AN3342
Application note

Getting started with microcontrollers of the STM8TL52/L53 line

Introduction

This application note describes the minimum hardware and software environment required to build an application with any microcontroller of the STM8TL52/L53 line.

The document includes a brief description of the main hardware components, details on power supply, reset control and ProxSense® lines and some hardware recommendations.

This application note also contains detailed reference design schematics with descriptions of the main components. The STM8 development tools and software toolchain are common to the STM8TL52/L53 line, the STM8L Series, the STM8S Series, the STM8AF Series and the STM8AL Series (please refer to *Section 9* and *Section 10*).

Section 11 describes how to set up the STM8 development environment.

Finally, *Section 12* provides a list of relevant documentation and online support resources.

This document applies to all microcontrollers of the STM8TL52/L53 line, but the examples provided are based on the STM8TL53xx devices.

Contents

List of tables

List of figures

1 Hardware requirements summary

To build an application around an STM8TL53xx device, the application board should provide the following features:

- Power supply (mandatory)
- Reset management (optional)
- ProxSense line management (optional)
- Debugging tool support: single-wire interface module (SWIM) connector (optional)

2 Power supply

2.1 Power supply overview

The microcontrollers of the STM8TL52/L53 line need to be powered by a 1.65 V to 3.6 V external source.

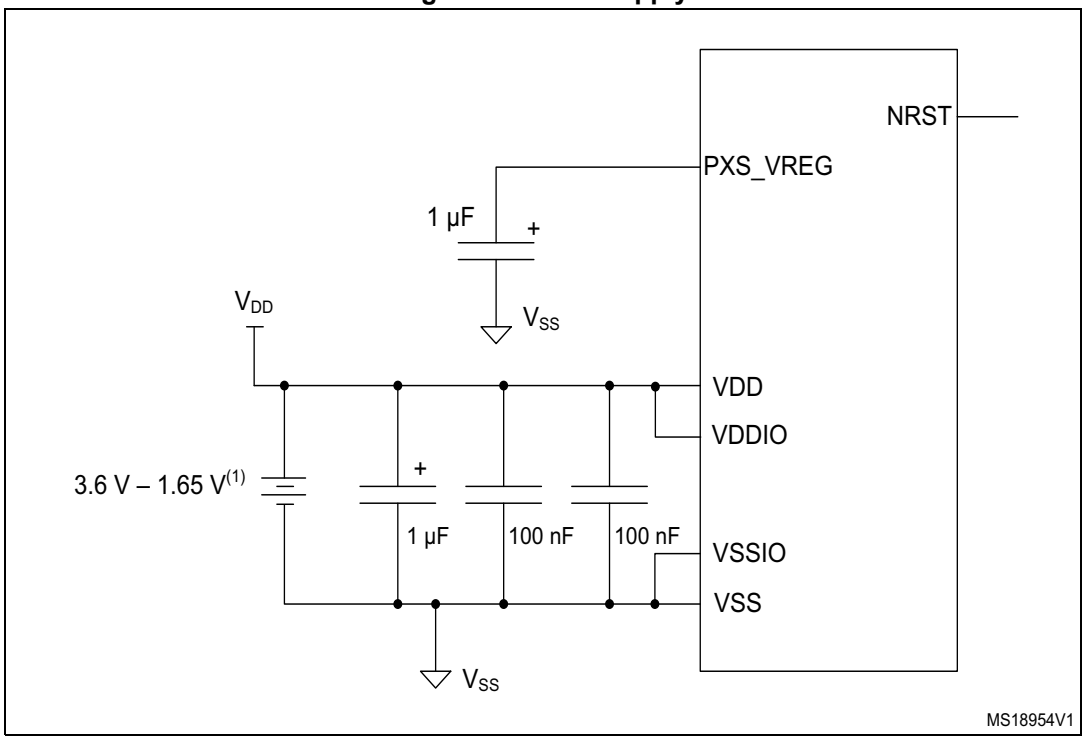
An on-chip power management system provides the constant digital supply to the core logic, both in normal and low-power modes. This on-chip management system ensures that the logic consumes a constant current level over the voltage range. It is also able to detect voltage drops and of generating a reset to avoid erratic behavior.

The STM8TL53xx devices provide:

- One pair of power supply pins (VDD/VSS) for the main operating voltage (1.65 V to 3.6 V)
- Another pair of power supply pins (depending of package) (VDDIO/VSSIO) for the IOs (1.65 V to 3.6 V)

The microcontrollers of the STM8TL52/L53 line manage the supply voltage needed by the ProxSense interface by connecting a 1 µF capacitor low ESR (≤1 Ω) to the PXS_V_{REG} pin (see *Figure 1*).

Figure 1. Power supply



1. The device keeps operating as long as the battery voltage is above 1.65 V and no reset is generated.

Note: The capacitors must be connected as close as possible to the device power supply pins (VDDx pins).

The decoupling capacitor must be connected as close as possible to the ground pins (VSSx pins).

2.2 Main operating voltages

The microcontrollers of the STM8TL52/L53 line embed an internal voltage regulator for generating the 1.8 V power supply for the core and peripherals, and a second internal voltage regulator providing a stable power supply (around 1.55 V) for the ProxSense peripheral.

2.3 Power-on/power-down reset (POR/PDR)

The input supply to the main and low power regulators is monitored by a power-on/power-down reset circuit. The monitoring voltage begins at 0.7 V.

During power-on, the POR/PDR keeps the device under reset until the supply voltage (V_{DD}) reach its specified working area. This internal reset is maintained for a period of ~1ms in order to wait for supply stabilization.

At power-on, a defined reset should be maintained below 0.7 V. The upper threshold for a reset release is defined in the electrical characteristics section of the product datasheets.

A hysteresis is implemented (POR > PDR) to ensure clean detection of voltage rise and fall.

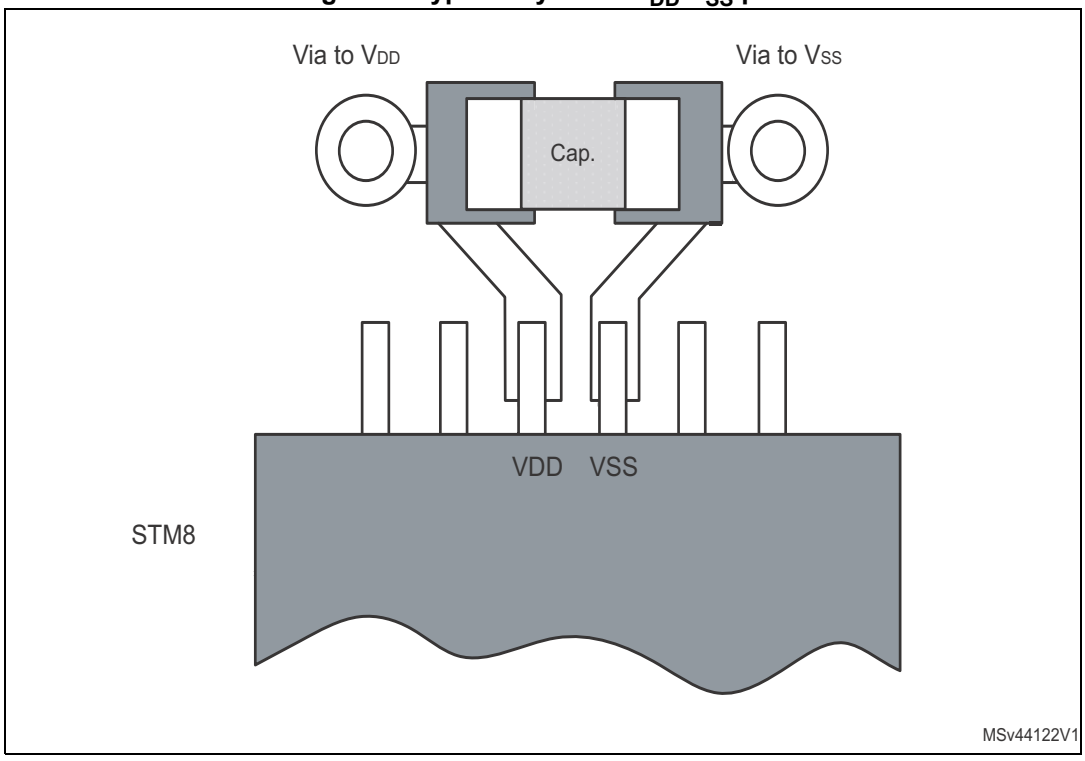
The POR/PDR also generates a reset when the supply voltage drops below the $V_{POR/PDR}$ threshold (isolated and repetitive events).

