

Integrating Touch Sensing Software (TSS 3.0.1) on Kinetis L Using GPIO Method

by: Xianhu Gao

Contents

1 Introduction

The Touch Sensing Software (TSS) solution transforms a standard Freescale microcontroller into a proximity capacitive touch sensor controller. In TSS 3.0 and higher versions, the touch sense library enables capacitive sensing not only for the entire Freescale S08, ColdFire V1 family and ARM®Cortex™-M4 Kinetis, but also ARM Cortex-M0+ Kinetis L family of microcontrollers.

In the TSS 3.0.1 Library, a demo called FRDMKL25Z_DEMO is offered using TSI method based on FRDM-KL25Z board. The FRDM-KL25Z is an ultra-low-cost development platform enabled by Kinetis L Series KL1 and KL2 MCU families built on ARM Cortex-M0+ processor.

In this application note, the demo FRDMKL25Z_DEMO is mostly reused and finally shows touch sensing with GPIO method, not TSI method. This is a good example of integrating TSS for those MCUs that have no TSI module such as S08 or KL02.

1.1 Kinetis L series freedom development platform

The FRDM-KL25Z is an ultra-low-cost development platform enabled by Kinetis L Series KL1 and KL2 MCU families built on ARM Cortex-M0+ processor. The features of FRDM-

1	Introduction.....	1
1.1	Kinetis L series freedom development platform.....	1
1.2	Touch-Sensing Software 3.0.1	2
1.3	Analog slider.....	2
2	GPIO-based capacitive touch sensing method.....	3
2.1	Touch sensing algorithm.....	3
2.2	Analog slider control API.....	4
2.3	Application interrupt management.....	5
3	Settings for GPIO method.....	5
3.1	Electrodes definition.....	5
3.2	Hardware timer.....	6
3.3	Peripheral initialization.....	6
3.4	Pullup resistor.....	7
3.5	Low-power function.....	7
4	FreeMASTER.....	9
5	Conclusion.....	12
6	Reference.....	12

Introduction

KL25Z include easy access to MCU I/O, battery-ready, low-power operation, a standard-based form factor with expansion board options and a built-in debug interface for flash programming and run-control. The FRDM-KL25Z is supported by a range of Freescale and third-party development software.

More information about FRDM-KL25Z can be found at: freescale.com/FRDM-KL25Z.

1.2 Touch-Sensing Software 3.0.1

The touch sensing software (TSS) solution transforms a standard Freescale microcontroller into a proximity capacitive touch sensor controller. Compared with the earlier version, TSS 3.0 and higher versions added a new text with the sections “Signal normalization”, “Automatic sensitivity calibration”, “Baseline initialization”, “Electrodes groups”, “Analog slider and analog rotary decoder API”, “Reading the instant delta in the control”, “Proximity function”, “Automatic Sensitivity Calibration”, “Shielding function and Water tolerance”, “Water tolerance mode”, “Analog rotary”, “Analog Slider”, and “Matrix”. (See the Revision History of TSSAPIRM : TSSAPIRM, Touch Sensing Software API Reference Manual, available on freescale.com.)

The latest TSS can be found at: freescale.com/TSS.

1.3 Analog slider

In this application note, the analog slider is used to control the brightness of the LEDs. An analog slider control works similar as the standard slider, but with less electrodes and the calculated position has a higher resolution. For example, a two-electrode analog slider can provide an analog position in the range 128. The shape of the electrodes need to meet the condition that increases and decreases the signal during the finger movement which needs to be linear. [Figure 1](#) shows the arrangement of electrodes used for a typical analog slider.



Figure 1. Electrode analog slider

2 GPIO-based capacitive touch sensing method

The GPIO low-level sensing method measures the capacitance using the GPIO capacitive touch sensing method as discussed in [Touch sensing algorithm](#). To measure the capacitance of an electrode, a Freescale MCU with GPIO and time measurement capabilities is required.

2.1 Touch sensing algorithm

The electrode connected to the MCU acts like a capacitor, and the external pullup resistor limits the current to charge the electrode. [Figure 2](#) shows the diagram of the circuit.

