A large, light blue graphic of a stylized color wheel or arc, with a small circle at the top left, serving as a background for the title.

XC800 Family

AP08113

Capacitive-Touch Color Wheel Implementation

Application Note

V1.0, 2010-08

Microcontrollers

Edition 2010-08

**Published by
Infineon Technologies AG
81726 Munich, Germany**

**© 2010 Infineon Technologies AG
All Rights Reserved.**

LEGAL DISCLAIMER

THE INFORMATION GIVEN IN THIS APPLICATION NOTE IS GIVEN AS A HINT FOR THE IMPLEMENTATION OF THE INFINEON TECHNOLOGIES COMPONENT ONLY AND SHALL NOT BE REGARDED AS ANY DESCRIPTION OR WARRANTY OF A CERTAIN FUNCTIONALITY, CONDITION OR QUALITY OF THE INFINEON TECHNOLOGIES COMPONENT. THE RECIPIENT OF THIS APPLICATION NOTE MUST VERIFY ANY FUNCTION DESCRIBED HEREIN IN THE REAL APPLICATION. INFINEON TECHNOLOGIES HEREBY DISCLAIMS ANY AND ALL WARRANTIES AND LIABILITIES OF ANY KIND (INCLUDING WITHOUT LIMITATION WARRANTIES OF NON-INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS OF ANY THIRD PARTY) WITH RESPECT TO ANY AND ALL INFORMATION GIVEN IN THIS APPLICATION NOTE.

Information

For further information on technology, delivery terms and conditions and prices, please contact the nearest Infineon Technologies Office (www.infineon.com).

Warnings

Due to technical requirements, components may contain dangerous substances. For information on the types in question, please contact the nearest Infineon Technologies Office.

Infineon Technologies components may be used in life-support devices or systems only with the express written approval of Infineon Technologies, if a failure of such components can reasonably be expected to cause the failure of that life-support device or system or to affect the safety or effectiveness of that device or system. Life support devices or systems are intended to be implanted in the human body or to support and/or maintain and sustain and/or protect human life. If they fail, it is reasonable to assume that the health of the user or other persons may be endangered.

XC822T**Revision History: V1.0 2010-08**

Previous Version(s):

Page	Subjects (major changes since last revision)
–	This is the first release ...

We Listen to Your Comments

Is there any information in this document that you feel is wrong, unclear or missing? Your feedback will help us to continuously improve the quality of this document. Please send your proposal (including a reference to this document) to:

mcdocu.comments@infineon.com



Table of Contents

1	Introduction	5
1.1	Overview	5
2	Color Wheel Features	6
3	Implementation of Color Wheel Software	7
3.1	Touch Detection Methodology (using Touch-Wheel and LEDTS ROM Library)	7
3.1.1	Touch Detection Methodology for the Centre Button	7
3.1.2	Touch Detection Methodology for the Touch-Wheel	8
3.2	Touch-Wheel Library	9
3.2.1	Calc_Touchwheel() Function	9
3.2.2	Calc_Compensation() Function	9
3.3	RGB LED	10
3.4	Overview Flow Chart of Color Wheel Software Code	11
4	Modifying Color Wheel Software Code	12
4.1	User-Defined Parameters	12
4.2	DAvE-Bench Tool	12
5	Downloading Modified Color Wheel Software Code	15
5.1	Access Code to UART BSL Mode	15
6	Implementation of Color Wheel with XC822T	17
6.1	Real-Time Clock (RTC) Module	17
6.2	Power Saving Control Module	17
6.3	CCU6 Module	17
6.4	Interrupt System Module	17
6.5	UART Module	17
6.6	LED and Touch-Sense Controller Unit Module	17
7	Schematics and Layout of the Color Wheel Board	18
8	Summary	21

1 Introduction

This application note describes the Color Wheel Application Kit, a ready-to-use implementation of the touch-sensing application to adjust light colors with a touch-wheel (**Figure 1**). The kit is built around the XC822T, a member of Infineon Technologies 8-bit microcontrollers XC800 Family, which is capable of capacitive touch detection and LED driving with color control.

The implementation comprises of a touch-wheel PCB board consisting of XC822T, the LEDs and the software code running on XC822T. This document takes the user through the steps necessary to create their own XC822T¹⁾ solution. The source code provided can be modified and compiled with the free DAVe-Bench tool chain. The user can download the code into flash, execute it and observe the touch-wheel controls in action.

Please refer to the XC82x User's Manual for detailed description of the XC822T product.

1.1 Overview

The topics covered in this application note:

- Color Wheel features
- Software:
 - Implementation details of Color Wheel software code
 - Overview flow chart
 - User-defined parameters
 - Modify and Compile code using DAVe-Bench
 - Download modified code using DAVe-Bench (XC800_FLOAD)
- Hardware:
 - On-chip modules utilized
 - Board schematic and layout

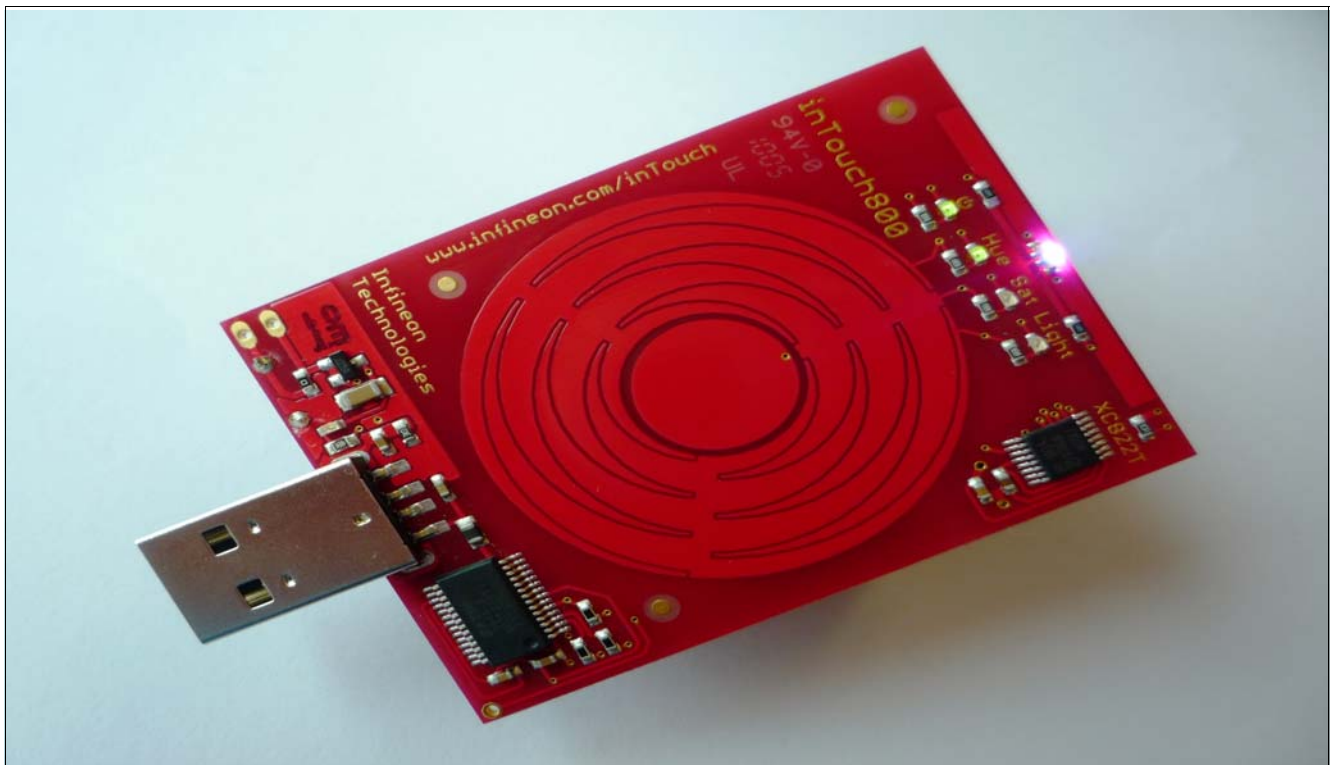


Figure 1 Color Wheel Application Kit

¹⁾ The steps and software code are also applicable for other XC82x/XC83x products that support capacitive touch control.

