AN2476
Application note

STR75x low power modes

Introduction

This application note is intended for system designers who require a hardware implementation overview of the low power modes of the STR75x product family. It describes how to use the STR75x product family and details the components of supply circuitry, clock systems, register settings and low power management in order to optimize the use of STR750 in applications where low power is key.

Contents

1 Power supply and clocks

1.1 Introduction

The device can be connected in any of the following schemes, depending on the application requirements:

- Power scheme 1: Single external 3.3V power source
- Power scheme 2: Dual external 3.3V and 1.8V power sources
- Power scheme 3: Single external 5.0V power source
- Power scheme 4: Dual external 5.0V and 1.8V power sources

1.2 Power supplies

The device has five power pins:

- V_{DD_IO}: power supply for I/Os (3.3V ±0.3V or 5V ±0.5V). Must be kept on, even in STANDBY mode.

Two embedded regulators are available to supply the internal 1.8V digital power:

- V_{18} (pins V_{18REG} and V_{18} which are internally shorted): Power Supply for Digital, SRAM and Flash: 1.8V ± 0.15V.
- V_{18_BKP}: Backup Power Supply for STANDBY or STOP Mode

V_{18} and V_{18_BKP} are normally generated internally by the embedded regulators:

The Main Voltage Regulator (MVREG) supplies V_{18} and V_{18_BKP}. It delivers a power supply of 1.8V.

The Low Power Voltage Regulator (LPVREG) can supply V_{18_BKP} or V_{18} in STOP or STANDBY mode (see *Section 1.8: Low power modes on page 14*). It delivers a power supply of 1.6V ± 0.2V.

When the embedded regulators are used, the Main Digital part of the chip (V_{18}) can be powered off (meaning V_{18} left hi-Z) while keeping the backup circuitry powered on (V_{18_BKP}).

It is also possible to supply V_{18} and V_{18_BKP} externally.

Two sensitive analog blocks have dedicated power pins:

- V_{DDA_PLL}: Analog Power supply for PLL (must have the same voltage level as V_{DD_IO})
- V_{DDA_ADC}: Analog Power supply for ADC (must have the same voltage level as V_{DD_IO})

1.3 Power supply schemes

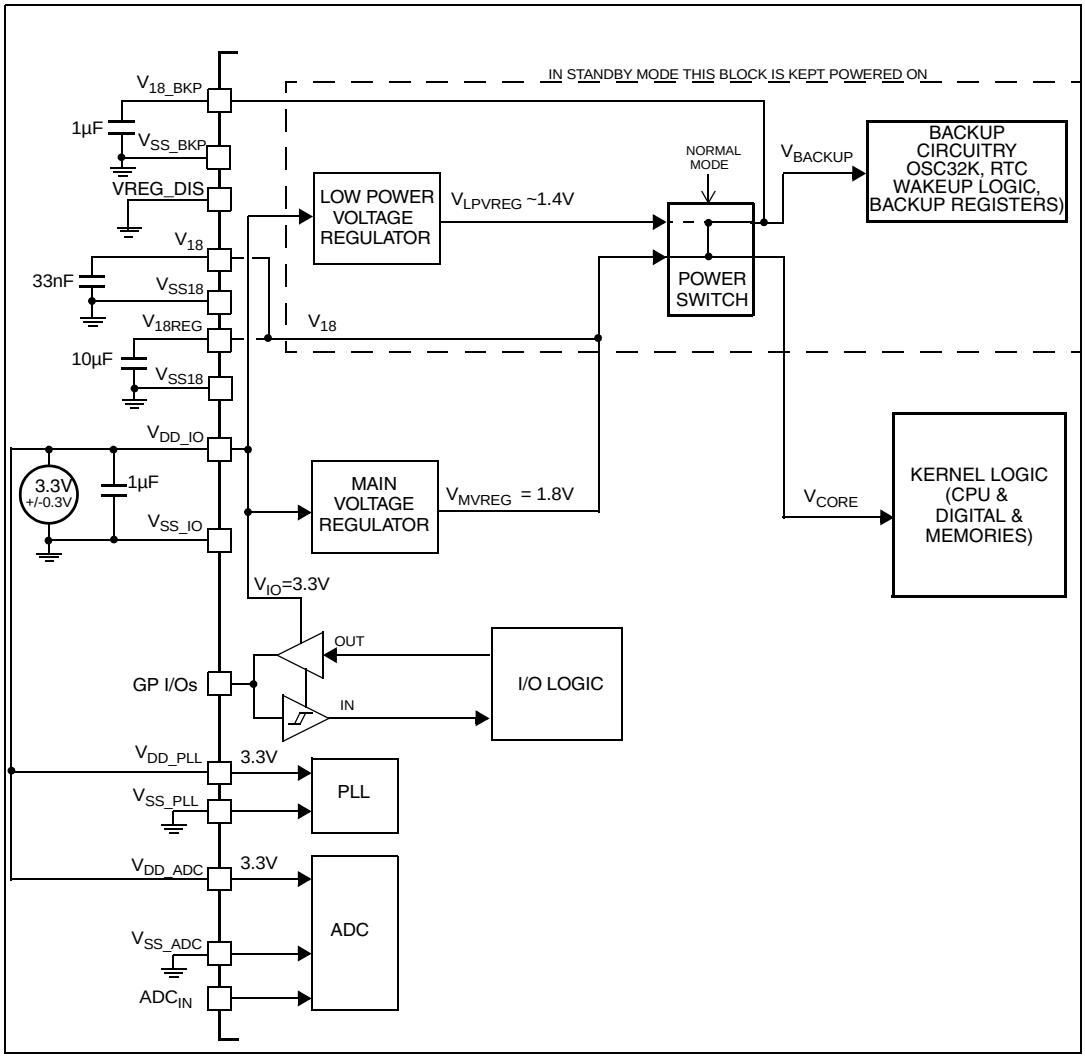
1.3.1 Power scheme 1: single external 3.3V power source

In this configuration, the internal voltage regulators are switched on by forcing the VREG_DIS pin to low level. The V_{CORE} supply required for the kernel logic and the V_{BACKUP} supply required for the backup circuitry are generated internally by the Main Voltage Regulator or the Low Power Voltage Regulator (depending on the selected low power mode). This scheme has the advantage of requiring only one power source. Refer to *Figure 1 on page 5*.

At power-up and during *NORMAL* mode (all operating modes except STOP and STANDBY Modes):

- The Main Voltage Regulator powers both V_{CORE} supply required for the kernel logic and the V_{BACKUP} supply required for the backup circuitry.
- The Low Power Regulator is not used

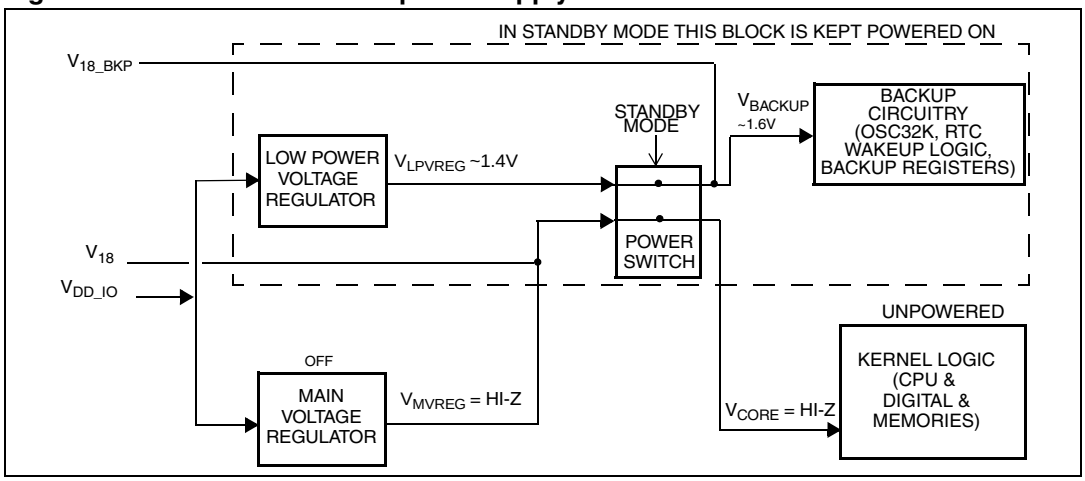
Figure 1. Power supply scheme 1 (single 3.3V supply VREGDIS=0) in NORMAL mode



In STANDBY mode (see *Section 1.8.6: STANDBY mode on page 22*):

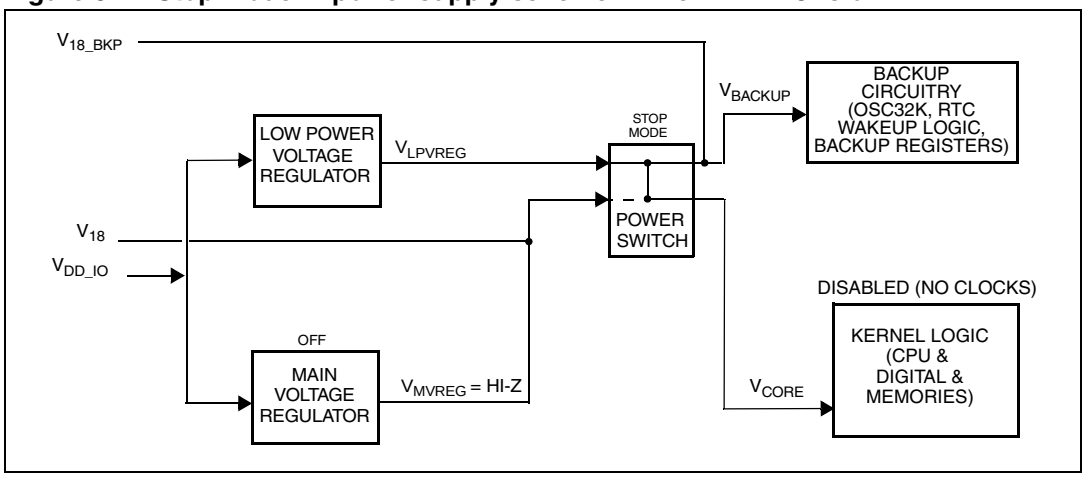
- the Main Voltage Regulator is disabled (output V_{CORE} is hi-Z), the kernel is powered-off
- the Low Power Voltage Regulator powers the backup circuitry.

Figure 2. STANDBY mode in power supply scheme 1



In STOP mode (where all clocks are disabled), it is possible to disable the Main Voltage Regulator and to power both the backup circuitry and the kernel logic with the Low Power Voltage Regulator (see *Section 1.8.5: STOP mode on page 18*)

