AN3139
Application note

Migration and compatibility guidelines for STM8L microcontroller applications

Introduction

For designers of STM8L microcontroller applications, it is important to be able to replace easily one microcontroller type by another one in the same product family. Migrating an application to a different microcontroller is often needed when product requirements grow, putting extra demands on the memory size and on the number of I/Os.

However, to achieve cost reduction objectives, the user may need to switch to smaller components and to shrink the PCB area. This application note aims at analyzing the steps required to migrate from an existing STM8L-based design to any one of the other microcontroller types in the fast-growing STM8L family.

This application note groups all the most important information and provides a list of the fundamental aspects.

The information included in this document can also be extremely useful in a first STM8 design. Studying the issues in this phase can allow the user to adapt from the beginning his design to any future requirement.

To benefit fully from the information in this application note, the user should be familiar with the STM8L microcontroller family. The STM8L family reference manuals (RM0013 and RM0031), the STM8L datasheets, and the STM8L Flash program memory / data EEPROM programming manual (PM0054) are available from www.st.com.

This application note is divided into four main sections:

- **Section 1: STM8L family compatibility**: This section presents a first-level view of the different aspects of the STM8L family architecture that must be taken into account for a new design or migration. The microcontroller blocks and peripherals are grouped and identified either as “compatible” or “compatible with minor limitations”.
- **Section 2: Planning for migration:** This section gives an overview of common migration cases. It provides a checklist of items which are potentially impacted by each case to allow the user to quickly analyze which subjects have to be anticipated.
- **Section 3**: **Block-by-block compatibility**: This section focuses on the migration between different packages and details the pin-to-pin compatibility between all STM8L sub-families.
- **Section 4: Peripheral pinout through all STM8L families**. This section shows the differences in the pinout for each peripheral.

Contents

List of tables

List of figures

1 STM8L family compatibility

1.1 Family concept

The STM8L family is one of a growing number of different STM8 microcontroller families.

All these STM8 microcontroller families are based on a common robust and low-cost 8-bit high performance core with a rich set of enhanced peripherals. This ensures a high level of compatibility within the STM8L 'world', especially in terms of software development, compilers, debugging environment, programming tools and driver libraries.

The STM8L product family offers a wide choice of memory sizes and package types to fit different application requirements as closely as possible. Consequently, when there are new requirements on the application side, it can make sense to switch to another STM8L type with different memory capacity or package size.

The STM8L family includes a product line divided into two main sub-families:

- **STM8L15x medium density** sub-family where the Flash memory density ranges between 16 and 32 Kbytes.
- **STM8L10x low density** sub-family where the Flash memory density ranges between 4 and 8 Kbytes. The STM8L10x MCUs are ideal for cost-sensitive applications with low code density.

Both sub-families provide a complete set of essential peripherals. STM8L10x devices target applications requiring reduced cost, lower memory capacity, fewer GPIOs and less advanced features.

The wide range of available pin-counts and package sizes is discussed in *Chapter 3.1: Package pinout*.

All STM8L family microcontrollers use the same application development tools:

- Embedded single wire interface module (SWIM)
- Software integrated development environment (IDE) tools including assembler, simulator, debugger, programmer:
  - ST Visual Develop (ST)
  - Ride (Raisonance)
- In-circuit debugging and programming tools
  - STIce from ST (full hardware emulator)
  - ST-Link from ST
  - RLink from Raisonance (low cost debug/programming tool)
- Starter kits and evaluation boards
- C compiler and assembler tool chains (Cosmic, Raisonance)
- Firmware libraries (peripheral control examples, MISRA or class B compliance, touch sensing)
- Application notes

By using a common development environment, you significantly reduce code maintenance effort and shorten the time-to-market, especially in cases when an application has to be migrated from one STM8 microcontroller to another.

By using the drivers provided in the STM8L firmware library to interface with the hardware, it becomes reasonably straightforward to move the application firmware from one STM8L product to another. The principle job is analyzing the details on the hardware side, taking care of the placement and availability of the peripheral I/O functions in the pinout. More details can be obtained in the STM8L datasheet and further in this document in *Section 3.1: Package pinout*.

Figure 1: STM8L family block diagram gives an overview of the STM8L blocks and their compatibility level, as discussed in the next sections.

Table 1. Overview of STM8L family peripherals

Peripheral	STM8L10x	STM8L15x
RAM	Up to 1.5 Kbytes	Up to 2 Kbytes
Flash program memory	Up to 8 Kbytes	From 16 to 32 Kbytes
Data EEPROM	Up to 2 Kbytes in Flash program memory Size configurable by option byte	Up to 1 Kbyte in separate memory array; Fixed size
Interrupt	Up to 26 peripheral interrupt vectors	Up to 32 peripheral interrupt vectors
CLK	Yes	Yes
AWU	Yes	Not available
RTC	Not available	Yes
Beep	Yes	Yes
IWDG	Yes	Yes
WWDG	Not available	Yes
COMP	Yes	Yes
RI & SYSCFG	Not available	Yes
GPIO	Up to 30 I/Os (GPIOA..D)	Up to 41 I/Os (GPIOA..F)
EXTI	Up to 29 external interrupt lines	Up to 40 external interrupt lines
DMA	Not available	DMA1 with 4 channels
ADC	Not available	ADC1
DAC	Not available	Yes
TIM	Basic TIM4 General-purpose TIM2/3	Basic TIM4 General-purpose TIM2/3 Advanced-control TIM1
Infrared interface IRTIM	Yes	Yes
I2C	I2C	I2C1
SPI	SPI	SPI1
USART	USART	USART1
LCD	Not available	Yes

1.2 Fully compatible blocks

The STM8L family embeds a set of system blocks which are by definition common to all products. Those blocks are identical, so they have the same structure, registers and control bits. There is no need to perform any software change to keep the same functionality at the application level after migration. When external components are needed (e.g. Vcap capacitor) no change is required from one product to another. All the features and behaviors remain the same. These blocks are shown in *Figure 1: STM8L family block diagram*.

Fully compatible parts and peripherals are:

- STM8 core
- Debug / SWIM module
- Power-on reset (POR)
- Voltage regulator
- Low speed internal RC (LSI)
- High speed internal RC (HSI)
- Independent watchdog
- Timers (TIM2, TIM3 and TIM4)
- IR (infrared interface)

1.3 Blocks that are compatible with minor exceptions

Some of the peripherals or functional blocks can have differences in their electrical parameters, structure, registers, control bits or other minor aspects but not in their main functionality.

Note: The RTC, ADC, DAC, DMA, WWDG, LCD and BootROM peripherals are not available in STM8L10x devices.
The AWU peripheral is not available in STM8L15x devices and is replaced by the RTC, so this aspect can also be considered as an incompatibility.

The following functional blocks can be considered as compatible with only a few negligible differences:

- GPIO (I/O capabilities)
- Interrupt management (interrupt vectors)
- Power control (wakeup from low power mode)
- I2C (true open drain)
- SPI
- USART
- Internal memories (Flash, SRAM, EEPROM)

You can find more details about these blocks in *Chapter 3: Block-by-block compatibility analysis*. You can also refer to *Figure 1: STM8L family block diagram*.