2 Color Wheel Features

The Color Wheel board starts automatically when the kit is plugged into computer (via USB). This is indicated by the 'On/Off' LED (see LED4 in [Figure 6](#)). When a finger moves over the touch-wheel, the software detects the absolute position of the current finger touch and the color of the RGB LED (D2) changes accordingly. The effect of the finger position depends on the selected color mode. There are three color modes available:

- Color ('Hue')
- Saturation ('Sat')
- Brightness ('Light')

The three color modes are represented by 'Hue', 'Sat' and 'Light' markings on the top of the respective LEDs as shown in [Figure 1](#) (or LED3, LED2 and LED1 respectively, as shown in [Figure 6](#)). The color modes can be selected by tapping the centre button. Successive taps on the button cycle through the available color modes, so the first tap selects 'Sat', the second selects 'Light' and the third selects 'Hue'. The respective LED above the touch-wheel will indicate the current color mode selection.

The main LED is a 3-channel RGB LED (D2), where the color is a combination of the three channels' relative RGB values, which are controlled by the PWM signals. When the Color Wheel board is in Color mode, a touch detected by the touch-wheel will adjust the color aspect. The saturation and brightness aspects remain unchanged, until the respective Saturation or Brightness mode is selected. The three modes set the HSL (Hue, Saturation, Lightness) color space, which is converted into RGB space, which is fed into the RGB LED via 3 PWM signals. The user can therefore adjust the light colors with the touch-wheel and centre button. Please refer to the [Section 3.3](#) for more details.

The Color Wheel board can be switched off with a continuous press of the centre button (~2s). The 'On/Off' LED indicates the status. When off, the touch-wheel is inactive. Press the centre button again to turn the board on. The current color mode selection is reset to 'Hue' and the RGB LED is reset to the default value.

3 Implementation of Color Wheel Software

The Color Wheel software enables some XC822T device functions, such as Real-Time Clock, Power Down Mode and LEDTSCU for example. When the Color Wheel board is turned off, it actually enters Power Down Mode. The device will wake-up every 1 second for less than 500µs, to check if a touch is detected on the centre button. If no touch is detected, Power Down Mode is re-entered. When a touch is detected, the device will initialize all parameters and jump to software execution from the main routine. The Color Wheel software flowchart is shown in [Section 3.4](#).

3.1 Touch Detection Methodology (using Touch-Wheel and LEDTS ROM Library)

The LEDTS ROM Library is enabled, with a customized trip point for touch detection threshold and an oscillation window for the 4 touch-sense inputs. The centre button uses the TSIN3 touch-sense input, while the touch-wheel is made up of 3 touch-sense inputs, TSIN0, TSIN1 and TSIN2, as shown in [Figure 2](#). The 4 touch-sense inputs are represented by bit 3-0 of the output parameters (PADFLAG, PADRESULT and PADERROR) of the LEDTS ROM Library, FINDTOUCHEDPAD function.

There are two touch detection methodologies used in the Color Wheel Application Kit:

- LEDTS ROM Library touch detection methodology
- Touch-Wheel Library (Library_TouchWheel.lib) touch detection methodology

The centre button uses the LEDTS ROM Library touch detection methodology, while the touch-wheel uses the Touch-Wheel Library touch detection methodology, as described in [Section 3.1.1](#) and [Section 3.1.2](#) respectively.

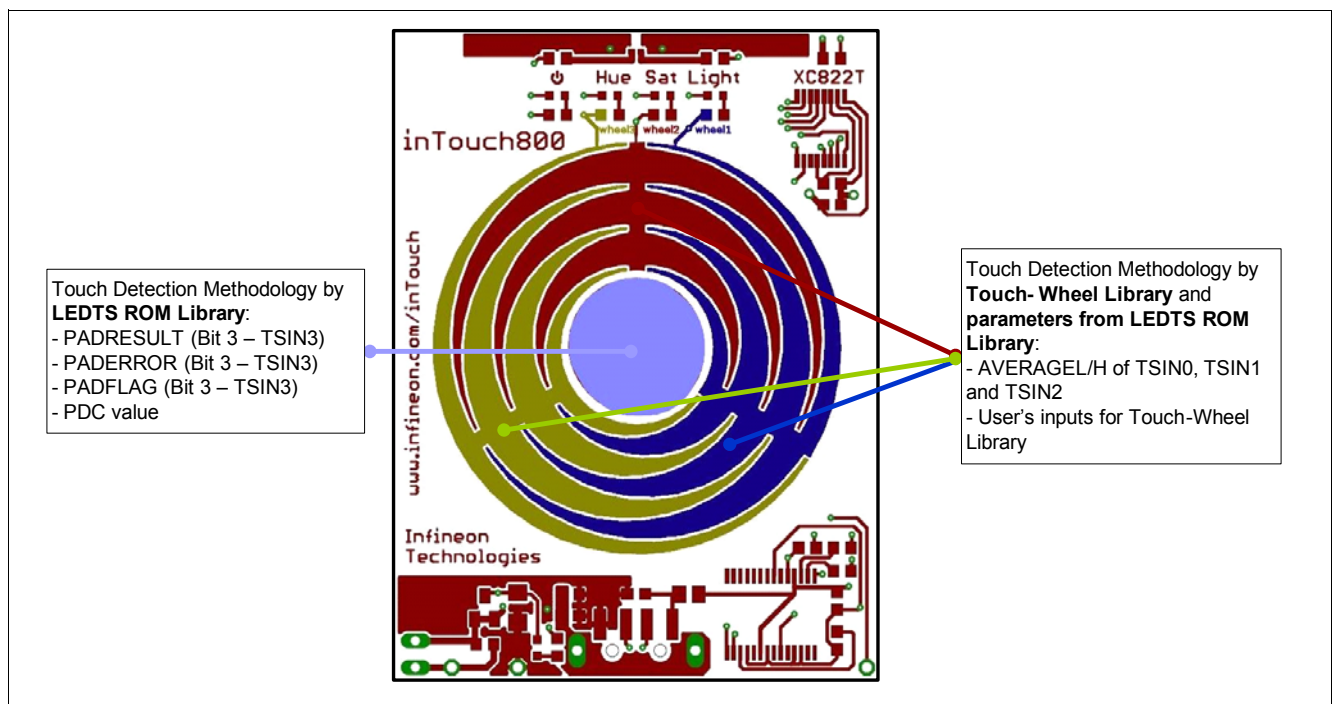


Figure 2 Four touch-sense inputs of Color Wheel Application Kit

Note: TSIN3, TSIN0, TSIN1 and TSIN2 are named WHEEL_C, wheel1 (WHEEL_1), wheel2 (WHEEL_2) and wheel3 (WHEEL_3) respectively, as shown in the diagrams in [Figure 2](#) and [Figure 7](#).

3.1.1 Touch Detection Methodology for the Centre Button

The LEDTS ROM Library is used to detect a touch on the centre button. The outputs of the FINDTOUCHEDPAD function show whether the centre pad is being touched; i.e. PADRESULT.3 = 1 indicates that the centre pad is being touched, and the color mode and its respective LED will be updated accordingly.

The pad detection period depends on many parameters, such as the number of pad turns and LED columns enabled and the accumulation of SFR LTS_TSVAl for example. The 'All Enabled Pads Accumulated Count Period' (AEPACP) is 8.96ms, as defined in [Equation \(1\)](#). The ShortCount determines the minimum 'valid pad detection period' (VPDP). The minimum valid pad detection period is 71.68ms, calculated as shown in [Equation \(2\)](#).