Figure 3. Stop mode in power supply scheme 1 when MVREG is off



1.3.2 Power scheme 2: dual external 3.3V and 1.8V power sources

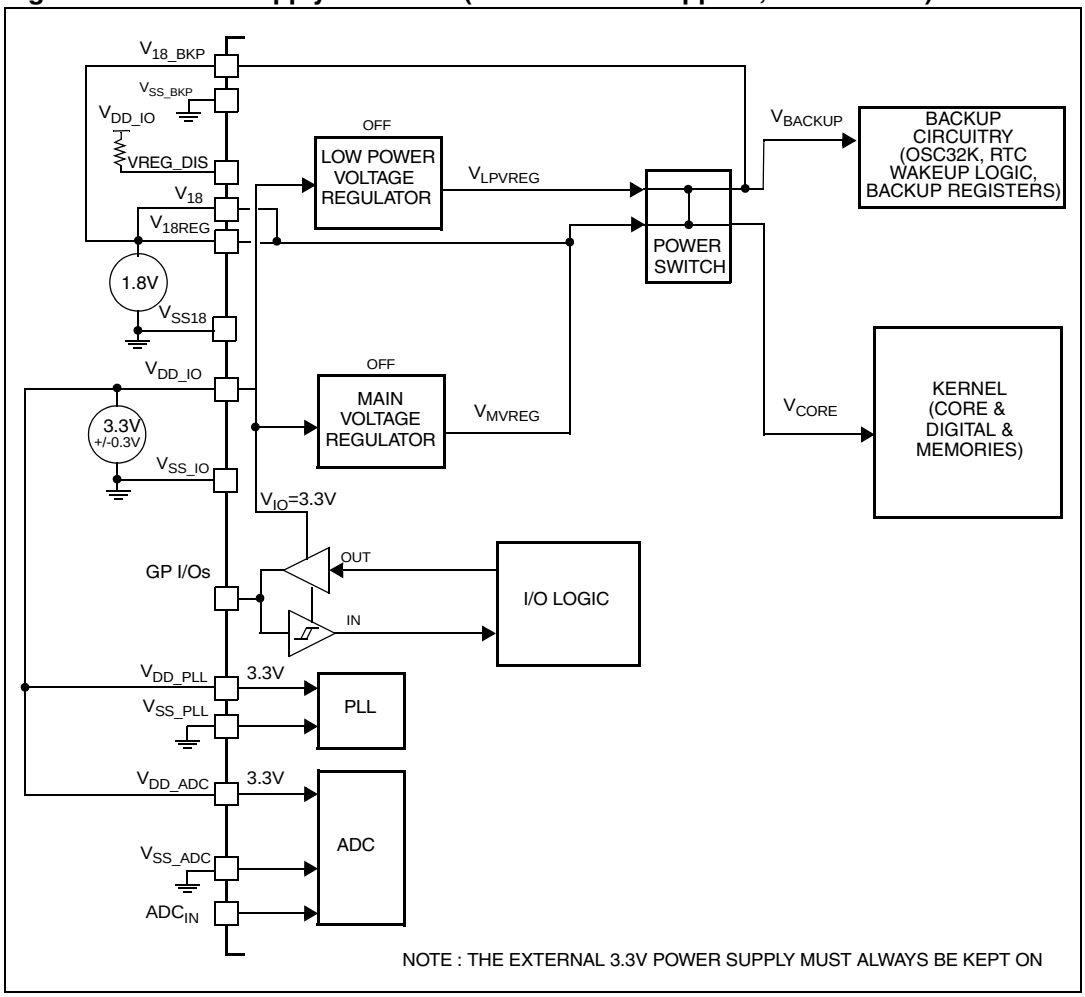
In this configuration, the internal voltage regulators are switched off by forcing the VREG_DIS pin to high level. This scheme has the advantage of saving power consumption when the 1.8V power supply is already available in the application. V_{18} and V_{18_BKP} are provided externally through the V_{18REG}, V_{18} and V_{18_BKP} power pins.

VREG_DIS pin is tied to high level which disables the Main Voltage Regulator and the Low Power Voltage Regulator.

All digital power pins (V_{18REG}, V_{18} and V_{18_BKP}) **must be externally shorted** to the same 1.8V power supply source. Internally, V_{CORE} and V_{BACKUP} are shorted by the Power Switch.

In this scheme, STANDBY Mode is not available.

Figure 4. Power supply scheme 2 (3.3V and 1.8V supplies, VREGDIS=1)



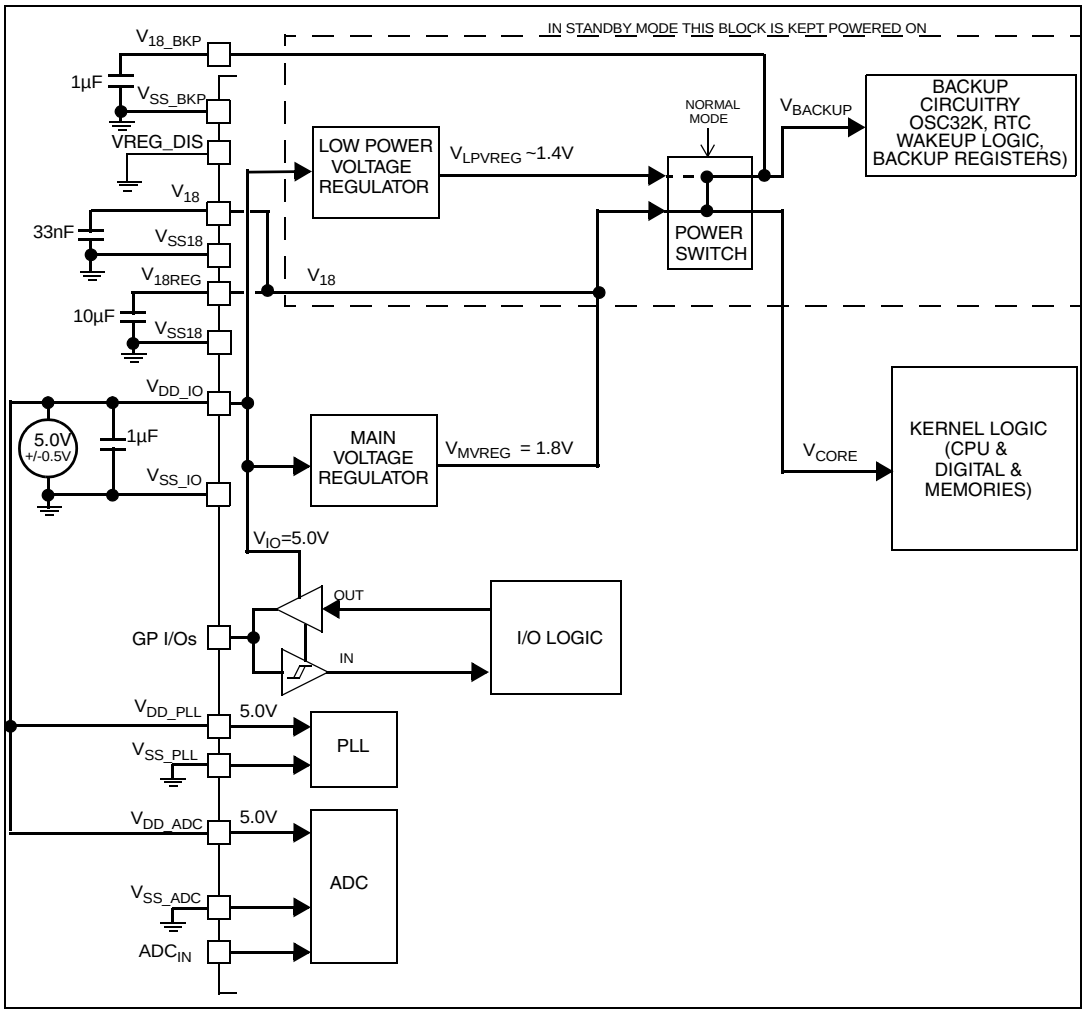
1.3.3 Power scheme 3: single external 5.0V power source

This power scheme is equivalent to Power Scheme 1 with the exception that:

- The external power supply is 5.0V +/-0.5V instead of 3.3V
- USB functionality is not available

STANDBY mode is supported in this scheme.

Figure 5. Power supply scheme 3 (single 5.0V supply VREGDIS=0) in NORMAL mode



1.3.4 Power scheme 4: dual external 5.0V and 1.8V power sources

In this configuration, the internal voltage regulators are switched off, by forcing the VREG_DIS pin to high level. This scheme has the advantage of saving power consumption when the 1.8V power supply is already available in the application and providing 5V I/O capability. V_{18} and V_{18_BKP} are provided externally through the V_{18REG}, V_{18} and V_{18_BKP} power pins.

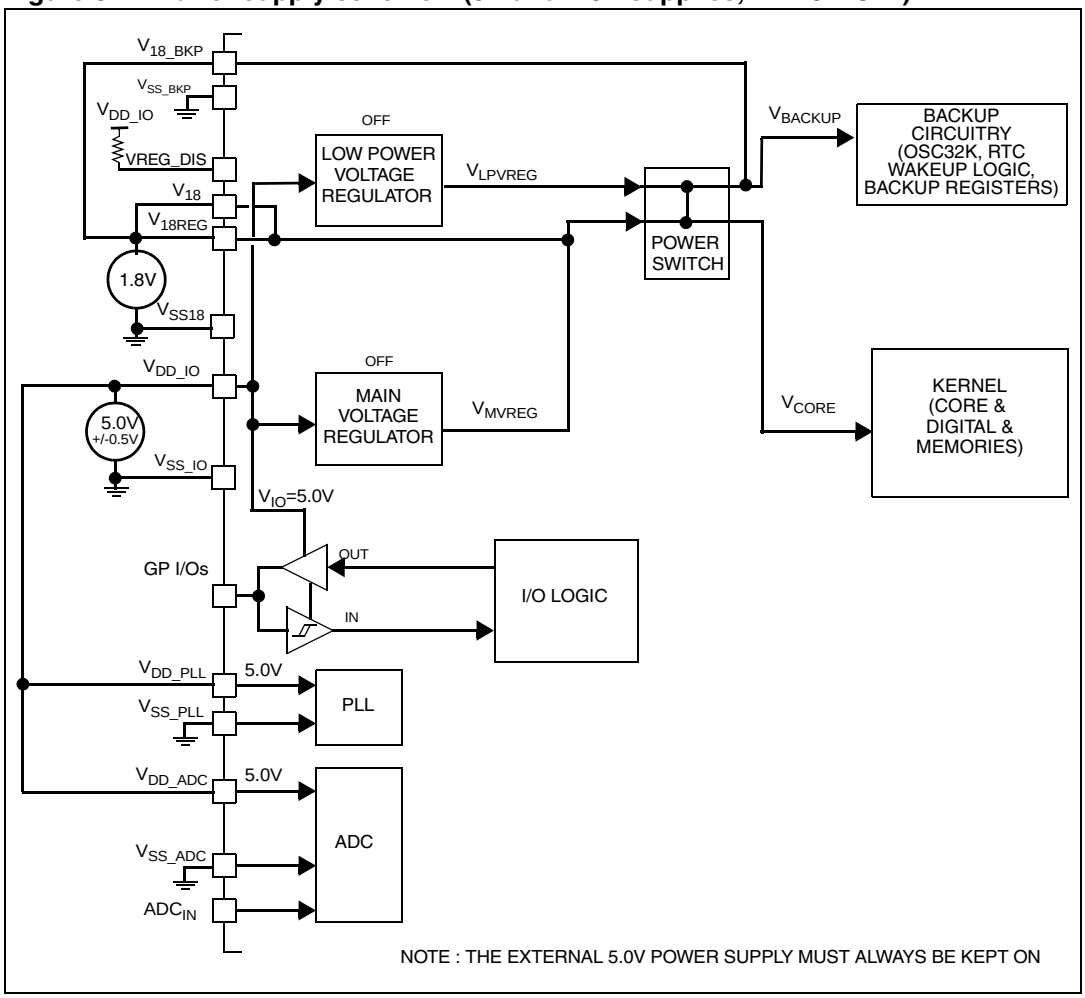
VREG_DIS pin is tied to high level which disables the Main Voltage Regulator and the Low Power Voltage Regulator.

All digital power pins (V_{18REG}, V_{18} and V_{18_BKP}) **must be externally shorted** to the same 1.8V power supply source. Internally, V_{CORE} and V_{BACKUP} are also shorted by the power switch shown in *Figure 6: Power supply scheme 4 (5V and 1.8V supplies, VREGDIS=1) on page 9*.

In this scheme:

- STANDBY mode is not available
- USB functionality is not available

Figure 6. Power supply scheme 4 (5V and 1.8V supplies, VREGDIS=1)



1.4 Main voltage regulator

In power scheme 1 or 3 (see *Section 1.3: Power supply schemes on page 5*) the Main Voltage Regulator provides the 1.8V power supply starting from V_{DD_IO}. The V_{18REG} pin must be connected to external stabilization capacitors (min. 10 uF Tantalum, low series resistance, and 33nF ceramic). The V_{18} pin must be left unconnected. A decoupling capacitor of 1µF must be added on the V_{DD_IO} pin which is closest to the V_{18REG} pin. Refer to the diagram in the application note, *AN2419, STR75x hardware development getting started*.

1.5 Low power voltage regulator

In power scheme 1 or 3 (see *Section 1.3: Power supply schemes on page 5*) the Low Power Voltage Regulator is used in STANDBY Mode and in STOP Mode (when MVREG is OFF in STOP mode).

The V_{18_BKP} pin must be connected to an external stabilization capacitor of 1µF. It generates a non-stabilized and non-thermally-compensated voltage of approximately 1.4V.

The output current is enough to power the backup circuitry (RTC and Wakeup Logic) or the V_{CORE} supply required for the kernel logic in STOP mode (with the low power control parameters FLASH and OSC4M OFF, see *Section 1.8.5: STOP mode on page 18*).

- In STANDBY mode, it provides the power supply starting from V_{DD_IO} to the backup circuitry while the Kernel Logic is un-powered.
- In STOP mode, if you program the LP_PARAM13 control bit (MVREG OFF) to disable the Main Voltage Regulator, the Low Power Voltage Regulator powers the whole digital circuitry, saving the static consumption of the Main Voltage Regulator (~100µA typical).

1.6 Regulator startup monitor (RSM)

The Main Voltage Regulator and the Low Power Voltage Regulator have an internal Regulator Startup Monitor (RSM) which monitors the regulated power supply.

