NXP Semiconductors
Application Notes

Document Number: AN_CSSWUG
Rev. 1.1, 05/2019

Capacitive Sensing SW User Guide

Using the S32K MCU

1. Introduction

This document describes settings and configuration options of the NXP GPIS Touch Sense solution software for the S32K general purpose MCU.

A GPIS Touch Sense solution is provided as a reference design. There are 3 different reference designs currently available for the S32K:

- S32K144 2-pad EVB
- S32K144 7-pad keypad
- S32K144 6-pad keypad with slider

Each reference design includes:

- Unique hardware (the demo board) with electrodes
- Software for capacitive touch sense applications
- Documentation

The Touch Sense solution software is common for all the reference designs. However, for proper functionality, the software must be configured according to the reference design board used. The software is configured by modifying pre-processor defines and macros in the configuration header files.

Contents

2. Reference Designs Overview

2.1. S32K144 2-pad EVB

The most straightforward and accessible option is the S32K144EVB Q100. This EVB includes 2 touch button electrodes. It is suitable for general testing of the GPIS Touch Sense solution.

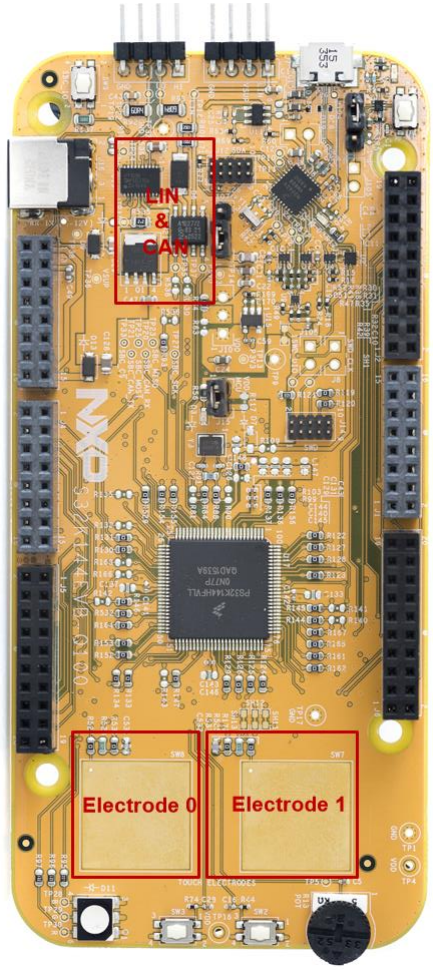

Figure 1. **S32K144EVB Q100**

2.2. S32K144 7-pad Keypad

The 7-pad keypad features 7 touch buttons with a 2 mm plastic overlay. To meet the 70uA average MCU current consumption requirement, a wake-up EGS (Electrode Global Sense) is also present on the board. This large electrode covers the area around all touch button electrodes. S32K144 7-pad keypad is the most suitable reference design for small or medium keypads solution evaluation.

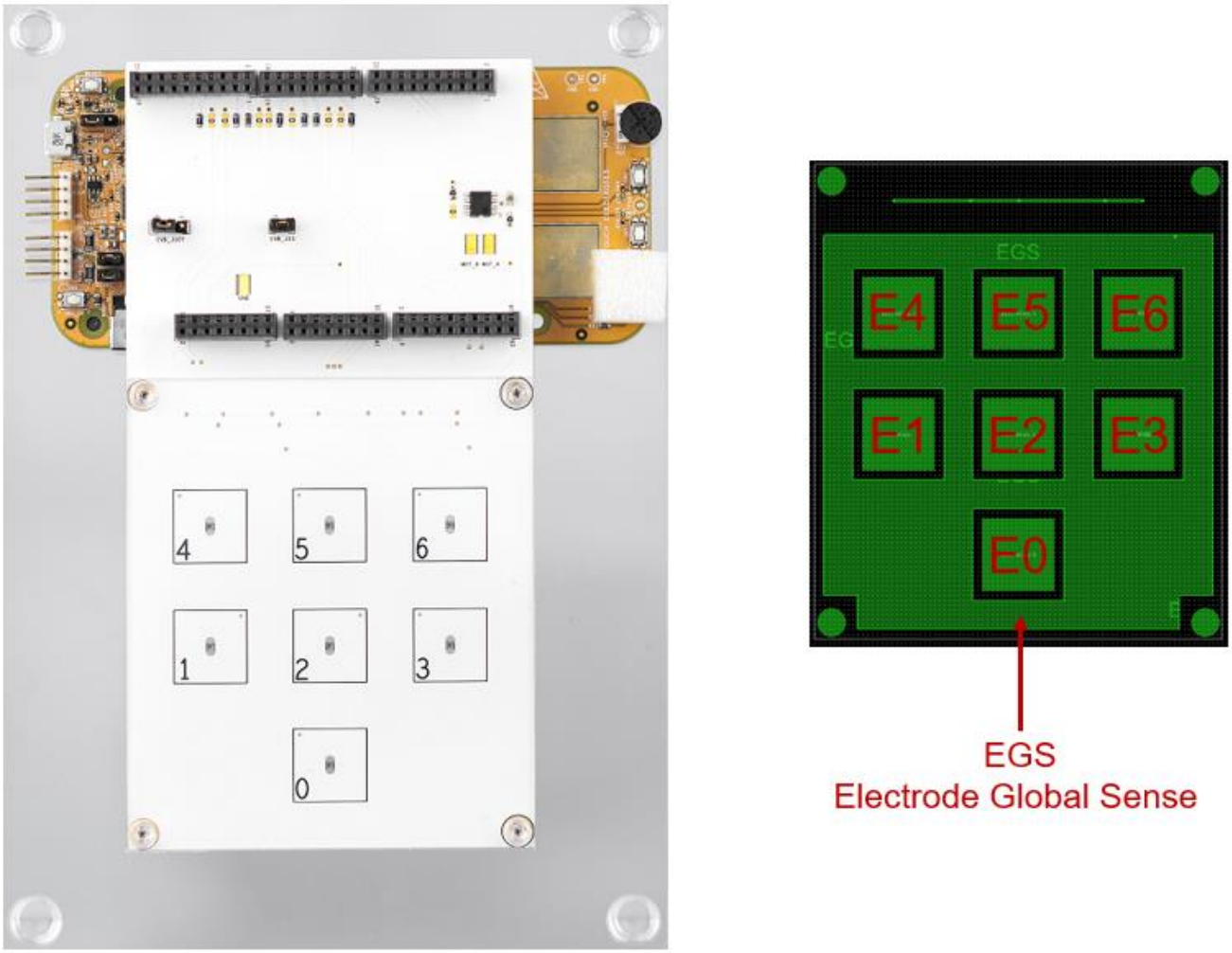

Figure 2. **S32K144 7-pad keypad reference design**

2.3. S32K144 6-pad Keypad with Slider

There are 6 touch button electrodes and a slider available on the S32K144 6-pad keypad with slider demo board. All electrodes are covered by a 2 mm plastic overlay. The 60 mm slider consists of two wedge shaped electrodes. As in the case of the S32K144 7-pad keypad reference design, a wake-up EGS electrode is also present on the board. The EGS shape is reduced to a thin grid for better slider sensitivity. This reference design is the most suitable if an application is needed using both touch buttons and slider.

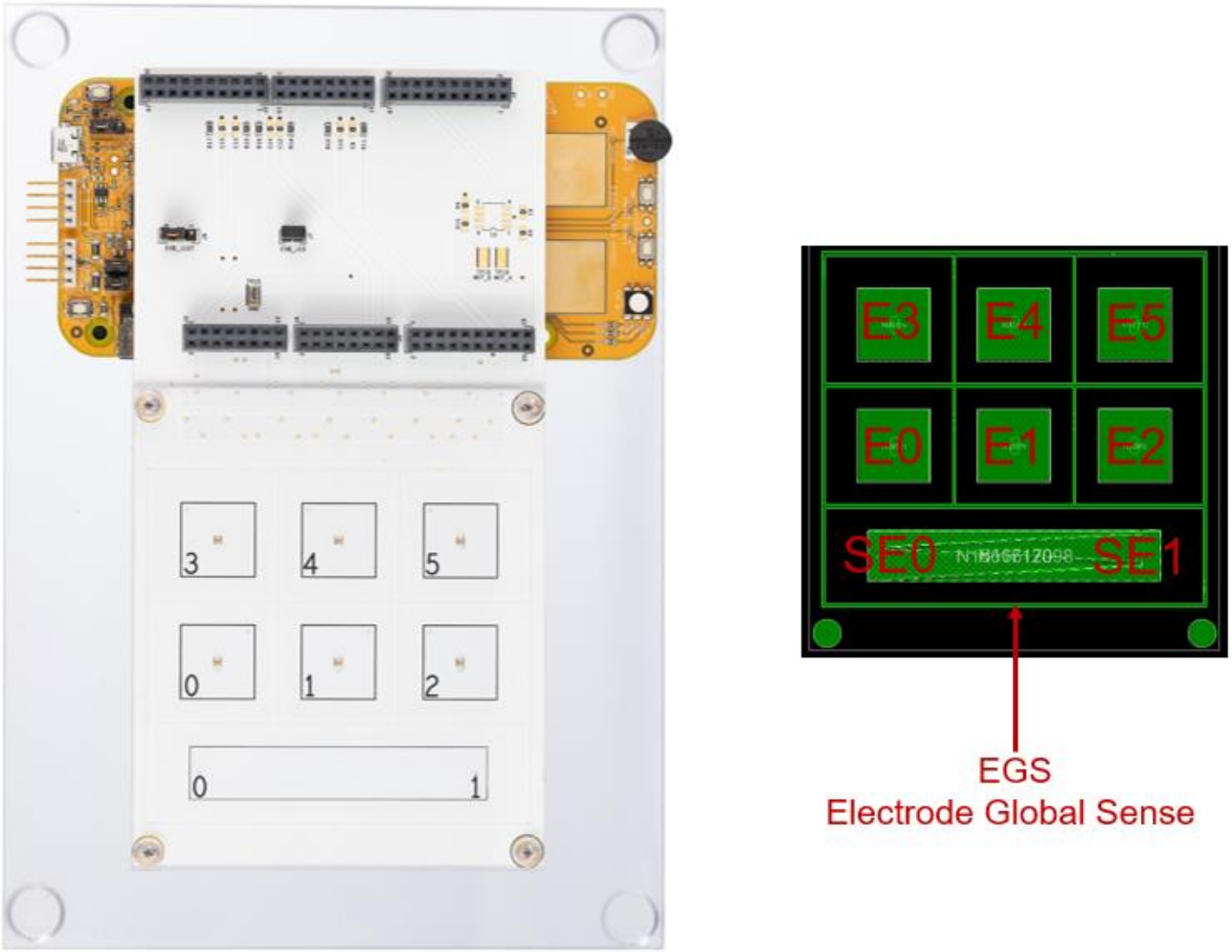

Figure 3. **S32K144 6-pad keypad with slider reference design**

3. Touch Sense Software Functionality

In this chapter, the software is described as a bottom-up hierarchical structure. To understand the software properly, follow the chapters in order.

Firstly, the core GPIS Touch Sense method for capacitive touch sensing (converting electrode capacitance change into a digital value) is explained (3.1). Secondly, individual software layers are described (3.2). Thirdly, a software overview and algorithm flowcharts are presented (3.3). And finally, Touch Sense software low power performance is explained.

3.1. Electrode Sensing Method

The GPIS Touch Sense method was developed for general purpose MCU and is easily portable between S32K MCU family. For each electrode pad, two GPIO pins are required. One of the pins must have ADC functionality and both pins must share the same MCU port.

The method consists of three steps:

1. Charge Distribution – both GPIO pins are simultaneously set as outputs, while the external capacitor (C_{EXT}) is charged, and the electrode is discharged. ADC conversion is started by a software trigger.

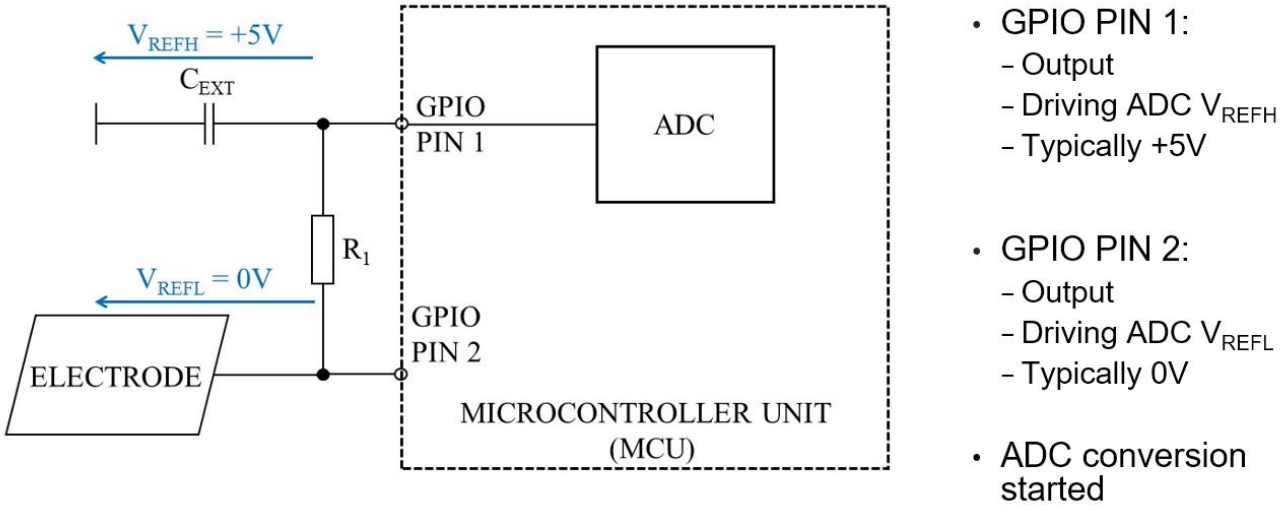

Figure 4. **Charge Distribution**

2. Charge Redistribution – both GPIO pins are simultaneously set as inputs (high impedance) and charge is redistributed between the external capacitor and electrode. ADC conversion is in progress.

