Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

THIS SPEC IS OBSOLETE

Spec No: 002-05226

Spec Title: AN205226 - F2MC-8FX Family MB95F430H Series Capacitance Touch Sensor Use TSC2_Demo_434.Lib

Replaced by: NONE

AN205226

F²MC-8FX Family MB95F430H Series Capacitance Touch Sensor Use TSC2_Demo_434.Lib

This application note describes Cypress TSC solution, and describes how to use TSC library and TSC GUI.

Contents

1 Introduction....1
1.1 Purpose....1
1.2 Definitions, Acronyms and Abbreviations....1
1.3 Document Overview....1
2 Solutions....1
2.1 AD Discharge Solution....1
2.2 Relaxation Oscillation Solution....4
3 Library....6
3.1 Library Overview....6
3.2 Parameters Setup....7
3.3 Application Interface....9
3.4 Cypress TSC Performance....11
3.5 How to Add Cypress TSC.lib....12
3.6 How to use Cypress TSC.lib....15
4 LIB Usage Notice....17
5 Additional Information....18
A Appendix....19
A.1 Sample code....19

OBSOLETE

1 Introduction

1.1 Purpose

This application note describes Cypress TSC solution, and describes how to use TSC library and TSC GUI.

1.2 Definitions, Acronyms and Abbreviations

1.3 Document Overview

The rest of document is organized as the following:

Chapter 2 explains the working principles of used solution.

Chapter 3 explains how to use TSC library.

Chapter 4 explains LIB usage notice.

2 Solutions

This chapter introduces working principles of TSC solutions used in TSC2_EVB_434.LIB.

2.1 AD Discharge Solution

The theory of capacitance touch sensor is to check capacitance increment. As follows, when finger not touch the C = Cp, when touching the C = Cp +Cғ, and the increment is ΔC = Cғ. This ΔC will affect the electrode charging-up time.

The AD discharge method detects the touch action by compare the voltage difference at the same time.

Let V_1 is voltage of electrode at time t in finger not touch situation. When finger touching, speed of electrode discharging will be delayed (because of ΔC). At the same time t, the voltage is V_2 which larger than V_1. MCU can detect the touch action by compare this voltage change.

Figure 1. AD Discharge method Physical Diagram

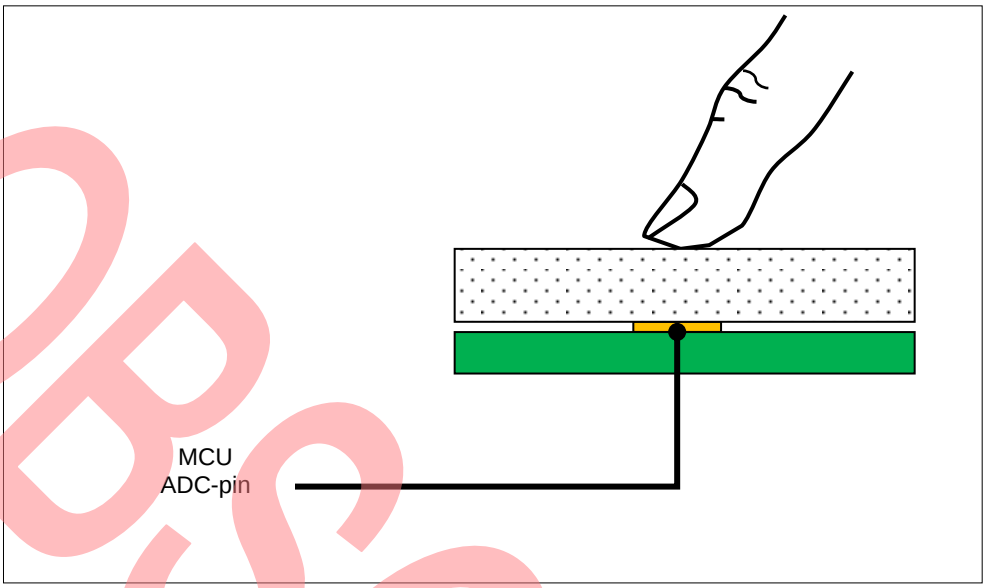
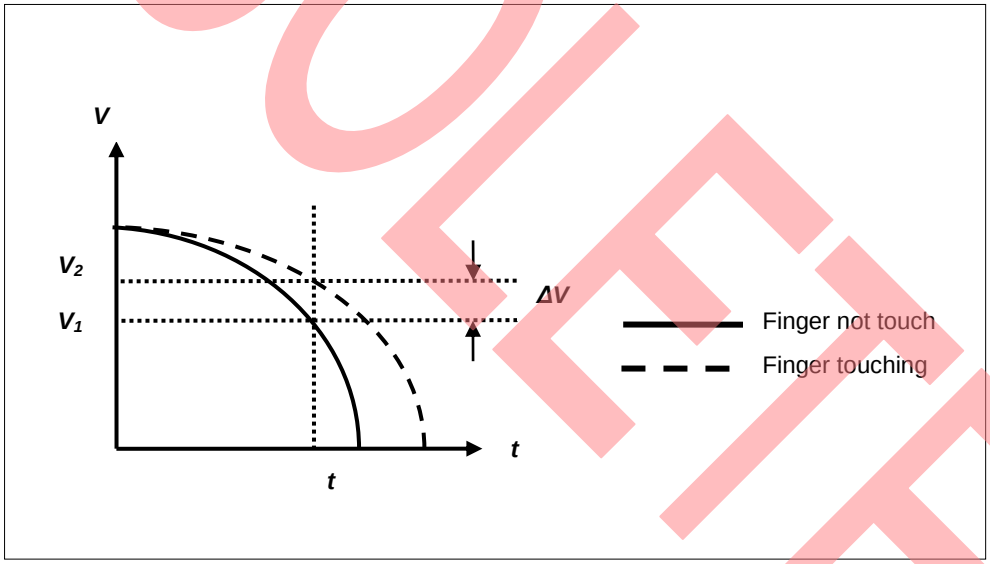


Figure 2. AD Discharge method Curve Graph



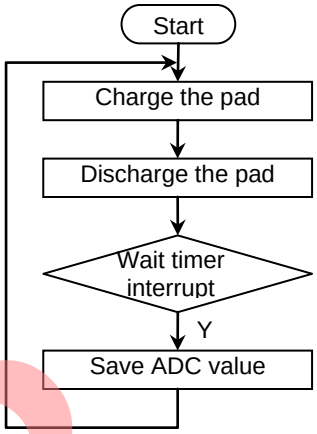
OBSOLETE

The AD Discharge method includes the following steps:

1. Set pin to input (high impedance) to charge the pad
2. Set pin to output 'L' to discharge the pad
3. Get the ADC value periodically
4. Save ADC value
5. Start next sample

