IS3XCS8977 Touch Key Application User Guide

Rev. A

2022-04-12

GENERAL DESCRIPTION

CS8977 is a general-purpose MCU with 64KB Code e-Flash memory with ECC and 2KB SRAM with ECC. The embedded flash for code storage has built-in ECC that corrects one-bit errors and detects two-bit errors. CPU accesses the e-Flash through program address read and through Flash Controller which can perform software read/write operations of e-Flash.

CS8977 has a 1-T 8051 with enhanced multiplication and division accelerator. There are three clock sources for system. One is a 16MHz/32MHz IOSC (manufacturer calibration+/- 2%), one is XCLK, and the other one is SOSC32KHz (typical 32KHz) which is divided from slow oscillator. ALL clock sources have a clock programmable divider for scaling down the frequency to save power dissipations. The clock selections are combined with flexible power management schemes, including NORMAL, STOP, and SLEEP modes to balance speed and power consumption.

There are T0/T1/T2/T3/T4/T5 timers coupled with CPU and three WDTs where WDT1 is clocked by SYSCLK, and WDT2/WDT3 are clocked by a non-stop SOSC32KHz. An 8-bit/16-bit checksum and 16-bit CRC accelerator is included. There are EUART/LIN controllers, I2C master/Slave controllers and SPI master/slave controller. The interfaces of these controllers are multiplexed with GPIO pins. Other useful peripherals include a buzzer control, six 8/10/12-bit PWMs, one channel of 16-bit timer/capture, and one 16-bit quadrature decoder. There are also 16 channels 8-bit PWM for LED control.

Analog peripherals include a 12-bit ADC with internal temperature sensor, an 8-bit voltage output DAC, and four analog comparators with programmable threshold. A touch key controller with up to 20-bit resolution is included. The touch key controller has shield output capability for moisture immunity and allows auto-detection wakeup from sleep mode (under 20uA). The maximum number of key inputs can be scanned is 27. The touch key controller can also be used for proximity sensing.

CS8977 provides a flexible means of flash programming that supports ISP and IAP. The protection of data loss is implemented in hardware by access restriction of critical storage segments. The code security is reinforced with sophisticated writer commands and ISP commands. The on-chip break point processor also allows easy debugging which can be integrated with ISP. Reliable power-on-reset circuit and low supply voltage detection allow reliable operations under harsh environments.

FEATURES

CPU and Memory

- Up to 32MHz 1-Cycle 8051 CPU core
- 16-bit Timers T0/T1/T2/T3/T4 and 24-bit Timer T5
- Checksum and CRC accelerator
- WDT1 by SYSCLK, WDT2/WDT3 by SOSC32KHz
- Clock fault monitoring
- Up to 6 external interrupts shared with GPIO pins
- Power saving modes – Normal, STOP, and SLEEP modes
- 256B IRAM and 1792B XRAM with ECC check
- 64KB Code e-Flash with ECC and two 128x16 Information Block
  - Code security and data loss protection
  - 100K endurance and 10 years retention

Clock Sources

- Internal oscillator at 16MHz/32MHz(+/- 2%)
  - Spread Spectrum option
- Internal low power oscillator 128KHz/256KHz
- External clock option and clock out

Digital Peripherals

- 6 CH 8/10/12-bit center-aligned PWM controller
  - Trigger interrupt and ADC conversion
- 16 CH 8-bit PWM left/right aligned
- One 16-bit Timer/Capture and One 16-bit quadrature decoder
- Buzzer/Melody generator
- One I²C Master
- One I²C Slave – also for ISP and debug
- One SPI Master/ Slave Controller
- One EUART1 and one EUART2/LIN

Analog Peripherals

- Capacitance sense touch-key controller - scan up to 27 key
  - Shield output for moisture immunity
  - Low power sleep mode wakeup (<20uA).
  - Active Proximity sensing front-end
- 12-Bit SAR ADC with GPIO analog input
  - Track and hold
  - Temperature sensor and supply measurement
- 8-Bit DAC and four analog comparators
- Power on reset and Low voltage detect (2.2V-4.5V)

Miscellaneous

- Up to 28 GPIO pins with multi-function options
  - Configurable IO structure and noise filters
- 2.3V to 5.5V single supply
- Active current < 150uA/MHz in Normal mode
- Low power standby (1uA) in SLEEP mode
- Operating temperature -40°C t0 85°C
- TSSOP20/24/28, QFN-32 and LQFP32 package (RoHS compliant)

ORDERING INFORMATION

Part No.	Temperature Range	Package
IS31CS8977-QFLS2-EBGUI	-40°C ~ +85°C	QFN-32, Lead-free

Table 1: Ordering Information

For pricing, delivery, and ordering information, please contact LUMISSIL's marketing and sales team at https://www.lumissil.com/company/office-locations or (408) 969-6600.

QUICK START

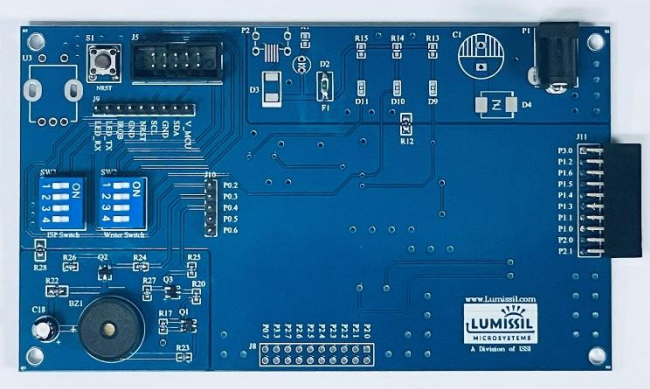


Figure 1: Photo of IS3XCS8977 EBGUI Top View

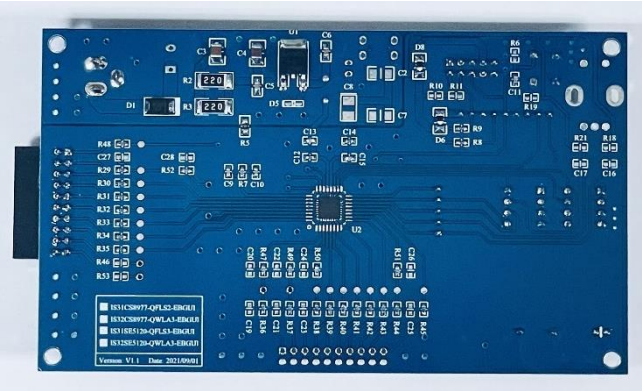


Figure 2: Photo of IS3XCS8977 EBGUI Bottom View

RECOMMENDED POWER SUPPLY

- 9V~15V, 1.0A power supply

ABSOLUTE MAXIMUM RATING

- ≤ 25V power supply

Caution: Do not exceed the conditions listed above, otherwise the board will be damaged.

PROCEDURE

IS3XCS8977 EBGUI board is fully assembled and tested. Follow the steps listed below to verify board operation.

Caution: Do not turn on the power supply until all connections are completed.

Plug in the DC adapter and turn on the power supply. Please pay attention to the supply current.

EzISP PROGARMMING BOARD OPERATION

IS3XCS8977 Touch Key evaluation board offers flexible flash programming that can be programmed via the ISP Mode, Write Mode or Fast Write Mode via EzISP Programming Board.

1) The ISP Mode of the EzISP Programming Board requires 4-pins connection (SCL, SDA, GND, VDD).

2) The Write Mode and Fast Writer Mode of the EzISP Programming Board requires 7-pins connection (RST, CK, CS, MO, MI, GND, VDD).

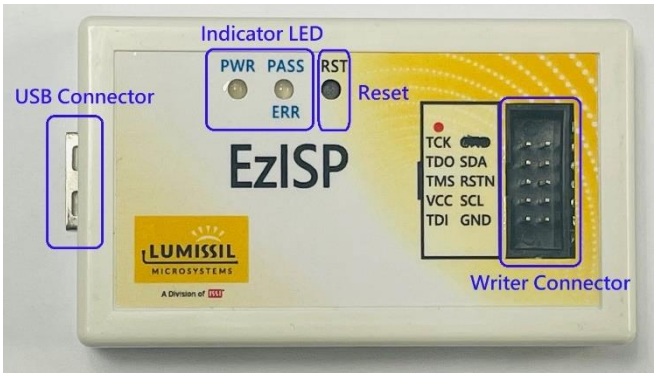


Figure 3: Photo of EzISP Programming Board

BOARD ASSEMBLY

Please refer Figure 4 and Figure 5 for IS3XCS8977 EBGUI top and bottom view. IS3XCS8977 EBGUI schematics and PCB layout can be found from Figure 13~19. Bill of Materials can be found in table 2.

