Bluetooth Low Energy (BLE)
Transparent Transmission Controller Programming User Guide

BC7601/BC7602/BC32F7611

Revision: V1.00 Date: December 28, 2018

www.holtek.com

Table of Contents

Introduction

Overview

The series of BLE devices discussed in this guide are Holtek's fully integrated single chip Bluetooth Low Energy transparent transmission controllers. The devices require a driver code to activate the BLE function for full BLE optimization. This driver code can be downloaded from an EEPROM or MCU for which the driver source can be selected by the IIC_SDA line status during the reset period. The driver interface can also be selected to be the UART or SPI interface determined by the SPI-UR_N line status during the reset period. The IIC_SDA and SPI-UR_N line level will be detected after the reset has completed. Therefore these two lines should be fixed at a certain level before the RST_N line goes high.

As these two lines selections are independent, there will be four driver modes as shown in the following table.

Driver Mode	Driver Code Description
Mode_A	Driver from the MCU using the SPI interface – 8K driver code memory capacity
Mode_B	Driver from the EEPROM using the SPI interface
Mode_C	Driver from the MCU using the UART interface – 8K driver code memory capacity
Mode_D	Driver from the EEPROM using the UART interface

Device Summary

The supported mode types is dependent upon which device is selected as shown in the following table.

Driver Mode / Device	Mode_A	Mode_B	Mode_C	Mode_D
BC7601	√	—	√	—
BC7602	—	√	—	√
BC32F7611	√	—	—	—

Pin Description

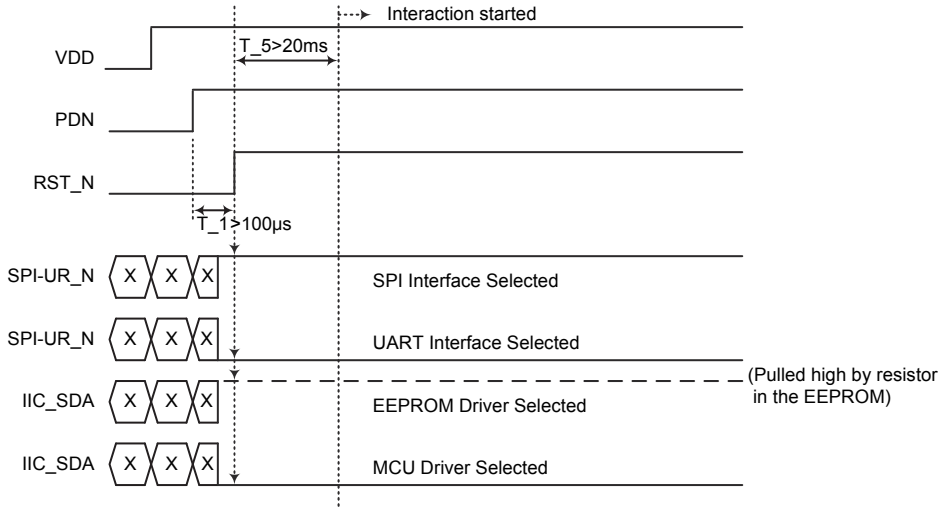
Name	Direction	Level	Description
RST_N	Input	0: Reset 1: Not reset	
PDN	Input	0: Power down 1: Not power down	If the PDN pin state is changed from low to high, it is necessary to reset the device and driver again. Note that the PDN pin cannot be connected to other I/O pins.
IIC_CLK	Input/ Output	1: Baud rate=115200 0: Baud rate=9600	In Mode_C, IIC_CLK pin is used to select the default baud rate. The IIC_CLK pin level should be fixed during a reset as it will be checked after the reset.
EE_WP	Output	0: EEPROM Write Protect Disable 1: EEPROM Write Protect Enable	Used for an EEPROM write protect. The MCU should not change the EE_WP pin level.
IIC_SDA	Input/ Output	0: Driver from MCU Pull-high resistor contained in EEPROM: Driver from EEPROM	The device will check the EEPROM after a reset.
STATE	Output	0: Device sleep 1: Device active	Device state indicator
INT_EXT	Output	0: Valid data ready 1: No data ready	SPI interrupt – software
SPI_INT	Output	0: Data valid 1: No data valid	SPI interrupt – hardware

Name	Direction	Level	Description
WAKEUP	Input	0: Enter Sleep mode 1: Wake up the device (delay<3ms)	MCU controls the WAKEUP pin WAKEUP=1 → STATE=1 when active
SPI-UR_N	Input	0: UART interface 1: SPI interface	Pin level should be fixed during a reset and checked after the reset
SPI_MOSI /UART_RXD	SPI/UART	SPI/UART	Driver interface selected by SPI-UR_N If the SPI-UR_N pin is pulled high the SPI interface pins are selected. If the SPI-UR_N pin is pulled low the UART interface pins are selected.
SPI_MISO /UART_TXD	SPI/UART	SPI/UART	
SPI_CS /UART_CTS	SPI/UART	SPI/UART	
SPI_CLK /UART_RTS	SPI/UART	SPI/UART	

Pin Functional Description

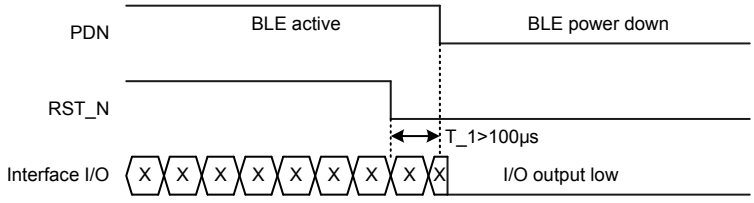
Reset Pin – RST_N

The RST_N pin must be kept low for a time period greater than 100μs after the PDN pin is set high when a power-on reset occurs. Then the device can boot from the EEPROM or MCU by reading a driver code using the SPI or UART interface. After a boot time period, denoted as T_5 in the following diagram and greater than 20 ms, then interaction can be initiated between the MCU and BLE device.



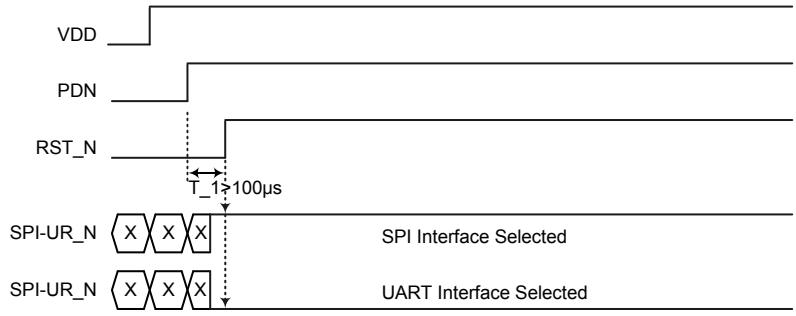
Power Down Control Pin – PDN

If the PDN pin is cleared to low the device will enter the power down mode. The RST_N and all interface I/O pins must be cleared to low. The Interface I/O includes the SPI_CS/UR_CTS, SPI_CLK/UR_RTS, SPI_MOSI/UR_RXD and SPI_MISO/UR_TXD lines.



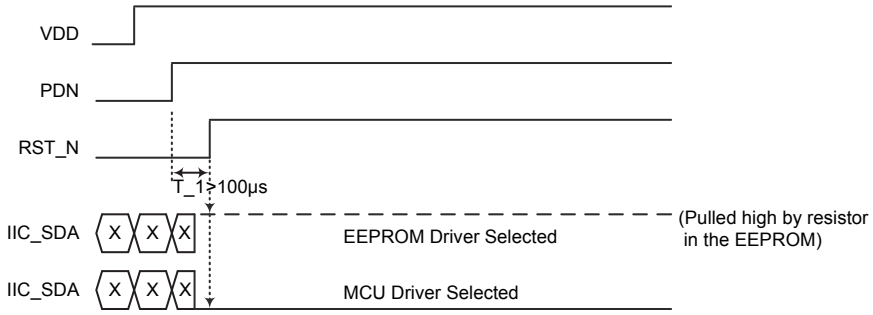
Interface Select Pin – SPI-UR_N

The SPI-UR_N pin is used to select which interface is used for data transfer. The SPI-UR_N pin status will be checked after a reset after which the interface selected can be determined. If the SPI-UR_N pin status is high the SPI interface will be selected while the UART interface will be selected if the SPI-UR_N pin status is low.



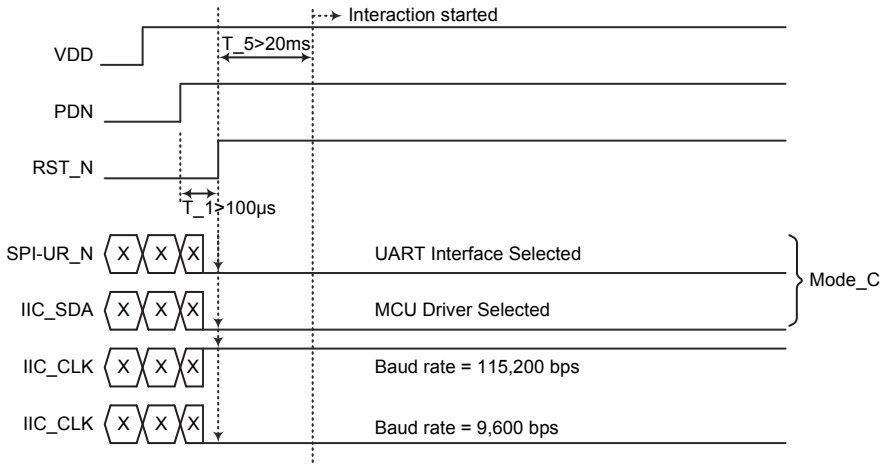
IIC_SDA Pin

A driver code can be sourced from either the EEPROM or MCU, which is selected by the IIC_SDA line status during the reset period. If the IIC_SDA line is externally fixed to low during the reset period, the driver code will be derived from the MCU after the reset completes. However the driver code will be sourced from the EEPROM after the reset completes if the IIC_SDA line is connected to the EEPROM SDA pin and pulled high by a resistor contained within the EEPROM.



IIC_CLK Pin

The IIC_CLK pin is used to select the default baud rate when the device operates in Mode_C, which means the driver code is derived from the MCU using the UART interface. The UART default baud rate is equal to 115,200 bps when the IIC_CLK pin status is set high. Otherwise, the UART default baud rate is equal to 9,600 bps as the IIC_CLK pin status is set low.

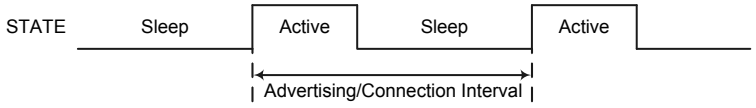


EEPROM Write Protect Control Pin – EE_WP

The EE_WP pin is the EEPROM write protect control pin which is use to control the EEPROM write protect function and only available for the device which is driven by the EEPROM memory. This pin should be kept in a floating status and also the host MCU should not change the pin status for devices driven by the MCU.

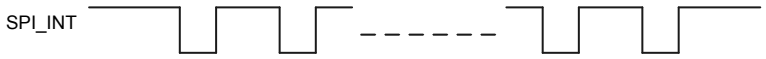
IC State Indicator Pin – STATE

The STATE pin is used to indicate that the device is in the operating or sleep mode. The external host MCU can read this pin status to know the BLE device condition. When the STATE pin status is high this indicates that the device is active while if the STATE pin status is low this indicates that the device is in the sleep mode.



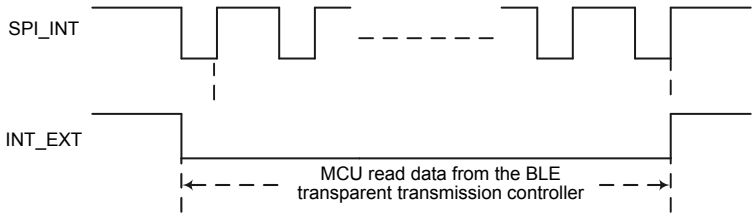
SPI Interrupt Pin – SPI_INT

The SPI_INT line signal is used to inform the MCU that data in the SPI buffer is available. The corresponding interrupt enable control bit in the control register, CR2, should first be set high before the SPI_INT signal is used. The MCU can obtain data from the BLE transparent transmission controller when the MCU receives an SPI_INT active signal. The BLE transparent transmission controller will generate a valid trigger signal on the SPI_INT line once data in the buffer is available.