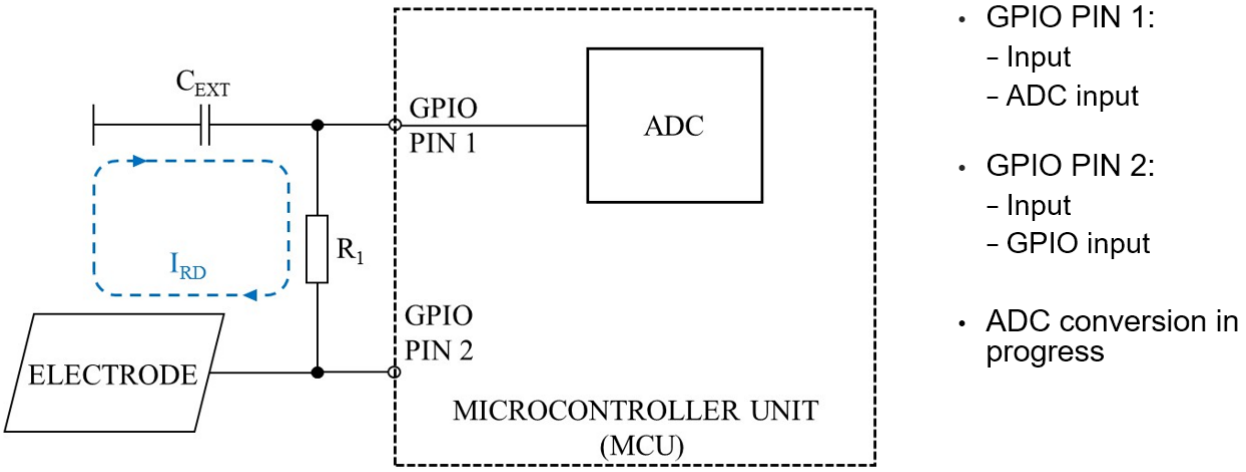

Figure 5. **Charge Redistribution**

3. Equivalent Voltage Digitalization – when voltage across the external capacitor and electrode no longer differs, the electrode equivalent voltage is sampled by the ADC and is thus converted from analog into the digital domain. The ADC conversion is finished.

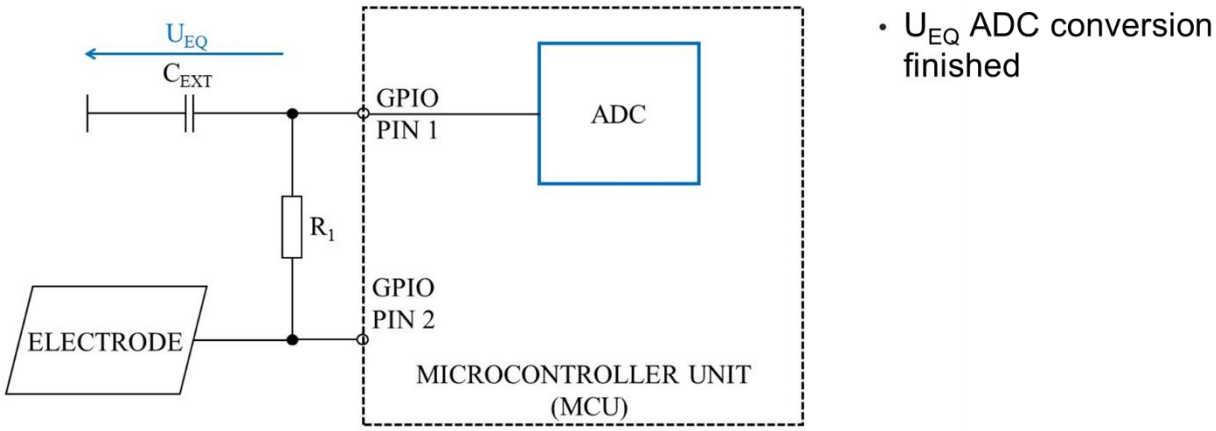

Figure 6. **Equivalent Voltage Digitalization**

These three steps of the TS method make up one electrode sensing cycle. One electrode sensing cycle takes 2µs and technically returns one ADC sample (Figure 7). However, the TS method is consecutively repeated multiple times (Figure 8) and the final electrode sample is typically calculated as an average from a set of multiple electrode sensing cycles. By default, every 30ms (electrodes sensing period), a new set of electrode sensing cycles is taken, and the new final electrode sample is calculated. This final electrode sample is also called the electrode raw data. The entire electrodes sensing period procedure described can be seen in Figure 9 and the Raw Data Sampling software layer (chapter 3.2) is responsible for this procedure.

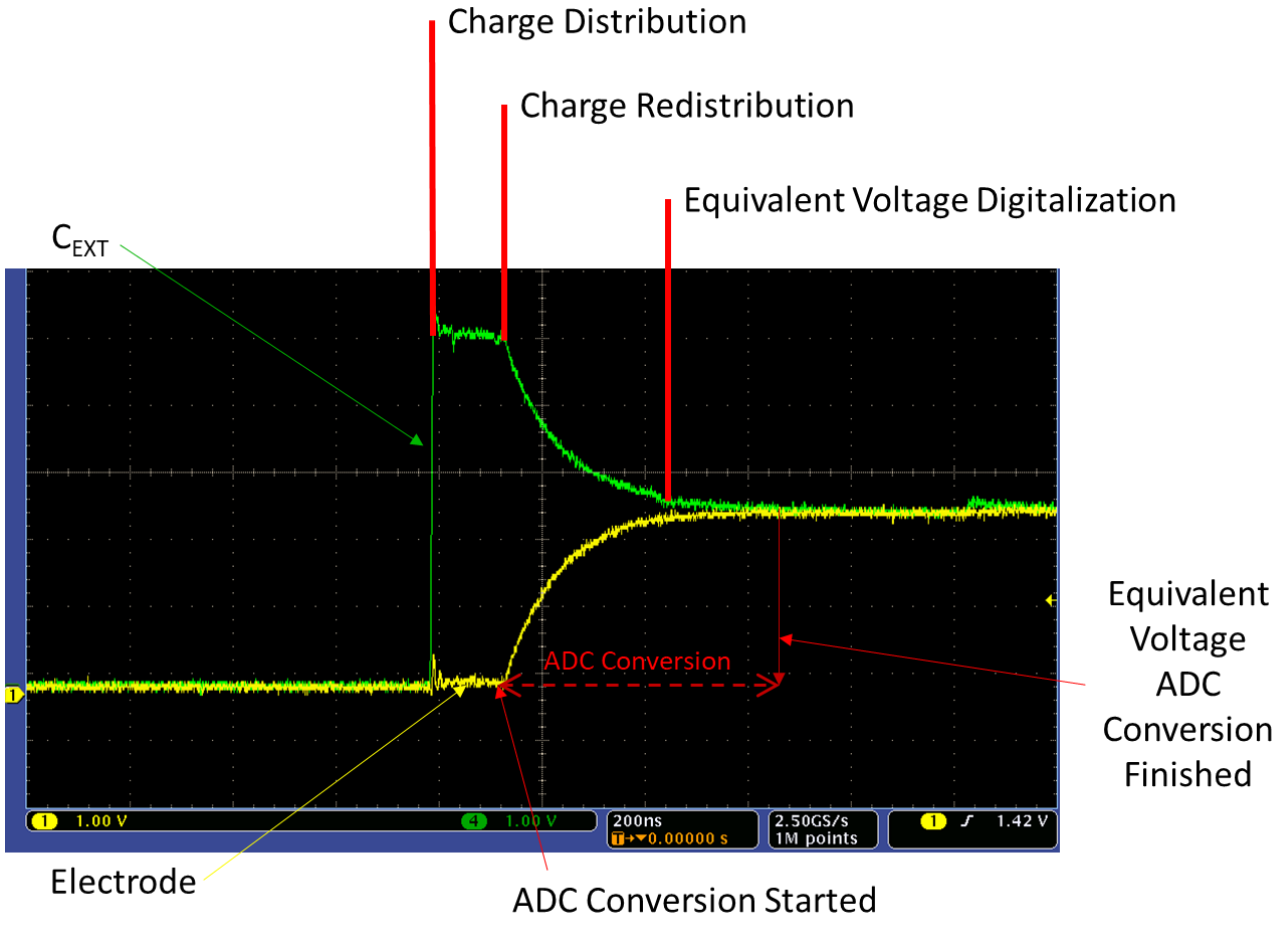

Figure 7. **One electrode sensing cycle signal shapes**

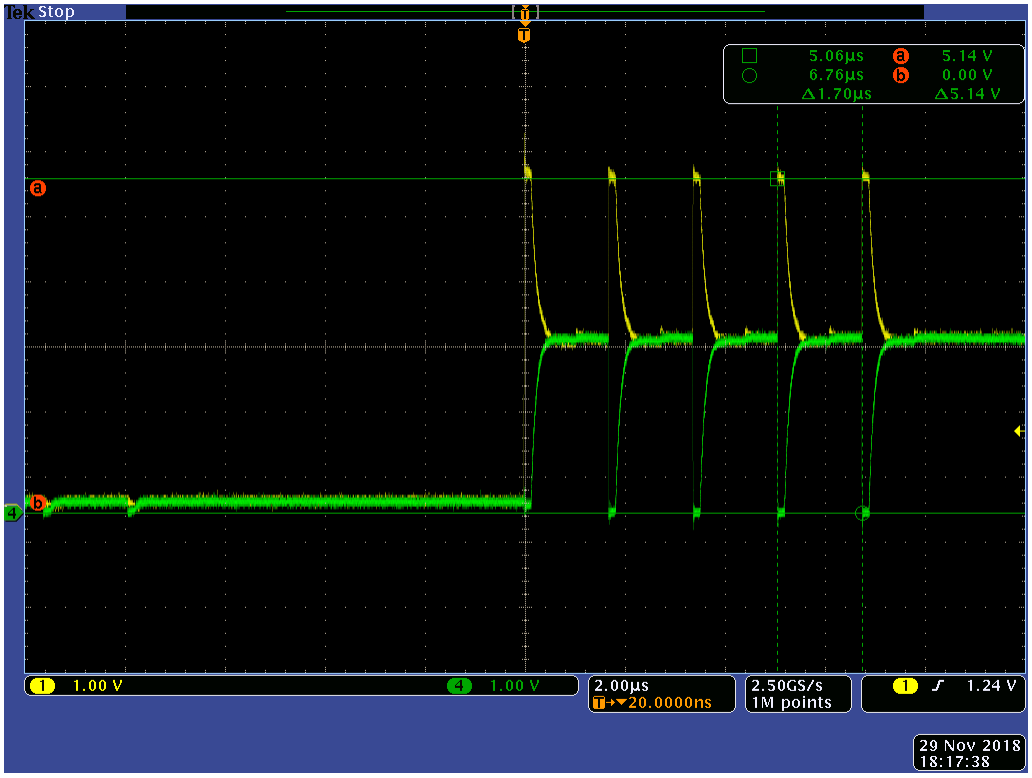

Figure 8. **Set of electrode sensing cycles**

When touching the keyboard 2 mm plastic overlay, the electrode self-capacitance increases by approx. 2%, which roughly equals 0.1pF. If the electrode capacitance increases, the scanned equivalent voltage decreases.

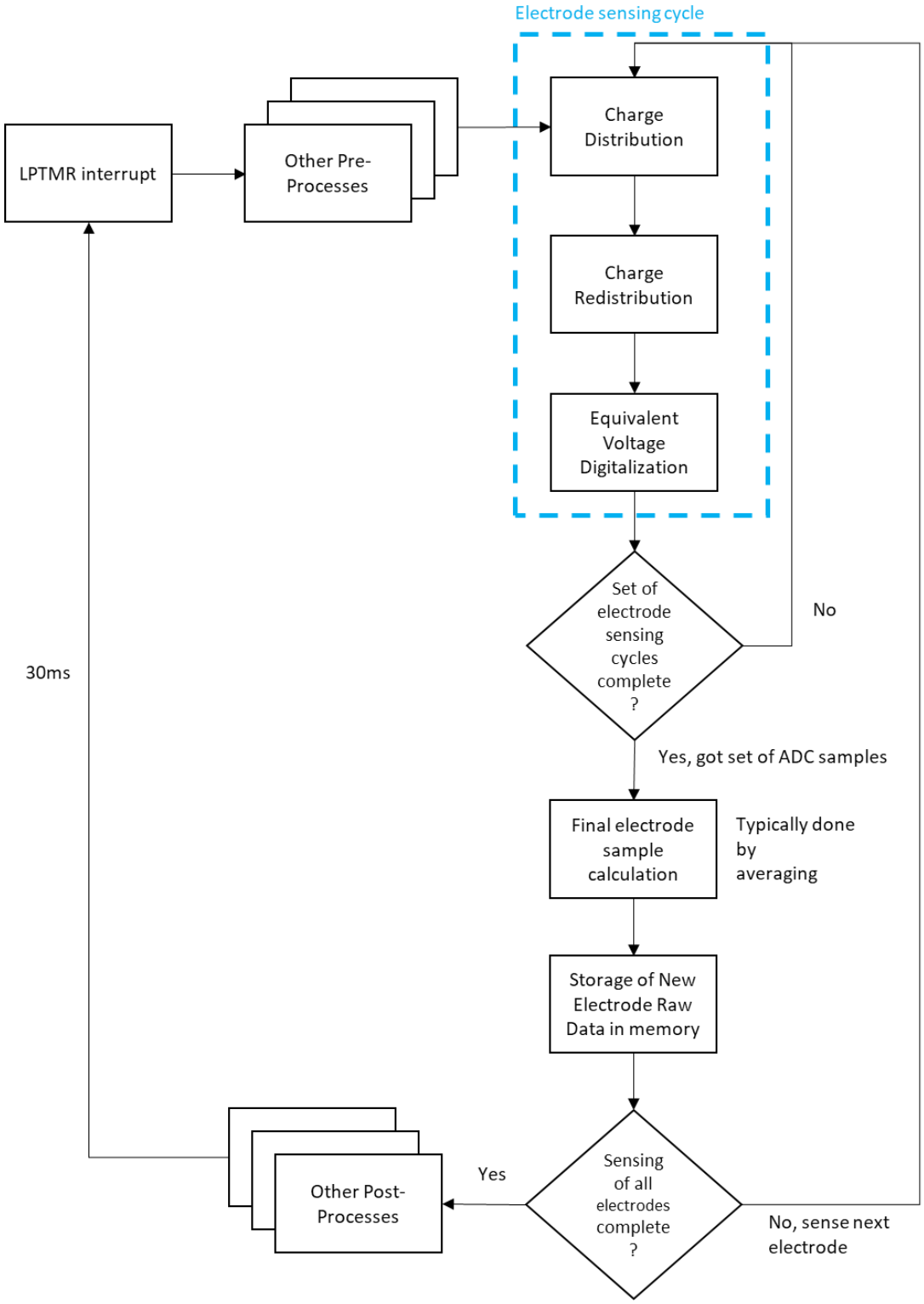

Figure 9. **Electrodes sensing procedure - Raw Data Sampling flowchart**

3.2. Software Layers

There are 3 main software layers (also part of the flowcharts in chapter 3.3):

1. Raw Data Sampling
    - Uses the GPIS Electrode Sensing Method (see previous chapter 3.1)
    - Delivers the electrode raw data – the final electrode sample calculated from a set of multiple repetitive electrode sensing cycles (Figure 9)
    - Operates a timer (LPTMR) – on every electrode sensing period (30ms) a new set of electrode sensing cycles is taken, and the new final electrode sample is calculated
2. Digital Signal Processing
    - Filters the electrode raw data – improving the S/N ratio (LP IIR filter)
    - Establishes the electrode baseline (DC tracker) to react on environment change using a very slow LP filter
    - Sets the electrode touch and release threshold relatively to the electrode baseline
3. Touch detect algorithm
    - Creates a touch sense application user interface
    - Detects and reports a touch or release event for all electrodes (Figure 10)
    - Qualifies the touch event for all electrodes – with on-board RGB LED signaling
    - Calculates slider data and determines the slider finger position (Figure 12)

When the LP filter data signal of a touch button electrode drops below the touch threshold, the touch event is detected (Figure 10).

