IS31CS8975 EB Board User Guide

GENERAL DESCRIPTION

CS8975 is a general-purpose MCU with 16KB Code e-Flash memory with ECC, 1K SRAM with ECC. The embedded flash for code storage has built-in ECC that corrects 1-bit error and detects two-bit errors. CPU accesses the e-Flash through program address read and through Flash Controller which can perform software read/writer operations of e-Flash.

CPU in CS8975 is 1-T 8051 with enhanced multiplication and division accelerator. There are two clock sources for the system, one is a 16MHz/32MHz IOSC (manufacturer calibrated +/- 2%) and another one is 128KHz SOSC. Both 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 SOSC. An 8-bit/16-bit checksum and 16-bit CRC accelerator is included. There are EUART/LIN controller and I2C master/Slave controller as well as SPI master/slave controller. The interfaces of these controllers are multiplexed with GPIO pins. Other useful peripherals include a buzzer control, 6 channels of 12-bit PWM, and one channel of 16-bit timer/capture and quadrature decoder.

Analog peripherals include an 11-bit ADC with an internal temperature sensor, an 8-bit voltage output DAC, and four analog comparators with a programmable threshold. A touch key controller with up to 20-bit resolutions is also included. The touch key controller also has shield output capability for moisture immunity. The touch key controller allows sleep mode (under 10uA) and uses auto-detection for wakeup. The maximum number of key inputs can be scanned is 11.

CS8975 also 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 breakpoint processor also allows easy debugging which can be integrated with ISP. Reliable power-on-reset circuits and low supply voltage detection allow reliable operations under harsh environments.

FEATURES

CPU and Memory

- 1-Cycle 8051 CPU core up to 32MHz
- 16-bit Timers T0/T1/T2/T3/T4 and 24-bit T5
- Checksum and CRC accelerator
- WDT1 by SYSCLK, WDT2/WDT3 by SOSC
- Clock fault monitoring
- Up to 6 external interrupts shared with GPIO pins
- Power saving modes – Normal, STOP, and SLEEP modes
- 256B IRAM and 768B XRAM with ECC check
- 16KB Code e-Flash with ECC and two 512x16 Information Block
  - Program read with hardware ECC
  - Software read/write direct access 16-bit wide
  - Code security and data loss protection
  - 100K endurance and 10 years retention

Clock Sources

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

Digital Peripherals

- 6 CH 8/10/12-bit center-aligned PWM controller
  - Trigger interrupt and ADC conversion
  - Output polarity
- One 16-bit Timer/Capture and One 16-bit quadrature decoder
- Buzzer/Melody generator
- One I^2C Master
- One I^2C 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 11 key inputs
  - Shield output for moisture immunity
  - Low power sleep mode wakeup (<5uA).
- 11-Bit SAR ADC with GPIO analog input
  - Temperature sensor and supply measurement
- 8-Bit DAC and four analog comparators
- Power on reset and Low voltage detect (2.3V-4.5V)

Miscellaneous

- Up to 12 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/125°C
- SOP-8/TSSOP-16 package
- RoHS & Halogen-Free compliant package

QUICK START

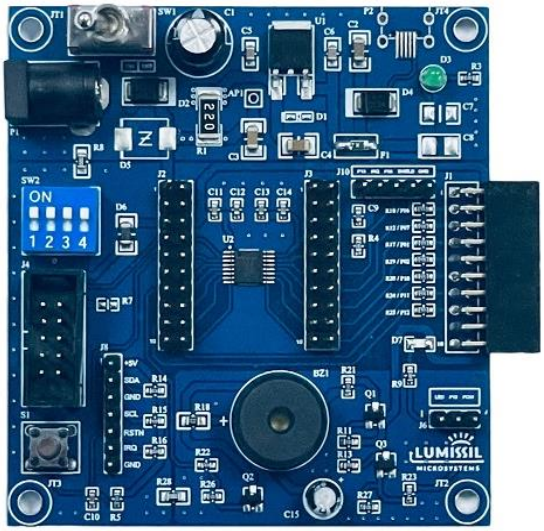


Figure 1: Photo of IS31CS8975 EB Top View

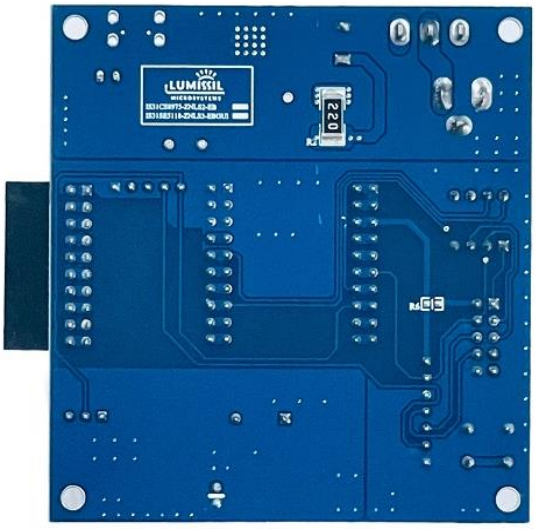


Figure 2: Photo of IS31CS8975 EB Bottom View

RECOMMENDED EQUIPMENT

- 9V~15V, 1.0A power supply

ABSOLUTE MAXIMUM RATINGS

- ≤ 25V power supply

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

PROCEDURE

The IS31CS8975 evaluation 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.

For standalone IS31CS8975 EB operation, the user can plug in the power adapter into the power jack P1 and make sure SW1 is ON. For operation with EzISP, the external power adapter is not necessary.

EzISP PROGRAMMING BOARD OPERATION

The IS31CS8975 EB offers flexible flash programming that can be programmed via the Write Mode or Fast Write Mode of the EzISP Programming Board. The Write Mode and Fast Writer Mode of the EzISP Programming Board require 7 pins of hardware (RSTN, TCK, TMS, TDO, TDI, GND, VDD, etc.).

