# Si4356 Standalone Receiver Development Kit User's Guide

## 1. Overview

Thank you for your interest in Silicon Laboratories EZRadio® Si4356 Standalone Receiver Development Kit (P/N 4356-LEDK1W-XXX). The kit contains everything you need to familiarize yourself with and evaluate the EZRadio® Si4356 standalone direct mode receiver. The kit has two versions: one for the 434 MHz band and one for the 868 MHz band. The key features of the development kit are as follows:

1. Transmitter node is a preprogrammed demo keyfob with Si4010 Transmitter SOC.
2. Receiver node is a switch configurable Si4356 demo board.
3. The receiver demo board has four LEDs to display information and four push-buttons to receive user commands.
4. The Silabs website provides all the documentation and files needed to develop a user application.
5. The kit supports the use of the Silicon Laboratories Integrated Development Environment (IDE) for software debugging and the use of the Keil & SDCC C compiler, assembler, and linker tool chain.

The demo boards come with a direct mode reception demo application. The demo implements deglitching, retiming, and decoding of the received data stream.

**Table 1. Kit Contents**

| Qty | Description | Part Number |
|---|---|---|
| | **Si4356 Standalone Receiver Development Kit 434 MHz** | **4356-LEDK1W-434** |
| 1 | Si4356 EZRadio RFStick receiver board 434 MHz with Direct Receive FW | 4356-LED-434-DIR |
| 1 | Si4010 Keyfob Board with Direct Transmit FW 434 MHz | 4010-KFOB-434-DIR |
| 1 | Keyfob Plastic Case (gray) | MSC-PLPB_1 |
| 1 | CR2032 3 V coin battery | CRD2032 |
| 2 | AAA alkaline battery | AAA |
| 1 | Toolstick Base Adapter | Toolstick BA |
| 1 | USB extender cable (USBA–USBA) | USB-XTEN-01 |
| 1 | Cardboard Box | S-KITBOX-03 |
| | **Si4356 Standalone Receiver Development Kit 868 MHz** | **4356-LEDK1W-868** |
| 1 | Si4356 EZRadio RFStick receiver board 868MHz with Direct Receive FW | 4356-LED-868-DIR |
| 1 | Si4010 Keyfob Board with Direct Transmit FW 868 MHz | 4010-KFOB-868-DIR |
| 1 | Keyfob Plastic Case (red) | MSC-PLPB_2 |
| 1 | CR2032 3 V coin battery | CRD2032 |

**Table 1. Kit Contents (Continued)**

| Qty | Description | Part Number |
|---|---|---|
| 2 | AAA alkaline battery | AAA |
| 1 | Toolstick Base Adapter | Toolstick BA |
| 1 | USB extender cable (USBA–USBA) | USB-XTEN-01 |
| 1 | Cardboard Box | S-KITBOX-03 |

## 2. Software Setup

Download the 8-bit software from the website (http://www.silabs.com/products/mcu/Pages/8-bit-microcontroller-software.aspx). At a minimum, the Si4356 development kit requires:

- **Silicon Labs IDE**—The Silicon Labs Integrated Development Environment (IDE) is a complete, stand-alone software program that includes a project manager, source editor, source-level debugger and other utilities. The IDE supports the entire Silicon Labs 8-bit microcontroller (MCU) portfolio.
- **Unlimited Keil® PK51 Professional Developer's Kit**—Keil 8051 C Compiler/Assembler/Linker toolchain. Register for free and receive the key to unlock your PK51 tools.

To install the selected components, download and run their installer exe files and follow the instructions on the screen.

### 2.1. Registering the Keil Toolset

The development kit includes the latest version of the C51 Keil 8051 toolset. This toolset is initially limited to a code size of 2 kB and programs start at code address 0x0800. After registration, the code size limit is removed entirely and programs will start at code address 0x0000.

To register the Keil toolset:

1. Register on the Silicon Labs website (http://pages.silabs.com/lp-keil-pk51.html) to obtain the free key (Product Serial Number).
2. Open the Keil µVision4 IDE from the installation directory with administrative privileges.
3. Select File → License Management to open the License Management window.
4. Click on the Get LIC via Internet... button to open the Obtaining a License IDE Code (LIC) window.
5. Press OK to open a browser window to the Keil website. If the window doesn't open, navigate to www.keil.com/license/install.htm.
6. Enter the Product Serial Number along with any additional required information.
7. Once the form is complete, click the Submit button. An email will be sent to the provided email address with the License ID Code.
8. Copy the License ID Code (LIC) from the email.
9. Paste the LIC into the New License ID Code (LIC) text box at the bottom of the License Management window in µVision4.
10. Press the Add LIC button. The window should now list the PK51 Professional Developer's Kit for Silabs as a licensed product.
11. Click the Close button.

# 3. Hardware Setup

After checking the completeness of the kit according to the kit content Table 1 on page 1, the kit can be put in operation by following this step-by-step guide.

