

EZRadioPRO Wireless Product Software Development Board

Platform: PIC18F2520

**User Guide
Version 1.11**



Integration Associates, Inc.
110 Pioneer Way, Unit L
Mountain View, California 94041

Tel: 650.969.4100
Fax: 650.969.4582
www.integration.com
marketing@integration.com
wireless.support@integration.com

EZRadioPRO Wireless Product Software Development Board
Platform: PIC18F2520
Version 1.11
Revision Date: May 22, 2008

The information is provided "as is" without any express or implied warranty of any kind, including warranties of merchantability, non-infringement of intellectual property, or fitness for any particular purpose. In no event shall Integration Associates, Inc., or its suppliers be liable for any damages whatsoever arising out of the use of or an inability to use the materials. Integration Associates, Inc., and its suppliers further do not warrant the accuracy or completeness of the information, text, graphics, or other items contained within these materials. Integration Associates, Inc., may make changes to these materials, or to the products described within, at any time, without notice.

© 2008 Integration Associates, Inc. All rights reserved. Integration Associates is a trademark of Integration Associates, Inc. All trademarks belong to their respective owners.

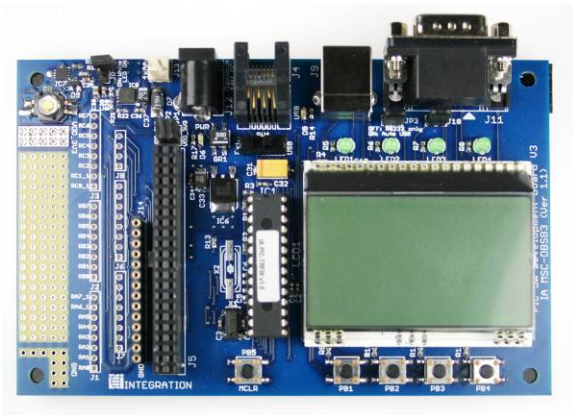
TABLE OF CONTENTS

General Introduction.....	4
System Introduction: Power Supply.....	5
System Introduction: IA MSC-DBSB3 ICD Connector.....	6
System Introduction: Schematic (IA MSC-DBSB3).....	7
Typical Testboard Schematic (IA443x Testcard).....	9
Using the SDB with a Standard Testcard.....	10
Demonstration Firmware.....	11
Setting up the USB Port for the PIC Development Board (IA MSC-DBSB3).....	14
Firmware 1.2r State Machine.....	15
IA MSC-DKSB3 Troubleshooting Guide.....	17
Appendix: EZRadioPRO Packet Error Rate Test Source Code (Firmware 1.2r).....	19

GENERAL INTRODUCTION

The IA Wireless Product Software Development Board, IA MSC-DBSB3, is designed to help engineers develop code specifically for Integration's EZRadio® and EZRadioPRO™ products when their application is developed on the Microchip PIC microcontroller platform.

The PIC Wireless Development Board (IA MSC-DBSB3) is designed specifically for code development and is designed a compliment tool to Integration's Loadboard with which exhaustive RF lab testing may be performed with the WDS environment. The Loadboard can be bought under the part number IA MSC-DKLB2 but also within the IA ISM-DK3 kit)



IA MSC-DBSB3

Software Development Board (SDB)



IA MSC-DBLB2

Testing Platform for controlled Lab Tests (Loadboard)

Both boards come with Integration's standard 40-pin socket for connecting the standard EZRadio® and EZRadioPRO™ evaluation testcards such as the IA4432-DKDB1. The IA MSC-DBSB3 software development board also comes with a DIP socket for various 28-pin PIC microcontrollers to be loaded into, the PIC provided by Integration is the PIC18F2520. The PIC18F2520 is preloaded with sample firmware to demonstrate a packet-based wireless link between two of these systems.

The IA MSC-DBSB3 PIC software development board includes:

- One 40-pin socket for EZRadio and EZRadioPRO testcards
- DIP socket for 28-pin PIC microcontrollers (16F and 18F series)
- PIC18F2520 preloaded with demonstration software
- Standard ICD connector for PIC programming and debugging
- 4 buttons and 4 LEDs for custom purposes
- LCD display for setup parameters and information display
- RS232 interface via a 9-pin DSUB male connector
- USB type B connector
- On board 3.3V PSU
- 5 x 19 through hole breadboard area for customer's application

Datasheets and application notes designed to support the Integration Associates Wireless chipsets can be found within the "Design Resources" section of the Integration website: <http://www.integration.com>.

SYSTEM INTRODUCTION: POWER SUPPLY

The board has three power options. The user can select between these options by the supply source selector switch (SW1).

On Board PSU

The on board PSU supplies 3.3VDC. In this mode, the board should be powered by a standard 9V AC or 9-12V DC adapter.

External PSU

In this mode, the board can be powered via the Direct DC supply connector by an external PSU. Any supply voltage can be used in the 3.3 - 4V range. Polarity is marked on the PCB.

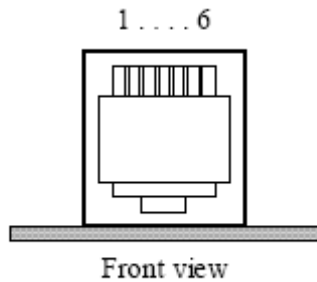
Powered by USB port

In this mode, the board can be powered via the USB connector.

(Note: when using the white LED Flash option it is recommend to use an alternative power supply)

SYSTEM INTRODUCTION: IA MSC-DBSB3 ICD CONNECTOR

ICD Connector (Emulator and Programmer Interface)

