Headset Reference Design nRD24V1

Software Developer Guide v1.0

February 2007

Liability disclaimer

Nordic Semiconductor ASA reserves the right to make changes without further notice to the product to improve reliability, function or design. Nordic Semiconductor ASA does not assume any liability arising out of the application or use of any product or circuits described herein.

Life support applications

These products are not designed for use in life support appliances, devices, or systems where malfunction of these products can reasonably be expected to result in personal injury. Nordic Semiconductor ASA customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Nordic Semiconductor ASA for any damages resulting from such improper use or sale.

Contact details

For your nearest dealer, please see http://www.nordicsemi.no

Receive available updates automatically by subscribing to eNews from our homepage or check our website regularly for any available updates.

Main office:

Otto Nielsen's vei 12
7004 Trondheim
Phone: +47 72 89 89 00
Fax: +47 72 89 89 89
www.nordicsemi.no

Revision History

Date	Version	Description
February 2007	1.0	

Contents

1 Introduction

This document describes how the Software for the 'Wireless Headset' is organized and implemented. The Headset reference design includes the following three modules, the USB Dongle, audio Dongle and headset. All modules share common software source files, and compile switches determines the module when generating executable code.

2 SW architecture

The software (SW) organization is shown in Figure 1. SW Architecture. A source and header file represents each block, which is further explained in the following chapters.

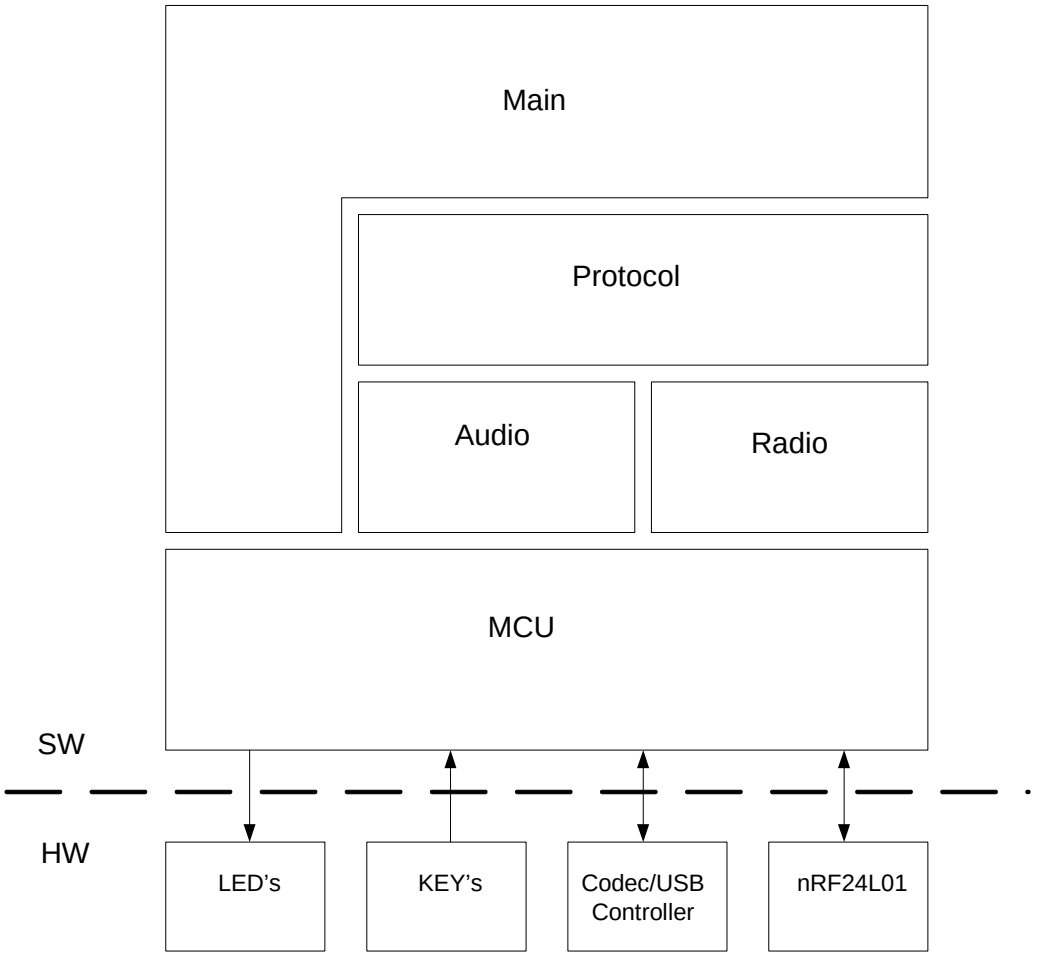

Figure 1. SW Architecture

2.1 MCU

The MCU represents the microcontroller resources and the interface to the physical hardware (HW). All HW interaction is performed using macros.

