UM1606
User manual

STMTouch driver user manual

Introduction

The document covers the following touch sensing library product codes:

- STM8L-TOUCH-LIB
- STM8TL-TOUCH-LIB
- 32L1-TOUCH-LIB
- 32F0-TOUCH-LIB
- 32F3-TOUCH-LIB

The STMTouch driver includes:

- A complete register address mapping with all bits, bitfields and registers declared in C. This avoids a cumbersome task and more importantly, it brings the benefits of a bug free reference mapping file, speeding up the early project phase.
- A collection of routines and data structures covering all functions to manage the touch sensing technology.

The STMTouch driver source code is developed using the ANSI-C standard. It is fully documented and is MISRA®-C 2004 compliant. Writing the whole library in 'Strict ANSI-C' makes it independent from the development tools. Only the start-up files depend on the development tools.

Run-time failure detection is also implemented by checking the input values for all library functions. Such dynamic checking contributes towards enhancing the robustness of the firmware. Run-time detection is suitable for user application development and debugging. It adds an overhead which can be removed from the final application code to minimize code size and execution speed. For more details refer to *Section 1.4: Run-time checking*.

Since the STMTouch driver is generic and covers many functionalities and devices, the size and/or execution speed of the application code may not be optimized. For many applications, the STMTouch driver may be used as is. However, for applications having tough constraints in terms of code size and/or execution speed, the STMTouch driver may need to be fine tuned.

Note: Additional information on the STMTouch driver functions, variables and parameters can be found in the CHM user manual present in the STMTouch Library installation folder.

Contents

List of tables

List of figures

1 Coding rules and conventions

1.1 Glossary

The table below summarizes all the terms and acronyms used inside this user manual.

Table 1. Terms and Acronyms

Name	Definition
Bank	A group of channels acquired simultaneously
Channel	Elementary acquisition item
Cs	Charge-Transfer sampling capacitor or capacitance
Ct	Equivalent touch capacitance
CT	Charge-Transfer acquisition principle
Cx	Equivalent sensor capacitance
Delta	Difference between the Measure and the Reference (for PXS acquisition)
DTO	Detection Time Out
DXS	Detection Exclusion System
ECS	Environment Change System
Linear sensor	Multi-channels sensor with the electrodes positioned in a linear way
LinRot sensor	A linear or rotary touch sensor
Measure or Meas	Current signal measured on a channel
PXS	ProxSense acquisition peripheral used in STM8TL5x devices
Reference or Ref	Measure of reference initialized during calibration and then regularly updated by the ECS
Rotary	Multi-channels sensor with the electrodes positioned in a circular way
Rs	ESD protection serial resistor
Sensor or Object	Any touch sensor (touchkey, linear, rotary,...)
Timer acquisition mode	Acquisition using two timers and PWM signals. Also called hardware acquisition mode. Available on STM32L1xx devices
Touchkey or TKey sensor	Single channel sensor
Zone	An ordered set of banks

1.2 Naming conventions

The following naming conventions are used in the STMTouch driver source files:

- Source and header files are in lower-case and preceded by 'tsl' or 'tsl_'.
- The microcontroller family is added at the end of the file name if needed.
- Functions, globals, typedefs and defines are preceded by 'TSL'.
- Constants are written in upper case and preceded by 'TSLPRM_'.
- Constants used in one file are defined within this file only.
- Constants used in more than one file are defined in a header file.
- Typedef names are suffixed with '_T'.
- Enum typedefs are suffixed with '_enum_T'.
- Functions are preceded 'TSL_[module]_[function]'.
  - [module]: abbreviation of the file (acq, tim, dxs, etc...)
  - [function]: the first letter in each word is in upper case

1.3 Coding rules

This section describes the coding rules used in the STMTouch driver source files.

1.3.1 General

- Source code complies with ANSI C standard.
- No warning after compilation. Any warnings that cannot be eliminated are commented in the source code.
- ANSI standard data types are used and defined in the ANSI C header file <stdint.h>.
- No blocking code is present and all required waiting loops (polling loops) are controlled by a timeout.

1.3.2 Variable types

Specific variable types are already defined with a fixed type and size.

- The types that are used by all modules are defined in the **tsl_types.h** file.
- Other variable types are defined in their corresponding module header file.

1.3.3 Peripheral registers

The peripheral registers are accessed using the pointers described in the standard peripherals library mapping file.

1.4 Run-time checking

The STMTouch driver implements run-time failure detection by checking the functions input parameters. The run-time checking is achieved using the **assert_param** macro defined in the standard peripherals library configuration file. It is enabled when the preprocessor constant **USE_FULL_ASSERT** is defined.

Because of the overhead it introduces, it is recommended to use run-time checking during application code development and debugging, and to remove it from the final application to improve code size and speed.

However if you want to keep this functionality in your final application, reuse the assert_param macro defined in the standard peripherals library to test the parameter values before calling the STMTouch driver functions.

Please see the standard peripherals library user manual for more informations.

1.5 MISRA-C 2004 compliance

1.5.1 Generalities

The C programming language is growing in importance for embedded systems. However, when it comes to developing code for safety-critical applications, this language has many drawbacks. There are several unspecified, implementation-defined, and undefined aspects of the C language that make it unsuited for developing safety-critical systems.

The motor industry software reliability association's guidelines for the use of the C language in critical systems (MISRA-C 2004 [1]) describe a subset of the C language well suited for developing safety-critical systems.

The STMTouch driver has been developed to be MISRA-C 2004 compliant.

The following section describes how the STMTouch driver complies with MISRA-C 2004 (as described in section 4.4 Claiming compliance of the standard [1]):

- A compliance matrix has been completed which shows how compliance has been enforced.
- The whole STMTouch driver source code is compliant with MISRA-C 2004 rules.
- Deviations are documented. A list of all instances of rules not being followed is being maintained, and for each instance there is an appropriately signed-off deviation.
- All the issues listed in section 4.2 “The programming language and coding context of the standard” [1], that need to be checked during the firmware development phase, have been addressed during the development of the STMTouch driver and appropriate measures have been taken.

1.5.2 Compliance matrix

The compliance of the STMTouch driver with MISRA-C 2004 has been checked in two ways:

- using PC-lint tool for C/C++ (NT) vers. 8.00v, copyright gimpel software 1985-2006
- performing regular code reviews.

The following table lists the MISRA-C 2004 rules that are frequently violated in the code:

Table 2. MISRA-C 2004 rules not followed

MISRA-C 2004 rule number	Required/ advisory	Summary	Reason of deviance
1.1 1.2	Required	All code shall conform to ISO 9899:1990 standard C, with no extensions permitted.	Compilers extensions are enabled. Comments starting with “//” symbol for code readability.
5.4	Required	A tag name shall be a unique identifier.	Due to the usage of objects methods.
8.1	Required	No prototype seen. Functions shall always have prototype declarations and the prototype shall be visible at both the function definition.	This rule is violated as there is no functions prototypes for the objects methods.
10.1 10.2	Required	The value of an expression of integer/floating type shall not be implicitly converted to a different underlying type.	Code complexity
10.3	Required	The value of a complex expression of integer type may only be cast to a type that is narrower and of the same signedness as the underlying type of the expression.	Code complexity
10.5	Required	If the bitwise operators are applied to an operand of underlying type unsigned char or unsigned short, the result shall be immediately cast to the underlying type of the operand.	Use shift on signed quantity for the linear/rotary position
11.3	Advisory	A cast should not be performed between a pointer type and an integral type.	Needed when addressing memory mapped registers.
12.7	Required	Bitwise operators shall not be applied to operands whose underlying type is signed.	Shift of signed value needed
14.3	Required	Before preprocessing, a null statement shall only occur on a line by itself.	Usage of macros to simplify the code
14.5	Required	The continue statement shall not be used.	Used to optimize the code speed execution.
19.11	Required	All macro identifiers in preprocessor directives shall be defined before use, except in ifdef and ifndef preprocessor directives and the defined() operator.	All parameters are checked in the check_config files

2 STMTouch driver

2.1 Supported devices and development tools

2.1.1 Supported devices

This STMTouch driver version supports the following devices and acquisition modes:

- Support of **STM8L1xx** devices
  - Surface charge-transfer acquisition principle managed by:
  - Two timers + routing interface (hardware acquisition mode, supported only by STM8L15x low-density devices)
  - GPIOs + routing interface (software acquisition mode, supported by all STM8L devices)
  - Up to 6 channels with up to 2 channels acquired simultaneously for the STM8L101 devices (see *Table 11* for more details)
  - Up to 20 channels with up to 8 channels acquired simultaneously for the STM8L15x/16x devices (see *Table 12* and *Table 13* for more details)
- **STM8TL5x** devices using the embedded ProxSense™ patented acquisition technology.
  - Projected ProxSense™ acquisition principle
  - Up to 300 channels
  - Up to 10 channels can be acquired simultaneously (see *Table 18* for more details)
  - Integrated sampling capacitor
  - Electrode parasitic capacitance compensation (EPCC)
  - On-chip integrated voltage regulator
- Support of **STM32L1xx** devices
  - Surface charge-transfer acquisition principle managed by:
  - Two timers + routing interface (hardware acquisition mode). This mode is not supported on Medium-density devices.
  - GPIOs + routing interface (software acquisition mode). This mode is supported by all devices.
  - Up to 34 channels
  - Up to 11 channels can be acquired simultaneously (see *Table 37*, *Table 38*, *Table 39*, *Table 40* and *Table 41* for more details)
- **STM32F0xx** and **STM32F3xx** devices using the embedded touch sensing controller IP (TSC).
  - Surface charge-transfer acquisition principle managed by the touch sensing controller
  - Up to 24 channels
  - Up to 8 channels can be acquired simultaneously (see *Table 22*, *Table 23*, *Table 27* and *Table 28* for more details)
  - Spread spectrum feature
  - Programmable charge transfer frequency and max count value

2.1.2 Development tools

The STM8 and STM32 microcontrollers are supported by a full range of development solutions from lead suppliers that deliver start-to-finish control of application development from a single integrated development environment. The STM8 are based on proprietary code and STM32 microcontrollers are based on Arm®(a) code.

The STMTouch driver has been developed with the following toolchains and compilers:

- **STVD** (STMicroelectronics) + Raisonance and Cosmic compilers
- **EWSTM8** and **EWArm** (IAR™)
- **MDK-Arm** (Keil®)
- **Tasking** (Altium®)
- **TrueSTUDIO®**-C (Atollic®)
- **Ride7** / **RKit-Arm** (Raisonance)

For more details about the compilers versions used, please see the STMTouch driver release note (present in the STMTouch Library installation folder).

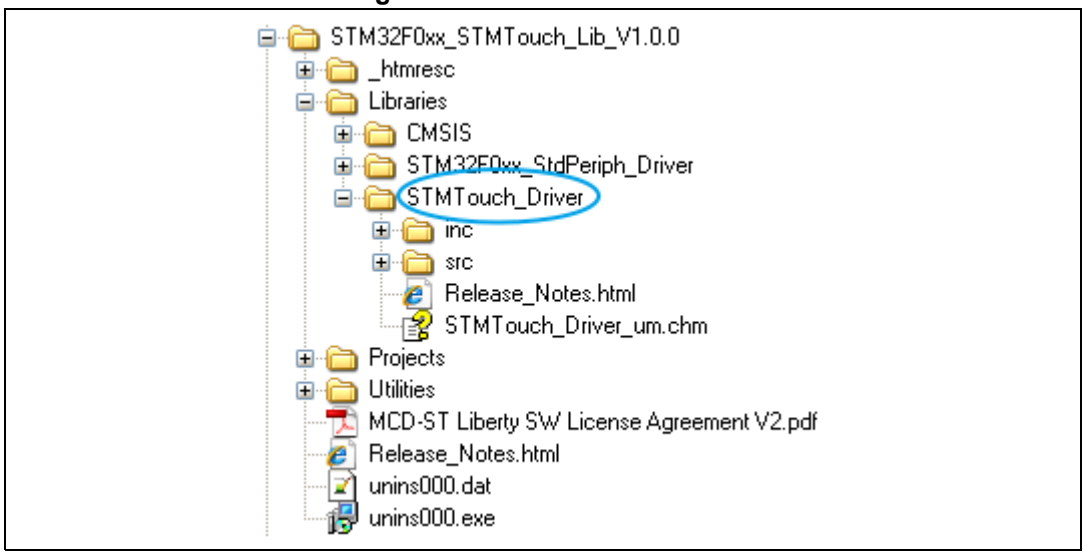
arm

2.2 Package description

The STMTouch driver is not supplied by itself. It is delivered instead inside each device STMTouch library (STMTouch_Driver folder present in the Libraries folder).

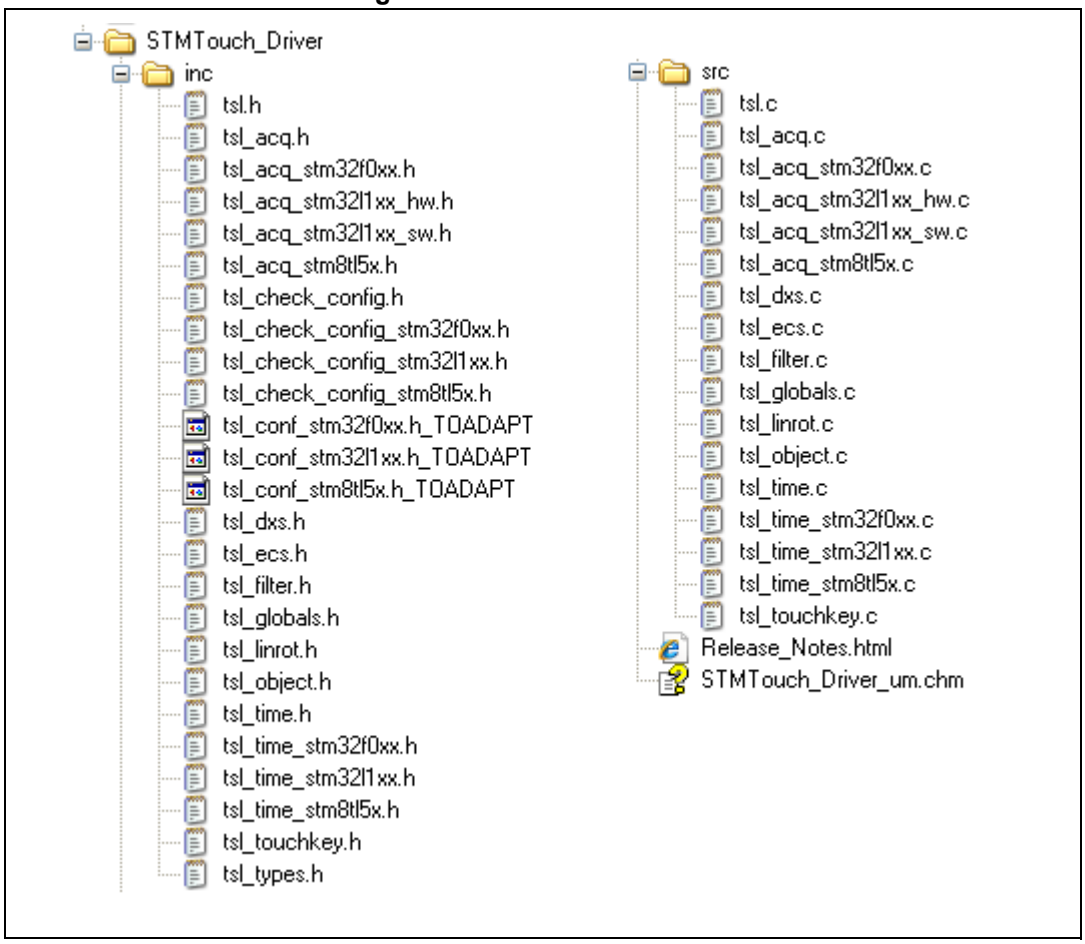
The following snapshots show an example of installation.

Figure 1. Installation folder 1/2



a. Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere

Figure 2. Installation folder 2/2



2.3 Main features

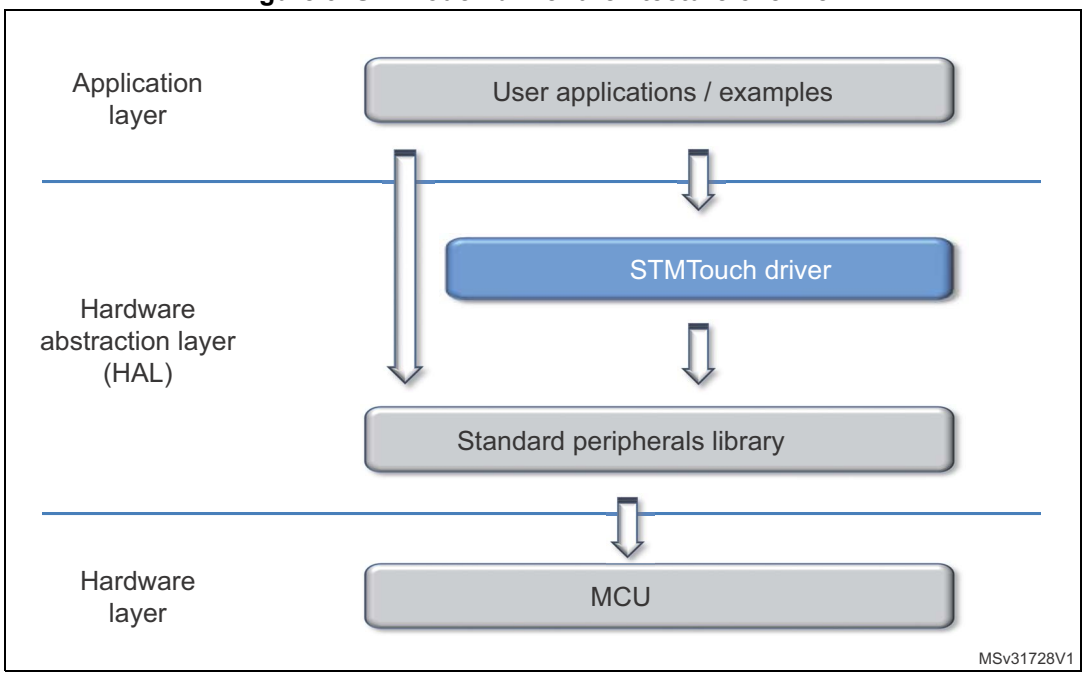
- Environment Change System (ECS)
- Detection Time Out (DTO)
- Detection Exclusion System (DXS)
- Noise filter
- Supports proximity, touchkeys and linear touch sensors
- Unlimited number of sensors
- Modular architecture allowing easy addition of new acquisitions or sensors
- Each sensor can have its own state machine
- Simplified timing management
- Run-time checking of functions parameters
- Management of error during acquisition

2.4 Architecture

2.4.1 Overview

The following figure shows the interactions between the STMTouch driver and the other firmware layers.

Figure 3. STMTouch driver architecture overview



The HAL is the hardware abstraction layer (HAL) which controls the device itself through hardware registers.

It is composed of different components:

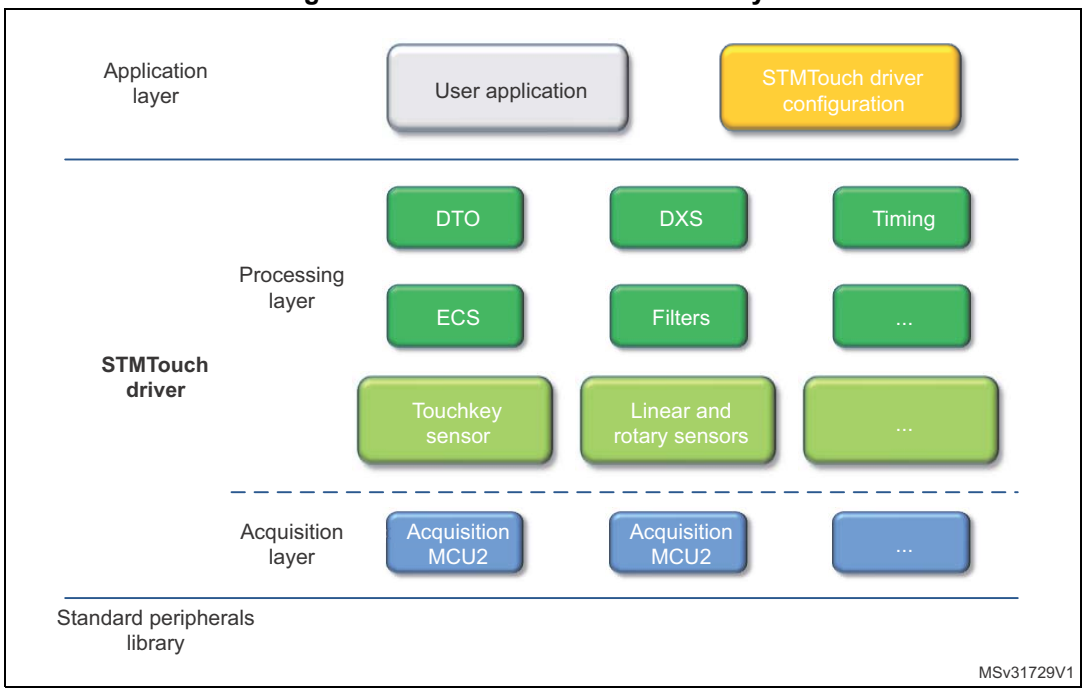
- The STMTouch driver
- The standard peripherals library
- The CMSIS firmware (for STM32 devices only)
- Utilities and third-parties firmwares

Note: The STMTouch driver can access directly to the MCU hardware registers using the map file provided by the standard peripherals library.

2.4.2 STMTouch driver layers

The following figure shows a more detailed view of the different STMTouch driver layers.

Figure 4. STMTouch driver detailed layers



The STMTouch driver is composed of three main layers:

- The acquisition layer
- The processing layer
- The configuration layer

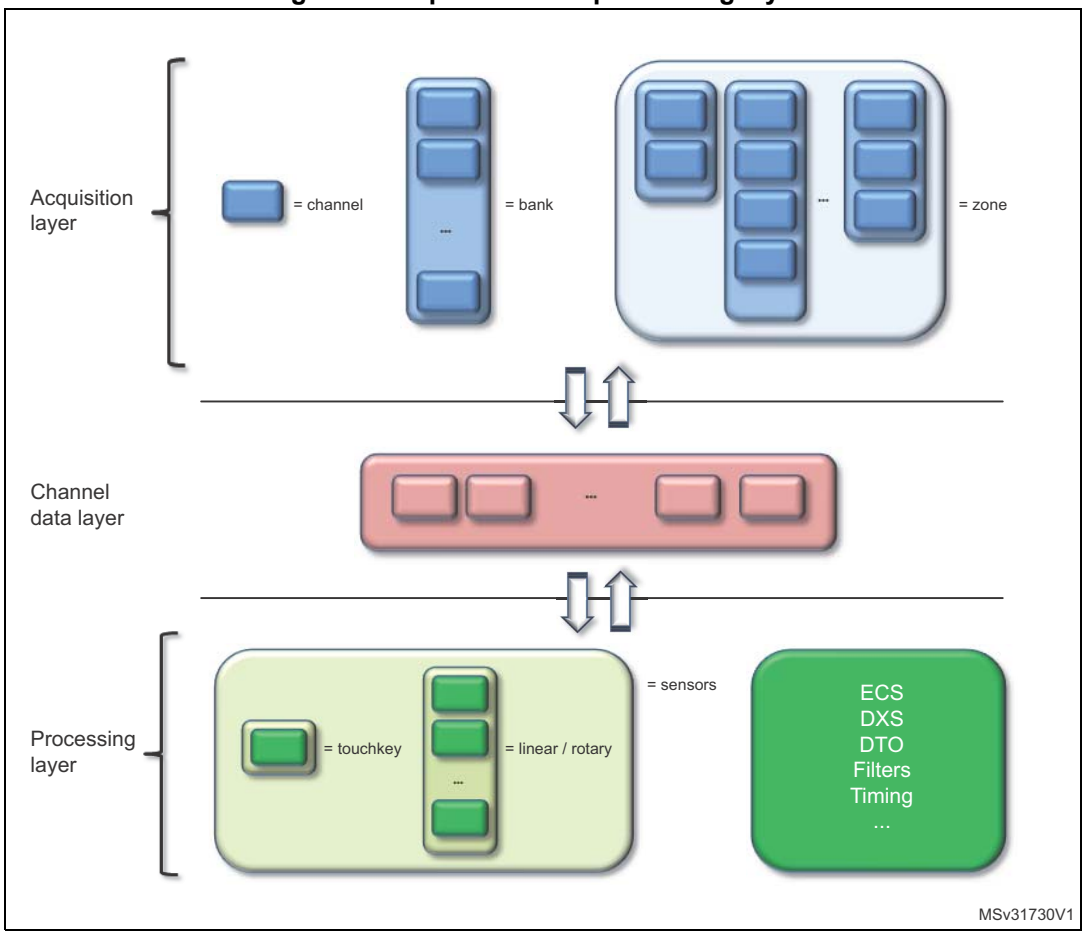
The configuration layer corresponds to what the user needs to write in its application code in order to use correctly the STMTouch driver. This includes all the channels and sensors declaration, the parameters, etc...

The acquisition and processing layers are described more in detail below.

2.4.3 Acquisition and processing layers

The following figure details the acquisition and processing layers and the different elements used in each layer.

Figure 5. Acquisition and processing layers



The **acquisition layer** role is to perform the acquisition of the different channels. The result of the acquisition (measure and flags) is stored inside the channel data layer. These informations will be accessed by the processing layer.

The acquisition layer has only access to the channel, bank and zone elements. It does not have access to the sensor elements.

The **channel data layer** role is to share information between the acquisition and processing layers. It stores the result of the acquisition (measure) for each channel and store different informations coming from the processing layer (reference, delta, flags, etc...).

Located in RAM, the ChannelData structure is the only interface between the acquisition and processing layers.

This **processing layer** consists in executing each sensors state machine, executing the different data processing like ECS, DXS, DTO and storing any useful information for the acquisition layer inside the channel data area.

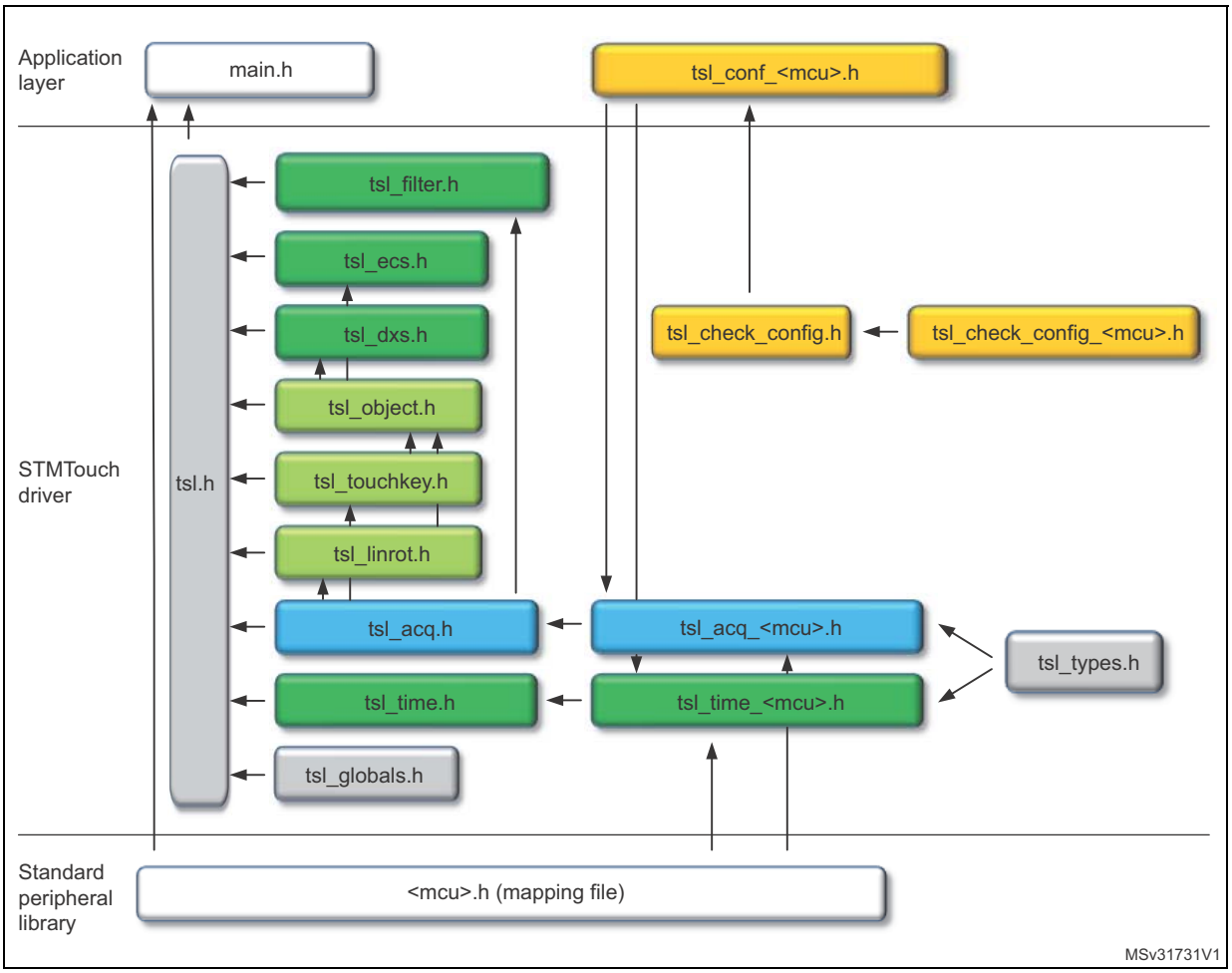
The processing layer does not have direct access to the channels, banks and zones. This access is made through the sensors.