Recommendations

All VDD and VSS pins, including VDDIO and VSSIO, need to be properly connected to the power supplies. These connections, including pads, tracks and vias should have the lowest possible impedance. This is typically achieved with thick track widths and preferably dedicated power supply planes in multi-layer printed circuit boards (PCBs).

In addition, the power supply pair should be decoupled with filtering ceramic capacitors (C) at 100 nF with one chemical C (1..2 µF) in parallel on the STM8TL53xx device. The ceramic capacitors should be placed as close as possible to the appropriate pins, or below the appropriate pins, on the opposite side of the PCB. Typical values are 10 nF to 100 nF, but exact values depend on the application needs. *Figure 2* shows the typical layout of such a V_{DD}/V_{SS} pair.

Figure 2. Typical layout of V_{DD}/V_{SS} pair



3 Clock management

The microcontrollers of the STM8TL52/L53 line do not have an external clock, so no precautionary measures are needed.

Internal clocks

The STM8TL53xx devices have three kinds of internal clock:

- a high speed internal clock (HSI) running at 16 MHz
- a low speed internal clock (LSI) running at 38 kHz
- a high speed internal clock dedicated to the ProxSense (HSI_PXS) running at 16 MHz.

The HSI_PXS clock runs once the ProxSense is enabled if the LowPower bit is reset. If LowPower bit is set HSI_PXS clock runs only when an acquisition is being performed.

After reset, the CPU starts at speed of 2 MHz driven by the internal RC (HSI clock signal) divided by 8.

4 Reset control

4.1 Reset management overview

The reset pin is a 3.3 V bidirectional I/O (supplied by V_{DDIO}). After startup it can be programmed by software to be used as a general purpose output.

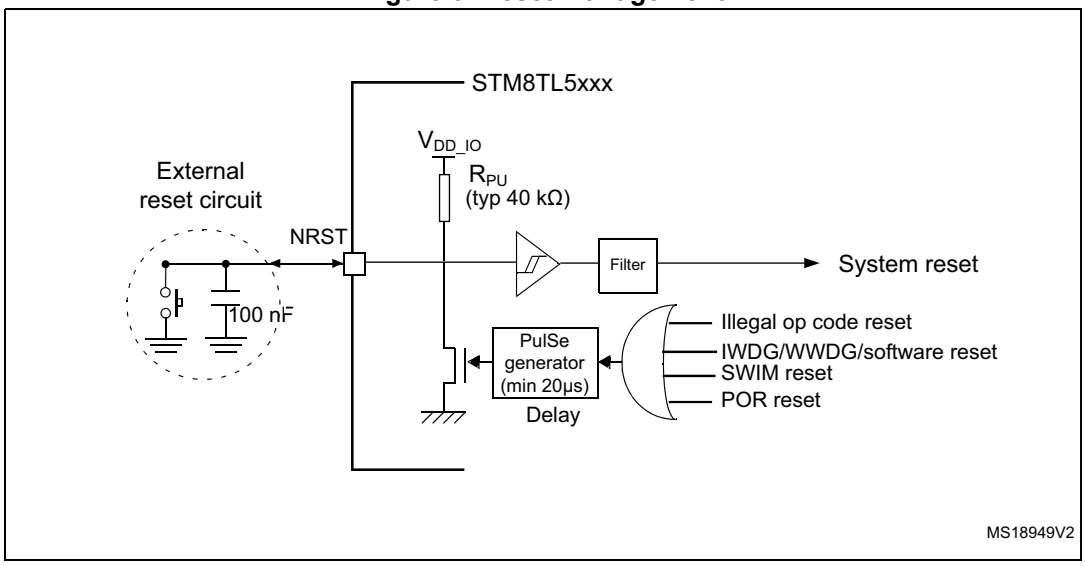
Its output buffer driving capability is fixed to Iol_{MIN} = 2 mA @ 0.45 V in the 1.65 V to 3.6 V range which includes a ~40 kΩ pull-up. Output buffer is reduced to the n-channel MOSFET (NMOS). The receiver includes a glitch filter, whereas the output buffer includes a 20 µs delay.

There are many reset sources, including:

- External reset through the NRST pin
- Power-on reset (POR): During power-on, the POR keeps the device under reset until the supply voltage (V_{DD}) reach the right voltage level.
- Independent watchdog reset (IWDG)
- Window watchdog reset (WWDG), featuring also software reset.
- SWIM reset: An external device connected to the SWIM interface can request the SWIM block to generate a microcontroller reset.
- Illegal opcode reset: If a code to be executed does not correspond to any opcode or prebyte value, a reset is generated.

Figure 3 shows a simplified functional I/O reset schematic.

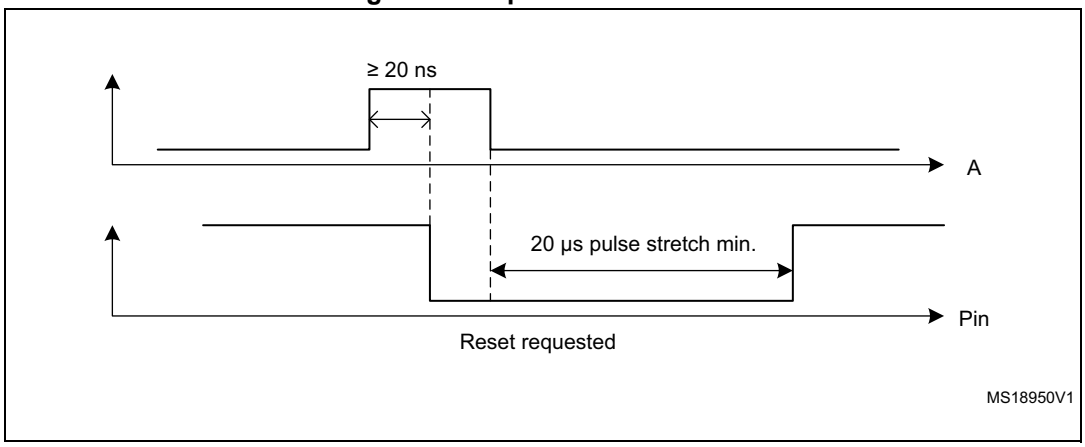
Figure 3. Reset management



4.1.1 Output characteristics

- A valid pulse on the pin is guaranteed with a ≥ 20 ns pulse duration on the internal output buffer.
- After a valid pulse is recognized, a pulse on the pin of at least 20 µs is guaranteed starting from the falling edge of A (output of the OR between the different reset sources).

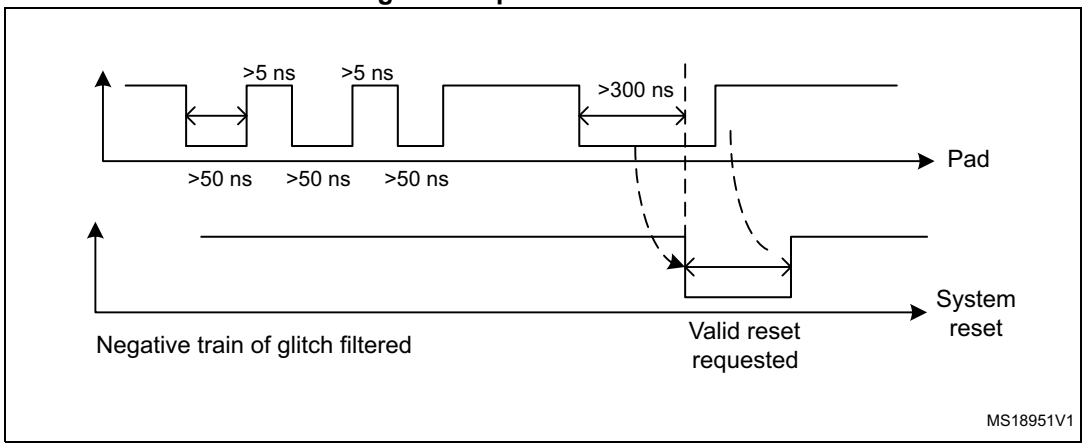
Figure 4. Output characteristics



4.1.2 Input characteristics

- All pulses with a duration less than 50 ns are filtered
- All train/burst spikes with a ratio of 1/10 must be filtered. This means that a negative spike of up to 50 ns is always filtered, when a 5 ns interval between spikes occurs (ratio 1/10).
- All pulses with duration more than 300 ns are recognized as valid pulses

Figure 5. Input characteristics



4.2 Hardware reset implementation

The microcontrollers of the STM8TL52/L53 line do not require an external reset circuit to power-up correctly. Only a pull-down capacitor is recommended (see *Figure 3*). However, charging/discharging the pull-down capacitor through an internal resistor has a negative influence on the device power consumption. Therefore, the recommended capacitor value of 100 nF can be reduced down to 10 nF to limit such power consumption.