The MCU defines the following external and local prototypes:

External prototypes:

```
void init_mcu(void);
void init_spi_rf(void);
void init_spi_codec(void);
char spi_byte(char data_byte);
void start_timer1(unsigned int time, unsigned int period, char prescaler);
void start_timer0(char timeout, char prescaler);
void sleep(char wdt_prescaler, char mode);
void start_usart(void);
void stop_usart(void);
char eeprom_read(unsigned int address);
void eeprom_write(char data,unsigned int address);
```

Local prototypes:

2.2 Audio

The Audio handles the voice streaming towards the Codec/USB Controller including initialization routines, data formatting and the I2S interrupt.

The Audio defines the following interrupts, external and local prototypes.

Interrupts

```
SIGNAL (SIG_OUTPUT_COMPARE2B)
SIGNAL (SIG_OUTPUT_COMPARE1A)
```

External prototypes:

```
void init_codec(void);
void start_codec(void);
void stop_codec(void);
```

Local prototypes:

```
void write_codec_reg(char address,char data_byte);
```

2.3 Radio

The Radio includes all functions necessary for initialization and interaction with the nRF24L01.

The Radio defines the following external and local prototypes.

External prototypes:

```
void init_rf(void);
void init_rf_signalling(void);
void set_rf_rx(char freq);
void set_rf_tx(char freq);
void set_rf_off(void);
void rf_clearint(void);
char read_rf_byte(char byte);
void write_rf_byte(char byte);
void write_rf_reg(char address,char data_byte);
void write_rf_data(char inst,char *data_ptr,char nbr_of_bytes);
void reuse_rf_payload(void);
void flush_rf_tx(void);
void flush_rf_rx(void);
void clear_rf_interrupt(void);
void rf_enable_pulse(void);
char wait_rf_irq(char timeout, char prescaler);
void init_rf_test(void);
```

Local prototypes:

2.4 Protocol

The Protocol includes all functions necessary to handle the air interface for both MASTER and SLAVE. The Dongle will be MASTER and the Headset is SLAVE.

Pairing between the MASTER and SLAVE is done static by programming the MASTER and SLAVE with the same 3-byte ID.

The Protocol defines the following external and local prototypes.

External prototypes:

```
void init_protocol(void);
void init_freq(void);
char call_setup(char *freq_ptr,char n_freqs);
char call_detect(char *freq_ptr,char n_freqs,unsigned int n_rep);
void audio_transfer(void);
char get_sync(void);
void test_rf_transmitter(void);
```

Local prototypes:

```
void put_audio_packet(void);
void get_audio_packet(void);
void stuff_packet(void);
```

```
char compress_audio(int audio_linear16);
int expand_audio(char audio_compressed8);
```

2.5 Main

The main application will be a state machine controlling the initiation and clearing of calls. The application controls the local keys and LED's using application specific macros.

The Main defines the following local prototypes.:

Local prototypes:

```
void init_buffer(void);
char read_key(void);
```

3 Call States

The USB Dongle, Audio Dongle and Headset are all based on a state machine having a number of states and handling the different stages in the call. The following 'Call States' exist, which will be further explained in the following sections:

- Call Idle
- Call Setup
- Call Connected
- Call Re-connect

The state transitions are performed according to Figure 2. Call State Transitions.