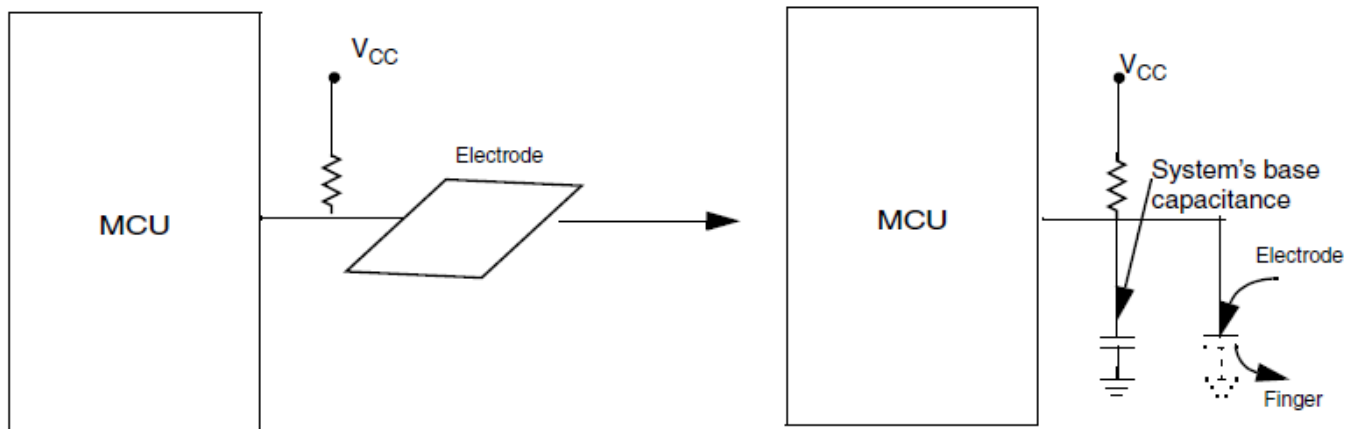


Figure 2. Electrode equivalent circuit

The RC circuit charging time constant (τ) is defined by the following equation:

$$\tau = RC$$

According to the above equation, if the pullup resistor remains the same, an increase in the capacitance will increase the circuit charging time. The MCU measures the charging time and uses this value to determine if the electrode has been touched or not. As the electrode is touched, the finger capacitance is added to the capacitance of the electrode. This increases the circuit capacitance, which increases the charge time measured by the timer. The following figure shows the charging time of the capacitor with finger added capacitance.