The reset state on microcontrollers of the STM8TL52/L53 line is released 1 ms after the POR value (1.35 V to 1.65 V) is reached. At this time, V_{DD} should be in the 1.65 V to 3.6 V range.

5 ProxSense line management

5.1 ProxSense line management overview

The ProxSense interface on microcontrollers of the STM8TL52/L53 line is mainly used to perform capacitance variation acquisition.

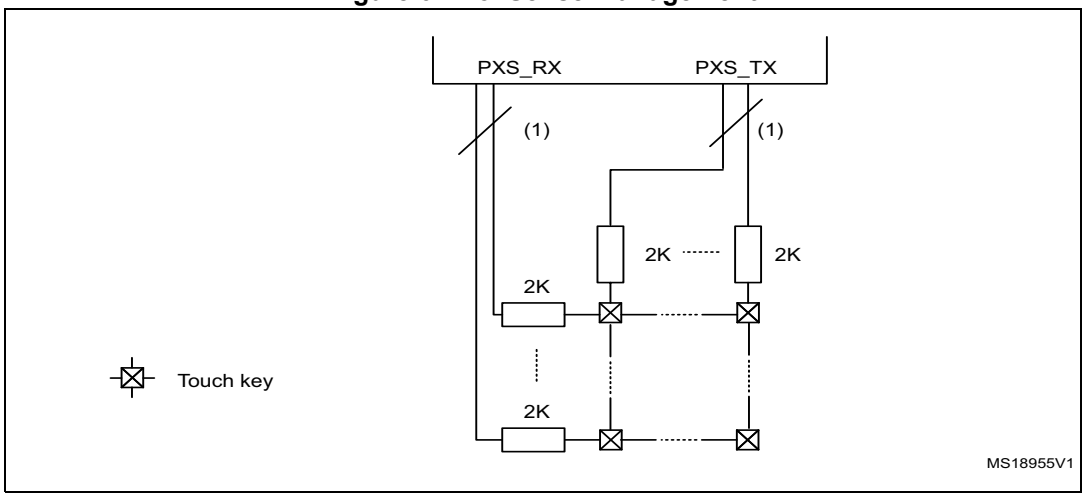
The principle of this interface is to transfer a charge from a capacitance (the electrode C_X) to another one (the sample C_S, inside the MCU) by driving the PXS_TX and activating some switches. This allows detection of proximity as well as touch by monitoring on PXS_RX.

The many features of the ProxSense interface include:

- 10 independent receiver channels, allowing 10 measurements to be performed in parallel
- Each of the 10 receiver channels can be associated with two different pins, effectively allowing an application to have up to 20 receiver channels.
- Each receiver channel can be independently configured to perform projected capacitance measurements.
- The size of each C_S capacitor can be independently configured with 5 bits of resolution.

Figure 6 shows a simplified functional schematic of the ProxSense interface.

Figure 6. ProxSense management



1. The receiver and transmit numbers are application dependent.

5.2 Hardware ProxSense implementation

The microcontrollers of the STM8TL52/L53 line do not require any external circuitry to transfer the charge correctly. However, 1KΩ serial resistors are recommended (see *Figure 6*) for ESD robustness. Several configurations are possible such as:

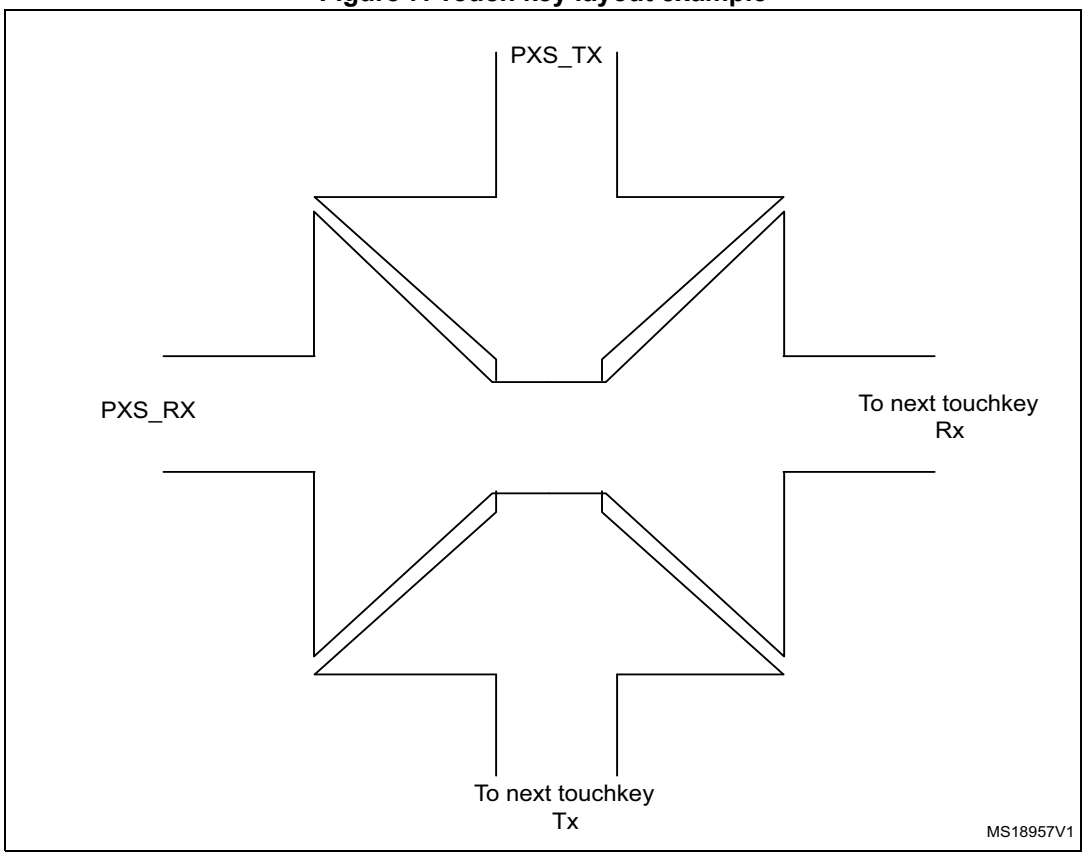
- 10 PXS_RX with 1 PXS_TX (10 measurements in parallel) or
- 10 PXS_TX with 1 PXS_RX (10 measurements in serial).

However, any combination of 20 RX and 15 TX is allowed. Up to 15 PXS_TX are allowed to be used. Furthermore the 20 RX are organized in 2 groups, all RX belonging to the same group being acquired simultaneously.

Advanced features such as an antenna (to detect noise) and external trigger may be implemented via the PXS_RFIN and PXS_TRIG pins in order to increase the robustness of the application.

The footprint shown in *Figure 7* gives an example of a touch key implementation on a PCB (for more details please refer to application note *Guidelines for designing touch sensing applications with projected sensors* (AN4313)).

Figure 7. Touch key layout example



The PXS_RX pins cannot be used as GPIOs, they are dedicated to ProxSense acquisition.

Nevertheless, any PXS_RX pin can also be configured as a transmitter:

- To ease the layout in some cases
- To get a greater number of keys in a matrix configuration for devices with a reduced number of PXS_TX pins
- To save some PXS_TX pins for general purpose functions.

This configuration as a transmitter is easily done using the firmware library by configuring the acquisition bank with a RXn instead of a TXm in the transmitter definition.

For devices with PXS_RXna and PXS_RXnb pins, such as devices with RX_GROUPA and RX_GROUPB, only the pins belonging to the same group are activated simultaneously. The disabled pins and the pins belonging to the not selected group are driven to V_{SS} or configured in high impedance according to PXS_RXINSR register.

6 Recommendations

6.1 Printed circuit board

For technical reasons, it is best to use a multi-layer PCB with a separate layer dedicated to the V_{SS} and another layer to the V_{DD} supply. This results in a good decoupling, as well as a good shielding effect. For many applications, economic requirements prohibit the use of this type of board. In this case, the most important requirement is to ensure a good structure for the V_{SS} and power supply.

6.2 Component position

A preliminary layout of the PCB must separate the different circuits according to their electromagnetic interference (EMI) contribution. This reduces cross-coupling on the PCB, for instance, noisy, high-current circuits, low voltage circuits, and digital components.

