MG0007

Migrating a microcontroller application from SXX32F103 to AT32F403A

Introduction

This migration guide is written to help users with the analysis of the steps required to migrate from an existing SXX32F103 series to an AT32F403A series device. It gathers the most important information and lists the mandatory aspects that need to address.

To migrate an application from SXX32F103 series to AT32F403A series, users have to analyze the hardware migration, the peripheral migration and firmware migration.

Applicable products：

Part numbers	AT32F403Axxxx

Contents

1 Similarities and differences between AT32F403 series and SXX32F103 series 6

1.1 Overview of similarities 6

1.2 Overview of differences 6

2 Quick replacement of SXX32F103 chip 8

2.1 Quick verification of compatibility 8

2.2 BSP project replacement steps 9

3 Compatibility analysis of AT32F403A 10

3.1 Feature enhanced 10

3.1.1 FPU ARM® 32-bit Cortex®-M4F 10

3.1.2 sLib (Security Library) protection 10

3.1.3 High frequency PLL setting 10

3.1.4 Additional prescaler 11

3.1.5 Expanded memory capacity 11

3.1.6 External SPI Flash via SPIM interface 12

3.1.7 Support additional SDIO2 12

3.1.8 Support additional I2C3 13

3.1.9 Support additional SPI4 13

3.1.10 Support additional full-duplex I2S 13

3.1.11 Support additional USART 和 UART 13

3.1.12 Support additional CAN2 function 14

3.1.13 Support the use of CAN and USB simultaneously 14

3.1.14 32-bit Timer 14

3.1.15 SPI1 supports I2S1 14

3.1.16 USBDEV buffer 14

3.1.17 Additional 48 MHz HSI supports USB peripherals 15

3.1.18 HSI automatic clock calibration ACC 15

3.1.19 64-pin package supports XMC 15

3.1.20 Support Flash memory CRC check 16

3.1.21 High speed GPIO 16

3.1.22 Additional DMA flexible mapping request function 18

3.2 Peripheral differences 19
3.2.1 Program will hang or enter hard fault due to PLL switching at high frequency 19
3.2.2 Internal temperature sensor 19
3.2.3 ADC requires longer sampling time 20
3.2.4 GPIO 5V-tolerant pin compatibility 20
3.2.5 BOOT0 with pull-down resistor 20
3.2.6 The pull-down resistors of PA0 pin is automatically enabled in Standby mode 20
3.2.7 The system clock frequency must not be less than 12 MHz when using USB 21
3.2.8 Differences between USB double-buffered SW_BUF and DTOG 21
3.2.9 USB_DP has internal pull-up resistor 21
3.2.10 DAC output glitches 21
3.2.11 Flash sector erase time is longer 21
3.2.12 SDIO receive data occurs after RX over run occurs 22
3.2.13 Enabling I2S slave may cause communication data errors and cannot be recovered automatically 22
3.2.14 GPIO has overshooting phenomenon at max 50MHz output 22
3.2.15 The program will hang in a loop that determines CAL control bit as zero when using ADC dual mode 23
3.2.16 EWIF flag cannot be cleared when using WWDG interrupt 23
3.2.17 MCO output cannot be stopped in Stop mode 24
3.2.18 Systick interrupt abnormality wakes up and executes Stop mode entered by WFE 24
3.2.19 Data reception error in USART smart card mode 24
3.2.20 PWR enters low-power mode through different WFE operation 24

4 Revision history 25

List of tables

Table 1. Differences between SXX32F103 and AT32F403A 6
Table 2. BSP replacement steps 9
Table 3. SPI Flash support list 12
Table 4. Pin support when EXT_SPIF_GRMP[2:0] is [000] 12
Table 5. Pin support when EXT_SPIF_GRMP[2:0] is [001] 12
Table 6. SXX32F103/403A GPIO toggle Max speed performance test 16
Table 7. Document revision history 25

List of figures

Figure 1. 403A GPIO Max toggle speed waveform at 192 MHz Max frequency 16
Figure 2. 403A GPIO Max toggle speed waverform at 72 MHz Max frequency 17
Figure 3. SXX32F103 GPIO Max toggle speed waveform at 72 MHz Max frequency 17
Figure 4. Ideal curve of V_{SENSE} vs.Tempeature 20
Figure 5. USB double-buffer instructions 21

1 Similarities and differences between AT32F403 series and SXX32F103 series

The AT32F403A series microcontrollers are basically compatible with the SXX32F103 series, and also optimize many functional relationships, some of which are different from the SXX32F103 series, which are detailed in this document.

1.1 Overview of similarities

- Pin definition: the same package has the same pin definition. Extended the pin alternate definition for additional peripherals
- Addressing space: the memory and register logical address remains the same. The additional peripherals occupy SXX32 reserved space.
- Function library files: the function library is the same. Some header files are optimized according to enhanced functions.
- Compiler tools: identical, for example, Keil, IAR

1.2 Overview of differences

Table 1. Differences between SXX32F103 and AT32F403A

	AT32F403A	SXX32F103
System		
Core	Cortex-M4 and supports DSP instruction and floating point unit (FPU)	Cortex-M3
System clock	Max frequency: 240 MHz Both APB1 and APB2 bus are 120 MHz	Max frequency : 72 MHz APB1 36MHz, APB2 72MHz
Startup	13 ms	2.5 ms
Reset	8 ms	-
Wake up from Standby mode	8 ms	50 us
SRAM capacity	Extend up to 224 KB	96 KB
External SPI Flash via SPIM interface	Supports external SPI Flash as SPIM, up to 16 Mbytes	Not support
System Memory	The whole series is 18 KB, supporting the following features compared to the SXX32F103 series: 1. USB DFU ISP programming 2. ISP programming of SPIM 3. Perform CRC check on the contents of Flash memory	6 KB/2 KB
Option Byte	48 Bytes, added the following features: 1. SRAM mode setting 2. 4 Bytes custom fields (such as developer ID) 3. 8 Bytes SPIM encryption key	16 Bytes
Flash memory 16-bit write time	50 µs	52.5 µs
Flash memory sector erase time	50 ms	40 ms
The whole Flash memory chip erase time	0.8 s (AT32F403AxC) 1.4 s (AT32F403AxE) 2.8 s (AT32F403AxG)	40 ms

