

CC1110/CC2510 Example Software

Packet Error Rate Application Example

By E. Syvertsen

Abstract

The Packet Error Rate (PER) application example is a tool for measuring the quality of your wireless communication channel. The software is adapted to run on two SmartRF®04EB (Evaluation Boards) with each having either a CC1110EM or CC2510EM (Evaluation Module) mounted. The program lets you in an easy manner navigate a menu with preset alternatives configure the main radio parameters such as radio frequency, data rate and number of packets to transmit. The one-way packet error rate test will show the accumulated PER at the receiver and instantaneous Received Signal Strength Indicator (RSSI) value during transmission. This makes it possible to easily perform a link range test or evaluate the CC1110/CC2510 performance against the specifications in the data sheet with various transmission settings.

Table of contents

1	Introduction.....	2
2	Using the software.....	3
2.1	<i>Prerequisites</i>	<i>3</i>
2.2	<i>General program behavior.....</i>	<i>4</i>
2.3	<i>The PER test transmitter</i>	<i>7</i>
2.4	<i>The PER test receiver.....</i>	<i>9</i>
2.4.1	Handling of packets with incorrect length.....	11
2.4.2	Calculation of PER and RSSI.....	11
3	Set your own settings	12
3.1	<i>Tuning your RF link range.....</i>	<i>13</i>
	Document History	14

1 Introduction

The Packet Error Rate (PER) is a quality measure of a radio channel. By transmitting data packets from one node to another while counting the number of packets that are lost or damaged at the receiver, the packet error rate can be calculated. This document describes the features and how to use the PER test application example for the CC1110 or CC2510, two low-cost system-on-chips (SoC) with 8051 MCU and low power RF transceiver from TI.

The application has an easy to set-up configuration menu where you select radio settings such as radio frequency, data rate, which evaluation board to be transmitter and which one to be the receiver, and the total number of packets to transmit for the test. Both the instantaneous PER in percent and the averaged instantaneous Received Signal Strength Indicator (RSSI) in dBm can be read from the receiver's LCD during testing.

The software is provided both as pre-built executables (HEX files), for quick and simple downloading to the SoC and as source code to allow adjustments of the PER test. The internals of the PER test will be explained in order to facilitate modifications. The application is designed to run on two SmartRF[®]04EBs, both with either a CC1110EM or CC2510EM and antennas mounted.



Figure 1: A SmartRF[®]04EB with a CC2510EM and antenna

The PER test can be used to measure the link range or test the performance of the radios in different environments and with various settings.

2 Using the software

2.1 Prerequisites

The quickest way to try the software is to download the pre-built HEX files to the CC1110EMs or CC2510EMs. To successfully download and run the software described in this document, the following hardware is needed:

- 2 x SmartRF[®]04EBs
- 2 x CC1110EMs or 2 x CC2510EMs with appropriate antennas
- Minimum 1 USB cable
- SmartRF04 Flash Programmer
- 2 x batteries (9 volt) or other power source (e.g. USB cable).

The SmartRF04 Flash Programmer is freely downloadable from the TI website. Go to www.ti.com/lprf and use the part number search in the upper right corner to find e.g. "CC1110". Click on one of the CC1110 Development Kits among your hits and you should find the Flash Programmer in the "Tools and Software" section on that page.

For detailed instructions on how to use the flash programmer, please refer to the Flash Programmer User Manual bundled with the software.

If you also want to modify the software to try other radio configurations, you will instead of the flash programmer need:

- IAR Embedded Workbench for 8051

More details about how to build your own version is found further below in this document, in section 3.

2.2 General program behavior

The packet error rate application example sets up a one-way RF link between two SmartRF[®]04EBs. One board will operate as the transmitter and the other board will operate as the receiver. Before any data can be transmitted or received, both radios need to be configured properly. Only some main parameters are configurable when running the software, but as the source code is provided the hard-coded radio settings can also be modified.

When you turn on power a splash screen with the Chipcon logo and the name and version of the system-on-chip is shown. To start the program press the S1 button in the lower right corner of the SmartRF04[®]EB. At the first step of the setup process, the program asks you to choose what radio frequency to use from a menu. The menu is navigated by moving the joystick up or down, and to select the currently displayed option the S1 button must be pressed. You will have four options, but if you are using the CC1110 SoC, you will have to pick a frequency that matches the frequency band(s) supported by your evaluation module (EM). Even though the CC1110 itself supports all four listed frequencies, the components and antenna on the EM are matched for use in either the 433 MHz or the 868/915 MHz frequency band.