(1)

$$\text{AEPACP} = \text{TFD} \times (\text{Number of Touch-sense inputs TSIN}[x]) \times (\text{AccumulatorCounter} + 1)$$

where:

- TFD is Time Frame Duration = (((PREscaler x 256)/fCLK) x (Number of Time Slice(s))) = ((42 x 256) / 48MHz) x 2 Time Slices = 448μs
- Number of Touch-sense inputs is 4
- AccumulatorCounter is defined as 4 in the software code.

(2)

$$\text{Minimum Valid Pad Detection Period (VPDP)} = (0\text{xFF} - \text{ShortCount} + 1) \times \text{AEPACP}$$

where:

- ShortCount is defined as 0xF8 in the software code

A long touch on the centre button is detected when both PadDownCounter (PDC) is 0x40 and PADFLAG.4 is 1. This long touch period is 1.71s, defined in [Equation \(3\)](#). If a longer detection period is required, modify PDC to a smaller value.

(3)

$$\text{Long Touch Period} = (0\text{xFF} - \text{PDC}) \times \text{AEPACP}$$

The maximum valid pad detection period is 2.28s, as defined in [Equation \(4\)](#). For this period, the code can be modified such that PADERROR.3 = 1, indicates a long touch on the centre button.

(4)

$$\text{Maximum Valid Pad Detection Period (VPDP)} = 0\text{xFF} \times \text{AEPACP}$$

Note: PDC is used to keep track of how long the pad is touched. PDC is initialized to be 0xFF at the start of a detected pad touch, and is decremented once at each All Enabled Pads Accumulated Count Period (AEPACP). When pad touch is removed, PDC is compared with the user input value, ShortCount, and/or 0x00 to decide if pad touch is valid, invalid or a long touch. PDC is one of the parameters stored in XRAM, at address 0xF0FF, by the FINDTOUCHEDPAD function.

Note: Please refer to LEDTS ROM Library in the User's Manual for more details of the FINDTOUCHEDPAD function, its inputs and its formulas.

3.1.2 Touch Detection Methodology for the Touch-Wheel

The Touch-Wheel Library (Library_TouchWheel.lib linked into the application code and programmed in the flash memory) is used to detect a touch on the touch-wheel. It is provided as a .lib file and is described in [Section 3.2](#). The Touch-Wheel Library makes use of the AverageL/H values calculated by the LEDTS ROM Library to determine the absolute location of the current finger touch on the touch-wheel. In order to keep the averaging active, the PADFLAG bits are cleared for touch-sense inputs TSIN0, TSIN1 and TSIN2.

3.2 Touch-Wheel Library

The Touch-Wheel Library (Library_TouchWheel.lib) has 2 main functions - calc_touchwheel() and calc_compensation(). The functions are described in this section.

3.2.1 Calc_Touchwheel() Function

The calc_touchwheel() function provides the status (i.e. whether touch-wheel is touched), the position (the absolute position of the last touch on the touch-wheel) and the touch-wheel average value as described in [Table 1](#).

Table 1 Specification of the Touch-Wheel Library Function - calc_touchwheel()

Function Name	calc_touchwheel
Prototype	void calc_touchwheel (int idata *parameter, char idata *angle, unsigned int idata *amplitude)
Input	<p>Address pointer of touch-wheel parameters (<i>wheel_param[0]</i>) The parameters include compensation for wheel2 and wheel1, amplitude threshold and filter gain. Refer to Section 4.1 for descriptions of the user-defined parameters.</p> <p>Address pointer of touch-wheel angle (<i>wheel_angle</i>) The angle value is updated when the touch-wheel is touched</p> <p>Address pointer of touch-wheel amplitude (<i>wheel_amplitude[0]</i>) The amplitude includes the touch-wheel position and the touch-wheel initial average value. Refer to Section 4.1 for descriptions of the user-defined parameters.</p>
Return	None
Output	<p>Touch-wheel Status (<i>wheel_amplitude[0]</i>) 00_H The touch-wheel is not touched >00_H The touch-wheel is touched</p> <p>Touch-wheel Position (<i>wheel_angle</i>) This value is updated by the Touch-Wheel Library when the touch-wheel status is more than 0x00 (i.e. a touch is detected on the touch-wheel). The HSL value is updated accordingly</p> <p>Touch-wheel Average Value (<i>wheel_amplitude[1]</i>) This value is calculated and updated by the Touch-Wheel Library when the touch-wheel is not touched</p>
Max Stack size	2
Resource used	XRAM address 0xF0F6 - 0xF0FF (10 bytes) ¹⁾

1) These parameters are stored by the FINDTOUCHEDPAD function and are used to calculate the Average value for the touch-sense inputs.

3.2.2 Calc_Compensation() Function

The compensation value is one of the user-defined parameters for calc_touchwheel function (see [Section 3.2.1](#)). The compensation values for Wheel1 and Wheel2 are calculated with respect to Wheel3, as shown in [Equation \(5\)](#) and [Equation \(6\)](#), to ensure that all values are equal under idle (no touch) condition.

(5)

$$\text{COMPENSATION_WHEEL2} = (\text{AVERAGEL/H of WHEEL3}) - (\text{AVERAGEL/H of WHEEL2})$$

(6)

$$\text{COMPENSATION_WHEEL1} = (\text{AVERAGEL/H of WHEEL3}) - (\text{AVERAGEL/H of WHEEL1})$$

The `calc_compensation()` function provides the compensation values for Wheel1 and Wheel2 as described in [Table 2](#). The user can call this function during board calibration to obtain an estimation of the compensation values. The values can be programmed to observe the sensitivity of the touch-wheel and further refined by the user. This function should be called after an initialization period to get a stable value. Suggested initialization period is about 640 Time Frame duration or interrupt and it can be calculated from [Equation \(7\)](#).

(7)

$$\text{Period} > 2^{\text{Divisor } n} \times 8 \times (\text{AccumulatorCounter} + 1) \times (\text{Number of Touch-sense inputs}) \times \text{TFD}$$

where:

- Divisor n is defined as 2
- AccumulatorCounter is defined as 4
- Number of Touch-sense inputs [TSIN(x)] is 4
- TFD is the Time Frame Duration

Table 2 Specification of the Touch-wheel Library Function - `calc_compensation()`

Function Name	<code>calc_compensation</code>
Prototype	<code>void calc_compensation (int idata *compensation)</code>
Input	Address pointer of touch-wheel parameters (<i>compensation[0]</i>) The parameters include the compensation for wheel2 and wheel1.
Return	None
Output	Wheel2 Compensation Value (<i>compensation[0]</i>) The Compensation_Wheel2 value with respect to Wheel3 Wheel1 Compensation Value (<i>compensation[1]</i>) The Compensation_Wheel1 value with respect to Wheel3
Max Stack size	2
Resource used	XRAM address 0xF0F6 - 0xF0FF (10 bytes) ¹⁾

1) These parameters are stored by the `FINDTOUCHEDPAD` function and are used to calculate the Average value for the touch-sense inputs.

3.3 RGB LED

[Figure 3](#) shows how the touch detection on the touch-wheel (based on the color mode) is reflected on the RGB LED (D2). The HSL value is obtained from the Touch-Wheel Library, converted to an RGB value (via the `HSL_TO_RGB()` function), and the CCU6 registers are programmed where their respective pins are connected to the RGB LED.