6.3 Ground and power supply (V_{SS}, V_{DD}, V_{SSIO}, V_{DDIO})

The V_{SS} should be distributed individually to every block (noisy, low level sensitive, and digital) with a single point for gathering all ground returns. Loops must be avoided or have a minimum surface. The power supply should be implemented close to the ground line to minimize the surface of the supply loop. This is due to the fact that the supply loop acts as an antenna, and is therefore the main emitter and receiver of EMI. All component-free surfaces of the PCB must be filled with additional grounding to create a kind of shield (especially when using single-layer PCBs).

6.4 Decoupling

The standard decoupler for the external power is a 1 µF capacitor**.** Supplementary 100 nF capacitors must be placed as close as possible to the VSS/VDD and VSSIO/VDDIO pins of the microcontroller to reduce the area of the current loop.

As a general rule, decoupling all sensitive or noisy signals improves electromagnetic compatibility (EMC) performances.

There are two types of decouplers:

- Capacitors close to components. Inductive characteristics, which apply to all capacitors beyond a certain frequency, must be taken into account. If possible, parallel capacitors with decreasing values (0.1, 0.01,... µF) should be used.
- Inductors. Ferrite beads for example, are excellent inductors with good dissipation of EMI energy and no loss of DC voltage (which is not the case when simple resistors are used).

6.5 Other signals

When designing an application, the following areas should be closely studied to improve EMC performances:

- Noisy signals (clock)
- Sensitive signals (high impedance)
- Signals for which a temporary disturbance affects operation of the application permanently, for example, interrupts and handshaking strobe signals (but not LED commands).

A surrounding V_{SS} trace for such signals increases EMC performances, as does a shorter length or absence of noisy and sensitive traces (crosstalk effect).

For digital signals, the best possible electrical margin must be reached for the two logical states. Slow Schmitt triggers are recommended for eliminating parasitic states.

6.6 Unused I/Os and features

Microcontrollers are designed for a variety of applications, where often a particular application does not use 100 % of the microcontroller resources.

To avoid unnecessary power consumption (especially important for battery powered applications) and also to improve EMC performance, unused clocks, counters, or I/Os, should not be left free. The I/Os should be forced externally (pull-up or pull-down to the unused I/O pins), and unused functions should be ‘frozen’ or disabled.

Alternatively, unused I/Os can be programmed as push-pull ‘low’ to keep them at a defined level without using external components. However, in this case the I/O is not driven during the power up phase, until the I/O is configured. This can add a little extra power consumption, and may be undesirable in very power sensitive applications.

The unused PXS_RX pins should follow the same rule and can be driven to V_{SS} by resetting the PXS_RXINSR register.

20-pin package

A special care of I/O configuration must be taken with STM8TL5xFx devices. The port A, B and D I/Os are not all configured by default as it is the case on 28-pin packages.

The user code must configure PA6, PA7, PD2, PD3, PD7 as output push pull low level. It is recommended to program all the unused I/Os to push-pull 'low level' even the ones not present in the device. While accessing to the GPIO registers, it is strongly recommended to mask the unused bit in order not to change their configuration or state.

Concerning the PXS_RX, PXS_RXENRH and PXS_RXINSRH (and more generally all the MSByte of the register related to PxS_RX and PXS_TX) must be kept clear.

7 Reference design

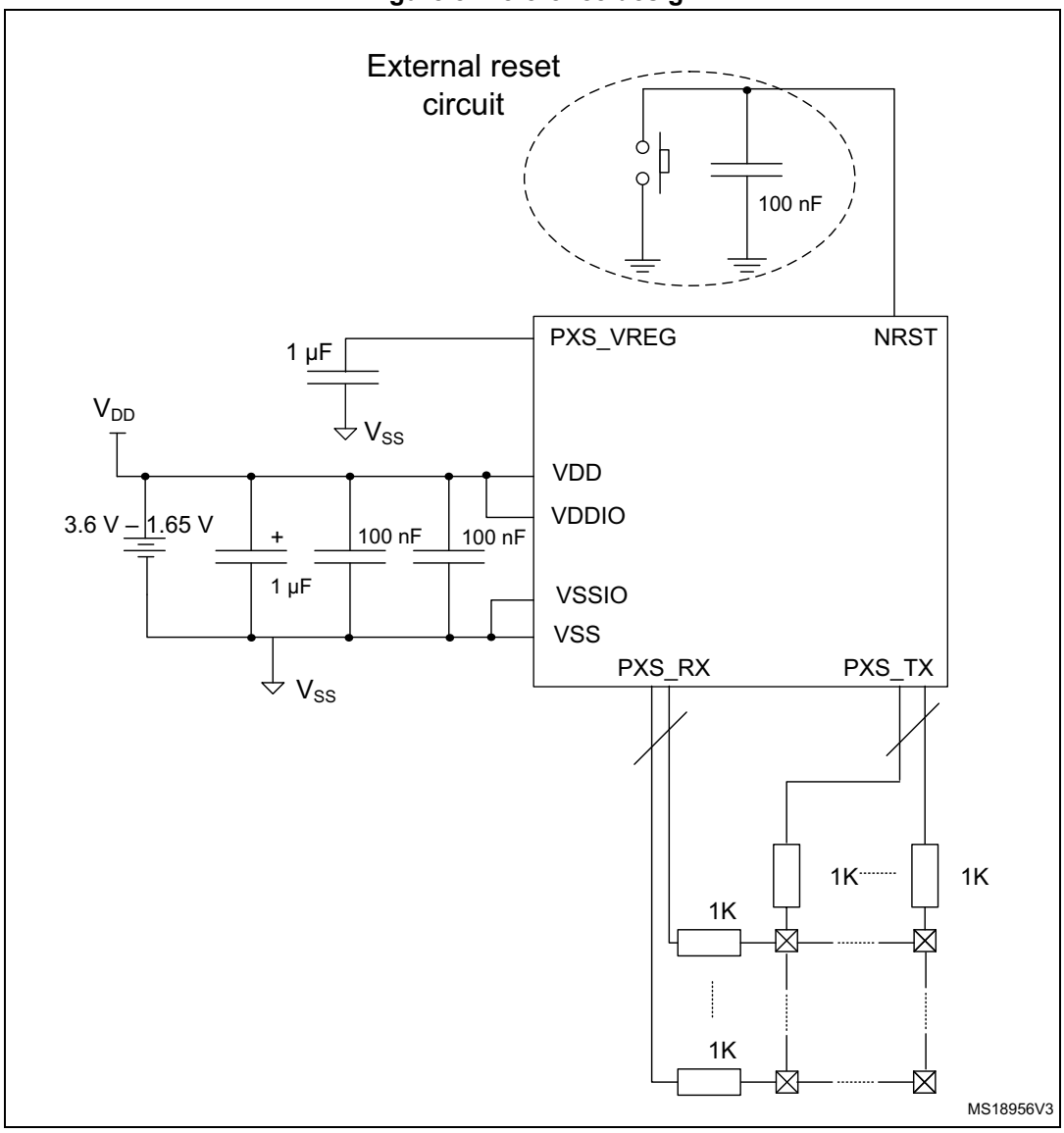
7.1 Component references

Table 1. Component list

ID	Component name	Reference	Quantity	Comments
1	Microcontroller	STM8TL53xx	1	Refer to the ‘pinouts and pin description’ and ‘package characteristics’ sections of the STM8TL53xx datasheets, to choose the right package
2	Battery	1.65 V to 3.6 V	1	-
3	Capacitor	1 µF	1	Decoupling capacitor
4	Capacitor	1 µF Low ESR (≤1 Ω)	1	Decoupling capacitor for the PXS regulator
5	Capacitor	100 nF	2	Ceramic capacitor (decoupling capacitor)
Components below are optional				
6	SWIM connector	4 pins	1	-
7	Resistor	1K	n	Serial resistors for PXS_TX and PXS_RX pins

7.2 Schematics

Figure 8. Reference design



1. For best performance, select a low ESR (≤ 1 Ω) capacitance.

8 STM8TL5x firmware libraries

In order to ease the development start-up, two firmware libraries are provided:

- The STM8TL5x standard peripheral library
- The STM8TL5x STMTouch library

8.1 STM8TL5x standard peripheral library

This STM8TL5x firmware library contains the standard peripheral drivers (timers, I2C, SPI, USART, watchdogs, etc...) and a complete set of source code examples for each STM8TL5x peripheral. It is written in strict ANSI-C and it is fully MISRA C 2004 compliant.

All examples can be used with four workspace and project definition files:

- One for the STVD (ST visual develop) and Cosmic C compiler
- One for the STVD and Raisonance compiler
- One for the Raisonance integrated debugging environment and compiler (RIDE7 IDE)
- One for the IAR embedded workbench for STM8 (EWSTM8).

The user is able to load and compile them easily into their preferred development environment.