1.4 Blocks that are compatible with significant exceptions

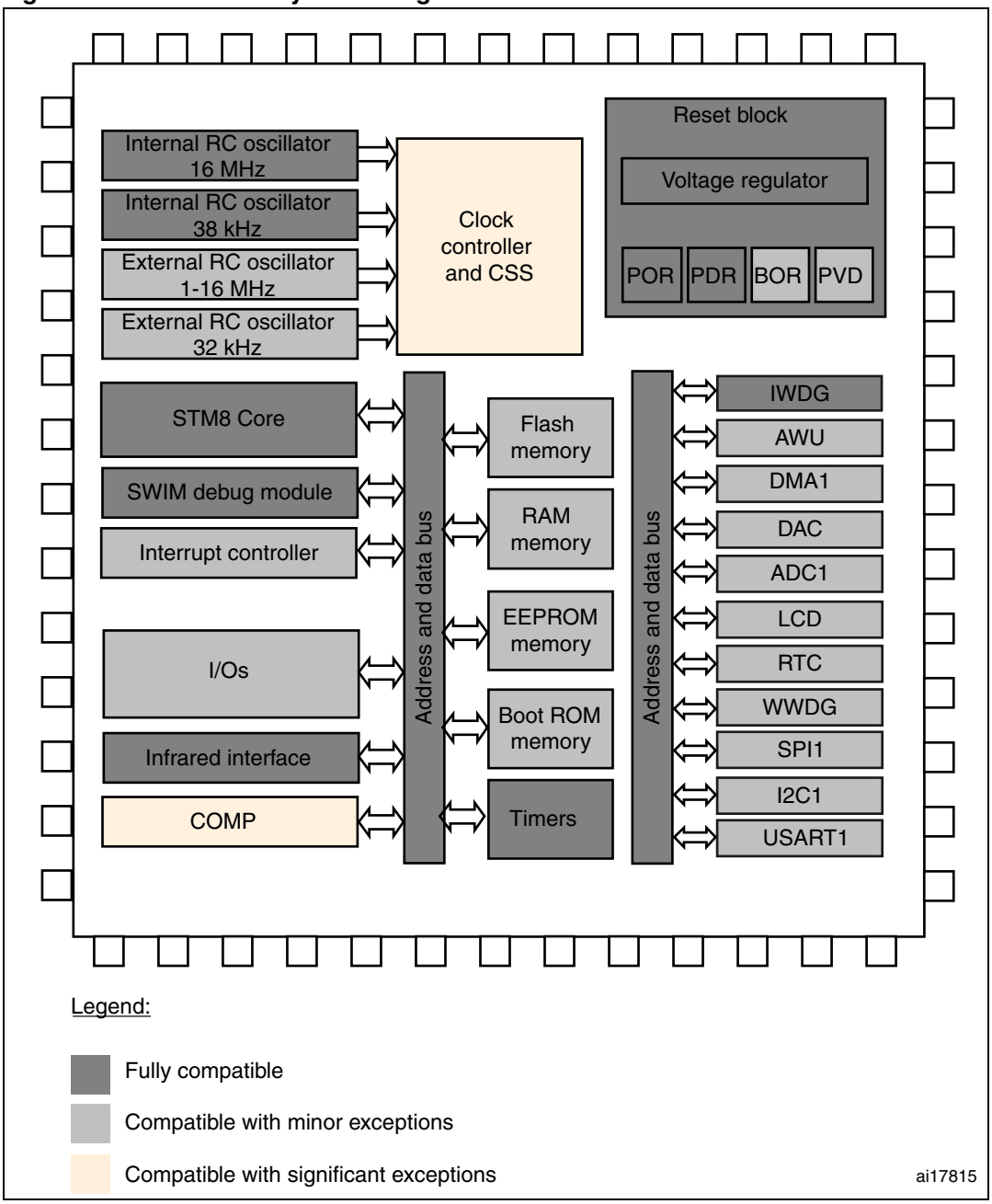
A few peripherals have additional features or less important functionalities compared to the same peripheral in another STM8L sub-family. For these particular peripherals you have to adapt the software drivers and check all possible hardware dependencies.

The peripheral and functional blocks in the following list are compatible with significant exceptions. The package pinout is high on the list as this aspect requires special attention:

- Package pinout
- CLK
- COMP

You can find more details in *Section 3: Block-by-block compatibility analysis*. You can also refer to *Figure 1: STM8L family block diagram*.

Figure 1. STM8L family block diagram



1. LCD, ADC1, WWDG, RTC, DAC, DMA, Boot ROM and AWU are fully compatible but not present in all STM8L devices.

1.5 Firmware library

The peripheral compatibility throughout STM8L MCU families promotes platform design and eases significantly the migration from one product line to the other. The software support is however essential during development time. Extensive software libraries are available for both STM8L10x and STM8L15x devices, providing the user with a hardware abstraction

layer (HAL) for all MCU resources. Moreover, there is not a single control/status bit that is not covered by a C function or an API.

The software library covers three abstraction levels, and it includes:

1. A complete register address map with all bits, bit fields and registers declared in C. By providing this map, the software library makes the designers' task much lighter and, even more importantly so, it gives all the benefits of a bug-free reference mapping file, thus speeding up the early project phase.
2. A collection of routines and data structures in API form, that covers all peripheral functions. This collection can directly be used as a reference framework, since it also includes macros for supporting core-related intrinsic features and common constant and data type definition. Moreover, it is compiler agnostic and can therefore be used with any existing or future toolchain. It was developed using the MISRA C automotive standard.
3. A set of examples covering all available peripherals (35 examples so far for the STM8L10x sub-family, 35 for the STM8L15x sub-family), with template projects for the most common development toolchains. With the appropriate hardware evaluation board, only a few hours are needed to get started with a brand new microcontroller. It is then up to you to choose how to use the library. You can either pick up the files useful for the design, use examples to get trained or quickly evaluate the product. You can also use the API to save development time.

Let us now have a look at the few key files and concepts. Two separate libraries support the STM8L10x and STM8L15x devices. In the file names below, you simply need to replace the "*stm8l1xx_*" prefix by "*stm8l10x*" or "*stm8l15x*" depending on the chosen product.

- **stm8l1xx_.h**

This file is the only header file that must be included in the C source code, usually in *main.c*. This file contains:

  – data structures and address mapping for all peripherals
  – macros to access peripheral register hardware (for bit manipulation for instance), plus STM8 core intrinsics
  – a configuration section used to select the device implemented in the target application. You also have the choice to use or not the peripheral drivers in the application code (that is code based on direct access to registers rather than through API drivers)

- **stm8l1xx_conf.h**

This is the peripheral driver configuration file, where you specify the peripherals you want to use in your application, plus a few application-specific parameters such as the crystal frequency.

- **stm8l1xxx_it.c**

This file contains the template IRQ handler to be filled, but this is already the first development step.

Once you have understood the above operating principle and file organization, for simple applications, you could virtually switch from one product to the other without referring to the reference manual. *Figure 2: STM8L10x code example* and *Figure 3: STM8L15x code example* show the initialization code (using the firmware library) for STM8L10x and STM8L15x products, respectively.

Figure 2. STM8L10x code example

```
/* SPI configuration */
SPI_Init(SPI_FirstBit_MSB,
         SPI_BaudRatePrescaler_2,
         SPI_Mode_Master,
         SPI_CPOL_High,
         SPI_CPHA_2Edge,
         SPI_Direction_2Lines_FullDuplex,
         SPI_NSS_Soft);

/* Enable SPI */
SPI_Cmd(ENABLE);
```

Figure 3. STM8L15x code example

```
/* SPI configuration */
SPI_Init(SPI1,
         SPI_FirstBit_MSB,
         SPI_BaudRatePrescaler_2,
         SPI_Mode_Master,
         SPI_CPOL_High,
         SPI_CPHA_2Edge,
         SPI_Direction_2Lines_FullDuplex,
         SPI_NSS_Soft,
         0x07);
/* Enable SPI  */
SPI_Cmd(SPI1, ENABLE);
```

All parameters are identical, and the procedure is similar with two function calls for both configuration and startup.