The flowchart shown as below

Figure 3. AD Discharge method Check Flowchart



OBSOLETE

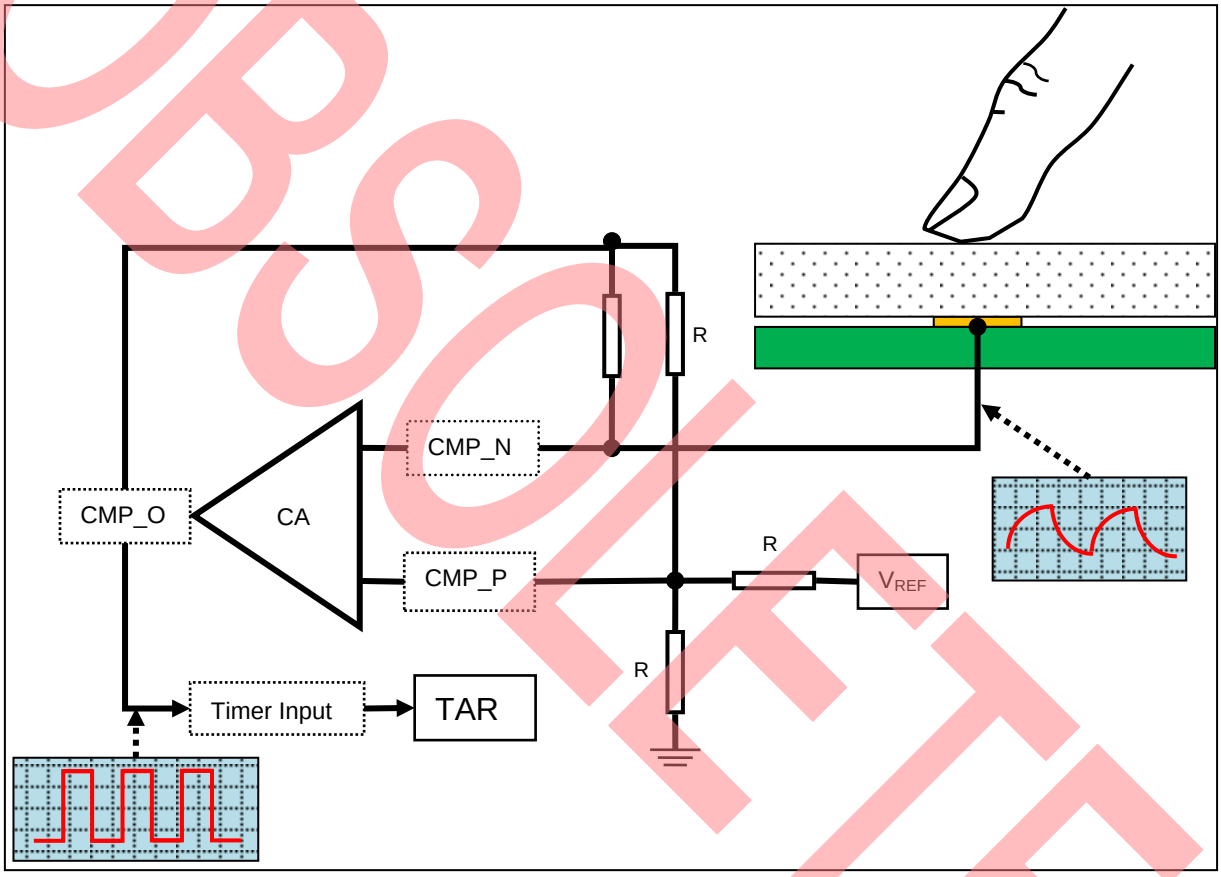
2.2 Relaxation Oscillation Solution

The theory of capacitance touch sensor is to check capacitance increment. As follows, when finger not touch the C = Cp, when touching the C = Cp +CF, and the increment is ΔC = CF. This ΔC will affect the electrode charging-up time.

The Relaxation Oscillation method detects the touch action by compare the frequency difference which is dependent upon the time.

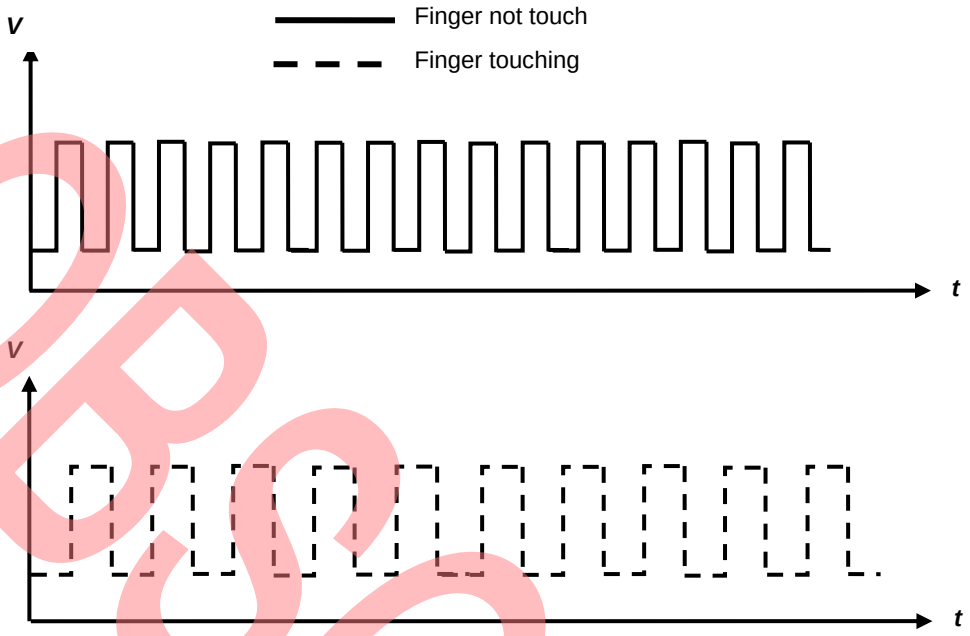
A simple relaxation oscillator can be created using the MCU's on-chip comparator and the capacitive sensor as the tuning element. Any change in capacitance of the sensor corresponds to a change in frequency which can be measured using the internal PWC (Pulse Width Capture) hardware of the MCU.

Figure 4. Relaxation Oscillation method physical diagram



OBSOLETE

Figure 5. Relaxation Oscillation method curve graph

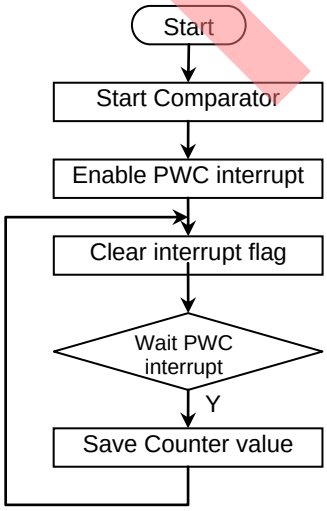


The Relaxation Oscillation method includes the following steps:

1. Configure register of Comparator
2. Enable PWC interrupt
3. Clear interrupt flag
4. Get the counter value in interrupt service routine
5. Save counter value
6. Start next sample

The flowchart shown as below

Figure 6. Relaxation Oscillation method Check Flowchart



OBSOLETE

3 Library

This chapter introduces how to use TSC library.

