



Getting Started with SimpliciTI™ and the eZ430

Mike Claassen

m-claassen@ti.com



Abstract

The first half of this module gives an overview of the components of the SimpliciTI network stack and their interaction in a SimpliciTI network.

The second half of this module is hands on and covers setting up a SimpliciTI network and the utilization of the Smart RF Studio Tool to set up the radio component of the stack.

The full IAR Studio Tool and an eZ430-RF2500 board are required for this module.



Agenda: Introducing SimpliCI™

- **SimpliCI™ Overview**
- **Topology Examples**
- **Stack Architecture and API Details**
- **eZ430-RF2500 Data Hub Example**
- **Device Objects Overview**
- **Build Time Configuration Options**



Benefits of this Training

- **Gain a working knowledge of low bandwidth RF communication topologies and capabilities**
- **Evaluate the multiple capabilities of the SimpliciTI™ protocol stack**
- **Be able to design and demonstrate a functional RF network**



SimpliciTI™ Overview

The basics behind the stack



What is SimpliciTI™?

- Low Power: Supports **sleeping devices** for low power consumption
- Low Cost: Uses **< 8K FLASH** and **< 1K RAM** depending on platform
- Flexible: Simple **star w/ extender** and/or **p2p** communication
- Simple: Utilizes a very basic core **6 instruction API**
- Versatile: Currently ported to the MSP430+CC1100/2500, CC111x/251x, CC2520 and CC2430/31

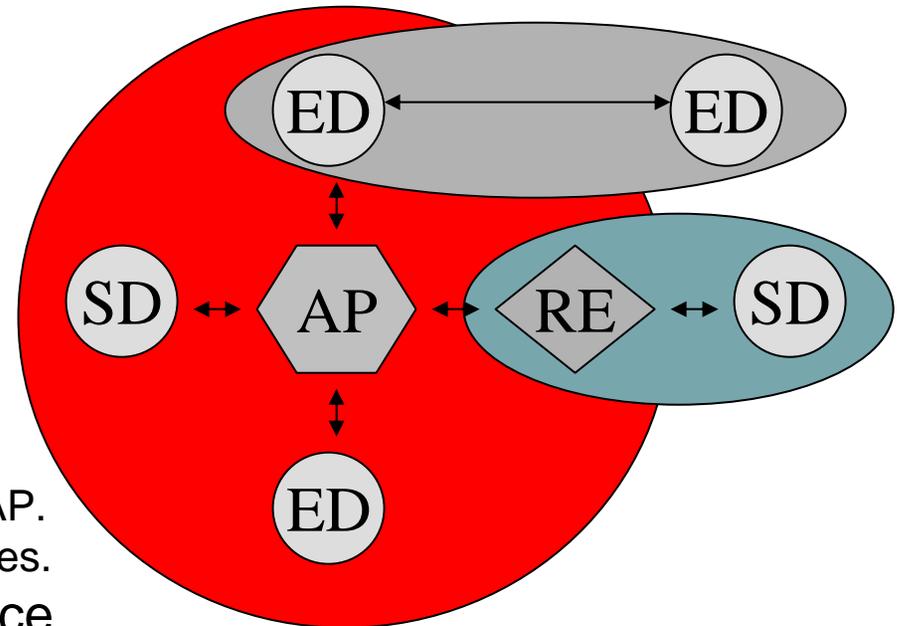
SimpliciTI targets quick time-to-market wireless solutions for **low power**, **low cost**, and **low data rate** networks without the need to know the details of the network support.



Basic Network Topology

Device Configurations:

-  **AP** AP – Access Point
 - Allows Access to the Network
 - Stores and Forward Messages
 - Serves as a Range Extender
-  **RE** RE – Range Extender
 - Repeats message Traffic
 - Like the AP, Device is always on
-  **ED** ED – End Device
 - No Store & Forward Services from AP.
 - Can sleep, but can't poll for messages.
-  **SD** SD – Sleeping (polling) End Device
 - Requires Store and Forward Services from the AP



Topologies:

- Access Point Star
- Access Point Star w/ Range Extender
- Peer to Peer



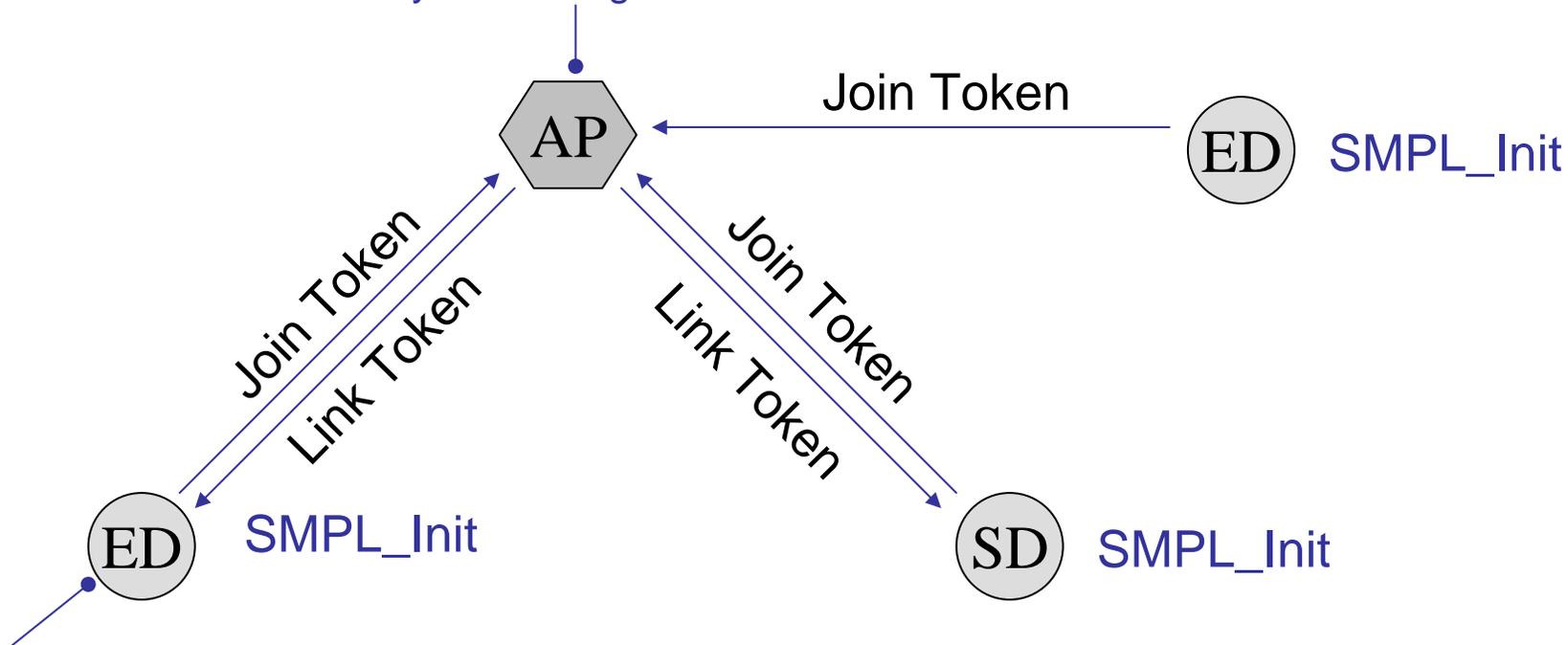
SimpliciTI™ Overview

Topology Examples



AP Managed: Peer to Peer Topology

1. Access Point must be powered on and initialized before the rest of the network.
3. If the Join Token matches, the AP will return a Link Token
6. The Access Point will establish Store and Forward data storage for the polling End Devices to ensure delivery of messages intended for them.