## 3.1. Keyfob Assembly

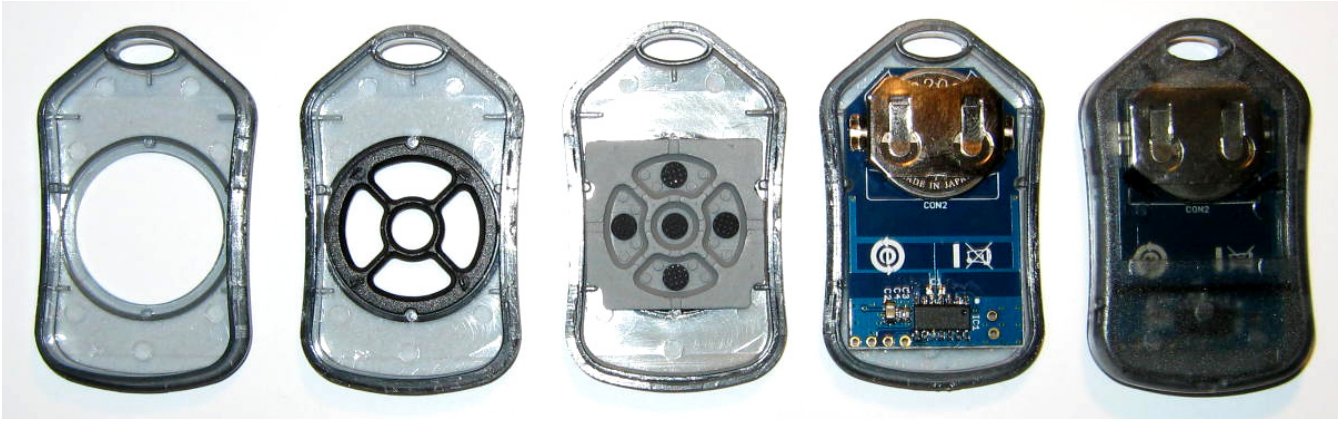


**Figure 1. Keyfob Assembly**

1. Snap the keypad frame in the top half of the keyfob case from inside.
2. Insert the rubber keypad in the frame from inside.
3. Insert the button cell battery in the battery holder on the keyfob PCB. Minus pole should face the PCB.
4. Insert the pcb in the top half of the case. Key contact side should face the rubber keys.
5. Snap on the back half of the case.
6. Check the operation. When any button is pushed on the keyfob, the LED has to flash four times.

## 3.2. RFStick Setup

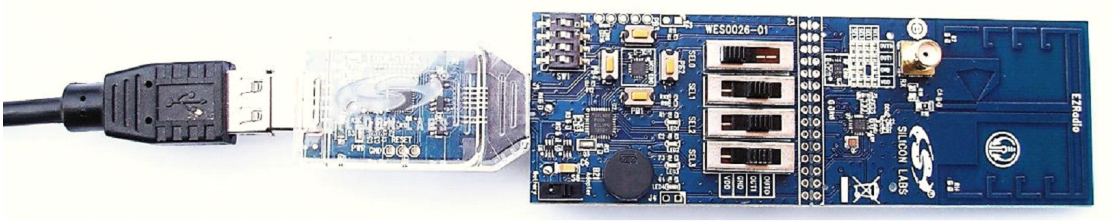


**Figure 2. RFStick Setup**

1. Set the S6 power switch to “Adapter” position.
2. Connect the RFStick to the Toolstick Base Adapter via the PCB edge connector.
3. Connect the Base Adapter to a USB port of a PC directly or using the USB extender cable.
4. If necessary wait for Windows to install the debug interface driver.
5. RFSticks can be shipped without preloaded software, or the software on the board can be out of date. So upon first use, it is advised to upgrade the firmware in the MCU of the RFStick. The complete Si4356 Development Kit RX side code project can be downloaded from http://www.silabs.com/Support%20Documents/Software/4356_rfstick_dir_SW.zip.
6. Start the Silabs IDE and close any open project.
7. Before connecting to the target device, several connection options may need to be set. Open the Connections Options window by selecting Options→Connection Options...in the IDE menu. First select the

USB Debug Adapter in the “Serial Adapter” section. Next, the C2 Debug Interface must be selected. Once all the selections are made, click the OK button to close the window.

8. Click the Connect button in the toolbar or select Debug→Connect from the menu to connect to the device.
9. Download the project to the target by clicking the Download Code button in the toolbar and browsing to the 4356-rfstick-dir.hex file of the out folder of the Si4356 Development Kit RX side code project.
10. Click the Disconnect button in the toolbar to start the operation of the downloaded example code.

To operate the RFStick without USB connection, insert two AAA batteries in the battery holder on the bottom side of the board and set the power switch to “Battery” position.

## 4. Operating the Direct Mode Reception Demo

The Si4010 keyfob transmitter and the Si4356 RFStick receiver are the transmitter and receiver nodes used in the direct mode reception demo. After setting up the nodes according to the previous chapter, the Si4356 has to be configured for the RF parameters of the demo, using the SEL0 – SEL3 slide switches and the SW1-1 to SW1-4 DIP switches on the RFStick. SEL0 and SEL1 set the center frequency according to Table 2. The part number of the board contains information on the frequency band (434 or 868) that the matching circuit of the board was designed for. Using the board in a different band results in significantly reduced sensitivity.

**Table 2. Center Frequency Setting**

| Kit Part Number | Center Freq. | SEL0 | SEL1 |
|---|---|---|---|
| 4356-LEDK1W-434 | 433.92 MHz | GND | VDD |
| 4356-LEDK1W-868 | 868.30 MHz | VDD | OUT1 |

The buttons on the keyfob have been programmed to transmit various packet formats. Details on the RF settings and packet formats associated with each button are provided in Section 5. The Si4356 RFStick will only respond to keyfob button transmissions that it has been configured to receive. The slide and DIP switches on the Si4356 RFStick are used to configure the Si4356 IC. Table 3 and Table 4 highlight the required configurations for each keyfob button. Note that the keyfob button names have been defined according to their position when the keyfob LED is oriented on top.