	AT32F403A	SXX32F103
Peripheral		
Security library (sLib) protection	Support	N/A
Backup registers	The full family supports 42 sets of half byte backup registers	Small-and medium-density series only support 10sets
Additional I^2C	One more I^2C	$I^2C1/2$
Additional SPI	One more SPI4	SPI1/2/3
Additional SDIO	One more SDIO2	SDIO1
Additional USART and UART	Supports USART6/UART7/UART8	Not support USART6/UART7/UART8
SPI1 supports I^2S	SPI1 can support I^2S	SPI1 only, not support I^2S
I^2S support	48-pin with I^2S $I^2S2/3$ supports full-duplex	48-pin without I^2S I^2S not support full-duplex
Additional CAN2	Supports CAN2	Not support CAN2
Support to use CAN and USB simultaneously	Support	Not support
Additional 48MHz HSI supports USB peripherals	Support	Not support
HSI auto clock calibration ACC	Support	Not support
XMC	1. Not support CF card; support multiplexed signal NOR/PSRAM or support non-multiplexed signal SRAM/PSRAM through external devices (refer to AN0068) 2. 2 chip selections 3. Not support external interrupts 4. 64-pin package supports 8-bit parallel LCD	1. Supports CF card and NOR/SRAM/PSRAM 2. 4 chip selections 3. Supports 2 external interrupts 4. Not support 64-pin packages
Support Flash memory CRC check	Support	Not support
High-speed GPIO	GPIO is connected to AHB bus	GPIO is connected to APB bus
Expanded memory interface	64-pin and above supports bus output (XMC)	Only 144-pin and more than 256 KB Flash memory supports bus output
32-bit timer	TMR2 and TMR5 are 32-bit timer	All for 16-bit timers.
USB buffer	Can be extended to 768 Byte	512 Byte
ADC	2 Msps (max ADCCLK=28MHz)	1 Msps (max ADCCLK=14 MHz)
ADC trigger event	Supports TMR1, TMR8 and TMR15	No TMR15
Temperature sensor	Positive temperature factor	Negative temperature factor
Electrical characteristics		
Voltage range	2.6 V~3.6 V	2.0 V~3.6 V
Ambient temperature T_A	-40°C~+105°C	-40°C~+85°C
Core voltage	1.2 V	1.8 V
ESD parameters	HBM:5KV, CDM:1000V	HBM:2KV, CDM:500V
Run mode	37.1 mA @ 72MHz	51 mA @ 72MHz
Power consumption at Sleep mode	31.8 mA @ 72MHz	29.5mA @ 72MHz
Power consumption at Stop mode	1.4 mA	25 uA
Power consumption at Standby mode	5.7 uA	2.1 uA

2 Quick replacement of SXX32F103 chip

2.1 Quick verification of compatibility

- Step 1: De-solder SXX32F103 and replace it with the corresponding AT32F403A part no.
- Step 2: Download SXX32F103 HEX file or BIN file using Artery ICP, ISP, KEIL or IAR.
- Step 3: If necessary, download information other than SXX32F103 HEX file or BIN file or perform System calibration.
- Step 4: Check if the program can run normally.
- Step 5: If the system clock source uses HSE, in order to ensure mass production stability, please modify the clock initialization source code as follows:
  Open system_at32f4xx.c to find the current system clock frequency configuration function,
  Such as 168 MHz function:
  Static void SetSysClockTo168(void)
  Then configure the automatic step-by-step frequency switching function as follows:

```
/* Wait till PLL is ready */
      while((RCC->CR & RCC_CR_PLLRDY) == 0)
      {
      }
      *((unsigned int   *)0x40021054) |= (0x30);// enable automatic step-by-step
frequency switching function

      /* Select PLL as system clock source */
      RCC->CFGR &= (uint32_t)((uint32_t)~(RCC_CFGR_SW));
      RCC->CFGR |= (uint32_t)RCC_CFGR_SW_PLL;

      /* Wait till PLL is used as system clock source */
      while ((RCC->CFGR & (uint32_t)RCC_CFGR_SWS) != (uint32_t)0x08)
      {
}
// no need to wait for 200us
*((unsigned int   *)0x40021054) &=~ (0x30); //disable the automatic step-by-step
frequency switching function
```

- Step 6: For quick troubleshooting of other issues, please refer to *3.2 Peripheral differences*
- Step 7: If the program still cannot run normally after following the above steps, please refer to Other Chapters in this document, or contact your agent and Artery technical support staff for assistance.

2.2 BSP project replacement steps

- The six application scenarios are summarized in the table below:

Table 2. Overview of BSP replacement steps

No.	BSP/Pack used	AT32F403A new features used or not	Solution
1	AT32F403A BSP/Pack	Y/N	1.Modify the corresponding program according to *3.2 Peripheral differences*
2	SXX32F103 BSP/Pack	N	1. Refer to section 2.1 “enable clock step-by-step mode” before the system clock switching 2. Refer to section 2.1 “disable clock step-by-step mode” after the system clock switching 3. Modify the corresponding program according to *3.2 Peripheral differences*
3	SXX32F103 register operation	N	1. Refer to Refer to section 2.1 “enable clock step-by-step mode” before the system clock switching. 2. Refer to section 2.1 “disable clock step-by-step mode” after the system clock switching. 3.Modify the corresponding program according to *3.2 Peripheral differences*
4	SXX32F103 BSP + AT32 Pack	N	1. Refer to Refer to section 2.1 “enable clock step-by-step mode” before the system clock switching. 2. Refer to section 2.1 “disable clock step-by-step mode” after the system clock switching. 3. Modify FPU setting. 4. Modify the corresponding program according to *3.2 Peripheral differences*
5	SXX32F103 register operation	Y	1. Refer to Refer to section 2.1 “enable clock step-by-step mode” before the system clock switching. 2. Refer to section 2.1 “disable clock step-by-step mode” after the system clock switching. 3. Modify the corresponding program according to *3.2 Peripheral differences*
6	SXX32F103 BSP/Pack	Y	1. Refer to Refer to section 2.1 “enable clock step-by-step mode” before the system clock switching. 2. Refer to section 2.1 “disable clock step-by-step mode” after the system clock switching. 3. Modify the corresponding program according to *3.2 Peripheral differences*

- For further information, please refer to Application Note of AT32F403A BSP and Pack

3 Compatibility analysis of AT32F403A

3.1 Feature enhanced

3.1.1 FPU ARM® 32-bit Cortex®-M4F

- **Description:**
  - With memory protection unit (MPU)
  - Embedded single-cycle multiplication and hardware division
  - Embedded floating point unit (FPU)
  - Supports DSP instructions
  - **Example code:**
    AT32F4xx_StdPeriph_Lib_V1.x.x\Project\Examples\AT_START_F403A\CortexM4\FPU
    AT32F4xx_StdPeriph_Lib_V1.x.x\Libraries\CMSIS\DSP_Lib\Examples

3.1.2 sLib (Security Library) protection

- **Description:**
  - At present, more and more microcontroller applications need to use complex algorithms and middleware solutions, therefore, how to protect the core algorithms and intellectual property codes (IP-Code) developed by software solutions providers has become a very important subject in the microcontroller applications.

