Application note

Migration of applications from the STM8L and STM8S series to the STM32C0 series microcontrollers

Introduction

This application note provides guidelines and a methodology to migrate easily from an application based on the STM8L and STM8S series to the STM32C0 series platform. It groups all of the most important information, and lists the main aspects that must be addressed. It describes a simple procedure using the HAL (hardware abstraction layer), and STM32Cube software, to access a larger portfolio.

The STM32C0 platform is a starting point for simple cost-focused applications. It offers easy further migration within a wide range of STM32 products, depending on the application needs (focused on costs, tailored to ultra low-power consumption, high performance, or for products embedding wireless communication).

This document provides details about the hardware, peripheral, and firmware migration.

In addition, this document gives an overview of the STM32 ecosystem, for example the hardware development and IDE/compiler available to start using the STM32C0 series.

For a better understanding, the user must be familiar with STM32 microcontrollers.

For additional information, refer to the documents in Table 1. This does not provide a full list of electrical parameters, for which the device datasheet is the reference document.

1 General information

This document applies to all STM32C0 series devices. All these products are Arm®-based microcontrollers.

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

Table 1. **Document and website references**

Reference	Document
RM0490	STM32C0x1 advanced Arm®-based 32-bit MCUs[1]
DS13866	Arm® Cortex®-M0+ 32-bit MCU, 32 KB flash, 6 KB RAM, 2 x USART, timers, ADC, communication interface I/Fs, 2-3.6V[1]
DS13867	Arm®Cortex®-M0+ 32-bit MCU, 32 KB flash, 12 KB RAM, 2x USART, timers, ADC, communication interface I/Fs, 2-3.6V [1]
PM0223	Cortex®-M0+ programming manual for STM32C0, STM32L0, STM32G0, STM32WL, and STM32WB series[1]
AN5673	Getting started with STM32C0 series hardware development [1]
AN2606	STM32 microcontroller system memory boot mode
AN4894	EEPROM emulation techniques and software for STM32 microcontrollers
AN4899	STM32 microcontroller GPIO hardware settings and low-power consumption
AN1709	EMC design guide for STM8, STM32, and legacy MCUs
STM32CubeProg	https://www.st.com/stm32cubeprog
RM0016	STM8S series and STM8AF series 8-bit microcontrollers
RM0013	STM8L001xx and STM8L101xx microcontroller families
RM0031	STM8L050J3, STM8L051F3, STM8L052C6, STM8L052R8 MCUs and STM8L151/L152, STM8L162, STM8AL31, STM8AL3L lines
PM0051	How to program STM8S and STM8A flash program memory and data EEPROM
STM8 family	All STM8S and STM8L datasheets
	All STM8S and STM8L errata sheets
ES0568	STM32C031xx device errata
ES0569	STM32C011xx device errata

1. *Available at www.st.com. Contact STMicroelectronics when more information is needed.*

2 STM32C0 series overview

The STM32C0 series includes all of the STM8 family standard peripherals, such as SPI and UART. (see Table 5 for more details.) It also has a set of peripherals with advanced features and optimized power consumption level, including:

- 32-bit CPU with maximum CPU frequency of 48 MHz
- DMA
- 12-bit ADC
- I^{2}S

3 Hardware migration

3.1 Pinout compatibility

STM32C0 devices use a different system of power distribution (single supply pair), with the merging of V_{DDA} and V_{DD} and the absence of VCAP, embedding capacitance required internally by the regulator. The STM32C0 provides better GPIO density than the STM8L/S series. It needs a 3.3 V supply voltage (compared to the STM8S 5 V supply voltage), with an allowed range of 2 V to 3.6 V.

Due to the significant difference between the STM8L/S series and the STM32C0 series, there is no pin-to-pin compatibility. In case of replacement, the PCB routing must be reworked.

Table 2. **Additional IOs for STM32C0 vs STM8**

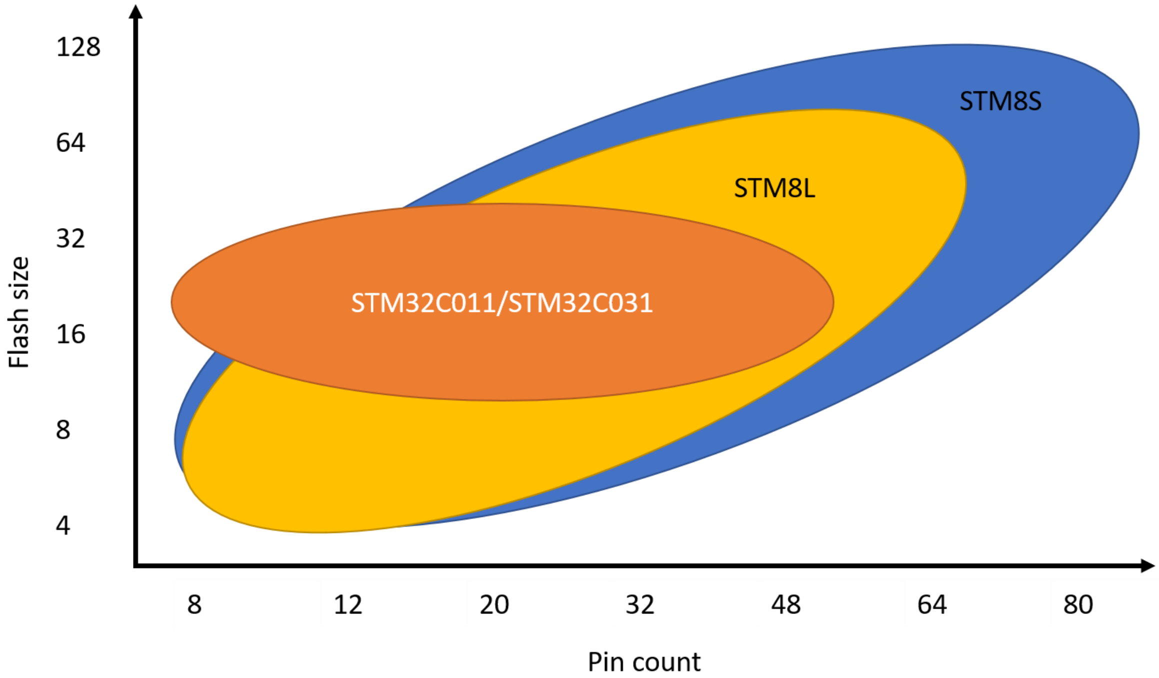
Package pin count	GPIO number in STM32C0 series	GPIO number in STM8S series	GPIO number in STM8L series	Difference I/Os
8	6	5	6	0 to +1
20	18	16	18	0 to +2
32	30	25 to 28	28 to 30	0 to +5
48	45	38	41	+4 to +7

Table 3. **Package type**

Package pins	STM32C0 series	STM8S series	STM8L series
8	SO8N	SO8N	SO8N
12	WLCSP	-	-
20	TSSOP/UFQFPN	TSSOP/UFQFPN/SO	TSSOP/UFQFPN
28	UFQFPN	-	UFQPN/CSP
32	UFQFPN / LQFP	UFQFPN / LQFP / SDIP	LQFP/UFQFPN/CSP
44	-	LQFP	-
48	UFQFPN / LQFP	LQFP	UFQFPN / LQFP
64	(1)(2)	LQFP	LQFP
80	(1)	LQFP	LQFP

1. *STM32G0 series support this package.*
2. *STM32L4 and STM32L0 series support this package.*

Figure 1. Flash memory size versus pin count

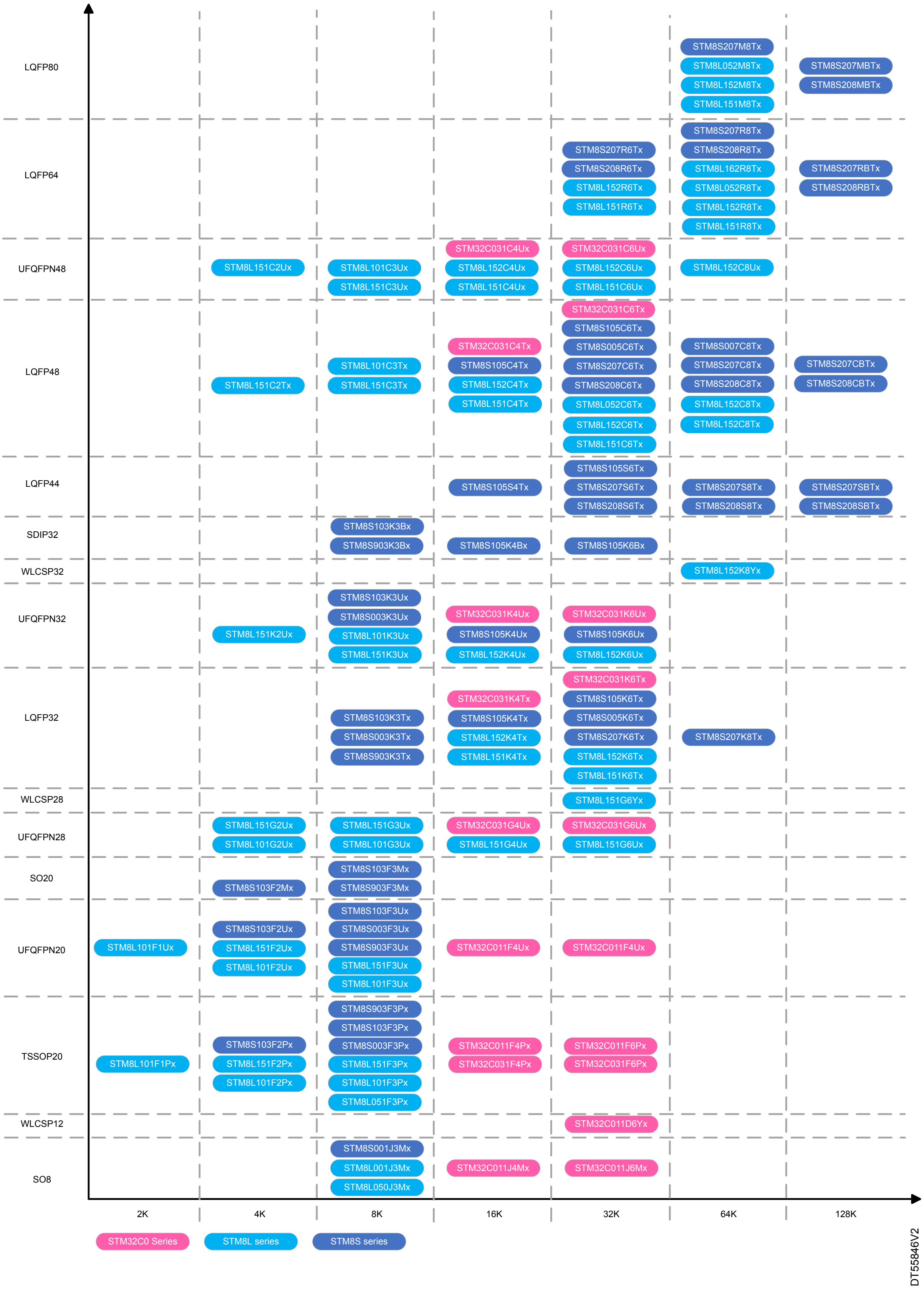


3.2 Sales type selection

Figure 2 helps the user to find the suitable sales type to migrate from the STM8L/S series to the STM32C0 series, with a flash memory size and package comparison.

