Solution to the Issue of LPTIM3 Failing to wake up STM32U545 from Stop2 Mode

Nianhang Hu

May 1, 2025

1 Background

During the development of STM32U545, it was found that LPTIM3 was unable to wake the MCU from STOP2 low-power mode. According to the reference manual RM0456. LPTIM3 should have this capability. Thus, it was initially suspected that the issue was related to configuration or usage.

2 Problem Analysis and Reproduction

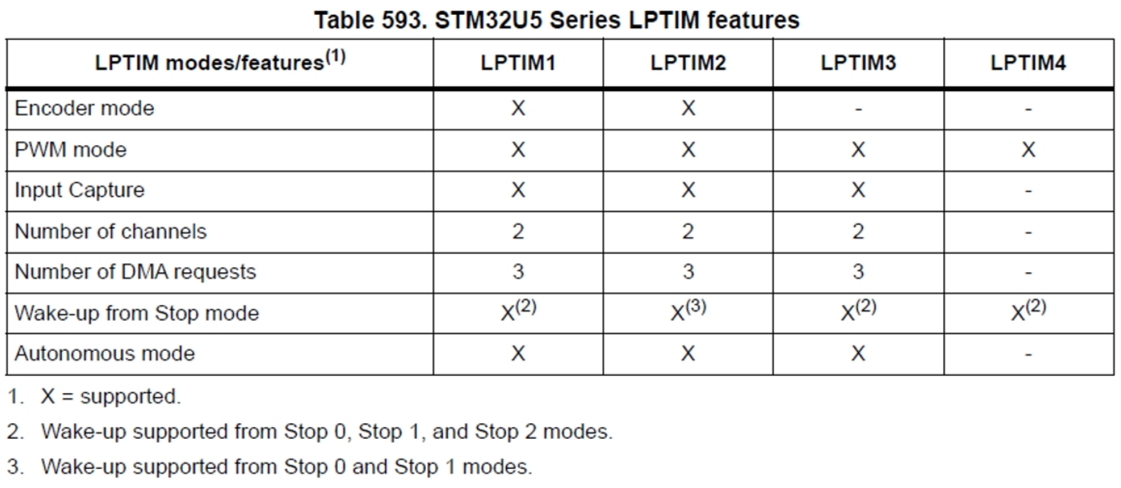


Table 593. STM32U5 Series LPTIM features

LPTIM modes/features[1]	LPTIM1	LPTIM2	LPTIM3	LPTIM4
Encoder mode	X	X	-	-
PWM mode	X	X	X	X
Input Capture	X	X	X	-
Number of channels	2	2	2	-
Number of DMA requests	3	3	3	-
Wake-up from Stop mode	X[2]	X[3]	X[2]	X[2]
Autonomous mode	X	X	X	-

1. X = supported.
2. Wake-up supported from Stop 0, Stop 1, and Stop 2 modes.
3. Wake-up supported from Stop 0 and Stop 1 modes.

Figure 1: LPTIM3 Wakeup in Stop2 Mode

2.1 Hardware and software checks

- Hardware: The schematic and PCB were reviewed against application note AN5373, and no issues were found.
- Software: The customer had ported code from another project and suspected of a possible silicon bug. However, no relevant issues were found in the errata.

2.2 Code Configuration

LPTIM3 was initialized with the following code:

```c
void MX_LPTIM3_Init(void)
{
  hlptim3.Instance = LPTIM3;
  hlptim3.Init.Clock.Source = LPTIM_CLOCKSOURCE_APBCLOCK_LPOSC;
  hlptim3.Init.Clock.Prescaler = LPTIM_PRESCALER_DIV1;
  hlptim3.Init.Trigger.Source = LPTIM_TRIGSOURCE_SOFTWARE;
  hlptim3.Init.Period = 65535;
  hlptim3.Init.UpdateMode = LPTIM_UPDATE_IMMEDIATE;
  hlptim3.Init.CounterSource = LPTIM_COUNTERSOURCE_INTERNAL;
  hlptim3.Init.Input1Source = LPTIM_INPUT1SOURCE_GPIO;
  hlptim3.Init.Input2Source = LPTIM_INPUT2SOURCE_GPIO;
  hlptim3.Init.RepetitionCounter = 0;

  if (HAL_LPTIM_Init(&hlptim3) != HAL_OK)
  {
    Error_Handler();
  }
}
```