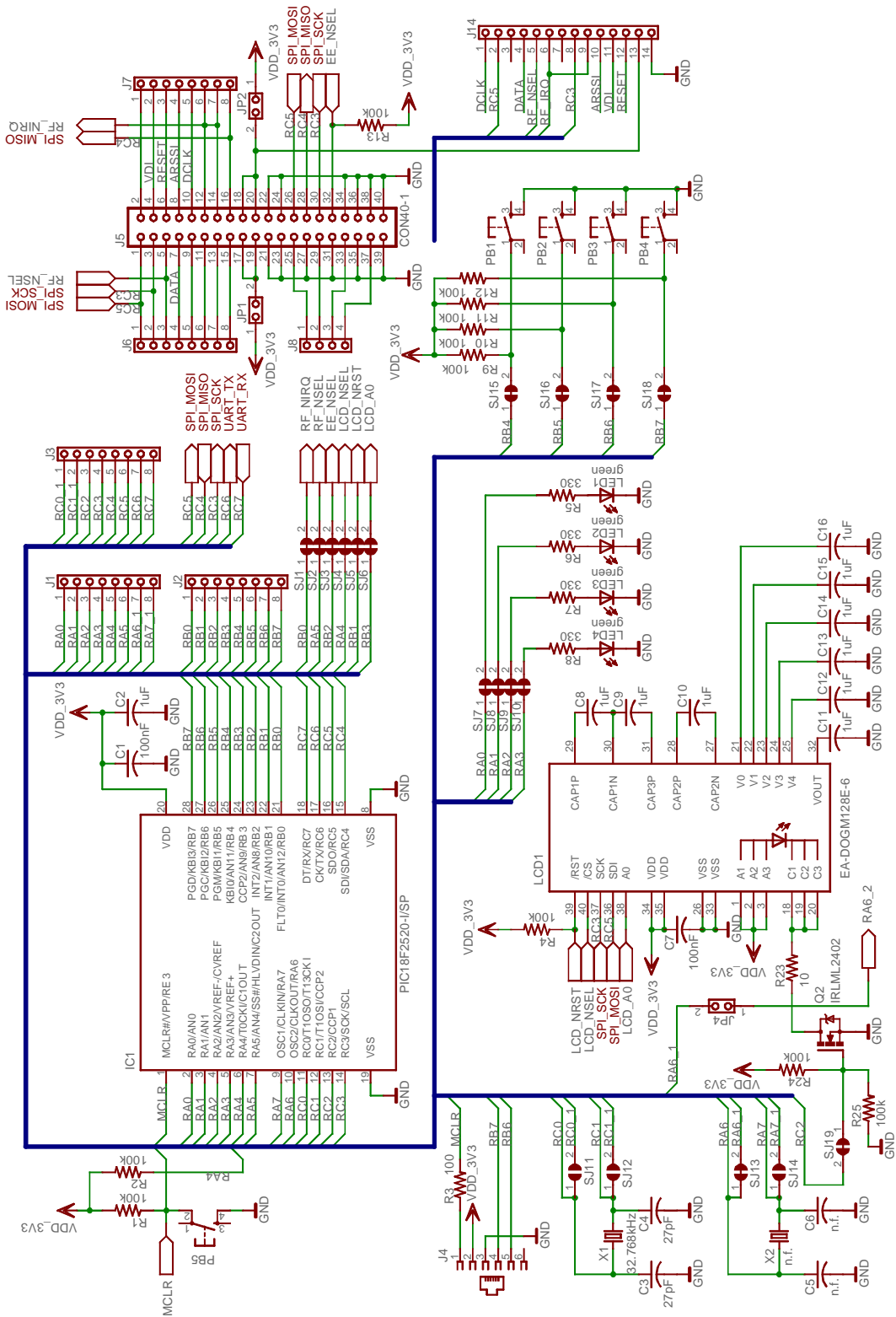


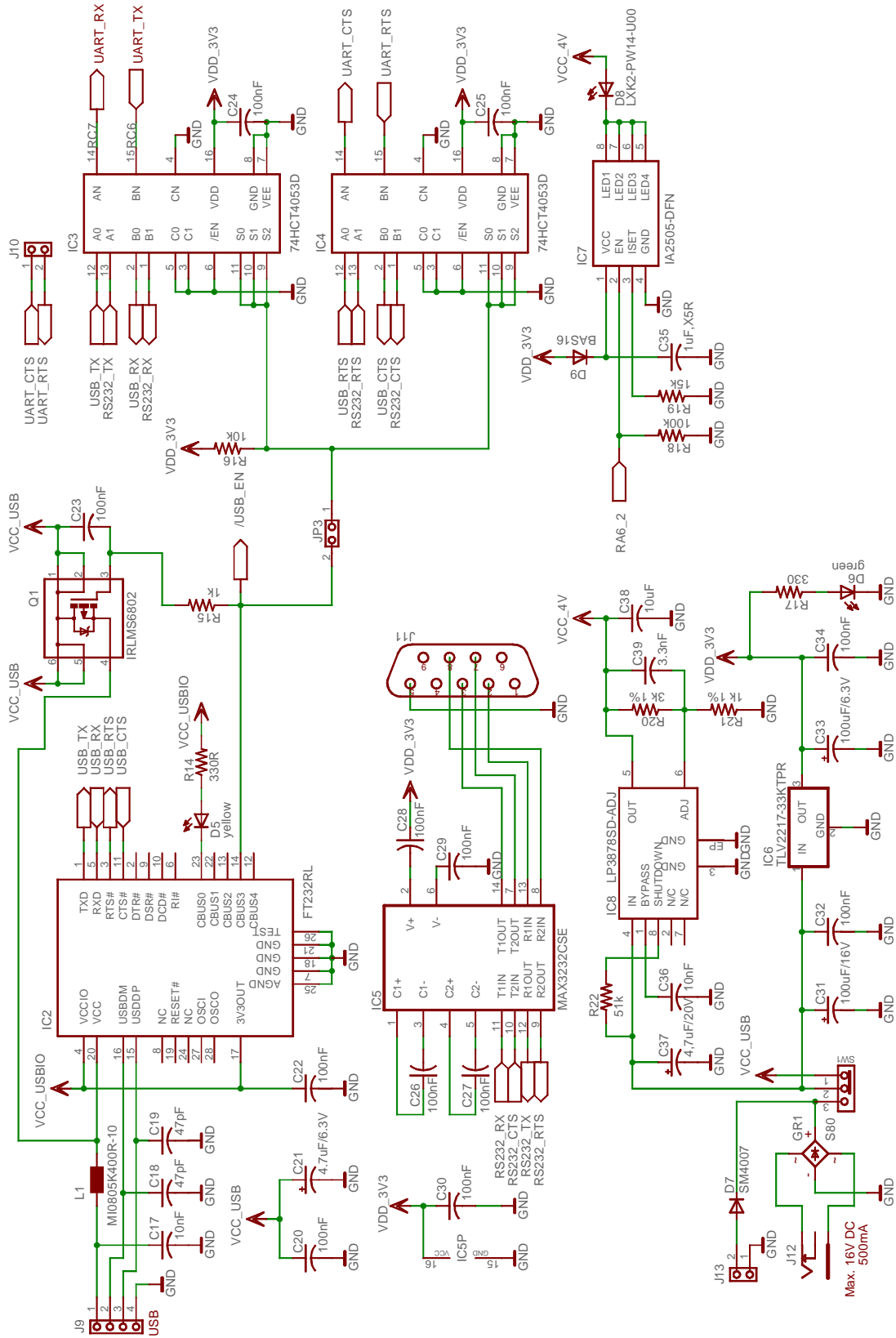
Pin Nr	Description
1	MCLR
2	VDD
3	GND
4	RB7
5	RB6
6	NC

40-Pin Evaluation Board Connector (J5)

PIN	Description	PIN	Description
1	J6/1 (SPI_MOSI)	2	J7/1
3	J6/2 (SPI_SCK)	4	J7/2(VDI)
5	J6/3 (RF_NSEL)	6	J7/3(RESET)
7	J6/4	8	J7/4
9	J6/5	10	J7/5
11	J6/6	12	J7/6
13	J6/7	14	J7/7
15	J6/8	16	J7/8(SPI_MISO)
17	VDD	18	VDD
19	VDD	20	VDD
21	GND	22	GND
23	GND	24	GND
25	J8/1	26	SPI_MOSI
27	GND	28	SPI_MISO
29	J8/2	30	SPI_SCK
31	GND	32	EE_NSEL
33	J8/3	34	GND
35	GND	36	GND
37	J8/4	38	GND
39	GND	40	GND

SYSTEM INTRODUCTION: SCHEMATIC (IA MSC-DBSB3)





USING THE SDB WITH A STANDARD TESTCARD

The standard EZRadio or EZRadioPRO testcards that are typically plugged into the IA MSC-DBLB2 Loadboard when engineers are performing RF tests on the radio ICs can also be plugged into the 40pin socket on the PIC software development board (SDB), as demonstrated below.

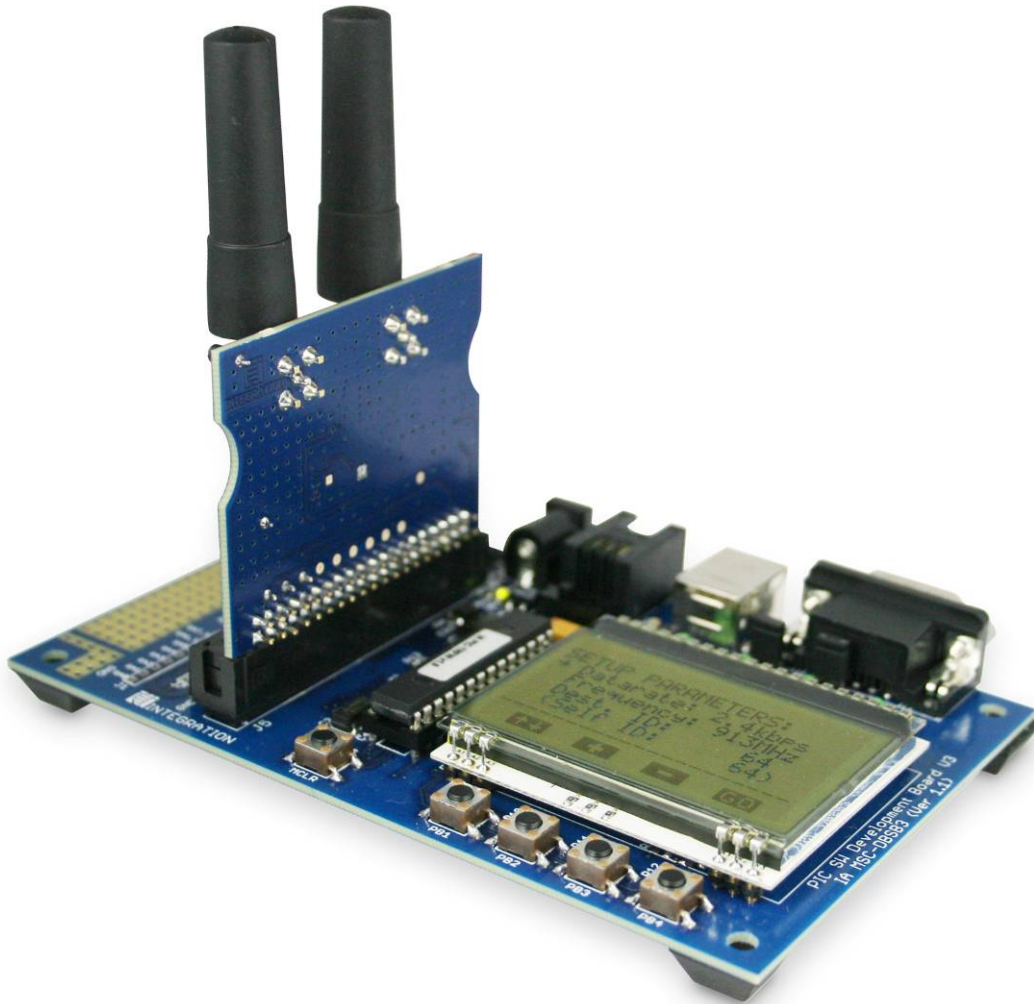


Figure 1: Software Development Board (IA MSC-DBSB3) with a standard IA I test card installed

DEMONSTRATION FIRMWARE

When shipped the IA MSC-DBSB3 comes with example firmware, this firmware is used to demonstrate the basic RF capabilities of a board under test. In the initial public release of this firmware (1.2r) only the EZRadioPRO IA4432 transceiver is supported, later releases are intended to demonstrate the ever increasing number of products from Integration.

Introducing Version 1.2r Firmware

Reference firmware v1.2r is designed to show the IA443x and a packet error rate test demonstration. This firmware is preloaded on to the PIC18F2520 microcontroller but can also be found on the WDS CDRoms in the SDB section. Source code to Integration's firmware should also be available in the same location.

The following screen shots reference firmware version 1.2r only, these screen shots may differ from the version you have received

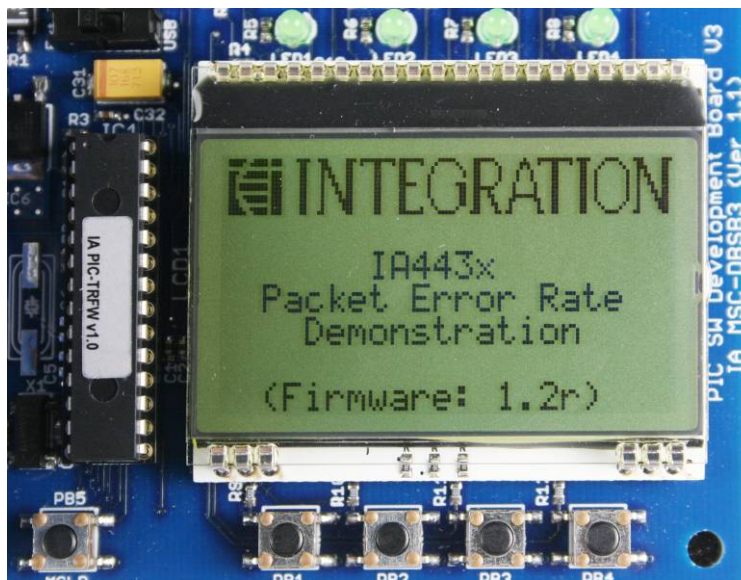


Figure 2: Reference Firmware Start Screen (ver 1.2r)

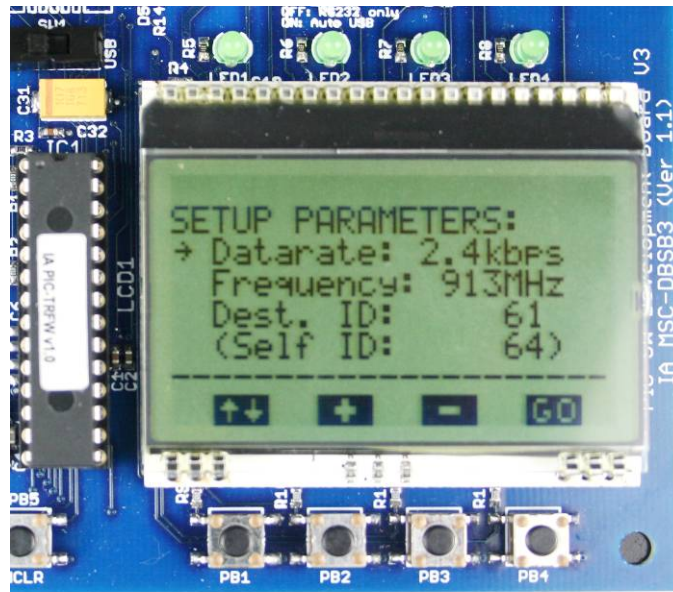


Figure 3: Set up screen

After 3 seconds delay (or immediately upon a button push), the radio parameters setup window will be displayed on the LCD.