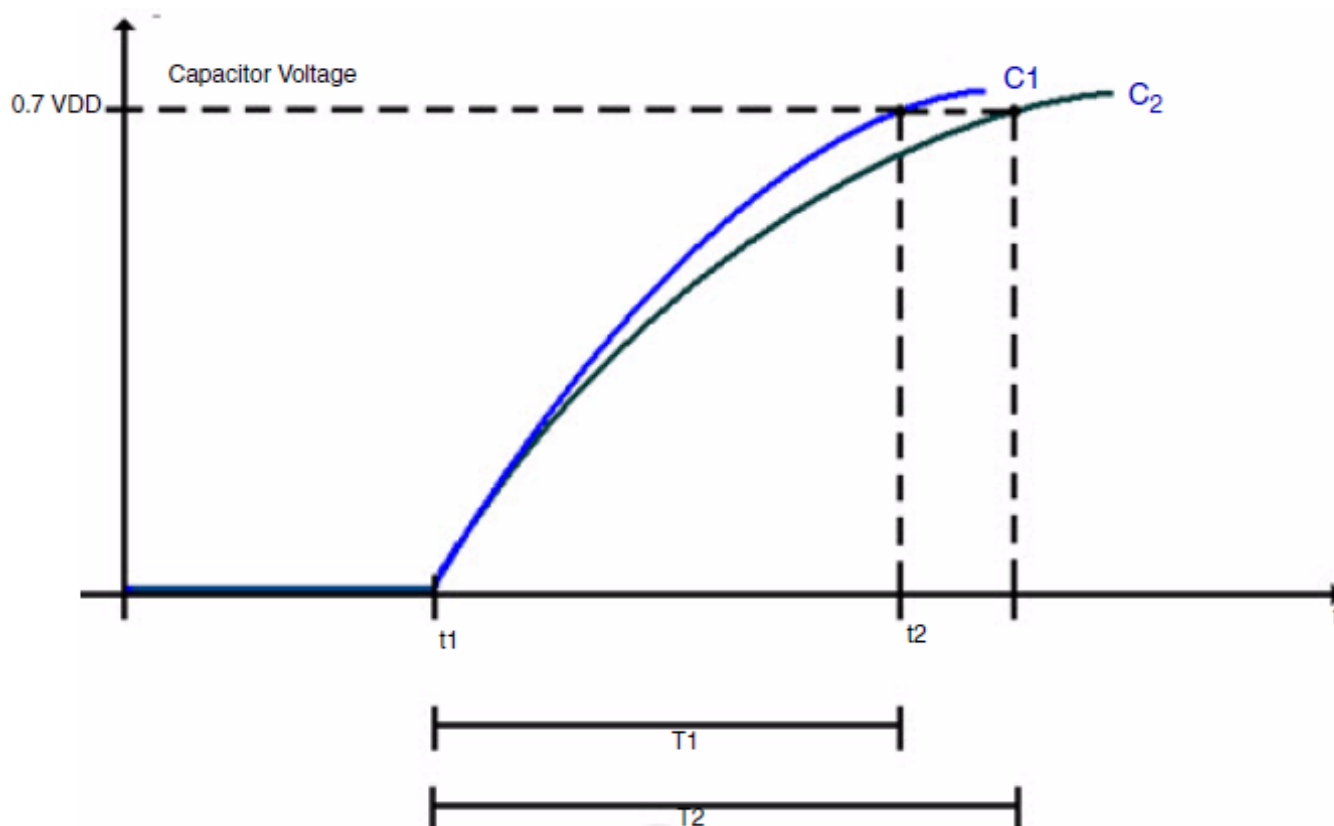


Figure 3. Charge time of capacitor with finger added capacitance

As shown in [Figure 3](#) :

- C1 — The charging time curve when there is no extra capacitance or when the electrode has not been touched.
- C2 — The charging time curve when the electrode has been touched.
- T1 — The charging time when the electrode has not been touched.
- T2 — The charging time when the electrode has been touched.

2.2 Analog slider control API

In TSS library, there exists TSS_SensorGPIO.c and TSS_SensorGPIO.h files, which contain the functions to perform the sensing of the electrodes and set the status for each electrode.

For analog slider, 8 configuration and status registers are offered. See [Table 1](#).

Table 1. Control configuration and status registers

Register number	Size(bytes)	Register name	Initial value	Brief description
0x00	1	ControlId	Application dependable	R—Displays the control type and control number
0x01	1	ControlConfig	0x00	RW—This register configures overall enablers of the object.
0x02	1	DynamicStatus	0x00	R—Displays the movement, direction, and displacement information

Table continues on the next page...

Table 1. Control configuration and status registers (continued)

Register number	Size(bytes)	Register name	Initial value	Brief description
0x03	1	Position	0x00	R—Displays the touch and absolute position information. Hold the invalid position status flag.
0x04	1	Events	0x00	RW—Configures the events that call the callback function
0x05	1	AutoRepeatRate	0x00	RW—Configures the rate at which keys will be reported when they are kept pressed and no movement is detected.
0x06	1	MovementTimeout	0x00	RW—Number of times the decoder must detect a no displacement before reporting a hold event and no movement
0x07	1	Range	0x40	RW—Defines the absolute range within which the position is reported.

But these registers are not supposed to be operated directly, or the data may get corrupt. The TSS_SetASliderConfig() and TSS_GetASliderConfig() are the two APIs prepared for users in TSS_API.h file.

All the registers mentioned in [Table 1](#) are read or written by the following two API functions.

```
UINT8 TSS_SetASliderConfig(TSS_CONTROL_ID u8ControlId,UINT8 u8Parameter,UINT8 u8Value)
UINT8 TSS_GetASliderConfig(TSS_CONTROL_ID u8ControlId,UINT8 u8Parameter);
```

2.3 Application interrupt management

When using GPIO method, all user interrupt handlers must register themselves with the TSS library by calling the TSS_SET_SAMPLE_INTERRUPTED() macro. The TSS leaves interrupts enabled while taking electrode measurements. The electrode measurement routine may get interrupted by a user application interrupt that causes the values sampled by the GPIO method to be invalid. So this is the user's responsibility to handle this micro.

3 Settings for GPIO method

FRDMKL25Z_DEMO is the demo special for KL25 Freedom board in TSS3.0 and above library, the default touch sensing method is TSI method. After downloading the code (see freescale.com/FRDM-KL25Z), the brightness of three LEDs on the board can be controlled by touching the analog slider. Besides, low power and wake up control are also shown in the demo.

However, in this application, GPIO method is the highlight, so some of the key points must be picked out and modified.

3.1 Electrodes definition

There are two types for electrodes: GPIO and TSIx_CHy. The default demo configuration is for TSI pin, so the Analog Slider pin is defined in TSS_SystemSetup.h file using the following code:

```
#define TSS_E0_TYPE          TSI0_CH9
#define TSS_E1_TYPE          TSI0_CH10
```

Settings for GPIO method

But in GPIO method, the following macro is defined:

```
#define TSS_E0_P          B
#define TSS_E0_B          16
#define TSS_E1_P          B
#define TSS_E1_B          17

#define TSS_E0_TYPE        GPIO
#define TSS_E1_TYPE        GPIO
```

TSS_E0_P and TSS_E0_B indicate the port and the bit in this port. For example, in the above configuration, E0 is PTB16 and E1 is PTB17. Although PTB16 and PTB17 are still the same pins with TSI0_CH9 and TSI0_CH10, these are now just normal GPIO and no longer in TSI control mode.