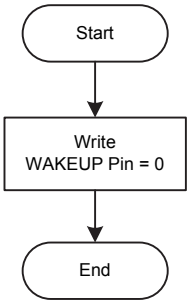
External Interrupt Pin – INT_EXT

The INT_EXT line signal is used to inform the MCU that data is available. The MCU can obtain data from the BLE transparent transmission controller when the MCU receives an INT_EXT active signal. The INT_EXT active level can be determined by configuring the accessible physical address 0x0020_0494[1]. The BLE transparent transmission controller will keep the INT_EXT line at an active level until there is no more available data in the buffer.

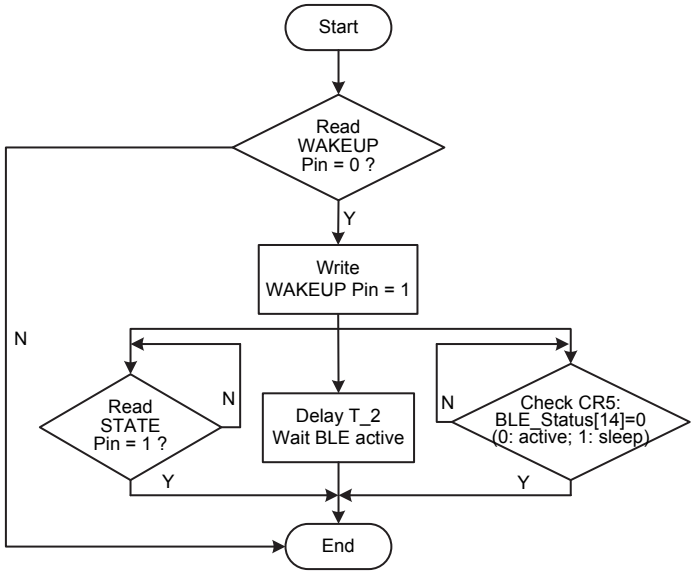


Wake-up Pin – WAKEUP

The WAKEUP pin is used to select the device operation mode while the STATE pin is used to indicate the device operation status. The external host MCU can check the device operation mode by monitoring the STATE pin. When the WAKEUP pin is pulled low, the device will enter the Sleep mode and the STATE pin will go low. If the device is in the Sleep Mode, it can be woken up using the WAKEUP pin. When the WAKEUP pin is pulled high, the device will be woken up and the STATE pin will go high.



Sleep Flow

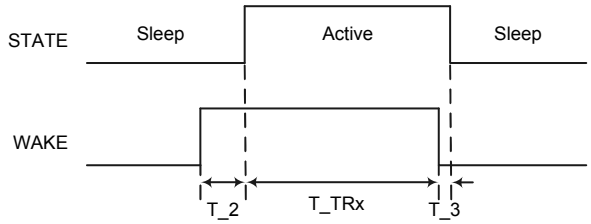


Wakeup Flow

Host Start Interaction

When the host MCU sets the WAKEUP pin high, the device will set the STATE pin high after a delay time, T_2, as shown in the following diagram. After this the BLE device will be in the active mode. The host MCU can read the BLE status register, CR5, bit 14 to check that the BLE device is in the sleep or active mode when using the SPI interface. It is recommended that the host MCU should wait for a delay time, T_2, greater than 3 ms when using the UART interface. When the device is in the active mode, the host MCU can transmit or receive packets.

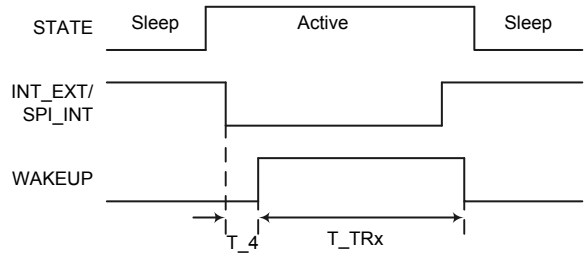
The packet will be ignored if the host MCU send a packet when the BLE device is in the sleep mode. If the host MCU sends packets during the T_2 delay time period, the BLE device will respond with a value of 26FF1C to the host to indicate that the packet type is not supported or with other error messages. After the interaction has completed the host MCU should clear the WAKEUP pin low to inform the BLE device to enter the sleep mode again.



Note: T_2 = 3ms(typ.); T_3 = 0.1ms(typ.)

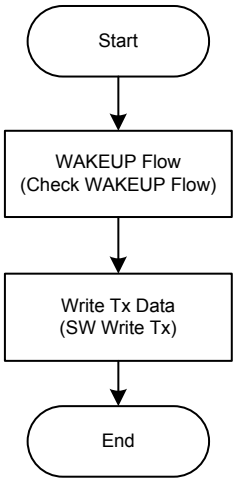
BLE Start Interaction

If the BLE device wishes to send packet to the external host, the BLE device should first wait until the STATE pin goes high which means that the BLE device is in the active mode. The BLE device should send an INT_EXT or SPI_INT signal to inform the host MCU after the STATE pin goes high. The host MCU should first set the WAKEUP pin high to ensure that the packet can be transferred completely. After this the host MCU can transmit or receive packets when the BLE device is in the active mode. If the packet is transferred by the host MCU before setting the WAKEUP pin high, i.e. during the T_4 period, the BLE device will respond with an error message to the host MCU. After the interaction has completed the host MCU should set the WAKEUP pin low to inform the BLE device to enter the sleep mode again. Note that the INT_EXT signal active level can be determined by configuring the accessible physical address 0x0020_0494[1] while the SPI_INT signal is always active low.



Transmission Flow – TX Flow

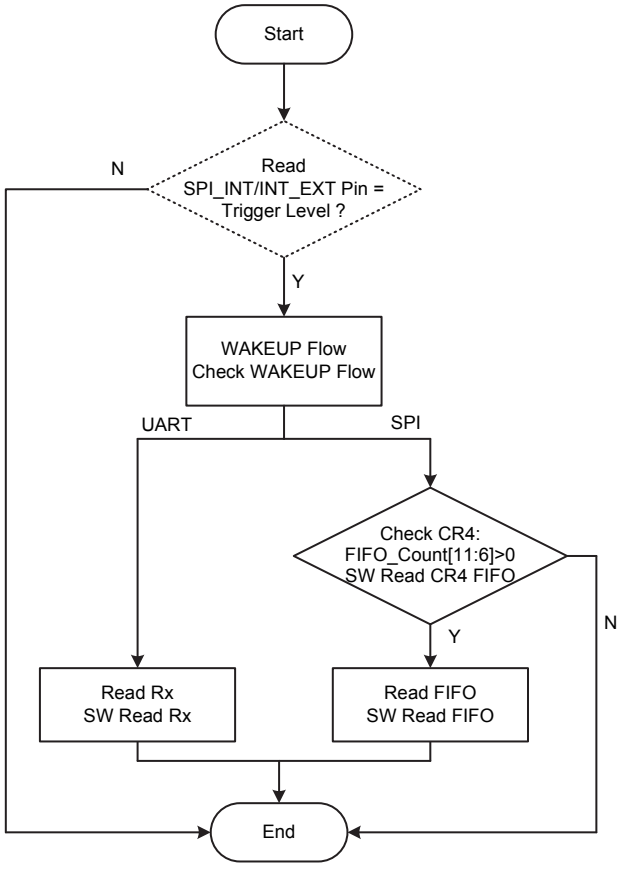
The MCU can transmit the Command or Payload to the BLE device using the corresponding format.



TX Flow

Reception Flow – RX Flow

The MCU can receive an Event or Payload from the BLE device using the corresponding format. Note that the SPI_INT/ INT_EXT check procedure can be ignored when using the UART interface.



RX Flow

BLE Device Command / Event

The BLE device supports several commands and events for communication with the host MCU. Some GATT services are also supported to communicate with cell phones. To more conveniently describe the command/event format several abbreviations are used which are listed in the following table.

Abbreviation	Explanation
BD	Bluetooth Device
BD_Addr	Bluetooth Device Address
BD_Name	Bluetooth Device Name

Abbreviation Summary

GATT Service Description

The BLE device supports two types of service for the communication with cell phones. The cell phone can access these services using the BLE protocol.

Service	Characteristics	Property	Data Type	Data Length	Description
Battery (UUID 0x180F)	Battery Level (UUID 0x2A19)	Read / Notify	8 bits	1 byte	Device Battery Level The cell phone sends a read command to the device after which the device battery level value will be sent to the cell phone.
Manufacture Define (UUID 0xFFF0)	Read (UUID 0xFFF1)	Notify	8 bits	20 bytes	Payload sent from the BLE device to the cell phone. If the cell phone enables the device “Notify” function, the BLE device will send the payload to the cell phone whenever the data is ready.
	Write (UUID 0xFFF2)	Write without Response	8 bits	20 bytes	Payload is sent from the cell phone to the BLE device without waiting for a device acknowledge.

Command (Cmd) / Event (Evt) Format

Both commands and events are supported by the BLE device to communicate with the host MCU. The write commands are used to configure the BLE device by the host MCU while the read commands are used to retrieve the information from the BLE device. The corresponding events are the responses which are returned from the device to the host MCU.

Type	Dir.	Header	Payload									
		1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th	...	Nth
Read Command	M→S	0x20	Opcode (byte*1)										
Read Event	M←S	0x21	Opcode (byte*1)	Length (byte*1)	Data (byte*Length, MSB)								
Payload Packet Command	M↔S	0x22	Length (byte*1 max=200)	RF Payload (byte*Length, MSB)									
Write Command	M→S	0x25	Opcode (byte*1, unit=byte*1)	Length (byte*1)	Data (byte*Length, MSB)								
Write Event/ Payload Packet Event	M←S	0x26	Opcode (byte*1, unit=byte*1)	Result (byte*1)									

Type	Dir.	Header	Payload									
		1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th	...	Nth
Write Physical Address	M→S	0x55	Length (byte*1, unit=byte*4, max=60)	Reserved (byte*2)		Address (byte*4, LSB)				Data (byte*Length*4, LSB)			
Read Physical Address	M→S	0x56	Length (byte*1, unit=byte*4, max=60)	Address (byte*4)									
Read Physical Address Return	M←S	0x57	Length (byte*1, unit=byte*4, max=60)	Reserved (byte*2)		Address (byte*4, LSB)				Data (byte*Length*4, LSB)			

Note: 1. The "Dir." word means the transfer direction.
2. "M→S": from Master to Slave;
"M←S": from Slave to Master;
"M↔S": both from Master to Slave and from Slave to Master;
where "M" is the MCU and "S" is the BLE device.
3. "Reserved" means 0x00.

Read Command (Cmd) / Event (Evt) Format

These commands are used to retrieve information from the device. The corresponding events are the information which is returned from the device to the host MCU.

Read Cmd/Evt	Header	Opcode	Length	Data							
	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	...	Nth
IntvRead Cmd	0x20	0x30									
IntvRead Evt	0x21	0x30	0x04	Connection Interval (4 bytes, Unit: 1ms, Valid Range=8~4000, LSB)							
Description: Read connection interval setting between the device and cell phone. Only available when connected.											
BDNameRead Cmd	0x20	0x31									
BDNameRead Evt	0x21	0x31	Length (up to 16 bytes)	BD_Name (up to 16 bytes, MSB)							
Description: Read Bluetooth Device Name											
BaudRateRead Cmd	0x20	0x32									
BaudRateRead Evt	0x21	0x32	0x04	BaudRate (4 bytes, LSB) 0: 2400 1: 9600 2: 14400 3: 19200 4: 38400 5: 57600 6: 115200 7: 256000							
Description: Read the UART baudrate. Only valid when using the UART interface.											
BDAddrRead Cmd	0x20	0x33									
BDAddrRead Evt	0x21	0x33	0x06	BD_Addr (6 bytes, public, LSB)							
Description: Read Bluetooth Device public Address.											
AdvIntvRead Cmd	0x20	0x35									

Read Cmd/Evt	Header	Opcode	Length	Data							
	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	...	Nth
AdvIntvRead Evt	0x21	0x35	0x04	Adv Interval (4 bytes, Unit: 1 ms, Valid Range=20~10000, LSB)							
Description: Read Bluetooth Device Advertising Interval											
AdvDataRead Cmd	0x20	0x36									
AdvDataRead Evt	0x21	0x36	Length (up to 25 bytes)	Adv Data (up to 25 bytes, manufacturer specific data field value, MSB)							
Description: Read manufacturer specific data field value of the advertising packet Ex: Advertising packet=020106_06FF04095E5F60 020106: 02: length=2 01: type=Flags 06: value=LE General Discoverable Mode/BR/EDR not supported 06FF04095E5F60: 06: length=6 FF: type=Manufacturer Specific Data 04095E5F60: value=(Adv Data)											
WhiteListRead Cmd	0x20	0x37									
WhiteListRead Evt	0x21	0x37	0x06	Connectable device address (white list) (6 bytes, LSB)							
Description: Read WhiteList setup value. If the external device address is different from the white list, it cannot be connected to the BLE device. If the WhiteList are all set to 0x00, all external devices can be connected to the BLE device.											
TxPowerRead Cmd	0x20	0x38									
TxPowerRead Evt	0x21	0x38	0x01	Pwr (1 byte) 0: 3dBm 1: 0dBm Others: Reserved							
Description: Read the BLE device RF TX power setup value.											