At power-up, the RSM extends the assertion of the RESET until the regulators are operating (until V_{18_BKP} and V_{18} are at the right level).

If there is a drop in the regulated power supply, the RSM automatically generates a RESET. This enhances the security of the system by preventing the MCU from going into an unpredictable state.

Note: An external reset circuit must be used to provide the RESET at VDD_IO power-up. It is not sufficient to rely on the RESET generated by the RSM in this case. This is because RSM operation is guaranteed only when VDD_IO is within the specification (minimum 3.0V).

When regulators are not used (VREG_DIS pin is tied to high level), the RSM is disabled.

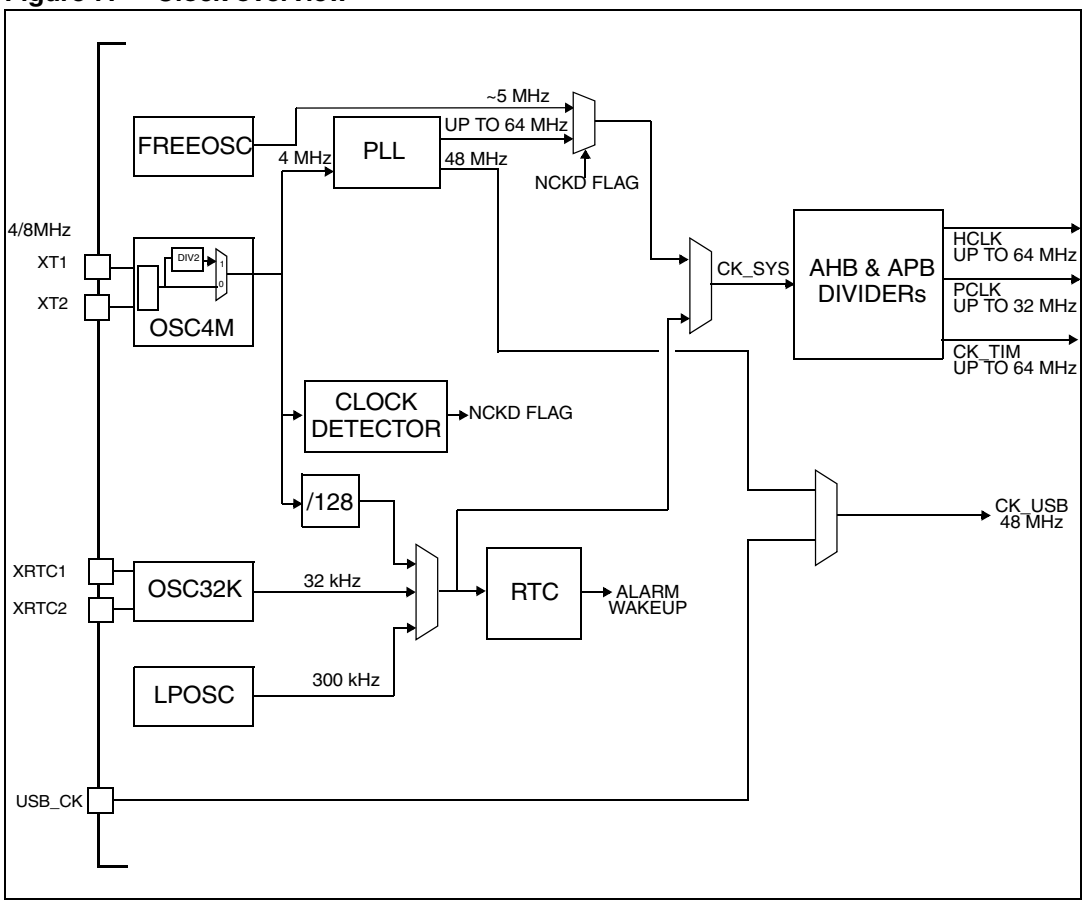
1.7 Clocks

The Clock controller provides the internal clocks needed by the different parts of the MCU:

- HCLK clocks all the peripherals mapped on the AHB bus
- PCLK clocks all the peripherals mapped on the APB bus
- CK_TIM clocks several timer counters independently from the APB clock
- CK_USB clocks the USB interface kernel

1.7.1 Clock overview

Figure 7. Clock overview



Several on-chip oscillators can feed the MCU system clock (CK_SYS) from which the HCLK and PCLK derive:

- FREEOSC: Internal Free Running Oscillator providing a clock of approximately 5 MHz, also used as emergency clock. It consists of the internal VCO of the PLL configured in free running mode
- OSC4M: 4 MHz Main Oscillator, based on:
  - a 4 MHz Crystal/Ceramic oscillator connected to XT1/XT2
  - or an 8 MHz Crystal/Ceramic oscillator to XT1/XT2 followed by an embedded divider by 2
  - or external clock connected to XT1

  which can be multiplied by the PLL to provide a wide range of frequencies (up to 64 MHz)
- OSC32K: 32.768kHz Oscillator (Crystal or Ceramic oscillator) which can drive either the system clock and/or the RTC.
- LPOSC: Internal Low Power RC Oscillator providing a clock around 300 kHz which can drive either the system clock and/or the RTC.

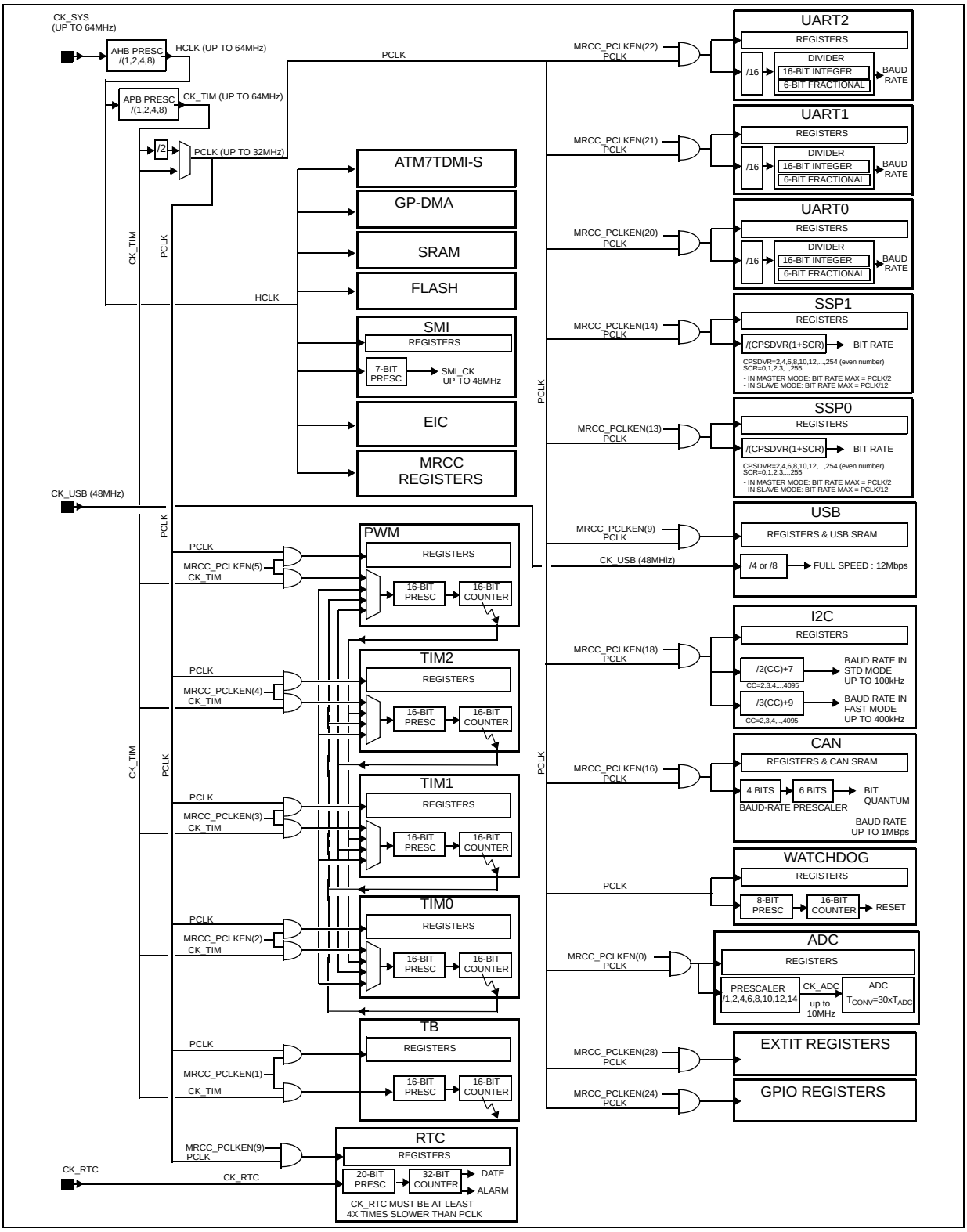
Several configurable dividers provide a high degree of flexibility to the application in the choice of the APB or AHB frequency, while keeping a fixed frequency value for the USB clock (48 MHz).

The Clock Detector (CKD) protects the Microcontroller against OSC4M or external clock failures.

The RTC provides calendar, alarm and wake-up functions and can be clocked by any of the oscillators other than FREEOSC.

1.7.2 Peripheral clock scheme

Figure 8. Peripheral clock Scheme



1.8 Low power modes

Several low power modes are implemented to reach the best compromise between lowest power consumption, fastest start-up time and available wake-up sources:

- SLOW MODE: System clock speed is reduced
- PCG MODE: APB Peripherals Clock Gating
- WFI MODE: Wait For Interrupts
- STOP MODE: All clocks are disabled
- STANDBY MODE: Part of the chip is unpowered

Software has to execute the *low power bit writing sequence* (see below) to enter WFI, STOP or STANDBY modes. This sequence protects the application from entering a low power mode unintentionally.

1.8.1 Low power bit writing sequence

WFI, STOP or STANDBY modes are entered by software writing the following specific sequence to the LP bit in the MRCC_PWRCTRL register.

Before executing the sequence, LP_DONE bit status must be previously cleared.

- Write LP bit to ‘1’
- Write LP bit to ‘1’
- Write LP bit to ‘0’
- Write LP bit to ‘1’
- Read LP: if successful, LP is first read at 1 and then cleared by hardware when entering in Low Power Mode.

This sequence is performed by internal function of the library *Section 2: STR750 library low power mode functions on page 26*.

If this sequence is not performed correctly or if LP_DONE was not cleared, the sequence is automatically reset and must be re-executed from the beginning.

To abort the sequence, simply write twice the LP bit to ‘0’.

The LP_DONE status flag is set after a successful LP bit writing sequence (meaning that the system has entered and exited the low power mode). This information is useful to know if the low power mode has been effectively entered or not.

As soon as the LP bit writing sequence is initiated:

- The MRCC_PWRCTRL register is locked (any write access other than to the LP bit will be discarded) until the sequence is completed or aborted
- Interrupts are masked (IRQ and FIQ signals to the ARM CPU are gated) until the sequence is completed or aborted

Any interrupts which occur during this sequence will not be lost: they will be served *after* the sequence is completed or aborted.

Before performing the sequence, all pending bit interrupts used to wake-up from STOP mode should be cleared.