Three types of parameters need to be setup as shown in Figure 3:

1. **Data rate**
2. **Frequency**
3. **Destination ID (should be the self ID of the 2nd board)**

To change the value of a parameter, push button 1 (labeled PB1), to move the arrow to the item what you want to change. In figure 3 it is shown currently highlighting the Data rate field. Using push buttons 2 & 3, highlighted "+" & "-" on the screen it is then possible to adjust that parameter accordingly. Once the parameters are adjusted as required simply press the "GO" button (PB4).

The LCD display will show following screen (Figure 4):

As firmware version 1.2r was only designed for limited field tests, antenna diversity was not supported; this is displayed on the screen by <NO-ANTDIV>. The feature is disabled in the firmware. A later firmware release will auto detect the capabilities of the testcard and in the event a testcard supports antenna diversity options to enable 1, 2 or both antennas will be available and displayed here accordingly.



Figure 4: Set up parameters displayed on LCD screen

To run the Packet Error Rate (PER) test as supplied in firmware v1.2r, two like systems are required since the test performs a 'PING-PONG' type transmission between two boards. To start the demo, push the 'TX ON' button on ONE of the boards, this board then transmits a ping (data) packet and waits for the pong (ACK) packets. The complete outbound and inbound response is used to calculate the packet error rate. In figure 5, it can be seen 24 packets were transmitted and 24 packets received thus providing a perfect 0.00% PER. In typical applications however it should be assumed engineers would 're-transmit' a packet where a packet is lost (or an ACK is not received).

For engineers wishing to visually see activity on both boards even when extended distances are shown, it is possible to fire the white LED upon packet receipt by placing a jumper on JP4 of the IA MSC-DBSB3.

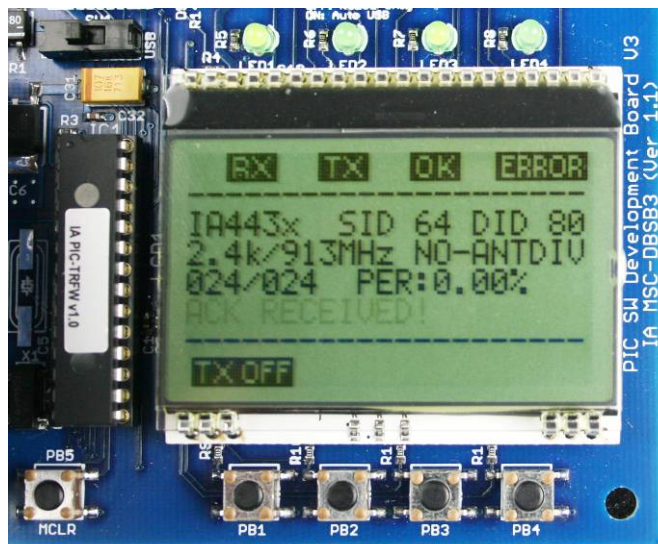


Figure 5: Packet error rate test underway

SETTING UP THE USB PORT FOR THE PIC DEVELOPMENT BOARD (IA MSC-DBSB3)

To configure the IA MSC-DBSB3 software development board to communicate with a PC via the USB port, a virtual serial port driver needs to be installed on the PC.

When the IA MSC-DBSB3 is connected, you will be prompted to install the Virtual COM port driver (FTD2xx).

This driver can be found on the WDS CDRROM or downloaded from the following WEB site:

Virtual COM port (VCP) Driver –

<http://www.ftdichip.com/Drivers/VCP.htm>

The Virtual COM port settings of the PIC development board are as follows:

- Data rate is 19.2kbps
- 1 stop bit
- No parity bit
- No handshake

If USB to virtual serial port driver is installed correctly, when the software development board is connected to PC by USB port and the WDS Terminal Emulator is running, test results like following can be seen (Figure 6):

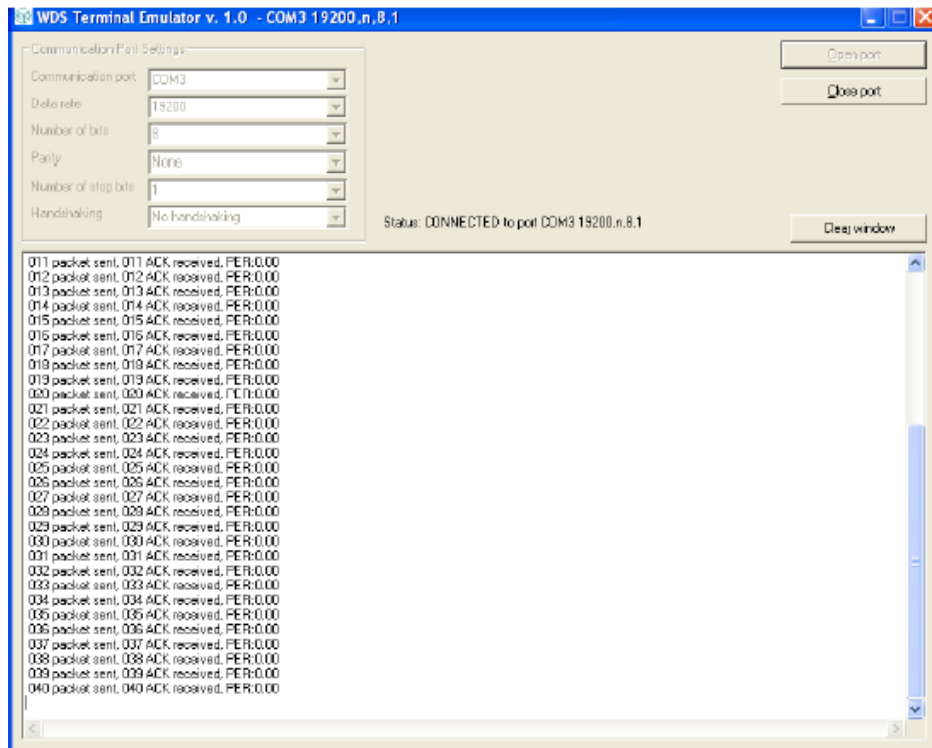


Figure 6: Test result displayed by USB virtual COM port

FIRMWARE 1.2R STATE MACHINE

The state machine for concluding the PER (Packet Error Rate) can be seen in figure 7 and 8. The state machine diagrams show the basic program setup in figure 7, and then in figure 8 shows how the packet error rate test procedure is calculated. Figure 9: shows the packet format that is used within the demonstration.