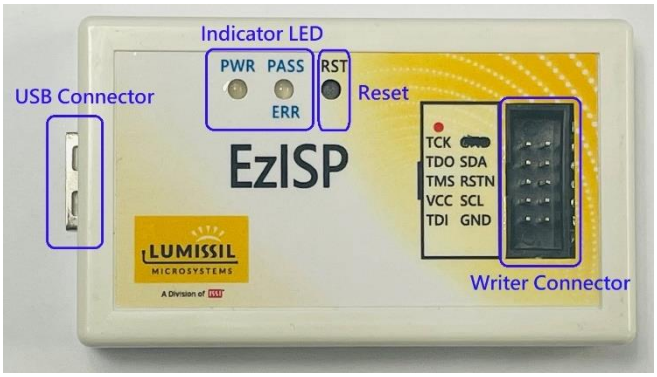


Figure 3: Photo of EzISP Programming Board

ORDERING INFORMATION

Part No.	Temperature Range	Package
IS31CS8975-ZNLS2-EB	-40°C to +85°C	TSSOP-16

Table 1: Ordering Information

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

BOARD ASSEMBLY

Please refer to the Bill of Materials for a description of components, *Figure 4 & 5* for components placement, and *Figure 11* for the EB Schematic.

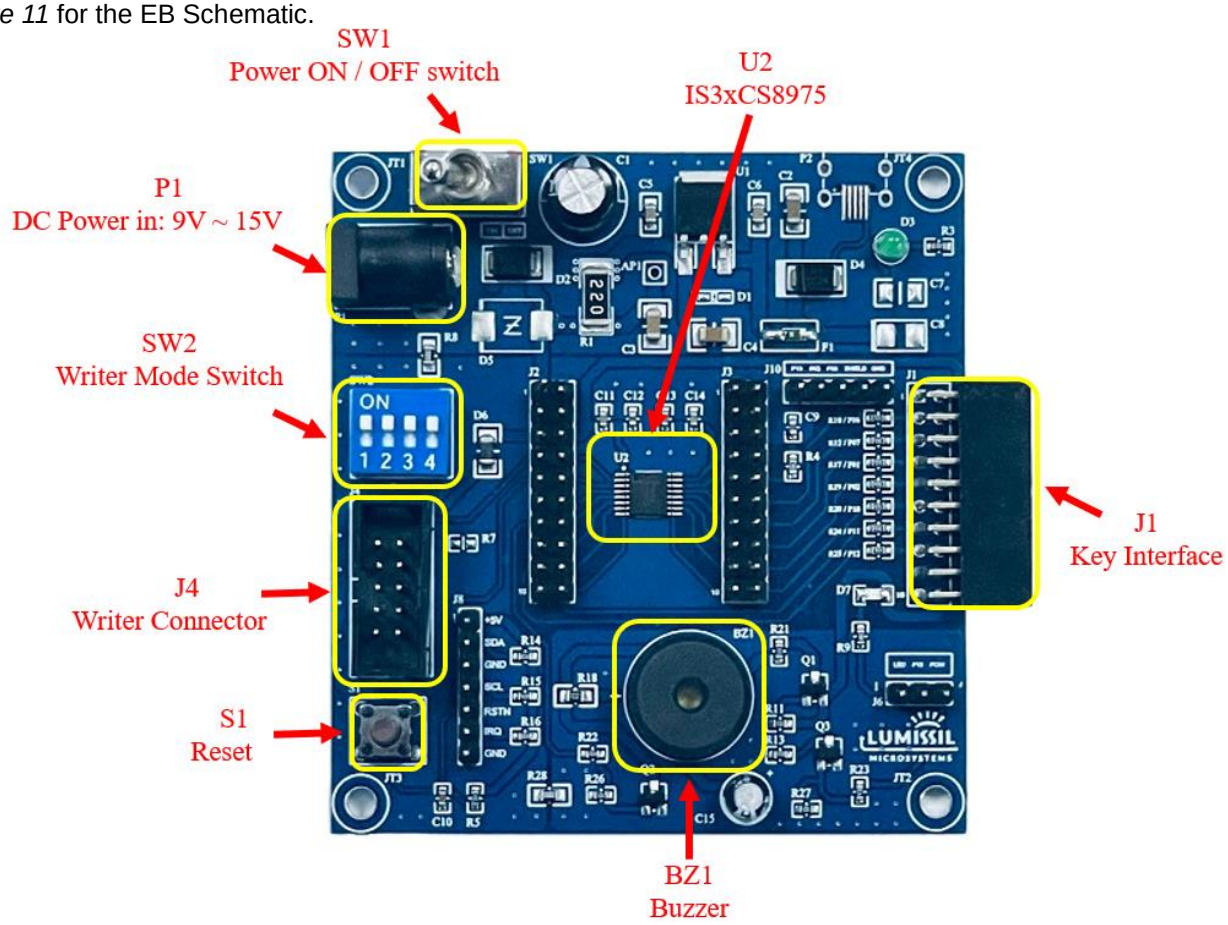


Figure 4: Photo of IS31CS8975 EB Assembly Top View