2.4.4 Header files inclusion

The figure below provides a global view of the STMTouch driver usage and the interaction between the different header files.

Note: To simplify the drawing, only the most important links are shown. For example the tsl_globals.h file is also included in different files.

Figure 6. Header files inclusion



2.5 Channel

2.5.1 Principle

A channel is the basic element that is used to store several information like:

- where the source measurement can be found after the acquisition is performed (i.e. RX registers for STM8TL5x devices or TSC_IOGxCR registers for STM32F0xx/STM32F3xx devices)
- where are stored the measure, the reference, the delta, flags etc...

2.5.2 Resources

A channel is defined by 3 data structures:

- **TSL_ChannelSrc_T**: contains all information about the source measurement (index of the register containing the measurement, masks,...)
- **TSL_ChannelDest_T**: contains all information about the measurement destination (index in the channel data array).
- **TSL_ChannelData_T**: contains all data for the channel (measure, delta, reference, ...)

The channel depends on the acquisition technology. This is why the contents of this structures are not common for all acquisitions. They are declared in each acquisition header files (tsl_acq_<mcu>.h):

- tsl_acq_stm8l_hw.h for STM8L devices using hardware acquisition mode
- tsl_acq_stm8l_sw.h for STM8L devices using software acquisition mode
- **tsl_acq_stm8tl5x.h** for STM8TL5x devices
- **tsl_acq_stm32l1xx_hw.h** for STM32L1xx devices using the hardware acquisition mode
- **tsl_acq_stm32l1xx_sw.h** for STM32L1xx devices using the software acquisition mode
- **tsl_acq_stm32f0xx.h** for STM32F0xx devices
- **tsl_acq_stm32f3xx.h** for STM32F3xx devices

The maximum number of channels is only limited by the device (memory size and channels supported).

The user must declare all the channels array in its application code. It can be done directly in the main.c file or in any other file.

2.5.3 Parameters

- TSLPRM_TOTAL_CHANNELS

2.5.4 Usage example

The 3 channels structures must be declared in the application code.

Example of **channel source** array declaration for STM32F0xx devices. This structure must always be placed in ROM.

```
const TSL_ChannelSrc_T MyChannels_Src[TSLPRM_TOTAL_CHANNELS] =
{ { CHANNEL_0_SRC },
  { CHANNEL_1_SRC },
  { CHANNEL_2_SRC }};
```

Example of **channel destination** array declaration for STM32F0xx devices. This structure must always be placed in ROM.

```
const TSL_ChannelDest_T MyChannels_Dest[TSLPRM_TOTAL_CHANNELS] =
{ { CHANNEL_0_DEST },
  { CHANNEL_1_DEST },
  { CHANNEL_2_DEST }};
```

Note: The "CHANNEL_x_SRC" and "CHANNEL_x_DEST" are "#define" constants and are used for readability. The values are acquisition dependant.

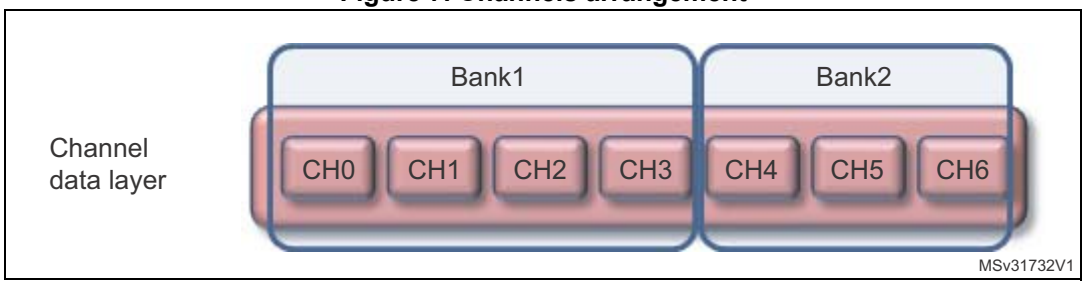
Example of **channel data** array declaration (i.e. channel data layer). This structure must always be placed in RAM.

```
TSL_ChannelData_T MyChannels_Data[TSLPRM_TOTAL_CHANNELS];
```

Warning: When several banks are present, it is mandatory to declare all channels of each bank consecutively in the source and destination structures.

Example:

Figure 7. Channels arrangement



Example of **channel source** array declaration for STM32F0xx devices.

```
CONST TSL_ChannelSrc_T MyChannels_Src[TSLPRM_TOTAL_CHANNELS] =
{
// Bank 1
{ CHANNEL_0_SRC, CHANNEL_0_IO_MSK, CHANNEL_0_GRP_MSK },
{ CHANNEL_1_SRC, CHANNEL_1_IO_MSK, CHANNEL_1_GRP_MSK },
{ CHANNEL_2_SRC, CHANNEL_2_IO_MSK, CHANNEL_2_GRP_MSK },
{ CHANNEL_3_SRC, CHANNEL_3_IO_MSK, CHANNEL_3_GRP_MSK },
// Bank 2
{ CHANNEL_4_SRC, CHANNEL_4_IO_MSK, CHANNEL_4_GRP_MSK },
{ CHANNEL_5_SRC, CHANNEL_5_IO_MSK, CHANNEL_5_GRP_MSK },
{ CHANNEL_6_SRC, CHANNEL_6_IO_MSK, CHANNEL_6_GRP_MSK }
};
```

2.6 Bank

2.6.1 Principle

A bank is a group of channels that are acquired simultaneously. The number of channels in the bank is variable.

2.6.2 Resources

The bank data are held by only one structure:

- TSL_Bank_T
- The bank depends also on the acquisition technology. These structures are declared in each acquisition header files (tsl_acq_<mcu>.h):

The maximum number of banks is only limited by the device.

The user must declare all the banks array in its application code. It can be done directly in the main.c file or in any other file.

The banks are used mainly by the functions described below. Some functions are common whatever the device and acquisition technology. Some others are dependent on the device.

Common functions:

- TSL_acq_BankGetResult()
- TSL_acq_BankCalibrate()

Device dependent functions:

- TSL_acq_BankConfig()
- TSL_acq_BankStartAcq()
- TSL_acq_BankWaitEOC()

2.6.3 Parameters

- TSLPRM_TOTAL_BANKS

2.6.4 Usage example

Example of 2 banks declaration for STM8TL5x devices:

```
// Always placed in ROM
const TSL_Bank_T MyBanks[TSLPRM_TOTAL_BANKS] =
{
  // Bank 0
  &MyChannels_Src[0],
  &MyChannels_Dest[0],
  MyChannels_Data,
  BANK_0_NBCHANNELS,
  // For STM8TL5x acquisition only
  BANK_0_MSK_CHANNELS,
  BIT_MASK_TX(BANK_0_TX),
  BANK_0_GROUP,
#if (BANK_0_MSK_TX < 0x8000) // a TX pin is used as transmitter
  BANK_0_MSK_CHANNELS,
#else // a RX pin is used as transmitter
  (BIT_MASK_RX(BANK_0_TX)| BANK_0_MSK_CHANNELS),
#endif

  // Bank 1
```

```
  &MyChannels_Src[4],
  &MyChannels_Dest[4],
  MyChannels_Data,
  BANK_1_NBCHANNELS,
  // For STM8TL5x acquisition only
  BANK_1_MSK_CHANNELS,
  BIT_MASK_TX(BANK_1_TX),
  BANK_1_GROUP,
#if (BANK_1_MSK_TX < 0x8000) // a TX pin is used as transmitter
  BANK_1_MSK_CHANNELS,
#else // a RX pin is used as transmitter
  (BIT_MASK_RX(BANK_1_TX)| BANK_1_MSK_CHANNELS),
#endif
};
```

The "BANK_x_NBCHANNELS", "BANK_x_MSK_CHANNELS", "BIT_x_MASK_TX", etc... are defines and are used for readability. These values are for STM8TL5x devices acquisition only.

Example of 3 banks declaration for STM32F0xx devices:

```
CONST TSL_Bank_T MyBanks[TSLPRM_TOTAL_BANKS] = {
  {&MyChannels_Src[0], &MyChannels_Dest[0], MyChannels_Data,
BANK_0_NBCHANNELS, BANK_0_MSK_CHANNELS, BANK_0_MSK_GROUPS},
  {&MyChannels_Src[1], &MyChannels_Dest[1], MyChannels_Data,
BANK_1_NBCHANNELS, BANK_1_MSK_CHANNELS, BANK_1_MSK_GROUPS},
  {&MyChannels_Src[2], &MyChannels_Dest[2], MyChannels_Data,
BANK_2_NBCHANNELS, BANK_2_MSK_CHANNELS, BANK_2_MSK_GROUPS}
};
```

2.7 Zone

2.7.1 Principle

A zone is an ordered group of banks. It is used to easily cascade the acquisition of many banks. The acquisition of the next bank in a zone is launched directly in the interrupt routine managing the acquisition result of the current acquired bank.

2.7.2 Resources

This feature is optional and is enabled/disabled using the **TSLPRM_USE_ZONE** parameter.

The zone data are held by only one structure:

- TSL_Zone_T

The zones are used mainly by the function:

- TSL_acq_ZoneConfig()

2.7.3 Parameters

- **TSLPRM_USE_ZONE**

2.7.4 Usage example

Example of a zone declaration containing 3 banks:

```
TSL_tIndex_T MyBankSorting[TSLPRM_TOTAL_BANKS] = {2, 0, 1, 3, 4, 5};
TSL_Zone_T MyZone = {
  MyBankSorting,
  0,
  3 // Number of Banks in the Zone
};
```

In this example the "MyBankSorting" array contains the list of the banks indexes to acquire. And only the 3 first Banks will be acquired (indexes 2, 0 and 1).

2.8 Objects

2.8.1 Principle

The term “object” or “sensor” stands for any sensors type (touchkeys, linear and rotary touch sensors, etc...) supported by the STMTouch driver.

2.8.2 Resources

All processing that affect the sensors in general are defined in:

- tsl_object.c
- tsl_object.h

The functions are:

- TSL_obj_GroupInit()
- TSL_obj_GroupProcess()
- TSL_obj_SetGlobalObj()

A sensor is described by the structures:

- TSL_Object_T
- TSL_ObjectGroup_T

2.8.3 Parameters

- TSLPRM_TOTAL_OBJECTS

2.8.4 Usage example

First, all touchkeys, linear and rotary touch sensors (described after) used in the application must be described first as 'generic' sensor or object.

Example:

```
// Mix of touchkeys and Linear touch sensors
```

```
const TSL_Object_T MyObjects[TSLPRM_TOTAL_OBJECTS] =
{
  // TKeys
  { TSL_OBJ_TOUCHKEYB, (TSL_TouchKeyB_T *)&MyTKeys[0]   },
  { TSL_OBJ_TOUCHKEYB, (TSL_TouchKeyB_T *)&MyTKeys[1]   },
  // Linear touch sensors
  { TSL_OBJ_LINEARB,   (TSL_LinRotB_T   *)&MyLinRots[0] }
};
```

These objects must be placed in ROM memory.

Once this done, it is necessary to create at least one group of sensors. These groups will be used by the different processing routines (ECS, DXS, etc...).

These groups of objects must placed in RAM.

Example:

```
TSL_ObjectGroup_T MyObjGroup_All = {
 MyObjects,
 3,
 0,
 TSL_STATE_NOT_CHANGED
};
```

Then, all the sensors must be initialized and “processed”. This is done in the main function of the application:

```
int main(void) {
  ...
  TSL_obj_GroupInit(&MyObjGroup_All);
  ...
  while (1) {
    ...
    TSL_obj_GroupProcess(&MyObjGroup_All);
    ...
  }
}
```

2.9 Touchkey sensor

2.9.1 Principle

The touchkey sensor is composed of only one channel. It acts as a simple “button” with two states RELEASE and DETECT (or TOUCH if DXS is enabled).

2.9.2 Resources

All the functions related to this sensor are described in the files:

- tsl_touchkey.c
- tsl_touchkey.h

Two types of touchkey sensor are available:

- Basic: defined by the **TSL_TouchKeyB_T** structure
- Extended: defined by the **TSL_TouchKey_T** structure

Two functions (called methods) are used to initialized the sensor parameters and to run the sensor state machine:

- TSL_tkey_Init()
- TSL_tkey_Process()

The difference between the “basic” and “extended” concerns the usage of the methods and sensor state machine.

For the “basic” sensor, the methods and state machine are those used in the **TSL_Params** structure.

For the “extended” sensor, the methods and state machine are those declared in their own structure.

2.9.3 Parameters

- TSLPRM_TOTAL_TKEYS

2.9.4 Usage example

The user must declared these methods in the application code.

Note: One can also use its own initialization and process functions instead:

```
const TSL_TouchKeyMethods_T MyTKeys_Methods =
{
  TSL_tkey_Init,
  TSL_tkey_Process
};
```

The declaration of the touchkey sensor is done by the user in the application code:

Example with “basic” sensor:

```
// "Basic" touchkeys: Always placed in ROM
const TSL_TouchKeyB_T MyTKeys[TSLPRM_TOTAL_TKEYS] =
{
  { &MyTKeys_Data[0], &MyTKeys_Param[0], &MyChannels_Data[0] },
  { &MyTKeys_Data[1], &MyTKeys_Param[1], &MyChannels_Data[1] },
  { &MyTKeys_Data[2], &MyTKeys_Param[2], &MyChannels_Data[2] }
};
```

Example with “extended” sensor:

```
// "Extended" TouchKeys: Always placed in ROM
const TSL_TouchKey_T MyTKeys[TSLPRM_TOTAL_TKEYS] =
{
  { &MyTKeys_Data[0], &MyTKeys_Param[0], &MyChannels_Data[0],
MyTKeys_StateMachine, &MyTKeys_Methods },
  { &MyTKeys_Data[1], &MyTKeys_Param[1], &MyChannels_Data[1],
MyTKeys_StateMachine, &MyTKeys_Methods },
```

```
  { &MyTKeys_Data[2], &MyTKeys_Param[2], &MyChannels_Data[2],
MyTKeys_StateMachine, &MyTKeys_Methods }
};
```

2.10 Linear and rotary sensors

2.10.1 Principle

The linear and rotary sensors are like a touchkey sensor except that they are composed of a variable number of channels. The difference between the linear and rotary touch sensors is how the electrodes are organized together.

The linear and rotary sensors have additional fields in their structure compared to touchkey sensors:

- Number of channels
- Delta coefficient table
- Position offset table
- Sector computation parameter
- Position correction parameter for linear sensor

The last 3 fields are used to calculate the position.

2.10.2 Number of channels

Only 1, 3, 4, 5 and 6 channels are supported today by the STMTouch driver. Additional number of channels can be added by the end-user. See the *Position offset table* bullet below for more detail.

Note: A Linear sensor with 1 channel is equivalent to one touchkey sensor. When an application uses both touchkey sensor and linear and rotary sensor, it is better to use touchkeys with a 1-channel linear touch sensor. In this case the gain in memory size is important as the touchkey sensor state machine is not used.

For optimum performance of a linear or rotary sensor, all channels of such a sensor must be acquired simultaneously.Therefore all channels must belong to the same bank, which means the I/Os must be selected from different analog I/O groups. Please refer to datasheet for more information regarding I/O groups and available capacitive sensing GPIOs.

2.10.3 Delta coefficient table

The delta coefficient table is used to adjust each channel of the linear and rotary sensors. Each value is a 16-bit integer. The MSB is the integer part, the LSB is the real part.

Examples:

To apply a factor of 1.10:

- MSB equal 0x01
- LSB equal 0x1A (0.10 x 256 = 25.6 -> rounded to 26 = 0x1A)

To apply a factor 1.00:

- MSB equal 0x01
- LSB equal 0x00

To apply a factor 0.90:

- MSB equal 0x00
- LSB equal 0xE6 (0.90 x 256 = 230.4 -> rounded to 230 = 0xE6)

This results in the following delta coefficient table:

```
CONST uint16_t MyLinRot0_DeltaCoeff[3] = {0x011A, 0x0100, 0x00E6};
```

The number of delta coefficient table is not limited. The same delta coefficient table can be shared by several linear and rotary sensors.

2.10.4 Electrodes placement

The placement (design) of the electrodes can be done in three different manners:

1. Mono electrode design

   The number of electrodes is equivalent to the number of channels. This design is used for linear and rotary sensors.

   Abbreviations: **LIN_M1**, **LIN_M2** and **ROT_M**

   Examples:

   – CH1 CH2 CH3
   – CH1 CH2 CH3 CH4
   – CH1 CH2 CH3 CH4 CH5

2. Dual electrode design

   All the electrodes are duplicated and interlaced together in order to increase the touch area.

   This design is used for linear and rotary sensors composed **with at least 5 channels**.

   Abbreviation: **ROT_D**

   Examples with 5 channels:

   – CH1 CH2 CH3 CH4 CH5 CH1 CH3 CH5 CH2 CH4
   – CH1 CH2 CH3 CH4 CH5 CH2 CH4 CH1 CH3 CH5
   – CH1 CH2 CH3 CH4 CH5 CH3 CH1 CH4 CH2 CH5

3. Half-ended electrode design

   The first electrode is duplicated and the replica is placed at the end. The size of the first and last electrode is **half the size** of the other electrodes. This design is used for **linear sensors only**. The 0 and 255 positions are obtained more easily compared to the Mono electrodes design.

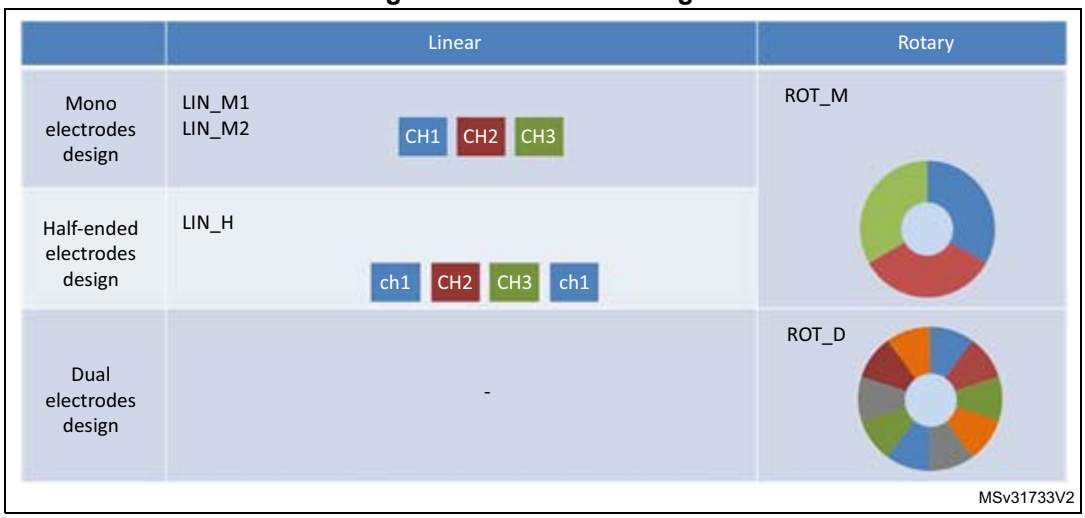
   Abbreviation: **LIN_H**

   Examples:

   – ch1 CH2 CH3 ch1
   – ch1 CH2 CH3 CH4 ch1
   – ch1 CH2 CH3 CH4 CH5 ch1

The following figure summarizes the different electrodes designs we can have on linear and rotary sensors:

Figure 8. Electrodes designs



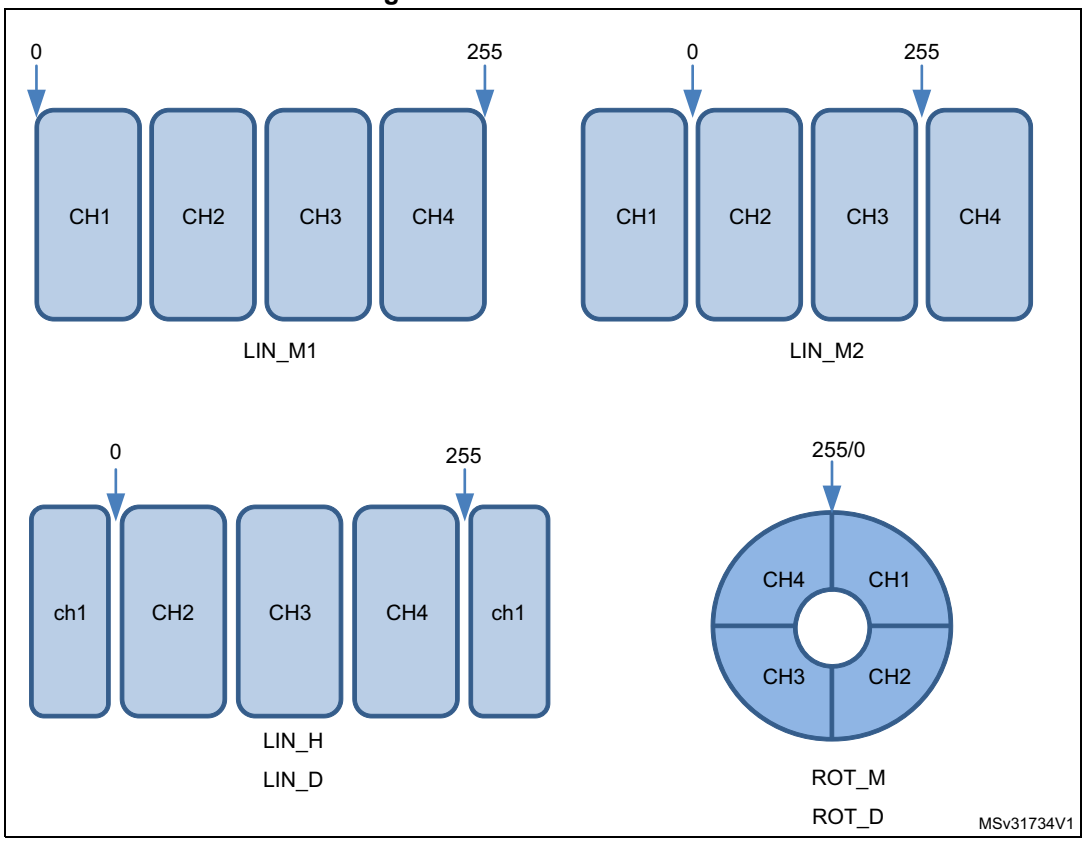
Positions 0 and 255

Special care must be taken for the 0 and 255 positions on linear sensors. These positions are placed differently depending on the electrodes design used:

- **LIN_M1**: the 0 and 255 positions are placed completely at the sensor's **extremities**. These positions can be obtain with difficulty if the electrodes are too big or if they are separated by an important space.
- **LIN_M2**, **LIN_H**: the 0 position is placed **between the first and second electrodes**. The 255 position is placed **between the last two electrodes**.
- **ROT_M** and **ROT_D**: the 0 and 255 positions are always placed **between the first and the last electrodes**.

The following figures summarizes the different placements of the 0 and 255 positions with 4 channels sensors:

Figure 9. Positions 0 and 255



The following table summarizes the different linear and rotary sensors electrodes designs supported by the STMTouch driver:

Table 3. Supported linear and rotary sensors

Number of Channels	LIN_M1	LIN_M2	LIN_H	ROT_M	ROT_D
3	Yes	Yes	Yes	Yes	No
4	Yes	Yes	Yes	Yes	No
5	Yes	Yes	Yes	Yes	Yes
6	Yes	Yes	Yes	Yes	No

Each supported electrode design is described by 3 fields in the **TSL_LinRot_T** or **TSL_LinRotB_T** structures:

- Position offset table
- Sector computation parameter
- Position correction parameter for linear sensor

These 3 fields are defined in the **tsl_linrot.c** and **tsl_linrot.h** files and follow the naming convention:

Position offset table: TSL_POSOFF_nCH_[LIN|ROT]_[M1|M2|H|D]

Sector computation parameter: TSL_SCTCOMP_nCH_[LIN|ROT]_[M1|M2|H|D]

Position correction parameter for linear sensor: TSL_POSCORR_nCH_LIN_[M1|M2|H|D]

With:

- n = number of channels
- LIN = linear sensor
- ROT = rotary sensor
- M1 = mono electrodes design with 0/255 position at extremities
- M2 = mono electrodes design
- H = half-ended electrodes design
- D = dual electrodes design

In order to gain memory space, each table is only compiled if its corresponding parameter is set in the configuration file:

TSLPRM_USE_nCH_[LIN|ROT]_[M1|M2|H|D]

2.10.5 Resources

All the functions related to this sensor are described in the files:

- tsl_linrot.c
- tsl_linrot.h

Two types of linear and rotary sensor are available:

- basic: defined by the **TSL_LinRotB_T** structure
- extended: defined by the **TSL_LinRot_T** structure

The difference between “basic” and “extended” is the same as for the touchkey sensor.

Three functions (called methods) are used to initialized the sensor parameters, run the sensor state machine and calculate the position.

- TSL_linrot_Init()
- TSL_linrot_Process()
- TSL_linrot_CalcPos()

2.10.6 Parameters

- TSLPRM_TOTAL_LINROTS

2.10.7 Usage example

The user must declared these methods in the application code.

Note: One can also use its own initialization and process functions instead:

```
CONST TSL_LinRotMethods_T MyLinRots_Methods =
{
  TSL_linrot_Init,
  TSL_linrot_Process,
  TSL_linrot_CalcPos
};
```

The declaration of the linear and rotary sensor is done by the user in the application code in the same manner as for touchkey sensor.

Example with 2 “basic” linear touch sensors, one with 3 channels half-ended and the other with 5 channels mono electrodes design:

```
CONST TSL_LinRotB_T MyLinRots[2] =
{
  // LinRot sensor 0
  &MyLinRots_Data[0],
  &MyLinRots_Param[0],
  &MyChannels_Data[CHANNEL_9_DEST],
  3, // Number of channels
  MyLinRot0_DeltaCoeff, // Delta coefficient table
  (TSL_tsignPosition_T *)TSL_POSOFF_3CH_LIN_H, // Position table
  TSL_SCTCOMP_3CH_LIN_H, // Sector compensation
  TSL_POSCORR_3CH_LIN_H, // Position correction
  // LinRot sensor 1
  &MyLinRots_Data[1],
  &MyLinRots_Param[1],
  &MyChannels_Data[CHANNEL_12_DEST],
  5, // Number of channels
  MyLinRot1_DeltaCoeff, // Delta coefficient table
  (TSL_tsignPosition_T *)TSL_POSOFF_5CH_LIN_M2, // Position table
  TSL_SCTCOMP_5CH_LIN_M2, // Sector compensation
  TSL_POSCORR_5CH_LIN_M2  // Position correction
};
```

Example of one “extended” (i.e. having its own state machine and methods) linear touch sensor with 3 channels half-ended:

```
CONST TSL_LinRot_T MyLinRots[1] =
{
  // LinRot sensor 0
  &MyLinRots_Data[0],
  &MyLinRots_Param[0],
  &MyChannels_Data[CHANNEL_0_DEST],
  3, // Number of channels
  MyLinRot0_DeltaCoeff,
  (TSL_tsignPosition_T *)TSL_POSOFF_3CH_LIN_H,
  TSL_SCTCOMP_3CH_LIN_H,
  TSL_POSCORR_3CH_LIN_H,
  MyLinRots_StateMachine, // Specific state machine
  &MyLinRots_Methods // Specific methods
};
```

Example of one “extended” rotary touch sensor with 3 channels mono electrode design:

```
CONST TSL_LinRot_T MyLinRots[0] =
{
```

```
  // LinRot sensor 0
  &MyLinRots_Data[0],
  &MyLinRots_Param[0],
  &MyChannels_Data[CHANNEL_0_DEST],
  3, // Number of channels
  MyLinRot0_DeltaCoeff,
  (TSL_tsignPosition_T *)TSL_POSOFF_3CH_ROT_M,
  TSL_SCTCOMP_3CH_ROT_M,
  0, // No position correction needed on a Rotary sensor
  MyLinRots_StateMachine, // Specific state machine
  &MyLinRots_Methods // Specific methods
};
```

2.11 Main state machine

The main state machine is managed by the user in the application layer. A set of functions are available to accomplish this task. The main state machine can be defined with polling or with interrupt modes, using one or several banks. The modularity of the STMTouch driver allows also the application code to be inserted between acquisition and processing tasks. Several examples are given below.

The functions to use for the acquisition are:

- TSL_acq_BankConfig()
- TSL_acq_BankStartAcq()
- TSL_acq_BankWaitEOC()
- TSL_acq_BankGetResult()

These functions are device dependent and are described in the **tsl_acq_<mcu>.c** files.

