AN2857
Application note

STM8S and STM8A family power management

Introduction

This application note is intended for system designers who require a hardware implementation overview of the low-power modes of the STM8S and STM8A product families. It shows how to use the STM8S and STM8A devices in these modes, describes how to take power consumption and wakeup time measurements, and gives results for such measurements.

Example firmware is provided with this application note for implementing and measuring the consumption and wakeup time of the different STM8S and STM8A functioning modes.

Contents

List of tables

List of figures

1 Power consumption factors

In complementary metal oxide semiconductor (CMOS) digital logic devices, power consumption is a sum of:

- Static power (caused mainly by transistor polarization and leakage)
- Dynamic power which depends on the supply voltage and the clock frequency through the formula: Dynamic power = C V^2 f, where:
  - C is the CMOS load capacitance
  - V is the supply voltage
  - f is the clock frequency.

Static consumption is negligible compared to dynamic consumption when the clock is running. In some low-power modes, when no clock is running, static consumption is the main consumption source.

Power consumption thus depends on:

- **Microcontroller unit (MCU) chip size:** Technology used, number of transistors, analog features/peripherals embedded and used.
- **MCU supply voltage**: The amount of current used in CMOS logic is directly proportional to the voltage of the power supply squared . Thus, power consumption may be reduced by lowering the MCU supply voltage.
- **Clock frequency**: Power consumption may be reduced by decreasing the clock frequency when fast processing is not required by the application.
- **Number of active peripherals or MCU features used** (CSS, BOR, PVD,...): The greater the number of active peripherals or features, the greater the power consumed.
- **Operating mode**: Power consumption varies depending on the mode a particular application is running in (CPU on/off, oscillator on/off,...).

For an application powered by a battery, consumption is very important. Usually, average consumption should be below a certain target to ensure an optimum battery lifetime. This means that an application can consume more for short periods of time and keep its average current consumption below the target.

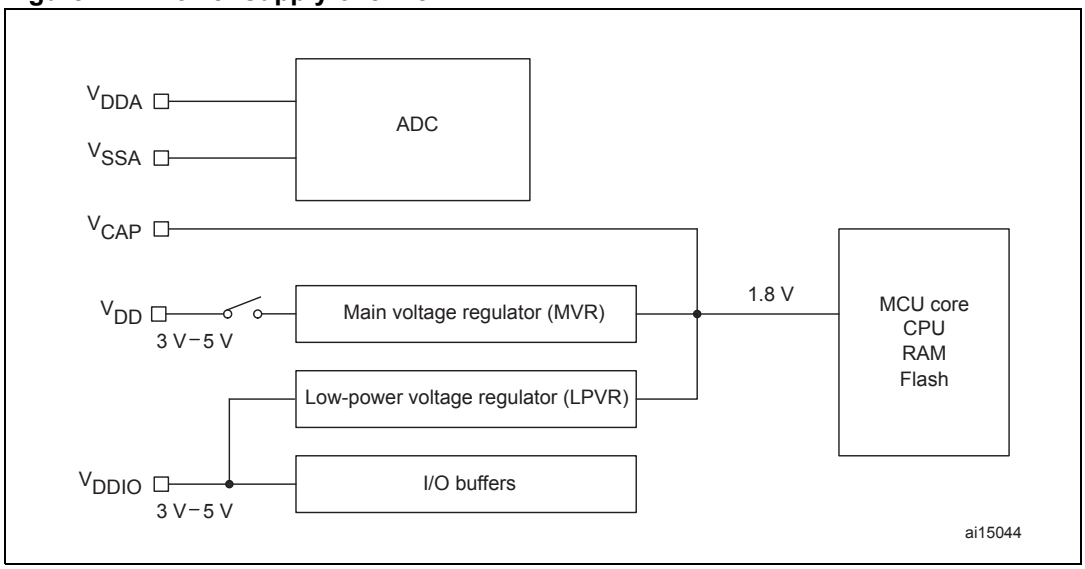
2 Power supply

2.1 Internal supply structure

STM8S and STM8A devices require a 3 V to 5.5 V operating voltage supply (V_{DD}). Two embedded regulators, the main voltage regulator (MVR) and the low-power voltage regulator (LPVR), are used to provide 1.8 V supply to the internal digital parts, depending on the functioning mode (see *Figure 1*).

The V_{CAP} pin is used to decouple the internal 1.8 V supply. A capacitor of minimum 470 nF has to be connected between this pin and the ground. The 1.8 V should never be provided by an external voltage regulator through this pin. Refer to the STM8S and STM8A reference manual (RM0016) which is available on *st.com* for more details.

Figure 1. Power supply overview



2.2 Analog supply

The analog-to-digital converter (ADC) of STM8S and STM8A devices is powered by an independent power supply. The digital and analog power supply have to be properly decoupled. Refer to the STM8S and STM8A reference manual for more details about decoupling capacitors.

2.3 IO supply

The IOs have dedicated pins for power supply which have to be properly decoupled with recommended capacitors. Refer to the STM8S and STM8A reference manual for more details about decoupling capacitors.

2.4 Voltage regulator

After reset, the MVR provides the 1.8 V to the internal digital parts of the microcontroller. Depending on the functioning mode, the MVR can be switched off at which time the LPVR provides the 1.8 V. For example:

- In run and wait mode, only the MVR provides the 1.8 V. The LPVR cannot be used in run mode.
- In active halt mode, during the halt phase, either the MVR or the LPVR can provide the 1.8 V. The user can select which regulator to be used.
- In halt mode, the LPVR is automatically used. The MVR cannot be used in halt mode.

3 Clock management

3.1 Clock system overview

Four different clock sources can be used to drive the master clock:

- 1-24 MHz high speed external clock from the crystal oscillator (HSE crystal)
- Up to 24 MHz high speed external clock provided by the user (HSE user-external)
- 16 MHz high speed internal RC oscillator (HSI)
- 128 kHz low speed internal RC oscillator (LSI)

Each peripheral clock source can be switched on or off independently when it is not used, to optimize power consumption. This is done by using the peripheral clock gating (PCG) feature. See the 'clock control' sections of the STM8S and STM8A reference manual for more details.