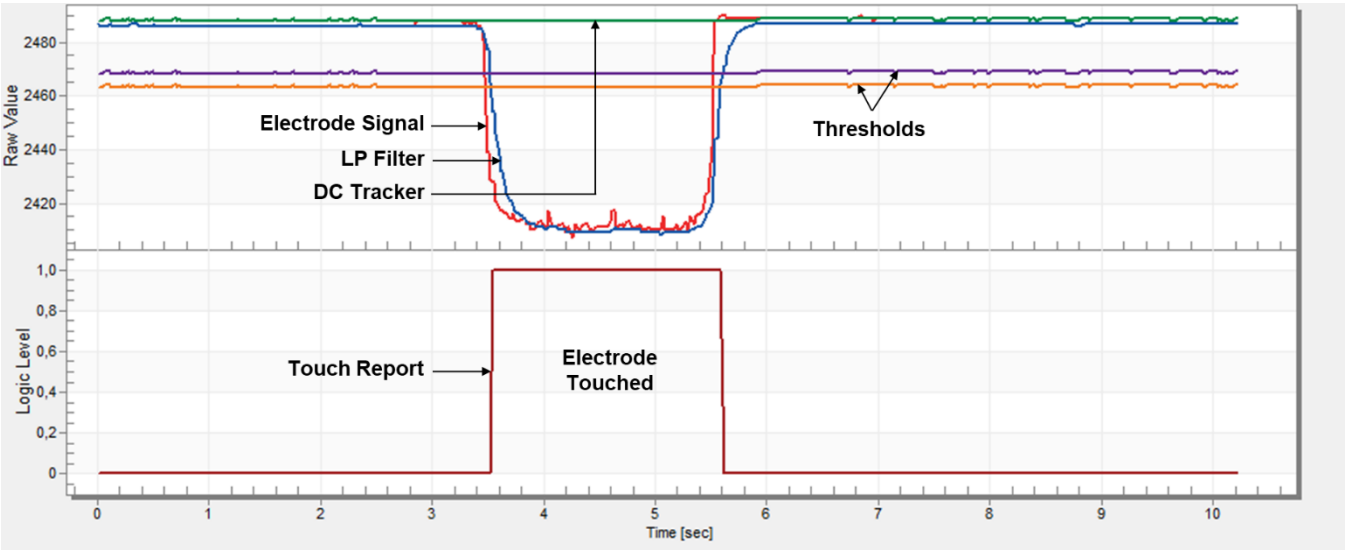

Figure 10. **Touch button electrode touch detection**

In the case of a slider application, the touch detect algorithm is more complicated and is described in the following section 3.2.1.

3.2.1. Slider Application Functionality

As in the case of a touch button electrode, the touch event is detected when the slider electrode LP filter data signal drops below the touch threshold. The touch event must be then confirmed – slider addition data must drop below the addition data threshold (Figure 11).

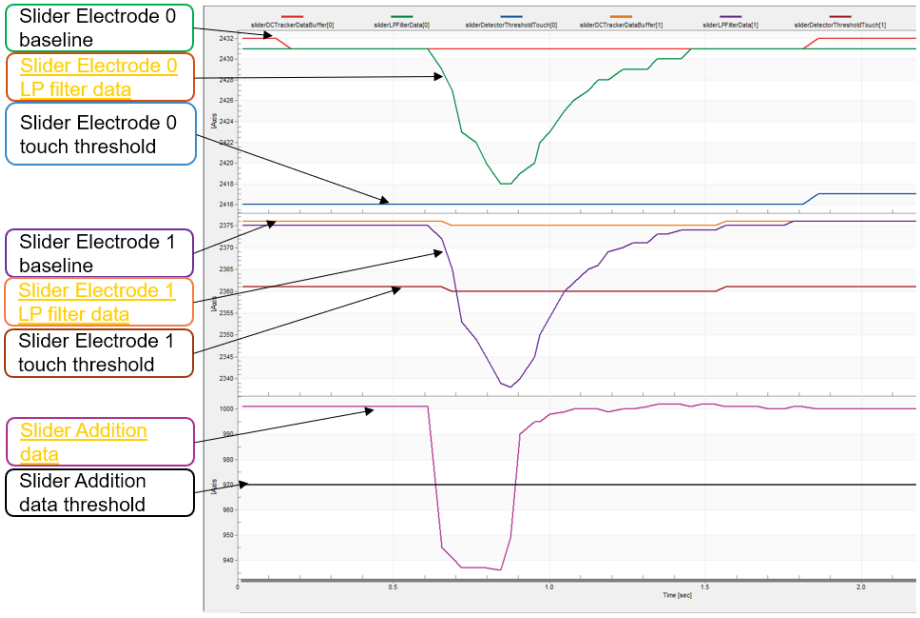

Figure 11. **Slider electrodes touch detection and confirmation**

When a slider touch event is detected and confirmed, the finger position is determined from the slider difference data. The slider difference data are calculated by subtracting the slider electrodes raw data from each other.

When using the S32K14x MCU family, 2 ADC modules can be used for scanning the 2 slider electrodes at one time (simultaneously). Therefore, the noise coupled is the same on both electrodes. If the slider electrodes raw data are simply subtracted from each other, a clean slider difference signal is obtained for the finger position determination process.

The slider resolution is typically expressed in bits. From this point of view, the GPIS slider application is nearly 6-bit as the slider data achieves a 50-step resolution.

In the application, the slider area (60 mm) is divided by software into 5 virtual segments, representing a typical HVAC fan speed control application (Figure 12). The slider data range (50) is split equally between the segments resulting in a 10- step bandwidth per segment. The number of virtual segments is configurable.

For more information, please see the Slider Application Quick Start Guide by clicking the link in chapter 7:Useful Links .

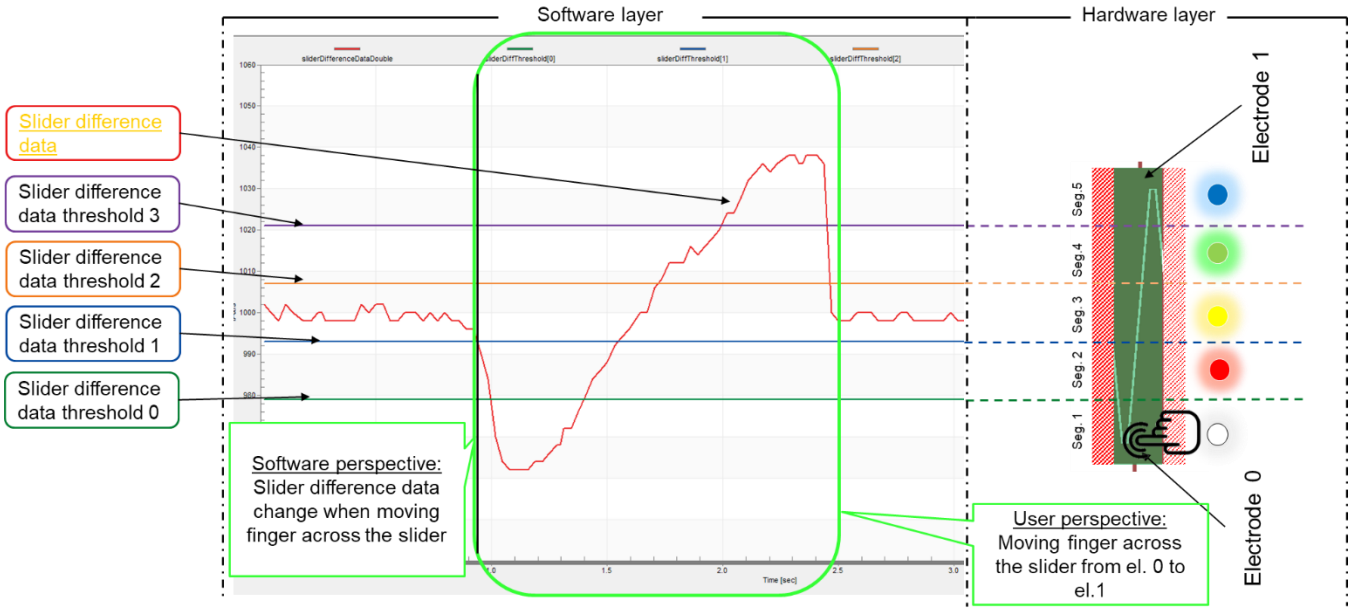

Figure 12. **Slider difference data and finger position determination**

3.3. Software Overview and Flowcharts

The Touch Sense software flow is bound to and adjusted to strict low power requirements. In general, the average MCU current consumption in the GPIS Touch Sense software application is dependent on 3 factors, sorted by impact from highest to lowest:

1. Electrodes sensing period
    - If the electrodes sensing period is decreased, electrodes are scanned more often, the MCU spends less time in SLEEP mode, and thus the average MCU current consumption increases.
    - The default electrodes sensing period is 30ms.

2. Presence and usage of the wake-up EGS (Electrode Global Sense) electrode
    - In general, the more electrodes to be scanned, the higher the power consumption.
    - Therefore, the wake-up EGS electrode was implemented. When the wake-up EGS electrode is used and no touch event is detected, only 2 electrodes (EGS and one more touch electrode) are scanned each cycle.
    - The software algorithm with and without the EGS is described in the following sections 3.3.1 and 3.3.2
    - The EGS electrode is available and can be enabled on S32K144 7-pad Keypad and S32K144 6-pad Keypad with Slider reference designs
    - How the EGS looks like on each reference design board is shown in chapter 2:Reference Designs Overview

3. Number of electrode sensing cycles per sample
    - As described in section 3.1, the final electrode sample is calculated as an average from a set of consecutive electrode sensing cycles.
    - The lower the number of electrode sensing cycles per sample, the faster each electrode sensing is completed, and the MCU current consumption decreases.
    - Related to the number of electrode sensing cycles per sample, two sensing modes are implemented in software: idle sensing mode and active sensing mode.
    - Idle sensing mode:
        i. The final electrode sample is typically calculated as an average from a set of only 4 electrode sensing cycles (idle number of sensing cycles per sample)
        ii. The smaller number of electrode sensing cycles contributes to the lower average MCU current consumption

iii. If a touch event detected, the number of electrode sensing cycles is changed to 16 (active number of sensing cycle per sample) for the next electrode sensing procedure

- Active sensing mode:
  i. The final electrode sample is typically calculated as an average from a set of 16 electrode sensing cycles (active number of sensing cycle per sample)
  ii. Touch can be qualified and signalized by the on-board LED
  iii. The higher number of electrode sensing cycles contributes to a higher EMI resistance
  iv. If a touch event is not detected, and thus not qualified, the software switches the number of sensing cycles back to 4 (idle number of sensing cycles per sample)

The following sections 3.3.1 and 3.3.2 describe the software algorithm with and without the wake-up EGS electrode.

3.3.1. Software Overview and Flowchart with Wake-up EGS

When the wake-up EGS electrode is present on the board and is enabled, the transition from idle to active sensing mode is based on the touch or release event detected on this EGS electrode.

With wake-up EGS, the software follows this algorithm (Figure 13):

1. The MCU is awoken by the LPTMR (Low Power Timer) interrupt
2. The MCU scans the EGS in the idle sensing mode and processes the EGS raw data
3. The MCU also scans also one other keypad single electrode “En” to the track electrode baseline; “En” is changed (to another keypad electrode) on each wake-up cycle
4. EGS touch or release event evaluation
   a. EGS release event? – the MCU enters SLEEP mode with a 30ms wake-up period
   b. EGS touch event? – the active sensing mode is set, the MCU senses all keyboard touch electrodes, processes their raw data and the touch electrodes touch or release event evaluation follows:
      i. All touch electrodes released – the MCU enters SLEEP mode with a 30ms wake-up period if the backlight 3s delay period is over. Then the idle sensing mode for the next sensing period is set
      ii. Any touch electrode touched – the touch is signalized by the on-board RGB LED, backlight LEDs are activated for the next 3 seconds. The MCU enters RUN mode with a 30ms electrodes sense period. The software senses all the touch electrodes (but not the EGS) in the active sensing mode. The MCU stays in RUN mode until all touch electrodes are released and the backlight 3 seconds delay period times out

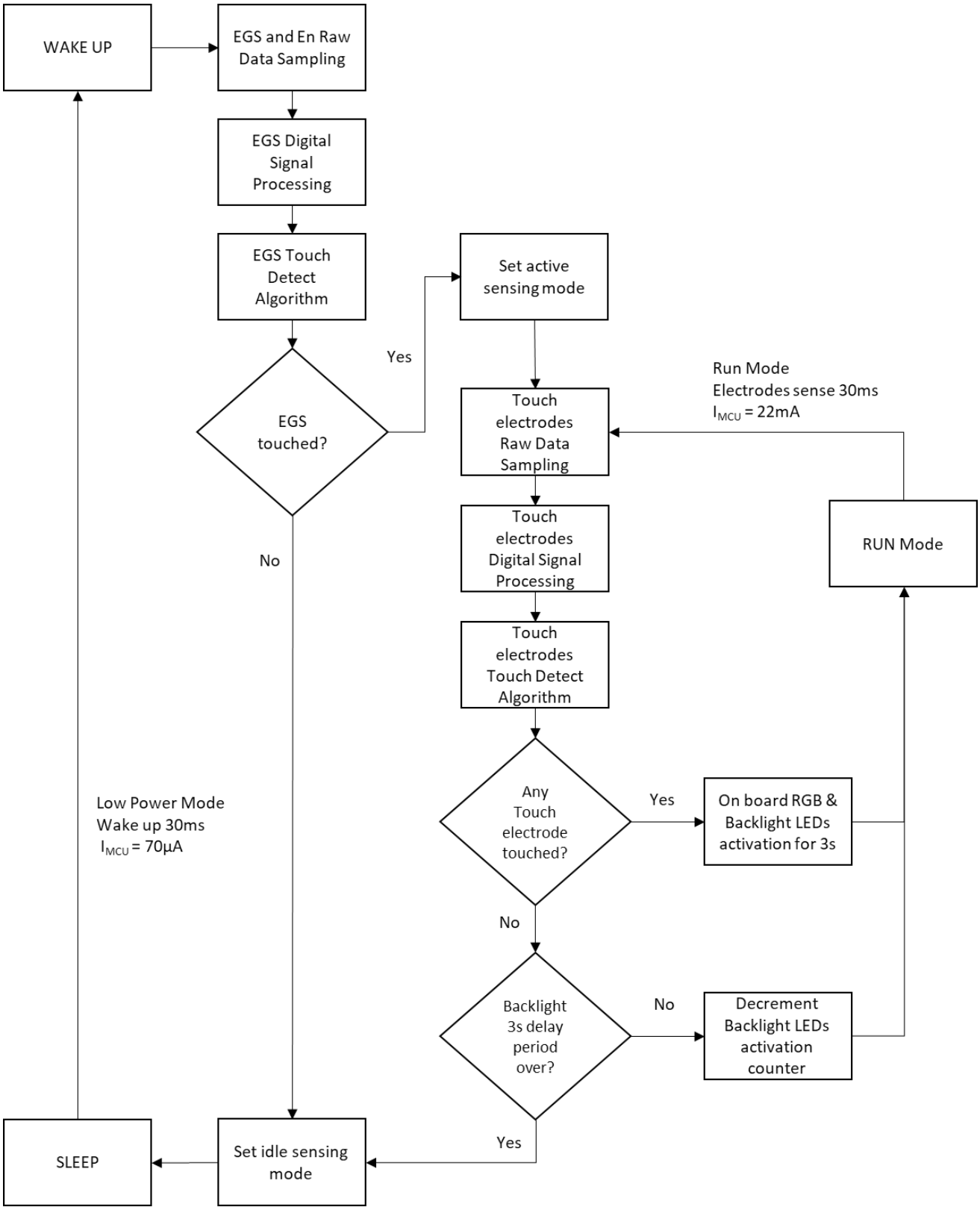

Figure 13. **Software flowchart when wake-up EGS electrode is used**

3.3.2. Software Overview and Flowchart without Wake-up EGS

When the wake-up EGS electrode is not present on the board or is disabled, the transition from idle to active sensing mode is done by the “Virtual EGS”. The virtual EGS is basically another touch threshold implemented for every touch electrode. The virtual EGS touch threshold is typically very close to the electrode baseline and can be modified by the user (see 5.4.9).