The main difference is the additional peripherals of the same type that are added in the STM8L15x sub-family:

- when migrating from STM8L10x sub-family to STM8L15x sub-family, the additional peripheral must be added in every firmware library function as the first parameter (refer to *Figure 3*)
- when migrating from STM8L15x sub-family to STM8L10x sub-family, the additional peripheral given as the first parameter in every firmware library function must be removed (refer to *Figure 2*)

Another difference in the SPI peripheral is the CRC capability which is supported only in the STM8L15x devices. In the above example (*Figure 3.*), the CRC polynomial “0x07” is explicitly defined. This CRC polynomial parameter must be removed from the “init” function in the case of a migration from the STM8L15x sub-family to the STM8L10x sub-family.

The following table shows an overview of the firmware library compatibility between the two families. It describes the differences in terms of:

1. **New features** like the IrDA, Smartcard, Halfduplex features
2. **Common features** but with different implementations, like the comparator initialization.

To migrate from the STM8L10x sub-family to the STM8L15x sub-family when using the communication peripherals (SPI, I2C or USART), you have to add the additional peripheral of the same type as the first parameter in every function used from the firmware library. A full compatibility, in terms of function parameter names, is guaranteed for all common features.

Table 2. STM8L firmware library compatibility

Peripheral	STM8L10x	STM8L15x
Flash		/* New functions */ 1/ void FLASH_ProgramOptionByte(uint16_t Address, uint8_t Data); 2/ void FLASH_EraseOptionByte(uint16_t Address); 3/ void FLASH_PowerWaitModeConfig(FLASH_Power_TypeDef FLASH_Power); 4/ void FLASH_PowerRunModeConfig(FLASH_Power_TypeDef FLASH_Power); 5/ FLASH_PowerStatus_TypeDef FLASH_GetPowerStatus(void);
CLK	/* New functions */ 1/ void CLK_CCOCmd(FunctionalState NewState); 2/ void CLK_PeripheralClockConfig(CLK_Peripheral_TypeDef CLK_Peripheral, FunctionalState NewState); 3/ void CLK_MasterPrescalerConfig(CLK_MasterPrescaler_TypeDef CLK_MasterPrescaler); 4/void CLK_CCOConfig(CLK_Output_TypeDef CLK_Output);	/* New functions */ 1/ void CLK_HSICmd(FunctionalState NewState); 2/ void CLK_AdjustHSICalibrationValue(uint8_t CLK_HSICalibrationValue); 3/ void CLK_LSICmd(FunctionalState NewState); 4/ void CLK_HSEConfig(CLK_HSE_TypeDef CLK_HSE); 5/ void CLK_LSEConfig(CLK_LSE_TypeDef CLK_LSE); 6/ void CLK_SYSCLKSourceConfig(CLK_SYSCLKSource_TypeDef CLK_SYSCLKSource); 7/ void CLK_SYSCLKDivConfig(CLK_SYSCLKDiv_TypeDef CLK_SYSCLKDiv); 8/ void CLK_SYSCLKSourceSwitchCmd(FunctionalState NewState); 9/ CLK_SYSCLKSource_TypeDef CLK_GetSYSCLKSource(void); 10/ void CLK_ClockSecuritySystemEnable(void); 11/ void CLK_ITConfig(CLK_IT_TypeDef CLK_IT, FunctionalState NewState); 12/ void CLK_CCOConfig(CLK_CCOSource_TypeDef CLK_CCOSource, CLK_CCODiv_TypeDef CLK_CCODiv); 13/ void CLK_RTCClockConfig(CLK_RTCCLKSource_TypeDef CLK_RTCCLKSource, CLK_RTCCLKDiv_TypeDef CLK_RTCCLKDiv); 14/ void CLK_BEEPClockConfig(CLK_BEEPCLKSource_TypeDef CLK_BEEPCLKSource); 15/ void CLK_PeripheralClockConfig(CLK_Peripheral_TypeDef CLK_Peripheral, FunctionalState NewState);

Table 2. STM8L firmware library compatibility (continued)

Peripheral	STM8L10x	STM8L15x
COMP	/*Updated functions*/ 1/ void COMP_Init(COMP_Selection_TypeDef COMP_Selection, COMP_Reference_TypeDef COMP_Reference, COMP_Polarity_TypeDef COMP_Polarity); 2/ void COMP_Cmd(FunctionalState NewState); 3/ void COMP_SelectionConfig(COMP_Selection_TypeDef COMP_Selection, FunctionalState NewState); 4/ void COMP_ITConfig(COMP_IT_TypeDef COMP_IT, FunctionalState NewState); 5/ void COMP_TIM2Config(COMP_TIM2Config_TypeDef COMP_TIM2Config); 6/ void COMP_SwitchConfig(COMP_Switch_TypeDef COMP_Switch, FunctionalState NewState); 7/ void COMP_TIMConnect(COMP_TimersConnection_TypeDef COMP_TIMConnection); 8/ void COMP_SelectPolarity(COMP_Polarity_TypeDef COMP_Polarity); 9/ void COMP_SetReference(COMP_Reference_TypeDef COMP_Reference); 10/ FlagStatus COMP_GetOutputStatus(COMP_Output_TypeDef COMP_Output); 11/ FlagStatus COMP_GetFlagStatus(COMP_FLAG_TypeDef COMP_Flag);	/*Updated functions*/ 1/ void COMP_Init(COMP_InvertingInput_Typedef COMP_InvertingInput, COMP_OutputSelect_Typedef COMP_OutputSelect, COMP_Speed_TypeDef COMP_Speed); 2/ void COMP_VrefintToCOMP1Connect(FunctionalState NewState); 3/ void COMP_EdgeConfig(COMP_Selection_TypeDef COMP_Selection, COMP_Edge_TypeDef COMP_Edge); 4/ COMP_OutputLevel_TypeDef COMP_GetOutputLevel(COMP_Selection_TypeDef COMP_Selection); 5/ void COMP_WindowCmd(FunctionalState NewState); 6/ void COMP_ITConfig(COMP_Selection_TypeDef COMP_Selection, FunctionalState NewState); 7/ void COMP_TriggerConfig(COMP_TriggerGroup_TypeDef COMP_TriggerGroup, COMP_TriggerPin_TypeDef COMP_TriggerPin, FunctionalState NewState); 8/ void COMP_VrefintOutputCmd(FunctionalState NewState); 9/ void COMP_SchmittTriggerCmd(FunctionalState NewState); 10/ FlagStatus COMP_GetFlagStatus(COMP_Selection_TypeDef COMP_Selection); 11/ void COMP_ClearFlag(COMP_Selection_TypeDef COMP_Selection); 12/ ITStatus COMP_GetITStatus(COMP_Selection_TypeDef COMP_Selection); 13/ void COMP_ClearITPendingBit(COMP_Selection_TypeDef COMP_Selection);

Table 2. STM8L firmware library compatibility (continued)