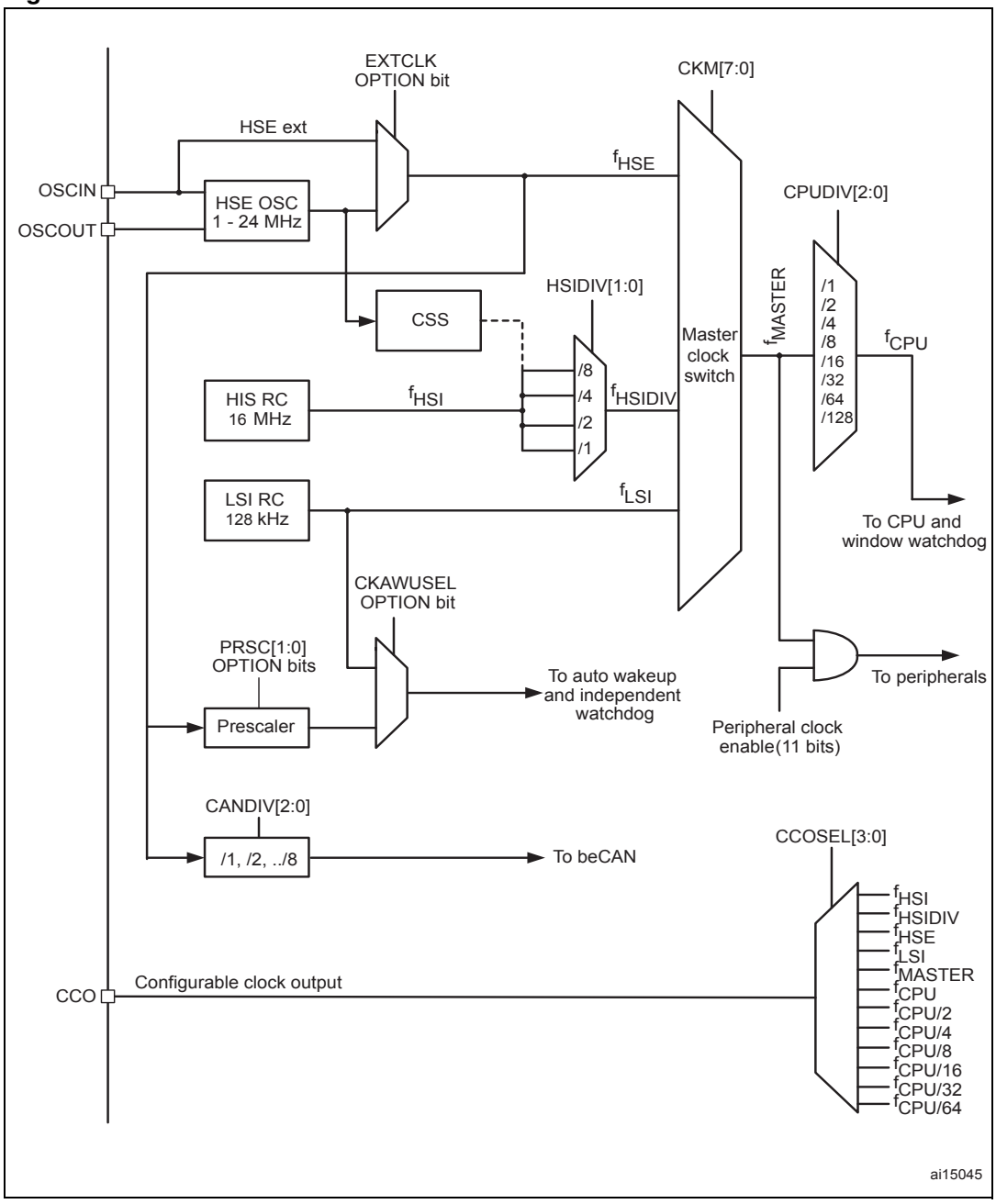
Table 1 summarizes the main features of each clock source. STM8S and STM8A devices offer a complete range of clock sources to fit customer application requirements in terms of cost, accuracy and consumption.

Table 1. Clock source comparison

	HSE crystal	HSE external	HSI	LSI
Cost	Low to medium	Free to high	Free	Free
Accuracy	Crystal	External source	±1 %(1)	±2.5 %(1)
Consumption(1)	High	Medium	Low	Very low
Other information	Resonator or crystal	Existing clock to very complex clock systems	-	-

1. Factory calibrated by STMicroelectronics at T_A = 25 °C.

Figure 2. Clock tree



The read-only CLK_CMSR register contains the current selection of master clock sources (for core and peripherals). Selection of the 'next' master clock is made through the writable CLK_SWR register, the content of which is copied into the CLK_CMSR register once the change is effective (see details below). Refer to *Table 2* for clock selection values.

The default clock after reset is HSI/8. The user can then switch the clock to different frequencies and sources by:

- Choosing another prescaler (/1, /2, /4 or /8) for an internal RC 16 MHz (HSI) clock through the HSIDIV[1:0] bits in the CLK_CKDIVR register.
- Changing the clock master (to HSE or LSI). Refer to the STM8S and STM8A reference manual for more details about the clock switching mechanism.

Before switching off a previous clock source when using automatic switching mode, the user must ensure that the core is no longer running on the current clock. This means that the previous clock must be turned off after the SWIF flag has been set. If the user tries to switch off the clock and the SWIF flag has not been set by hardware, the current clock is not switched off as the microcontroller is still running on it.

Such clock switching can be combined with wait mode (for example, if the HSE crystal is the new clock) as clock switch execution interrupt can wake up the MCU from wait mode. This allows the MCU wakeup to be synchronized with the new clock availability.

The clock switching feature can also be used at the beginning and end of a regular or interrupt routine to speed it up (for instance, if the clock master is LSI RC, but, some parts of the code have to be executed quickly with HSI RC).

Note: When the device is running on an external clock with a frequency above 16 MHz, the wait state bit in the option bytes has to be set. This adds a wait state to the Flash memory access.

Table 2. Clock selection table

Register value	Clock master source (CLK_CMSR)	Next clock master (CLK_SWR)
E1h	HSI	HSI
D2h	LSI	LSI
B4h	HSE	HSE
Other	Reset	Current clock master kept

3.2 Clock configuration and power management

In addition to the flexibility of the clock sources, different complementary clock configurations and features are available to optimize power consumption of the device:

- PCG: Each peripheral clock can be switched off through the CLK_PCKENRx registers.
- CPU clock divider from 1 to 128 (CPUDIV[0:2] bits from the CLK_CKDIVR register): The CPU frequency can be decreased and not the frequency of the peripherals.