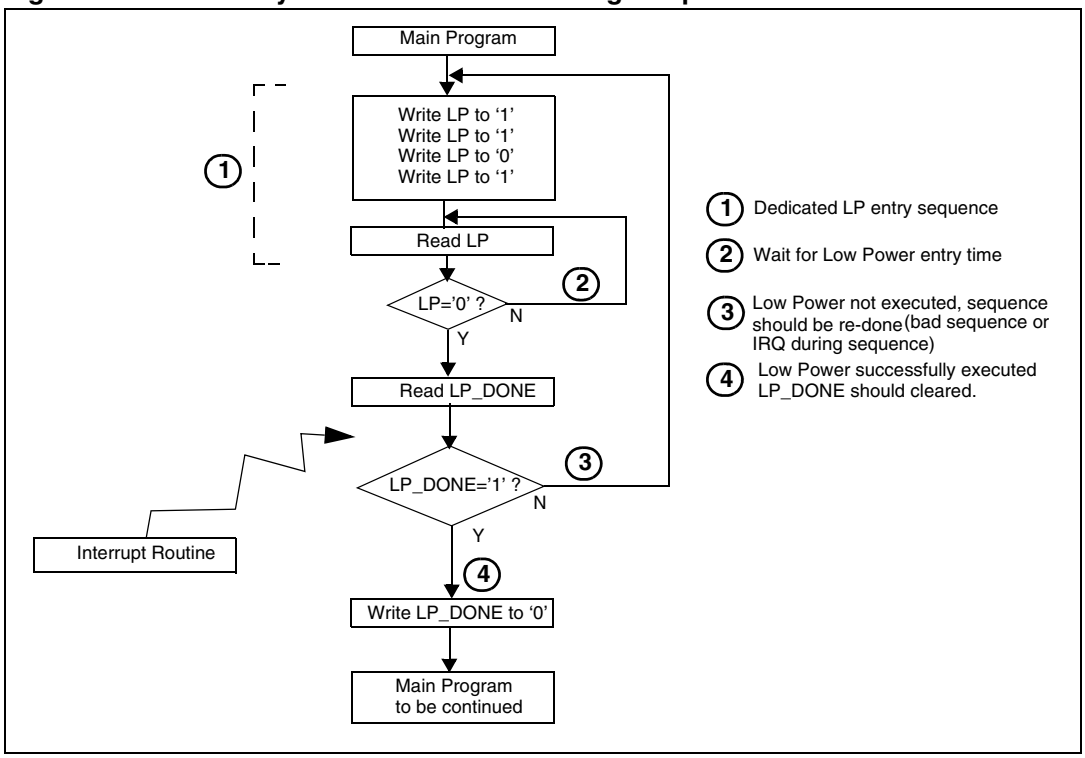
If any interrupts from an external interrupt line, an RTC alarm or NCKDF (no clock detected) event are still pending, the sequence will not performed correctly.

If an interrupt occurs during this sequence, the MCU will not enter Low Power Mode after completing the sequence, but will serve the interrupt instead.

In both cases, software must then reexecute the low power bit writing sequence to effectively go into low power mode. It is possible to determine if the low power mode was entered by reading the LP and LP_DONE bits status after wake-up.

It is mandatory to follow this flow chart to manage the low power mode:

Figure 9. Mandatory software flow for entering low power mode



The choice of low power mode entered with this sequence depends on the state of the LPMC bits described in the *Table 1 on page 15*.
These bits can be set in the MRCC_PWRCTRL register. See Reference Manual for more detail.

Table 1. Low Power mode selection

Control bits				Low Power Mode Selected
LPMC[1:0]		BURST	WFI_FLASH_EN	
0	0	-	-	STOP Mode
0	1	-	-	Software Reset
1	0	0	0	WFI, Flash disabled (DMA not allowed in WFI)
		0	1	WFI, Flash enabled (DMA allowed in WFI)
		1	0	Forbidden
		1	1	WFI, Flash enabled (DMA allowed in WFI)
1	1	-	-	STANDBY Mode

1.8.2 SLOW mode

In SLOW mode, power consumption is reduced by slowing down the main clocks. Please refer to *Section 2.1: MRCC_CKSYSConfig on page 26* to use dedicated function of the library. It is possible to use all the device functions of the chip, but at reduced speed.

To enter Slow Mode, the HPRESC[1:0] and PPRESC[2:0] bits prescaler in the MRCC_PWRCTL register must be programmed to reduce the CPU and/or peripheral clock frequency. Please refer to the following sections to use dedicated function of the library:

Section 2.2: MRCC_HCLKConfig on page 27
Section 2.3: MRCC_CKTIMConfig on page 28
Section 2.4: MRCC_PCLKConfig on page 29

It is also possible to use the RTC clock as system clock. This is done by programming the CKOSCSEL control bit with a special sequence of write instructions. In this configuration, the PLL can be disabled by software to reduce power consumption. When exiting from this mode, if the PLL has been disabled, the software must re-enable it in the same way as after a RESET.

1.8.3 PCG mode: peripherals clocks gated mode

It is possible to gate or ungate each clock feeding the APB peripherals independently. This is done by writing to the MRCC_PCLKEN register. This can be done at any time, allowing you to save power whenever these peripherals are not used, adapting dynamically to the needs of the application. To use a dedicated function of the library please refer to *Section 2.8: MRCC_PeripheralClockConfig on page 31*.

By default, after reset, all APB peripherals are in PCG Mode (their clock is disabled).

The clock to the Watchdog is an exception, it cannot be gated in PCG mode, it is always clocked by PCLK.

Some peripherals (USB, TB, TIM, PWM) use a clock source other than PCLK which can be gated using specific control bits:

- CK_USB 48 MHz clock gated by the PLL2EN bit in the Clock Control Register (MRCC_CLKCTL).
- CK_TIM: clock to TB, TIM, PWM timers gated by specific bits in the Peripheral Clock Enable register (MRCC_PCLKEN).

1.8.4 WFI mode: Wait For Interrupt mode

In WFI mode power consumption is reduced by stopping the CPU. The program stops executing but peripherals are kept running and the register contents are preserved. Execution restarts when an interrupt request is sent to the EIC.

WFI mode entry procedure

Use the following procedure to enter WFI mode:

1. First select WFI Mode by programming the LPMC[1:0] and the WFI_FLASH_EN bits in the Power Management Control register (MRCC_PWRCTRL). Program the LP_PARAM bits in the MRCC_PWRCTRL register to enable or disable selected parts of the MCU circuitry in WFI mode
2. Execute the Low Power Bit Writing Sequence as described in *Section 1.8.1: Low power bit writing sequence on page 14*.

This procedure is performed by the function of the library described in the *Section 2.5: MRCC_EnterWFIMode on page 29*.

LP mode control parameters

If the Flash is used in **Burst mode**, it must be kept enabled in WFI mode (WFI_FLASH_EN bit = 1).

- If the WFI_FLASH_EN bit is set, the Flash memory is kept enabled in WFI mode and the system can perform DMA transfers.
- The LP_PARAM bits in the MRCC_PWRCTRL register have no effect in this case.

If the Flash is used in **Non-Burst mode**, the Flash may be disabled in WFI mode (WFI_FLASH_EN bit = 0) and the LP_PARAM bits in the MRCC_PWRCTRL register used to further reduce power consumption in WFI mode as follows:

- If WFI_FLASH_EN is reset, DMA must be disabled when entering WFI mode.
- FLASH OFF (LP_PARAM14). The Flash can be put in low power mode (LP_PARAM14=0) or disabled (LP_PARAM14=1). In the second case, the static consumption of the Flash is removed, but latency is incurred when exiting WFI mode.
- Other LP_PARAM bits have no effect in WFI mode.

Wake-up events

Wake-up from WFI mode can be performed either by an external interrupt from an I/O pin or an internal interrupt generated by one of the internal peripherals (assuming that its clock has not been disabled by software) or by NCKDF (no clock detected) interrupt.

DMA interrupts can also be used, but only if the WFI_FLASH_EN bit in the Power Management Control register (MRCC_PWRCTRL) has been previously set.

You can also combine WFI mode with PCG mode. However in this case, the peripherals that are disabled cannot generate the interrupt used to exit from WFI mode and another interrupt source has to be available.

It is also possible to wake-up from WFI when triggered by an external interrupt line (configured in the EXTIT registers) without having enabled the corresponding IRQ in the EIC. In this case, the external interrupt event will cause a wake-up from WFI, but no IRQ routine will be invoked. The program will simply resume execution.

The RTC can wake up the MCU from WFI in two ways:

- either using the global interrupt of the RTC (Alarm, Second or Overflow)
- or through external interrupt line #15 which is connected only to the RTC alarm. In this case, this external interrupt must be properly configured.

However, it is recommended to use the global interrupt of the RTC to wake-up from WFI.

Wake-up latency

If the Flash memory is kept enabled (WFI_FLASH_EN) or in low power mode (LP_PARAM14=0), there will be no latency when resuming after wake-up from WFI mode.

If Flash memory is disabled in WFI Mode, using the LP_PARAM14 bit in the Power Management Control register (MRCC_PWRCTRL), latency occurs when restarting due to the Flash start-up time. Wait states are inserted until the Flash is ready. This latency is not deterministic and may vary, depending on temperature or process dispersion.

Note: This latency does not occur if the software fetches from SRAM when it restarts. Any access to Flash will be performed correctly, but will result in wait states being inserted if the Flash is not ready.

The Flash must be configured all the above control parameters, even if SRAM is used to fetch parts of the software.

Table 2. WFI mode functionality

Control bits		Functionality in WFI mode											
							Wake-up resources						
LP_PARAM14: Flash off	RTC enabled	RTC clock source	Voltage Regulators	OSC4M Oscillator	PLL + Clock Detector (CKD)	Flash	NRSTIN	Internal interrupts	External interrupts	RTC Alarm	Clock Detector	Wake-up latency due to:	Clock config. state after wake-up
1	No	All	Main VREG	On	On	Off (in non Burst mode)	Yes	Yes if enabled	All	No	Yes	Flash	Unchanged
	Yes									Yes			
0	No			On	On	On (enabled or low power mode)				No		No latency	
	Yes									Yes			

1.8.5 STOP mode

In this mode CK_SYS is stopped. This means that the CPU and all the peripherals clocked by CK_SYS or derived clocks are disabled. As a result, the AHB and APB peripherals are not clocked, so the digital part of the MCU consumes no power other than the leakage current due to the process technology.

In STOP mode, all the registers and SRAM contents are preserved, the PLL and clock configuration can remain unchanged.

STOP mode entry procedure

Use the following procedure to enter STOP mode:

1. First select STOP Mode by programming the LPMC[1:0] bits in the MRCC_PWRCTRL register.
2. Program the LP_PARAM bits in the MRCC_PWRCTRL register to enable or disable some peripherals in STOP mode
3. Execute the Low Power Bit Writing Sequence as described in *Section 1.8.1: Low power bit writing sequence on page 14*.

This procedure is performed by the function of the library described in the *Section 2.6: MRCC_EnterSTOPMode on page 30*

LP mode control parameters

To further reduce power consumption in STOP mode, additional parts of the MCU can be disabled when entering STOP Mode. This can be configured by programming the LP_PARAM[15:13] bits in the MRCC_PWRCTRL register:

- **OSC4M and PLL OFF (LP_PARAM15)**: This removes the power consumption of the PLL and the OSC4M oscillator, however the clock configuration returns to its reset state. So when resuming from Stop mode after a wake-up event, it is necessary to manage the start-up time for oscillator stabilization and to re-enable the PLL.
- **FLASH OFF (LP_PARAM14)**: This saves the static power consumption of the Flash, but some latency for Flash start-up is incurred when waking up from STOP mode.
- **MVREG OFF (LP_PARAM13)**: When this parameter is set, the **LPVREG** (Low Power Voltage Regulator) powers the whole circuit, saving the static consumption of the MVREG (Main Voltage Regulator). In addition, the OSC4M oscillator, PLL and Flash are also switched off (all three LP_PARAM bits 13, 14 and 14 are set) because the Low power regulator does not have enough power to drive them. When resuming from Stop mode after a wake-up event, start-up latency is incurred.

Note: 1 When using the LP_PARAM15 bit to switch off OSC4M in STOP mode, it is recommended to clear the NCKDF bit just before entering STOP mode. Otherwise, if NCKDF=1 when entering STOP mode, OSC4M will not be disabled and will continue to use power.

2 If OSC4M is not used as the system clock source, it is strongly recommended to switch it off by setting the OSC4MOFF bin STOP mode even if LP_PARAM 15 is set and NCKDF is cleared as recommended in note 1 above.

Table 3. STOP mode functionality

Control bits				Functionality in STOP mode											
									Wake-up resources						
LP_PARAM13: MVREG off	LP_PARAM15: OSC4M off	LP_PARAM14: Flash off	RTC enabled	RTC clock source	Voltage Regulators	OSC4M Oscillator	PLL + Clock Detector (CKD)	Flash	NRSTIN	Internal interrupts	External interrupts	RTC Alarm	Clock Detector interrupt	Wake-up latency due to:	Clock configuration state after wake-up
1	1	1	No	None	LP VREG	Off	Off	Off	Yes	No	All	No	No	MVREG + Flash + OSC4M	Reset State
			Yes	LPOSC, OSC32K								Yes			
0	1	1	No	None	Main VREG	Off	Off	Off				No		FLASH + OSC4M	
			Yes	LPOSC, OSC32K								Yes			
	1	0	No	None		Off	Off	On				No		OSC4M	
			Yes	LPOSC, OSC32K								Yes			
	0	1	No	All		On	On	Off				No	Yes	Flash	Unchanged
			Yes									Yes			
	0	0	No			On	On	On				No		No latency	
			Yes									Yes			

Power Consumption in STOP mode

Table 4 gives an approximate indication of the power savings that can be gained using the LP_PARAM15:13 bits in the Power Management Control register (MRCC_PWRCTRL).