If the STM32C0 series does not support the desired package or flash memory, the user can check the part available on the STM32G0, STM32L0, and STM32L4 series.

Figure 2. Sales type help selection



	2K	4K	8K	16K	32K	64K	128K
LQFP80						STM8S207M8Tx STM8L052M8Tx STM8L152M8Tx STM8L151M8Tx	STM8S207MBTx STM8S208MBTx
LQFP64					STM8S207R6Tx STM8S208R6Tx STM8L152R6Tx STM8L151R6Tx	STM8S207R8Tx STM8S208R8Tx STM8L162R8Tx STM8L052R8Tx STM8L152R8Tx STM8L151R8Tx	STM8S207RBTx STM8S208RBTx
UFQFPN48		STM8L151C2Ux	STM8L101C3Ux STM8L151C3Ux	STM32C031C4Ux STM8L152C4Ux STM8L151C4Ux	STM32C031C6Ux STM8L152C6Ux STM8L151C6Ux	STM8L152C8Ux	
LQFP48		STM8L151C2Tx	STM8L101C3Tx STM8L151C3Tx	STM32C031C4Tx STM8S105C4Tx STM8L152C4Tx STM8L151C4Tx	STM32C031C6Tx STM8S105C6Tx STM8S005C6Tx STM8S207C6Tx STM8S208C6Tx STM8L052C6Tx STM8L152C6Tx STM8L151C6Tx	STM8S007C8Tx STM8S207C8Tx STM8S208C8Tx STM8L152C8Tx STM8L152C8Tx	STM8S207CBTx STM8S208CBTx
LQFP44				STM8S105S4Tx	STM8S105S6Tx STM8S207S6Tx STM8S208S6Tx	STM8S207S8Tx STM8S208S8Tx	STM8S207SBTx STM8S208SBTx
SDIP32			STM8S103K3Bx STM8S903K3Bx	STM8S105K4Bx	STM8S105K6Bx		
WLCSP32						STM8L152K8Yx	
UFQFPN32		STM8L151K2Ux	STM8S103K3Ux STM8S003K3Ux STM8L101K3Ux STM8L151K3Ux	STM32C031K4Ux STM8S105K4Ux STM8L152K4Ux	STM32C031K6Ux STM8S105K6Ux STM8L152K6Ux		
LQFP32			STM8S103K3Tx STM8S003K3Tx STM8S903K3Tx	STM32C031K4Tx STM8S105K4Tx STM8L152K4Tx STM8L151K4Tx	STM32C031K6Tx STM8S105K6Tx STM8S005K6Tx STM8S207K6Tx STM8L152K6Tx STM8L151K6Tx	STM8S207K8Tx	
WLCSP28					STM8L151G6Yx		
UFQFPN28		STM8L151G2Ux STM8L101G2Ux	STM8L151G3Ux STM8L101G3Ux	STM32C031G4Ux STM8L151G4Ux	STM32C031G6Ux STM8L151G6Ux		
SO20		STM8S103F2Mx	STM8S103F3Mx STM8S903F3Mx				
UFQFPN20	STM8L101F1Ux	STM8S103F2Ux STM8L151F2Ux STM8L101F2Ux	STM8S103F3Ux STM8S003F3Ux STM8S903F3Ux STM8L151F3Ux STM8L101F3Ux	STM32C011F4Ux	STM32C011F4Ux		
TSSOP20	STM8L101F1Px	STM8S103F2Px STM8L151F2Px STM8L101F2Px	STM8S903F3Px STM8S103F3Px STM8S003F3Px STM8L151F3Px STM8L101F3Px STM8L051F3Px	STM32C011F4Px STM32C031F4Px	STM32C011F6Px STM32C031F6Px		
WLCSP12					STM32C011D6Yx		
SO8			STM8S001J3Mx STM8L001J3Mx STM8L050J3Mx	STM32C011J4Mx	STM32C011J6Mx		

3.3 Pinout migration

A comparison between two packages is available to help the customer to evaluate how much the PCB needs to be reworked. The comparison considers only the different position of power pins, reset, and oscillator input/output. The product datasheets give more details in case the user would like to check timer, communication peripherals, or even ADC channel similarities.

Figure 3. TSSOP20 GPIO comparison

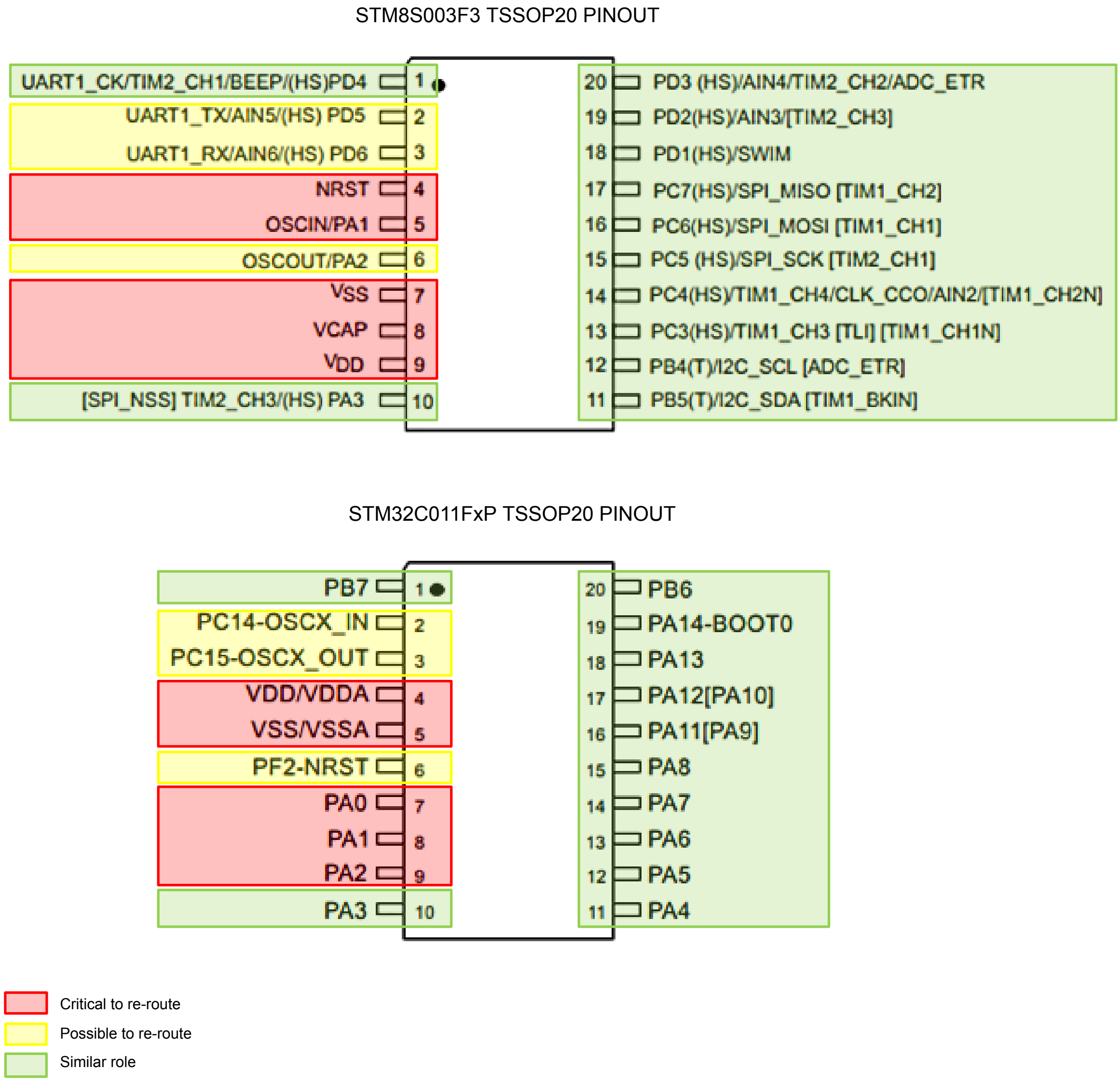
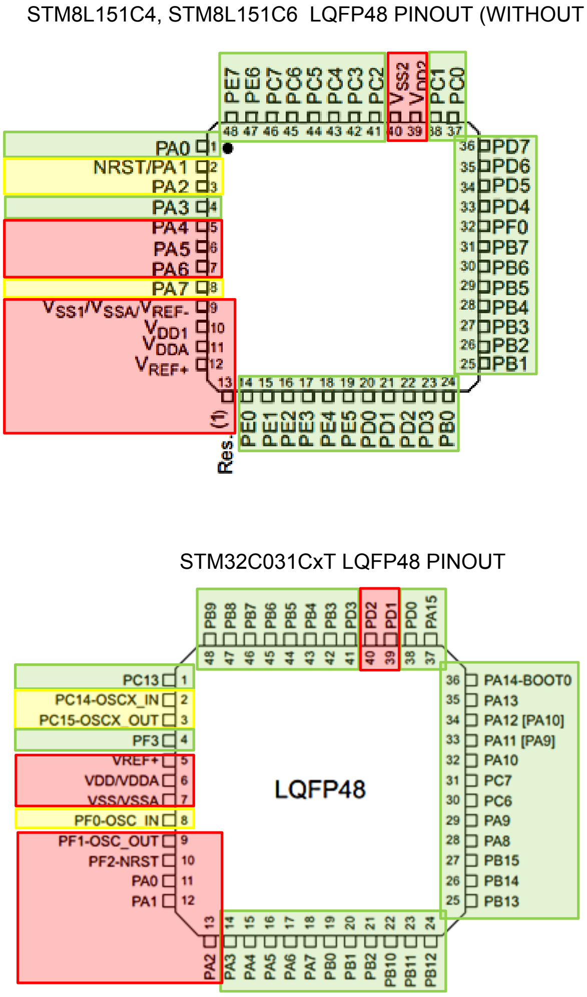