4 Run and low-power modes

By default, after a reset, the microcontroller is in run mode. The default CPU clock is a 16 MHz HSI divided by eight due to the HSIDIV register reset value.

Several low-power modes are available to save power when it is not needed to keep the CPU running, for example, when waiting for an external event. It is up to the user to select the mode that gives the best compromise between low power consumption, short startup time, good peripheral functionality, and availability of wakeup sources.

STM8S and STM8A devices feature three main low-power modes:

- Wait mode (CPU stopped, peripherals kept running)
- Active halt mode (CPU stopped, AWU (auto wakeup) and IWDG (independent watchdog) kept running if activated).
- Halt mode (everything is stopped)

Power consumption in run and wait mode can also be reduced by one of the following means:

- Slowing down the system clocks
- Gating the clocks to the peripherals when they are not used

Table 3 summarizes the functioning modes of STM8S and STM8A devices. It gives an overview of the mode combinations which can be used to fit an application's requirements in terms of consumption and wakeup time.

Table 3. Functioning modes

Mode	Voltage regulator	Flash	Oscillators	CPU	Peripherals	Entry	Wake up trigger event
Run	MVR	On	On	On	On	-	-
Wait	MVR	On	On	Off	On	Execute WFI(1) instruction	All internal or external interrupts and reset
Active halt	MVR	On	Off except LSI or HSE	Off	AWU and IWDG on (if activated)	Enable the AWU then execute HALT instruction	AWU or external interrupts and reset
		Off					
	LPVR	On	Off except LSI or HSE				
		Off					
Halt	LPVR	On	Off	Off	Off	Execute HALT instruction	External interrupts and reset
		Off					

1. WFI = Wait for interrupt

Note: When the LPVR is used, the MVR is automatically switched off.

4.1 Run mode

Run mode is the default functioning mode of STM8S and STM8A devices. This mode is used for normal operations and consumes the most power. Use of PCG and low speed clock sources reduce power consumption of the device in run mode.

Clock frequency can be slowed down in several ways. For example, the LSI clock can be used as the f_{MASTER} clock. To enable the MCU to run on the LSI clock, the LSI_EN option bit must be set. Refer to the STM8S and STM8A datasheets (which are available on *st.com)* for more details about the option byte configurations.

4.2 Wait mode

Wait mode, or more exactly wait for interrupt mode, is designed to reduce STM8S and STM8A device power consumption by switching off the core when it is not used. Wait mode is mainly used when the STM8S or STM8A device is waiting for an external or internal event which allows the program to continue its execution. Instead of waiting for the event in run mode, the device can be switched to wait mode. This mode can be used with PCG and with a low speed clock source to further reduce power consumption of the device.

4.2.1 Entering wait mode

Wait mode is entered by executing the WFI assembly instruction. This stops the CPU, but, allows other peripherals and the interrupt controller to continue to run.

When entering wait mode, the interrupts are automatically enabled.

4.2.2 Exiting wait mode

When an internal or external interrupt request occurs, the CPU wakes up from wait mode and resumes processing. This mode offers the lowest wakeup time.

Examples of peripherals or features with interrupts having exit-from-wait capability include:

- I^2C
- USART
- SPI
- CAN
- ADC
- AWU
- External interrupt
- Timers
- Clock controller (clock switch execution)

Refer to the STM8S and STM8A reference manual for more details on the functioning of the above peripherals and features. Refer to the STM8S and STM8A datasheets for availability of above peripherals and features on particular devices.

4.2.3 Activation level/low-power mode control

In a very low-power application, the MCU spends most of the time in WFI/halt mode and is woken up (through interrupts) at specific moments to execute particular tasks. Some of these tasks are recurring and short enough to be treated directly in an interrupt service routine (ISR), rather than returning to the main program. In this case, setting the AL bit before going to wait mode, ensures the run time/ISR execution is reduced because the 'context' (core register content) is not saved/restored each time.

In very simple applications, all operations can be executed in ISR only. In more complex applications, an interrupt routine may relaunch the main program by resetting the AL bit.

The activation level/low-power mode control feature works only with wait mode. It is not available for active halt or halt mode.

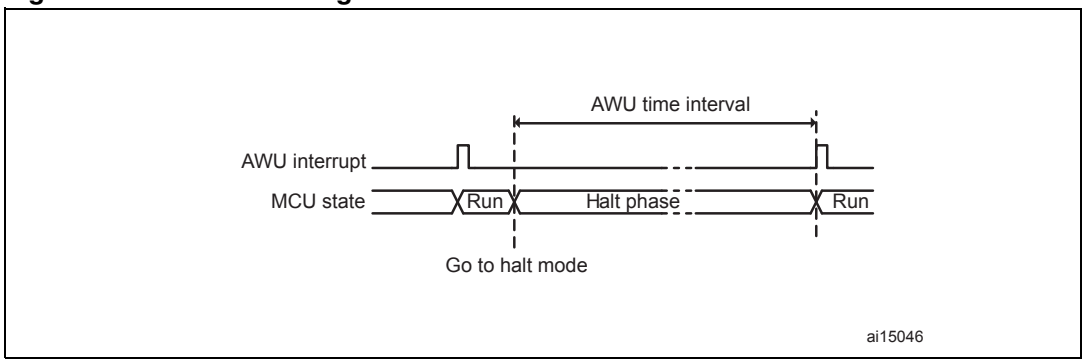
4.3 Active halt mode

Active halt mode can be defined as a 'hybrid' mode between run and halt mode. It is composed of two phases:

- Halt phase: In this phase, the MCU is in halt mode except that the AWU unit and the LSI clock are kept running if they are used as the AWU clock source.
- Active phase: In this phase, the MCU is in run mode.

When entering active halt mode, the AWU counters start to run. The AWU interrupt wakes up the CPU at regular programmed intervals. Once the device is in run mode the AWU counters are stopped.

Figure 3. Active halt diagram



In active halt mode during halt phase, the user can choose the regulator (either MVR or LPVR) and the Flash state (either operating or power-down mode). Using the LPVR and putting the Flash in power-down mode reduces power consumption, but, increases wakeup time.