Table 4. Typical power consumption of circuits controlled by LP mode parameters

Control bit	Circuit	Power Consumption Saving (under typical conditions with 3.3V single power scheme)
LP_PARAM15	OSC4M oscillator and PLL	1.8 mA
LP_PARAM14	Flash memory	515 µA
LP_PARAM13	MVREG Main Voltage Regulator	130 µA

When all the LP_PARAM bits are enabled, the MCU power consumption consists only of the static consumption of the Low Power Regulator (and the RSM) plus the leakage currents of the digital logic (a total of 20µA typical at T_A= 25° C for 3.3 V supply).

In dual supply mode (VREG_DIS pin=1), the power consumption consists only of the leakage current (10 µA typical at T_A = 25° C on V_{18} supply).

Note: The Low Power RC Oscillator (LPOSC) and the 32.768kHz crystal oscillator (OSC32K) are still active if previously programmed (RTC is still working if clocked by one of these two clocks).

The ADC or USB can also consume power unless they are disabled before entering STOP mode.

Wake-up events

Wake-up from STOP mode can be triggered by a reset, an external interrupt event or an RTC alarm or NCKDF (no clock detected) interrupt. Refer to *Table 3: STOP mode functionality on page 20*.

It is also possible to wake-up from STOP mode triggered by an external interrupt line (configured in the EXTIT registers) without having enabled the corresponding IRQ in the EIC. In this case, the External Interrupt Event will exit from STOP, but no IRQ routine will be invoked. The program will simply resume execution.

The user program must clear the external interrupt line pending bit in the Pending Register (EXTIT_PR) which caused the wake-up from STOP mode. If this pending bit is not cleared, the next STOP mode entry procedure will be ignored and program execution will continue.

The RTC can wake up the MCU from STOP mode only through external interrupt line #15 which is internally connected to the RTC alarm. In this case, this external interrupt must be properly configured. Note that only the RTC alarm is connected to external interrupt line #15 (not the RTC second or overflow interrupt).

Wake-up latency

When the OSC4M oscillator, the Flash memory and the MVREG voltage regulator are kept active, the application is able to restart immediately with almost no latency.

If Flash memory is disabled in STOP Mode, using the LP_PARAM14 bit in the Power Management Control register (MRCC_PWRCTRL), latency occurs when restarting due to the Flash start-up time. Wait states are inserted until the Flash is ready. This latency is not deterministic and may vary, depending on temperature or process dispersion.

Note: This latency does not occur if the software fetches from SRAM when it restarts. Any access to Flash will be performed correctly, but will result in wait states being inserted if the Flash is not ready.

If you disable the OSC4M oscillator in STOP Mode, using the LP_PARAM15 bit in the Power Management Control register (MRCC_PWRCTRL), the clock configuration is lost and returns to reset state. Software must re-configure the clocks when the MCU wakes-up from STOP mode.

Debug mode

If using the debug features of the ARM7TDMI-S, and the current application puts the MCU in STOP mode, the debug connection will be lost, because the ARM7TDMI-S core is no longer clocked.

However, using Low Power Debug mode, the software can be debugged even with extensive use of STOP mode. To enable this feature, set the LPMC_DBG bit after RESET. In this configuration, whenever a STOP mode entry sequence is successfully executed, the MCU goes into WFI Mode instead of STOP mode, so the core keeps running and its debug features remain active.

As WFI is performed instead of STOP mode, the WFI_FLASH_EN must be set (see *Section 1.8.4: WFI mode: Wait For Interrupt mode on page 16*);

1.8.6 STANDBY mode

This mode is only available when using the embedded regulators (pin VREG_DIS tied to 0).

In STANDBY Mode, the MVREG (Main Voltage Regulator) is disabled and the internal power switch opened (see *Figure 2: STANDBY mode in power supply scheme 1 on page 6*). This powers off the kernel circuitry, reducing power consumption to almost zero. Only the backup circuitry remains powered by the LPVREG (Low Power Voltage Regulator).

This mode is particularly important for applications requiring very low consumption even when a rise in temperature occurs: there is almost no dispersion in temperature because the leakage currents are very low (most of the digital part is not powered).

The contents of the registers and SRAM are lost. The backup circuitry can be used to wake-up the microcontroller. Some backup registers also remain powered and can be used to store some parameters. This backup circuitry consists of:

- RTC counter and alarm mechanism
- Wake-up logic
- Backup registers (8 bytes)

The Wake-up logic switches the power to the kernel back on. The kernel is kept under RESET until the internal voltage is correctly regulated; at this point the interface between the kernel and Backup block is reconnected, and the CPU restarts from its RESET sequence.

STANDBY mode entry procedure

Use the following procedure to enter STANDBY mode:

1. First select STANDBY Mode by programming the LPMC[1:0] bits in the MRCC_PWRCTRL register.
2. Execute the Low Power Bit Writing Sequence as described in *Section 1.8.1: Low power bit writing sequence on page 14*.

This procedure is performed by the function of the library described in the *Section 2.7: MRCC_EnterSTANDBYMode on page 31*

The Low Power Mode Control Parameters (LP_PARAM bits in the MRCC_PWRCTRL) have no effect in STANDBY mode.

To enter STANDBY mode, the WKPF bit must be reset. Otherwise, the STANDBY mode entry procedure will be ignored and program execution will continue.

The user program should also take into account the case of a wake-up event occurring during the STANDBY mode entry procedure: in this case the STANDBY mode entry procedure will be ignored and program execution will continue.

Consumption in STANDBY mode

All the peripherals are disabled except the LPVREG (and its associated RSM). The RTC and its clock source (OSC32K or LPOSC) can be either kept disabled or activated.

If the RTC is disabled, the consumption is reduced to the static consumption of the Low Power Regulator (and the RSM) which is typically 20µA. The leakage currents due to the technology are negligible and there is almost no dispersion in temperature (most of the digital part is not powered)

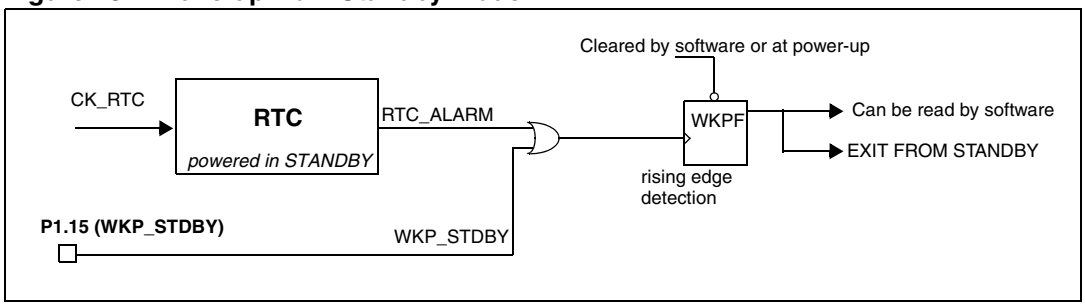
Wake-up events

Wake-up from STANDBY mode can only be performed by:

- A reset
- A rising edge RTC alarm event. See *Figure 10: Wake-up from Standby mode on page 23*
- A rising edge on the WKP_STDBY pin

Refer to *Table 5: STANDBY mode functionality on page 25*.

Figure 10. Wake-up from Standby mode



After wake-up from STANDBY mode, program execution restarts in the same way as after a RESET (the vector reset will be fetched). However, if a wake-up event occurs during the STANDBY entry procedure, the STANDBY mode entry procedure is ignored and the program execution continues. The user program should take into account the case of a wake-up event occurring during the STANDBY mode entry procedure.

Some status flags helps software to determine if it is resuming from RESET or from STANDBY mode and if STANDBY is exited because of a wake-up event or because of an external reset.

Refer to the *Reference manual* for further information on the status bits STDBF and WKPF in the status register (MRCC_RFSR)

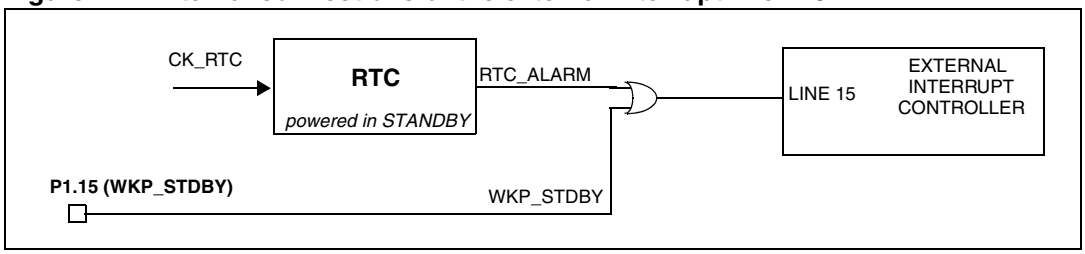
The RTC can wake-up the MCU from STANDBY mode at each RTC alarm event, independently from external interrupt line #15 (no need to configure it). Note that the RTC Second or Overflow interrupts do not wake-up the MCU from STANDBY.

The condition which wakes-up the MCU from STANDBY is a rising edge of the output of an OR gate between the P1.15 input pin and the RTC alarm event, see *Figure 11: Internal connections of the external interrupt line #15 on page 24*.

Consequently, to wake-up from STANDBY by an RTC alarm, you must:

- Apply low level to P1.15 pin.
- Configure the RTC to generate the RTC alarm.

Figure 11. Internal connections of the external interrupt line #15



Wake-up latency

The latency is the same as when resuming from reset.

I/O states in STANDBY mode

In STANDBY mode, all GPIOs are configured in high impedance except the WKP_STDBY pin which is kept in input mode.

All other pins are disabled (XT1, XT2, XRTC1/2, USB, etc.) except the NRSTIN and NRSTOUT pins which are still functional.

Debug mode

If using the debug features of the ARM7TDMI-S, and the current application puts the MCU in STANDBY mode, the debug connection will be lost, because the ARM7TDMI-S core is no longer powered.

Table 5. STANDBY mode functionality

Feature		Functionality in STANDBY mode	
		RTC enabled	RTC disabled
RTC clock source		LPOSC or OSC32K	None
Voltage Regulators		LPVREG	
OSC4M oscillator		Not powered	
PLL + Clock Detector (CKD)		Not powered	
Flash memory		Not powered	
Wakeup resources	NRSTIN	Yes	
	External Interrupts	WKP_STDBY pin	
	RTC Alarm	Yes	No
	Clock Detector	No	
Wake-up latency due to:		MVREG + FLASH + OSC4M	
Clock configuration state after wake-up		Reset state	

1.8.7 Auto-Wake-Up (AWU) from low power mode

The RTC can be used to wake-up the MCU from Low Power Mode without depending on an external interrupt (Auto-Wake-Up mode). The RTC provides a programmable time base for waking-up from STOP or STANDBY mode at selectable intervals (using the alarm event). For this purpose, two alternative RTC clock sources can be selected by programming the CKRTCSEL[1:0] bits in the MRCC_PWRCTRL register:

- **Low Power 32.768kHz crystal oscillator (OSC32K).**

  This clock source provides a precise time base with very low power consumption (less than 1µA added consumption in typical conditions)

- **Internal Low Power RC Oscillator (LPOSC)**

  This clock source has the advantage of saving the cost of the 32.768kHz crystal. This internal RC Oscillator is designed to add minimum power consumption (2µA added consumption in typical conditions).

2 STR750 library low power mode functions

2.1 MRCC_CKSYSConfig

Function Name	MRCC_CKSYSConfig
Function Prototype	`ErrorStatus MRCC_CKSYSConfig(u32 MRCC_CKSYS, u32 MRCC_PLL)`
Behavior Description	Configures the system clock (CK_SYS).
Input Parameter1	MRCC_CKSYS: specifies the clock source used as system clock. Refer to section "*MRCC_CKSYS*" for more details on the allowed values of this parameter.
Input Parameter2	MRCC_PLL: specifies the PLL configuration. Refer to section "*MRCC_PLL*" for more details on the allowed values of this parameter.
Output Parameter	None
Return Parameter	An ErrorStatus enumeration value: – SUCCESS: Clock configuration succeeded – ERROR: Clock configuration failed
Required preconditions	To use the RTC clock as system clock, the RTC clock source must be previously configured using *MRCC_CKRTCConfig()* function.
Called functions	None

MRCC_CKSYS

To select the system clock, use one of the following values:

MRCC_CKSYS	Meaning
MRCC_CKSYS_FREEOSC	Internal VCO of the PLL configured in free running mode used as system clock
MRCC_CKSYS_OSC4M	4MHz main oscillator (OSC4M) used as system clock
MRCC_CKSYS_OSC4MPLL	4MHz main oscillator (OSC4M) followed by PLL used as system clock
MRCC_CKSYS_RTC	RTC clock used as system clock

MRCC_PLL

To configure the PLL, use one of the following values:

MRCC_PLL	Meaning
MRCC_PLL_Disabled	PLL disabled
MRCC_PLL_NoChange	No change on PLL configuration
MRCC_PLL_Mul_12	Multiplication by 12
MRCC_PLL_Mul_14	Multiplication by 14
MRCC_PLL_Mul_15	Multiplication by 15
MRCC_PLL_Mul_16	Multiplication by 16

Note: When fetching from internal Flash the max System Cock (CK_SYS) frequency is 60MHz (64MHz when fetching from SRAM).