STM8S003F3 TSSOP20 PINOUT
UART1_CK/TIM2_CH1/BEEP/(HS)PD4 1
UART1_TX/AIN5/(HS) PD5 2
UART1_RX/AIN6/(HS) PD6 3
NRST 4
OSCIN/PA1 5
OSCOUT/PA2 6
VSS 7
VCAP 8
VDD 9
[SPI_NSS] TIM2_CH3/(HS) PA3 10
20 PD3 (HS)/AIN4/TIM2_CH2/ADC_ETR
19 PD2(HS)/AIN3/[TIM2_CH3]
18 PD1(HS)/SWIM
17 PC7(HS)/SPI_MISO [TIM1_CH2]
16 PC6(HS)/SPI_MOSI [TIM1_CH1]
15 PC5 (HS)/SPI_SCK [TIM2_CH1]
14 PC4(HS)/TIM1_CH4/CLK_CCO/AIN2/[TIM1_CH2N]
13 PC3(HS)/TIM1_CH3 [TLI] [TIM1_CH1N]
12 PB4(T)/I2C_SCL [ADC_ETR]
11 PB5(T)/I2C_SDA [TIM1_BKIN]
STM32C011FxP TSSOP20 PINOUT
PB7 1
PC14-OSCX_IN 2
PC15-OSCX_OUT 3
VDD/VDDA 4
VSS/VSSA 5
PF2-NRST 6
PA0 7
PA1 8
PA2 9
PA3 10
20 PB6
19 PA14-BOOT0
18 PA13
17 PA12[PA10]
16 PA11[PA9]
15 PA8
14 PA7
13 PA6
12 PA5
11 PA4
🟥 Critical to re-route
🟨 Possible to re-route
🟩 Similar role

Figure 4. LQFP48 GPIO comparison



STM8L151C4, STM8L151C6 LQFP48 PINOUT (WITHOUT LCD)

STM32C031CxT LQFP48 PINOUT

🟥 Critical to re-route

🟨 Possible to re-route

🟩 Similar role

4 Boot mode selection

The boot configuration of the STM32C0 is based on the STM32 M0+ core products.

In the STM8L/S series, the software can boot only from the flash memory or the system bootloader. The STM32C0 series permits to locate the BOOT vector in the flash memory, the system memory (bootloader), or the RAM, based on Table 4. It relocates the boot memory start address if, for example, the user chooses to boot from the main flash memory. This memory area is aliased in the boot memory space (0x0000 0000), but is still accessible from its original memory space (0x0800 0000). It is reciprocal to other boot area.

A feature to check if the device is virgin is implemented on the STM32C0 series. If the BOOT0 pin defines the main flash memory as the target boot area, and after loading the option byte, the flash memory interface checks if the first location of the main memory is programmed. It returns the result on the FLASH_ACR register.

Table 4. Boot mode configuration

Boot mode configuration					Selected boot area
BOOT_LOCK bit	nBOOT1 bit	BOOT0 pin	nBOOT_SEL bit	nBOOT0 bit	
0	X	0	0	X	Main flash memory
0	1	1	0	X	System memory
0	0	1	0	X	Embedded SRAM
0	X	X	1	1	Main flash memory
0	1	X	1	0	System memory
0	0	X	1	0	Embedded SRAM
1	X	X	X	X	Main flash memory forced

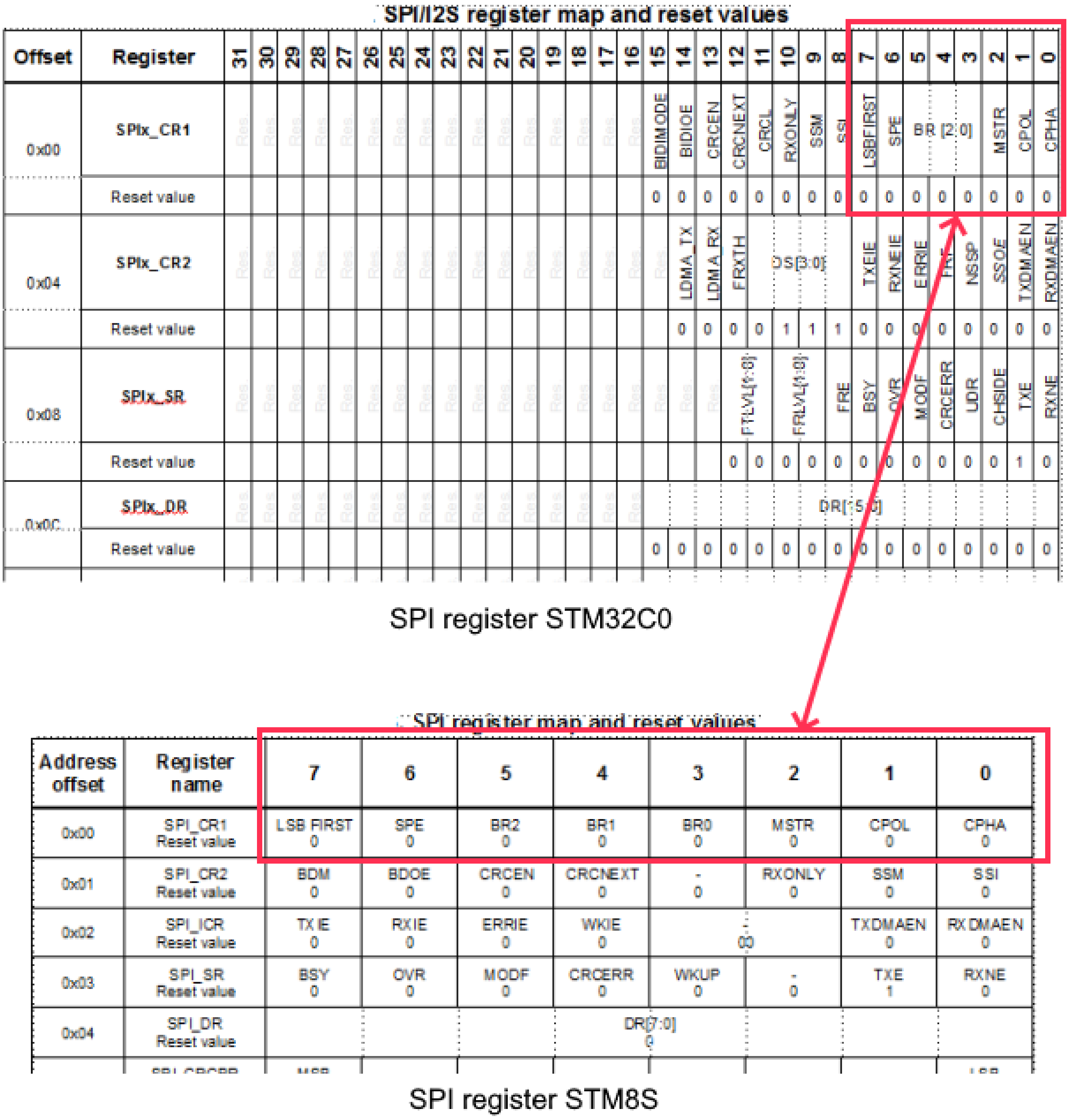
5 Peripheral migration

5.1 STM32 product cross-compatibility

The peripheral platform shares a common base. There are some differences between STM8 and STM32C0 peripherals due to continuous improvement, and the addition of new functionalities. The comparison below helps the user to identify and use these improvements.

The major difference between the STM8 and the STM32 is the number of register bits: 32 or 16 bits in STM32. Only 8 bits in STM8. Sometimes, register names are similar.

Figure 5. Register name sharing



SPI/I2S register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	SPIx_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BIDIMODE	BIDIOE	CRCEN	CRCNEXT	CRCL	RXONLY	SSM	SSI	LSBFIRST	SPE	BR [2:0]			MSTR	CPOL	CPHA
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	SPIx_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LDMA_TX	LDMA_RX	FRXTH	DS[3:0]				TXEIE	RXNEIE	ERRIE	FRF	NSSP	SSOE	TXDMAEN	RXDMAEN
	Reset value																		0	0	0	0	1	1	1	0	0	0	0	0	0	0	0
0x08	SPIx_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FTLVL[1:0]		FRLVL[1:0]		FRE	BSY	OVR	MODF	CRCERR	UDR	CHSIDE	TXE	RXNE
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	1	0
0x0C	SPIx_DR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SPI register STM32C0

SPI register map and reset values

Address offset	Register name	7	6	5	4	3	2	1	0
0x00	SPI_CR1 Reset value	LSB FIRST 0	SPE 0	BR2 0	BR1 0	BR0 0	MSTR 0	CPOL 0	CPHA 0
0x01	SPI_CR2 Reset value	BDM 0	BDOE 0	CRCEN 0	CRCNEXT 0	- 0	RXONLY 0	SSM 0	SSI 0
0x02	SPI_ICR Reset value	TXIE 0	RXIE 0	ERRIE 0	WKIE 0	- 00		TXDMAEN 0	RXDMAEN 0
0x03	SPI_SR Reset value	BSY 0	OVR 0	MODF 0	CRCERR 0	WKUP 0	- 0	TXE 1	RXNE 0
0x04	SPI_DR Reset value	DR[7:0] 0							
	SPI_CRCPR	MSB							LSB

SPI register STM8S

Table 5. Peripheral summary of STM32C0 series and STM8S and STM8L series

Peripheral		STM32C0 series	STM8S series	STM8L series
Power supply		See Table 9		
Core		Cortex® M0+ (32-bit)	STM8 core (8-bit)	STM8 core (8-bit)
Maximum frequency		48 MHz	Up to 24 MHz	16 MHz
Flash memory		Up to 32 Kbytes	Up to 128 Kbytes	Up to 64 Kbytes
SRAM		Up to 12 Kbytes	Up to 6 Kbytes	Up to 4 Kbytes
EEPROM		Emulated in the flash memory[1]	Up to 2 Kbytes	Up to 2 Kbytes
TIMER	General purpose (16-bit)	4	Up to 2	Up to 3
	Advanced (16-bit)	1	Up to 1	Up to 1
	Basic (8-bit)	0	Up to 1	Up to 1
ADC		1	1	1
DAC		[3][4][5]	-	Up to 2
DMA (number of independently configurable channels request)		3	-	Up to 4
USART		2	1 to 2 (UART only)	1 to 3
SPI		1	1	1 to 2
I^{2}C		1	1 to 4	1 to 2
I^{2}S (Inter-IC-sound)		1	-	-
CRC		X	-	-
RTC		X	-	X[6]
WWDG		X	X	X[6]
IWDG		X	X	X[6]
LCD		-[4][5]	-	X
COMP		-[3][4][5]	-	Up to 2
CAN		-[3][5]	X[6]	X[6]
Bootloader supported peripheral		USART/I^{2}C	UART/SPI	UART/SPI

1. *Refer to the AN4894 in Table 1. Document and website references*
2. *Available at www.st.com. Contact STMicroelectronics when more information is needed.*
3. *STM32G0 series supports this feature.*
4. *STM32L0 series supports this feature.*
5. *STM32L4 series supports this feature.*
6. *Not on all devices.*

5.2 System architecture

The STM32C0 series implement an Arm® 32-bit architecture with Cortex®-M0+ core, while the STM8L/S series use the STM8 8-bit proprietary core. The STM32 uses a RISC instruction set, while the STM8 uses a CISC instruction set. This allows the STM32 to be faster, at the price of greater code size, as described below. See in Table 6 the full list of differences.