Listing 1: LPTIM3 Init Code

LPTIM3 was started with:

```
if (HAL_LPTIM_TimeOut_Start_IT(&hlptim3, TIMEOUT) != HAL_OK)
{
    Error_Handler();
}
```

Listing 2: LPTIM3 Start Code

In the main loop, the MCU enters STOP2 mode:

```
HAL_PWREx_EnterSTOP2Mode(PWR_STOPENTRY_WFI);
```

Listing 3: LPTIM3 Start Code

2.3 Low Power Mode Indicator Pin Configuration

To debug the low-power state, pins PC6,PC7, and PC8 were configured to indicate the CSLEEP, CSTOP, and SRDSTOP status, respectively.

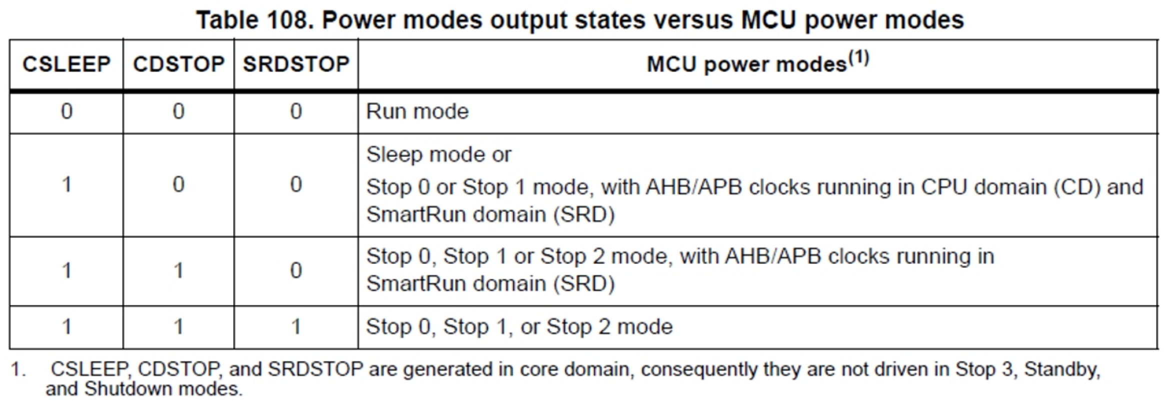


Table 108. Power modes output states versus MCU power modes

CSLEEP	CDSTOP	SRDSTOP	MCU power modes[1]
0	0	0	Run mode
1	0	0	Sleep mode or Stop 0 or Stop 1 mode, with AHB/APB clocks running in CPU domain (CD) and SmartRun domain (SRD)
1	1	0	Stop 0, Stop 1 or Stop 2 mode, with AHB/APB clocks running in SmartRun domain (SRD)
1	1	1	Stop 0, Stop 1, or Stop 2 mode

1. CSLEEP, CDSTOP, and SRDSTOP are generated in core domain, consequently they are not driven in Stop 3, Standby, and Shutdown modes.

Figure 2: STOP Mode Indicator

2.4 Test Result

Logical analyzer Saleae showed that CSLEEP, CSTOP, and SRDSTOP pins remained high, indicating that the MCU stayed in STOP2 mode and was not being woken up by LPTIM3.

3 In-depth Analysis and Solution

3.1 Autonomous Mode Configuration

According to section 58.4.20 of RM0456, when clocked by oscillator,LPTIM3 can operate in Autonomous Mode, allowing it to function even when the APB3 clock is stopped in STOP mode. We could get the informmation from the MCU architecture that LPTIM3 is in APB3 bus. Section 11.4.24 of the RCC chapter states that for autonomous peripherals which can not use EXTI to wake the MCU, the AMEN bit must be set to 1 to allow interrupt capability in STOP mode.