3.1 Library Overview

There are parameters which user need to setup, and 4 functions as API for user's situations. All the parameters and functions are introduced as follow.

Table 1. Parameters List

Name	Description	Remark
Key_Const type	TSC Key data structure type define	N/A
TSC_ADKEY	TSC AD data structure array	N/A
TSC_CMPKEY	TSC CMP data structure array	N/A
TSC_ADKeyNum	TSC AD Key Number	N/A
TSC_CMPKeyNum	TSC CMP Key Number	N/A
TSCKEY_ADPORT [1..3]	TSC AD Key IO port	N/A
TSCKEY_ CMPREFDATA	TSC CMP Key REF pin data	N/A
TSCKEY_ CMPREFDIR	TSC CMP Key REF pin direction	N/A
TSCKEY_ADPIN [0..7]	TSC Key IO pin	N/A
TSCKEY_ADSampleNumConst	TSC GPIO Key sample number	N/A
TSCKEY_CMPSampleNumConst	TSC AD Key sample number	N/A
TSC_ADThreshold	TSC GPIO Key Threshold data array	N/A
TSC_CMPThreshold	TSC AD Key Threshold data array	N/A
TSC_TimerData	TSC Key Timer Dare register	N/A
TSC_TimerEn	TSC Key Timer Start/Stop control bit	N/A

Table 2. Functions List

Prototype	Function Description	Remark
TSCKey_ADInit	This function initializes the timer used, the data structures of the library and calculates the initial capacitance of the electrodes.	N/A
TSCKey_ADCheck	This function is used to Check if the charge process are completed and save the value.	N/A
TSCKey_GetADValue	This function is used to detect the key touched status, and generate a specific key word.	N/A
TSCKey_CMPInit	This function initializes the timer used, the data structures of the library and calculates the initial capacitance of the electrodes.	N/A
T00PWC_ISR	This function as the interrupt service routine to capture the oscillation wave and count the value	N/A
TSCKey_GetGPIOValue	This function is used to detect the key touched status, and generate a specific key word.	N/A
TSCKey_UpdateThr	This function is used to update the threshold set by the GUI and refresh the detect difference	N/A

OBSOLETE

3.2 Parameters Setup

3.2.1 Structure type define

Every TSC Key have 4 attributes include: initial status, count value, threshold value and difference threshold value. User can define new key directly using the following structure.

```
typedef struct
{
      unsigned char InitFlag;          //**< Contains the initial status
      unsigned int  Value;             //**< Contains the count value
      unsigned int  Threshold;         //**< Contains the threshold value now
      unsigned int  DifThreshold;      //**< Contains the threshold temp value
} Key_Const;
```

All the keys which have been defined can make up a array which length will defined by the parameter *TSC_GPIOKeyNum and TSC_ADKeyNum*

3.2.2 Port and Pin define

Parameter *TSCKEY_ADPORT[1..3], TSCKEY_CMPPORT* are used to configure the TSC pin. The *TSCKEY_ADPORT, TSCKEY_CMPPORT,* directly use the register value on the data sheet, and the *TSCKEY_ADPIN[0..7]* configure the pin (use the continuous pin in same port are recommend, because that will be good at the sample speed).

3.2.3 Sample parameters setup

The parametes:

```
#define TSCKEY_ADSampleNumConst   10          //Sample number
#define TSCKEY_CMPSampleNumConst  20          //Sample number

#define TSCKEY_ADDifThrOffset     130         //Threshold Difference Offset
#define TSCKEY_CMPDifThrOffset    120         //Threshold Difference Offset

#define TSC_TimerData             T00DR       //Timer Data
#define TSC_TimerEn               T00CR1_STA  //Timer Enable

extern unsigned int TSC_ADThreshold[TSC_KeyNum];
extern unsigned int TSC_CMPThreshold[TSC_KeyNum];
```

used to configure the sample number, threshold, and timer initial value.

OBSOLETE

3.2.4 Example of Parameters Setup

```
#define TSCKEY_ADSampleNumConst   10            //Sample number
#define TSCKEY_CMPSampleNumConst  20            //Sample number

#define TSCKEY_ADDifThrOffset     130           //Threshold Difference Offset
#define TSCKEY_CMPDifThrOffset    120           //Threshold Difference Offset

#define TSC_ADKeyNum              8             //TSCKey number
#define TSC_CMPKeyNum             1             //TSCKey number

#define TSCKEY_CMPREFDATA         PDR7_P76      //TSCKey pin data          >>P76
#define TSCKEY_CMPREFDIR          DDR7_P76      //TSCKey pin dir           >>P76

#define TSCKEY_ADPORT1            0x0000        //TSCKey port register     >>P0
#define TSCKEY_ADPORT2            0x0018        //TSCKey port register     >>P7
#define TSCKEY_ADPORT3            0x0016        //TSCKey port register     >>P6

#define TSCKEY_ADPIN1             0x01          //TSCKey pin dir & data    >>P00
#define TSCKEY_ADPIN2             0x02          //TSCKey pin dir & data    >>P01
#define TSCKEY_ADPIN3             0x04          //TSCKey pin dir & data    >>P02
#define TSCKEY_ADPIN4             0x20          //TSCKey pin dir & data    >>P04
#define TSCKEY_ADPIN5             0x40          //TSCKey pin dir & data    >>P05
#define TSCKEY_ADPIN6             0x02          //TSCKey pin dir & data    >>P71
#define TSCKEY_ADPIN7             0x04          //TSCKey pin dir & data    >>P72
#define TSCKEY_ADPIN8             0x08          //TSCKey pin dir & data    >>P63

#define TSC_TimerData             T00DR         //Timer Data
#define TSC_TimerEn               T00CR1_STA    //Timer Enable

typedef struct
{
      unsigned char InitFlag;           //<Contains the initial status
      unsigned int  Value;              //<Contains the count value
      unsigned int  Threshold;          //<Contains the threshold value
      unsigned int  DifThreshold;       //<Contains the difference threshold
} Key_Const;

extern unsigned int TSC_GPIOThreshold[TSC_KeyNum];
extern unsigned int TSC_ADThreshold[TSC_KeyNum];

extern Key_Const TSC_GPIOKEY[TSC_GPIOKeyNum];
extern Key_Const TSC_ADKEY[TSC_ADKeyNum];
```

OBSOLETE

3.3 Application Interface

All the functions supplied by the TSC.lib will be introduced below, include the function prototype, input parameter(s), return value(s), and the function description.