Payload Format

The BLE device receives the payload stream from the host MCU with a length of up to 200 bytes. However, the whole payload stream received from the host MCU will be divided into several payload packets with each packet having a length of 27 bytes. These packets with a length of up to 27 bytes will be sent to the cell phone.

Payload Packet Cmd	Header	Length /Result	RF Payload							
	1st	2nd	3rd	4th	5th	6th	7th	8th	...	Nth
Payload Packet Command	0x22	Length (up to 200 bytes)	RF Payload (MSB)							

Payload Packet Evt	Header	Opcode	Length /Result							
	1st	2nd	3rd							
Payload Packet Evt	0x26	0x22	Result 0: ok others: fail							

Description:
The payload length should ≤ 200 bytes.
The host MCU can send payload packet to the device with the length ≤ 200 bytes.
The BLE device air payload packet length cannot be greater than 27 bytes. Therefore if the MCU sends a payload packet with a length greater than 27 bytes, the device will automatically partition the whole payload stream into several packets with each air payload packet having a length of 27 bytes and then send these packets to the cell phone. Finally the device will send an event packet message to the host MCU after all payload packets have been received.

Example: The MCU sends 50 bytes payload to the BLE device after which the payload packet will be sent to the cell phone.

MCU	UART/SPI (wired)	BLE Device	BLE packet (air)	Cell Phone
22_32_616263646566676869 6A6B6C6D6E6F7071727374 75767778797A7B7C7D7E 7F808182838485868788 898A8B8C8D8E8F909192	→			
		(Header: 7bytes) + 6162636465666768696A6B6C6D6E 6F7071727374	→	
		75767778797A7B7C7D7E 7F808182838485868788 898A8B8C8D8E8F	→	
		909192	→	
	←	262200		

Write Command (Cmd) / Event (Evt) Format

These commands are used to configure the device parameters. The corresponding events are the results after the configuration has taken place.

Write Cmd/Evt	Header	Opcode	Length /Result	Data							
	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	...	Nth
BDNameWrite Cmd	0x25	0x31	Length (up to 16 bytes)	BD_Name (up to 16 bytes, MSB)							
BDNameWrite Evt	0x26	0x31	Result 0: ok others: fail								
Description: Setup the device friendly name.											
BaudRateWrite Cmd	0x25	0x32	0x04	BaudRate (4 bytes, LSB) 0: 2400 1: 9600 2: 14400 3: 19200 4: 38400 5: 57600 6: 115200 7: 256000							
BaudRateWrite Evt	0x26	0x32	Result 0: ok others: fail								
Description: Setup the device baudrate is only available when using the UART interface. The baudrate will be updated at the next reboot or after a BaudRateUpdate command has been executed.											
BDAddrWrite Cmd	0x25	0x33	0x06	BD_Addr (6 bytes, public, LSB)							
BDAddrWrite Evt	0x26	0x33	Result 0: ok others: fail								
Description: Setup the Bluetooth device public address.											
AdvIntvWrite Cmd	0x25	0x35	0x04	Adv Interval (Unit: 1 ms, Valid Range=20~10000, LSB)							
AdvIntvWrite Evt	0x26	0x35	Result 0: ok others: fail								
Description: Setup the advertising interval.											
AdvDataWrite Cmd	0x25	0x36	Length (up to 25 bytes)	Adv Data (up to 25 bytes, Manufacturer specific data field value, MSB)							
AdvDataWrite Evt	0x26	0x36	Result 0: ok others: fail								
Description: This command is used to setup only the “Manufacturer Specific Data” field value of the advertising packet and should be used before the Driver_B execution. Ex: Advertising data packet=020106 06FF04095E5F60, Adv Data=04095E5F60. 020106: 02: length=2 01: type=Flags 06: value=LE General Discoverable Mode/BR/EDR not supported 06FF04095E5F60: 06: length=6 FF: type=Manufacturer Specific Data 04095E5F60: value=Adv Data											
WhiteListWrite Cmd	0x25	0x37	0x06	Connectable device address – white list (6 bytes, LSB)							
WhiteListWrite Evt	0x26	0x37	Result 0: ok others: fail								
Description: Write the WhiteList setup value. If the external device address is different from the white list, it cannot connect to the BLE device. If the WhiteList is set to 0x00 then all external devices can connect to the BLE device.											

Write Cmd/Evt	Header	Opcode	Length /Result	Data							
	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	...	Nth
TxPowerWrite Cmd	0x25	0x38	0x01	TxPwr (1 byte) 0: 3dBm 1: 0dBm Others: Reserved							
TxPowerWrite Evt	0x26	0x38	Result 0: ok others: fail								
Description: Configure the device RF TX power setting value.											
BatteryLevelWrite Cmd	0x25	0x3B	0x01	Battery Level (1 byte)							
BatteryLevelWrite Evt	0x26	0x3B	Result 0: ok others: fail								
Description: The host MCU detects the device battery level and stores the value using this command. The cell phone can obtain the battery level value using the GATT "Battery" service supported by the BLE device.											
BaudRateUpdate Cmd	0x25	0x3F	0x00	*Old baudrate							
BaudRateUpdate Evt	0x26	0x3F	0x00	*New baudrate							
Description: This command is used to activate a new baudrate update. It is only available when using the UART interface. The new baudrate value is written into the BLE device by the host MCU using BaudRateWrite command. However, the new baudrate value will not be available until the BaudRateUpdate command is executed. After this update command is executed the desired new baudrate will be available and the BaudRateUpdate event will be sent out by the BLE device driven by the new baudrate. If the BaudRateUpdate event is not successfully received by the host MCU the MCU should use other read commands to synchronize the UART interface again. Note that a new baudrate will be updated and available when the next reboot occurs if this command is not used.											
IntvLatencyWrite2 Cmd	0x25	0x40	0x08	MinInterval (2 bytes, unit=1.25ms, LSB)	MaxInterval (2 bytes, unit=1.25ms, LSB)	0x00	0x00	Timeout (2 bytes, unit=10ms, LSB)			
IntvLatencyWrite2 Evt	0x26	0x30	0x00								
Description: This command is used to set the BLE device connection interval and latency together with the time-out period. It is only available after connection. There will be an error event if the device is disconnected with the cell phone. After the setup process has completed, the corresponding event packet will be returned. For example, when this command is executed with a total configuration of "0x25_40_08_0800_1000_00_00_5802", the iphone sytem will prefer to use the maximum interval setting of 0x10 while the android system will prefer to use the minimum interval setting of 0x08. The following interval settings are the configurations for users to connect with different system cell phones. 0x25_40_08_0800_1000_00_00_5802, //10~20ms 0x25_40_08_4000_5000_00_00_5802, //80~100ms 0x25_40_08_5000_7800_00_00_5802, //100~150ms 0x25_40_08_7800_A000_00_00_5802, //150~200ms 0x25_40_08_A000_C800_00_00_5802, //200~250ms 0x25_40_08_C800_F000_00_00_5802, //250~300ms 0x25_40_08_2003_B004_00_00_5802, //1000~1500ms 0x25_40_08_B004_4006_00_00_5802, //1500~2000ms											
AdvDataWrite2 Cmd	0x25	0x50	Length max=31, 0=no action	Adv Data (up to 31 bytes, whole Advertising Data field value, MSB)							
AdvDataWrite2 Evt	0x26	0x50	Result 0: ok others: fail								
Description: This command is used to setup the whole advertising data packet fields and can only be used after a Driver_B execution. Note that the "AdvDataWrite" command is used to modify only the "Manufacturer Specific Data" field value while the "AdvDataWrite2" command is used to modify the whole Advertising data field value.											
ScanResDataWrite Cmd	0x25	0x51	Length max=31 0=no action	ScanRes Data (up to 31 bytes, whole Scan Response Data field value, MSB)							

Write Cmd/Evt	Header	Opcode	Length /Result	Data							
	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	...	Nth
ScanResDataWrite Evt	0x26	0x51	Result 0: ok others: fail								
Description: The ScanResDataWrite command is used to setup the whole scan response data field value and can only be used after a Driver_B execution. The ScanResDataWrite event is used to response the Scan Request in the BLE protocol.											
DisconnectWrite Cmd	0x25	0x5F	0x00								
DisconnectWrite Evt	0x26	0x5F	Result 0: ok others: fail								
Description: This command is used to disconnect the device when the device is connected. After executing this command, the device will return the corresponding event with an event packet of 26FF10, which means a disconnection status.											

Event Packet

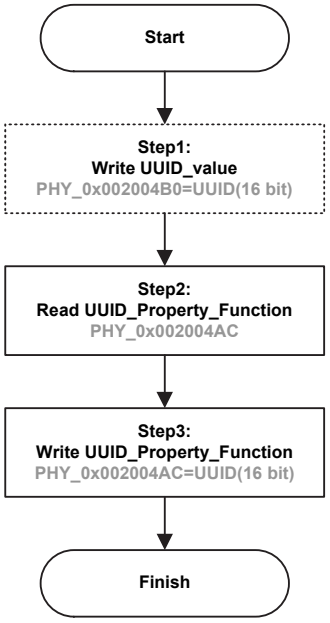
Packet Type	Opcode	Payload	Description
0x26	0x22	0x00	Tx finish – Tx buffer empty
		0x01	Connection not established – ignore this Payload Packet
		0x02	Indication or Notification not enabled – ignore this Payload Packet
		0x03	Unknown error – ignore all Payload Packets
		0x04	Send data error – ignore this Payload Packet
		0x05	Tx buffer full – ignore this Payload Packet
	0x30	0x00	Changing the Connection Interval is successful
		0x01	Master rejected the Connection Interval setup
		0x02	Command send fail
		0x12	Interval or length out of valid range
		0x13	
		0x14	
		0x15	No Connection
	0x40	0x12	Interval or length out of valid range
	0xFF	0x10	Disconnected
		0x11	Connected
		0x19	Packet Type is not supported
		0x1C	Packet Type is not supported

Accessible Physical Address

Address	Description	R/W	Note	
0x0000_5FFC	ROM Version Available version: All version	R	0x99000000	A5
0x0020_0134	Disable EEPROM Check **Write before Driver_A** Available version: All version	W	[0]	0: Reserved 1: Disable EEPROM check
0x0020_0480	Tx Buffer Valid Size Available version: All version	R	0x0000~0x00B4	Buffer size
			0xFFFF	Buffer error
			0xFFB4	Not in connection mode
0x0020_0484	Driver_B Version **Valid after Driver_B executed** Available version: All version	R		
0x0020_0488	RF parameter Address **Read/write before Driver_A executed** Available version: All version	R		
0x0020_048C	RF parameter Size **Read/write before Driver_A executed** Available version: All version	R	Unit: Byte	
0x0020_0490	Driver_A Version **Valid after Driver_A executed** Available version: All version	R		
0x0020_0494	Function **valid after Driver_B execute** Available version: BLE_180410A5 or later	R/W	[0]	0: Enable “2622 00” event 1: Disable “2622 00” event When the host MCU writes a payload into the BLE device to transmit air packets to a cell phone, the BLE device will send a “2622 00” event packet when the Tx operation has finished. If this bit is set high, the “262200” event packet will be disabled.
			[1]	0: INT_EXT low active (default) 1: INT_EXT high active
			[2]	0: STATE low active 1: STATE high active (default)
			[31:3]	Reserved
0x0020_04A4	CapConfig **Read/write before Driver_A executed** Available version: All version	R/W	[7:0]	CapConfig [1:0] should be kept as 11b.
			[14:8]	Reserved
			[15]	0: Disable update CapConfig 1: Enable update CapConfig
			[31:16]	CheckSum of CapConfig 0xFFFF-0x0020_04A4[15:0]
			0x0060_8038[29:22] map to 0x0020_04A4[7:0] Ex: If [7:0]=0x3F,[15:8]=0x80 [23:16] will be 0xC0 [31:24] will be 0x7F (0xFFFF-0x803F=0x7FC0)	