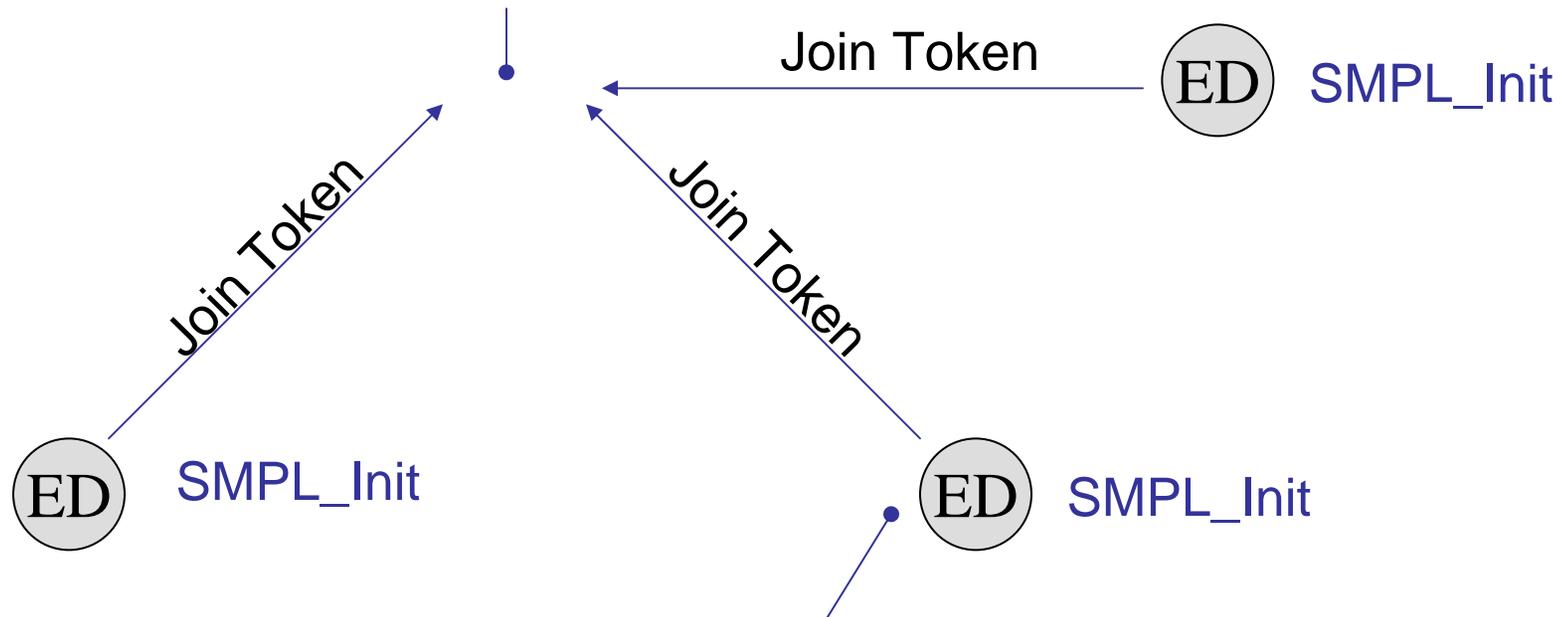


2. End Devices upon initialization, send a Join Token request to the network
4. The End Device overwrites its default Link Token w/ the new Link Token received from the AP
5. Sleeping End Devices follow the same process.
6. The AP is using information in the Join frame to determine this is a polling device.



No AP: Peer to Peer Topology

No Access Point exists, so the Join Token request falls on deaf ears. The SMPL_Init returns nothing, the ED's time out and use their default link token. A smpl_Status_t code is returned to the ED that the join request failed.



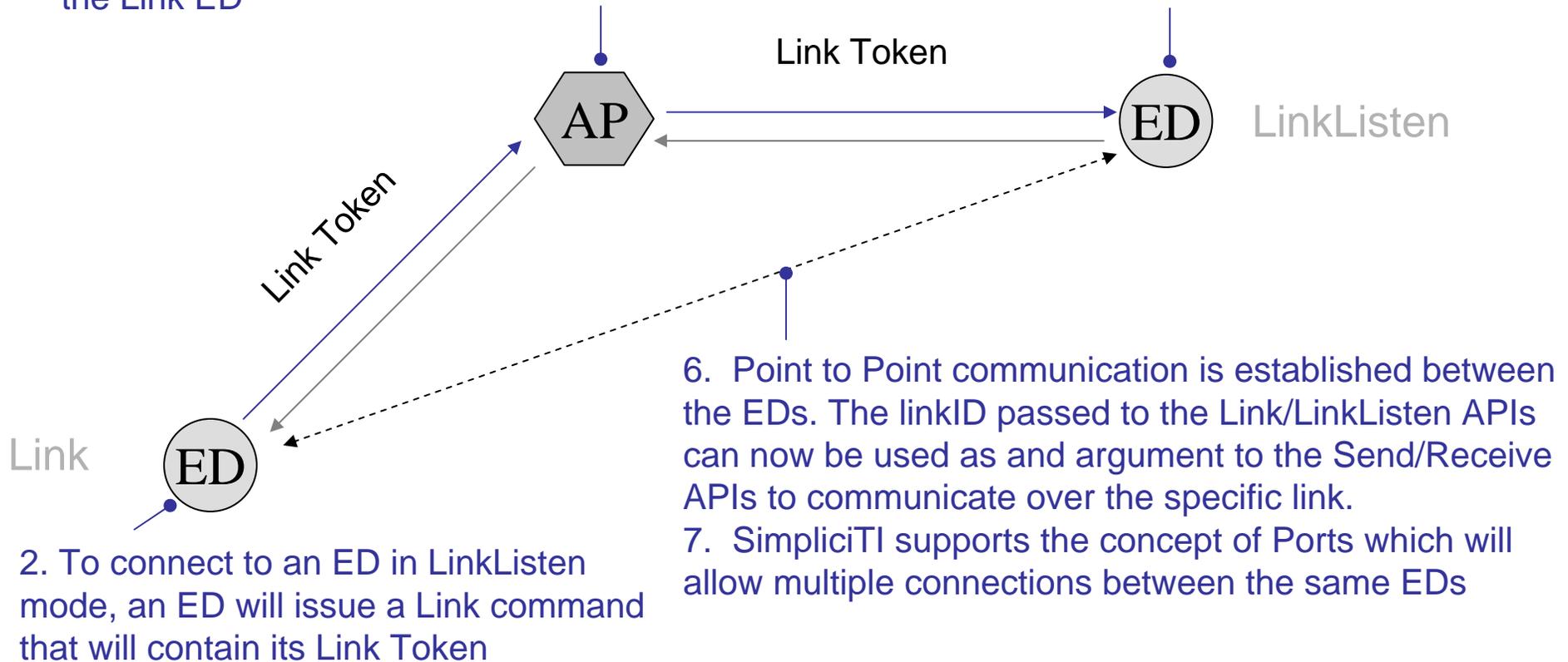
Since no Store and Forward Service exists, the polling End Devices cannot sleep as they may miss any asynchronous messages sent to them. An AP to required enable polling End Device functionality.



End Device Point to Point links

- 3. Serving the function of a range extender, the AP will repeat the message to extend the radio range of the Link ED
- 5. The Message is relayed back by the AP to the Link ED

- 1. An ED will accept invitations to Link with the LinkListen command
- 4. If the Link Token matches, the LinkListen ED will reply back with it's corresponding Link information



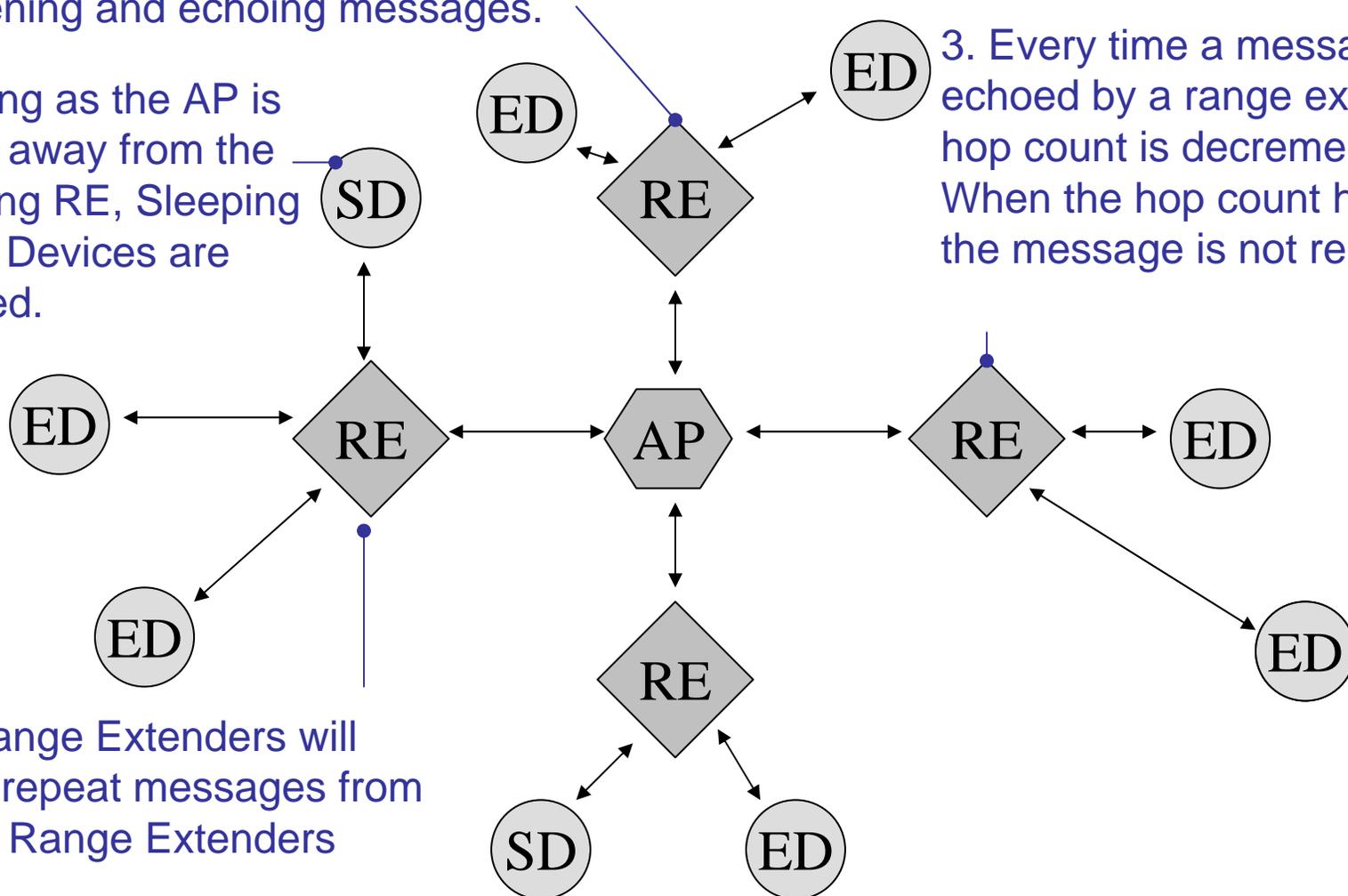


Adding a Range Extender

1. The Range Extender extends the network by listening and echoing messages.

5. As long as the AP is one hop away from the supporting RE, Sleeping (polling) Devices are supported.

3. Every time a message is echoed by a range extender a hop count is decremented. When the hop count hits 0, the message is not repeated