Active halt mode is very useful for reducing average consumption of a battery based application.

Table 4. Active halt mode configuration

Voltage regulator	Flash state	Power consumption(1)	Wakeup time(1)
MVR	On	****	*
	Off	***	**
LPVR	On	**	***
	Off	*	****

1. The greater the number of stars, the greater the power consumption or the wakeup time.

4.3.1 Entering active halt mode

To enter active halt mode, configure and enable the AWU. Then execute the HALT instruction.

Note: If the IWDG is enabled before the HALT instruction is executed, the device does not switch to halt mode, but, switches to the halt phase of active halt mode. In this case, if the AWU is not enabled, the MCU does not wake up automatically. The MCU wakes up by an IWDG reset or external reset.

4.3.2 Exiting active halt mode

Each time an AWU event occurs, the MCU goes into run mode. So, the AWU is used as a wakeup source. However, during active halt mode, the MCU can be woken up by any halt mode wakeup source (see *Section 4.4: Halt mode* for more details about wake-up sources).

Fast clock wakeup

A fast wakeup time is very important in active halt mode. It supplements the effect of the CPU processing performance by helping to minimize the time the MCU stays in run mode between two periods in low-power mode. Fast wakeup time thus reduces overall average power consumption.

After a wakeup event, device startup is on the clock selected before entering active halt mode. The longest wakeup time is obtained when the clock is an HSE crystal. This is mainly due to the oscillator stabilization time. To reduce this wakeup time, STM8S and STM8A devices offer a feature called ‘fast clock wakeup’ which allows the device to start automatically on an HSI clock after a wakeup event. The user can decide to switch back to the former clock or stay on the HSI clock.

By default, the fast clock wakeup feature is disabled. To enable it, the FHWU bit of the internal clock register (CLK_ICKR) must be set before entering low-power mode.

4.3.3 Voltage regulator and Flash configuration during halt phase

When entering active halt mode, the MVR is the default voltage regulator and the Flash is in operating mode.

To use the LPVR, the SWUAH bit of the internal clock register (CLK_ICKR) has to be set. To put the Flash in power-down mode, the AHALT bit of Flash control register 1 (FLASH_CR1) must be set.

4.3.4 AWU unit

The AWU unit is the heart of active halt mode. It generates regular time-spaced interrupts used for waking up the MCU from halt mode. Two clock sources can feed the AWU:

- LSI clock
- HSE crystal divided by a prescaler (PRESC) and selected by the option bytes to give an input clock of about 128 kHz.

The AWU counters only run in halt phase of active halt mode. Once a wakeup event occurs, the device wakes up and the counters are stopped. It is impossible to keep the AWU counters running in run mode. When the device enters halt phase again, the counters restart from zero.

The AWU delivers regular time-spaced interrupts in the range 15.625 µs to 30.720 s (for an AWU input clock of about 128 kHz).

The average power consumption of the device (I_{TOT}) in active halt mode can be estimated using *Equation 1*.

Equation 1

$$I_{TOT} = [(t_{RUN} / (t_{RUN} + t_{AWU}) \times I_{RUN}] + [(t_{AWU} / (t_{RUN} + t_{AWU}) \times I_{AHALT}]$$

t_{AWU} is the duration of halt phase of active halt mode. It is the time base of the AWU.

I_{RUN} is the consumption in run mode which depends on many factors. Good approximations of this parameter are given in *Table 5* and *Table 6*.

I_{AHALT} is the consumption of halt phase of active halt mode. Values given in *Table 8*, *Table 9*, and *Table 10* can be used.

t_{RUN} is the time the MCU spends in run phase between two halt phases. This time can be calculated using *Equation 2*.

Equation 2

$$t_{RUN} = t_{WKUP} + t_{CODE} + t_{SHDW}$$

t_{WKUP} is the time spent waking up the microcontroller and executing the first instruction of the interrupt routine. It depends on the halt phase configuration (regulator used and Flash state). Some typical values for this parameter are given in *Table 11*, *Table 12*, *Table 13*, and *Table 14*.

t_{CODE} is the execution time of the user routine. This time depends on the user code implementation.

t_{SHDW} is the time taken for entering low-power mode. It depends on the halt phase configuration (regulator used and Flash state). This time includes the context saving.

4.4 Halt mode

In halt mode, all clocks are stopped while the RAM content and all registers are preserved. In this mode, the MVR regulator is switched off to limit power consumption. Only the LPVR regulator is active.

4.4.1 Entering halt mode

The MCU enters halt mode when the HALT instruction is executed.

4.4.2 Exiting halt mode

Wakeup from halt mode is triggered by an external interrupt, sourced by a GPIO pin configured as an interrupt input or a peripheral interrupt. Interrupts which can wake up the device from halt mode include:

- External interrupt (GPIO)
- CAN receive interrupt
- SPI end of transfer
- I^2C interrupt (slave address match)
- Reset

4.4.3 Flash configuration during halt mode

By default, when entering halt mode, the Flash is in power-down mode. To maintain the Flash in operating mode during halt mode the HALT bit of Flash control register 1 (FLASH_CR1) must be set.

When the Flash is in power-down mode, the power consumption of the device is reduced but the wakeup time increases.

5 Power consumption and wakeup time measurements and results

The following measurements and results aim to highlight the impact of different low-power modes on MCU consumption and wakeup time.

Power consumptions given are typical values measured at 25 °C. They are for guidance only. A zip file containing the software used to take these measurements is attached to this application note. Refer to the electrical characteristics sections of the STM8S and STM8A datasheets for real specifications.

5.1 Power consumption measurements and results

Power consumption measurements are taken in run mode and low power modes. Results are given in *Table 5* to *Table 10*.

5.1.1 Measurement configuration

- V_{DDA}, V_{DDIO1}, V_{DDIO2}, and V_{DD} are connected together to V_{DD}
- V_{SSA}, V_{SSIO1}, V_{SSIO2}, and V_{SS} are connected together to V_{SS}
- All ports are set as output low level (single wire interface module, SWIM, is disabled).
- All peripherals are disabled (even if enabled by default)
- If possible, all peripheral clocks are stopped (see the CLK_PCKENRx definition in the STM8S and STM8A reference manual).

