Freescale Semiconductor
Application Note

Document Number: AN4213
Rev. 0, 09/2010

Migration to TSS 2.0

by: Eduardo Viramontes Guerra
Freescale MSG Industrial and Multimarket Division
Tlaquepaque, Mexico

1 Introduction

This document describes how to migrate from TSS 1.x versions to TSS 2.0. It also covers the principal differences between the two versions.

This document assumes that the users have a functional TSS 1.x project and TSS 2.0 is ready to use.

2 References

Use this book in conjunction with:

- Touch Sensing Software API Reference Manual (TSSAPIRM)
- Touch Sensing Software User Guide (TSSUG)

Contents

2.1 Acronyms and Abbreviations

Table 1. Acronyms

EGT	Electrode Graphing Tool
SSC	System Setup Creator
TSS	Touch Sensing Software

3 Migration procedure

Once you have the TSS 2.0 library files ready, follow the steps below:

1. Replace the TSS files.
2. Modify the TSS_SystemSetup.h file.
3. Add the OnFault callback function if you want to use it to detect the fault error.
4. Erase the TMP Over Flow vector in the prm file.

3.1 Replace TSS Library Files

1. Locate the TSS 2.0 files in the TSS 2.0 Lib folder.
2. Click Start/Programs/Freescale/Touch Sensing Software v2.0/Browse Library Files.
3. Copy the following files to your project:
   - TSS_S08.lib for HCS08 Freescale MCU family
   - TSS_CFV1.a for ColdFire® V1 Freescale MCU family
   - ATL_Sensor.c
   - ATL_Sensor.h
   - ATL_Timer.h
   - CTS_LowTypes.h
   - CTS_McuTypes.h
   - CTS_Sensor.c
   - CTS.Sensor.h
   - TSS_API.h
   - TSS_DataTypes.h
   - TSS_GPIO.h
   - TSS_StatusCodes.h
   - TSS_SystemSetupData.c
   - TSS_SystemSetupVal.h

3.2 Modify TSS_SystemSetup.h

There are two ways to edit the TSS_SystemSetup.h file.

- Generate a new file with the System Setup Creator (SSC).

- Modify the file directly.

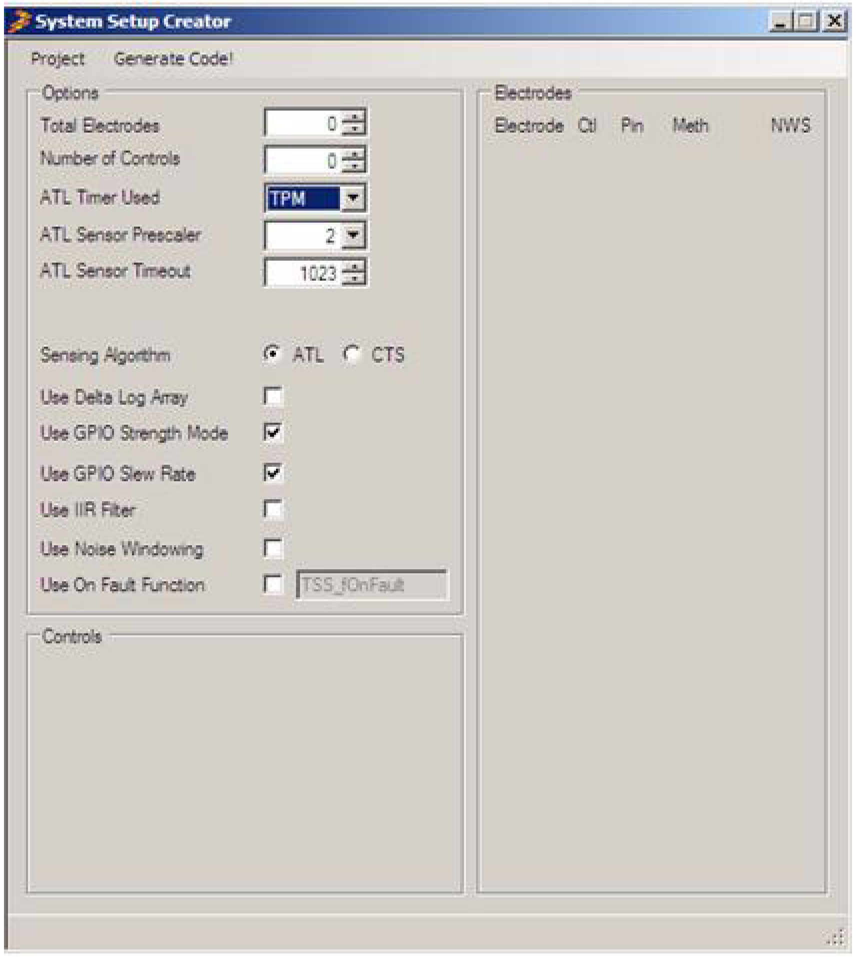
The former is recommended if you are using TSS 2.0 for the first time.