The functions to use for the processing are:

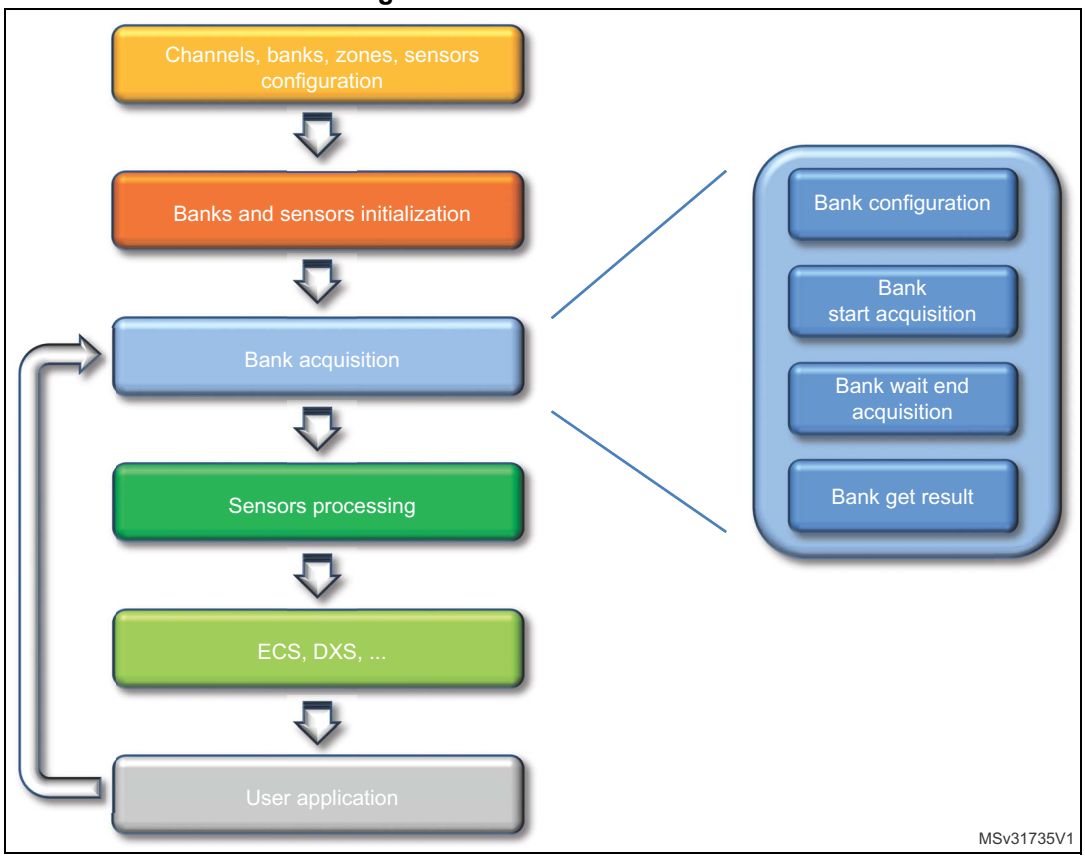
- TSL_obj_GroupProcess()
- TSL_ecs_Process()
- TSL_dxs_FirstObj()

Other functions that can be used during the processing:

- TSL_tim_CheckDelay_ms()
- TSL_obj_SetGlobalObj()
- TSL_tkey_GetStateId()
- TSL_tkey_GetStateMask()
- TSL_linrot_SetStateOff()
- TSL_linrot_SetStateCalibration()

The main state machine principle is illustrated by the figure below:

Figure 10. Main state machine



The main state machine steps are:

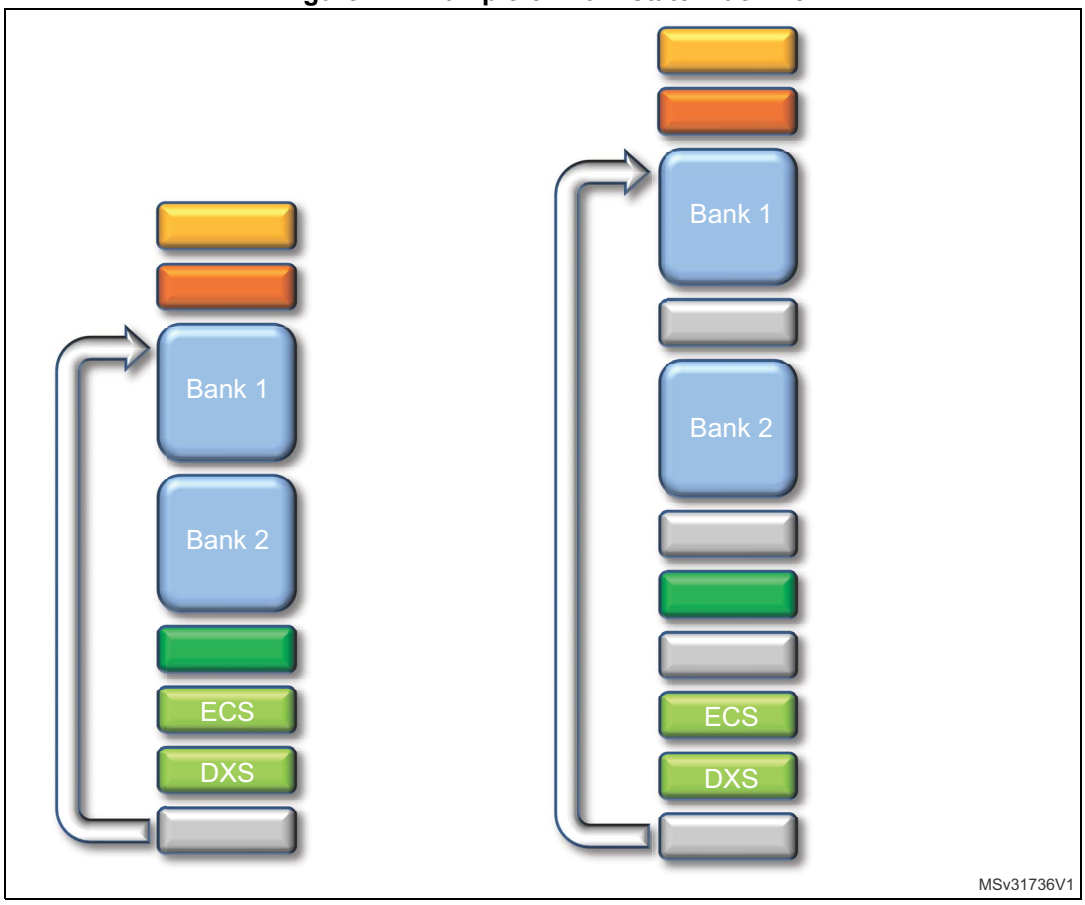
1. The **channels, banks, zones and sensors configuration** step are used to declare all the different elements. This is done in the global declaration section in the main application file. See the section associated to each element for more details.
2. The **banks and sensors initialization** step is used to initialize the STMTouch driver modules. The sensors parameters are initialized with their default value defined in the configuration files.
3. **The banks acquisition** step is used to perform the acquisition of the banks. It is composed of 4 sub-steps:
   - **configuration**: used to configure all channels of the bank
   - **start acquisition**: used to launch the measurement on all channels of the bank
   - **wait end acquisition**: used to wait the end of acquisition of all channels of the bank
   - **get result**: used to read all the channels measurements and to store them in the channel data layer.
4. **The sensors processing** step is used to execute the state machine of the sensors.

Note: The debouncing, Detection Time Out and re-calibration are automatically performed inside this step.

5. The **ECS, DXS** step is used to execute other algorithms that are not performed in the sensor state machine like the ECS, DXS, other filters, etc... This step is optional and it can be executed at certain time intervals (mainly for ECS).
6. The **user application** step is used to execute the application layer (read the sensors state, decide which actions to perform, manage ERROR states, etc...). The user application can also be placed between other steps, for example it can be done between the “sensors processing” step and the "ECS/DXS".

There are multiple manners to perform the main state machine. The following figures show some examples with two banks.

Figure 11. Example of main state machine



2.12 Sensors state machine

2.12.1 Overview

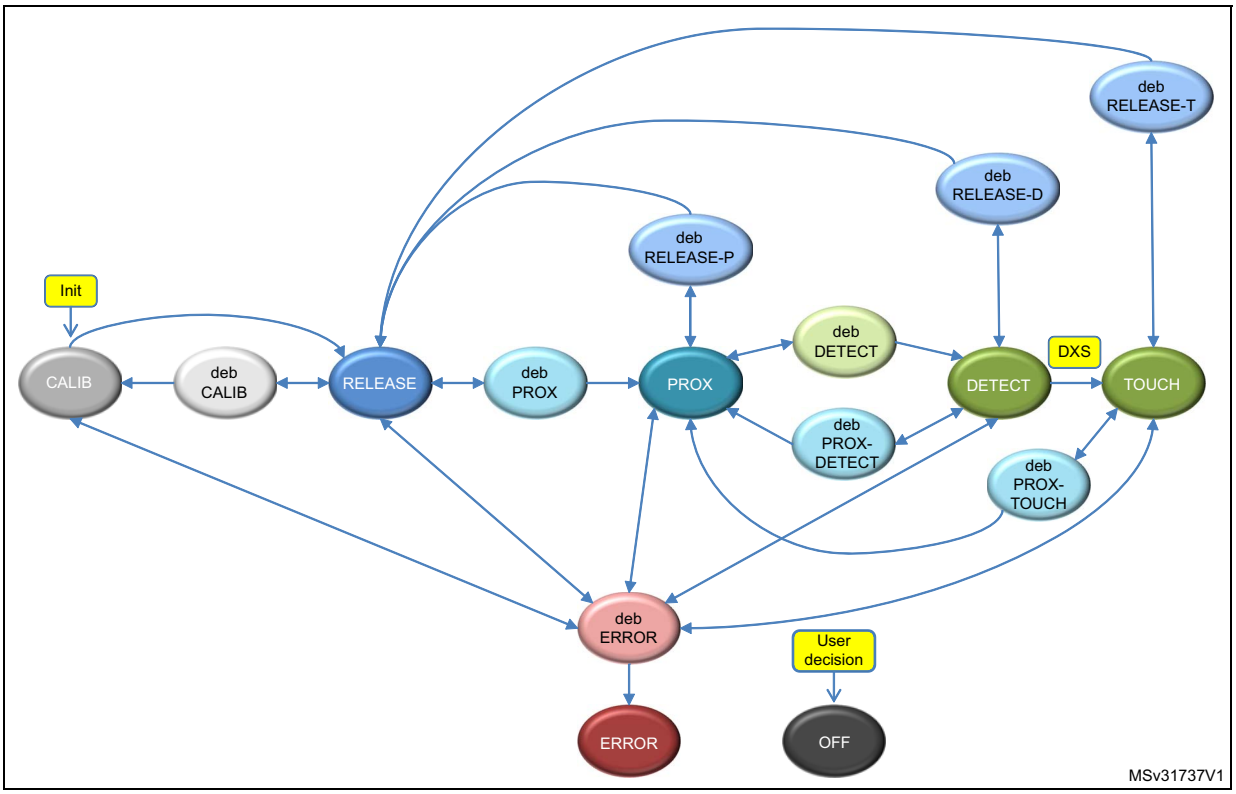
The state machine is managed in the files:

- **tsl_touchkey.c** and **tsl_touchkey.h** for the touchkey sensors
- **tsl_linrot.c** and **tsl_linrot.h** for the linear and rotary sensors

There is a total of **20 states** defined in the **TSL_StateId_enum_T** structure.

The following figure shows the simplified state machine used by any sensor (for clarity not all the connections between states are shown).

Figure 12. Simplified sensors state machine



2.12.2 States constant table

Each state ID is associated to a mask and a function. The association STATE_ID-mask-function is made in the user application code using a constant table of the **TSL_State_T** type. The name of this table is free and user can give any name he wants. If no function is needed simply put a zero instead of the function name.

Here below an example of touchkey sensors state machine:

```
// Touchkeys state machine
const TSL_State_T MyTKeys_StateMachine[] =
{
//-------------------------------------------------------------------------
// ID      MASK                                FUNCTION
//-------------------------------------------------------------------------
// Calibration states
/*  0 */ { TSL_STATEMASK_CALIB, TSL_tkey_CalibrationStateProcess },
/*  1 */ { TSL_STATEMASK_DEB_CALIB, TSL_tkey_DebCalibrationStateProcess },
// RELEASE states
/*  2 */ { TSL_STATEMASK_RELEASE,           TSL_tkey_ReleaseStateProcess },
#if TSLPRM_USE_PROX > 0
```

```
/*  3 */ { TSL_STATEMASK_DEB_RELEASE_PROX,
TSL_tkey_DebReleaseProxStateProcess },
#else
/*  3 */ { TSL_STATEMASK_DEB_RELEASE_PROX, 0 },
#endif
/*  4 */ { TSL_STATEMASK_DEB_RELEASE_DETECT,
TSL_tkey_DebReleaseDetectStateProcess },
/*  5 */ { TSL_STATEMASK_DEB_RELEASE_TOUCH,
TSL_tkey_DebReleaseTouchStateProcess },
#if TSLPRM_USE_PROX > 0
// Proximity states
/*  6 */ { TSL_STATEMASK_PROX,               TSL_tkey_ProxStateProcess },
/*  7 */ { TSL_STATEMASK_DEB_PROX,          TSL_tkey_DebProxStateProcess },
/*  8 */ { TSL_STATEMASK_DEB_PROX_DETECT,
TSL_tkey_DebProxDetectStateProcess },
/*  9 */ { TSL_STATEMASK_DEB_PROX_TOUCH,
TSL_tkey_DebProxTouchStateProcess },
#else
/*  6 */ { TSL_STATEMASK_PROX,               0 },
/*  7 */ { TSL_STATEMASK_DEB_PROX,           0 },
/*  8 */ { TSL_STATEMASK_DEB_PROX_DETECT,    0 },
/*  9 */ { TSL_STATEMASK_DEB_PROX_TOUCH,     0 },
#endif
// DETECT states
/* 10 */ { TSL_STATEMASK_DETECT,             TSL_tkey_DetectStateProcess },
/* 11 */ { TSL_STATEMASK_DEB_DETECT,        TSL_tkey_DebDetectStateProcess
},
// TOUCH state
/* 12 */ { TSL_STATEMASK_TOUCH,              TSL_tkey_TouchStateProcess },
// ERROR states
/* 13 */ { TSL_STATEMASK_ERROR,              MyTKeys_ErrorStateProcess },
/* 14 */ { TSL_STATEMASK_DEB_ERROR_CALIB,    TSL_tkey_DebErrorStateProcess
},
/* 15 */ { TSL_STATEMASK_DEB_ERROR_RELEASE,  TSL_tkey_DebErrorStateProcess
},
/* 16 */ { TSL_STATEMASK_DEB_ERROR_PROX,     TSL_tkey_DebErrorStateProcess
},
/* 17 */ { TSL_STATEMASK_DEB_ERROR_DETECT,   TSL_tkey_DebErrorStateProcess
},
/* 18 */ { TSL_STATEMASK_DEB_ERROR_TOUCH,    TSL_tkey_DebErrorStateProcess
},
// Other states
/* 19 */ { TSL_STATEMASK_OFF,                MyTKeys_OffStateProcess }
};
```

The STMTouch driver contains all the functions needed to manage each state. However the user can copy and adapt one or several functions to fit its application requirements.

Example:

```
/*  0 */ { TSL_STATEMASK_CALIB, MyTkeys_CalibrationStateProcess },
```

Note: The two functions used to manage the ERROR and OFF states are not part of the STMTouch driver. These functions are managed by the application.

For linear and rotary sensor state machine, it is the same principle. The functions used to manage each state start with the prefix “TSL_linrot_”:

```
CONST TSL_State_T MyLinRots_StateMachine[] =
{
// Calibration states
/*  0 */ { TSL_STATEMASK_CALIB, TSL_linrot_CalibrationStateProcess },
```

2.12.3 States detail

The two tables below show the detail of how each state is entered following the thresholds measured.

Table 4. Detailed sensors states 1/2

<table>
<tr><th>Previous state</th><th>all excepted 13</th><th>all excepted 13</th><th>2p,10p,12p,3, 4p,5p,7,8,9,11p</th><th>2,4,11</th><th>2p,6,4p,7,8, 11p</th><th>DXS,5</th><th>DXS,5p,9</th><th>2,2p,1</th><th>2,2p,6,10, 10p,12,12p,0,14..18</th></tr>
<tr><th>state nb</th><th>2</th><th>2p</th><th>6</th><th>10</th><th>10p</th><th>12</th><th>12p</th><th>0</th><th>13</th></tr>
<tr><th>Current state</th><th>RELEASE</th><th>RELEASE with PROX</th><th>PROX</th><th>DETECT</th><th>DETECT with PROX</th><th>TOUCH</th><th>TOUCH with PROX</th><th>CALIB</th><th>ERROR</th></tr>
<tr><td>Delta</td><td colspan="9"></td></tr>
<tr><td>DETECT IN Th</td><td>deb DETECT or DETECT+DTO</td><td>deb DETECT or DETECT+DTO</td><td>deb DETECT or DETECT+DTO</td><td rowspan="2">same or CALIB if DTO</td><td rowspan="2">same or CALIB if DTO</td><td rowspan="2">same or CALIB if DTO</td><td rowspan="2">same or CALIB if DTO</td><td rowspan="5">RELEASE or ERROR</td><td rowspan="5">same</td></tr>
<tr><td></td><td rowspan="4">same</td><td rowspan="2">deb PROX or PROX+DTO</td><td rowspan="3">same or CALIB if DTO</td></tr>
<tr><td>DETECT OUT Th

PROX IN Th</td><td rowspan="3">deb RELEASE-DETECT or RELEASE</td><td rowspan="2">deb PROX-DETECT or PROX+DTO</td><td rowspan="3">deb RELEASE-TOUCH or RELEASE</td><td rowspan="2">deb PROX-TOUCH or PROX+DTO</td></tr>
<tr><td></td><td rowspan="2">same</td></tr>
<tr><td>PROX OUT Th</td><td rowspan="2">deb RELEASE-PROX or RELEASE</td><td>deb RELEASE-DETECT or RELEASE</td><td>deb RELEASE-TOUCH or RELEASE</td></tr>
<tr><td>CALIB Th</td><td>deb CALIB or CALIB</td><td>deb CALIB or CALIB</td><td>deb ERROR or ERROR</td><td>deb ERROR or ERROR</td><td>deb ERROR or ERROR</td><td>deb ERROR or ERROR</td><td>deb ERROR or ERROR</td><td>same</td></tr>
</table>

Table 5. Detailed sensors states 2/2

Previous state	**6**	**10**	**10p,8**	**12**	**12p,9**	**2p,11p**	**10p**	**12p**	**2**	**2p,6,7**	**2,2p**	**2,2p,6,10, 10p,12,12p ,0**
state nb	**3**	**4**	**4p**	**5**	**5p**	**7**	**8**	**9**	**11**	**11p**	**1**	**14..18**
Current state	**deb RELEASE-PROX**	**deb RELEASE-DETECT**	**deb RELEASE-DETECT with PROX**	**deb RELEASE-TOUCH**	**deb RELEASE-TOUCH with PROX**	**deb PROX**	**deb PROX-DETECT**	**deb PROX-TOUCH**	**deb DETECT**	**deb DETECT with PROX**	**deb CALIB**	**deb ERROR**
Delta												
DETECT IN Th	PROX	DETECT	DETECT	TOUCH	TOUCH	deb DETECT or DETECT+ DTO	DETECT	TOUCH	same or DETECT+ DTO	same or DETECT+ DTO	RELEASE	RELEASE PROX DETECT TOUCH CALIB
						same or PROX+ DTO			RELEASE	deb PROX or PROX+ DTO		
DETECT OUT Th PROX IN Th		same or RELEASE	PROX	same or RELEASE	PROX		same or PROX+ DTO	same or PROX+ DTO				
						RELEASE				RELEASE		
PROX OUT Th	same or RELEASE		same or RELEASE		same or RELEASE		deb RELEASE-DETECT or RELEASE	deb RELEASE-TOUCH or RELEASE				
CALIB Th											same or CALIB	
if ACQ ERROR	PROX	DETECT	DETECT	TOUCH	TOUCH	RELEASE	DETECT	TOUCH	RELEASE	RELEASE	RELEASE	ERROR

2.12.4 Calibration state

It consists in calculating the reference for all the channels of a sensor. An average of a certain number of measurements is done.

The number of measurement samples to use for the calibration is defined by the **TSLPRM_CALIB_SAMPLES** parameter.

After reset the initialization method of each object is called. This method initializes the sensor parameters and then goes in the calibration state. After the calibration is done, the sensor goes in the RELEASE state or ERROR state except if an error occurred.

Related functions:

- TSL_tkey_CalibrationStateProcess()
- TSL_linrot_CalibrationStateProcess()
- TSL_tkey_SetStateCalibration()
- TSL_linrot_SetStateCalibration()

Calibration delay

If a noise filter is used it should be necessary to wait a certain amount of measurement samples before to start the reference calculation. This number of samples to wait is defined by the **TSLPRM_CALIB_DELAY** parameter.

Re-calibration

If the calibration threshold is reached while in RELEASE state, a new calibration is performed. This “re-calibration” prevents the application to get stuck if something touches permanently the sensor like a drop of water for example.

2.12.5 RELEASE state

Corresponds to the “idle” state of the sensor when no presence is detected.

Related functions:

- TSL_tkey_ReleaseStateProcess()
- TSL_linrot_ReleaseStateProcess()

2.12.6 Proximity state

This state is optional and is enabled or disabled using the **TSLPRM_USE_PROX** parameter.

Related functions:

- TSL_tkey_ProxStateProcess()
- TSL_linrot_ProxStateProcess()

2.12.7 DETECT state

It is the “normal” state when the sensor is touched.

Related functions:

- TSL_tkey_DetectStateProcess()
- TSL_linrot_DetectStateProcess()

2.12.8 TOUCH state

Same as DETECT state excepted that it is entered only by the DXS processing. If the DXS is not used this state is never entered.

Related functions:

- TSL_tkey_TouchStateProcess()
- TSL_linrot_TouchStateProcess()

2.12.9 ERROR state

It is used to catch all acquisition errors detected in the other states.

The management of this state must be performed at application level.

2.12.10 OFF state

It is used to inform the acquisition module to stop the burst and/or acquisition on the sensor's channels.

The management of this state must be performed at application level.

2.12.11 Debounce states

The debounce is optional and is enabled/disabled using the different debounce counters parameters: **TSLPRM_DEBOUNCE_PROX**, **TSLPRM_DEBOUNCE_DETECT**, **TSLPRM_DEBOUNCE_RELEASE**, **TSLPRM_DEBOUNCE_CALIB**, **TSLPRM_DEBOUNCE_ERROR**

The debounce is off if the corresponding parameter is equal to zero.

2.12.12 Reading the current state

The current state can be obtained by using the functions:

For touchkey sensor:

- TSL_tkey_GetStateId()
- TSL_tkey_GetStateMask()

For linear and rotary sensor:

- TSL_linrot_GetStateId()
- TSL_linrot_GetStateMask()

The functions **TSL_tkey_IsChanged()** or **TSL_linrot_IsChanged()** allows to check if a sensor state has changed.

You can also directly read the state inside the sensor data structure:

```
if MyTKeys[0].p_Data->StateId == TSL_STATEID_DETECT)
```

2.12.13 Accessing a specific state

It is possible to enter directly in the calibration, OFF and “burst only” states. The “burst only” state consists in only bursting the electrode without performing acquisition on it. It can be used in specific cases to improve the robustness against noise.

Note: This feature is not available for STM8TL5x devices.

This is done by using the following functions:

For touchkey sensor:

- TSL_tkey_SetStateCalibration()
- TSL_tkey_SetStateOff()
- TSL_tkey_SetStateBurstOnly()

For linear and rotary sensor:

- TSL_linrot_SetStateCalibration()
- TSL_linrot_SetStateOff()
- TSL_linrot_SetStateBurstOnly()

2.13 Environment Change System (ECS)

2.13.1 Principle

Power supply voltage, temperature and air humidity, may induce a slow variation of the measured signal. The Environment Change System (ECS) is used to adapt the reference to these environment changes.

The ECS processing is based on an infinite response digital low pass filter of the first order (IIR filter):

$$Y(n) = K \times X(n) + (1 - K) \times Y(n - 1)$$

with:

Y = reference

X = acquisition value (last measurement)

K = coefficient.

The higher value is K, the faster is the response time. Two default K coefficients are available to obtain fast and slow responses.

The sampling frequency is programmable using a timing utility routine (see example below).

If the sensor is in PROX, DETECT or TOUCH states, the ECS is disabled for the duration of the detection timeout or for the duration of the touch (whichever ends first).

When the ECS is disabled, Yn=Yn-1

As soon as the recalibration times out or the detection ends, the filter is set active again.

2.13.2 Resources

The ECS functions are provided in the files:

- tsl_ecs.c
- tsl_ecs.h

The functions are:

- **TSL_ecs_Process()**: main function to be used by the user
- **TSL_ecs_CalcK()**: additional function
- **TSL_ecs_ProcessK()**: additional function

2.13.3 Parameters

- TSLPRM_ECS_K_FAST
- TSLPRM_ECS_K_SLOW
- TSLPRM_ECS_DELAY

2.13.4 Usage example

The ECS processing is usually performed in the main state machine at regular time intervals defined by the user. But it can be done also in interrupt routines. It must be performed after the sensors state machine is processed.

The ECS is activated only when all the sensors are in RELEASE, ERROR or OFF states, with at least one sensor in RELEASE state. It can also be delayed from milli-seconds to few seconds.

The ECS processing is performed on a group of sensors defined by the user. Different groups can be created and ECS applied on these groups with different K coefficients.

It's up to the user to decide the best thing to do for its application.

The simplest way is to call the **TSL_ecs_Process()** function in the main application loop using the default K coefficients defined in the configuration file:

```
TSL_ecs_Process(&MyObjGroup);
```

To call this functions at regular time intervals you can use the provide timing routine **TSL_tim_CheckDelay_ms()**.

Example with ECS executed every 100ms:

```
TSL_tTick_ms_T time_ECS_tick;
int main(void) {
  while (1) {
    ...
    // ECS every 100 ms
    if (TSL_tim_CheckDelay_ms(100, &time_ECS_tick) == TSL_STATUS_OK)
    {
      TSL_ecs_Process(&MyObjGroup);
    }
    ...
  }
}
```

The **TSL_ecs_ProcessK()** function allows to use a K coefficient different than the default value:

```
    if (TSL_tim_CheckDelay_ms(100, &time_ECS_tick) == TSL_STATUS_OK)
    {
      if ((MyObjGroup->StateMask & TSL_STATE_RELEASE_BIT_MASK) &&
          !(MyObjGroup->StateMask & TSL_STATEMASK_ACTIVE))
      {
        TSL_ecs_ProcessK(&MyObjGroup, 120);
      }
    }
```

2.14 Detection Exclusion System (DXS)

2.14.1 Principle

The DXS processing is used to prevent several sensors to be in the DETECT state at the same time. This could happen if the sensors are closed to each other or if their sensitivity is too high. This can be useful also in some applications to prevent the user to touch at the same time several sensors with "opposite" meaning (volume up and volume down for example).

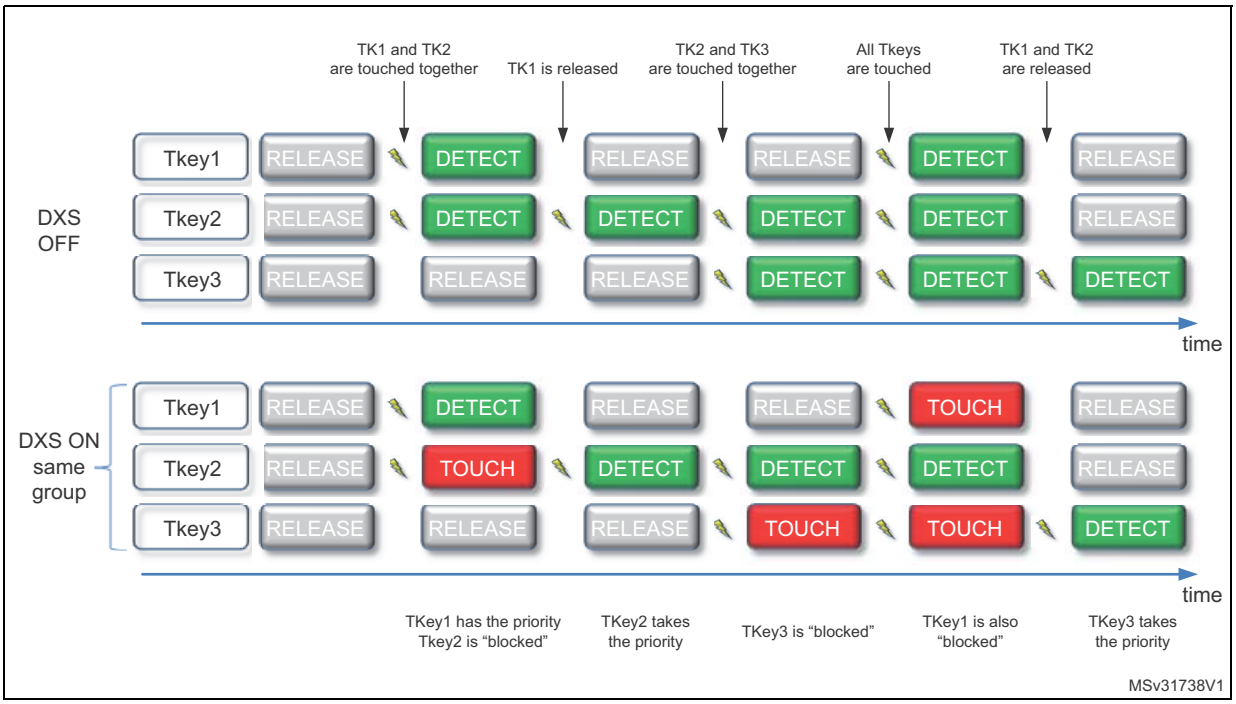
The first sensor in the group of sensors has the priority and enters in the DETECT state (with the DxSLock flag set). The other sensors are “blocked” and enter instead in the TOUCH state.

Note: A particular care must be taken when designing sensors that are shared between multiple DXS groups. The sensor that will be assigned in the DETECT state depends on the sensors position in the DXS groups and also on the order of the DXS groups processing. See the examples 1 and 2 for more detail.

The figure below illustrates the difference in behavior for a group of 3 sensors (touchkeys) when the DXS is OFF and ON. The three touchkeys are part of the same DXS group.

Note: The touchkeys can be replaced by a linear or a rotary sensor.