Next, you will have to choose data rate, whether the node should operate as receiver or transmitter, and in the latter case specify how many packets to transmit. To perform a PER test between two nodes, they have to be set up with a similar configuration, with the only exception of one being configured as receiver and the other as a transmitter. The menu layouts when running the software on CC1110 and CC2510 are shown in Figure 2 and

Figure 3, respectively.

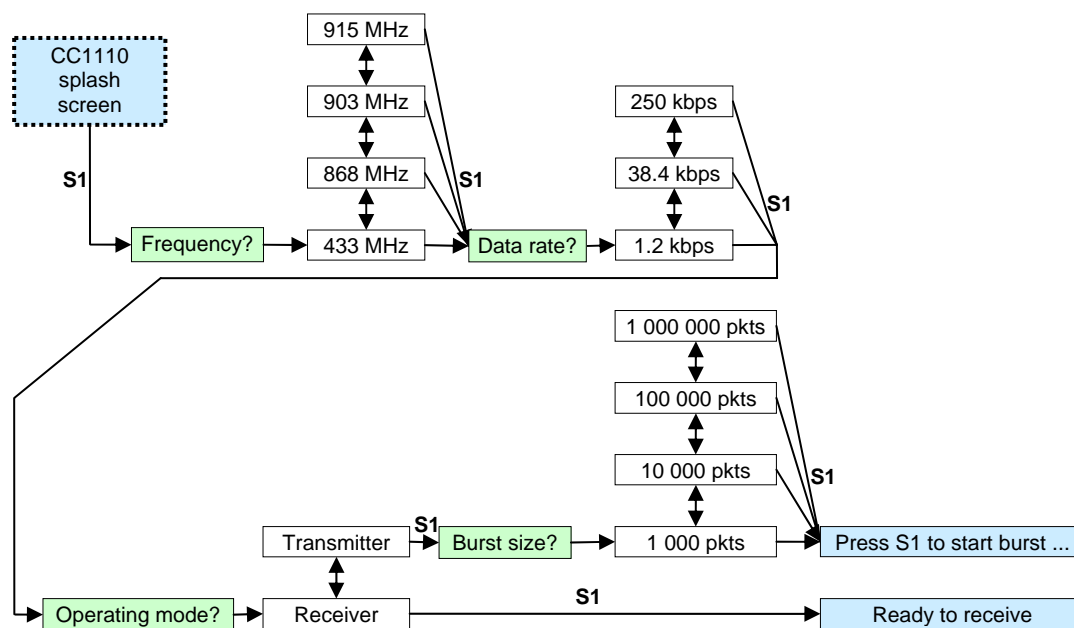


Figure 2: The setup menu layout on CC1110

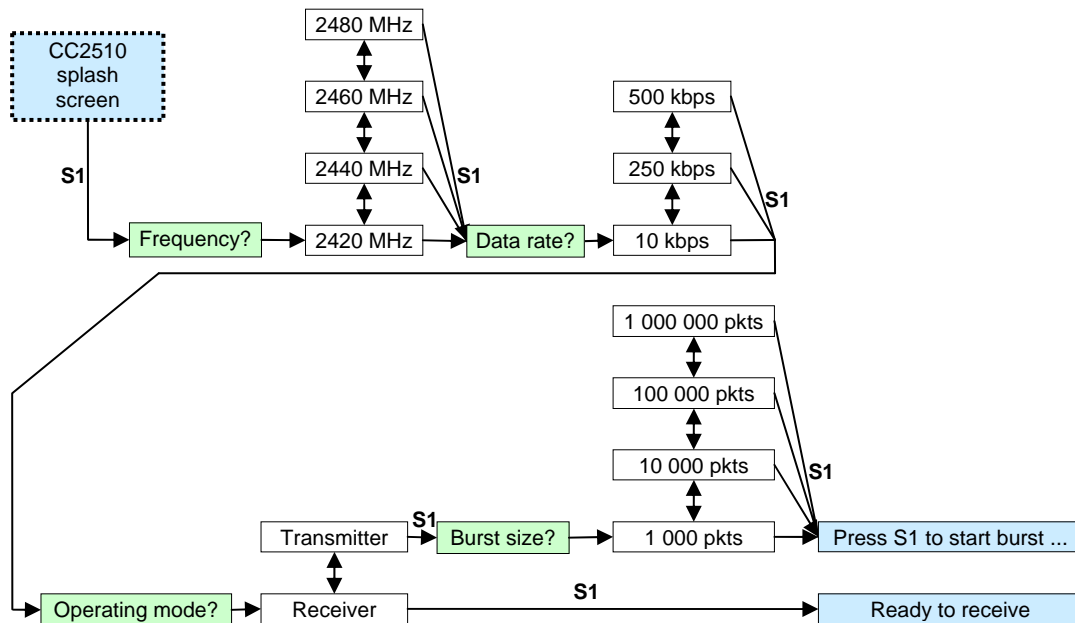


Figure 3: The setup menu layout on CC2510

The CC1110 transmitter is programmed for a +10 dBm output power, and the transmitting CC2510 is programmed for 0 dBm. The modulation settings for the predefined menu options are as follows:

System-on-Chip	Data rate	Modulation settings
CC1110	1.2 kbps	GFSK modulation, 58 kHz RX filter bandwidth, 5 kHz deviation.
	38.4 kbps	GFSK modulation, 100 kHz RX filter bandwidth, 19 kHz deviation.
	250.0 kbps	GFSK modulation, 540 kHz RX filter bandwidth, 127 kHz deviation.
CC2510	10.0 kbps	2-FSK modulation, 232 kHz RX filter bandwidth, 38 kHz deviation.
	250.0 kbps	MSK modulation, 540 kHz RX filter bandwidth.
	500.0 kbps	MSK modulation, 812 kHz RX filter bandwidth.

Table 1: Modulation settings for the various data rates

Some radio settings are not configurable from the menu and are common for both system-on-chips independent of the chosen frequency or data rate. The radio is configured for a 4 byte preamble and a 4 byte sync word (2 bytes repeated twice) to prefix each data packet. Even though all transmitted packets have the same length, the packets use the variable packet length format with a length byte specifying the length, defined as *PACKET_LENGTH* in the program. The next 2 bytes are the network identifier which is used to identify the transmitter/receiver pair constituting the PER network. This makes it robust against other interfering sources transmitting at the same frequency.

_____ Included in length byte (PACKET_LENGTH) _____



Name Of Register Bit(s)	Value	Meaning