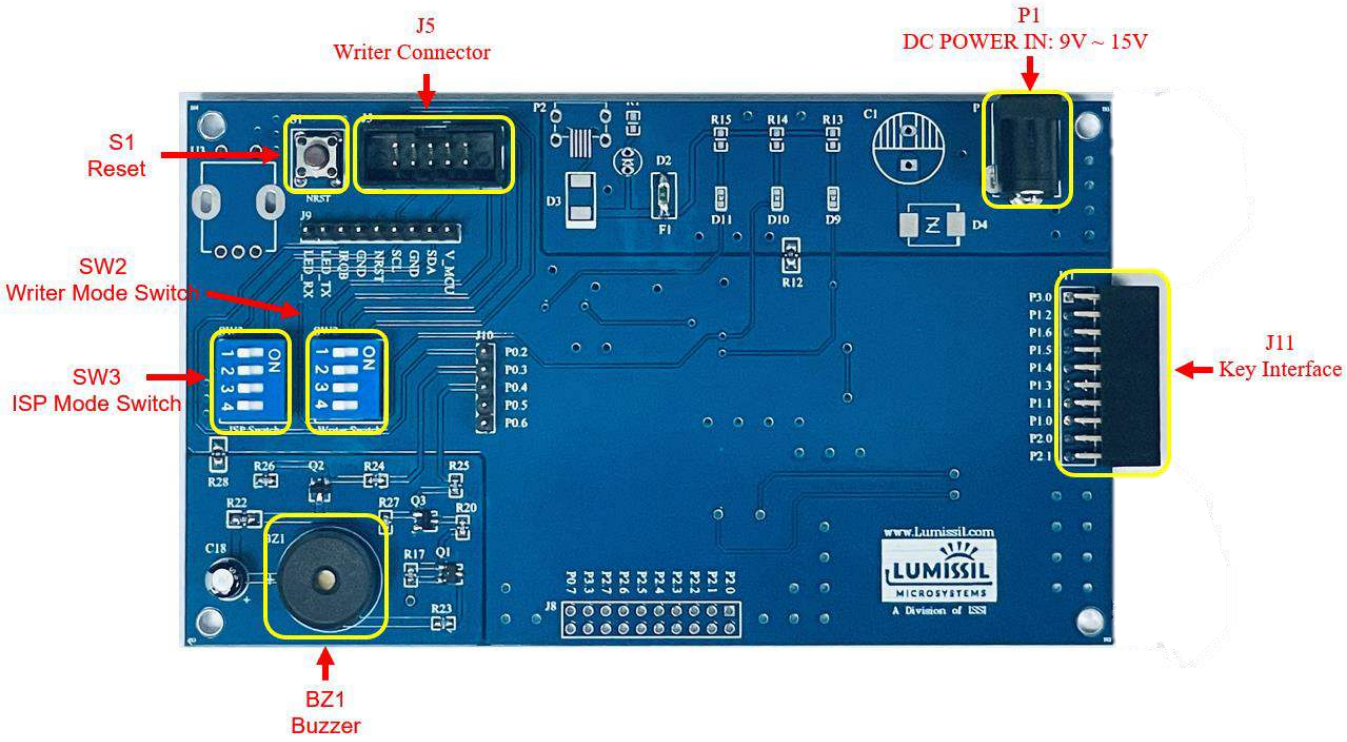


Figure 4: Photo of Main Board Top View

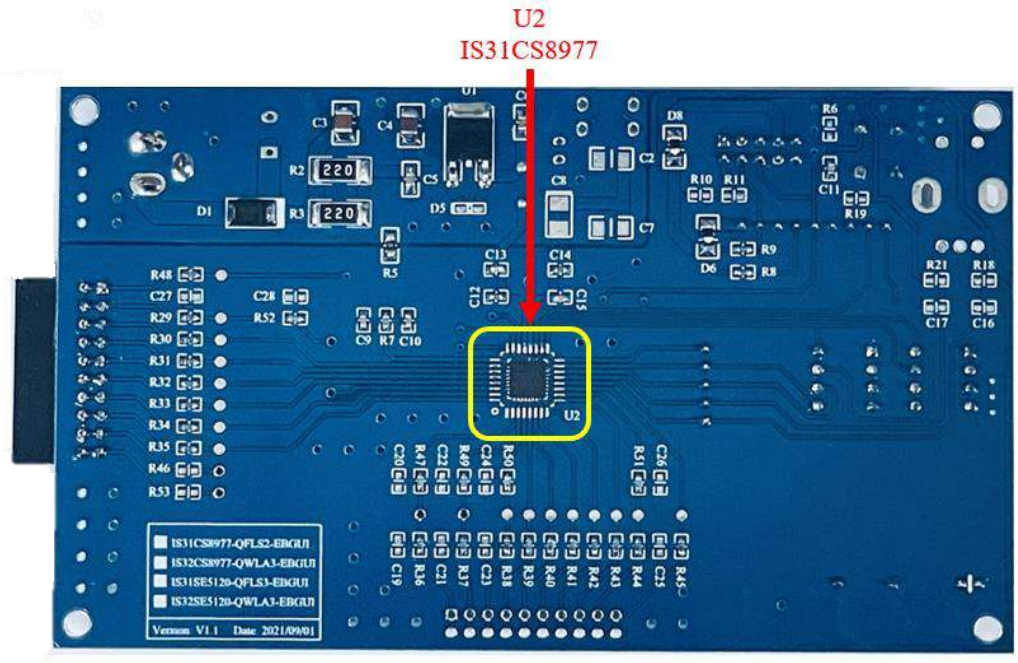


Figure 5: Photo of Main Board Bottom View

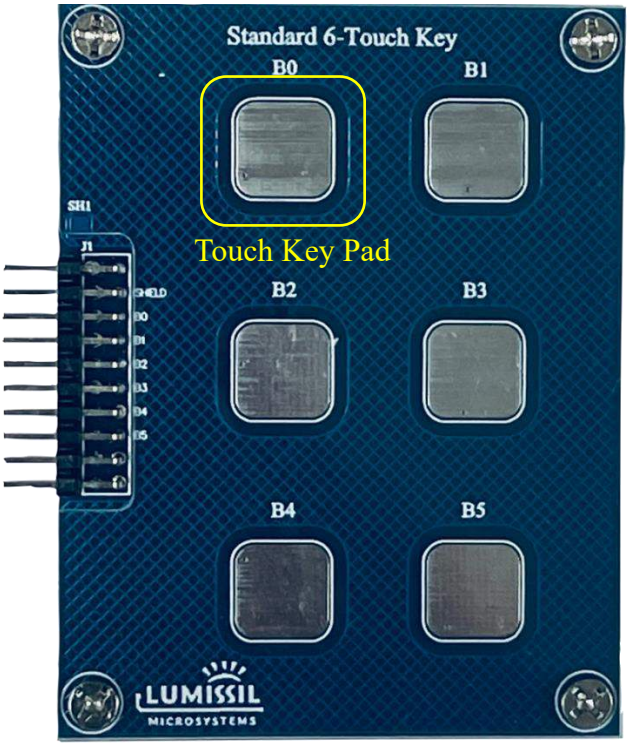


Figure 6: Photo of 6- Touch Key Daughter Board

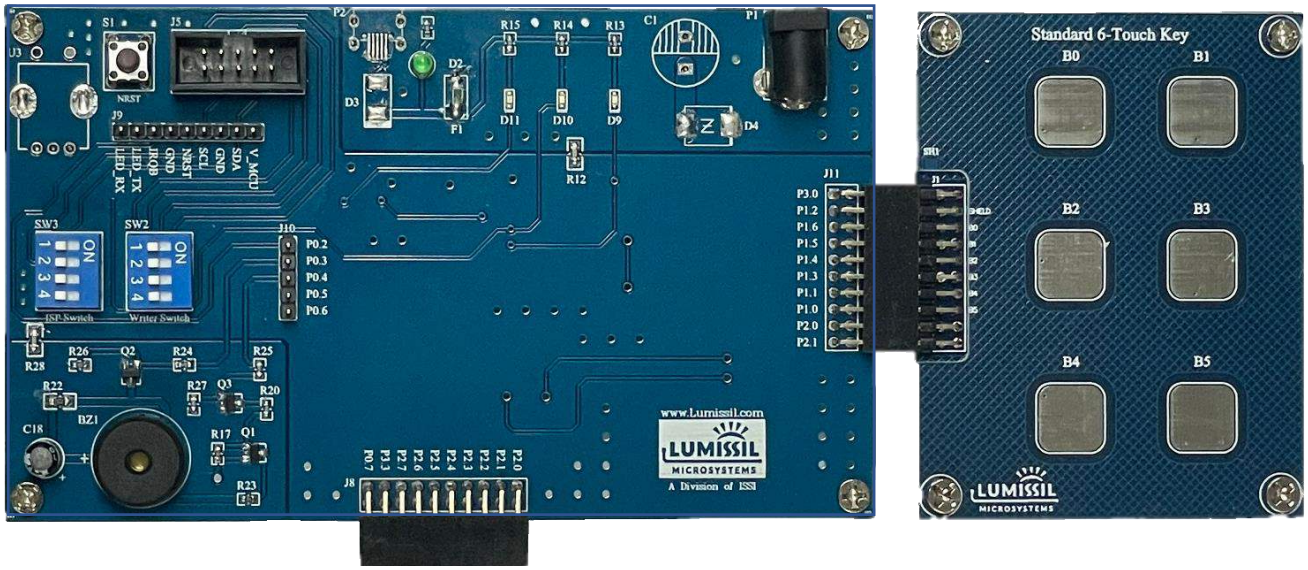


Figure 7: Photo of IS3XCS8977 EBGUI Board Connection with Touch Key Daughter Board

SOFTWARE SUPPORT

Before starting up IS3XCS8977 Touch Key GUI software, it is required that the PC is installed with the EzISP USB driver and related files (for example: Microsoft Framework and C++ library) to check and update device boot code and flash if necessary. User can unzip file EzISP v3.3.x.zip(downloadable from our company web site) and run EzISP setup v3.3.x.exe for installation. And this installation needs the login Windows user with administrator privilege.

EzISP software supports WinXP / Win7 / Win8 / Win10 operating system.

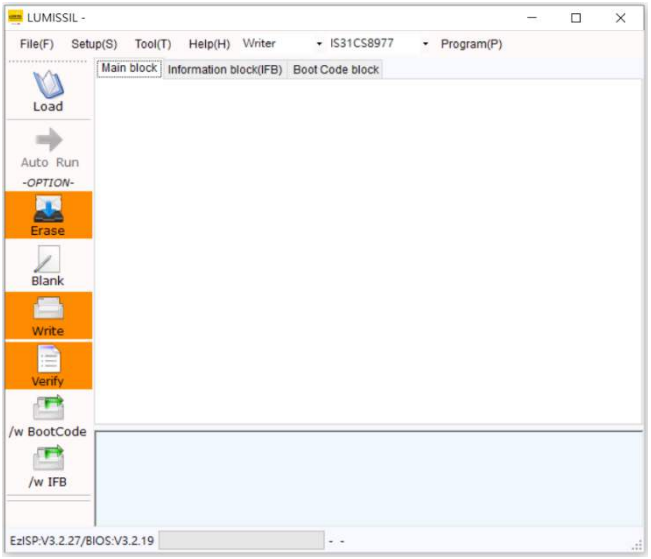


Figure 8: Photo of EzISP Writer Mode operation interface

EzISP Writer Mode programming operation process is as following:

1) Connect USB cable between the writer connector of the EzISP Programming Board and the USB port of your PC.
2) Connect a 10-pin 2x5 Socket-Socket 1.27mm IDC cable from the writer connector of the EzISP Programming Board to the writer connector on the IS3XCS8977 EBGUI.
3) Turn all switches of DIP switch (SW2) to ON position.
4) Run EzISP software.
5) Select the MCU chip type and programming mode (for example: Writer Mode or Fast Writer Mode).
6) Click the "Load" button and select the programming code (*.hex) to load.
7) Click the “Auto Run” button, and the EzISP software will immediately perform “Erase”, “Write” and “Verify”. Then the prompt information will be displayed at the bottom of the window. The prompt message includes the programming result and the running time.