Peripheral	STM8L10x	STM8L15x
I2C		/* New functions */ 1/ void I2C_DMACmd(I2C_TypeDef* I2Cx, FunctionalState NewState); 2/ void I2C_DMALastTransferCmd(I2C_TypeDef* I2Cx, FunctionalState NewState); 3/ void I2C_ARPCmd(I2C_TypeDef* I2Cx, FunctionalState NewState); 4/ void I2C_SMBusAlertConfig(I2C_TypeDef* I2Cx, I2C_SMBusAlert_TypeDef I2C_SMBusAlert); 5/ void I2C_PECPositionConfig(I2C_TypeDef* I2Cx, I2C_PECPosition_TypeDef I2C_PECPosition); 6/ void I2C_TransmitPEC(I2C_TypeDef* I2Cx, FunctionalState NewState); 7/ void I2C_CalculatePEC(I2C_TypeDef* I2Cx, FunctionalState NewState);
SPI	/*Updated functions*/ 1/ void SPI_Init(SPI_FirstBit_TypeDef SPI_FirstBit, SPI_BaudRatePrescaler_TypeDef SPI_BaudRatePrescaler, SPI_Mode_TypeDef SPI_Mode, SPI_CPOL_TypeDef SPI_CPOL, SPI_CPHA_TypeDef SPI_CPHA, SPI_DirectionMode_TypeDef SPI_Data_Direction, SPI_NSS_TypeDef SPI_Slave_Management);	/* New functions */ 1/ void SPI_TransmitCRC(SPI_TypeDef* SPIx); 2/ void SPI_CalculateCRCCmd(SPI_TypeDef* SPIx, FunctionalState NewState); 3/ uint8_t SPI_GetCRC(SPI_TypeDef* SPIx, SPI_CRC_TypeDef SPI_CRC); 4/ void SPI_ResetCRC(SPI_TypeDef* SPIx); 5/ uint8_t SPI_GetCRCPolynomial(SPI_TypeDef* SPIx); 6/ void SPI_DMACmd(SPI_TypeDef* SPIx, SPI_DMAReq_TypeDef SPI_DMAReq, FunctionalState NewState); /*Updated functions*/ 1/ void SPI_Init(SPI_TypeDef* SPIx, SPI_FirstBit_TypeDef SPI_FirstBit, SPI_BaudRatePrescaler_TypeDef SPI_BaudRatePrescaler, SPI_Mode_TypeDef SPI_Mode, SPI_CPOL_TypeDef SPI_CPOL, SPI_CPHA_TypeDef SPI_CPHA, SPI_DirectionMode_TypeDef SPI_Data_Direction, SPI_NSS_TypeDef SPI_Slave_Management, uint8_t CRCPolynomial);

Table 2. STM8L firmware library compatibility (continued)

Peripheral	STM8L10x	STM8L15x
USART		/* New functions */ 1/ void USART_HalfDuplexCmd(USART_TypeDef* USARTx, FunctionalState NewState); 2/ void USART_IrDAConfig(USART_TypeDef* USARTx, USART_IrDAMode_TypeDef USART_IrDAMode); 3/ void USART_IrDACmd(USART_TypeDef* USARTx, FunctionalState NewState); 4/ void USART_SmartCardCmd(USART_TypeDef* USARTx, FunctionalState NewState); 5/ void USART_SmartCardNACKCmd(USART_TypeDef* USARTx, FunctionalState NewState); 6/ void USART_SetGuardTime(USART_TypeDef* USARTx, uint8_t USART_GuardTime); 7/ void USART_SetPrescaler(USART_TypeDef* USARTx, uint8_t USART_Prescaler);
ITC	Same peripheral	Same peripheral
PWR	Not available	New peripheral
WFE	Same peripheral	Same peripheral
RST	Same peripheral	Same peripheral
GPIO	Same peripheral	Same peripheral
EXTI	Same peripheral	Same peripheral
DMA1	Not available	New peripheral
ADC1	Not available	New peripheral
DAC	Not available	New peripheral
IRTIM	Same peripheral	Same peripheral
TIMx	Same peripheral	Same peripheral
AWU	New peripheral	Not available
RTC	Not available	New peripheral
BEEP	Same peripheral	Same peripheral
WWDG	Same peripheral	Same peripheral
IWDG	Same peripheral	Same peripheral
SYSCFG	Not available	New peripheral

2 Planning for migration

To migrate your application from one sub-family to another, you have to analyze the hardware migration as well as the application resources and firmware migration.

2.1 Hardware migration

If you use the same package and the same pin numbers, you can use the same PCB without any modification. All sub-families are pin-to-pin compatible.

2.2 Application resources and firmware migration

You have to analyze the compatibility level of your peripherals between the initial sub-family and the new sub-family in the following cases:

- If you use the same resources and peripherals of the same type
  - you do not have to modify anything in your code except the clock configuration (Example: TIMx)
- If you use the same resources and different peripherals of the same type
  - If one peripheral of the same type is not present any more, you can change the reference to this peripheral and all related features (pin, clock and interrupt configuration).
- If you use new resources and/or new peripherals of the same type
  - If you need to migrate the application from STM8L10x devices to STM815x devices without using the new peripherals, the user software can be kept as is without any modification except the clock configuration. Using the standard Peripherals library has several advantages: it saves coding time while simultaneously reducing application development and integration costs.

The following table explains how to migrate from one sub-family to another and from one package to another.

The table is intended to be used as follows:

- Sub-families are listed in rows. Moving between the rows means changing the sub-family.
- Available package sizes are listed in columns. Moving between columns means changing the pin-count.
- The gray fields represent the migration between each column or row and give the impacted features.

The impact of moves between two subfamilies is common for all available package pairs. Therefore all gray cells in rows are merged into common fields. The text in these common fields is as follows:

- When migrating downwards: the row between both sub-families lists the features that are lost due to the migration.
- When migrating upwards, the row between both sub-families lists the features that are added due to the migration.

A move to the right towards smaller packages mainly leads to a loss of I/O pins. So the content of these cells is a simple list of impacted items only.

This section mainly discusses cases of migration between neighboring pairs. However your project may be a migration over several rows or columns in *Table 3* or even in a diagonal direction. In this case, you should check the differences indicated in each step passed by the vertical and horizontal moves through the following table.

Table 3. STM8L family migration products

		Pin-count ←						→
		48		32		28		20
functionality ↑	STM8L15x		Pin-to-pin compatible		Pin-to-pin compatible			
				Pin-to-pin compatible TIM1 / RTC / ADC / LCD				
functionality ↓	STM8L10x				Pin-to-pin compatible		Pin-to-pin compatible	

3 Block-by-block compatibility analysis

3.1 Package pinout

The following table gives an overview of the available packages in each STM8L family

Table 4. Overview of STM8L family packages

Package	STM8L10x	STM8L15x
LQFP48	Not available	x
LQFP32	x	x
VFQFPN48	Not available	x
WFQFPN32	Not available	x
WFQFPN28	Not available	x
UFQFPN28	Not available	x
UFQFPN32	x	Not available
UFQFPN28	x	Not available
UFQFPN28	x	Not available

3.1.1 Digital power supply

The digital power supply design includes two supply sources. The purpose is to distribute the current flowing through the I/O logic separately from the rest of the digital microcontroller circuits. Up to two V_{DDIO} / V_{SSIO} pin pairs are used as well as the main V_{DD} / V_{SS} digital supply pair, depending on the pin-count:

- Both V_{DDIO} pairs are present in the 48-pin package in the STM8L15x sub-family.
- Only one V_{DDIO} pin is available to supply power to the I/Os in the STM8L10x and STM8L15x sub-families from 32-pin packages to 20-pin packages.