Figure 13. DXS principle



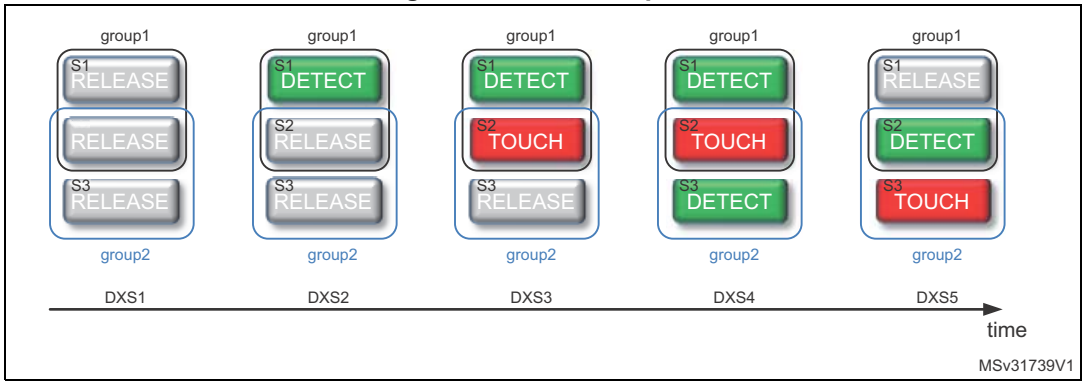
Example 1: 3 sensors with one shared between two groups.

In this example the group1 is composed of the two sensors s1 and s2 in this order and the group2 of the two sensors s2 and s3 in this order.

The DXS groups are processed in this order: group1 first and then group2.

We can see in the step DXS5 that the sensor 2 (s2) goes in DETECT state instead of the sensor 3 (s3). This is simply because s2 is placed first in the group2.

Figure 14. DXS example 1



Example 2: 4 sensors with one share between three groups.

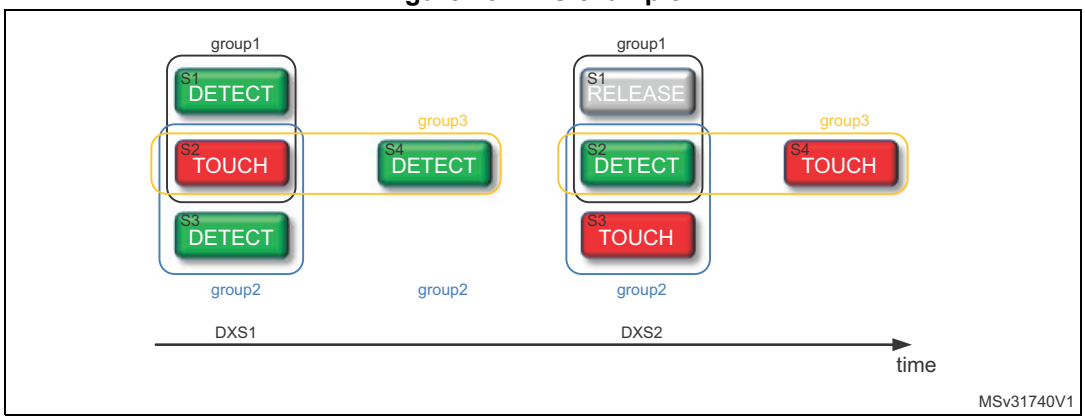
In this example the group1 is composed of the two sensors s1 and s2 in this order, the group2 of the two sensors s2 and s3 in this order and the group3 of the two sensors s2 and s4 in this order.

The DXS groups are processed in this order: group1 first, then group2 and finally group3.

We can see in the step DXS2 that the sensor 2 takes the priority over the sensors 3 and 4.

To summarize, the decision to be in DETECT state depends on the sensors placement inside the group and also on the order of the groups processing.

Figure 15. DXS example 2



2.14.2 Resources

The DXS functions are provided in the files:

- tsl_dxs.c
- tsl_dxs.h

The functions to use are:

- TSL_dxs_FirstObj()

2.14.3 Parameters

- TSLPRM_USE_DXS

2.14.4 Usage example

The DXS processing is performed usually in the main state machine but it can also be done in interrupt routines.

Warning: The DXS must be absolutely performed after the sensors state machine is processed, that is after the call to the TSL_obj_GroupProcess() function (see the main state machine for more details).

The DXS processing is performed on a **group of sensors** defined by the user. Different groups of DXS can be created.

It's up to the user to decide the best partitioning for his application.

Example:

```
int main(void) {
  while (1) {
    ...
    TSL_obj_GroupProcess(&MyObjGroup1);
    TSL_obj_GroupProcess(&MyObjGroup2);
    TSL_dxs_FirstObj(&MyObjGroup1);
    TSL_dxs_FirstObj(&MyObjGroup2);
    ...
  }
}
```

2.15 Detection Time Out (DTO)

2.15.1 Principle

The Detection Time Out (DTO) introduces a simple way to cope with water film and any obstacle that may come in contact with a sensor. It introduces a maximum duration for the 'detected' state of any sensor called the Detection Time Out (DTO).

After this period of time, the sensor is automatically recalibrated. This allows to make the sensor touch sensitive again, even if the obstacle or the liquid film is still present on the board.

This feature is application dependent and the time out must be tuned according to the user interface specifications.

The DTO is applied on the PROX, DETECT and TOUCH states and can be disabled.

2.15.2 Resources

The DTO functions are provided in the files:

- tsl_touchkey.c
- tsl_touchkey.h
- tsl_linrot.c
- tsl_linrot.h

The functions used by the DTO are:

- TSL_tkey_DTOGetTime()
- TSL_linrot_DTOGetTime()
- TSL_tim_CheckDelay_sec()

Note: The user doesn't need to call these functions to perform the DTO.

2.15.3 Parameters

- TSLPRM_DTO

2.15.4 Usage

The DTO is automatically performed inside the sensor state machine. The user doesn't need to call any function in the application code.

The DTO is disabled by writing zero in the **TSLPRM_DTO** parameter.

2.16 Noise filters

2.16.1 Principle

The STMTouch driver has been designed to facilitate the implementation of different noise filters. These filters can be used for many purpose and can range from very simple design to very complicated.

2.16.2 Resources

The filters are defined in the files:

- tsl_filter.c
- tsl_filter.h

Each filter is described by a function:

- **TSL_filt_MeasFilter()**: filter on measurement values
- **TSL_filt_DeltaFilter()**: filter on delta values

2.16.3 Parameters

There is no parameter for the filter module.

2.16.4 Usage

The filter functions can be called at anytime in the main application. In order to speed-up the execution time and to gain RAM space, the measure and delta filters are called by the **TSL_acq_BankGetResult()** function.

Examples:

```
// Apply a filter on the measures only
TSL_acq_BankGetResult(0, TSL_filt_MeasFilter, 0);
// Get the measures without applying any filter
TSL_acq_BankGetResult(0, 0, 0);
```

Note: The user can also create its own filter functions.

2.17 Timing management

2.17.1 Principle

The STMTouch driver needs an internal clock (“timing”), in particular for the ECS and DTO processings. This timing can be also used by the application layer for any purpose (LEDs blinking for example).

The timing process consists in incrementing a global variable at a regular interval. Different functions are then used to compare the current “time” and to check if a certain delay has elapsed.

2.17.2 Resources

The common timing routines are described in the files:

- tsl_time.c
- tsl_time.h

The initialization of the timing is made using hardware timer, systick, etc... and is implemented differently on each device. This is described in the files:

- tsl_time_<mcu>.c
- tsl_time_<mcu>.h

Functions:

- TSL_tim_ProcessIT()
- TSL_tim_CheckDelay_ms()
- TSL_tim_CheckDelay_sec()

2.17.3 Parameters

- TSLPRM_TICK_FREQ

2.17.4 Usage

The timing is started when the function **TSL_Init()** is called.

The function **TSL_tim_CheckDelay_ms()** can be used in the main application code to execute some code (for example the ECS) at a regular interval.

Example:

```
TSL_tTick_ms_T time_ECS_tick;
TSL_tTick_ms_T time_LED_tick;
int main(void) {
  TSL_Init(MyBanks); // The timing starts...
  while (1) {
    ...
    // Launch the ECS every 100 ms
    if (TSL_tim_CheckDelay_ms(100, &time_ECS_tick) == TSL_STATUS_OK)
    {
      TSL_ecs_Process(&MyObjGroup);
    }
    // Toggle LED every 500 ms
    if (TSL_tim_CheckDelay_ms(500, &time_LED_tick) == TSL_STATUS_OK)
    {
      ToggleLED();
    }
    ...
  }
}
```

2.18 Parameters

All the parameters are described in the **tsl_conf_<mcu>.h** file.

*Note: The **tsl_conf_<mcu>.h_TOADAPT** file present in the STMTouch_Driver/inc folder must be copied in the application project (inc folder) and adapted to your application.*

The structure **TSL_Params_T** is used to hold certain parameters that are common to all sensors. These parameters can be changed by the user while the application is running.

Parameters checking

All common parameters are verified (presence and value range) in the file:

- tsl_check_config.h

All device specific parameters are verified in the **tsl_check_config_<mcu>.h** file.

2.19 STM8L1xx devices

2.19.1 Acquisition

The STM8L1xx devices **hardware acquisition mode** (using two timers) is done in the files:

- tsl_acq_stm8l_hw.c
- tsl_acq_stm8l_hw.h

Warning: This acquisition mode is available for the STM8L15x Low-density devices only.

The STM8L1xx devices **software acquisition mode** is done in the files:

- tsl_acq_stm8l_sw.c
- tsl_acq_stm8l_sw.h

This acquisition is available for all STM8L1xx devices.

Note: The hardware acquisition mode is selected per default for the STM8L15x Low-density and devices. If you want to use the software acquisition mode you must add the following constant in the toolchain compiler preprocessor:

- TSLPRM_STM8L1XX_SW_ACQ

Functions used by the application layer and that are device dependent:

- TSL_acq_BankConfig()
- TSL_acq_BankStartAcq()
- TSL_acq_BankWaitEOC()
- TSL_acq_GetMeas()

The other functions in this file are for internal use and the user doesn't need to call them directly.

2.19.2 Timings

The STM8L1xx devices timing management is done in the files:

- tsl_time_stm8l.c
- tsl_time_stm8l.h

The **TIM4** is used to generate a timebase for the ECS and DTO modules.

Warning: The auto reload counter is calculated for a F_{CPU} equal to 16 MHz. If you use another F_{CPU} value in your application you must change the ARR value inside the TSL_tim_Init() function.

Functions used:

- TSL_tim_Init()

2.19.3 Parameters

The parameters specific to the STM8L1xx devices are described in the file:

- tsl_conf_stm8l.h

and are checked in the file:

- tsl_check_config_stm8l.h

2.19.4 Memory footprint

Conditions

- Cosmic STM8 C compiler 32K version v4.3.6
- Compiler options: +modsl0 -pxp +compact +split -pp
- Cosmic library not counted
- STMTouch driver default options: ECS=ON, DTO=OFF, PROX=OFF
- Each sensor has its own parameters, all parameters placed in RAM

The following tables summarize the memory footprint taken by the STMTouch driver with different STM8L devices, acquisition mode and sensors:

Table 6. STM8L101 memory footprint with software acquisition(1)

Channels	Banks	Sensors	ROM (Kbytes)	RAM (bytes)	Specific options
3	2	3 TKeys	~5.3	~160	ZONE=OFF DXS=ON

1. The content of this table is provided for information purposes only.

Table 7. STM8L15x memory footprint with hardware acquisition(1)

Channels	Banks	Sensors	ROM (Kbytes)	RAM (bytes)	Specific options
1	1	1 TKey	~5.2	~140	ZONE=OFF DXS=OFF
10	2	10 TKeys	~5.4	~300	ZONE=ON DXS=ON
16	2	10 TKeys 1 Linear-3ch 1 Rotary-3ch	~7.4	~450	ZONE=ON DXS=ON

1. The content of this table is provided for information purposes only.

Table 8. STM8L15x memory footprint with software acquisition(1)

Channels	Banks	Sensors	ROM (Kbytes)	RAM (bytes)	Specific options
1	1	1 TKey	~4.6	~130	ZONE=OFF DXS=OFF
10	2	10 TKeys	~4.8	~280	ZONE=OFF DXS=ON
16	2	10 TKeys 1 Linear-3ch 1 Rotary-3ch	~6.9	~430	ZONE=OFF DXS=ON

1. The content of this table is provided for information purposes only.

2.19.5 MCU resources

The tables below show the peripherals that are used by the STMTouch driver on STM8L1xx devices. Care must be taken when using them to avoid any unwanted behavior.

Table 9. MCU resources used on STM8L1xx with hardware acquisition

Peripheral	Function
GPIOs	Acquisition
TIM4	Time base for ECS and DTO
TIM2, TIM3	Acquisition
Routing interface	Acquisition

Table 10. MCU resources used on STM8L1xx with software acquisition

Peripheral	Function
GPIOs	Acquisition
TIM4	Time base for ECS and DTO
Routing interface	Acquisition

2.19.6 STM8L available touch-sensing channels

The tables below provide an overview of the available touch sensing channels for the STM8L1xx devices.

Note: The following tables are not restrictive in term of part numbers supported by the STMTouch driver. The STMTouch driver can be used on any new device that may become available as part of ST microcontrollers portfolio. Please contact your ST representative for support.

Note: For n available pins in an I/O group, one pin is used as sampling capacitor and n-1 pins are used as channels.

The I/O group cannot be used if the number of available pins in less or equal to one.

Table 11. Available touch-sensing channels for STM8L101

Subfamily			STM8L101									
Packages			TSSOP20 / UFQFPN20				UFQFPN28			UFQFPN32 / LQFP32		
Part numbers			STM8L101F[23]U				STM8L101G[23]U			STM8L101K3[UT]		
				STM8L101F[23]P								
Analog I/O group	Gx_IOy	GPIO	Pin TSSOP	Pin UFQFPN	Number of available pins	Usage	Pin	Number of available pins	Usage	Pin	Number of available pins	Usage
Group1	G1_IO1	PB0	10	7	3	2 channels with 1 sampling capacitor	12	4	3 channels with 1 sampling capacitor	13	4	3 channels with 1 sampling capacitor
	G1_IO2	PB1	11	8			13			14		
	G1_IO3	PD0	9	6			8			9		
	G1_IO4	PD1	-	-			9			10		
Group2	G2_IO1	PB2	12	9	2	1 channel with 1 sampling capacitor	14	4	3 channels with 1 sampling capacitor	15	4	3 channels with 1 sampling capacitor
	G2_IO2	PB3	13	10			15			16		
	G2_IO3	PD2	-	-			10			11		
	G2_IO4	PD3	-	-			11			12		
Maximum number of channels			3 with 2 sampling capacitors				6 with 2 sampling capacitors			6 with 2 sampling capacitors		

Table 12. Available touch-sensing channels for STM8L15x / STM8L16x (table 1/2)

Subfamily			STM8L151F				STM8L151G				STM8L151K		
Packages			UFQFPN20 / TSSOP20				UFQFPN28 / WLCSP28				UFQFPN32 / LQFP32		
Part numbers			STM8L151F[23]U(1) (UFQFPN)				STM8L151G[346]U(1) (UFQFPN)				STM8L152K[46][UT]		
				STM8L151F[23]P(1) (TSSOP)				STM8L151G[46]Y (WLCSP)					
Analog I/O group	Gx_IOy	GPIO	Pin	Pin	Number of available pins	Usage	Pin	Pin	Number of available pins	Usage	Pin	Number of available pins	Usage
Group 1	G1_IO1	PA6	-	-	0	cannot be used for touch sensing	-	-	2	1 channel with 1 sampling capacitor	6	3	2 channels with 1 sampling capacitor
	G1_IO2	PA5	-	-			5	D4			5		
	G1_IO3	PA4	-	-			4	D3			4		
	G1_IO4	PA7	-	-			-	-			-		
Group 2	G2_IO1	PC7	-	-	1	cannot be used for touch sensing	-	-	2	1 channel with 1 sampling capacitor	-	2	1 channel with 1 sampling capacitor
	G2_IO2	PC4	17	20			25	C2			29		
	G2_IO3	PC3	-	-			24	A2			28		
	G2_IO4	PE7	-	-			-	-			-		
Group 3	G3_IO1	PC2	-	-	0	cannot be used for touch sensing	23	B2	1	cannot be used for touch sensing	27	3	2 channels with 1 sampling capacitor
	G3_IO2	PD7	-	-			-	-			24		
	G3_IO3	PD6	-	-			-	-			23		
Group 4	G4_IO1	PD5	-	-	1	cannot be used for touch sensing	-	-	2	1 channel with 1 sampling capacitor	22	3	2 channels with 1 sampling capacitor
	G4_IO2	PD4	-	-			20	C1			21		
	G4_IO3	PB7	14	17			19	E1			20		

Table 12. Available touch-sensing channels for STM8L15x / STM8L16x (table 1/2) (continued)

Subfamily			STM8L151F				STM8L151G				STM8L151K		
Packages			UFQFPN20 / TSSOP20				UFQFPN28 / WLCSP28				UFQFPN32 / LQFP32		
Part numbers			STM8L151F[23]U(1) (UFQFPN)	STM8L151F[23]P(1) (TSSOP)			STM8L151G[346]U(1) (UFQFPN)	STM8L151G[46]Y (WLCSP)			STM8L152K[46][UT]		
Analog I/O group	Gx_IOy	GPIO	Pin	Pin	Number of available pins	Usage	Pin	Pin	Number of available pins	Usage	Pin	Number of available pins	Usage
Group 5	G5_IO1	PB6	13	16	3	2 channels with1 sampling capacitor	18	F1	3	2 channels with 1 sampling capacitor	19	3	2 channels with 1 sampling capacitor
	G5_IO2	PB5	12	15			17	D1			18		
	G5_IO3	PB4	11	14			16	D2			17		
Group 6	G6_IO1	PB3	10	13	3	2 channels with 1 sampling capacitor	15	E2	3	2 channels with 1 sampling capacitor	16	3	2 channels with 1 sampling capacitor
	G6_IO2	PB2	9	12			14	F2			15		
	G6_IO3	PB1	8	11			13	G1			14		
Group 7	G7_IO1	PB0	7	10	1	cannot be used for touch sensing	12	E3	3	2 channels with 1 sampling capacitor	13	3	2 channels with 1 sampling capacitor
	G7_IO2	PD3	-	-			11	F3			12		
	G7_IO3	PD2	-	-			10	E4			11		
	G7_IO4	PE3	-	-			-	-			-		
Group 8	G8_IO1	PD1	-	-	1	cannot be used for touch sensing	9	G2	2	1 channel with 1 sampling capacitor	10	1	cannot be used for touch sensing
	G8_IO2	PD0	6	9			8	G3			-		
	G8_IO3	PE5	-	-			-	-			-		
	G8_IO4	PE4	-	-			-	-			-		
Maximum number of channels			4 channels with 2 sampling capacitors				10 channels with 7 sampling capacitors				13 channels with 7 sampling capacitors		

1. The product has an hardware acceleration cell for touch sensing.

Table 13. Available touch-sensing channels for STM8L15x / STM8L16x (table 2/2)

Subfamily			STM8L151K			STM8L151C medium/medium+/high density STM8L151R/M STM8L152C/R/M STM8L162R/M					STM8L151C low density		
Packages			UFQFPN32 / LQFP32			UFQFPN48 / LQFP48 / LQFP64 / LQFP80					LQFP48		
Part numbers			STM8L151K3U(1) STM8L151K[46][UT]			STM8L151C[468][UT] STM8L152C[468][UT] (48 pins)	STM8L151R[68]T STM8L152R[68]T STM8L162R8T (64 pins)	STM8L151M8T STM8L152M8T STM8L162M8T (80 pins)			STM8L151C3T(1)		
Analog I/O group	Gx_IOy	GPIO	Pin	Number of available pins	Usage	Pin	Pin	Pin	Number of available pins	Usage	Pin	Number of available pins	Usage
Group 1	G1_IO1	PA6	6	3	2 channels with 1 sampling capacitor	7	7	11	3	2 channels with 1 sampling capacitor	7	4	3 channels with 1 sampling capacitor
	G1_IO2	PA5	5			6	6	10			6		
	G1_IO3	PA4	4			5	5	9			5		
	G1_IO4	PA7	-			(2)	(2)	(2)			8		
Group 2	G2_IO1	PC7	-	2	1 channel with 1 sampling capacitor	46	62	74	3	2 channels with 1 sampling capacitor	46	4	3 channels with 1 sampling capacitor
	G2_IO2	PC4	29			43	59	71			43		
	G2_IO3	PC3	28			42	58	70			42		
	G2_IO4	PE7	-			(2)	(2)	(2)			48		

Table 13. Available touch-sensing channels for STM8L15x / STM8L16x (table 2/2) (continued)

Subfamily			STM8L151K			STM8L151C medium/medium+/high density STM8L151R/M STM8L152C/R/M STM8L162R/M					STM8L151C low density		
Packages			UFQFPN32 / LQFP32			UFQFPN48 / LQFP48 / LQFP64 / LQFP80					LQFP48		
Part numbers			STM8L151K3U(1) STM8L151K[46][UT]			STM8L151C[468][UT] STM8L152C[468][UT] (48 pins)	STM8L151R[68]T STM8L152R[68]T STM8L162R8T (64 pins)	STM8L151M8T STM8L152M8T STM8L162M8T (80 pins)			STM8L151C3T(1)		
Analog I/O group	Gx_IOy	GPIO	Pin	Number of available pins	Usage	Pin	Pin	Pin	Number of available pins	Usage	Pin	Number of available pins	Usage
Group 3	G3_IO1	PC2	27	3	2 channels with 1 sampling capacitor	41	57	69	3	2 channels with 1 sampling capacitor	41	3	1 channel with 1 sampling capacitor
	G3_IO2	PD7	24			36	48	60			36		
	G3_IO3	PD6	23			35	47	59			35		
Group 4	G4_IO1	PD5	22	3	2 channels with 1 sampling capacitor	34	46	58	3	2 channels with 1 sampling capacitor	34	3	2 channels with 1 sampling capacitor
	G4_IO2	PD4	21			33	45	57			33		
	G4_IO3	PB7	20			31	38	46			31		
Group 5	G5_IO1	PB6	19	3	2 channels with 1 sampling capacitor	30	37	45	3	2 channels with 1 sampling capacitor	30	3	2 channels with 1 sampling capacitor
	G5_IO2	PB5	18			29	36	44			29		
	G5_IO3	PB4	17			28	35	43			28		

Table 13. Available touch-sensing channels for STM8L15x / STM8L16x (table 2/2) (continued)

Subfamily			STM8L151K			STM8L151C medium/medium+/high density STM8L151R/M STM8L152C/R/M STM8L162R/M					STM8L151C low density		
Packages			UFQFPN32 / LQFP32			UFQFPN48 / LQFP48 / LQFP64 / LQFP80					LQFP48		
Part numbers			STM8L151K3U(1) STM8L151K[46][UT]			STM8L151C[468][UT] STM8L152C[468][UT] (48 pins)	STM8L151R[68]T STM8L152R[68]T STM8L162R8T (64 pins)	STM8L151M8T STM8L152M8T STM8L162M8T (80 pins)			STM8L151C3T(1)		
Analog I/O group	Gx_IOy	GPIO	Pin	Number of available pins	Usage	Pin	Pin	Pin	Number of available pins	Usage	Pin	Number of available pins	Usage
Group 6	G6_IO1	PB3	16	3	2 channels with 1 sampling capacitor	27	34	42	3	2 channels with 1 sampling capacitor	27	3	2 channels with 1 sampling capacitor
	G6_IO2	PB2	15			26	33	41			26		
	G6_IO3	PB1	14			25	32	40			25		
Group 7	G7_IO1	PB0	13	3	2 channels with 1 sampling capacitor	24	31	39	3	2 channels with 1 sampling capacitor	24	4	3 channels with 1 sampling capacitor
	G7_IO2	PD3	12			23	28	32			23		
	G7_IO3	PD2	11			22	27	31			22		
	G7_IO4	PE3	-			(2)	(2)	(2)			17		

Table 13. Available touch-sensing channels for STM8L15x / STM8L16x (table 2/2) (continued)

Subfamily			STM8L151K			STM8L151C medium/medium+/high density STM8L151R/M STM8L152C/R/M STM8L162R/M					STM8L151C low density		
Packages			UFQFPN32 / LQFP32			UFQFPN48 / LQFP48 / LQFP64 / LQFP80					LQFP48		
Part numbers			STM8L151K3U(1) STM8L151K[46][UT]			STM8L151C[468][UT] STM8L152C[468][UT] (48 pins)	STM8L151R[68]T STM8L152R[68]T STM8L162R8T (64 pins)	STM8L151M8T STM8L152M8T STM8L162M8T (80 pins)			STM8L151C3T(1)		
Analog I/O group	Gx_IOy	GPIO	Pin	Number of available pins	Usage	Pin	Pin	Pin	Number of available pins	Usage	Pin	Number of available pins	Usage
Group 8	G8_IO1	PD1	10	2	1 channel with 1 sampling capacitor	21	26	30	3	2 channels with 1 sampling capacitor	21	4	3 channels with 1 sampling capacitor
	G8_IO2	PD0	9			20	25	29			20		
	G8_IO3	PE5	-			19	24	28			19		
	G8_IO4	PE4	-			(2)	(2)	(2)			18		
Maximum number of channels			14 channels with 8 sampling capacitors			16 channels with 8 sampling capacitors					20 channels with 8 sampling capacitors		

1. The product has an hardware acceleration cell for touch sensing.
2. This IO does not belong to the analog IO group.

2.19.7 Hardware implementation example

Figure 16 shows an example of hardware implementation on STM8L1xx devices.

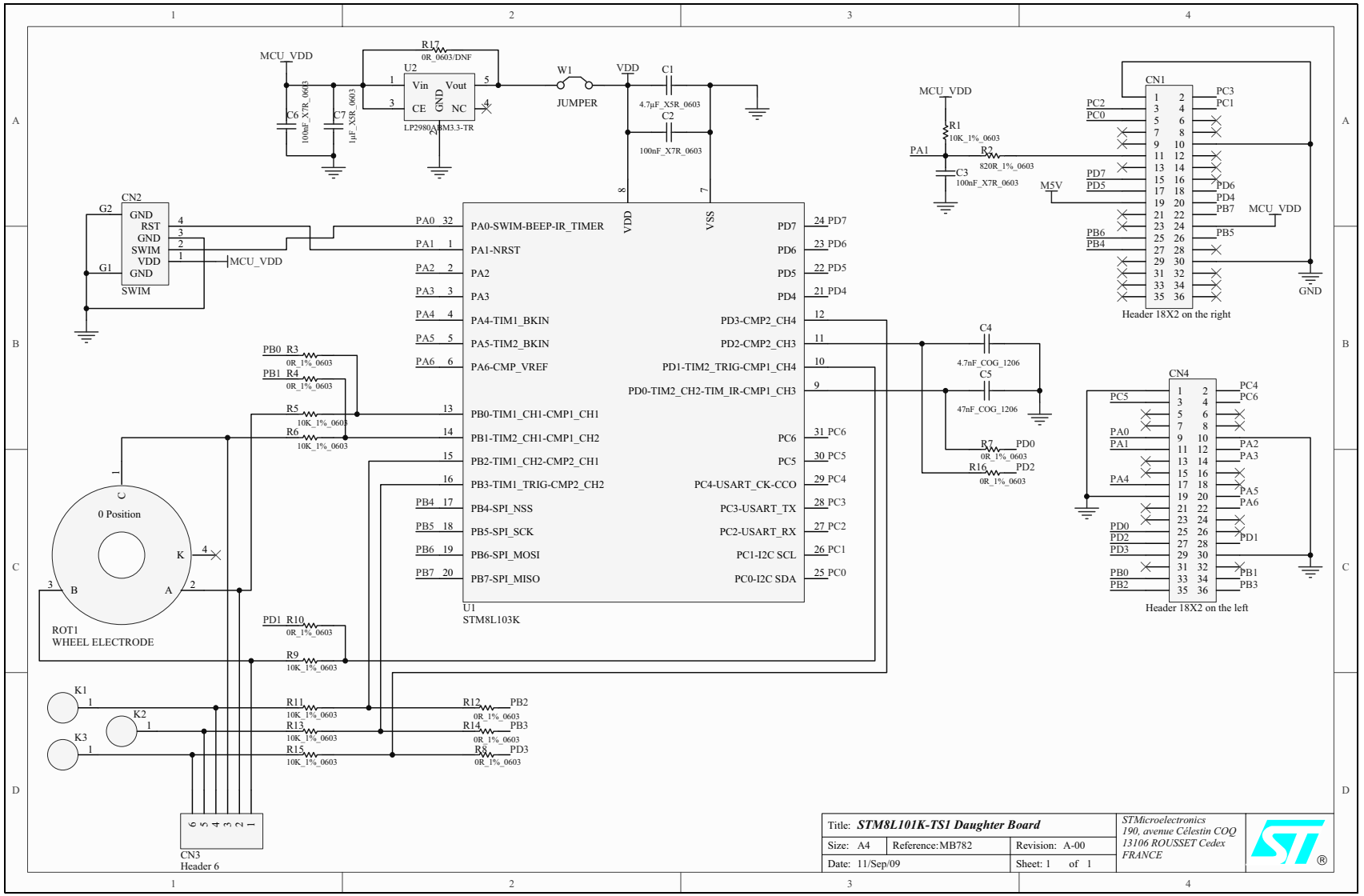
Recommendations to increase the noise immunity on the PCB

To ensure a correct operation in noisy environment, the floating nets must be avoided (tracks, copper elements, conductive frames, etc...).

As a consequence:

- All unused touch controller I/Os must be either configured to output push-pull low or externally tied to GND.
- The parameter TSLPRM_IODEF should also be configured to the output push-pull low state.
- We recommend to drive the sampling capacitor common node using a standard I/O of the touch controller configured in output push-pull low mode.