Once the device is configured, the MCU is then put into the different functioning modes described in *Section 4: Run and low-power modes*. Measurements are performed on an STM8S208MBT6 in an LQFP80 package. This is the ‘super set’ of the family which has the highest consumption rates.

Hardware environment

Run and low-power modes power consumption are measured simultaneously on V_{DD}, V_{DDIO} and V_{DDA}. The hardware configuration is given in *Figure 4*: $I_{TOT} = I_{VDDIO} + I_{VDD} + I_{VDDA}$.

Figure 4. Power supply setup

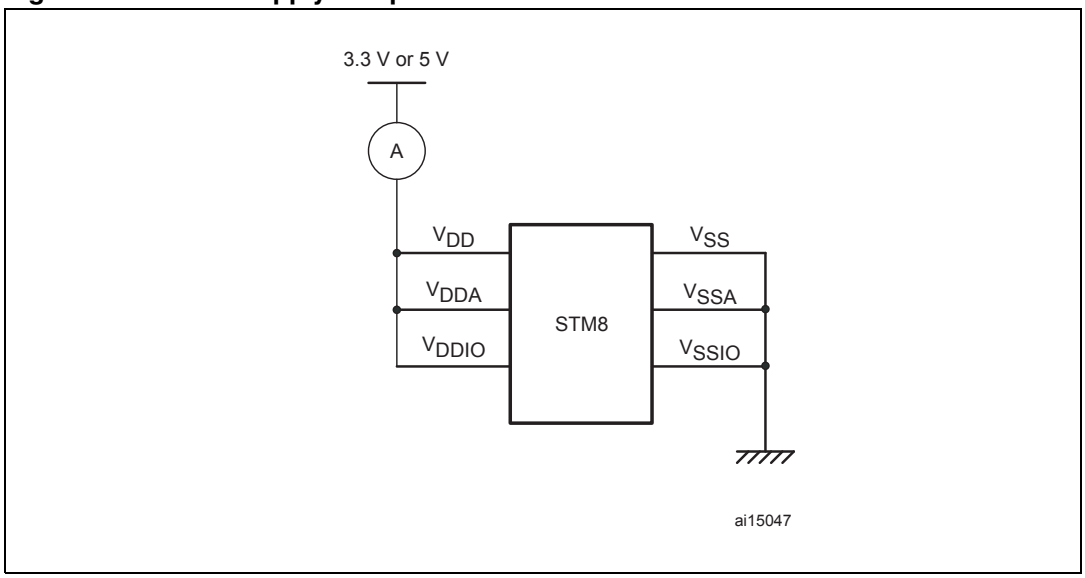
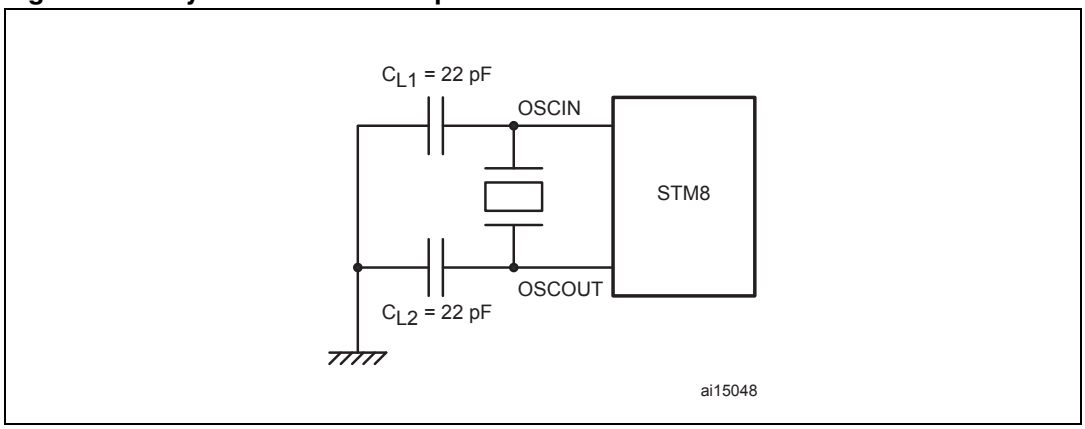


Figure 4 does not show the decoupling capacitors for all power supplies. Ceramic decoupling capacitors, with very low parasitc serial resistance, are used for the power measurements. Current leakage through such decoupling capacitors is not visible.

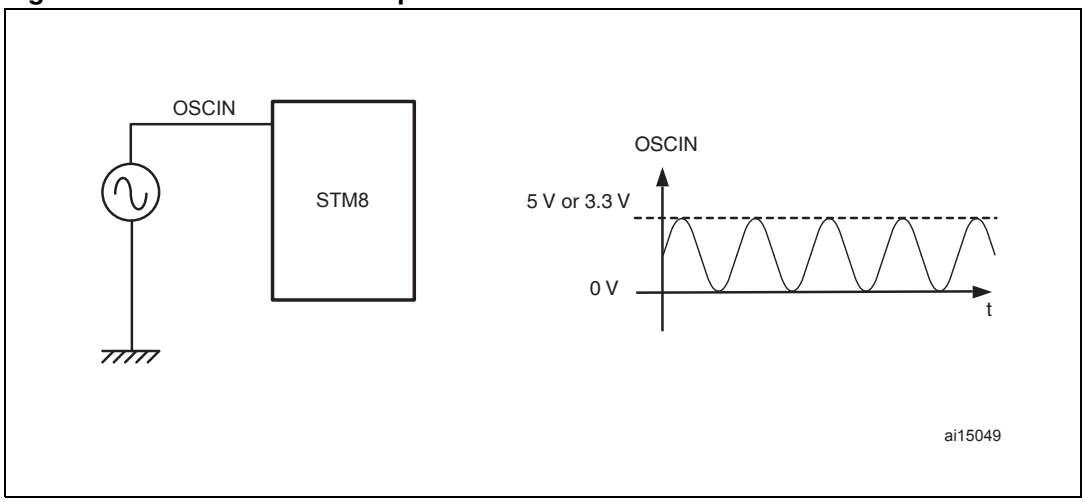
When the power measurements are performed with an external crystal, there is no internal clock division. The crystal is physically changed for each frequency. *Figure 5* shows the crystal oscillator setup.