8) Turn all switches of DIP switch (SW2) to OFF position.

After successful programming, IS3XCS8977 can run IS3XCS8977 Touch Key tool program.

Caution: In the Fast Writer/Writer mode, on-chip Boot code WILL BE erased! Once it’s erased, ISP mode cannot function anymore.

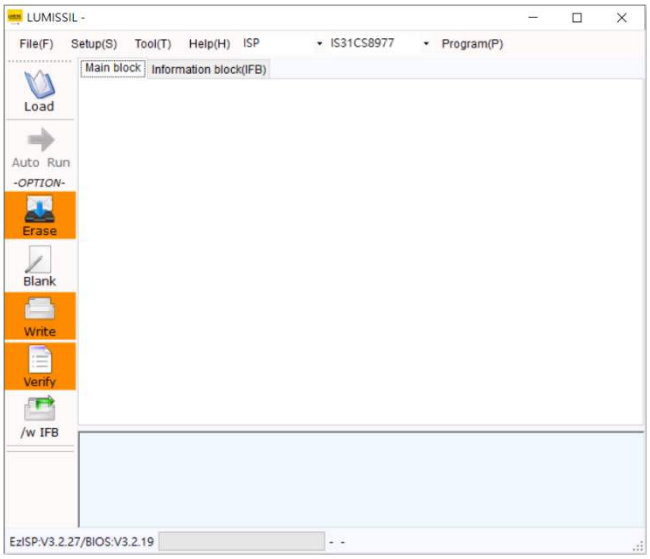


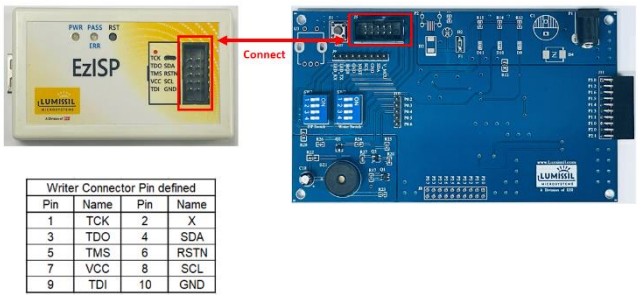
Figure 9: Photo of EzISP ISP Mode operation interface

EzISP ISP Mode programming operation process is as following:

1) Connect USB cable between the writer connector of the EzISP Programming Board and the USB port of your PC.
2) Use a 10-pin 2x5 Socket-Socket 1.27mm IDC cable from the writer connector on the EzISP Programming Board to the writer connector on the IS3XCS8977 EBGUI.
3) Turn all switches of DIP switch (SW3) to ON position.
4) Runb EzISP software.
5) Select the MCU chip type and programming mode (for example: ISP Mode).
6) Click the "Load" button and select the programming code (*.hex) to load.
7) Click the “Auto Run” button, and the EzISP software will immediately perform “Erase”, “Write” and “Verify”. Then the prompt information will be displayed at the bottom of the window. The prompt

message includes the programming result and the running time.

8) Turn all switches of DIP switch (SW3) to OFF position.



Writer Connector Pin defined			
Pin	Name	Pin	Name
1	TCK	2	X
3	TDO	4	SDA
5	TMS	6	RSTN
7	VCC	8	SCL
9	TDI	10	GND

Figure 10: Photo of EzISP Programming Board Connecting with IS3XCS8977 EBGUI Board

IS3XCS8977 EBGUI board requires 9V~15V/1.0A DC power supply. It must use a dedicated DC power supply.

The steps listed below are examples of GPIO control using the IS3XCS8977.