    In response to this important demand, AT32F403A series provides the security library (sLib) function to prevent important IP-Code from being modified or read by end-user programs to achieve protection.

- **Example code:**
  - Please refer to Application Note of AT32F403A sLib (security library).

3.1.3 High frequency PLL setting

- **Description:**
  - The AT32F403A embeds a PLL that can output 240 MHz clock with slightly different settings.
  - The PLL supports two frequency ranges, with 72 MHz as the boundary, up to 240 MHz, and the PLLRANGE register must be set according to the output frequency.
- **Example code:**
  - SXX32F103 PLL setting example:
    RCC->CFGR |= (uint32_t)(RCC_CFGR_PLLSRC_HSE | RCC_CFGR_PLLMULL9);
  - AT32F403A PLL setting example:

```
#define RCC_CFG_PLLMULT1 ((uint32_t)0x20000000) /*!< PLL input clock    * 17 */
#define RCC_CFG_PLLMULT18 ((uint32_t)0x20040000) /*!< PLL input clock * 18 */
#define RCC_CFG_PLLMULT19 ((uint32_t)0x20080000) /*!< PLL input clock * 19 */
#define RCC_CFG_PLLMULT20 ((uint32_t)0x200C0000) /*!< PLL input clock * 20 */
…
#define RCC_CFG_PLLMULT61 ((uint32_t)0x60300000) /*!< PLL input clock * 61 */
#define RCC_CFG_PLLMULT62 ((uint32_t)0x60340000) /*!< PLL input clock * 62 */
#define RCC_CFG_PLLMULT63 ((uint32_t)0x60380000) /*!< PLL input clock * 63 */
#define RCC_CFG_PLLMULT64 ((uint32_t)0x603C0000) /*!< PLL input clock * 64 */

#define RCC_CFG_PLLRANGE ((uint32_t)0x80000000) /*!< PLL Frequency range */
#define RCC_CFG_PLLRANGE_LE72MHZ ((uint32_t)0x00000000) /*!< When PLL
frequency is less than or equal to 72MHz */
#define   RCC_CFG_PLLRANGE_GT72MHZ   ((uint32_t)0x80000000) /*!< When PLL
frequency is greater than 72MHz */

Take setting 72 MHz as an example：
RCC->CFG |= (uint32_t)(RCC_CFG_PLLRC_HSE | RCC_CFG_PLLMULT9 |
RCC_CFG_PLLRANGE_LE72MHZ);
Take setting 200 MHz as an example：
RCC->CFG |= (uint32_t)(RCC_CFG_PLLRC_HSE | RCC_CFG_PLLMULT25 |
RCC_CFG_PLLRANGE_GT72MHZ);
```

3.1.4 Additional prescaler

- **Description:**
  - As the max frequency is increased to 240 MHz, the relevant prescalers are expanded.
  - USB prescaler supports additional /2, /2.5, /3, /3.5, /4 output
  - ADC prescaler supports additional /12, /16 output
  - HSE prescaler supports /3, /4, /5 output
  - The clock output supports additional CLKOUT prescaler, which can implement CLKOUT/2. CLKOUT/4…CLKOUT/512 frequency division.
  - The clock output (CLKOUT) supports additional LSE, LSI, PLLCLK/4, USB48M and ADCCLK output.
  - Please refer to 3.3.2 RCC_CFG register and 3.3.12 extra register (RCC_MISC) sections in the AT32F403A Reference Manual

3.1.5 Expanded memory capacity

- **Description:**
  - The AT32F403A series supports memory capacity expansion from 96 KB to 224 KB.
  - If this mode is enabled, the Flash memory address space with zero wait feature will be limited to 128 KB
  - The memory expansion function is set through the option byte EOPB0[7:0] (0x1FFF F810).
    0xFE: 224 Kbytes of on-chip memory
    0xFF: 96 Kbytes of on-chip memory
    Other settings reserved.
  - After the option byte EOPB0 is updated, the system must be reset for EOPB0 to take effect.
  - Please refer to 5.3.4 section in the AT32F403A Reference Manual.
  - Example code:
    AT32F4xx_StdPeriph_Lib_V1.x.x\Project\Examples\AT_START_F403\SRAM

3.1.6 External SPI Flash via SPIM interface

- **Description:**
  - AT32F403A supports external SPI Flash, the following part numbers and compatible part numbers with a capacity of up to 16 Mbyte. If it is impossible to judge whether the SPI FLASH part is compatible, the SPIM automatic detection function of AT32 ICP tools can be used.
  - Please refer to 5.2.2 and 5.4.14 sections in the AT32F403A Reference Manual
  - In addition to the SPI Flash part numbers in the table below, the SPI Flash chips compatible with the command sets in the table below are also supported.

Table 3. SPI Flash support list

Vendor	SPI Flash parts
ESMT	EN25F20A/EN25QH128A
Winbond	W25Q128V
GD	GD25Q16C/GD25Q32C/GD25Q64C/GD25Q80C/GD25Q127C

  - SPIM supports access to addresses 0x0840 0000~0x1FFF 0000
  - The Flash memory function is supported in all types of packages. The SPIM_IO0 and SPIM_IO01 pin can be remapping configured according to EXT_SPIF_GRMP[2:0]. The pins used are shown in the table below:

Table 4. Pin support when EXT_SPIF_GRMP[2:0] is [000]

SPIM	Pin	LQFP48 QFN48	LQFP 64	LQFP 100	Shared pin description
SPIM_SCK	PB1	19	27	36	ADC12_IN9/TMR3_CH4/TMR8_CH3N
SPIM_NSS	PA8	29	41	67	TMR1_CH1/CLKOUT/USART1_CK/I2C3_SCL
SPIM_IO0	PA11	32	44	70	USB_DM/TMR1_CH4/USART1_CTS/CAN_RX
SPIM_IO1	PA12	33	45	71	USB_DP/CAN_TX/USART1_RTS/TMR1_ETR
SPIM_IO2	PB7	43	59	93	TMR4_CH2/I2C1_SDA/XMC_NADV
SPIM_IO3	PB6	42	58	92	TMR4_CH1/I2C1_SCL

Table 5. Pin support when EXT_SPIF_GRMP[2:0] is [001]