The table below describes the allowed combination of ***MRCC_CKSYS*** and ***MRCC_PLL*** parameters:

		MRCC_CKSYS			
		MRCC_CKSYS_FREEOSC	MRCC_CKSYS_OSC4M	MRCC_CKSYS_OSC4MPLL	MRCC_CKSYS_RTC
MRCC_PLL	MRCC_PLL_Disabled	X	X	-	X
	MRCC_PLL_NoChange	-	X	-	X
	MRCC_PLL_Mul_12	-	-	X	-
	MRCC_PLL_Mul_14	-	-	X	-
	MRCC_PLL_Mul_15	-	-	X	-
	MRCC_PLL_Mul_16	-	-	X	-

Note: "x:" allowed combination

"-" : not allowed combination

Example:

```
/* Set CK_SYS to 60 MHz */
  if(MRCC_CKSYSConfig(MRCC_CKSYS_OSC4MPLL, MRCC_PLL_Mul_15) == SUCCESS)
  {
    /* Place your code here */
  }
  else
  {
     /* Add here some code to deal with this error */
  }
```

2.2 MRCC_HCLKConfig

Function Name	MRCC_HCLKConfig
Function Prototype	`void MRCC_HCLKConfig(u32 MRCC_HCLK)`
Behavior Description	Configures the AHB clock (HCLK).
Input Parameter	MRCC_HCLK: defines the AHB clock. This clock is derived from the system clock(CK_SYS). Refer to section "*MRCC_HCLK*" for more details on the allowed values of this parameter.
Output Parameter	None
Return Parameter	None
Required preconditions	None
Called functions	None

MRCC_HCLK

To configure the AHB clock, use one of the following values:

MRCC_HCLK	Meaning
MRCC_CKSYS_Div1	AHB clock = CK_SYS
MRCC_CKSYS_Div2	AHB clock = CK_SYS/2
MRCC_CKSYS_Div4	AHB clock = CK_SYS/4
MRCC_CKSYS_Div8	AHB clock = CK_SYS/8

Example:

```
/* Configure HCLK such as HCLK = CK_SYS */
  MRCC_HCLKConfig(MRCC_CKSYS_Div1);
```

2.3 MRCC_CKTIMConfig

Function Name	MRCC_CKTIMConfig
Function Prototype	`void MRCC_CKTIMConfig(u32 MRCC_CKTIM)`
Behavior Description	Configures the TIM clock (CK_TIM).
Input Parameter	MRCC_CKTIM: defines the TIM clock. This clock is derived from the AHB clock(HCLK). Refer to section "*MRCC_CKTIM*" for more details on the allowed values of this parameter.
Output Parameter	None
Return Parameter	None
Required preconditions	None
Called functions	None

MRCC_CKTIM

To configure the TIM clock, use one of the following values:

MRCC_CKTIM	Meaning
MRCC_HCLK_Div1	TIM clock = HCLK
MRCC_HCLK_Div2	TIM clock = HCLK/2
MRCC_HCLK_Div4	TIM clock = HCLK/4
MRCC_HCLK_Div8	TIM clock = HCLK/8

Example:

```
/* Configure CKTIM such as CKTIM = HCLK/2 */
  MRCC_CKTIMConfig(MRCC_HCLK_Div2);
```

2.4 MRCC_PCLKConfig

Function Name	MRCC_PCLKConfig
Function Prototype	`void MRCC_PCLKConfig(u32 MRCC_PCLK)`
Behavior Description	Configures the APB clock (PCLK).
Input Parameter	MRCC_PCLK: defines the APB clock. This clock is derived from the TIM clock(CK_TIM). Refer to section "*MRCC_PCLK*" for more details on the allowed values of this parameter.
Output Parameter	None
Return Parameter	None
Required preconditions	None
Called functions	None

MRCC_PCLK

To configure the APB clock, use one of the following values:

MRCC_PCLK	Meaning
MRCC_CKTIM_Div1	APB clock = CKTIM
MRCC_CKTIM_Div2	APB clock = CKTIM/2

Example:

```
/* Configure PCLK such as PCLK = CKTIM */
  MRCC_PCLKConfig(MRCC_CKTIM_Div1);
```

2.5 MRCC_EnterWFIMode

Function Name	MRCC_EnterWFIMode
Function Prototype	`void MRCC_EnterWFIMode(u32 MRCC_WFIParam)`
Behavior Description	Enters WFI mode.
Input Parameter	MRCC_WFIParam: specifies the WFI mode control parameters. Refer to section "*MRCC_WFIParam*" for more details on the allowed values of this parameter.
Output Parameter	None
Return Parameter	None
Required preconditions	None
Called functions	None

MRCC_WFIParam

To select the WFI mode control parameters, use one of the following values:

MRCC_WFIParam	Meaning
MRCC_WFIParam_Default	DMA disabled and FLASH enabled in WFI mode
MRCC_WFIParam_DMAEnabled[1)]	Enable DMA in WFI mode
MRCC_WFIParam_FLASHOff[1)]	Disable FLASH in WFI mode

Note: 1 These two parameters can be used together.

Example:

```
/* Enter WFI mode with DMA enabled */
  MRCC_EnterWFIMode(MRCC_WFIParam_DMAEnabled);
```

2.6 MRCC_EnterSTOPMode

Function Name	MRCC_EnterSTOPMode
Function Prototype	`void MRCC_EnterSTOPMode(u32 MRCC_STOPParam)`
Behavior Description	Enters STOP mode.
Input Parameter	MRCC_STOPParam: specifies the STOP mode control parameters. Refer to section "*MRCC_STOPParam*" for more details on the allowed values of this parameter.
Output Parameter	None
Return Parameter	None
Required preconditions	None
Called functions	None

MRCC_STOPParam

To select the STOP mode control parameters, use one of the following values:

MRCC_STOPParam	Meaning
MRCC_STOPParam_Default	OSC4M, FLASH and MVREG enabled in STOP mode
MRCC_STOPParam_OSC4MOff[1)]	Disable OSC4M and PLL in STOP mode
MRCC_STOPParam_FLASHOff[1)]	Disable FLASH in STOP mode
MRCC_STOPParam_MVREGOff[2)]	Disable main voltage regulator in STOP mode

Note: 1 These two parameters can be used together.

2 If this parameter is selected, OSC4M and FLASH are also disabled.

Example:

```
/* Enter STOP mode with OSC4M and FLASH OFF */
  MRCC_EnterSTOPMode(MRCC_STOPParam_FLASHOff | MRCC_STOPParam_OSC4MOff);
```

2.7 MRCC_EnterSTANDBYMode

Function Name	MRCC_EnterSTANDBYMode
Function Prototype	`void MRCC_EnterSTANDBYMode(void)`
Behavior Description	Enters STANDBY mode.
Input Parameter	None
Output Parameter	None
Return Parameter	None
Required preconditions	None
Called functions	None

Example:

```
/* Enter STANDBY mode */
  MRCC_EnterSTANDBYMode();
```

2.8 MRCC_PeripheralClockConfig

Function Name	MRCC_PeripheralClockConfig

Function Prototype	`void MRCC_PeripheralClockConfig(`
	`  u32 MRCC_Peripheral,`
	`  FunctionalState NewState)`

Behavior Description	Enables or disables the specified peripheral clock.
Input Parameter1	MRCC_Peripheral: specifies the peripheral to gates its clock. Refer to section "*MRCC_Peripheral*" for more details on the allowed values of this parameter.
Input Parameter2	NewState: new state of the specified peripheral clock. This parameter can be: ENABLE or DISABLE.
Output Parameter	None
Return Parameter	None
Required preconditions	None
Called functions	None

MRCC_Peripheral

To select the peripheral to gates its clock, use a combination of one or more of the following values:

MRCC_Peripheral	Meaning
MRCC_Peripheral_ALL	All peripherals clock
MRCC_Peripheral_EXTIT	EXTIT clock
MRCC_Peripheral_RTC	RTC clock
MRCC_Peripheral_GPIO	GPIO clock
MRCC_Peripheral_UART2	UART2 clock
MRCC_Peripheral_UART1	UART1 clock
MRCC_Peripheral_UART0	UART0 clock
MRCC_Peripheral_I2C	I2C clock
MRCC_Peripheral_CAN	CAN clock
MRCC_Peripheral_SSP1	SSP1 clock
MRCC_Peripheral_SSP0	SSP0 clock
MRCC_Peripheral_USB	USB clock
MRCC_Peripheral_PWM	PWM clock
MRCC_Peripheral_TIM2	TIM2 clock
MRCC_Peripheral_TIM1	TIM1 clock
MRCC_Peripheral_TIM0	TIM0 clock
MRCC_Peripheral_TB	TB clock
MRCC_Peripheral_ADC	ADC clock

Example:

```
/* Enable GPIOs and TB clocks */
  MRCC_PeripheralClockConfig(MRCC_Peripheral_GPIO | MRCC_Peripheral_TB, ENABLE);
```

3 Operating measurements

3.1 Setup board

The STR750 evaluation board (order code STR750-EVAL) and Uniboard (order code UB-QFP100-1414O/) are available for evaluation and testing purposes. Contact your local ST sales office for further details.

3.1.1 Measurement with STR750_EVAL Board

Before starting the measurements, the potentiometer RV2 should be removed from the boards MB469 and MB469 rev. B.

For correct usage instructions, refer to the *User Manual STR750_EVAL Board, UM0268*. With this board measurements can only be made at 3.3V (Single supply) or 3.3V and 1.8V (Dual supply).

Consumption measurements

- **In Single Supply:** measurements can be made by replacing JP3 by an ampermeter. JP4, JP5, JP6 and JP13 must not be fitted.
- **In Dual Supply:** measurements can be made by replacing JP3 by an ampermeter, setting jumper JP4, and replacing JP5 and JP6 by an ampermeter to measure the consumption on V18BKP and V18 respectively. JP13 must not be fitted.
- **Caution**: the standby mode is not available in dual supply.