Figure 16. STM8L101 hardware implementation example



2.20 STM8TL5x devices

2.20.1 Acquisition

The STM8TL5x devices acquisition is done in the files:

- tsl_acq_stm8tl5x.c
- tsl_acq_stm8tl5x.h

Functions used by the application layer and that are device dependent:

- TSL_acq_BankConfig()
- TSL_acq_BankStartAcq()
- TSL_acq_BankWaitEOC()
- TSL_acq_GetMeas()

The other functions in this file are for internal use and the user doesn't need to call them directly.

2.20.2 Timings

The STM8TL5x devices timing management is done in the files:

- tsl_time_stm8tl5x.c
- tsl_time_stm8tl5x.h

The **TIM4** is used to generate a timebase for the ECS and DTO modules.

Warning: **The auto reload counter is calculated for a F_{CPU} equal to 16 MHz. If you use another F_{CPU} value in your application you must change the ARR value inside the TSL_tim_Init() function.**

Functions used

- TSL_tim_Init()
- TSL_Timer_ISR()

2.20.3 Parameters

The parameters specific to the STM8TL5x devices are described in the file:

- tsl_conf_stm8tl5x.h

and are checked in the file:

- tsl_check_config_stm8tl5x.h

2.20.4 Memory footprint

Conditions

- Cosmic STM8 C compiler 32K version v4.3.6
- Compiler options: +modsl0 -pxp +compact +split -pp
- Cosmic library not counted
- STMTouch driver default options: ECS=ON, DTO=ON, ZONE=OFF, DXS=ON (excepted if only one sensor is used)
- Each sensor has its own parameters, all parameters placed in RAM

The following tables summarize the memory footprint with different configurations:

Table 14. STM8TL5x memory footprint without proximity(1)

Channels	Banks	Sensors	ROM (Kbytes)	RAM (bytes)
1	1	1 TKey	~4.3	~70
3	1	3 TKeys	~4.4	~110
19	4	19 TKeys	~4.7	~440
26	6	16 Linears-1ch 2 Linears-5ch	~6.2	~680
26	6	16 TKeys 2 Linears-5ch	~7.8	~570

1. The content of this table is provided for information purposes only.

Table 15. STM8TL5x memory footprint with proximity(1)

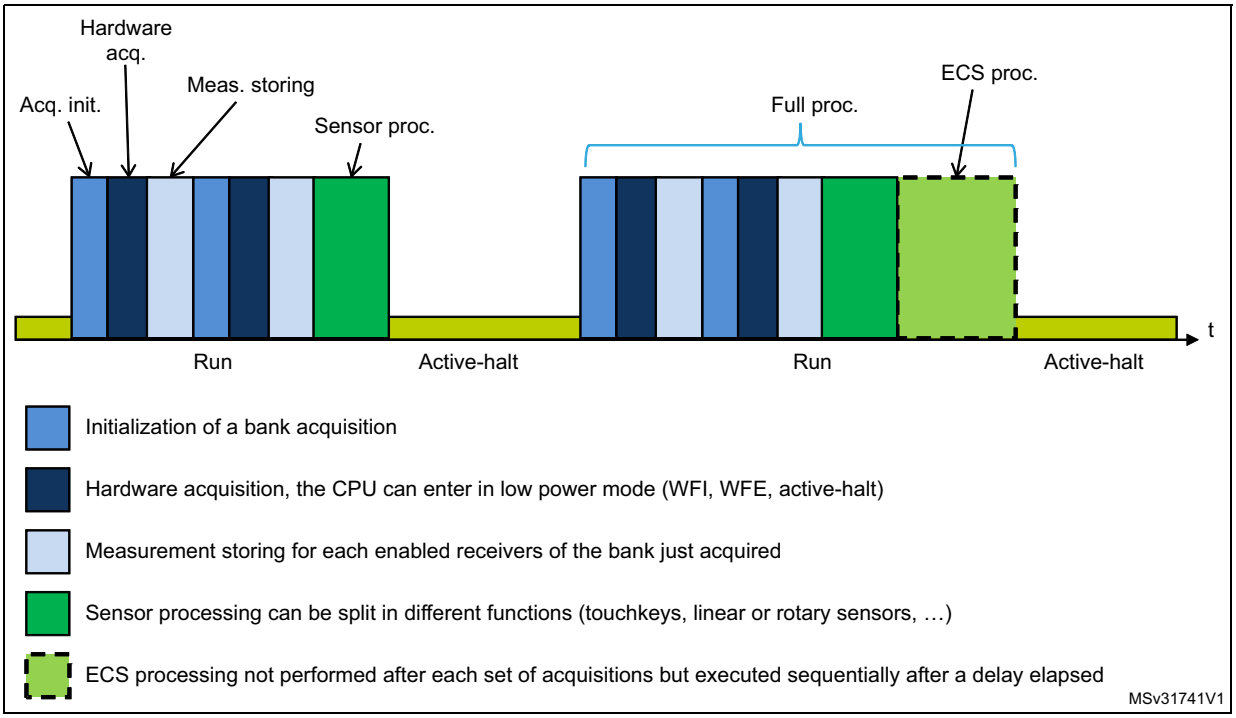
Channels	Banks	Sensors	ROM (Kbytes)	RAM (bytes)
1	1	1 TKey	~5.2	~70
3	1	3 TKeys	~5.4	~110
19	4	19 TKeys	~5.6	~490
26	6	16 Linears-1ch 2 Linears-5ch	~7.0	~730
26	6	16 TKeys 2 Linears-5ch	~9.5	~620

1. The content of this table is provided for information purposes only.

2.20.5 Acquisition timings

The following figure shows the **simplified** sequencing for a 2 bank acquisition.

Figure 17. Simplified acquisition sequencing



Conditions

- Cosmic STM8 C compiler 32K version v4.3.6
- Compiler options: +modsl0 -pxp +split -pp
- STMTouch driver default options: ECS=ON, DTO=ON, DXS=ON (excepted if one channel is used)
- Each sensor has its own parameters, all parameters placed in RAM

The following table summarizes the timings measured for the different acquisition sequences:

Table 16. STM8TL5x acquisition timings(1)

Process	Function	Conditions	Duration (us)	Comment
Startup before main()		Any	~60000	Compiler dependent
TSL initialization	TSL_obj_GroupInit() TSL_Init()	3 touchkeys / 1bank	~7500	Time before the driver is ready to report a touch including the calibration
		19 touchkeys / 4banks	~30000	
Acquisition Initialization	TSL_acq_BankConfig()	Any	66	This must be repeated for each bank

Table 16. STM8TL5x acquisition timings(1) (continued)

Process	Function	Conditions	Duration (us)	Comment
Hardware acquisition	None	UP = 1, PASS = 1	~125	Reference target set to 500 whatever the number of enabled receivers PXS_CKCR1 = 0x70 (i.e. HSI_PXS = 16 MHz)
		UP = 1, PASS = 2	~187	
		UP = 2, PASS = 2	~250	
		UP = 3, PASS = 3	~350	
Measurement storing	TSL_acq_BankGetResult()	3 receivers	81	
		5 receivers	135	
		by enabled receiver	~27	
Touchkey processing	TSL_obj_GroupProcess()	3 touchkeys	63	One object group
		19 touchkeys	460	Two object groups
Full processing	TSL_action()	3 touchkeys	302	Including hardware acquisition but without the ECS which is executed each 100 ms
		19 touchkeys	1680	
ECS processing	TSL_ecs_Process()	3 touchkeys	220	Not performed after each acquisition but on scheduling
		19 touchkeys	1400	
		by touchkey	~75	

1. The content of this table is provided for information purposes only.

2.20.6 MCU resources

The table below shows the peripherals that are used by the STMTouch driver on STM8TL5x devices. Care must be taken when using them to avoid any unwanted behavior.

Table 17. STM8TL5x MCU resources used

Peripheral	Function
GPIOs	Acquisition
8-bit timer (TIM 4)	Time base for ECS and DTO
ProxSense (PXS)	Acquisition

2.20.7 STM8TL5x available touch-sensing channels

The table below provides an overview of the available touch sensing channels for the STM8TL5x devices.

Note: The following table is not restrictive in term of part numbers supported by the STMTouch driver. The STMTouch driver can be used on any new device that may become available as part of ST microcontrollers portfolio. Please contact your ST representative for support.

Table 18. Available touch-sensing channels for STM8TL5x

Subfamily			STM8TL5x							
Packages			TSSOP20			UFQFPN28			UFQFPN48	
Part numbers			STM8TL52F4P	STM8TL53F4P		STM8TL52G4U	STM8TL53G4U		STM8TL53C4U	
PXS function		GPIO	Pin	Pin	Usage	Pin	Pin	Usage	Pin	Usage
Receiver A(1)	RX0a	-	11	11	5 Receivers / Transmitters	10	10	8 Receivers / Transmitters	13	10 Receivers / Transmitters
	RX1a	-	12	12		11	11		15	
	RX2a	-	13	13		12	12		17	
	RX3a	-	-	-		13	13		19	
	RX4a	-	-	-		14	14		21	
	RX5a	-	-	-		15	15		23	
	RX6a	-	14	14		16	16		25	
	RX7a	-	15	15		17	17		27	
	RX8a	-	-	-		-	-		29	
	RX9a	-	-	-		-	-		31	
Receiver B(1)	RX0b	-	-	-	0 Receivers / Transmitters	-	-	0 Receivers / Transmitters	14	10 Receivers / Transmitters
	RX1b	-	-	-		-	-		16	
	RX2b	-	-	-		-	-		18	
	RX3b	-	-	-		-	-		20	
	RX4b	-	-	-		-	-		22	
	RX5b	-	-	-		-	-		24	
	RX6b	-	-	-		-	-		26	
	RX7b	-	-	-		-	-		28	
	RX8b	-	-	-		-	-		30	
	RX9b	-	-	-		-	-		32	

Table 18. Available touch-sensing channels for STM8TL5x (continued)

Subfamily			STM8TL5x							
Packages			TSSOP20			UFQFPN28			UFQFPN48	
Part numbers			STM8TL52F4P	STM8TL53F4P		STM8TL52G4U	STM8TL53G4U		STM8TL53C4U	
PXS function		GPIO	Pin	Pin	Usage	Pin	Pin	Usage	Pin	Usage
Transmitter	TX0	PD0	16	16	STM8TL52F4P: 2 Transmitters STM8TL53F4P: 5 Transmitters	18	18	STM8TL52G4U: 2 Transmitters STM8TL53G4U: 9 Transmitters	33	15 Transmitters
	TX1	PD1	17	17		19	19		34	
	TX2	PD2	-	-		20(2)	20		35	
	TX3	PD3	-	-		21(2)	21		36	
	TX4	PD4	18(2)	18		22(2)	22		39	
	TX5	PD5	19(2)	19		23(2)	23		40	
	TX6	PD6	20(2)	20		24(2)	24		41	
	TX7	PD7	-	-		27(2)	27		42	
	TX8	PB0	-	-		28(2)	28		43	
	TX9	PB1	-	-		-	-		44	
	TX10	PB2	-	-		-	-		45	
	TX11	PB3	-	-		-	-		46	
	TX12	PB4	-	-		-	-		47	
	TX13	PB5	-	-		-	-		48	
	TX14	PB6	-	-		-	-		1	
Maximum number of channels			STM8TL52F4P: 12 channels with a 4RX*3TX matrix STM8TL53F4P: 25 channels with a 5RX*5TX matrix			STM8TL52G4U: 25 channels with a 5RX*5TX matrix STM8TL53G4U: 72 channels with a 8RX*9TX matrix			300 channels with a 20RX*15TX matrix	

1. The receivers can also be used as transmitters. This is used to define the square matrix to address the maximum number of channels (please refer to product datasheet for further information).
2. On STM8TL52 devices, this GPIO is present but does not support the PXS alternate function.

2.20.8 Hardware implementation example

Figure 18. shows an example of hardware implementation on STM8TL5x devices.

Recommendations to increase the noise immunity on the PCB:

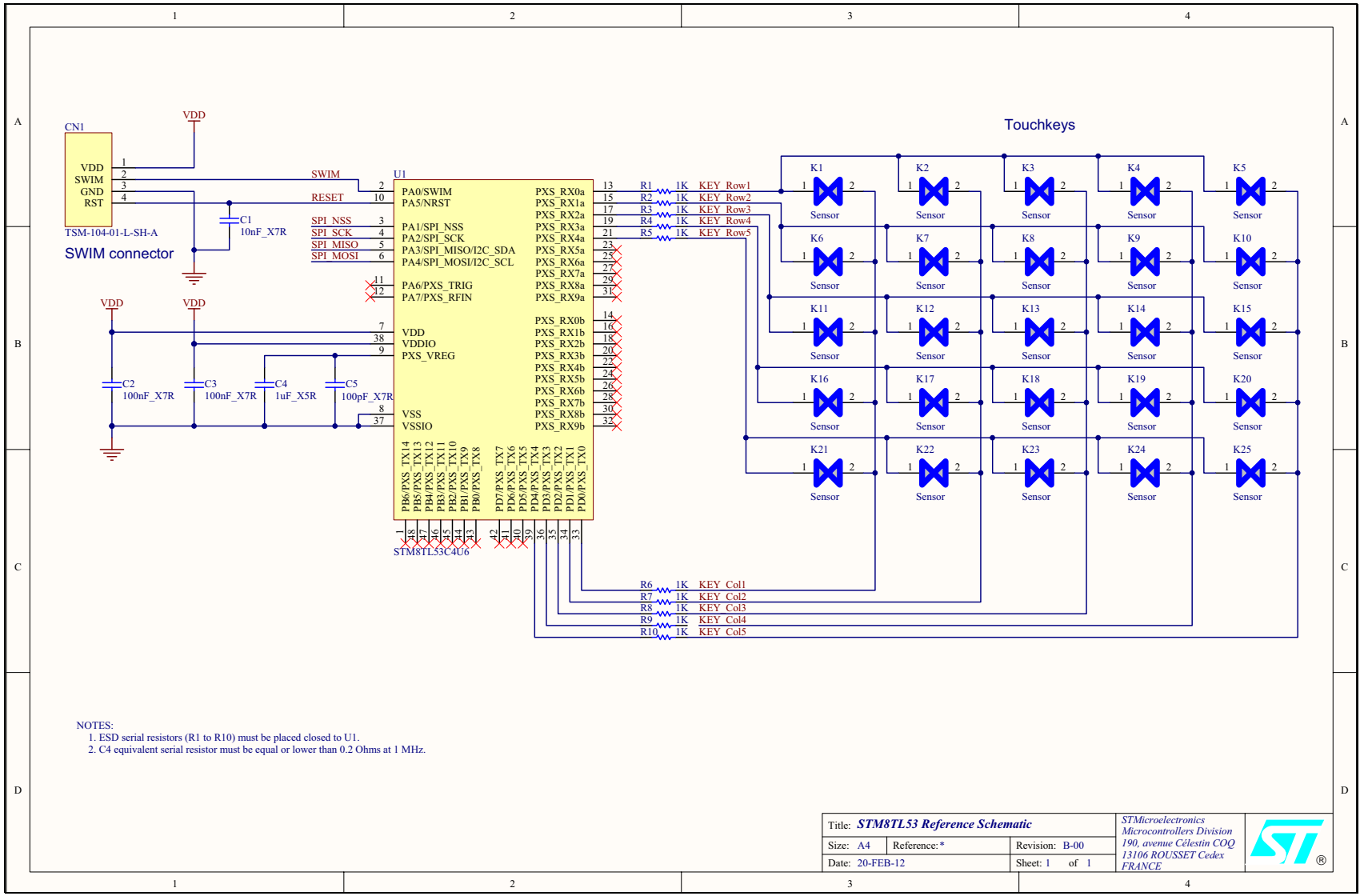
To ensure a correct operation in noisy environment, the floating nets must be avoided (tracks, copper elements, conductive frames, etc...).

As a consequence:

All unused touch controller I/Os must be either configured to output push-pull low or externally tied to GND.

The parameters TSLPRM_PXS_INACTIVE_TX and TSLPRM_PXS_INACTIVE_RX should also be configured to the Grounded state.

Figure 18. STM8TL5x hardware implementation example



2.21 STM32F0xx devices

2.21.1 Acquisition

The STM32F0xx devices acquisition is done in the files:

- tsl_acq_stm32f0xx.c
- tsl_acq_stm32f0xx.h

Functions used by the application layer and that are device dependent:

- TSL_acq_BankConfig()
- TSL_acq_BankStartAcq()
- TSL_acq_BankWaitEOC()
- TSL_acq_GetMeas()

The other functions in this file are for internal use and the user doesn't need to call them directly.

2.21.2 Timings

The STM32F0xx devices timing management is done in the files:

- tsl_time_stm32f0xx.c
- tsl_time_stm32f0xx.h

The **systick** is used to generate a timebase for the ECS and DTO modules.

Functions used:

- TSL_tim_Init()

2.21.3 Parameters

The parameters specific to the STM32F0xx devices are described in the file:

- tsl_conf_stm32f0xx.h

and are checked in the file:

- tsl_check_config_stm32f0xx.h

2.21.4 Memory footprint

Conditions

- IAR ANSI C/C++ Compiler V6.40.1 for Arm®
- Compiler options: optimization high-balanced
- IAR library not counted
- STMTouch driver default options: ECS=ON, DTO=ON, ZONE=OFF, DXS=ON (excepted if only one sensor is used)
- Each sensor has its own parameters, all parameters placed in RAM

The following tables summarize the memory footprint with different configurations:

Table 19. STM32F0xx memory footprint without proximity(1)

Channels	Banks	Sensors	ROM (Kbytes)	RAM (bytes)
3	3	3 TKeys	~3.9	~130
3	3	1 Linear-3ch	~4.9	~120
15	6	9 TKeys 1 Linear-3ch 1 Rotary-3ch	~7.7	~350

1. The content of this table is provided for information purposes only.

Table 20. STM32F0xx memory footprint with proximity(1)

Channels	Banks	Sensors	ROM (Kbytes)	RAM (bytes)
3	3	3 TKeys	~4.7	~140
3	3	1 Linear-3ch	~5.8	~130
15	6	9 TKeys 1 Linear-3ch 1 Rotary-3ch	~9.5	~390

1. The content of this table is provided for information purposes only.

2.21.5 MCU resources

The table below shows the peripherals that are used by the STMTouch driver on STM32F0xx devices. Care must be taken when using them to avoid any unwanted behavior.

Table 21. STM32F0xx MCU resources used

Peripheral	Function
GPIOs	Acquisition
Systick	Time base for ECS and DTO
Touch sense controller (TSC)	Acquisition

2.21.6 STM32F0xx available touch-sensing channels

The tables below provide an overview of the available touch sensing channels for the STM32F0xx devices.

Note: The following tables are not restrictive in term of part numbers supported by the STMTouch driver. The STMTouch driver can be used on any new device that may become available as part of ST microcontrollers portfolio. Please contact your ST representative for support.

Note: For n available pins in an I/O group, one pin is used as sampling capacitor and n-1 pins are used as channels.

The I/O group cannot be used if the number of available pins in less or equal to one.

Table 22. Available touch sensing channels for STM32F042

Subfamily			STM32F042															
Packages			TSSOP20			UFQFPN28			LQFP32 / UFQFPN32				WLCSP36			LQFP48 / UFQFPN48		
Flash memory size			4=16K, 6=32K															
Part numbers			STM32F042F[46]			STM32F042G[46]			STM32F042K[46]				STM32F042T[46]			STM32F042C[46]		
Analog I/O group	Gx_IOy	GPIO	Pin	Number of available pins	Usage	Pin	Number of available pins	Usage	Pin LQFP	Pin UFQFPN	Number of available pins	Usage	Pin	Number of available pins	Usage	Pin	Number of available pins	Usage
Group 1	G1_IO1	PA0	6	4	3 channels with 1 sampling capacitor	6	4	3 channels with 1 sampling capacitor	6	6	4	3 channels with 1 sampling capacitor	F6	4	3 channels with 1 sampling capacitor	10	4	3 channels with 1 sampling capacitor
	G1_IO2	PA1	7			7			7	7			D4			11		
	G1_IO3	PA2	8			8			8	8			E4			12		
	G1_IO4	PA3	9			9			9	9			F5			13		
Group 2	G2_IO1	PA4(1)	10	4	3 channels with 1 sampling capacitor	10	4	3 channels with 1 sampling capacitor	10	10	4	3 channels with 1 sampling capacitor	C3	4	3 channels with 1 sampling capacitor	14	4	3 channels with 1 sampling capacitor
	G2_IO2	PA5(1)	11			11			11	11			D3			15		
	G2_IO3	PA6	12			12			12	12			E3			16		
	G2_IO4	PA7	13			13			13	13			F4			17		
Group 3	G3_IO1	-	-	1	Cannot be used for touch sensing	-	2	1 channel with 1 sampling capacitor	-	-	2/3	1/2 channels with 1 sampling capacitor	-	3	2channels with 1 sampling capacitor	-	3	2 channels with 1 sampling capacitor
	G3_IO2	PB0	-			14			14	14			F3			18		
	G3_IO3	PB1	14			15			15	15			F2			19		
	G3_IO4	PB2	-			-			-	16			C2			20		
Group 4	G4_IO1	PA9	17(2)	2	1 channel with 1 sampling capacitor	19(2)	2	1 channel with 1 sampling capacitor	19	19	4	3 channels with 1 sampling capacitor	D1	4	3 channels with 1 sampling capacitor	30	4	3 channels with 1 sampling capacitor
	G4_IO2	PA10	18(2)			20(2)			20	20			D2			31		
	G4_IO3	PA11	17(2)			19(2)			21	21			C1			32		
	G4_IO4	PA12	18(2)			20(2)			22	22			A1			33		

Table 22. Available touch sensing channels for STM32F042 (continued)

Subfamily			STM32F042															
Packages			TSSOP20			UFQFPN28			LQFP32 / UFQFPN32				WLCSP36			LQFP48 / UFQFPN48		
Flash memory size			4=16K, 6=32K															
Part numbers			STM32F042F[46]			STM32F042G[46]			STM32F042K[46]				STM32F042T[46]			STM32F042C[46]		
Analog I/O group	Gx_IOy	GPIO	Pin	Number of available pins	Usage	Pin	Number of available pins	Usage	Pin LQFP	Pin UFQFPN	Number of available pins	Usage	Pin	Number of available pins	Usage	Pin	Number of available pins	Usage
Group 5	G5_IO1	PB3	-	0	Cannot be used for touch sensing	24	4	3 channels with 1 sampling capacitor	26	26	4	3 channels with 1 sampling capacitor	B3	4	3 channels with 1 sampling capacitor	39	4	3 channels with 1 sampling capacitor
	G5_IO2	PB4	-			25			27	27			A3			40		
	G5_IO3	PB6	-			27			29	29			C4			42		
	G5_IO4	PB7	-			28			30	30			A4			43		
Group 6	not available		-	0	Cannot be used for touch sensing	-	0	Cannot be used for Touch sensing	-	-	0	Cannot be used for Touch sensing	-	0	Cannot be used for Touch sensing	-	0	Cannot be used for Touch sensing
			-			-			-	-			-			-		
			-			-			-	-			-			-		
			-			-			-	-			-			-		
Group 7			-	0	Cannot be used for Touch sensing	-	0	Cannot be used for Touch sensing	-	-	0	Cannot be used for Touch sensing	-	0	Cannot be used for Touch sensing	-	0	Cannot be used for Touch sensing
			-			-			-	-			-			-		
			-			-			-	-			-			-		
			-			-			-	-			-			-		
Group 8			-	0	Cannot be used for Touch sensing	-	0	Cannot be used for Touch sensing	-	-	0	Cannot be used for Touch sensing	-	0	Cannot be used for Touch sensing	-	0	Cannot be used for Touch sensing
			-			-			-	-			-			-		
			-			-			-	-			-			-		
			-			-			-	-			-			-		
Maximum number of channels			7 with 3 sampling capacitors			11 with 5 sampling capacitors			13/14 with 5 sampling capacitors				14 with 5 sampling capacitors			14 with 5 sampling capacitors		

1. This GPIO offers a reduced touch sensing sensitivity. It is thus recommended to use it as sampling capacitor I/O.
2. Pin pair PA11/PA12 can be remapped instead of pin pair PA9/PA10 using SYS_CTRL register.

Table 23. Available touch sensing channels for STM32F051 and STM32F072

Subfamily			STM32F051/STM32F072												
Packages			LQFP32/UFQFPN32				LQFP48			LQFP64			LQFP100		
Flash memory size			4=16K, 6=32K, 8=64K, B=128K, C=256K												
Part numbers			STM32F051K[468]				STM32F051C[468B] STM32F072C[8B]			STM32F051R[468B] STM32F072R[8B]			STM32F051VB STM32F052V[8B]		
Analog I/O group	Gx_IOy	GPIO	Pin LQFP	Pin UFQFPN	Number of available pins	Usage	Pin	Number of available pins	Usage	Pin	Number of available pins	Usage	Pin	Number of available pins	Usage
Group 1	G1_IO1	PA0	6	6	4	3 channels with 1 sampling capacitor	10	4	3 channels with 1 sampling capacitor	14	4	3 channels with 1 sampling capacitor	23	4	3 channels with 1 sampling capacitor
	G1_IO2	PA1	7	7			11			15			24		
	G1_IO3	PA2	8	8			12			16			25		
	G1_IO4	PA3	9	9			13			17			26		
Group 2	G2_IO1	PA4(1)	10	10	4	3 channels with 1 sampling capacitor	14	4	3 channels with 1 sampling capacitor	20	4	3 channels with 1 sampling capacitor	29	4	3 channels with 1 sampling capacitor
	G2_IO2	PA5(1)	11	11			15			21			30		
	G2_IO3	PA6	12	12			16			22			31		
	G2_IO4	PA7	13	13			17			23			32		
Group 3	G3_IO1	PC5	-	-	2/3	1/2 channels with 1 sampling capacitor	-	3	2 channels with 1 sampling capacitor	25	4	3 channels with 1 sampling capacitor	34	4	3 channels with 1 sampling capacitor
	G3_IO2	PB0	14	14			18			26			35		
	G3_IO3	PB1	15	15			19			27			36		
	G3_IO4	PB2	-	16			20			28			37		
Group 4	G4_IO1	PA9	19	19	4	3 channels with 1 sampling capacitor	30	4	3 channels with 1 sampling capacitor	42	4	3 channels with 1 sampling capacitor	68	4	3 channels with 1 sampling capacitor
	G4_IO2	PA10	20	20			31			43			69		
	G4_IO3	PA11	21	21			32			44			70		
	G4_IO4	PA12	22	22			33			45			71		
Group 5	G5_IO1	PB3	26	26	4	3 channels with 1 sampling capacitor	39	4	3 channels with 1 sampling capacitor	55	4	3 channels with 1 sampling capacitor	89	4	3 channels with 1 sampling capacitor
	G5_IO2	PB4	27	27			40			56			90		
	G5_IO3	PB6	29	29			42			58			92		
	G5_IO4	PB7	30	30			43			59			93		

Table 23. Available touch sensing channels for STM32F051 and STM32F072 (continued)

Subfamily			STM32F051/STM32F072												
Packages			LQFP32/UFQFPN32				LQFP48			LQFP64			LQFP100		
Flash memory size			4=16K, 6=32K, 8=64K, B=128K, C=256K												
Part numbers			STM32F051K[468]				STM32F051C[468B] STM32F072C[8B]			STM32F051R[468B] STM32F072R[8B]			STM32F051VB STM32F052V[8B]		
Analog I/O group	Gx_IOy	GPIO	Pin LQFP	Pin UFQFPN	Number of available pins	Usage	Pin	Number of available pins	Usage	Pin	Number of available pins	Usage	Pin	Number of available pins	Usage
Group 6	G6_IO1	PB11	-	-	0	Cannot be used for touch sensing	22	4	3 channels with 1 sampling capacitor	30	4	3 channels with 1 sampling capacitor	48	4	3 channels with 1 sampling capacitor
	G6_IO2	PB12	-	-			25			33			51		
	G6_IO3	PB13	-	-			26			34			52		
	G6_IO4	PB14	-	-			27			35			53		
Group 7	G7_IO1	PE2	-	-	0	Cannot be used for Touch sensing	-	0	Cannot be used for Touch sensing	-	0	Cannot be used for Touch sensing	1	4	3 channels with 1 sampling capacitor
	G7_IO2	PE3	-	-			-			-			2		
	G7_IO3	PE4	-	-			-			-			3		
	G7_IO4	PE5	-	-			-			-			4		
Group 8	G8_IO1	PD12	-	-	0	Cannot be used for Touch sensing	-	0	Cannot be used for Touch sensing	-	0	Cannot be used for Touch sensing	59	4	3 channels with 1 sampling capacitor
	G8_IO2	PD13	-	-			-			-			60		
	G8_IO3	PD14	-	-			-			-			61		
	G8_IO4	PD15	-	-			-			-			62		
Maximum number of channels			13/14 with 5 sampling capacitors				17 with 6 sampling capacitors			18 with 6 sampling capacitors			24 with 8 sampling capacitors		

1. This GPIO offers a reduced touch sensing sensitivity. It is thus recommended to use it as sampling capacitor I/O.

2.21.7 Hardware implementation example

Figure 19 shows an example of hardware implementation on STM32F0xx devices.