Address	Description	R/W	Note	
0x0020_04AC	UUID_Property_Function **Valid after Driver_B executed** Available version: BLE_170827A5 or later	R/W	[0]	Reserved
			[1]	0: Disable 0xFFF0 UUID modification function 1: Enable 0xFFF0 UUID modification function
			[2]	0: Disable 0xFFF2 UUID modification function 1: Enable 0xFFF2 UUID modification function
			[3]	0: Disable 0xFFF2 Property modification function 1: Enable 0xFFF2 Property modification function
			[4]	0: Disable 0xFFF1 UUID modification function 1: Enable 0xFFF1 UUID modification function
			[5]	0: Disable 0xFFF1 Property modification function 1: Enable 0xFFF1 Property modification function
			[8:6]	Reserved
			[9]	0xFFF2 Property selection 0: WriteNoRes 1: Write This bit is only available when bit 3 is set high.
			[10]	Reserved
			[11]	0xFFF1 Property selection 0: Notify 1: Indicate This bit is only available when bit 5 is set high.
			[31:12]	Should be kept as the original value.
			Refer to the UUID Modification Flow for UUID access operations.	
0x0020_04B0	UUID_Value **Valid after Driver_B executed** Available version: BLE_170827A5 or later	R/W	[15:0]	Target UUID 0x0020_04AC[1]=1: Set 0xFFF0 0x0020_04AC[2]=1: Set 0xFFF2 0x0020_04AC[4]=1: Set 0xFFF1
			[31:16]	Reserved
0x0020_04C4	EEPROM_Function **Valid after Driver_A executed** Available version: BLE_180511A5 or later	R/W	[0]	IIC_WRITE_EN 0: Command not sync to EEPROM(0x0040) (default) 1: Command sync to EEPROM(0x0040)
			[1]	IIC_NOT_CHECK_BLE_PARAM 0: 0x0040 checksum fail read 0x1F40 and MP_BD_ADDR (default) (Valid when MP_BD_ADDR_UPDATE_FLAG=0x5AF0/0x0055) 1: Not check
			[2]	IIC_UPDATE_BLE_PARAM_BACKUP_EN: 0: Command not sync to EEPROM_Backup (0x1F40) (default) 1: Command sync to EEPROM_Backup (0x1F40) (auto clear)
			[31:3]	Reserved
			Refer to the EEPROM Setup Flow for EEPROM access operations.	

Address	Description	R/W	Note	
0x0020_0500	Driver Start Address Available version: All version	W		
0x0020_1F80	Driver Execute Trigger Available version: All version	W	Write 0x00200500 to Execute Driver_A Write 0x00000000 to Execute Driver_B	
0x004003E4	PeerAddrType **Valid when connected** Available version: BLE_180410A5 or later	R	[31:24]	Reserved
			[23:16]	0: Public 1: Random
			[15:0]	Reserved
0x004003E8	PeerAddr[31:0] **Valid when connected** Available version: BLE_180410A5 or later	R	[31:0]	PeerAddr[31:0] (LSB)
0x004003EC	PeerAddr[47:32] **Valid when connected** Available version: BLE_180410A5 or later	R	[31:16]	Reserved
			[15:0]	PeerAddr[47:32](LSB)
0x0060_2020	Driver State(map to CR5) Available version: All version	R	[6]	0: Device Driver_A successful 1: Device wait for Driver_A
			[10]	0: Device wait for Driver_B 1: Device Driver_B successful
			[15]	0: Device ROM boot fail 1: Device ROM boot ok



UUID Modification Flow

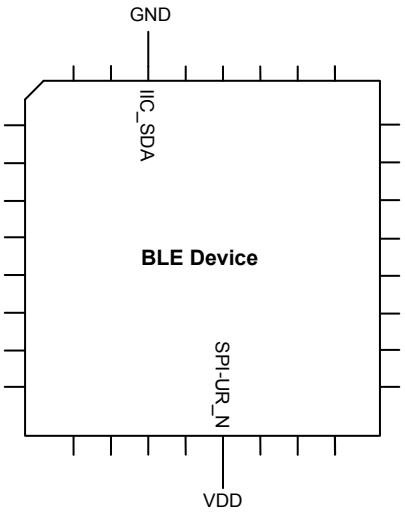
EEPROM Setup Flow

SPI Interface

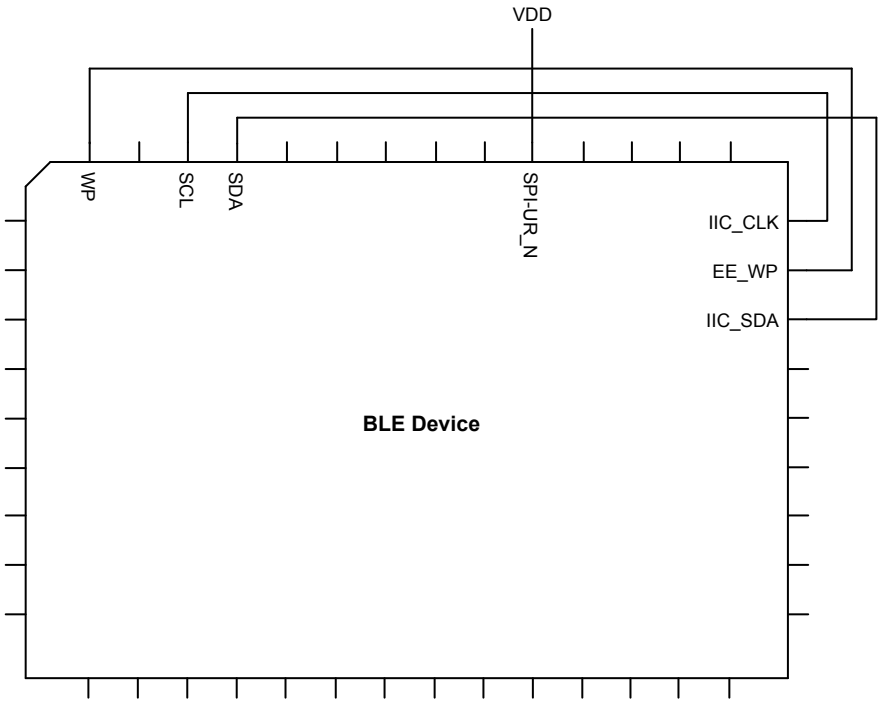
Hardware setup

The hardware environment should be established before using the SPI interface.

- Keep the SPI-UR_N pin=1 before the RST_N pin=1
- If driven by the MCU, keep the IIC_SDA pin=0
- If driven by an EEPROM, connect the IIC_CLK/IIC_SDA/EE_WP pins to the EEPROM corresponding pins.



Driven by MCU using SPI Interface



Driven by EEPROM using SPI Interface

SPI interface default setting

5-Wire : SPI_MISO, SPI_MOSI, SPI_CS, SPI_CLK, SPI_INT/INT_EXT
Data length : 8 bits (MSB first)
Format : CPOL=0, CPHA=0
Operating mode : Duplex
Max. Frequency : 10MHz

SPI Command Format

Type	Header			SPI Interface Action
Read Register	Bit[7:5]	Bit[4:1]	Bit[0]	SPI_MOSI: Read Register 8bit SPI_MISO: Register [15:8] 8bit, Register [7:0] 8bit
	000b	0~8 (CR0~CR8)	1	
Write Register	Bit[7:5]	Bit[4:1]	Bit[0]	SPI_MOSI: Read Register 8bit, Register [15:8] 8bit, Register [7:0] 8bit SPI_MISO
	001b	0~8 (CR0~CR8)	1	
Read FIFO	Bit[7:5]	Bit[4:0]		SPI_MOSI: Read FIFO 8bit SPI_MISO: Data 0 8bit, ..., Data (Length-1) 8bit
	011b	Data Length		
		0 means 32 byte		
Write FIFO	Bit[7:5]	Bit[4:0]		SPI_MOSI: Write FIFO 8bit, Data 0 8bit, ..., Data (Length-1) 8bit SPI_MISO
	101b	Data Length		
		0 means 32 byte		

Example:
- "090300" means Read Register, Register Address=0x04, Register Value=0x0300
- "A3253400" means Write FIFO, Data Length=0x03, Data=0x253400

Control Register Table (CR0~CR8)

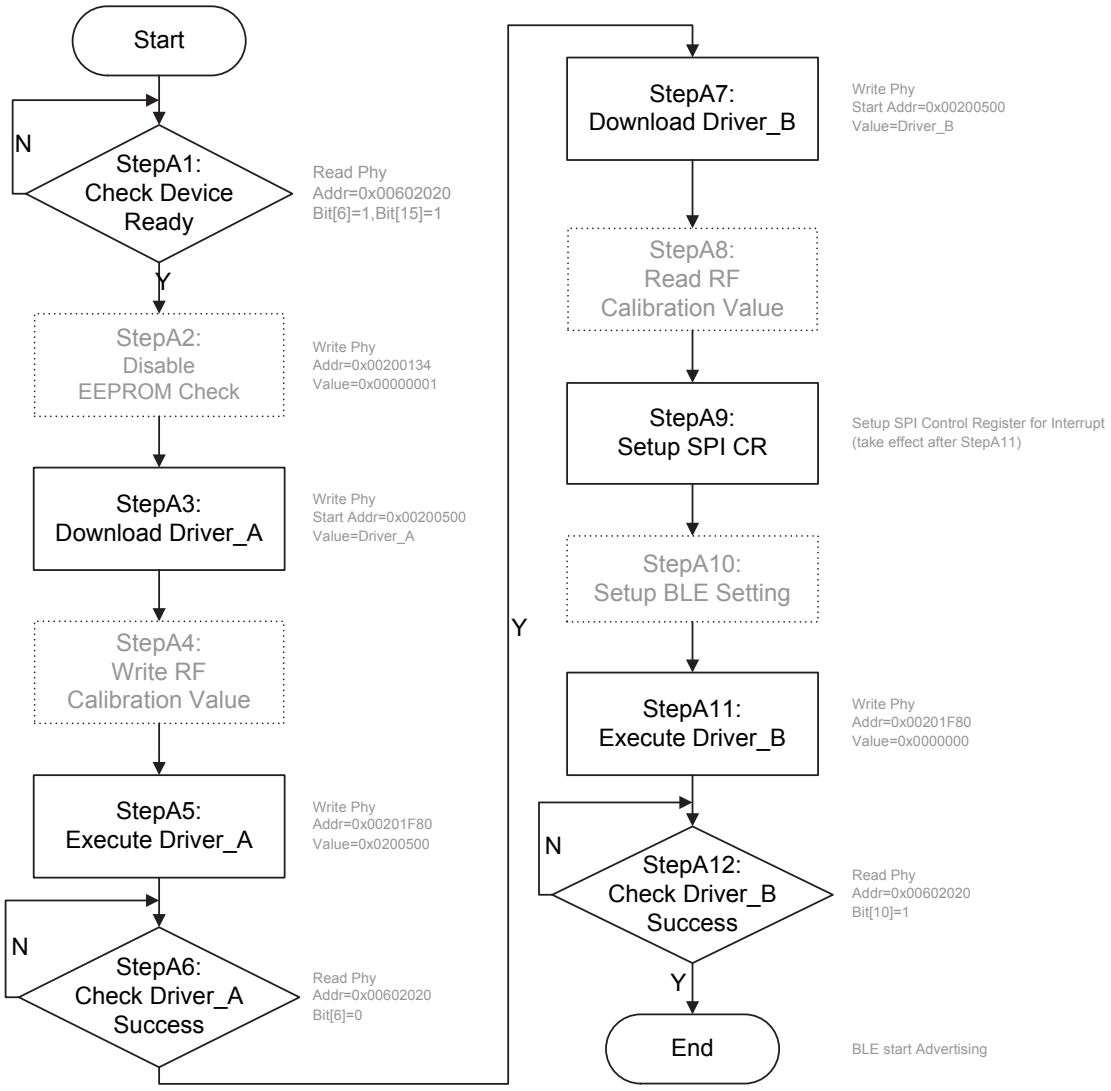
Name	Addr	Description	
CR0: Threshold	0x00	SPI FIFO threshold If the data length is greater than this value, the corresponding interrupt will be generated.	
		Bit[15:12]	Reserved
		Bit[11:6]	SPI Tx FIFO threshold (BLE → MCU)
		Bit[5:0]	SPI Rx FIFO threshold (BLE ← MCU)
CR1: Int_Status	0x01	SPI Interrupt status Used to indicate which interrupt is triggered. 0: Corresponding interrupt condition did not occur 1: Corresponding interrupt condition occurs	
		Bit[15:5]	Reserved
		Bit[4]	BLE Tx FIFO not empty
		Bit[3]	BLE Tx FIFO overflow
		Bit[2]	BLE Tx FIFO over threshold
		Bit[1]	BLE Rx FIFO empty
		Bit[0]	BLE Rx FIFO under threshold