When the software is in idle sensing mode and LP filter data of a touch electrode drop below the virtual EGS touch threshold, the software switches to active sensing mode for next sensing period. In the next sensing period after waking up from SLEEP, the electrodes are scanned in active sensing mode. Therefore, the software can proceed to the touch electrodes touch or release event evaluation and eventually enter the RUN mode in case the touch event was detected.

Without wake-up EGS, the software follows this algorithm (Figure 14):

1. The MCU is awoken by the LPTMR (Low Power Timer) interrupt
2. The MCU senses all keyboard touch electrodes in idle or active sensing mode based on previous sensing period
3. The MCU processes all keyboard touch electrodes raw data
4. Virtual EGS threshold touch or release event evaluation:
    a. The virtual EGS released? – the idle sensing mode is set and the MCU enters SLEEP mode with a 30ms wake-up period
    b. The virtual EGS touched, but the idle sensing mode is currently set? – the software switches to active sensing mode for next sensing period and the MCU enters SLEEP mode with a 30ms wake-up period
    c. The virtual EGS touched and the active sensing mode is currently set? – touch electrodes touch or release event evaluation:
        i. All touch electrodes released – the MCU enters SLEEP mode with a 30ms wake-up period if the backlight 3s delay period is over. Then the idle sensing mode for the next sensing period is set
        ii. Any touch electrode touched – the touch is signalized by the on-board RGB LED, backlight LEDs are activated for the next 3 seconds. The MCU enters RUN mode with a 30ms electrodes sense period. The software senses all the touch electrodes (but not the EGS) in the active sensing mode. The MCU stays in RUN mode until all touch electrodes are released and the backlight 3 seconds delay period times out

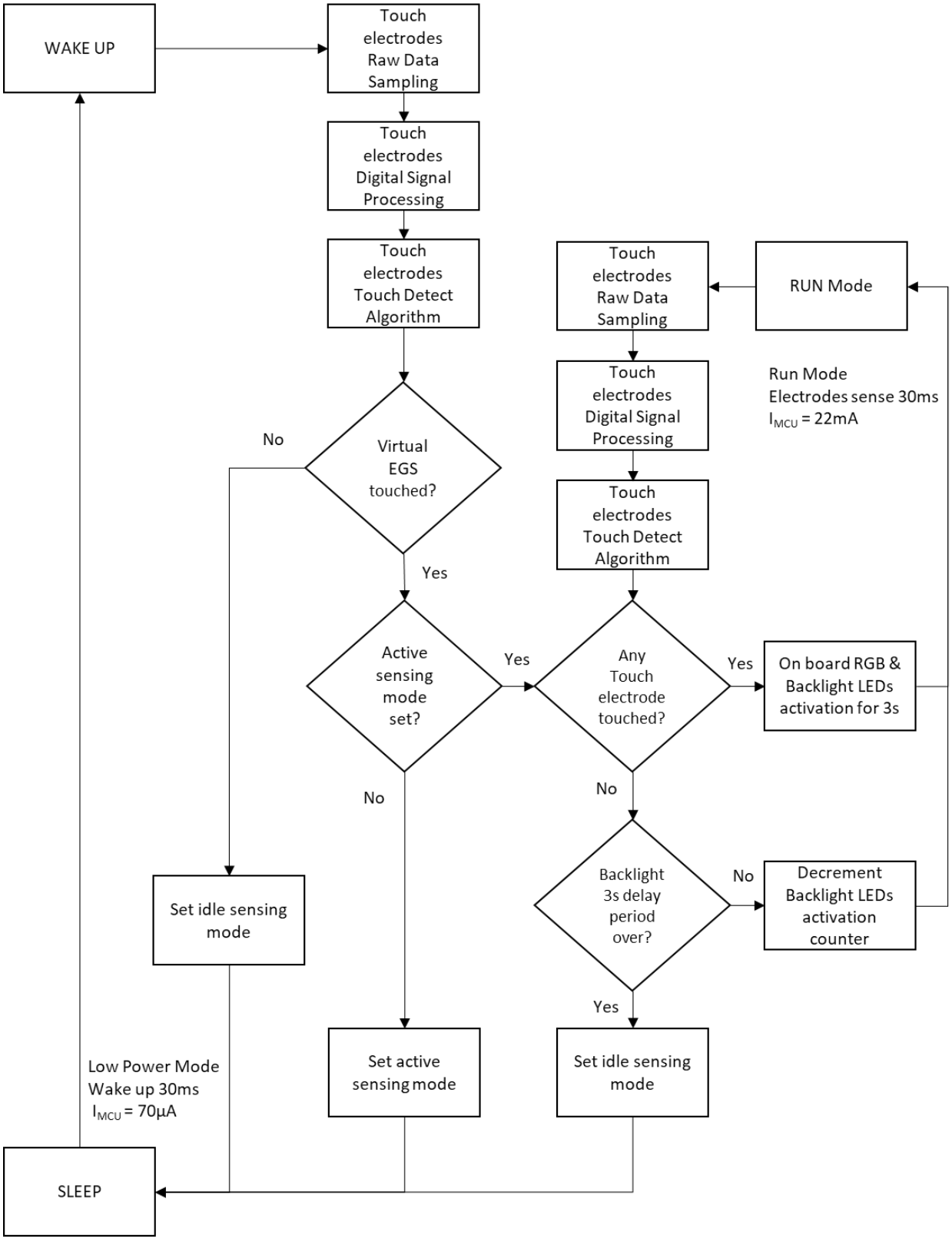

Figure 14. **Software flowchart without wake-up EGS electrode**

4. Tools Installation and Setup

To be able to flash and configure the Touch Sense software into the chosen reference design board, follow these steps:

1. Download and install the latest version of the following tools:
   a. S32 Design Studio IDE for Arm® based MCUs
   b. FreeMASTER Run-Time Debugging Tool
   c. Touch Sense software package:" S32K144_TS_40_COMMON_TOUCH_SENSE_SOFTWARE_SOLUTION.exe"
2. Run the S32DS and import the installed project "S32K144_TS_40" into workspace: File → Import → Existing projects into Workspace
3. Check the project compiler optimization level (must be -O3): Right-click imported project in Project explorer section → Properties → C/C++ Build → Settings → Tool Settings card → Standard S32DS C Complier → Optimization

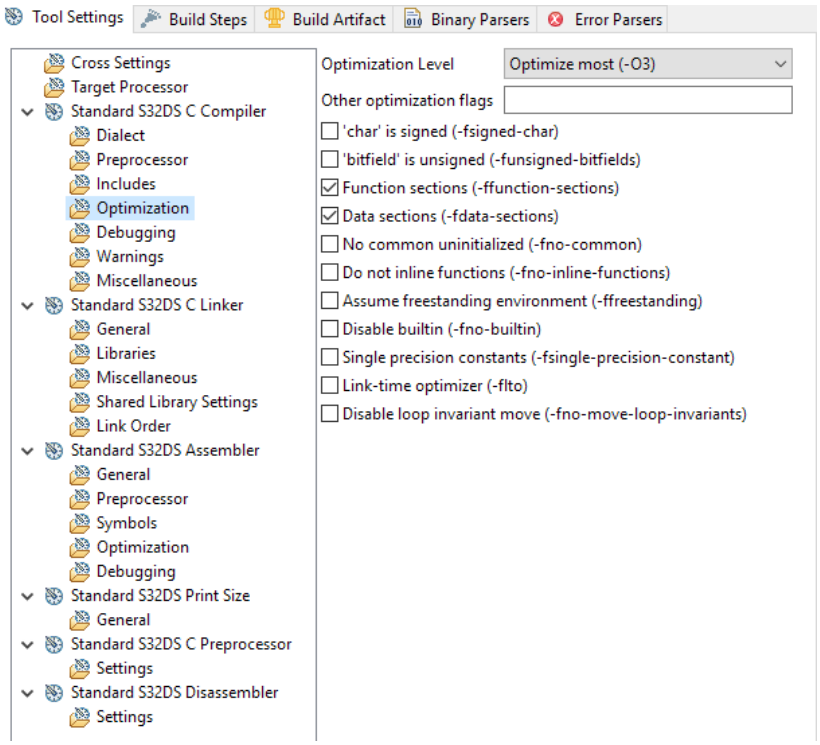

Figure 15. **Optimization settings**

5. Touch Sense Software Configuration

As mentioned in the Introduction, the header files of the Touch Sense software must be configured with respect to the used reference design for correct functionality. All configuration header files are in the TS project “Cfg” folder (Figure 16).

In the configuration files there are preset default configuration settings for all the reference designs. If the user is satisfied with these default configuration settings, only the reference design board has to be chosen in the header files. To set the correct reference design board, move to the Reference Design Board Selection (5.2.1).

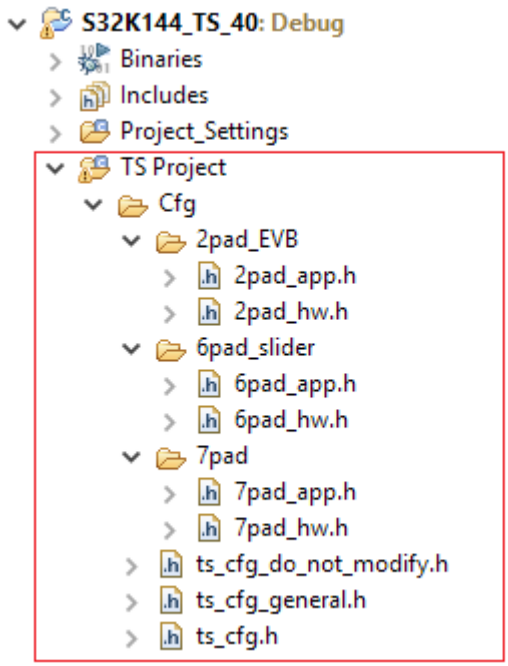

Figure 16. **Configuration folder with header files**

.

5.1. Configuration Header Files Organization

The configuration header files inclusion procedure is depicted in Figure 17. The .h files in blue are the ones that can be modified by the user:

1. A general configuration header file (“ts_cfg_general.h”) is included into hardware (“*pad_hw.h”) and application (“*pad_app.h”) header files of each reference design.
2. The user then modifies the hardware (“*pad_hw.h”) and application (“*pad_app.h”) header files of the chosen reference design board only.
3. Hardware (“*pad_hw.h”) and application (“*pad_app.h”) header files are included in the final common “ts_cfg.h”, which is included in all the other touch sense application .c and .h files.

A detailed description of all the defines, which can be modified and set by the user, is presented in chapters 5.2 General Configuration, 5.3 TS Hardware Configuration, 5.4 TS Application Configuration and 5.5 Common Application Settings.

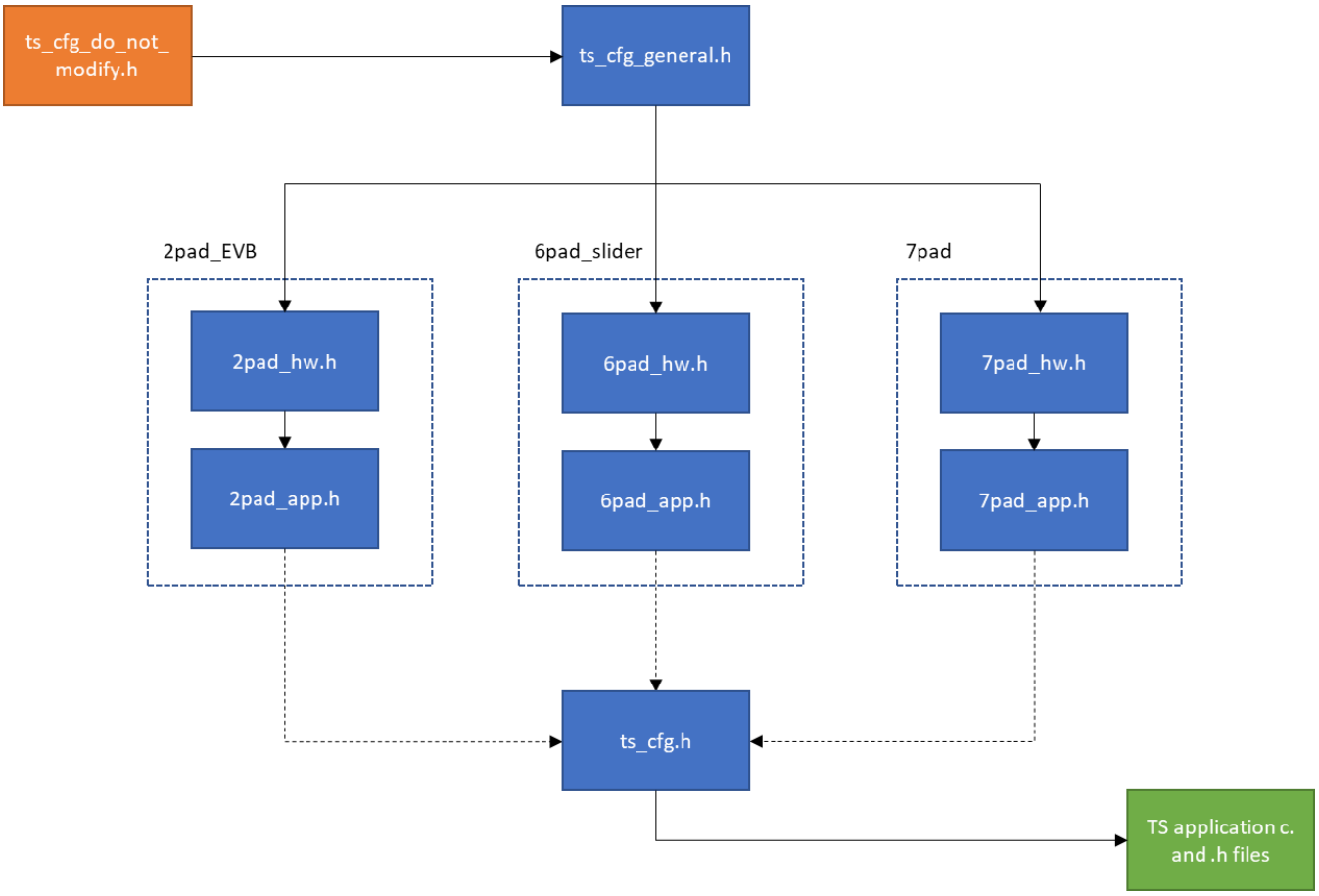

Figure 17. **Touch Sense configuration header files inclusion procedure**

5.2. General Configuration

This configuration is done in the “ts_cfg_general.h” header file. All general configuration defines which can be modified by the user are described in subchapters 5.2.1 and 5.2.2.