**Table 3. Si4356 RFStick Configurations for Si4356-LEDK1W-434 Kit**

| Key Fob Button | SEL2 | SEL3 | SW1-1 | SW1-2 | SW1-3 | SW1-4 | Valid Packet RX |
|---|---|---|---|---|---|---|---|
| Right | OUT1 | OUT1 | OFF | OFF | OFF | OFF | LED2 and 3 |
| Left | OUT1 | OUT1 | OFF | OFF | OFF | OFF | LED2, 3 and 4 |
| Middle | OUT1 | GND | ON | OFF | OFF | OFF | LED2 and 4 |
| Bottom | GND | GND | ON | ON | OFF | OFF | LED4 |
| Top | GND | GND | ON | ON | OFF | OFF | LED3 |
| **Note:** The slide and DIP switches MUST be configured prior to powering up the board. | | | | | | | |

**Table 4. Si4356 RFStick Configurations for Si4356-LEDK1W-868 Kit**

| Key Fob Button | SEL2 | SEL3 | SW1-1 | SW1-2 | SW1-3 | SW1-4 | Valid Packet RX |
|---|---|---|---|---|---|---|---|
| Right | GND | OUT0 | OFF | OFF | OFF | OFF | LED2 and 3 |
| Left | GND | OUT0 | OFF | OFF | OFF | OFF | LED2, 3, and 4 |
| Middle | OUT1 | GND | ON | OFF | OFF | OFF | LED2 and 4 |
| Bottom | GND | GND | ON | ON | OFF | OFF | LED4 |
| Top | GND | GND | ON | ON | OFF | OFF | LED3 |
| **Note:** The slide and DIP switches MUST be configured prior to powering up the board. | | | | | | | |

After turning on the demo the first time, it needs to be paired with a keyfob. LED1, LED2, LED3, and LED4 will begin blinking after the device has been initialized, showing that it is waiting for association. The Si4356 RFStick will associate with the keyfob that sends the first valid packet. If a proper packet is received, the receiver node stores the unique ID of the keyfob into its MCU's RAM memory and will stop flashing the four LEDs (indicating a successful association) and will enter receive mode. In receive mode, LED1 will be continuously on. The Si4356 will accept packets from the associated keyfob and flash the LEDs listed in the "Valid Packet RX" column of Table 3 and Table 4 when receiving a valid packet. Pressing the PB1 button will clear the associated keyfob and cause the Si4356 RFStick to wait for a new association.

## 5. Transmitter Node

**Figure 3. Demo Keyfob**

The Si4010 transmitter SOC in the keyfob of the kit is factory programmed with the 4010-kfob-dir program. Since the program is burned in the NVM (OTP) memory of the Si4010, it cannot be changed.

For keyfob development with the Si4010 transmitter SOC, it is recommended to use the **Si4010/Si4355 Development Kit (P/N 4010-KFOBDEV-xxx),** available from Silabs. The complete 4010-kfob-dir software project is available in the example programs package of the Si4010 product webpage.

The schematic of the keyfob can be found at the end of the document.

The keyfob transmits an RF packet four times each time a button is pressed, and it also blinks the LED on the keyfob. The transmitted packet is different for each button, according to Table 5. Note that the keyfob button names have been defined according to their position when the keyfob LED is on top.

**Table 5. Packet Configurations**

| Keyfob Button | Packet Type | Modulation |
|---|---|---|
| Right button | Normal RF packet | FSK |
| Middle button | Normal RF packet | FSK |
| Left button | Normal RF packet | OOK/MANCH |
| Bottom button | Legacy encoder packet | OOK |
| Top button | Legacy encoder packet | OOK |

## 5.1. RF Parameters

The demo kit uses the following RF parameters according to the kit's frequency band.

**Table 6. RF Parameters of the Demo**

| Kit Part Number | Center Freq. | FSK Deviation | Symbol Rate |
|---|---|---|---|
| **4356-LEDK1W-434** | 433.92 MHz | ±59 khz | 9.6 kBd |
| **4356-LEDK1W-868** | 868.30 MHz | ±119 khz | 9.6 kBd |

## 5.2. Normal RF Packet

This packet is for demonstrating on the receiver side the reception of a packet that is optimized for RF communication. It consists of 16 bytes. The packet structure is as follows:

**Table 7. Normal RF packet structure**

| Number of Bytes | Field Name | Description |
|---|---|---|
| 5 | Preamble | 0xaa |
| 2 | Sync | 0x2d, 0xd4 |
| 4 | Chip ID | Unique, factory burnt chip ID of the Si4010 |
| 1 | Status | Lower 5 bits are the button information |
| 2 | Packet count | Rolling counter for PER measurement |
| 2 | CRC-16 | Generator X16+X15+X2+1, start value 0xFFFF |

### 5.3. Legacy Encoder Packet

This packet is for demonstrating on the receiver side the reception of legacy packets that were not optimized for RF communication. The packet structure follows the structure of the bit stream generated by a Holtek HT6P20 encoder chip. The figure below describes the packet structure. The only difference is that this demo implementation uses 22 address bits and 2 data bits. The address bits are the 22 LSb bits of the four byte factory programmed unique ID of the Si4010. The data bits reflect which button was pressed. The bit structure can be observed in the fixed anti-code pattern. One bit takes three clock cycles. The nominal value of fosc is 3 kHz.