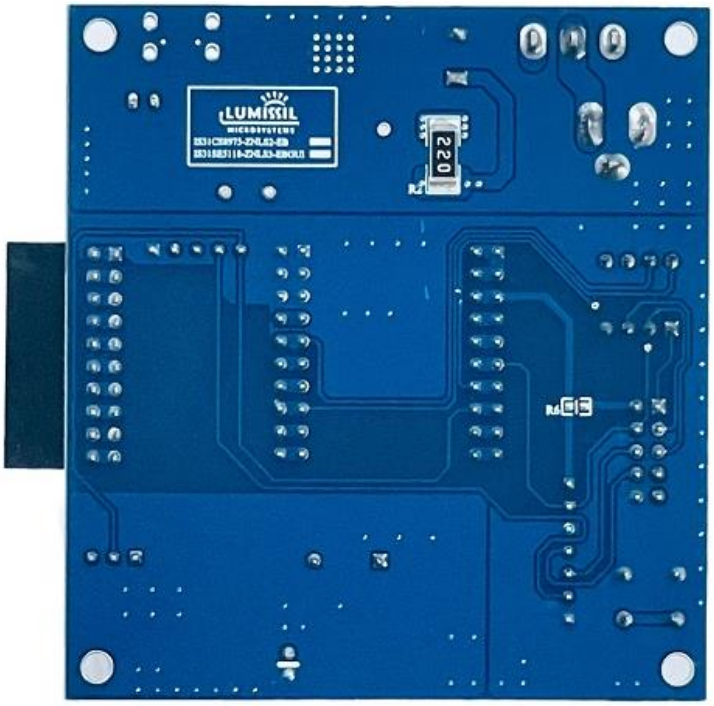


Figure 5: Photo of IS31CS8975 EB Assembly Bottom View

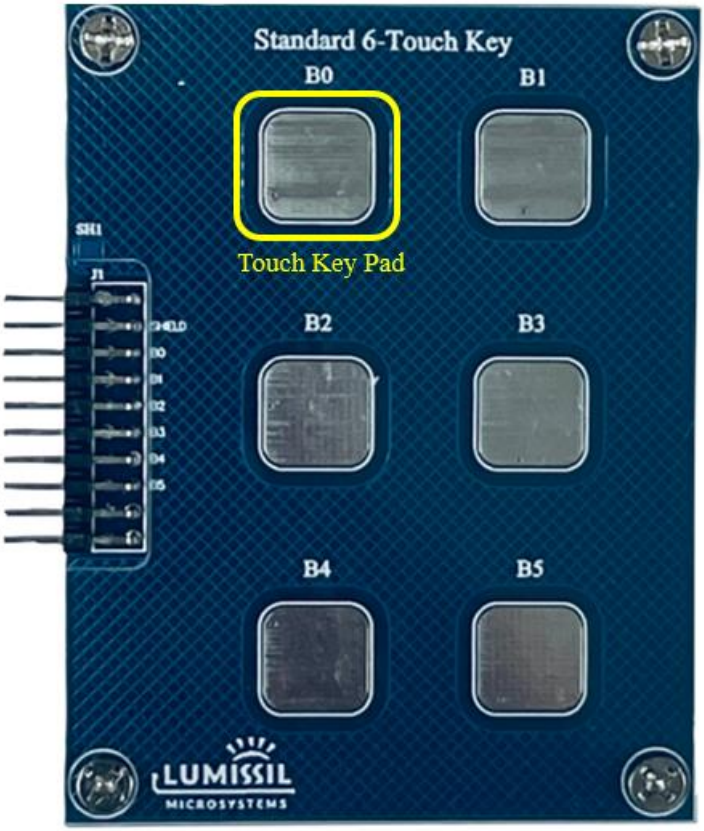


Figure 6: Photo of Accessory Board

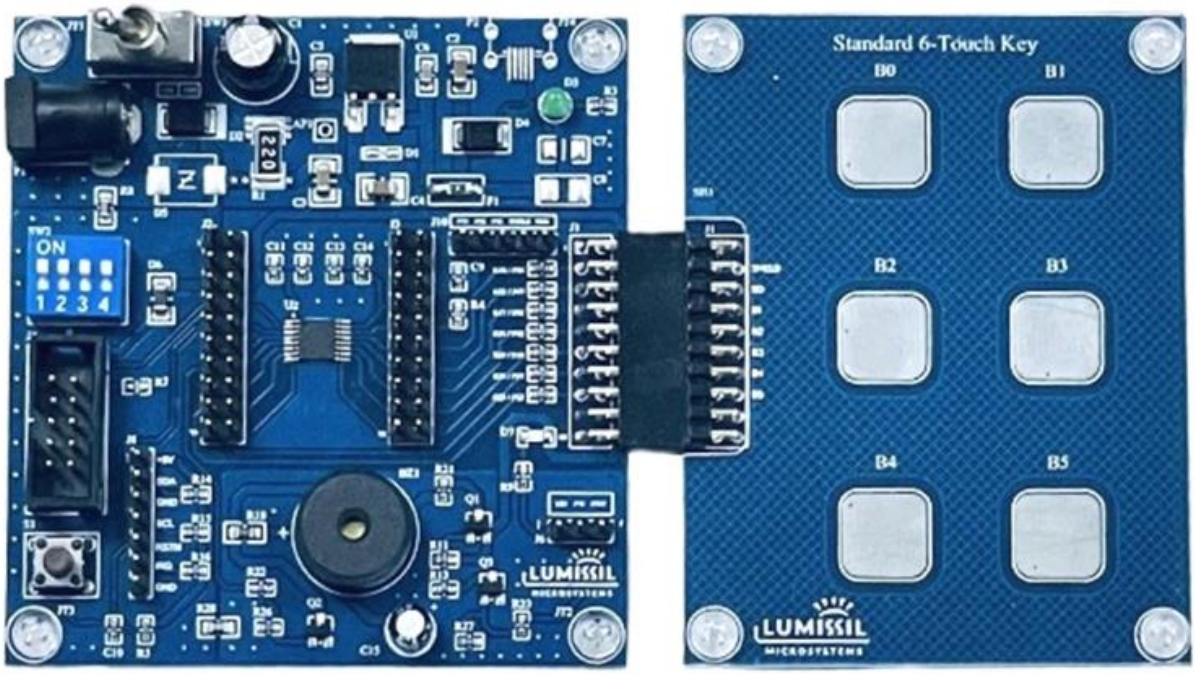


Figure 7: Photo of IS31CS8975 EB Environment

SOFTWARE SUPPORT

Before starting up IS31CS8975 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. Users can unzip the 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 / Win 7 / Win 8 / Win 10/Win 11 operating system.