5.2.1. Reference Design Board Selection

Choose your REFERENCE_DESIGN_BOARD (Figure 18).

```
// Select the available type of hardware for TS application (S32K144_2PAD_EVB, S32K144_7PAD_KEYPAD or S32K144_6PAD_KEYPAD_SLIDER)
#define REFERENCE_DESIGN_BOARD S32K144_6PAD_KEYPAD_SLIDER
```

Figure 18. **Reference design board selection define**

After choosing your reference design board, these three steps must be followed:

1. Clean the project: Right-click the imported project in the Project explorer section → Clean Project
2. Rebuild indexing: Right-click the imported project in the Project explorer section → Index → Rebuild
3. Build project: Right-click the imported project in the Project explorer section → Build Project

In the rest of the configuration header files, there are preset default configuration settings for all the reference designs. If the user is satisfied with these default configuration settings**,** proceed to Chapter 6: Building and Debugging the Application.

5.2.2. Raw Data Calculation Formula

As mentioned in chapter 3.1, the final electrode sample is calculated from a set of multiple consecutive sensing cycles. Choose the formula for raw data calculation (TS_RAW_DATA_CALCULATION):

- Averaging: Electrode sample is calculated as an average of multiple electrode sensing cycles
  - Advantage: Typical approach, better EMI resistance
  - Disadvantage: Lower sensitivity
- Oversampling: Electrode sample is calculated as a sum of multiple electrode sensing cycles
  - Advantage: Susceptible to EMI
  - Disadvantage: Higher sensitivity

```
// Define the main raw data calculation formula (AVERAGING OR OVERSAMPLING)
#define TS_RAW_DATA_CALCULATION   AVERAGING
```

Figure 19. **Touch sensing raw data calculation define**

5.3. TS Hardware Configuration

All hardware configuration options (defines) which can be modified by the user are described in Table 1. Table 1 also shows to which hardware configuration files (with respect to the chosen reference design) the configuration defines are applicable. All hardware configuration options are also described in detail in subchapters 5.3.1 - 5.3.4.

Table 1. **Hardware configuration defines applicability**

Hardware configuration option (define)	Applicable for:
Number of Touch Button Electrodes	“2pad_hw.h”, “7pad_hw.h”, “6pad_hw.h”
Wake-up EGS Electrode	“7pad_hw.h”, “6pad_hw.h”
Slider Enable Define	“6pad_hw.h”
Electrodes Pin Settings [1]	“2pad_hw.h”, “7pad_hw.h”, “6pad_hw.h”

1. Slider electrodes pin settings applicable only for “6pad_hw.h”

This configuration is done in the hardware configuration header file (“*pad_hw.h”) of the selected reference design – for guidance see Table 2. The user must configure only the appropriate hardware configuration header file according to Table 2.

Table 2. **Hardware configuration header files for reference designs**

Chosen reference design	Hardware configuration header file name
S32K144 2-pad EVB	“2pad_hw.h”
S32K144 7-pad keypad	“7pad_hw.h”
S32K144 6-pad keypad with slider	“6pad_hw.h”

5.3.1. Number of Touch Button Electrodes

Define the NUMBER_OF_TOUCH_ELECTRODES (Figure 20), which represents the number of touch button electrodes.

```
/*******************************************************************************
* Modify: Define number of electrodes
*******************************************************************************/
// Number of touch button electrodes from 1 to 6
#define NUMBER_OF_TOUCH_ELECTRODES   6
```

Figure 20. **Number of touch button electrodes define**

5.3.2. Wake-up EGS Electrode

Modify to enable or disable OPTIONAL_WAKE_UP_ELECTRODE (Figure 21).

```
// YES (WAKE_ELEC_YES) or NO (WAKE_ELEC_NO) optional wake-up electrode
// If YES, an additional electrode defined, assigned the highest number
#define OPTIONAL_WAKE_UP_ELECTRODE   WAKE_ELEC_YES
```

Figure 21. **Optional wake-up EGS electrode define**

What the wake-up EGS electrode is, and how the software functions with and without the EGS electrode is described in chapter 3.3: Software Overview and Flowchart. If EGS is not available on your reference design board, do not enable (define) the EGS electrode.

5.3.3. Slider Enable Define

This define is available only when using the S32K144 6 pad keypad with slider reference design. If applicable, define SLIDER_ENABLE (Figure 22). The software is designed for one slider only, therefore do not modify the number of slider electrodes.

```
// YES (SLIDER_YES) or NO (SLIDER_NO) optional Slider electrodes
#define SLIDER_ENABLE SLIDER_YES
```

Figure 22. **Slider enable define**

5.3.4. Electrodes Pin Settings

All electrodes pin settings are preconfigured for each reference design demo board, thus do not modify unless different (non on-board) electrodes are used.

```
/*******************************************************************************
* Modify: Electrode 0 defines
*******************************************************************************/
#if (NUMBER_OF_ELECTRODES > 0)
    #define ELEC0
    #define ELEC0_ADC              ADC1      // Modify: ADC module
    #define ELEC0_ADC_CHANNEL      1         // Modify: Cext ADC channel
    #define ELEC0_PORT             PORTA     // Modify: Electrode and Cext PORT
    #define ELEC0_GPIO             PTA       // Modify: Electrode and Cext GPIO
    #define ELEC0_ELEC_GPIO_PIN    2         // Modify: Electrode GPIO pin
    #define ELEC0_CEXT_GPIO_PIN    3         // Modify: Cext GPIO pin
    #define ELEC0_PORT_MASK        (1 << ELEC0_ELEC_GPIO_PIN) | (1 << ELEC0_CEXT_GPIO_PIN)
#endif
```

Figure 23. **Example: Touch button electrode 0 pins configuration**

```
/*******************************************************************************
* Modify: HW Slider Electrode 0 defines
*******************************************************************************/
    #define SLIDER_ELEC0_ADC             ADC0       // Modify: ADC module
    #define SLIDER_ELEC0_ADC_CHANNEL     6          // Modify: Cext ADC channel
    #define SLIDER_ELEC0_PORT            PORTB      // Modify: Electrode and Cext PORT
    #define SLIDER_ELEC0_GPIO            PTB        // Modify: Electrode and Cext GPIO
    #define SLIDER_ELEC0_ELEC_GPIO_PIN   4          // Modify: Electrode GPIO pin
    #define SLIDER_ELEC0_CEXT_GPIO_PIN   2          // Modify: Cext GPIO pin
    #define SLIDER_ELEC0_PORT_MASK       (1 << SLIDER_ELEC0_ELEC_GPIO_PIN) | (1 << SLIDER_ELEC0_CEXT_GPIO_PIN)
```

Figure 24. **Example: Slider electrode 0 pins configuration**

Each touch button or wake up EGS electrode requires two GPIO pins (Figure 23), one of them with ADC functionality. Both pins must be on the same port.

Slider electrodes pin settings (Figure 24) are applicable only when using S32K144 6-pad keypad with slider reference design. The slider consists of 2 wedge shaped electrodes. Each slider electrode requires two GPIO pins (the same as any other electrode), one of them with ADC functionality. However, all 4 slider electrodes pins must be on the same port.

5.4. TS Application Configuration

All application configuration options (defines) which can be modified by the user are described in Table 3. Table 3 also shows to which application configuration files (with respect to the chosen reference design) the configuration defines are applicable. All application configuration options are also described in detail in subchapters 5.4.1 - 5.4.8.

Table 3. **Application configuration defines applicability**

Application configuration option (define)	Applicable for:
Electrode Sensing Cycles per Sample[1]	“2pad_app.h”, “7pad_app.h”, “6pad_app.h”
Reaction Time Mode	“2pad_app.h”, “7pad_app.h”, “6pad_app.h”
IIR Filter Coefficients[2]	“2pad_app.h”, “7pad_app.h”, “6pad_app.h”
DC Tracker Filter Factor[3]	“2pad_app.h”, “7pad_app.h”, “6pad_app.h”
Electrode Touch and Release Thresholds[4]	“2pad_app.h”, “7pad_app.h”, “6pad_app.h”
Jittering	“2pad_app.h”, “7pad_app.h”, “6pad_app.h”
Frequency Hopping	“2pad_app.h”, “7pad_app.h”, “6pad_app.h”
Decimation Filter	“2pad_app.h”, “7pad_app.h”, “6pad_app.h”
Virtual EGS Touch Threshold	“2pad_app.h”, “7pad_app.h”, “6pad_app.h”

1. Slider electrodes sensing cycles per sample applicable only for “6pad_app.h”
2. IIR LP filter coefficients for slider electrodes applicable only for “6pad_app.h”
3. Slider DC tracker filter factor settings applicable only for “6pad_app.h”
4. Slider Electrodes thresholds configuration applicable only for “6pad_app.h”

This configuration is done in the application configuration header file (“*pad_app.h”) of the selected reference design – for guidance, see Table 4. The user must configure only the appropriate application configuration header file according to Table 4 .

Table 4. **Application configuration header files for reference designs**

Chosen reference design	Application configuration header file name
S32K144 2-pad EVB	“2pad_app.h”
S32K144 7-pad keypad	“7pad_app.h”
S32K144 6-pad keypad with slider	“6pad_app.h”

5.4.1. Electrode Sensing Cycles per Sample

To understand this configuration setting, go through chapter 3:Touch Sense Software Functionality. Focus on what the electrode sensing cycle is and how final electrode sample is calculated (chapters 3.1 and 3.2). In chapter 3.3 focus on the active and idle sensing mode descriptions.

The set of electrode sensing cycles for calculation of the final electrode sample consists of:

1. Number of sensing pre-cycles – usually 1. This sample is discarded.
2. Number of sensing cycles when idle or active
    a. Idle number of sensing cycles – typically 4 cycles, when wake-up EGS electrode release event is detected (3.3.1) or touch electrodes release event is detected (3.3.2)

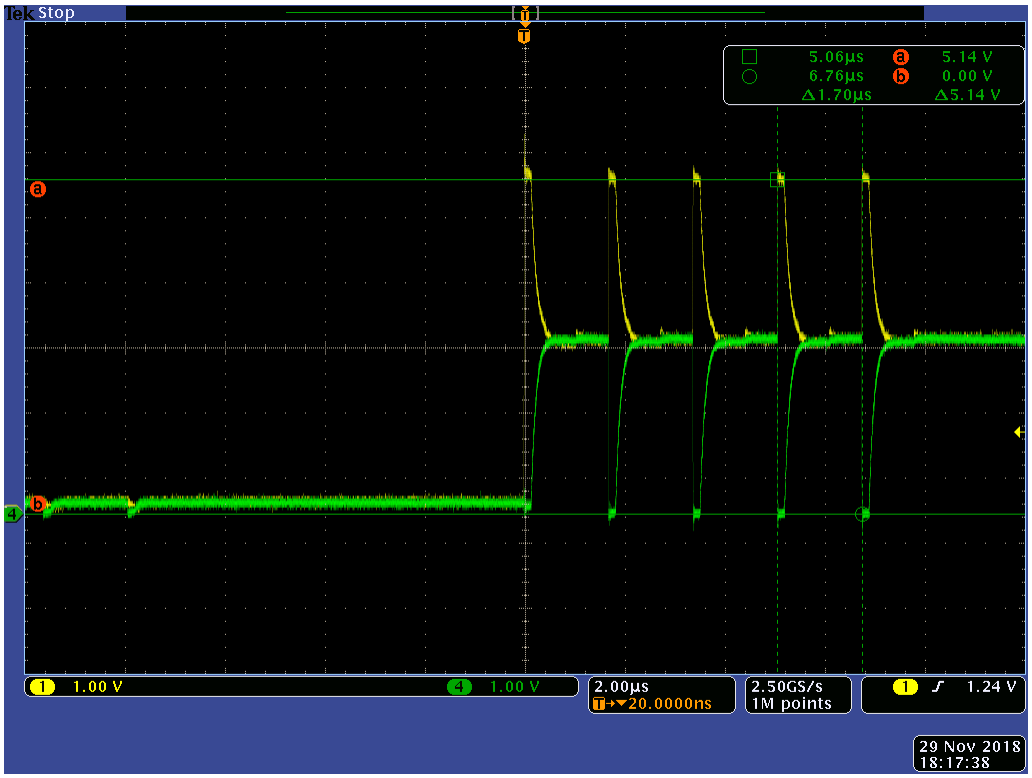

Figure 25. **Idle sensing – 1 sensing pre-cycle and 4 sensing cycles**

    b. Active number of sensing cycles – typically 16 cycles, when an EGS electrode touch event is detected (3.3.1) or a touch electrodes touch event is detected (3.3.2) . It is defined as a multiplication of the number of cycles when idle.