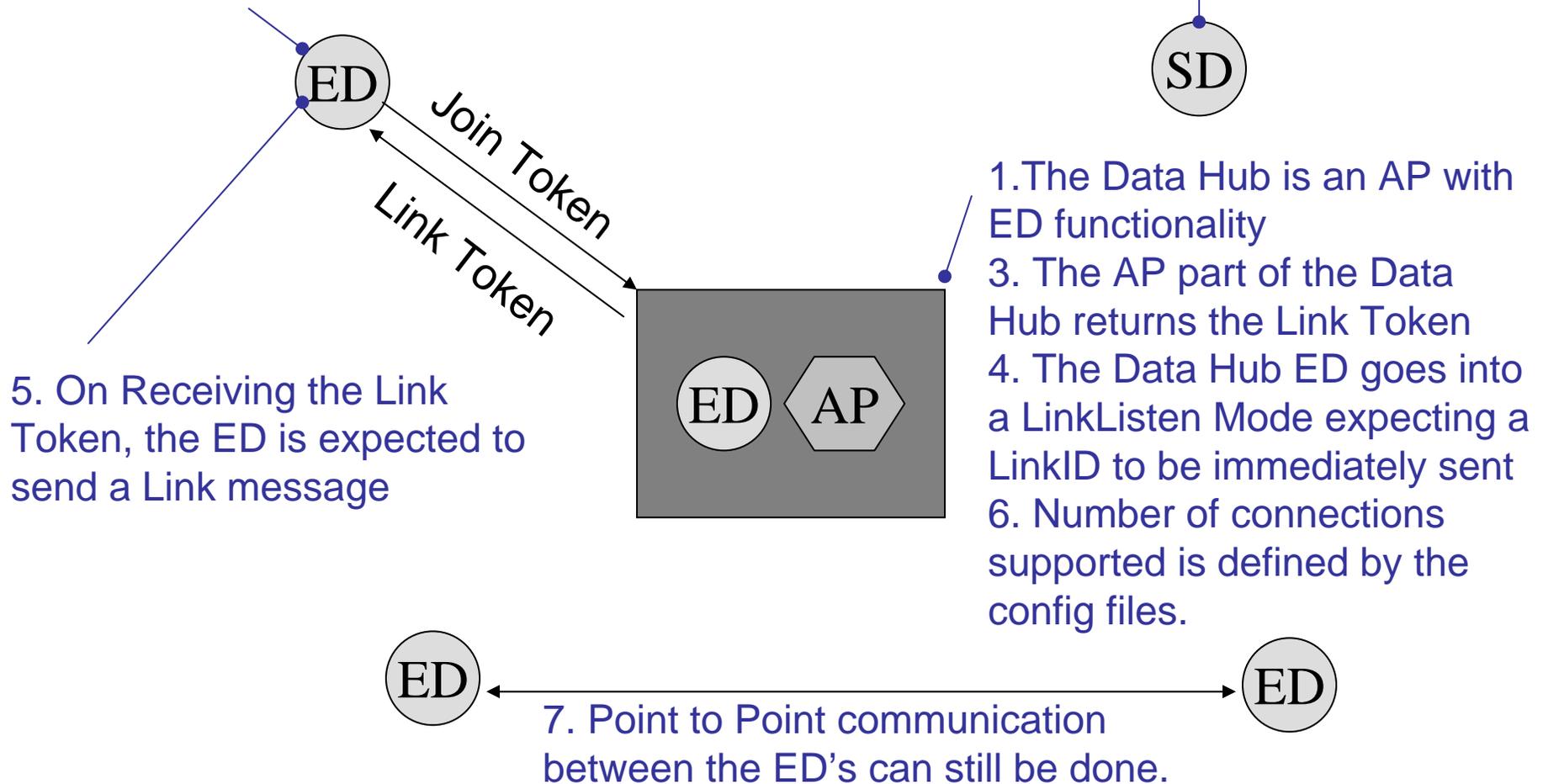
2. Range Extenders will NOT repeat messages from other Range Extenders



Star Topology using a Data Hub

2. On initialization, the ED sends a Join Token to the network

8. Sleeping End Devices can also be supported as the AP still performs Store and Forward.





SimpliciTI™ Overview

Stack Architecture and API Details



SimpliciTI™ API

- **Initialization**

- `smplStatus_t SMPL_Init(uint8_t (*callback)(linkID_t));`

- **Linking (bi-directional by default)**

- `smplStatus_t SMPL_Link(linkID_t *linkID);`
- `smplStatus_t SMPL_LinkListen(linkID_t *linkID);`

- **Peer-to-peer messaging**

- `smplStatus_t SMPL_Send(lid, *msg, len);`
- `smplStatus_t SMPL_Receive(lid, *msg, *len);`

- **Configuration**

- `smplStatus_t SMPL_Ioctl(object, action, *val);`



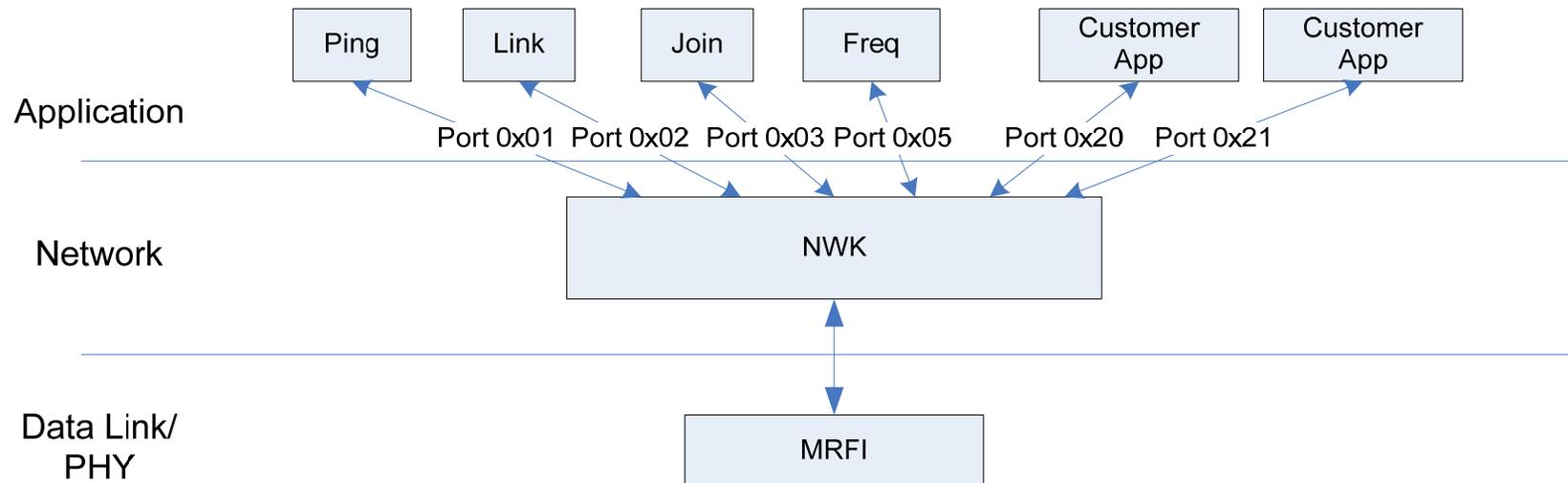
Architectural Overview

- Network Support

- Init (Join Token/Link Token exchange)
- Ping (Debug Only)
- link / linklisten (Establish peer-to-peer connections)
- nwk mgmt (General nwk mgmt, poll port)
- send / receive
- I/O

- Layers

- MRFI
- NWK
- nwk applications (Ports < 0x20)
- customer applications (Ports \geq 0x20)