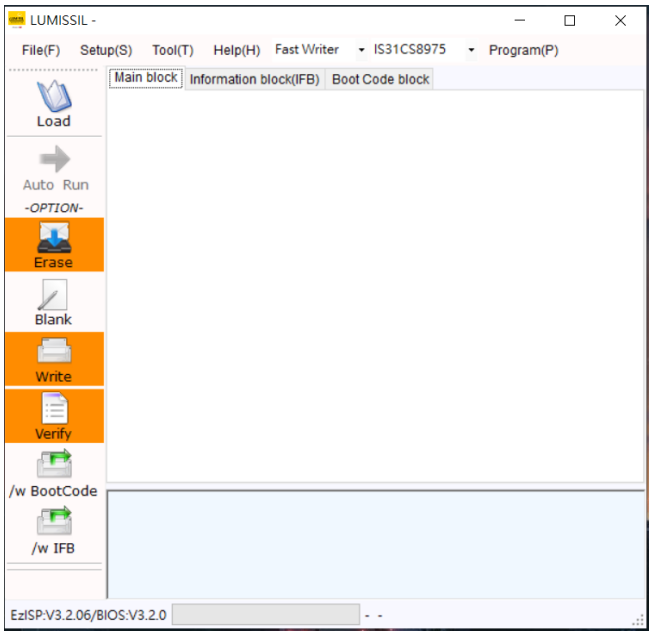


Figure 8: Photo of EzISP software operation interface

EzISP Writer Mode programming operation procedures are as follows:

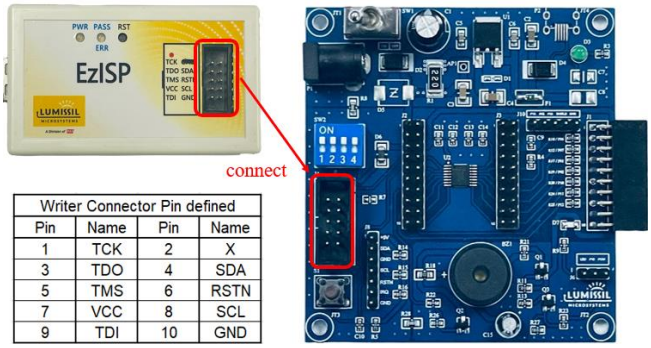
1) Connect USB cable between the write connector of the EzISP Programming Board and the USB port of your PC.
2) Use a 10-pin 2x5 Socket-Socket 1.27mm IDC cable from the writer connector on the EzISP Programming Board to the writer connector on the IS31CS8975 EB Board.
3) Turn all switches of DIP switch (SW2) to ON position.
4) Execute 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, 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, the IS31CS8975 chip will start running the program.

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

EzISP ISP Mode programming operation process is as following items:

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 IS31CS8975 EB.
3) Run EzISP software.
4) Select the MCU chip type and programming mode (for example: ISP Mode).
5) Click the "Load" button and select the programming code (*.hex) to load.
6) 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.



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 9: Photo of EzISP Programming Board and IS31CS8975 EB Board connection

The steps listed below are examples of GPIO control using the IS31CS8975.

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 in the EzISP software to update the firmware to the IS31CS8975 flash.

3) After the firmware update is complete, the IS31CS8975 chip will automatically reset and execute the program.

4) In this example, you can measure if the P02 pin on the IS31CS8975 EB Board has been toggled.

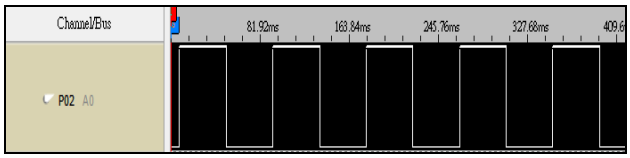


Figure 10: Photo of P02 pin toggling on the IS31CS8975 EB Board

SCHEMATICS OF IS31CS8975 EB BOARD

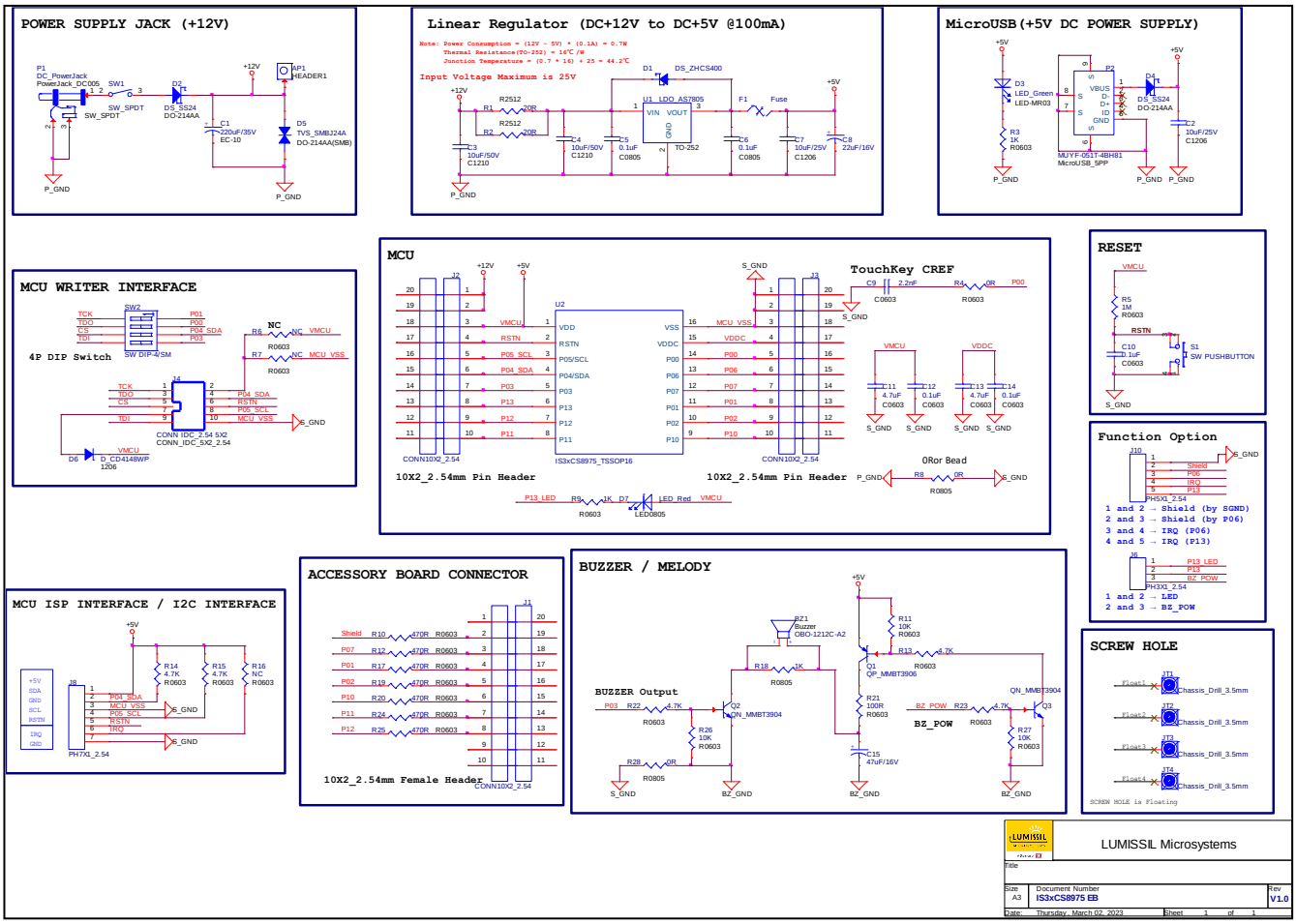


Figure 11: IS31CS8975 Evaluation Board Application Schematic

Note:

The DIP switch SW2 should be set to all OFF if ISP mode is selected (boot code pre-programmed)

The DIP switch SW2 should be set to all ON if writer mode is selected (The boot code will be erased, and ISP won't be available

BILL OF MATERIALS

Item	Q'ty	Symbol	Value	PCB Footprint	Description
1	1	AP1	TP/NC	HEADER1	1x1 Header
2	1	BZ1	OBO-1212C-A2	BUZZER$2F$2P-3$2F$7.6MM	NC
3	1	C1	100uF/50V	EC-10	Capacitance
4	2	C2,C7	10uF/25V	C1206	SMD Capacitance
5	2	C3,C4	10uF/50V	C1210	SMD Capacitance
6	2	C5,C6	0.1uF	C0805	SMD Capacitance
7	1	C8	22uF/16V	C3528-21	SMD Capacitance
8	1	C9	2.2nF	C0603	SMD Capacitance
9	3	C10,C12, C14	0.1uF	C0603	SMD Capacitance
10	2	C11,C13	4.7uF	C0603	SMD Capacitance
11	1	C15	47uF/16V	C_AL6.3_7	Capacitance
12	1	D1	DS_ZHCS400	SOD323-Z	NC
13	2	D2,D4	DS_SS24	DO-214AA	Dioed
14	1	D3	LED_Green	LED-MR03	LED Green
15	1	D5	TVS_SMBJ24A	DO-214AA(SMB)	NC
16	1	D6	0R	D1206	0R or Bead
17	1	D7	LED_Red	LED0805	LED Red
18	1	F1	Fuse	FUSE-30R110	1.5A Fuse
19	4	JT1,JT2, JT3,JT4	Chassis_Drill_3.5mm	MTHOLE1	
20	1	J1	CONN10X2_2.54	HEADER10X2	10x2 Header
21	2	J2,J3	CONN10X2_2.54	PH10X2_2.54	10x2 Header
22	1	J4	CONN IDC_2.54 5X2	CONN_IDC_5X2_2.54	Writer connector
23	1	J6	PH3X1_2.54	PH3X1_2.54	3x1 Header
24	1	J8	PH7X1_2.54	PH7X1_2.54	7x1 Header
25	1	J10	PH5X1_2.54	PH5X1_2.54	5x1 Header
26	1	P1	DC_PowerJack	PowerJack_DC005	DC PowerJack
27	1	P2	MUYF-051T-4BH81	MIRCO_USB5PP	NC
28	1	Q1	QP_MMBT3906	SOT23-123	PNP
29	2	Q2,Q3	QN_MMBT3904	SOT23-123	NPN
30	2	R1,R2	20R	R2512	SMD Resistor
31	2	R3,R9	1K	R0603	SMD Resistor
32	1	R4	0R	R0603	SMD Resistor
33	1	R5	1M	R0603	SMD Resistor

34	2	R6,R7	NC	R0603	NC
35	2	R8,R28	0R	R0805	SMD Resistor
36	7	R10,R12, R17,R19, R20,R24, R25	470R	R0603	SMD Resistor
37	3	R11,R26, R27	10K	R0603	SMD Resistor
38	6	R13,R14, R15,R16, R22,R23	4.7K	R0603	SMD Resistor
39	1	R18	1K	R0805	SMD Resistor
40	1	R21	100R	R0603	SMD Resistor
41	1	SW1	SW_SPDT	SW_SPDT	SPDT Switch
42	1	SW2	SW DIP-4/SM	SW_EDS_SPST_4P	Switch
43	1	S1	SW PUSHBUTTON	SW_TACT_4.5_SMD	Push Button
44	1	U1	LDO_AS7805	TO-252	LDO
45	1	U2	IS31CS8975_TSSOP16	TSSOP16_IS31CS8975	MCU