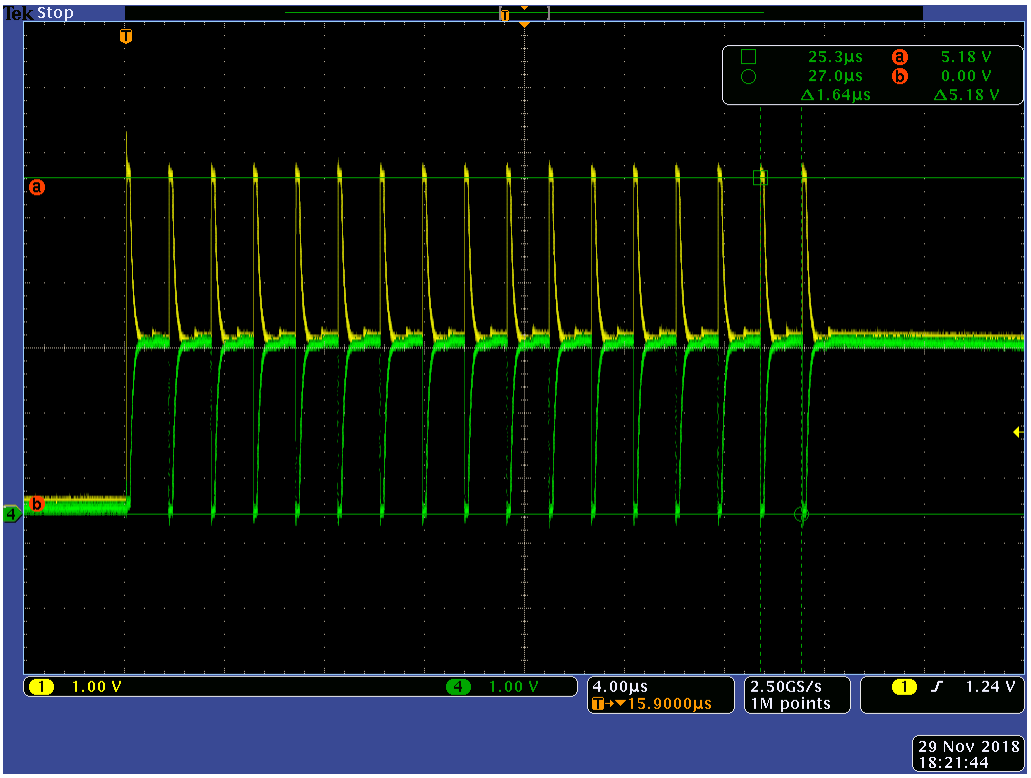

Figure 26. **Active sensing - 1 sensing pre-cycle and 16 sensing cycles**

In the application header file, the user can modify (Figure 27, and if applicable Figure 28):

- NUMBER_OF_ELECTRODE_SENSING_CYCLES_PER_SAMPLE_IDLE
- NUMBER_OF_ELECTRODE_SENSING_CYCLES_PER_SAMPLE_MULTIPLY
- NUMBER_OF_ELECTRODE_SENSING_CYCLES_PER_SAMPLE_SLIDER_IDLE
- NUMBER_OF_ELECTRODE_SENSING_CYCLES_PER_SAMPLE_SLIDER _MULTIPLY

For the idle number of electrode sensing cycles per sample, the user modifies the NUMBER_OF_ELECTRODE_SENSING_CYCLES_PER_SAMPLE_IDLE define.

For the active number of electrode sensing cycles per sample, the user modifies the NUMBER_OF_ELECTRODE_SENSING_CYCLES_PER_SAMPLE_MULTIPLY define, and the following equation applies:

$$number\ of\ electrode\ sensing\ cycles\ active = number\ of\ electrode\ sensing\ cycles\ idle * number\ of\ electrode\ sensing\ cycles\ multiply$$

This way of defining the number of sensing cycles ensures better software functionality when oversampling is chosen as the raw data calculation formula (see section 5.2.2).

```
/*****************************************************************************
* Modify: Number of following electrode pre-cycles per single sample
*****************************************************************************/
// Value from 1 to 2
#define NUMBER_OF_ELECTRODE_SENSING_PRECYCLES_PER_SAMPLE    1

/*****************************************************************************
* Modify: Number of following electrode cycles per single sample
*****************************************************************************/
// IDLE MODE, when EGS/touch buttons are not touched
// Value from 1 to 4  to achieve MCU 70uA max average current consumption
#define NUMBER_OF_ELECTRODE_SENSING_CYCLES_PER_SAMPLE_IDLE   4

// ACTIVE MODE, when EGS/touch buttons are touched
// Define value as the multiply of NUMBER_OF_ELECTRODE_SENSING_CYCLES_PER_SAMPLE_IDLE
// Value must be a power of two
// The NUMBER_OF_ELECTRODE_SENSING_CYCLES_PER_SAMPLE_MULTIPLY * NUMBER_OF_ELECTRODE_SENSING_CYCLES_PER_SAMPLE_IDLE
// Result of multiplication must be below or equal 128
#define NUMBER_OF_ELECTRODE_SENSING_CYCLES_PER_SAMPLE_MULTIPLY  4
```

Figure 27. **Touch button and wake-up EGS electrodes sensing cycles per sample settings**

```
/*****************************************************************************
* Modify: Number of slider electrode cycles per single sample, slider segments
*****************************************************************************/
// Value from 1 to 6 (when slider idle)
#define NUMBER_OF_ELECTRODE_SENSING_CYCLES_PER_SAMPLE_SLIDER_IDLE   4

// ACTIVE MODE, when EGS/touch buttons are touched
// Define value as the multiply of NUMBER_OF_ELECTRODE_SENSING_CYCLES_PER_SAMPLE_SLIDER_IDLE
// Value must be a power of two
// The NUMBER_OF_ELECTRODE_SENSING_CYCLES_PER_SAMPLE_SLIDER_MULTIPLY * NUMBER_OF_ELECTRODE_SENSING_CYCLES_PER_SAMPLE_SLIDER_IDLE
// Result of multiplication must be below or equal 128
#define NUMBER_OF_ELECTRODE_SENSING_CYCLES_PER_SAMPLE_SLIDER_MULTIPLY  8
```

Figure 28. **Slider electrodes sensing cycles per sample settings**

5.4.2. Reaction Time Mode

Response time stands for the electrodes sensing period, by default 30ms. Every 30ms new electrodes raw data are obtained (see chapter 3.1 for a detailed explanation).

Reaction time is the period (delay) between a physical touch of an electrode and the actual application report that the electrode has been touched. Reaction time is dependent on:

- Electrode sensing period
- IIR LP filter coefficients (5.4.3)

Electrodes raw data are smoothed by the IIR LP filter, creating electrode filtered data. In the default configuration, it takes the filtered data 90ms (3 * electrodes sensing period) to cross the touch threshold and thus report the touch event that the electrode is physically touched.

User can choose between (Figure 29):

1. default mode (recommended) MODE_REACTION_TIME_90MS
2. select his own MODE_CUSTOM

Consequently, user can define the electrodes sensing period (ELECTRODES_SENSE_PERIOD). That means LPTMR (Low Power Timer) interrupt period in milliseconds, by default 30ms.

If the electrodes sensing period is changed, IIR filter coefficients must be updated accordingly (5.4.3).

```
/*****************************************************************************
* Modify: Define application mode, select only one option designed for 7 electrode keyboard:
*         MODE_REACTION_TIME_90MS and MODE_CUSTOM
*
* Note: Response time stands for period of electrodes sensing (scan period)
*       Reaction time stands for period between electrode touch and application report "electrode touched"
*****************************************************************************/
// If "#define OPTIONAL_WAKE_UP_ELECTRODE   WAKE_ELEC_YES": MCU response time 30ms, MCU reaction time 90ms, MCU average current consumption 7
// If "#define OPTIONAL_WAKE_UP_ELECTRODE   WAKE_ELEC_NO": MCU response time 30ms, MCU reaction time 90ms, MCU average current consumption 13
#define MODE_REACTION_TIME_90MS

// Custom settings:
// Preset to same operation as in case of MODE_REACTION_TIME_90MS
// If "#define OPTIONAL_WAKE_UP_ELECTRODE   WAKE_ELEC_YES": MCU response time 30ms, MCU reaction time 90ms, MCU average current consumption 7
// If "#define OPTIONAL_WAKE_UP_ELECTRODE   WAKE_ELEC_NO": MCU response time 30ms, MCU reaction time 90ms, MCU average current consumption 13
//#define MODE_CUSTOM

/*****************************************************************************
* Modify: Define electrodes sensing (scanning) cycle in milliseconds [ms].
*         Value from 1 to 65535.
*****************************************************************************/
#if (OPTIONAL_WAKE_UP_ELECTRODE == WAKE_ELEC_YES)
    #ifdef MODE_REACTION_TIME_90MS
        #define ELECTRODES_SENSE_PERIOD   30
    #endif
```

Figure 29. **Reaction time mode and response time**

5.4.3. IIR Filter Coefficients

The IIR LP filter is an algorithm that reduces the noise at the cost of a slower response time. If the electrode LP filter data drop below the touch threshold, an electrode touch event is detected. In Touch Sense application, the purpose of the LP filter is to prevent detection of a false touch.

Figure 30 shows the electrode data subjected to electromagnetic interference from a fluorescent lamp. Electrode raw data represent the noise and the Electrode LP filter data represent the signal. LP filter data peak-to-peak amplitude is reduced approx. 3 times in comparison with the raw data peak-to-peak amplitude.

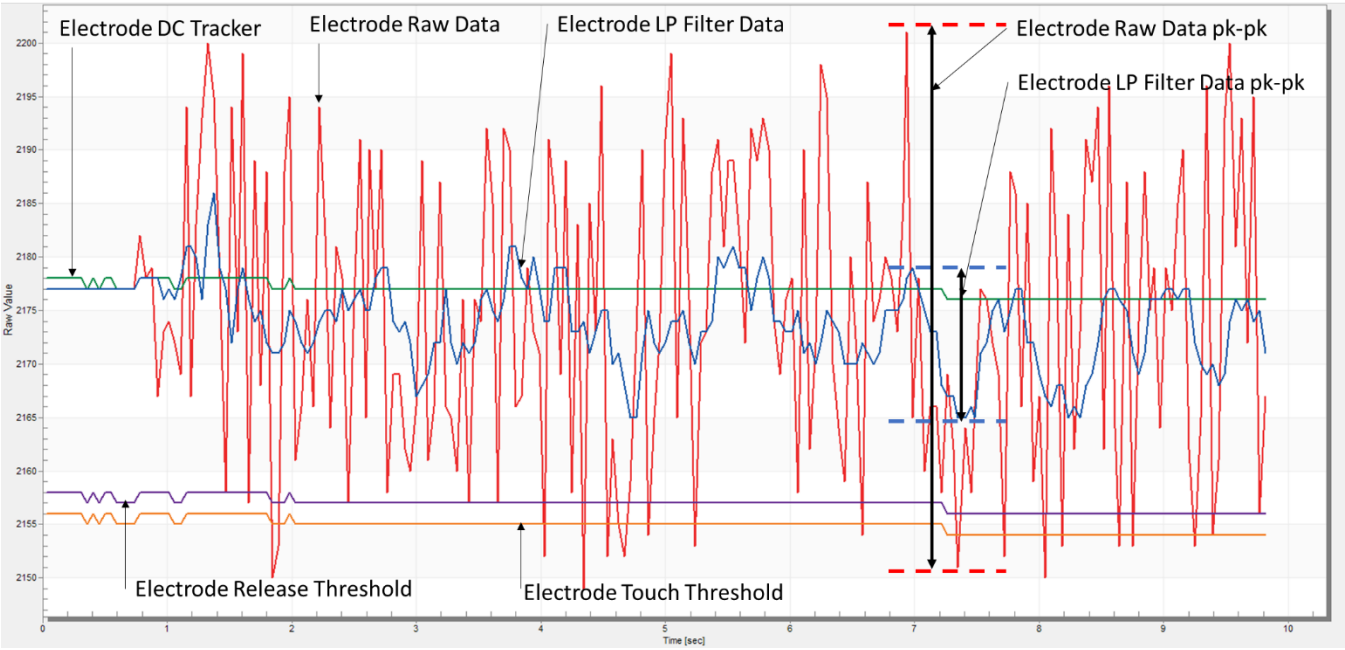

Figure 30. **Electrode data affected by noise**

Current default settings of the first order IIR LP Butterworth filter coefficients are set for electrodes scanning period of 30ms and a cut/off frequency of 1 Hz. For calculation of the filter coefficients, use for example MATLAB.

For proper IIR filter functionality, 15 decimal places for the coefficients are required.

First order IIR LP filter coefficients are linked with application response and reaction times (5.4.2) and they can be modified separately for (Figure 31):

1. Touch buttons (all application header files)
2. Wake-up EGS electrode (“6pad_app.h” and “7pad_app.h”.)
3. Slider electrodes (“6pad_app.h” only)

```
// Define electrodes IIR filter parameters
// ELECTRODES_SENSE_PERIOD   30
#define FILTER_1                    0
// Cutoff frequency in Hertz [Hz]
#define FILTER_1_CUTOFF_FREQ_HZ     1
// Coefficient B0
#define FILTER_1_COEF_B0            0.086364027013762
// Coefficient B1
#define FILTER_1_COEF_B1            0.086364027013762
// Coefficient A0
#define FILTER_1_COEF_A0            -0.827271945972476

// Define EGS IIR filter parameters
#define FILTER_2                    1
#define FILTER_2_CUTOFF_FREQ_HZ     1
#define FILTER_2_COEF_B0            0.086364027013762
#define FILTER_2_COEF_B1            0.086364027013762
#define FILTER_2_COEF_A0            -0.827271945972476

#if SLIDER_ENABLE
    // Slider LPF 1 Hz
    #define SLIDER_FILTER_1             2
    #define FILTER_3                    SLIDER_FILTER_1
    #define FILTER_3_CUTOFF_FREQ_HZ     1
    #define FILTER_3_COEF_B0            0.086364027013762
    #define FILTER_3_COEF_B1            0.086364027013762
    #define FILTER_3_COEF_A0            -0.827271945972476
#endif
```

Figure 31. **IIR filter coefficients**

5.4.4. DC Tracker Filter Factor

The DC tracker is a very slow filter which tracks the electrodes baseline to react on long term environment changes. The DC tracker filter response can be modified.

When oversampling is selected as the raw data calculation formula (5.2.2), the software is always using both idle and active factors, no matter any other configurations.

When averaging is selected as the raw data calculation formula (5.2.2), the software is using both idle and active factors only if the EGS electrode is enabled (5.3.2) and Frequency Hopping (5.4.7) or Decimation Filter (5.4.8) is enabled, else it is using just the active factor.

The filter factor value must be from 1 to 8, where 8 represents the slowest DC tracker filter response.