3.2.1 Using the System Setup Creator

Using the SSC, you can create a new file with the specific options for the project. To create a new file, fill out the options in the SSC. The TSS_SystemSetup.h file will include new macros to enhance the TSS performance and use.

To create a file using SSC, follow the steps below.

- Open the System Setup Creator (SSC), and click Start/Programs/Freescale/Touch Sensing Software v2.0/System Setup Creator. Figure 1 shows SSC.



Total Electrodes 0
Number of Controls 0
ATL Timer Used TPM
ATL Sensor Prescaler 2
ATL Sensor Timeout 1023
Sensing Algorithm [✓ ATL] CTS
Use Delta Log Array ☐
Use GPIO Strength Mode ✓
Use GPIO Slew Rate ✓
Use IIR Filter ☐
Use Noise Windowing ☐
Use On Fault Function ☐ TSS_fOnFault

Figure 1. System Setup Creator (SSC)

Unlike previous versions, now you can modify the ATL Prescaler and ATL Sensor Timeout from this file, include IIR filter as the new noise amplitude filter, and add the new callback error detection function. Additionally, the TSS 2.0 provides new capacitive sensing algorithms using port interrupts like KBI and TMP. For details, refer to the Touch Sensing Software API Reference Manual.

NOTE

The error callback function replaces the SWI interrupt in the TSS 1.x. If your project uses the SWI interrupt for fault detection, it is recommended to replace it with the new callback function as mentioned above. Backwards compatibility with the SWI was included for S08 devices, but if creating a project with other devices, the callback needs to be used. For details about writing the callback function, refer to Section 3.3, “Add OnFault Callback.”

4. Enter the electrode configuration with the same parameters as your original project. If you think the enhancements in TSS 2.0 are useful, you can select them. (If you want to implement the amplitude filter, choose an amplitude filter size. The basic recommendation is to use an amplitude filter size that is 10% - 20% the sensitivity threshold for the application. If this is less than 5, then use 5 (this is the minimum recommended value).

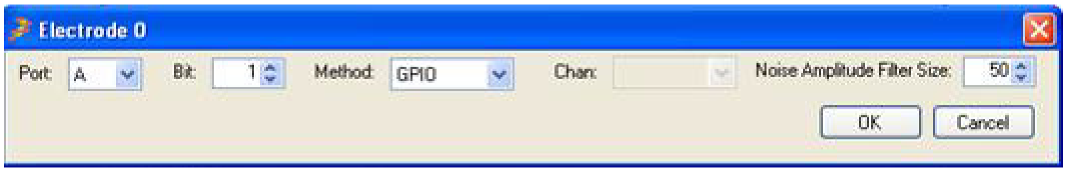


Figure 2. Parameters for each electrode with amplitude filter

5. Once you complete the information, go to the Generate Code! option (Figure 3). Select the folder that contains TSS_SystemSetup.h file.

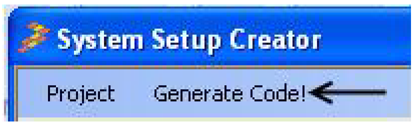
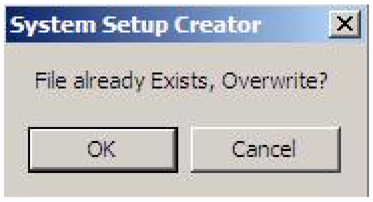


Figure 3. Generate Code! option

System prompts you to overwrite it. Click OK.



File already Exists, Overwrite?