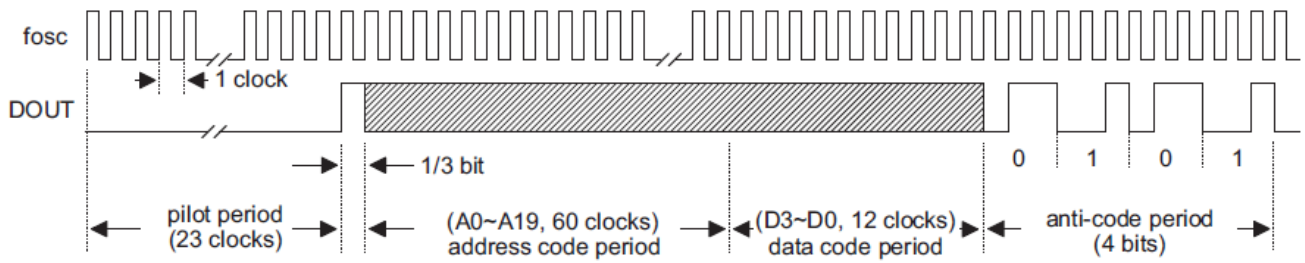



**Figure 4. A Complete Code word of the HT6P20D**

## 6. Receiver Node

The receiver node is based on the EZRadio Si4356 standalone sub-GHz receiver IC. Standalone means that the Si4356 is pin-strap configurable; there is no need for an external MCU to configure and control the receiver. Integrated configuration tables allow the Si4356 to be completely configured using the four selector pins. The state of each of these pins is read internally at startup and used to determine which pre-loaded configuration should be used. After loading the configuration, the receiver automatically starts reception. The receiver demodulates the incoming data asynchronously by oversampling the incoming transmission. The resulting demodulated signal is output to the system MCU through a data output pin.

### 6.1. The RFStick platform

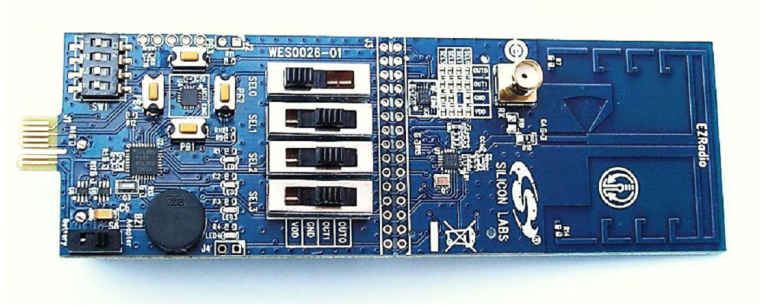



**Figure 5. EZRadio RFStick Receiver Board (P/N 4356-LED-434-DIR)**

The RFStick is a basic demo system for the evaluation of the EZradio chips. The board has two main parts, the MCU part and the radio part. The MCU part of the board contains a Silabs C8051F930 MCU and basic human interface devices (four push-buttons, four LEDs, four switches and a buzzer). Additionally the MCU part of the Si4356 RFStick has four four-position switches (SEL0-SEL3) to configure the radio. The radio part contains the EZRadio chip, the matching circuit and the antenna. The RF input is selectable via a 0 Ω resistor between a PCB antenna and a 50 Ω SMA input connector. The PCB antenna is connected by default. The MCU is connected to the Si4356 EZRadio chip via four GPIOs according to the Table 8. The RF section of the board can be broken off along a perforation and installed in the user's own hardware as a radio module. In this case the radio can be configured via solderable jumpers on the RF section. These solderable jumpers have to be left open when the RF

part is not detached from the MCU part, to avoid conflict with the configuration switches.

**Table 8. Connections Between the Si4356 and the MCU**

| Si4356 | | | RFStick | | C80C51F930 |
|---|---|---|---|---|---|
| **Pin Number** | **Pin Name** | **Pin Function** | **J3 Pins Connector** | **Signal Name** | **Pin Name** |
| EP, 1, 6, 9, 16 | GND | Ground | 3–4 | GND | GND |
| 7, 8 | VDD | Supply Voltage input | 1–2 | VDD | VDD |
| 12 | MSTAT/OUT0 | Mode status/Config output | 19–20 | MSTAT/OUT0 | P1.0 |
| 14 | RX_DATA/OUT1 | Received data/Config output | 15–16 | MCU_RXDATA/OUT1 | P1.1 |
| 13 | STBY | Standby mode toggle input | 13–14 | MCU_STDBY | P1.2 |
| 2 | RST | Reset input, active high | 5–6 | MCU_RST | P1.5 |
| 10 | CLK_OUT | System reference clock output | 7 -x- 8* | MCU_GPIO_3 | P0.3/XTAL2 |
| 11 | SEL0 | Configuration selector input | 11–12 | SEL0 | — |
| 15 | SEL1 | Configuration selector input | 17–18 | SEL1 | — |
| 19 | SEL2 | Configuration selector input | 23–24 | SEL2 | — |
| 20 | SEL3 | Configuration selector input | 21–22 | SEL3 | — |

***Note:** The reference clock output of the Si4356 is not connected to the MCU by default (represented by -x- in the table above). The user can connect it by soldering in a jumper across pin7 and pin8 of J3.

Power source of the board can be selected with the power-supply selector switch (S6). If S6 is in ‘Adapter’ position, supply voltage is provided by the Toolstick Base Adapter that is connected to the J1 PCB edge connector. If S6 is in ‘Battery’ position, the supply voltage is provided by two AAA batteries in the battery holder on the bottom side of the board. Current consumption of the RF part (RFVDD) can be measured on J6. Since J6 is shorted by a PCB track on the bottom side of the board, the user have to cut the track if this feature is used. The board also contains two board identification memories (U3 and U4) to provide information for the WDS about the connected boards through the debug interface.

Schematics can be found at the end of the document.

## 6.2. Operation of the Receiver Demo Code

The 8051F930 MCU on the Si4356 RFStick receiver board is factory programmed with the 4356-rfstick-dir demo program. Since the program is programmed in flash memory, it can be overwritten using the Silicon Labs Toolstick Base Adapter that is part of the development kit (P/N Toolstick_BA). The complete 4356-rfstick-dir software project can be found on the Si4356 product page.

Upon power up the demo code reads the state of the four DIP switches to determine which type of packet has to be received. Later changes of switch settings have no effect.

The Si4356 receives and demodulates the transmission of the keyfob and outputs the raw data signal to the F930 MCU. First the receiver program deglitches and retimes the received signal to improve sensitivity and evaluates the received packets. If the packet is OK, the program flashes the LEDs according to the button information in the status byte of the received packet.