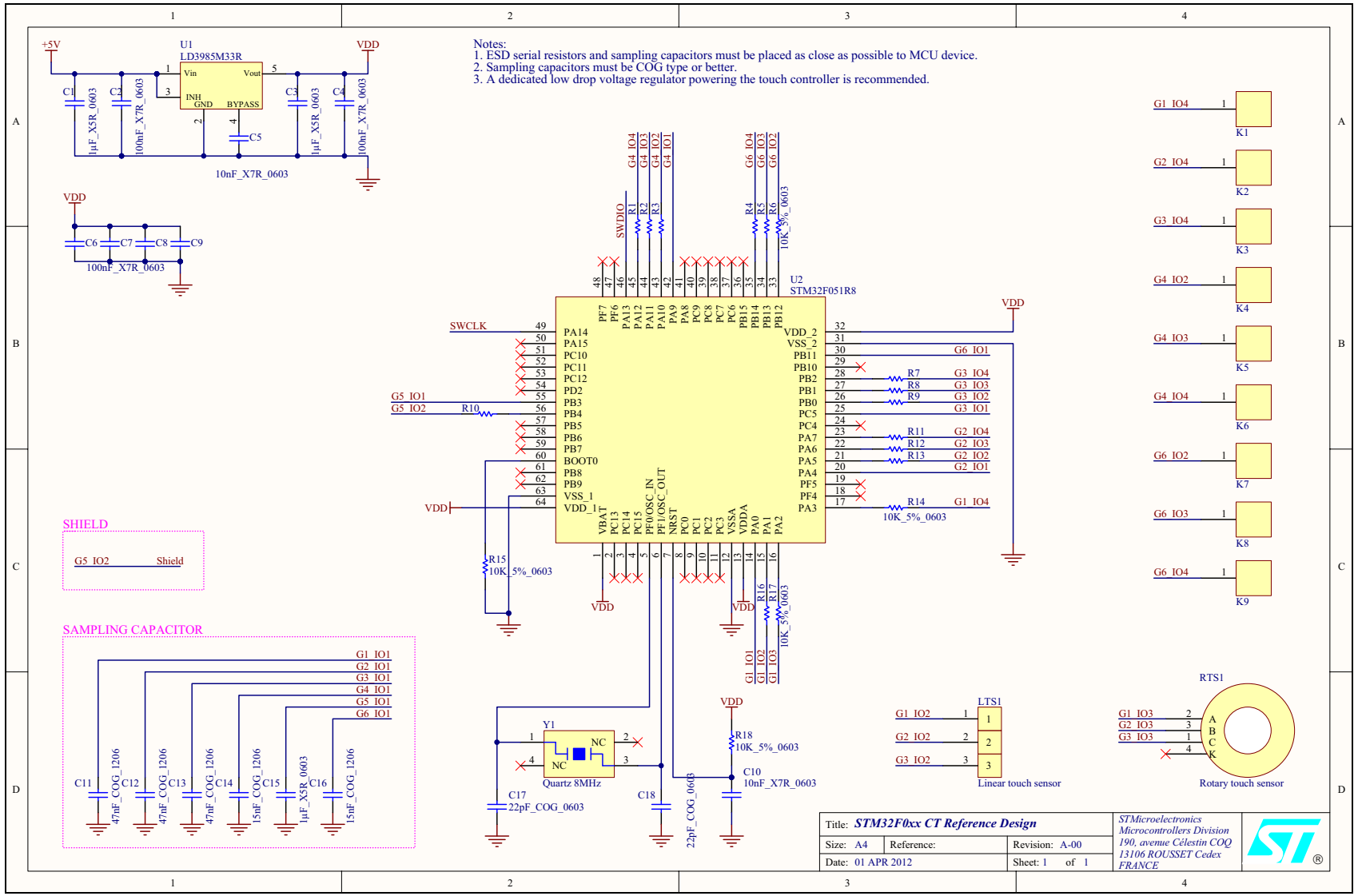
Recommendations to increase the noise immunity on the PCB

To ensure a correct operation in noisy environment, the floating nets must be avoided (tracks, copper elements, conductive frames, etc...).

As a consequence:

- All unused touch controller I/Os must be either configured to output push-pull low or externally tied to GND.
- The parameter TSLPRM_TSC_IODEF should also be configured to the output push-pull low state.
- We recommend to drive the sampling capacitor common node using a standard I/O of the touch controller configured in output push-pull low mode.
- It may also be required to add a capacitor-input filter (pi filter) on each channel line.

Figure 19. STM32F0xx hardware implementation example



2.22 STM32F3xx devices

2.22.1 Acquisition

The STM32F3xx devices acquisition is done in the files:

- tsl_acq_stm32f3xx.c
- tsl_acq_stm32f3xx.h

Functions used by the application layer and that are device dependent:

- TSL_acq_BankConfig()
- TSL_acq_BankStartAcq()
- TSL_acq_BankWaitEOC()
- TSL_acq_GetMeas()

The other functions in this file are for internal use and the user doesn't need to call them directly.

2.22.2 Timings

The STM32F3xx devices timing management is done in the files:

- tsl_time_stm32f3xx.c
- tsl_time_stm32f3xx.h

The **systick** is used to generate a timebase for the ECS and DTO modules.

Functions used:

- TSL_tim_Init()

2.22.3 Parameters

The parameters specific to the STM32F3xx devices are described in the file:

- tsl_conf_stm32f3xx.h

and are checked in the file:

- tsl_check_config_stm32f3xx.h

2.22.4 Memory footprint

Conditions

- IAR ANSI C/C++ Compiler V6.40.1 for Arm®
- Compiler options: optimization high-balanced
- IAR library not counted
- STMTouch driver default options: ECS=ON, DTO=ON, PROX=OFF, ZONE=OFF, DXS=OFF
- Each sensor has its own parameters, all parameters placed in RAM

The following tables summarize the memory footprint with different configurations:

Table 24. STM32F30x memory footprint(1)

Channels	Banks	Sensors	ROM (Kbytes)	RAM (bytes)
2	2	2 TKeys	~3.2	~120

1. The content of this table is provided for information purposes only.

Table 25. STM32F37x memory footprint(1)

Channels	Banks	Sensors	ROM (Kbytes)	RAM (bytes)
3	3	3 TKeys	~3.3	~140
3	3	1 Linear-3ch	~4.3	~130

1. The content of this table is provided for information purposes only.

2.22.5 MCU resources

The table below shows the peripherals that are used by the STMTouch driver on STM32F3xx devices. Care must be taken when using them to avoid any unwanted behavior.

Table 26. STM32F3xx MCU resources used

Peripheral	Function
GPIOs	Acquisition
Systick	Time base for ECS and DTO
Touch sense controller (TSC)	Acquisition

2.22.6 STM32F3xx available touch-sensing channels

The tables below provide an overview of the available touch sensing channels for the STM32F30x and STM32F37x devices.

Note: The following tables are not restrictive in term of part numbers supported by the STMTouch driver. The STMTouch driver can be used on any new device that may become available as part of ST microcontrollers portfolio. Please contact your ST representative for support.

Note: For n available pins in an I/O group, one pin is used as sampling capacitor and n-1 pins are used as channels.

The I/O group cannot be used if the number of available pins in less or equal to one.

Table 27. Available touch sensing channels for STM32F30x

Subfamily			STM32F30x												
Packages			LQFP32			LQFP48			LQFP64			LQFP100			
Part numbers			STM32F301K[468] STM32F302K[468] STM32F303K[468] STM32F333K[468]			STM32F301C[468] STM32F302C[468BC] STM32F303C[468BC] STM32F333C[468]			STM32F301R[468] STM32F302R[468BC] STM32F303R[468BC] STM32F333R[468]			STM32F302V[BC] STM32F303V[BC]		
Analog I/O group	Gx_IOy	GPIO	Pin	Number of available pins	Usage	Pin	Number of available pins	Usage	Pin	Number of available pins	Usage	Pin	Number of available pins	Usage
Group 1	G1_IO1	PA0	7	4	3 channels with 1 sampling capacitor	10	4	3 channels with 1 sampling capacitor	14	4	3 channels with 1 sampling capacitor	23	4	3 channels with 1 sampling capacitor
	G1_IO2	PA1	8			11			15			24		
	G1_IO3	PA2	9			12			16			25		
	G1_IO4	PA3	10			13			17			26		
Group 2	G2_IO1	PA4(1)	11	4	3 channels with 1 sampling capacitor	14	4	3 channels with 1 sampling capacitor	20	4	3 channels with 1 sampling capacitor	29	4	3 channels with 1 sampling capacitor
	G2_IO2	PA5(1)	12			15			21			30		
	G2_IO3	PA6	13			16			22			31		
	G2_IO4	PA7	14			17			23			32		
Group 3	G3_IO1	PC5	-	1	Cannot be used for touch sensing	-	3	2 channels with 1 sampling capacitor	25	4	3 channels with 1 sampling capacitor	34	4	3 channels with 1 sampling capacitor
	G3_IO2	PB0	15			18			26			35		
	G3_IO3	PB1	-			19			27			36		
	G3_IO4	PB2	-			20			28			37		
Group 4	G4_IO1	PA9	19	4	3 channels with 1 sampling capacitor	30	4	3 channels with 1 sampling capacitor	42	4	3 channels with 1 sampling capacitor	68	4	3 channels with 1 sampling capacitor
	G4_IO2	PA10	20			31			43			69		
	G4_IO3	PA13	23			34			46			72		
	G4_IO4	PA14	24			37			49			76		

Table 27. Available touch sensing channels for STM32F30x (continued)

Subfamily			STM32F30x												
Packages			LQFP32			LQFP48			LQFP64			LQFP100			
Part numbers			STM32F301K[468] STM32F302K[468] STM32F303K[468] STM32F333K[468]			STM32F301C[468] STM32F302C[468BC] STM32F303C[468BC] STM32F333C[468]			STM32F301R[468] STM32F302R[468BC] STM32F303R[468BC] STM32F333R[468]			STM32F302V[BC] STM32F303V[BC]		
Analog I/O group	Gx_IOy	GPIO	Pin	Number of available pins	Usage	Pin	Number of available pins	Usage	Pin	Number of available pins	Usage	Pin	Number of available pins	Usage
Group 5	G5_IO1	PB3	26	4	3 channels with 1 sampling capacitor	39	4	3 channels with 1 sampling capacitor	55	4	3 channels with 1 sampling capacitor	89	4	3 channels with 1 sampling capacitor
	G5_IO2	PB4	27			40			56			90		
	G5_IO3	PB6	29			42			58			92		
	G5_IO4	PB7	30			43			59			93		
Group 6	G6_IO1	PB11	-	0	Cannot be used for touch sensing	22	4	3 channels with 1 sampling capacitor	30	4	3 channels with 1 sampling capacitor	48	4	3 channels with 1 sampling capacitor
	G6_IO2	PB12	-			25			33			51		
	G6_IO3	PB13	-			26			34			52		
	G6_IO4	PB14	-			27			35			53		
Group 7	G7_IO1	PE2	-	0	Cannot be used for touch sensing	-	0	Cannot be used for touch sensing	-	0	Cannot be used for touch sensing	1	4	3 channels with 1 sampling capacitor
	G7_IO2	PE3	-			-			-			2		
	G7_IO3	PE4	-			-			-			3		
	G7_IO4	PE5	-			-			-			4		
Group 8	G8_IO1	PD12	-	0	Cannot be used for touch sensing	-	0	Cannot be used for touch sensing	-	0	Cannot be used for touch sensing	59	4	3 channels with 1 sampling capacitor
	G8_IO2	PD13	-			-			-			60		
	G8_IO3	PD14	-			-			-			61		
	G8_IO4	PD15	-			-			-			62		
Maximum number of channels			12 with 4 sampling capacitors			17 with 6 sampling capacitors			18 with 6 sampling capacitors			24 with 8 sampling capacitors		

1. This GPIO offers a reduced touch sensing sensitivity. It is thus recommended to use it as sampling capacitor I/O.

Table 28. Available touch sensing channels for STM32F37x

Subfamily			STM32F37x									
Packages			LQFP48			LQFP64			LQFP100 / UFBGA100			
Flash memory size			8=64K, B=128K, C=256K									
Part numbers			STM32F373C[8BC]			STM32F373R[8BC]			STM32F373V[8BC]			
Analog I/O group	Gx_IOy	GPIO	Pin	Number of available pins	Usage	Pin	Number of available pins	Usage	LQFP Pin	BGA Pin	Number of available pins	Usage
Group 1	G1_IO1	PA0	10	4	3 channels with 1 sampling capacitor	14	4	3 channels with 1 sampling capacitor	23	L2	4	3 channels with 1 sampling capacitor
	G1_IO2	PA1	11			15			24	M2		
	G1_IO3	PA2	12			16			25	K3		
	G1_IO4	PA3	13			18			26	L3		
Group 2	G2_IO1	PA4(1)	14	3	2 channels with 1 sampling capacitor	20	4	3 channels with 1 sampling capacitor	29	M3	4	3 channels with 1 sampling capacitor
	G2_IO2	PA5(1)	15			21			30	K4		
	G2_IO3	PA6(1)	16			22			31	L4		
	G2_IO4	PA7	-			23			32	M4		
Group 3	G3_IO1	PC4	-	2	1 channel with 1 sampling capacitor	24	4	3 channels with 1 sampling capacitor	33	K5	4	3 channels with 1 sampling capacitor
	G3_IO2	PC5	-			25			34	L5		
	G3_IO3	PB0	18			26			35	M5		
	G3_IO4	PB1	19			27			36	M6		
Group 4	G4_IO1	PA9	30	4	3 channels with 1 sampling capacitor	42	4	3 channels with 1 sampling capacitor	68	D10	4	3 channels with 1 sampling capacitor
	G4_IO2	PA10	31			43			69	C12		
	G4_IO3	PA13	34			46			72	A11		
	G4_IO4	PA14	37			49			76	A10		

Table 28. Available touch sensing channels for STM32F37x (continued)

Subfamily			STM32F37x									
Packages			LQFP48			LQFP64			LQFP100 / UFBGA100			
Flash memory size			8=64K, B=128K, C=256K									
Part numbers			STM32F373C[8BC]			STM32F373R[8BC]			STM32F373V[8BC]			
Analog I/O group	Gx_IOy	GPIO	Pin	Number of available pins	Usage	Pin	Number of available pins	Usage	LQFP Pin	BGA Pin	Number of available pins	Usage
Group 5	G5_IO1	PB3	39	4	3 channels with 1 sampling capacitor	55	4	3 channels with 1 sampling capacitor	89	A8	4	3 channels with 1 sampling capacitor
	G5_IO2	PB4	40			56			90	A7		
	G5_IO3	PB6	42			58			92	B5		
	G5_IO4	PB7	43			59			93	B4		
Group 6	G6_IO1	PB14	26	3	2 channels with 1 sampling capacitor	34	3	2 channels with 1 sampling capacitor	53	K11	4	3 channels with 1 sampling capacitor
	G6_IO2	PB15	27			35			54	K10		
	G6_IO3	PD8	28			36			55	K9		
	G6_IO4	PD9	-			-			56	K8		
Group 7	G7_IO1	PE2	-	0	Cannot be used for touch sensing	-	0	Cannot be used for touch sensing	1	B2	4	3 channels with 1 sampling capacitor
	G7_IO2	PE3	-			-			2	A1		
	G7_IO3	PE4	-			-			3	B1		
	G7_IO4	PE5	-			-			4	C2		

Table 28. Available touch sensing channels for STM32F37x (continued)

Subfamily			STM32F37x									
Packages			LQFP48			LQFP64			LQFP100 / UFBGA100			
Flash memory size			8=64K, B=128K, C=256K									
Part numbers			STM32F373C[8BC]			STM32F373R[8BC]			STM32F373V[8BC]			
Analog I/O group	Gx_IOy	GPIO	Pin	Number of available pins	Usage	Pin	Number of available pins	Usage	LQFP Pin	BGA Pin	Number of available pins	Usage
Group 8	G8_IO1	PD12	-	0	Cannot be used for touch sensing	-	0	Cannot be used for touch sensing	59	J10	4	3 channels with 1 sampling capacitor
	G8_IO2	PD13	-			-			60	H12		
	G8_IO3	PD14	-			-			61	H11		
	G8_IO4	PD15	-			-			62	H10		
Maximum number of channels			14 with 6 sampling capacitors			17 with 6 sampling capacitors			24 with 8 sampling capacitors			

1. This GPIO offers a reduced touch sensing sensitivity. It is thus recommended to use it as sampling capacitor I/O.

2.22.7 Hardware implementation example

Figure 20 shows an example of hardware implementation on STM32F3xx devices.

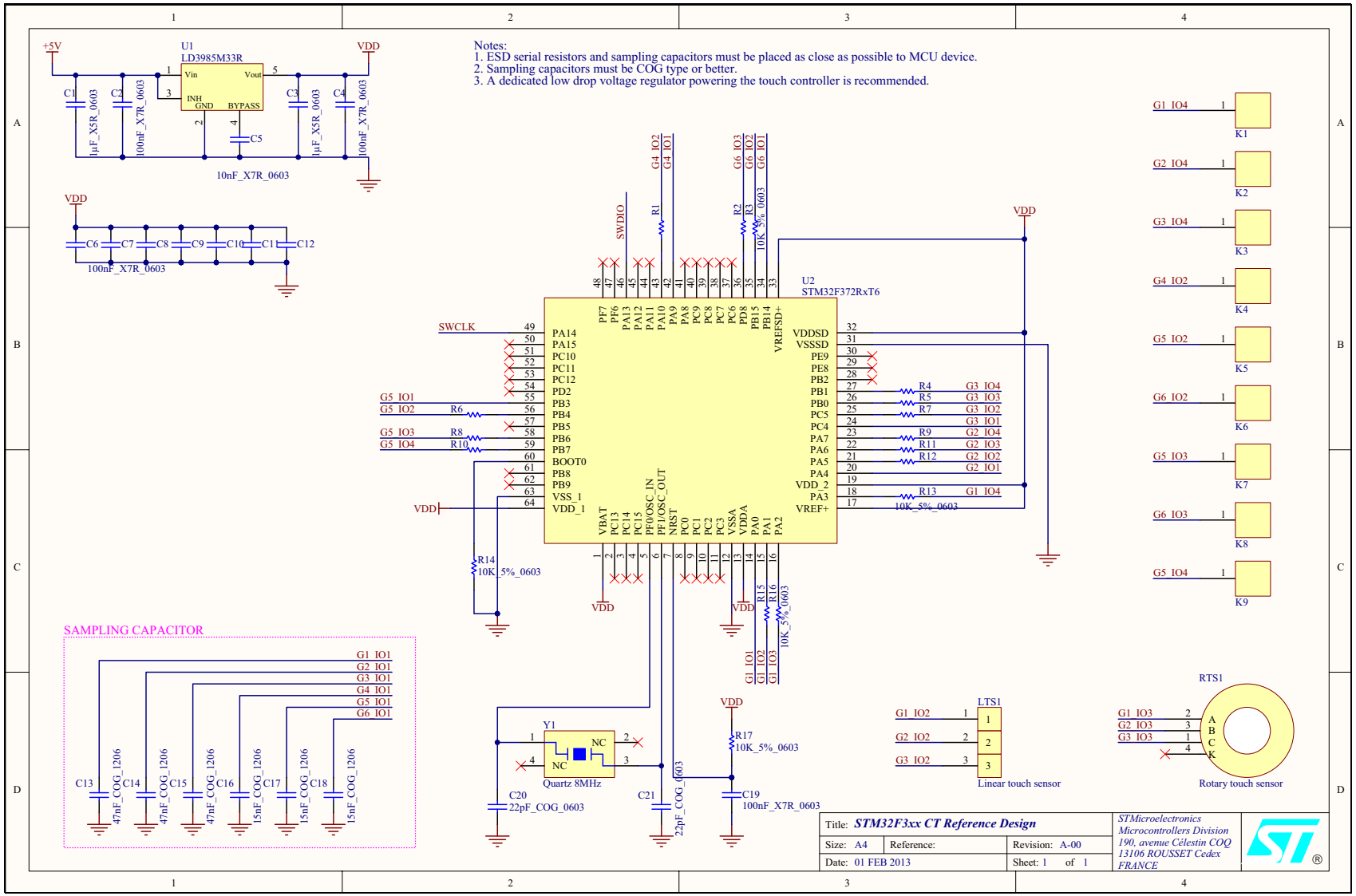
Recommendations to increase the noise immunity on the PCB

To ensure a correct operation in noisy environment, the floating nets must be avoided (tracks, copper elements, conductive frames, etc...).

As a consequence:

- All unused touch controller I/Os must be either configured to output push-pull low or externally tied to GND.
- The parameter TSLPRM_TSC_IODEF should also be configured to the output push-pull low state.
- We recommend to drive the sampling capacitor common node using a standard I/O of the touch controller configured in output push-pull low mode.
- It may also be required to add a capacitor-input filter (pi filter) on each channel line.

Figure 20. STM32F3xx hardware implementation example



2.23 STM32L1xx devices

2.23.1 Acquisition

The STM32L1xx devices **hardware acquisition mode** (using two timers) is done in the files:

- tsl_acq_stm32l1xx_hw.c
- tsl_acq_stm32l1xx_hw.h

Warning: This acquisition mode is available for the STM32L1xx High-density and STM32L1xx Medium-density Plus devices only.

The STM32L1xx devices **software acquisition mode** is done in the files:

- tsl_acq_stm32l1xx_sw.c
- tsl_acq_stm32l1xx_sw.h

This acquisition is available for all STM32L1xx devices.

Note: The hardware acquisition mode is selected per default for the STM32L1xx High-density and Medium-density Plus devices. If you want to use the software acquisition mode you must add the following constant in the toolchain compiler preprocessor:

- TSLPRM_STM32L1XX_SW_ACQ

Functions used by the application layer and that are device dependent:

- TSL_acq_BankConfig()
- TSL_acq_BankStartAcq()
- TSL_acq_BankWaitEOC()
- TSL_acq_GetMeas()

The other functions in this file are for internal use and the user doesn't need to call them directly.

2.23.2 Timings

The STM32L1xx devices timing management is done in the files:

- tsl_time_stm32l1xx.c
- tsl_time_stm32l1xx.h

The **systick** is used to generate a timebase for the ECS and DTO modules.

Functions used:

- TSL_tim_Init()

2.23.3 Parameters

The parameters specific to the STM32L1xx devices are described in the file:

- tsl_conf_stm32l1xx.h

and are checked in the file:

- tsl_check_config_stm32l1xx.h

2.23.4 Memory footprint

Conditions

- IAR ANSI C/C++ Compiler V6.30.1 for Arm®
- Compiler options: optimization high-balanced
- IAR library not counted
- STMTouch driver default options: ECS=ON, DTO=ON, ZONE=OFF, DXS=ON (excepted if only one sensor is used)
- Each sensor has its own parameters, all parameters placed in RAM

The following tables summarize the memory footprint taken by the STMTouch driver using the **hardware acquisition mode** (using Timers) on **STM32L1xx High-density devices**:

Table 29. STM32L1xx_HD(1) memory footprint without proximity(2)

Channels	Banks	Sensors	ROM (Kbytes)	RAM (bytes)
3	3	3 TKeys	~6.2	~370
3	3	1 Linear-3ch	~7.2	~360
16	3	10 TKeys 1 Linear-3ch 1 Rotary-3ch	~9.1	~630

1. See *Section 3.1.1: Toolchain compiler preprocessor section* for definition of STM32L1xx_HD.
2. The content of this table is provided for information purposes only.

Table 30. STM32L1xx_HD(1) memory footprint with proximity(2)

Channels	Banks	Sensors	ROM (Kbytes)	RAM (bytes)
3	3	3 TKeys	~7.0	~380
3	3	1 Linear-3ch	~8.1	~370
16	3	10 TKeys 1 Linear-3ch 1 Rotary-3ch	~10.9	~680

1. See *Section 3.1.1: Toolchain compiler preprocessor section* for definition of STM32L1xx_HD.
2. The content of this table is provided for information purposes only.

The following tables summarize the memory footprint taken by the STMTouch driver using the **software acquisition mode** on **STM32L1xx Medium-density Plus devices**:

Table 31. STM32L1xx_MDP(1) memory footprint without proximity(2)

Channels	Banks	Sensors	ROM (Kbytes)	RAM (bytes)
2	2	2 TKeys	~5.9	~350

1. See *Section 3.1.1: Toolchain compiler preprocessor section* for definition of STM32L1xx_MDP.

2. The content of this table is provided for information purposes only

Table 32. STM32L1xx_MDP(1) memory footprint with proximity(2)

Channels	Banks	Sensors	ROM (Kbytes)	RAM (bytes)
2	2	2 TKeys	~6.7	~360

1. See *Section 3.1.1: Toolchain compiler preprocessor section* for definition of STM32L1xx_MDP.
2. The content of this table is provided for information purposes only.

The following tables summarize the memory footprint taken by the STMTouch driver using the **software acquisition mode** on **STM32L1xx Medium-density devices**:

Table 33. STM32L1xx_MD(1) memory footprint without proximity(2)

Channels	Banks	Sensors	ROM (Kbytes)	RAM (bytes)
2	2	2 TKeys	~5.2	~400
3	3	1 Linear-3ch	~6.2	~420
16	3	10 TKeys 1 Linear-3ch 1 Rotary-3ch	~8.7	~690

1. See *Section 3.1.1: Toolchain compiler preprocessor section* for definition of STM32L1xx_MD.
2. The content of this table is provided for information purposes only.

Table 34. STM32L1xx_MD(1) memory footprint with proximity(2)

Channels	Banks	Sensors	ROM (Kbytes)	RAM (bytes)
2	2	2 TKeys	~6.0	~400
3	3	1 Linear-3ch	~7.2	~420
16	3	10 TKeys 1 Linear-3ch 1 Rotary-3ch	~10.4	~730

1. See *Section 3.1.1: Toolchain compiler preprocessor section* for definition of STM32L1xx_MD.
2. The content of this table is provided for information purposes only.

2.23.5 MCU resources

The tables below show the peripherals that are used by the STMTouch driver on STM32L1xx devices. Care must be taken when using them to avoid any unwanted behavior.

Table 35. MCU resources used on STM32L1xx with hardware acquisition

Peripheral	Function
GPIOs	Acquisition
Systick	Time base for ECS and DTO
2 Timers (TIM9, TIM11)	Acquisition
Routing interface	Acquisition

Table 36. MCU resources used on STM32L1xx with software acquisition

Peripheral	Function
GPIOs	Acquisition
Systick	Time base for ECS and DTO
Routing interface	Acquisition

2.23.6 STM32L1xx available touch-sensing channels

The tables below provide an overview of the available touch sensing channels for the STM32L1xx devices.

Note: The following tables are not restrictive in term of part numbers supported by the STMTouch driver. The STMTouch driver can be used on any new device that may become available as part of ST microcontrollers portfolio. Please contact your ST representative for support.

Note: For n available pins in an I/O group, one pin is used as sampling capacitor and n-1 pins are used as channels. The I/O group cannot be used if the number of available pins in less or equal to one.