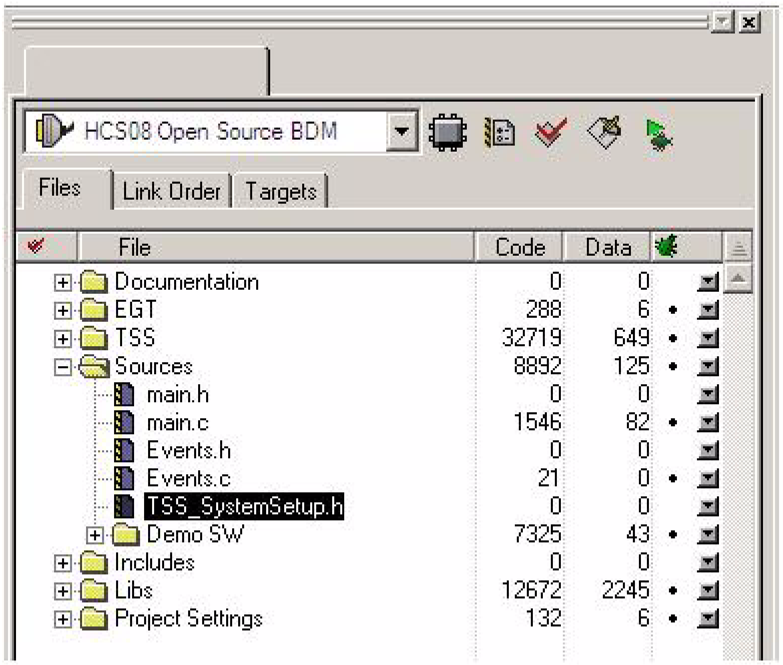
Figure 4. Overwrite protection window

6. Click OK. A new TSS_SystemSetup.h will be created in your project.

3.2.2 Directly Modifying TSS_SystemSetup.h

This is a faster option, but is only recommended when you have knowledge about the defines and TSS 2.0 changes.

1. Open the project with CodeWarrior, and go to the TSS_SystemSetup.h file (see Figure 5).



File	Code	Data	
Documentation	0	0	
EGT	288	6	•
TSS	32719	649	•
Sources	8892	125	•
main.h	0	0	
main.c	1546	82	•
Events.h	0	0	
Events.c	21	0	•
TSS_SystemSetup.h	0	0	
Demo SW	7325	43	•
Includes	0	0	
Libs	12672	2245	•
Project Settings	132	6	•

Figure 5. TSS_SystemSetup.h in CodeWarrior

2. Define the Prescaler and Timeout. If you want to implement a new option, define it. New macros are listed below. Set value as 1 to activate a new option, or 0 to deactivate.

```
#define ATL_SENSOR_PRESCALER               2
#define ATL_SENSOR_TIMEOUT                 0x1FF
#define TSS_USE_IIR_FILTER                 1
#define TSS_ONFAULT_CALLBACK               TSS_fOnFault
#define TSS_USE_NOISE_AMPLITUDE_FILTER     1
#define TSS_En_NOISE_AMPLITUDE_FILTER_SIZE 30
```

If you use the amplitude filter method, define the amplitude filter size. If you do not define this value, the window method will not work properly.

NOTE

The "n" letter represents the number for each electrode in the noise window size define.

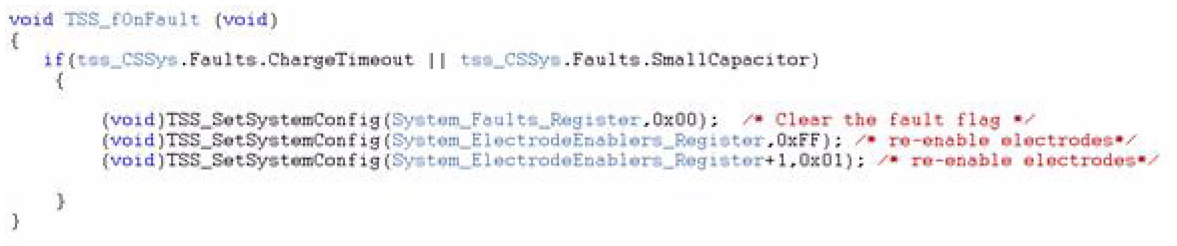
3.3 Add OnFault Callback

To use this callback to detect the fault error, add this function. This is an alternative to the SWI functionality of the HCS08 version of the library. The OnFault callback function is available for both

HCS08 and Coldfire V1 version of library and is a new recommended way to handle fault events detected by TSS code.

Steps to enable this option:

1. Write the callback define in the TSS_SystemSetup.h.
2. Create the callback function (see Figure 6).



```
void TSS_fOnFault (void)
{
   if(tss_CSSys.Faults.ChargeTimeout || tss_CSSys.Faults.SmallCapacitor)
    {

        (void)TSS_SetSystemConfig(System_Faults_Register,0x00);  /* Clear the fault flag */
        (void)TSS_SetSystemConfig(System_ElectrodeEnablers_Register,0xFF); /* re-enable electrodes*/
        (void)TSS_SetSystemConfig(System_ElectrodeEnablers_Register+1,0x01); /* re-enable electrodes*/

    }
}
```

Figure 6. Creating OnFault Callback

In this example, we clear the fault register and re-enable the electrodes. You can write the appropriate instructions for your project.