Name	Addr	Description	
CR2: Int_Set	0x02	SPI Interrupt enable control Used to enable the corresponding interrupt function. 0: Corresponding interrupt not enabled 1: Corresponding interrupt enabled	
		Bit[15:5]	Reserved
		Bit[4]	BLE Tx FIFO not empty
		Bit[3]	BLE Tx FIFO overflow
		Bit[2]	BLE Tx FIFO over threshold
		Bit[1]	BLE Rx FIFO empty
		Bit[0]	BLE Rx FIFO under threshold
CR3: Int_Clr	0x03	SPI Interrupt status clear control – write only. Used to clear the corresponding interrupt status. 0: No action 1: Clear the corresponding interrupt status	
		Bit[15:5]	Reserved
		Bit[4]	BLE Tx FIFO not empty
		Bit[3]	BLE Tx FIFO overflow
		Bit[2]	BLE Tx FIFO over threshold
		Bit[1]	BLE Rx FIFO empty
		Bit[0]	BLE Rx FIFO under threshold
CR4: FIFO_Count	0x04	SPI FIFO count Used to set the Tx/Rx FIFO depth.	
		Bit[15:12]	Reserved
		Bit[11:6]	SPI Tx FIFO count (BLE → MCU)
		Bit[5:0]	SPI Rx FIFO count (BLE ← MCU)

Name	Addr	Description	
CR5: BLE_Status	0x05	BLE device status, read only. The host MCU can read this register to know the BLE device status.	
		Bit[15]	Status_BLE_ROM_Init 0: Not ready 1: Ready
		Bit[14]	Status_BLE_Sleep 0: Active 1: Sleep
		Bit[13:12]	Reserved
		Bit[11]	Status INT_EXT_STATUS 0: No INT_EXT occurs 1: INT_EXT occurs
		Bit[10]	Status_BLE_Can_Sleep 0: BLE device is kept active 1: BLE device sleep/active switched automatically
		Bit[9]	Reserved
		Bit[8]	Status_Soft_Reset 0: Soft_Reset event does not occur or Soft_Reset event has finished 1: Soft_Reset event occurs This bit will be 1 when the Soft_Reset event occurs after setting the CR7 register bit[8] to 1. After this the bit value will be 0 and the register value will be 0x8040 after the Soft_Reset event has finished.
		Bit[7]	Status_Buffer_Reset 0: Buffer_Reset event does not occur or Buffer_Reset event has finished 1: Buffer_Reset occurs This bit will be 1 when the Buffer_Reset event occurs after setting the CR7 register bit[7] to 1. After this the bit value will be 0 after the Buffer_Reset event has finished.
		Bit[6]	Status_Wait_For_Driver 0: Not necessary to download the driver 1: Wait for driver download
		Bit[5:3]	Reserved
		Bit[2]	Status_BLE_UUID_0xFFF1 0: Disable Notify 1: Enable Notify
		Bit[1]	Status_BLE_Connection 0: Not connected 1: Connected
		Bit[0]	Status_BLE_Cmd_Analysing 0: No analysis 1: Analysing If this bit is kept as “1” for a long period a Ctrl_Buffer_Reset execution is necessary.

Name	Addr	Description	
CR7: BLE_Status_Set	0x07	BLE device status set control, write only. The MCU can set the corresponding bit high to trigger various BLE device actions.	
		Bit[15:11]	Reserved
		Bit[10]	Set_BLE_Can_Sleep Setting this bit high will enable the BLE device active/sleep auto-switch function.
		Bit[9]	Reserved
		Bit[8]	Set_Soft_Reset Setting this bit high will trigger a BLE device Soft_Reset operation. The corresponding status bit, CR5[8], will be 1. After the Soft_Reset operation has completed, the CR5[8] value will be zero.
		Bit[7]	Set_Buffer_Reset Setting this bit high will trigger a BLE device Buffer_Reset operation. The corresponding status bit, CR5[7], will be 1. After the Buffer_Reset operation has completed, the CR5[7] value will be zero.
		Bit[6:0]	Reserved
CR8: BLE_Status_Clr	0x08	BLE device status clear control, write only. The MCU can set the corresponding bit high to clear the BLE device action.	
		Bit[15:11]	Reserved
		Bit[10]	Clr_BLE_Can_Sleep Setting this bit high will disable the BLE device active/sleep auto-switch function.
		Bit[9]	Reserved
		Bit[8]	Clr_Soft_Reset Setting this bit high will cancel a BLE device Soft_Reset action. After this the corresponding status bit, CR5[8], will be zero after a cancellation.
		Bit[7]	Clr_Buffer_Reset Setting this bit high will cancel a BLE device Buffer_Reset action. After this the corresponding status bit, CR5[7] will be zero.
		Bit[6:0]	Reserved

Initialization Flow for the Mode_A – Driven by the MCU using the SPI Interface



Mode A – Driven by the MCU using the SPI Interface Flow

- **StepA1: Check Device Ready**
  - Read the Physical Address 0x0060_2020 to check whether the device Power-On is ready or not. If Bit[15]=1, it means that the device Power-On is ready and is waiting for the Driver.
  - Reading the Physical Address 0x0060_2020 should be bit[15]=1, bit[10]=0, bit[6]=1.
- **StepA2: Disable the EEPROM Check – optional**
  - If the IIC_SDA pin is kept high or floating, the EEPROM Check function must be disabled. Otherwise, the device will be in an endless check loop.
  - Write the Physical Address 0x0020_0134 with a value of 0x0000_0001.
- **StepA3: Download Driver_A**
  - The MCU sends the Driver_A code to the device and writes the Driver_A code to the Physical Address 0x0020_0500.
- **StepA4: Write RF Calibration Value – optional**
  - This step should be ignored for the first boot. Otherwise, update the RF Calibration Value in this step if it has already been obtained from StepA8.
  - Obtain the address where the RF Calibration Value is stored from the Physical Address 0x0020_0488.
  - Write the RF Calibration Value to the address where the RF Calibration Value is stored.
  - There should be at least 128 bytes of memory capacity to store the RF calibration value if this step is selected to be executed.
- **StepA5: Execute Driver_A**
  - Write the Physical Address 0x0020_1F80 with a value of 0x0020_0500 after which the Driver_A code will start to boot for the variable setup, RF calibration, etc.
- **StepA6: Check Driver_A Success**
  - Read the Physical Address 0x0060_2020 to check whether the Driver_A execution is ready or not. If the Bit[6]=0, it means that the device Driver_A boot is successful.
  - Reading the Physical Address 0x0060_2020 should be bit[15]=1,bit[10]=0,bit[6]=0.
- **StepA7: Download Driver_B**
  - The MCU sends the Driver_B code to the device and writes the Driver_B code to the Physical Address 0x0020_0500.
- **StepA8: Read RF Calibration Value – optional**
  - This operation should be executed for the first boot. Otherwise, this step can be ignored.
  - Obtain the RF Calibration Value Address from the Physical Address 0x0020_0488.
  - Obtain the RF Calibration Value Size from the Physical Address 0x0020_048C (unit: byte).
  - Read the RF Calibration Value from the RF Calibration Value Address and save it for next initialization flow.
  - There should be at least 128 bytes of memory capacity to store the RF calibration value if this step is selected to be executed.
- **StepA9: Setup SPI CR**
  - The MCU configures the SPI Control Register for the SPI_INT Interrupt that is used. If the EXT_INT is only used this step can be ignored.
  - The interrupt configurations will be available after StepA12.
- **StepA10: Setup BLE Setting – optional**
  - The MCU sends the Read/Write Command to setup the BLE device, such as BD_ADDR, BD_Name, etc.
- **StepA11: Execute Driver_B**
  - The MCU writes the Physical Address 0x0020_1F80 with a value of 0x0000_0000 after which the Driver_B code will start to boot.
- **StepA12: Check Driver_B Success**
  - Read the Physical Address 0x0060_2020 to check whether the Driver_B boot is ready or not. If Bit[10]=1, than this means that the device Driver_B boot is successful.
  - Reading the Physical Address 0x00602020 should be bit[15]=1, bit[10]=1, bit[6]=0.

Example for Mode_A – Driven by MCU using SPI Interface

Text Font	Normal	**Bold**	*Italic*
Subject	MCU_Rx	**MCU_Tx**	*Driver*
• **StepA1: Check Device Ready** ♦ Read the Physical Address 0x0060_2020 to check whether the device Power-On is ready or not. If Bit[15]=1, it means that the device Power-On is ready and is waiting for the Driver. ♦ Reading the Physical Address 0x0060_2020 should be bit[15]=1, bit[10]=0, bit[6]=1.			
A6560120206000 (Write FIFO)			
090000 (Read Register: SPI Tx FIFO count(CR4[11:6])=0)			
...			
090300 (Read Register: SPI Tx FIFO count(CR4[11:6])=0x0C)			
6C5701000020206000040800000 (Read FIFO)			
• **StepA2: Disable EEPROM Check – optional** ♦ If the IIC_SDA pin is kept high or floating, the EEPROM Check function must be disabled. Otherwise the device will be in an endless check loop. Write the Physical Address 0x0020_0134 with a value of 0x0000_0001.			
AC55010000340120000100000 (Write FIFO)			
• **StepA3: Download Driver_A** ♦ The MCU sends the Driver_A code to the device and write the Driver_A code to the Physical Address 0x0020_0500.			
A85538000000052000 (Write FIFO)			
A0*D01400E035029014D0147A1060935093403A030C7093CD7B9014D01475106093* (Write FIFO)			
A0*5393403A030C7393CD7B9014D014F0E313369014D014EFE37FFE9014D0146D10* (Write FIFO)			
A0*60935393403A030C7393CD7B9014D014681060935393403A030C7393CD7B9014* (Write FIFO)			
A0*D014F0E3BD079014D014F0E3B907901478074006E1304B327B348B3023246B3* (Write FIFO)			
A0*013246B33C78D114C1348F44789463E4FF3063EC090078B4661369B4679463EC* (Write FIFO)			
A0*A20067B46D94803B6DB46894A13B68B46894803B68B4363361B46D94A53B0130* (Write FIFO)			
A0*6DB4436C6432FFE3D7FF00C02050FA300540F0E3C9277612003246B344B36194* (Write FIFO)			
A855380000E0052000 (Write FIFO)			
A0*853B61B49114C11445740032C36C03290B6C0604885B80840022105889744964* (Write FIFO)			
A0*FA0B81146C126093403B040801326B124EA36A126E83312349126C5B6C5A418B* (Write FIFO)			
A0*E0336F4343DC0822C1336F434493A83A44B32031401222B28031214123B25993* (Write FIFO)			
A0*42EC1C0059B35993853A59B302EACBFF52B33C113CB35393843A53B35393A53A* (Write FIFO)			
A0*53B35393A23A53B35393A43A53B37611003240B3351124B33C78C214C0323411* (Write FIFO)			
A0*434200EA3206E0358242AF45945C80940022E03380A90A646F430121F50B3093* (Write FIFO)			
A0*471122DC3310319322DC3410329322DC3510339322DC3610349322DC371023D8* (Write FIFO)			
...			
• **StepA4: Write RF Calibration Value – optional** ♦ This step should be ignored for the first boot. Otherwise update the RF Calibration Value in this step if it has already been obtained from StepA8. ♦ Obtain the address where the RF Calibration Value is stored from the Physical Address 0x0020_0488. ♦ Write the RF Calibration Value to the address where the RF Calibration Value is stored. ♦ There should be at least 128 bytes of memory capacity to store the RF calibration value if this step is selected to be executed.			
A055060000A01E2000AA55757375757775777777979797B7B7B7B7D7D81818181 (Write FIFO)			
A055060000B81E2000838383838585858587878787898989898B8B8B8B918D9191 (Write FIFO)			
A055060000D01E2000919193930D0B1111131315131715191719191B1B21212321 (Write FIFO)			
A055060000E81E2000232325252727292729272B292B2B31313331333333535373737937 (Write FIFO)			
A055060000001F200039393B3B3D3D441643164218420C410C2401003812240000 (Write FIFO)			