SPIM	Pin	LQFP48 QFN48	LQFP 64	LQFP 100	Shared pin description
SPIM_SCK	PB1	19	27	36	ADC12_IN9/TMR3_CH4/TMR8_CH3N
SPIM_NSS	PA8	29	41	67	TMR1_CH1/CLKOUT/USART1_CK/I2C3_SCL
SPIM_IO0	PB10	21	29	47	USART3_TX/I2C2_SCL/I2S3_MCK/TMR2_CH3
SPIM_IO1	PB11	22	30	48	USART3_RX/I2C2_SDA/TMR2_CH4
SPIM_IO2	PB7	43	59	93	TMR4_CH2/I2C1_SDA/XMC_NADV
SPIM_IO3	PB6	42	58	92	TMR4_CH1/I2C1_SCL

  - Example code:
AT32F4xx_StdPeriph_Lib_V1.x.x\Project\Examples\AT_START_F403A\FLASH

3.1.7 Support additional SDIO2

- **Description:**
  - AT32F403A supports additional SDIO2.
  - Available on 48-pin/64-pin/100-pin package where 48-pin package only supports SDIO2
  - Register start address: 0x4002_3400
  - Interrupt vector number: IRQ 60

- DMA: use DMA2 channel 5
- Alternate function setting of pin:

  Extend SDIO alternate remapping SDIO2_REMAP[1:0] in AFIO_MAP2 register bit [20:19] (please refer to section 7.4.11 of AT32F403A Reference Manual)

3.1.8 Support additional I2C3

- **Description:**
  - AT32F403A supports additional I²C3
  - Available on 64-pin/100-pin package
  - Register start address: 0x4000_6800
  - Interrupt vector number: IRQ 61, IRQ 62
  - DMA: use DMA1 channel 2(TX) and channel 3(RX)
  - Alternate function setting of pin:

    Extend I²C3 alternate function remapping I2C3_REMAP in AFIO_MAP2 register bit [18] (please refer to section 7.4.8 of AT32F403A Reference Manual)

3.1.9 Support additional SPI4

- **Description:**
  - AT32F403A supports additional SPI4/I²S4
  - Available on 64-pin/100-pin package
  - Register start address: 0x4002_3400
  - Interrupt vector number: IRQ 63
  - DMA: use DMA2 channel 3(RX) and channel 4(TX)
  - Alternate function setting of pin

    Extend SPI4 alternate function remapping SPI4_REMAP in AFIO_MAP2 register bit [17] (please refer to 7.4.10 section in the AT32F403A Reference Manual)

3.1.10 Support additional full-duplex I2S

- **Description:**
  - AT32F403A added two modules(I2S2_ext，I2S3_ext) to support I²S full-duplex mode
  - Available on 48-pin/64-pin/100-pin package
  - I2Sx_ext cannot be used independently, only can be used in full-duplex mode with I²Sx, and I2Sx_ext needs to be configured as a slave.
  - In full-duplex mode, the unused SPI_MISO pin of the original I²S mode is used as an additional data line, other pins remain unchanged.
  - I2Sx_ext has no independent interrupt vector number(shared with I²Sx),but has independent register addresses and DMA channels(please refer to 1.2.1 and 9.3.8 section in the AT32F403A Reference Manual)

3.1.11 Support additional USART 和 UART

- **Description:**
  - AT32F403A supports additional USART6/UART7/UART8
  - Register start address: USART6-0x4001_6000, UART7-0x4001_6400, UART8-0x4001_6800
  - Interrupt vector number: IRQ 76, IRQ 77, IRQ 78
  - DMA: use flexible requests (please refer to section 9.3.8 of AT32F403A Reference Manual)
  - Alternate function setting of pin:

Extend USART6 alternate function remapping USART6_GRMP in AFIO_MAP8 register bit [23:20] (please refer to section 7.4.7 of AT32F403A Reference Manual)

Extend UART7 alternate function remapping UART7_GRMP in AFIO_MAP8 register bit [27:24] (please refer to section 7.4.7 of AT32F403A Reference Manual)

Extend UART8 alternate function remapping UART8_GRMP in AFIO_MAP8 register bit [31:28] (please refer to section 7.4.7 of AT32F403A Reference Manual)

3.1.12 Support additional CAN2 function

- **Description:**
  - AT32F403A supports additional CAN2 function.
  - Available on 48-pin/64-pin/100-pin package
  - Register start address: 0x4000 6800
  - Register start address: IRQ 68，IRQ 69，IRQ 70，IRQ 71
  - Alternate function setting of pin:

    Extend CAN2 alternate function CAN2_REMAP in AFIO_MAP register bit [22] (please refer to 7.4.3 section in the AT32F403A Reference Manual)

3.1.13 Support the use of CAN and USB simultaneously

- **Description:**
  - AT32F403A supports to use CAN and USB simultaneously.
  - Available on 48-pin/64-pin/100-pin package
  - CAN is responsible for managing its own independent 512 byte SRAM storage space
  - USB also has its own independent SRAM storage space, and the storage space of the disabled CAN can also be superimposed and allocated to USB(for more information ,please refer to 21.3.2.1 section in the AT32F403A Reference Manual)

3.1.14 32-bit Timer

- **Description:**
  - AT32F403A TMR2/TMR5 can be configured as a 32-bit timer
- Example code:
  - TMR2->CTRL1|=0X0400; //enable TMR2 32-bit mode
  - TMR5->CTRL1|=0X0400; //enable TMR5 32-bit mode

(The relevant variables must be changed to 32-bit, please refer to AT32F403A library file at32f40x_tim.c at32f40x_tim.h)

3.1.15 SPI1 supports I2S1

- **Description:**
  - AT32F403A SPI1 supports I^2S1
  - Available on 48-pin/64-pin/100-pin package
  - Added I^2S1_MCK pin
  - Other pins, register space and interrupt vector number, DMA channels are the same as SPI1

3.1.16 USBDEV buffer

- **Description:**
  - AT32F403A USB device (USBDEV) buffer can be expanded up to 768 Bytes.
  - Set in the USB768B(RCC_MISC[24]) (please refer to 3.3.11 section in the AT32F403A Reference Manual)

- 512 Byte mode: buffer address range 0x4000 6000~0x4000 63FF
- 768 Byte mode: when the USB768B is 1, the SRAM size is 768~1280 Byte, the address range is 0x4000 7800~0x4000 81FF, the buffer size depends on whether CAN1 and CAN2 are enabled. When CAN1 and CAN2 are not enabled, the USB packet buffer can be set to max 1280 Bytes, when either CAN1 or CAN2 is enabled, the USB packet buffer can be set to max 1024 Bytes; when both CAN1 and CAN2 are enabled, the USB packet buffer can be set to max 768 bytes.