8.2 STM8TL5x STMTouch library

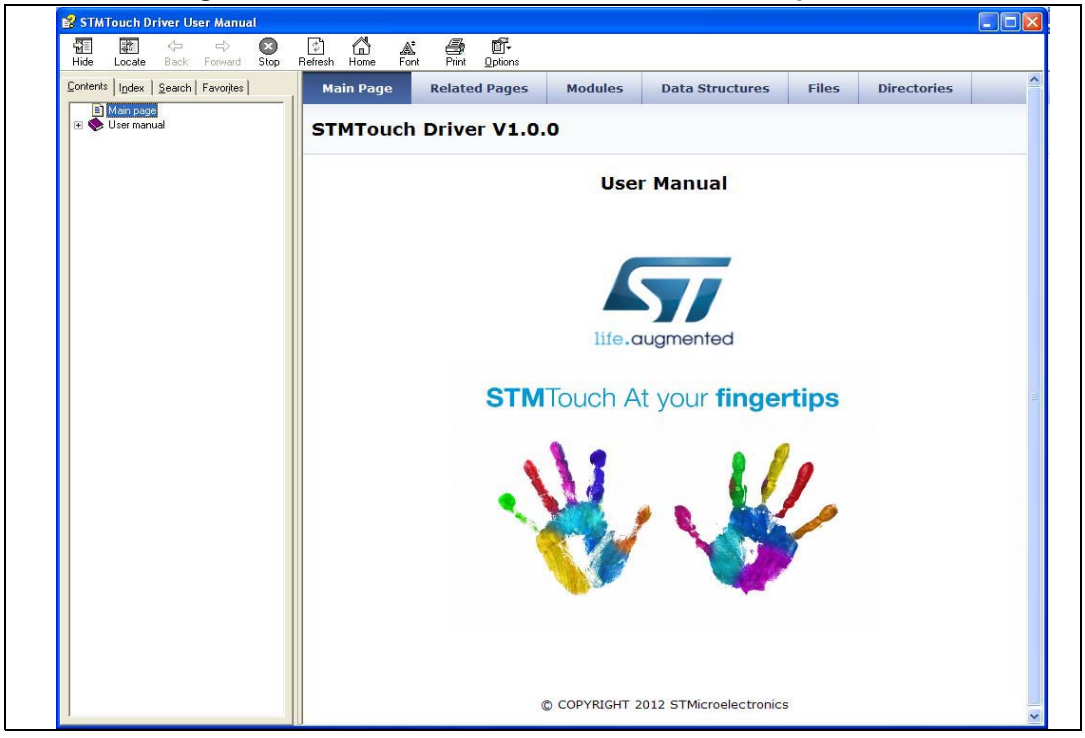
The STM8TL5x STMTouch library is dedicated to the management of the ProxSense (PXS) peripheral. It follows the same coding rules as the standard peripheral library. The STM8TL5x STMTouch library allows the user to enable touch sensing capabilities on STM8TL5x devices. This simple firmware offers a complete and robust solution to manage capacitive sensing keys, wheels or sliders.

8.3 Online help

For each firmware driver, an online help is directly available from the firmware installation directory (see *Figure 9* and *Figure 10*).

Figure 9. STM8TL5x standard peripheral driver online help manual

Figure 10. STM8TL5x STMTouch driver online help manual



The STM8TL5x STMTouch library is provided with a set of examples all together with their online help manual, available from the examples directory (see *Figure 11*).

Figure 11. STM8TL5x STMTouch examples online help manual

9 STM8 development tools

Typically, the following tools are needed to get started:

- STVD, IAR workbench or RIDE for integrated development environment
- STM8 C compiler (from Cosmic, Raisonance, or IAR)
- ST toolset from STMicroelectronics
- Firmware libraries from STMicroelectronics (STM8TL5x_StdPeriph_Lib and STM8TL5x_STMTouch_Lib for STM8TL5x)
- The user may need the hardware debug interface ST-Link from STMicroelectronics or "Rlink" from Raisonance.
- STMStudio-STM8 is a graphical user interface that allows sampling and viewing user variables in real time using any hardware debugging tool while the application is running.

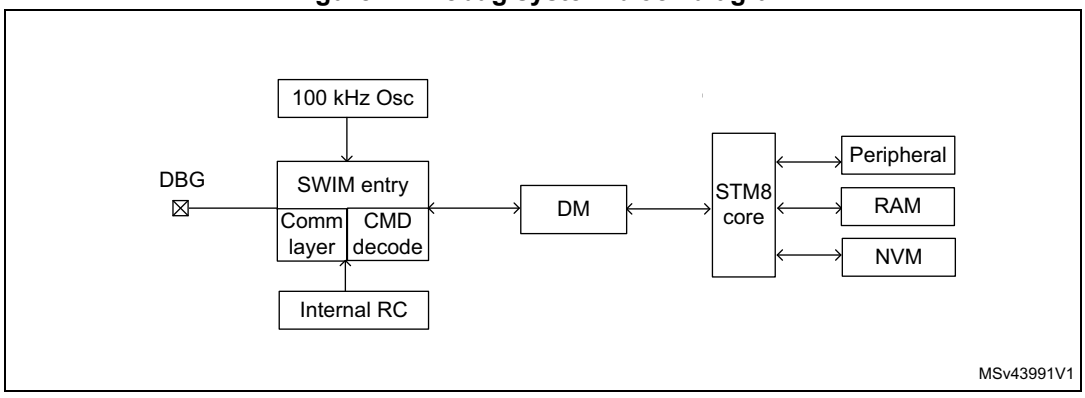
9.1 Single wire interface module (SWIM)

9.1.1 SWIM overview

In-circuit debugging mode or in-circuit programming mode are managed through a single wire hardware interface based on an open-drain line, featuring ultra fast memory programming. Coupled with an in-circuit debugging module, the SWIM also offers a non-intrusive read/write to RAM and peripherals. This makes the in-circuit debugger extremely powerful and close in performance to a full-featured emulator.

The SWIM pin can be used as a standard I/O (with 8 mA capability) which has some restrictions if the user wants to use it for debugging. The most secure way to use it is to provide a strap option on the PCB. Please refer to the STM8 SWIM communication protocol and debug module user manual (UM0470) for more SWIM protocol details.

Figure 12. Debug system block diagram



9.1.2 SWIM connector pins

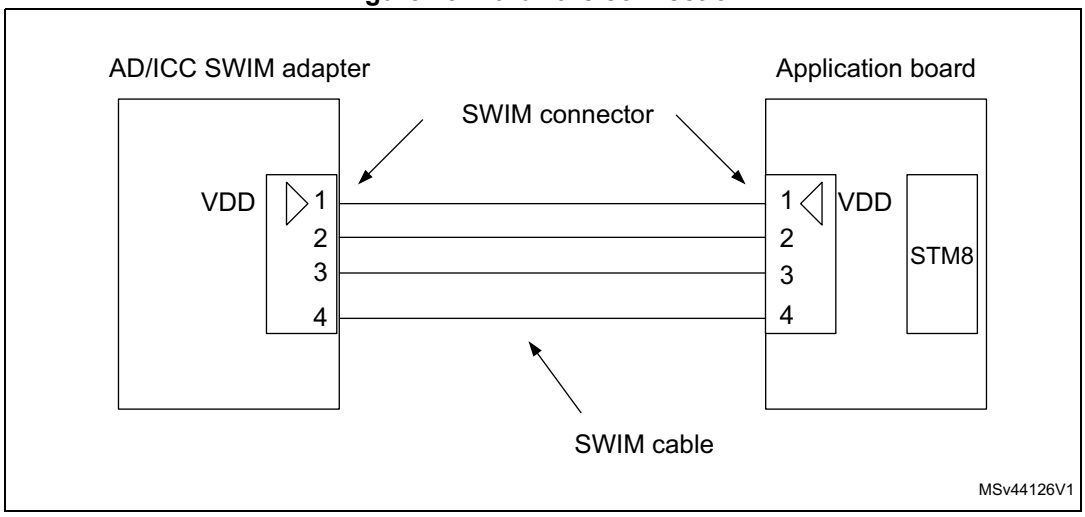
The SWIM connector pins consist of four pins as described in *Table 2*.

Table 2. SWIM connector pins

Pin number	Pin name
Pin 1	VDD
Pin 2	SWIM
Pin 3	VSS
Pin 4	Reset

9.1.3 Hardware connection

Figure 13. Hardware connection



Caution: It is recommended to place the SWIM header as close as possible to the STM8TL53xx device, minimizing any possible signal degradation caused by long PCB tracks.