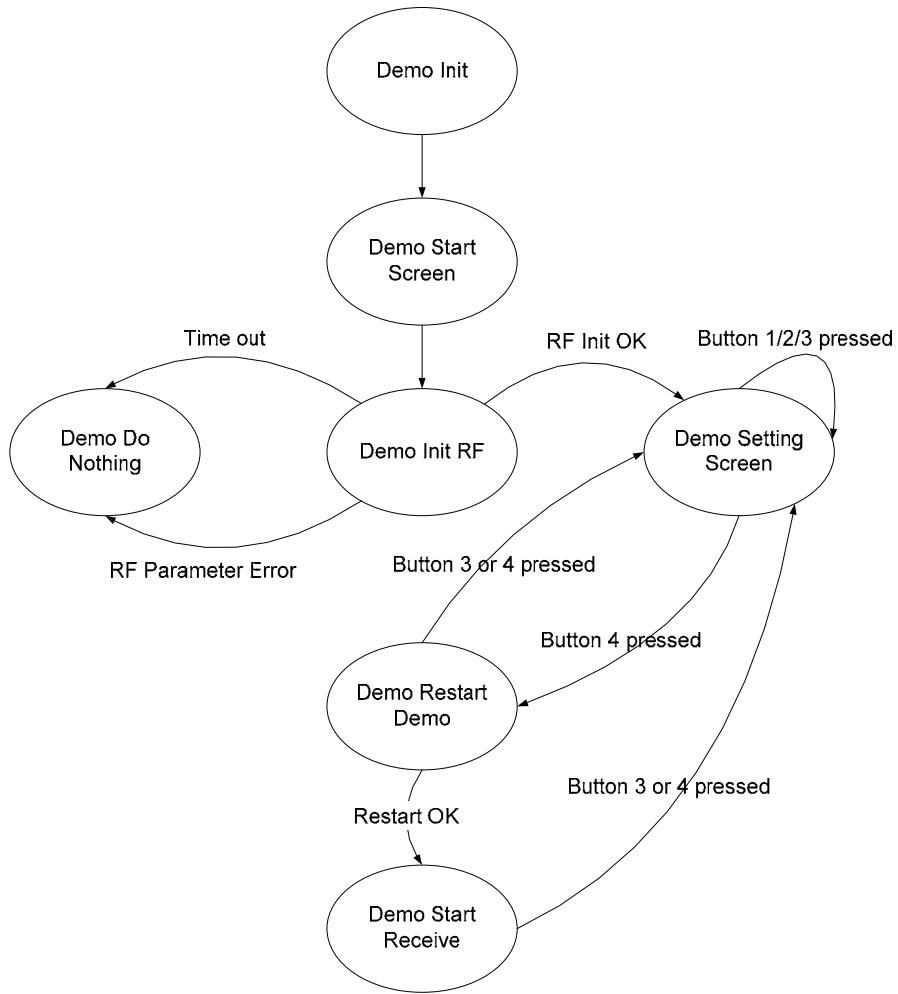


Figure 7: EZRadioPRO system initial state machines

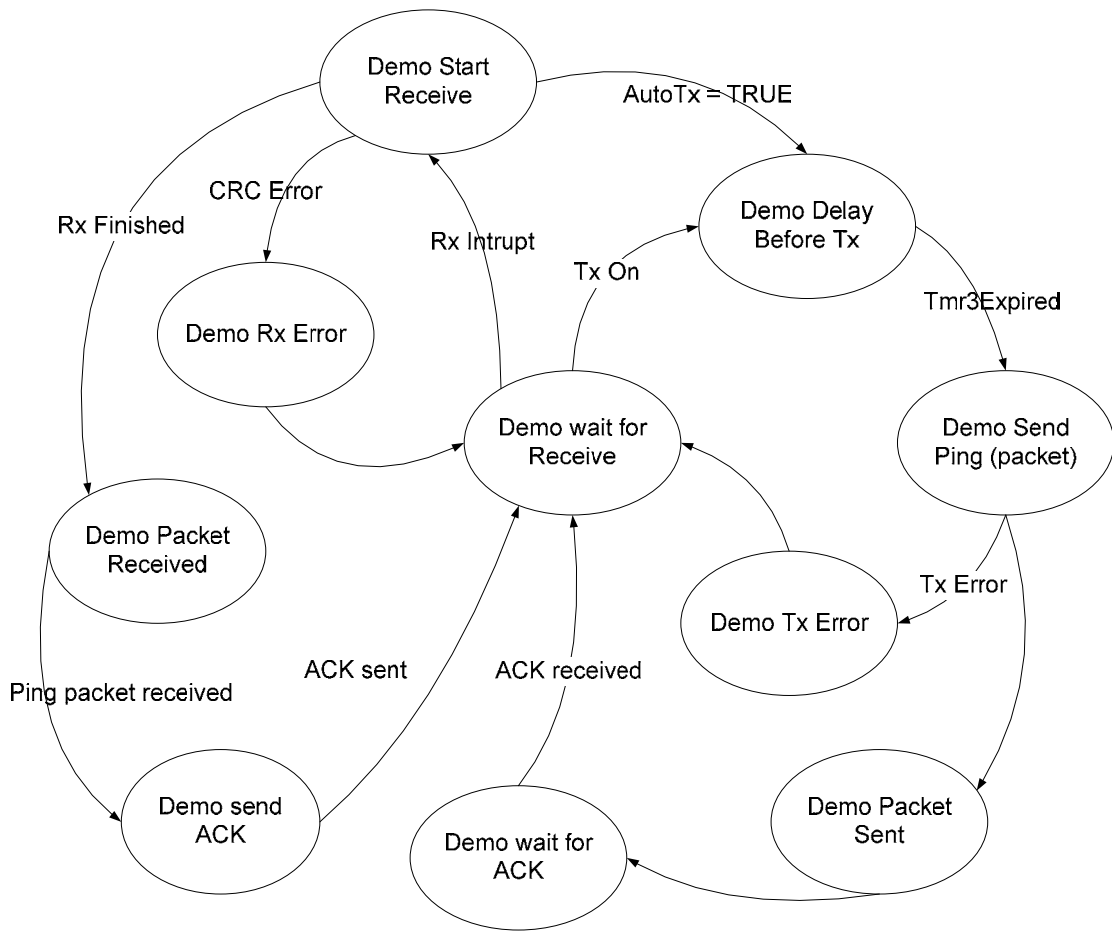


Figure 8: Packet Error Rate (PER) state machine

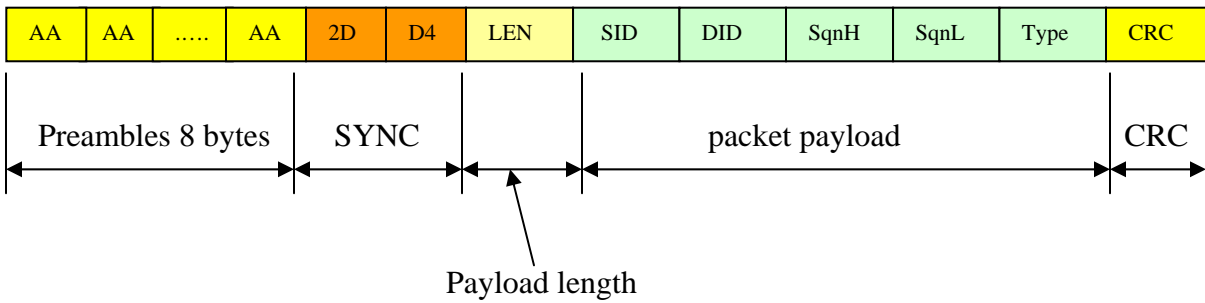


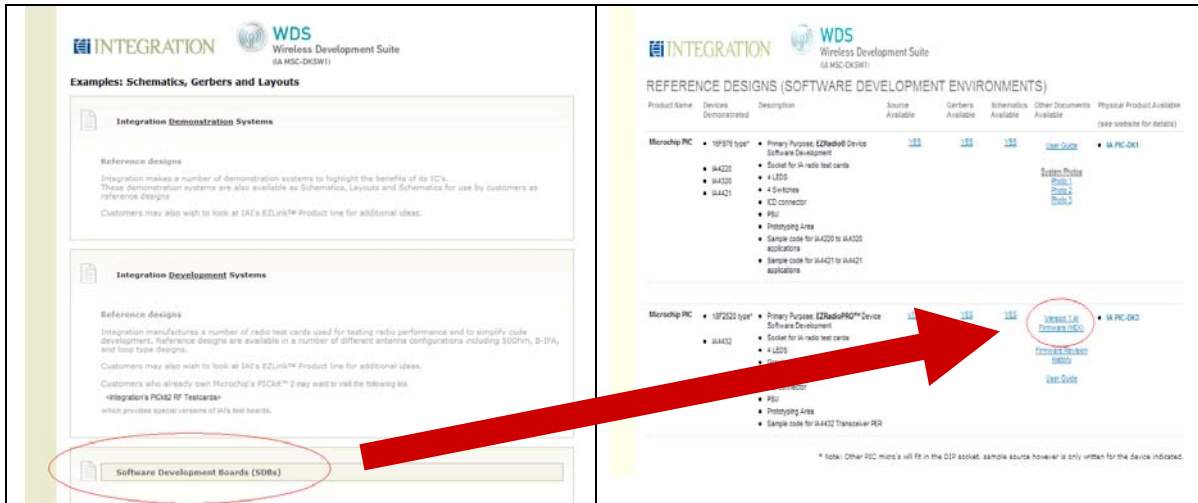
Figure 9: Packet format defined in the packet error rate test

IA MSC-DKSB3 TROUBLESHOOTING GUIDE

1. The LCD screen displays: **'ERROR: no Testcard or the board ident. EEPROM missing/empty!'** What does this mean and what do I have to do?

For future enhancements Integration has changed the content of the board identification EEPROMs found on the Testcards. The new EEPROM content is supported only by the firmware version 1.4r or greater.

Please update the firmware in the PIC SW Development board. The updated firmware can be found on the WDS CDROM under the SDB (Software Development Boards) section.



2. I'm using version 1.4r firmware and the board shows low battery even if it is plugged into the main power supply! Why is this and what can I do?

At the time of release, version 1.4r firmware supported EZRadioPRO transceiver and receiver silicon, the silicon at that time was considered pre-release, this silicon was version X2.

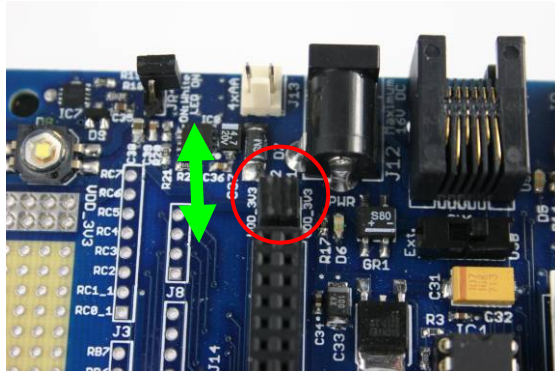
On X2 silicon a 'potential' bug has been found in the software to support the low battery detection (LBD) circuit. The LBD feature therefore has been disabled in the firmware rev 1.5r until the issue is fully investigated and resolved.

Please update the firmware to version 1.5r

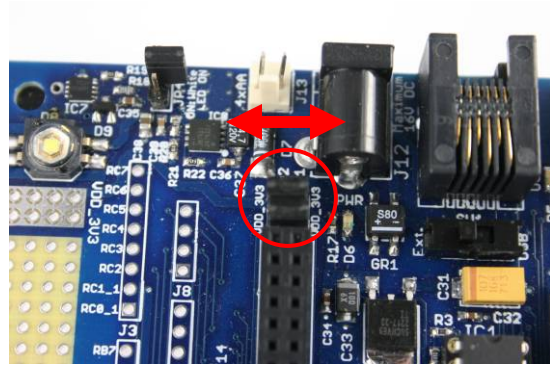
3. When using an EZRadioPRO testcard, I can setup the parameters (Data rate, frequency, etc.), but when I press the 'TX ON' button the TX LED illuminates but the system seems to lock-up. What can I do?

Check the orientation of the JP1 and JP2 jumpers, these jumpers have to be parallel with the 40-way connector. If the orientation is wrong, switch off the board, and adjust the jumpers before continuing.

If the system still does not work correctly, please contact to the Integration Support Team.



CORRECT orientation of Jumpers
JP1 & JP2



INCORRECT orientation of Jumpers
JP1 & JP2

4. When I press the TX ON button, the following error message appears on the LCD screen: 'PING TRANSMIT ERROR'. What can I do?

It shows hardware error, so please contact to the Integration Support Team!

However, this is usually caused by the interrupt from the testcard not being received by the microcontroller on the Software Development Board (SDB). If you have modified your testcard for experimentation purposes please ensure that if the test card is fitted with a pullup resistor on the NIRQ pin that its value lies between 100K and 1M.

APPENDIX: EZRADIOPRO PACKET ERROR RATE TEST SOURCE CODE (FIRMWARE 1.2R)

1. Main.c file

```
/*
** =====
**
** FILE
** $Id: main.c,v 1.0 2007/11/28 krk Exp apache $
**
** DESCRIPTION
** Main file of the IA443x demo
**
** CREATED
** Integration Hungary Ltd
**
** COPYRIGHT
** Copyright 2006 Integration Associates Inc. All rights reserved.
**
** Permission to use, copy, modify, revise, translate, abridge, condense, expand, collect,
** compile, link, recast, distribute, transform or adapt this software is granted only by
** specific written license from Integration Associates Inc. Contact Integration Associates
** at 110 Pioneer Way, Unit L, Mountain View, California 94041, Attention Sales & Marketing,
** or call (650) 969-4100 for information about obtaining a license.
**
** Any copy or modification made must satisfy the following conditions:
**
** 1. Both the copyright notice and this permission notice appear in all copies of the software,
** derivative works or modified versions, and any portions thereof, and that both notices
** appear in supporting documentation.
**
** 2. All copies of the software shall contain the following acknowledgement: "Portions of this
** software are used under license from Integration Associates Inc. and are copyrighted."
**
** 3. Neither the name of Integration Associates Inc. nor any of its subsidiaries may be used
** to endorse or promote products derived from this software without specific prior written
** permission.
**
** THIS SOFTWARE IS PROVIDED BY "AS IS" AND ALL WARRANTIES OF ANY KIND, INCLUDING THE IMPLIED
** WARRANTIES OF MERCHANTABILITY AND FITNESS FOR USE, ARE EXPRESSLY DISCLAIMED. THE DEVELOPER
** SHALL NOT BE LIABLE FOR ANY DAMAGES WHATSOEVER RESULTING FROM THE USE OF THIS SOFTWARE.
** THIS SOFTWARE MAY NOT BE USED IN PRODUCTS INTENDED FOR USE IN IMPLANTATION OR OTHER DIRECT
** LIFE SUPPORT APPLICATIONS WHERE MALFUNCTION MAY RESULT IN THE DIRECT PHYSICAL HARM OR INJURY
** TO PERSONS. ALL SUCH IS USE IS EXPRESSLY PROHIBITED.
**
** =====
**/
```

```

/*-----*/
/*-----*/
/*                                     INCLUDE                                     */
/*-----*/
/*-----*/
#include "PIC.h"
#include "IA443x_rf.h"
#include "uart.h"
#include "pic18_timers.h"
#include <stdio.h>
#include <string.h>
#include "dog_glcd.h"
#include "IA443x_demo.h"
#ifdef TEST_ENABLED
    #include "IA443x_test.h"
#endif

/*-----*/
/*-----*/
/*                                     GLOBAL variables                                     */
/*-----*/
/*-----*/
#ifdef TEST_ENABLED
    volatile S_TEST sDemo;
#endif
extern volatile uint8 fUartCommandReceived;
extern uint16 SpiCommand;

/*-----*/
/*-----*/
/*                                     FUNCTION prototypes                                     */
/*-----*/
/*-----*/
void _InitHw(void);

/*-----*/
/*-----*/
/*                                     MAIN routine                                     */
/*-----*/
/*-----*/
void main(void)
{
    //initialize the HW
    _InitHw();
    //initialize the demo
    DemoInit();
    //start the 32.768kHz system timer
    StartSystemTimer();

    //main loop
    while(1)
    {

#ifdef TEST_ENABLED
        //run the demo state machine
        switch( sDemo )
        {
            case DEMO:
                DemoStateMachine();
                break;

            case PER:
                TestPerStateMachine();
                break;

            case BER:
                TestBerStateMachine();
                break;
            case CW:

```

```

        TestCwStateMachine( MOD_TYPE_CW );
        break;

        case PN9:
            TestCwStateMachine( MOD_PN9 );
            break;

        default:
            break;
    }
#else
    DemoStateMachine();
#endif

    //if uart command received, handle it
    if(fUartCommandReceived == TRUE)
    {
        fUartCommandReceived = FALSE;
        UartCommandInterpreter();
    }
}

/*-----*/
/*          FUNCTIONS used in the main.c          */
/*-----*/
void _InitHw(void)
{
    //set 8MHz internal oscillator
    Set8MHzInternalRc();
    while( InternalRcInstable() );
    //disable all nonused peripherals
    //Capcture compare module: OFF
    DisableCCPModules();
    //Comparator: OFF
    DisableComparator();
    //ADC: OFF, all pins are I/O
    DisableAdc();

    //initialize I/O port directions
    LED1_DIR      = OUTPUT;
    LED2_DIR      = OUTPUT;
    LED3_DIR      = OUTPUT;
    LED4_DIR      = OUTPUT;
    BLED_DIR      = OUTPUT;
    PB1_DIR       = INPUT;
    PB2_DIR       = INPUT;
    PB3_DIR       = INPUT;
    PB4_DIR       = INPUT;
    RF_NIRQ_DIR   = INPUT;
    RF_NSEL_DIR   = OUTPUT;
    EE_NSEL_DIR   = OUTPUT;
    LCD_RESET_DIR = OUTPUT;
    LCD_NSEL_DIR  = OUTPUT;
    LCD_A0_DIR    = OUTPUT;

    //default I/O port
    LED1_PIN      = 0;
    LED2_PIN      = 0;
    LED3_PIN      = 0;
    LED4_PIN      = 0;
    BLED_PIN      = 0;
    RF_NSEL_PIN   = 1;
    EE_NSEL_PIN   = 1;
    LCD_NSEL_PIN  = 1;
    LCD_A0_PIN    = 0;
    LCD_RESET_PIN = 1;
    //initialize SPI port
    SetHwMasterSpi();
}

```

```

//initialize the UART peripherals
UartInit();

//initialize the LCD
LcdInit();

//enable interrupts
EnablePeripheralIt();
DisablePbIt();
EnableGlobalIt();
}

```

2. IA443x_demo.c file

```

/*
** =====
**
** FILE
** $Id: IA443x_demo.c,v 1.0 2007/11/29 krk Exp apache $
**
** DESCRIPTION
** Contains the RF dependent functions
**
** CREATED
** Integration Hungary Ltd
**
** COPYRIGHT
** Copyright 2006 Integration Associates Inc. All rights reserved.
**
** Permission to use, copy, modify, revise, translate, abridge, condense, expand, collect,
** compile, link, recast, distribute, transform or adapt this software is granted only by
** specific written license from Integration Associates Inc. Contact Integration Associates
** at 110 Pioneer Way, Unit L, Mountain View, California 94041, Attention Sales & Marketing,
** or call (650) 969-4100 for information about obtaining a license.
**
** Any copy or modification made must satisfy the following conditions:
**
** 1. Both the copyright notice and this permission notice appear in all copies of the software,
** derivative works or modified versions, and any portions thereof, and that both notices
** appear in supporting documentation.
**
** 2. All copies of the software shall contain the following acknowledgement: "Portions of this
** software are used under license from Integration Associates Inc. and are copyrighted."
**
** 3. Neither the name of Integration Associates Inc. nor any of its subsidiaries may be used
** to endorse or promote products derived from this software without specific prior written
** permission.
**
** THIS SOFTWARE IS PROVIDED BY "AS IS" AND ALL WARRANTIES OF ANY KIND, INCLUDING THE IMPLIED
** WARRANTIES OF MERCHANTABILITY AND FITNESS FOR USE, ARE EXPRESSLY DISCLAIMED. THE DEVELOPER
** SHALL NOT BE LIABLE FOR ANY DAMAGES WHATSOEVER RESULTING FROM THE USE OF THIS SOFTWARE.
** THIS SOFTWARE MAY NOT BE USED IN PRODUCTS INTENDED FOR USE IN IMPLANTATION OR OTHER DIRECT
** LIFE SUPPORT APPLICATIONS WHERE MALFUNCTION MAY RESULT IN THE DIRECT PHYSICAL HARM OR INJURY
** TO PERSONS. ALL SUCH IS USE IS EXPRESSLY PROHIBITED.
**
** =====
*/
#include <stdio.h>
#include <string.h>
#include "IA443x_demo.h"
#include "IA443x_rf.h"
#include "IA443x_rf_callback.h"
#include "dog_glcd.h"
#include "uart.h"

/*-----*/
/* GLOBAL variables */

```

```

/*-----*/
-----*/
volatile DEMO_STATES DemoStates;
PACKET_TYPE fWaitForPacketOrAck, fPacketSent;
uint8 lcd_data[22], lcd[22], rf_param[10], AutoTx, LongDelay;
uint16 NbrOfSentPackets, NbrOfReceivedPackets;

PHY_STATUS RfStatus;
uint8 RxTxBuffer[MAX_LENGTH_OF_PAYLOAD];
HEADER HeaderBuffer;
MESSAGE MessageBuffer;
RECEIVE_COMMAND RxCommand;
RECEIVED_DATA ReceivedData;
volatile DEMO_PARAMETERS demo_param;
volatile uint8 customer_name[CUSTOMER_NAME_SIZE];
uint8 ParamToChange, ButtonPressed;

const volatile DEMO_FREQ demo_freqs[MAX_FREQ_SETTING] =
{
    {0x75, 0xA280, 913},
    {0x75, 0xD480, 917},
};

const uint8 FirmwareVersion[] = "1.2r";

extern volatile RF_CONFIG chip_settings;
//pictures
extern const uint8 ok_inverted16x8[];
extern const uint8 rx_inverted16x8[];
extern const uint8 tx_inverted16x8[];
extern const uint8 txon_inverted29x8[];
extern const uint8 iai_logol28x16[];
extern const uint8 txon_inverted32x8[];
extern const uint8 txoff_inverted32x8[];
extern const uint8 setting_inverted48x8[];
extern const uint8 minus_inverted16x8[];
extern const uint8 plus_inverted16x8[];
extern const uint8 up_down_inverted16x8[];
extern const uint8 go_inverted16x8[];

/*-----*/
-----*/
/*
LOCAL function prototypes
*/
/*-----*/
-----*/
void DemoSendPing(void);
void DemoPingSent(uint8 success, uint16 pid);
void DemoAckSent(uint8 success);
void DemoPingReceived(uint16 pid, uint8 rssi_level);
void DemoAckReceived(uint8 success, uint16 pid, uint8 rssi_level);
void GetDataRate(RF_SAMPLE_SETTINGS datarate, uint8 * string);
void DemoSimplePacketReceived(uint8 rssi_level, uint8 length);

/*-----*/
-----*/
/*
LOCAL definitions
*/
/*-----*/
-----*/
#define PER_RESET_INPUT PB2_PIN
#define SEND_PACKET_INPUT PB1_PIN
#define LED_RX LED1_PIN
#define LED_TX LED2_PIN
#define LED_OK LED3_PIN
#define LED_ERROR LED4_PIN

/*+++++
+
+ FUNCTION NAME: void DemoInit(void)
+
+ DESCRIPTION: initializes the demo
*/

```

```

+
+ RETURN:          None
+
+ NOTES:
+
+******/
void DemoInit(void)
{
    //initialize the demo variables
    DemoStates = sDemoInit;
}

/*+*****+
+
+ FUNCTION NAME:   void DemoStateMachine(void)
+
+ DESCRIPTION:    state machine of the demo
+
+ RETURN:         None
+
+ NOTES:
+
+******/
void DemoStateMachine(void)
{
    uint8 temp8,i,string[6];
    uint16 temp16, delay;
    static uint8 demo_mode;

    switch( DemoStates )
    {
        case sDemoInit:
            //set default variables
            demo_param.datarate = DR10KBPS_DEV5KHZ_MOD1;
            DemoStates = sDemoStartScreen;
            break;

        case sDemoStartScreen:
            //clear LCD
            LcdClearDisplay();
            //draw startup screen
            LcdSetPictureCursor(LCD_LINE_1,1);
            LcdDrawPicture(iai_logol28x16);
            memcpy(lcd_data,"    IA443x          ",21);
            LcdWriteLine(LCD_LINE_4,lcd_data);
            memcpy(lcd_data," Packet Error Rate ",21);
            LcdWriteLine(LCD_LINE_5,lcd_data);
            memcpy(lcd_data," Demonstration      ",21);
            LcdWriteLine(LCD_LINE_6,lcd_data);
            sprintf(lcd_data," (Firmware: %s)      ",FirmwareVersion);
            LcdWriteLine(LCD_LINE_8,lcd_data);
            printf("\r\n\r\nINTEGRATION, IA443x Packet Error Rate Demonstration\r\n");
            printf("Firmware: %s\r\n",FirmwareVersion);

            //start long timeout -> 3s
            LongDelay = 24;
            StartTmr3(TMR3_INTERNAL_ENABLE, TMR3_8, 62500, FALSE);
            DemoStates = sDemoDelayAfterScreens;
            break;

        case sDemoDelayAfterScreens:
            if( Tmr3Expired() == TRUE )
            { //TMR expired
                StopTmr3();
                if( LongDelay > 0 )
                { //restart if needed for another 250ms
                    StartTmr3(TMR3_INTERNAL_ENABLE, TMR3_8, 62500, FALSE);
                    LongDelay--;
                }
            }
            else
            {