PKTLEN	PACKET_LENGTH	As variable length packets are used, this sets the maximum packet length allowed.
PKTCTRL1.APPEND_STATUS	1	Append two bytes (the RSSI byte and the CRC/LQI byte) to the payload of the received packet.
PKTCTRL0.WHITE_DATA	1	Automatic data whitening (and de-whitening) over the air.
PKTCTRL0.CRC_EN	1	CRC calculation appended in TX and CRC check in RX enabled.
PKTCTRL0.LENGTH_CONFIG	01	Variable length packet mode.
MCSM1.RXOFF_MODE	00	When a packet has been received, the radio goes automatically to IDLE state.
MCSM1.TXOFF_MODE	00	When a packet has been transmitted, the radio goes automatically to IDLE state.
MCSM0.FS_AUTOCAL	01	Start calibration when going from IDLE to TX or RX.
MCSM0.CLOSE_IN_RX (only available for CC1110)	00	No RX attenuation that can prevent saturation in RX at short distances.

Table 2: Some common radio settings

TEXAS
INSTRUMENTS

2.3 The PER test transmitter

This section describes how the PER test program works when the unit is set to operate as a transmitter. At this point the radio is already configured, and the DMA will be set to transport data from a packet buffer to the radio's RFD register. The DMA channel is armed, and the user is prompted in a menu for how many packets to transmit.

Figure 5 shows how the application works in transmitting mode. The HW initialization and configuration is short for clock management, LCD initialization, splash screen, verification that the chip/revision is supported, radio configuration, selection of transmitter mode and the DMA setup. After all this, the packet initialization is performed, which includes insertion of the length byte, the network ID bytes and the dummy payload into the packet appropriately. The user then has to select the packet burst size, another S1 button press is needed to start the packet burst. When triggered, the packet is prepared for transmission by inserting the appropriate sequence number. The DMA channel is then armed, the LED toggled on to indicate activity and the radio is set in transmit mode. The DMA is configured to automatically feed the radio's data register RFD with data from the packet buffer when the radio has entered TX.