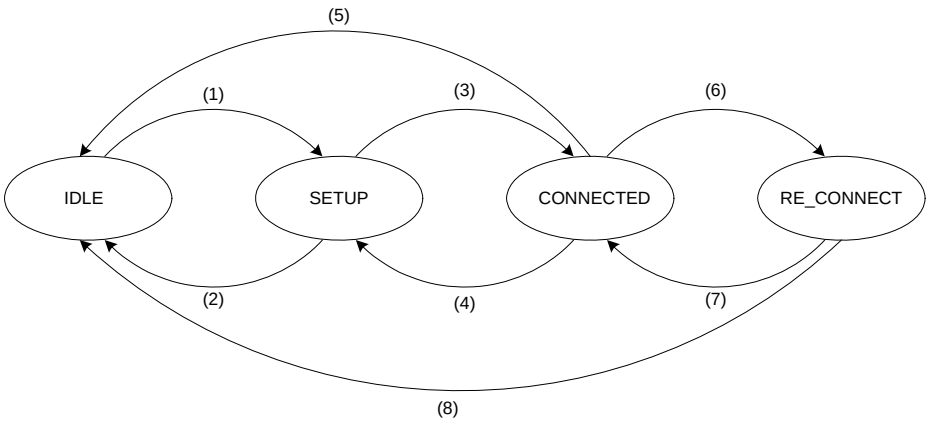

Figure 2. Call State Transitions

Signal	State Transition Description
1	High level on LEDN signal at pin 14 on the USB Controller chip SN11220ACF
2	Low level on LEDN signal at pin 14 on the USB Controller chip SN11220ACF
3	Call Setup Acknowledge packet received from Headset
4	
5	Low level on LEDN signal at pin 14 on the USB Controller chip SN11220ACF
6	No valid packets received from Headset within time limit
7	Call Setup Acknowledge packet received from Headset
8	Low level on LEDN signal at pin 14 on the USB Controller chip SN11220ACF

Table 1. USB Dongle State Transitions

Signal	State Transition Description
1	Call Setup Key (SW2) at Dongle pressed
2	Call Clear Key (SW4) at Dongle pressed
3	Call Setup Acknowledge packet received from Headset
4	
5	Call Clear Key (SW4) at Dongle or Headset pressed
6	No valid packets received from Headset within time limit
7	Call Setup Acknowledge packet received from Headset
8	Call Clear Key (SW4) at Dongle pressed

Table 2. Audio Dongle State Transitions

Signal	State Transition Description
1	Controlled by Watchdog wake up signal
2	Call Setup packet not received from Dongle within time limit
3	Call Setup packet received from Dongle
4	
5	Call Clear Key (SW4) at Headset pressed, or Call Clear Signal received from Dongle
6	No valid packets received from Dongle within time limit
7	Call Setup packet received from Dongle
8	Call Setup packet not received from Dongle within time limit

Table 3. Headset State Transitions

3.1 Call Idle State

In ‘Call Idle’ state the Dongle will continuously monitor the call setup criteria. The USB Dongle uses the LEDN signal at pin 14 on the USB Controller chip SN11220ACF to detect when there is voice data activity. The Audio Dongle uses the call setup key (SW2). When the call setup criteria is detected true the Dongle switches state to ‘Call Setup’ state.

In ‘Call Idle’ state the Headset is in power down sleep mode due to battery conservation. The watchdog timer in the microcontroller will wake the system at regular intervals and the ‘Call Setup’ state is entered to check if the Dongle is trying to set up a call.

3.2 Call Setup State

In ‘Call Setup’ state the Dongle transmits Call Setup packets as defined in Figure 3. Signaling Packet. Packets are transmitted at four fixed frequencies distributed throughout the entire frequency band. After each Call Setup packet transmission, the Dongle tries to receive a Call Setup Acknowledge packet from the Headset as shown in Figure 4. RF Protocol - Signaling Format. The Dongle will continue to transmit Call setup packets until a Call setup Acknowledge packet is received or the call setup criteria is detected false.

The Headset will try to receive a Call setup packet at each of the four predefined frequencies. It will stay tuned to one frequency sufficiently long to receive the Call Setup packet if transmitted by the Dongle. If no Call Setup packet is received at any of the frequencies, the Headset will resume ‘Call Idle’ state.

3.3 Call Connected State

In the ‘Call Connected’ state, the Audio Frame format is used as described in chapter 4.2 Audio Format. The audio interface circuits are powered up and the Dongle starts to transmit Audio packets. Each packet is transmitted twice at two different frequencies to increase packet throughput probability. The Headset is initially listening alternately at both frequencies long enough to receive the packet regardless of the timing differences between the Dongle and Headset. When the Headset receives a valid Audio packet an internal timer is set to the correct timing instant such that all subsequent transmissions and receptions of Audio packets between the Dongle and Headset occurs at fixed timing instants referred to the Audio Frame format as shown in Figure 6. Audio Frame Format.

Each Audio packet contains in addition to audio data, two byte of signaling information. The Dongle uses one byte to transfer the frequency difference between the two frequencies used in the Audio Frame format. When the Headset receives an Audio packet it can then calculate the frequency used for the other Audio

packet as further explained in chapter 4.3 Frequency Hopping. The Dongle uses the other signaling byte for call control. The Headset uses one signaling byte to transfer the state of each of the five keys to the Dongle every Audio Frame (3.75 ms). The other signaling byte are used to inform the Dongle of the status of the packet reception of the previous frame such that the frequency hopping scheme can be based on both the Dongle and Headset reception statistics.