Table 6. **Comparison of CPU core**

Feature	Cortex®-M0+	STM8 core
Data path	32-bit	8-bit
Architecture	Von Neumann	Harvard
Pipeline	Two stages	Three stages
Instruction set	RISC	CISC
Program bus data width	32-bit	32-bit
Prefetch buffer	2 x 32-bit	2 x 32-bit
Debug interface	2-wire (SWD)	1-wire (SWIM)
Number of registers	15 x 32-bit, 1 x 64-bit, 3 special registers	11 x 8-bit
Cache instruction	16 bytes	NA

Aligned memory, is an address where an "n-byte" value is stored. It must be divisible by "n". This means:

- Word (32-bit) aligned to an address divisible by 4 [UINT32/INT32].
- Half-word (16-bit) aligned to an address divisible by 2 [UINT16/INT16].
- Byte accesses are always aligned [UINT8/INT8].

The Cortex® -M0+ uses the ARM-v6M. This architecture does not permit an unaligned memory access. If attempted, the CPU raises a hard fault exception.

Usually, compilers are aware of the aligned access requirement, so they automatically adjust in several ways:

- Automatically place variables in aligned addresses.
- Use of packed structures to align members.
- Use of byte-by-byte access whenever a variable is unaligned for some reason.

5.3 Code density and CoreMark®

To help the developer to find the appropiate sales type for their needs, a comparison code size has been made on the CoreMark®. It is easily portable between both families. It ensures that compilers cannot precompute the results. Moreover, it provides the user with a benchmark comparison based on the IAR Embedded Workbench®, with different code optimization.

For further information about the code density between the libraries available on the STM32C0 series, check the Programming part.

Table 7. **Code density between STM32C0 series and STM8L series**

Optimization	Size	Balanced	Medium	Speed	Unit
STM32C0	16073	16921	17085	20473	Bytes
STM8L	15188	14752	15371	18935	

The code size does not increase excessively. However, it is necessary to accommodate an increase of 6 to 15% in code size.

The CoreMark® is not the perfect benchmark to compare both families. This is because one uses 8 bits, and the other uses 32 bits, while the CoreMark® uses a 16/32-bit variable. However, it indicates that the Cortex®-M0+ is seven times better than the STM8 8-bit core.

Table 8. CoreMark® comparison

STM32C0 series	STM8 family	Unit
2.22	0.30	CoreMark/MHz

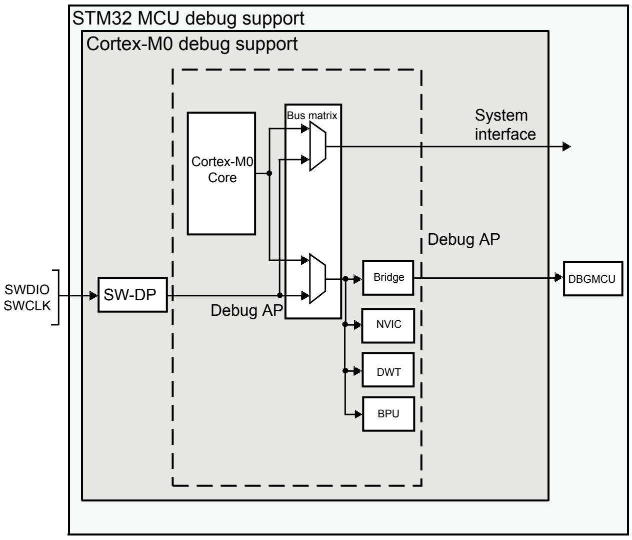
5.4 Debug

The STM32 series use a different debug methodology with respect to the STM8 family. The STM32 devices need two wires for debug, while just one is needed in STM8 devices (SWIM).

The new debug methodology allows:

- SW-DP: serial wire
- BPU: break point unit
- DWT: data watchpoint trigger
- Flexible debug pinout assignment
- NVIC debug
- MCU debug box (support for low-power modes, control over peripheral clocks, etc.)

Figure 6. Block diagram of STM32C0 MCU and Cortex®-M0 +-level debug support



DT19240V1

5.5 Power control peripheral

In the STM32C0 series, the PWR controller presents some differences compared to the STM8S/L series. This is especially the case for the STM8S series, which has a 5.0 V supply.

Table 9. Power control peripheral

PWR	STM32C0 Series	STM8S series	STM8L series
Power supplies	• V_{DD}: 2.0 V to 3.6 V (one pair V_{DD}/ V_{SS}) is the external power supply for the internal regulator and the system analog such as reset, power management, and internal clocks • V_{DDA}: is the analog power supply for the A/D converter and shorted to V_{DD} due to the low number of pins • V_{DDIO}: is the power supply for the I/Os and shorted to V_{DD}, due to the low number of pins • V_{REF+}: 2.0 V to V_{DDA} is the input reference voltage for the ADC, on a lower pin-count package V_{REF+} is shorted to V_{DD}	• V_{DD}: 2.95 V to 5.5 V (one pair V_{DD}/ V_{SS}) is the external power supply for the main regulator ballast transistor supply • V_{DDIO}: 3.0 V to 5.0 V is the power supply for the I/Os and on a lower pin-count package is shorted to V_{DD}, due to the low number of pins • V_{DDA}: 3.0 V to 5.5 V (one pair of V_{DDA}/ V_{SSA}) is the analog power supply for the A/D converter and on a lower pin-count package is shorted to V_{DD}, due to the low number of pins • V_{REF+}: 2.0 V to V_{DDA} is the input reference voltage for the ADC, on a lower pin-count package V_{REF+} is shorted to V_{DD}	• V_{DD}: 1.65 V or 1.8 V to 3.6 V is the external power supply for the main regulator • V_{DDA}: 1.8 V to 3.6 V is the analog power supply for the analog part and on lower pin-count package is shorted to V_{DD}, due to the low number of pins • V_{DDIO}: 1.8 V to 3.6 V is the power supply for the I/Os and on lower pin-count package is shorted to V_{DD}, due to the low number of pins • V_{REF+}: If V_{DDA} > 2.4 V: 2.4 V to V_{DDA} else: V_{REF+} = V_{DDA} is the input reference voltage for the ADC, on lower pin-count package V_{REF+} is shorted to V_{DD}
Power supply supervisor	• Integrated POR/PDR/BOR circuitry	• Integrated POR/PDR circuitry	• Integrated POR/PDR/BOR circuitry • Programmable voltage detector (PVD)

5.6 Power consumption mode

The STM32C0 series and the STM32 family generally have different low-power modes compared to the STM8 family. There are four low-power modes:

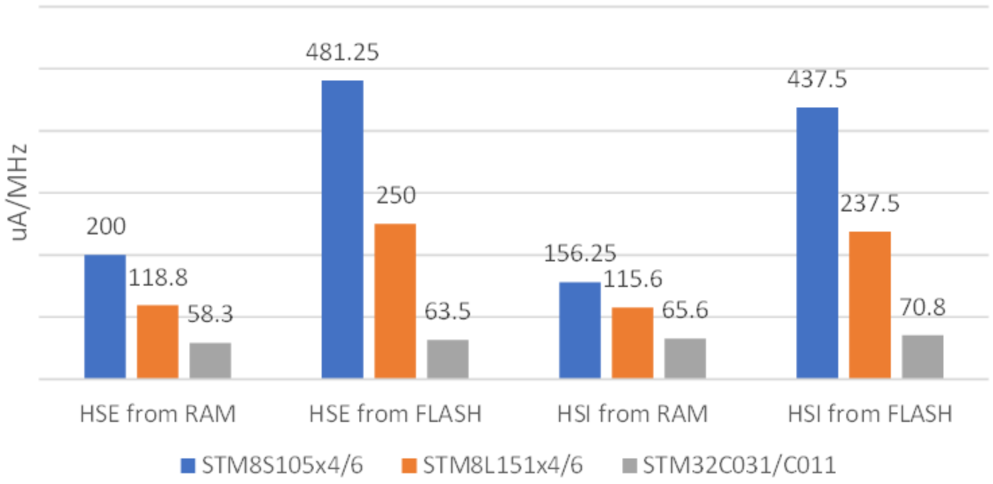
- Sleep
- Stop
- Standby
- Shutdown

The STM32C0 series has lower consumption in most conditions than the STM8L. The different consumption modes are described below.

Dynamic power consumption

Respective to the STM8, the dynamic consumption of the STM32C0 is lower. Regarding the Figure 7, the STM32C0 series can be up to 7.5 times more efficient than the STM8S series, and up to twice as efficient as the STM8L series.

Figure 7. STM32C0 versus STM8 dynamic consumption



Static power consumption

In the STM32C0 series and the STM8 family, the low-power modes have different names. However, the low-power modes have some similarities, so it is possible to compare them.