Text Font	Normal	**Bold**	*Italic*
Subject	MCU_Rx	**MCU_Tx**	*Driver*
• **StepA5: Execute Driver_A** ♦ Write the Physical Address 0x0020_1F80 with a value of 0x0020_0500 after which the Driver_A code will start to boot for the variable setup, RF calibration, etc.			
AC55010000801F200000052000 (Write FIFO)			
• **StepA6: Check Driver_A Success** ♦ Read the Physical Address 0x0060_2020 to check whether the Driver_A execution is ready or not. If the Bit[6]=0, it means that the device Driver_A boot is successful. ♦ Reading the Physical Address 0x0060_2020 should be bit[15]=1,bit[10]=0,bit[6]=0.			
A6560120206000 (Write FIFO)			
090000 (Read Register: SPI Tx FIFO count(CR4[11:6])=0)			
...			
090300 (Read Register: SPI Tx FIFO count(CR4[11:6])=0x0C)			
6C570100002020600000800000 (Read FIFO)			
• **StepA7: Download Driver_B** ♦ The MCU sends the Driver_B code to the device and writes the Driver_B code to the Physical Address 0x0020_0500.			
A85538000000052000 (Write FIFO)			
A0*D01400E039009014D0147A1060935093403A030C7093CD7B9014D01475106093* (Write FIFO)			
A0*5393403A030C7393CD7B9014D01400E0D7009014D014EFE37FFE9014D0146D10* (Write FIFO)			
A0*60935393403A030C7393CD7B9014D014681060935393403A030C7393CD7B9014* (Write FIFO)			
A0*D014F0E3BD079014D014F0E3B907901478074000631060934310 41B33C780000* (Write FIFO)			
A0*780740002C0520006B12003121B3409302C48028030C013122B322C48028040C* (Write FIFO)			
A0*6512023242B36412013121B33C78D014F0E3B42961124093933A40B3C1336F43* (Write FIFO)			
***A0*4493943A44B39014D4149D116084403B0508FFE3EEFF00A43E047A114083403A** (Write FIFO)			
A855380000E0052000 (Write FIFO)			
A0*0308193240A378118083403C1608FFE3E0FF03EAFF01C2642E0C71116083015B* (Write FIFO)			
A0*0074721100A30274E0C78028020C015C6C1160830D641F0CAD110036E5D86320* (Write FIFO)			
A0*C5DC632080C12070871100842258F0E3A90880C12074E5DC63206084153B0408* (Write FIFO)			
A0*002360A408047E10401120834082 4959C0A440A394145810639203C48028030C* (Write FIFO)			
A0*013125B223C48028040C7310023245B33C78C1336F43449342EC001844B34593* (Write FIFO)			
A0*42EC001845B380336443521060B272106093321033B30F3267104DB301EAFEFF* (Write FIFO)			
A0*2EB300324CB3013241B302312BB34BB33C78000000706000049060008200 2000* (Write FIFO)			
...			
• **StepA8: Read RF Calibration Value – optional** ♦ This operation should be executed for the first boot. Otherwise, this step can be ignored. ♦ Obtain the RF Calibration Value Address from the Physical Address 0x0020_0488. ♦ Obtain the RF Calibration Value Size from the Physical Address 0x0020_048C (unit: byte). ♦ Read the RF Calibration Value from the RF Calibration Value Address and save it for next initialization flow. ♦ There should be at least 128 bytes of memory capacity to store the RF calibration value if this step is selected to be executed.			
A656018C042000 (Write FIFO)			
090000 (Read Register: SPI Tx FIFO count(CR4[11:6])=0)			
...			
090300 (Read Register: SPI Tx FIFO count(CR4[11:6])=0x0C)			
6C570100008C04200078000000 (Read FIFO: RF Calibration Value's Size=0x78)			
A6560188042000 (Write FIFO)			
090000 (Read Register: SPI Tx FIFO count(CR4[11:6])=0)			
...			
090300 (Read Register: SPI Tx FIFO count(CR4[11:6])=0x0C)			

Text Font	Normal	**Bold**	*Italic*
Subject	MCU_Rx	**MCU_Tx**	*Driver*
6C5701000088042000A0E12000 (Read FIFO: RF Calibration Value's Address=0x00201E0A)			
A65606A01E2000 (Write FIFO)			
090000 (Read Register: SPI Tx FIFO count(CR4[11:6])=0)			
...			
090800 (Read Register: SPI Tx FIFO count(CR4[11:6])=0x20)			
6056060000A01E2000AA55757375757775777797979797B7B7B7B7D7D81818181 (Read FIFO: RF Calibration Value)			
A65606B81E2000 (Write FIFO)			
090000 (Read Register: SPI Tx FIFO count(CR4[11:6])=0)			
...			
090800 (Read Register: SPI Tx FIFO count(CR4[11:6])=0x20)			
6056060000B81E2000838383838585858587878787898989898B8B8B8B918D9191 (Read FIFO: RF Calibration Value)			
A65606D01E2000 (Write FIFO)			
090000 (Read Register: SPI Tx FIFO count(CR4[11:6])=0)			
...			
090800 (Read Register: SPI Tx FIFO count(CR4[11:6])=0x20)			
6056060000D01E2000919193930D0B1111131315131715191719191B1B21212321 (Read FIFO: RF Calibration Value)			
A65606E81E2000 (Write FIFO)			
090000 (Read Register: SPI Tx FIFO count(CR4[11:6])=0)			
...			
090800 (Read Register: SPI Tx FIFO count(CR4[11:6])=0x20)			
6056060000E81E2000232325252727292729272B292B2B31313331333333535373737937 (Read FIFO: RF Calibration Value)			
A65606001F2000 (Write FIFO)			
090000 (Read Register: SPI Tx FIFO count(CR4[11:6])=0)			
...			
090800 (Read Register: SPI Tx FIFO count(CR4[11:6])=0x20)			
6056060000001F200039393B3B3D3D441643164218420C410C2401003812240000 (Read FIFO: RF Calibration Value)			
• **StepA9: Setup SPI CR** ♦ The MCU configures the SPI Control Register for the SPI_INT Interrupt that is used. If the EXT_INT is only used this step can be ignored. ♦ The interrupt configurations will be available after StepA12.			
010000 (Read Register: CR0_Threshold=0x0000)			
030017 (Read Register: CR1_Int_Status=0x0017)			
070000 (Read Register: CR3_Int_Clr=0x0000)			
050000 (Read Register: CR2_Int_Set=0x0000)			
090000 (Read Register: CR4_FIFO_Count=0x0000)			
0B8400 (Read Register: CR5_BLE_Status=8400)			
0F0000 (Read Register: CR7_BLE_Status_Set=0000)			
110000 (Read Register: CR8_BLE_Status_Clr=0000)			
27000F (Write Register: CR3_Int_Clr=000F)			
250010 (Write Register: CR2_Int_Set=0010)			
010000 (Read Register: CR0_Threshold=0000, read back for check)			
030017 (Read Register: CR1_Int_Status=0017, read back for check)			
070000 (Read Register: CR3_Int_Clr=0000, read back for check)			
050010 (Read Register: CR2_Int_Set=0010, read back for check)			

Text Font	Normal	**Bold**	*Italic*
Subject	MCU_Rx	**MCU_Tx**	*Driver*
090000 (Read Register: CR4_FIFO_Count=0000, read back for check)			
0B8400 (Read Register: CR5_BLE_Status=8400, read back for check)			
0F0000 (Read Register: CR7_BLE_Status_Set=0000, read back for check)			
• **StepA10: Setup BLE Setting – optional** ♦ The MCU sends the Read/Write Command to setup the BLE device, such as BD_ADDR, BD_Name, etc.			
A6253103313131 (Write FIFO: BDNameWrite Cmd)			
090000 (Read Register: SPI Tx FIFO count(CR4[11:6])=0)			
...			
0900C0 (Read Register: SPI Tx FIFO count(CR4[11:6])=0x03)			
63263100 (Read FIFO: BDNameWrite Evt)			
A9253306112233445566 (Write FIFO: BDAddrWrite Command)			
090000 (Read Register: SPI Tx FIFO count(CR4[11:6])=0)			
...			
0900C0 (Read Register: SPI Tx FIFO count(CR4[11:6])=0x03)			
63263300 (Read FIFO: BDAddrWrite Evt)			
A725350464000000 (Write FIFO: AdvIntvWrite Command)			
090000 (Read Register: SPI Tx FIFO count(CR4[11:6])=0)			
...			
0900C0 (Read Register: SPI Tx FIFO count(CR4[11:6])=0x03)			
63263500 (Read FIFO: AdvIntvWrite Evt)			
A425380100 (Write FIFO: TxPowerWrite Command)			
090000 (Read Register: SPI Tx FIFO count(CR4[11:6])=0)			
...			
0900C0 (Read Register: SPI Tx FIFO count(CR4[11:6])=0x03)			
63263800 (Read FIFO: TxPowerWrite Evt)			
• **StepA11: Execute Driver_B** ♦ The MCU writes the Physical Address 0x0020_1F80 with a value of 0x0000_0000 after which the Driver_B code will start to boot.			
AC55010000801F200000000000 (Write FIFO)			
• **StepA12: Check Driver_B Success** ♦ Read the Physical Address 0x0060_2020 to check whether the Driver_B boot is ready or not. If Bit[10]=1, then this means that the device Driver_B boot is successful. ♦ Reading the Physical Address 0x00602020 should be bit[15]=1, bit[10]=1, bit[6]=0.			
A6560120206000 (Write FIFO)			
090000 (Read Register: SPI Tx FIFO count(CR4[11:6])=0)			
...			
090300 (Read Register: SPI Tx FIFO count(CR4[11:6])=0x0C)			
6C57010000202060004080000 (Read FIFO)			
• **StepN: Connection Established**			
090000 (Read Register: SPI Tx FIFO count(CR4[11:6])=0)			
...			
0900C0 (Read Register: SPI Tx FIFO count(CR4[11:6])=0x03)			
6322FF11 (Read FIFO)			
• **StepN+1: Receive Payload Packet from cell phone**			
090000 (Read Register: SPI Tx FIFO count(CR4[11:6])=0)			
...			
090140 (Read Register: SPI Tx FIFO count(CR4[11:6])=0x05)			
652203123456 (Read FIFO: payload=123456)			

Text Font	Normal	**Bold**	*Italic*
Subject	MCU_Rx	**MCU_Tx**	*Driver*
...			

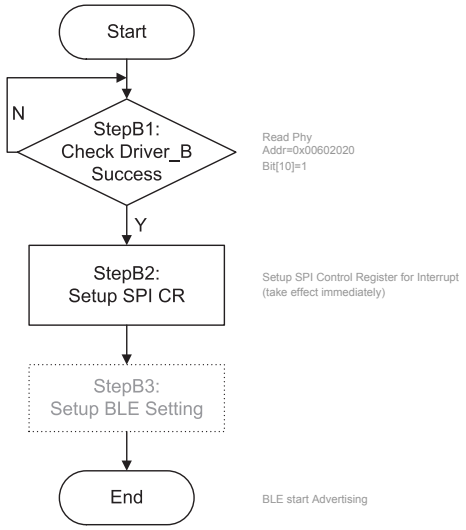
Driver Time for the Mode_A – Driven by the MCU using the SPI Interface

The Driver time may be different due to various versions. The following Driver time is measured for the Driver version 161025A5, in which the Driver_A size=0x02C0 bytes and the Driver_B size=0x1480 bytes.