- **768 Byte mode example:**
  - RCC->MISC |= 0x00000001 << 24; /*enable USB768Byte mode*/
  - #define PMAAddr 0x40007800 /*use 768 Byte mode, be sure to modify the buffer address*/

3.1.17 Additional 48 MHz HSI supports USB peripherals

- **Description:**
  - AT32F403A supports 48 MHz clock for USB use
  - Applicable to the condition when the system clock is synchronized with the USB clock. For example, if the system clock runs at 200 MHz (PLL from HSE), USB peripherals still need to be used.
  - For more information, please refer to 3.2.2 HSI clock section in the AT32F403A Reference Manual.
- **Example code:**
  - Please refer to AT32F4xx_StdPeriph_Lib_V1.x.x/Project/AT_START_F403A/Examples/USB_Device

3.1.18 HSI automatic clock calibration ACC

- **Description:**
  - AT32F403A added ACC module.
    HSI automatic clock calibrator (HSI ACC) uses the SOF signal (1ms cycle) generated by USB module as a reference signal to implement the sampling and calibration of the HSI clock.
  - The main function of this module is to provide 48 MHz±0.25% precision clock for USB devices.
  - Available on all packages
  - Register start address: 0x4001_5800
  - Interrupt vector number: IRQ 72
- **Example code:**
  - Please refer to the following example: AT32F4xx_StdPeriph_Lib_V1.x.x\Project\AT_START_F403A\Examples\ACC

3.1.19 64-pin package supports XMC

- **Description:**
  - AT32F403A 64-pin package supports XMC, but XMC only supports 8-bit 8080/6800 mode to drive LCD.
  - Alternate function setting of pin: XMC internal remapping in AFIO_MAP7 register bit [17:16] (please refer to 7.4.13 section in the AT32F403A Reference Manual）
- **Example code:**
  - Please refer to AT32F4xx_StdPeriph_Lib_V1.x.x/Project/AT_START_F403A/Examples/XMC/XMC_LCD_8BIT

3.1.20 Support Flash memory CRC check

- **Description:**
  - AT32F403A supports Flash memory CRC check by configuring Flash memory CRC check control register (CRC_DR), that is, the Flash memory contents are checked by the Flash memory CRC check result register (CRC_OUTR).This feature is also applicable in the case of read protection or sLib.
  - For complete information, please refer to 5.4.26 Flash CRC check control register (CRC_DR) and 5.4.27 Flash CRC check result register (CRC_OUTR) sections in the AT32F403A Technical Manual.

3.1.21 High speed GPIO

- **Description:**
  - AT32F403A upgraded GPIO by connecting GPIO clock to AHB bus, while SXX32F103 GPIO connecting to APB bus.
  - When doing GPIO toggle test or using GPIO to simulate SPI/USART/I^{2}C and other peripherals, AT32F403A GPIO toggle speed has faster frequency than SXX32F103, see the following comparison list:

Table 6. SXX32F103/403A GPIO toggle Max speed performance test

Clock frequency configuration(MHz)	AHB=192; APB2=96	AHB=72;APB2=72	AHB=36;APB2=36
SXX32F103 I/O toggle (MHz)	Not support	18	9
AT32F403A I/O toggle (MHz)	96	36	18