The total output current is limited by the number of supply pins. The total output current capability declines for smaller packages, regardless of the number of I/Os with high-sink capability.

In the STM8L15x sub-family, the V_{LCD} pin for connecting the external voltage source for the LCD is present on the 48-pin and 32-pin packages when the LCD is available.

3.1.2 ADC power supply and voltage reference

The analog power supply design includes an extra power supply for the analog parts of the microcontroller and an external reference voltage connected via an extra pin pair.

- The V_{DDA}/V_{SSA} and the V_{REF+}/V_{REF-} analog supply pin pair are present on the 28-pin and 48-pin packages in the STM8L15x sub-family. Without these pins, it is not possible

to use the ADC zooming function feature (see *Section 3.4: ADC modes* for more details).

- For the rest of the STM8L15x sub-family (32-pin package) the ADC reference is taken from the main analog supply V_{DDA}.

Note: As the ADC is not present in the STM8L10x family, V_{DDA}/V_{SSA} and V_{REF+}/V_{REF} analog supply pins are not available on packages for this family.

3.1.3 Alternate functions

The main purpose of the alternate feature concept is to keep the microcontroller configurable for different user and application needs. This is especially important and useful in low pin-count packages.

Default alternate functions can be enabled on dedicated pins by settings in the peripheral registers.

In the STM8L15x sub-family, a large number of alternate functions can be remapped to other pins by programming the system configuration controller registers. Consequently, many functions that would otherwise be lost by migrating to a smaller package size are preserved by remapping alternate functions to the remaining pins. For more details on pinout and packages, please refer to the related datasheet.

3.2 GPIO and peripheral registers

3.2.1 Mapping overview

The space for the GPIO and peripheral registers is mapped in the memory area between addresses 0x5000 and 0x57FF. The register blocks for each peripheral available in the sub-family have the same start addresses and the same register names.

The STM8L15x family has some peripherals that are not supported in the STM8L10x family. In addition, some peripherals are not fully compatible between the two families, which explains why two firmware libraries are developed: one for the STM8L10x family and the other for the STM8L15x family. To avoid any confusion for the user, when a peripheral feature is available in more than one family, this feature has the same function name.

The following tables *Table 5: Overview of STM8L family memory addresses* and *Table 6: Overview of STM8L family peripheral addresses* give an overview of the mapping in the STM8L family.

Table 5. Overview of STM8L family memory addresses

Memory	STM8L10x	STM8L15x
RAM including stack	0x00 0000 - 0x00 5FF	0x00 0000 - 0x00 7FF
Reserved	NA	0x00 0800 - 0x00 0FFF
Data EEPROM		0x00 1000 - 0x00 13FF
Reserved	0x00 600 - 0x00 47FF	0x00 1400 - 0x00 47FF
Option bytes	0x00 4800 - 0x00 48FF	0x00 4800 - 0x00 48FF
Reserved	0x00 4900 - 0x00 49FF	0x00 4900 - 0x00 49FF
GPIO and peripheral register	0x00 5000 - 0x00 57FF	0x00 5000 - 0x00 57FF
Reserved	NA	0x00 5800 - 0x00 59FF
Boot ROM		0x00 6000 - 0x00 67FF
Reserved	0x00 5800 - 0x00 7EFF	0x00 6800 - 0x00 7EFF
CPU/SWIM/Debug/ITC Registers	0X00 7F00 - 0x00 7FFF	0X00 7F00 - 0x00 7FFF
Flash program memory	0X00 8000 - 0x00 9FFF	0X00 8000 - 0x00 FFFF

Note: The gray cells show that the memory type is not present.

Table 6. Overview of STM8L family peripheral addresses

Bus	Peripheral	STM8L10x	STM8L15x
Address data bus	GPIO	0x00 5000 - 0x00 5013	0x00 5000 - 0x00 501D
	Reserved	0x00 5014 - 0x00 5049	0x00 501E - 0x00 5049
	Flash	0x00 5050 - 0x00 5054	0x00 5050 - 0x00 5054
	Reserved	NA	0x00 5055 - 0x00 506F
	DMA1		0x00 5070 - 0x00 509A
	Reserved	0x00 5055 - 0x00 509F	0x00 509B - 0x00 509D
	SYSCFG		0x00 509E - 0x00 509F
	ITC-EXTI	0x00 50A0 - 0x00 50A5	0x00 50A0 - 0x00 50A5
	WFE	0x00 50A6 - 0x00 50A7	0x00 50A6 - 0x00 50A8
	Reserved	0x00 50A8 - 0x00 50AF	0x00 50A9 - 0x00 50AF
	RST	0x00 50B0 - 0x00 50B1	0x00 50B0 - 0x00 50B1
	PWR		0x00 50B2 - 0x00 50B3
	Reserved	0x00 50B2 - 0x00 50BF	0x00 50B4 - 0x00 50BF
	CLK	0x00 50C0 - 0x00 50C5	0x00 50C0 - 0x00 50CF
	Reserved	0x00 50C6 - 0x00 50DF	0x00 50D0 - 0x00 50D2
	WWDG		0x00 50D3 - 0x00 50D4
	Reserved	0x00 50C6 - 0x00 50DF	0x00 50D5 - 0x00 50DF
	IWDG	0x00 50E0 - 0x00 50E2	0x00 50E0 - 0x00 50E2
	Reserved	0x00 50E3 - 0x00 50EF	0x00 50E3 - 0x00 50EF
	BEEP/AWU	0x00 50F0 - 0x00 50F3	0x00 50F0 - 0x00 50F3
	Reserved	NA	0x00 50F4 - 0x00 513F
	RTC		0x00 5140 - 0x00 515F
	Reserved	0x00 50F4 - 0x00 51FF	0x00 5160 - 0x00 51FF
	SPI1	0x00 5200 - 0x00 5204	0x00 5200 - 0x00 5207
	Reserved	0x00 5205 - 0x00 520F	0x00 5208 - 0x00 520F
	I2C1	0x00 5210 - 0x00 521D	0x00 5210 - 0x00 521E
	Reserved	0x00 521E - 0x00 522F	0x00 521F - 0x00 522F
	USART1	0x00 5230 - 0x00 5237	0x00 5230 - 0x00 523A
	Reserved	0x00 5238 - 0x00 524F	0x00 523B - 0x00 524F
	TIM2	0x00 5250 - 0x00 5265	0x00 5250 - 0x00 5266
	Reserved	0x00 5266 - 0x00 527F	0x00 5267 - 0x00 527F
	TIM3	0x00 5280 - 0x00 5295	0x00 5280 - 0x00 5296
	Reserved	NA	0x00 5297 - 0x00 52AF
	TIM1		0x00 52B0 - 0x00 52D3
	Reserved	0x00 5296 - 0x00 52DF	0x00 52D4 - 0x00 52DF

Table 6. Overview of STM8L family peripheral addresses (continued)

Bus	Peripheral	STM8L10x	STM8L15x
Address data bus	TIM4	0x00 52E0 - 0x00 52E8	0x00 52E0 - 0x00 52E9
	Reserved	0x00 52EA - 0x00 52FE	0x00 52EA - 0x00 52FE
	IRTIM	0x00 52FF	0x00 52FF
	Reserved	NA	0x00 5300 - 0x00 533F
	ADC1		0x00 5340 - 0x00 5351
	Reserved	NA	0x00 5352 - 0x00 537F
	DAC		0x00 5380 - 0x00 53AD
	Reserved	NA	0x00 53AE - 0x00 53FF
	LCD		0x00 5400 - 0x00 5419
	Reserved	NA	0x00 541A - 0x00 542F
	RI		0x00 5430 - 0x00 543F
	COMP	0x00 5300 - 0x00 5302	0x00 5440 - 0x00 5444

Note: The gray cells show that the peripheral is not present.