Once the whole packet is transmitted, an IRQ_DONE interrupt is triggered. The interrupt service routine will clear the interrupt flags and set the `pktSentFlag`, showing the main loop that a packet has been sent. At this point the DMA is finished and the channel is disarmed and the radio will switch back to IDLE state because of the TX_OFFMODE in MCSM1 register is set to IDLE (which is the radio's default setting). In addition to the time delay introduced from switching between TX and IDLE for each packet and the automatic frequency synthesizer calibration before entering TX a small wait loop is used to ensure there are enough spacing between packets for the receiver to process the received data and get back to RX before the next one arrives.

The number on the LCD indicating the total number of packets that have been transmitted will be incremented. A quick poll of the S1 button is performed, which will halt the burst for the possibility of starting all over again if it is pushed. If not, the next packet will be prepared. As the next packet to be sent is read from the same memory buffer as the one just sent, the sequence number has to be incremented. The DMA channel is rearmed and the radio is put in transmit mode. This repeats itself until the specified number of packets is sent. The transmitter will then be ready to start on another burst of packets.

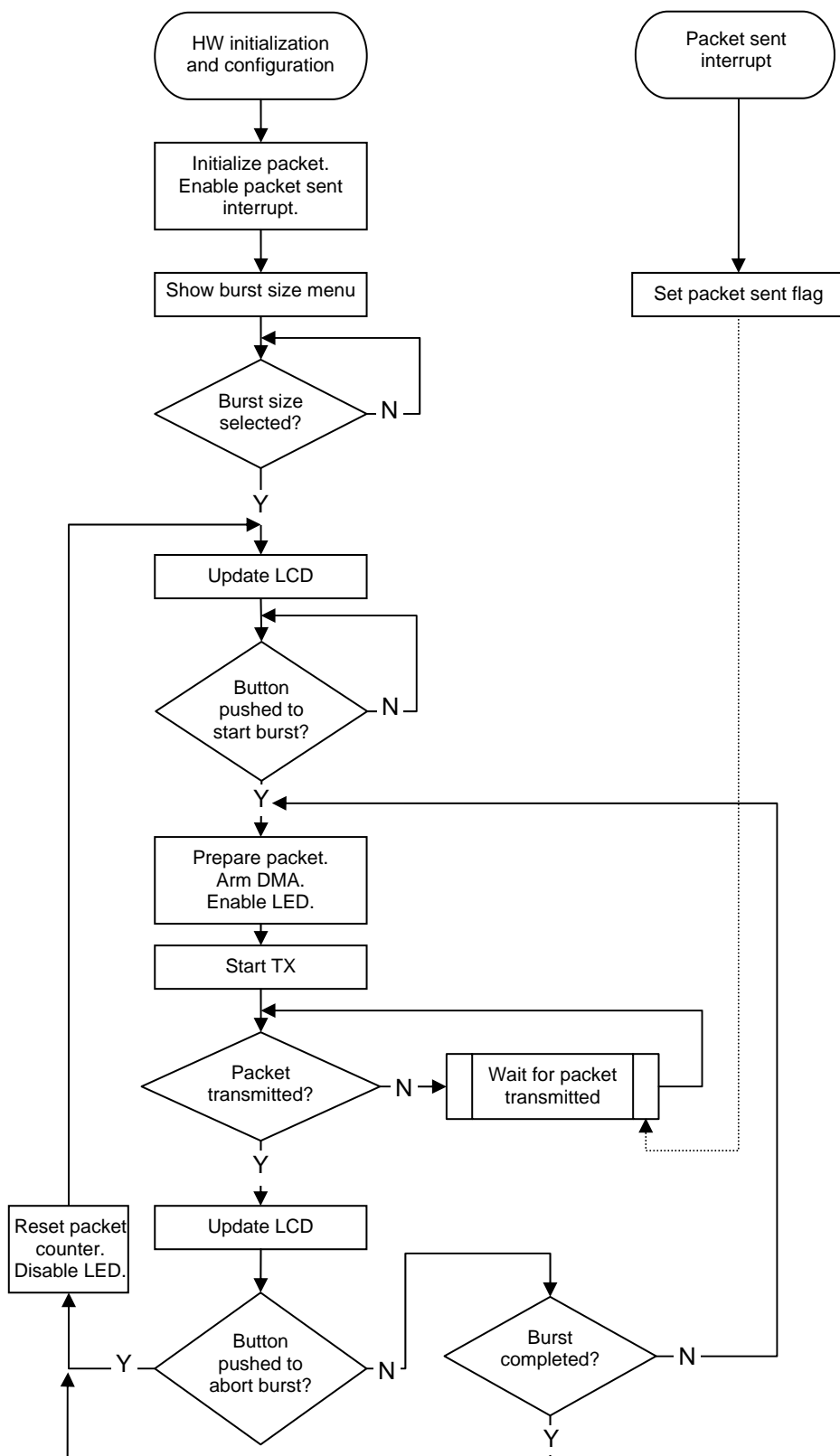


Figure 5: Flow chart of the PER test application in transmitter mode

2.4 The PER test receiver

Figure 6 shows how the PER test application works in receive mode. To compact the flow chart the “HW init and configuration” includes clock management, LCD initialization, splash screen, validation of the chip and revision, radio configuration and selection of receiver mode in the menu.

The CC1110/CC2510 prepares for packet reception by configuring a DMA channel for transport of incoming data from radio register RFD to a packet buffer. Also, the IRQ_DONE interrupt from the radio is enabled which will trigger on every received packet. The LCD will then display “Ready to receive”, the DMA is armed and the radio is put in RX to wait for packets. The receiver will when initiated expect a packet with sequence number 1.

When a packet is received an interrupt service routine will run. This clears the interrupt flags and sets the `pktRcvdFlag` variable. The radio will automatically go back to idle mode because it is configured with `MCSM1.RXOFF_MODE = IDLE`. The received data packet will now be found in the packet buffer and it has to be validated as a PER test packet to avoid confusion from interfering traffic at the same frequency. This is done by the `pktCheckValidity()` function, and based on the result one of the following scenarios will occur:

1. The packet length, the network ID, the CRC and the packet's sequence number are all OK. This means that the packet is a successful PER test packet! The expected sequence number of the next packet to be received is incremented as well as the number of received packets.
2. The packet length, the network ID and the CRC are OK, but the sequence number is not as expected. In this case there are two sub-scenarios:
 - a. The received sequence number is less than the expected sequence number causing the received packet to be considered as a part of a new burst of packets, i.e. a new/restarted PER test. This will clear the accumulated PER statistics and the expected sequence number of the next packet is set to the received sequence number incremented by 1.
 - b. The received sequence number is larger than the expected sequence number. This means that this packet was received correctly, but a number of packets before this one have been lost. The expected sequence number of the next PER test packet is increased to the received sequence number plus 1 and the counter for number of received packets is incremented. The packet loss is accounted for as the difference between (expected sequence number – 1) and the number of received packets.
3. The packet length and the network ID are correct, but the CRC fails. The number of bad packets and the number of received packets are both incremented. The expected sequence number of the next packet to be received is also increased by 1.
4. The packet length is OK, but the network ID does not match. The packet comes from either an interfering source transmitting with similar packet length, or the packet is from the transmitter, but the identifier has corrupted. For this scenario the packet will not be considered a relevant PER test packet and no action is taken. If the packet actually was from the transmitter, the corrupted packet will be counted as a lost packet when the next good packet arrives, because the next sequence number which is received is higher than the expected and scenario 2b occurs.
5. The packet length is incorrect and will be ignored. There are however some details about the handling of packets with incorrect length that are worth knowing. These are further described in section 2.4.1.

For each packet received, LED1 on the SmartRF04[®]EB will flash briefly. Additionally, a symbol in the upper right corner of the display will toggle for every 32nd received PER test packet. Under stable conditions when the PER and RSSI is left unchanged, these are indicators to determine whether the RF link is still preserved.