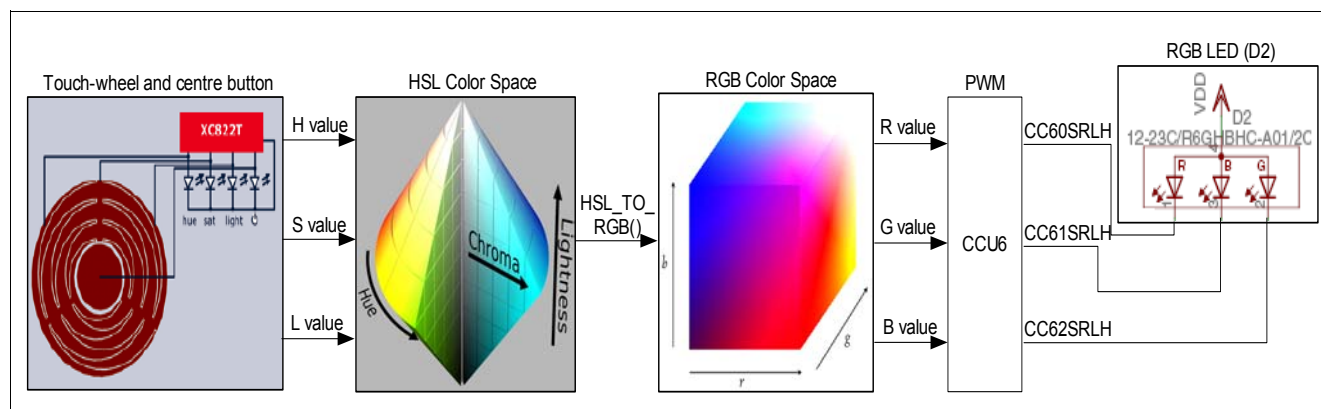


Figure 3 HSL to RGB LED Block Diagram

3.4 Overview Flow Chart of Color Wheel Software Code

Figure 4 shows the overview flow chart of the Color Wheel software code.

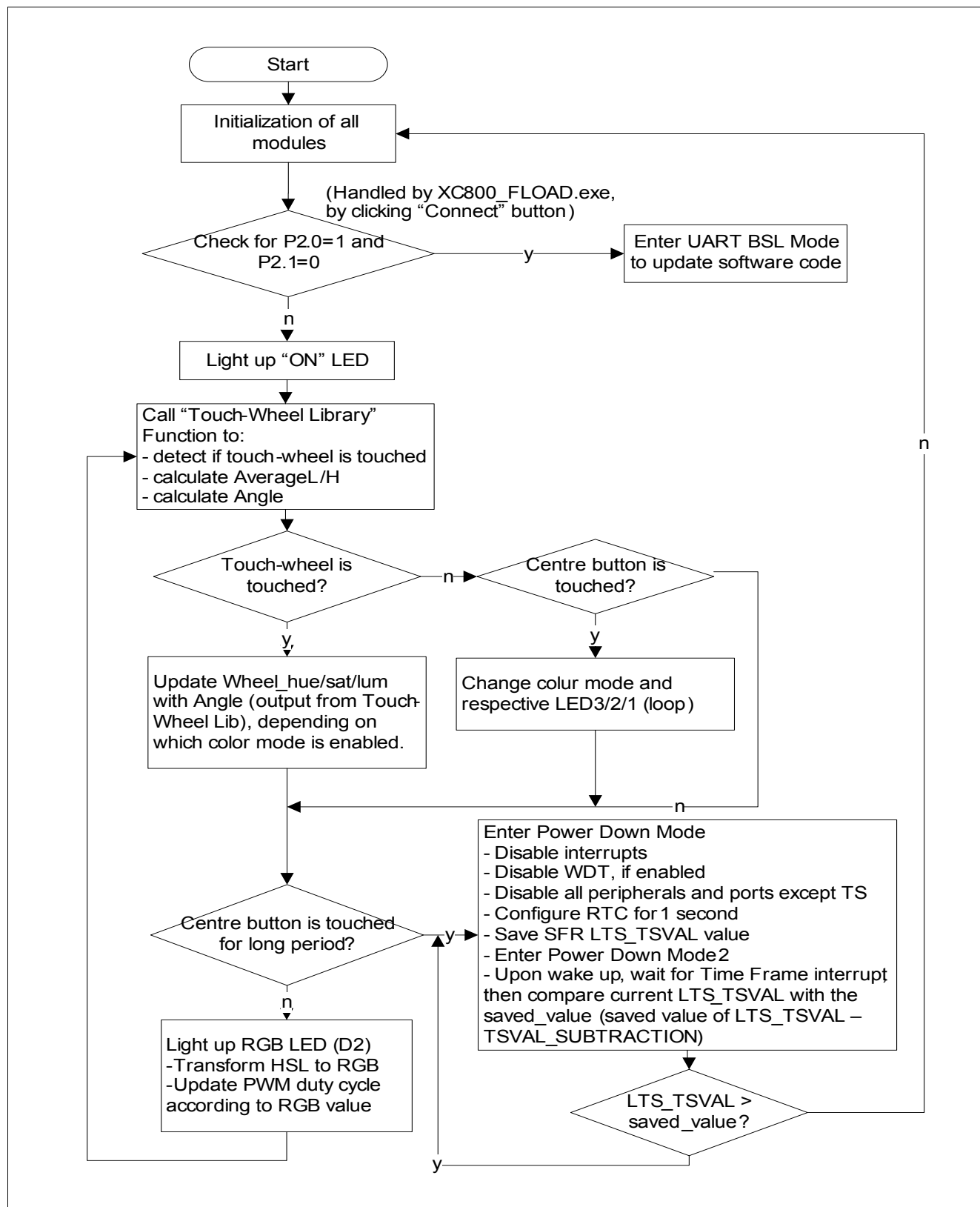


Figure 4 Overview Flow Chart of the Color Wheel Software Code

4 Modifying Color Wheel Software Code

The Color Wheel software example code can be accessed and modified by the user with DAVe-Bench - Infineon's free tool chain. The user can modify the user-defined parameters (as described in [Section 4.1](#)), the timing, code flow, enabling/disabling modules, and interrupts handling for example, as required for a specific application. The Color Wheel software example code can be downloaded together with this application note.

4.1 User-Defined Parameters

There are 6 parameters which the user can modify for the Touch-Wheel Library based on their application board and usage. The respective values and description are listed in [Table 3](#).

Table 3 User-Defined Parameters for Color Wheel Software Code

Parameter	Current Value	Description
COMPENSATION_WHEEL2	-11	Compensation value for Wheel2 due to environment
COMPENSATION_WHEEL1	-11	Compensation value for Wheel1 due to environment
AMPLITUDE_THRESHOLD	0x400	Threshold value for amplitude. Above threshold value is considered as touch on the touch-wheel
AMPLITUDE_FILTER_GAIN	5	Filter gain value for amplitude
AMPLITUDE_START_VALUE	0x240	Start value for amplitude during initialization/start-up
TSVAL_SUBTRACTION	3	To wake up the device via centre button Subtraction value to determine the threshold level. Current TSVAL > (Saved TSVAL - SUBTRACTION) means the centre button is detected as "no touch" - device remains in Power Down Mode

The threshold value can be reduced to increase sensitivity, and the filter gain value can be reduced to allow a faster response time. The user can modify the user-defined parameters to match their application. The estimated compensation values for Wheel1 and Wheel2 can be obtained by calling Touch-Wheel Library function as described in [Section 3.2.2](#).

4.2 DAVe-Bench Tool

The Infineon's free tools DAVe and DAVe-Bench are used to generate and compile the code for Color Wheel application. The user should follow the procedure shown below and in [Figure 5](#), when using DAVe-Bench to modify and compile the code.

Procedure to modify Color Wheel Application software code:

- Download and install DAVe-Bench (www.infineon.com/dave-bench)
- Define a workspace for DAVe-Bench
- Click "File" -> "Import" -> "XC800" -> "Existing Project"
- Select the desired folder where the project exists (RGB_Wheel folder)
- Default: Tick option - copy project into workspace (Future changes will be reflected at the project in the workspace, while original copy remain unchanged)
- Click "Finish" to import the project
- Modify the software code
- Click "Project" -> "Build Active Project" to compile the code
- Check the status of compilation on "Console" box
- If compilation is not successful, the bugs are listed in the "Problems" box
- Once hex file is generated, download the hex file using DAVe-Bench (XC800_FLOAD), described further in [Chapter 5](#).