```



```

        DemoStates = sDemoInitRf;
    }
}
if( (PB1_PIN == 0) || (PB2_PIN == 0) || (PB3_PIN == 0) || (PB4_PIN == 0) )
{
while( (PB1_PIN == 0) || (PB2_PIN == 0) || (PB3_PIN == 0) || (PB4_PIN == 0) );
DemoStates = sDemoInitRf;
}
break;

case sDemoInitRf:
//clear LCD
LcdClearDisplay();
//initialize the RF stack
temp8 = RfInitHw();
switch( temp8 )
{
    case RF_OK:
        //initialization is OK
        break;

    case RF_ERROR_TIMING:
        //POR ERROR
        //clear LCD
        LcdClearDisplay();
printf("\r\nERROR: POR timeout! Insert a Testcard or try a
        different one!\r\n");
        memcpy(lcd_data,"ERROR: POR timeout! ",21);
        LcdWriteLine(LCD_LINE_2,lcd_data);
        memcpy(lcd_data,"Insert a Testcard or ",21);
        LcdWriteLine(LCD_LINE_5,lcd_data);
        memcpy(lcd_data,"try a different one! ",21);
        LcdWriteLine(LCD_LINE_6,lcd_data);
        DemoStates = sDemoDoNothing;
        return;
        break;

    case RF_ERROR_PARAMETER:
        //clear LCD
        LcdClearDisplay();
        //EEPROM is empty
printf("\r\nERROR: the board identification EEPROM is missing or empty!");
        printf("\r\nTry a different Testcard!\r\n");
        memcpy(lcd_data,"ERROR: board ident. ",21);
        LcdWriteLine(LCD_LINE_2,lcd_data);
        memcpy(lcd_data,"EEPROM missing/empty!",21);
        LcdWriteLine(LCD_LINE_3,lcd_data);
        memcpy(lcd_data," Try a different ",21);
        LcdWriteLine(LCD_LINE_5,lcd_data);
        memcpy(lcd_data," Testcard! ",21);
        LcdWriteLine(LCD_LINE_6,lcd_data);
        DemoStates = sDemoDoNothing;
        return;
        break;
}

//get default settings
//addresses
demo_param.self_address = chip_settings.name.board_number;
if( EEPROM_READ(EE_ADDRESS_FIRST_PLUG) == 0 )
{
demo_param.destination_address = chip_settings.name.board_number;
}
else
{
demo_param.destination_address = EEPROM_READ(EE_ADDRESS_DID);
}
//frequency
demo_param.freq_number = EPROM_READ(EE_ADDRESS_DEFAULT_FREQ);
//data rate
demo_param.datarate = EEPROM_READ(EE_ADDRESS_DEFAULT_BR);

```

```

//get customer name
for(i=0;i<CUSTOMER_NAME_SIZE;i++)
{
customer_name[i] = EEPROM_READ(i);
}
//set variables
ParamToChange = PARAM_DR;
ButtonPressed = BUTTON_NON;
//next state
DemoStates = sDemoSettingScreen;
break;

case sDemoDoNothing:
break;

case sDemoSettingScreen:
//stop RF stack and timer
RFIdle();
StopTmr3();
//clear LCD
LcdClearDisplay();
//DRAW welcome-setting screen
memcpy(lcd_data, "                ",21);
sprintf(lcd_data,"%s",customer_name);
LcdWriteLine(LCD_LINE_1,lcd_data);
//setting page
memcpy(lcd_data,"SETUP PARAMETERS: ",21);
LcdWriteLine(LCD_LINE_2,lcd_data);
//check whether the parameter select pushbutton is pressed or not
if( ButtonPressed == BUTTON_1 )
{
    if( ParamToChange < PARAM_DID )
    {
        ParamToChange++;
    }
    else
    {
        ParamToChange = PARAM_DR;
    }
}
//draw push button labels
LcdSetPictureCursor(8,13);
LcdDrawPicture(up_down_inverted16x8);
LcdSetPictureCursor(8,43);
LcdDrawPicture(plus_inverted16x8);
LcdSetPictureCursor(8,73);
LcdDrawPicture(minus_inverted16x8);
LcdSetPictureCursor(8,103);
LcdDrawPicture(go_inverted16x8);

//check whether the parameter has to be changed or not
switch( ParamToChange)
{
    case PARAM_DR:
        //check PBs
        if( ButtonPressed == BUTTON_2 )
        {
            //inc
            //change data rate setting
            if(demo_param.datarate< NMBR_OF_SAMPLE_SETTING - 1 )
            {
                demo_param.datarate++;
            }
            else
            {
                demo_param.datarate = 0;
            }
        }
        if( ButtonPressed == BUTTON_3 )
        {
            //dec
            //change data rate setting

```

```

        if( demo_param.datarate == 0 )
        {
            demo_param.datarate=NMBR_OF_SAMPLE_SETTING - 1;
        }
        else
        {
            demo_param.datarate--;
        }
    }
    break;

case PARAM_FREQ:
    //check PBs
    if( ButtonPressed == BUTTON_2 )
    {
        //inc
        if( demo_param.freq_number < (MAX_FREQ_SETTING - 1) )
        {
            demo_param.freq_number++;
        }
        else
        {
            demo_param.freq_number = 0;
        }
    }
    if( ButtonPressed == BUTTON_3 )
    {
        //dec
        if( demo_param.freq_number == 0 )
        {
            demo_param.freq_number = MAX_FREQ_SETTING - 1;
        }
        else
        {
            demo_param.freq_number--;
        }
    }
    break;

case PARAM_DID:
    //check PBs
    if( ButtonPressed == BUTTON_2 )
    {
        //inc
        if( demo_param.destination_address < 255 )
        {
            demo_param.destination_address++;
        }
        else
        {
            demo_param.destination_address = 0;
        }
    }
    if( ButtonPressed == BUTTON_3 )
    {
        //dec
        if( demo_param.destination_address == 0 )
        {
            demo_param.destination_address = 255;
        }
        else
        {
            demo_param.destination_address--;
        }
    }
    break;
}

GetDataRate( demo_param.datarate, &string[0]);
string[3] = 0;
sprintf(lcd_data," Datarate: %skbps ",string);
LcdWriteLine(LCD_LINE_3,lcd_data);
sprintf(lcd_data," Frequency:
%3uMHz",demo_freqs[demo_param.freq_number].freq_decimal);
LcdWriteLine(LCD_LINE_4,lcd_data);

```

```

sprintf(lcd_data," Dest. ID: %5u  ",demo_param.destination_address);
LcdWriteLine(LCD_LINE_5,lcd_data);
sprintf(lcd_data," (Self ID: %5u)  ",demo_param.self_address);
LcdWriteLine(LCD_LINE_6,lcd_data);
memcpy(lcd_data,"-----",21);
LcdWriteLine(LCD_LINE_7,lcd_data);

//draw the arrow
switch( ParamToChange)
{
    case PARAM_DR:          LcdSetCharCursor(3,1); break;
    case PARAM_FREQ:       LcdSetCharCursor(4,1); break;
    case PARAM_DID:       LcdSetCharCursor(5,1); break;
    break;
}
LcdPutCh(''+1);

DemoStates = sDemoCheckPb;
break;

case sDemoCheckPb:
    ButtonPressed = BUTTON_NON;
    if( PB1_PIN == 0 )
    {
        //wait for release the button
        while( PB1_PIN == 0);
        ButtonPressed = BUTTON_1;
        DemoStates = sDemoSettingScreen;
    }

    if( PB2_PIN == 0 )
    {
        //wait for release the button
        while( PB2_PIN == 0);
        ButtonPressed = BUTTON_2;
        DemoStates = sDemoSettingScreen;
    }

    if( PB3_PIN == 0 )
    {
        //wait for release the button
        while( PB3_PIN == 0);
        ButtonPressed = BUTTON_3;
        DemoStates = sDemoSettingScreen;
    }
    //run demo
    if( PB4_PIN == 0 )
    {
        //wait for release the button
        while( PB4_PIN == 0);
        ButtonPressed = BUTTON_4;
        DemoStates = sDemoRestartDemo;
    }
break;

case sDemoRestartDemo:
    //clear LCD
    LcdClearDisplay();
    //put info to the LCD
    LcdSetPictureCursor(1,13);
    LcdDrawPicture(rx_inverted16x8);
    LcdSetPictureCursor(1,42);
    LcdDrawPicture(tx_inverted16x8);
    LcdSetPictureCursor(1,71);
    LcdDrawPicture(ok_inverted16x8);
    LcdSetPictureCursor(1,98);
    LcdDrawPicture(txon_inverted29x8);

    memcpy(lcd_data,"-----",21);
    LcdWriteLine(LCD_LINE_2,lcd_data);
    LcdWriteLine(LCD_LINE_7,lcd_data);

```

```

LcdSetPictureCursor(8,3);
LcdDrawPicture(txon_inverted32x8);
LcdSetPictureCursor(8,73);
LcdDrawPicture(setting_inverted48x8);

//print chip, freq, data rate
sprintf(lcd_data,"IA443x SID%3u DID%3u",

demo_param.self_address,demo_param.destination_address);
LcdWriteLine(LCD_LINE_3,lcd_data);
printf("%s\r\n",lcd_data);
GetDataRate( demo_param.datarate, &string[0]);
string[3] = 0;
sprintf(lcd_data,"%sk/%uMHz NO-ANTDIV",

string,demo_freqs[demo_param.freq_number].freq_decimal);
LcdWriteLine(LCD_LINE_4,lcd_data);
//save basic settings
EEPROM_WRITE(EE_ADDRESS_FIRST_PLUG, 1);
EEPROM_WRITE(EE_ADDRESS_DEFAULT_BR, demo_param.datarate);
EEPROM_WRITE(EE_ADDRESS_DEFAULT_FREQ, demo_param.freq_number);
EEPROM_WRITE(EE_ADDRESS_DID, demo_param.destination_address);

/*****
*/
/*initialize RF stack for the demo*/
/*****
*/
SetRadio( demo_param.datarate );
//clear flags
NmbrOfSentPackets = 0;
NmbrOfReceivedPackets = 0;
AutoTx = FALSE;
LED_TX = 0;
LED_RX = 0;
BLED_PIN = 0;
LED_OK = 0;
LED_ERROR = 0;
//next state
DemoStates = sDemoStartReceive;
break;

/*PER TEST DEMO --> Demo purposes*/
case sDemoStartPerTest:
break;

case sDemoStartReceive:
//reset the RF stack
RFIdle();

//set variables
if( NmbrOfSentPackets == 999 )
{
NmbrOfSentPackets = 1;
NmbrOfReceivedPackets = 0;
}
else
{
NmbrOfSentPackets++;
}
//set flags -> wait for PING packet and not for ACK
fWaitForPacketOrAck = PING;

//if auto tx on -> check PB / send next packet
if( AutoTx == TRUE )
{
LongDelay = 1;
StartTmr3(TMR3_INTERNAL_ENABLE, TMR3_8, 62500, FALSE);
DemoStates = sDemoDelayBeforeTx;
return;
}