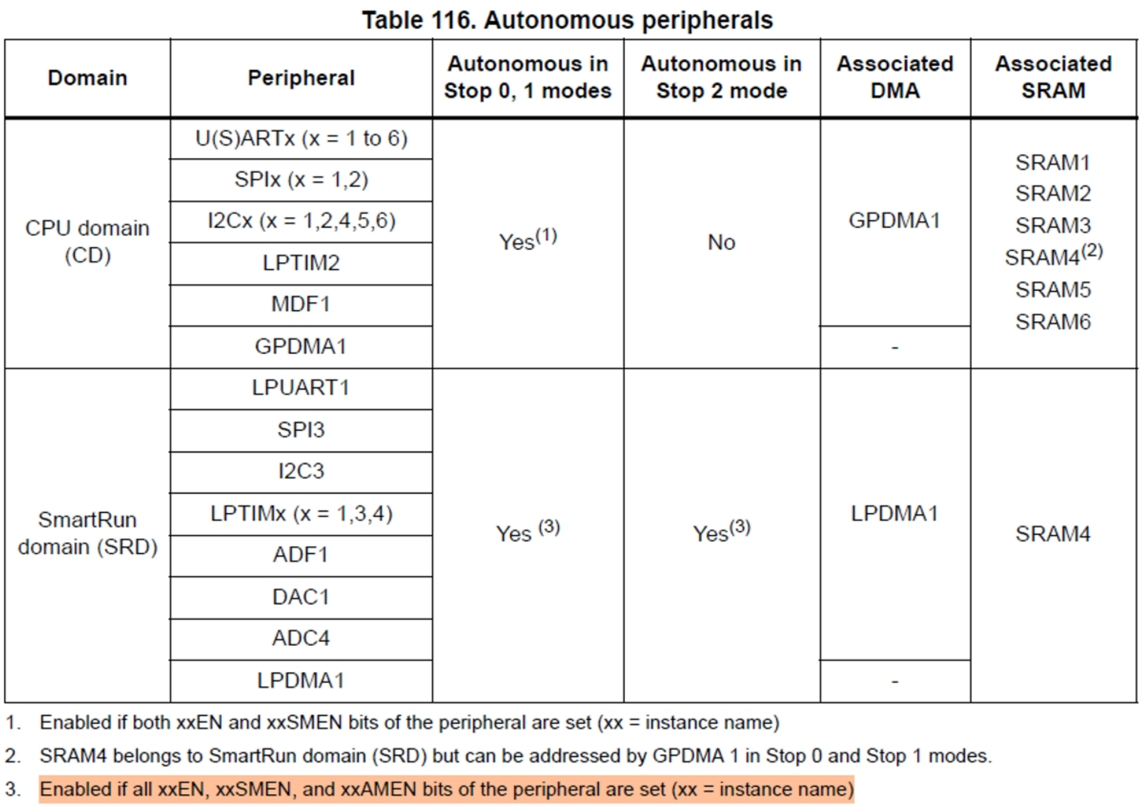
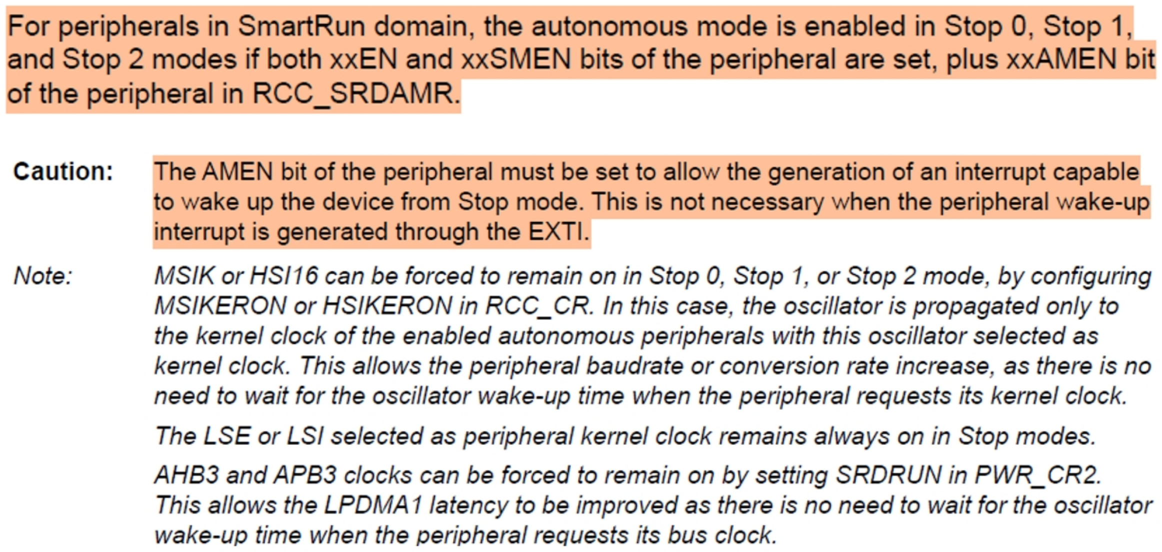