Figure 5. Crystal oscillator setup



When the power measurements are taken with an external clock, the following configuration is used: The function generator provides a ‘SIN’ waveform at a given frequency. The wave oscillates from 0 V to 5 V or 3.3 V depending on the supply voltage. As the OSCOUT pin is not used, it is configured like other GPIOs (output push-pull outputting low level).

Figure 6. External clock setup



Selection between the external clock and the crystal oscillator is made by the external clock selection (EXTCLK) option bit. Refer to the option byte sections of the STM8S and STM8A datasheets for more details.

Firmware description for run mode measurements

Power consumption in run mode depends on several factors which are linked more to the application than to the mode itself. Therefore, it is difficult to give some typical values. For example, consumption in run mode depends on the code executed and the code placement in the memory (the latter because of the pipeline and prefetch buffer).

Section 5.1.2: Power consumption results in run mode gives an overview of consumption in run mode with the code executed from Flash and RAM memory. The code starts with the MCU configuration (GPIO, PCG, ...). The MCU then enters an endless loop, executing a sequence of various instructions one hundred times. This is to benefit from the prefetch buffer.

Note: The same code is executed from Flash and from RAM memory.

Firmware description for low-power modes measurements

The code starts with the MCU configuration (GPIO, PCG, Flash mode, and regulator used during the low-power mode). The low-power mode entering instruction is then executed and the microcontroller switches to the corresponding low-power mode. The measurement is taken when the MCU is actually in low-power mode.

For more details about firmware configurations please refer to the zip file which contains the source codes related to this application note.

5.1.2 Power consumption results in run mode

Table 5. Power consumption results in run mode, code executed from Flash

Symbol	Parameter	Condition	V_{DD} = 3.3 V	V_{DD} = 5 V	Unit
I_{TOT}	Supply current in run mode	HSE crystal $f_{CPU} = f_{MASTER}$ = 24 MHz	11.0	11.4	mA
		HSE external clock $f_{CPU} = f_{MASTER}$ = 24 MHz	10.8	10.8	
		HSE crystal $f_{CPU} = f_{MASTER}$ = 16 MHz	8.4	9.0	
		HSE external clock $f_{CPU} = f_{MASTER}$ = 16 MHz	8.2	8.2	
		HSI $f_{CPU} = f_{MASTER}$ = 16 MHz	8.1	8.1	
		HSI $f_{CPU} = f_{MASTER}/128$ = 125 kHz	1.1	1.1	
		LSI $f_{CPU} = f_{MASTER}$ = 128 kHz	0.55	0.55	

Table 6. Power consumption results in run mode, code executed from RAM

Symbol	Parameter	Condition	V_{DD} = 3.3 V	V_{DD} = 5 V	Unit
I_{TOT}	Supply current in run mode	HSE crystal $f_{CPU} = f_{MASTER}$ = 24 MHz	5.2	5.6	mA
		HSE external clock $f_{CPU} = f_{MASTER}$ = 24 MHz	5.0	5.0	
		HSE crystal $f_{CPU} = f_{MASTER}$ = 16 MHz	3.7	4.1	
		HSE external clock $f_{CPU} = f_{MASTER}$ = 16 MHz	3.5	3.5	
		HSI $f_{CPU} = f_{MASTER}$ = 16 MHz	3.4	3.4	
		HSI $f_{CPU} = f_{MASTER}/128$ = 125 kHz	1.0	1.0	
		LSI $f_{CPU} = f_{MASTER}$ = 128 kHz	0.47	0.47	

Note: The results given above are different from those in the datasheets because a different reference code is used. The reference code used for Flash and RAM execution is the same.

Observations

- Power consumption increases with clock frequency (f_{MASTER}).
- When the code is executed from RAM, the code execution time is longer than when the code is executed from Flash.
- Consumption does not depend on the supply voltage when an internal clock is used (HSI or LSI). This is because the internal oscillators are supplied by the internal 1.8 V.
- The effect of memory placement is not shown in *Table 5* and *Table 6*. However, if the size of a loop is smaller than the size of a memory block, placing the loop in the same memory block can reduce power consumption. In this case, only one block of memory is active.
- The effect of the instruction executed is not shown in *Table 5* and *Table 6*. However, consumption in run mode depends on the executed instruction and the operands of this instruction. Examples of instructions which do not consume at a high rate include: INC A and NOP. Examples of instructions which consume at a high rate include LD (from Flash) and ADD A, #$01.
- Consumption of the crystal oscillator is about:
  - 5 V: 0.6 mA
  - 3.3 V: 0.3 mA

5.1.3 Power consumption results in wait mode

Table 7. Power consumption results in wait mode

Symbol	Parameter	Condition	V_{DD} = 3.3 V	V_{DD} = 5 V	Unit
I_{TOT}	Supply current in wait mode	HSE crystal $f_{CPU} = f_{MASTER}$ = 24 MHz	2.0	2.4	mA
		HSE external clock $f_{CPU} = f_{MASTER}$ = 24 MHz	1.8	1.8	
		HSE crystal $f_{CPU} = f_{MASTER}$ = 16 MHz	1.6	2.0	
		HSE external clock $f_{CPU} = f_{MASTER}$ = 16 MHz	1.4	1.4	
		HSI $f_{CPU} = f_{MASTER}$ = 16 MHz	1.2	1.2	
		HSI $f_{CPU} = f_{MASTER}/128$ = 125 kHz	1.0	1.0	
		LSI $f_{CPU} = f_{MASTER}$ = 128 kHz	0.50	0.50	

Observations:

- Power consumption increases with clock frequency ($f_{FMASTER}$).
- Consumption does not depend on the supply voltage when an internal clock is used (HSI or LSI). This is because the internal oscillators are supplied by the internal 1.8 V.
- Consumption of the crystal oscillator is about:
  - 5 V: 0.6 mA
  - 3.3 V: 0.3 mA

5.1.4 Power consumption results in active halt mode

These measurements are taken when the device is in halt phase of active halt mode.