Besides, if user needs to set the GPIO strength and slew rate, the following macro can be defined:

```
#define TSS_USE_GPIO_STRENGTH    1
#define TSS_USE_GPIO_SLEW_RATE  1
```

These are the basic changes from TSI method to GPIO method.

3.2 Hardware timer

The algorithm of the touch sensing control is based on a counter; the counter number indicates whether a touch is detected or how large the capacitance is added on the electrode. For GPIO method, a hardware timer must be selected from the on-chip timer: TPMx, FTMx, MTIMx.

In this application, TPM1 is selected as the timer, so the following macro is defined in TSS_SystemSetup.h file:

```
#define TSS_HW_TIMER          TPM1
```

However, in the original demo, TPM1 is used as the delay timer, so the delay function DelayMS() must be modified. In this application, the delay function is just deleted.

The timer prescaler is required to be adjusted to the application changing the counter speed that is used to measure the capacitance. The timer frequency depends on the MCU bus frequency. The timer configuration uses a prescaler value to adjust the time frequency relative to the MCU bus frequency.

This adjustment is made using the following define code present in the TSS_SystemSetup.h file:

```
#define TSS_SENSOR_PRESCALER    4    // adjust in different applications
```

Sometimes, the application may use a non-standard electrode size with specific values of capacitance. The touch sense timer interrupt provides an error handling if an electrode is never charged and also provides the code to exit the electrode charge loop in the event of a timeout. This ensures that the capacitive sensing module does not block the application execution. The adjustment of the timer overflow timeout value is made using the following macro present in the TSS_SystemSetup.h file:

```
#define TSS_SENSOR_TIMEOUT      65535 // adjust in different applications
```

3.3 Peripheral initialization

The TSS library provides the OnInit Callback function. Use of this callback is mandatory when executing TSS_Init function call. This function is used to initialize the peripheral clock, setup pin multiplexers, and so on.

In this function, the definition is like this:

```
#define TSS_ONINIT_CALLBACK      TSS_fOnInit    // in TSS_SystemSetup.h file

/* in events.c file */
void TSS_fOnInit(void)
```

```

{
    SIM_MemMapPtr sim = SIM_BASE_PTR;

    /* Modules Clock enablement */
    sim->SCGC5 |= SIM_SCGC5_PORTB_MASK; /* Port B clock enablement */

    // HW Timer clock must be opened here, or enter hard fault
    sim->SCGC6 |= SIM_SCGC6_TPM1_MASK;

    /* Set Electrodes for GPIO function */
    PORTB_PCR16 = PORT_PCR_MUX(MUX_ALT1);
    PORTB_PCR17 = PORT_PCR_MUX(MUX_ALT1);
}

```

This is very important for TPM1 and PORTB, lack of which will cause hardware fault when running the code.

3.4 Pullup resistor

As per the GPIO method algorithm, a pullup resistor is necessary to influence a lot to the charging and discharging of electrodes. In general, larger the value of the resistor, higher will be the resolution and smaller will be the range of capacitance measuring. With the proper resistor value, the user can modify the capacitance range and the sensitivity to a range suitable for application. [Table 2](#) shows the maximum capacitance range and capacitance resolution at different pullup resistance values from 500 k Ω to 2 M Ω .

Table 2. Maximum capacitance range and resolution at different pull-up resistance values

Voltage level	Pullup resistor (Ω)	Capacitance range (max) (pF)	Capacitance resolution (min) (pF)
VDD > 2.3 V	500 k	169.44	0.6645
VDD > 2.3 V	680 k	124.59	0.4886
VDD > 2.3 V	810 k	104.59	0.4102
VDD > 2.3 V	1 M	84.72	0.3322
VDD > 2.3 V	1.5 M	56.48	0.2215
VDD > 2.3 V	2 M	42.36	0.1661
1.8 V < VDD < 2.3 V	500 k	107.53	0.4217
1.8 V < VDD < 2.3 V	680 k	79.07	0.3101
1.8 V < VDD < 2.3 V	810 k	66.38	0.2603
1.8 V < VDD < 2.3 V	1 M	53.77	0.2108
1.8 V < VDD < 2.3 V	1.5 M	35.84	0.1406
1.8 V < VDD < 2.3 V	2 M	26.88	0.1054

In this application, a 500 k Ω resistor is connected between P3V3 in the jumper J9 (pin 8) on board and the electrode. If a 10 k Ω resistor is pulled up, the charging and discharging time is too small, so the touch may not be even detected; if a 10 M Ω resistor is pulled up, the dynamic feature is not so good. So, the proper pullup resistor must be adjusted by the application.