Table 116. Autonomous peripherals

Domain	Peripheral	Autonomous in Stop 0, 1 modes	Autonomous in Stop 2 mode	Associated DMA	Associated SRAM
CPU domain (CD)	U(S)ARTx (x = 1 to 6)	Yes(1)	No	GPDMA1	SRAM1 SRAM2 SRAM3 SRAM4(2) SRAM5 SRAM6
	SPIx (x = 1,2)				
	I2Cx (x = 1,2,4,5,6)				
	LPTIM2				
	MDF1				
	GPDMA1			-	
SmartRun domain (SRD)	LPUART1	Yes (3)	Yes(3)	LPDMA1	SRAM4
	SPI3				
	I2C3				
	LPTIMx (x = 1,3,4)				
	ADF1				
	DAC1				
	ADC4				
	LPDMA1			-	

1. Enabled if both xxEN and xxSMEN bits of the peripheral are set (xx = instance name)
2. SRAM4 belongs to SmartRun domain (SRD) but can be addressed by GPDMA 1 in Stop 0 and Stop 1 modes.
3. Enabled if all xxEN, xxSMEN, and xxAMEN bits of the peripheral are set (xx = instance name)

Figure 3: Note1


For peripherals in SmartRun domain, the autonomous mode is enabled in Stop 0, Stop 1, and Stop 2 modes if both xxEN and xxSMEN bits of the peripheral are set, plus xxAMEN bit of the peripheral in RCC_SRDAMR.

Caution: The AMEN bit of the peripheral must be set to allow the generation of an interrupt capable to wake up the device from Stop mode. This is not necessary when the peripheral wake-up interrupt is generated through the EXTI.

Note: MSIK or HSI16 can be forced to remain on in Stop 0, Stop 1, or Stop 2 mode, by configuring MSIKERON or HSIKERON in RCC_CR. In this case, the oscillator is propagated only to the kernel clock of the enabled autonomous peripherals with this oscillator selected as kernel clock. This allows the peripheral baudrate or conversion rate increase, as there is no need to wait for the oscillator wake-up time when the peripheral requests its kernel clock.

The LSE or LSI selected as peripheral kernel clock remains always on in Stop modes.

AHB3 and APB3 clocks can be forced to remain on by setting SRDRUN in PWR_CR2. This allows the LPDMA1 latency to be improved as there is no need to wait for the oscillator wake-up time when the peripheral requests its bus clock.

Figure 4: Note2

3.2 Register Setting

To ensure LPTIM3 can function properly, the following bits must be set to 1:

- LPTIM3EN: Enables the LPTIM3 clock(RCC_APB3ENR)

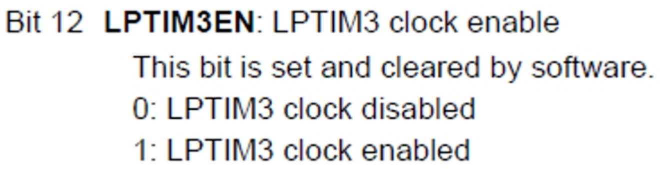

Bit 12 **LPTIM3EN**: LPTIM3 clock enable
This bit is set and cleared by software.
0: LPTIM3 clock disabled
1: LPTIM3 clock enabled