Bill of Materials, refer to Figure 11

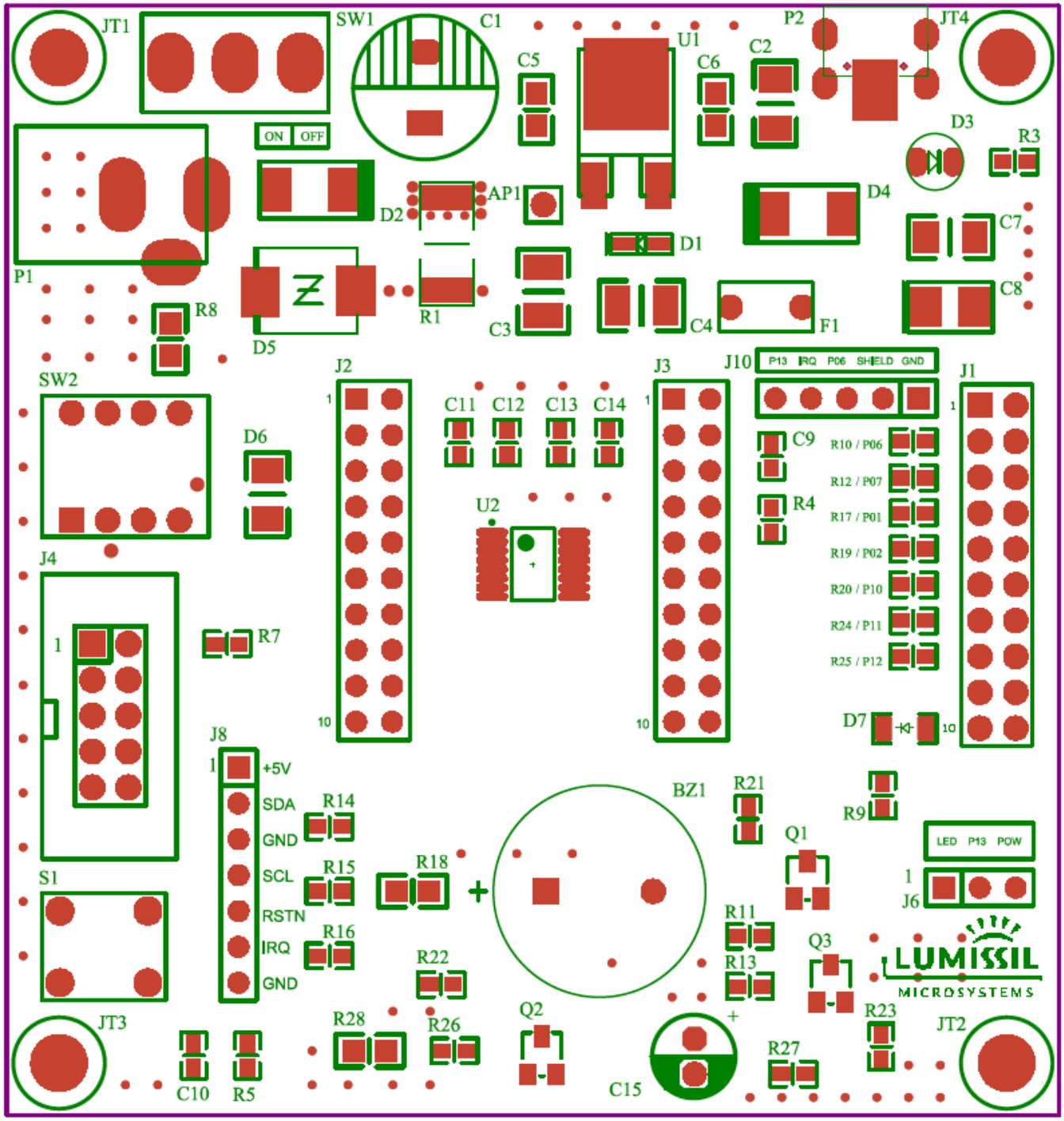


Figure 12: IS31CS8975 Evaluation Board Component Placement Guide - Top Layer

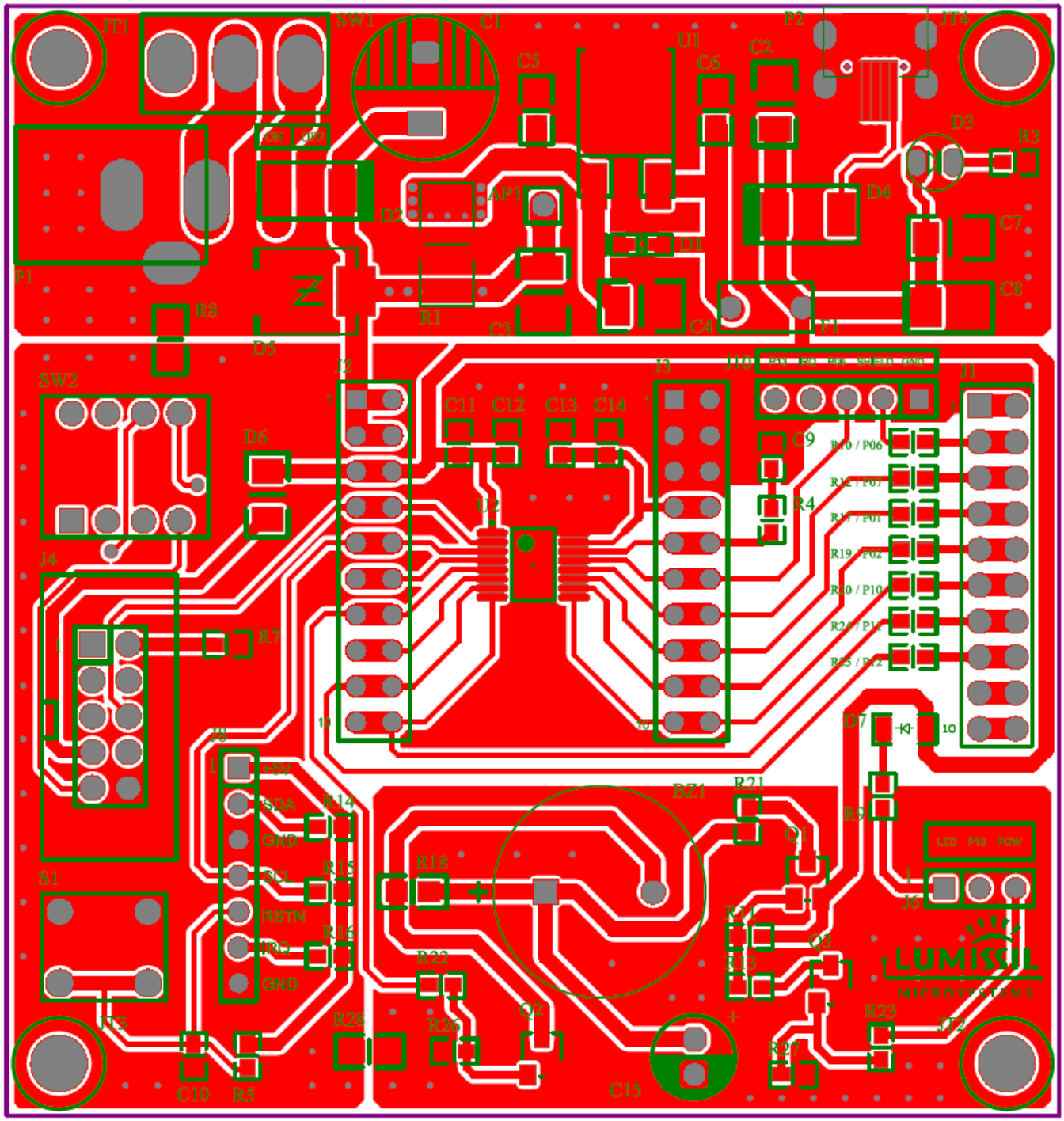


Figure 13: IS31CS8975 Evaluation Board PCB Layout - Top Layer

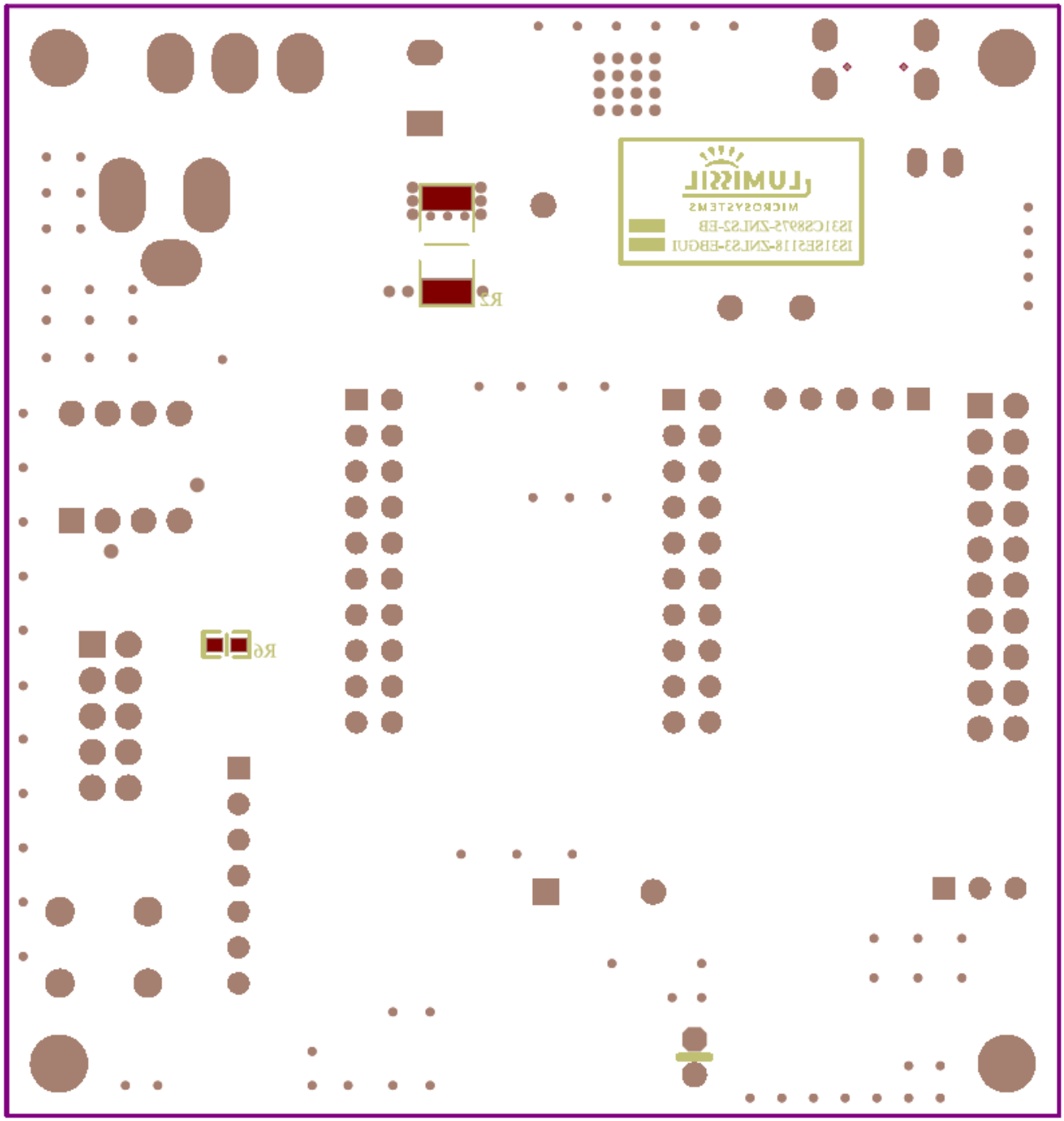


Figure 14: IS31CS8975 Evaluation Board Component Placement Guide - Bottom Layer

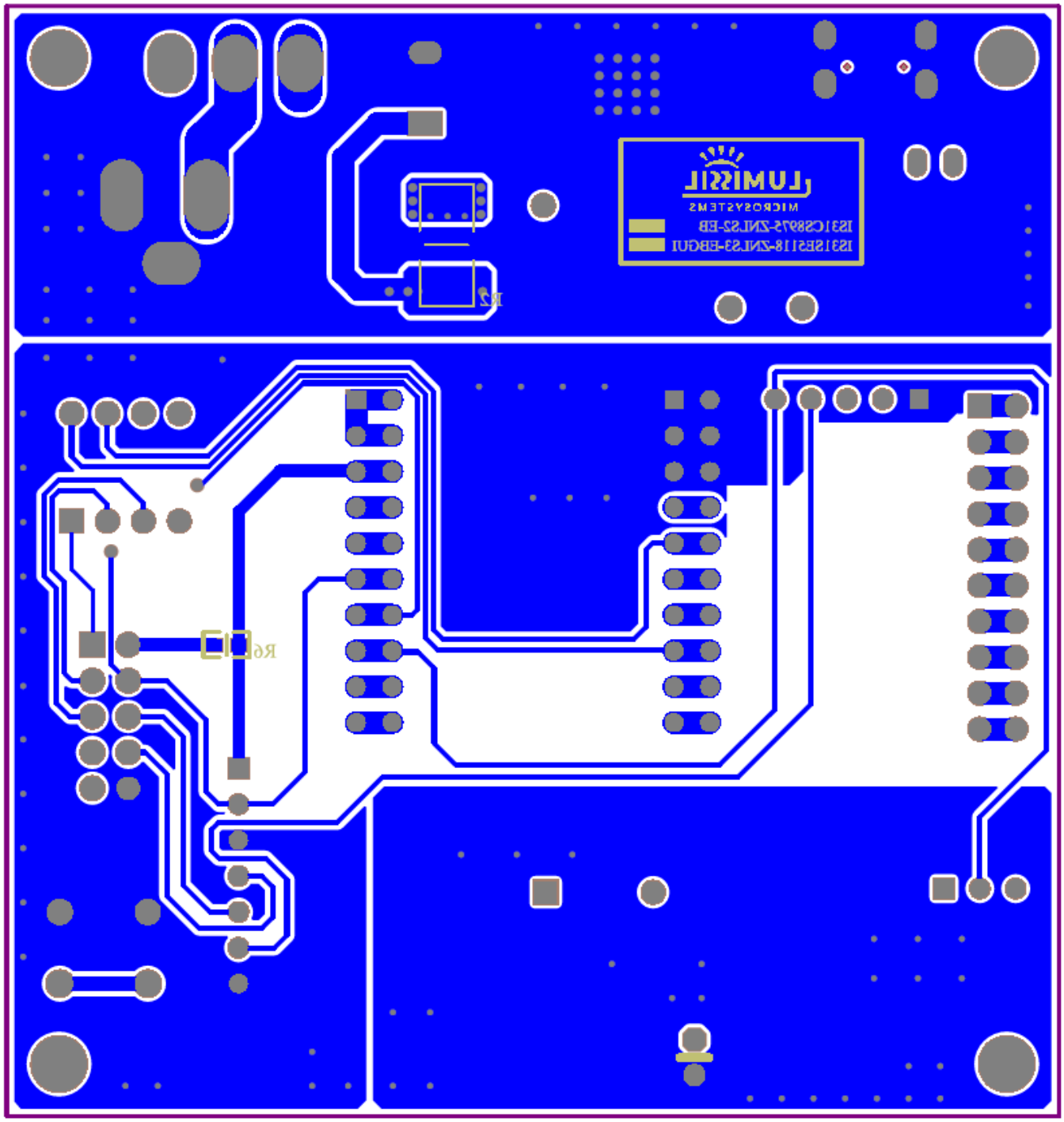


Figure 15: IS31CS8975 Evaluation Board PCB Layout - Bottom Layer

REVISION HISTORY

Revision	Detailed Information	Date
A	Initial release	2020.06.24
B	1. Add Figures 4 & 5 for Photos of IS31CS8975 EB Assembly Top and Bottom Views 2. Add Figures 6 for Photo of Accessory Board 3. Add Figures 7 for Photo of Connection of IS31CS8975 EB and Accessory Board 4. Add ISP mode procedures 5. Add Win 11 support for IS31CS8975 EB software	2023.03.06

APPENDIX Ⅰ: IS31CS8975 Test Code – GPIO toggle

```
/***************************************************/
/**
*       IS31CS8975 Sample Code
* Function: GPIO Toggle Test
* Version: 1.0
* Copyright(c)      All right reserved
*/
/***************************************************/
#define _main_

#include "CS8975_SFR.h"
#include "CS8975_XFR.h"
#include <intrins.h>

/* SYSCLK */
#define SYSCLK          iSYSCLK
#define iSYSCLK         16000000
#define xSYSCLK             22118400
#define SOSC                128000

