User Manual

PY32_Bootloader_USART_UserManual

Introduction

This application note describes the USART protocol used in the PY32 microcontroller bootloader, providing details on each supported command.

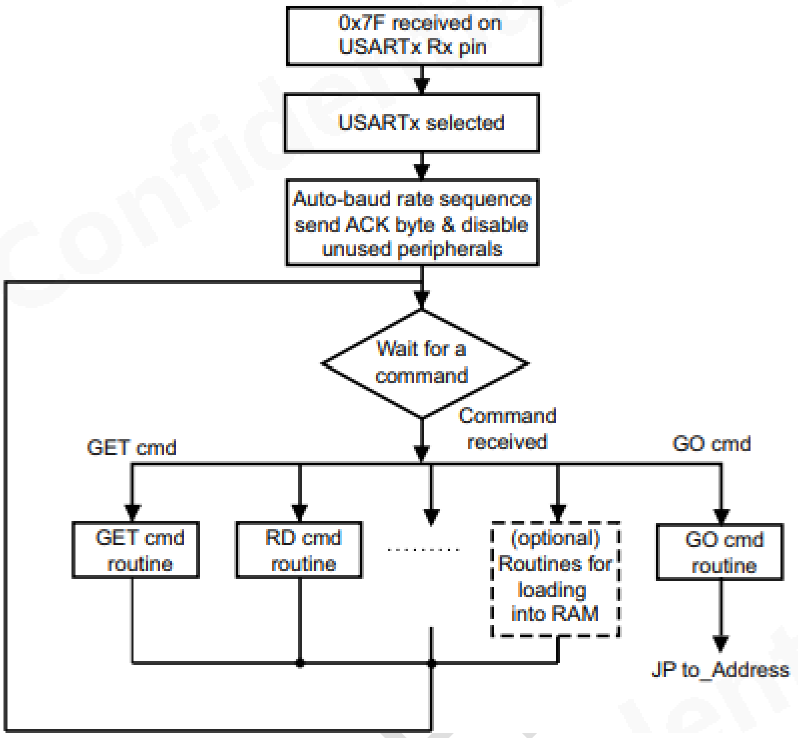
This document applies to PY32 products embedding any bootloader version, as specified in "PY32_Bootloader _UserManual" PY32 system memory boot mode. These products are referred to as PY32 throughout the document.

For more information about the USART hardware resources and requirements for your device bootloader, refer to the already mentioned "PY32_Bootloader _UserManual".

Contents

1 USART bootloader code sequence

Figure 1. Bootloader for PY32 with USART


0x7F received on USARTx Rx pin
USARTx selected
Auto-baud rate sequence send ACK byte & disable unused peripherals
Wait for a command
Command received
GET cmd
GO cmd
GET cmd routine
RD cmd routine
....
(optional) Routines for loading into RAM
GO cmd routine
JP to_Address

Once the system memory boot mode is entered and the PY32 microcontroller (based on Arm®(a) cores) has been configured (for more details refer to Bootloader UserManual), the bootloader code begins to scan the USARTx_RX line pin, waiting to receive the 0x7F Bytes Data frame: a start bit, 0x7F Bytes Data bits, even parity bit, and a stop bit. Depending on the implementation, the baud rate detection is based on the HW (IP supporting auto baud rate), or on the SW. The following paragraphs explain the SW detection mode. The duration of this Bytes Data frame is measured using the Systick timer. The count value of the timer is then used to calculate the corresponding baud rate factor respect to the current system clock. Then, the code initializes the serial interface accordingly. Using this calculated baud rate, an acknowledge byte (0x79) is returned to the host, who signals that the PY32 is ready to receive commands.

2 USART auto baud rate detection

The USART is able to detect and automatically set the USART_BRR register value based on the reception o one character. Automatic baud rate detection is useful under two circumstances:

1) The communication speed of the system is not known in advance
2) The system is using a relatively low accuracy clock source and this mechanism allows the correct baud rate to be obtained without measuring the clock deviation.

The clock source frequency must be compatible with the expected communication speed.