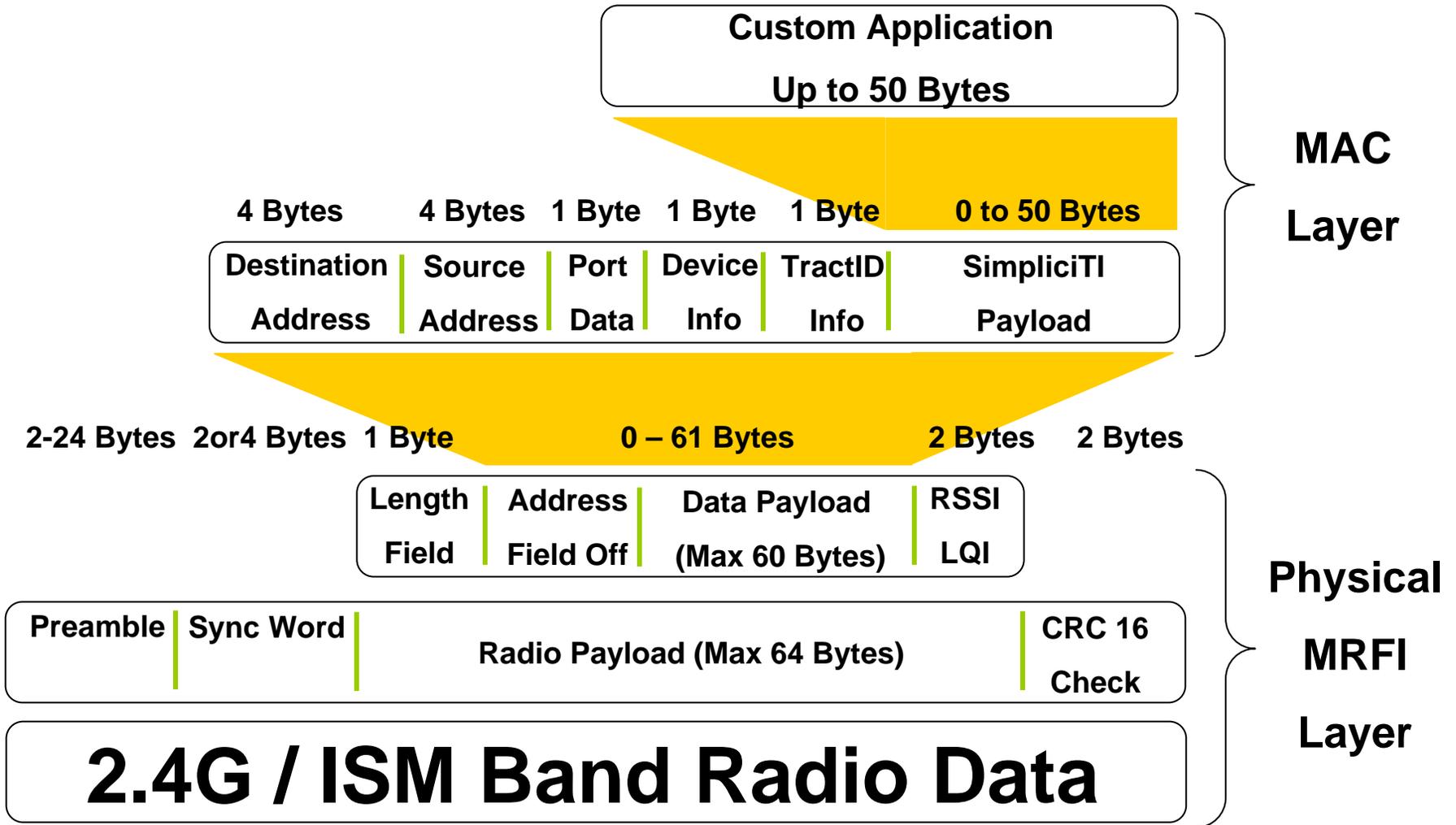


- SimpliciTI Address

- HW addr (4 byte) + Port
- Statically assigned HW addr
- Ports allow for more than one link between two devices



SimpliciTI™ – CC2500/CC1100



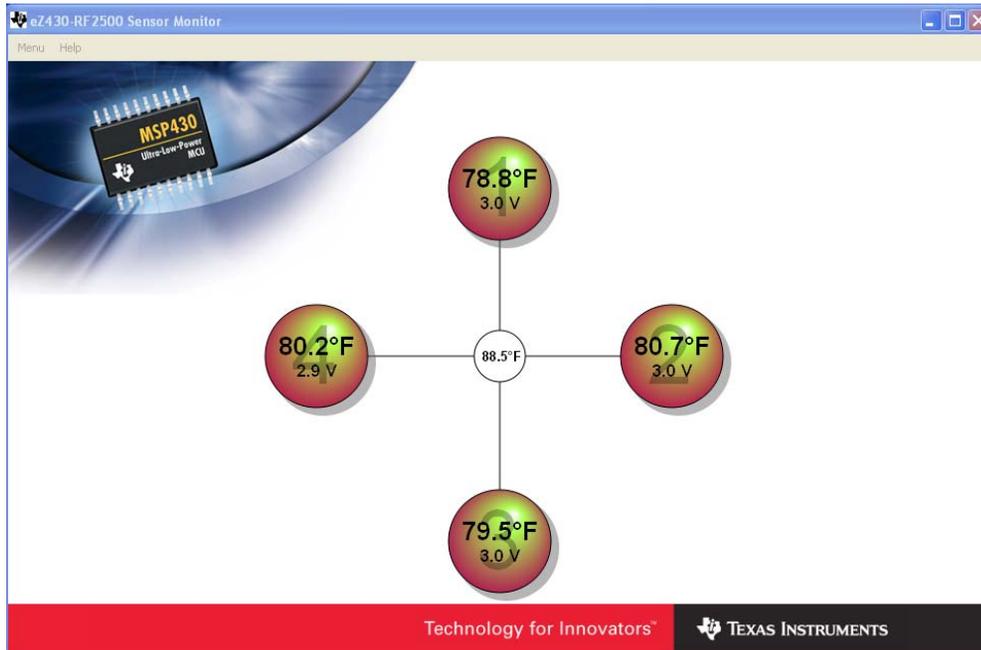


SimpliciTI™ Device Objects Overview

eZ430-RF2500 Data Hub Example



Temperature Monitor Demo



eZ430-RF2500
Wireless Development Tool



P.nbr.	Time (us)	Length	Dest. Address	Source Address	Port		Device Info			Transaction ID	Applicaton payload	Unknown Port	RSSI (dBm)	LOI	FCS
					Encryption Number	HCount	Rec.Type	Send.Type	HCount						
1141	+332509 =538690411	16	0x865B93D4	0x9E5CA29F	NO	0x20	ALWAYS_LISTEN	END_DEVICE	03	0x55	04 01 1E	Unknown Port 0x20	-53	11	OK
1142	+224373 =538914784	16	0x865B93D4	0x5CE2BCD6	NO	0x20	ALWAYS_LISTEN	END_DEVICE	03	0x77	17 01 1E	Unknown Port 0x20	-61	11	OK
1143	+123656 =539038440	16	0x865B93D4	0x138B4378	NO	0x20	ALWAYS_LISTEN	END_DEVICE	03	0xF9	08 01 1D	Unknown Port 0x20	-67	13	OK
1144	+136375 =539174815	16	0x865B93D4	0xC71F648A	NO	0x20	ALWAYS_LISTEN	END_DEVICE	03	0xCC	04 01 1E	Unknown Port 0x20	-69	17	OK



Getting Started

1. Download IAR Kickstart Tool

- A. <http://focus.ti.com/docs/toolsw/folders/print/iar-kickstart.html>
- B. Loads the Driver for the eZ430 USB UART

2. Download eZ430-RF2500 Sensor Monitor Demo

- A. <http://www.ti.com/litv/zip/slac139b>
- B. Loads the Sensor Monitor Application

3. Download SmartRF Studio

- A. <http://focus.ti.com/docs/toolsw/folders/print/smartrfstm-studio.html>
- B. Can be used to modify the Radio settings

4. Install IAR Kickstart Tool

5. Plug in eZ430-RF2400 Toolstick in the USB port

- A. USB Human Interface Device Installed
- B. USB Communications Port Installed
- C. C:\Program Files\IAR Systems\Embedded Workbench 5.0\430\drivers\TIUSBFET\WinXP

6. Install the eZ430-RF2500 Sensor Monitor Demo

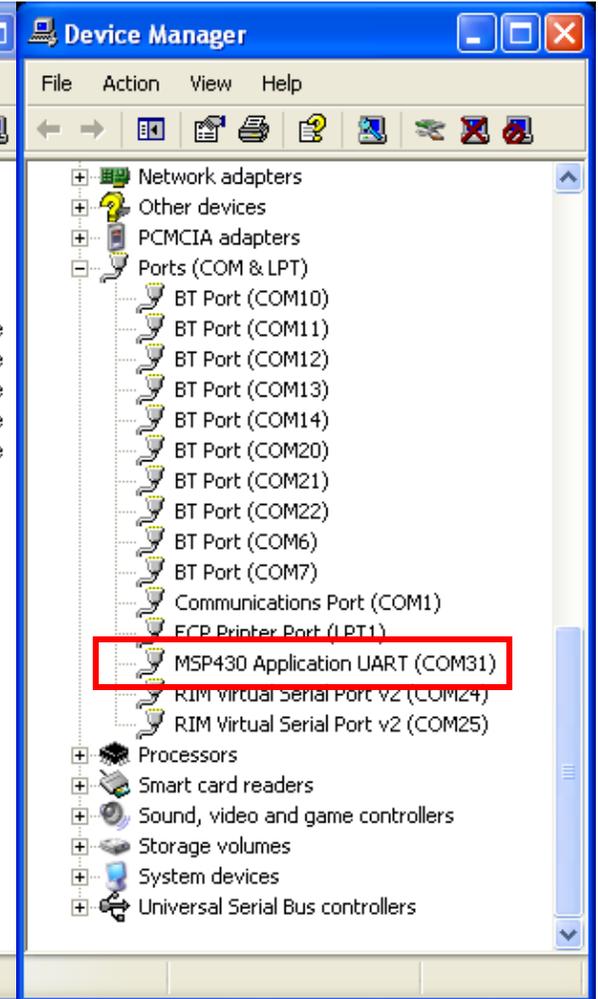
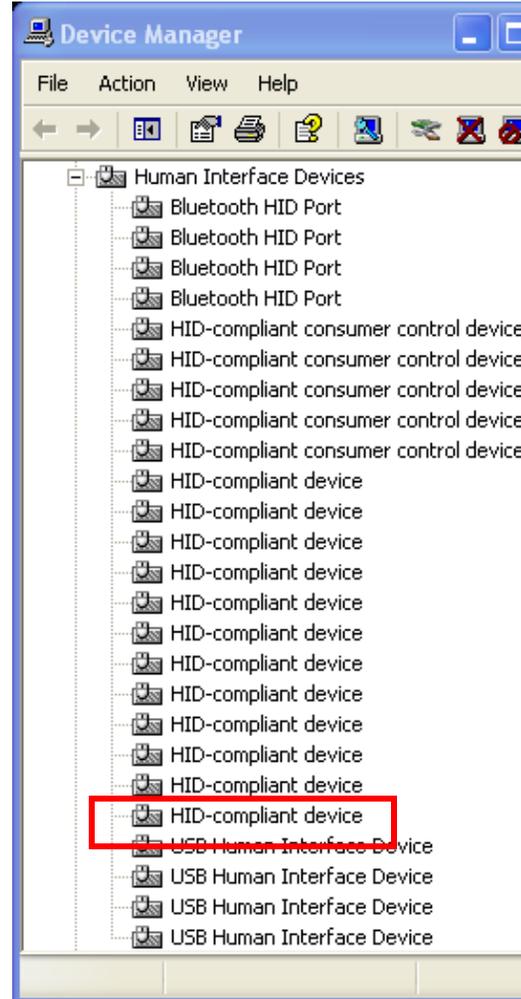
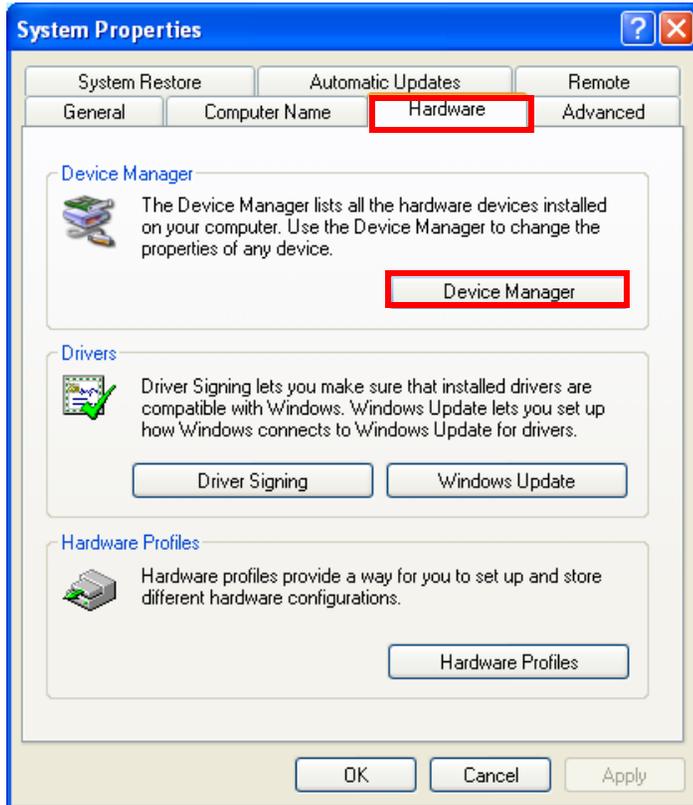
<http://www.ti.com/litv/zip/slac139b>