1) Use the test code in Appendix I and compile the test code in the Keil C51 development environment (IDE, Keil µVision).
2) Create a Hex file of the test code in Keil C51 and load the hex file of the test code via the EzISP software to update the firmware to the IS3XCS8977 flash.
3) After the firmware update is complete, IS3XCS8977 chip will automatically reset and execute the program.
4) In this example, you can measure if the P02 pin on the IS3XCS8977 EBGUI has been toggled.

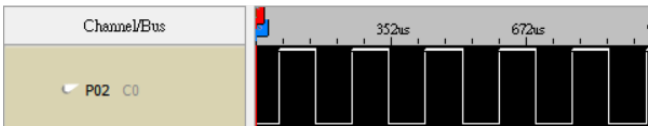


Figure 11: Photo of P02 pin toggling on the IS3XCS8977 EBGUI

SCHEMATICS AND LAYOUT GUIDE

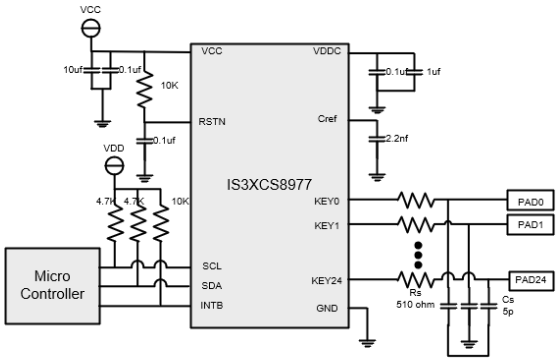


Figure 12: IS3XCS8977 Typical Application

Below are layout guideline for your reference. Please refer above Figure 12 for below descriptions.

1.) Four layers of PCB is recommended to have better performance.
2.) In case, two-layer PCB is a must. Put the IC, sensor-related components, and sensor traces on the bottom layer. It is highly recommended to provide a whole ground plane upon the top layer over IC and sensor related-components to have less emissions.
3.) Rs and Cs should be placed as close as possible to IC.
4.) Choose qualified by-pass capacitors and place them as close as possible to the IC VDDC and VCC pins to provide the best decoupling.
5.) Cref should be placed as close as possible to IC.
6.) Reduce the capacitance of a touch sensor. There should be no copper-filled, grounding or signal routing within the sensor area.
7.) Reduce signals trace length. A long PCB trace length can form a proximity sensor.
8.) Sensor traces should not be close to communication lines like I2C, UART.
9.) Sensor traces should not be close to control signal lines like LED control lines, LED power lines, …, etc..
10.) Place a guard ring on the perimeter of the PCB and connect the guard ring to the chassis ground. Mechanical design will also play an important role for better performance in addition to PCB layout.