If the Dongle or Headset receives no valid Audio packets within a certain time interval, it will switch state to the ‘Call Re-connect’ state.

If the Dongle detects that the call setup criteria is no longer valid, it will transmit a call clear message in the call control byte and then resume to ‘Call Idle’ state. The Headset will resume the ‘Call Idle’ state as soon as the call clear message is received.

3.4 Call Re-connect State

The ‘Call Re-connect’ state is very similar to the ‘Call setup’ state. The main difference for the Dongle is that the audio interface is maintained in this state. For the Headset the main difference is that the number of attempts to receive a Call setup packet before resuming to ‘Call Idle’ state is significantly increased.

4 RF Protocol

The ‘Air Interface’ is based on the following two communication formats.

- Signaling Format
- Audio Format

The Signaling Format is used in the ‘Call Setup’ and ‘Call Re-connect’ states, and the Audio Format is used in the ‘Call Connected’ state.

A 3-byte address and 2-byte CRC are used for both formats. The Dongle and Headset are paired in factory and the 3-byte address must be loaded into the EEPROM at address 0.

4.1 Signaling Format

The Signaling packet format is shown in Figure 3. Signaling Packet. The 3-byte payload for a Call Setup packet and Call Setup Acknowledge packet is defined in Table 4. Payload Definition, Call Setup packet and Table 5. Payload Definition, Call Setup Acknowledge packet.

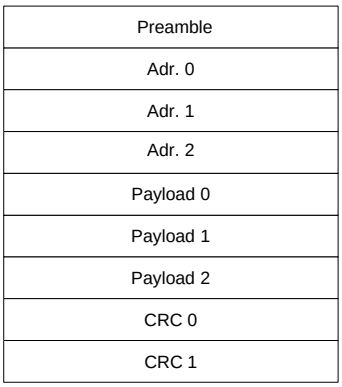

Figure 3. Signaling Packet

Payload Byte	Description
0	Call Setup packet code
1	Frequency 0, lower frequency to be used in Call Connected state
2	Frequency 1, higher frequency to be used in Call Connected state

Table 4.Payload Definition, Call Setup packet

Payload Byte	Description
0	Call Setup Acknowledge packet code
1	Call Setup Acknowledge packet code
2	Spare

Table 5. Payload Definition, Call Setup Acknowledge packet

The Dongle will alternately transmit a Call Setup packet and try to receive a Call Setup Acknowledge packet at each frequency as shown in Figure 4. RF Protocol - Signaling Format. The Headset is unsynchronized to the Dongle transmissions and must stay tuned to each frequency the time the Dongle uses to transmit at all frequencies. When the Headset receives a Call Setup packet it will immediately transmit a Call Setup Acknowledge packet.

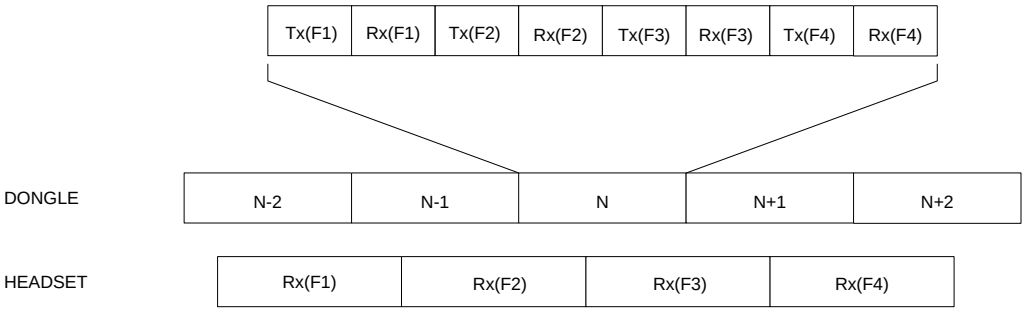

Figure 4. RF Protocol - Signaling Format

4.2 Audio Format

In the 'Call Connected' state all RF transmissions are using the Audio Format with a 32-byte payload as shown in Figure 5. Audio Packet. The payload consists of 2-byte signaling information and 30-byte audio data as defined in Table 6. Payload Definition, Audio Packet from Dongle and Table 7. Payload Definition, Audio Packet from Headset.