7. Install the Smart RF Studio Tool

<http://www.ti.com/litv/zip/swrc046k>

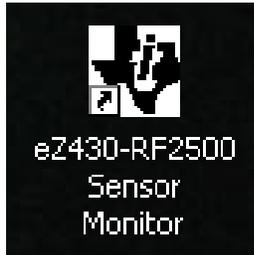


USB Driver Locations





eZ430-RF2500 Sensor Monitor



eZ430-RF2500 Sensor Monitor

Menu Help

- Disconnect
- Reset
- Settings
- Full Screen
- Console
- Exit

MSP430
Ultra-Low-Power
MCU

94.4°F

Settings

Settings

Port: Communications Port(COM1)

- BT Port(COM14)
- BT Port(COM20)
- BT Port(COM21)
- BT Port(COM22)
- BT Port(COM6)
- BT Port(COM7)
- RIM Virtual Serial Port v2(COM24)
- RIM Virtual Serial Port v2(COM25)

Maximum Temperature: [Slider]

Minimum Temperature: [Slider]

Temperature: MSP430 Application UART (COM31)

Minimum Distance: [Slider]

Disable Animations

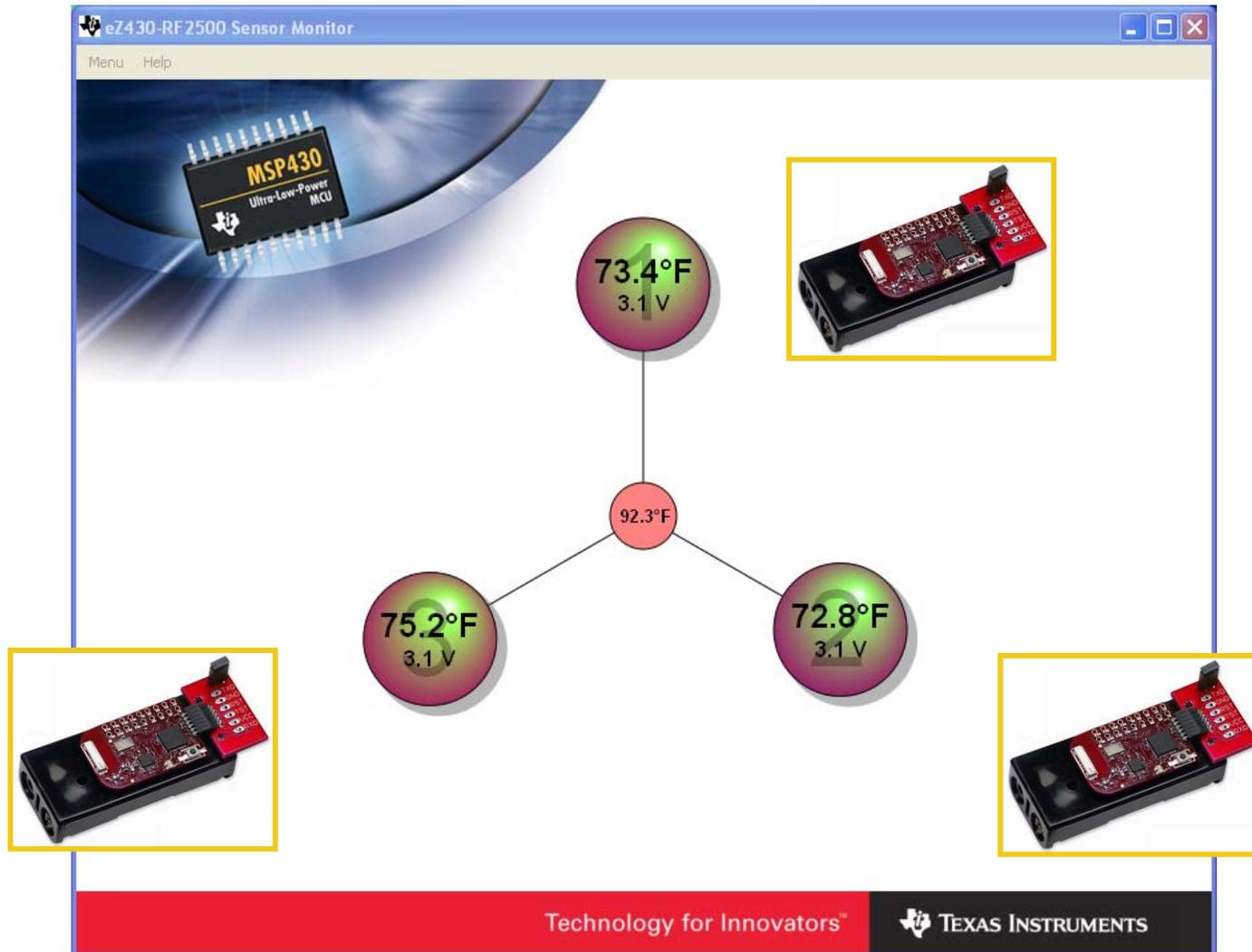
OK Cancel

Technology for Innovators™

TEXAS INSTRUMENTS



Power On Battery RF2500





View of Join Token Transaction

Texas Instruments General Packet Sniffer SimplicTI

File Help

P.nbr.	Time (us)	Length	Dest. Address	Source Address	Port	Device Info			Transaction ID	Application payload	Join	RSI (dBm)	LOI	FCS
					Encryption Number	Rec.Type	Send.Type	HCount			App.Info Token Nbr_Conn.			
1	+0	19	0x00000000	0x9E5CA29F	NO 0x03	ALWAYS_LISTEN	END_DEVICE	03	0xEF	05 08 07 06 05 02	0x05 0x5060708 0x02	-66	42	OK
2	+1395	19	0x9E5CA29F	0x865B93D4	NO 0x03	ALWAYS_LISTEN	ACCESS_POINT	01	0xEF	85 EF BE AD DE 00	0x85 0xDEADBEEF NO	-51	10	OK
3	+42312	21	0x00000000	0x9E5CA29F	NO 0x02	ALWAYS_LISTEN	END_DEVICE	03	0xF0	07 EF BE AD DE 3D 01 00	0x07 0xDEADBEEF 0x3D 0x01 ALWAYS_LISTEN	-63	33	OK
4	+3538	16	0x9E5CA29F	0x865B93D4	NO 0x02	ALWAYS_LISTEN	ACCESS_POINT	00	0xF0	82 20 00	0x82 0x20 ALWAYS_LISTEN	-51	15	OK
5	+832706	16	0x865B93D4	0x9E5CA29F	NO 0x20	ALWAYS_LISTEN	END_DEVICE	03	0xF1	00 01 1E	Unknown Port 0x20	-76	10	OK
6	+880847	16	0x865B93D4	0x9E5CA29F	NO 0x20	ALWAYS_LISTEN	END_DEVICE	03	0xF2	04 01 1E	Unknown Port 0x20	-72	10	OK
7	+879414	16	0x865B93D4	0x9E5CA29F	NO 0x20	ALWAYS_LISTEN	END_DEVICE	03	0xF3	04 01 1E	Unknown Port 0x20	-68	18	OK
8	+880933	16	0x865B93D4	0x9E5CA29F	NO 0x20	ALWAYS_LISTEN	END_DEVICE	03	0xF4	04 01 1E	Unknown Port 0x20	-72	11	OK
9	+882679	16	0x865B93D4	0x9E5CA29F	NO 0x20	ALWAYS_LISTEN	END_DEVICE	03	0xF5	00 01 1E	Unknown Port 0x20	-69	12	OK
10	+878420	16	0x865B93D4	0x9E5CA29F	NO 0x20	ALWAYS_LISTEN	END_DEVICE	03	0xF6	04 01 1E	Unknown Port 0x20	-69	12	OK