Before activating the auto baud rate detection, the auto baud rate detection mode must be chosen. There are various modes based on different character patterns(They can be chosen through the ABRMOD[1:0] field in the USART_CR3 register). In these auto baud rate modes, the baud rate is measured several times during the synchronization Bytes Data reception and each measurement is compared to the previous one.

These modes are:

Mode 0: Any character starting with a bit at 1. In this case the USART measures the duration of the Start bit (falling edge to rising edge).

Mode 1: Any character starting with a 10xx bit pattern. In this case, the USART measures the duration of the Start and of the 1st Bytes Data bit. The measurement is done falling edge to falling edge, ensuring better accuracy in the case of slow signal slopes.

In parallel, another check is performed for each intermediate transition of RX line. An error is generated if the transitions on RX are not sufficiently synchronized with the receiver (the receiver being based on the baud rate calculated on bit 0).

Prior to activating auto baud rate detection, the USART_BRR register must be initialized by writing a non-zero baud rate value.

The automatic baud rate detection is activated by setting the ABREN bit in the USART_CR2 register. The USART will then wait for the first character on the RX line. The auto baud rate operation completion is indicated by the setting of the ABRF flag in the USART_ISR register. If the line is noisy, the correct baud rate detection cannot be guaranteed. In this case the BRR value may be corrupted and the ABRE error flag will be set. This also happens if the communication speed is not compatible with the automatic baud rate detection range (bit duration not between 16 and 65536 clock periods (oversampling by 16) and not between 8 and 65536 clock periods (oversampling by 8)).

The RXNE interrupt will signal the end of the operation. At any later time, the auto baud rate detection may be relaunched by resetting the ABRF flag (by writing a 0).

Note: If the USART is disabled (UE = 0) during an auto baud rate operation, the BRR value may be corrupted.

3 Bootloader command set

The supported commands are listed in Table 3-1, all of them are described in this section.

Table 3-1. USART bootloader commands

command	Code	Description
Get	0x00	Gets the version and the allowed commands supported by the current version of the protocol.
Get ID	0x02	Gets the chip ID.
Read Memory	0x11	Reads up to 256 Bytes Number of memory starting from an address specified by the application.
Go	0x21	Jumps to user application code located in the internal flash memory or in the SRAM.
Write Memory	0x31	Writes up to 256 Bytes Number to the RAM or flash memory starting from an address specified by the application.
Erase Memory	0x44	Erases from one to all the flash memory pages using two-byte addressing mode (available only for USART bootloader v3.0 and higher).

1. If a denied command is received or an error occurs during the command execution, the bootloader sends an NACK byte, and goes back to command checking.

Communication safety

The communication from the programming tool (PC) to the device is verified by
1. checksum: received blocks of Bytes Data Bytes Number are XOR-ed. A byte containing the computed XOR of all previous Bytes Number is added to the end of each communication (checksum byte). By XOR-ing all received Bytes Number (Bytes Data plus checksum), the result at the end of the packet must be 0x00.
2. for each command, the host sends a byte and its complement (XOR = 0x00)
3. UART: parity check active (even parity).
Each packet is either accepted (ACK answer), or discarded (NACK answer):
• ACK = 0x79
• NACK = 0x1F

3.1 Get command

This command allows the user to get the protocol version and the supported commands. When the bootloader receives the command, it transmits the protocol version and the supported command codes to the host, as shown in Table 3.2-1.

Table 3.2-1. Get command: host side(PC), device side(PY32)

Host/Device	Bytes Number	Bytes Data	Description
host side	1	0x00	Get command
	2	0xFF	command checksum
device side	1	0x79	ACK
	2	0x06	N = the number of bytes to follow – 1, except current and ACKs
	3	0x10	Protocol version (0 < version < 255), example: 0x10 = version 1.0
	4	0x00	Get command
	5	0x02	Get ID command
	6	0x11	Read Memory command
	7	0x21	Go command
	8	0x31	Write Memory command
	9	0x44	Extended Erase command
	10	0x79	ACK

3.2 Get ID command

This command is used to get the version of the chip ID (identification). When the bootloader receives the command, it transmits the product ID to the host.