3.2.2 GPIO

All STM8L sub-families share the same GPIO architecture with a different number of ports used in each family and package. All products are pin-to-pin compatible.

The following table presents the number of ports and pins used in each superset of each sub-family. For more details on pinout and packages, please refer to the related datasheet.

Table 7. STM8L family GPIOs overview

Ports	STM8L10x	STM8L15x
Port A	PA0 - PA6	PA0 - PA7
Port B	PB0 - PB7	PB0 - PB7
Port C	PC0 - PC6	PC0 - PC7
Port D	PD0 - PD7	PD0 - PD7
Port E		PE0 - PE7
Port F		PF0

Note: All I/Os available in the package are mapped on external interrupt vectors.

3.3 Advanced, general purpose and basic timers

All STM8L sub-families are equipped with the following timers:

- TIM2 general purpose timer (3x16-bit Cap/Com channels)
- TIM3 general purpose timers (2x16-bit Cap/Com channels)
- TIM4 basic timer: (1x8bit, no Cap/Com channel, no output)

Note: In the STM815x devices, the basic timer is especially used to trigger the DAC.

In addition to these timers, the following timer is available only in the STM8L15x sub-family:

- TIM1 advanced control timer: 16-bit up/down auto-reload counter with 16-bit prescaler, wide range of modes, 4x16-bit Cap/Com channels, 3 of them have a complementary output.

All these timers are identical in all products. They are based on the same architecture and are pin-to-pin compatible. On the software side, in all products, they use the same fully compatible driver. The difference lies in the DMA capability feature that is supported only on the STM8L15x sub-family. The differences are shown in the following table.

The only difference between all products is the number of peripherals of the same type. In addition, if a timer is not present in a product, the related trigger is also absent.

Table 8. Features of advanced, general purpose and basic timers

Timer	Counter type	Prescaler	Cap /Com. channels	Complem. outputs	Repet. counter	Ext. trigger / break inputs	Interrupt sources	DMA requests
TIM1 16-bit advanced control timer	Up/ down	from 1 to 65536 (Any integer)	3+1*	3	Yes	Yes	Break Trigger Commutation Capture/Compare i Update event	Commutation Capture/Compare i Update event
TIM2 & TIM3 16-bit general purpose timers	Up/ down	from 1 to 128 (Any power of 2)	2	No	No	Yes	Break Trigger Capture/Compare i Update event	Capture/Compare i Update event
TIM4 8-bit basic timer	Up	from 1 to 32768 (Any power of 2)	0			No	Trigger Update event	Update event

Note: The grey cells show that the functions are not available in the STM8L10x devices (they are available only in the STM8L15x family).

The following table shows the availability of timers in all products.

Table 9. STM8L family timers overview

Timer type	STM8L10x	STM8L15x
Advanced	Not available	TIM1
General purpose	TIM2 TIM3	TIM2 TIM3
Basic	TIM4	TIM4

The following table shows the internal triggers available for timer synchronisation.

Table 10. Overview of STM8L family timer internal trigger

Timers	TIM internal triggers	STM8L10x	STM8L15x
		TIM TRGO events	
TIM1	ITR0		TIM4 TRGO
	ITR1		TIM3 TRGO
	ITR2		TIM2 TRGO
	ITR3		
TIM2	ITR0	TIM4 TRGO	TIM4 TRGO
	ITR1		TIM3 TRGO
	ITR2	TIM3 TRGO	
	ITR3		TIM1 TRGO
TIM3	ITR0	TIM4 TRGO	TIM4 TRGO
	ITR1		
	ITR2		TIM2 TRGO
	ITR3	TIM2 TRGO	TIM1 TRGO
TIM4	ITR0		
	ITR1		TIM3 TRGO
	ITR2	TIM3 TRGO	TIM2 TRGO
	ITR3	TIM2 TRGO	TIM1 TRGO

3.4 ADC modes

This peripheral is only available in the STM8L15x sub-family. The table below presents the pinout of the ADC in this sub-family.

Table 11. Overview of STM8L family ADC channels(1)

Channels	ADC1 (STM8L15x)
AIN0	PA6
AIN1	PA5
AIN2	PA4
AIN3	PC7
AIN4	PC4
AIN5	PC3
AIN6	PC2
AIN7	PD7
AIN8	PD6
AIN9	PD5
AIN10	PD4
AIN11	PB7
AIN12	PB6
AIN13	PB5
AIN14	PB4
AIN15	PB3
AIN16	PB2
AIN17	PB1
AIN18	PB0
AIN19	PD3
AIN20	PD2
AIN21	PD1
AIN22	PD0
AIN23	PE5
AIN24	PF0

1. No AIN channel available in the STM8L10x devices.

The table below lists all TIMx internal triggers that can be used with ADC for the STM8L15x sub-family:

Table 12. TIMx internal triggers(1)

ADC1 TIMx trigger	STM8L15x
TIM1	TIM1_TRGO event
TIM2	TIM2_TRGO event
TIM3	
TIM4	

1. No ADC TIMx trigger available in the STM8L10x devices.

3.5 DAC peripheral

This peripheral is only available in the STM8L15x sub-family. All TIMx internal triggers that can be used with DAC for this sub-family are given in the following table:

Table 13. Overview of STM8L family DACTIMx triggers (1)

DAC TIMx trigger	STM8L15x
TIM1-TRGO	
TIM2-TRGO	
TIM3-TRGO	
TIM4-TRGO	X

1. No DAC TIMx trigger available in the STM8L10x devices.

3.6 COMP peripherals

The STM8L products feature two zero-crossing comparators COMP1 and COMP2 that share the same current bias. For the STM8L10x sub-family, the COMP1 and COMP2 share also the reference voltage.

Each comparator has two inputs: inverting and non inverting inputs. The STM8L15x device offers more capabilities than the STM8L10x device in terms of number and configuration of these inputs.

The following table shows the availability of the COMPx inverting and non inverting inputs in both sub-families.

Table 14. Overview of comparator inputs

	Comparator Input	Low density STM8L10x	Medium density STM8L15x
COMP1	Non inverting input	PB0	PA6
		PB1	PA5
		PD0	PA4
		PD1	PC7
			PC4
			PC3
			PC2
			PD7
			PD6
			PD5
			PD4
			PB7
			PB6
			PB5
			PB4
			PB3
			PB2
			PB1
			PB0
			PD3
			PD2
			PD1
			PD0
			PE5
			PF0
COMP1	Inverting input	PA6	
		V_{SS}	V_{REFINT}

Table 14. Overview of comparator inputs (continued)

	Comparator Input	Low density STM8L10x	Medium density STM8L15x
COMP2	Non inverting input	PB2	PD1
		PB3	PD0
		PD2	PE5
		PD3	
COMP2	Inverting input	PA6	PC7
		V_{SS}	PC4
			PC3
			DAC output
			V_{REFINT}
			$3/4V_{REFINT}$
			$1/2V_{REFINT}$
			$1/4V_{REFINT}$

3.7 LCD peripheral

This peripheral is only available in the STM8L15x sub-family. The table below gives the pinout of the LCD in the STM8L15x sub-family.