3.5 Low-power function

In the FRDMKL25Z_DEMO demo, low-power function is added; if no touch is detected in about 8 seconds, the system will enter low-power mode LOW_POWER_MODE that is defined in main.h file. The CPU will wake up if a touch is detected, because the TSI out-of-range interrupt can wake the MCU in all kinds of low-power modes.

Settings for GPIO method

There is an issue in the current KL25 chips that the out-of-range interrupt function is closely connected with the end of scan interrupt. The user can not wake the CPU with the out-of-range interrupt independently, so the demo gives out a workaround. The LPTMR is used as the LLWU source too. Every time the LPTMR interrupt occurs, the TSI0_GENCS register will be polled and the End of Scan flag is cleared, or the scan of electrode will not be executed.

The TSS does not fully manage the MCU low-power mode. The TSS just prepares the TSS system and the selected lower power control source device for entering the low-power mode of the MCU. Then, the user initiates entering into low-power mode by himself.

The following steps show an example of typical use of low-power function.

1. The peripheral module which is responsible for low-power control and synchronization is defined by the TSS_USE_LOWPOWER_CONTROL_SOURCE function contained in TSS_SystemSetup.h file.

```
#define TSS_USE_LOWPOWER_CONTROL_SOURCE          TSI0
```

2. The user defines LowPowerScanPeriod, LowPowerElectrode and LowPowerElectrodeSensitivity registers by the TSS_SetSystemConfig.

```
TSS_SetSystemConfig(System_LowPowerScanPeriod_Register, 0x08);  
TSS_SetSystemConfig(System_LowPowerElectrode_Register, 29u);  
TSS_SetSystemConfig(System_LowPowerElectrodeSensitivity_Register, 0x1A);
```

3. If the user wants to enter the MCU low-power mode, the Low Power Enabler bit in the System Configuration register, has to be enabled by the TSS_SetSystemConfig function.

```
(void) TSS_SetSystemConfig(System_SystemConfig_Register, (TSS_SYSTEM_EN_MASK |  
TSS_DC_TRACKER_EN_MASK | TSS_LOWPOWER_EN_MASK));
```

4. The user may now force the MCU to enter low-power mode by instructions related to the used MCU platform.

```
LowPowerControl(); // in the main loop in main.c
```

5. If the selected lower power control source device detects a touch, the MCU wakes and the program continues to run. The Low-Power Enabler bit is automatically disabled.

However, in this application, GPIO method is selected and TSI module is not used at all, so the TSI associated configuration must be removed. Therefore, the following macro in the original demo code must be removed:

```
/* in isr.h */  
#undef VECTOR_042  
#define VECTOR_042 TSS_TSI0Isr  
  
/* in TSS_SystemSetup.h */  
#define TSS_USE_LOWPOWER_CONTROL_SOURCE          TSI0  
  
/* TSI Autocalibration Settings */  
#define TSS_TSI_RESOLUTION                        6  
#define TSS_TSI_EXTCHRG_LOW_LIMIT                1  
#define TSS_TSI_EXTCHRG_HIGH_LIMIT              7  
#define TSS_TSI_PS_LOW_LIMIT                    0  
#define TSS_TSI_PS_HIGH_LIMIT                   7  
  
/* Active Mode Clock Settings */  
#define TSS_TSI_AMCLKS                           0  
#define TSS_TSI_AMPSC                           0  
#define TSS_TSI_AMCLKDIV                       1  
  
/* Low Power TSI definition */  
#define TSS_TSI_LPCLKS                          0  
  
/* in TSS_fOnInit(void) in events.c file */  
sim->SCGC5 |= SIM_SCGC5_TSI_MASK;  
/* Set Electrodes for TSI function */  
PORTB_PCR16 = PORT_PCR_MUX(MUX_ALT0);  
PORTB_PCR17 = PORT_PCR_MUX(MUX_ALT0);
```


4 FreeMASTER

The FreeMASTER software is one of the off-chip drivers, which supports communication between the target microcontroller and PC. The TSS library can use the FreeMASTER (GUI) tool for visualization and configuration of internal library variables. This tool also enables to observe the signal behavior, tune sensitivities, and setup the TSS system control registers. The GUI for this application has been built and saved in TSS 3.0\examples\FRDMKL25Z_DEMO\gui. The following steps discuss the usage of the GUI.