3.3.1 Initialization Function

Table 3. Initialization Function Table (AD Method)

Prototype	void TSCKey_ADInit (Key_Const *Key);		
Parameter:	Key_Const *	Key	Pointer to the TSC Key array
Return	void		
Description	This function initializes the timer used, the data structures of the library and calculates the initial capacitance of the electrodes.		
Remark	N/A		

Table 4. Initialization Function Table (RO Method)

Prototype	void TSCKey_CMPInit (Key_Const *Key);		
Parameter:	Key_Const *	Key	Pointer to the TSC Key array
Return	void		
Description	This function initializes the timer used, the data structures of the library and calculates the initial capacitance of the electrodes.		
Remark	N/A		

3.3.2 Check Sensor Function

Table 5. Check Sensor Function Table (AD Method)

Prototype	unsigned int TSCKey_ADCheck(unsigned char TSC_Port,unsigned char TSC_Pin,unsigned char TSCKEY_SampleNum)		
Parameter:	unsigned char*	TSC_Port	Port to be checked
	unsigned char	TSC_Pin	Pin to be checked
	unsigned char	TSCKEY_SampleNum	Sample number
Return	unsigned int	Check_count	The counter value
Description	This function is used to Check if the charge process are completed and save the value.		
Remark	N/A		

Table 6. Check Sensor Function Table

Prototype	__interrupt void T00PWC_ISR (void)
Parameter:	void
Return	void
Description	This function as the interrupt service routine to capture the oscillation wave and count the value
Remark	N/A

OBSOLETE

3.3.3 Get Value Function

Table 7. Get Value Function Table (AD Method)

Prototype	unsigned char TSCKey_GetADValue(Key_Const *Key)		
Parameter:	Key_Const *	Key	Pointer to the TSC Key array
Return	unsigned char	TSCKeyValue	The Key word for the service function Single touch mode: Only the latest detected key will be set with '0' Multi touch mode: All touched keys be set with '0'
Description	This function is used to detect the key touched status, and generate a specific key word.		
Remark	N/A		

Table 8. Get Value Function Table (RO Method)

Prototype	unsigned char TSCKey_GetGPIOValue(Key_Const *Key)		
Parameter:	Key_Const *	Key	Pointer to the TSC Key array
Return	unsigned char	TSCKeyValue	The Key word for the service function Single touch mode: Only the latest detected key will be set with '0' Multi touch mode: All touched keys be set with '0'
Description	This function is used to detect the key touched status, and generate a specific key word.		
Remark	N/A		

3.3.4 Update Threshold Function

Table 9. Update Threshold Function Table

Prototype	void TSCKey_UpdateThr(Key_Const *Key,unsigned char Key_Num,unsigned int *Key_Thr)		
Parameter:	Key_Const *	Key	Pointer to the TSC Key array
	unsigned char	Key_Num	Number of TSC Keys
	unsigned int *	TSCKEY_SampleNum	Pointer to the TSC Threshold array
Return	void		
Description	This function is used to update the threshold set by the GUI and refresh the detect difference		
Remark	N/A		

OBSOLETE

3.4 Cypress TSC Performance

Table 10. Performance

Resource	Amount	Description
ROM	1272 Byte	ROM space: If just need support 1 sensor, ROM of lib can be reduced to: 300 Byte If just need support 2 sensors, ROM of lib can be reduced to: 960 Byte If just need support 3 sensors, ROM of lib can be reduced to: 1120 Byte If need support 4 sensors, ROM of Lib is:1272 Byte
RAM	36 Byte	RAM space: If just need support 1 sensor, RAM of lib can be reduced to: 11 Byte If just need support 2 sensors, RAM of lib can be reduced to: 26 Byte If just need support 3 sensors, RAM of lib can be reduced to: 31 Byte If need support 4 sensors, RAM of Lib is:36 Byte
IO Port	1	P0
IO Pin	4	P04,P05,P06,P07
Timer	2	Timebase Timer
Serial Port	1	Serial Port(P04,P05),only use in check threshold mode
Machine Clock	>=8M	
Scan 1 Key	2 ms	

OBSOLETE

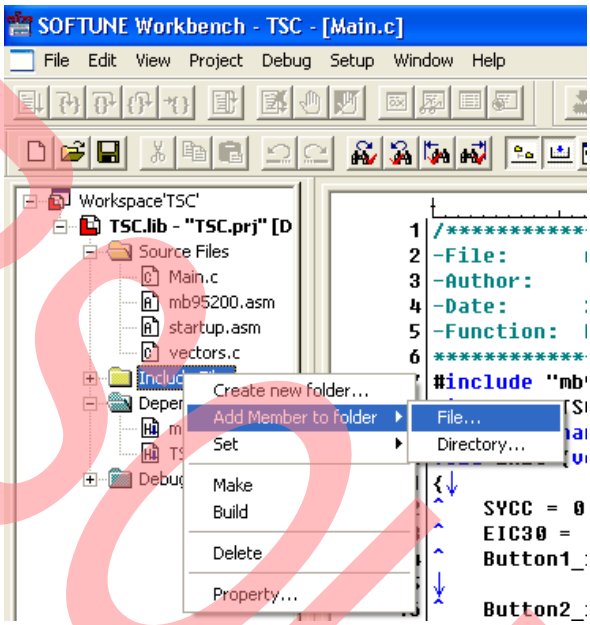
3.5 How to Add Cypress TSC.lib

3.5.1 Add Cypress TSC.lib to User's Project

Download the software from the website: www.cypress.com/mb95f430h.

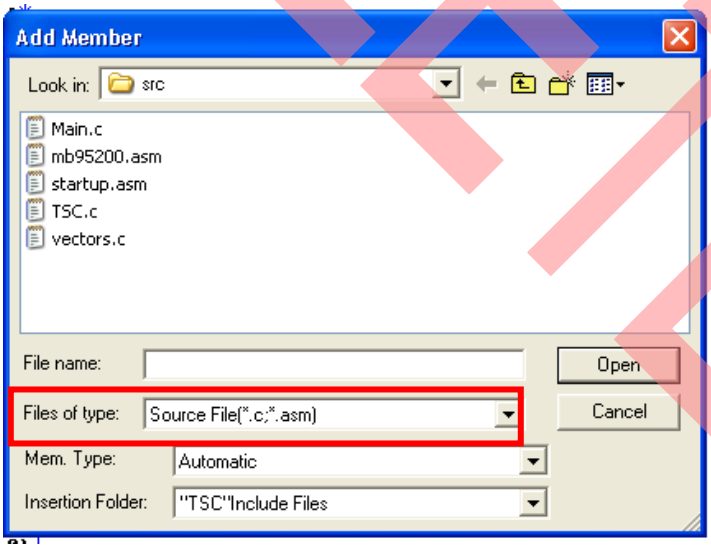
1. In Softune, Right click on folder *Include Files* → select *Add member to folder* from the menu →select *File*.