Table 37. Available touch sensing channels for *STM32L1xx 512K*

Subfamily			STM32L1xx 512K													
Packages			LQFP64			LQFP100 / WLCSP104				UFBGA132			LQFP144		
Part numbers			STM32L151RE STM32L152RE STM32L162RE			STM32L151VE STM32L152VE STM32L162VE				STM32L151QE STM32L152QE STM32L162QE			STM32L151ZE STM32L152ZE STM32L162ZE		
Analog I/O group	Gx_IOy	GPIO	LQFP pin	Number of available pins	Usage	LQFP Pin	WLCSP ball	Number of available pins	Usage	BGA ball	Number of available pins	Usage	LQFP pin	Number of available pins	Usage
Group 1	G1_IO1	PA0	14	4	3 channels with 1 sampling capacitor	23	K9	4	3 channels with 1 sampling capacitor	L2	4	3 channels with 1 sampling capacitor	34	4	3 channels with 1 sampling capacitor
	G1_IO2	PA1	15			24	L9			M2			35		
	G1_IO3	PA2	16			25	J8			K3			36		
	G1_IO4	PA3(1)	17			26	H7			L3			37		
Group 2	G2_IO1	PA6	22	2	1 channel with 1 sampling capacitor	31	H6	2	1 channel with 1 sampling capacitor	L4	4(2)	3 channels with 1 sampling capacitor	42	4(2)	3 channels with 1 sampling capacitor
	G2_IO2	PA7	23			32	K7			J5			43		
	G2_IO3	PF15	-			-	-			J9			55		
	G2_IO4	PG0(3)	-			-	-			H9			56		
	G2_IO5	PG1(3)	-			-	-			G9			57		
Group 3	G3_IO1	PB0(1)	26	3	2 channels with 1 sampling capacitor	35	J6	3	2 channels with 1 sampling capacitor	M5	5	4 channels with 1 sampling capacitor	46	5	4 channels with 1 sampling capacitor
	G3_IO2	PB1	27			36	K6			M6			47		
	G3_IO3	PB2	28			37	M6			L6			48		
	G3_IO4	PF11	-			-	-			K6			49		
	G3_IO5	PF12	-			-	-			J7			50		

Table 37. Available touch sensing channels for *STM32L1xx 512K* (continued)

Subfamily			STM32L1xx 512K												
Packages			LQFP64			LQFP100 / WLCSP104				UFBGA132			LQFP144		
Part numbers			STM32L151RE STM32L152RE STM32L162RE			STM32L151VE STM32L152VE STM32L162VE				STM32L151QE STM32L152QE STM32L162QE			STM32L151ZE STM32L152ZE STM32L162ZE		
Analog I/O group	Gx_IOy	GPIO	LQFP pin	Number of available pins	Usage	LQFP Pin	WLCSP ball	Number of available pins	Usage	BGA ball	Number of available pins	Usage	LQFP pin	Number of available pins	Usage
Group 4	G4_IO1	PA8	41	3	2 channels with 1 sampling capacitor	67	F3	3	2 channels with 1 sampling capacitor	D11	3	2 channels with 1 sampling capacitor	100	3	2 channels with 1 sampling capacitor
	G4_IO2	PA9	42			68	F1			D10			101		
	G4_IO3	PA10	43			69	F2			C12			102		
Group 5	G5_IO1	PA13	46	3	2 channels with 1 sampling capacitor	72	E3	3	2 channels with 1 sampling capacitor	A11	3	2 channels with 1 sampling capacitor	105	3	2 channels with 1 sampling capacitor
	G5_IO2	PA14	49			76	D3			A10			109		
	G5_IO3	PA15	50			77	B1			A9			110		
Group 6	G6_IO1	PB4	56	4	3 channels with 1 sampling capacitor	90	A5	4	3 channels with 1 sampling capacitor	A7	4	3 channels with 1 sampling capacitor	134	4	3 channels with 1 sampling capacitor
	G6_IO2	PB5	57			91	A6			C5			135		
	G6_IO3	PB6	58			92	C5			B5			136		
	G6_IO4	PB7	59			93	C7			B4			137		

Table 37. Available touch sensing channels for *STM32L1xx 512K* (continued)

Subfamily			STM32L1xx 512K													
Packages			LQFP64			LQFP100 / WLCSP104				UFBGA132			LQFP144			
Part numbers			STM32L151RE STM32L152RE STM32L162RE			STM32L151VE STM32L152VE STM32L162VE				STM32L151QE STM32L152QE STM32L162QE			STM32L151ZE STM32L152ZE STM32L162ZE			
Analog I/O group	Gx_IOy	GPIO	LQFP pin	Number of available pins	Usage	LQFP Pin	WLCSP ball	Number of available pins	Usage	BGA ball	Number of available pins	Usage	LQFP pin	Number of available pins	Usage
Group 7	G7_IO1	PB12	33	4	3 channels with 1 sampling capacitor	51	J4	4	3 channels with 1 sampling capacitor	L12	5(2)	4 channels with 1 sampling capacitor	73	5(2)	4 channels with 1 sampling capacitor
	G7_IO2	PB13	34			52	J3			K12			74		
	G7_IO3	PB14	35			53	L1			K11			75		
	G7_IO4	PB15	36			54	K2			K10			76		
	G7_IO5	PG2(3)	-			-	-			G10			87		
	G7_IO6	PG3(3)	-			-	-			F9			88		
	G7_IO7	PG4(3)	-			-	-			F10			89		
Group 8	G8_IO1	PC0	8	4	3 channels with 1 sampling capacitor	15	F6	4	3 channels with 1 sampling capacitor	H1	4	3 channels with 1 sampling capacitor	26	4	3 channels with 1 sampling capacitor
	G8_IO2	PC1	9			16	H9			J2			27		
	G8_IO3	PC2	10			17	G9			J3			28		
	G8_IO4	PC3	11			18	G8			K2			29		
Group 9	G9_IO1	PC4	24	2	1 channel with 1 sampling capacitor	33	L7	2	1 channel with 1 sampling capacitor	K5	4	3 channels with 1 sampling capacitor	44	4	3 channels with 1 sampling capacitor
	G9_IO2	PC5	25			34	M7			L5			45		
	G9_IO3	PF13	-			-	-			K7			53		
	G9_IO4	PF14	-			-	-			J8			54		

Table 37. Available touch sensing channels for *STM32L1xx 512K* (continued)

Subfamily			STM32L1xx 512K												
Packages			LQFP64			LQFP100 / WLCSP104				UFBGA132			LQFP144		
Part numbers			STM32L151RE STM32L152RE STM32L162RE			STM32L151VE STM32L152VE STM32L162VE				STM32L151QE STM32L152QE STM32L162QE			STM32L151ZE STM32L152ZE STM32L162ZE		
Analog I/O group	Gx_IOy	GPIO	LQFP pin	Number of available pins	Usage	LQFP Pin	WLCSP ball	Number of available pins	Usage	BGA ball	Number of available pins	Usage	LQFP pin	Number of available pins	Usage
Group 10	G10_IO1	PC6	37	4	3 channels with 1 sampling capacitor	63	H1	4	3 channels with 1 sampling capacitor	E12	4	3 channels with 1 sampling capacitor	96	4	3 channels with 1 sampling capacitor
	G10_IO2	PC7	38			64	G1			E11			97		
	G10_IO3	PC8	39			65	G2			E10			98		
	G10_IO4	PC9	40			66	F4			D12			99		
Group 11	G11_IO1	PF6	-	0	Cannot be used for touch sensing	-	-	0	Cannot be used for touch sensing	G3	4	3 channels with 1 sampling capacitor	18	5	4 channels with 1 sampling capacitor
	G11_IO2	PF7	-			-	-			G4			19		
	G11_IO3	PF8	-			-	-			H4			20		
	G11_IO4	PF9	-			-	-			J6			21		
	G11_IO5	PF10	-			-	-			-			22		
Maximum number of channels			23 channels with 10 sampling capacitors			23 channels with 10 sampling capacitors				33 channels with 11 sampling capacitors			34 channels with 11 sampling capacitors		

1. This GPIO offers a reduced touch sensing sensitivity. It is thus recommended to use it as sampling capacitor I/O.
2. Not all the pins are available simultaneously on this group.
3. This GPIO can only be configured as sampling capacitor I/O when using HW acquisition mode and as channel I/O when using SW acquisition mode.

Table 38. Available touch sensing channels for *STM32L1xx 384K*

Subfamily			STM32L1xx 384K													
Packages			LQFP64 / WLCSP64				LQFP100			UFBGA132			LQFP144			
Part numbers			STM32L151RD STM32L152RD STM32L162RD				STM32L151VD STM32L152VD STM32L162VD			STM32L151QD STM32L152QD STM32L162QD			STM32L151ZD STM32L152ZD STM32L162ZD			
Analog I/O group	Gx_IOy	GPIO	LQFP pin	WLCSP ball	Number of available pins	Usage	LQFP Pin	Number of available pins	Usage	BGA ball	Number of available pins	Usage	LQFP pin	Number of available pins	Usage
Group 1	G1_IO1	PA0	14	F6	4	3 channels with 1 sampling capacitor	23	4	3 channels with 1 sampling capacitor	L2	4	3 channels with 1 sampling capacitor	34	4	3 channels with 1 sampling capacitor
	G1_IO2	PA1	15	E6			24			M2			35		
	G1_IO3	PA2	16	H8			25			K3			36		
	G1_IO4	PA3(1)	17	G7			26			L3			37		
Group 2	G2_IO1	PA6	22	G5	2	1 channel with 1 sampling capacitor	31	2	1 channel with 1 sampling capacitor	L4	4(2)	3 channels with 1 sampling capacitor	42	4(2)	3 channels with 1 sampling capacitor
	G2_IO2	PA7	23	G4			32			J5			43		
	G2_IO3	PF15	-	-			-			J9			55		
	G2_IO4	PG0(3)	-	-			-			H9			56		
	G2_IO5	PG1(3)	-	-			-			G9			57		
Group 3	G3_IO1	PB0(1)	26	H4	3	2 channels with 1 sampling capacitor	35	3	2 channels with 1 sampling capacitor	M5	5	4 channels with 1 sampling capacitor	46	5	4 channels with 1 sampling capacitor
	G3_IO2	PB1	27	F4			36			M6			47		
	G3_IO3	PB2	28	H3			37			L6			48		
	G3_IO4	PF11	-	-			-			K6			49		
	G3_IO5	PF12	-	-			-			J7			50		

Table 38. Available touch sensing channels for *STM32L1xx 384K* (continued)

Subfamily			STM32L1xx 384K												
Packages			LQFP64 / WLCSP64				LQFP100			UFBGA132			LQFP144		
Part numbers			STM32L151RD STM32L152RD STM32L162RD				STM32L151VD STM32L152VD STM32L162VD			STM32L151QD STM32L152QD STM32L162QD			STM32L151ZD STM32L152ZD STM32L162ZD		
Analog I/O group	Gx_IOy	GPIO	LQFP pin	WLCSP ball	Number of available pins	Usage	LQFP Pin	Number of available pins	Usage	BGA ball	Number of available pins	Usage	LQFP pin	Number of available pins	Usage
Group 4	G4_IO1	PA8	41	E4	3	2 channels with 1 sampling capacitor	67	3	2 channels with 1 sampling capacitor	D11	3	2 channels with 1 sampling capacitor	100	3	2 channels with 1 sampling capacitor
	G4_IO2	PA9	42	D2			68			D10			101		
	G4_IO3	PA10	43	D3			69			C12			102		
Group 5	G5_IO1	PA13	46	D4	3	2 channels with 1 sampling capacitor	72	3	2 channels with 1 sampling capacitor	A11	3	2 channels with 1 sampling capacitor	105	3	2 channels with 1 sampling capacitor
	G5_IO2	PA14	49	B2			76			A10			109		
	G5_IO3	PA15	50	C3			77			A9			110		
Group 6	G6_IO1	PB4	56	B4	4	3 channels with 1 sampling capacitor	90	4	3 channels with 1 sampling capacitor	A7	4	3 channels with 1 sampling capacitor	134	4	3 channels with 1 sampling capacitor
	G6_IO2	PB5	57	A5			91			C5			135		
	G6_IO3	PB6	58	B5			92			B5			136		
	G6_IO4	PB7	59	C5			93			B4			137		

Table 38. Available touch sensing channels for *STM32L1xx 384K* (continued)

Subfamily			STM32L1xx 384K												
Packages			LQFP64 / WLCSP64				LQFP100			UFBGA132			LQFP144		
Part numbers			STM32L151RD STM32L152RD STM32L162RD				STM32L151VD STM32L152VD STM32L162VD			STM32L151QD STM32L152QD STM32L162QD			STM32L151ZD STM32L152ZD STM32L162ZD		
Analog I/O group	Gx_IOy	GPIO	LQFP pin	WLCSP ball	Number of available pins	Usage	LQFP Pin	Number of available pins	Usage	BGA ball	Number of available pins	Usage	LQFP pin	Number of available pins	Usage
Group 7	G7_IO1	PB12	33	G2	4	3 channels with 1 sampling capacitor	51	4	3 channels with 1 sampling capacitor	L12	5(2)	4 channels with 1 sampling capacitor	73	5(2)	4 channels with 1 sampling capacitor
	G7_IO2	PB13	34	G1			52			K12			74		
	G7_IO3	PB14	35	F2			53			K11			75		
	G7_IO4	PB15	36	F1			54			K10			76		
	G7_IO5	PG2(3)	-	-			-			G10			87		
	G7_IO6	PG3(3)	-	-			-			F9			88		
	G7_IO7	PG4(3)	-	-			-			F10			89		
Group 8	G8_IO1	PC0	8	E8	4	3 channels with 1 sampling capacitor	15	4	3 channels with 1 sampling capacitor	H1	4	3 channels with 1 sampling capacitor	26	4	3 channels with 1 sampling capacitor
	G8_IO2	PC1	9	F8			16			J2			27		
	G8_IO3	PC2	10	D6			17			J3			28		
	G8_IO4	PC3(1)	11	F7			18			K2			29		
Group 9	G9_IO1	PC4	24	H6	2	1 channel with 1 sampling capacitor	33	2	1 channel with 1 sampling capacitor	K5	4	3 channels with 1 sampling capacitor	44	4	3 channels with 1 sampling capacitor
	G9_IO2	PC5	25	H5			34			L5			45		
	G9_IO3	PF13	-	-			-			K7			53		
	G9_IO4	PF14	-	-			-			J8			54		

Table 38. Available touch sensing channels for *STM32L1xx 384K* (continued)

Subfamily			STM32L1xx 384K												
Packages			LQFP64 / WLCSP64				LQFP100			UFBGA132			LQFP144		
Part numbers			STM32L151RD STM32L152RD STM32L162RD				STM32L151VD STM32L152VD STM32L162VD			STM32L151QD STM32L152QD STM32L162QD			STM32L151ZD STM32L152ZD STM32L162ZD		
Analog I/O group	Gx_IOy	GPIO	LQFP pin	WLCSP ball	Number of available pins	Usage	LQFP Pin	Number of available pins	Usage	BGA ball	Number of available pins	Usage	LQFP pin	Number of available pins	Usage
Group 10	G10_IO1	PC6	37	E1	4	3 channels with 1 sampling capacitor	63	4	3 channels with 1 sampling capacitor	E12	4	3 channels with 1 sampling capacitor	96	4	3 channels with 1 sampling capacitor
	G10_IO2	PC7	38	E2			64			E11			97		
	G10_IO3	PC8	39	E3			65			E10			98		
	G10_IO4	PC9	40	D1			66			D12			99		
Group 11	G11_IO1	PF6	-	-	0	Cannot be used for touch sensing	-	0	Cannot be used for touch sensing	G3	4	3 channels with 1 sampling capacitor	18	5	4 channels with 1 sampling capacitor
	G11_IO2	PF7	-	-			-			G4			19		
	G11_IO3	PF8	-	-			-			H4			20		
	G11_IO4	PF9	-	-			-			J6			21		
	G11_IO5	PF10	-	-			-			-			22		
Maximum number of channels			23 channels with 10 sampling capacitors				23 channels with 10 sampling capacitors			33 channels with 11 sampling capacitors			34 channels with 11 sampling capacitors		

1. This GPIO offers a reduced touch sensing sensitivity. It is thus recommended to use it as sampling capacitor I/O.
2. Not all the pins are available simultaneously on this group.
3. This GPIO can only be configured as sampling capacitor I/O when using HW acquisition mode and as channel I/O when using SW acquisition mode.

Table 39. Available touch sensing channels for STM32L1xx 256K (table 1/2)

Subfamily			STM32L1xx 256K									
Packages			LQFP48 or UFQFPN48			WLCSP63			LQFP64 / WLCSP64			
Part numbers			STM32L152CC			STM32L151UC			STM32L151RC STM32L152RC STM32L162RC			
Analog I/O group	Gx_IOy	GPIO	Pin	Number of available pins	Usage	WLCSP ball	Number of available pins	Usage	LQFP pin	WLCSP ball	Number of available pins	Usage
Group 1	G1_IO1	PA0	10	4	3 channels with 1 sampling capacitor	E4	4	3 channels with 1 sampling capacitor	14	F6	4	3 channels with 1 sampling capacitor
	G1_IO2	PA1	11			G5			15	E6		
	G1_IO3	PA2	12			H6			16	H8		
	G1_IO4	PA3(1)	13			J7			17	G7		
Group 2	G2_IO1	PA6	16	2	1 channel with 1 sampling capacitor	G4	2	1 channel with 1 sampling capacitor	22	G5	2	1 channel with 1 sampling capacitor
	G2_IO2	PA7	17			J5			23	G4		
	G2_IO3	PF15	-			-			-	-		
	G2_IO4	PG0(2)	-			-			-	-		
	G2_IO5	PG1(2)	-			-			-	-		
Group 3	G3_IO1	PB0(1)	18	3	2 channels with 1 sampling capacitor	J3	3	2 channels with 1 sampling capacitor	26	H4	3	2 channels with 1 sampling capacitor
	G3_IO2	PB1	19			H3			27	F4		
	G3_IO3	PB2	20			G3			28	H3		
	G3_IO4	PF11	-			-			-	-		
	G3_IO5	PF12	-			-			-	-		
Group 4	G4_IO1	PA8	29	3	2 channels with 1 sampling capacitor	E3	3	2 channels with 1 sampling capacitor	41	E4	3	2 channels with 1 sampling capacitor
	G4_IO2	PA9	30			C1			42	D2		
	G4_IO3	PA10	31			D2			43	D3		

Table 39. Available touch sensing channels for STM32L1xx 256K (table 1/2) (continued)

Subfamily			STM32L1xx 256K									
Packages			LQFP48 or UFQFPN48			WLCSP63			LQFP64 / WLCSP64			
Part numbers			STM32L152CC			STM32L151UC			STM32L151RC STM32L152RC STM32L162RC			
Analog I/O group	Gx_IOy	GPIO	Pin	Number of available pins	Usage	WLCSP ball	Number of available pins	Usage	LQFP pin	WLCSP ball	Number of available pins	Usage
Group 5	G5_IO1	PA13	34	3	2 channels with 1 sampling capacitor	C2	3	2 channels with 1 sampling capacitor	46	D4	3	2 channels with 1 sampling capacitor
	G5_IO2	PA14	37			C3			49	B2		
	G5_IO3	PA15	38			A2			50	C3		
Group 6	G6_IO1	PB4	40	4	3 channels with 1 sampling capacitor	D4	4	3 channels with 1 sampling capacitor	56	B4	4	3 channels with 1 sampling capacitor
	G6_IO2	PB5	41			A5			57	A5		
	G6_IO3	PB6	42			B5			58	B5		
	G6_IO4	PB7	43			C5			59	C5		
Group 7	G7_IO1	PB12	25	4	3 channels with 1 sampling capacitor	G2	4	3 channels with 1 sampling capacitor	33	G2	4	3 channels with 1 sampling capacitor
	G7_IO2	PB13	26			G1			34	G1		
	G7_IO3	PB14	27			F3			35	F2		
	G7_IO4	PB15	28			F2			36	F1		
	G7_IO5	PG2(2)	-			-			-	-		
	G7_IO6	PG3(2)	-			-			-	-		
	G7_IO7	PG4(2)	-			-			-	-		

Table 39. Available touch sensing channels for STM32L1xx 256K (table 1/2) (continued)

Subfamily			STM32L1xx 256K									
Packages			LQFP48 or UFQFPN48			WLCSP63			LQFP64 / WLCSP64			
Part numbers			STM32L152CC			STM32L151UC			STM32L151RC STM32L152RC STM32L162RC			
Analog I/O group	Gx_IOy	GPIO	Pin	Number of available pins	Usage	WLCSP ball	Number of available pins	Usage	LQFP pin	WLCSP ball	Number of available pins	Usage
Group 8	G8_IO1	PC0	-	0	Cannot be used for touch sensing	E6	4	3 channels with 1 sampling capacitor	8	E8	4	3 channels with 1 sampling capacitor
	G8_IO2	PC1	-			E5			9	F8		
	G8_IO3	PC2	-			G7			10	D6		
	G8_IO4	PC3	-			G6			11	F7		
Group 9	G9_IO1	PC4	-	0		F4	2	1 channel with 1 sampling capacitor	24	H6	2	1 channel with 1 sampling capacitor
	G9_IO2	PC5	-			J4			25	H5		
	G9_IO3	PF13	-			-			-	-		
	G9_IO4	PF14	-			-			-	-		
Group 10	G10_IO1	PC6	-	0		F1	4	3 channels with 1 sampling capacitor	37	E1	4	3 channels with 1 sampling capacitor
	G10_IO2	PC7	-			E1			38	E2		
	G10_IO3	PC8	-			D1			39	E3		
	G10_IO4	PC9	-			E2			40	D1		
Group11	G11_IO1	PF6	-	0		-	0	Cannot be used for touch sensing	-	-	0	Cannot be used for touch sensing
	G11_IO2	PF7	-			-			-	-		
	G11_IO3	PF8	-			-			-	-		
	G11_IO4	PF9	-			-			-	-		
	G11_IO5	PF10	-			-			-	-		

Table 39. Available touch sensing channels for STM32L1xx 256K (table 1/2) (continued)

Subfamily			STM32L1xx 256K									
Packages			LQFP48 or UFQFPN48			WLCSP63			LQFP64 / WLCSP64			
Part numbers			STM32L152CC			STM32L151UC			STM32L151RC STM32L152RC STM32L162RC			
Analog I/O group	Gx_IOy	GPIO	Pin	Number of available pins	Usage	WLCSP ball	Number of available pins	Usage	LQFP pin	WLCSP ball	Number of available pins	Usage
Maximum number of channels			16 channels with 7 sampling capacitors			23 channels with 10 sampling capacitors			23 channels with 10 sampling capacitors			

1. This GPIO offers a reduced touch sensing sensitivity. It is thus recommended to use it as sampling capacitor I/O.
2. This GPIO can only be configured as sampling capacitor I/O when using HW acquisition mode and as channel I/O when using SW acquisition mode.

Table 40. Available touch sensing channels for STM32L1xx 256K (table 2/2)

Subfamily			STM32L1xx 256K									
Packages			LQFP100 / UFBGA100				UFBGA132			LQFP144		
Part numbers			STM32L151VC STM32L152VC STM32L162VC				STM32L151QC STM32L152QC STM32L162QC			STM32L151ZC STM32L152ZC STM32L162ZC		
Analog I/O group	Gx_IOy	GPIO	LQFP pin	BGA ball	Number of available pins	Usage	BGA ball	Number of available pins	Usage	LQFP pin	Number of available pins	Usage
Group 1	G1_IO1	PA0	23	L2	4	3 channels with 1 sampling capacitor	L2	4	3 channels with 1 sampling capacitor	34	4	3 channels with 1 sampling capacitor
	G1_IO2	PA1	24	M2			M2			35		
	G1_IO3	PA2	25	K3			K3			36		
	G1_IO4	PA3(1)	26	L3			L3			37		
Group 2	G2_IO1	PA6	31	L4	2	1 channel with 1 sampling capacitor	L4	4(2)	3 channels with 1 sampling capacitor	42	4(2)	3 channels with 1 sampling capacitor
	G2_IO2	PA7	32	M4			J5			43		
	G2_IO3	PF15	-	-			J9			55		
	G2_IO4	PG0(3)	-	-			H9			56		
	G2_IO5	PG1(3)	-	-			G9			57		
Group 3	G3_IO1	PB0(1)	35	M5	3	2 channels with 1 sampling capacitor	M5	5	4 channels with 1 sampling capacitor	46	5	4 channels with 1 sampling capacitor
	G3_IO2	PB1	36	M6			M6			47		
	G3_IO3	PB2	37	L6			L6			48		
	G3_IO4	PF11	-	-			K6			49		
	G3_IO5	PF12	-	-			J7			50		
Group 4	G4_IO1	PA8	67	D11	3	2 channels with 1 sampling capacitor	D11	3	2 channels with 1 sampling capacitor	100	3	2 channels with 1 sampling capacitor
	G4_IO2	PA9	68	D10			D10			101		
	G4_IO3	PA10	69	C12			C12			102		

Table 40. Available touch sensing channels for STM32L1xx 256K (table 2/2) (continued)

Subfamily			STM32L1xx 256K									
Packages			LQFP100 / UFBGA100				UFBGA132			LQFP144		
Part numbers			STM32L151VC STM32L152VC STM32L162VC				STM32L151QC STM32L152QC STM32L162QC			STM32L151ZC STM32L152ZC STM32L162ZC		
Analog I/O group	Gx_IOy	GPIO	LQFP pin	BGA ball	Number of available pins	Usage	BGA ball	Number of available pins	Usage	LQFP pin	Number of available pins	Usage
Group 5	G5_IO1	PA13	72	A11	3	2 channels with 1 sampling capacitor	A11	3	2 channels with 1 sampling capacitor	105	3	2 channels with 1 sampling capacitor
	G5_IO2	PA14	76	A10			A10			109		
	G5_IO3	PA15	77	A9			A9			110		
Group 6	G6_IO1	PB4	90	A7	4	3 channels with 1 sampling capacitor	A7	4	3 channels with 1 sampling capacitor	134	4	3 channels with 1 sampling capacitor
	G6_IO2	PB5	91	C5			C5			135		
	G6_IO3	PB6	92	B5			B5			136		
	G6_IO4	PB7	93	B4			B4			137		
Group 7	G7_IO1	PB12	51	L12	4	3 channels with 1 sampling capacitor	L12	5(2)	4 channels with 1 sampling capacitor	73	5(2)	4 channels with 1 sampling capacitor
	G7_IO2	PB13	52	K12			K12			74		
	G7_IO3	PB14	53	K11			K11			75		
	G7_IO4	PB15	54	K10			K10			76		
	G7_IO5	PG2(3)	-	-			G10			87		
	G7_IO6	PG3(3)	-	-			F9			88		
	G7_IO7	PG4(3)	-	-			F10			89		
Group 8	G8_IO1	PC0	15	H1	4	3 channels with 1 sampling capacitor	H1	4	3 channels with 1 sampling capacitor	26	4	3 channels with 1 sampling capacitor
	G8_IO2	PC1	16	J2			J2			27		
	G8_IO3	PC2	17	J3			J3			28		
	G8_IO4	PC3	18	K2			K2(3)			29(3)		

Table 40. Available touch sensing channels for STM32L1xx 256K (table 2/2) (continued)

Subfamily			STM32L1xx 256K									
Packages			LQFP100 / UFBGA100				UFBGA132			LQFP144		
Part numbers			STM32L151VC STM32L152VC STM32L162VC				STM32L151QC STM32L152QC STM32L162QC			STM32L151ZC STM32L152ZC STM32L162ZC		
Analog I/O group	Gx_IOy	GPIO	LQFP pin	BGA ball	Number of available pins	Usage	BGA ball	Number of available pins	Usage	LQFP pin	Number of available pins	Usage
Group 9	G9_IO1	PC4	33	K5	2	1 channel with 1 sampling capacitor	K5	4	3 channels with 1 sampling capacitor	44	4	3 channels with 1 sampling capacitor
	G9_IO2	PC5	34	L5			L5			45		
	G9_IO3	PF13	-	-			K7			53		
	G9_IO4	PF14	-	-			J8			54		
Group 10	G10_IO1	PC6	63	E12	4	3 channels with 1 sampling capacitor	E12	4	3 channels with 1 sampling capacitor	96	4	3 channels with 1 sampling capacitor
	G10_IO2	PC7	64	E11			E11			97		
	G10_IO3	PC8	65	E10			E10			98		
	G10_IO4	PC9	66	D12			D12			99		
Group11	G11_IO1	PF6	-	-	0	Cannot be used for touch sensing	G3	4	3 channels with 1 sampling capacitor	18	5	4 channels with 1 sampling capacitor
	G11_IO2	PF7	-	-			G4			19		
	G11_IO3	PF8	-	-			H4			20		
	G11_IO4	PF9	-	-			J6			21		
	G11_IO5	PF10	-	-			-			22		
Maximum number of channels			23 channels with 10 sampling capacitors				33 channels with 11 sampling capacitors			34 channels with 11 sampling capacitors		

1. This GPIO offers a reduced touch sensing sensitivity. It is thus recommended to use it as sampling capacitor I/O.
2. Not all the pins are available simultaneously on this group.
3. This GPIO can only be configured as sampling capacitor I/O when using HW acquisition mode and as channel I/O when using SW acquisition mode.

Table 41. Available touch sensing channels for STM32L15x 32K to 128K

Subfamily			STM32L15x 32K to 128K										
Packages			LQFP48 / VFQFPN48			LQFP64 / BGA64				LQFP100 / BGA100			
Part numbers			STM32L151C6 STM32L151C8 STM32L151CB STM32L152C6 STM32L152C8 STM32L152CB			STM32L151R6 STM32L151R8 STM32L151RB STM32L152R6 STM32L152R8 STM32L152RB				STM32L151V8 STM32L151VB STM32L152V8 STM32L152VB			
Analog I/O group	Gx_IOy	GPIO	Pin	Number of available pins	Usage	LQFP pin	BGA ball	Number of available pins	Usage	LQFP pin	BGA ball	Number of available pins	Usage
Group 1	G1_IO1	PA0	10	4	3 channels with 1 sampling capacitor	14	G2	4	3 channels with 1 sampling capacitor	23	L2	4	3 channels with 1 sampling capacitor
	G1_IO2	PA1	11			15	H2			24	M2		
	G1_IO3	PA2	12			16	F3			25	K3		
	G1_IO4	PA3	13			17	G3			26	L3		
Group 2	G2_IO1	PA6	16	2	1 channel with 1 sampling capacitor	22	G4	2	1 channel with 1 sampling capacitor	31	L4	2	1 channel with 1 sampling capacitor
	G2_IO2	PA7	17			23	H4			32	M4		
Group 3	G3_IO1	PB0	18	2	1 channel with 1 sampling capacitor	26	F5	2	1 channel with 1 sampling capacitor	35	M5	2	1 channel with 1 sampling capacitor
	G3_IO2	PB1	19			27	G5			36	M6		
Group 4	G4_IO1	PA8	29	3	2 channels with 1 sampling capacitor	41	D7	3	2 channels with 1 sampling capacitor	67	D11	3	2 channels with 1 sampling capacitor
	G4_IO2	PA9	30			42	C7			68	D10		
	G4_IO3	PA10	31			43	C6			69	C12		
Group 5	G5_IO1	PA13	34	3	2 channels with 1 sampling capacitor	46	A8	3	2 channels with 1 sampling capacitor	72	A11	3	2 channels with 1 sampling capacitor
	G5_IO2	PA14	37			49	A7			76	A10		
	G5_IO3	PA15	38			50	A6			77	A9		

Table 41. Available touch sensing channels for STM32L15x 32K to 128K (continued)

Subfamily			STM32L15x 32K to 128K										
Packages			LQFP48 / VFQFPN48			LQFP64 / BGA64				LQFP100 / BGA100			
Part numbers			STM32L151C6 STM32L151C8 STM32L151CB STM32L152C6 STM32L152C8 STM32L152CB			STM32L151R6 STM32L151R8 STM32L151RB STM32L152R6 STM32L152R8 STM32L152RB				STM32L151V8 STM32L151VB STM32L152V8 STM32L152VB			
Analog I/O group	Gx_IOy	GPIO	Pin	Number of available pins	Usage	LQFP pin	BGA ball	Number of available pins	Usage	LQFP pin	BGA ball	Number of available pins	Usage
Group 6	G6_IO1	PB4	40	2	1 channel with 1 sampling capacitor	56	A4	2	1 channel with 1 sampling capacitor	90	A7	2	1 channel with 1 sampling capacitor
	G6_IO2	PB5	41			57	C4			91	C5		
Group 7	G7_IO1	PB12	25	4	3 channels with 1 sampling capacitor	33	H8	4	3 channels with 1 sampling capacitor	51	L12	4	3 channels with 1 sampling capacitor
	G7_IO2	PB13	26			34	G8			52	K12		
	G7_IO3	PB14	27			35	F8			53	K11		
	G7_IO4	PB15	28			36	F7			54	K10		

Table 41. Available touch sensing channels for STM32L15x 32K to 128K (continued)

Subfamily			STM32L15x 32K to 128K										
Packages			LQFP48 / VFQFPN48			LQFP64 / BGA64				LQFP100 / BGA100			
Part numbers			STM32L151C6 STM32L151C8 STM32L151CB STM32L152C6 STM32L152C8 STM32L152CB			STM32L151R6 STM32L151R8 STM32L151RB STM32L152R6 STM32L152R8 STM32L152RB				STM32L151V8 STM32L151VB STM32L152V8 STM32L152VB			
Analog I/O group	Gx_IOy	GPIO	Pin	Number of available pins	Usage	LQFP pin	BGA ball	Number of available pins	Usage	LQFP pin	BGA ball	Number of available pins	Usage
Group 8	G8_IO1	PC0	-	0	Cannot be used for touch sensing	8	E3	4/3	3/2 channels with 1 sampling capacitor	15	H1	4	3 channels with 1 sampling capacitor
	G8_IO2	PC1	-			9	E2			16	J2		
	G8_IO3	PC2	-			10	F2			17	J3		
	G8_IO4	PC3	-			11	-			18	K2		
Group 9	G9_IO1	PC4	-	0		24	H5	2	1 channel with 1 sampling capacitor	33	K5	2	1 channel with 1 sampling capacitor
	G9_IO2	PC5	-			25	H6			34	L5		
Group 10	G10_IO1	PC6	-	0		37	F6	4	3 channels with 1 sampling capacitor	63	E12	4	3 channels with 1 sampling capacitor
	G10_IO2	PC7	-			38	E7			64	E11		
	G10_IO3	PC8	-			39	E8			65	E10		
	G10_IO4	PC9	-			40	D8			66	D12		
Maximum number of channels			13 channels with 7 sampling capacitors			20/19 channels with 10 sampling capacitors				20 channels with 10 sampling capacitors			

2.23.7 Hardware implementation example

Figure 21 shows an example of hardware implementation on STM32L1xx devices.

