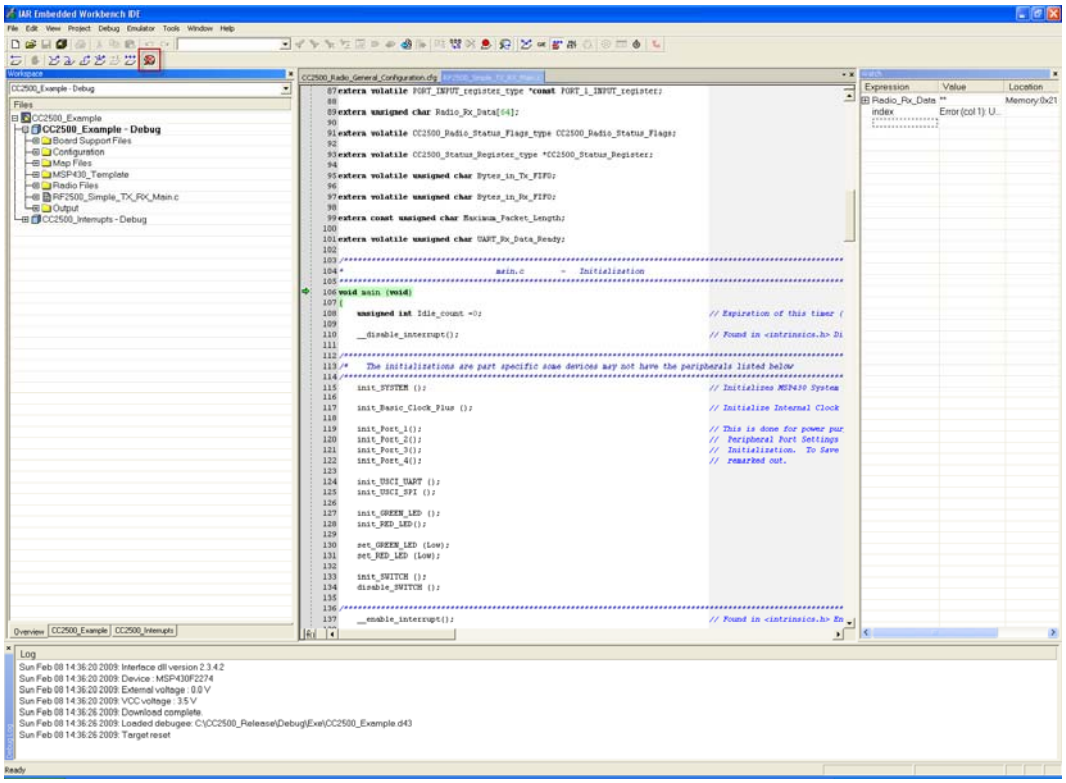


//-DUSE_GDO_0
//-DUSE_GDO_2
//-DVARIABLE_PACKET
-DFIXED_PACKET
//-DALLOW_ADDRESSING
//-DAPPEND_STATUS
//-DFORCE_TRANSMIT
//-DFORCE_RECEIVE
-DTERMINAL
//-DVERIFY
//-DDEBUG
//-DMAX_POWER

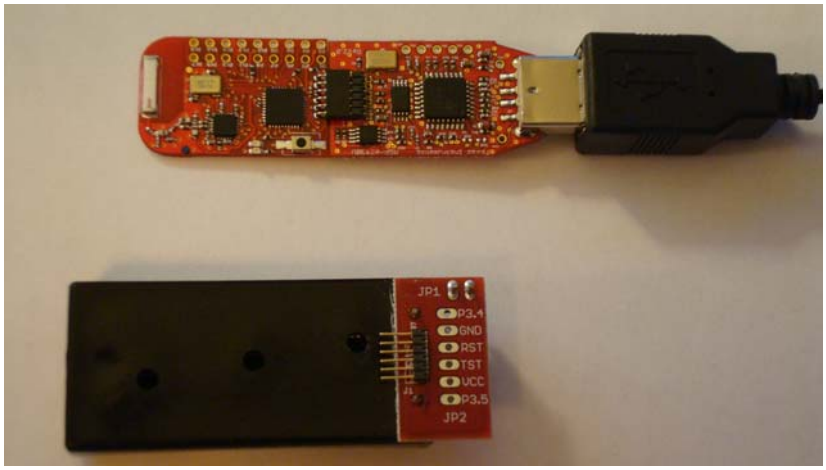
5. Build and Download the code:



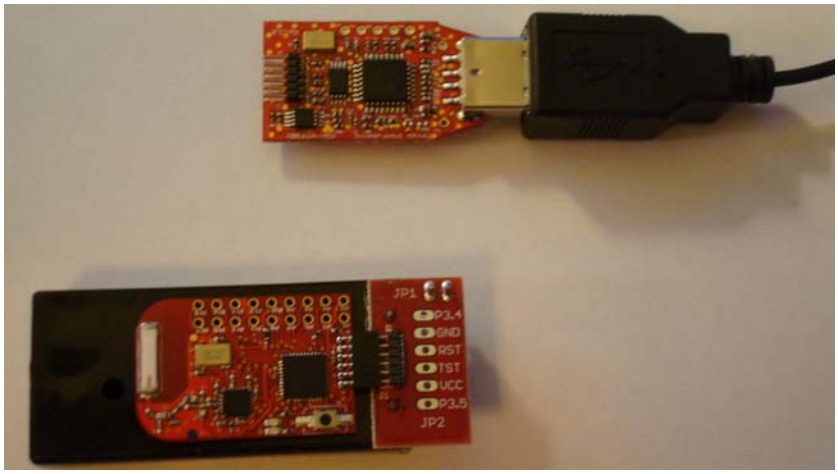
5. Stop Debugging:



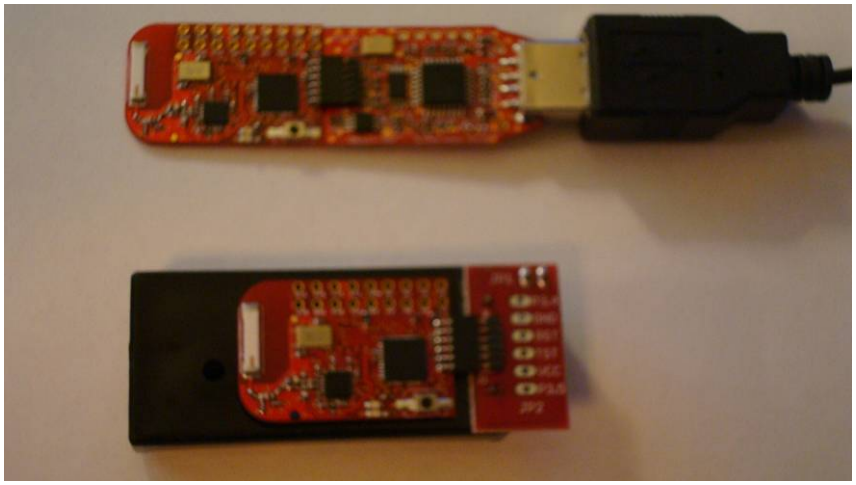
6. Remove the Programmed RF2500 Board from the eZ430:



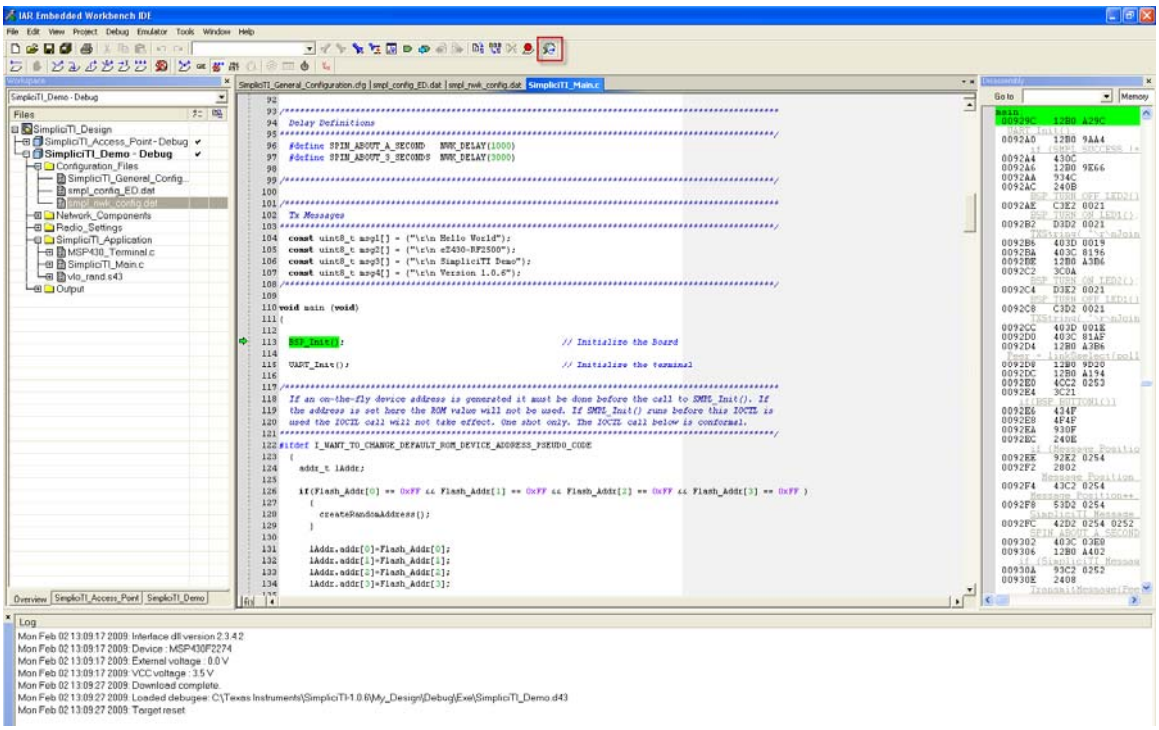
7. Put the Programmed RF2500 Board onto the Battery Board:



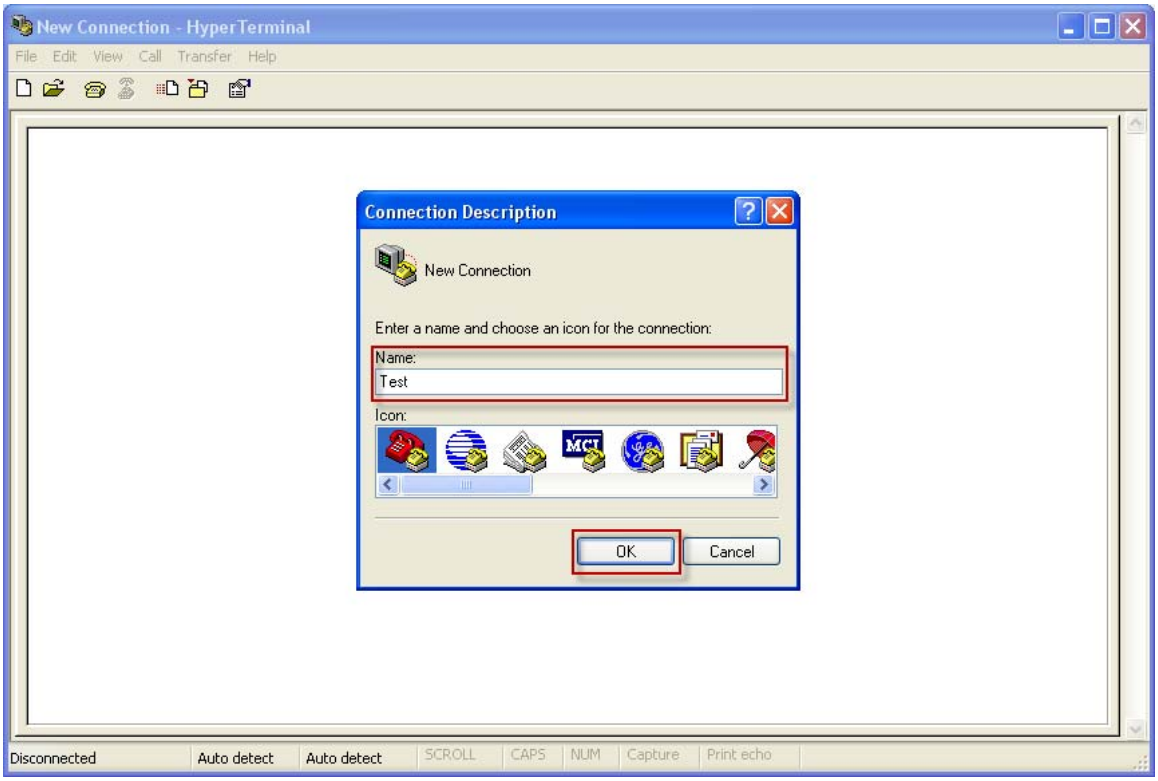
8. Put the un-programmed RF2500 Board onto the eZ430:



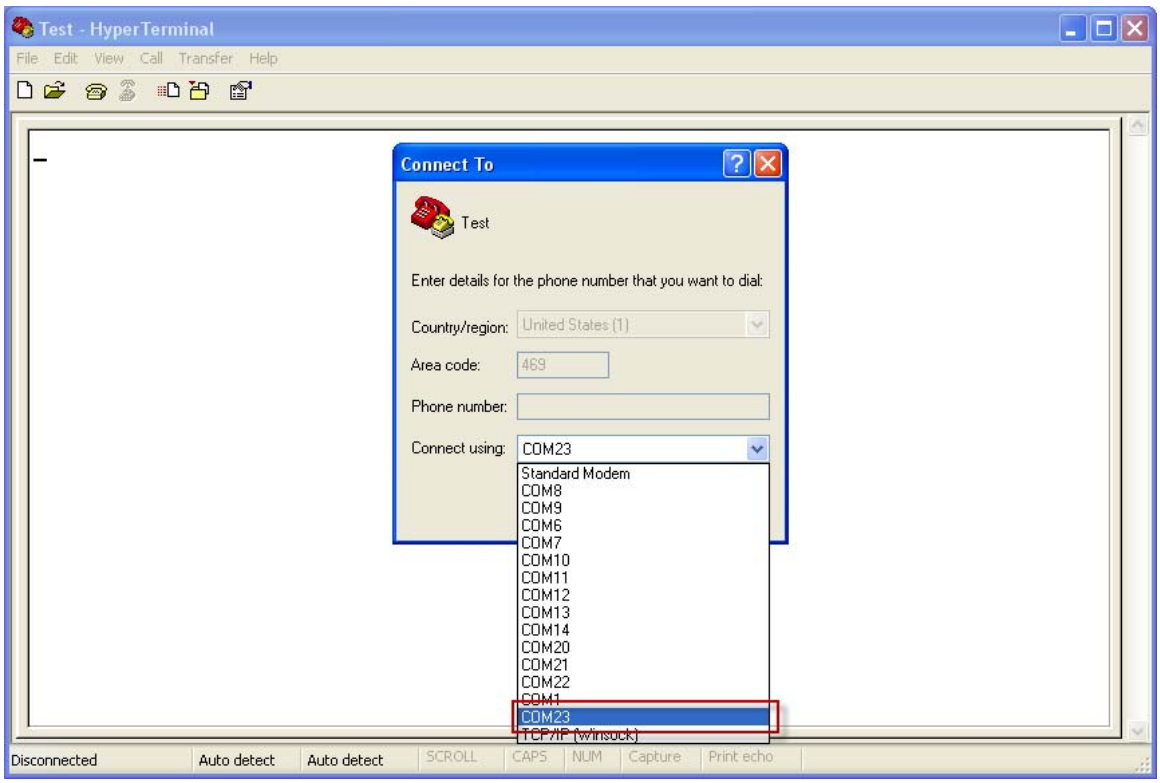
9. Build and Download the code into the un-programmed board:



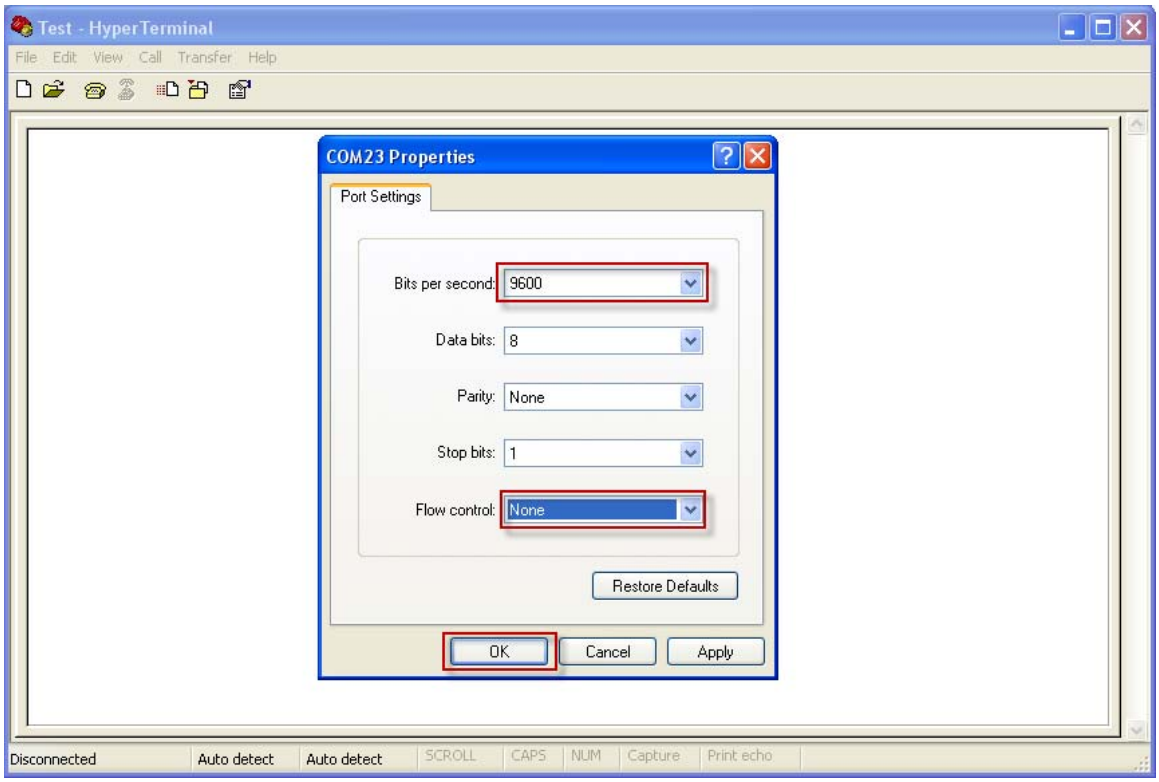
10. Start HyperTerminal:
Start->All Programs->Accessories->Communication->HyperTerminal



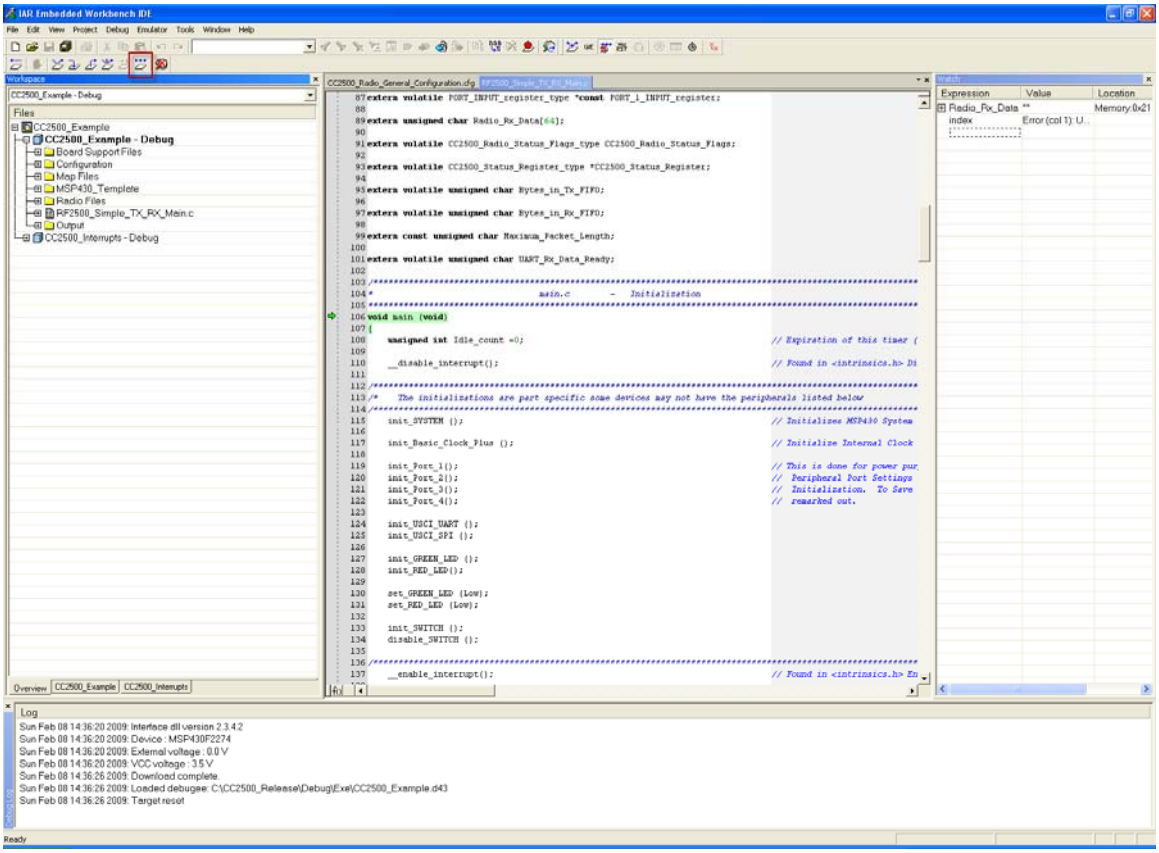
11. Set the Com Port to the MSP430 Application UART, and press OK



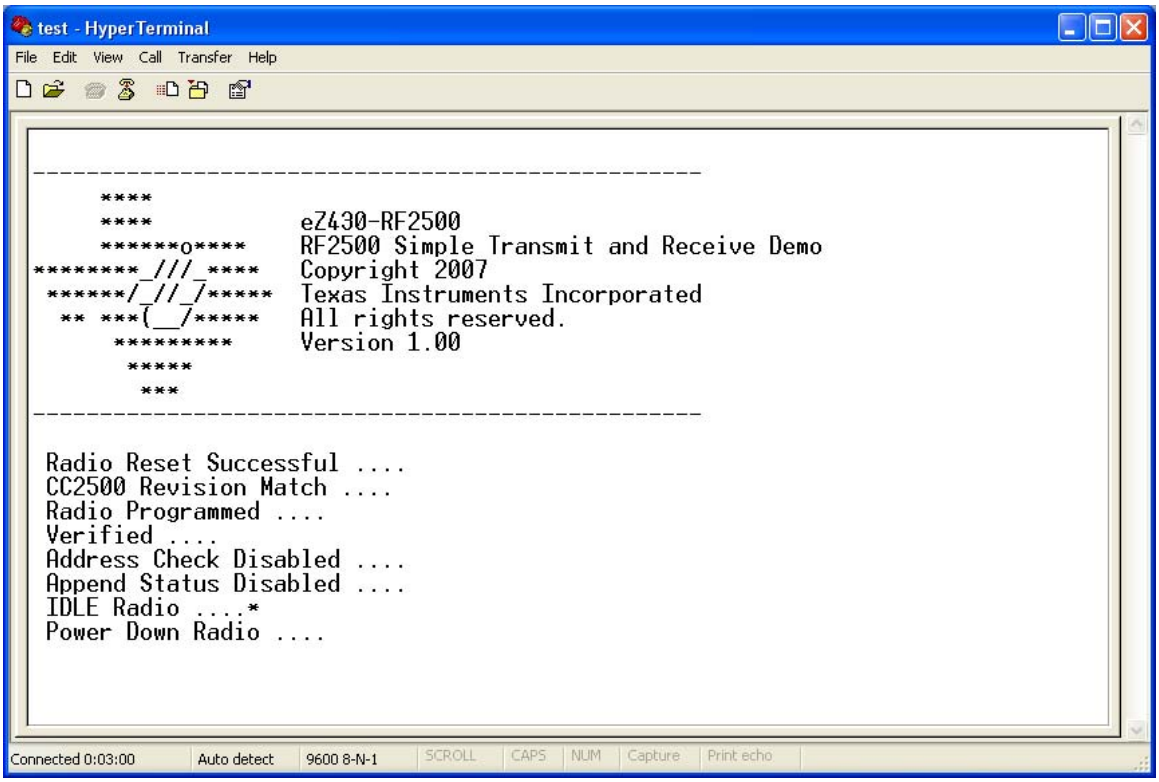
12. Set the Com Port to 9600 Baud, 8N1:



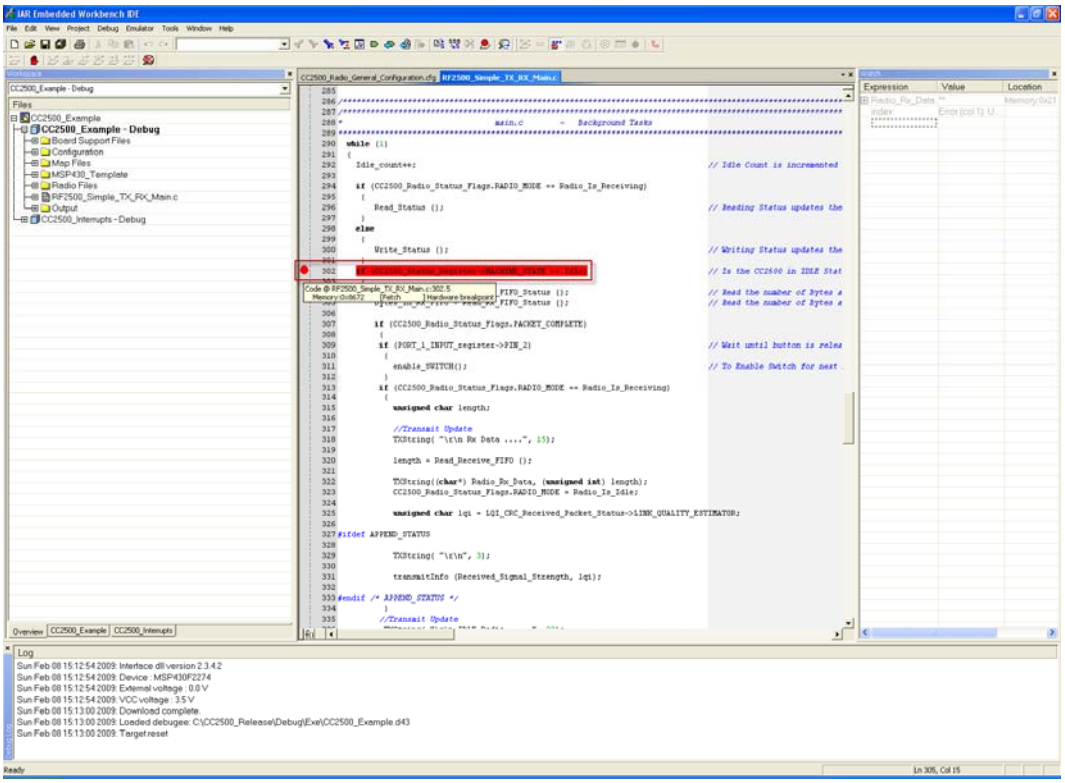
13. Run the Code:



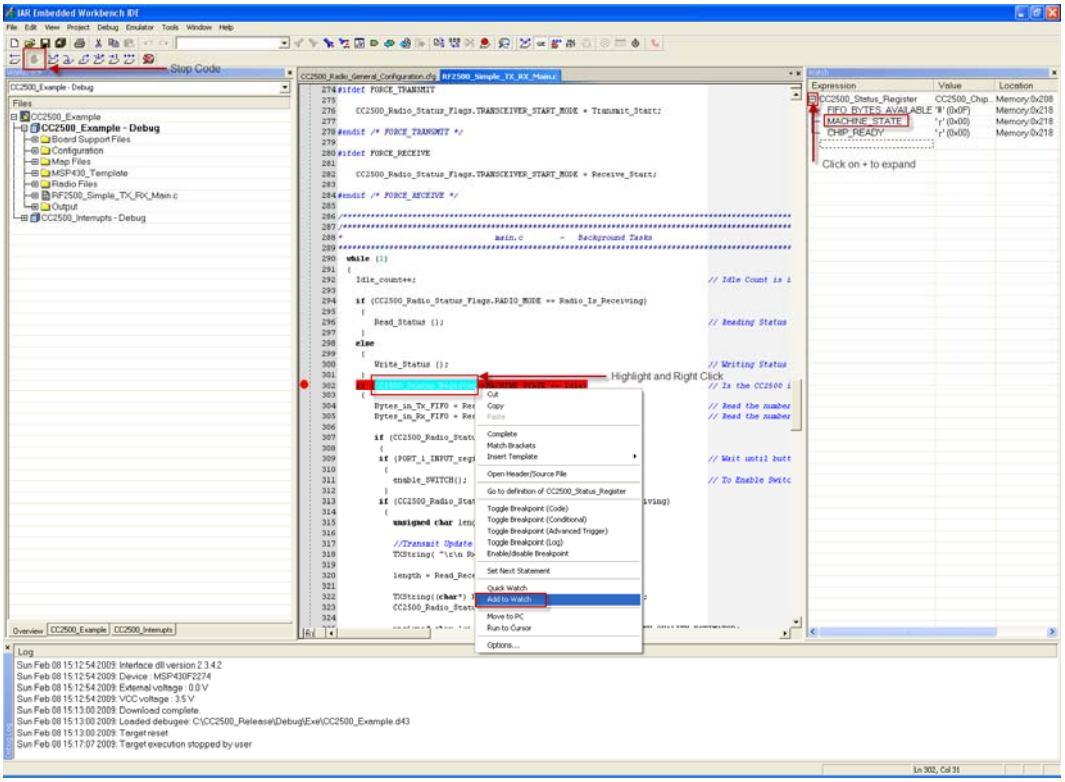
The HyperTerminal Screen should show:



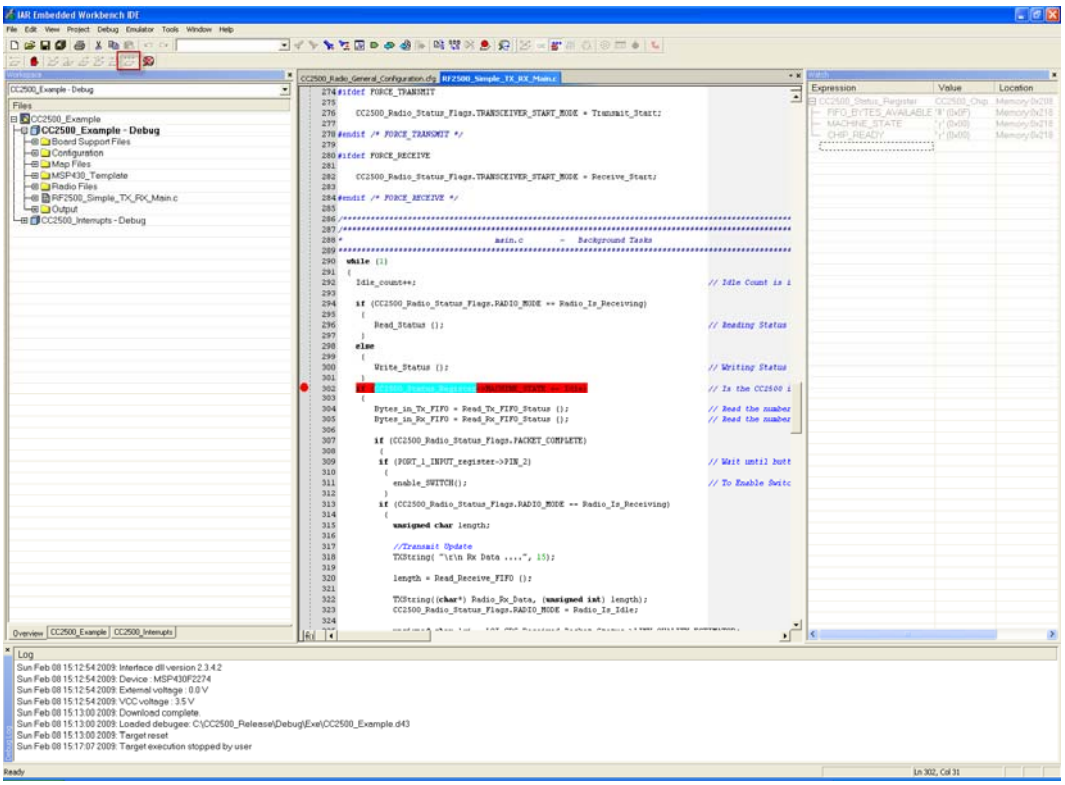
14. Set a breakpoint at line 302:



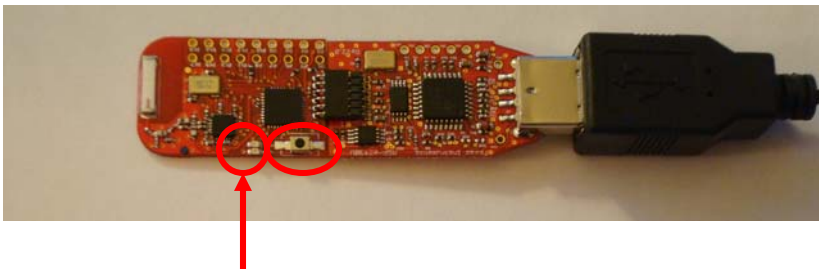
15. Highlight and watch the variable: CC2500_Status_Register



16. Continue the code:

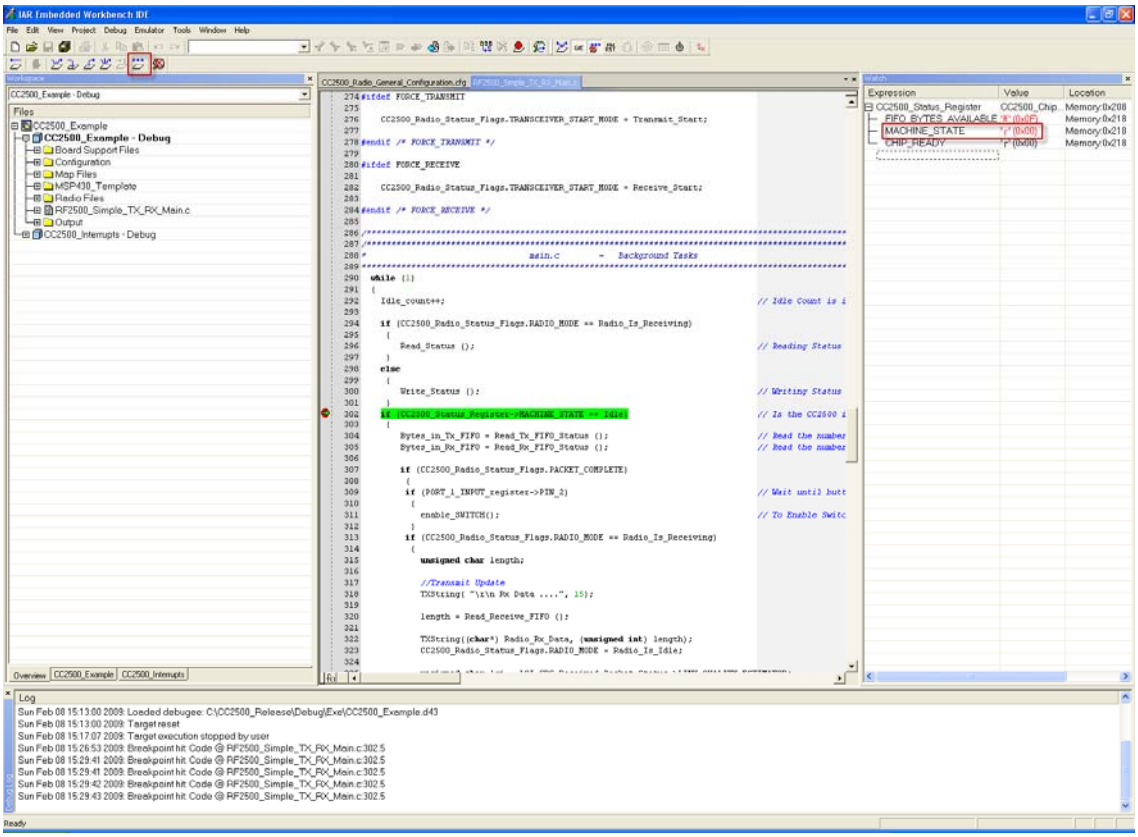


17. Press and hold button to transmit:

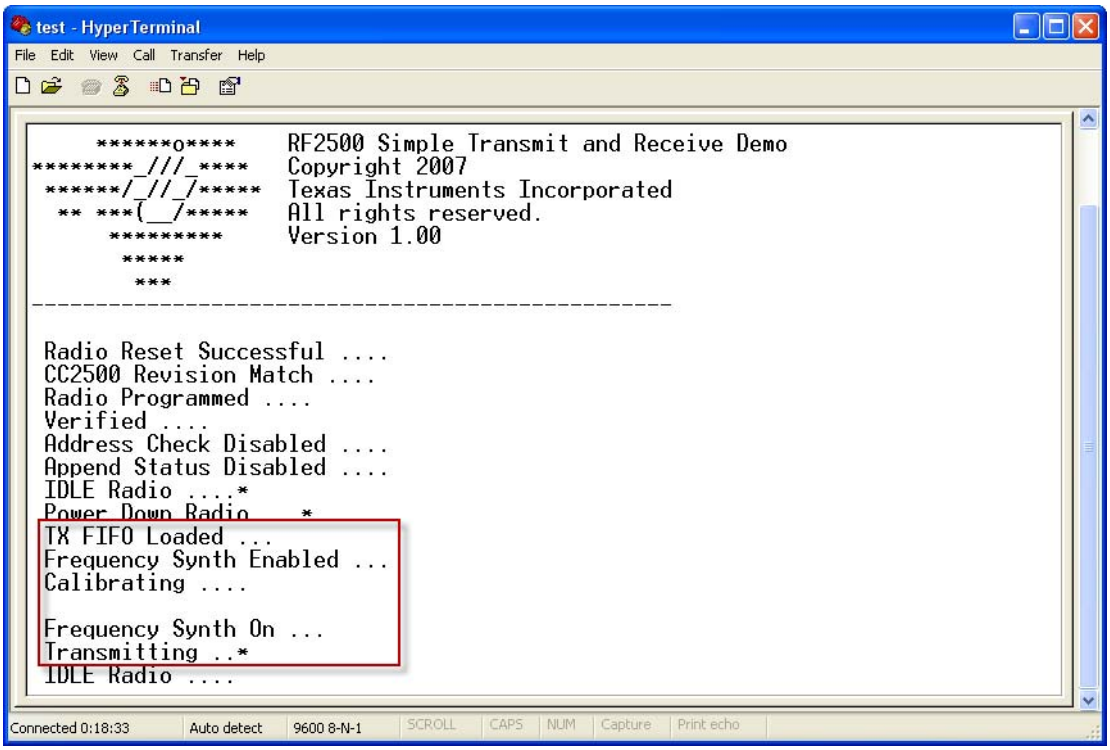


Green LED will light (showing the radio is in transmit mode).

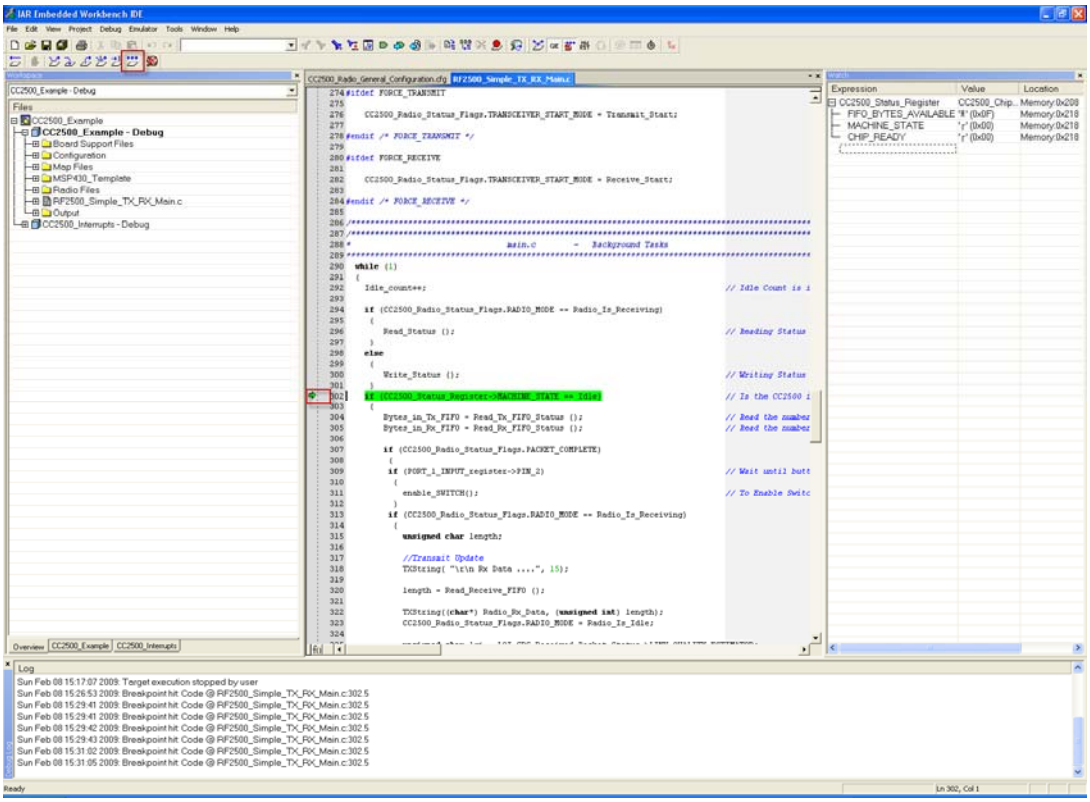
18. Code stops…. Execute the code 4 times, watching the MACHINE_STATE it will cycle from 4 ->3 ->2->0



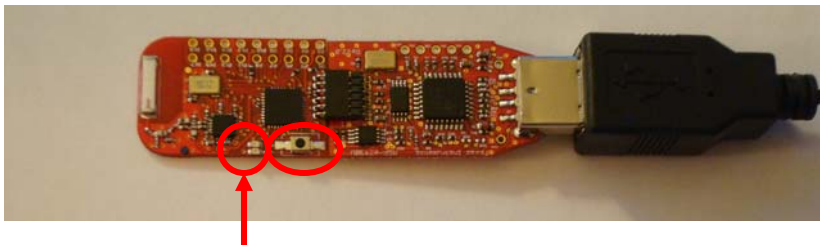
Hyperterminal will show the stages the radio goes through to transmit.



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

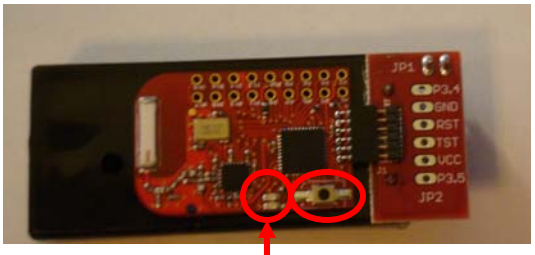


20. Press and release the switch on the eZ430-RF2500 board:



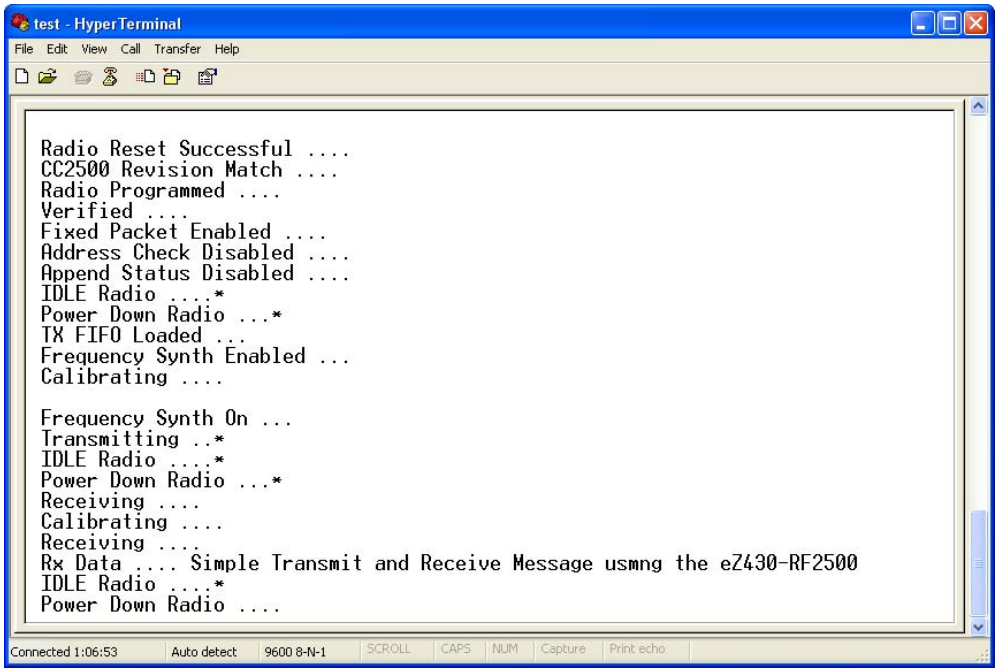
The red LED will light, the radio is in Receive Mode.