1. First, open the GUI file in FRDMKL25Z_DEMO\gui and build the connection. Choose Project > Options and select the Comm tab. Set Port as the virtual serial port on the computer and set baud rate as 115,200. See [Figure 4](#) as an example.

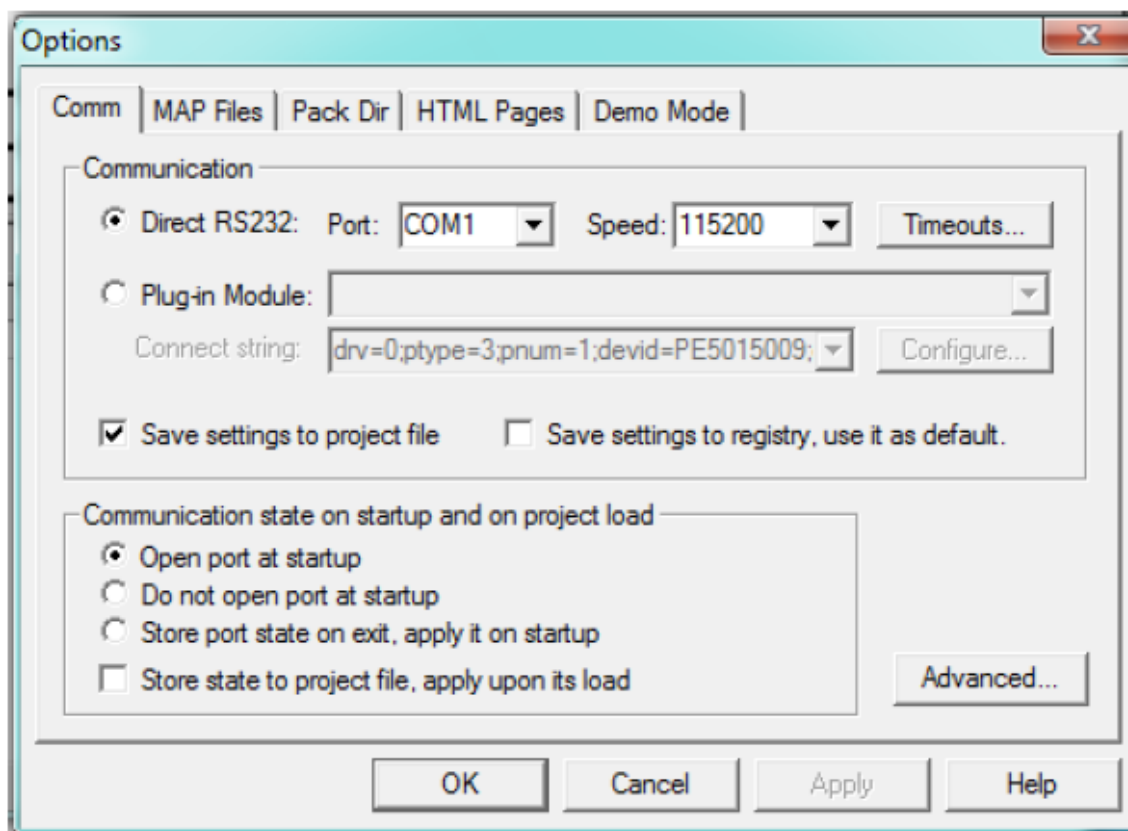


Figure 4. Build communication

2. Next, choose File > Start Communication. The GUI can read the data from MCU through the serial port, and the user can check the System configurations and the electrode signal changing curve. [Figure 5](#) shows the system configurations user sets in the code, [Figure 6](#) shows the electrodes signal data, [Figure 7](#) shows the control status, and [Figure 8](#) shows the electrodes signal data curve.