Table 3.3-1. Get ID command: host side(PC), device side(PY32)

Host/Device	Bytes Number	Bytes Data	Description
host side	1	0x02	Get ID command
	2	0xFD	command checksum
device side	1	0x79	ACK
	2	0x01	N = the number of bytes to follow – 1, except current and ACKs
	3	0x00	PID MSB
	4	0x64	PID LSB
	5	0x79	ACK

3.3 Read Memory command

This command is used to read data from any valid memory address (refer to the product datasheets and to Bootloader UserManual for more details) in RAM, flash memory, and in the information block (system memory or option byte areas).

When the bootloader receives the command, it transmits the ACK byte to the application. After the transmission of the ACK byte, the bootloader waits for an address (four bytes, byte 1 is the address MSB and byte 4 is the LSB) and a checksum byte, then it checks the received address. If the address is valid and the checksum is correct, the bootloader transmits an ACK byte, otherwise it transmits an NACK byte, and aborts the command.

When the address is valid and the checksum is correct, the bootloader waits for the number of bytes to be transmitted – 1 (N bytes) and for its complemented byte (checksum). If the checksum is correct it then transmits the needed data ((N + 1) bytes) to the application, starting from the received address. If the checksum is not correct, it sends an NACK before aborting the command.

Table 3.4-1. Read Memory command: host side(PC), device side(PY32)

Host/Device	Bytes Number	Bytes Data	Description
host side	1	0x11	Read Memory command
	2	0xEE	command checksum
device side	1	0x79	ACK
host side	1~4	-	Start address byte 1: MSB, byte 4: LSB
	5	-	Checksum: XOR (byte 1, byte 2, byte 3, byte 4)
device side	1	0x79	ACK
host side	1	-	The number of bytes to be read – 1 (0 < N ≤ 255)
	2	-	Checksum: XOR byte 8 (complement of byte 8)
device side	1	0x79	ACK
	2~(N+2)	-	Send data to the host（N+1 Bytes）

3.4 Go command

This command is used to execute the downloaded code or any other code by branching to an address specified by the application. When the bootloader receives the command, it transmits the ACK byte to the application. After the transmission of the ACK byte, the bootloader waits for an address (four bytes, byte 1 is the address MSB and byte 4 is LSB) and a checksum byte, then it checks the received address. If the address is valid and the checksum is correct, the bootloader transmits an ACK byte, otherwise it transmits an NACK byte, and aborts the command.

When the address is valid and the checksum is correct, the bootloader firmware:

• initializes the registers of the peripherals used by the bootloader to their default reset values

• initializes the main stack pointer of the user application

• jumps to the memory location programmed in the received address + 4 (corresponding to the address of the application reset handler). For example, if the received address is 0x0800 0000, the bootloader jumps to the memory location programmed at address 0x0800 0004.

In general, the host must send the base address where the application to jump to is programmed.。

Table 3.5-1. Go command: host side(PC), device side(PY32)

Host/Device	Bytes Number	Bytes Data	Description
host side	1	0x21	Go command
	2	0xDE	command checksum
device side	1	0x79	ACK
host side	1~4	-	Start address (byte 1: MSB, byte 4: LSB)
	5	-	Checksum: XOR (byte 1, byte 2, byte 3, byte 4)
device side	1	0x79	ACK

3.5 Write Memory command

This command is used to write data to any valid memory address (see note below), such as RAM, flash memory, or option byte area.

When the bootloader receives the command, it transmits the ACK byte to the application. After the transmission of the ACK byte, the bootloader waits for an address (four bytes, byte 1 is the address MSB and byte 4 is the LSB) and a checksum byte, it then checks the received address. For the option byte area, the start address must be the base address of the option byte area (see note below) to avoid writing inopportunely in this area.