Table 10. Low-power consumption comparison

Consumption mode		Clock	STM32C011/31	STM8S105C4/6	STM8L151x4/6	Unit
Wait/sleep mode	From flash memory	HSE 16 MHz	0.33	1.55	1.00	mA
	From RAM		0.32	1.55	0.76	
Stop/active halt mode		LSI/LSE	80	200	0.90	µA
Standby/halt mode		All clocks off	7.45	6.50	0.35	µA
Shutdown mode		All clocks off	19	NA	NA	nA

- Sleep mode, corresponds to the wait mode in the STM8, the CPU is clocked off, but other peripherals and the interrupt controller continue to run.
- Stop mode is like the active halt mode. The HSI/HSE clocks are stopped, and the SRAM is retained.
- Standby mode is similar to the halt mode. The HSE/HSI clocks are off. The LSI and LSE clocks can be running if the application uses IWDG, but the main difference is that the RAM is powered off in the STM32C0.
- Shutdown mode has no equivalent in the STM8. It is the ultimate low-power mode. All clocks and peripherals are off.

Wake-up source

Table 11. **Wake-up source comparison**

PWR	STM32C0 series	STM8S series	STM8L series
Low-power modes and wake up sources	Sleep mode • Peripheral event/interrupt • EXTI interrupt/event • NVIC IRQ interrupt • IWDG • Reset Stop mode • Peripheral event/interrupt • EXTI interrupt/event • NVIC IRQ interrupt • IWDG • Reset Standby mode • Wake-up pins • IWDG • Reset Shutdown mode • Wake-up pins • Reset	Wait mode • All internal or external interrupts (including auto wake-up) • Reset • IWDG Active halt mode • Auto wake-up • External interrupts • Reset • IWDG Halt mode • External interrupts • Reset • IWDG	Wait mode • All internal or external interrupts • Wake-up events • Reset • IWDG Low-power run mode • Software sequence • Reset • IWDG Low-power wait mode • Internal or external event • Reset • IWDG Active halt mode • External interrupts • RTC interrupts • Reset • IWDG Halt mode • External interrupts • Reset • IWDG

5.7 Reset and clock controller (RCC) interface

5.7.1 Clocks

Table 12. RCC peripheral STM32C0 series versus STM8S/L series

RCC	STM32C0 series	STM8S series	STM8L series
HSI48	48 MHz high-speed internal RC oscillator	-	-
HSI16	-	16 MHz high-speed internal RC oscillator	16 MHz high-speed internal RC oscillator
LSI	32 kHz low-speed internal RC	128 kHz low-speed internal RC	38 kHz low-speed internal RC
HSE	4 to 48 MHz	1 to 24 MHz	1 to 16 MHz
LSE	32.768 kHz	-	32.768 kHz
System clock source	HSI48, HSE, LSI, LSE	HSI16, HSE, LSI	HSI16, HSE, LSI, LSE
System clock frequency	• Up to 48 MHz • 12 MHz after reset based on HSI	• Up to 24 MHz • 2 MHz after reset based on HSI	• Up to 16 MHz • 2 MHz after reset based on HSI
APB frequency	Up to 24 MHz	-	-
RTC clock source	LSI, LSE, or HSE clocks divided by 32	-	HSI, HSE, LSI, LSE
Clock output	MCO1/2: LSI, LSE, SYSCLK, HSI48, HSE LSCO: LSI, LSE available in stop mode	CCO: HSE, HSI, HSIDIV, LSI, MASTER, CPU	CCO: HSE, HSI, LSI, LSE
Internal oscillator measurement and calibration	Internal/external clock measurement inputs • TIM14 inputs: GPIO, RTC, HSE/32, MCO, MCO2 • TIM16 inputs: GPIO, LSI, LSE, MCO2 • TIM17 inputs: GPIO, HSE/32, MCO, MCO2	-	Internal/external clock measurement inputs • TIM2/3: LSE

Table 13. High-speed and low-speed clock internal accuracy comparison

Clock accuracy	Temperature	STM32C03/C01	STM8S105C6	STM8L101F1
HSI factory calibrated	Full range	-2.5% to 2%	±3%	-4.5% to 3%
	0°C to 85°C	±1%	±2%	-2.5% to 2%
	30°C	-0.83% to 0.2%	±1%	±1%
LSI	Full range	±7%	±20%	-12% to 11%

The STM32C0 has a better clock accuracy than the STM8. It can clock other peripherals with the MCO output. The HSI can be used for USART communication.

5.7.2 Reset

The STM32C0 series has several types of reset:

- Power reset: this sets all registers to their reset values. Exiting Standby mode is an exception. In this case the registers outside the VCORE domain (back up registers, IWDG, Standby/Shutdown mode control) are not impacted.
- System reset: this resets all registers to their reset value, except the reset flags, and the RTC registers.
- RTC domain reset: this only affects the RTC domains (LSE oscillator, RTC and RCC_CSR1 register).

The main difference is the addition of the software reset. It is no longer mandatory to use a trick with the WWDG to emulate a software reset, as in the STM8.

Table 14. **Reset source comparison**

Reset source	STM32C0 series	STM8S series	STM8L series
Power-on Reset/Power-down reset	X[1][2]	X	X
Brown-out reset	X[1]	X	X
Power voltage detection (PVD)	-	-	X
Exit from Standby mode	X[1]	-	-
Exit from Shutdown mode	X[1]	-	-
Low level on the NRST pin	X[2]	X	X
WWDG reset	X[2]	X	X
IWDG reset	X[2]	X	X
Software reset	X[2]	-	-
Low-power mode security reset	X[2]	-	-
Option-byte loader reset	X[2]	-	-
EMC reset	-	X	-
Illegal opcode reset	-	X	X

1. *Power reset*
2. *System reset*

5.8 Nested vectored interrupt controller (NVIC)

STM32C0 devices do not use the same interrupt system as STM8 devices. They use a nested vectored interrupt controller (NVIC). There are some similarities with the STM8 family such as: interrupt vector, priority management, and EXTI. In the STM32C0, each IP has its own vector, so there is no interrupt sharing (as in STM8L151x6/8 STM8L152x6/8).

Table 15. **Interrupt features comparison**

Parameter	STM32C0 series	STM8S series	STM8L series
Interrupt vectors	Up to 32 interrupt vectors (+ 5 system ones)	Up to 32 interrupt vectors	
Interrupt priorities	4 levels lower number = higher priority	3 levels	
Disable interrupts	yes, apart from NMI and HardFault	Yes	
External interrupts	16 external interrupt channels linked to IO lines	5 external interrupts linked to ports	8 external interrupts linked to IO lines + 4 linked to ports
Reset vector	4 bytes (address of the IRQ procedure)	4 bytes (0x82 code + 24bit address of the IRQ procedure)	
Interrupt latency	16 cycles to save context 16 cycles to restore context Tail chaining supported	9 cycles to save context 9 cycles to restore context Tail chaining supported	

The Cortex®-M0+ has six system interrupts (three more than the STM8). The priority of Reset, NMI and HardFault are fixed, in contrast to SVC, PendSV, and SysTick, which are programmable.

Table 16. **System interrupts comparison**

Offset	STM32C0 series	STM8 family
0x00	-	Reset: Address of the application start
0x04	Reset: Address of the application start	TRAP: Software interrupt
0x08	NMI: Non Maskable Interrupt connected to SRAM parity error, HSE and LSE clock security systems (may be slightly different in other STM32 lines)	TLI: Top level interrupt. (It is assigned to various interrupt sources depending on the family, that is, in STM32L15xx8 it is TIM2 and TIM4 overflow IRQ.)
0x0C	HardFault: Reports all issues related to bus/memory accesses	-
0x2C	SVC: System service call, software interrupt. Used by operating systems	-
0x38	PendSV: Pendable request for system service software interrupt. Used by operating systems	-
0x3C	SysTick: Interrupt from built-in 24-bit counter (part of the core), used for delays, timeouts, and operating system timing	-

There are two ways to handle the interrupts with the help of STM32CubeMx: the hardware abstraction layer (HAL), and the low layer (LL). The first one takes longer, due to the high level, but it is easier to implement the interrupt processing flow.

Table 17. **Interrupt handler comparison**

Features	STM32C0 LL library	STM32C0 HAL library	STM8 SPL
Vector table definition	startup_stm32c0xx.s	startup_stm32c0xx.s	stm8_interrupt_vector.c
Interrupt processing flow	startup file with complete IRQ table definition ↓ Interrupt handler in stm32c0xx_it.c	startup file with complete IRQ table definition ↓ Interrupt handler in stm32c0xx_it.c ↓ HAL IRQ handler in stm32c0xx_hal_ppp.c [1] to handle flags and status bits ↓ Final callback overwriting „weak” callback within HAL library	stm8_interrupt_vector.c with complete IRQ table definition ↓ Interrupt handler in stm8xx_it.c

1. *“ppp” = peripheral name (adc, uart, rcc etc.)*

5.9 DMA

The DMA IP is new in the STM32C0 series, compared to the STM8S series and some STM8L lines. It is clearly a major asset to improve the product consumption when it is possible to make a memory transfer without CPU.

Table 18. **DMA peripheral**

Feature	STM32C0 series	STM8S	STM8L
DMA channel	3	-	4 (only on STM8L05xxx/15xxx, STM8L162xx)
DMA controller	DMAMUX: • The trigger for each channel is either a peripheral request, or any of four generated requests	-	Up to 3 requests per channel
Transfer size	Byte, half-word, word	-	Byte, half-word
Transfer type	Peripherals to memory, memory to peripherals, memory to memory, and peripherals to peripherals	-	Peripherals to memory, memory to peripherals and memory to memory
Interrupt request per channel	Transfer complete, half transfer, or transfer error	-	Transfer complete or half transfer
Addressing mode	Incrementing	-	Incrementing and decrementing

5.10 GPIO interface

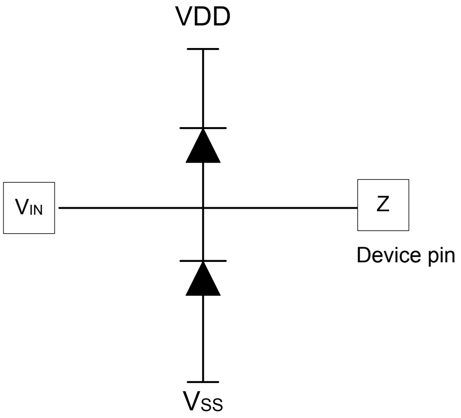
The STM32C0 GPIOs are different to those of the STM8 family.