Figure 1. AT32F403A GPIO Max toggle speed waveform at 192 MHz Max frequency

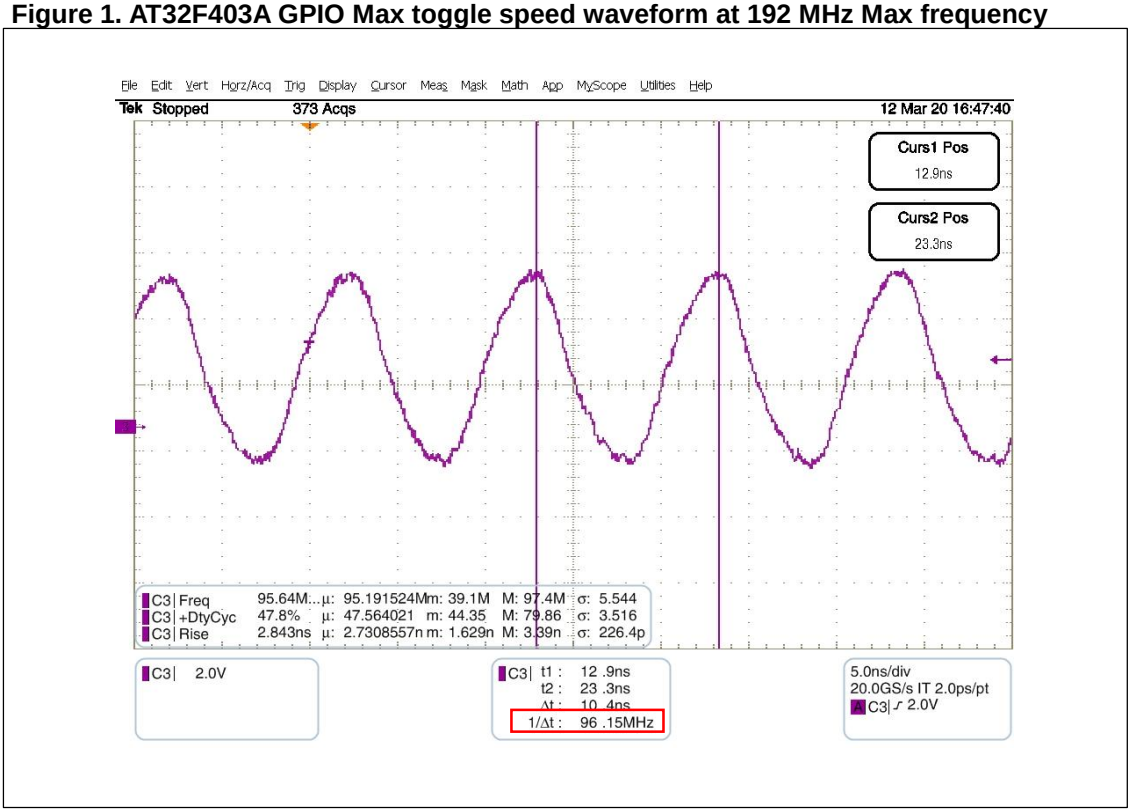


Figure 2. AT32F403A GPIO Max toggle speed waveform at 72 MHz Max frequency

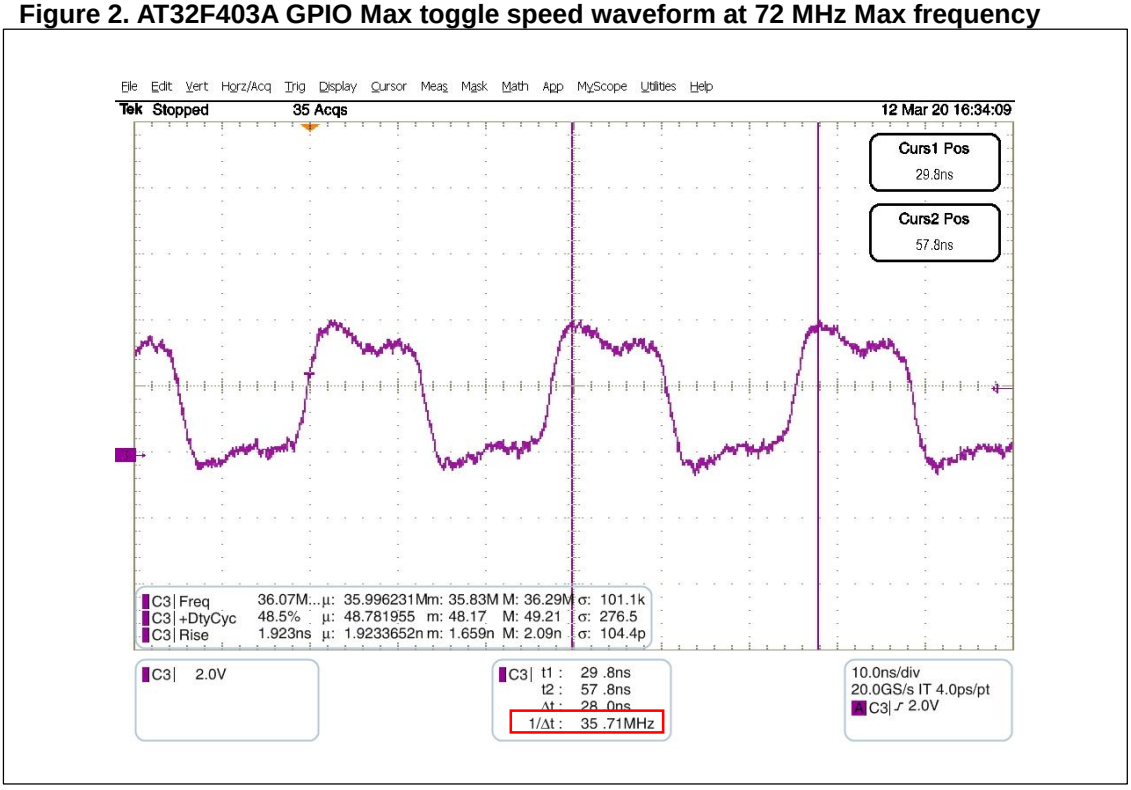


Figure 3. SXX32F103 GPIO Max toggle speed waveform at 72 MHz Max frequency

3.1.22 Additional DMA flexible mapping request function

- **Description:**
  - The DMA1/DMA2 of AT32F403A added flexible mapping request function.
  - Available on 48-pin/64-pin/100-pin package.
  - Only one of the flexible mapping and fixed mapping can be enabled, the two cannot be enabled at the same time.
  - Flexible mapping request has no independent interrupt vector number (shared with the fixed mapping request), but has independent configuration register address (please refer to AT32F403A Reference Manual)
- **Example code:**
  - Please refer to the following example: AT32F4xx_StdPeriph_Lib_V1.x.x\Project\AT_START_F403A\Examples\DMA\SPI_RAM_FLEXIBLE。

3.2 Peripheral differences

3.2.1 Program will hang or enter hard fault due to PLL switching at high frequency

- **Description:**
  During the initialization of RCC, when the system clock switches from the low-frequency state (HSE/HSI) to high-frequency PLL state (PLL >108MHz and above), the program may hang or enter hard fault.
- Special attention should be required as follows:
  - When user sets AT32F403A's PLL to 108 MHz or higher frequency, the PLL setting is slightly different, users need to operate the automatic step-by-step frequency switching function.
- 168MHz PLL example:

Open system_sxx32f10x.c to find the current system clock frequency configuration function (need to go through the above PLL configuration), such as 168MHz function:
Static void SetSysClockTo168(void)
Added the following italic bold:

```
 /* Wait till PLL is ready */
while((RCC->CR & RCC_CR_PLLRDY) == 0)
{
}
*((unsigned int   *)0x40021054) |= (0x30);// enable automatic step-by-step frequency switching function

/* Select PLL as system clock source */
RCC->CFGR &= (uint32_t)((uint32_t)~(RCC_CFGR_SW));
RCC->CFGR |= (uint32_t)RCC_CFGR_SW_PLL;

/* Wait till PLL is used as system clock source */
while ((RCC->CFGR & (uint32_t)RCC_CFGR_SWS) != (uint32_t)0x08)
{
}
  *((unsigned int   *)0x40021054) &=~ (0x30); //disable automatic step-by-step frequency switching function
```

3.2.2 Internal temperature sensor

- **Description:**
  AT32F403A temperature sensor is a positive temperature factor, while SXX32F103 is a Negative temperature factor.
- **Solution:**

  Follow the values in the datasheet and use the below formula to obtain the temperature：

  Temperature (℃) = {(V_{25} - V_{SENSE}) / Avg_Slope} + 25

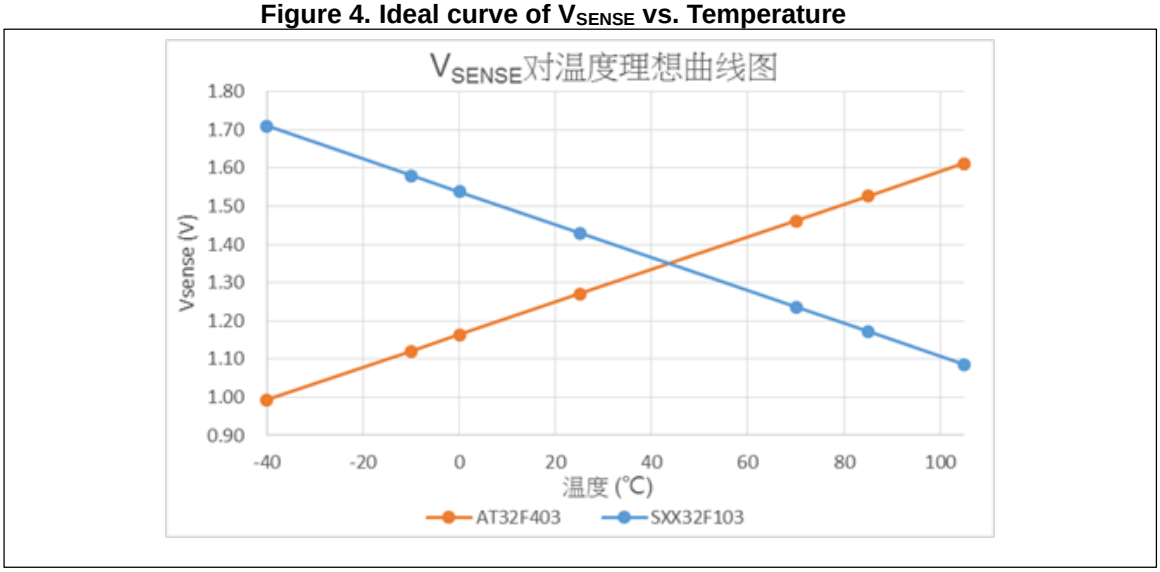
  Here:

  V_{25} = V_{SENSE} value for 25°C

  Avg_Slope = Average slope for curve between temperature vs.V_{SENSE} curve (unit: mV/℃).