If the received address is valid and the checksum is correct, the bootloader transmits an ACK byte, otherwise it transmits an NACK byte, and aborts the command. When the address is valid and the checksum is correct, the bootloader

• gets a byte, N, which contains the number of data bytes to be received

• receives the user data ((N + 1) bytes) and the checksum (XOR of N and of all data bytes)

• programs the user data to memory starting from the received address

• at the end of the command, if the write operation was successful, the bootloader transmits the ACK byte; otherwise it transmits an NACK byte to the application and aborts the command.

The maximum length of the block to be written for the PY32 is 256 bytes.

If the Write Memory command is issued to the option byte area, all bytes are erased before writing the new values, and at the end of the command the bootloader generates a system reset to take into account the new configuration of the option bytes.

Note:

When writing to the RAM, user must not overlap the first area used by the bootloader firmware.

No error is returned when performing write operations on write-protected sectors. No error is returned when the start address is invalid.

Table 3.6-1. Write Memory command: host side(PC), device side(PY32)

Host/Device	Bytes Number	Bytes Data	Description
host side	1	0x31	Write Memory command
	2	0xCE	command checksum
device side	1	0x79	ACK
host side	1~4	-	Start address (byte 1: MSB, byte 4: LSB)
	5	-	Checksum: XOR (byte 1, byte 2, byte 3, byte 4)
device side	1	0x79	ACK
host side	1	-	Number of bytes to be received (0 < N ≤ 255)
	2~N+2		N +1 data bytes: Max 256 bytes
	N+3	-	Checksum byte: XOR (N, N+1 data bytes)
device side	1	0x79	ACK

3.6 Erase Memory command

This command allows the host to erase flash memory pages using two bytes addressing mode. When the bootloader receives the command, it transmits the ACK byte to the host. After the transmission of the ACK byte, the bootloader receives two bytes (number of pages to be erased), the fl/ash memory page codes (each one coded on two bytes, MSB first) and a checksum byte (XOR of the sent bytes); if the checksum is correct, the bootloader erases the memory and sends an ACK byte to the host. Otherwise, it sends an NACK byte to the host, and the command is aborted.

Extended Erase Memory command specifications:

1. The bootloader receives one half-word (two bytes) that contains N, the number of pages to be erased:

a) For N = 0xFFFY (where Y is from 0 to F) special erase is performed:

- 0xFFFF for global mass erase

b) $N_{(MSB)}$ = 0x10 Page erase, 0⩽ $N_{(LSB)}$< Max Page Number: Erase $N_{(LSB)}$+ 1 Pages.

c) $N_{(MSB)}$ = 0x20 Sector erase, 0⩽$N_{(LSB)}$< Max Sector Number: Erase $N_{(LSB)}$+ 1 Sectors.

2. The bootloader receives:

a) In the case of a special erase, one byte: checksum of the previous bytes:

- 0x00 for 0xFFFF

b) $N_{(MSB)}$ = 0x10 Page erase, 0⩽ $N_{(LSB)}$< Max Page Number: Erase $N_{(LSB)}$+ 1 Pages.

c) $N_{(MSB)}$ = 0x20 Sector erase, 0⩽$N_{(LSB)}$< Max Sector Number: Erase $N_{(LSB)}$+ 1 Sectors.

Note:

No error is returned when performing erase operations on write-protected sectors.

The maximum number of pages is relative to the product and must be respected.

Table 3.3-1. Erase Memory command: host side(PC), device side(PY32)

Host/Device	Bytes Number	Bytes Data	Description
host side	1	0x44	Erase Memory command
	2	0xBB	command checksum
device side	1	0x79	ACK
host side	1	0x20	Sector Erase Code（Page Erase Code is 0x10）
	2	-	Number of sectors to be erased (N+1 where 0 ⩽ N < Maximum number of sectors).
	3~2*(N+2)		(2 x (N + 1)) bytes (sectors numbers coded on two bytes MSB first)
	2*(N+2)+1		checksum of all sent bytes (N (2 bytes), 2x(N+1) bytes)
device side	1	0x79	ACK

4 Version history

Version	Date	Description
V1.0	2023-04-20	Initial Version

Puya Semiconductor Co., Ltd.