Table 8. Power consumption results in active halt mode (MVR on LPVR off)

Symbol	Parameter	Condition		V_{DD} = 3.3 V	V_{DD} = 5 V	Unit
		Clock	Flash			
I_{TOT}	Supply current in active halt mode (halt phase)	HSE crystal 24 MHz f_{AWU} = 24 MHz/PRSC = 128 kHz	On	700	1260	µA
			Off	640	1200	
		HSE crystal 16MHz f_{AWU} = 16 MHz/PRSC = 128 kHz	On	600	1000	
			Off	540	940	
		HSE crystal 8 MHz f_{AWU} = 8 MHz/PRSC = 128 kHz	On	500	1040	
			Off	440	980	
		HSE crystal 4 MHz f_{AWU} = 4 MHz/PRSC = 128 kHz	On	460	970	
			Off	400	910	
		LSI 128 kHz f_{AWU} = 128 kHz	On	200	200	
			Off	140	140	

Table 9. Power consumption results in active halt mode (MVR off LPVR on)

Symbol	Parameter	Condition		V_{DD} = 3.3 V	V_{DD} = 5 V	Unit
		Clock	Flash			
I_{TOT}	Supply current in active halt mode (halt phase)	LSI 128 kHz f_{AWU} = 128 kHz	On	66	68	µA
			Off	9	11	

5.1.5 Power consumption results in halt mode

Table 10. Power consumption results in halt mode (MVR off LPVR on)

Symbol	Parameter	Condition	V_{DD} = 3.3 V	V_{DD} = 5 V	Unit
I_{TOT}	Supply current in halt mode	Flash in operating mode	61.5	63.5	µA
		Flash in power-down mode	4.5	6.5	

5.1.6 Conclusions

Power consumption depends on:

- Supply voltage
- Clock frequency

The results presented in this section, show that the functioning mode (run, wait, active halt and halt) of the MCU impacts consumption and can greatly reduce it.

For active halt and halt mode, the user can choose which regulator to be used (MVR or LPVR) and the Flash state (operating mode or power-down mode). Using the results presented in *Section 5.1: Power consumption measurements and results*, consumption is estimated as follows:

- MVR: About 135 µA
- Flash: About 60 µA
- AWU + LSI: About 4 µA

5.2 Wakeup time measurements and results

Wakeup time measurements are taken in low-power modes. Results are given in *Table 11* to *Table 14*.

The wake-up time is measured from the interrupt event to the first instruction execution in the interrupt routine. *Figure 7* explains the wakeup time measurement.

Note: In the STM8S and STM8A datasheets the wake-up time is not defined in the same way. it does not include the fetch of the interrupt vector.

5.2.1 Measurement configuration

Hardware environment

The following hardware configuration is used to measure the MCU wakeup time: one pin of the microcontroller is used as an interrupt input pin. Another pin is used as an output pin. The CPU clock is outputted on the CLK_CCO pin. These three pins are monitored on an oscilloscope.

Firmware description

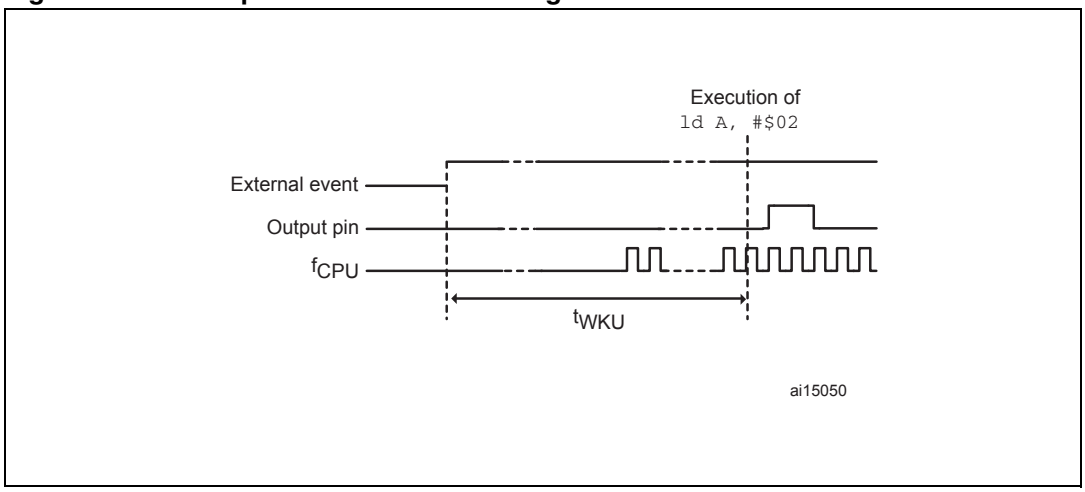
The code starts with the MCU configuration. One pin (PB5) is configured as an input interrupt. After configuration, the MCU switches into low-power mode. When an external interrupt is triggered, the MCU wakes up, and executes the interrupt routine. The interrupt routine causes a pin (PB1) to toggle.

Note: The results given in this section take into account the interrupt latency times because they include the time between the external event occurring and the first instruction of the interrupt routine being executed.

The code executed in the interrupt routine is described below:

```
ld A,#$02
ld $5005,A
ld A,#$00
ld $5005,A
```

Figure 7. Wakeup time measurement diagram



5.2.2 Wakeup time results

In most of cases, power supply voltage has no effect on wakeup time. The minus sign (-) in *Table 11* to *Table 14* below indicates that the wakeup time is the same for both power supply voltages.

5.2.3 Wakeup time results in wait mode

The wakeup time from wait mode is not always the same because the wakeup event is synchronized with the CPU clock. The variation range is about one period of the CPU clock. Data given in *Table 11* are maximum values.

Table 11. Wakeup time results in wait mode

Symbol	Parameter	Condition	V_{DD} = 3.3 V(1)	V_{DD} = 5 V	Unit
$t_{WU(WFI)}$	Wakeup time from wait mode	HSE crystal $f_{CPU} = F_{MASTER}$ = 24 MHz	-	1.0	µs
		HSE crystal $f_{CPU} = F_{MASTER}$ = 16 MHz	-	0.75	
		HSE crystal $f_{CPU} = F_{MASTER}$ = 8 MHz	-	1.4	
		HSE crystal $f_{CPU} = F_{MASTER}$ = 4 MHz	-	2.7	
		HSI $f_{CPU} = F_{MASTER}$ = 16 MHz	-	0.75	
		HSI $f_{CPU} = F_{MASTER}/128$ = 125 kHz	-	80	
		LSI $f_{CPU} = F_{MASTER}$ = 128 kHz	-	94	

1. The minus sign (-) indicates that the wakeup time is the same as for V_{DD} = 5 V.

Observations

- When no wait state is added, wakeup time from wait mode is linked only to the f_{MASTER} and to the f_{CPU} clock frequencies. Wakeup time from wait mode can be calculated using *Equation 3* and *Equation 4*.