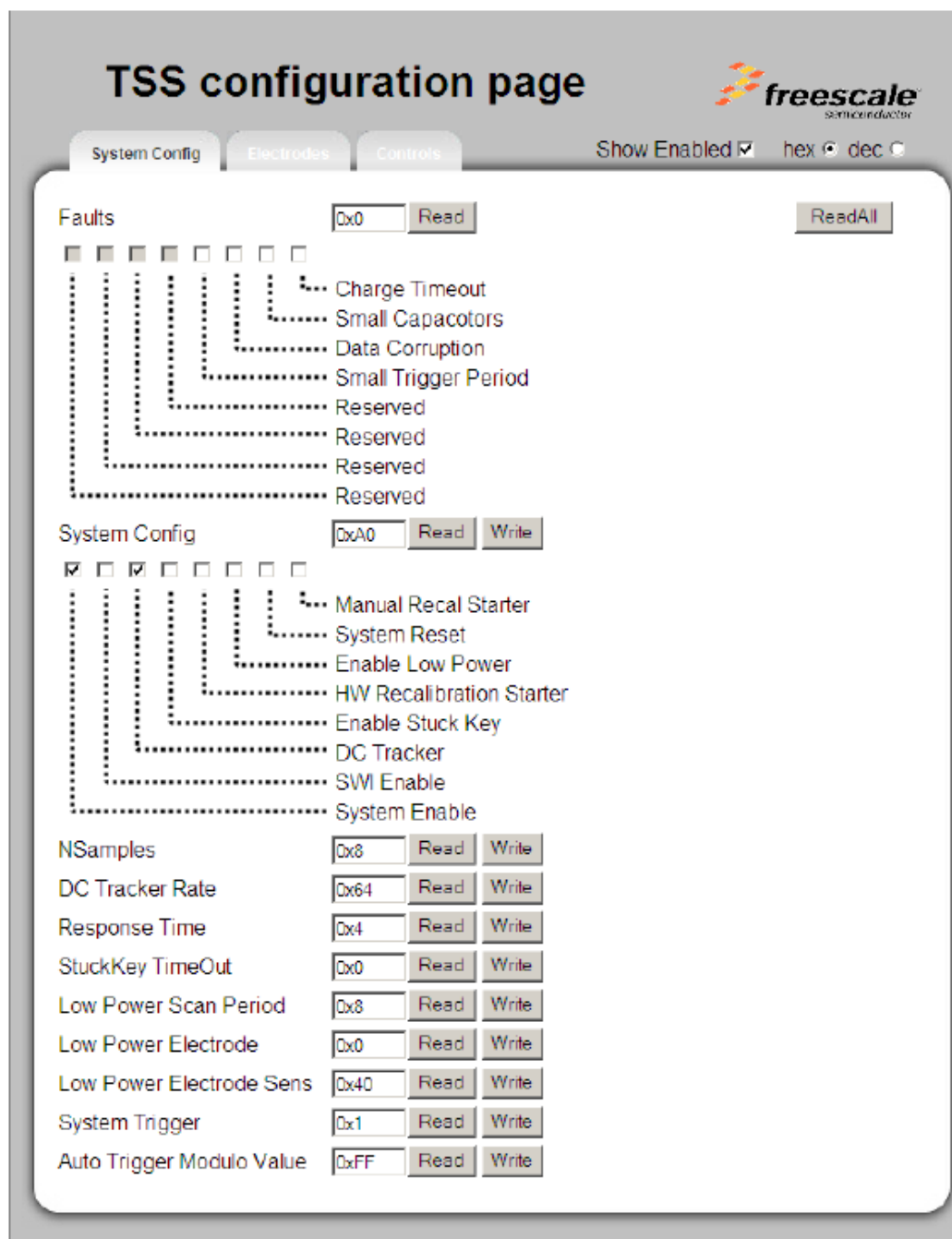



Figure 5. System configuration

TSS configuration page



System Config
Electrodes
Controls

Show Enabled ☒
 hex ☒ dec ☐




El. Index	Signal Ex	Baseline Ex	Delta Ex	Electrode state	Sensitivity	Enable electrodes
0	0xDB	0x53	0x7F		<input type="text" value="0xD"/>	<input checked="" type="checkbox"/>
1	0x7B	0x56	0x25		<input type="text" value="0x11"/>	<input checked="" type="checkbox"/>

Figure 6. Electrodes data

TSS configuration page



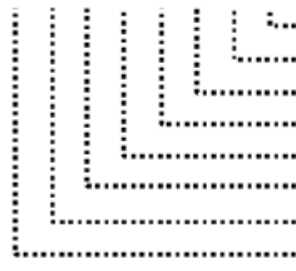
System Config
Electrodes
Controls

Show Enabled ☒
 hex ☒ dec ☐

Available controls:


Control Config

☒
☒
☐




0xC0
Read
Write

Dynamic Status

Direction: 

Displacement: 0

Position

Touch state: 