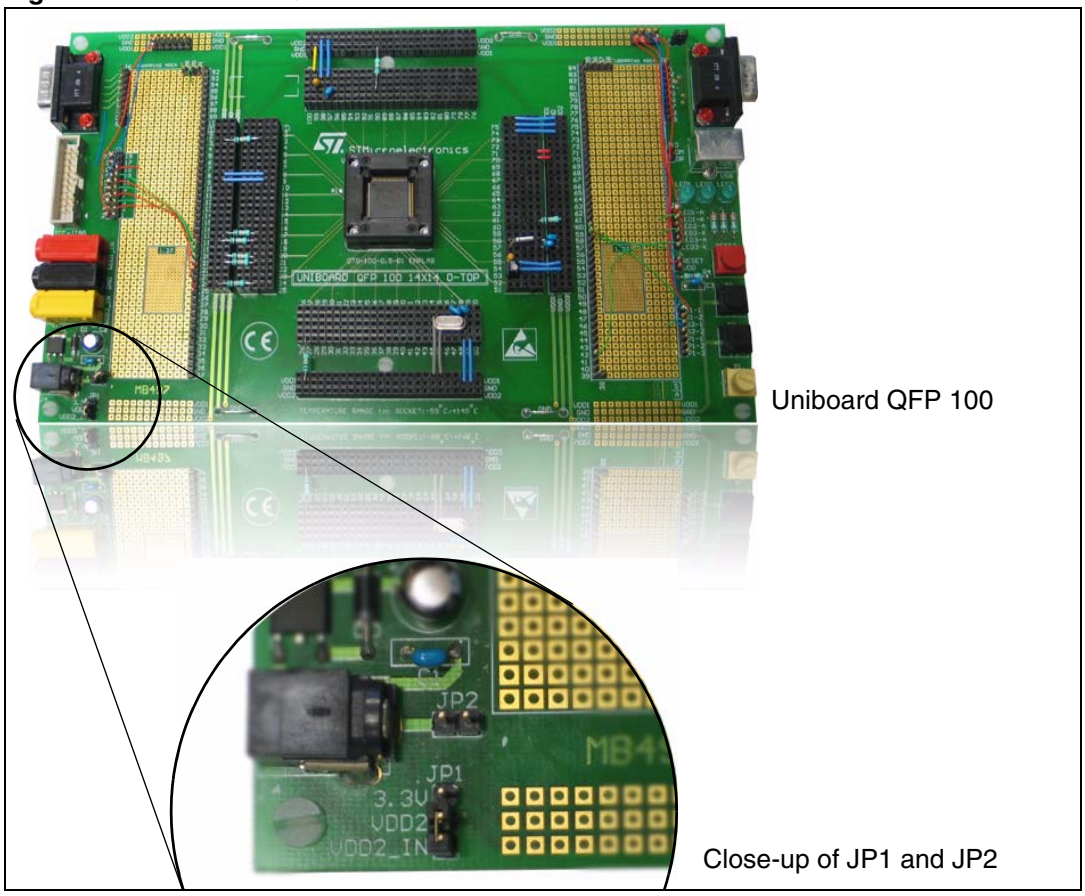
Wake-up time measurements

- **Definition:**

  Two wake-up time measurements can be made. The STR75x can be restarted either from the low-power mode on the internal free oscillator (FREEOSC) or on the main oscillator, restarting it if necessary (OSC4M PLL).
  - FREEOSC: Time between wake-up trigger event and the first fetched instruction by the core.
  - OSC4M PLL: Time between the wake-up trigger event and the execution of the instruction once the clock has been started by either the library function or in the interrupt handler routine.
- **Measurement:** Place the oscilloscope on:
  - Pins P1.15 (WKP_STDBY pin) and P0.16 to measure the wake-up-time in a FREEOSC configuration. This time is measured between the rising edge on P1.15 and a falling edge on P0.16.
  - Pins P1.15 (WKP_STDBY pin) and P2.18 to measure the wake-up-time in a OSC4M PLL configuration. This time is measured between the rising edge on P1.15 and a falling edge on P2.18.

3.1.2 Measurements with Uniboard QFP 100 in single supply (3.3V or 5V)

Figure 12. Uniboard QFP 100



Before making measurements, the board must be configured as describe below:

- First JP2 should be removed and JP1 set between VDD and VDD2_IN
- Don't use J4 to power supply the Board
- These external components are needed to update uniboard QFP100

Table 6. Bill of material

Components	Value/Reference	Quantity
UNIBOARD QFP 100 14x14	MB497	1
STR750 Chip	STR750FL2T6	1
Capacitor	1µF	2
Capacitor	10µF	1
Capacitor	33nF	1
Capacitor	22pF	4
Resistor	10K	9
Quartz	4MHz	1

Components	Value/Reference	Quantity
Quartz	32 KHz	1
Connector	HE-20	1

This table describes how the STR750 chip can be powered:

Table 7. Power supply

Signal Name	pin n° to VDD1	pin n° to GND
TEST		9
VSS_IO		10
VDD_IO	44	
VDDA_PLL	45	
VSS_IO		48
VSSA_PLL		49
VSS18		53
VSSBKP		54
VDD_IO	69	
VDDA_ADC	70	
VSSA_ADC		73
VSS_IO		74
VREG_DIS		75
VSS18		97
VSS_IO		98
VDD_IO	99	

This table describes how the STR750 chip can be decoupled

Table 8. Decoupling capacitor

SIgnal Name	Pin	Connection
V18REG	52	Decoupling with 10µF capacitor and VSS18 pin 53.
V18_BKP	55	Decoupling with 1µF capacitor and VSSBKP pin 54.
V18	96	Decoupling with 33nF capacitor and VSS18 pin 97.
VDD_IO	99	Decoupling with 1µF capacitor and VSS_IO pin 98.

This table describes how to connect the Pull-up and Pull-down resistors:

Table 9. Pull-up and pull-down resistor

Signal Name	Pin	Connection
Boot 0	4	Pull-down to **GND** via 10K resistor
Boot 1	27	Pull-down to **GND** via 10K resistor

Signal Name	Pin	Connection
JTMS	18	Pull-up to **VDD1** via 10K resistor
JTCK	19	Pull-down to **GND** via 10K resistor
JTDI	21	Pull-up to **VDD1** via 10K resistor
NJTSRST	22	Pull-up to **VDD1** via 10K resistor
RTCK	25	Pull-down to **GND** via 10K resistor
WKP_STDBY	60	Pull-down to **VDD1** via 10K resistor
P1.05	90	Pull-up to **VDD1** via 10K resistor

This table describes how to connect the JTAG connector:

Table 10. JTAG connection (J3)

Signal Name	Pin of HE20 connector	Connection to the Chip
VTref	1	VDD1
GND	2	VDD1
NJTSRST	3	Pin 22
GND	4	GND
JTDI	5	Pin 21
GND	6	GND
JTMS	7	Pin 18
GND	8	GND
JTCK	9	Pin 19
GND	10	GND
RTCK	11	Pin 25
GND	12	GND
JTDO	13	pin 20
GND	14	GND
RESET_IN	15	pin 59
GND	16	GND
DBGRQ	17	No connect
GND	18	GND
DBGACK	19	No connect
GND	20	GND

This table describes how to connect the Reset Pin and Oscillators to the Chip

Table 11. Other pin connections

Signal Name	Pin	Connection
RESET_IN	59	RESET J8
X1, X2 MCU	46,47	4MHz quartz with two capacitor 22pF to GND see Figure5.
X1, X2 RTC	56,57	32KHz quartz with two capacitor 22pFto GND, see Figure6.

Figure 13. Main Clock circuit

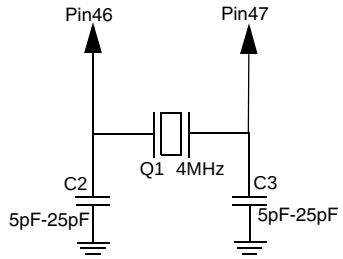


Figure 14. Real time clock circuit

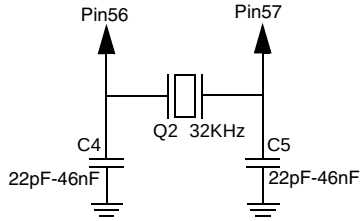


Table 12. Other pin connections for the uniboard

Connector Name	Pin	Connection
J8	VDD	VDD1
J7	S1-1	pin 60 of the chip
J7	S1-2	VDD1
J7	S2-1	pin 90 of the chip
J7	S2-2	GND
J9	LED1-A	VDD1
J9	LED1-K	pin 32
J9	LED2-A	VDD1
J9	LED2-K	pin 31
J9	LED3-A	VDD1
J9	LED3-K	pin 42

- Put the STR750 in the Socket
- Connect the JTAG link on J2 connector
- Connect the board to the power supply at 3.3V or 5V (GND black connector and VDD1_IN red connector)

Measurement with Uniboard QFP 100 in dual supply (3.3V or 5V and 1.8V)

To make measurements in dual supply mode the board must be configured as described before with the following modifications:

Signal Name	pin n° to VDD1	pin n° to GND	pin n° to VDD2
VREG_DIS	75		
V18REG			52
V18_BKP			55
V18			96

Connect the board to the power supply at 3.3V or 5V (GND black connector and VDD1_IN red connector) and 1.8V to VDD2_IN (Yellow connector).

Consumption measurements

- **In Single Supply:** measurements can be made by placing an ampmeter between the power supply and the VDD1_IN.
- **In Dual Supply:** measurements can be made by placing an ampmeter between the power supply and the VDD1_IN (to determine the consumption at 3.3 V / 5 V) or between the power supply and the VDD2_IN (to determine the consumption at 1.8 V).
- **Caution**: the standby mode is not available in dual supply.

Wake-up time measurements

- **Definition:**

  Two wake-up time measurements can be made. The STR75x can be restarted either from the low-power mode on the internal free oscillator (FREEOSC) or on the main oscillator, restarting it if necessary (OSC4M PLL).

  – FREEOSC: Time between wake-up trigger event and the first fetched instruction by the core.

  – OSC4M PLL: Time between the wake-up trigger event and the execution of the instruction once the clock has been started by either the library function or in the interrupt handler routine.

- **Measurement:** Place the oscilloscope on:

  – Pin 60 (I/O P1.15: WKP_STDBY) and pin 42 (I/O P0.16) to measure the wake-up-time in a FREEOSC configuration. This time is measured between the rising edge on pin 60 and a falling edge on pin 42.

  – Pin 60 (I/O P1.15: WKP_STDBY) and Pin 32 (I/O P2.18) to measure the wake-up-time in a OSC4M PLL configuration. This time is measured between the rising edge on pin 60 and a falling edge on pin 32.

3.2 Software provided with this application note

There are 3 folders in the ZIP file provided with this application note:

3.2.1 Standby mode

This example shows how to put the system in STANDBY mode and how to wake-up again using external RESET, WKP_STDBY pin or RTC Alarm (if RTC_ON is set in *75x_conf.h* file).

Of course this software used the standard STR750 library and this example show how to used the function `MRCC_EnterSTANDBYMode()`. Refer to *Chapter 2.7: MRCC_EnterSTANDBYMode on page 31* for further information.

- **Choose RTC using clock:**

  Comment the following line in the *75x_conf.h* file, if the RTC clock is not required in STANDBY mode in the application:

  ```
  #define RTC_ON
  ```

  In the associated software, the system clock is set to 60 MHz, the EXTIT line7 is configured to generate an interrupt on falling edge.

- **If RTC ON is set** (by uncommenting the line `#define RTC_ON` in *75x_conf.h* file):

  The RTC is programmed to generate an interrupt each 1 second. In the RTC interrupt handler, an LED connected to P2.18 pin is toggled, and an LED connected to P0.16 is on. This is used to indicate whether the MCU is in STANDBY or RUN mode.

  – When a falling edge is detected on EXTIT line7, an interrupt is generated. In the EXTIT handler routine the RTC is configured to generate an alarm event after 10 seconds. Then the system enters STANDBY mode causing the P2.18 pin to stop toggling.

  – A rising edge on WKP_STDBY pin or an external RESET will wake-up the system from STANDBY. If within 10 seconds neither rising edge on WKP_STDBY nor external RESET are generated, the RTC alarm wakes-up the system. After that

the LED connected to P0.16 is switched on and the VCO of the PLL (FREEOSC) supplies the system clock.

After The clock configuration (system clock set to 60 MHz), the P2.18 pin restarts toggling.

- **If RTC ON is not set** (by commenting the line `#define RTC_ON` in *75x_conf.h* file):

  An LED connected to P2.18 pin is on, to indicate the MCU is in RUN mode.

  – When a falling edge is detected on EXTIT line7, an interrupt is generated, the system enters STANDBY mode causing the P2.18 pin to switch off.

  – A rising edge on WKP_STDBY pin or an external RESET will wake-up the system from STANDBY. Then the LED connected to P0.16 is switched on and the VCO of the PLL(FREEOSC) supplies the system clock.

  After The clock configuration (system clock set to 60 MHz), the P2.18 pin is switched on.

- **Directory contents:**
  – *75x_conf.h* : Library Configuration file
  – *75x_it.c* : Interrupt handlers
  – *main.c* : Main program
- **Hardware environment:**
  – Connect an LED to pin P0.16 (LD3 on STR75x-EVAL board).
  – Connect an LED to pin P2.18 (LD4 on STR75x-EVAL board).
  – Connect a push-button to the WKP_STDBY pin (P1.15) (the wake-up push-button on the STR75x-EVAL board).
  – Connect a push-button to EXTIT line7 pin (P1.05) (the key push-button on the STR75x-EVAL board)
- **How to use it:**

  In order to make the program work, you must do the following :

  a) Create a project and setup all your toolchain's start-up files
  b) Compile the directory content files and required Library files :

  75x_lib.c
  75x_rtc.c
  75x_gpio.c
  75x_mrcc.c
  75x_eic.c
  75x_extit.c
  75x_cfg.c

  c) Link all compiled files and load your image into Flash
  d) Run the example

Note: If the line `#Define RTC_ON` is modified through commenting or uncommenting in the file 75x_conf.h, the project must be rebuilt

3.2.2 STOP mode

This example shows how to put the system in STOP mode and then to wake-up from this mode using an external interrupt, an RTC alarm or an NCKD (no clock detected) interrupt.