Figure 7. Add member to folder



2. Because the default option of file type filters is *.c and * .asm, you can't found *TSC.lib* in dialog box of *Add Member*.

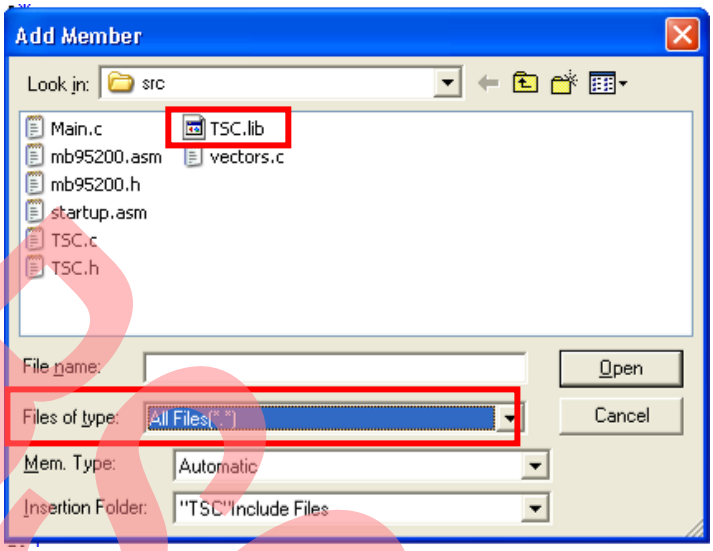
Figure 8. Popup Add Member dialog box



OBSOLETE

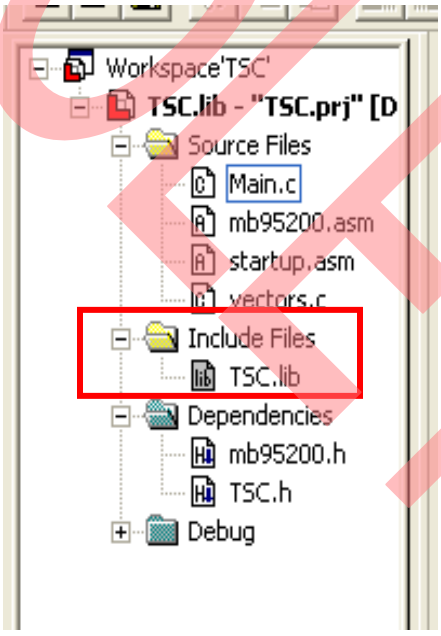
3. In *Add Member* dialog box, select 'ALL Files' from 'Files of Type', and then you will find the TSC.lib

Figure 9. Found the lib file



4. Double click TSC.lib, and then you can see it has been added in the folder *Include Files*

Figure 10. Add TSC.lib

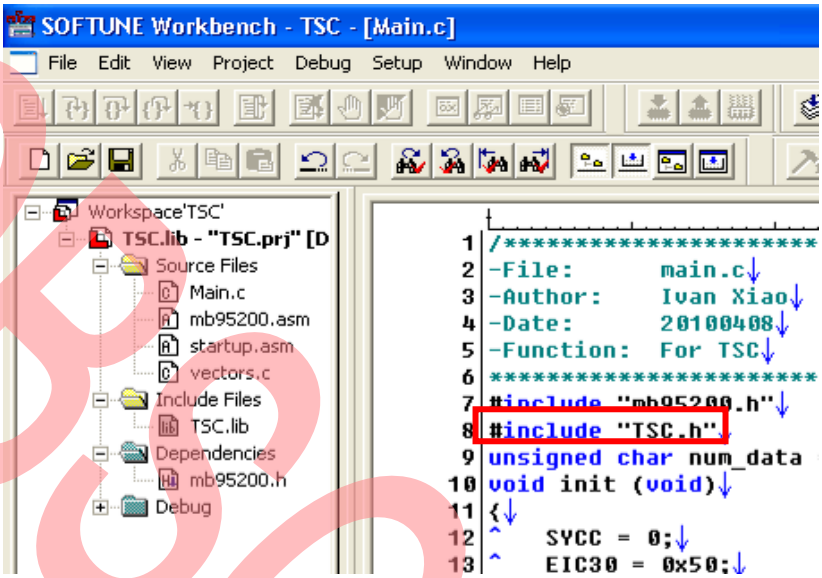


OBSOLETE

3.5.2 Include Header File

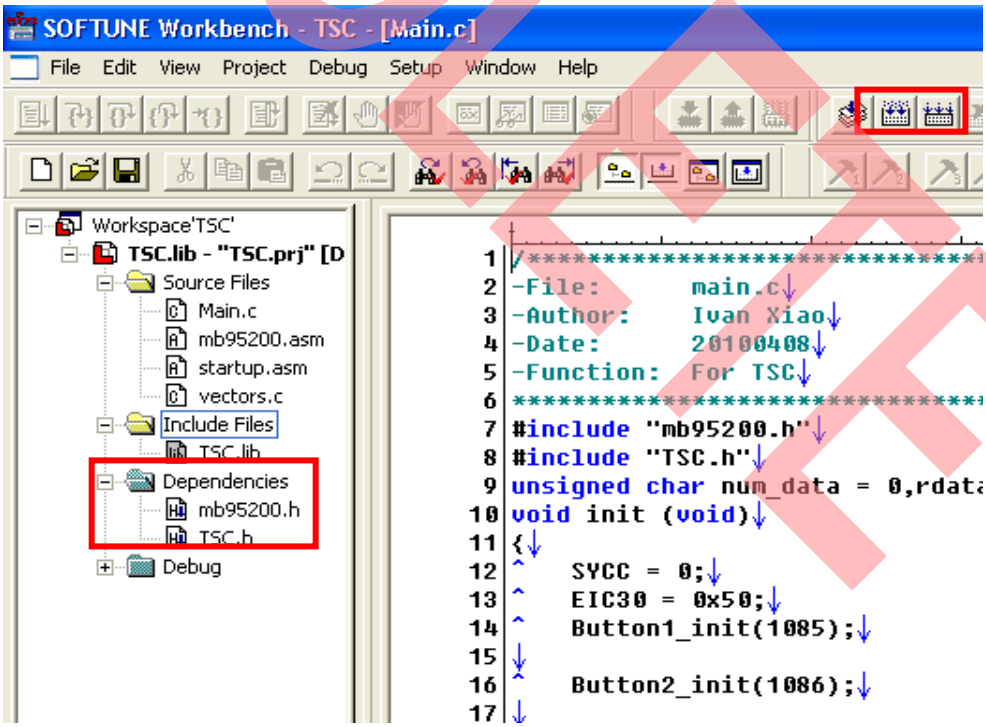
1. Add "#include"TSC.h "" in C file, such as in "main.c ".