Modifying Color Wheel Software Code

There are two configurations, Debug and Release, available to compile the Color Wheel software code. Select the hex file from the respective folder. To select the required configuration, select:

"Project" -> "Properties" -> "C/C++ Build" -> "Settings" -> "Manage Configurations"

Activate this option.

To view the differences between the Debug and Release configurations, select:

"Project" -> "Properties" -> "C/C++ Build" -> "Settings"

User the "Tool Settings" tab, select:

"SDCC Compiler" -> "Optimization"

Select between Debug and Release configurations. The default configuration is Debug.

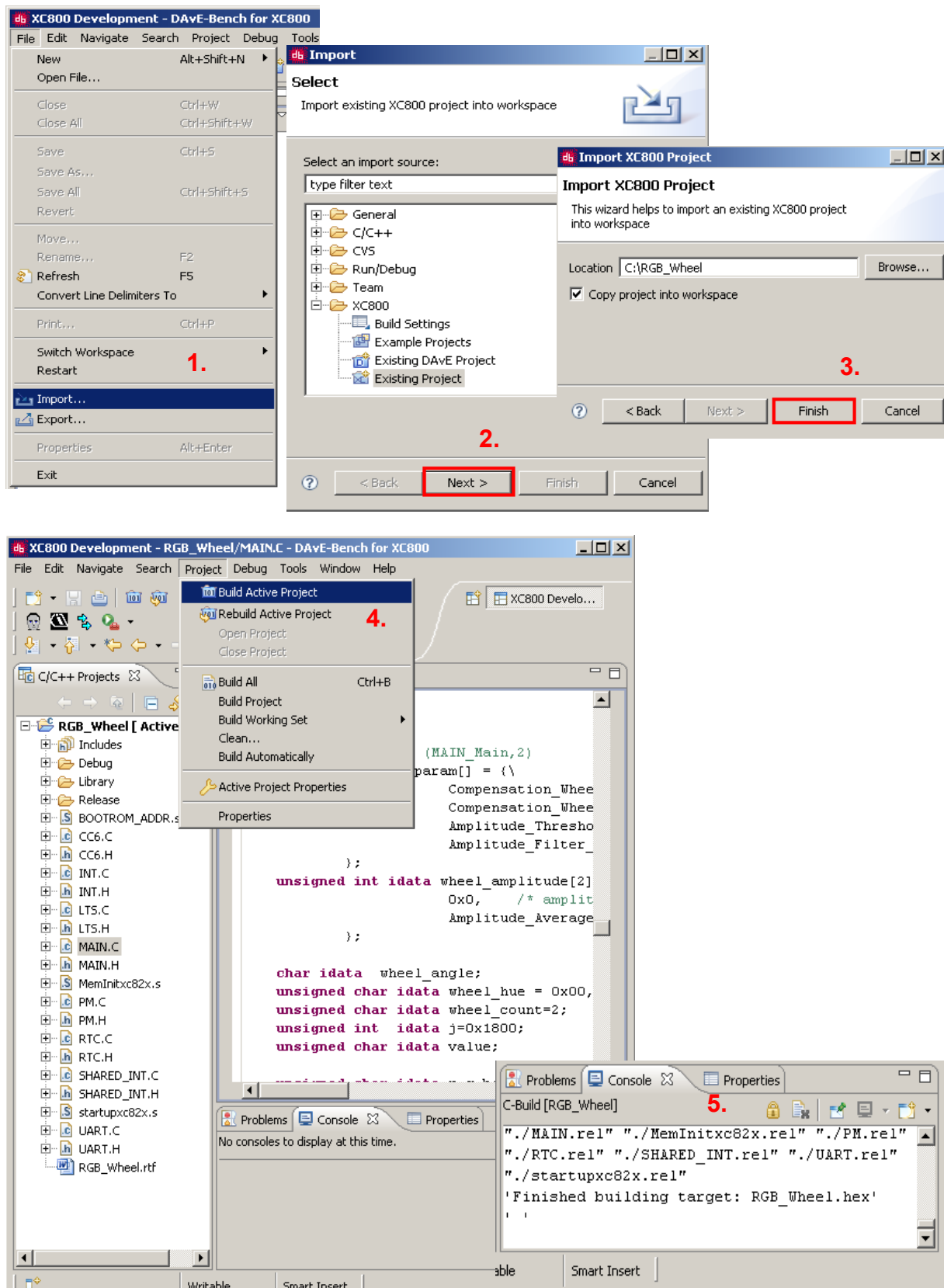


Figure 5 DAvE-Bench GUI for Compiling the Color Wheel Software Code

5 Downloading Modified Color Wheel Software Code

The Color Wheel Application Kit is prepared to start up automatically after power-on reset. This is achieved by programming the Boot Mode Index (BMI) to User Mode (Productive). To download the software code, the “accessing of UART BSL Mode” code must be included in the user flash code at all times (see [Section 5.1](#)).

The procedure to enter the UART BSL Mode to update software code is as follows:

- Plug in Color Wheel board to computer, via USB, to start up the board
- Open DAVE-Bench (XC800_FLOAD) tool (Device: XC82x-1F; Protocol: UART; Physical Interface: Half Duplex; set COM port)
- Click on “Connect” button to connect UART BSL Mode (*P2.0 and P2.1 are controlled by XC800_FLOAD*)
- Device is in UART BSL Mode (*Notice that the “On/Off” and color mode LEDs are off*)
- Click on “Write User ID” button to program BMI to UART BSL Mode
- Erase flash (optional)
- Open hex file and download code to Flash.
- Execute Flash and observe Color Wheel board

Note: Ensure that the access code to UART BSL Mode is included in code to allow future software updating. Once the code is verified, the user can re-program the device to User Mode (Productive).

Attention: It is recommended to program BMI to UART BSL Mode before updating the software code. This is a safer approach and it will ensure that the device can be accessed via UART BSL Mode at all times. Only program the BMI back to User Mode (Productive) when the updated software code is proven to be working and is in a finished state.

5.1 Access Code to UART BSL Mode

In order to allow software updating, the following code must be included in the user flash code at all times:

- Soft reset - triggered by external interrupt 0 (low level triggered at P2.0). This emulates a Reset pin. [INT.C]
- Enter UART BSL Mode - upon start up, check P2.1 for low to enter UART BSL Mode. [MAIN.C]

Note: The 2 port pins are controlled by the XC800_FLOAD tool to trigger a soft reset and to enter UART BSL Mode at next boot-up.

Note: The user can define specific conditions to program the BMI value to access to UART BSL Mode.

MAIN.C

```
void main(void)
{
    // USER CODE BEGIN (MAIN_Main,2)
    ....
    .....
    if( !(P2_DATAIN & 0x02)) && (P2_DATAIN & 0x01) )
    {
        while(--j);                // wait for FTDI device to enumerate and startup
        if( !(P2_DATAIN & 0x02) ) // Only enter BSL in case "MBC" is low
        {
            EA = 0;                // disable interrupts
            NMICON = 0x00;         // disable NMIs
            SP = 0xE0;             // move stack pointer outside flash buffer
            BR_UART_BSL();         // start UART BSL
        }
    }
    // USER CODE END
```

```
MAIN_vInit();
...
...
}
```

INT.C

```
void INT_vInit(void)
{
    NMICON = 0x00; //NMI Control Register
    IEN0 |= 0x01; // interrupt enable register 0
    IEN1 |= 0x00; // interrupt enable register 1
    EXICON0 = 0xF3; // External interrupt Control Register 0
    SFR_PAGE(_su3, noSST); // switch to page 3 without saving
    MODPISEL1 |= 0x18; // peripheral input select register 1
    SFR_PAGE(_su0, noSST); // switch to page 0 without saving
} // End of function INT_vInit

void INT_viExt0(void) interrupt EXT0INT
{
    // USER CODE BEGIN (IR_IsrExt0,2)
        // emulate Reset Pin an falling edge
        SFR_PAGE(_su1, noSST); // switch to page1
        while( !(P2_DATAIN & 0x01) );
        MAIN_vSoftReset();
    // USER CODE END
    ...
    .....
} // End of function INT_viExt0
```

6 Implementation of Color Wheel with XC822T

There are several on-chip modules used in the Color Wheel Application Kit. The configuration and usage of these modules are described in this chapter.