Table 15. Overview of STM8L family LCD pins (1)

Channels	LCD (STM8L15x)
COM0	PA4
COM1	PA5
COM2	PA6
COM3	PD1
SEG0	PA7
SEG1	PE0
SEG2	PE1
SEG3	PE2
SEG4	PE3
SEG5	PE4
SEG6	PE5
SEG7	PD0

Table 15. Overview of STM8L family LCD pins (1) (continued)

Channels	LCD (STM8L15x)
SEG8	PD2
SEG9	PD3
SEG10	PB0
SEG11	PB1
SEG12	PB2
SEG13	PB3
SEG14	PB4
SEG15	PB5
SEG16	PB6
SEG17	PB7
SEG18	PD4
SEG19	PD5
SEG20	PD6
SEG21	PD7
SEG22	PC2
SEG23	PC3
SEG24	PC4
SEG25	PC7
SEG26	PE6
SEG27	PE7

1. No LCD COMx and SEGx available in the STM8L10x devices.

3.8 Communication peripherals

3.8.1 SPI

This peripheral is available in all STM8L sub-families. The differences are the supported features:

- DMA capability not supported in the STM8L10x sub-family
- Hardware CRC feature for reliable communication not supported in the STM8L10x sub-family.

When migrating from the STM8L10x sub-family to the STM8L15x sub-family, you have to add the additional peripheral of the same type as first parameter in every function used from the firmware library. You also have to add the CRC polynomial parameter in the *SPI_Init()* function. A full compatibility, in terms of function parameter names, is guaranteed for all common features.

3.8.2 I2C

This peripheral is available in all STM8L sub-families.

Note: If true open drain lines are required, make sure that the default mapping is kept for SDA & SCL functions as these functions are generally mapped to pins with true open drain capabilities (which is not the case when SDA & SCL are remapped to other pins).

The differences are in the supported features:

- DMA capability not supported in the STM8L10x sub-family
- SMBUS 2.0/ PMBus not supported in the STM8L10x sub-family.

When migrating from the STM8L10x sub-family to the STM8L15x sub-family, you have to add the additional peripheral of the same type as first parameter in every function used from the firmware library. For the common features, a full compatibility, in terms of function parameter names, is guaranteed.

Note: I/Os with true open drain capabilities are available for I2C in all STM8L sub-families.

3.8.3 USART

This peripheral is present in the whole STM8L family. The differences lie in the supported features. The following table gives an overview of the supported features for each family.

When migrating from the STM8L10x sub-family to the STM8L15x sub-family, you have to add the additional peripheral of the same type as first parameter in every function used from the firmware library. For the common features, a full compatibility, in terms of function parameter names, is guaranteed.

Refer to the STM8L10x microcontroller family reference manual (RM0013), STM8L15x microcontroller family reference manual (RM0031), STM8L101x datasheet and STM8L15x datasheet for more detailed information.

Table 16. USART special features

USART features	STM8L10x	STM8L15x
Asynchronous mode	x	x
Multibuffer communication (DMA)		x

Table 16. USART special features (continued)

USART features	STM8L10x	STM8L15x
Multiprocessor communication	x	x
Synchronous	x	x
Smartcard		x
Half-duplex (single-wire mode)		x
IrDA		x

3.9 Clock controller

The clock controller in the STM8L15x sub-family is a superset of the clock controller used in the STM8L10x sub-family:

- The common features are:
  - HSI/8 is the default system clock after startup
  - CCO feature, with more options for the STM8L15x sub-family
  - HSI and LSI features (low power, low cost but not accurate)
  - Peripheral clocks are disabled after reset. The user should enable a peripheral clock before using it.
- The additional features in the STM8L15x sub-family are:

1. System clock switching: the clock switching feature provides an easy to use, fast and secure way for the application to switch from one system clock source to another: HSE, HSI, LSE and LSI (with configurable divider). The only system clock available in the STM8L10x sub-family is the HSI (with configurable divider).
2. The clock security system (CSS): if the HSE clock fails due to a broken or disconnected resonator or any other reason, the clock controller activates a stall-safe recovery mechanism by automatically switching to the HSI with the same division factor as that used before the HSE clock failure.
3. More HSE and LSE oscillators (higher cost but better accuracy)
4. More clock settings for new peripherals (LCD, RTC, DAC, DMA...)

The following table gives an overview of clocks and oscillators supported in the STM8L family.

Table 17. Overview of the clocks in the STM8L family

Clocks & Osc	STM8L10x	STM8L15x	Comments
LSE		32.768 kHz OSC	New
LSI	38 kHz RC	38 kHz RC	No change
HSE		1 - 16 MHz OSC	New
HSI	16 MHz RC	16 MHz RC	No change

Table 17. Overview of the clocks in the STM8L family (continued)

Clocks & Osc	STM8L10x	STM8L15x	Comments
CCO CLK	f_{SYSCLK} $f_{SYSCLK}/2$ $f_{SYSCLK}/4$ $f_{SYSCLK}/16$	HSI/Prescaler LSI/Prescaler HSE/Prescaler LSE/Prescaler	f_{SYSCLK} = HSI/1;HSI/2;HSI/4;HSI/8 Prescaler = 1;2;4;8;16;32;64
RTC CLK		32.768 kHz OSC 38 kHz RC 1 - 16 MHz OSC 16 MHz RC	New

3.9.1 LSI clock frequency

In the whole STM8L family, the low-speed internal clock (LSI) is an ultralow internal power source (a few µA), that can be permanently enabled to be used as the clock source for the application during the Active-halt mode. It is not accurate (error of a few 10%), but it can be periodically measured using the precise HSI clock to compensate for chip manufacturing variations or for the drift due to temperature changes for instance.

3.9.2 HSI clock frequency

In the whole STM8L family, the HSI internal oscillator is factory calibrated in intervals of +/-2% of the temperature range “5 to 25°C”. This value can be trimmed by the user within an interval of +3/-4% of the range in eight steps using three trimming register bits. This enables calibration in steps of about 1% of the range.

3.10 BEEP

In the STM8L15x sub-family, the BEEP clock sources can be either the LSE or LSI clocks. However this feature is not available in the STM8L10x sub-family and the BEEP operates by default using the LSI clock source.

3.11 DMA

The STM8L15x sub-family is equipped with one DMA with 4 channels. The following table presents all peripheral DMA requests present in the sub-family.

Table 18. Overview of STM8L family DMA requests(1)

	Channel	Requests	STM8L15x
DMA1	Channel 0	ADC	x
		I2C_RX	x
		TIM1_CC3	x
		TIM2_CC1	x
		TIM3_U	x
		TIM4_U	x
	Channel 1	ADC	x
		SPI_RX	x
		USART_TX	x
DMA1	Channel 1	TIM1_CC4	x
		TIM2_U	x
		TIM3_CC1	x
		TIM4_U	x
	Channel 2	ADC	x
		SPI_TX	x
		USART_RX	x
		TIM1_U	x
		TIM1_CC1	x
		TIM1_COM	x
		TIM3_CC2	x
		TIM4_U	x
	Channel 3	ADC	x
		I2C_TX	x
		DAC	x
		TIM1_CC2	x
		TIM2_CC2	x
		TIM4_U	x

1. No DMA channel available in STM8L10x devices.

3.12 Memory

3.12.1 Flash program memory

The Flash program memory organization differs slightly in each sub-family.

The memory is organized in pages:

- Medium density STM8L15x Flash memory is organized in up to 256 pages of 128 bytes each. Each page is equal to a block of 128 bytes. A block is the maximum amount of memory that can be programmed in a single programming cycle.
- Low density STM8L10x Flash memory is organized in up to 128 pages of 64 bytes. A page consists of a single block of 64 bytes.