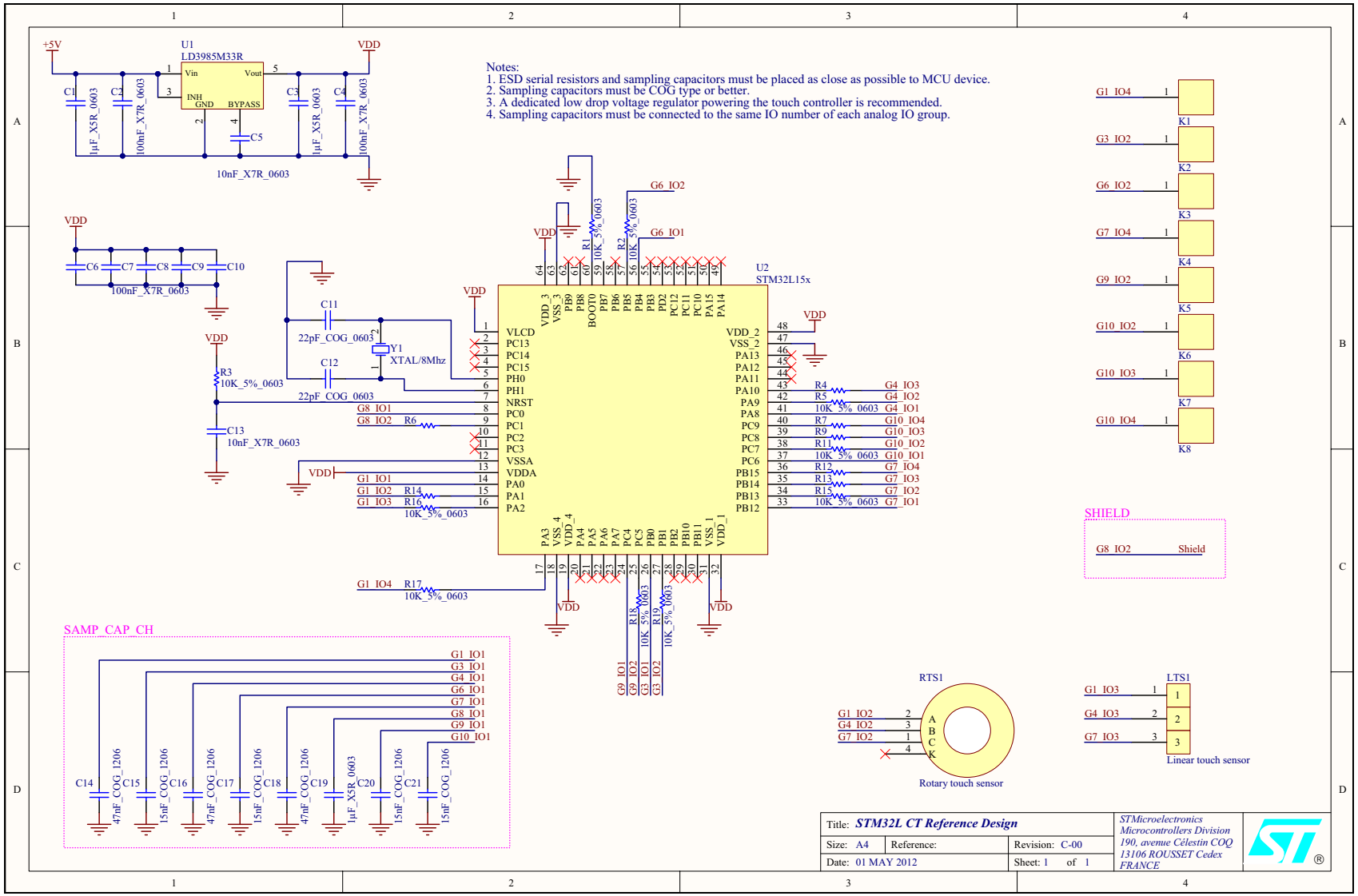
Recommendations to increase the noise immunity on the PCB

To ensure a correct operation in noisy environment, the floating nets must be avoided (tracks, copper elements, conductive frames, etc...).

As a consequence:

- All unused touch controller I/Os must be either configured to output push-pull low or externally tied to GND.
- The parameter TSLPRM_IODEF should also be configured to the output push-pull low state.
- We recommend to drive the sampling capacitor common node using a standard I/O of the touch controller configured in output push-pull low mode.
- It may also be required to add a capacitor-input filter (pi filter) on each channel line.

Figure 21. STM32L1xx hardware implementation example



3 Getting started

3.1 Create your application

Start with an example present in the STMTouch library of the device you intend to use. Take an example that is close in term of number of channels/sensors with your target application. Copy and paste the example in the same parent folder and rename it according your target application. Then modify the files as described below.

The following sections describe the necessary steps to create a new application project.

3.1.1 Toolchain compiler preprocessor section

The device that you intend to use must be written in the **toolchain compiler preprocessor section** of your project.

These defines are the same as those for the standard peripherals library. Please see the stm<xxx>.h map file to have the list of the devices definition**.**

Note: The hardware acquisition mode is selected per default for the STM8L15x Low-density devices, and STM32L1xx (excepted Medium-density) devices. If you want to use the software acquisition mode you must add the following constant in the toolchain compiler preprocessor:

- TSLPRM_STM8L1XX_SW_ACQ
- TSLPRM_STM32L1XX_SW_ACQ

3.1.2 The tsl_conf file

The **tsl_conf_<mcu>.h** file contains all the STMTouch driver parameters. The following edits must be done:

1. Change the number of channels, banks, sensors according your application.
2. Change the common parameters: thresholds, debounce, ECS, DTO, etc...
3. Change the parameters specific to the device.

3.1.3 The main file

The **main.c** and **main.h** files contain the application code itself (LEDs and LCD management, etc...) and the call to the STMTouch driver initialization and action functions.

3.1.4 The tsl_user file

The **tsl_user.c** and **tsl_user.h** files contain the STMTouch driver configuration (definition of channels, banks, zones, sensors, etc...) and the STMTouch driver initialization (**TSL_user_Init**) and action (**TSL_user_Action**) functions.

Create the channels variables using the structures (**mandatory**):

- TSL_ChannelSrc_T
- TSL_ChannelDest_T
- TSL_ChannelData_T

Create the Banks variables using the structures (**mandatory**):

- TSL_Bank_T

Create the Zone variables using the structure (optional):

- TSL_Zone_T

Create the touchkeys variables using the structures (optional):

- TSL_TouchKeyData_T
- TSL_TouchKeyParam_T
- TSL_State_T
- TSL_TouchKeyMethods_T
- TSL_TouchKeyB_T
- TSL_TouchKey_T

Create the Linear and Rotary touch sensors variables using the structures (optional):

- TSL_LinRotData_T
- TSL_LinRotParam_T
- TSL_State_T
- TSL_LinRotMethods_T
- TSL_LinRotB_T
- TSL_LinRot_T

Create the generic sensors (objects) variables using the structures (**mandatory**):

- TSL_Object_T
- TSL_ObjectGroup_T

The **TSL_user_Init()** function contains the initialization of the STMTouch driver. Modify this function to take into account your bank array name and object groups names.

The **TSL_user_Action()** function contains the main state machine. Modify it also if you have several object groups to process or to change the ECS period, etc...

3.2 Debug with STM Studio

The STM Studio software is very useful to observe variables of the STMTouch driver. Thanks to its powerful features you will be able to better understand how the sensors behave and to find the better parameters to apply.

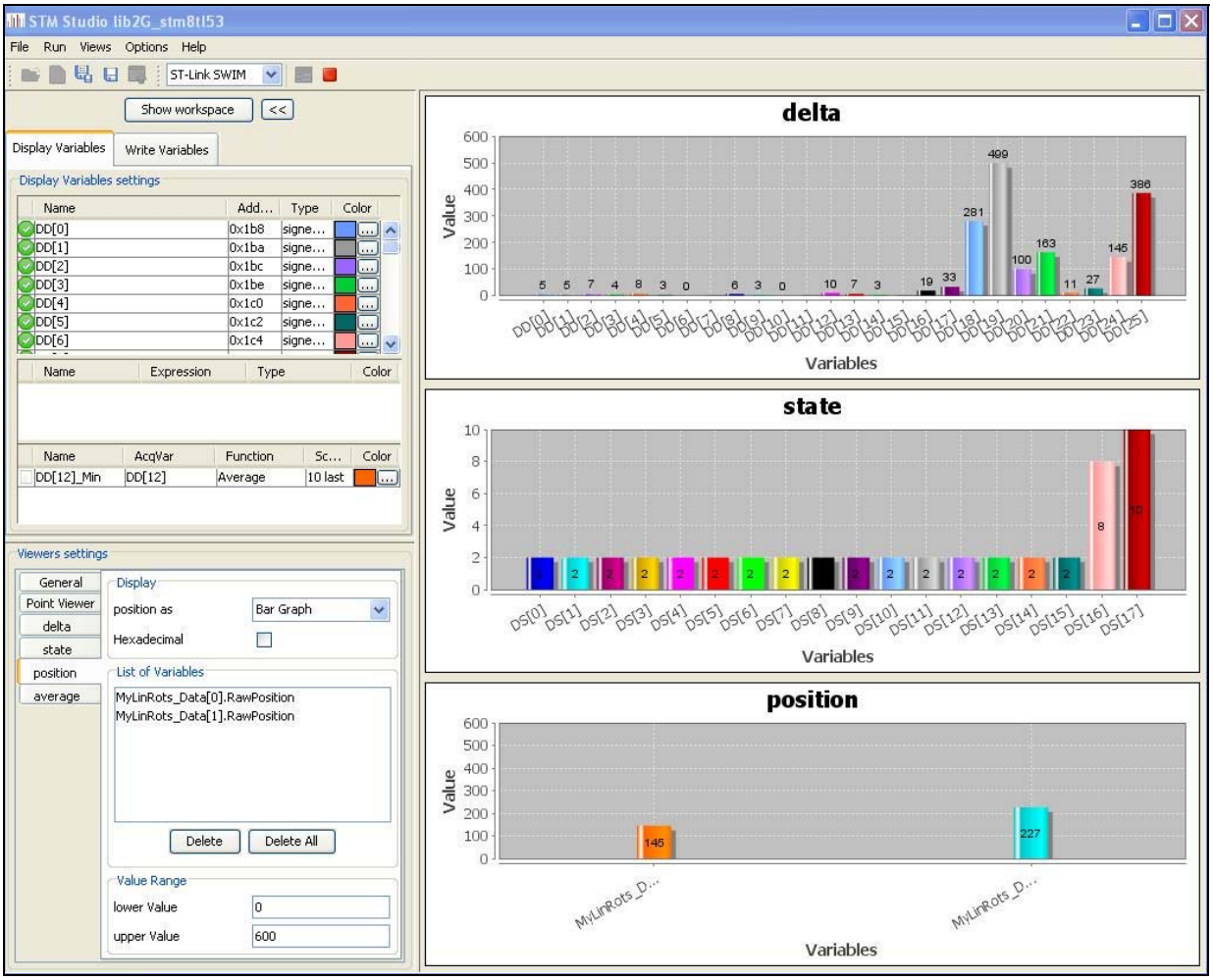
This section does not intend to explain how to use this tool, but give some advice to better understand and debug your application.

This is a non-exhaustive list of the STMTouch driver variables to observe:

- The **channels measure**, **reference** and **delta**. These variables are present inside the **TSL_ChannelData_T** structure. This is useful to adjust the **thresholds** parameters.
- The **sensors state** present in the **TSL_TouchKeyData_T** and **TSL_LinRotData_T** structures. This is useful to adjust the **Debounce**, **ECS** and **DTO** parameters.
- The linear and rotary touch sensors **position** in the **TSL_LinRotData_T** structure.

The following snapshot is an example of data visualization on STM Studio:

Figure 22. STM Studio snapshot

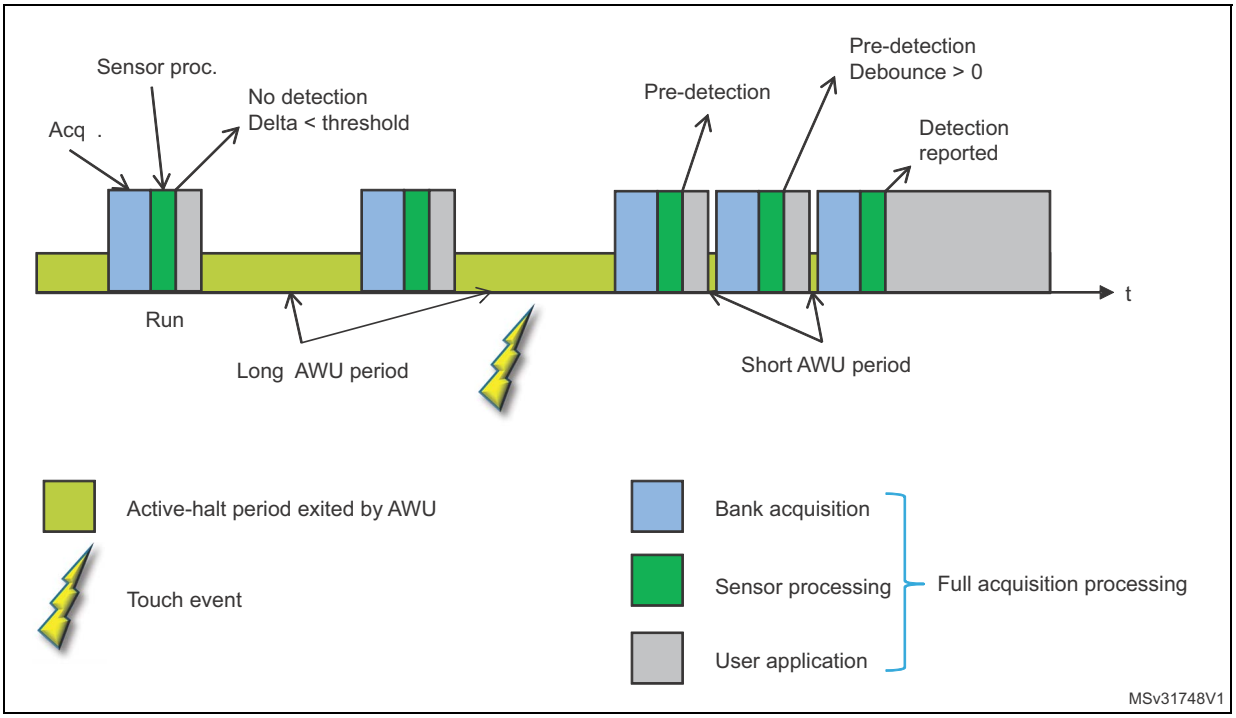


3.3 Low-power strategy

The following figure shows the acquisition sequencing for a single bank acquisition in low-power mode.

To reduce the power consumption, the acquisitions are sequenced with a long delay in between. During this delay, the CPU can be in low-power mode (active-halt for STM8 or STOP for STM32). This delay can be shortened or even removed between two consecutive acquisitions when the delta becomes greater than a detection threshold (proximity or touch). The long delay is restored if all the sensors return in RELEASE state.

Figure 23. Low_power strategy



This approach allows to save power consumption without increasing the response time. The maximum response time is obtained when a touch occurs during the sensor processing. It can be expressed as followed:

Max Response Time = long AWU period + (n) x short AWU period + (n+2) x full acquisition processing - bank acquisition

with n being the debounce value.

3.4 Main differences with previous library

This section describes the main differences between the previous touch sensing library (named **STMxxx_TouchSensing_Lib**) and this new STMTouch Library (named **STMxxx_STMTouch_Lib**).

3.4.1 Files

Previous Library

- Files are prefixed with “stm8_tsl_” or “stm32_tsl_” or “<mcu>_tsl_”.
- Copy <mcu>_tsl_conf_CT_TOADAPT.h or <mcu>_tsl_conf_RC_TOADAPT.h in your project and rename it in <mcu>_tsl_conf.h
- Include <mcu>_tsl_api.h in main.h

STMTouch Library

- All files are prefixed with “tsl” or “tsl_”. There is no more “stm8” nor “stm32” prefix.
- Copy **tsl_conf_<mcu>.h_TOADAPT** in your project and rename it in **tsl_conf_<mcu>.h**
- Include **tsl.h** in **tsl_user.h**

3.4.2 Channels, banks and sensors configuration

Previous Library

- Done in stm8_tsl_conf.h
- Limited number of channels, banks and sensors.
- SCKeys (= single channel keys) and MCKeys (multi-channels keys) sensors.

STMTouch library

- Done in the application (**tsl_user.c** and **tsl_user.h**).
- No limitation in the number of Cchannels, banks and sensors.
- Touchkey and LinRot sensors with extended and basic types.

3.4.3 Parameters configuration

Previous library

- Done in stm8_tsl_conf.h
- No common prefix for the parameters.
- Thresholds are on 7 bits (signed).
- No thresholds for proximity.
- Fixed calibration samples.
- Debounce counters common to all sensors and states.
- Fixed timer tick frequency.

STMTouch Library

- Done in **tsl_conf_<mcu>.h**
- All parameters are prefixed with “TSLPRM_”.
- Thresholds are on 8 bits (unsigned) and multiplier coefficient can be applied on them.
- Thresholds for proximity.
- Programmable calibration samples and delay.
- Debounce counters different for each sensor and each state.
- Programmable timer tick frequency.

3.4.4 Usage

Previous Library

- Initialization by calling TSL_Init() function.
- Execution by calling TSL_Action() continuously.
- Main state machine and sensors state machine are in the TS driver layer.
- ECS and DXS process are called by the TS driver layer.
- Test of the main state machine value using TSLState variable:

```
if (TSLState == TSL_IDLE_STATE) {
...
}
if ((TSLState == TSL_SCKEY_P1_ACQ_STATE) ||
(TSLState == TSL_SCKEY_P2_ACQ_STATE) ||
(TSLState == TSL_MCKEY1_ACQ_STATE) ||
(TSLState == TSL_MCKEY2_ACQ_STATE)) {
...
}
```

  - Test of sensor state using TSL_GlobalSetting, sSCKeyInfo and sMCKeyInfo variables:

```
if (TSL_GlobalSetting.b.CHANGED) {
TSL_GlobalSetting.b.CHANGED = 0;
if (sSCKeyInfo[0].Setting.b.DETECTED) {...}
```

STMTouch library

- Initialization by calling **TSL_obj_GroupInit()** and **TSL_Init()** functions.
- Execution by calling **TSL_Action()** continuously.
- Main state machine and sensors state machine are in the application layer and so can be changed easily.
- ECS and DXS process are called by the application layer.
- The test of the main state machine value is not needed as the main state machine is managed by the application layer.
- Test of sensor state using **StateId** variable:

```
if ((MyTKeys[0].p_Data->StateId == TSL_STATEID_DETECT)||
(MyTKeys[0].p_Data->StateId == TSL_STATEID_DEB_RELEASE_DETECT)) {...}
```

3.4.5 Variables monitoring

Many variables can be monitored in order to debug your application. The list below is not exhaustive but shows only the most important variables used in both libraries:

Previous library

- The sSCKeyInfo, sMCKeyInfo and sAcqBankInfo structures contain all informations related to touchkey and linear/rotary sensors: reference, last measure, thresholds, state, position, EPCC, ..

STMTouch library

- The array of **TSL_ChannelData_T** structure contains the reference, measure and delta values of ALL channels used by the application:

Figure 24. Debug of TSL_ChannelData_T structure

Expression	Value	Location	Type
MyChannels_Data	<array>	0x20000030	TSL_ChannelData_T[3]
[0]	<struct>	0x20000030	TSL_ChannelData_T
Flags	<struct>	0x20000030	TSL_ChannelFlags_T
Ref	2414	0x20000032	uint16_t
RefRest	208	0x20000034	uint8_t
Delta	0	0x20000036	int16_t
Meas	2414	0x20000038	uint16_t
[1]	<struct>	0x2000003A	TSL_ChannelData_T
Flags	<struct>	0x2000003A	TSL_ChannelFlags_T
Ref	2642	0x2000003C	uint16_t
RefRest	245	0x2000003E	uint8_t
Delta	-1	0x20000040	int16_t
Meas	2643	0x20000042	uint16_t
[2]	<struct>	0x20000044	TSL_ChannelData_T
Flags	<struct>	0x20000044	TSL_ChannelFlags_T
Ref	2583	0x20000046	uint16_t
RefRest	46	0x20000048	uint8_t
Delta	1	0x2000004A	int16_t
Meas	2582	0x2000004C	uint16_t

- The array of **TSL_TouchKeyData_T** contains mainly the touchkey sensors state:

Figure 25. Debug of TSL_TouchKeyData_T structure

Expression	Value	Location	Type
MyTKeys_Data	<array>	0x20000084	TSL_TouchKeyData_T[3]
[0]	<struct>	0x20000084	TSL_TouchKeyData_T
StateId	TSL_STATEID_PROX	0x20000084	enum <Unnamed 90>
Counter	0	0x20000085	uint8_t
Change	TSL_STATE_NOT_CHANGED	0x20000085	enum <Unnamed 31>
DxSLock	TSL_FALSE	0x20000085	TSL_Bool_enum_T
[1]	<struct>	0x20000086	TSL_TouchKeyData_T
StateId	TSL_STATEID_DETECT	0x20000086	enum <Unnamed 90>
Counter	0	0x20000087	uint8_t
Change	TSL_STATE_NOT_CHANGED	0x20000087	enum <Unnamed 31>
DxSLock	TSL_FALSE	0x20000087	TSL_Bool_enum_T
[2]	<struct>	0x20000088	TSL_TouchKeyData_T
StateId	TSL_STATEID_PROX	0x20000088	enum <Unnamed 90>
Counter	0	0x20000089	uint8_t
Change	TSL_STATE_NOT_CHANGED	0x20000089	enum <Unnamed 31>
DxSLock	TSL_FALSE	0x20000089	TSL_Bool_enum_T

- The array of **TSL_LinRotData_T** contains the linear and rotary sensors state and position:

Figure 26. Debug of TSL_LinRotData_T structure

Expression	Value	Location	Type
MyLinRots_Data	<array>	0x20000074	TSL_LinRotData_T[1]
[0]	<struct>	0x20000074	TSL_LinRotData_T
StateId	TSL_STATEID_DETECT	0x20000074	enum <Unnamed 99>
RawPosition	157	0x20000075	uint8_t
Position	78	0x20000076	uint8_t
Counter	8	0x20000077	uint8_t
Change	TSL_STATE_NOT_CHANGED	0x20000077	enum <Unnamed 31>
PosChange	TSL_STATE_NOT_CHANGED	0x20000077	enum <Unnamed 31>
Counter2	3	0x20000078	uint8_t
DxSLock	TSL_FALSE	0x20000078	TSL_Bool_enum_T
Direction	TSL_FALSE	0x20000078	TSL_Bool_enum_T

- The arrays of **TSL_TouchKeyParam_T** and **TSL_LinRotParam_T** contain the touchkeys, linear and rotary sensors parameters (mainly thresholds and debounce counters):

Figure 27. Debug of TSL_TouchKeyParam_T

Expression	Value	Location	Type
MyTKeys_Param	<array>	0x20000050	TSL_TouchKeyParam_T[3]
[0]	<struct>	0x20000050	TSL_TouchKeyParam_T
ProxInTh	10	0x20000050	uint8_t
ProxOutTh	5	0x20000051	uint8_t
DetectInTh	80	0x20000052	uint8_t
DetectOutTh	70	0x20000053	uint8_t
CalibTh	50	0x20000054	uint8_t
CounterDebCalib	3	0x20000055	uint8_t
CounterDebProx	3	0x20000056	uint8_t
CounterDebDetect	3	0x20000057	uint8_t
CounterDebRelease	3	0x20000058	uint8_t
CounterDebError	3	0x20000059	uint8_t
[1]	<struct>	0x2000005A	TSL_TouchKeyParam_T
[2]	<struct>	0x20000064	TSL_TouchKeyParam_T

Figure 28. Debug of TSL_LinRotParam_T structures

Expression	Value	Location	Type
MyLinRots_Param	<array>	0x20000064	TSL_LinRotParam_T[1]
[0]	<struct>	0x20000064	TSL_LinRotParam_T
ProxInTh	15	0x20000064	uint8_t
ProxOutTh	10	0x20000065	uint8_t
DetectInTh	80	0x20000066	uint8_t
DetectOutTh	75	0x20000067	uint8_t
CalibTh	50	0x20000068	uint8_t
CounterDebCalib	3	0x20000069	uint8_t
CounterDebProx	3	0x2000006A	uint8_t
CounterDebDetect	3	0x2000006B	uint8_t
CounterDebRelease	3	0x2000006C	uint8_t
CounterDebError	3	0x2000006D	uint8_t
CounterDebDirection	3	0x2000006E	uint8_t
Resolution	7	0x2000006F	uint8_t
DirChangePos	10	0x20000070	uint8_t

3.5 Tips and tricks

3.5.1 Bank definition

For optimum sensitivity and position reporting, all the channels composing a linear or a rotary touch sensor must be acquired **simultaneously**. This means that all the channels must belong to the **same bank**.

Note: The library allows to define a linear or a rotary touch sensor with channels belonging to different banks. A such configuration induces a loss of sensitivity and a higher noise level. Moreover, depending on the acquisition time, it is also possible to observe a position change when removing the finger from the sensor.

For optimum performance of a linear or rotary sensor, all channels of such a sensor must be acquired simultaneously.Therefore all channels must belong to the same bank, which means the I/Os must be selected from different analog I/O groups. Please refer to datasheet for more information regarding I/O groups and available capacitive sensing GPIOs.

3.5.2 Channel assignment

It is recommended to assign GPIOs offering the same sensitivity level to all the channels composing a linear or a rotary touch sensor. Moreover, it is not recommended to use GPIOs offering a reduced sensitivity.

3.5.3 IO Default state parameter

For optimum acquisition noise level, it is recommended to set the **TSLPRM_TSC_IODEF**

or **TSLPRM_IODEF** parameter to **output push-pull low**.

However, if your application is using a linear or a rotary touch sensor with channels belonging to different banks, this parameter must be set to **input floating**. This will ensure optimum sensitivity.

4 Revision history

Table 42. Document revision history

Date	Revision	Changes
19-Mar-2013	1	Initial release.
03-Sep-2013	2	Updated document title. Updated *Section : Introduction*. Updated *Section 1.2: Naming conventions* Updated *Section 2.1.1: Supported devices*. Updated *Section 2.3: Main features*. Updated *Section 2.5.2: Resources*. Updated *Section 2.5.4: Usage example*. Updated *Section 2.6.2: Resources*. Added *Section 2.19: STM8L1xx devices*. Updated *Table 22: Available touch sensing channels for STM32F0xx* Updated *Section 3.1.1: Toolchain compiler preprocessor section*. Updated *Table 28: Available touch sensing channels for STM32F37x*. Updated *Table 35: MCU resources used on STM32L1xx with hardware acquisition*. Updated *Table 36: MCU resources used on STM32L1xx with software acquisition*.
17-Mar-2014	3	- Updated all channel tables of STM8L, STM32L and STM32F0 products. - Added STM32L1xx 512K channel table *Table 22: Available touch sensing channels for STM32F042* - Added STM32L1xx 512K channel table *Table 37: Available touch sensing channels for STM32L1xx 512K*: - Added *Chapter 3.5: Tips and tricks* - Updated *Section 2.1.1*; - Updated *Table 22* - Updated *Section 3.1.1* - Removed Table 42: STM8L1xx acquisition selection - Removed Table 43: STM32L1xx acquisition selection
22-Apr-2014	4	- Added note 1 on Pa4 of *Table 28*
14-Mar-2018	5	Removed UN_D Updated: – *Section 2.10.4: Electrodes placement* – *Figure 8: Electrodes designs* – *Table 3: Supported linear and rotary sensors*
17-Sep-2018	6	Updated: – *Section 2.10.2: Number of channels* – *Section 3.5.1: Bank definition*