RTC/API setup for wake-up in SPC58 family

Introduction

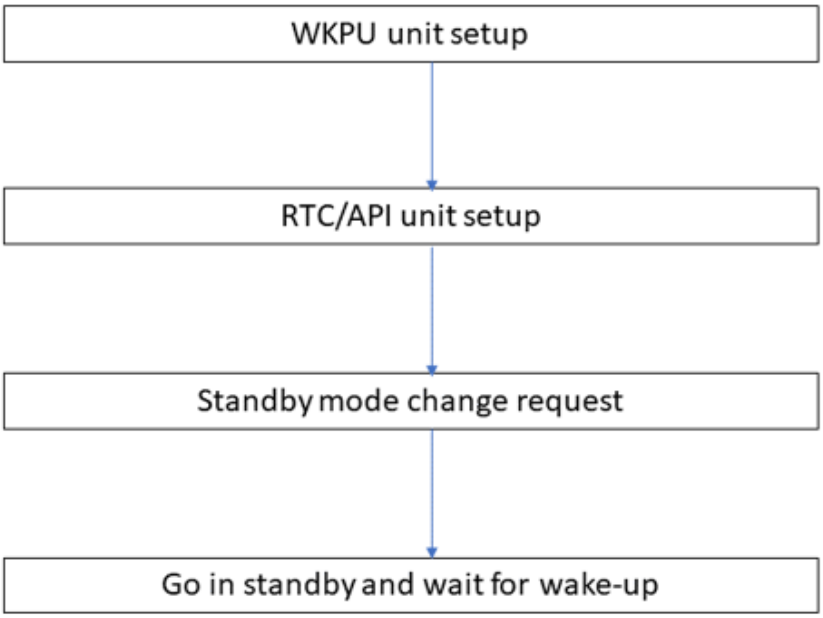
The aim of this document is to give recommendations for hardware designers using STMicroelectronics SPC58 lines devices for the correct usage of real-time clock / autonomous periodic interrupt (RTC/API) for wake-up purposes, once in standby.

Particular attention is dedicated on clock source selection, as RTC can be set up with IRC, LPRC and SXOSC.

Note: Refer to the device reference manual for current availability of clock sources.

The basic flow of the correct device setup for the standby and wake-up from RTC/API is the one in Figure 1. Standby/wake-up flow.

Figure 1. Standby/wake-up flow



1 WKPU setup for waking-up via RTC/API

Wake-up unit needs to be programmed for waking-up the system, once in standby.

One of the possible wake-up sources is from RTC/API.

To accomplish the setup, the user has to program the relative RTC/API bit in:

- The wake-up rising edge event register (WIREER)
- The wake-up request enable register (WRER)
- To clean the wake-up status flag register (WISR)

Optionally, but recommended, the wake-up filter enable register (WIFER) can be programmed.

WKPU bit mask for each register is the following:

- Bit 31 is for API
- Bit 30 is for RTC

Refer to the relevant device reference manual for all the details about the WKPU unit.

1.1 Example of WKPU setup

The following is an example of the code the user can use for programming the WKPU:

```
WKPU.WIFER.B.IFEx = 1; // configure the glitch filter
WKPU.WIREER.B.IREEx = 1; // configure the rising edge
WKPU.WRER.B.WREx = 1; // enable the wake-up request
WKPU.WISR.B.EIFx = 1; // clean the wake-up status
```

Where “x” is 0 for API, 1 for RTC

Note: The devices in SPC58 family implement one instance of WKPU IP. Only devices of SPC58 H line support two WKPU units, so the user needs to distinguish them by programming the proper one they want to use. Refer to proper reference manual.

1.2 Setup of API in the WKPU unit

In the SP58 family the API source is used for timing the smart standby wake-up unit (SSWU) and so it cannot be used as wake-up source by STANDBY.

The devices of SPC58 2B line are the only exception inside the SPC58 family because they do not support SWWU IP and the wake-up is possible by API wake up event.

Note: bit[31] of WKPU registers in all the other devices is reserved for SSWU wake-up, not for API.

2 RTC/API setup

The unit needs to be programmed before going into standby, in order to let the counter start and to trigger when the comparison matches.

There are few steps for the unit setup:

- Clean the CNTEN bit[0] in the control register (RTCC)
- Clean the RTCIF/APIF bits in the status register (RTCS)
  - Use the RTCIF, RTCVAL in case of RTC setup; use APIF, APIVAL in case of API setup
- Set the counter value in the RTC/API compare value register (RTCVAL, APIVAL)
- Set the clock source in the control register (RTCC) CLKSEL bit[18:19]
- Set, if needed, clock divisors using control register (RTCC) DIV512EN bit[20] and DIV32EN bit[21]
- Set, in case of API, the control register (RTCC) APIEN bit[16]
- Set CNTEN bit[0] in the control register (RTCC)

2.1 CLKSEL setup in the control register (RTCC)

Features of the RTC/API include three selectable counter clock sources:

- LPRC prescaled by 8 (128 kHz) low-power RC oscillator
- SXOSC (32 kHz) slow external crystal
- IRCOSC (16 MHz) internal RC oscillator

CLKSEL bit[18:19] provides the possibility to choose one of those three sources, see next table.