Table 19. Overview of the STM8L family Flash interface

Flash features	STM8L10x	STM8L15x
Interface	Common read/write/protection interface	
Size range	4 Kbytes 8 Kbytes	16 Kbytes 32 Kbytes
Address range	0x8000 - 0x9FFF	0x1000 - 0x13FF 0x8000 - 0xFFFF
Block size	64 bytes	128 bytes
Page size	64 bytes	128 bytes
User boot code (UBC)	From 3 pages up to 127 pages	From 2 pages up to 255 pages
Read-while-write (RWW)	Not supported	Supported
Low power wait mode	Not supported	Supported
Low power run mode	Not supported	Supported

3.12.2 Data EEPROM memory

The data EEPROM memory organization differs slightly in each sub-family. The data EEPROM is organized in pages:

- Medium density STM8L15x data EEPROM is organized in up to 8 pages of 128 bytes each.
- Low density STM8L10x data EEPROM is organized in up to 32 pages of 64 bytes.

The start address of the data EEPROM is configured through the DATASIZE option byte in the STM8L10x sub-family and through the address 0x1000 in the STM8L10x sub-family. Refer to the STM8L datasheets for more details.

3.12.3 Boot ROM memory

The Boot ROM memory containing the bootloader code is present in the STM8L15x sub-family. It is not present in the STM8L10x sub-family. The Boot ROM size is 2 Kbytes and its start address is always 0x6000.

3.12.4 RAM memory

The RAM memory always starts from address 0. The first 256 bytes make up the zero page.

3.12.5 Stack

The space for the stack is always located at the end of the RAM memory. So its position in the address space varies depending on the RAM memory size. The stack pointer value is initialized to the upper address at reset and it is decremented each time a byte is pushed onto the stack. The stack size is 513 bytes in all STM8L devices.

3.13 Interrupt mapping

The interrupt vector mapping is compatible for STM8L10x and STM8L15x devices except:

- The new added vectors to support the new peripherals in the STM8L15x devices
- The timer 4 trigger interrupt which is not supported in the STM8L10x devices
- The auto-wakeup interrupt in the STM8L10x that is replaced by the RTC interrupt in the STM8L15x

The following table lists these differences between all sub-families.

Table 20. STM8L interrupt vector differences

N	STM8L10x	STM8L15x
2		DMA1_CHANNEL0_1
3		DMA1_CHANNEL2_3
4	AWU	RTC
5		EXTIE_F_PVD
. . .	. . .	. . .
16		LCD
17		SWITCH_CSS_BREAK_DAC
. . .	. . .	. . .
23		TIM1_UPD_OVF_TRG_COM
24		TIM1_CAP
25	TIM4_UPD_OVF	TIM4_UPD_OVF_TRG

Except for the differences listed above, all other interrupt vectors are identical for all products as shown in the following table.

Table 21. Overview of the STM8L family interrupt vectors

N	STM8L10x	STM8L15x
	RESET	RESET
	TRAP	TRAP
0		
1	Flash	Flash
2		DMA1_CHANNEL0_1
3		DMA1_CHANNEL2_3
4	AWU	RTC
5		EXTIE_F_PVD
6	EXTIB	EXTIB
7	EXTID	EXTID
8	EXTI0	EXTI0
9	EXTI1	EXTI1
10	EXTI2	EXTI2
11	EXTI3	EXTI3
12	EXTI4	EXTI4
13	EXTI5	EXTI5
14	EXTI6	EXTI6
15	EXTI7	EXTI7
16		LCD
17		SWITCH_CSS_BREAK_DAC
18	COMP	ADC1_COMP
19	TIM2_UPD_OVF_TRG_BRK	TIM2_UPD_OVF_TRG_BRK
20	TIM2_CAP	TIM2_CAP
21	TIM3_UPD_OVF_TRG_BRK	TIM3_UPD_OVF_TRG_BRK
22	TIM3_CAP	TIM3_CAP
23		TIM1_UPD_OVF_TRG_COM
24		TIM1_CAP
25	TIM4_UPD_OVF	TIM4_UPD_OVF_TRG
26	SPI	SPI1
27	USART_TX	USART1_TX
28	USART_RX	USART1_RX
29	I2C	I2C1

3.14 Option bytes

The basic structure of the option byte area is compatible across the family. There are some differences detailed in the following table:

1. OPT4 and OPT5 (Number of stabilization clock cycles for HSE and LSE oscillators and Brownout reset respectively) are available only in the STM8L15x family and can be modified on the fly by the application in IAP mode.
2. The independent watchdog option is the OPT4 in the STM8L10x and OPT3 in the STM8L15x.
3. The user boot code size is the OPT2 in the STM8L10x and OPT1 in the STM8L15x.
4. The DATASIZE option byte is available only for the STM8L10x family.

Table 22. Option byte addresses

Address	Option byte name	Low density STM8L10x	Medium density STM8L15x
0x4800	ROP (read-out protection)	OPT1	OPT0
0x4801			
0x4802	UBC (User boot code size)	OPT2	OPT1
0x4803	DATASIZE	OPT3	
0x4804			
0x4805			
0x4806			
0x4807			
0x4808	Independent watchdog option byte	OPT4	OPT3
0x4809	Number of stabilization clock cycles for HSE and LSE oscillators		OPT4
0x480A	BOR (brownout reset)		OPT5
0x480B	Bootloader option bytes (OPTBL)		OPTBL
0x480C			

4 Peripheral pinout through all STM8L families

4.1 Timer pinout

The following table describes the timer pinout for both STM8L families.

Table 23. Timer pinout

Channels	TIM1 pinout				TIM2 pinout				TIM3 pinout			
	STM8L10x		STM8L15x		STM8L10x		STM8L15x		STM8L10x		STM8L15x	
	Reset	Remap	Reset	Remap	Reset	Remap	Reset	Remap	Reset	Remap	Reset	Remap
CH1			PD2		PB0		PB0		PB1		PB1	
CH1N			PD7									
CH2			PD4		PB2		PB2		PD0		PD0	
CH2N			PE1									
CH3			PD5									
CH3N			PE2									
ETR			PD3		PB3		PB3	PA4	PD1		PD1	PA5
BRK			PD6		PA4		PA4		PA5		PA5	

4.2 SPI

The following table describes the SPI pinout for both STM8L families.

Table 24. SPI pinout

Channels	SPI1			
	STM8L10x		STM8L15x	
	Reset	Remap	Reset	Remap
NSS	PB4		PB4	PC5
SCK	PB5		PB5	PC6
MOSI	PB6		PB6	PA3
MISO	PB7		PB7	PA2

4.3 I2C

The following table describes the I2C pinout for both STM8L families.

Table 25. I2C pinout

Channels	I2C1			
	STM8L10x		STM8L15x	
	Reset	Remap	Reset	Remap
SCL	PC1		PC1	
SDA	PC0		PC0	
SMB			PC4	

4.4 USART

The following table describes the USART pinout for both STM8L families.

Table 26. USART pinout

Channels	USART1 pinout			
	STM8L10x		STM8L15x	
	default	Remap	default	Remap
TX	PC3		PC3	PA2 PC5
RX	PC2		PC2	PA3 PC6
CK	PC4		PC4	PA0

5 Revision history

Table 27. Document revision history

Date	Revision	Changes
07-Jun-2010	1	Initial release