Setup | Select fields | Packet details | Address book | Display filter | Time line | Radio Settings

Field Name: Template:

Filter condition:

Filter management:

First And Add Remove Open Save Merge

Packet count: 47 Error count: 0 Filter Off



View of End Device Interaction

P.nbr.	Time (us)	Length	Dest. Address	Source Address	Port	Device Info			Transaction ID	Applicaton payload	Unknown Port	RSSI (dBm)	LOI	FCS
					Encryption Number	Rec.Type	Send.Type	HCount						
1131	+268883 =536422108	16	0x865B93D4	0x138B4378	NO 0x20	ALWAYS LISTEN	END DEVICE	03	0xF6	08 01 1D	0x20	-66	16	OK
1132	+297505 =536719613	16	0x865B93D4	0xC71F648A	NO 0x20	ALWAYS LISTEN	END DEVICE	03	0xC9	04 01 1E	0x20	-68	9	OK
1133	+212661 =536932274	16	0x865B93D4	0x9E5CA29F	NO 0x20	ALWAYS LISTEN	END DEVICE	03	0x53	08 01 1E	0x20	-53	12	OK
1134	+139426 =537071700	16	0x865B93D4	0x5CE2BCD6	NO 0x20	ALWAYS LISTEN	END DEVICE	03	0x75	17 01 1E	0x20	-61	15	OK
1135	+223891 =537295591	16	0x865B93D4	0x138B4378	NO 0x20	ALWAYS LISTEN	END DEVICE	03	0xF7	08 01 1D	0x20	-66	10	OK
1136	+241051 =537536642	16	0x865B93D4	0xC71F648A	NO 0x20	ALWAYS LISTEN	END DEVICE	03	0xCA	04 01 1E	0x20	-70	17	OK
1137	+274649 =537811291	16	0x865B93D4	0x9E5CA29F	NO 0x20	ALWAYS LISTEN	END DEVICE	03	0x54	08 01 1E	0x20	-53	11	OK
1138	+182979 =537994270	16	0x865B93D4	0x5CE2BCD6	NO 0x20	ALWAYS LISTEN	END DEVICE	03	0x76	13 01 1E	0x20	-60	8	OK
1139	+170673 =538164943	16	0x865B93D4	0x138B4378	NO 0x20	ALWAYS LISTEN	END DEVICE	03	0xF8	08 01 1D	0x20	-67	14	OK
1140	+192959 =538357902	16	0x865B93D4	0xC71F648A	NO 0x20	ALWAYS LISTEN	END DEVICE	03	0xCE	08 01 1E	0x20	-69	10	OK
1141	+332509 =538690411	16	0x865B93D4	0x9E5CA29F	NO 0x20	ALWAYS LISTEN	END DEVICE	03	0x55	04 01 1E	0x20	-53	11	OK
1142	+224373 =538914784	16	0x865B93D4	0x5CE2BCD6	NO 0x20	ALWAYS LISTEN	END DEVICE	03	0x77	17 01 1E	0x20	-61	11	OK
1143	+123656 =539038440	16	0x865B93D4	0x138B4378	NO 0x20	ALWAYS LISTEN	END DEVICE	03	0xF9	08 01 1D	0x20	-67	13	OK
1144	+136375 =539174815	16	0x865B93D4	0xC71F648A	NO 0x20	ALWAYS LISTEN	END DEVICE	03	0xCC	04 01 1E	0x20	-69	17	OK

Packet count: 1145 Error count: 10 Filter Off



Device Objects - Access Point

- **MSP430 Initialized**
- **Board initialized**
- **SMPL_Init**
 - **Set up as an Access Point**
 - **Callback utilized – Sets Semaphore**

```
/*  
 * Runs in ISR context. Reading the frame should be done in the  
 * application thread not in the ISR thread.  
 */  
static uint8_t sCB(LinkID_t lid)  
{  
    if (lid)  
    {  
        sPeerFrameSem++;  
    }  
    else  
    {  
        sJoinSem++;  
    }  
    // leave frame to be read by application.  
    return 0;  
}
```

- **LinkListen invoked**
 - **Until # of Connections are met**
 - **Allows additional EDs to Join**
 - **BSP_ENTER_CRITICAL_SECTION**
 - **Context Saves SimpliciTI Int State**
 - **Disables Interrupts**
 - **BSP_EXIT_CRITICAL_SECTION**
 - **Context Restores SimpliciTI Int State**
 - **Enables Interrupts**

The screenshot shows the IAR Embedded Workbench IDE interface. The project structure on the left includes 'Access Point - Debug' with sub-items like 'Application', 'demo_AP.c', 'vlo_rand.s43', 'Components', 'Configuration', 'Output', and 'End Device - Debug'. The main window displays the source code for 'demo_AP.c'. Two red boxes highlight specific code sections: the first box highlights the 'SMPL_Init(sCB);' call, and the second box highlights the 'main work loop' which includes a 'while (1)' loop with logic for joining peers and listening for connections.

```
SMPL_Init(sCB);  
  
// network initialized  
TXString( "Done\r\n", 6);  
  
// main work loop  
while (1)  
{  
    // Wait for the Join semaphore to be set by the receipt of a Join frame from a  
    // device that supports and End Device.  
  
    if (sJoinSem && (sNumCurrentPeers < NUM_CONNECTIONS))  
    {  
        // listen for a new connection  
        SMPL_LinkListen(&sLID[sNumCurrentPeers]);  
        sNumCurrentPeers++;  
        BSP_ENTER_CRITICAL_SECTION(intState);  
        if (sJoinSem)  
        {  
            sJoinSem--;  
        }  
        BSP_EXIT_CRITICAL_SECTION(intState);  
    }  
  
    // if it is time to measure our own temperature...  
    if (sSelfMeasureSem)  
    {  
        char msg [6];  
        char addr[] = {"HUB0"};  
        char rssi[] = {"0000"};
```



Device Objects - End Device

- **MSP430 Initialized**
- **Board initialized**
- **SMPL_Init**
 - Set up as End Device
 - Option to Sleep
 - No Call Back
 - Join Token Sent, & repeat if no reply
- **Link process Initiated**
 - Assumes LinkListen ED from AP
- **ADC Sampled**
 - Twice per Second (Temp & Vcc)
- **SMPL_Send**
 - Two Byte Temperature
 - One Byte Vcc
- **MSP430 Functionality**
 - Controls Radio On/Off
 - ADC Interrupt on Conversion

```
Flash_Addr[2] == 0xFF &&
Flash_Addr[3] == 0xFF )
{
    createRandomAddress();           // set Random device address at initial startup
}
lAddr.addr[0]=Flash_Addr[0];
lAddr.addr[1]=Flash_Addr[1];
lAddr.addr[2]=Flash_Addr[2];
lAddr.addr[3]=Flash_Addr[3];
SMPL_Iocctl(IOCTL_OBJ_ADDR, IOCTL_ACT_SET, &lAddr);
BCSCTL1 = CALBC1_8MHZ;             // Set DCO after random function
DCOCTL = CALDCO_8MHZ;

BCSCTL3 |= LFX1S_2;                // LFX1S = VLO
TACCTL0 = CCIE;                    // TACCR0 interrupt enabled
TACCR0 = 12000;                    // ~ 1 sec
TACTL = TASSEL_1 + MC_1;           // ACLK, upmode

// keep trying to join until successful. toggle LEDs to indicate that
// joining has not occurred. LED3 is red but labeled LED 4 on the EXP
// board silkscreen. LED1 is green.
while (SMPL_NO_JOIN == SMPL_Init((uint8_t *)(&linkID_t))0)
{
    BSP_TOGGLE_LED1();
    BSP_TOGGLE_LED2();
    __bis_SR_register(LPM3_bits + GIE); // LPM3 with interrupts enabled
}
// unconditional link to AP which is listening due to successful join.
linkTo();
```



Device Objects - Access Point

- **MSP430 Functionality**
 - **ADC Interrupt on Conversion**
 - Local Temperature
 - Vcc Conversion
 - **USB Backchannel UART**
 - Local Temperature
 - Address Hub
 - Remote Temp and Vcc
 - LinkID Address
 - Two Modes
 - eZ430-RF2500 Sensor Monitor
 - UART Terminal Program
- **SimpliciTI Configuration**
 - **Star hub in the network**
 - 1 AP per net (Join Token)
 - **Always-on**
 - **Configured as a Data Hub**
 - AP + ED

```
void transmitDataString(char addr[4],char rssi[3], char msg[MESSAGE_LENGTH] )
{
    char temp_string[] = {" XX.XC"};
    int temp = msg[0] + (msg[1]<<8);

    if( !degCMode )
    {
        temp = (((float)temp)*1.8)+320;
        temp_string[5] = 'F';
    }
    if( temp < 0 )
    {
        temp_string[0] = '-';
        temp = temp * -1;
    }
    else if( ((temp/1000)%10) != 0 )
    {
        temp_string[0] = '0'+((temp/1000)%10);
    }
    temp_string[4] = '0'+(temp%10);
    temp_string[2] = '0'+((temp/10)%10);
    temp_string[1] = '0'+((temp/100)%10);

    if( verboseMode )
    {
        char output_verbose[] = {"\r\nNode:XXXX,Temp:-XX.XC,Battery:X.XV,Strength:XXX%,RE:no "};

        output_verbose[46] = rssi[2];
        output_verbose[47] = rssi[1];
        output_verbose[48] = rssi[0];
    }
}
```