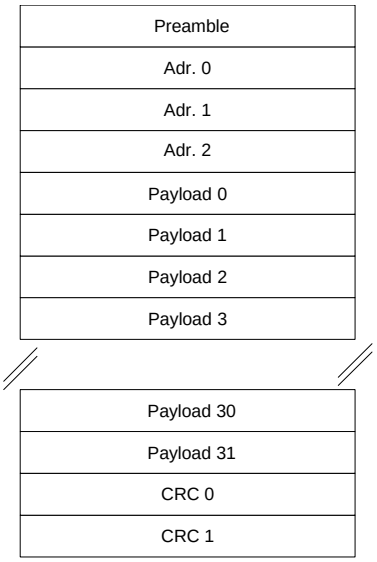

Figure 5. Audio Packet

Payload Byte	Description
0	Call Control Byte
1	Frequency difference, Frequency 1 – Frequency 0
2 - 31	Audio data, 8-bit A-law encoded

Table 6. Payload Definition, Audio Packet from Dongle

Payload Byte	Description
0	Packet Errors in previous frame
1	Key code
2 - 31	Audio data, 8-bit A-law encoded

Table 7.Payload Definition, Audio Packet from Headset

The Dongle starts to transmit and receive Audio packets using two different frequencies as shown in Figure 6. Audio Frame Format. The Headset will initially be unsynchronized to the Dongle transmissions and must stay tuned to each frequency time the Dongle uses to transmit at both frequencies. As soon as the Headset receives an Audio packet it is synchronized to the Dongle Audio Frame timing and an internal timer sets the event time for all subsequent transmissions and receptions. The Headset adjusts this timer at each Audio packet reception to compensate for clock differences in the Dongle and Headset.

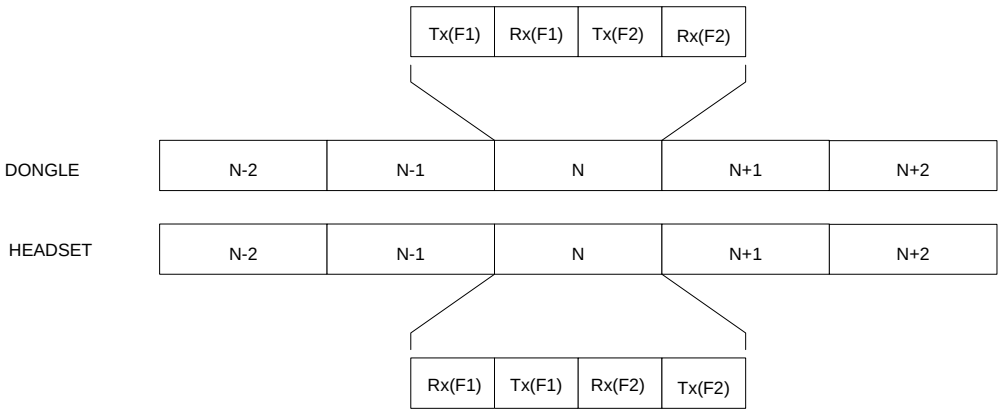

Figure 6. Audio Frame Format

4.3 Frequency Hopping

The frequency hopping is fully controlled by the Dongle. The two frequencies being used when a call is successfully cleared is stored in EEPROM and will be used as start frequencies at the next call. The Dongle is monitoring the packet error rate at both the Dongle and Headset receiver and maintains a ‘Packet Loss’ quantity for each frequency. The calculation is based on incrementing a number if the packet was lost on one or both sides and decrementing a number if the packet was successfully received at both sides. By weighting the increment and decrement numbers, the ‘Packet Loss’ quantity is either an increasing or decreasing number at a given bit error rate. If the number increases to a given threshold, the frequency is considered to be disturbed and the Dongle starts using another frequency. The Headset will be informed by this frequency change using the signaling bytes in the Audio packet.

5 Audio Interface

The Audio Dongle and Headset interfaces the Semtech audio codec XE3005. The audio codec is used in Slave mode with Long Frame Synchronization format. The MCU generates the FSYNC signal by dividing the MCU clock, and each transition of the FSYNC signal generates a MCU interrupt. The interrupt routine transfers 2 bytes of data to/from the input/output audio buffers using the SPI interface.

The USB Dongle interfaces the Sonix USB Controller SN11220ACF. The USB controller operates in Slave mode using a proprietary frame format. The USB controller outputs a 2,048MHz clock signal synchronized to the USB audio stream. The MCU generates the FSYNC signal by dividing the 2,048MHz clock, and each transition of the FSYNC signal generates a MCU interrupt. The interrupt routine transfers 1 byte of data to/from the input/output audio buffers using the SPI interface.