### 6.2.1. Deglitching

The Si4356 demodulates the incoming data asynchronously by oversampling the incoming transmission. This allows for faster acquisition as well as tracking of non-standard preamble or synchronization patterns. Since the received data is sampled by a high sampling clock, glitches may be present as the signal level approaches the sensitivity limit. The widths of the glitches are one or two clock cycles of the oversampling clock. A deglitching algorithm can be run on the host microcontroller to remove these glitches.

The chosen algorithm is Median Filtering that is typically used to filter spikes. In this algorithm, the elements are sorted and the element in the middle of the window is selected as the output. Since there are only two logical states (0 or 1) it is enough to count the number of 1s in a window. If this number is greater than the half size of the window, the output will be logic 1, otherwise a logic 0.

For example:

01011 sorting 00111 the median (element in the middle) is 1, so the output is 1. The number of the 1s is three, which is greater than the half size of the window (2.5), so the output is 1.
00100 sorting 00001 the median is 0, the output is 0. The number of 1s is 1, which is less than 2.5, so the output is 0.

The data output of the radio is sampled by Timer3 interrupt service routine. The sampling rate is chosen as an integer multiple (5-10 times) of the expected data rate. The deglitching and retiming is made bit by bit in the same ISR. In the algorithm, a running sum of 1's is maintained. The algorithm is updated every RX_Clk edge. The bitmask variable is used to set a mask for the MSB bit of the shift register. For example, with windowsize=5, the bitmask is 10000 (binary). Then it is checked if the MSB is 1 or 0. When the data is shifted, the MSB will leave the window. If the MSB was 1, then the number of 1s (which is stored in the count variable) is decreased.

```
if ( wShiftreg & wBitMask )
        {
              bOneSampleCount--;
        }
```

Here we shift the shift register:

```
wShiftreg <<= 1;
```

If the incoming data is 1, the LSB of the shift register is set to1. Also the count is increased by 1. If the RX_DATA is 0, shift register is unaltered i.e. 0 enters the shift register.

```
if ( MCU_RX_DATA )
      {
            wShiftreg++;
            bOneSampleCount++;
}
```

The algorithm also allows for a selection of window sizes from 1 to 15.

If the number of 1s (count) is greater than the half size of the window (bitlimit = windowsize / 2) the output will be 1, otherwise 0.

```
if ( bOneSampleCount > bBitLimit )
        {
                bCurrentDeglitchedSample = 1;
        }
        else
        {
                bCurrentDeglitchedSample = 0;
        }
```

### 6.2.2. Retiming

The retiming algorithm will estimate the no. of raw data bits between two data transitions. It will also account for any jitter that is on the RX Data. This is performed bit by bit in the Timer3 ISR.

In the software implementation, the data rate and sample rate are defined in the application_defs.h file. The output of the deglitching/retiming is a one-byte buffer bRetimedBitBuffer. After reception of eight bits the fRetimedBitBufferFull flag indicates that the buffer is full and can be read.

```
if (bCurrentDeglitchedSample == bPreviousDeglitchedSample )
        {
                bSampleperbitCounter--;
                if ( bSampleperbitCounter == 0)
                {
                        bSampleperbitCounter = bSamplePerBitRatio;
                        bRetimedBitShiftreg <<= 1;
                        bRetimedBitShiftreg |= bCurrentDeglitchedSample;
                        bNumberOfRetimedBits++;
                  if ( bNumberOfRetimedBits == 8 )
                  {
                                bNumberOfRetimedBits = 0;
                                bRetimedBitBuffer = bRetimedBitShiftreg;
                    fRetimedBitBufferFull = TRUE;
                        }
                 }
         }
         else//if the next sample isn't the same
         {
                bSampleperbitCounter = bSamplePerBitRatio / 2;
                bPreviousDeglitchedSample = bCurrentDeglitchedSample;
         }
```

### 6.2.3. Packet Evaluation

The output of the retiming algorithm is an input parameter to the packet handling algorithm. The packet handler performs typical packet handling functionalities such as synchronization word detection, CRC check and payload validation.

#### 6.2.3.1. Normal RF Packet

**Table 9. Normal RF Packet Structure**

| Number of Bytes | Field Name | Description |
|---|---|---|
| 5 | Preamble | 0xaa |
| 2 | Sync | 0x2d, 0xd4 |
| 4 | Chip ID | Unique, factory burnt chip ID of the Si4010 |
| 1 | Status | Lower 5 bits are the button information |
| 2 | Packet Count | Rolling counter for PER measurement |
| 2 | CRC-16 | Generator X16+X15+X2+1, start value 0xFFFF |

- **Preamble.** Since preamble is not necessary for raw data mode, this part of the normal RF packet is not used during the reception and will be skipped during the packet evaluation.
- **Sync word detection.** Packet evaluation looks for the sync pattern to find the exact position of the received packet in the received data buffer.
- **CRC check.** After synchronization, the expected position of the CRC is known. The received CRC is checked against the calculated CRC of the packet to validate packet payload content.
- **Transmitter ID check.** If the payload content is valid, the transmitted ID can be checked against the ID that was registered during association.
- **Button information display.** If the ID is accepted, button information is displayed according to Table 2 on page 4.