9.2 RLink and STLink

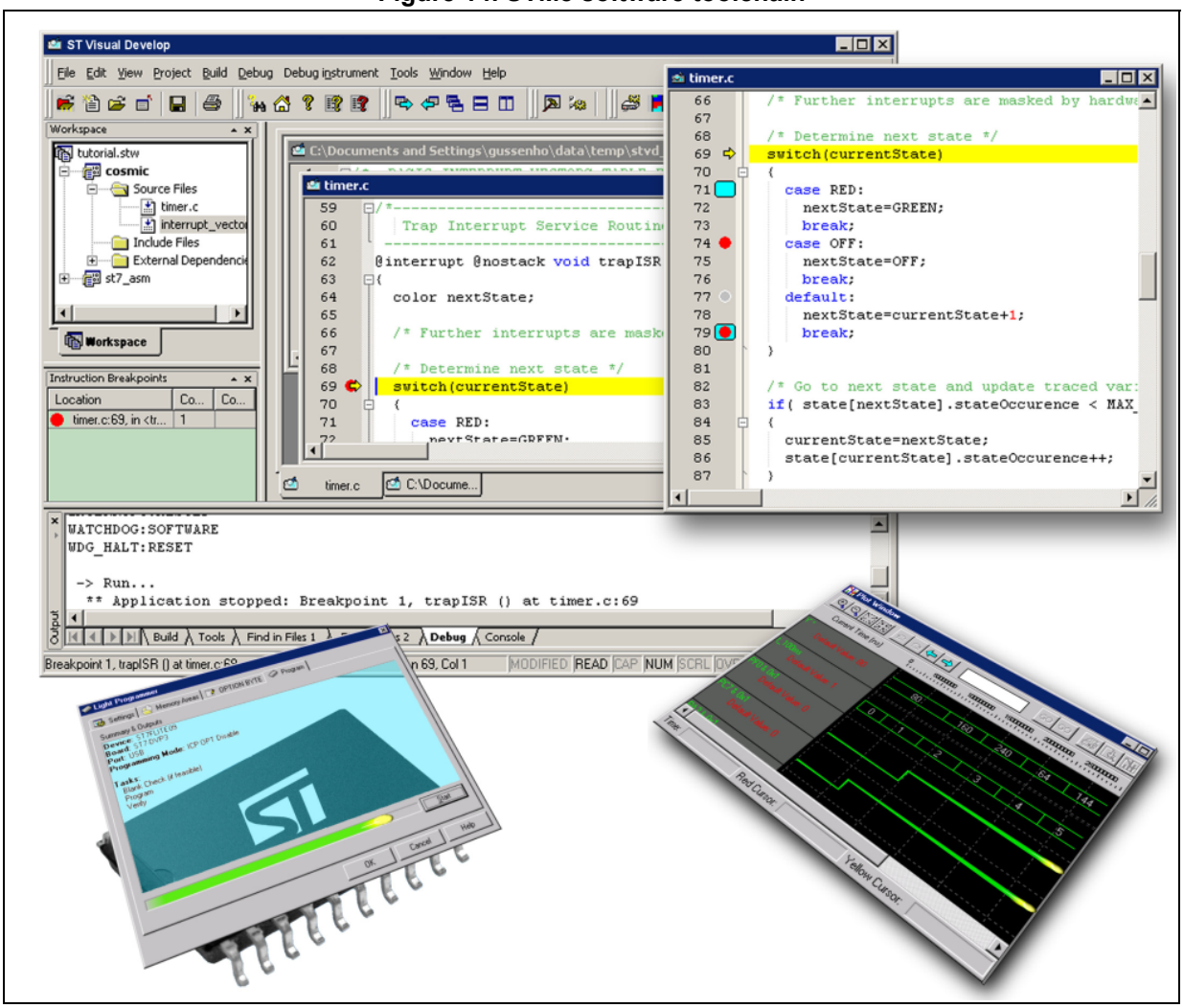
RLink and STLink are debug tools that allow any user application board with the SWIM interface to be connected to a host PC via USB for debugging and programming. See *Section 11.3.3: Connecting the hardware on page 30*.

10 STM8 software toolchain

To write, compile and run the first software on an STM8TL53xx device, the following components of the software toolchain are required (see *Figure 14*):

- Integrated development environment
- Compiler
- Firmware library (optional, used to ease the startup)

Figure 14. STM8 software toolchain



10.1 Integrated development environment

The integrated development environment ST visual develop (STVD) provides an easy-to-use, efficient environment for start-to-finish control of application development. It provides an environment from building and debugging the application code to programming the microcontroller. The STVD is delivered as part of the free ST toolset, which also includes the ST visual programmer (STVP) programming interface and the ST assembler linker.

To build applications, the STVD provides seamless integration of C and assembly tool chains for ST including the Cosmic and Raisonance C compilers and the ST assembler linker. When debugging, STVD provides an integrated simulator (software) and supports a complete range of hardware tools including the low-cost RLink in-circuit debugger/programmer and the high-end STice emulator.

To program applications to an STM8TL53xx, the STVD also provides an interface for reading from the microcontroller memories, writing to them and verifying them. This interface is based on the ST visual programmer (STVP), and supports all the target devices and programming tools supported by STVP.

The free ST toolset for STM8 is available from STMicroelectronics home page (see *www.st.com*).

10.2 Compiler

The STM8TL53xx devices can be programmed by a free assembler toolchain which is included in the ST toolset.

As the core is designed for optimized high-level-language support, use of a C compiler is recommended.

C compilers for STM8 are offered by the third party companies Cosmic, Raisonance, and IAR.

A free version of the C compiler with up to 32 Kbytes of generated code is available at: www.cosmic-software.com and www.raisonance.com.

11 Setting up the STM8 development environment

11.1 Installing the tools

All software tools are delivered with a setup wizard which guides the user through the installation process. It is recommended to install the tools in the following order:

1. C compiler
2. ST toolset
3. STM8TL5x firmware libraries

ST-LINK does not need any dedicated software installation in the STM8 development environment because the necessary drivers are delivered with the ST toolset.

The R-link drivers must be launched separately as follows:
Start/Programs/STtoolset/Setup/Install Rlink driver.

11.2 Using the tools

Once the tools installation is complete, the ST visual develop (STVD) integrated development environment can be launched.

The user then has the choice to generate either a new workspace with a new project or to open an existing workspace. If using the STVD for the first time, it is recommended to open an existing project from one of the STM8TL52/L53 firmware libraries.

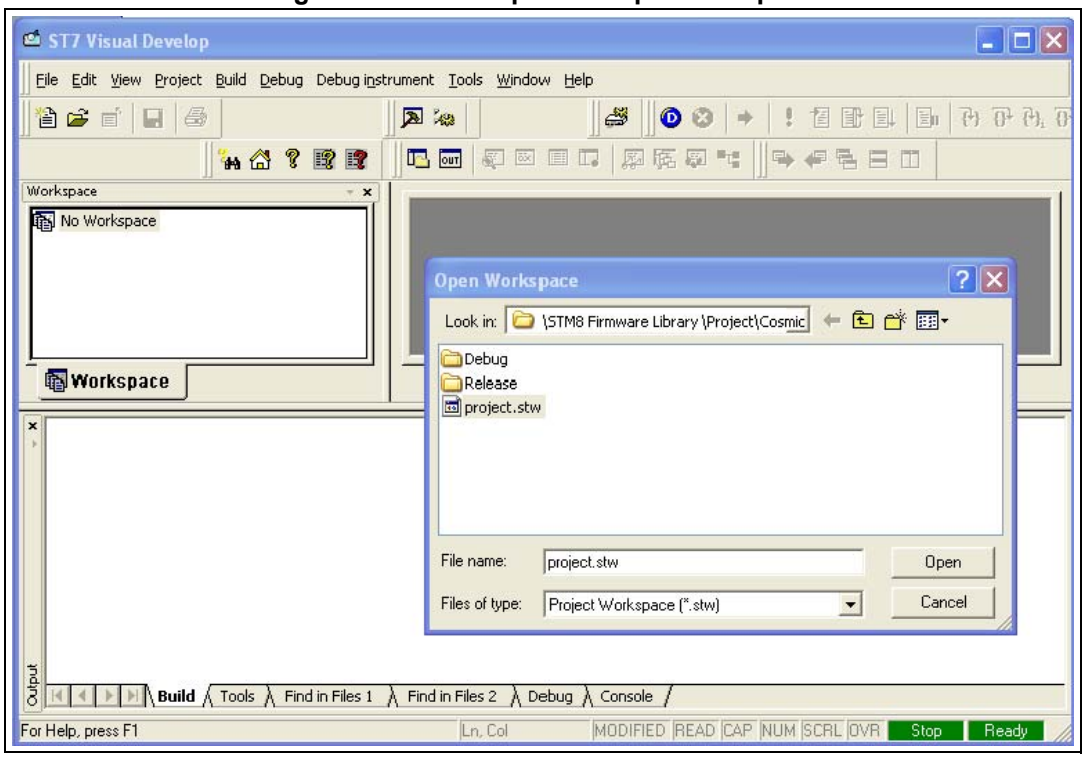
Note: Even if the user is not intending to use the library, an existing library project can be used as a template to configure all the compiler options. The user may enter their own code after main().