SPI=1Mhz, total time=500ms (typical), A1+A2+A3+A5+A6+A7+A9+A11+A12

SPI=1Mhz, total time=140ms (typical), A1+A2+A3+A4+A5+A6+ A7+A9+A11+A12

Initialization Flow for the Mode_B – Driven by the EEPROM using the SPI Interface



Procedures for the Mode B – Driven by the EEPROM using the SPI Interface

- **StepB1: Check Driver_B Success**
  - ♦ Read the Physical Address 0x0060_2020 to check whether the Driver_B execution is ready or not. If Bit[10]=1, then this means that device Driver_B boot is successful.
  - ♦ Reading the Physical Address 0x0060_2020 should be bit[15]=1, bit[10]=1, bit[6]=0.
- **StepB2: Setup SPI CR**
  - ♦ The MCU configures the SPI Control Register for the SPI_INT Interrupt which is used. If the EXT_INT is only used this step can be ignored.
  - ♦ The interrupt configurations are available immediately.
- **StepB3: Setup BLE Setting (Optional)**
  - ♦ The MCU sends the Read/Write Command to setup the BLE device, such as BD_ADDR, BD_ Name, etc.

Example for the Mode_B – Driven by the EEPROM using the SPI Interface

Text Font	Normal	**Bold**	*Italic*
Subject	MCU_Rx	**MCU_Tx**	*Driver*
• **StepB1: Check Driver_B Success** ♦ Read the Physical Address 0x0060_2020 to check whether the Driver_B execution is ready or not. If Bit[10]=1, then this means that device Driver_B boot is successful. ♦ Reading the Physical Address 0x0060_2020 should be bit[15]=1, bit[10]=1, bit[6]=0.			
A6560120206000 (Write FIFO)			
090000 (Read Register: SPI Tx FIFO count(CR4[11:6])=0)			
...			
090300 (Read Register: SPI Tx FIFO count(CR4[11:6])=0x0C)			
6C570100002020600040800000 (Read FIFO)			
• **StepB2: Setup SPI CR** ♦ The MCU configures the SPI Control Register for the SPI_INT Interrupt which is used. If the EXT_INT is only used, this step can be ignored. ♦ The interrupt configurations are available immediately.			
010000 (Read Register: CR0_Threshold=0x0000)			
030017 (Read Register: CR1_Int_Status=0x0017)			
070000 (Read Register: CR3_Int_Clr=0x0000)			
050000 (Read Register: CR2_Int_Set=0x0000)			
090000 (Read Register: CR4_FIFO_Count=0x0000)			
0B8400 (Read Register: CR5_BLE_Status=8400)			
0F0000 (Read Register: CR7_BLE_Status_Set=0000)			
110000 (Read Register: CR8_BLE_Status_Clr=0000)			
27000F (Write Register: CR3_Int_Clr=000F)			
250010 (Write Register: CR2_Int_Set=0010)			
010000 (Read Register: CR0_Threshold=0000, read back for check)			
030017 (Read Register: CR1_Int_Status=0017, read back for check)			
070000 (Read Register: CR3_Int_Clr=0000, read back for check)			
050010 (Read Register: CR2_Int_Set=0010, read back for check)			
090000 (Read Register: CR4_FIFO_Count=0000, read back for check)			
0B8400 (Read Register: CR5_BLE_Status=8400, read back for check)			
0F0000 (Read Register: CR7_BLE_Status_Set=0000, read back for check)			
• **StepB3: Setup BLE Setting (Optional)** ♦ The MCU sends the Read/Write Command to setup the BLE device, such as BD_ADDR, BD_Name, etc.			
A6253103313131 (Write FIFO: BDNameWrite Cmd)			
090000 (Read Register: SPI Tx FIFO count(CR4[11:6])=0)			
...			
0900C0 (Read Register: SPI Tx FIFO count(CR4[11:6])=0x03)			
63263100 (Read FIFO: BDNameWrite Evt)			
A9253306112233445566 (Write FIFO: BDAddrWrite Command)			
090000 (Read Register: SPI Tx FIFO count(CR4[11:6])=0)			
...			
0900C0 (Read Register: SPI Tx FIFO count(CR4[11:6])=0x03)			
63263300 (Read FIFO: BDAddrWrite Evt)			
A725350464000000 (Write FIFO: AdvIntvWrite Command)			
090000 (Read Register: SPI Tx FIFO count(CR4[11:6])=0)			
...			
0900C0 (Read Register: SPI Tx FIFO count(CR4[11:6])=0x03)			
63263500 (Read FIFO: AdvIntvWrite Evt)			

Text Font	Normal	**Bold**	*Italic*
Subject	MCU_Rx	**MCU_Tx**	*Driver*
A425380100 (Write FIFO: TxPowerWrite Command)			
090000 (Read Register: SPI Tx FIFO count(CR4[11:6])=0)			
...			
0900C0 (Read Register: SPI Tx FIFO count(CR4[11:6])=0x03)			
63263800 (Read FIFO: TxPowerWrite Evt)			
• StepN: Connection Established			
090000 (Read Register: SPI Tx FIFO count(CR4[11:6])=0)			
...			
0900C0 (Read Register: SPI Tx FIFO count(CR4[11:6])=0x03)			
6322FF11 (Read FIFO)			
• StepN+1: Receive Payload Packet from cell phone			
090000 (Read Register SPI Tx FIFO count(CR4[11:6])=0)			
...			
090140 (Read Register SPI Tx FIFO count(CR4[11:6])=0x05)			
652203123456 (Read FIFO: payload=123456)			
...			

Driver Time for the Mode_B – Driven by the EEPROM using the SPI Interface

The driver time may be different due to various versions. The following driver time is measured for the driver version 161025A5, in which the Driver_A size=0x02C0 bytes and the Driver_B size=0x1480 bytes.

First boot with the RF calibration: SPI=1MHz, total time=850ms (typical), B1+B2

After boot the device will write the RF calibration value into the EEPROM.

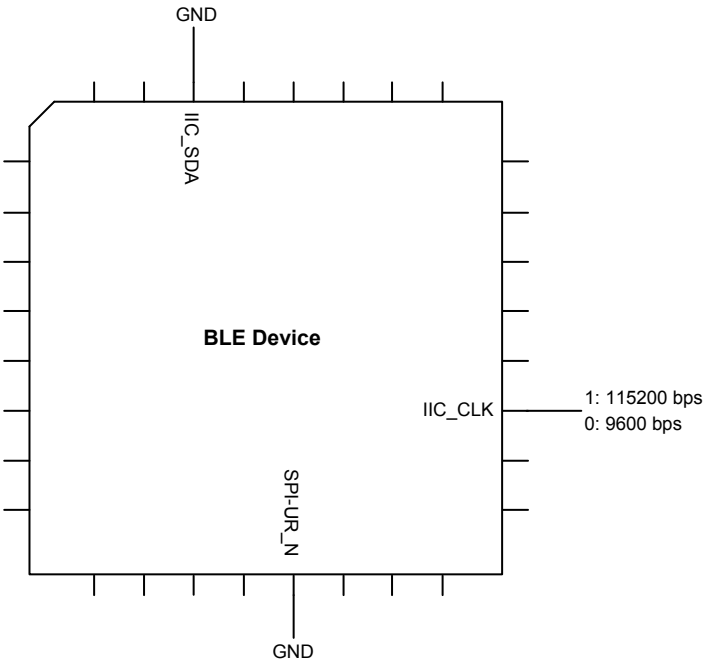
Second boot without RF calibration: SPI=1MHz, total time=350ms (typical), B1+B2

UART Interface

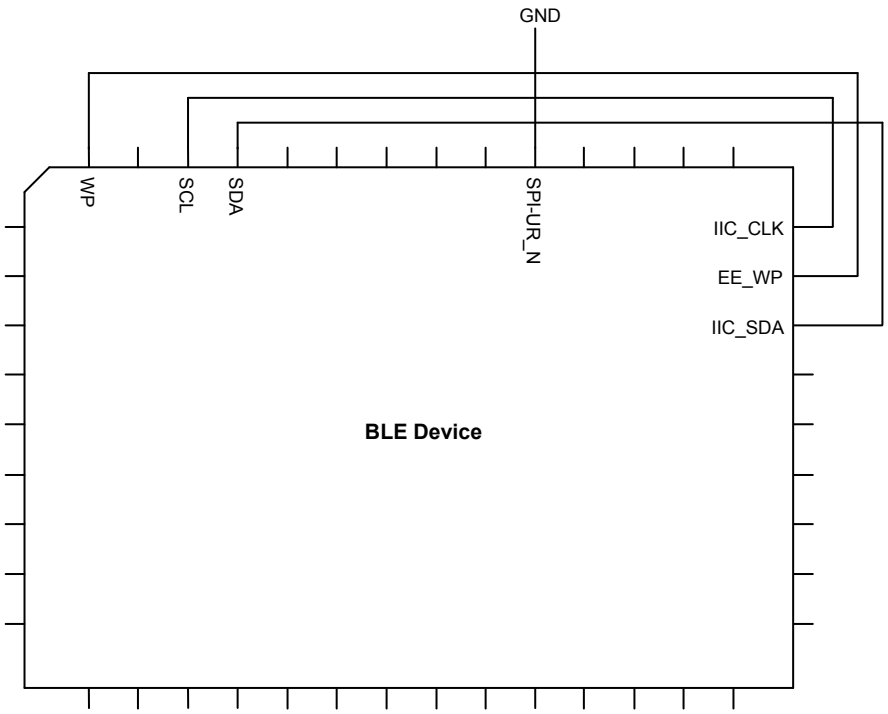
Hardware Setup

The hardware environment should be established before using the UART interface.

- Keep the SPI-UR_N pin=0 before the RST_N pin=1
- If driven by the MCU, keep the IIC_SDA pin=0. The IIC_CLK pin can then be used to select the default baudrate.
- If driven by the EEPROM, connect the IIC_CLK/IIC_SDA/EE_WP pins to the EEPROM corresponding pins.
- The recommended retry period is 100ms to avoid the BLE buffer overflowing when using the UART interface.



Driven by MCU using UART Interface

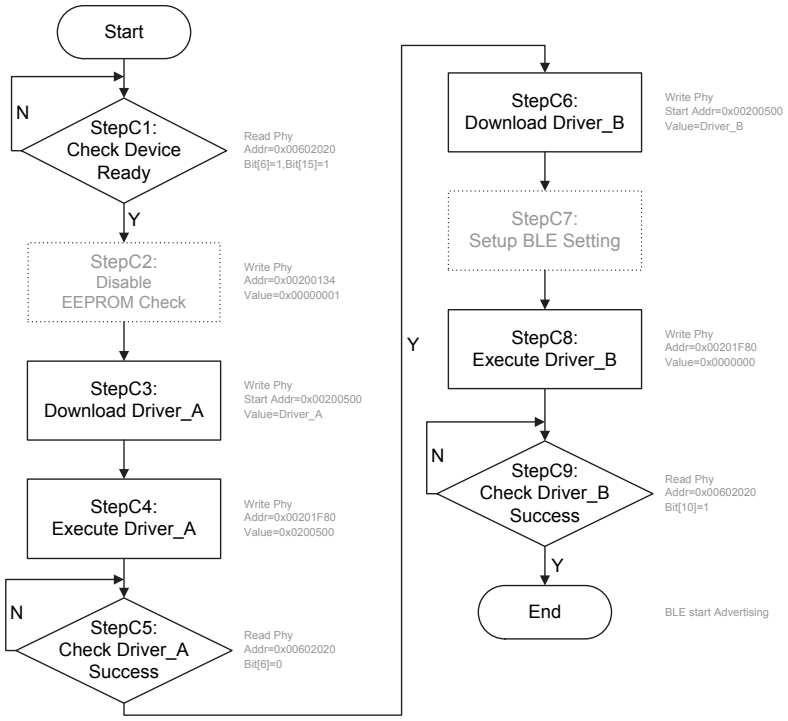


Driven by EEPROM using UART Interface

UART Interface Default Setup

Flow control : Enable
Data length : 8 bits
Stop bit : 1 bit
Parity : None
Baudrate : 115200 bps

Initialization Flow for the Mode C – Driven by the MCU using the UART Interface



Procedures for the Mode C – Driven by the MCU using the UART Interface

- **StepC1: Check Device Ready**
  - Read the Physical Address 0x0060_2020 to check whether the device is powered on and is ready or not. If Bit[15]=1, this means that the device is powered on and waiting for the Driver.
  - Reading the Physical Address 0x0060_2020 should be bit[15]=1,bit[10]=0,bit[6]=1.
- **StepC2: Disable EEPROM Check – optional**
  - If the IIC_SDA pin is kept high or floating, the EEPROM Check function must be disabled. Otherwise the device will remain in an endless check loop.
  - Write the Physical Address 0x0020_0134 with a value of 0x0000_0001.
- **StepC3: Download Driver_A**
  - The MCU sends the Driver_A code to the device and writes the Driver_A code to the Physical Address 0x0020_0500.
- **StepC4: Execute Driver_A**
  - Write the Physical Address 0x0020_1F80 with a value of 0x0020_0500 after which the Driver_A code will start to boot for the variable setup, RF calibration etc.
- **StepC5: Check Driver_A Success**
  - Read the Physical Address 0x0060_2020 to check whether the Driver_A execution is ready or not. If Bit[6]=0, then this means that the device Driver_A boot has been successful.
  - Reading the Physical Address 0x0060_2020 should be bit[15]=1, bit[10]=0, bit[6]=0.
  - The recommended retry period is 100ms to avoid the BLE buffer overflowing when using the UART interface.
- **StepC6: Download Driver_B**
  - The MCU sends the Driver_B code to the device and writes the Driver_B code to the Physical Address 0x0020_0500.
- **StepC7: Setup BLE Setting – optional**
  - The MCU sends the Read/Write Command to setup the BLE device, such as BD_ADDR, BD_ Name, etc.