6.1 Real-Time Clock (RTC) Module

Real-Time Clock is configured for 1 second and is used for wake-up in the Power saving control module.

6.2 Power Saving Control Module

The power saving mode saves on the power consumption when the board is not used. Power Down Mode is entered when the centre button is pressed for a long period. The configurations are:

- Power Down Mode 2
- Power down with RTC running
- Wake-up without reset

SFR LTS_TSVAL value is saved before entering Power Down Mode. At wake-up (every 1 second, by the RTC module), the software waits for the Time Frame flag (indicates touch-sensing on one pad completed) before comparing the current SFR LTS_TSVAL with the saved_value (i.e. saved value of LTS_TSVAL - user-defined TSVAL_SUBTRACTION).

If the latest SFR LTS_TSVAL value is more than the saved_value, it means that the centre button is not pressed. The Power Down Mode is therefore re-entered and the latest SFR LTS_TSVAL value is re-saved. The cycle then repeats. If the SFR LTS_TSVAL value is less than or equal to the saved_value, it means that the centre button is being pressed, therefore Power Down Mode is exited and the Color Wheel board will turn on ('On/Off' LED light turns on). With above mentioned protocol, timing to enter Power Down Mode is consistent, while timing to exit Power Down Mode varies, depending on when the user touches the centre button.

6.3 CCU6 Module

CCU6 is used to control the color, saturation and brightness level at RGB LED (D2). RGB LED utilizes COUT60 (P1.0), COUT61 (P1.2) and COUT62 (P0.5) for PWM control.

6.4 Interrupt System Module

External interrupt 0 (P2.0) (low level triggered) is enabled. This emulates a Reset pin. If a negative edge is detected at P2.0, soft reset will be activated. It is triggered by XC800_FLOAD tool to enter UART BSL Mode at next boot-up.

Time Slice and Time Frame interrupts are also enabled to set up the configurations for LEDTSCU.

6.5 UART Module

UART module is enabled for debugging to transmit and receive data via P0.6 (TXD/RXD).

6.6 LED and Touch-Sense Controller Unit Module

LEDTSCU module is used for touch-sensing and driving the LEDs. The configuration is:

- 4 Touch-Sense inputs (P0.0, P0.1, P0.2 and P0.3)
- 1 LED column (P0.4)
- 4 LED line pins (P0.0, P0.1, P0.2 and P0.3)

7 Schematics and Layout of the Color Wheel Board

Figure 6 shows the component layout of the Color Wheel board, while Figure 7, Figure 8 and Figure 9 show the schematics of the XC822T device, USB interface and power supply on the Color Wheel board respectively.