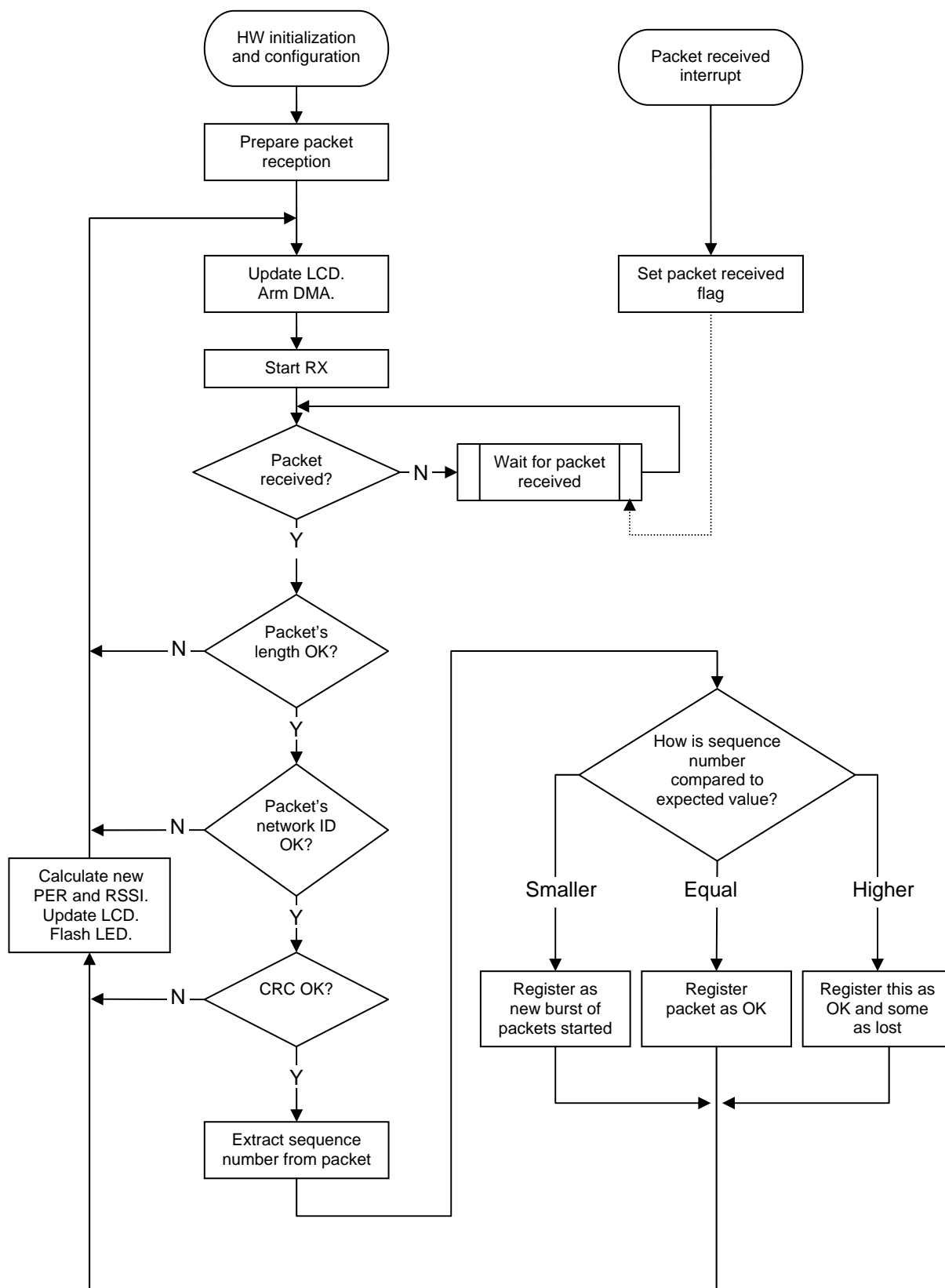


Figure 6: Flow chart of the PER test application in receiver mode

2.4.1 Handling of packets with incorrect length

When the radio is used in variable packet length mode, the first byte received after the sync word specifies the length of the packet payload. In this mode the PKTLEN radio register then sets the upper limit on the packet length. The PER test application utilizes the value *PACKET_LENGTH* both as the length byte value for transmitted packets and as the value set in PKTLEN register in the receiver.

The packet length of the detected packet may be either longer or shorter than expected. If the radio receives a packet with a length byte larger than what is set in the PKTLEN, the radio will discard this packet, but it will still generate an interrupt (IRQ_DONE), making it seem as a new packet is received. The DMA will not write anything new to the incoming packet buffer, so the previous packet will still remain intact. To prevent the old packet in memory to be accepted as a new one, the function `pktCheckId()` resets the network ID bytes in the packet buffer to 0 after each check. Any received (and discarded) packet will then not cause any false positives as it will not pass the network ID check and be ignored in the PER statistics.