Some new features are available in STM32C0 Series devices:

- internal pull-down resistor
- output open-drain with pull-up or pull-down capability
- output push-pull with pull-up or pull-down capability
- alternate function push-pull with pull-up or pull-down capability
- alternate function open-drain with pull-up or pull-down capability
- analog input
- V_{IN} is no longer limited by V_{DD} + 0.3 V, but by 5.5 V (see Table 20).

Due to different pin protection architectures, and as for all STM32 devices, positive current injection is not allowed on the STM32C0. (An exception is certain MCUs with switchable diodes.) In fact, there is no clamping diode between the IO and V_{DD} (due to 5 V tolerant capability). If the user application needs to be protected against positive injection, it is necessary to add external clamping diodes (see Figure 8). For further details about the GPIO (FT) and EMC design, refer respectively to the AN4899 and the AN1709 mentioned in the Table 1. Document and website references.

Figure 8. **Clamping diodes protection**



DT57077V2

The STM32C0 Series can share the same pin for Reset or GPIO functionality. One specific pin, PF2, is configured by an appropriate value in the option bytes. In the small package, due to the limited number of pins, multiple GPIOs are connected in the IO.

The user can also freeze the GPIO control register by applying a specific write sequence. Moreover, each pin of the GPIO can be set as an analog input (Schmitt trigger deactivated), in order to reduce the power consumption.

Table 19. **GPIO differences between STM32C0 Series and STM8 family**

Feature	STM32C0 Series	STM8 family
Speeds	(2 bits → 4 SPEED) 3 MHz 15 MHz 60 MHz 80 MHz	1 bit (2 SPEED) 2 MHz 10 MHz
Pull-up/down	YES	Pull-up only

Table 20. **GPIO input voltage comparison**

Voltage	STM32C0 Series	STM8S series	STM8L series
V_{IL}	0 V to 0.3xV_{DD}	-0.3 V to 0.3xV_{DD}	V_{SS}-0.3 V to 0.3xV_{DD}
V_{IH}	0.7xV_{DD} to 5.5 V	0.7xV_{DD} to V_{DD}+0.3	0.7xV_{DD} to 5 V (for 5 V tolerant input)

5.11 RTC

The STM8S series does not have an RTC. However, an RTC is present on the STM8L series.

Table 21. RTC peripheral

Peripheral	Feature	STM32C0 series	STM8L (low-density devices)	STM8L (medium-density devices)	STM8L (medium+ and high-density devices)
RTC	Number of alarms	1	1 (or wake-up signal)		1 (or wake-up signal)
RTC	Number of outputs	2 (RTC calibration + alarm/wake-up signal)	2 (RTC calibration + alarm/wake-up signal)		2 (RTC calibration + alarm/wake-up signal)
Tamper	Number of events	0	0		3

5.12 USART

Table 22. **USART peripheral**

Feature	STM32C0 series	STM8S series	STM8L series
Configurable oversampling method	16 or 8	-	-
RX/TX FIFO	2 x 8 bytes	2 x 1-byte (TDR/ RDR)	2 x 1-byte (TDR/RDR)
Common programmable transmit and receive baud rate	YES	-	-
Programmable data word length	7, 8 or 9 bits	8 or 9 bits	8 or 9 bits
Programmable data order with MSB-first or LSB-first shifting	YES	-	-
SPI slave transmission underrun error flag	YES	-	-
DMA	Continuous communications using DMA Received/transmitted bytes are buffered in reserved SRAM using centralized DMA	-	Configurable multibuffered communication using DMA
Separate signal polarity control for transmission and reception	YES	-	-
Swappable Tx/Rx pin configuration	YES	-	-
Hardware flow control for modem and RS-485 transceiver	YES	-	-
Wake-up from low-power mode	YES	-	-
Modbus	YES	-	-

5.13 I^{2}C

Table 23. **I^{2}C configuration**

Feature	STM32C0 series	STM8S series	STM8L series
Communication speeds	– Standard-mode (up to 100 kHz) – Fast-mode (up to 400 kHz) – Fast-mode plus (up to 1 MHz)	Standard speed (up to 100 kHz Fast speed (up to 400 kHz)	Standard speed (up to 100 kHz Fast speed (up to 400 kHz)
SMBus	3.0	-	2.0
PMBus	1.3	-	YES
DMA capability	1-byte buffer	-	1-byte buffer
Clock selection	PCLK, SYSCLK, HSIKER	-	-

5.14 Flash memory

The STM32C0 series has a maximum frequency of 48 MHz, and the flash memory's maximum frequency is 24 MHz. To compensate the flash memory speed, and to be sure to have valid and uncorrupted data, a wait state feature is added. The wait state feature is not implemented on most STM8 products, because the CPU speed does not go above the flash memory speed.

Table 24. **Flash memory**

Feature	STM32C0 series	STM8S series	STM8L series
Page size	• 2 Kbytes	• 64 bytes (low and medium density) • 128 bytes (high density)	• 64 bytes (low density) • 128 bytes (medium and medium+ density) • 256 bytes (high density)
Data width	64-bit	32-bit	32-bit
Programming granularity	8-byte	4-byte	4-byte
Flash read protection (RDP)	Three levels: • No protection • Read and write protection • No debug	Two levels: • No protection • Read and write protection	Two levels: • No protection • Read and write protection
Flash writes protection area	Two configurable areas (WRP)	One configurable area (UBC)	One configurable area (UBC)
Flash proprietary code readout protection	Two configurable areas (PCROP)	-	One configurable area (ROP)
"On Time" programmable area	1 Kbytes	-	-

Table 25. **Flash memory characteristics comparison**

Parameter	STM32C0x1x6	STM8S105x6	STM8L152x6	Unit
Page size	2 k	128	128	byte
Programing time for one page	21.8	6	6	ms
	10.6	46.9	46.9	µs/byte
Fast programing time for one page	13.7	3	3	ms
	6.7	23.4	23.4	µs/byte
Page erase time	22	3	3	ms
	10.7	23.4	23.4	µs/byte

5.15 SRAM

Table 26. **SRAM density, STM32C0 series versus STM8L/S series**

Maximum flash memory density	STM32C0 series	STM8S series	STM8L series
4 Kbytes	-	STM8S103F2: 1 Kbytes	STM8L151x2: 1 Kbytes
8 Kbytes	-	STM8S103x3/STM8S001J3/ STM8S003x3/STM8S903x3: 1 Kbytes	STM8L151x3: 1 Kbytes
16 Kbytes	STM32C011x4: 6 Kbytes	STM8S105x4: 2 Kbytes	STM8L151x4/STM8L152x4: 2 Kbytes
	STM32C031x4:12 Kbytes		

Maximum flash memory density	STM32C0 series	STM8S series	STM8L series
32 Kbytes	STM32C011x6: 6 Kbytes	STM8S005x6/STM8S105x6: 2 Kbytes	STM8L151x6/STM8L152x6: 2 Kbytes
	STM32C031x6:12 Kbytes	STM8S207x6/STM8S208x6: 6 Kbytes	
64 Kbytes	-	STM8S007x8/STM8S207x8/ STM8S208x8: 6 Kbytes	STM8L151x8/STM8L152x8: 4 Kbytes
128 Kbytes	-	STM8S207xB/STM8S208xB: 6 Kbytes	-

5.16 Timers

Table 27. **Timers available in STM32C0 series MCUs**

Timer type	Timer	Counter resolution	Counter type	Prescaler factor	DMA request generation	Capture/ compare channels	Complementary outputs
Advanced control	TIM1	16-bit	Up, down, up/down	integer from 1 to 65536	YES	4 (6 internals)	3
General purpose	TIM3	16-bit	Up, down, up/down	integer from 1 to 65536	YES	4	-
	TIM14	16-bit	Up	integer from 1 to 65536	-	1	-
	TIM16 TIM17	16-bit	Up	integer from 1 to 65536	YES	1	1
Systick	STK	24-bit	Down	HCLK/8	-	-	-

All timers in the STM32C0 are 16-bit. The maximum clock frequency is now 48 MHz. There is one 24-bit timer inside the Cortex®-M0+ core, which is generally used as a 1 ms time base.

The new functions are listed below:

- Advanced timer (TIM1):
  - 3 more independent channels
  - 1 more break input
  - Asymmetric, combined, combined 3-phase PWM
  - Bidirectional break inputs
  - UIF bit remapping
- General purpose timers (TIM3/14/16/17)

In STM32 MCUs, 8-bit timers are not present.

5.17 ADC

Table 28. **ADC differences between STM32C0 series and STM8L/S series**

Feature	STM32C0 series	STM8S series	STM8L series
Resolution	12-bit, 10-bit, 8-bit, or 6-bit configurable	10 bits	12-bit, 10-bit, 8-bit, or 6-bit configurable
Conversion time	Up to 0.4 µs (2.5 Msps)	Up to 2.33 µs (0.43 Msps)	Up to 1 µs (1 Msps)
Self-calibration	YES	-	-
Programmable sampling time	YES	-	YES
DMA support	YES	NA	YES
Oversampling	YES	-	-
Number of external channels	Up to 19	Up to 16	Up to 28
Internal channel	• V_{sense} (temp sensor) • V_{refint} • V_{DDA} • V_{SSA}	-	• V_{temp_sensor} • V_{refint}

5.18 SPI/I^{2}S

Table 29. **SPI comparison**

Feature	STM32C0 series	STM8S series	STM8L series
Half-duplex synchronous transfer on two lines (with bidirectional data line)	YES	-	-
Data size selection	4 to 16-bit data size selection	Only 8-bit	Only 8-bit
Multimaster mode capability	YES	-	-
Faster communication - maximum SPI speed	12 MHz	10 MHz	8 MHz
SPI Motorola support	YES	-	-
DMA capability	Two 32-bit embedded Rx and Tx FIFOs	-	1-byte transmission and reception buffer
Enhanced TI and NSS pulse modes support	YES	-	-

STM32C0 devices no longer have a beep output (sound generated). However, they have an I^{2}S IP, so it is possible to connect the STM32C0 to an audio interphase.

5.19 Independent watchdog (IWDG)

In STM32C0 series devices, the independent watchdog (IWDG) can be driven in two ways:

- Without the window option activated, the IWDG works in the same way as that in the STM8. The counter value is reloaded when the key is written in IWDG_KR. The chip is reset when the down counter value becomes lower than 0x000.
- With the window option activated, the counter value can be reloaded in two ways. The first is the same methodology as the STM8, by writing a special key in IWDG_KR. The second, new, way is to refresh the counter value, with a value written in the window register. This new feature adds a new conditional reset, in addition to the one previously described. The circuit is reset if the down counter is reloaded outside the window.