/* EUART configuration macro*/
#define baud_rate 115200
#define BUAD_H (Uint8)((SYSCLK/baud_rate)/256) // MSB
#define BUAD_L (Uint8)(SYSCLK/baud_rate)

#define UART_SB 0           // 0: 1 Stop bit, 1: 2 Stop bit
#define UART_WLS 3          // frame length = WordLength+5 range 0~3
#define UART_Break 0        // set TX as low if 1 until Break ast as 0
#define UART_OP 0           // 0: even parity, 1: odd parity
#define UART_PE 0           // 0: no parity, 1: parity enable
#define UART_SP 0           // 1: always set parity bit as 1

#define UART_SCON (UART_SB<<6)+(UART_WLS<<4)+(UART_Break<<3)+(UART_OP<<2)+(UART_PE<<1)+UART_SP

#define dfn8_RX_CMD_length 3

//=============================================================================
/**
*       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++);
}
//===========================================================================
/**
*       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;
}
//=============================================================================
/**
*	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;
}
//================================================================================
/**
*    Initial_IO
*
*/
void Initial_IO(void)
{
    /* Initial Port0 / 1 / 2 / 3 */
    P0 = 0;
    P1 = 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);
}
//================================================================================
/**
*    Initial_EUART1
*
*    Initial EUART1 Configuration
*
*/
void Initial_EUART1(void)
{
    /* P0_7 : TXD1, P0_6: RXD1 */
    IOCFGO07 = 0x60;
    IOCFGI07 = 0x00;
    MFCFG07 = 0x0A;
    IOCFGO06 = 0x02;
    IOCFGI06 = 0x08;
    MFCFG06 = 0x0B;
```

```
        SBR1L = BUAD_L;
        SBR1H = BUAD_H;
        SCON1 = UART_SCON;
        SFIFO1 = 0x30;   //RX FIFO = 3, TX FIFO = 0
        SCON1 = UART_SCON | 0x80;
}
//=============================================================================
/**
*       EUART1_tx_byte
*
*       EUART1 transmit 1 byte
*
*/
void EUART1_tx_byte(unsigned char p)
{
        while((SFIFO1 & 0x08));
        SBUF1 = 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 EUART */
        Initial_EUART1();

        /* Initial IO */
        Initial_IO();

        EUART1_tx_byte(0x55);
```

```
        EUART1_tx_byte(0xAA);

        tmp = 0;

        while(1)
        {
                Reset_WDT3();   //Clear WDT3
                EUART1_tx_byte(tmp);
                P0_0 = !P0_0;
                P0_1 = !P0_1;
                P0_2 = !P0_2;
                P0_4 = !P0_4;
                P0_5 = !P0_5;
                P0_6 = !P0_6;
                P0_7 = !P0_7;
                P1_0 = !P1_0;
                P1_1 = !P1_1;
                P1_2 = !P1_2;
                P1_3 = !P1_3;
                tmp ++;
        }
}
```