In the application header file, the user can modify (Figure 32, and if applicable Figure 33):

- ELEC_DCTRACKER_FILTER_FACTOR_IDLE
- ELEC_DCTRACKER_FILTER_FACTOR_ACTIVE
- SLIDER_ELEC_DCTRACKER_FILTER_FACTOR_IDLE
- SLIDER_ELEC_DCTRACKER_FILTER_FACTOR_ACTIVE

```
// DC tracker response, when in Idle mode
// Value 5 equals 1s at 30ms sampling period
#define ELEC_DCTRACKER_FILTER_FACTOR_IDLE     5
// DC tracker response, when in Active mode
// Value 5 equals 1s at 30ms sampling period
#define ELEC_DCTRACKER_FILTER_FACTOR_ACTIVE   (5 + DF_DCTRACKER_FILTER_FACTOR + FH_DCTRACKER_FILTER_FACTOR)
```

Figure 32. **Touch button and Wake Up electrode DC tracker filter factor**

```
// DC tracker response, when in Idle mode
// Value 5 equals 0.5s at 30ms sampling period
#define SLIDER_ELEC_DCTRACKER_FILTER_FACTOR_IDLE   5
// DC tracker response, when in Active mode
// Value 5 equals 0.5s at 30ms sampling period
#define SLIDER_ELEC_DCTRACKER_FILTER_FACTOR_ACTIVE   (5 + DF_DCTRACKER_FILTER_FACTOR + FH_DCTRACKER_FILTER_FACTOR)
```

Figure 33. **Slider electrodes DC tracker filter factor**

5.4.5. Electrode Touch and Release Thresholds

Every touch button and wake-up EGS electrode has touch and release thresholds. The thresholds influence electrode sensitivity and are calculated relative to the electrode baseline (Figure 34). The thresholds can be changed by modifying the ELEC_TOUCH_THRESHOLD_DELTA and ELEC_RELEASE_THRESHOLD_DELTA values (Figure 35). When the electrode LP Filter data signal drops below the touch threshold, an electrode touch event is detected.

If the delta number is decreased, the threshold is closer to the electrode DC Tracker baseline, and sensitivity increases. Also, the application is more susceptible to electromagnetic interference (EMI) and false touches.

However, if the delta number is increased, the application is less susceptible to EMI and false touches, but also less sensitive in general.

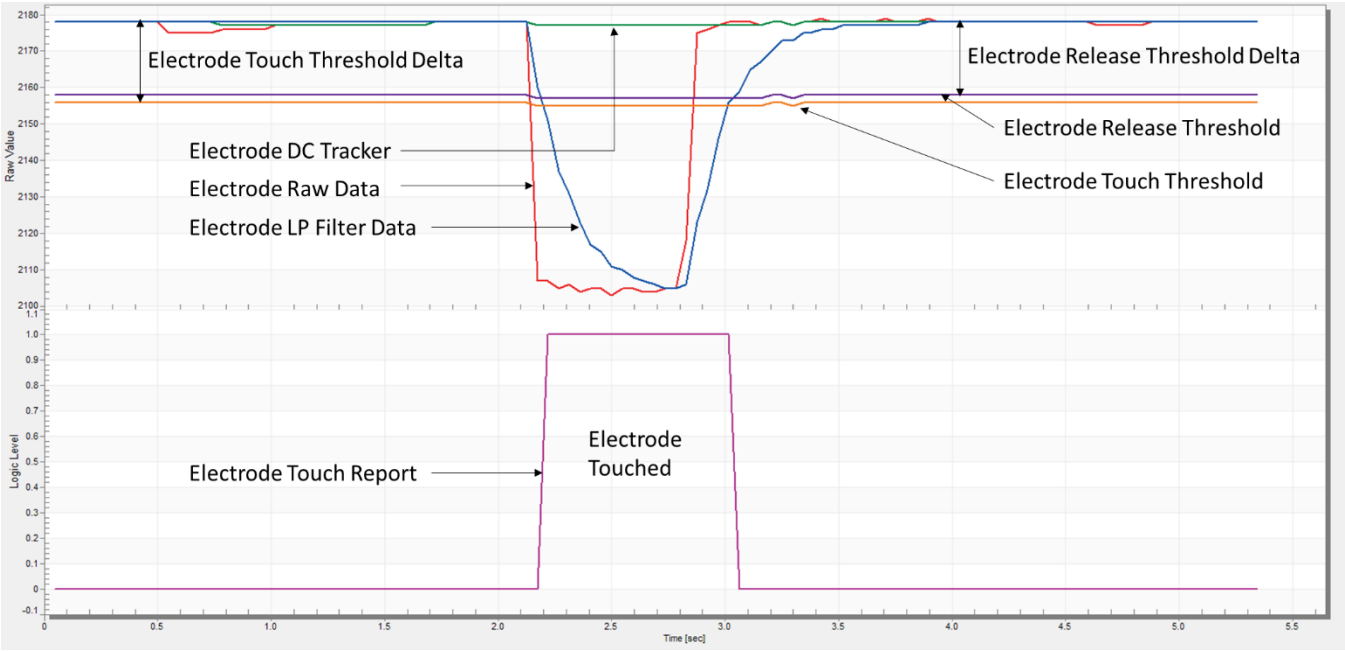

Figure 34. **Electrode Touch and Release Thresholds**

```
// Electrode touch threshold relative to DC tracker value
#define ELEC_TOUCH_THRESHOLD_DELTA       22
// Electrode release threshold relative to DC tracker value
#define ELEC RELEASE THRESHOLD DELTA     20
```

Figure 35. **Touch button electrodes common touch and release threshold delta**

Slider electrode thresholds are more complicated than touch button electrodes. The slider application configuration and tuning is described in detail in the Slider Application Quick Start Guide.

5.4.6. Jittering

To avoid hitting any harmonics when periodically scanning the electrodes, the user can change (jitter) the sample rate by a very small amount each time a sample is taken by using a simple “jittering” technique.

In Touch Sense software:

- Jittering function creates a delay loop, which decrements a random value calculated by masking the lower 2 to 8 bits of the electrode raw data. To prolong the delay loop, increase the NUMBER_OF_JITTERING_BITS and vice versa.
- Jittering is enabled/disabled by modifying the JITTERING define (Figure 36).
- Jittering of the sample rate can be allowed for two scenarios:
    - JITTERING_OPTION 1 – jitters the sample rate only at the beginning of the timer IRQ handler
    - JITTERING_OPTION 2 – jitters in between each electrode sensing cycle (Figure 37)

```
/*****************************************************************************
* Modify: Jittering ON/OFF and jittering bits
*****************************************************************************/
// YES (JITTERING_ON) or NO (JITTERING_OFF) optional jittering sample rate control
#define JITTERING JITTERING_ON

    #if JITTERING
        // Modify: Define jittering option: 1 - jitter at the beginning of IRQ handler, 2 - jitter in b
        #define JITTERING_OPTION 2

        // Modify: Value of 2 to 8
        #define NUMBER_OF_JITTERING_BITS 3

        // Do not modify!: the jittering_mask_macro
        #define JITTERING_MASK ((1 << NUMBER_OF_JITTERING_BITS)-1)
    #endif
```

Figure 36. **Jittering defines**

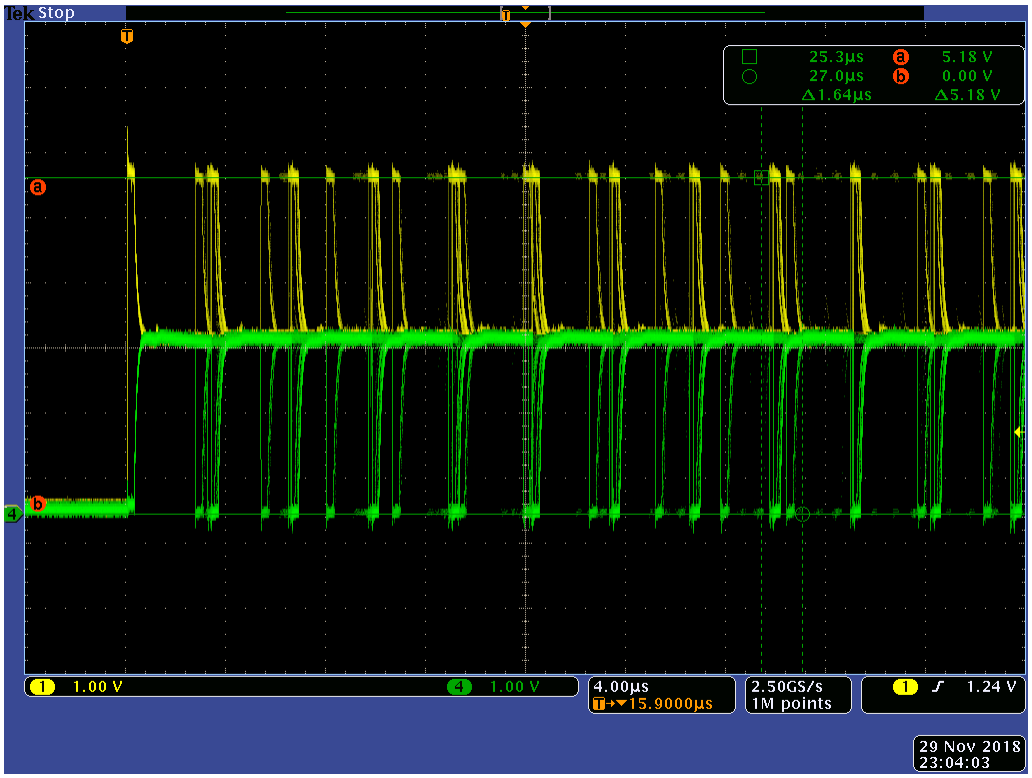

Figure 37. **Jittering example: jittering option 2 – in between samples**

5.4.7. Frequency Hopping

Frequency hopping enables the user to gather redundant electrode raw data by scanning on a secondary electrode sensing period/frequency. This feature extends the TS application noise immunity, as the noise would have to exist at both electrode sensing frequencies or their harmonics to potentially cause a false touch detection.

In Touch Sense software:

- The frequency hopping function directs the software to perform another scan of all electrodes on the second scanning period/frequency. Basically, after all the electrodes are sensed on core 30ms scanning period, the LPIT (low power interrupt timer) is enabled and after 330 µs (by default) it triggers another (second) scan of all electrodes. A touch event must be confirmed on both periods/frequencies to proceed to the touch qualification.
- Frequency hopping is enabled/disabled by modifying the FREQUENCY_HOPPING define (Figure 38).
- The software is written only for two hopping frequencies. However, the user can modify the LPIT period (ELECTRODES_SENSE_PERIOD_FH) which triggers the electrodes sensing on the second frequency.
- Also, with more frequent electrodes sensing, the DC tracker filter is updated more frequently, therefore the user can define an additional frequency hopping DC tracker filter factor (FH_DCTRACKER_FILTER_FACTOR), which contributes to the total active DC tracker filter factor (5.4.4).

```
// YES (FREQUENCY_HOPPING_ON) or NO (FREQUENCY_HOPPING_OFF) optional frequency hopping
#define FREQUENCY_HOPPING FREQUENCY_HOPPING_ON

    #if FREQUENCY_HOPPING
        // Do not modify!: 1 core scanning period/frequency (30ms) + 1 extra scanning period/frequency
        // SW written for maximum of 2 scanning periods/frequencies only
        #define NUMBER_OF_HOPPING_FREQUENCIES 2

        // Modify: Extra scanning period/frequency 330us after the core period
        // (LPIT fed by 48 MHz clock => Value 48 000 equals delay 1 ms)
        #define ELECTRODES_SENSE_PERIOD_FH   15840

        // Modify: Frequency hopping filter factor addition for DCTracker
        #define FH_DCTRACKER_FILTER_FACTOR 1
```

Figure 38. **Frequency hopping defines**

Figure 39. **Frequency hopping – first and second scanning period**

5.4.8. Decimation Filter

The decimation filter is a slew rate limiter designed to reject impulse noise (spikes). The decimation filter typically either decrements or increments data by one, based on whether they are higher or lower than the currently scanned electrode raw data. This technique requires a faster sample rate (electrodes sensing period) to keep the same application response time (5.4.2).

In Touch Sense software:

- When the wake-up EGS electrode touch event is detected, (3.3.1) the electrodes sensing period (LPTMR period) is changed from 30ms to 3ms (ELECTRODES_SENSE_PERIOD_DF). Response and reaction times (5.4.2) are improved since the electrodes are sensed every 3ms (Figure 41).
- If the electrode raw data are smaller than the decimation filter data, the decimation filter data value decrements by a preset DECIMATION_STEP and vice versa. Decimation filter data are fed into the IIR LP filter instead of the electrode raw data.
- Decimation filtering is enabled/disabled by modifying the DECIMATION_FILTER define (Figure 40).
- Also, with more frequent electrodes sensing, the DC tracker filter is updated more frequently, therefore the user can define an additional decimation filter DC filter factor (DF_DCTRACKER_FILTER_FACTOR), which contributes to total active DC tracker filter factor (5.4.4).

```
// YES (DECIMATION_FILTER_ON) or NO (DECIMATION_FILTER_OFF) optional decimation filter
#define DECIMATION_FILTER DECIMATION_FILTER_ON

    #if DECIMATION_FILTER
        // Modify: LPTMR new wake-up period - 3 ms
        // Value between 3 and 30
        #define ELECTRODES_SENSE_PERIOD_DF 3

        // Modify: Decimation filter filter factor addition for DCTracker
        #define DF_DCTRACKER_FILTER_FACTOR (3 - FH_DCTRACKER_FILTER_FACTOR)

        #if (TS_METHOD == OVERSAMPLING)
            // Modify: Decimation filter step
            #define DECIMATION_STEP 10
        #elif (TS_METHOD == AVERAGING)
            // Modify: Decimation filter step
            #define DECIMATION_STEP 1
```

Figure 40. **Decimation filter defines**

Figure 41. **Decimation filter sensing period**

5.4.9. Virtual EGS Touch Threshold

When the wake-up EGS electrode is not present on the board or is disabled, the transition from idle to active sensing mode is done by the “Virtual EGS”. (see 3.3.2)

The virtual EGS is basically another touch threshold implemented for every touch electrode. The virtual EGS touch threshold is typically very close to the electrode baseline.

The threshold can be changed by modifying the VIRTUAL_EGS_TOUCH_THRESHOLD_DELTA value. (Figure 42) If the delta number is decreased, the threshold is closer to the electrode DC Tracker baseline, and the software is switched into the active sensing mode sooner.

```
/*****************************************************************************
* Modify: Virtual EGS Electrode touch threshold (for samples switching)
*****************************************************************************/
#ifndef WAKE_UP_ELECTRODE
    #if(TS_RAW_DATA_CALCULATION == AVERAGING)
        // Virtual EGS touch threshold relative to DC tracker value
        #define VIRTUAL_EGS_TOUCH_THRESHOLD_DELTA 3
    #elif (TS_RAW_DATA_CALCULATION == OVERSAMPLING)
        // Virtual EGS touch threshold relative to DC tracker value
        #define VIRTUAL_EGS_TOUCH_THRESHOLD_DELTA 15
    #else
        #error Please select valid raw data calculation method in ts_cfg_general.h
    #endif
#endif
```

Figure 42. **Virtual EGS touch threshold**

5.5. Common Application Settings

This configuration is done in the “ts_cfg.h” file. All common application configuration defines which can be modified by the user are described in subchapters 5.5.1 - 5.5.3.