Equation 3

$$t_{WU(WFI)min} = (2 \times 1/ f_{MASTER}) + (9 \times 1/ f_{CPU})$$

Equation 4

$$t_{WU(WFI)max} = (2 \times 1/ f_{MASTER}) + (10 \times 1/ f_{CPU})$$

- The wakeup time at 24 MHz is longer than at 16 MHz. This is because the wait state has to be introduced for accessing the Flash memory.

5.2.4 Wakeup time results in active halt mode

For active halt mode measurements, the AWU clock is always LSI. In *Table 12* and *Table 13* below, the clock in the 'Condition/Clock' column is the clock used before entering active halt mode.

The wakeup time from active halt mode is not always the same due to a sampling effect, similar to that for wait mode. The variation range is about one period of the AWU clock (7.8125 µs). Data given in *Table 12* and *Table 13* are maximum values which include sampling of the AWU clock cycle.

Table 12. Wakeup time results in active halt mode (MVR)

Symbol	Parameter	Condition		V_{DD} = 3.3 V(1)	V_{DD} = 5 V	Unit
		Clock	Others			
$t_{WU(AH)}$	Wakeup time from active halt mode	HSI (16 MHz)	Flash on	-	9	µs
			Flash off	-	11	
		HSE (16 MHz crystal)	Fast CLK wakeup on Flash off	-	11	
			Fast CLK wakeup off Flash off	394	543	
		HSE (4 MHz crystal)	Fast CLK wakeup on Flash off	-	11	
			Fast CLK wakeup off Flash off	1627	1274	

1. The minus sign (-) indicates that the wakeup time is the same for V_{DD} = 5 V.

Table 13. Wakeup time results in active halt mode (LPVR)

Symbol	Parameter	Condition		V_{DD} = 3.3 V(1)	V_{DD} = 5 V	Unit
		Clock	Others			
$t_{WU(AH)}$	Wakeup time from active halt mode	HSI (16 MHz)	Flash on	-	56	µs
			Flash off	-	58	

1. The minus sign (-) indicates that the wakeup time is the same for V_{DD} = 5 V.

Observations

- When fast clock wakeup is enabled, wakeup time always equals HSI wakeup time because the MCU automatically wakes up on HSI.
- When the wakeup clock is the HSE crystal, wakeup time is very long and depends on the supply voltage and the crystal used. This is due to the stabilization time of the oscillator. This long wakeup time can be avoided by:
  - Using the fast clock wakeup feature.
  - Programing a shorter stabilization time in the HSECNT option byte (refer to the option bytes section of the STM8S and STM8A datasheets). By default, a delay of 2048 oscillator cycles is inserted before the clock signal is released (the default value of 2048 oscillator cycles is used in the above measurements).
- Data for 'HSE crystal/Flash on' are not given in *Table 12* because the wakeup time of the Flash is always the same. The wakeup time of the Flash is about 2 µs.
- The wakeup time of the MVR is about 50 µs.

5.2.5 Wakeup time results in halt mode

Table 14. Wakeup time results in halt mode (LPVR)

Symbol	Parameter	Condition		V_{DD} = 3.3 V	V_{DD} = 5 V	Unit
		Clock	Others			
$t_{WU(H)}$	Wakeup time from halt mode	HSI (16 MHz)	Flash on	-	52	µs
			Flash off	-	54	

Observations

- The wakeup time of the Flash is about 2 µs (the same as for active halt mode).
- The wakeup time in halt mode (LPVR) is shorter than in active halt mode (see *Table 12*). Note that the data in *Table 12* include one AWU clock cycle sampling delay.

5.2.6 Conclusions

The global wakeup time of the device can be viewed as the sum of the following:

- Wakeup time of the main voltage regulator (if it is switched off during the low-power mode).
- Stabilization time of the oscillator (if it is switched off during the low-power mode).
- Wakeup time of the Flash memory (if it is switched off during the low-power mode).
- Interrupt latency for the wake up event trigger.

6 Power management tips

6.1 Rules to help minimize power consumption

- Switch off all unused peripherals (peripherals are switched off by default except the USART, LINUART and SWIM) and use the PCG feature (through the CLK_PCKENRx registers) to disable the clock provided to the unused peripherals (the clock is provided by default). Refer to *Section 3.2: Clock configuration and power management* for more details.
- In run mode, if the size of a loop is smaller than the size of a block, the loop code must be located in one block.
- All unused port pins should be configured as output low level. Do not leave any unused I/O pin configured as a floating input which could lead to useless high consumption.
- Use wait mode if you need external interrupt capability in low-power mode and if some peripherals have to remain active.
- Use the appropriate V_{DD} value because the higher the V_{DD} value, the more power is consumed.
- Use the minimum possible frequency for your application. The eight CPU prescalers and four HSI prescalers allow the required frequency value to be fitted to the application.

6.2 Choosing the optimal low-power mode for an application

- Application powered by a battery where the MCU is in sleep mode most of the time:
  - If the MCU is woken up due to external events, no time tracking is necessary and power consumption has to be as low as possible. In this case, halt mode is advised to extend battery life as much as possible.
  - Active halt mode with AWU is advised if the application does not depend on external events but needs a non accurate periodic wakeup.
- Application powered by a battery where the MCU is awake most of the time:
  - Active halt mode is advised if the MCU has to perform a few periodic actions during which no peripheral has to stay on.
  - Wait mode is advised if at least one peripheral has to stay on all the time and an interrupt can wake up the MCU.
- Application supplied by mains but where consumption is critical:
  - Run mode, with a clock prescaler adapted to the application requirement, is advised if the MCU has to run all the time.

7 Revision history

Table 15. Revision history

Date	Revision	Description of changes
09-Jan-2009	1	Initial release
08-Jul-2009	2	Updated *Table 3 on page 12* Updated *Figure 7 on page 25*
25-Aug-2011	3	Updated to refer to both STM8S and STM8A products