SCHEMATICS OF IS3XCS8977 TOUCH KEY EVALUATION BOARD - EBGUI

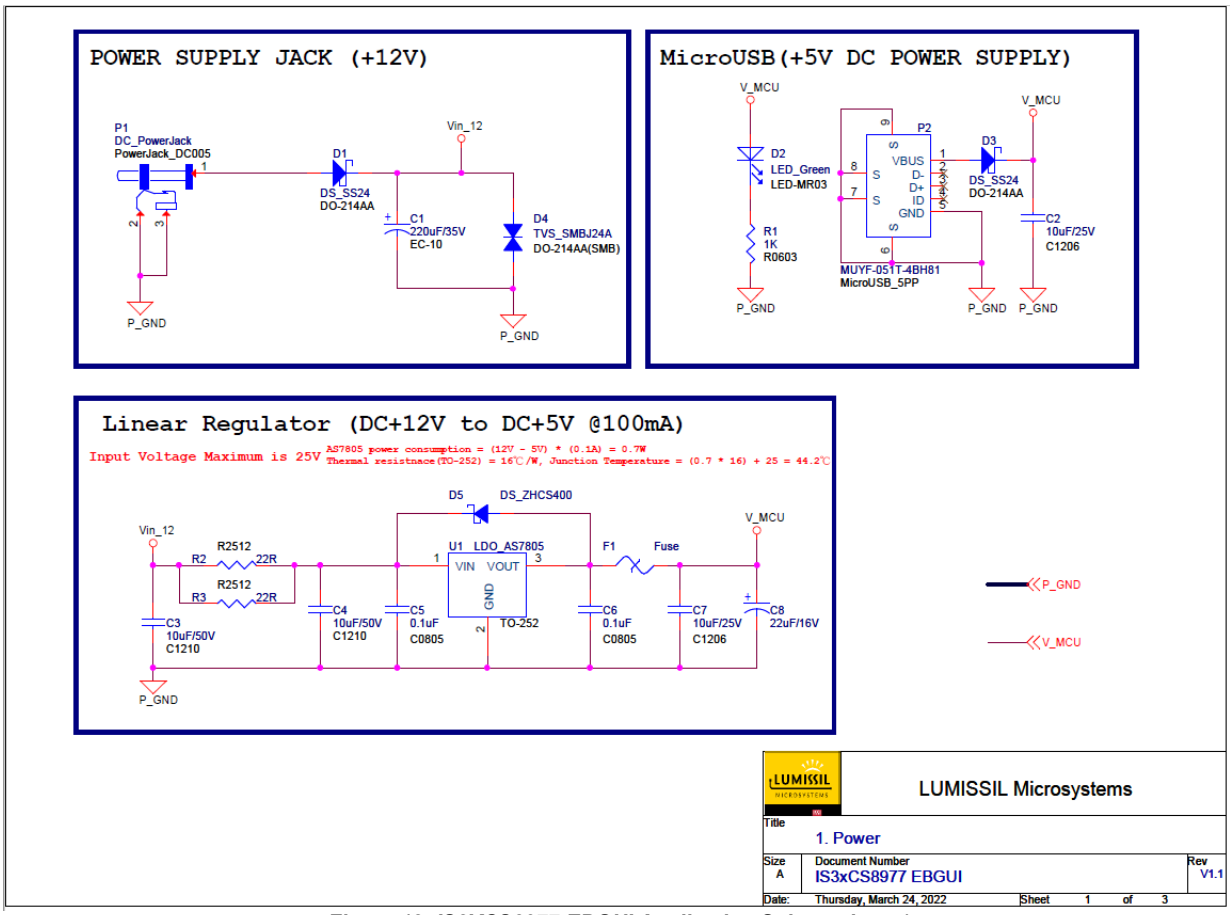


Figure 13: IS3XCS8977 EBGUI Application Schematics - 1

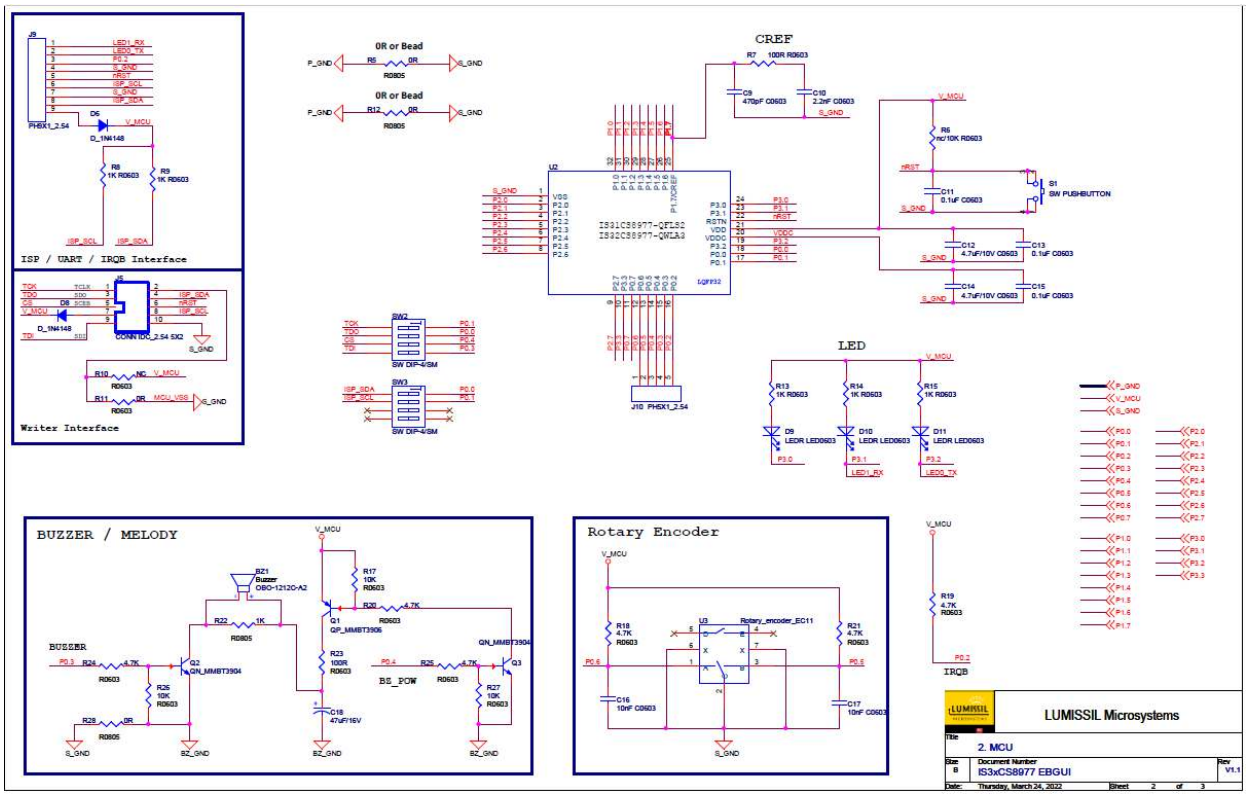


Figure 14: IS3XCS8977 EBGUI Application Schematics - 2

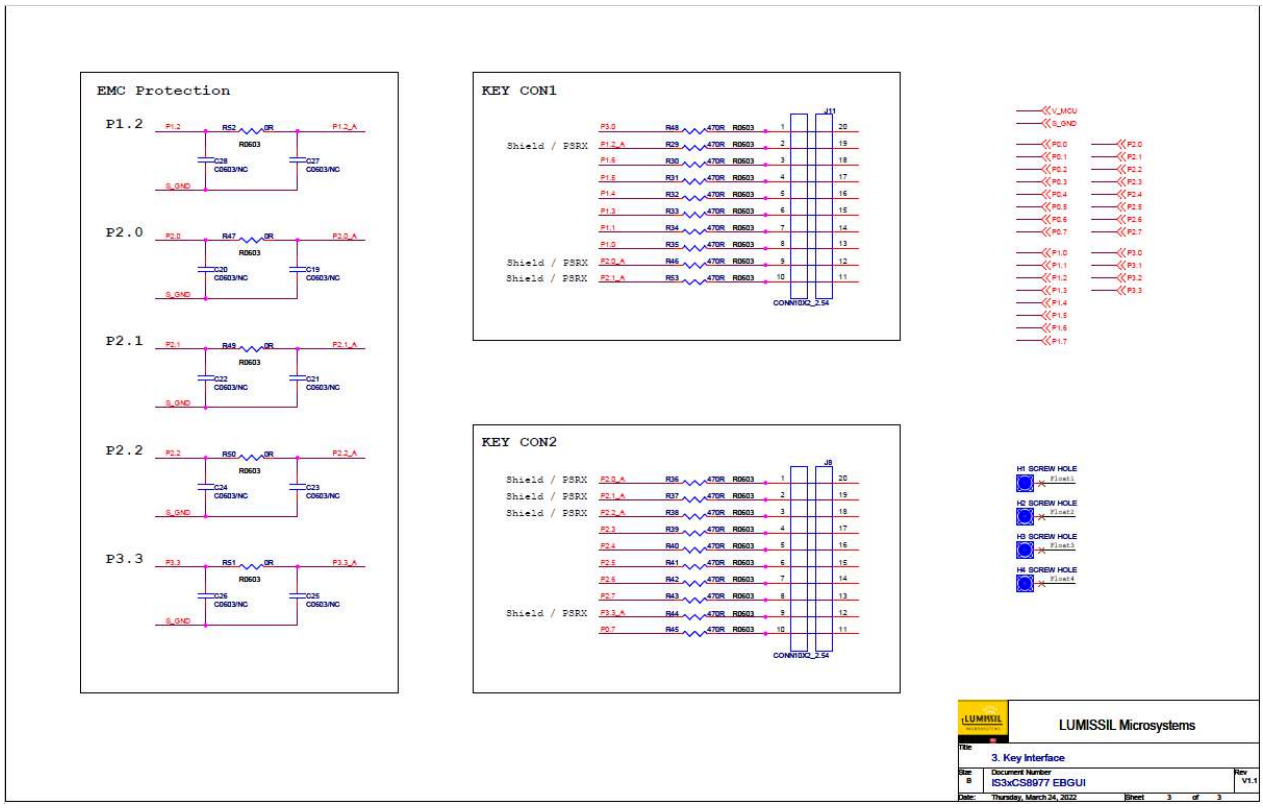


Figure 15: IS3XCS8977 EBGUI Application Schematics - 3

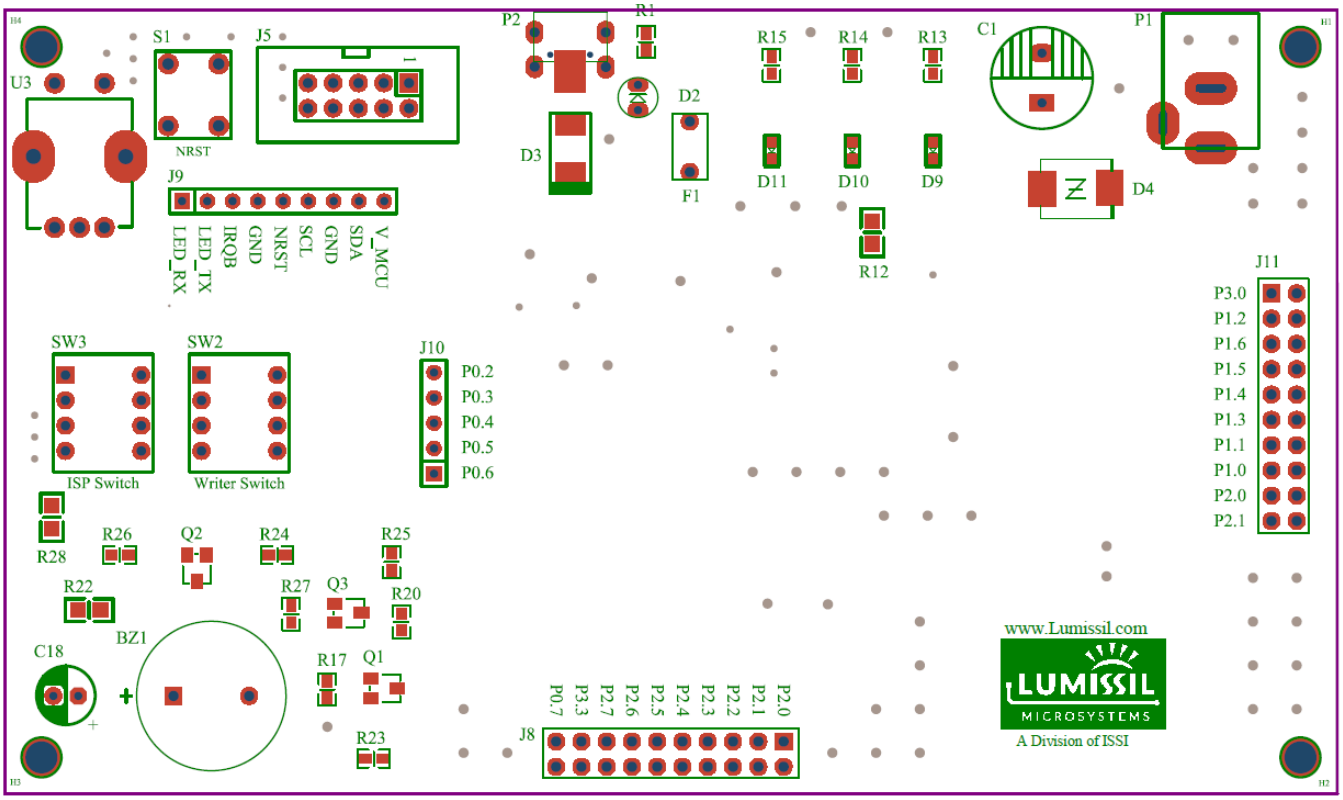


Figure 16: IS3XCS8977 EBGUI Component Placement - Top Layer

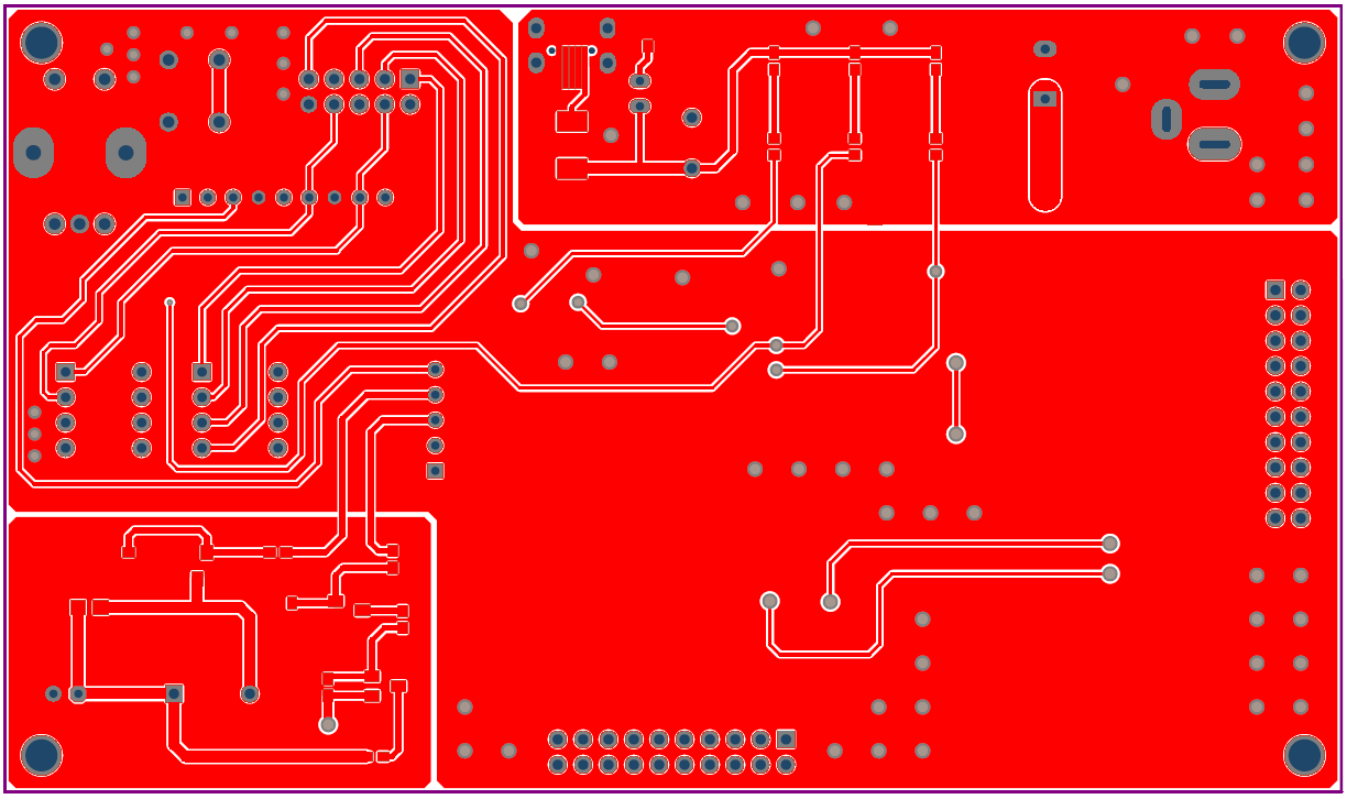


Figure 17: IS3XCS8977 EBGUI PCB Layout - Top Layer

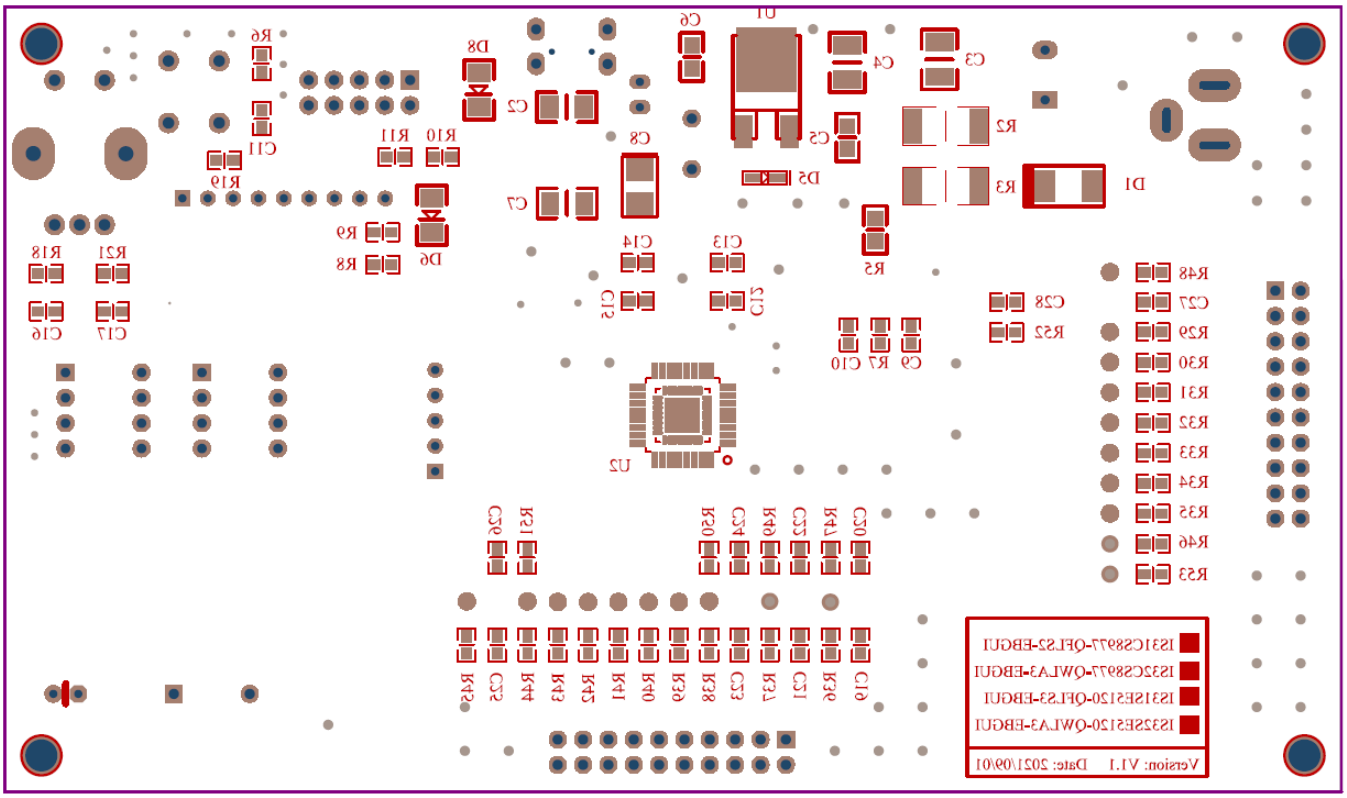


Figure 18: IS3XCS8977 EBGUI Component Placement - Bottom Layer

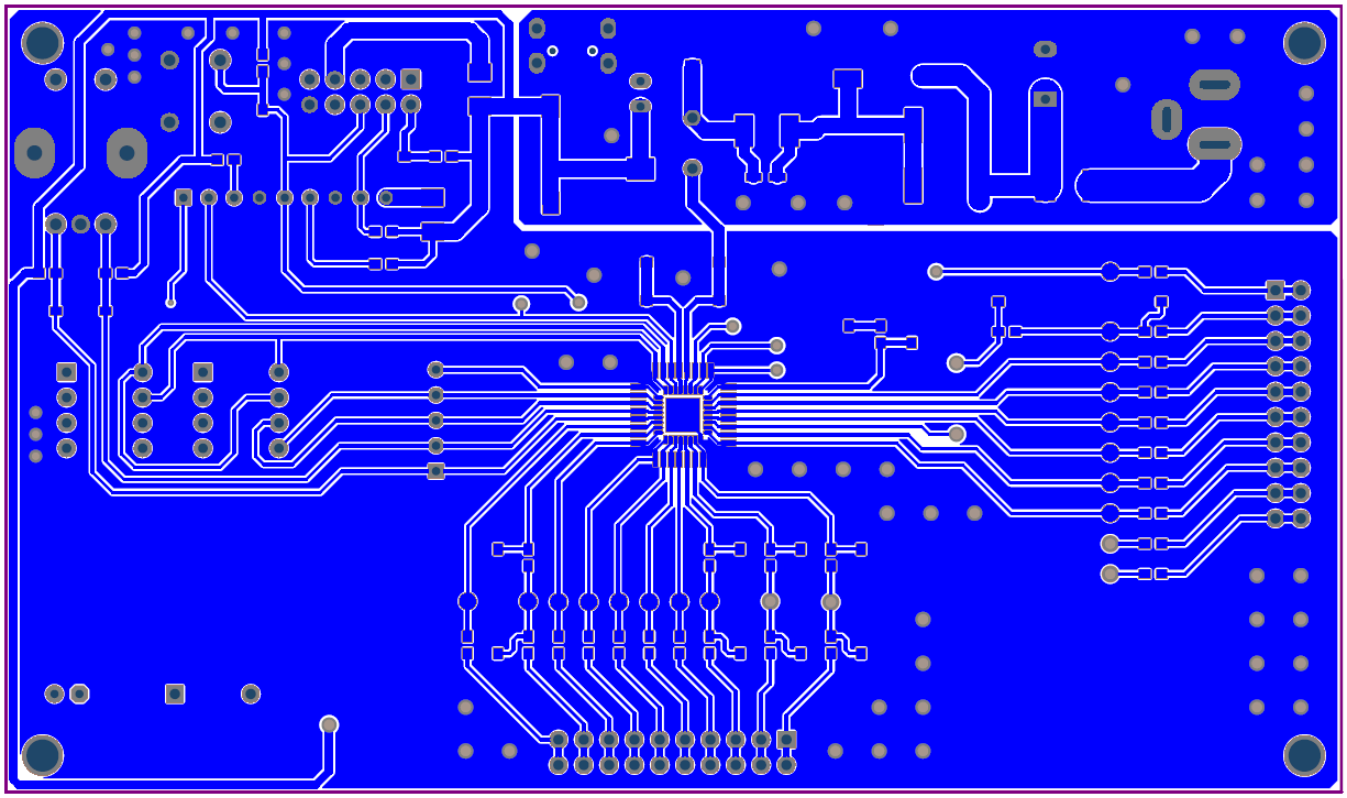