#### 6.2.3.2. Legacy Encoder Packet

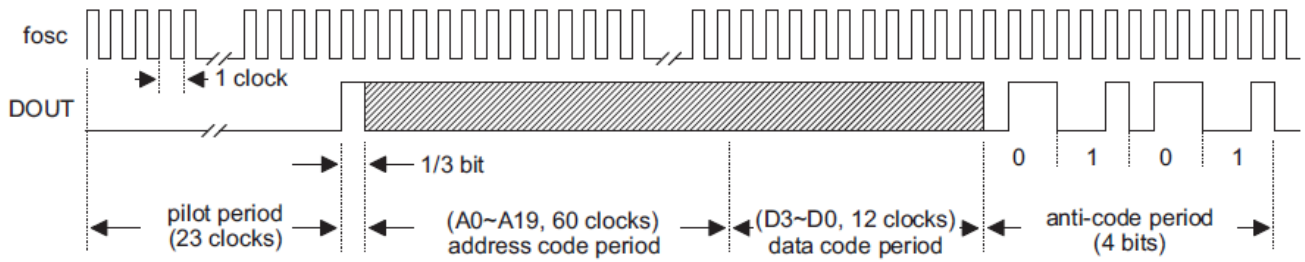



**Figure 6. A Complete Code Word of the HT6P20D**

- **Sync word detection.** Legacy packets may not contain fixed part for synchronization purposes. In our case the pilot period and start bit can be used as sync pattern (0x01) with limited sensitivity because the long zero (pilot) part is prone to glitches. Better results can be achieved by using the anti-code period as sync pattern.
- **Transmitter ID check.** In our packet 22LSb bits of the transmitter ID is used, so this can be compared to the relevant part of the stored associated ID
- **Button information display**

### 6.2.4. Association

When the receiver program is in association mode, instead of checking the ID received in the incoming packet, it stores this ID as associated ID. The two MSb bits of the ID is always set to zero to allow mixing of normal and legacy packets during association and normal reception.

# 7. Schematics

The schematics of the transmitter and receiver boards of the kit can be found on the following pages. Complete manufacturing file packs with CAD/CAM files and BOMs are available at silabs.com