UART Terminal Output Types

```

IAR Embedded Workbench IDE
File Edit View Project Emulator Tools Window Help
Debug
Files
  Access Point - Debug
    Application
    Components
      bsp
      mrfi
      nwk
      SimpliciTI EZ430RF NWK AP Data Collector Library.lib
    Configuration
    Output
Overview Access Point End Device
demo_AP.c
//data for terminal output
const char splash[] = ("r\n-----r\n ****r\n ****
no_init volatile int tempOffset @ 0x10F4; // Temperature offset set at production
no_init volatile char Flash_Addr[4] @ 0x10F0; // Flash address set randomly
// reserve space for the maximum possible peer Link IDs
static linkID_t sLID(NUM_CONNECTIONS);
static uint8_t sNumCurrentPeers;
// callback handler
static uint8_t sCB(linkID_t);
// work loop semaphores
static uint8_t sPeerFrameSem;
static uint8_t sJoinSem;
static uint8_t sSelfMeasureSem;
// mode data verbose = default, deg F = default
char verboseMode = 1;
char degCMode = 0;
void main (void)
{
  addr_t lAddr;
  bspIState_t intState;
  WDTCTL = WDTPW + WDTHOLD; // Stop WDT
  // delay loop to ensure proper startup before SimpliciTI increases DCO

```

```

****
****
*****0*****
*****///*****
*****///*****
** **{/*****
*****
****
****
eZ430-RF2500
Temperature Sensor Network
Copyright 2007
Texas Instruments Incorporated
All rights reserved.
Version 1.02

```

```

Test - HyperTerminal
File Edit View Call Transfer Help
Node:0001,Temp: 76.1F,Battery:2.9V,Strength:059%,RE:no
Node:0003,Temp: 70.5F,Battery:3.1V,Strength:064%,RE:no
Node:0002,Temp: 71.9F,Battery:3.1V,Strength:060%,RE:no
Node:HUB0,Temp: 93.0F,Battery:3.5V,Strength:000%,RE:no
Node:0001,Temp: 75.2F,Battery:2.9V,Strength:057%,RE:no
Node:0003,Temp: 69.8F,Battery:3.1V,Strength:064%,RE:no
Node:HUB0,Temp: 93.0F,Battery:3.5V,Strength:000%,RE:no
Node:0002,Temp: 71.9F,Battery:3.1V,Strength:059%,RE:no

```

↑
verboseMode = 1

→
verboseMode = 0

```

Test - HyperTerminal
File Edit View Call Transfer Help
$HUB0, 93.0F,3.5,000,N#
$0002, 68.3F,3.1,043,N#
$0003, 70.8F,2.9,044,N#
$0001, 71.9F,3.1,059,N#
$HUB0, 93.0F,3.5,000,N#
$0002, 68.3F,3.1,043,N#
$0003, 70.8F,2.9,043,N#
$HUB0, 93.0F,3.5,000,N#
$0001, 72.6F,3.1,059,N#
$0002, 68.3F,3.1,046,N#

```



SimpliciTI™ Overview

Radio Settings and RF Studio



SimpliciTI™ Radio Settings

```

//*****
// SmartRF Studio(tm) Export
//
// Radio register settings specified with C-code
// compatible #define statements.
//*****

#ifndef SMARTRF_CC2500_H
#define SMARTRF_CC2500_H

#define SMARTRF_RADIO_CC2500

#define SMARTRF_SETTING_FSCTRL1 0x07
#define SMARTRF_SETTING_FSCTRL0 0x00
#define SMARTRF_SETTING_FREQ2 0x5D
#define SMARTRF_SETTING_FREQ1 0x93
#define SMARTRF_SETTING_FREQ0 0xB1
#define SMARTRF_SETTING_MDCFG4 0x2D
#define SMARTRF_SETTING_MDCFG3 0x3B
#define SMARTRF_SETTING_MDCFG2 0x73
#define SMARTRF_SETTING_MDCFG1 0x22
#define SMARTRF_SETTING_MDCFG0 0xEB
#define SMARTRF_SETTING_CHANNR 0xEB
#define SMARTRF_SETTING_DEVIATN 0x81
#define SMARTRF_SETTING_FREND1 0xB6
#define SMARTRF_SETTING_FREND0 0x10
#define SMARTRF_SETTING_MCSM0 0x18
#define SMARTRF_SETTING_FOCCFG 0x1D
#define SMARTRF_SETTING_BSCFG 0x1C
#define SMARTRF_SETTING_AGCCTRL2 0xC7
#define SMARTRF_SETTING_AGCCTRL1 0x00
#define SMARTRF_SETTING_AGCCTRL0 0xB0
#define SMARTRF_SETTING_FSCAL3 0xEA
#define SMARTRF_SETTING_FSCAL2 0x0A
#define SMARTRF_SETTING_FSCAL1 0x00
#define SMARTRF_SETTING_FSCAL0 0x11
#define SMARTRF_SETTING_FSTEST 0x59
#define SMARTRF_SETTING_TEST2 0x88
#define SMARTRF_SETTING_TEST1 0x31
#define SMARTRF_SETTING_TEST0 0x0B
#define SMARTRF_SETTING_IOCFG0D 0x06
#define SMARTRF_SETTING_IOCFG0C 0x01
#define SMARTRF_SETTING_PKTCTRL1 0x04
#define SMARTRF_SETTING_PKTCTRL0 0x05
#define SMARTRF_SETTING_ADDR 0x00
#define SMARTRF_SETTING_PKTLEN 0xFF

#endif

```

CHANNR – Sets the Channel

IOCFG0D – Sets the ISR Pin



SmartRF Studio



Calculation Window - CC2500 - SmartRF® Studio

File Settings Help

Current chip values:

- IOCFG2 [0x00]: 0x00
- IOCFG1 [0x01]: 0x00
- IOCFG0 [0x02]: 0x00
- IOCFG0A1 [0x02]: 0x00
- IOCFG0A2 [0x02]: 0x00
- FIFOTHRESH [0x03]: 0x00
- SYNCR1 [0x04]: 0x00
- SYNCR0 [0x05]: 0x00
- PKTLEN [0x06]: 0x00
- PKTCTRL1 [0x07]: 0x00
- PKTCTRL0 [0x08]: 0x00
- ADDR [0x09]: 0x00
- CHANNR [0x0A]: 0x00

Normal View Register View Notes

Chip revision: E (VERSION = 0x03)

X-tal frequency: 26.000000 MHz RF output power: 0 dBm PA ramping

Deviation: 38.085938 kHz Datarate: 2.398968 kBaud Modulation: 2-FSK Manchester

RF frequency: 2432.999908 MHz Channel: 199.951172 kHz Channel number: 0 RX filterbandwidth: 203.125000 kHz

Preferred settings:

Datarate	Deviation	Modulation	RX filterbandwidth	Optimization
2-FSK	203 kHz	Sensitivity		
2-FSK	203 kHz	Current 0 - 85 C		
2-FSK	232 kHz	Sensitivity		
2-FSK	232 kHz	Current		
MSK	540 kHz	Sensitivity		
MSK	540 kHz	Current		
MSK	812 kHz	Sensitivity		

Correlation:

Register Components

- PA value = 0xFE
- RF output power -> PATABLE
- FREQ2 = 0x5D
- RF Frequency -> FREQ[23:16]
- FREQ1 = 0x93
- RF Frequency -> FREQ[15:8]
- FREQ0 = 0xB1
- RF Frequency -> FREQ[7:0]
- FSCTRL1 = 0x08
- IF Frequency -> FREQ_IF[4:0] -> 203.13 kHz
- FSCTRL0 = 0x00
- RF Frequency offset -> FREQOFF[7:0]
- MDMCFG4 = 0x86
- Data rate (exponent) -> DRATE_E
- Channel bandwidth (exponent) -> CHANBW_E
- Channel bandwidth (mantissa) -> CHANBW_M
- MDMCFG3 = 0x83
- Data rate (mantissa) -> DRATE_M
- MDMCFG2 = 0x03

and write settings Copy settings to Register View

Packet RX Packet TX PER test

Sync word: 30/32 sy Address config: No addr CRC Manual Init

Packet count: 200 Address: FIFO Autoflush

MDMCFG1 = 0x20 Forward Error Correction -> FEC_EN

MDMCFG2 = 0x03 Sync mode -> SYNC_MODE[2:0]

PKTCTRL0 = 0x05 Packetformat -> PKT_FORMAT[5:4]

Forced to 0 by FW.

CRC operation -> CRC_EN[2]

Forced to 1 by FW.

Packet config -> LENGTH_CONFIG[1:0]

Forced to 1 by FW.

Start buffered RX Stop RX

SmartRF® Studio

SmartRF® 01 DK | SmartRF® 02 DK | SmartRF® 03 DK | SmartRF® 04 DK | SmartRF® 05 DK

Current Status USB DID FW ID

- Calculation Window - CC1100
- Calculation Window - CC1101
- Calculation Window - CC1110
- Calculation Window - CC1111
- Calculation Window - CC1150
- Calculation Window - CC2430
- Calculation Window - CC2431
- Calculation Window - CC2500
- Calculation Window - CC2510
- Calculation Window - CC2511
- Calculation Window - CC2550

Product info: [SmartRF® productline](#)

Load USB Firmware Load MCU prototype firmware Start

File versions...