Figure 19: IS3XCS8977 EBGUI PCB Layout - Bottom Layer

BILL OF MATERIALS

Bill Of Materials					
IS31CS8977 EBGUI V1.1					
Item	Quantity	Reference	Part	PCB Footprint	Note
1	1	BZ1	OBO-1212C-A2		Buzzer
2	2	C3,C4	10uF/50V	1210	
3	2	C5,C6	0.1uF	0805	
4	1	C9	470pF	0603	
5	1	C10	2.2nF	0603	
6	3	C11,C13,C15	0.1uF	0603	
7	2	C12,C14	4.7uF/10V	0603	
8	1	C18	47uF/16V	DIP	
9	1	D1	DS_SS24	DO-214AA	
10	1	D2	LED_Red	DIP	
11	2	D6, D8	0R	1206	
12	3	D9,D10,D11	LED_Red	0603	
13	1	F1	Fuse	FUSE-30R110	Fuse
14	1	J5	CONN IDC_2.54 5X2	CONN_IDC_5X2_2.54	5X2
15	1	J8	CONN10X2_2.54	HEADER10X2	10X2(right angle)
16	1	J11	CONN10X2_2.54	HEADER10X2	10X2(right angle)
17	1	J9	PH9X1_2.54	PH9X1_2.54	9X1
18	1	J10	PH5X1_2.54	PH5X1_2.54	5X1