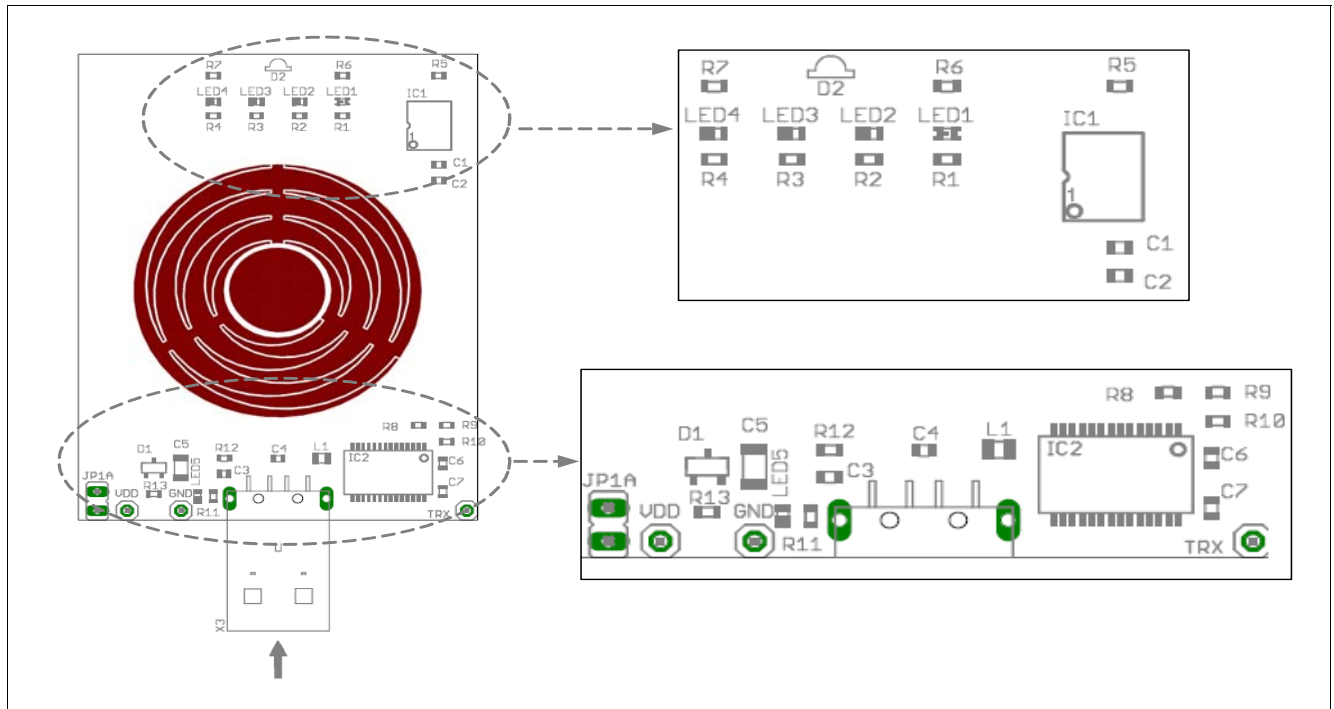


Figure 6 Component Layout of the Color Wheel Board

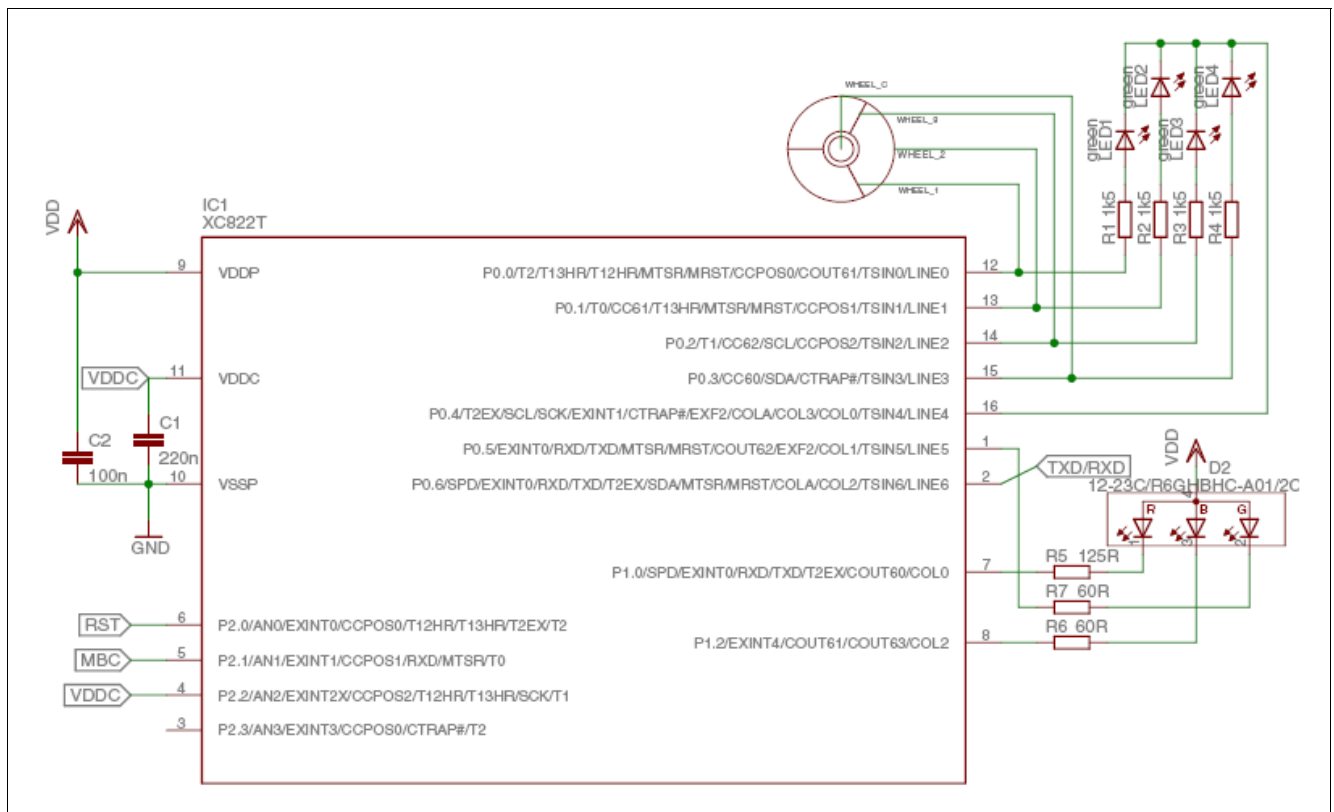


Figure 7 Schematic of the Color Wheel Board

Schematics and Layout of the Color Wheel Board

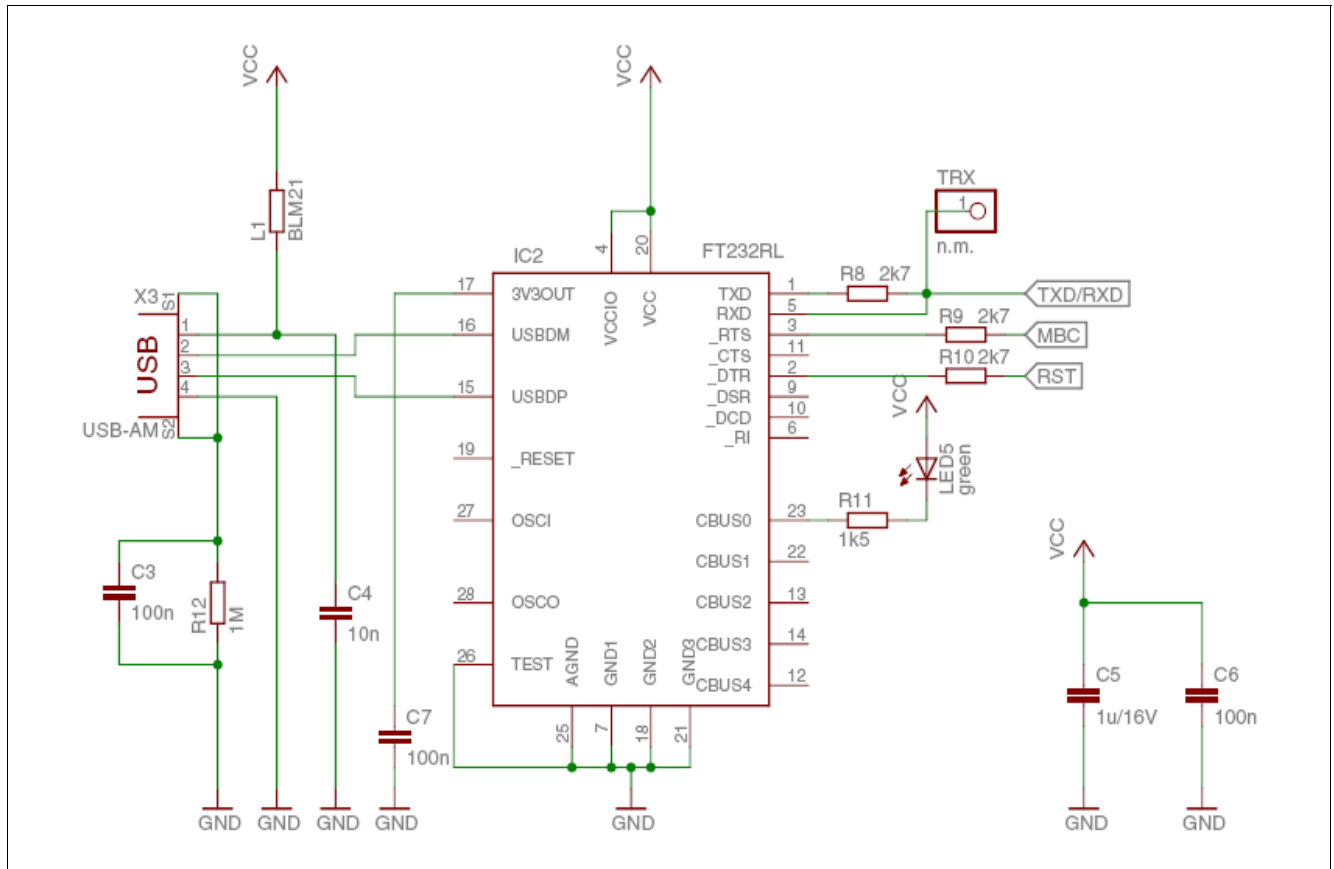


Figure 8 Schematic of USB Interface on the Color Wheel Board

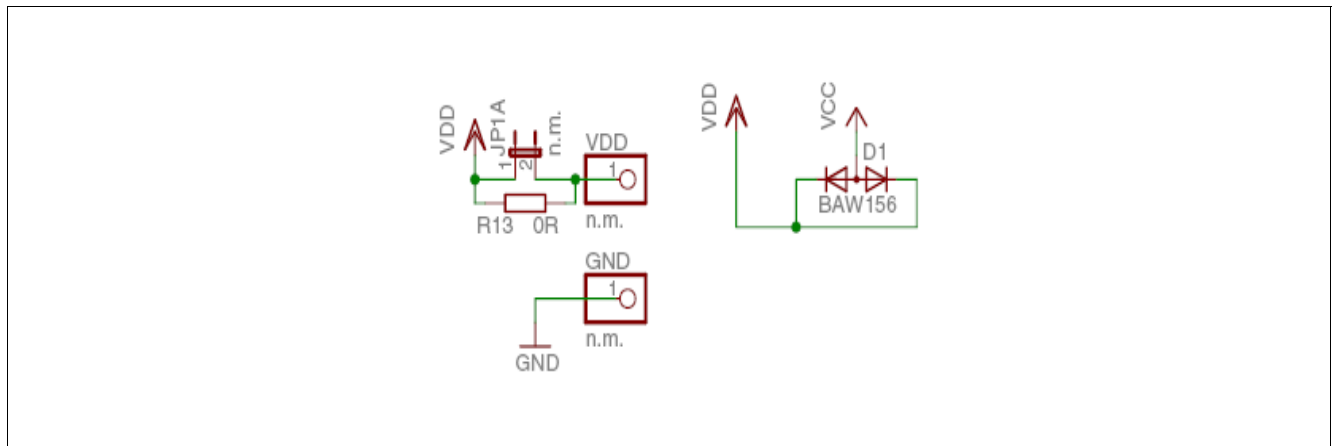


Figure 9 Schematic of power supply of the Color Wheel Board

Figure 10 and Figure 11 show the top and bottom layout of the Color Wheel board respectively.