Device ID: Not Connected Last executed command: Date: 06.08.2008, Time: 15:08:03



Example

Set Radio To:

1. MSK Modulation
2. 1MHz Channels
3. Base Frequency of 2.4GHz
4. Data Rate of 500kB/s
5. RF output power 0dBm
6. Set Channel to 15

2. Max Channel is 405kHz:
 $333.25\text{MHz} \times 3 = 1\text{MHz}$

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

1. The Preferred Settings Window can get you started with the 500kB/s Data Rate and the MSK Modulation

Values will change in the Register Window

3. Channel 0 defines the Base Frequency



Example

Set Radio To:

1. MSK Modulation
2. 1MHz Channels
3. Base Frequency of 2.4GHz
4. Data Rate of 500kB/s
5. RF output power to 0dBm
6. Set Channel to 15

5. RF Output Power using pull down

6. Channel 15 x 3 = 45

The screenshot shows the TI RF Studio software interface with several key settings highlighted by red boxes and arrows:

- RF frequency:** 2414.996368 MHz
- Channel:** 333.251953 kHz
- Channel number:** 45
- RF output power:** -10 dBm
- Modulation:** MSK
- Datarate:** 499.877930 kBaud
- RF filter bandwidth:** 812.500000 kHz

The interface also displays a list of register values on the left and a 'Correlation' window on the right showing register components like PA value, RF output power, and various frequency and data rate settings.



To Review Actual Register Settings

Selecting the Copy Settings to Register View puts the new values in the Register View Window

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

Register View details:

- IF frequency -> FREQ_IF[4:0] => 304.63 kHz
- FSCTRL0 = 0x00
- RF Frequency offset -> FREQOFF[7:0]
- MDMCFG4 = 0x0E
- Data rate (exponent) -> DRATE_E
- Channel bandwidth (exponent) -> CHANBW_E
- Channel bandwidth (mantissa) -> CHANBW_M
- MDMCFG3 = 0x3B
- Data rate (mantissa) -> DRATE_M
- MDMCFG2 = 0x73

Register View controls:

- Write TX FIFO
- Insert length Length: [red bar]
- Read RX FIFO
- Write PATABLE: 97 00 00 00 00 00 00
- Read PATABLE

Register View status:

SRES	SKOFF	SFSTXON	SCAL	SRX	STX	SIDLE
SAFC	SWDR	SPWD	SFRX	SFTX	SWORRST	SNOP



Exporting the settings to 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

Normal View Register View

IOCFG2 (0x00) Write 0 0x 28 Read value: 0x00

IOCFG1 (0x01) Write 1 0x 28 Read value: 0x00

IOCFG0D (0x02) Write 2 0x 00 Read value: 0x00

IOCFG0A1 (0x02) Write 3 0x 00 Read value: 0x00

IOCFG0A2 (0x02) Write 4 0x 00 Read value: 0x00

Write TX FIFO

MARCSTATE: 0

FREQOFF_EST: 0 kHz CRC OK

RSSI: 0 dB

DBW: 0 kHz Lock

SRES SXOFF SFSTXON SCAL SRX STX SIDLE

SmartRF® Studio Code Export

Export format

Template name: SimpliciTI settings Save

Make chip specific

Normal View summary Comment delimiters: /* */

Header: /*****
* SmartRF Studio(tm) Export
* Radio register settings specified with C-code
* compatible #define statements.
*****/

For each register: **#define SMARTRF_SETTING_@RN@ @<<<@ 0x@VH@**

Footer: #endif

Templates

Refresh list Delete

- C51 SFR definitions
- MSP430_Template
- Packet sniffer settings
- RF settings
- RF settings SoC
- RF settings struct typedet**
- SimpliciTI settings**

Copy to clipboard Write to file Filename: smartrf_CC@CHIPID@.h Close

Output Help

```
*****  
* SmartRF Studio(tm) Export  
* Radio register settings specified with C-code  
* compatible #define statements.  
*****  
  
#ifndef SMARTRF_CC2500_H  
#define SMARTRF_CC2500_H  
  
#define SMARTRF_RADIO_CC2500  
  
#define SMARTRF_SETTING_FSCTRL1 0x0C  
#define SMARTRF_SETTING_FSCTRL0 0x00  
#define SMARTRF_SETTING_FREQ2 0x5C  
#define SMARTRF_SETTING_FREQ1 0x4E  
#define SMARTRF_SETTING_FREQ0 0xC5  
#define SMARTRF_SETTING_MDMCFG4 0x0E  
#define SMARTRF_SETTING_MDMCFG3 0x3B  
#define SMARTRF_SETTING_MDMCFG2 0x73  
#define SMARTRF_SETTING_MDMCFG1 0x43  
#define SMARTRF_SETTING_MDMCFG0 0xA4  
#define SMARTRF_SETTING_CHANNR 0x2D  
#define SMARTRF_SETTING_DEVIATN 0x00  
#define SMARTRF_SETTING_FREND1 0xB6  
#define SMARTRF_SETTING_FREND0 0x10  
#define SMARTRF_SETTING_MCSM0 0x18  
#define SMARTRF_SETTING_FOCCFG 0x1D  
#define SMARTRF_SETTING_BSCFG 0x1C
```



Over-Writing the SimpliciTI™ Radio File

The image shows the SmartRF Studio Code Export dialog box. The 'Export format' section has 'SimpliciTI settings' selected. The 'Header' section contains the following text:

```
/*  
 * SmartRF Studio(tm) Export  
 * Radio register settings specified with C-code  
 * compatible #define statements.  
 */
```

The 'For each register' section has the text: `#define SMARTRF_SETTING_@RN@ @<<@ 0x@VH@`. The 'Footer' section has the text: `#endif`.

The 'Save As' dialog box is overlaid on the main dialog. It shows the 'Save in' location as 'CC2500'. The file list includes:

- mrfi_CC2500.c
- mrfi_CC2500_defs.h
- mrfi_CC2500_spi.c
- mrfi_CC2500_spi.h
- smartrf.srfs2500
- smartrf.srfs2510
- smartrf_CC2500.h
- smartrf_CC2510.prs

The 'File name' field is set to 'smartrf_CC2500.h' and the 'Save as type' is 'All Files (*.*)'. The 'Write to file' button in the main dialog is highlighted with a red box, and the 'Filename' field in the main dialog is also highlighted with a red box, showing the value 'smartrf_CC@CHIPID@.h'.

C:\eZ430-RF2500 Wireless Sensor Monitor IAR Source v1.02\Components\mrfi\radios\CC2500



SimpliciTI™ Radio File Updated

```
*****  
* SmartRF Studio(tm) Export  
*  
* Radio register settings specified with C-code  
* compatible #define statements.  
*  
*****  
  
#ifndef SMARTRF_CC2500_H  
#define SMARTRF_CC2500_H  
  
#define SMARTRF_RADIO_CC2500  
  
#define SMARTRF_SETTING_FSCTRLL1 0x0C  
#define SMARTRF_SETTING_FSCTRLO 0x00  
#define SMARTRF_SETTING_FREQ2 0x5C  
#define SMARTRF_SETTING_FREQ1 0x4E  
#define SMARTRF_SETTING_FREQ0 0xC5  
#define SMARTRF_SETTING_MDMCFG4 0x0E  
#define SMARTRF_SETTING_MDMCFG3 0x3B  
#define SMARTRF_SETTING_MDMCFG2 0x73  
#define SMARTRF_SETTING_MDMCFG1 0x43  
#define SMARTRF_SETTING_MDMCFG0 0x45  
#define SMARTRF_SETTING_CHANNR 0x2D  
#define SMARTRF_SETTING_DEVIATN 0x00  
#define SMARTRF_SETTING_FREND1 0xB6  
#define SMARTRF_SETTING_FREND0 0x10  
#define SMARTRF_SETTING_MCSM0 0x18  
#define SMARTRF_SETTING_FOCCFG 0x1D  
#define SMARTRF_SETTING_BSCFG 0x1C  
#define SMARTRF_SETTING_AGCCTRL2 0xC7  
#define SMARTRF_SETTING_AGCCTRL1 0x40  
#define SMARTRF_SETTING_AGCCTRL0 0xB0  
#define SMARTRF_SETTING_FSCAL3 0xEA  
#define SMARTRF_SETTING_FSCAL2 0x0A  
#define SMARTRF_SETTING_FSCAL1 0x00  
#define SMARTRF_SETTING_FSCAL0 0x19  
#define SMARTRF_SETTING_FSTEST 0x59  
#define SMARTRF_SETTING_TEST2 0x88  
#define SMARTRF_SETTING_TEST1 0x31  
#define SMARTRF_SETTING_TEST0 0x0B  
#define SMARTRF_SETTING_FIFOTH 0x07  
#define SMARTRF_SETTING_IOCFG2 0x29  
#define SMARTRF_SETTING_IOCFG0D 0x06  
#define SMARTRF_SETTING_PKTCTRL1 0x04  
#define SMARTRF_SETTING_PKTCTRL0 0x05  
#define SMARTRF_SETTING_ADDR 0x00  
#define SMARTRF_SETTING_PKTLEN 0xFF  
  
#endif
```

CHANNR – 2D = 45
(2*16 + 13)

IOCFG0D – Still needs to be
0x06 for SimpliciTI to work



Getting Started with SimpliciTI™ and the eZ430

Questions?

m-claassen@ti.com