Figure 11. Add include statement in C file



2. Compile the whole project, "TSC.h" will link TSC.lib to c file, so that user program can use API functions in TSC.lib.

Figure 12. Include header file successfully

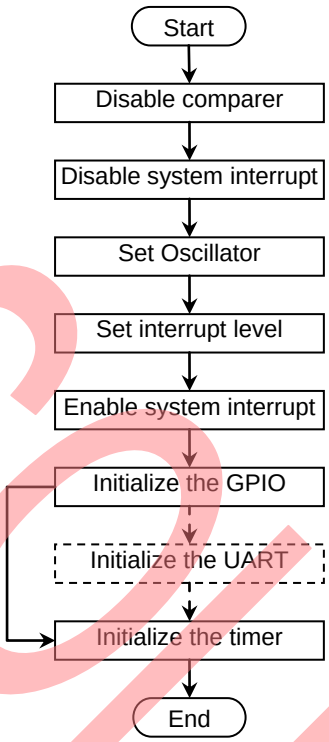


OBSOLETE

3.6 How to use Cypress TSC.lib

After complete above works, we can start use the TSC library now. The operation categorised by if connect to GUI. Both of two operations need initialize TSC, initialize Timer, initialize GPIO and initialize interrupt. If connect to GUI, the UART module should also be initialized.

Figure 13. Initialize Flow chart



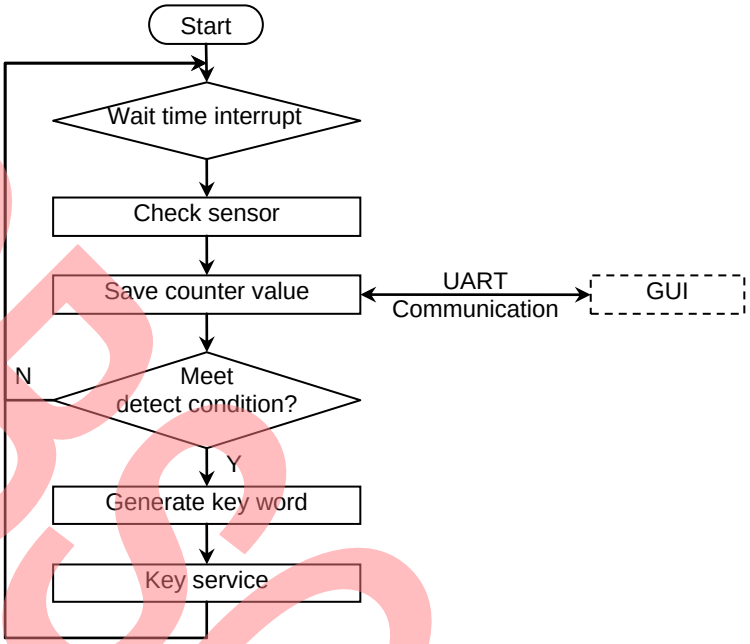
After all used modules have been initialized, program go into a endless loop, to do below step circularly:

1. Wait the time interval interrupt is generate, clear the interrupt flag
2. Check the electrodes and save the counter value
3. Judgement if the new value bigger or lower than the threshold, if meet the condition, generate the key word
4. According to the key word jump to corresponding operation (such as drive LED)
5. Go to another new loop

If connect GUI, in above process, there also have the UART interrupt, MCU need send out counter value or get new threshold from GUI. The direct of transmission is determined by the user-defined protocol.

OBSOLETE

Figure 14. Main loop Flowchart



OBSOLETE

4 LIB Usage Notice

This chapter introduces LIB usage notice.

- Machine clock

The machine clock should be set to 8M or above. If the machine clock is less than 8M, the sensor response is slow.

- Interrupt

This solution needs to use Timebase Timer interrupt to check sensor, and the interval time is 512us. It is unnecessary to use UART if the threshold is not gotten or tested with TSC GUI, and the interrupt setting in “vector.c” as following:

```
#include " mb95390.h"
/*****************************************************************************
   InitIrqLevels()
   This function  pre-sets all interrupt control registers. It can be used
   to set all interrupt priorities in static applications. If this file
   contains assignments to dedicated resources, verify  that the
   appropriate controller is used.
*****************************************************************************/
void InitIrqLevels(void)
{
       #ifdef enableUART
       ILR2 = 0xFE;          //  IRQ8:  LIN-UART (transmission)
       #else
       ILR2 = 0xFF;          //  IRQ8:  LIN-UART (transmission)
       #endif
       ILR4 = 0x3F;          //  IRQ19: Timebase timer
}
/*****************************************************************************
   Prototypes
   Add your own prototypes here. Each vector definition needs is prototype.
   Either do it here or include a header file containing them.
*****************************************************************************/
__interrupt void DefaultIRQHandler (void);
#ifdef enableUART
__interrupt void UART_ISR (void);
#endif
__interrupt void TBT_ISR (void);
```

OBSOLETE

```
/*****************************************************************************
   Vector definition

   Use following statements to define vectors.
   All resource related vectors are predefined.
   Remaining software interrupts can be added hereas well.
*****************************************************************************/
#ifdef enableUART
#pragma intvect UART_T                  8   //  IRQ8:  LIN-UART (transmission)
#else
#pragma intvect DefaultIRQHandler       8   //  IRQ8:  LIN-UART (transmission)
#endif
#pragma intvect TBT_ISR                 19  //  IRQ19: Time base timer
```

OBSOLETE

5 Additional Information

For more Information on Cypress products, visit the following website:

www.cypress.com/spansionproducts

A Appendix

A.1 Sample code

A.1.1 Main Function

File Name: main.c

Function: Initialize and configure

```
/*****************************************************************************************
-File:      main.c
-Author:    Lee Song
-Date:      20110510
-Function:Initialize and configure
*****************************************************************************************/
#include "Lcd_StateGrid.h"
#include "main.h"
#include "PortDef.h"
#include "HardKey.h"
#include "TSC.h"
extern UCHAR disUpdateFlage;
extern UCHAR HideDataBuff[8];
unsigned char TSCUpdateFlag;
extern void TBT_Init(void);
volatile unsigned int Div_Counter = 0;
//=====================================================================
// Initial GPIO peripheral
//=====================================================================
void GPIO_Init(void)
{
        LCDCE1 = 0;                     //Set for use GPIO
        LCDCE1_PICTL = 1;       //Set for use GPIO
        LED1_Off;
        LED2_Off;
        LEDCtrl_DirOut;
        HKEYInput_DirIn;
        BeepCtrl_Off;
        BeepCtrl_DirOut; //BEEP_Dir
}
```