- **Choose the Board:**

  The choice of board to be used to perform the measurements is defined in the *75x_conf.h* file. Comment/uncomment the appropriate lines, depending on the chosen board:

```
#define STR750_EVAL
//#define Uniboard
```

- **Choose the STOP mode:**

  Uncomment one of these lines in the *75x_conf.h* file to select the STOP mode configuration:

```
//#define MRCC_STOP_Default
//#define STOP_MVREG_Off
//#define STOP_OSC4M_Off
//#define STOP_FLASH_Off
//#define STOP_FLASH_PLL_Off
#define STOP_FLASH_OSC4M_Off
```

- **Choose RTC using clock:**
  Comment the following line in the *75x_conf.h* file if the RTC is not used to wake-up from STOP mode:

```
#define RTC_ON
```

  In the associated software, the system clock is set to 60 MHz and an interrupt is generated if no clock is present on OSC4M.

- **If the RTC is on** (by uncommenting the line `#define RTC_ON` in the *75x_conf.h* file):

  The EXTIT line7(P1.05) and EXTIT line15 are configured to generate interrupts on a rising edge and the RTC is programmed to generate an interrupt each 1 second. The EXTIT line15 is shared between the RTC_Alarm event and the WKP_STDBY pin (P1.15).

  In the RTC interrupt handler, an LED connected to P2.18 pin is toggled and used to indicate whether the MCU is in STOP or RUN mode.

  The system enters mode as follows:

  a) After system power on, an LED connected to P2.18 pin is toggled each 1 second
  b) After 5 seconds the RTC is configured to generate an alarm event after a delay of 10 seconds
  c) The system then enters STOP mode causing the P2.18 pin to stop toggling.

  The system exits STOP mode as follows:

  a) To wake-up from STOP mode either a rising edge can be applied on EXTIT line15, EXTIT line7 or the 4MHz external Quartz oscillator can be disconnected.
  b) If within 10 seconds none of those actions are performed, the RTC alarm wakes-up the system.
  c) The LED connected to P0.16 is then switched on and the VCO of the PLL (FREEOSC) supplies the system clock.

d) After the clock configuration (system clock set to 60 MHz), the P2.18 pin restarts toggling and in 5 seconds the system enters STOP mode again before exiting in the manner described above.

This behavior is repeated in an infinite loop.

- **If RTC is not on** (by commenting the line `#define RTC_ON` in the *75x_conf.h* file): The EXTIT line7(P1.05) and EXTIT line15 (WKP_STDBY pin P1.15) are configured to generate interrupts on a rising edge.

  An LED connected to P2.18 pin is switched on to indicate when the MCU is in RUN mode.

  The system enters STOP mode as follows:

  a) After system power on, an LED connected to P2.18 pin is switched on.

  b) When a falling edge is detected on EXTIT line7 the system enters STOP mode causing the P2.18 pin to switch off the LED.

  The system exits STOP mode as follows:

  a) To wake-up from STOP mode, either a rising edge can be applied on EXTIT line15, EXTIT line7 or the 4MHz external Quartz oscillator can be disconnected.

  b) The LED connected to P0.16 is switched on and the VCO of the PLL (FREEOSC) supplies the system clock.

  c) After the clock configuration (system clock set to 60 MHz), the LED connected to P2.18 pin is switched on.

When the 4MHz external Quartz oscillator is disconnected, the VCO of the PLL (FREEOSC) supplies the system clock. Once the 4MHz clock has recovered (by connecting the 4MHz Quartz oscillator) the system clock is reconfigured to 60 MHz.

If the system fails to enter STOP mode, a led connected to P2.19 pin is turned on.

- Directory contents:
  - *75x_conf.h* Library Configuration file
  - *75x_it.c* Interrupt handlers
  - *main.c* Main program
- Hardware environment:
  - Connect three LEDS to P016, P2.18 and P2.19 pins (respectively LD3, LD4 and LD5 on STR75x-EVAL board).
  - Connect a push-button to WKP_STDBY pin (P1.15) (the wakeup push-button on STR75x-EVAL board).
  - Connect a push-button to EXTIT line7 pin (P1.05) (the key push-button on STR75x-EVAL board).
  - On STR75x-EVAL board the 4MHz Quartz oscillator is mounted on socket so it is easy to disconnect.
- **How to use it:**

  In order to make the program work, you must do the following :

  a) Create a project and setup all your toolchain's start-up files

  b) Compile the directory content files and required Library files :

  75x_lib.c

  75x_rtc.c

  75x_gpio.c

75x_mrcc.c
75x_eic.c
75x_extit.c
75x_cfg.c

c) Link all compiled files and load your image into Flash
d) Run the example

Note: If the line `#Define RTC_ON` is modified through commenting/uncommenting in the 75x_conf.h file, the project must be rebuilt.

3.2.3 WFI mode

This example shows how to enter the system to WFI mode and wake-up from this mode using an external Interrupt, an RTC Alarm or a NCKD(no clock detected) interrupt.

- **Choose the Board:**

  The chosen board for measurements can be selected in the *75x_conf.h* file. Uncomment the appropriate line:

```
#define STR750_EVAL
//#define Uniboard
```

- **Choose the WFI Mode:**

  Uncomment one of these lines in the *75x_conf.h* file, to choose the required WFI mode for the application:

```
//#define WFI_FLASH_Off
//#define WFI_FLASH_PowerDown
#define WFI_FLASH_On
```

In the associated software, the system clock is set to 64 MHz and an interrupt is generated if no clock is present on OSC4M.

The EXTIT line7(P1.05) and EXTIT line15 are configured to generate interrupts on a rising edge and the RTC is programmed to generate an interrupt each 1 second. The EXTIT line15 is shared between the RTC_Alarm event and the WKP_STDBY pin (P1.15).

In the RTC interrupt handler, an LED connected to P2.18 pin is toggled and used to indicate whether the MCU is in WFI or RUN mode.

The system enters WFI mode as follows:

a) After system is powered on, an LED connected to P2.18 pin is toggled each 1 second
b) After 5 seconds, the RTC is configured to generate an Alarm event with a trigger delay of 10 seconds.
c) The system then enters WFI mode causing the P2.18 pin to stop toggling.

The system exits WFI mode as follows:

a) To wake-up from WFI mode a rising edge can be applied on EXTIT line15, EXTIT line7 or the 4MHz external Quartz oscillator can be disconnected.
b) If within 10 seconds none of those actions are performed, the RTC Alarm wakes the system up.
c) The P2.18 pin then restarts toggling and after 5 seconds the system enters WFI mode again before exiting in the manner described above. This behavior is repeated in an infinite loop.

When the 4MHz external Quartz oscillator is disconnected, the VCO of the PLL (FREEOSC) supplies the system clock. Once the 4MHz clock has recovered (by connecting the 4MHz Quartz oscillator) the system clock is reconfigured to 64 MHz.

If the system fails to enter WFI mode, an LED connected to P2.19 pin is turned on.

- **Directory contents**
  - *75x_conf.h* : Library Configuration file
  - *75x_it.c* : Interrupt handlers
  - *main.c* : Main program
- **Hardware environment**
  - Connect two LEDs to P2.18 and P2.19 pins (respectively LD4 and LD5 on STR75x-EVAL board).
  - Connect a push-button to WKP_STDBY pin (P1.15) (Wakeup push-button on STR75x-EVAL board).
  - Connect a push-button to EXTIT line7 pin (P1.05) (Key push-button on STR75x-EVAL board).
  - On the STR75x-EVAL board the 4MHz Quartz oscillator is mounted on socket so it is easy to disconnect it.
- **How to use it**

  In order to make the program work, you must do the following :

  a) Create a project and setup all your toolchain's start-up files

  b) Compile the directory content files and required Library files :

  75x_lib.c

  75x_rtc.c

  75x_gpio.c

  75x_mrcc.c

  75x_eic.c

  75x_extit.c

  75x_cfg.c

  c) Link all compiled files and load your image into Flash

  d) Run the example

3.3 Measurement and typical value

Table 13 below shows the low power mode configurations for the examples described in *Section 3.2 on page 39*.

Table 13. Low power mode configurations for example measurements

Example mode name	Mode	MVREG	OSC4M	PLL	FLASH	RTC
Standby	STANDBY from Flash	OFF	OFF	OFF	OFF	OFF
Standby (with `Define RTC_ON`)		OFF	OFF	OFF	OFF	ON 32k
STOP_MVREG_Off	STOP from Flash	OFF	OFF	OFF	OFF	OFF
STOP_MVREG_Off (with `Define RTC_ON`)		OFF	OFF	OFF	OFF	ON 32k
STOP_FLASH_OSC4M_Off		ON	OFF	OFF	OFF	OFF
STOP_OSC4M_Off		ON	OFF	OFF	ON	OFF
STOP_FLASH_PLL_Off		ON	ON	OFF	OFF	OFF
STOP_FLASH_Off		ON	ON	ON	OFF	OFF
STOP_Default		ON	ON	ON	ON	OFF
WFI_FLASH_Off	WFI from Flash	ON	ON	ON	OFF	ON 32k
WFI_FLASH_PowerDown [1)]		ON	ON	ON	P DOWN	ON 32k
WFI_FLASH_On		ON	ON	ON	ON	ON 32k

Note: 1 For this configuration, it is forbidden to put the Flash into BURST mode.

Table 14 shows the typical consumption and timing values measured with these mode configurations.

Table 14. Typical consumption and timing measurements

Example mode name	Consumption with TA= +25°C (µA)				Wake-up times (µs) [2)]	
	Single Supply 3.3V Idd(V33)	Dual Supply 3.3V Idd(V33)/ (V18)	Single Supply 5V Idd(V33)	Dual Supply 5V Idd(V33)/ (V18)	OSC4M PLL	FREEOSC
Standby	**10.8**	NA	14.9	NA	3250	85
Standby (with `Define RTC_ON`)	14.1	NA	19.6	NA	3250	85
STOP_MVREG_Off	**11.9**	0.3 / 2.4	15.4	0.8/ 2.4	3250	38
STOP_MVREG_Off (with `Define RTC_ON`)	15.8	3.0 / 2.6	21.1	5.3 / 2.6	3250	40

Table 14. Typical consumption and timing measurements

Example mode name	Consumption with TA= +25°C (µA)				Wake-up times (µs) 2)	
	Single Supply 3.3V Idd(V33)	Dual Supply 3.3V Idd(V33)/ (V18)	Single Supply 5V Idd(V33)	Dual Supply 5V Idd(V33)/ (V18)	OSC4M PLL	FREEOSC
STOP_FLASH_OSC4M_Off	110	0.3/ 2.4	118	0.8 / 2.4	3380	40
STOP_OSC4M_Off	618	0.3/ 507	626	0.8/ 507	3290	40
STOP_FLASH_PLL_Off	**1104**	878/ 114	1020	740/ 114	**200**	40
STOP_FLASH_Off	5947	1865/ 3860	5849	1782/ 3860	9.5	NA
STOP_Default	6458	1865/ 4352	6427	1782/ 4352	9	NA
WFI_FLASH_Off	21940	1758/ 18466	21950	1666/ 18466	10.6	NA
WFI_FLASH_PowerDown 1)	22480	1758/ 18914	22440	1666/ 18914	3.08	NA
WFI_FLASH_On	34180	1758/ 28130	34160	1666/ 28130	3.08	NA

Note: 1 For this configuration, it is forbidden to put the Flash into BURST mode.

2 Measurements for the wake-up time have been made on pin 1.15. Refer for to Section 3.1 on page 33 for further details.

4 Conclusion

Before choosing clock and regulator configuration it is important to verify the application requirements not only for the consumption but also in terms of the wake-up time, the peripheral availibilty and whether the RAM contents need to be preserved until wake-up. For example, in STOP mode the SRAM content is preserved, however in Standby mode, SRAM content is lost.

There are therefore compromises to be made between firstly the wake-up time and the consumption, and secondly the preservation of the SRAM contents and the peripheral availability.

For example, the measured mode *STOP_FLASH_PLL_Off* shown in *Table 14 on page 45* uses roughly double the power consumption as the *STOP_OSC4M_Off* mode, however it is able to react faster by having a much reduced wake-up time.

5 Revision history

Table 15. Document revision history

Date	Revision	Changes
26-Mar-2007	1	Initial release.
09-Jul-2007	2	Connections between V_{BACKUP} and V_{CORE} in *Figure 1*, *Figure 5* updated