```

```

        //start receive
        RFReceive(&RxCommand);
        //next state
        DemoStates = sDemoWaitForReceive;
break;

case sDemoDelayBeforeTx:
    //check the push button
    if(SEND_PACKET_INPUT == 0)
    {
        //wait for release the button
        while(SEND_PACKET_INPUT == 0);
        AutoTx = FALSE;
        DemoStates = sDemoRestartDemo;
        return;
    }
    if( PB3_PIN == 0 || PB4_PIN == 0 )
    {
        while( PB3_PIN == 0 || PB4_PIN == 0 );
        //goto setting page
        DemoStates = sDemoSettingScreen;
        return;
    }
    if( Tmr3Expired() == TRUE )
    { //TMR expired
        StopTmr3();
        if( LongDelay > 0 )
        { //restart if needed for another 250ms
            StartTmr3(TMR3_INTERNAL_ENABLE, TMR3_8, 62500, FALSE);
            LongDelay--;
            return;
        }
        else
        {
            DemoStates = sDemoSendPing;
            return;
        }
    }
}
break;

case sDemoWaitForReceive:
    //check whether the push button is pressed for sending a PING packet
    if( (SEND_PACKET_INPUT == 0) && (fWaitForPacketOrAck == PING) &&
        (AutoTx == FALSE) )
    { //push button is pressed -> stop receive and send a PING packet
        //stop receiving
        RFIdle();
        //wait for release TX push button
        while(SEND_PACKET_INPUT == 0);
        //switch on auto tx
        AutoTx = TRUE;
        memcpy(lcd_data, "                ", 21);
        LcdWriteLine(LCD_LINE_8, lcd_data);
        LcdSetPictureCursor(8,3);
        LcdDrawPicture(txoff_inverted32x8);

        //clear counters
        NbrOfSentPackets = 1;
        NbrOfReceivedPackets = 0;
        LED_TX = 0;
        LED_RX = 0;
        BLED_PIN = 0;
        LED_OK = 0;
        LED_ERROR = 0;

        //next state
        DemoStates = sDemoSendPing;
        return;
    }
    if( PB3_PIN == 0 || PB4_PIN == 0 )
    {

```

```

        while( PB3_PIN == 0 || PB4_PIN == 0 );
        //goto setting page
        DemoStates = sDemoSettingScreen;
        return;
    }
break;

case sDemoPacketReceived:
    //stop timer
    StopTmr3();
    //get received packet and put it into the buffer
    MessageBuffer.data = &RxTxBuffer[0];
    ReceivedData.message = MessageBuffer;
    RFGetBuffer(&ReceivedData);
    //stop RF stack
    RFIdle();
    //check what kind of packet received
    if( fWaitForPacketOrAck == PING )
    {
        //simple packet received -> send an ACK
        if( (RxTxBuffer[1] == demo_param.self_address) && (RxTxBuffer[4] == PING)
)
        {
            //if the packet is sent to this device, then send ACK
            //report the result
            DemoPingReceived(((uint16)(RxTxBuffer[2]<<8) +
(uint16)RxTxBuffer[3]),
ReceivedData.RSSI_level);

            //set ACK packet
            memcpy(&string[0],&RxTxBuffer[0],4);
            RxTxBuffer[0] = string[1];
            RxTxBuffer[1] = string[0];
            RxTxBuffer[2] = string[2];
            RxTxBuffer[3] = string[3];
            RxTxBuffer[4] = ACK;
            MessageBuffer.header = HeaderBuffer;
            MessageBuffer.length = 5;
            MessageBuffer.data = &RxTxBuffer[0];
            //send the packet
            LED_TX = 1;
            RFTransmit(&MessageBuffer);
            //set flag ->ACK packet has sent
            fPacketSent = ACK;
            //next state
            DemoStates = sDemoWaitForSendPacket;
        }
        else
        {
            //the packet is not PING or sent to another device, so discard it -> goto
            // RX state:
            //wait for push button or another PING packet
            //not known, but valid packet received
            DemoSimplePacketReceived(ReceivedData.RSSI_level,ReceivedData.message.length);
            //goto RX state: wait for push button or another packet
            DemoStates = sDemoStartReceive;
        }
        return;
    }
    if( fWaitForPacketOrAck == ACK )
    {
        //ACK packet received -> calculate the PER and print the result
        //report the result
        DemoAckReceived(TRUE, NnbrOfSentPackets, ReceivedData.RSSI_level);
        //goto RX state: wait for push button or another PING packet
        DemoStates = sDemoStartReceive;
        return;
    }
break;

case sDemoRxError:
    //stop RF stack and timer
    RFIdle();
    StopTmr3();

```

```

        if( fWaitForPacketOrAck == PING )
        {
            //error occurred during receiving a PING packet
            DemoStates = sDemoStartReceive;
        }
        if( fWaitForPacketOrAck == ACK )
        {
            //error occurred during receiving an ACK packet
            //report that ACK didn't receive
            DemoAckReceived(FALSE, NbrOfSentPackets, 0xFF);
            //goto RX state: wait for push button or another PING packet
            DemoStates = sDemoStartReceive;
        }
    }
    break;

    case sDemoSendPing:
        //set the payload of PING message
        RxTxBuffer[0] = demo_param.self_address;
//source address field
        RxTxBuffer[1] = demo_param.destination_address;
//destination address field
        RxTxBuffer[2] = (uint8)(NbrOfSentPackets >> 8);
//16bit packet ID
//((incremented after every
        RxTxBuffer[3] = (uint8)(NbrOfSentPackets & 0x00FF);
//PING packet transmission
        RxTxBuffer[4] = PING;
//packet type field
        //set the PING message
        MessageBuffer.header = HeaderBuffer;
        MessageBuffer.length = 5;
        MessageBuffer.data = &RxTxBuffer[0];
        DemoSendPing();
        //start sending the PING packet
        LED_TX = 1;
        RFTransmit(&MessageBuffer);
        //set flag ->PING packet sent
        fPacketSent = PING;
        //next state
        DemoStates = sDemoWaitForSendPacket;
    break;

    case sDemoWaitForSendPacket:
        //do nothing, wait for the RF stack to send the packet
    break;

    case sDemoPacketSent:
        //check what kind of packet has sent
        LED_TX = 0;
        if( fPacketSent == PING )
        {
            //PING packet sent -> goto RX state immediately: wait for ACK
            //stop the RF stack
            RFIdle();
            //set flags -> wait for simple packet and not for ACK
            fWaitForPacketOrAck = ACK;
            //start receiving
            RFReceive(&RxCommand);
            //start timeout (100ms is enough even at 1.2kbps data rate)
            StartTmr3(TMR3_INTERNAL_ENABLE, TMR3_8, ACK_TIMEOUT, FALSE);
            //report that PING sent correctly
            //DemoPingSent(TRUE, NbrOfSentPackets);
            //next state
            DemoStates = sDemoWaitForAck;
        }
        if( fPacketSent == ACK )
        {
            //ACK packet sent -> goto RX state: wait for push button or another PING packet
            //stop the RF stack
            RFIdle();
            //report that ACK sent correctly
            DemoAckSent(TRUE);
            //next state
            DemoStates = sDemoStartReceive;
        }
    }
}

```



```

        }
        break;

        case sDemoTxError:
            //error occured during packet transmission
            LED_TX = 0;
            if( fPacketSent == PING)
            { //ping packet didn't send, because error occured
                DemoPingSent(FALSE, NmbrOfSentPackets);
                DemoStates = sDemoStartReceive;
            }
            if( fPacketSent == ACK )
            { //ACK packet didn't send, because error occured
            }
        break;

        case sDemoWaitForAck:
            if( Tmr3Expired() == TRUE )
            { //ACK didn't receive within timeout
                //stop timer & RF Stack
                StopTmr3();
                RFIdle();
                //report that timeout occured and ACK didn't receive
                DemoAckReceived(FALSE, NmbrOfSentPackets, 0xff);
                //goto RX state: wait for push button or another PING packet
                DemoStates = sDemoStartReceive;
            }
        break;

        default:
        break;
    }
}

```

```

void DemoSendPing(void)
{
}

```

```

/*+++++
+
+ FUNCTION NAME: void DemoPingSent(uint8 success, uint16 pid)
+
+ DESCRIPTION: the function is called if the PING packet sent correctly
+ or there was a problem to send it. It reports the result
+ on the LCD and to the UART.
+
+ INPUT: success - status of the PING transmit
+ pid - packet ID of the transmitted PING packet
+
+ RETURN: None
+
+ NOTES:
+
+******/
void DemoPingSent(uint8 success, uint16 pid)
{
    if( success == TRUE )
    { //packet sent correctly
        //report the result on the LCD

        //report the result to the UART
    }
    else
    { //error occured during packet transmission
        LED_ERROR = 1;
        //report the result on the LCD
        sprintf(lcd_data,"PING TRANSMIT ERROR! ");
        LcdWriteLine(LCD_LINE_6,lcd_data);
    }
}

```

```

        //report the result to the UART
        printf("PING TRANSMIT ERROR!\r\n>");
        LED_ERROR = 0;
    }
}

/*+++++
+
+ FUNCTION NAME: void DemoAckSent(uint8 success)
+
+ DESCRIPTION: the function is called if the ACK packet sent correctly
+ or there was a problem to send it. It reports the result
+ on the LCD and to the UART.
+
+ INPUT: success - status of the ACK transmission (TRUE/FALSE)
+
+ RETURN: None
+
+ NOTES:
+
+******/
void DemoAckSent(uint8 success)
{
    if( success == TRUE )
    { //packet sent correctly
        LED_TX = 1;
        //report the result on the LCD
        memcpy(lcd_data,"ACK SENT",21);
        LcdWriteLine(LCD_LINE_6,lcd_data);
        //report the result to the UART
        printf("ACK SENT!\r\n>");
        LED_TX = 0;
    }
    else
    { //error occured during packet transmission
        //report the result on the LCD
        LED_ERROR = 1;
        memcpy(lcd_data,"ACK TX ERROR!",21);
        LcdWriteLine(LCD_LINE_6,lcd_data);
        //report the result to the UART
        printf("ACK TX ERROR!\r\n>");
        LED_ERROR = 0;
    }
    //timeout to show the result
    StartTmr3(TMR3_INTERNAL_ENABLE, TMR3_8, 25000, FALSE);
    while( Tmr3Expired() == FALSE );
    StopTmr3();
    memcpy(lcd_data,"",21);
    LcdWriteLine(LCD_LINE_6,lcd_data);
}

/*+++++
+
+ FUNCTION NAME: void DemoPingReceived(uint16 pid, uint8 rssi_level)
+
+ DESCRIPTION: the function is called if a PING packet received correctly
+ It reports the result on the LCD and to the UART.
+
+ INPUT: pid - packet ID of the received PING packet
+ rssi_level - RSSI level during packet reception
+
+ RETURN: None
+
+ NOTES:
+
+******/
void DemoPingReceived(uint16 pid, uint8 rssi_level)
{
    LED_RX = 1;
    BLED_PIN = 1;
}