  V_{25} and Avg_Slope must be calculated according to the typical values in the datasheet, AT32F403A is different from SXX32F103.

Figure 4. Ideal curve of V_{SENSE} vs. Temperature



- Example code:
  AT32F4xx_StdPeriph_Lib_V1.x.x\Project\Examples\AT_START_F403\ADC\Temperature

3.2.3 ADC requires longer sampling time

- **Description:**
  In order to adapt to the sampling rate of the ADC to 2 Msps, the internal equivalent R_{ADC} and C_{ADC} values are large, requiring a long sampling time. In particular, when the impedance of ADC input source is large, the sufficient sampling time should be met to obtain accurate conversion data and eliminate the cross-talk during the different ADC input channels conversion.
- **Solution:**
  If the conversion value is not as expected during use, users can first try to set the sampling time to max 239.5 ADC clocks, and then gradually reduce the sampling time to the appropriate set. If a longer sampling interval can be acceptable, inserting V_{REFINT} between different channel conversions also can get the accurate conversion data.

3.2.4 GPIO 5V-tolerant pin compatibility

- **Description:**
  PA11 & PA12 in each package and PD0 & PD1 in 64/48 pin package are not a 5-Volt tolerant pins, so the input level of these pins should not exceed VDD + 0.3V.
- **Solution:**
  Please pay attention to this restriction when using.

3.2.5 BOOT0 with pull-down resistor

- **Description:**

  When the BOOT0 is used, the internal pull-down resistor of BOOT0 is about 90 KΩ (cannot be disabled), so there is no need for additional external pull-down when using it.

3.2.6 The pull-down resistors of PA0 pin is automatically enabled in Standby mode

- **Description:**
  In Standby mode, the pull-down resistors of PA0 pin will be automatically enabled by the on-chip control circuit to avoid the floating leakage of the pin.
- **Solution:**
- Please let the PA0 external circuit maintain 0V input level of these two pins, otherwise, the system will consume a small amount of power.

3.2.7 The system clock frequency must not be less than 12 MHz when using USB

- **Description:**
  When the system clock frequency is less than 12 MHz, the USB is unable to transmit or Receive data normally.
- **Solution:**
  To make sure that USB can transmit and receive data normally, the system clock frequency Must not be less than 12 MHz.

3.2.8 Differences between USB double-buffered SW_BUF and DTOG

- **Description:**
  SXX32F103: When the software toggles SW_BUF, it is considered to release a buffer and can continue to transmit or receive data regardless of whether SW_BUF and DTOG conflict or not. The design of SXX32F103 is inconsistent with its documentation.
  AT32403A: The software toggles SW_BUF. When SW_BUF and DTOG are equal, the state is NAK at this time, cannot transmit or receive data. The design completely matches its documentation.
- **Solution:**
  Please refer to the instructions in the AT32F403A Reference Manual.

Figure 5. USB double-buffer instructions

End point	DTOG bit	SW_BUF bit	Buffer used by USB module	Buffer used by application program
IN end point	0	1	ADRn_TX_0/CNTn_TX_0	ADRn_TX_1/CNTn_TX_1
	1	0	ADRn_TX_1/CNTn_TX_1	ADRn_TX_0/CNTn_TX_0
	0	0	NA	ADRn_TX_0/CNTn_TX_0
	1	1	NA	ADRn_TX_0/CNTn_TX_0
OUT end point	0	1	ADRn_RX_0/CNTn_RX_0	ADRn_RX_1/CNTn_RX_1
	1	0	ADRn_RX_1/CNTn_RX_1	ADRn_RX_0/CNTn_RX_0
	0	0	NA	ADRn_RX_0/CNTn_RX_0
	1	1	NA	ADRn_RX_1/CNTn_RX_1

3.2.9 USB_DP has internal pull-up resistor

- **Description:**
  The USB_DP, by default, enables internal pull-up resistors to support full-speed devices. Users can set the PUO bit in the register USB_SOFEN to 1 to disable the internal pull-up.

3.2.10 DAC output glitches

- **Description:**
  Glitches may be generated in the process of using Timer to trigger DAC output waveform to change the BFF bit state to change the driver ability.
- **Solution:**
  The state of BFF bit should be configured before triggering DAC output.

3.2.11 Flash sector erase time is longer

- **Description:**
  Since the Flash sector erase time of AT32F403A is about 50 ms, longer than 22 ms of SXX32F103, this leads to the different time requirements in some applications. If the user enters Flash sector erase immediately after enabling WWDG, the reset may occur before feeding the dog.
- **Solution:**
  For above applications, users should disable WWDG before erasing, and enable WWDG after erasing.

3.2.12 SDIO receive data occurs after RX over run occurs

- **Description:**
  If the related FIFO flags, such as RX FIFO half full, are used to judge the data received, the data received will be abnormal. The reason is that when RX OVER RUN occurs, the AT32F403A will directly enter the IDLE state, and the related FIFO flags will be cleared by hardware, while SXXX32F103 will remain in the receiving state, will not clear FIFO flags. Therefore, the AT32 is more reasonable than SXXX32F103.
- **Solution:**
  Enable RX over run interrupt to restart DPSM。

3.2.13 Enabling I2S slave may cause communication data errors and cannot be recovered automatically

- **Description:**
  If the slave is enabled between the 2nd and 3rd CLK sent by the master, it will cause the current and subsequent data transmission out of order.
- **Solution:**
  Ensure that the slave enabling must be completed before the clock on the master CK pin Arrives.