Figure 5: Enables the LPTIM3 clock

- LPTIM3SMEN: Enables the LPTIM3 clock during sleep mode(RCC_APB3SMENR)

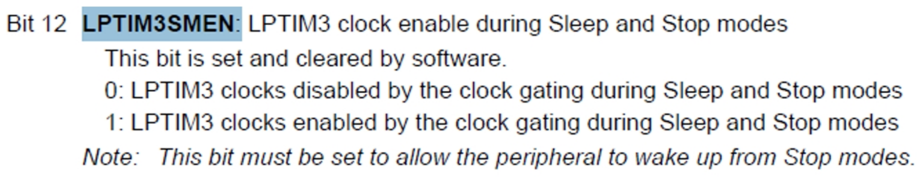

Bit 12 **LPTIM3SMEN**: LPTIM3 clock enable during Sleep and Stop modes
This bit is set and cleared by software.
0: LPTIM3 clocks disabled by the clock gating during Sleep and Stop modes
1: LPTIM3 clocks enabled by the clock gating during Sleep and Stop modes
Note: This bit must be set to allow the peripheral to wake up from Stop modes.

Figure 6: Enables the LPTIM3 clock during sleep mode

- LPTIM3AMEN: Enables the LPTIM3 autonomous mode(RCC_SRDAMR)

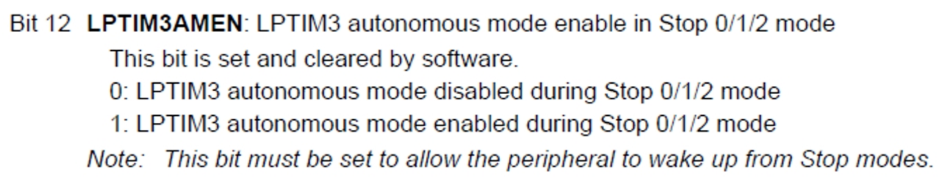

Bit 12 **LPTIM3AMEN**: LPTIM3 autonomous mode enable in Stop 0/1/2 mode
This bit is set and cleared by software.
0: LPTIM3 autonomous mode disabled during Stop 0/1/2 mode
1: LPTIM3 autonomous mode enabled during Stop 0/1/2 mode
Note: This bit must be set to allow the peripheral to wake up from Stop modes.

Figure 7: Enables the LPTIM3 autonomous mode

Before starting LPTIM3, the following code was added:

```
__HAL_RCC_LPTIM3_CLK_SLEEP_ENABLE();  // Enable LPTIM3 clock in sleep
__HAL_RCC_LPTIM3_CLKAM_ENABLE();      // Enable LPTIM3 Autonomous Mode
```

Listing 4: LPTIM3 Start Code

3.3 Test Verification

After adding the above configurations, the test was repeated. It was confirmed that the MCU could now be periodically woken up from STOP2 mode by LPTIM3, and the waveforms were as expected – problem resolved.

4 Summary and Recommendations

- Understand the MCU's internal mechanisms: Peripheral behavior may vary across MCU models. Read the reference manual carefully to understand how each module functions.
- Correct Autonomous Mode Setup: For peripherals expected to run during low-power modes, ensure Autonomous mode is properly enabled, especially the AMEN bit.
- Debugging Techniques: Use low-power indicator pins like CSLEEP, CSTOP,and SRDSTOP to asist with debugging power modes.
- Pay attention to inter-module dependencies: Though datasheets describe modules separately, many features require coordination between modules like RCC, PWR, and peripheral registers.

5 Note

To enable LPTIM3 to wake up the CPU from STOP2 mode, it is essential to configure LPTIM3 to work in Autonomous mode. This requires proper setup of the RCC and PWR modules to provide the necessary clocks and permissions for independent operation.

6 Reference Source Code

https://github.com/hunianhang/prj_experience/002_LTIM3_HSI_WakeUp.

References