OBSOLETE

```
//=====================================================================
// Beeper control function
//=====================================================================
void Beep(unsigned char num,unsigned char delay)
{
        unsigned int i,j,k;
        for(k=0;k<num;k++)
        {
                BeepCtrl_On;
                for(i=0;i<100;i++)
                {
                        for(j=0;j<delay;j++);
                }
                BeepCtrl_Off;
                for(i=0;i<100;i++)
                {
                        for(j=0;j<delay;j++);
                }
        }
}

//=====================================================================
// FunctionName:  main
// Description:   main function
// Input: None
// Return: None
//=====================================================================
void main(void)
{
        UCHAR dataBuff[8] = {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08};
        unsigned char i = 0;
        unsigned char j = 1;
        CMR0_VCID = 1;                                  //Disable CMP input
        __DI();
        Osc_Setup();                                    //Setup MCU main oscillator/FLL
        InitIrqLevels();        //Initialise Interrupt level register and IRQ vector table
        __EI();                 //Enable system interrupt now
        GPIO_Init();
```

OBSOLETE

```
        TBT_Init();
        IIC_Init();
        #ifdef enableUART
        UART_Init();
        #endif
        do
        {
                LED1_On;
                LED2_On;
                if(TSCUpdateFlag == 1)
                {
                        Div_Counter++;
                        TSCUpdateFlag = 0;
                }
        }
        while(Div_Counter < 30);
        Div_Counter = 0;
        LED1_Off;
        LED2_Off;
        Beep(2,500);
        TSCKey_Init(TSCKEY);
        while(1)
        {
                CallUpDateKey();
                TSCKey_Service();
                if(TSCUpdateFlag == 1)
                {
                        TSCUpdateFlag = 0;
                        if((Div_Counter % 9) == 0)
                        {
                                TSCKey_UpdateThr(TSCKEY,TSC_KeyNum,TSC_Threshold);
                        }
                        if(Div_Counter == 360)
                        {
                                Div_Counter = 0;
                        }
                        Div_Counter++;
                }
        }
}
```

OBSOLETE

```
//=====================================================================
// Setup MCU main oscillator/FLL
// Internal clock 243KHz. Set FLL for 11MHz Bus frequency
//=====================================================================
void Osc_Setup(void)
{
        SYSC &=0xbf;            //Set PF0/PF1 as main CR pin,set PG1/PG2 as GPIO
        SYCC  =0xf0;
        WATR  =0xf3;
        SYCC2 =0xf4;
        while(STBC_MRDY == 0);
        RESET_WATCHDOG();
}
//=====================================================================
// Reset the watch dog timer
//=====================================================================
void RESET_WATCHDOG(void)
{
        WDTC = 0x35;
}
```

OBSOLETE

5.1.1 Uart Function

File Name: Uart.c

Function: Initialize and configure UART module

```
/*******************************************************************************************
-File:      Uart.c
-Author:    Lee Song
-Date:      20110610
-Function:  Initialize and configure UART module
*******************************************************************************************/
#include "mb95390.h"
#include "TSC.h"
#include "Uart.h"

unsigned char UART_Start = 0;
unsigned char UART_Sent = 0;
unsigned char UART_Get = 0;
unsigned char Threshold_N = 0;
unsigned char UART_Setfinish = 0;
unsigned char *pTSC_Threshold = (unsigned char *)TSC_Threshold;
unsigned char UART_TSCValue[UART_DBUFF_LEN] = {0};
unsigned char UART_TSCStatus[UART_SBUFF_LEN] = {0};

unsigned char UART_DBUFF_PTR = 0;
unsigned char UART_SBUFF_PTR = 0;

void UART_Init(void)
{
        UART_Dir_IN;
        UART_Dir_OUT;
        SMC10 = 0x0C;   // UART/SIO Serial Mode Control Register 1 (SMC10)
        // |||||||+------Operating mode select bit (Clock asynchronous mode (UART))
        // |||||++-------      Clock select bit(Dedicated baud rate generator)
        // ||||+---------      Character bit length control bits(8 bits)
        // |||+----------      Stop bit length control bit(2-bit length)
        // ||+-----------      Parity control bit(No parity)
        // |+------------      Parity polarity bit(Even parity)
        // +------------      Serial data direction control bit(Transmit/receive data from MSB side sequentially)
```

OBSOLETE

```
    SMC20 = 0x5E;// UART/SIO Serial Mode Control Register 2 (SMC20)
    // |||||||+-------      Transmission data register empty interrupt enable bit(Disables
    //                      transmission data register empty interrupts)
    // ||||||+-------       Transmission completion interrupt enable bit(Enables transmission
    //                      completion interrupts)
    // |||||+--------       Reception interrupt enable bit(Enables reception interrupts)
    // ||||+---------       Transmission operation enable bit(Enables transmission operation)
    // |||+----------       Reception operation enable bit(Enables reception operation)
    // ||+-----------       Reception error flag clear bit(Clears the error flags in the SSR0 register)
    // |+------------       Serial data output enable bit(Enables serial data output)
    // +------------        Serial data output enable bit(Disables serial clock output (usable as
    //                      a general-purpose port))
    PSSR0 = 0x04;   // UART/SIO Dedicated Baud Rate Generator Prescaler Select Register (PSSR0)
    // ||||||++-------      Prescaler select bits(1/1)
    // |||||+----------      Baud rate clock output enable bit(Enables transmission completion interrupts)
    // +++++--------Undefined bit
    BRSR0 = 26;     // UART/SIO Dedicated Baud Rate Generator Baud Rate Setting
                    // Register (BRSR0)
                    // 2 ~ 255
    ////////////////////////////////////////////////////////////////////////////////////////////////////////////
    //////// Sample Asynchronous Transfer Rates by Baud Rate Generator ////////////////////////////
    ////////////////////////////////////////////////////////////////////////////////////////////////////////////
    //          Total division ratio                Baud rate          ////////////////////////////////
    //          (PSS ×  BRS ×  4)           (Machine Clock/Total division ratio)//////////////////////
    ////////////////////////////////////////////////////////////////////////////////////////////////////////////
    ////////////////////////////////////////////////////////////////////////////////////////////////////////////

    SSR0 = 0x00;    // UART/SIO Serial Status Register (SSR0)
    // |||||||+------       Transmit data register empty flag
    //                      (0:Transmit data present / 1:Transmit data absent)
    // ||||||+-------       Transmission completion flag
    // |+------------       Parity polarity bit(Even parity)
    // +------------        Serial data direction control bit(Transmit/receive data from MSB side sequentially)
}
```