Table 30. **IWDG comparison**

Feature	STM32C0 series	STM8S series	STM8L series
Clock source	LSI (32 kHz)	LSI/2 (64 kHz)	38 kHz
Down counter size	12-bit	8-bit	8-bit
Window option	Yes	-	-
Minimum time out period	125 µs	62.5 µs	110 µs
Maximum time out period	32.76 s	1.07 s	1.72 s
Debug mode (suspend the IWDG when the core is halted)	Yes	-	-
Freeze IWDG in low power mode	Yes (STOP and STANDBY)	-	Yes (HALT)

5.20 System window watchdog (WWDG)

Table 31. **WWDG comparison**

Feature	STM32C0 series	STM8S series	STM8L series
Clock source	HSI, HSE, LSI, LSE	HSI, HSE	HSI, HSE
Static prescaler	4096	12288	12288
Variable prescaler	1-128	-	-
Minimum time out period for F_{WWDG}=16 MHz	0.512 ms	0.768 ms	0.768 ms
Maximum time out period for F_{WWDG}=16 MHz	2097.152 ms	49.152 ms	49.152 ms
WWDG interrupt	Yes	-	-
Debug mode (suspend the WWDG when the core is halted)	Yes	-	-

5.21 Option and engineering bytes

For the option bytes, the STM32C0 series and the STM8 family share the same methodology. However, the implementation is different.

In the STM8 family, there is no special protocol to program the option bytes. It is done on-the-fly by the application, or through the SWIM interface by accessing the EEPROM address.

Conversely, for the STM32C0 series, it is no longer possible to write the option bytes directly to the flash memory address. There is a dedicated programming protocol with a locking mechanism to protect the option bytes from unwanted writes.

Table 32. **Option bytes comparison**

Feature	STM32C0 series	STM8S series	STM8L series
Register size	64 bits (32-bit option byte + 32-bit complemented option byte)	16-bit (8-bit option byte + 8-bit complemented option byte)	8-bit
Register location	Flash memory	EEPROM memory	
Peripheral configuration	• NRST pin, reset holder, BOR, and low power mode entry protection • Boot configuration • Multiple bonding • Clock remapping • Watchdog selection and freeze option • Flash protection (RDP, PCROP, WRP, SEC)	• Alternate function remapping • Bootloader option byte • Watchdog selection and freeze option • Clock • Flash protection (ROP, UBC)	• BOR • Bootloader option byte • Watchdog selection and freeze option • Clock • Flash protection (ROP, UBC, PCODESIZE)

In addition to the option bytes, the user can find the engineering bytes on the STM32C0. They contain some useful information written during the production test, such as:

- Unique device ID
- Flash size
- Package type
- Calibration value of internal voltage reference and temperature sensor

6 Getting started with STM32C0Cube

Due to the huge difference between the STM8 proprietary core and the Cortex®-M0+, the software is not portable between the STM8 family and STM32C0 series devices. The user needs to rewrite the code. Nevertheless, some useful software and libraries are available to simplify the transfer.

6.1 Initialization code from STM32CubeMX

The STM32CubeMX helps the user to generate the MCU initialization functions. A graphical interface and a different menu help to configure the STM32C0 as needed. This facilitates starting from a healthy working environment.

By default, STM32CubeMX generates initialization code based on the HAL, but this could be modified to generate code based on an LL driver.

Then, the user just needs to integrate their own functions to get the desired behavior. If the chip does not correspond to the desired configuration, it is possible to change it in the tool, and regenerate the code. This can be done without deleting the user functions.

Compared to the SPL utilization, the STM32CubeMX performs the configuration modification for the customer. It no longer needs to add or delete any libraries.

6.2 Migration

The standard tools used when developing the STM8 family are:

1. ST Visual Develop IDE with Cosmic compiler
2. IAR Embedded Workbench® for STM8

The first tool is selected as the starting point. The paragraphs below explain how to get started with STM32C0 series devices, and help to understand the system behavior. For this, they detail, step-by-step, the migration of a simple application from the STM8S105C6 to the STM32C031C6. The goal is to fill a memory buffer by A-D conversion with a timer trigger, while using sleep/wait mode.

Getting the workspace

There are three main IDEs for STM32C0 series devices, and for the STM32 family in general. Two are fully free of charge, such as STM32CubeIDE or µVision® from Keil® (only for M0+ core). The third needs a subscription or a free-of-charge part with code limitation, such as IAR Embedded Workbench®. All these IDEs include the essential features, such as compiler, STLINK, and a driver that permits programming and debug.

When the toolchain is installed and ready to use, the user must download the various needed libraries. If the user chooses to use STM32CubeMX, the tool downloads the latest available library. Otherwise, it is possible to download these libraries from the www.st.com website.

The user can find many examples to begin their application development easily, or to learn the different ways to use the MCU, both in HAL and LL.

To start developing an application easily, a preconfiguration is also available for the STM32 board (Nucleo, Discovery, and so on).

Starting configuration

By default, the STM32C0 device boots in flash memory as soon as the code is flashed. It is possible that the application needs to use the bore on reset, memory protection, or modify the system reset. In such cases, the option byte must be correctly configured for the required function when the product is powered on. To do this, installing the STM32CubeProg tool is recommended.

The first-level system initialization is done in the SystemInit() function. It is located in *system_stm32c0xx.c*. After a reset, HSI 48 MHz is selected by default with division by four. Hence, the system clock starts at 12 MHz. It is easy to modify the clock configuration in STM32CubeMX, or in the SystemClock_Config() function.

Programming part

To develop code for the STM8 family, it is advisable to use the standard peripheral library (SPL). This gives the base function to configure and use the MCU peripherals. Moreover, STMicroelectronics offers some useful examples to help users to develop their application. Despite the use of libraries, the SPL is close to the register-level programming.

Also, the STM32 series uses the hardware abstraction level (HAL) and the low level (LL) libraries, which are an evolution of the SPL.

The HAL allows the user to develop an application quickly and port it to the entire STM32 portfolio. The downside is that it is not optimized in terms of code size and execution time. However, this library is perfect for discovering the functionalities of the STM32C0 series.

To compensate for the size of the HAL library, the customer can use the LL, which is closer to the SPL.

The lower level is the best compromise between software development time and code size.

If the code size is a problem, the user can also program the software at register level. This is the best way to optimize the code size. However, it makes development more time consuming.

Table 33 gives the user a comparative idea of the different abstraction levels, in terms of code size, and development time.

Table 33. **Abstraction level programming**

Abstraction level	Development time	Code size
HAL	+	+++
LL	++	++
Register level	+++	+

In addition to the Table 8, a comparison based on a classic use case has been made. This uses the different available software libraries (standard peripheral library for the STM8 family, and the LL and HAL for the STM32C0 series). The goal is a more precise comparison of the code size between STM8S, STM8L, and STM32C0. This is based on a simple application using ADC, TIM, USART, and DMA (if available).

Table 34. **Use case ADC code comparison**

Product	Libraries	Speed	Size	Balance	Medium	Low	None	Unit
STM32C0	LL	10090	9462	10082	11158	11728	11862	Byte
	HAL	15782	15666	15742	15898	17162	17432	
STM8L	SPL	8567	8454	8454	8490	9010	9192	
STM8S	SPL	10839	10461	10471	10859	11417	11454	

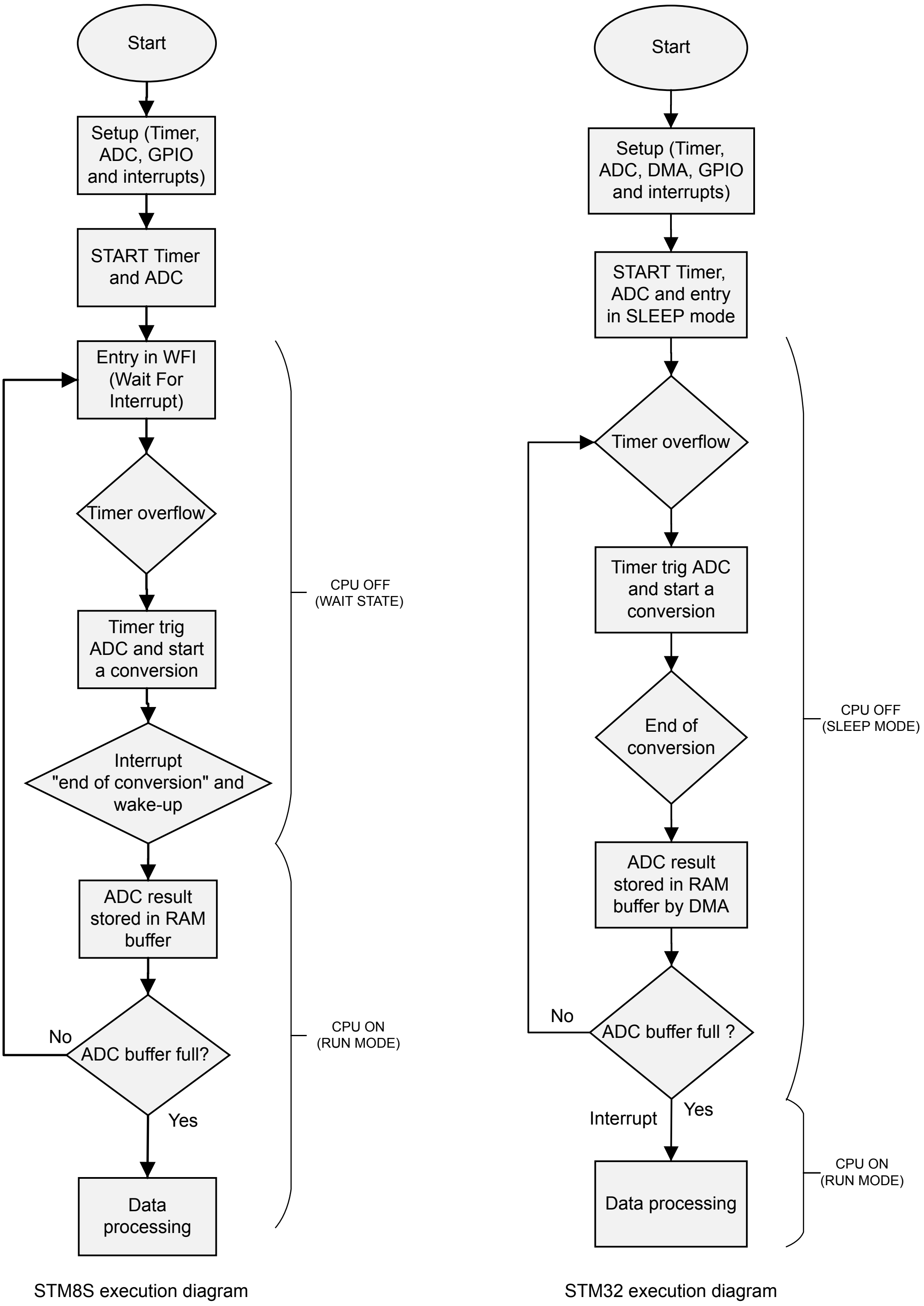
The data in Table 34 supports Table 33. The HAL takes 50% more space in the memory compared to the LL. Due to the limited flash memory size of the STM32C0, using the LL library as early as possible is recommended.