The STM8TL5x standard peripheral firmware library includes several examples for each peripheral plus one workspace containing a blank project which is ready to receive the user's C code. It is located in the firmware subdirectory \Project\Examples (see *Figure 15*). The user can choose between STVD\Cosmic, STVD\Raisonance, RIDE, or EWSTM8.

The STM8TL52/L53 STMTouch library provides all functions required for an easy and quick development of an own touch sensing application, using the full set of STM8TL52/L53 touch sensing features.

The STM8TL5x STMTouch library (STM8TL-TOUCH-LIB) contains the touch sensing drivers dedicated to the STM8TL52/L53 and their ProxSense peripheral, and also a set of examples showing the STM8TL52/L53 performances.

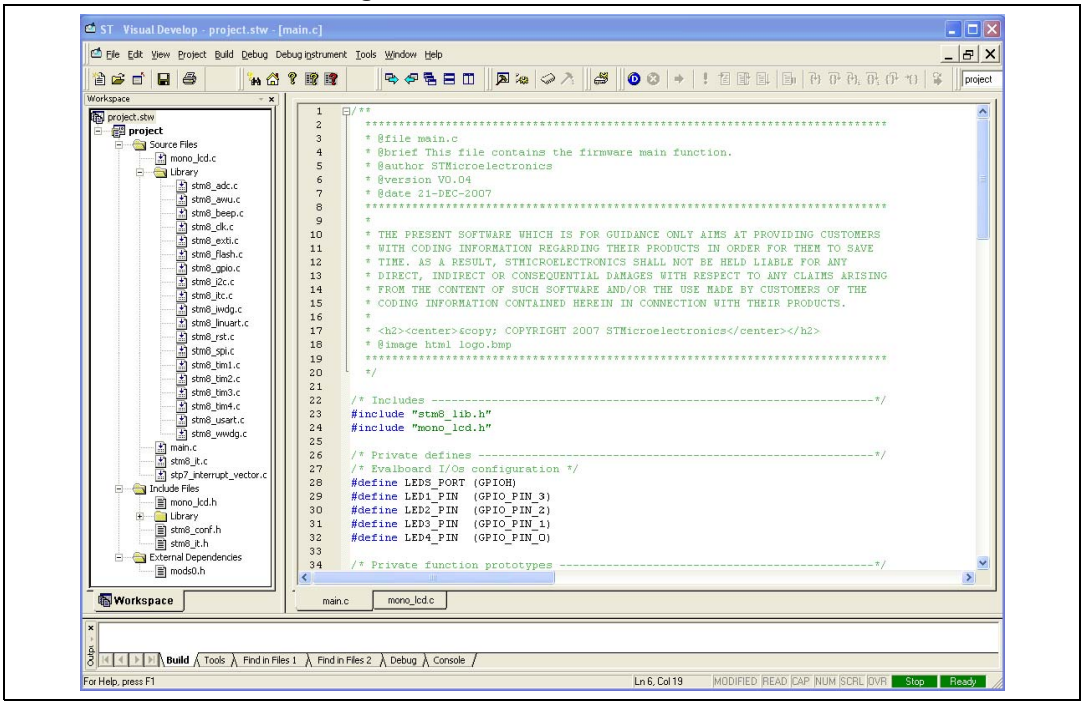
Figure 15. STVD: Open example workspace



Project editing

All project source files are visible and can be edited (see *Figure 16*).

Figure 16. STVD: MCU edit mode



11.3 Running the demonstration software

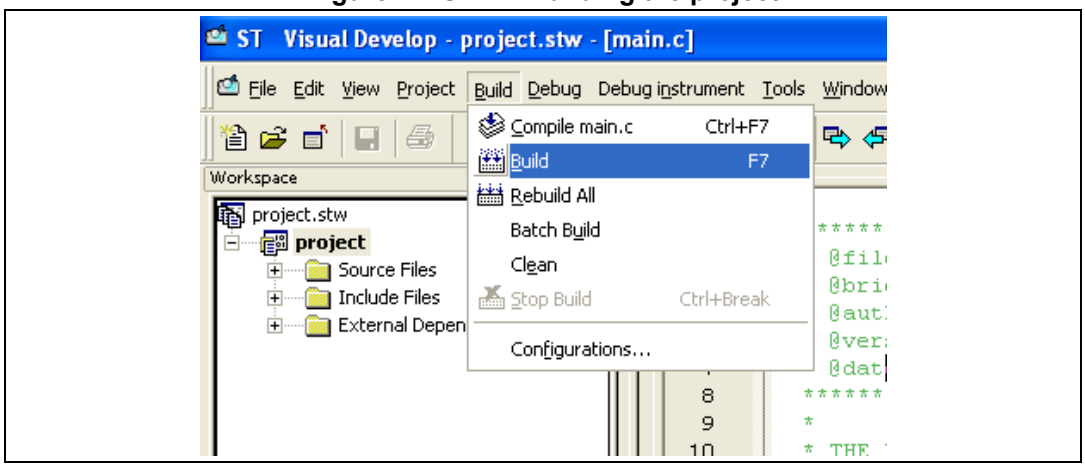
Open the desired project workspace within the chosen demonstration firmware package.

To run the demonstration software, the project has to be compiled and the correct HW tool must be selected before starting the debug session.

11.3.1 Compiling the project

The project can be compiled using the ‘Build’ function in the ‘Build’ menu (see *Figure 17*).

Figure 17. STVD: Building the project



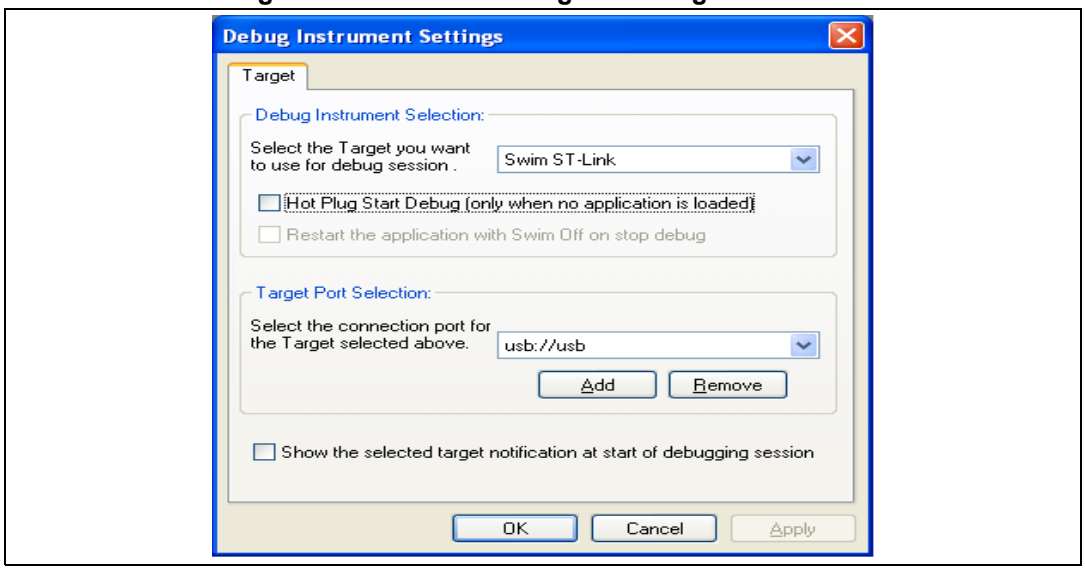
11.3.2 Selecting the correct debug instrument

STM8TL53xx Touch keypad evaluation board in standalone

In the example below, the ST-Link tool is used for communicating via the SWIM interface with the on-board debug module of the STM8.

The ST-Link tool can be selected from the ‘Debug Instrument Selection’ list in the ‘Debug Instrument Settings’ dialog (see *Figure 18*).

Figure 18. STVD: Selecting the debug instrument



Note: The Rlink can also be used for communicating via the SWIM interface.