OBSOLETE

```
void UART_Transmit(void)
{
    if(UART_Sent == 1)
    {
        if(UART_DBUFF_PTR < UART_DBUFF_LEN)          //Sent the data
        {
            TDR0 = UART_TSCValue[UART_DBUFF_PTR++];
        }
        else if(UART_DBUFF_PTR < (UART_DBUFF_LEN << 1))      //Sent the threshold
        {
            TDR0 = pTSC_Threshold[UART_DBUFF_PTR - UART_DBUFF_LEN];
            UART_DBUFF_PTR++;
        }
        else if(UART_DBUFF_PTR == (UART_DBUFF_LEN << 1))
        {
            UART_Sent = 0;
            UART_DBUFF_PTR = 0;
        }
    }
    else if(UART_Sent == 2)                          //Sent the board status
    {
        if(UART_SBUFF_PTR < UART_SBUFF_LEN)
        {
            TDR0 = UART_TSCStatus[UART_SBUFF_PTR++];
        }
    else
        {
            UART_Sent = 0;
            UART_SBUFF_PTR = 0;
        }
    }
    else
    {
        SSR0_TCPL = 0;
    }
}
```

OBSOLETE

```
__interrupt void UART_ISR (void)
{
        unsigned char UART_Temp;
        SMC20_RERC = 0;
        if(SSR0_RDRF == 1)                                          //Receive Data
        {
                SSR0_RDRF = 0;
                if(UART_Get == 0)
                {
                        UART_Temp = RDR0;
                        switch(UART_Temp)
                        {
                                case UART_Connect:
                                {
                                        TDR0 = UART_Check_OK;
                                        UART_Start = 1;
                                        break;
                                }
                                case UART_Getdata:
                                {
                                        UART_Sent = 1;
                                        SSR0_TCPL = 1;
                                        TDR0 = UART_TSCValue[UART_DBUFF_PTR++];
                                        break;
                                }
                                case UART_Getstatus:
                                {
                                        UART_Sent = 2;
                                        SSR0_TCPL = 1;
                                        TDR0 = UART_TSCStatus[UART_SBUFF_PTR++];
                                        break;
                                }
                                case UART_Setdata:
                                {
                                        TDR0 = UART_SetBack;
                                        Threshold_N = 0;
                                        UART_Get = 1;
                                        break;
                                }
```

OBSOLETE

```
                        case UART_Disconnect:
                        {
                                UART_Start = 0;
                                break;
                        }
                        default:
                        {
                                TDR0 = UART_Check_Erro;
                                break;
                        }
                }
        }
        else                                                    //Set threshold by GUI
        {
                if(Threshold_N < UART_DBUFF_LEN)
                {
                        pTSC_Threshold[Threshold_N] = RDR0;
                        TDR0 = UART_SetBack;
                        Threshold_N++;
                }
                else if(Threshold_N == UART_DBUFF_LEN)
                {
                        UART_Get = 0;
                        Threshold_N = 0;
                        UART_Setfinish = 1;
                        TSCKey_UpdateThr(TSCKEY,TSC_KeyNum);//,TSC_Threshold);
                }
        }
    }
    else if(SSR0_TDRE == 1)                                     //Sent Data
    {
        SSR0_TDRE = 0;
        if(UART_Get == 0)
                UART_Transmit();
    }
}
```

OBSOLETE

File Name: Uart.h

Function: Header file of UART module

```
/*****************************************************************************************
-File:      Uart.h
-Function:  Header file of UART module
*****************************************************************************************/
#include "TSC.h"
#define UART_DBUFF_LEN                  (TSC_KeyNum<<1)
#define UART_SBUFF_LEN                  10
#define UART_Dir_IN                     DDR1_P10 = 0
#define UART_Dir_OUT                DDR1_P11 = 1
#define UART_Connect                    0x0F
#define UART_Getdata                0x5F
#define UART_Getstatus                  0x6F
#define UART_Setdata                0xAF
#define UART_SetBack                0xCF
#define UART_Disconnect                 0xBF
#define UART_Check_OK                   0xFF
#define UART_Check_Erro                 0xEF
extern unsigned char UART_TSCValue[UART_DBUFF_LEN];
extern unsigned char UART_TSCStatus[UART_SBUFF_LEN];
void UART_Init(void);
void UART_Transmit(void);
```

OBSOLETE

5.1.2 Key_Service Function

File Name: Key_Service.c

Function: Key press service function

```
/*****************************************************************************************
-File:      Key_Service.c
-Author:    Lee Song
-Date:      20110610
-Function:  Key press service function
*****************************************************************************************/
#include "mb95390.h"
#include "PortDef.h"
#include "TSC.h"

unsigned char TSCKeyCode = 0xFF;
unsigned char mTSCKeyCode = 0xFF;
unsigned int Key_Task_Time=0;

extern unsigned char TSCKeyCode;
extern volatile      unsigned int Now_Time;

void CallUpDateKey(void)
{
        unsigned char TmpCode = 0xff;
        unsigned char TSCTmpCode = 0xff;
        if(Now_Time-Key_Task_Time >= 1)
        {
                Key_Task_Time = Now_Time;
                TSCTmpCode = TSCKey_GetValue(TSCKEY);
                if (TSCTmpCode == TSCOldCode)
                        TSCKeyCode = TSCTmpCode;
                else
                        TSCOldCode = TSCTmpCode;
        }
}
```

OBSOLETE

```
void TSCKey_Service(void)
{
        unsigned char i;
        if(mTSCKeyCode != TSCKeyCode)
        {
                if(TSCKeyCode != 0xFF)
                {
                        for(i=0;i<4;i++)
                        {
                                if((TSCKeyCode | (1 << i)) == 0xFF)break;
                        }
                        Beep(1,500);
                }
                mTSCKeyCode = TSCKeyCode;
        }
}
```

OBSOLETE

Document History

Document Title: AN205226 - F²MC-8FX Family MB95F430H Series Capacitance Touch Sensor Use TSC2_Demo_434.Lib

Document Number: 002-05226

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	—	HUAL	06/03/2011	Initial release.
			08/01/2011	1. Update template to the latest version. 2. Modify API name and function description based on new TSC.lib. 3. Add new sample code, include UART and Key service functions.
*A	5266025	HUAL	06/28//2016	Migrated Spansion Application note from MCU-AN-500124-E-11 to Cypress format. Lib file doesn't exist and this AN to be Obsolete.

OBSOLETE

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

ARM® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Lighting & Power Control	cypress.com/powerpsoc
Memory	cypress.com/memory
PSoC	cypress.com/psoc
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless/RF	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP

Cypress Developer Community

Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

PSoC is a registered trademark and PSoC Creator is a trademark of Cypress Semiconductor Corporation. All other trademarks or registered trademarks referenced herein are the property of their respective owners.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

Phone : 408-943-2600
Fax : 408-943-4730
Website : www.cypress.com

© Cypress Semiconductor Corporation, 2011-2016. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.

OBSOLETE