19	1	P1	DC_PowerJack	PowerJack_DC005	
20	1	Q1	QP_MMBT3906	SOT23-123	
21	2	Q2,Q3	QN_MMBT3904	SOT23-123	
22	2	R2,R3	22R	2512	
23	3	R5,R12,R28	0R	0805	
24	1	R7,R23	100R	0603	
25	6	R1,R8,R9,R13,R14,R15	1K	0603	
26	5	R47,R49,R50,R51,R52	0R	0603	
27	4	R6,R17,R26,R27	10K	0603	
28	4	R19,R20,R24,R25	4.7K	0603	
29	1	R22	1K	0805	
30	18	R29,R30,R31,R32,R33,R34,R35,R36,R37,R38, R39,R40,R41,R42,R43,R44,R45,R48	470R	0603	
31	2	SW2,SW3	SW DIP-4/SM	SW_EDS_SPST_4P	DIP Switch
32	1	S1	SW PUSHBUTTON	SW_TACT_4.5_SMD	Switch Button
33	1	U1	LDO_AS7805	TO-252	TO-252
34	1	U2	IS31CS8977-QFLS2-TR/IS32CS8977-LQLA3	QFN-32/LQFN-32	QFN-32/LQFN-32

Table 2: Bill of Materials

For Bill of Materials, please refer above Figure 13, Figure 14 and Figure 15.