It also shows the same gap between the STM8 and the STM32 as seen in the CoreMark part. It is less visible for the STM8S due to the bad library optimization.

Execution

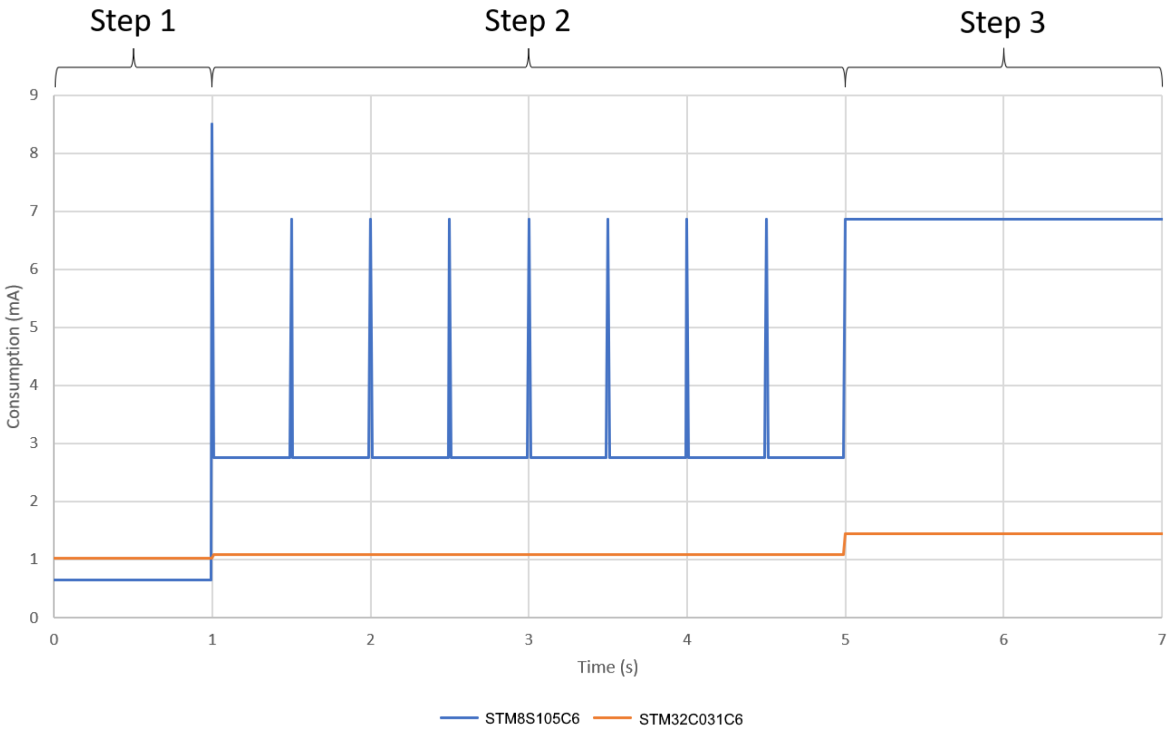
One of the main advantages of the STM32C0 compared to the STM8S is the possibility to use different peripherals without the CPU. This optimizes the current consumption and therefore the battery lifespan. For example, the use of DMA permits data transfer between the ADC and the RAM memory without the CPU. The STM32C0 stays in sleep mode during the whole operation, whereas the STM8S needs to wake up from wait state to save the data in the buffer. The flow chart below represents the hardware execution of the example.

Figure 9. Example code diagram


Start
Setup (Timer, ADC, GPIO and interrupts)
START Timer and ADC
Entry in WFI (Wait For Interrupt)
Timer overflow
Timer trig ADC and start a conversion
Interrupt "end of conversion" and wake-up
ADC result stored in RAM buffer
ADC buffer full?
No
Yes
Data processing
CPU OFF (WAIT STATE)
CPU ON (RUN MODE)
STM8S execution diagram
Start
Setup (Timer, ADC, DMA, GPIO and interrupts)
START Timer, ADC and entry in SLEEP mode
Timer overflow
Timer trig ADC and start a conversion
End of conversion
ADC result stored in RAM buffer by DMA
ADC buffer full ?
No
Interrupt
Yes
Data processing
CPU OFF (SLEEP MODE)
CPU ON (RUN MODE)
STM32 execution diagram

In each case, the HSI clock is used as the clock source. It is clocked at 12 MHz for the STM32C0, and 16 MHz for the STM8S.

Figure 10. **Consumption mode and timeline behavior**



- Step 1: Both products are under reset
- Step 2: The STM32C031C6 stays in sleep mode during the whole operation (ADC conversion and DMA transfer).
  While the STM8S105C6 makes a wait-run-wait transition to wake up after an ADC conversion to fill the RAM buffer. That is why there are eight spikes.
- Step 3: This is the final step, when the buffer is full, both MCUs treat the value in run mode.

As shown in Figure 10, the STM32C0 is more efficient than the STM8S, whether in run mode, or low-power mode.

7 Ecosystem

Compared to the STM8 family, the STM32 series offers a large choice of different software. This helps the user to program the application with the new STM32CubeMX configuration tool, or the many available IDEs.

The next sections describe how the developer can choose and set up the software part to start to use the STM32C0 series.

7.1 Compilers Cosmic, IAR™ for STM8 family versus Keil® IAR™, and STM32CubeIDE for STM32 series

To develop code for the STM8 family, there are two main possibilities:

- STVD and Cosmic compiler: free of charge
- IAR Embedded Workbench® for STM8: paid license needed

For the STM32C0, there is wider choice:

- Keil® IDE by Arm®, using the MDK-ARM compiler: free of charge
- STM32CubeIDE with GCC compile: free of charge
- IAR Embedded Workbench® for STM32 with EWARM toolchain: a paid license or free limited version (32-Kbyte code size limitation)

All these IDEs and compilers are compatible with STM32CubeMX.

7.2 STM8CubeMX versus STM32CubeMX

In the migration example, the STM8CubeMX is not mentioned because this tool does not have code-generation capability (initialization code for the RCC, GPIOs, and IPs). This tool only gives the customer an idea about STM8 usage with the GPIO distribution, the clock configuration, or even an estimation of the product consumption. It can also help the designer to build the layout.

All these functionalities are found in the STM32CubeMX, but with a direct interaction with the code. Programmers need to be careful to write their own function in the associated boxes:

```
/* USER CODE BEGIN 1   */
/*USER CODE END 1 */
```

7.3 STVP and FLASHER-STM8 versus STM32CubeProgrammer

There are two ways to program the STM8. These are part of the programming tool integrated into IDEs:

- STVP (ST Visual Programmer) using the debug pin (SWIM) it is possible to use:
  - S19 and HEX format
  - erase, program, view, and verify the device memory
  - project mode, to automate the configuration and programming tasks
- FLASHER-STM8. This software can program and communicate with the STM8 system bootloader through the RS232 interface.

Compared to the STM8, STM32 MCUs use STM32CubeProgrammer. This merges all the functionalities described above. It uses the SWD/JTAG debug interface (only SWD is available on STM32C0 series devices), or the system bootloader. Moreover, it offers new functionalities:

- ELF and binary format
- In addition to USART, it is possible to use USB DFU/I^{2}C/SPI/CAN bootloader interface. (Only I^{2}C and USART is available on STM32C0 series devices.)
- Command-line interface for automation through scripting

7.4 STM32C0 hardware available

The STM32 inherits the board methodology developed for the STM8 family, and STM32 series devices. There are two board families that help to learn on the product, and to develop first prototypes quickly.

1. Nucleo 64 boards:
   These are the mainstream board. They allow the user to learn about, and evaluate, the STM32C0 features. They use a simple PCB that is common to all Nucleo 64 boards. This includes an STLINK for chip debugging, and also to provide an RX-TX link between the computer and the MCU.
   Moreover, two buttons (one user button and reset) and two LEDs (user LED and power-up LED) are present.

   It is also possible to use some add-ons that are compatible with the ARDUINO® Uno and ST morpho connector.
2. Discovery boards:
   The discovery boards are cheaper than the Nucleo boards. They are relatively simple pieces of hardware, to test the key features of the product.

   There are three add-on connectors: STMod+, DIP28 ARDUINO® compatible pinout, and a Bluetooth® connector. However, it is necessary to add an STLINK (for example the MB1762A board).

Revision history

Table 35. Document revision history

Date	Version	Changes
12-Apr-2022	1	Initial release.
19-Sep-2022	2	Updated: • Section 4 Boot mode selection, Section 5.2 System architecture, Section 5.7.1 Clocks, Section 5.10 GPIO interface, Section 5.16 Timers, Section 6 Getting started with STM32C0Cube, Section 6.2 Migration, Section 7.1 Compilers Cosmic, IAR™ for STM8 family versus Keil® IAR™, and STM32CubeIDE for STM32 series • Table 6, Table 9, Table 18, Table 21, Table 24. Flash memory , Table 27. Timers available in STM32C0 series MCUs, Table 28. ADC differences between STM32C0 series and STM8L/S series Added: Section 5.7.1 Clocks, Section 5.19 Independent watchdog (IWDG), Section 5.20 System window watchdog (WWDG), Section 5.21 Option and engineering bytes, Section 7.3 STVP and FLASHER-STM8 versus STM32CubeProgrammer
20-Dec-2022	3	Updated : • Programming part • Figure 10. Consumption mode and timeline behavior • Figure 8. Clamping diodes protection • Generated a public version of the document.
21-Dec-22	3	Updated the title and information about the STM8 series (STM8L/S series).

Contents

List of tables

List of figures