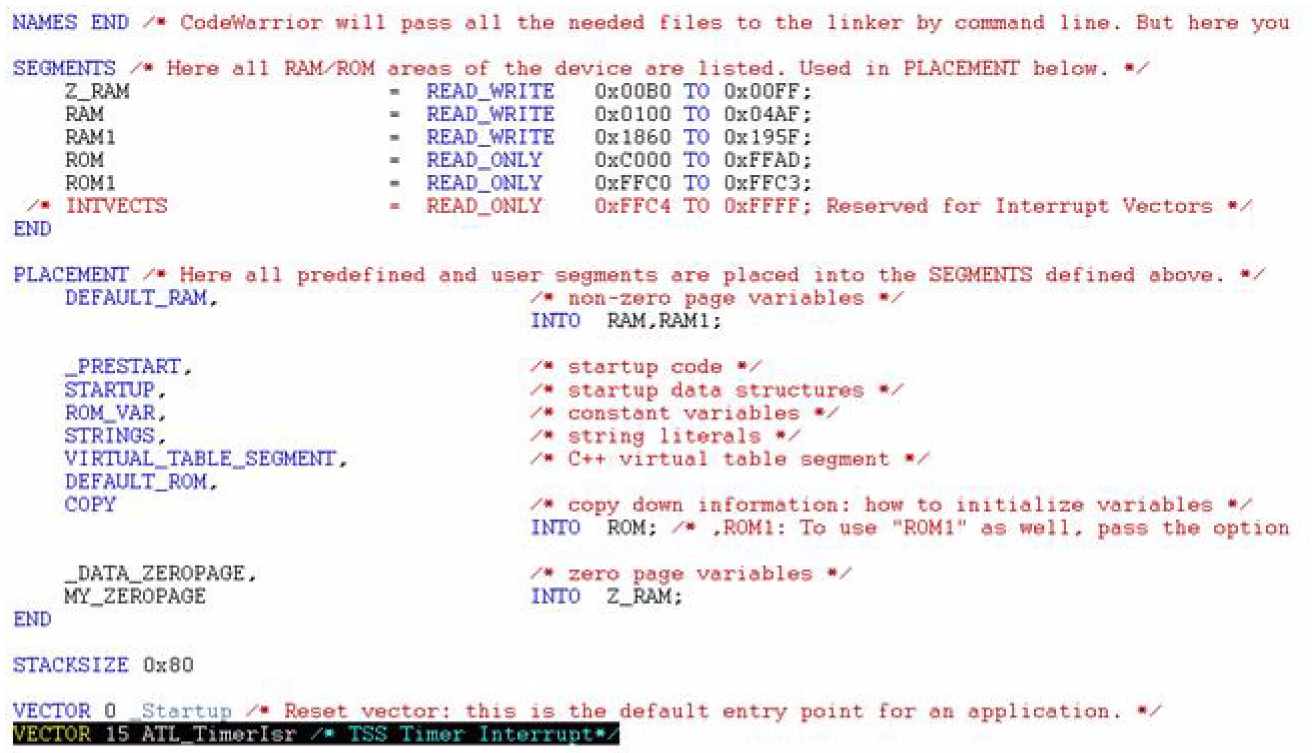
NOTE

Erase and deactivate the SWI if you were previously using it.

3.4 Erase TPM Over Flow Vector

Previously, in all TSS projects, it was required to declare the TMP Over Flow in the prm file. However, it is not necessary in the TSS 2.0 project. The final step in the migration is to erase this vector.

1. Go to the prm file in your project and erase the overflow vector.



```
NAMES END /* CodeWarrior will pass all the needed files to the linker by command line. But here you

SEGMENTS /* Here all RAM/ROM areas of the device are listed. Used in PLACEMENT below. */
    Z_RAM                    =  READ_WRITE   0x00B0 TO 0x00FF;
    RAM                      =  READ_WRITE   0x0100 TO 0x04AF;
    RAM1                     =  READ_WRITE   0x1860 TO 0x195F;
    ROM                      =  READ_ONLY    0xC000 TO 0xFFAD;
    ROM1                     =  READ_ONLY    0xFFC0 TO 0xFFC3;
 /* INTVECTS                 =  READ_ONLY    0xFFC4 TO 0xFFFF; Reserved for Interrupt Vectors */
END

PLACEMENT /* Here all predefined and user segments are placed into the SEGMENTS defined above. */
    DEFAULT_RAM,                      /* non-zero page variables */
                                      INTO  RAM,RAM1;

    _PRESTART,                        /* startup code */
    STARTUP,                          /* startup data structures */
    ROM_VAR,                          /* constant variables */
    STRINGS,                          /* string literals */
    VIRTUAL_TABLE_SEGMENT,            /* C++ virtual table segment */
    DEFAULT_ROM,
    COPY                              /* copy down information: how to initialize variables */
                                      INTO  ROM; /* ,ROM1: To use "ROM1" as well, pass the option

    _DATA_ZEROPAGE,                   /* zero page variables */
    MY_ZEROPAGE                       INTO  Z_RAM;
END

STACKSIZE 0x80

VECTOR 0 _Startup /* Reset vector: this is the default entry point for an application. */
VECTOR 15 ATL_TimerIsr /* TSS Timer Interrupt*/
```

Figure 7. Erasing TPM Over Flow Vector

With these changes, your project should be ready to work, compile, and flash the program in your MCU.

Document Number: AN4213
Rev. 0
09/2010