5.5.1. Low power mode

In chapter 3.3: Software Overview and Flowcharts a transition of the Touch Sense software to low power mode is shown. The low power mode of the MCU ensures the required 70µA average MCU current consumption.

The average MCU current consumption in low power mode is proportional to:

1. Electrodes sensing period – by default, each 30ms the MCU is awoken and scans the electrodes (see chapter 3: Touch Sense Software Functionality for details). If the electrodes sensing period value is decreased, the MCU is awoken more often and therefore the current consumption rises.
2. Number of sensing cycles per sample – with each electrodes sensing period, a final electrode sample is calculated from a set of sensing cycles (see chapter 3.1: Electrode Sensing Method for details). If number the of sensing cycles per sample is increased, the MCU performs more sensing cycles per electrode and therefore the current consumption rises.

Disable LOW_POWER_MODE to enable the application debug and FreeMASTER visualization. Enable LOW_POWER_MODE for low power MCU performance (Figure 43).

```
/*****************************************************************************
* Modify: Low power mode enable
*         If low power mode is enabled (LPM_ENABLE), the application debug and
*         FreeMASTER data visualization are disabled.
*         If low power mode is disabled (LPM_DISABLE), the application debug and
*         FreeMASTER data visualization are enabled.
*****************************************************************************/
#define LOW_POWER_MODE    LPM_DISABLE
```

Figure 43. **Low power mode**

In Table 5, a comparison of reference designs in terms of the measured average MCU current consumption is shown. All measured values are for the default software configurations for all the reference designs. For 7-pad keypad and 6-pad keypad with slider, a difference of the current consumption is shown if the wake-up EGS electrode is enabled/disabled.

Table 5. **MCU average current consumption**

Reference design	Wake-up EGS?	MCU current consumption [µA]
7pad keypad	ON	65
7pad keypad	OFF	126
6pad keypad with slider	ON	68
6pad keypad with slider	OFF	170
2pad EVB	OFF	62

In RUN mode, when touch electrodes are touched (see 3.3), the MCU typically has a 22mA current consumption.

5.5.2. Debug pins

The user can also debug the Touch Sense application using debug pins.

Figure 45 shows two waveforms in time when touch electrodes are touched. The waveforms represent two types of processes which can be tracked in time using the debug pins:

1. Electrodes sensing procedure during electrodes sensing period (Figure 45, yellow waveform)
    - For more information, see chapters 3.1, 3.2
    - Uncomment DEBUG_ELECTRODE_SENSE to enable debugging (Figure 44)
2. Whole Touch Sense application algorithm execution time (Figure 45, green waveform)
    - For more information see chapter 3.3
    - Uncomment DEBUG_ ALGORITHM to enable debugging (Figure 44)

```
/*****************************************************************************
* Modify: If needed, configure debug pins. Uncomment required option.
*****************************************************************************/
// Debug electrode sensing cycle
//#define DEBUG_ELECTRODE_SENSE
// Debug application algorithm
//#define DEBUG_ALGORITHM

// Defined?
#ifdef DEBUG_ELECTRODE_SENSE
    // Assign PORT
    #define DES_PORT  PORTE
    // Assign GPIO
    #define DES_GPIO  PTE
    // Assign pin
    #define DES_PIN   10

#endif

// Defined?
#ifdef DEBUG_ALGORITHM
    // Assign PORT
    #define DA_PORT  PORTE
    // Assign GPIO
    #define DA_GPIO  PTE
    // Assign pin
    #define DA_PIN   11
#endif
```

Figure 44. **Debug pins**

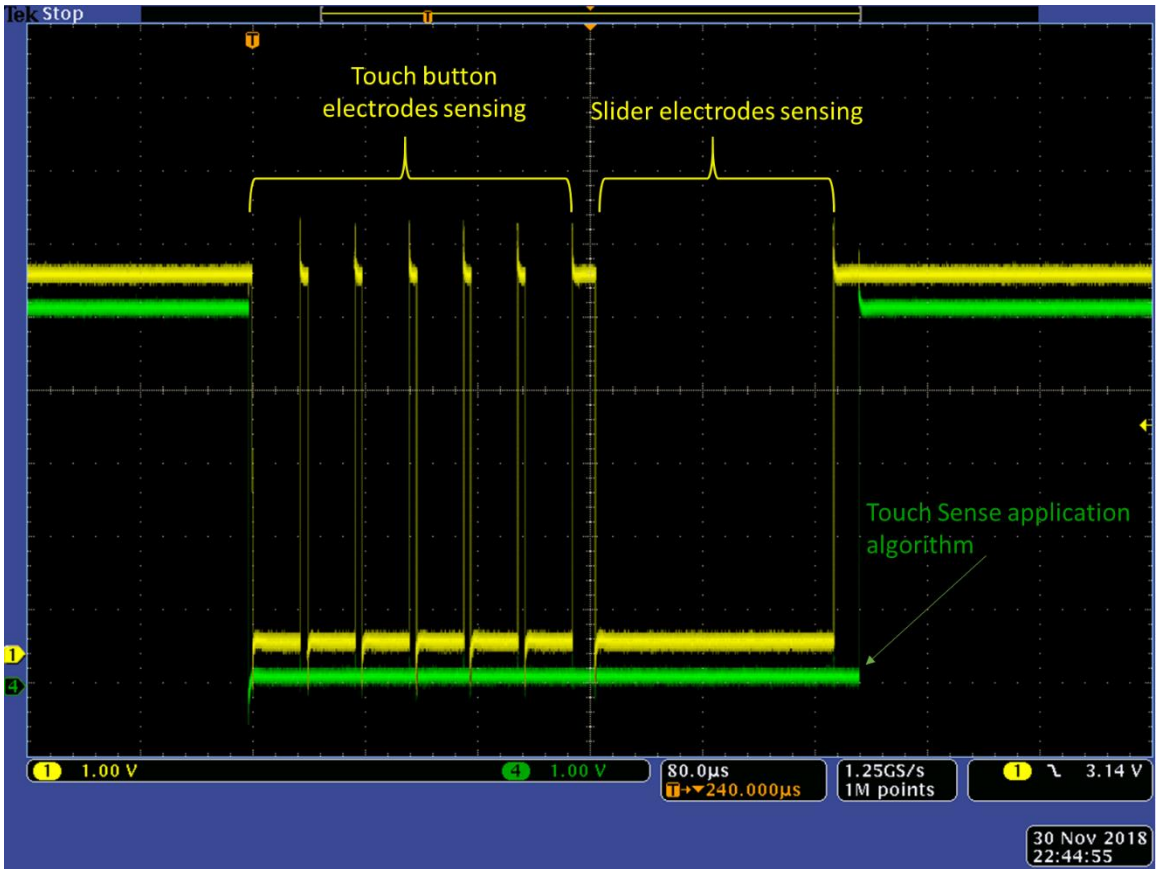

Figure 45. **Example of time waveforms of debug pins signals**

5.5.3. Assembly optimization

The Touch Sense software runs with -O3 compiler optimization for best performance. If any other optimization level is required, then TS_ASM_OPTIMIZE must be set to 1 (Figure 46). See Table 6 for the detailed relation between the chosen optimization and the TS_ASM_OPTIMIZE define.

Table 6. **Optimization vs TS_ASM_OPTIMIZE define**

Optimization	TS_ASM_OPTIMIZE
-O3	0
-O2	1
-O1	1
-O0	1

```
/*****************************************************************************
* Modify: Assembly optimization to avoid dependency on -O3 compilation optimization (1-ON, 0-OFF)
*         Modify only if lower than -O3 optimization needed
*****************************************************************************/
#define TS_ASM_OPTIMIZE     0
```

Figure 46. **Assembly optimization**

6. Building and Debugging the Application

After all the necessary configurations are done, follow these steps:

1. Click on the hammer icon in the toolbar to Build the project
2. Connect the PC and S32K144 EVB board with the USB cable
3. Download the code to the S32K144 EVB board MCU using the S32K144_TS_40 debug configuration and on-board OPENSDA interface (Figure 47): Right-click the imported project in Project explorer section → Debug As → Debug Configurations
4. In the debug session, resume, pause or terminate at your will

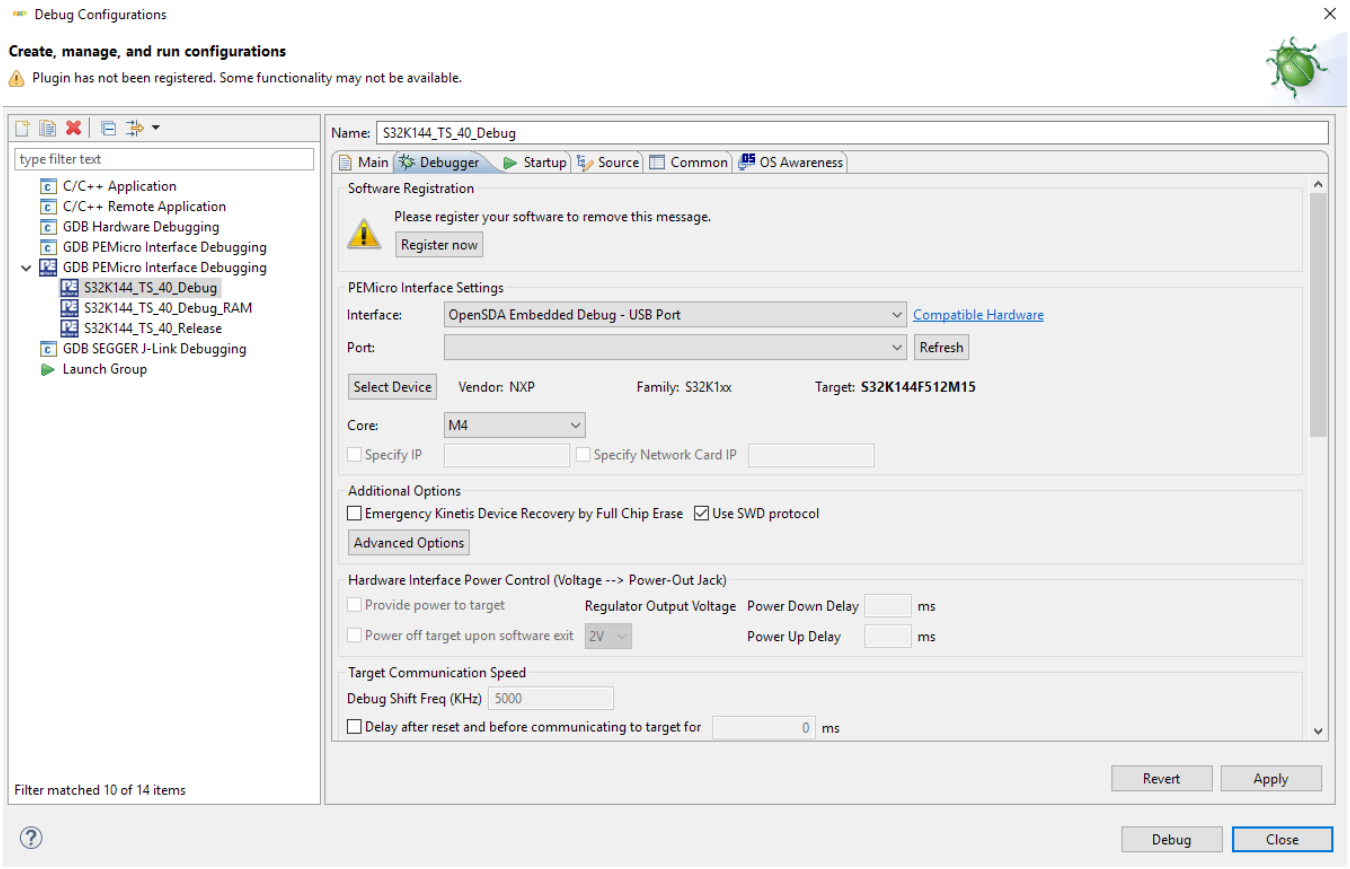

Figure 47. **Debug configuration**

6.1. Real Time Debugging with FreeMASTER

1. Open the correct (with respect to the chosen reference design) FreeMASTER project located in project folder

Figure 48. **FreeMASTER projects in project folder**

   If frequency hopping or the decimation filter is used, open the project with the *FH_DF ending.

2. Establish communication
   a. If Frequency hopping or decimation not used: Project → Options → Comm bookmark → RS232 → Port with OpenSDA, speed 9600
   b. If Frequency hopping or decimation used: Project → Options → Comm bookmark → Plug-in Module → FreeMASTER BDM Communication Plug-in → Click configure → Driver: P&E CortexM → Select Connection: OpenSDA

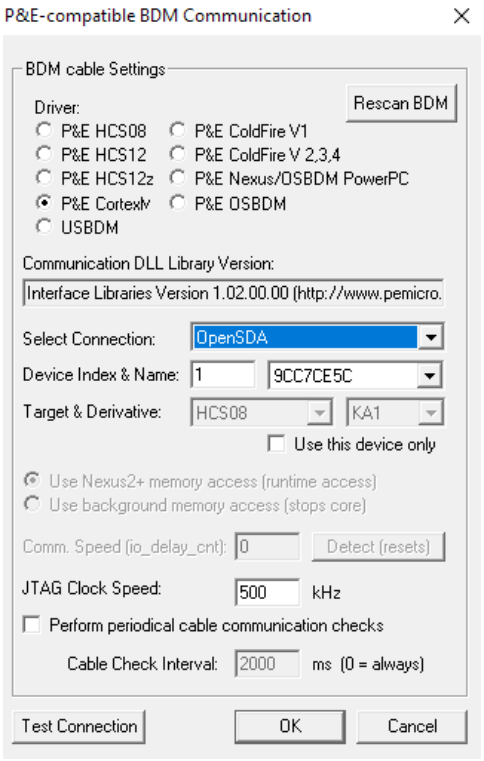

Figure 49. **FreeMASTER Communication settings when Frequency hopping or Decimation filter used**

3. Start communication – note that when communication is set via BDM, all running debug sessions (e.g. in S32 DS) must be terminated.

Figure 50. **FreeMASTER Start-Stop button**

7. Useful Links

Within the NXP organization, these materials are available:

1. Touch Sense solution video
    - GPIS Touch Sense solution video
    - Motivation behind touch sensing in general
    - GPIS electrodes sensing method
    - Opportunities
    - Reference designs overview
    - How to sell
2. GPIS Touch Sense Solution – General Presentations
    - GPIS electrodes sensing method
    - Reference designs overview
    - Reference designs performance
3. Slider Application Quick Start Guide
    - Slider application performance and concept of work
    - Slider electrodes touch detection
    - Slider finger position determination
    - Slider application configuration – threshold settings
    - Slider application tuning
4. FAQ
    - Document with answers on the most frequently asked questions

RoHS

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number: AN_CSSWUG
Rev. 1.1
05/2019

arm