For the opposite case where the radio receives a packet with a length byte smaller than PKTLEN the packet will be written by the DMA to the buffer. This packet will also be ignored since it will not pass the length check.

2.4.2 Calculation of PER and RSSI

To obtain the statistics used for calculating the packet error rate, the three following variables are used:

<code>perExpectedSeqNum</code>	The expected sequence number for the next packet that should arrive. This is equivalent to the number of received packets + lost packets + 1.
<code>perRcvdPkts</code>	The number of received packets that are identified as PER test packets, both with and without CRC error.
<code>perBadPkts</code>	The number of corrupted PER test packets received, i.e. with failed CRC.

Both corrupted packets and lost packets contribute to the packet error rate. The receiver is not using any timers to detect lost packets, e.g. if the packets were transmitted at a given timeslot or regular interval the receiver would immediately know if a packet was lost due to the signal being too weak. Instead, lost packets are detected through a jump in the sequence numbering. However, this implies that a series of lost packets will not be detected until a subsequent packet has been received correctly.

The PER value in percent is calculated by the formula:

$$\text{PER} = 100 * (\text{lost packets} + \text{bad packets}) / (\text{lost packets} + \text{received packets})$$

The receiver will show the instantaneous PER as well as an RSSI value. The RSSI value at sync word detection for each PER test packet is fetched from one of the two status bytes which are appended to the received packet by the radio. To convert this RSSI byte to an absolute RSSI value (in dBm) it has to be corrected with an RSSI offset. This offset is specified in the SoC's data sheet, and may vary with the frequency band in use. The RSSI value shown on the LCD is the average of the 32 (defined by *RSSI_AVG_WINDOW_SIZE*) last received PER packets. A circular buffer is used to calculate a sliding window average.

3 Set your own settings

The source code is provided to let you replace the predefined radio settings with your own. This assumes the use of IAR Embedded Workbench for 8051, version 7.30A or newer running on Microsoft Windows. It is also recommended to use the SmartRF Studio software to help you set up the radio registers.

These are the steps needed to modify the packet error rate application example:

1. Install IAR Embedded Workbench for 8051, version 7.3A or newer. The KickStart will currently not work due to its code size limitation.
2. Install the SmartRF Studio software. It is freely available from the TI website (<http://www.ti.com/smartrfstudio>).
3. If it's not already done, download the CC1110/CC2510 Example Software and unzip the files to your working directory.
4. Open the IAR IDE Workspace file (.eww) found in the unzipped "iar" folder.
5. Start SmartRF Studio which will help you configure new radio registers. See the SmartRF Studio User Manual for details on how to use the software.
6. Use SmartRF Studio's code export function. It's found in the upper menu: "File" -> "Export CCxx10 code...". Choose "RF settings SoC" for output format.
7. In IAR Embedded Workbench, overwrite the radio registers with the values from SmartRF Studio. All radio register settings are found in the function `radioConfigure()` in the file `per_test_radio.c`. You will also have to edit the menu options displayed on the LCD. These are found in the included `per_test_main.h` file. Make sure you modify the settings for the chip you are using. **Note:** Some registers are frequency dependent (like `FREQx` and `PA_TABLE0`), some registers are data rate (modulation) dependent, and some are even both. Because of this the registers are located in groups at various places inside this function, and you cannot copy all register settings directly from SmartRF Studio in one code block but rather have to go through the register values manually. For new RSSI offset values, please consult the respective system-on-chip's data sheet.