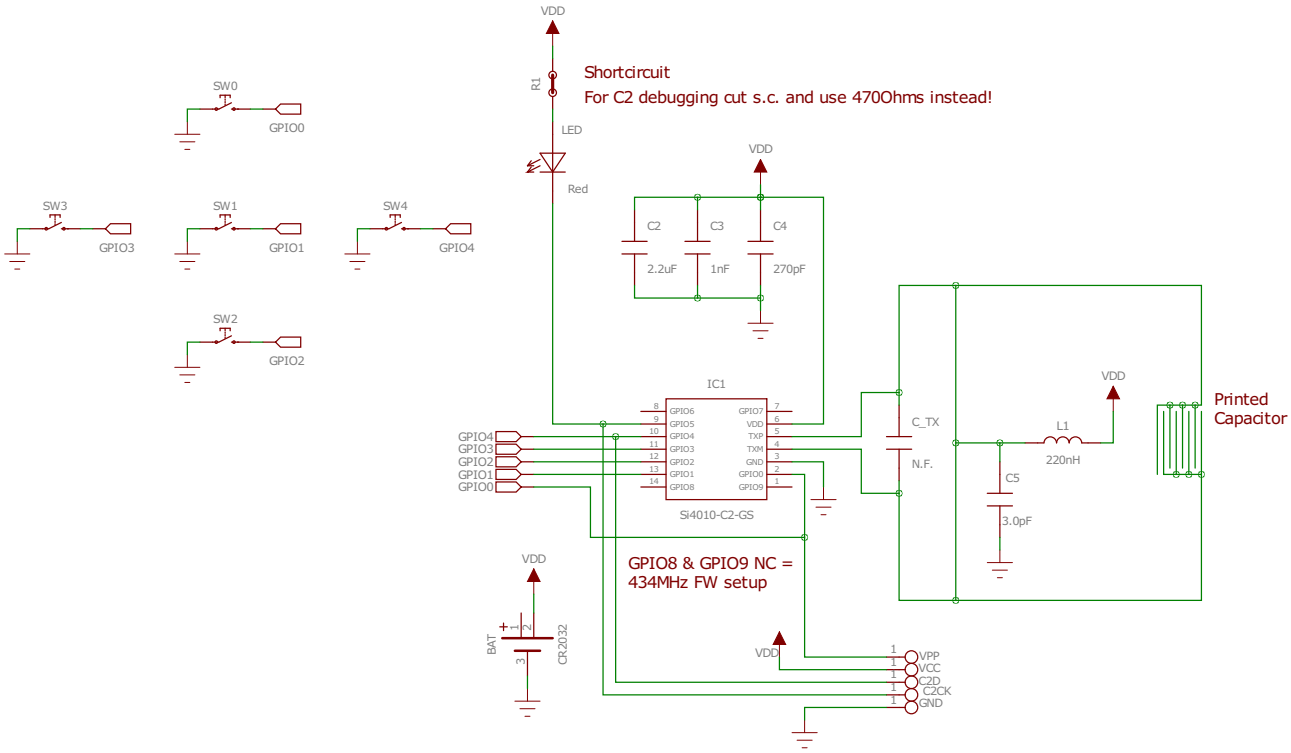



**Figure 7. Schematic of the 4010-KFOB-434 Board**

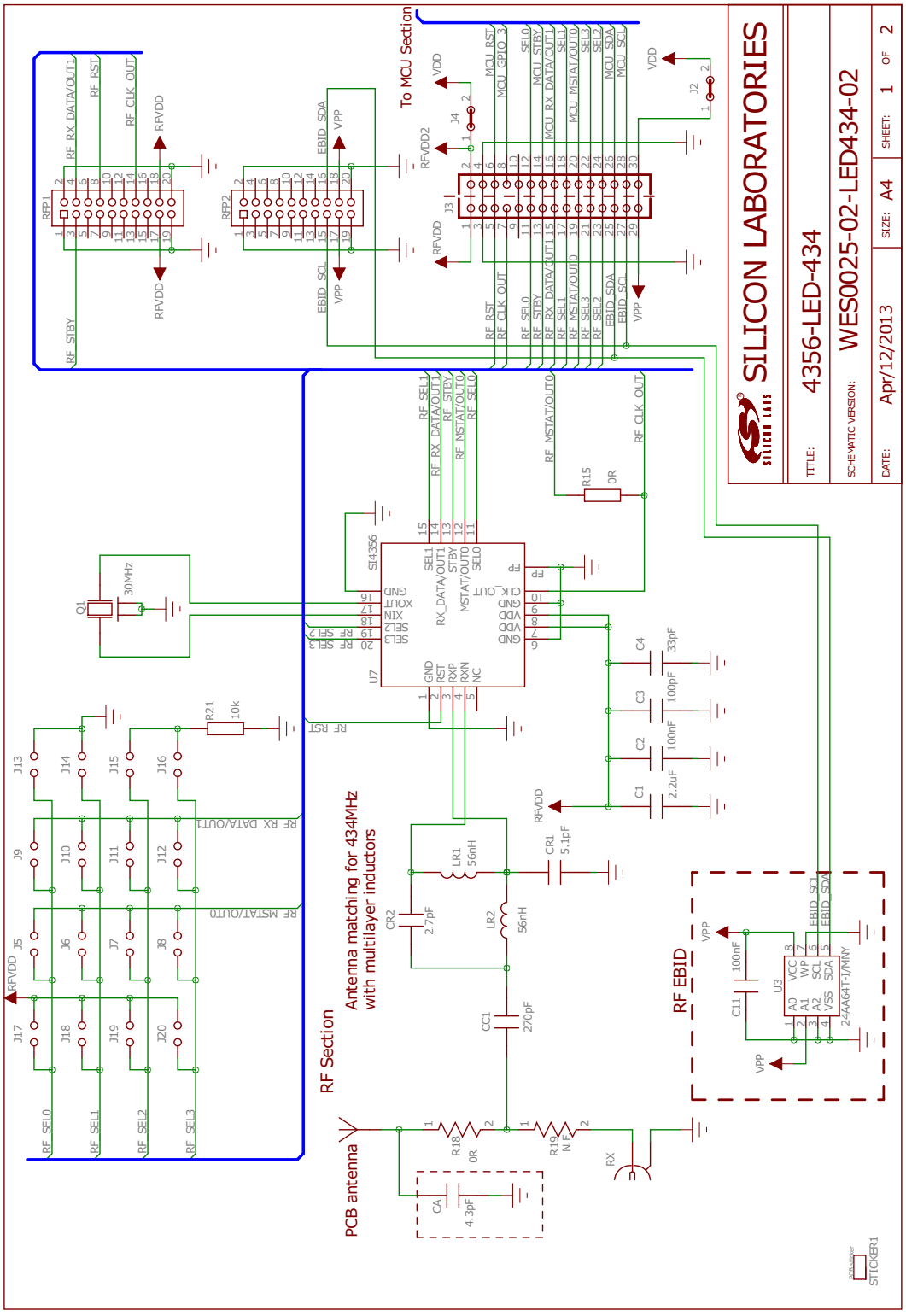



**Figure 8. Schematic of the 4356-LED-434 Board (1 of 2)**

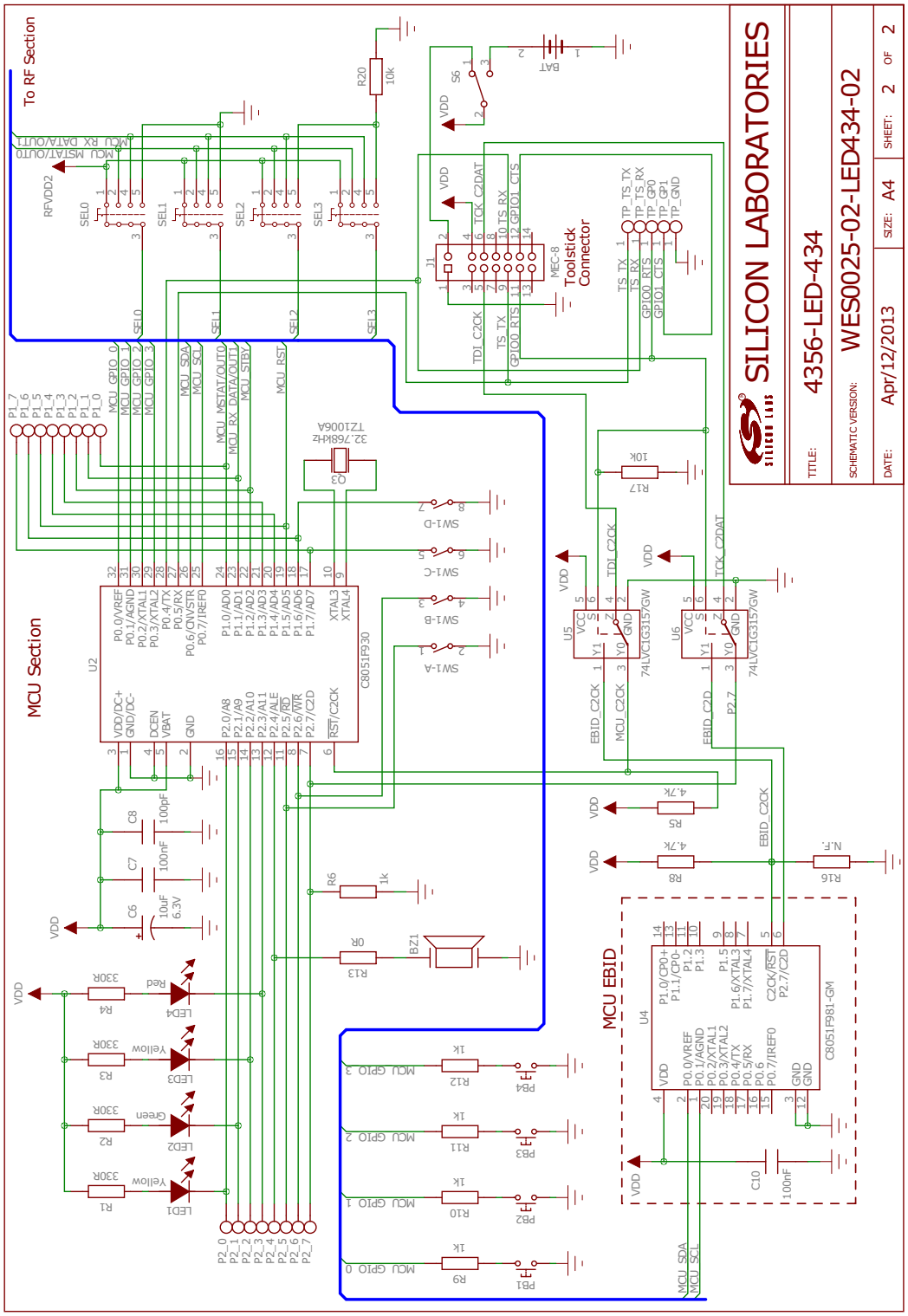
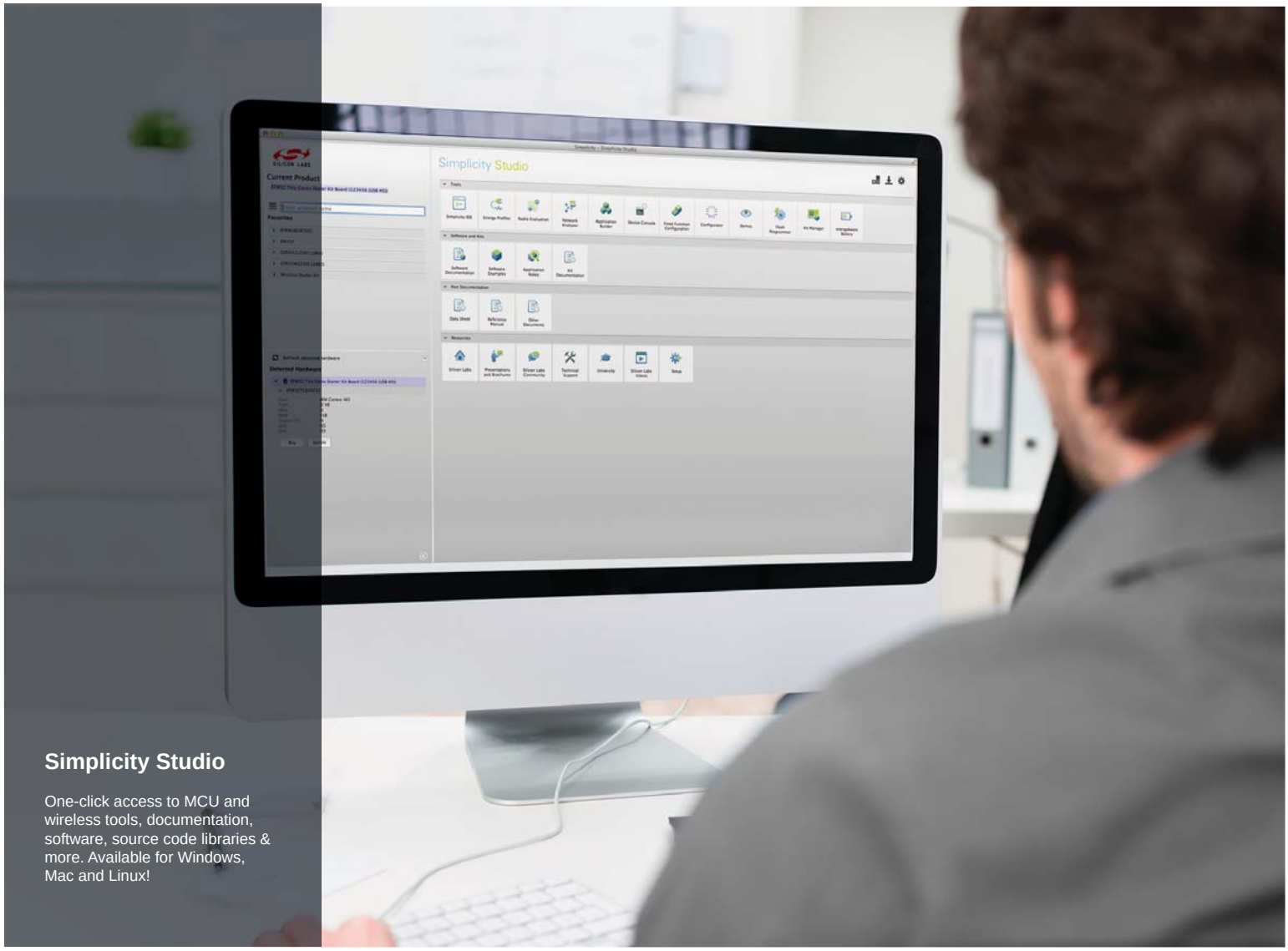



**Figure 9. Schematic of the 4356-LED-434 Board (2 of 2)**




## Simplicity Studio

One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!

**IoT Portfolio**
*www.silabs.com/IoT*

**SW/HW**
*www.silabs.com/simplicity*

**Quality**
*www.silabs.com/quality*

**Support and Community**
*community.silabs.com*



SILICON LABS

**Silicon Laboratories Inc.**
**400 West Cesar Chavez**
**Austin, TX 78701**
**USA**

**http://www.silabs.com**