```

```

    StartTmr3(TMR3_INTERNAL_ENABLE, TMR3_8, 2500, FALSE);
    while( Tmr3Expired() == FALSE );
    StopTmr3();
    LED_RX = 0;
    BLED_PIN = 0;
}
/*+++++
+
+ FUNCTION NAME:  void DemoAckReceived(uint8 success, uint16 pid, uint8 rssi_level)
+
+ DESCRIPTION:   the function is called if an ACK packet received correctly
+                or there was a problem during the ACK reception.
+                It reports the result on the LCD and to the UART.
+
+ INPUT:         success - status of the ACK reception (TRUE/FALSE)
+                pid    - packet ID of the received ACK
+                rssi_level - RSSI level during packet reception
+
+ RETURN:        None
+
+ NOTES:
+
+******/
void DemoAckReceived(uint8 success, uint16 pid, uint8 rssi_level)
{
uint8 i;
float temp_float;

    //set the variable
    if( success == TRUE )
    {
        LED_OK = 1;
        LED_RX = 1;
        BLED_PIN = 1;
        NbrOfReceivedPackets++;
        sprintf(lcd_data,"ACK RECEIVED!      ");
        LcdWriteLine(LCD_LINE_6,lcd_data);
    }
    else
    {
        LED_ERROR = 1;
        sprintf(lcd_data,"ACK RECEIVING ERROR! ");
        LcdWriteLine(LCD_LINE_6,lcd_data);
    }
    //calc the PER
    temp_float = CalcPer(NbrOfSentPackets,NbrOfReceivedPackets);

    //report the result
    printf(" %03u packet sent, %03u ACK received, ",NbrOfSentPackets,NbrOfReceivedPackets);
    printf("PER:%3.2f\r\n",temp_float);
    sprintf(lcd_data,"                ");
    sprintf(lcd_data,"%03u/%03u PER:%3.2f%%",NbrOfReceivedPackets,NbrOfSentPackets,temp_float);
    LcdWriteLine(LCD_LINE_5,lcd_data);

    LED_OK = 0;
    LED_RX = 0;
    BLED_PIN = 0;
    LED_ERROR = 0;
    StartTmr3(TMR3_INTERNAL_ENABLE, TMR3_8, 50000, FALSE);
    while( Tmr3Expired() == FALSE );
    StopTmr3();
    sprintf(lcd_data,"                ");
    LcdWriteLine(LCD_LINE_6,lcd_data);
}
/*+++++
+
+ FUNCTION NAME:  void DemoSimplePacketReceived(uint8 rssi_level, uint8 length)
+
+ DESCRIPTION:   print a message when a simple packet received
+
+

```

```

+ RETURN:          None
+
+ INPUT:           rssi_level - signal strength of te received packet
+                  length - packet length
+
+ NOTES:
+
+******/
void DemoSimplePacketReceived(uint8 rssi_level, uint8 length)
{
    LED_RX = 1;
    BLED_PIN = 1;

    //report the result on the LCD
    sprintf(lcd_data,"PACKET RECEIVED:      ");
    LcdWriteLine(LCD_LINE_5,lcd_data);
    sprintf(lcd_data,"Length:%2u, RSSI:%3u  ",length,rssi_level);
    LcdWriteLine(LCD_LINE_6,lcd_data);
    //report the result to the UART
    printf("PACKET RECEIVED! Length: %2u, RSSI: %3u\r\n",length,rssi_level);
    //some delay to see the message on the screen
    StartTmr3(TMR3_INTERNAL_ENABLE, TMR3_8, 20000, FALSE);
    while( Tmr3Expired() == FALSE );
    //clear display
    sprintf(lcd_data,"                    ");
    LcdWriteLine(LCD_LINE_5,lcd_data);
    sprintf(lcd_data,"                    ");
    LcdWriteLine(LCD_LINE_6,lcd_data);

    StopTmr3();
    BLED_PIN = 0;
    LED_RX = 0;
}

/*******/
+
+ FUNCTION NAME:   float CalcPer(uint16 total, uint16 received)
+
+ DESCRIPTION:    calculate the PER
+
+ RETURN:         PER in percentage
+
+ INPUT:          total - total number of expected packages
+                  received - number of received packages
+
+ NOTES:
+
+******/
float CalcPer(uint16 total, uint16 received)
{
    float temp_float;

    temp_float = 0;

    if( total != 0 )
    {
        temp_float = ((float)(total - received) / (float)total) * 100;
        if(temp_float > 100)
        {
            temp_float = 100;
        }
    }

    return temp_float;
}

/*******/
+
+ FUNCTION NAME:   void SetRadio(void)

```

```

+
+ DESCRIPTION:      set radio parameters
+
+ RETURN:          Freq number
+
+ INPUT:           data rate
+
+ NOTES:
+
+******/
uint8 SetRadio(uint8 data_rate)
{
uint8 temp8;
uint16 temp16;

//wake up the RF chip if it was in standby state
temp8 = RFWakeUp();
if( temp8 == RF_OK )
{//wait for waking up
do{RFGetStatus(&RfStatus);} while( RfStatus.status == sRFWakingUp );
RFGetStatus(&RfStatus);
if( RfStatus.status == sRFWakeUpError )
{
LcdClearDisplay();
printf("\r\nERROR: XTAL startup error!");
printf("\r\nTry a different Testcard!\r\n");
memcpy(lcd_data,"ERROR: XTAL startup! ",21);
LcdWriteLine(LCD_LINE_2,lcd_data);
memcpy(lcd_data," Try a different ",21);
LcdWriteLine(LCD_LINE_5,lcd_data);
memcpy(lcd_data," Testcard! ",21);
LcdWriteLine(LCD_LINE_6,lcd_data);
while(1);
}
}
else
{//reset the RF stack
RFIdle();
}
demo_param.frequency_band = demo_freqs[demo_param.freq_number].frequency_band;
demo_param.frequency = demo_freqs[demo_param.freq_number].frequency;

rf_param[0] = demo_param.frequency_band;
rf_param[1] = (uint8)(demo_param.frequency >> 8);
rf_param[2] = (uint8)(demo_param.frequency & 0x00FF);
RFSetParameter(pRfFrequency, rf_param, 3);

//header fields are not used in the packet
HeaderBuffer.enabled_headers = 0;
RxCommand.header = HeaderBuffer;
//preamble length
temp16 = PREAMBLE_LENGTH * 2;
rf_param[0] = ((uint8)(temp16 >> 8) & 0x01);
rf_param[1] = (uint8)(temp16 & 0x00FF);
RFSetParameter(pRfPreambleLength, rf_param, 1);
//preamble detection
rf_param[0] = PD_LENGTH;
rf_param[1] = PD_ERROR;
RFSetParameter(pRfPreambleDetection, rf_param, 2);
//synch word
rf_param[0] = 2;
rf_param[1] = 0x2D;
rf_param[2] = 0xD4;
RFSetParameter(pRfSynchronWord, rf_param, 3);
//GPIOs
rf_param[0] = GPIO_TX_CLK;
rf_param[1] = GPIO_RX_DATA;
rf_param[2] = GPIO_VALID_PREA;
RFSetParameter(pRfGpios, rf_param, 3);
//set rf parameters according the requirements
RFSetRfParameters( data_rate );

```

```

        return demo_param.freq_number;
    }

/*+++++
+
+ FUNCTION NAME: void GetDataRate(RF_SAMPLE_SETTINGS datarate, uint8 * string)
+
+ DESCRIPTION: it gives back the datarate in string format
+
+ RETURN: string
+
+ INPUT: data rate
+
+ NOTES:
+
+******/
void GetDataRate(RF_SAMPLE_SETTINGS datarate, uint8 * string)
{
    switch( datarate)
    {
        case DR2400BPS_DEV45KHZ: memcpy(string,"2.4",3); break;
        case DR4800BPS_DEV45KHZ: memcpy(string,"4.8",3); break;
        case DR9600BPS_DEV45KHZ: memcpy(string,"9.6",3); break;
        case DR10KBPS_DEV5KHZ_MOD1: memcpy(string," 10",3); break;
        case DR20KBPS_DEV10KHZ_MOD1: memcpy(string," 20",3); break;
        case DR40KBPS_DEV20KHZ_MOD1: memcpy(string," 40",3); break;
        case DR50KBPS_DEV25KHZ_MOD1: memcpy(string," 50",3); break;
        case DR100KBPS_DEV50KHZ_MOD1: memcpy(string,"100",3); break;
        case DR125KBPS_DEV62KHZ5_MOD1: memcpy(string,"128",3); break;
    }
}

/*+++++
+
+ FUNCTION NAME: uint16 GetFreq(uint8 freq_number)
+
+ DESCRIPTION: it gives back the center freq in decimal format
+
+ RETURN: frequency in decimal
+
+ INPUT: serial number of the frequency setting
+
+ NOTES:
+
+******/
uint16 GetFreq(uint8 freq_number)
{
    return demo_freqs[freq_number].freq_decimal;
}

```

This page has been intentionally left blank.

Integration Associates, Inc.
110 Pioneer Way, Unit L
Mountain View, California 94041
Tel: 650.969.4100
Fax: 650.969.4582
www.integration.com
marketing@integration.com
wireless.support@integration.com
P840

The specifications and descriptions in this document are based on information available at the time of publication and are subject to change without notice. Integration Associates assumes no responsibility for errors or omissions, and disclaims responsibility for any consequences resulting from the use of information included herein. Additionally, Integration Associates assumes no responsibility for the functioning of undescribed features or parameters. Integration Associates reserves the right to make changes to the product and its documentation at any time. Integration Associates makes no representations, warranties, or guarantees regarding the suitability of its products for any particular purpose and does not assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability for consequential or incidental damages arising out of use or failure of the product. Nothing in this document shall operate as an express or implied license or indemnity under the intellectual property rights of Integration Associates or third parties. The products described in this document are not intended for use in implantation or other direct life support applications where malfunction may result in the direct physical harm or injury to persons. **NO WARRANTIES OF ANY KIND, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, ARE OFFERED IN THIS DOCUMENT.**

©2008 Integration Associates, Inc. All rights reserved. Integration Associates is a trademark of Integration Associates, Inc. All other trademarks belong to their respective owners.