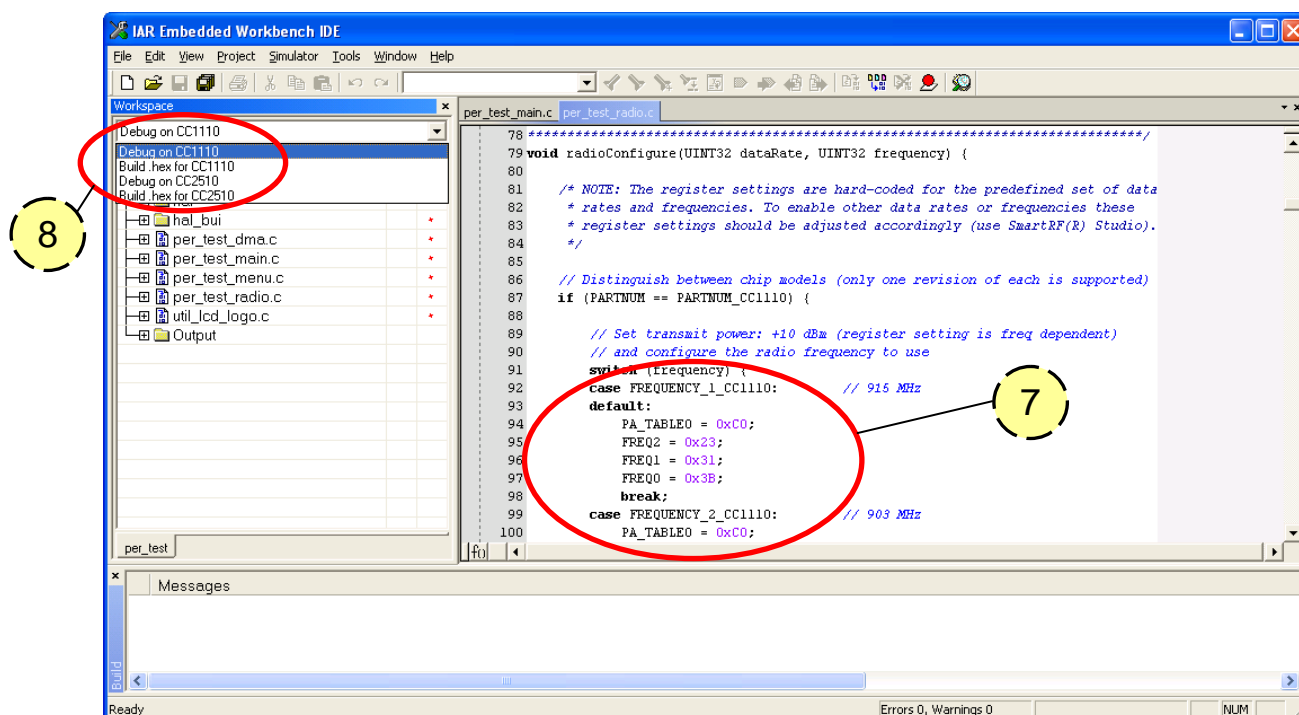


Figure 7: Overview of the workspace in IAR Embedded Workbench IDE

8. When done editing the source code, select your build target from the menu in the upper left corner (see Figure 7). This step-by-step guide assumes that one of the debug builds is chosen.
9. Select "Project" -> "Rebuild All". There should be no errors or warnings when IAR builds the executables. (If the "Build .hex for CCxxxx" option was chosen, it will now be output to the folder named "hex", potentially overwriting any previous version.)
10. Mount your CC1110EM or CC2510EM onto your SmartRF04[®]EB and attach it to your computer with a USB cable.
11. Make sure the power switch on the evaluation board is set in the rightmost position to power the board via the USB cable.
12. Select "Project" -> "Debug". IAR will now establish a connection with the target, download the application and program the system-on-chip. The debugger will be started, halting the target at `main()`.
13. Start the application by selecting "Debug" -> "Go". The LCD should display a splash screen with the EM's system-on-chip name and revision number.
14. Stop debugging by selecting "Debug" -> "Stop Debugging".
15. The SoC is now programmed and can now be operated independently from the computer by disconnecting the USB cable and use e.g. batteries as power source.
16. Repeat steps 9 through 15 for the other evaluation module (EM). You should now be able to perform your packet error rate test.

3.1 Tuning your RF link range

If the goal is to have a long communication link range, there are some radio parameters to be aware of that to great extent affect the RF link range. These are:

- Transmitter output power. A stronger signal will propagate longer.
- Data rate. A higher data rate will reduce the maximum range because the shorter symbol period makes it harder to demodulate correctly.
- Preamble length. A longer preamble may improve the sensitivity of the receiver and thus improve the link range.
- Frequency. In general, reducing the carrier frequency will improve the range.
- Antenna. A directional antenna with high antenna gain can improve your link range significantly.

Also, transmitting packets with shorter length will reduce the PER, but this does not affect the bit error rate (BER). As a result, fewer bytes will have to be retransmitted because of a packet error, but shorter packets generate more overhead.

Document History

Revision	Date	Description/Changes
-	2007-09-11	Initial release
A	2008-04-08	Updated section 3 to reflect the newly added build configurations (separate for debugging and for building hex file).