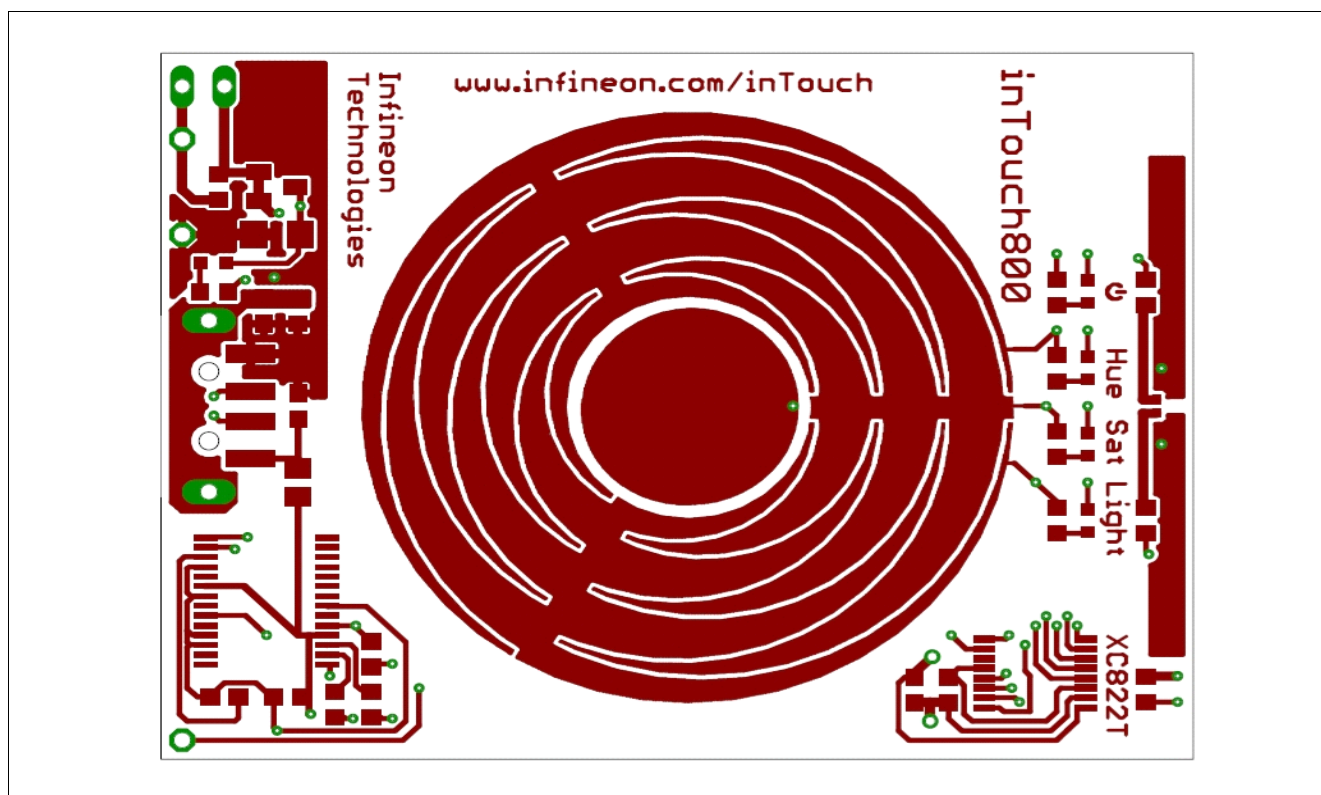


Figure 10 Top Layout of the Color Wheel Board

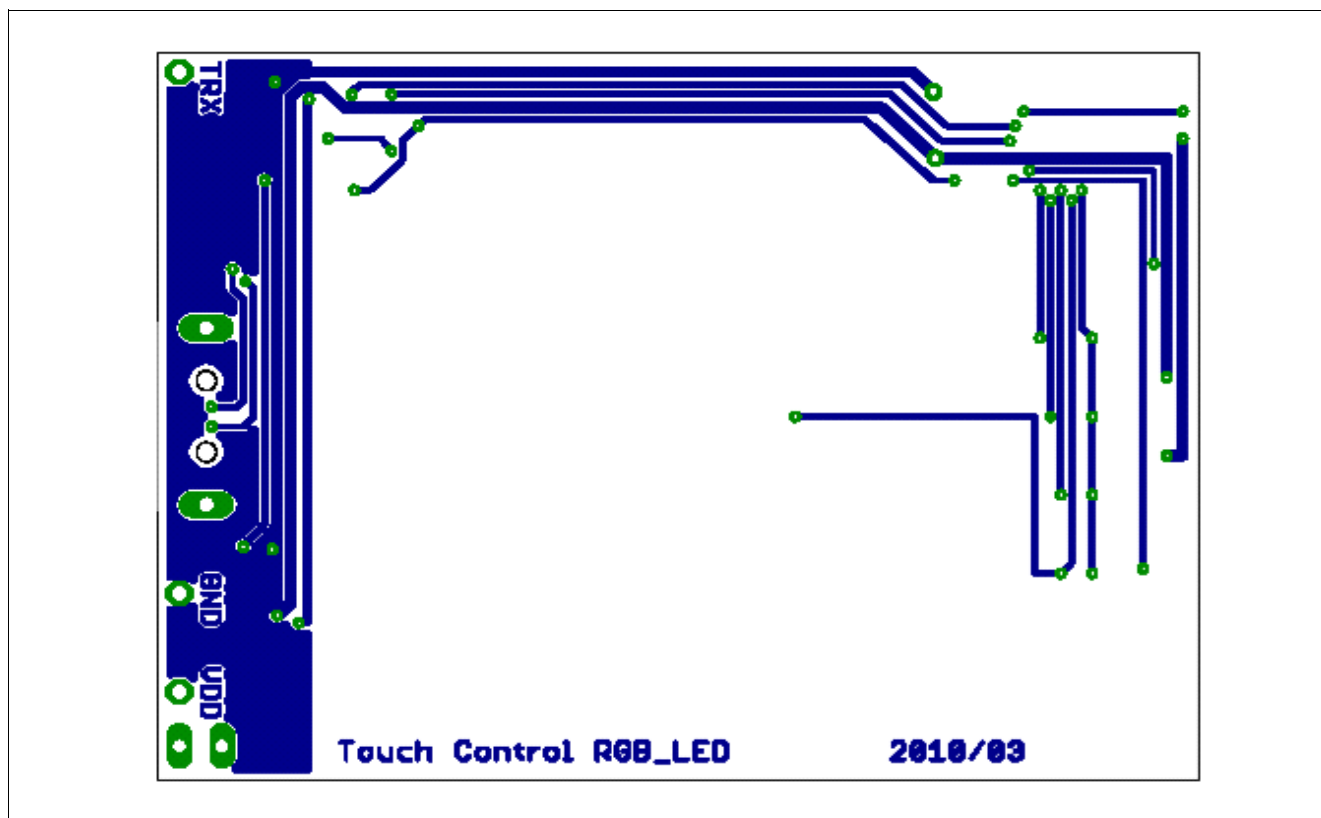


Figure 11 Bottom Layout of the Color Wheel Board

8 Summary

The capacitive touch-sensing capability of the Infineon XC82x and XC83x Microcontrollers opens the door to a wide variety of applications. The Infineon touch-sensing solution described in this document is a total solution on-chip, comprising optimized hardware and effective software functions running from ROM. By taking the example provided and working with the Infineon's free tool-chain, the development of new applications for these devices is both simplified and straight-forward.

www.infineon.com