Actual Position: 8

ReadAll

Figure 7. Control status

Conclusion

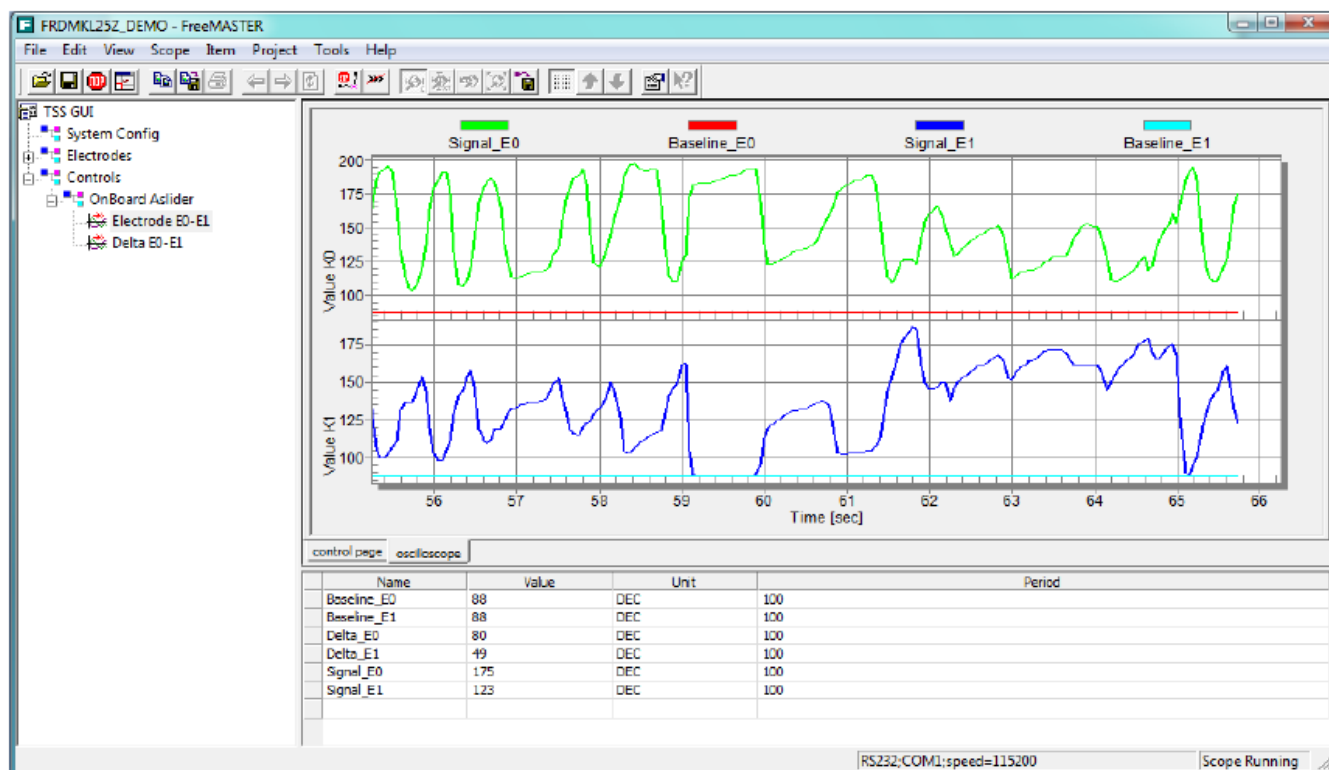


Figure 8. Electrodes data curve

5 Conclusion

This application is based on the KL25 demo; the difference is in the touch sensing method, the original demo is based on TSI module while this application is based on GPIO. This is a good example for the user to integrate the TSS library into the existing project or modify an existing TSS project using GPIO method. Using GPIO as the touch sensing electrode only needs 1 free I/O and a hardware timer. This is very useful to those chips with no TSI module such as KL02 and S08, for example and also leads to cost reduction. This application has been successfully executed on the KL25 and KL02 Freedom boards with the same results as the TSI method.

6 Reference

The following documents are available at freescale.com for further reference.

- TSSUG: TSSUG, Touch Sensing Software Users' Guide (Rev. 5)
- TSSAPIRM: TSSAPIRM, Touch Sensing Software API Reference Manual (Rev. 7)
- FRDM-KL25Z: Kinetis L Series Freedom Development Platform

How to Reach Us:

Home Page:

www.freescale.com

Web Support:

<http://www.freescale.com/support>

USA/Europe or Locations Not Listed:

Freescale Semiconductor
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:

Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:

Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:

Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductors products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals", must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claims alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

RoHS-compliant and/or Pb-free versions of Freescale products have the functionality and electrical characteristics as their non-RoHS-complaint and/or non-Pb-free counterparts. For further information, see <http://www.freescale.com> or contact your Freescale sales representative.

For information on Freescale's Environmental Products program, go to <http://www.freescale.com/epp>.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© 2012 Freescale Semiconductor, Inc.