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



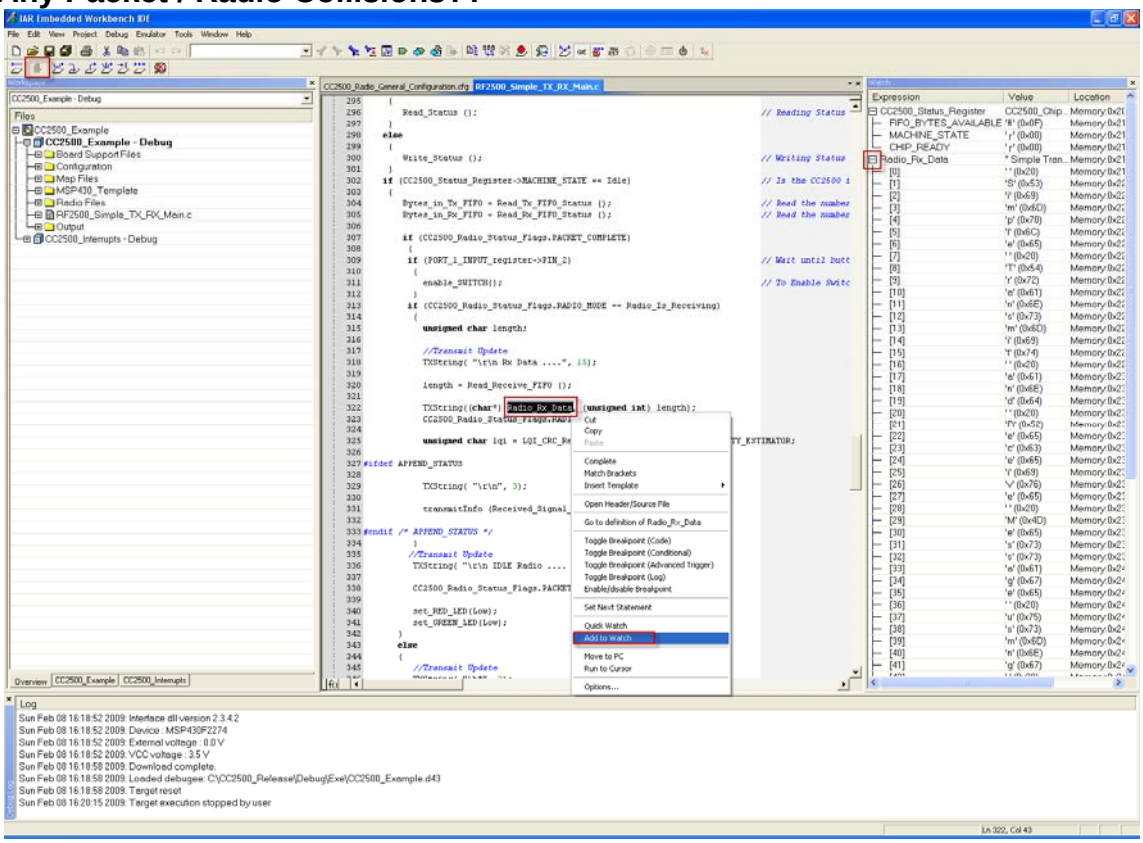
The green LED will flash as a message is transmitted.

The message will appear on the hyperterminal session.



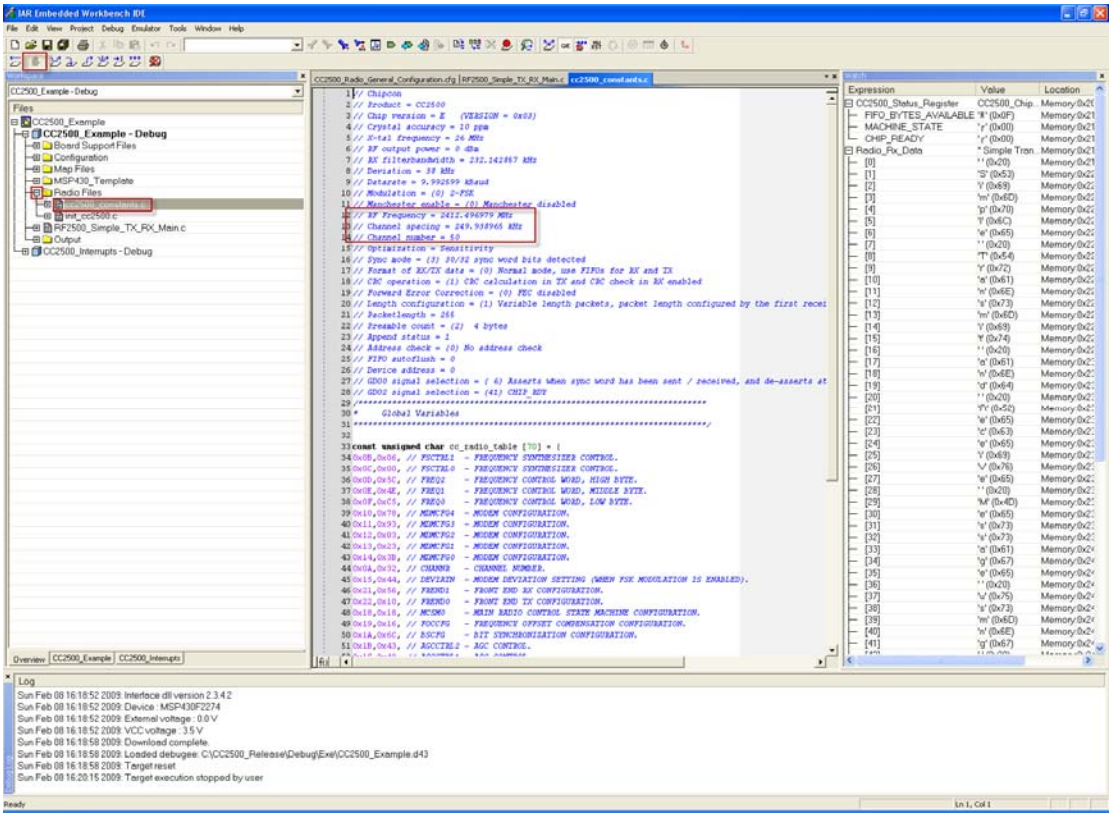
22. Stop the debugging session, highlight Radio_Rx_Data on line 322, right click on it and add the watch. Click on the + next to the Radio_Rx_Data in the watch window:

Any Packet / Radio Collisions??



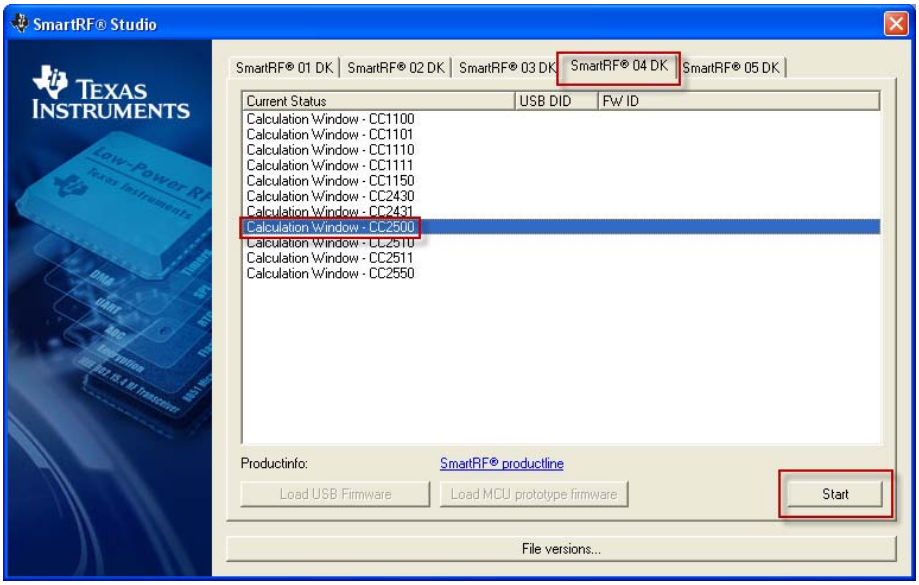
Lab 2: Editing the Radio Settings

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

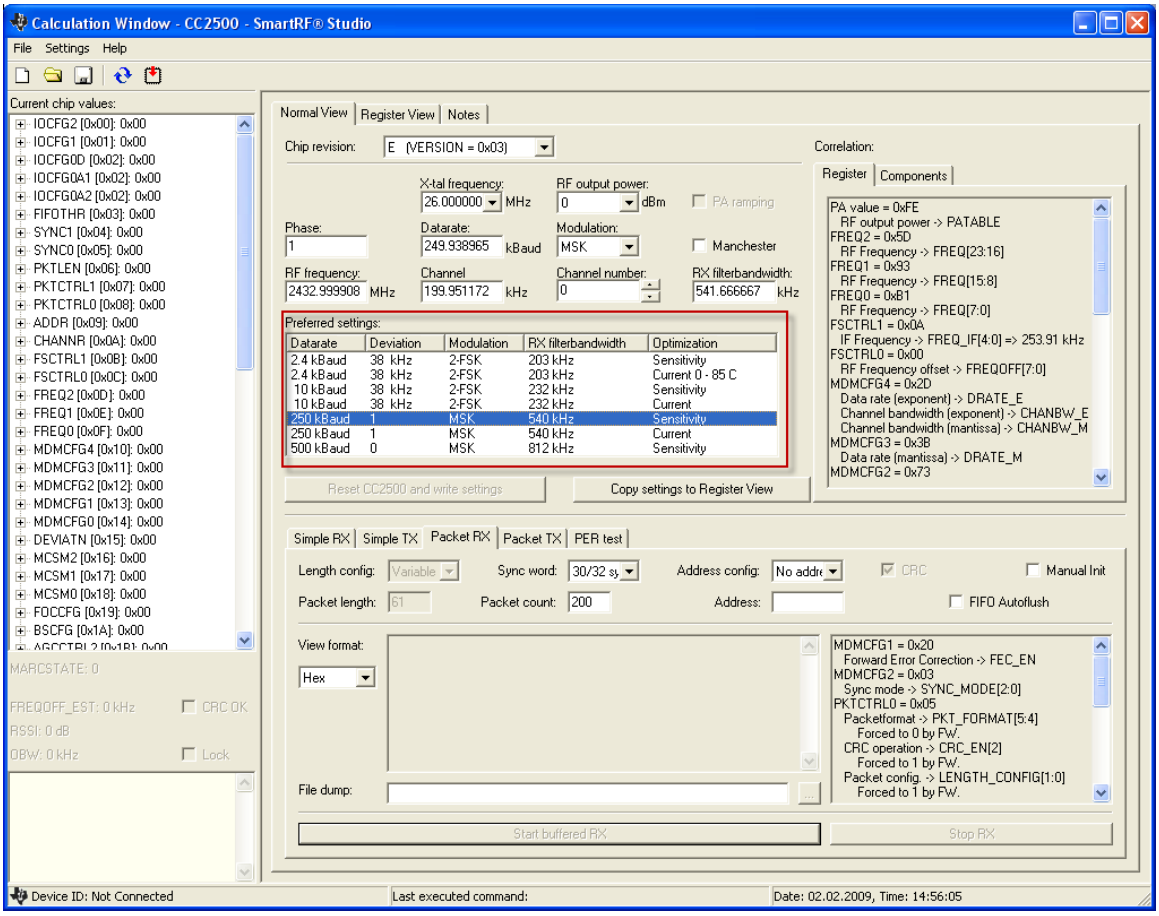


2. Open SmartRF Studio:

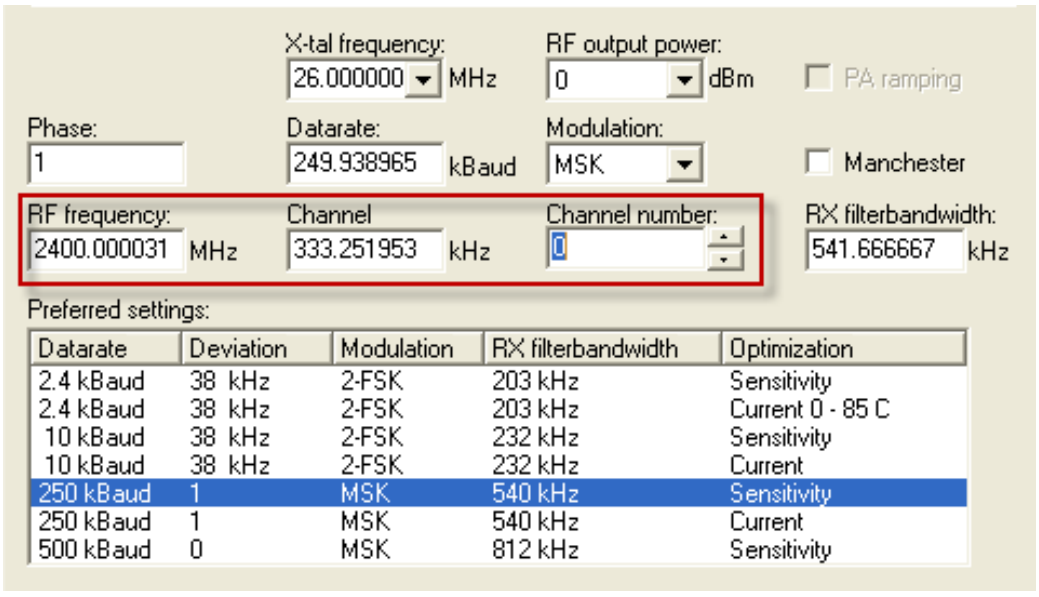
Start->All Programs->Texas Instruments->Smart RF Studio->Smart RF Studio



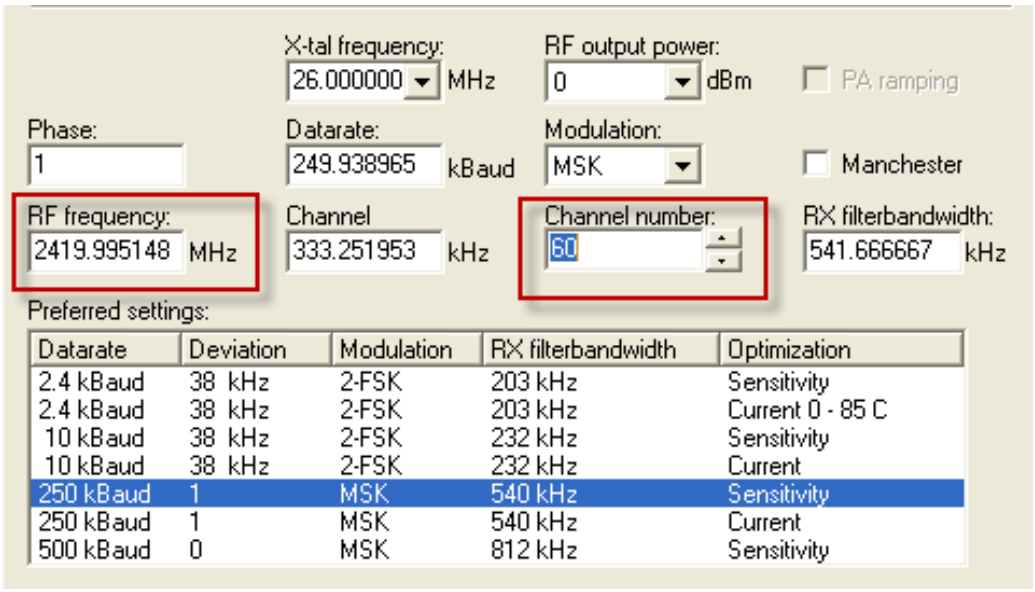
3. Select a Preferred Setting to start with:



4. Set your base frequency and Channel Bandwidth:

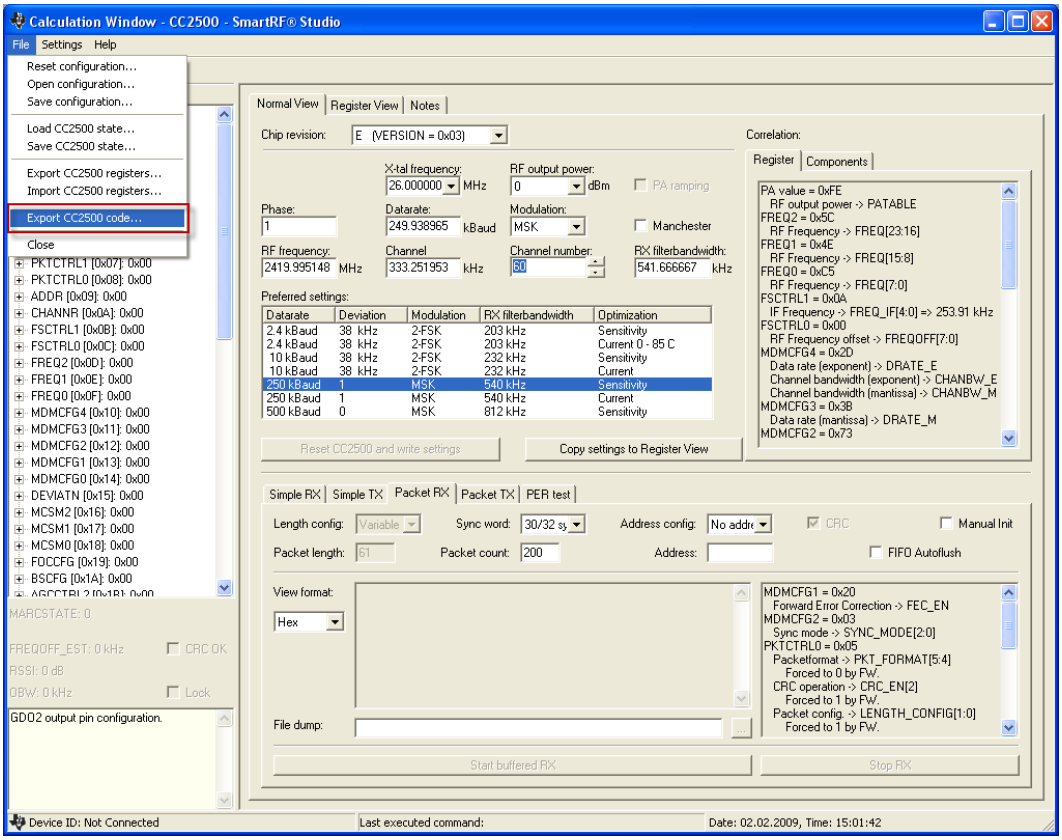


5. And set your target channel:

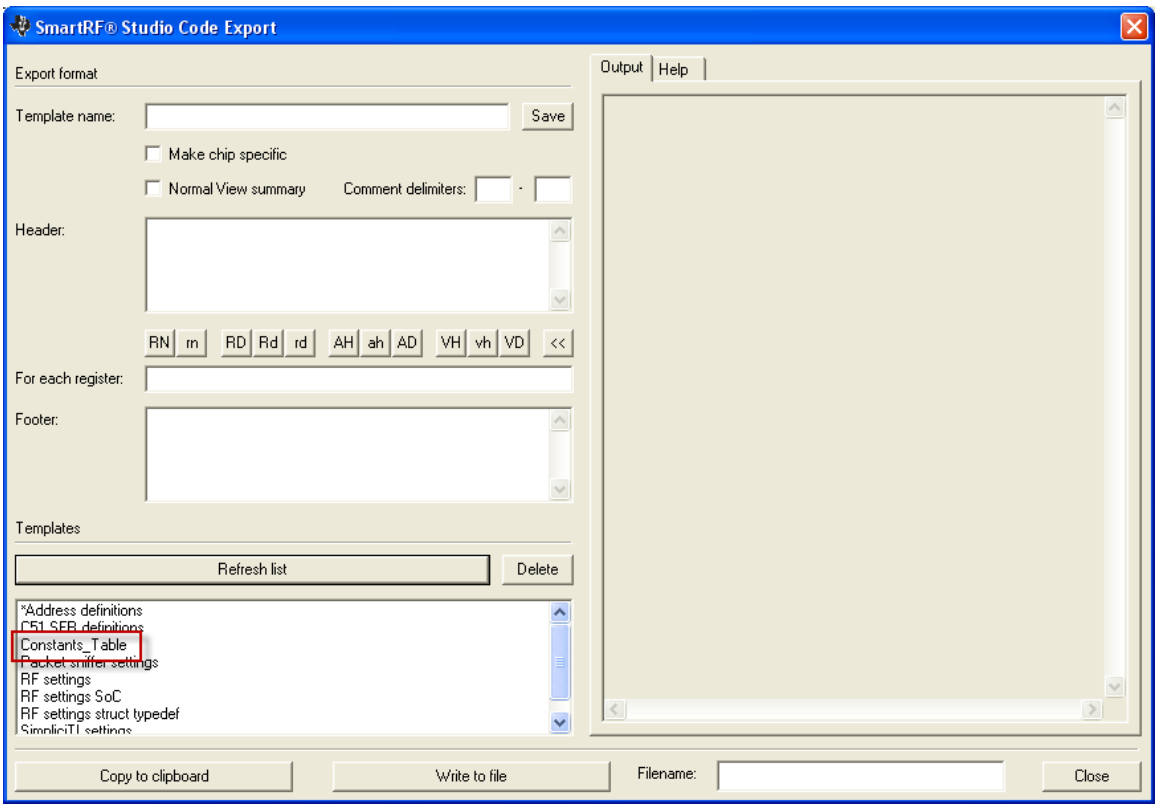


Datarate	Deviation	Modulation	RX filterbandwidth	Optimization
2.4 kBaud	38 kHz	2-FSK	203 kHz	Sensitivity
2.4 kBaud	38 kHz	2-FSK	203 kHz	Current 0 - 85 C
10 kBaud	38 kHz	2-FSK	232 kHz	Sensitivity
10 kBaud	38 kHz	2-FSK	232 kHz	Current
250 kBaud	1	MSK	540 kHz	Sensitivity
250 kBaud	1	MSK	540 kHz	Current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

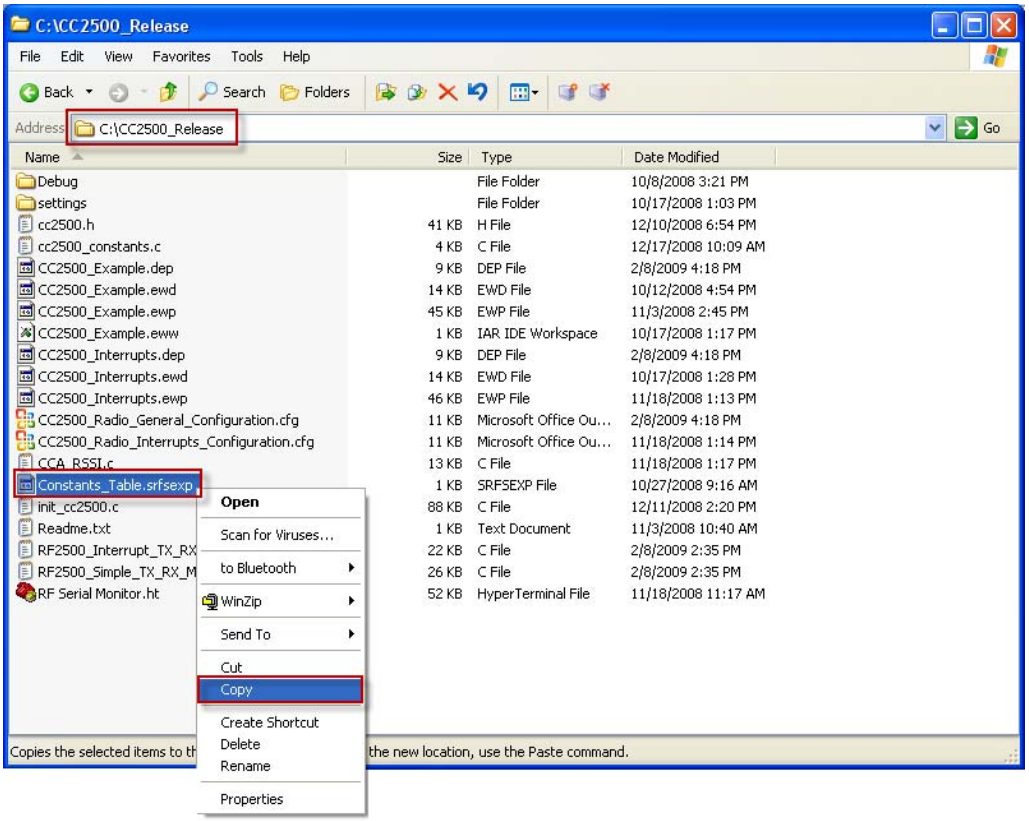
6. Export your Code:



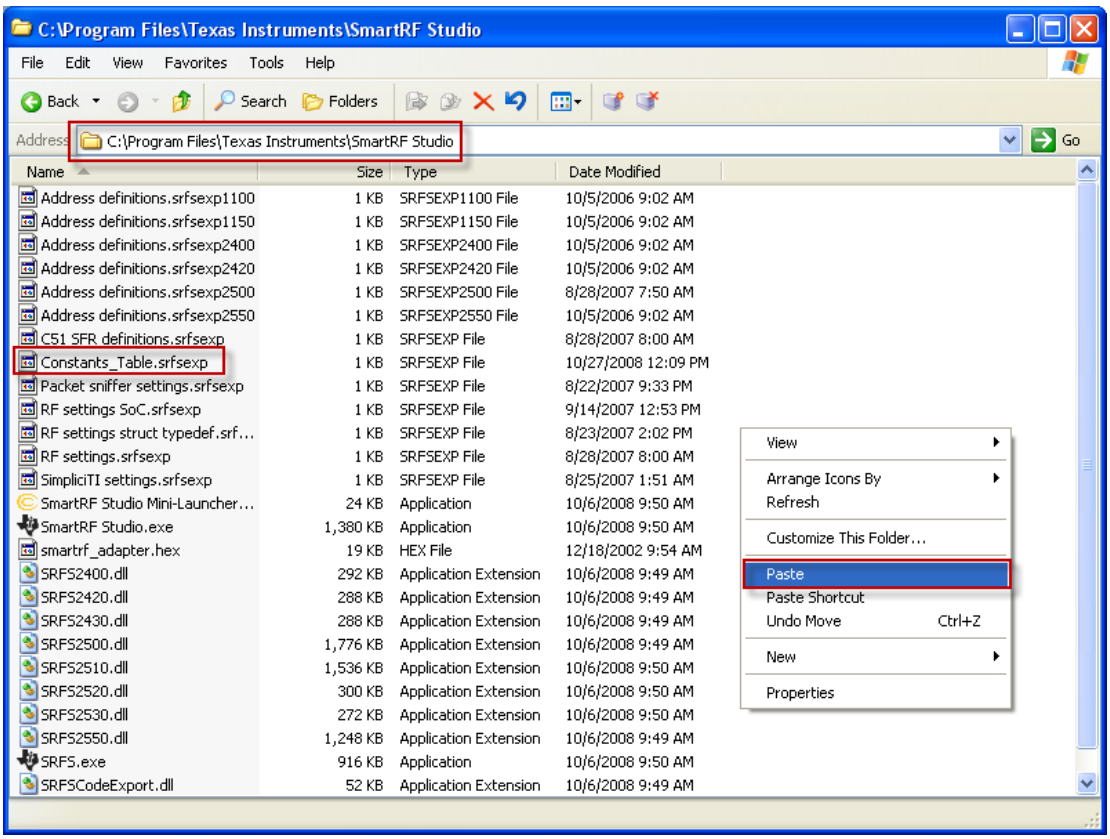
7. Importing a Settings Template



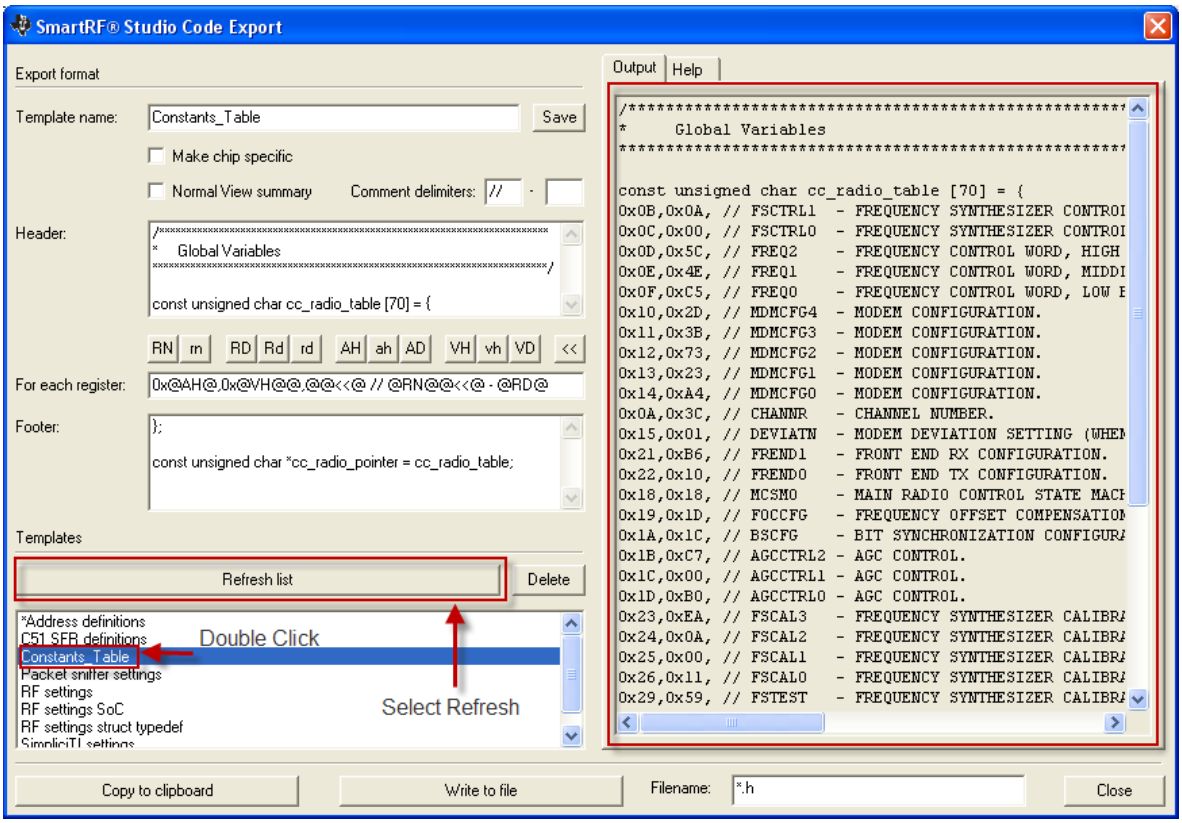
8. Copy Constants_Table.srfsexp in the CC2500_Release Directory:



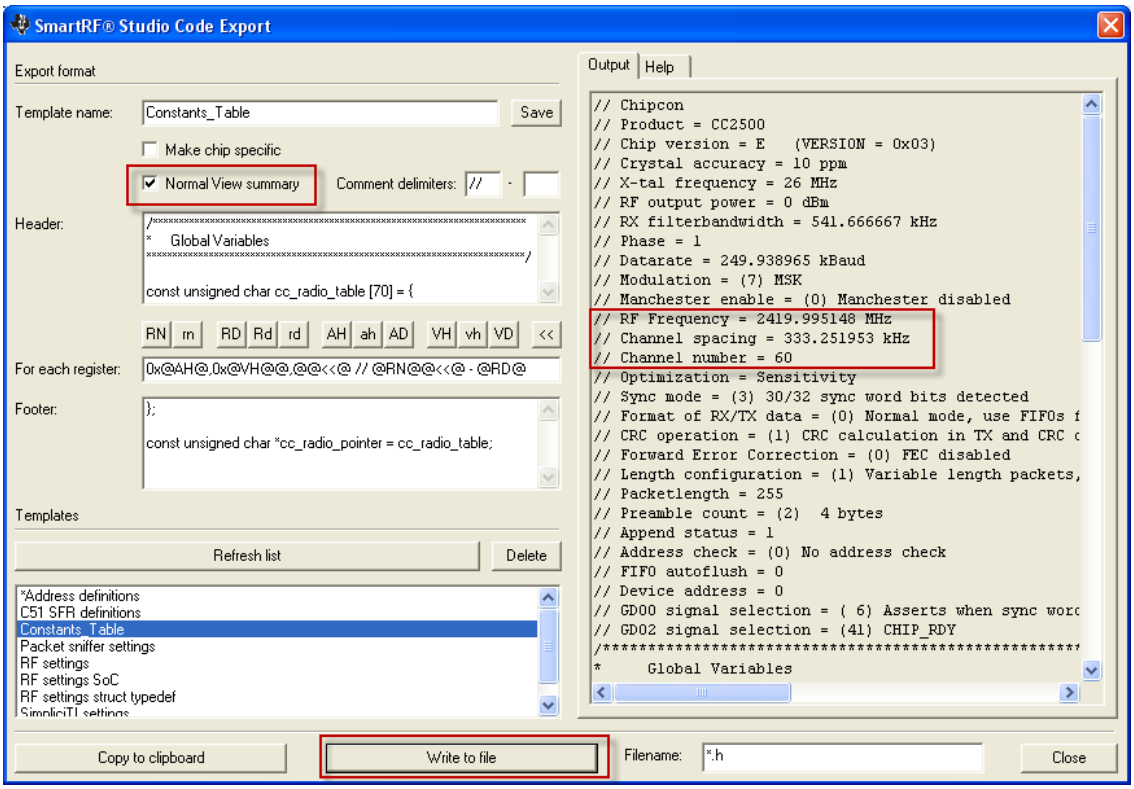
9. And paste it in the SmartRF Studio Directory:



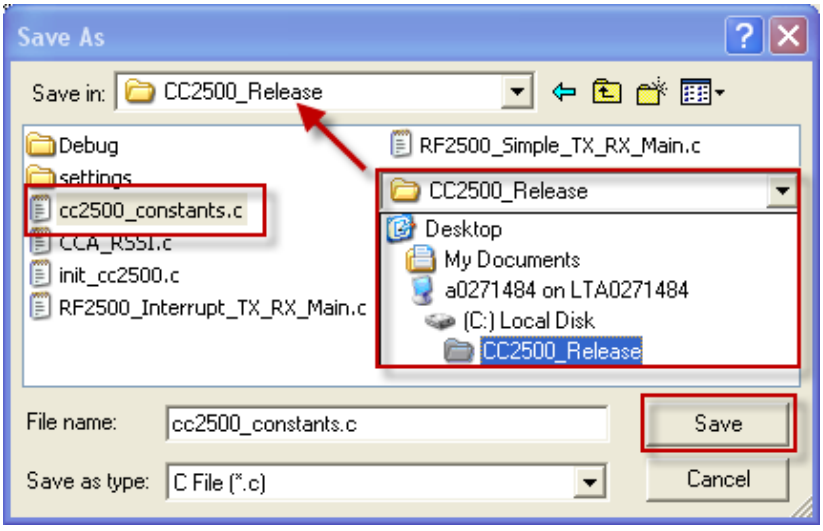
10. Refresh the list for the Constants_Table to appear. Double click on it to use the template to generate the code:

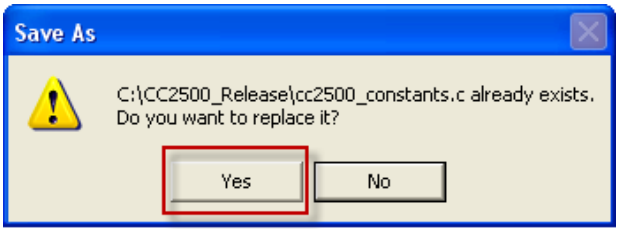


11. Select Normal View summary to show the radio settings. Then select Write to file to update the code:

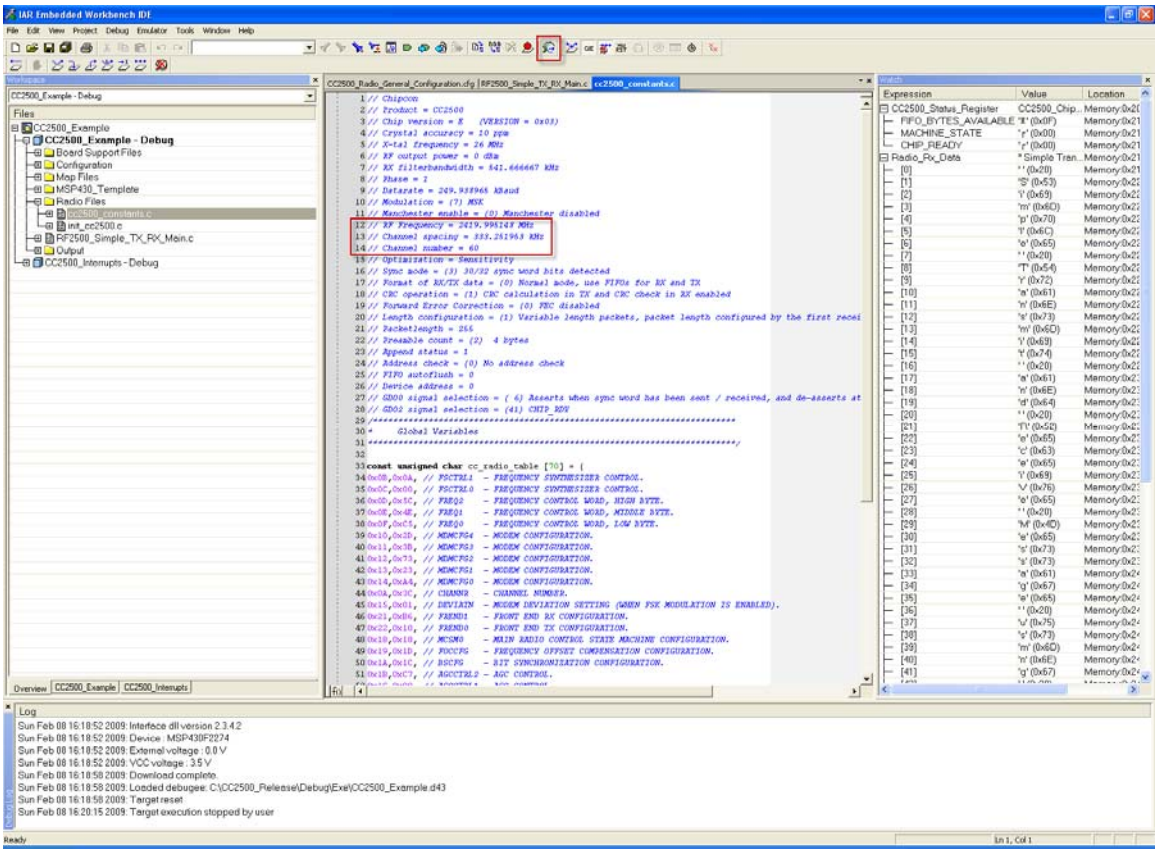


12. Overwrite cc2500_constants.c:

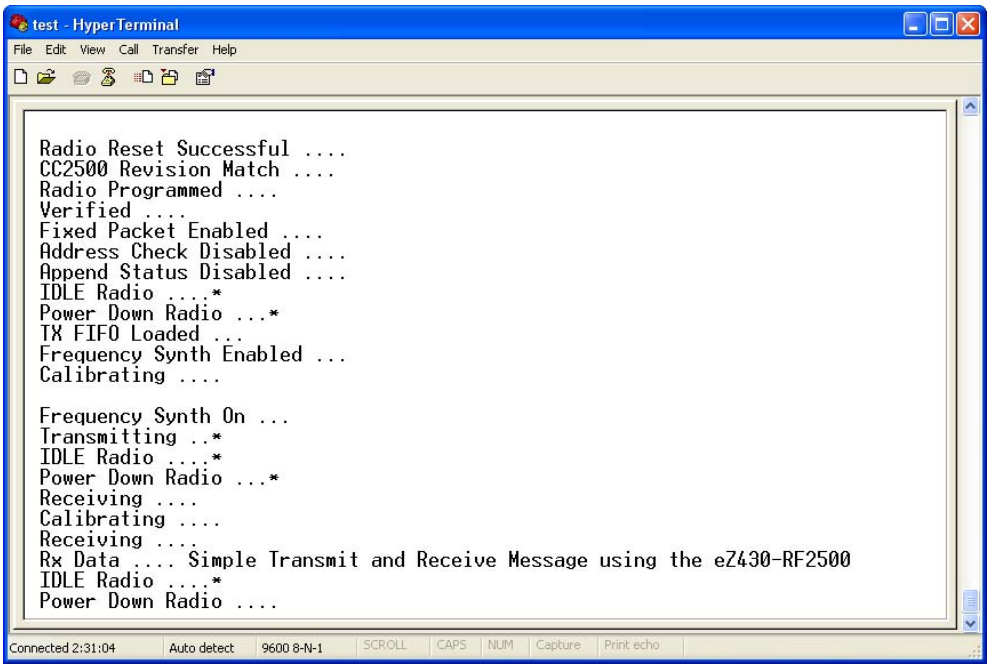




Was the file overwritten?

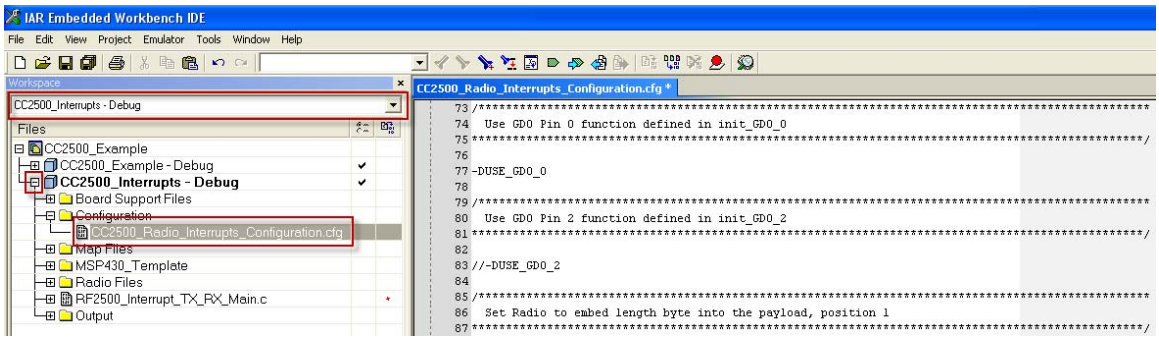


13. Rebuild both devices. The functionality of Lab 1 should occur but with less interference from your neighbor.



Lab 3: Eliminating the State Machine

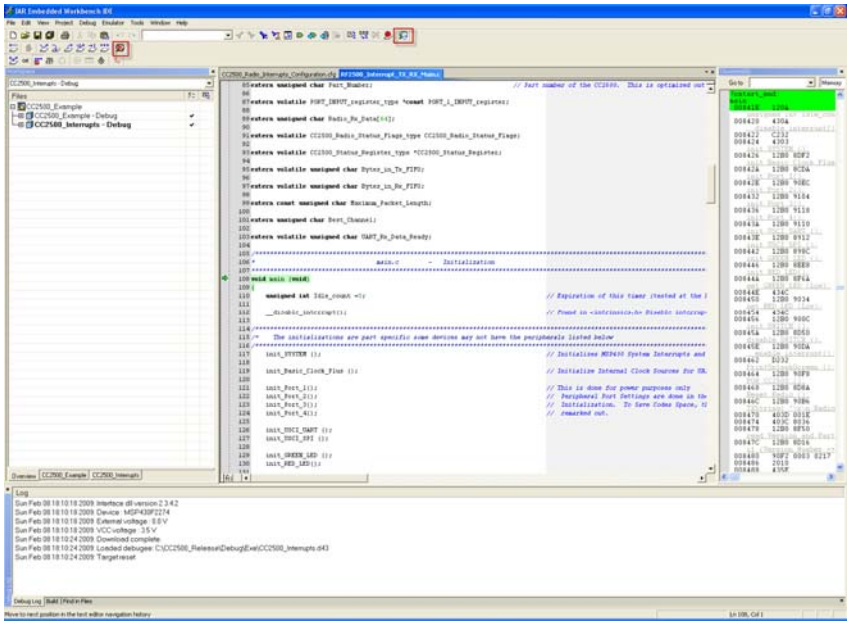
1. Select the CC2500_Interrupts – Debug Project and open the CC2500_Radio_Interrupts_Configurations.cfg file:



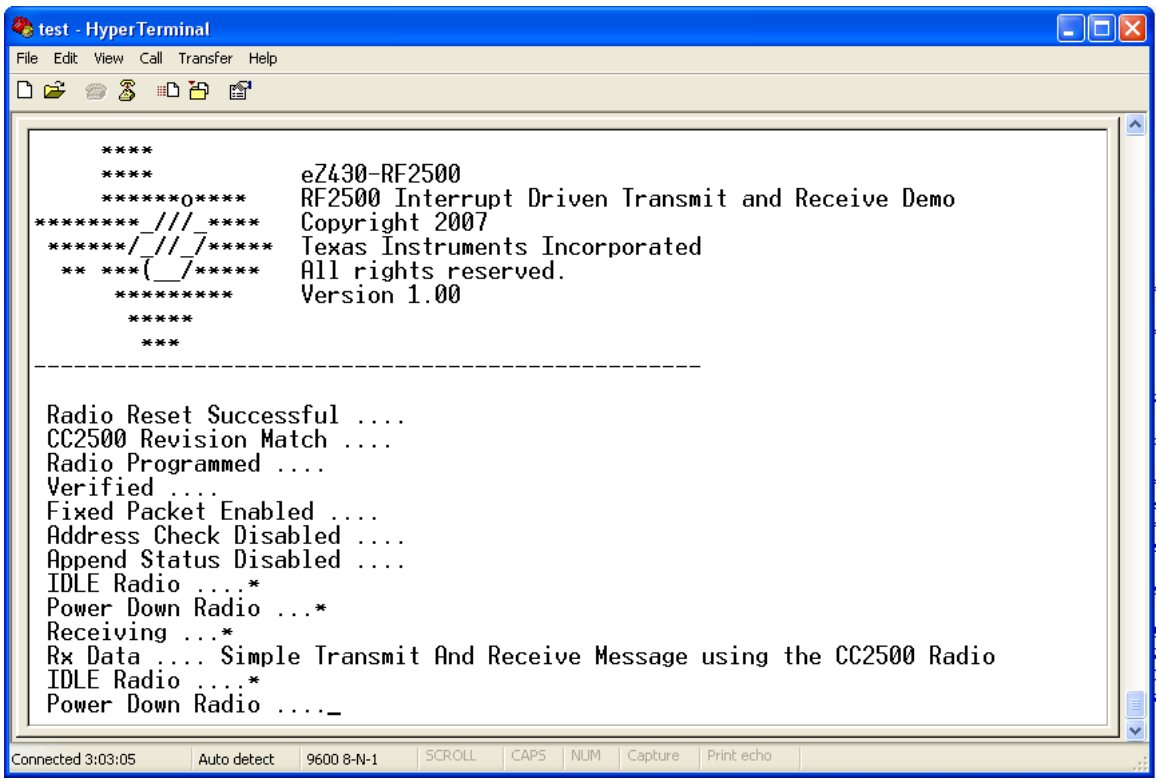
2. Set in CC2500_Radio_Interrupts_Configurations.cfg:

```
-DUSE_GDO_0
//-DUSE_GDO_2
//-DVARIABLE_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:

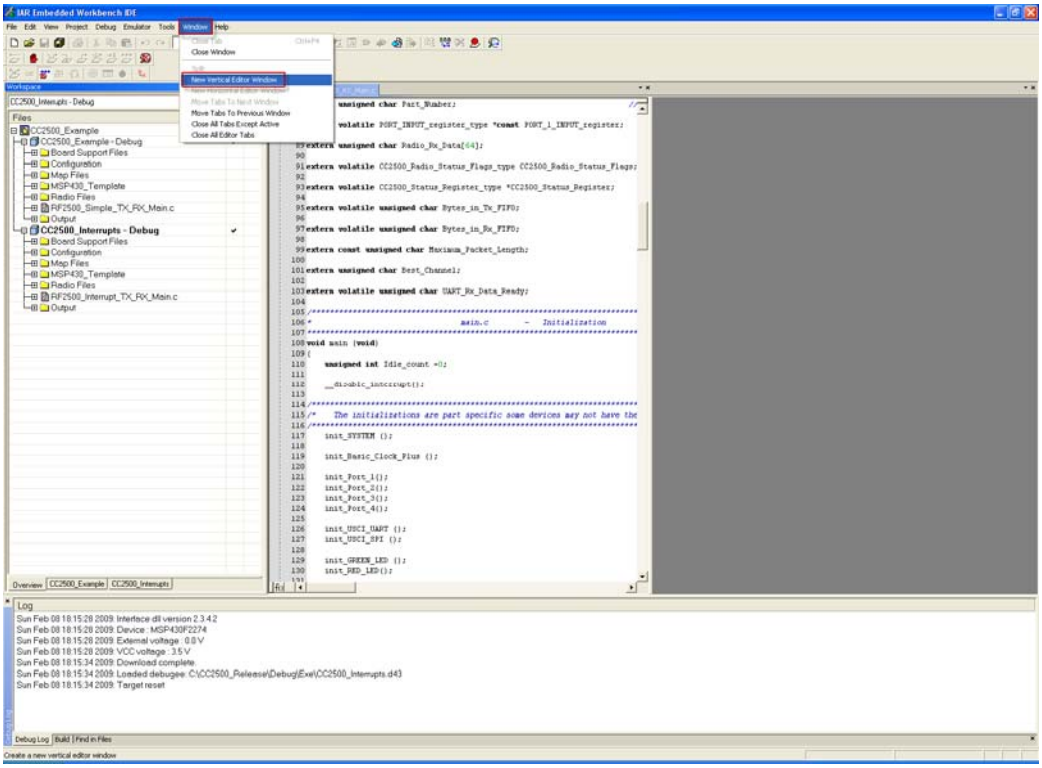


Output to the hyperterminal:

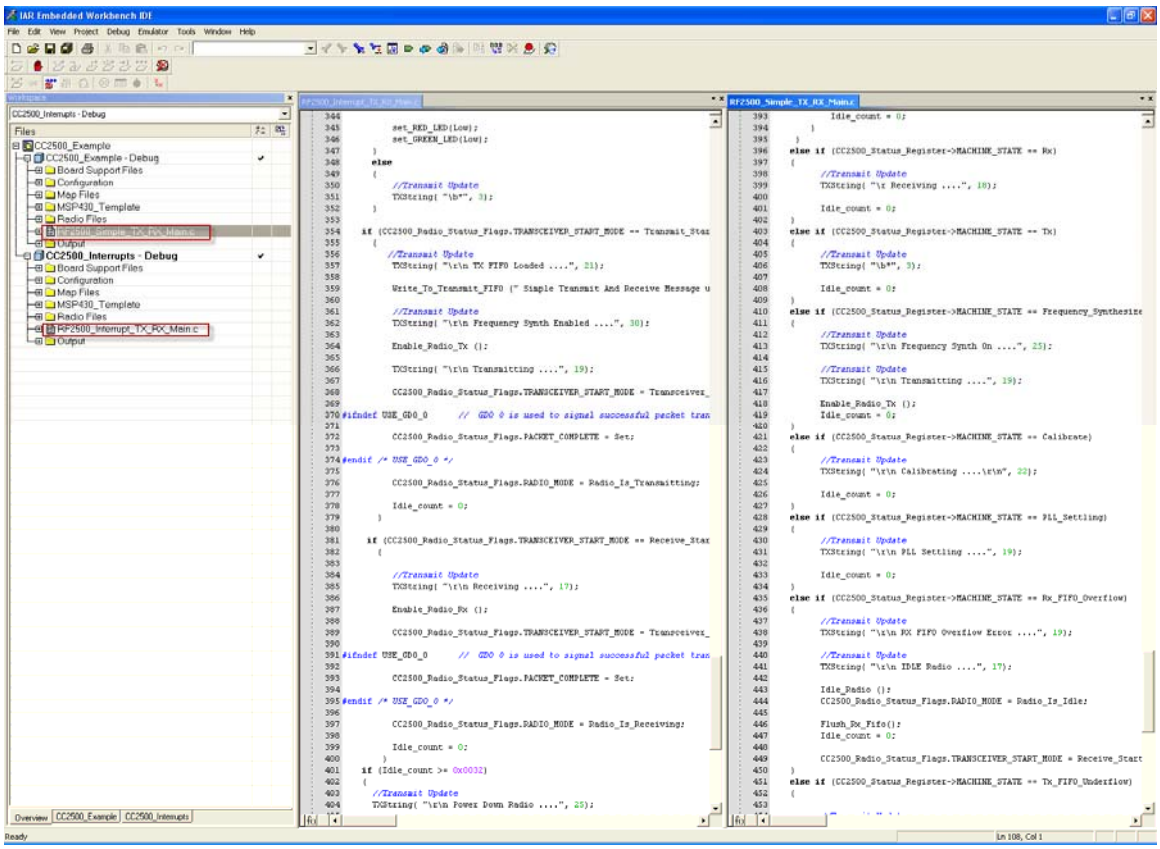


What changed?

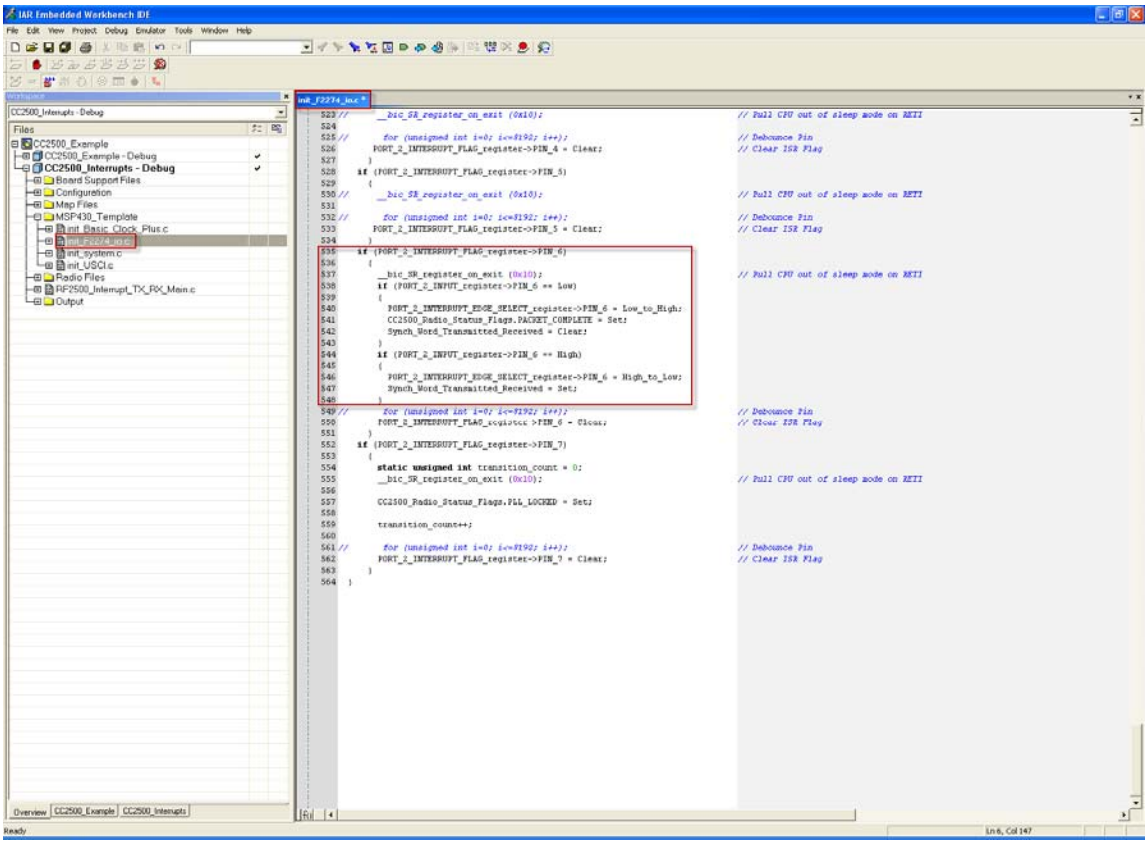
4. Add a vertical window:



5. Compare the Interrupt Main versus the Simple Main:

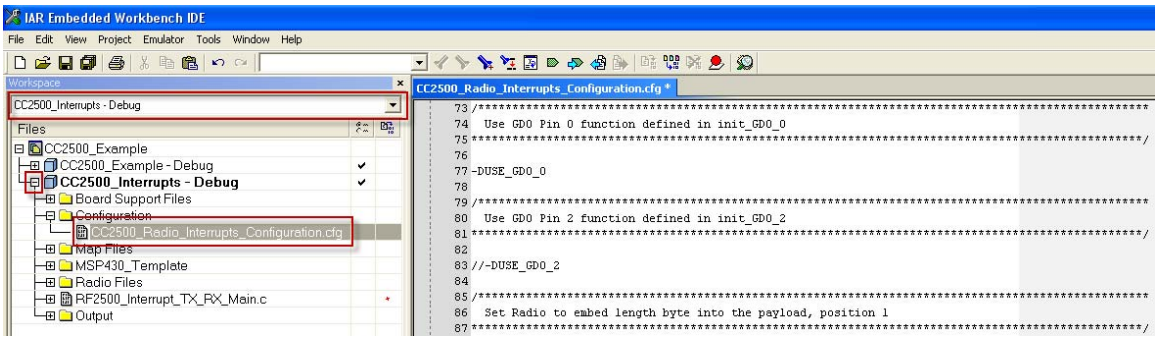


Interrupt Service Routine Location:



Lab 4: Adding Radio Information

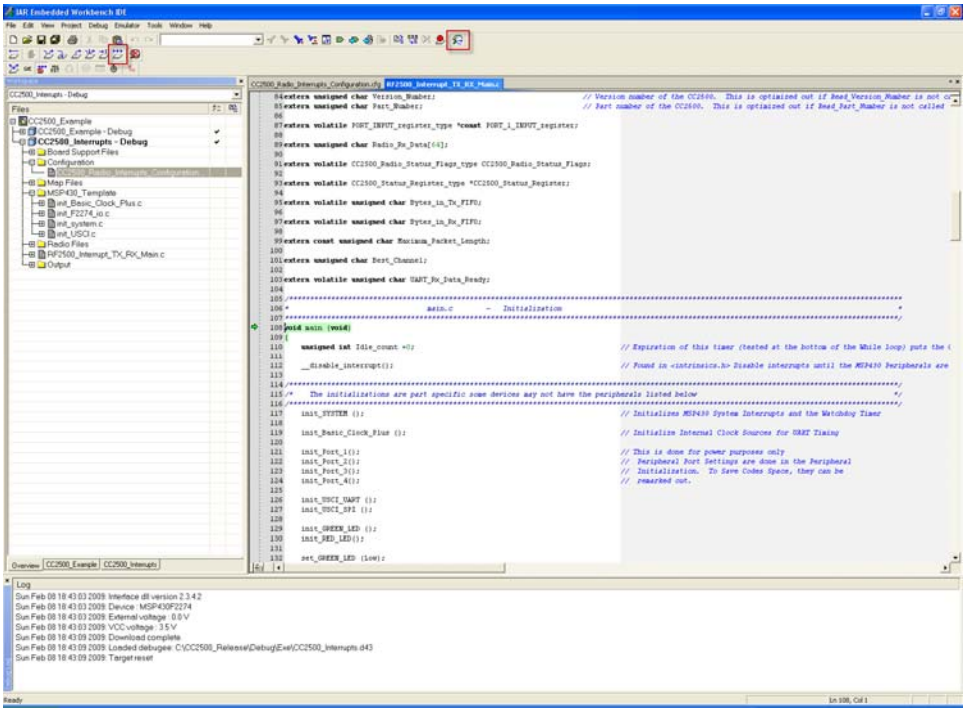
1. Select the CC2500_Interrupts – Debug Project and open the CC2500_Radio_Interrupts_Configurations.cfg file:



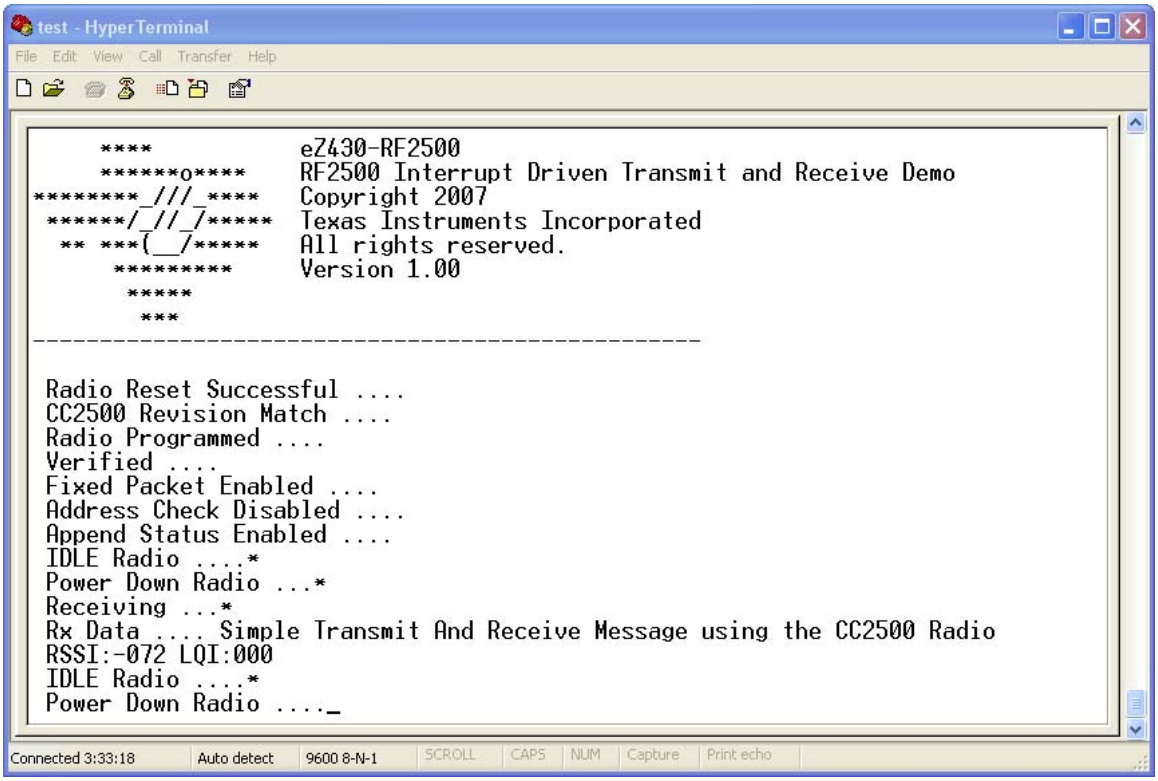
2. Set in CC2500_Radio_Interrupts_Configurations.cfg:

```
-DUSE_GDO_0
//-DUSE_GDO_2
//-DVARIABLE_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:

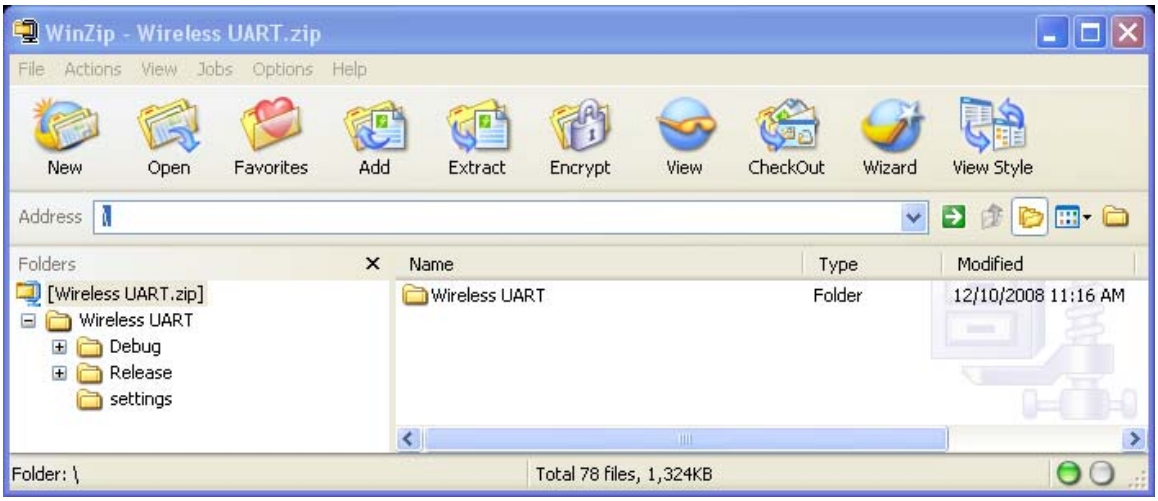


Output to Hyperterminal:

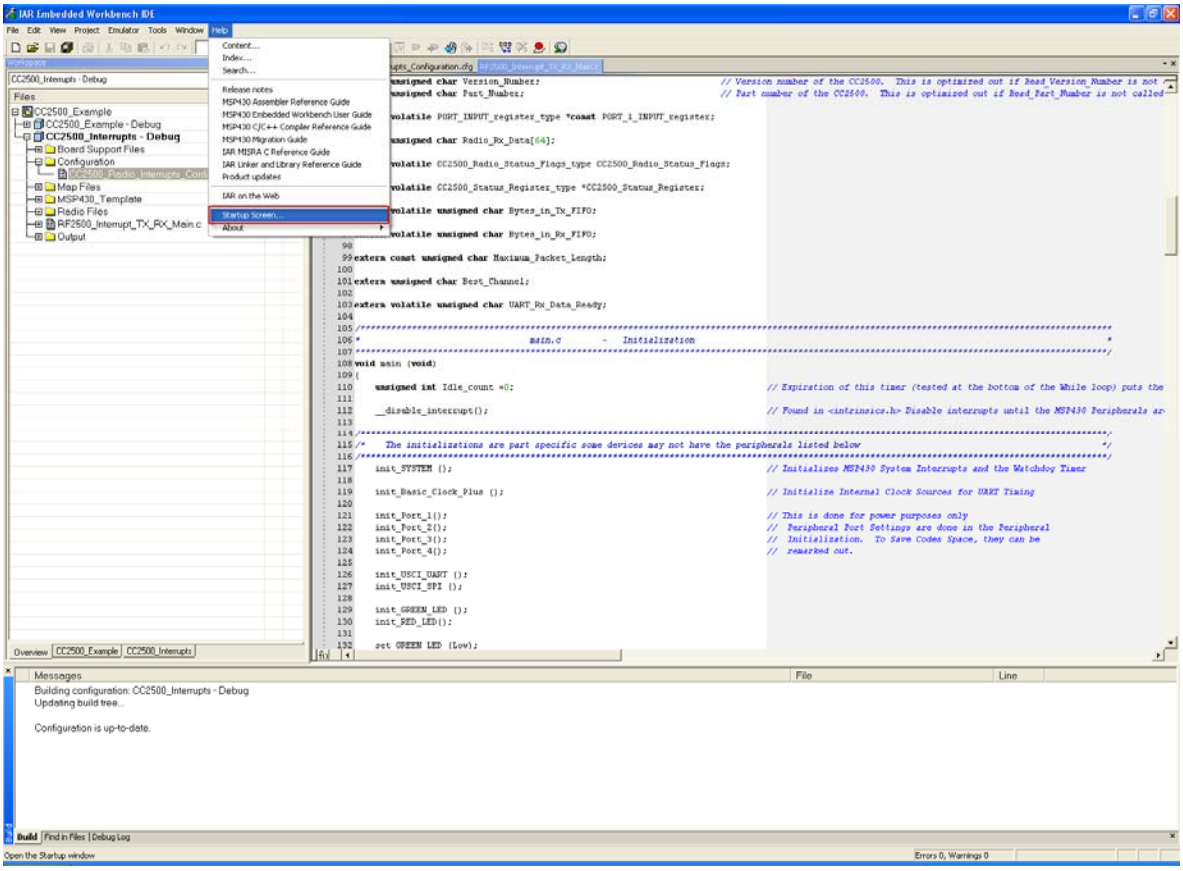


Lab 5: Wireless UART

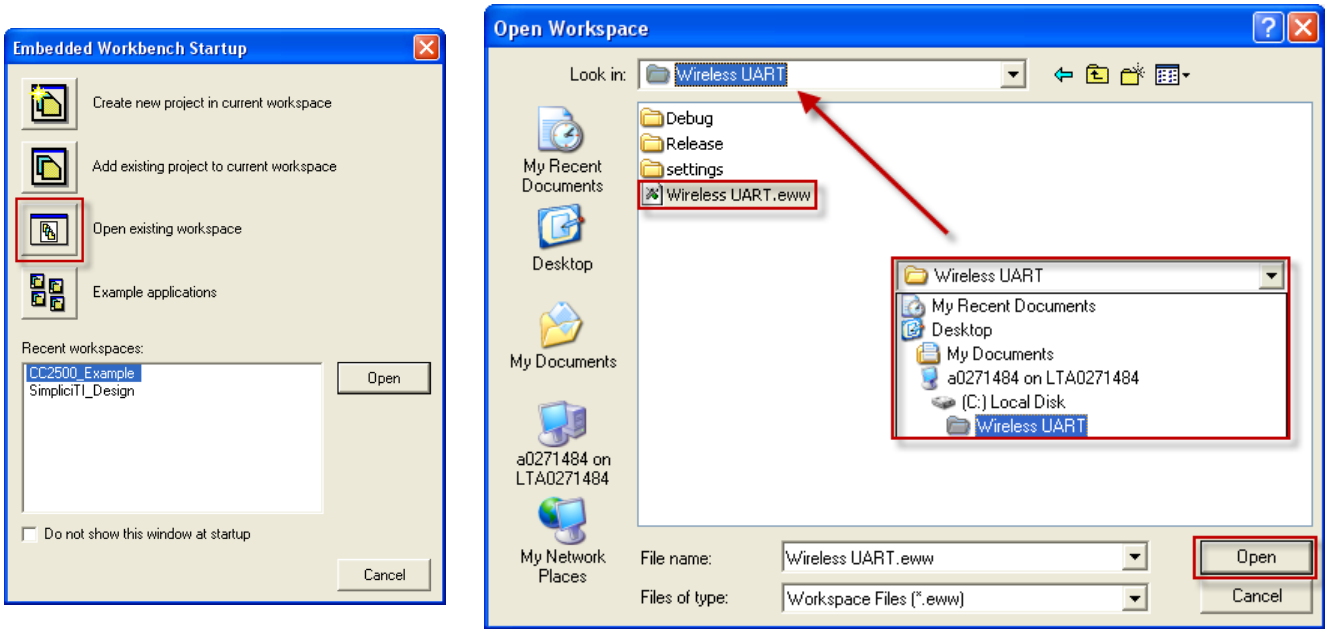
1. Unzip Wireless UART.zip



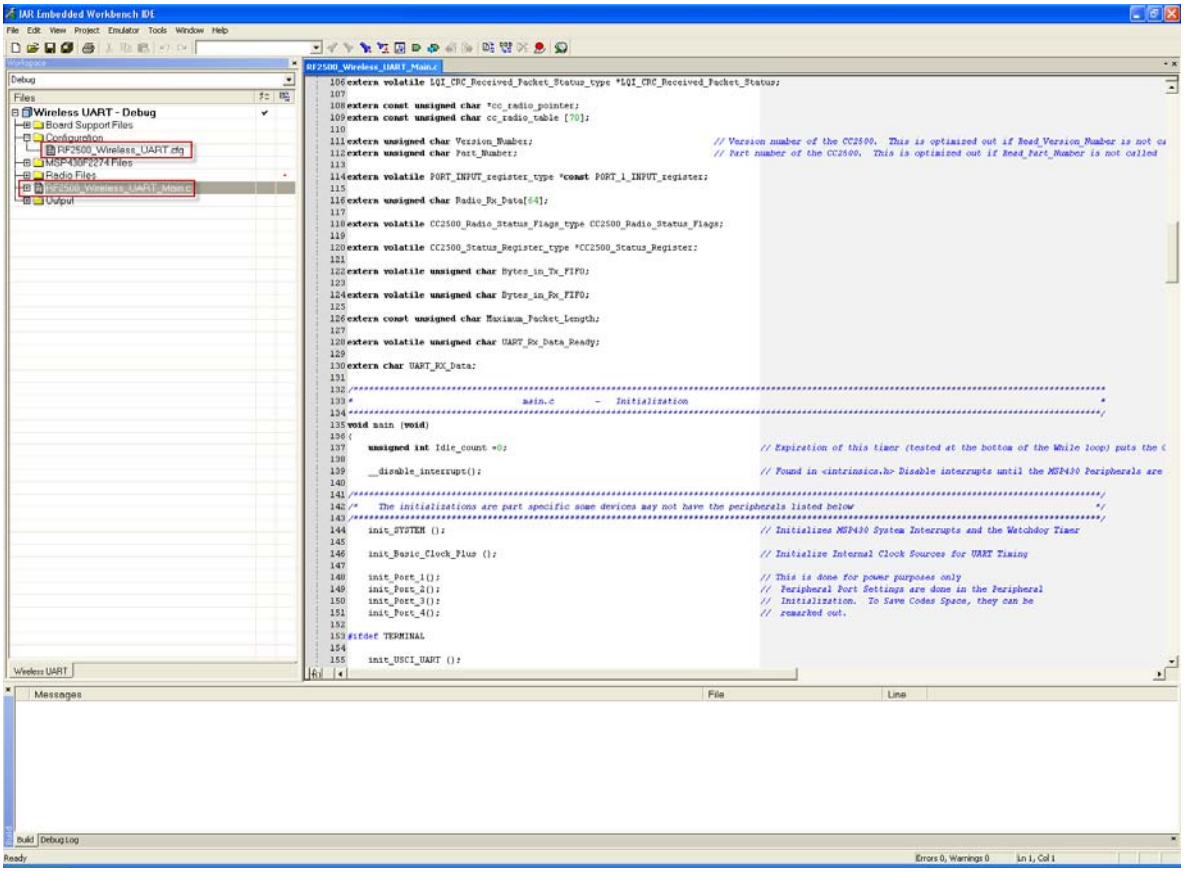
2. To start the New Workspace Wizard, make sure to Stop Debugging, then go to Help->Startup Screen



3. Open an existing workspace:



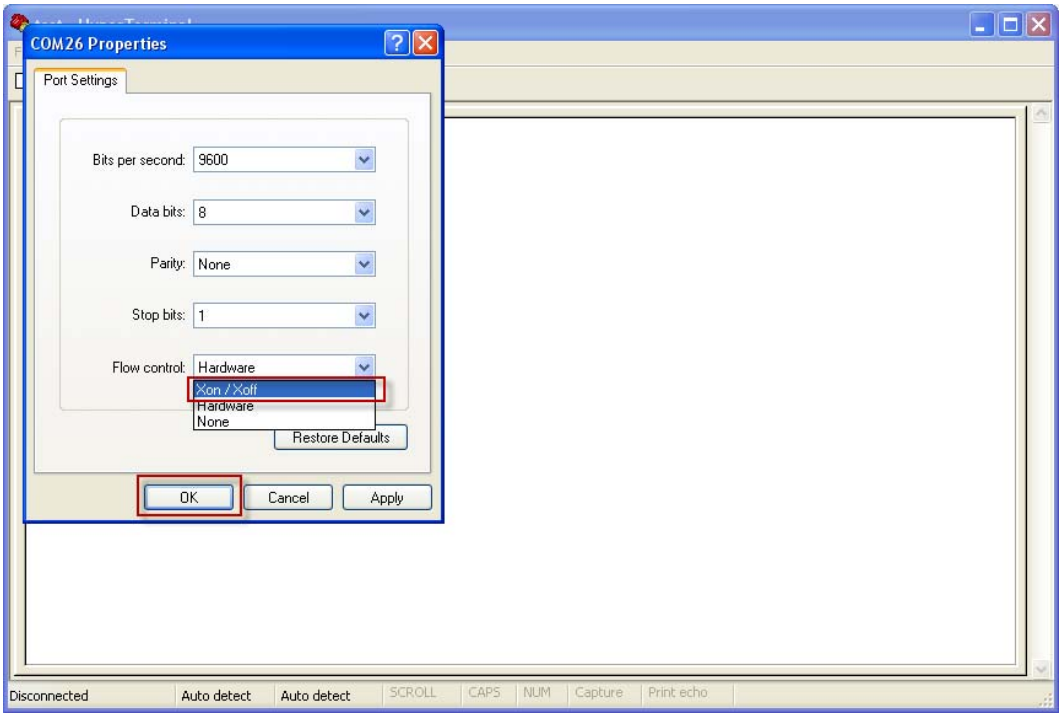
The Wireless UART is the General Radio project with a dedicated Main file and specific configuration file.



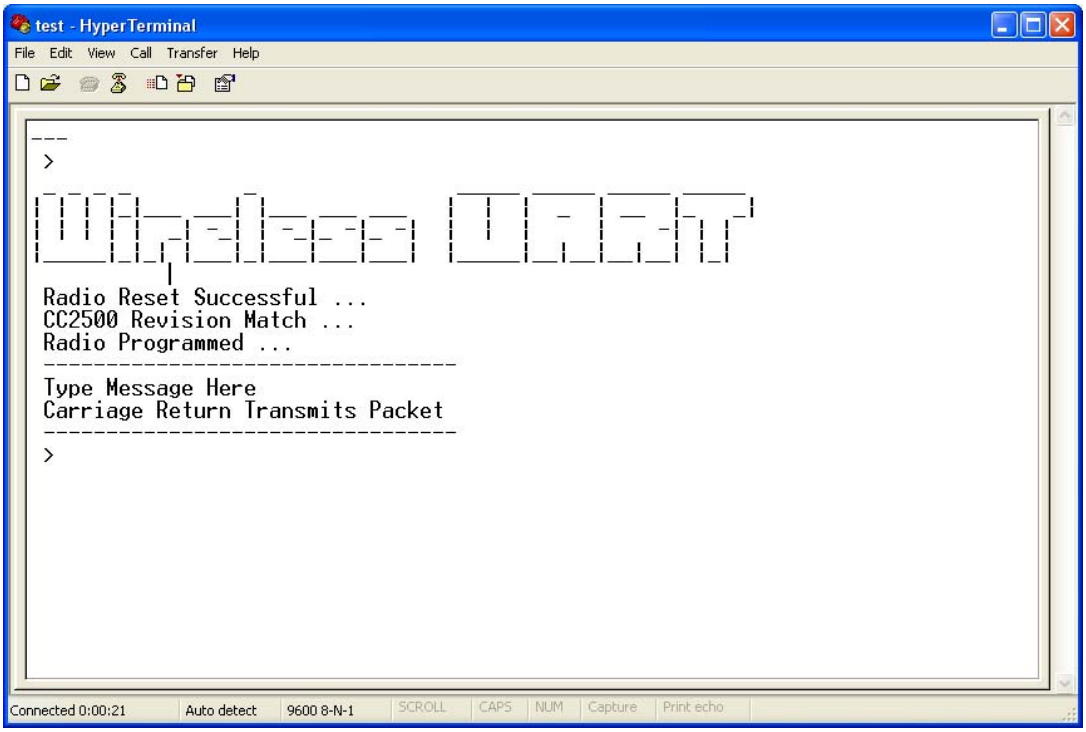
The Configuration is set as such:

```
-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 HyperTeminal 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.