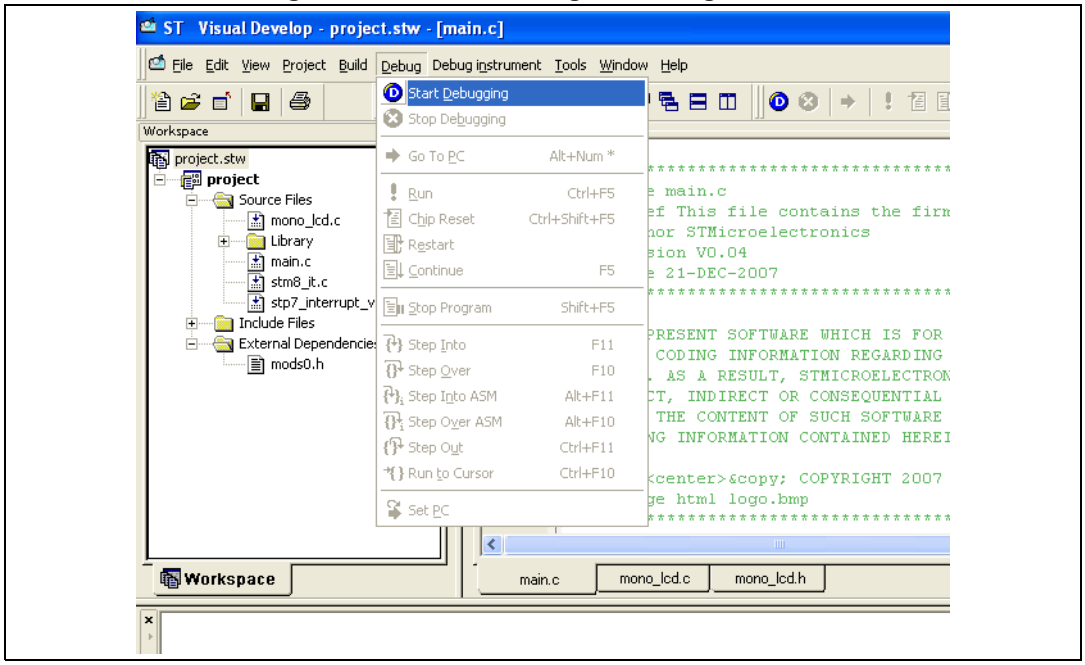
11.3.3 Connecting the hardware

The debug tool, ST-Link or Rlink, must be connected to the USB connector of the PC. The SWIM connector for STM8 of the ST-Link or Rlink must be connected to the 4-pin SWIM connector of the board. This board must also be powered through its USB connector.

11.3.4 Starting the debug session

Debug mode can be entered by the command ‘Debug Start Debugging’ (see *Figure 19*).

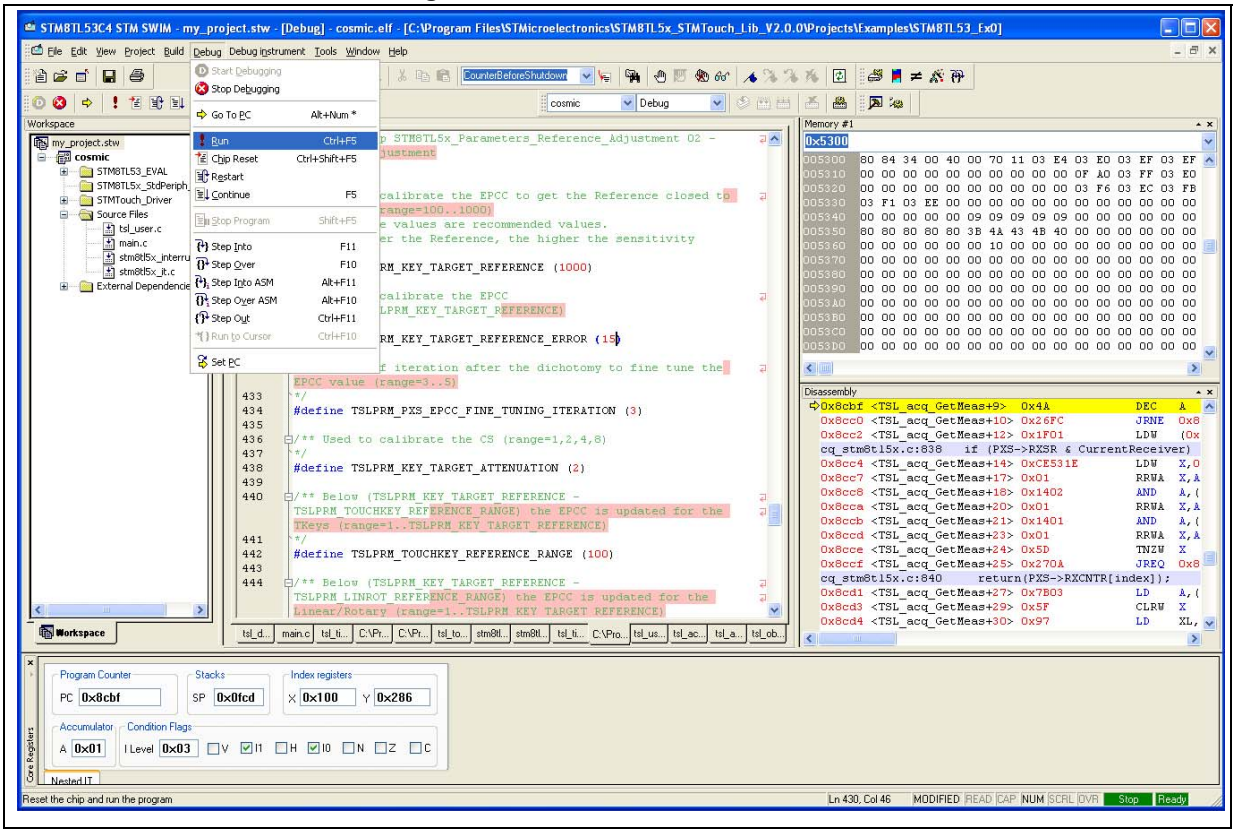
Figure 19. STVD: Starting the debug session



11.3.5 Running the software

After entering debug mode, the software can be started by the run command in the menu ‘Debug Run’ (see *Figure 20*).

Figure 20. STVD: Run the software



11.3.6 Follow up

Step by step, additional peripherals of STM8TL53xx devices can be run, following on from the initial debug session described above.

The necessary software drivers, including STM8TL52/L53 peripheral drivers (USART, I2C, SPI) and driver for the touch sensing modules (buttons, wheel, slider), are delivered in the STM8TL52/L53 firmware standard peripheral library and the STM8TL5x STMTouch library.

12 Documentation and online support

Documentation resources related to tool usage includes:

Applications

- STM8TL52x4 STM8TL53x4 datasheet.
- How to program Flash memory and data EEPROM on STM8TL5xxx microcontrollers (PM0212).
- STM8TL5xxx reference manual (RM0312)
- STM8 CPU programming manual (PM0044)
- Guidelines for designing touch sensing applications with projected sensors (AN4313)

Tools

- STM8TL5xxx firmware standard peripheral library and release note (detailed descriptions of the library are included as help files).
- STM8TL5x STMTouch library and release note (detailed descriptions of the library are included as help files).
- Cosmic, Raisonnance, or IAR C compiler user manual
- STVD tutorial (included as help files in the ST-toolchain)
- STVD user manual (UM0036)
- STM8 SWIM communication protocol and debug module user manual (UM0470)

The microcontroller discussion forum on *www.st.com* is available for developers to exchange ideas or find different application ideas. In addition, the website has a knowledge base of FAQs for microcontrollers, which provide answers to many queries and solutions to many problems.

13 Revision history

Table 3. Document revision history

Date	Revision	Changes
10-Mar-2011	1	Initial release
06-Dec-2011	2	STM8TL53xx product name update Changed references to associated documents Updated *Section 8: STM8TL5x firmware libraries* Updated *Section 11.2: Using the tools*.
05-Sep-2012	3	STM8TL5xxx product name update Updated *Figure 6: ProxSense management* Added references to associated documents Updated *Section 5.2: Hardware ProxSense implementation* Added *Section 6.6.1: 20-pin package on page 17* Updated *Table 1: Component list* Updated *Figure 8: Reference design* Updated *Figure 9: STM8TL5x standard peripheral driver online help manual* Updated *Figure 10: STM8TL5x STMTouch driver online help manual* Added *Figure 11: STM8TL5x STMTouch Examples online help manual* Updated *Section 11.3.2: Selecting the correct debug instrument* Replaced *Section 11.3.3: Connecting the hardware*
05-Apr-2017	4	Updated: – Title – *Introduction* – AN2869 replaced by AN4313 Removed: – all information on the STMT-BOX,STM8TL53 Touch keypad Evaluation board – Table1. Applicable products