Table 1. CLKSEL setup in RTCC register

	Bit18	Bit19
LPRC	0	1
SXOSC	0	0
IRCOSC	1	0
RESERVED	1	1

2.2 Clock source status before going into standby

Some considerations need to be reported for the three clock sources.

2.2.1 IRCOSC

IRCOSC is always enabled, also in reset.

2.2.2 SXOSC

SXOSC is disabled by default and needs to be enabled by the user before using it as RTC clock source.

To accomplish this the user has to setup the SXOSC_CTL register.

Once enabled, the SXOSC will be available until the next device reset.

Refer to the device reference manual for the setup details.

2.2.3 LPRC

LPRC setup needs to be differentiated across devices, because there is a subset of them on which the clock is always enabled and another subset on which it needs to be enabled by the user.

In some circumstances, the difference depends on the device's cut version.

Refer to the following table for a summary and to the proper reference manual for the details.

Table 2. List of devices and LPRC status

Line	LPRC default status
SPC582B line	OFF except in standby
SPC584B line	OFF except in standby for cut1.0 and cut1.1 Always ON in cut1.2
SPC58xC line	OFF except in standby for cut1.0 and 1.1 Always ON in cut2.0
SPC58xE/G line	OFF except in standby
SPC58xH line	Always ON

Note: SPC58xN not included in the list because it does not provide RTC/API unit.

Note: User is recommended to maintain the LPRC "always enabled" because there could be a possible glitch during the turn-off of the LPRC clock. This is obtained by programming to 0 the MISC_CTRL_REG[RCOSC1M_ENB] bit in the PMC_DIG interface.

This recommendation has been managed by erratum valid for all the devices on which the LPRC can be enabled/disabled by the user.

Next table reports the errata entries for the impacted devices.

Table 3. List of errata reporting the LPRC recommendation

Device name and cut	JTAG_ID	Errata entry
SPC582B cut1.0	0x0014_4041	PS2161
SPC584B cut1.1	0x1014_4041	PS2161
SPC58xC cut1.1	0x1014_2041	PS2161
SPC58xE/G	All cut versions	PS2161

Note: SPC58xH not impacted by erratum.

In this way, the LPRC is maintained enabled in any run mode (also in standby) avoiding the device to disable it after the wake-up.

In fact, the clock, in all the devices, is restored to the previous state once exiting from standby.

So, at the wake-up it will be disabled in case it was maintained off before entering the standby.

Note: In case of LPRC disabling after the wake-up, the RTC/API counter value cannot be read because the unit will be unclocked.

As for SXOSC, LPRC status will restore to default after a device reset.

2.3 Example of programming the RTC/API

The following codes can be used to program the unit for triggering a wake-up once a given time runs over.

Codes are based on the LPRC source clock.

2.3.1 Using RTC compare value register (RTC_RTCVAL)

```
RTC_API.RTCC.B.CNTEN = 0; // Disable the counter
RTC_API.RTCS.B.RTCF = 1; // Clean the interrupt flag
```

```
RTC_API.RTCVAL.R = 0x<value>; // Set the counter value to the desired <value>
RTC_API.RTCC.B.CLKSEL = 0x1; // Set LPRC as RTC clock
RTC_API.RTCC.B.CNTEN = 1; // Enable the counter
```

2.3.2 Using API compare value register (RTC_APIVAL)

```
RTC_API.RTCC.B.CNTEN = 0; // Disable the counter
RTC_API.RTCS.B.APIF = 1; // Clean the interrupt flag
RTC_API.APIVAL.R = 0x<value>; // Set the counter value to the desired <value>
RTC_API.RTCC.B.CLKSEL = 0x1; // Set LPRC as RTC clock
RTC_API.RTCC.B.APIEN = 0x1; // API enabled
RTC_API.RTCC.B.CNTEN = 1; // Enable the counter
```

Appendix A Reference documents

Table 4. Reference documents

Doc name	ID	Title
DM00201129	027949	SPC58 2B Line - 32 bit Power Architecture automotive MCU z2 core 80 MHz, 1 MByte Flash, ASIL-B
DM00400546	030699	SPC58 4B Line - 32 bit Power Architecture automotive MCU z4 core 120 MHz, 2 MBytes Flash, HSM, ASIL-B
DM00215423	028117	SPC58 C Line - 32 bit Power Architecture automotive MCU Dual z4 cores 180 MHz, 4 MBytes Flash, HSM, ASIL-B
DM00148989	027214	SPC58 E/G Line - 32 bit Power Architecture automotive MCU Triple z4 cores 180 MHz, 6 MBytes Flash, HSM, ASIL-D
DM00449842	031241	SPC58 H Line - 32 bit Power Architecture automotive MCU Triple z4 cores 200 MHz, 10 MBytes Flash, HSM, ASIL-D

Revision history

Table 5. **Document revision history**

Date	Revision	Changes
02-Feb-2022	1	Initial release.

Contents