3.2.14 GPIO has overshooting phenomenon at max 50MHz output

- **Description:**
  For AT32F403A series, when the GPIO port is configured as output mode and the output frequency is selected to be max 50 MHz, there will be overshooting in the output high-frequency signal, which may interfere with the circuit and affect the application.
- **Solution:**
  - Scenario 1: for general use, modify the code to be GPIO_Speed_2 MHz configuration.
  - Scenario 2: for high current drive, users can configure different speed parameters MDEx according to the section 5.3.12 in the AT32F403A datasheet.
- **Example of scenario 1:**
  - For the use of SXX32F103 BSP/Pack, the modification method is shown in the following bond font:

```
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
/* the following modification is for the use of SXX32F103 BSP/Pack */
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_2MHz;
GPIO_Init(GPIOB, &GPIO_InitStructure);
```

- **Example of scenario 2:**
  - Omitted.

3.2.15 The program will hang in a loop that determines CAL control bit as zero when using ADC dual mode

- **Description:**
  - Since the hardware will not clear the CAL control bit, the program will hang in the ADC calibration function while(ADC_GetCalibrationStatus(ADC1)) loop when using ADC dual mode

- **Solution:**
  - Modify the ADC enable and calibration function sequence, such as the original ADC initialization function.

```
ADC_Configuration()
{
...
ADC_Ctrl(ADC1, ENABLE);
...
while(ADC_GetCalibrationStatus(ADC1));//the program will hang here

...
ADC_Ctrl(ADC2, ENABLE);
...
while(ADC_GetCalibrationStatus(ADC2));
...
}
```

  - Modification method:

```
ADC_Configuration()
{
...
ADC_Ctrl(ADC1, ENABLE);
...
ADC_Ctrl(ADC2, ENABLE);
...
while(ADC_GetCalibrationStatus(ADC1));
while(ADC_GetCalibrationStatus(ADC2));
...
...
}
```

3.2.16 EWIF flag cannot be cleared when using WWDG interrupt

- **Description:**
  - When using the WWDG interrupt, the EWIF flag cannot be cleared in the process of entering the interrupt program WWDG_CNT = 0x40, so after entering the interrupt, it is necessary to first feed the dog and then clear the EWIF flag.

- **Solution:**
  - In the WWDG interrupt processing function, first feed the dog, and then clear the EWIF flag.

```
void WWDG_IRQHandler(void)
{
  WWDG_SetCounter(127);
  WWDG_ClearFlag();
}
```

3.2.17 MCO output cannot be stopped in Stop mode

- **Description:**

  After entering Stop Mode, MCO will also have clk output (LSICLK clock frequency), increasing power consumption.

- **Solution:**

  Before entering Stop mode, disable MCO output by software, reconfigure after waking up from Stop mode.

```
/*******First disable MCO output before entering Stop mode********/
RCC_CLKOUTConfig(RCC_CLKOUT_NOCLK, /*User's original desired frequency division configuration*/);
PWR_EnterSTOPMode(PWR_Regulator_ON, PWR_STOPEntry_WFI);
/****** Reconfigure MCO output after waking up from Stop mode******/
RCC_CLKOUTConfig(/*User's original desired MCO clock source*/, /*User's original desired frequency division configuration*/);
```

3.2.18 Systick interrupt abnormality wakes up and executes Stop mode entered by WFE

- **Description:**

  After entering Stop mode, Systick does not stop counting. When the counter reaches 0 interrupt, it will wake up Stop mode by mistake.

- **Solution:**

  Disable Systick before entering Stop mode, then re-enable it when the user exits Stop mode.

```
/********Disable Systick before entering Stop mode********/
SysTick->CTRL &= 0xFFFFFFFE;
PWR_EnterSTOPMode(PWR_Regulator_ON, PWR_STOPEntry_WFE);
/****** Re-enable Systick after waking up from Stop mode ******/
SysTick->CTRL |= 0x1;
```

3.2.19 Data reception error in USART smart card mode

- **Description:**

  When initializing the USART smart card mode, if the USART is enabled first before the smart card mode being enabled, it is not possible to receive data within the first frame of data length after the USART smart card mode is enabled.

- **Solution:**

  Enable the smart card mode before enabling USART.

3.2.20 PWR enters low-power mode through different WFE operation

- AT32F403A adopts Cortex-M4 core, so the WFE operation is a little different, requiring to follow the example provided below.

- **Example:**

```
/* Request Wait For Event */
__SEV();
__WFE();
__WFE();
```

4 Revision history

Table 7. Document revision history

Date	Revision	Changes
2019.12.19	1.0.0	Initial release
2020.02.24	1.0.1	1.Modified system frequency and internal AHB clock frequency up to 240 MHz, and internal APB clock frequency up to 120 MHz 2. Updated the document format.
2020.03.13	1.0.2	Added GPIO waveform comparison chart in V1.0.1 *High-speed GPIO*
2020.04.06	1.0.3	Revised the section *3.2.15 The program will hang in a loop that determines CAL control bit as zero when using ADC dual mode*
2020.04.09	1.0.4	Modified Table 2: *BSP replacement steps*
2020.6.11	1.0.5	Added *3.2.17 MCO output cannot be stopped in Stop mode* Added *3.2.18 Systick interrupt abnormality wakes up and executes Stop mode entered by WFE*
2020.08.11	1.0.6	Added *3.2.19 Data reception error in USART smart card mode*
2021.03.04	1.0.7	Added *3.2.20 PWR enters low-power mode through different WFE operation*
2021.03.11	1.0.8	Modified the description of NOR/PSRAM/SRAM in XMC of *Table 1.*

IMPORTANT NOTICE – PLEASE READ CAREFULLY

Purchasers understand and agree that purchasers are solely responsible for the selection and use of Artery's products and services.

Artery's products and services are provided "AS IS" and Artery provides no warranties express, implied or statutory, including, without limitation, any implied warranties of merchantability, satisfactory quality, non-infringement, or fitness for a particular purpose with respect to the Artery's products and services.

Notwithstanding anything to the contrary, purchasers acquires no right, title or interest in any Artery's products and services or any intellectual property rights embodied therein. In no event shall Artery's products and services provided be construed as (a) granting purchasers, expressly or by implication, estoppel or otherwise, a license to use third party's products and services; or (b) licensing the third parties' intellectual property rights; or (c) warranting the third party's products and services and its intellectual property rights.

Purchasers hereby agrees that Artery's products are not authorized for use as, and purchasers shall not integrate, promote, sell or otherwise transfer any Artery's product to any customer or end user for use as critical components in (a) any medical, life saving or life support device or system, or (b) any safety device or system in any automotive application and mechanism (including but not limited to automotive brake or airbag systems), or (c) any nuclear facilities, or (d) any air traffic control device, application or system, or (e) any weapons device, application or system, or (f) any other device, application or system where it is reasonably foreseeable that failure of the Artery's products as used in such device, application or system would lead to death, bodily injury or catastrophic property damage.