REVISION HISTORY

Revision	Detailed Information	Date
A	Initial release	2022.04.12

APPENDIX I: IS3XCS8977 Test Code – GPIO toggle

```
#include "CS8977_SFR.h"
#include "CS8977_XFR.h"
#include "CS8977_MCU.h"
#include "CS8977_EUART2.h"
#include "Global.h"
#include <intrins.h>

//=====================================================================
/**
*     Initial_REGTRM
*
*     REGTRM value for 1.5V
*
*/
void Initial_REGTRM(unsigned char regtrm)
{
      TB = 0xAA;
      TB = 0x55;
      REGTRM = regtrm;
   TB = 0x00;
}
//=====================================================================
/**
*     Initial_IOSC
*
*     IOSC ITRM value and IOSC VTRM value for 16MHz
*
*/
void Initial_IOSC(unsigned char ITRM, unsigned char VTRM)
{
   TB = 0xAA;
   TB = 0x55;
   IOSCITRM = ITRM;
   TB = 0x00;

   Delay10ms(1);

   TB = 0xAA;
   TB = 0x55;
   IOSCVTRM = VTRM;
   TB = 0x00;
}

//=====================================================================
/**
*     IFB_Read_1Byte
*
*     Read 1 byte from Information block IFB
*
*/
unsigned char IFB_Read_1Byte(unsigned char ADD)
{
      unsigned char IFB_DAT;

      TB = 0xAA;
      TB = 0x55;
      FLSHADH = 0x00;
```

```
	FLSHADL = ADD;
	FLSHCMD = 0x02;    //IFB Read
	TB = 0x00;

	TB = 0xAA;
	TB = 0x55;
	IFB_DAT = FLSHDATL;
	TB = 0x00;

	return IFB_DAT;
}
//=====================================================================
/**
*	Reset_WDT3
*
*	Watchdog Timer3 Configuration
*
*/
void Reset_WDT3(void)
{
	TB = 0xAA;
	TB = 0x55;
	WDT3CF = 0xD0;  //clear WDT3 counter, stop WDT3 increment in STOP/SLEEP mode, clear Reset flag
	TB = 0x00;
}
//=====================================================================
/**
*	Initial_IO
*
*/
void Initial_IO(void)
{
	/* Initial Port0 / 1 / 2 / 3 */
	P0 = 0;
	P1 = 0;
	P2 = 0;
	P3 = 0;
	/* Port0 */
	IO_setting(&IOCFGO00, IO_PNDRV, IO_Input_EN, MFCFG_GPIO);
	IO_setting(&IOCFGO01, IO_PNDRV, IO_Input_EN, MFCFG_GPIO);
	IO_setting(&IOCFGO02, IO_PNDRV, IO_Input_EN, MFCFG_GPIO);
	//IO_setting(&IOCFGO03, IO_PNDRV, IO_Input_EN, MFCFG_GPIO);
	IO_setting(&IOCFGO04, IO_PNDRV, IO_Input_EN, MFCFG_GPIO);
	IO_setting(&IOCFGO05, IO_PNDRV, IO_Input_EN, MFCFG_GPIO);
	IO_setting(&IOCFGO06, IO_PNDRV, IO_Input_EN, MFCFG_GPIO);
	IO_setting(&IOCFGO07, IO_PNDRV, IO_Input_EN, MFCFG_GPIO);
	/* Port1 */
	IO_setting(&IOCFGO10, IO_PNDRV, IO_Input_EN, MFCFG_GPIO);
	IO_setting(&IOCFGO11, IO_PNDRV, IO_Input_EN, MFCFG_GPIO);
	IO_setting(&IOCFGO12, IO_PNDRV, IO_Input_EN, MFCFG_GPIO);
	IO_setting(&IOCFGO13, IO_PNDRV, IO_Input_EN, MFCFG_GPIO);
	IO_setting(&IOCFGO14, IO_PNDRV, IO_Input_EN, MFCFG_GPIO);
	IO_setting(&IOCFGO15, IO_PNDRV, IO_Input_EN, MFCFG_GPIO);
	IO_setting(&IOCFGO16, IO_PNDRV, IO_Input_EN, MFCFG_GPIO);
	IO_setting(&IOCFGO17, IO_PNDRV, IO_Input_EN, MFCFG_GPIO);
	/* Port2 */
	IO_setting(&IOCFGO20, IO_PNDRV, IO_Input_EN, MFCFG_GPIO);
	IO_setting(&IOCFGO21, IO_PNDRV, IO_Input_EN, MFCFG_GPIO);
	IO_setting(&IOCFGO22, IO_PNDRV, IO_Input_EN, MFCFG_GPIO);
	IO_setting(&IOCFGO23, IO_PNDRV, IO_Input_EN, MFCFG_GPIO);
```

```
	IO_setting(&IOCFGO24, IO_PNDRV, IO_Input_EN, MFCFG_GPIO);
	IO_setting(&IOCFGO25, IO_PNDRV, IO_Input_EN, MFCFG_GPIO);
	IO_setting(&IOCFGO26, IO_PNDRV, IO_Input_EN, MFCFG_GPIO);
	IO_setting(&IOCFGO27, IO_PNDRV, IO_Input_EN, MFCFG_GPIO);
	/* Port3 */
	IO_setting(&IOCFGO30, IO_PNDRV, IO_Input_EN, MFCFG_GPIO);
	//IO_setting(&IOCFGO31, IO_PNDRV, IO_Input_EN, MFCFG_GPIO);
	//IO_setting(&IOCFGO32, IO_PNDRV, IO_Input_EN, MFCFG_GPIO);
	IO_setting(&IOCFGO33, IO_PNDRV, IO_Input_EN, MFCFG_GPIO);
}
//=====================================================================
/**
*	Initial_Variable
*
*/
void Initial_Variable(void)
{
	/* Initial variable */
	u8_EUART2_cnt = 0;
}
//=====================================================================
/**
*	Delay10us
*
*/
/*
void Delay10us(unsigned char delay)
{
   unsigned char i, j;
	for(i=0; i<delay;i++)
		for (j = 0; j < 16; j++);
}
*/
//=====================================================================
/**
*	Delay1ms
*
*/
/*
void Delay1ms(unsigned char delay)
{
	unsigned char i, j, k;
	for(i=0; i<delay; i++)
		for(j=0; j<10; j++)
			for(k=0; k<200; k++);
}
*/
//=====================================================================
/**
*	Delay10ms
*
*/
void Delay10ms(unsigned char delay)
{
	unsigned char i, j, k;

	for(i=0; i<delay; i++)
		for(j=0; j<100; j++)
			for(k=0; k<200; k++);
}
```

```
//=====================================================================
/**
*     IO_setting
*
* Setting IO Configuration and Multi-Function(IOCFGOxx, IOCFGIxx, MFCFGxx)
*
*/
void IO_setting(unsigned char* xIOCFGO, unsigned IOCFGO_V, unsigned IOCFGI_V, unsigned MFCFG_V)
{
      unsigned char* tmp;
      tmp = xIOCFGO;
      tmp += 0x20;
      *tmp = MFCFG_V;
      tmp -= 0x10;
      *tmp = IOCFGI_V;
      tmp -= 0x10;
      *tmp = IOCFGO_V;
}

unsigned char u8_EUART2_cnt;
//=====================================================================
/**
*     Initial_EUART2
*
*     Initial EUART2 Configuration
*
*/
void Initial_EUART2(void)
{
      /* P0_7 : TXD2, P0_6: RXD2 */
      IOCFGO32 = IO_NDRV | IO_PDRV;
      IOCFGI32 = 0x00;
      MFCFG32 = MFCFG_TXD2;

      IOCFGO31 = IO_PU;
      IOCFGI31 = IO_Input_EN;
      MFCFG31 = MFCFG_RXD2;

      /* EUART2 Configuration */
      LINSBRL = BUAD_L;          // EUART2 baud rate Low byte
      LINSBRH = BUAD_H;          // EUART2 baud rate    High byte
      SCON2 = UART_SCON;               // EUART2 Configuration Register
      SFIFO2 = 0x00;             // Receive FIFO trigger threshold level = 0, Transmit FIFO trigger threshold level
                                  // = 0
      SCON2 = UART_SCON | 0x80;         // EUART2 Enable
}
//=====================================================================
/**
*     EUART2_tx_byte
*
*     EUART2 transmit 1 byte
*
*/
void EUART2_tx_byte(unsigned char p)
{
      while((SFIFO2 & 0x08));
      SBUF2 = p;
}
//=====================================================================
void main(void)
```

```
{
    unsigned char tmp;

    /* Disable Watchdog timer */
    TA = 0xAA; //Clear and disable watchdog
    TA = 0x55;
    WDCON = 0x00;
    TA = 0x00;

    /* Setup Wait Stat */
    TA = 0xAA;    //Wait State Cycle = 0
    TA = 0x55;
    WTST = 0;
    TA = 0x00;

    /* Regulator Trim & IOSC Trim */
    Initial_REGTRM(IFB_Read_1Byte(0x20));                          //Regulator Voltage
    Initial_IOSC(IFB_Read_1Byte(0x21), IFB_Read_1Byte(0x22));      //IOSC = 16MHz

    /* Initial variable */
    Initial_Variable();

    /* Initial EUART2 */
    Initial_EUART2();

    /* Initial IO */
    Initial_IO();

    EUART2_tx_byte(0x55);
    EUART2_tx_byte(0xAA);

    tmp = 0;

    while(1)
    {
        Reset_WDT3();    //Clear WDT3
        EUART2_tx_byte(tmp);
        /* Port 0 */
        P0_0 = !P0_0;
        P0_1 = !P0_1;
        P0_2 = !P0_2;
        //P0_3 = !P0_3;
        P0_4 = !P0_4;
        P0_5 = !P0_5;
        P0_6 = !P0_6;
        P0_7 = !P0_7;
        /* Port 1 */
        P1_0 = !P1_0;
        P1_1 = !P1_1;
        P1_2 = !P1_2;
        P1_3 = !P1_3;
        P1_4 = !P1_4;
        P1_5 = !P1_5;
        P1_6 = !P1_6;
        P1_7 = !P1_7;
        /* Port 2 */
        P2_0 = !P2_0;
        P2_1 = !P2_1;
        P2_2 = !P2_2;
        P2_3 = !P2_3;
```

```
            P2_4 = !P2_4;
            P2_5 = !P2_5;
            P2_6 = !P2_6;
            P2_7 = !P2_7;
            /* Port 3 */
            P3_0 = !P3_0;
            //P3_1 = !P3_1;
            //P3_2 = !P3_2;
            P3_3 = !P3_3;
            tmp++;
        }
}
```