• **StepC8: Execute Driver_B** ♦ The MCU writes the Physical Address 0x0020_1F80 with a value 0f 0x0000_0000 after which the Driver_B code will start to boot.
• **StepC9: Check Driver_B Success** ♦ Read the Physical Address 0x0060_2020 to check whether the Driver_B boot is ready or not. If Bit[10]=1, than this means that the device Driver_B boot has been successful. ♦ Reading the Physical Address 0x0060_2020 should be bit[15]=1, bit[10]=1, bit[6]=0.

Example for the Mode_C – Driven by the MCU using the UART Interface

Text Font	Normal	**Bold**	*Italic*
Subject	MCU_Rx	**MCU_Tx**	*Driver*
• **StepC1: Check Device Ready** ♦ Read the Physical Address 0x0060_2020 to check whether the device is powered on and is ready or not. If Bit[15]=1, this means that the device is powered on and waiting for the Driver. ♦ Reading the Physical Address 0x0060_2020 should be bit[15]=1, bit[10]=0, bit[6]=1.			
UART_Tx=560120206000 UART_Rx=57010000202060004080000 0			
• **StepC2: Disable EEPROM Check – optional** ♦ If the IIC_SDA pin is kept high or floating, the EEPROM Check function must be disabled. Otherwise the device will remain in an endless check loop. ♦ Write the Physical Address 0x0020_0134 with a value of 0x0000_0001.			
UART_Tx=55010000340120000100000 0			
• **StepC3: Download Driver_A** ♦ The MCU sends the Driver_A code to the device and writes the Driver_A code to the Physical Address 0x0020_0500.			
UART_Tx=5538000000052000 **UART_Tx=*D01400E035029014D0147A1060935093403A030C7093CD7B9014D01475106093*** **UART_Tx=*5393403A030C7393CD7B9014D014F0E313369014D014EFE37FFE9014D0146D10*** **UART_Tx=*60935393403A030C7393CD7B9014D014681060935393403A030C7393CD7B9014*** **UART_Tx=*D014F0E3BD079014D014F0E3B9079014780740006E1304B327B348B3023246B3*** **UART_Tx=*013246B33C78D114C1348F44789463E4FF3063EC090078B4661369B4679463EC*** **UART_Tx=*A20067B46D94803B6DB46894A13B68B46894803B68B4363361B46D94A53B0130*** **UART_Tx=*6DB4436C6432FFE3D7FF00C02050FA300540F0E3C9277612003246B344B36194***			
UART_Tx=55380000E0052000 **UART_Tx=*853B61B49114C11445740032C36C03290B6C0604885B80840022105889744964*** **UART_Tx=*FA0B81146C126093403B040801326B124EA36A126E83312349126C5B6C5A418B*** **UART_Tx=*E0336F4343DC0822C1336F434493A83A44B32031401222B28031214123B25993*** **UART_Tx=*42EC1C0059B35993853A59B302EACBFF52B33C113CB35393843A53B35393A53A*** **UART_Tx=*53B35393A23A53B35393A43A53B37611003240B3351124B33C78C214C0323411*** **UART_Tx=*434200EA3206E0358242AF45945C80940022E03380A90A646F430121F50B3093*** **UART_Tx=*471122DC3310319322DC3410329322DC3510339322DC3610349322DC371023D8*** ***...***			
• **StepC4: Execute Driver_A** ♦ Write the Physical Address 0x0020_1F80 with a value of 0x0020_0500 after which the Driver_A code will start to boot for the variable setup, RF calibration etc.			
UART_Tx=55010000801F200000052000			
• **StepC5: Check Driver_A Success** ♦ Read the Physical Address 0x0060_2020 to check whether the Driver_A execution is ready or not. If Bit[6]=0, then this means that the device Driver_A boot has been successful. ♦ Reading the Physical Address 0x0060_2020 should be bit[15]=1, bit[10]=0, bit[6]=0. ♦ The recommended retry period is 100ms to avoid the BLE buffer overflowing when using the UART interface.			
UART_Tx=560120206000 UART_Rx=57010000202060000080000 0			

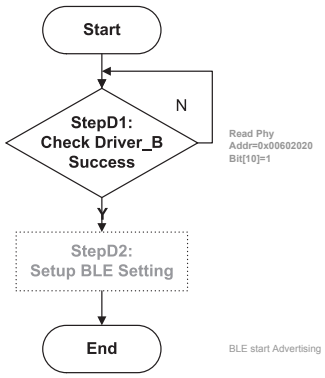
Text Font	Normal	**Bold**	*Italic*
Subject	MCU_Rx	**MCU_Tx**	*Driver*
• **StepC6: Download Driver_B** ♦ The MCU sends the Driver_B code to the device and writes the Driver_B code to the Physical Address 0x0020_0500.			
UART_Tx=5538000000052000 **UART_Tx=*D01400E039009014D0147A1060935093403A030C7093CD7B9014D01475106093*** **UART_Tx=*5393403A030C7393CD7B9014D01400E0D7009014D014EFE37FFE9014D0146D10*** **UART_Tx=*60935393403A030C7393CD7B9014D0146810609353934O3A030C7393CD7B9014*** **UART_Tx=*D014F0E3BD079014D014F0E3B907901478074000631060934310418B33C780000*** **UART_Tx=*780740002C0520006B12003121B3409302C48028030C013122B322C48028040C*** **UART_Tx=*6512023242B36412013121B33C78D014F0E3B42961124093933A40B3C1336F43*** **UART_Tx=*4493943A44B39014D4149D116084403B0508FFE3EEFF00A43E047A114083403A***			
UART_Tx=55380000E0052000 **UART_Tx=*0308193240A378118083403C1608FFE3E0FF03EAFF01C2642E0C71116083015B*** **UART_Tx=*0074721100A30274E0C78028020C015C6C1160830D641F0CAD110036E5D86320*** **UART_Tx=*C5DC632080C12070871100842258F0E3A90880C12074E5DC63206084153B0408*** **UART_Tx=*002360A408047E1040112083408249590C0A440A39414581063920 3C48028030C*** **UART_Tx=*013125B223C48028040C7310023245B33C78C1336F43449342EC001844B34593*** **UART_Tx=*42EC001845B380336443521060B272106093321033B30F3267104DB301EAFEFF*** **UART_Tx=*2EB300324CB3013241B302312BB34BB33C78000000706000049060008200 2000***			
...			
• **StepC7: Setup BLE Setting – optional** ♦ The MCU sends the Read/Write Command to setup the BLE device, such as BD_ADDR, BD_Name, etc.			
UART_Tx=253110493D3530306D735F544F3D3673656300 UART_Rx=263100			
UART_Tx=2031 UART_Rx=213110493D3530306D735F544F3D3673656300			
UART_Tx=25320406000000 UART_Rx=263200			
UART_Tx=2032 UART_Rx=21320406000000			
UART_Tx=25330615181409B0BA UART_Rx=263300			
UART_Tx=2033 UART_Rx=21330615181409B0BA			
...			
• **StepC8: Execute Driver_B** ♦ The MCU writes the Physical Address 0x0020_1F80 with a value 0f 0x0000_0000 after which the Driver_B code will start to boot.			
UART_Tx=55010000801F200000000000			
• **StepC9: Check Driver_B Success** ♦ Read the Physical Address 0x0060_2020 to check whether the Driver_B boot is ready or not. If Bit[10]=1, then this means that the device Driver_B boot is successful. ♦ Reading the Physical Address 0x0060_2020 should be bit[15]=1, bit[10]=1, bit[6]=0.			
UART_Tx=560120206000 UART_Rx=570101012020600001840000			
• **StepN: Connection Established**			
UART_Rx=22FF11			
• **StepN+1: Receive Payload Packet from cell phone**			
UART_Rx=2203123456 (payload=123456)			
...			

Driver Time for the Mode_C – Driven by the MCU using the UART Interface

The driver time may be different due to various versions. The following driver time is measured for the driver version 161025A5, in which the Driver_A size=0x02C0 bytes and the Driver_B size=0x1480 bytes.

UART Baudrate=115200 bps, total time=1000ms (typical), C1+C2+C3+C4+C5+C6+C8+C9

Initialization Flow for the Mode D – Driven by the EEPROM using the UART Interface



Procedures for the Mode D – Driven by the EEPROM using the UART Interface

- **StepD1: Check Driver_B Success**
  - ♦ Read the Physical Address 0x0060_2020 to check whether the Driver_B execution is ready or not. If Bit[10]=1, then this means that the device Driver_B boot has been successful.
  - ♦ Reading the Physical Address 0x0060_2020 should be bit[15]=1, bit[10]=1, bit[6]=0.
- **StepD2: Setup BLE Setting – optional**
  - ♦ The MCU sends the Read/Write Command to setup the BLE device, such as BD_ADDR, BD_Name, etc.

Example for the Mode_D – Driven by the EEPROM using the UART Interface

Text Font	Normal	**Bold**	*Italic*
Subject	MCU_Rx	**MCU_Tx**	*Driver*
• **StepD1: Check Driver_B Success** ♦ Read the Physical Address 0x0060_2020 to check whether the Driver_B execution is ready or not. If Bit[10]=1,then this means that the device Driver_B boot has been successful. ♦ Reading the Physical Address 0x0060_2020 should be bit[15]=1, bit[10]=1, bit[6]=0.			
UART_Tx=560120206000 UART_Rx=57010101202060000184000 0			
• **StepD2: Setup BLE Setting – optional** ♦ The MCU sends the Read/Write Command to setup the BLE device, such as BD_ADDR, BD_Name, etc.			
UART_Tx=253110493D3530306D735F544F3D3673656300 UART_Rx=263100			
UART_Tx=2031 UART_Rx=213110493D3530306D735F544F3D3673656300			
UART_Tx=25320406000000 UART_Rx=263200			
UART_Tx=2032 UART_Rx=21320406000000			
UART_Tx=25330615181409B0BA UART_Rx=263300			
UART_Tx=2033 UART_Rx=21330615181409B0BA			
...			

Text Font	Normal	**Bold**	*Italic*
Subject	MCU_Rx	**MCU_Tx**	*Driver*
• StepN: Connection Established			
UART_Rx=22FF11			
• StepN+1: Receive Payload Packet from cell phone			
UART_Rx=2203123456 (payload=123456)			
...			

Driver Time for the Mode_D – Driven by the EEPROM using the UART Interface

The driver time may be different due to various versions. The following driver time is measured for the driver version 161025A5, in which the Driver_A size=0x02C0 bytes and the Driver_B size=0x1480 bytes.

First boot with the RF calibration: UART Baudrate=115200 bps, total time=850ms (typical), D1 only.

After booting the device will write the RF calibration value into the EEPROM.

Second boot without RF calibration: UART Baudrate=115200 bps, total time=350ms (typical), D1 only.