

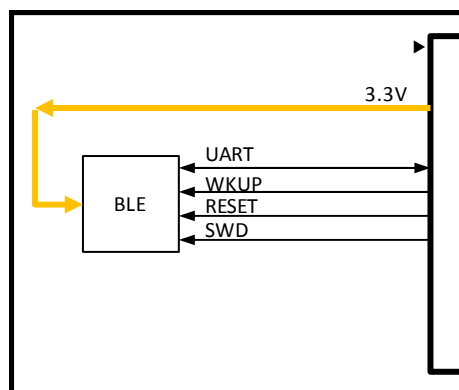
AC02A BLE Sugar

1. Description

The leaf which is equipped with the technical certification satisfied BLE module BGM11S22F256GA-V of Silicon Labs can connect with MCU leaf by UART.

2. Leaf specification

2-1. Block diagram



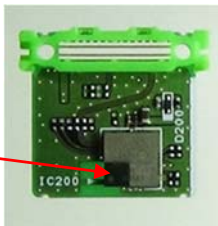

2-2. Power supply specification

Symbol	Parameter	Condition	Min.	Typ.	Max.
Vdd	Power Supply Voltage	—	2.4V	3.3V	3.8V
Idd	Operating current	Active	-	3.8mA	-
		Sleep	-	2.8uA	-

2-3. Main parts

Reference No.	Part name	Part number	Vendor name	note
IC200	BLE module	BGM11S22F256GA-V2	Silicon Labs	—

2-4. Appearance

Top Side	Back Side
	

2-5. Pinout

Name	Function
A2	TXD : UART send D9 can also be the alternative due to the replacement of chip's resistor
A1	RXD : UART receive D8 can also be the alternative due to the replacement of chip's resistor
D7	WAKEUP : wakeup H : wakeup
RESET	RST : reset
SWCLK	Debug I/F clock
SWDIO	Debug I/F data input/output
3V3	3.3V power input
GND	GND

3. BLEModule(BGM11S22F256GA-V2) Specifications

3-1. Description

Item	内容
SoC	EFR32BG1 (ARM Cortex-M4)
Bluetooth version	4.2
Frequency range	2400M ~ 2483.5MHz
Internet Security	<ul style="list-style-type: none"> General Purpose CRC Random Number Generator Hardware Cryptographic Acceleration for AES 128/256,SHA-1, SHA-2 (SHA-224 and SHA-256) and ECC
RX sensitivity	-90 dBm @ 1 Mbit/s GFSK
TX power	+8dBm
RF certification	CE, full FCC, ISED Canada, Japan and South-Korea
Flash	256KB
RAM	32KB
Interfaces	UART

3-2. Electrical characteristics

3-2-1. Absolute Maximum Ratings

Parameter	Value
Operating Temperature	-40°C to +85°C
Maximum Operation Voltage	3.8V

3-2-2. Rating

Symbol	Parameter	Condition	Min.	Typ.	Max.
Vdd	Power Supply Voltage	–	2.4V	3.3V	3.8V
Idd	EM0 Active mode	38 MHz HFRCO all peripherals disabled	-	3.8mA	3.99mA
	EM1 Sleep mode	38 MHz HFRCO	-	1.33mA	1.44mA

		all peripherals disabled			
	EM2 Deep Sleep mode	Full RAM retention and RTCC running from LFXO	-	33uA	-
	EM3 Stop mode	Full RAM retention and CRYOTIMER running from ULFRCO	-	2.8uA	6uA
	EM4H Hibernate mode	128 byte RAM retention, RTCC running from LFXO	-	1.1uA	-
	EM4S Shutoff mode	no RAM retention, no RTCC	-	0.04uA	0.20uA
	Receive mode, active packet reception (MCU in EM1 @38.4 MHz, peripheral clocks disabled)	1 Mbit/s, 2GFSK, F = 2.4 GHz, Radio clock prescaled by 4	-	9.0mA	-
	Transmit mode (MCU in EM1@ 38.4 MHz, peripheral clocks disabled)	0 dBm output power, Radio clock prescaled by 3	-	8.2mA	-
		2 dBm output power	-	16.5mA	-
		8 dBm output power	-	24.6mA	-

3-3. Link destination of data sheet

<https://jp.silabs.com/products/wireless/bluetooth/bluetooth-low-energy-modules/bgm11s-bluetooth-sip-module>

3-4.

3-4-1. The control of BLE

include file : BGLib.h(Leaf Libraies)

Definition	Description
BGLib ble112(HardwareSerial *module, HardwareSerial *output, uint8_t pMode)	<p>Creating an instance of BGLib</p> <p>【statement】</p> <p>BGLib ble112(HardwareSerial *module, HardwareSerial *output, pMode)</p> <p>【parameter】</p> <p>ble112: the name of instance</p> <p>module: The instance of serial board communicating with BLE leaf</p> <p>output: The instance of serial board to which BLE leaf output</p> <p>Null fixed</p> <p>pMode: packet mode 0 fixed</p> <p>【return value】</p> <p>null</p>

<p>ble112.ble_cmd_le_gap_set_adv_parameters(interval_min, interval_max, channel_map)</p>	<p>Set advertisement parameters</p> <p>【statement】</p> <p>ble_cmd_le_gap_set_adv_parameters(uint16 interval_min, uint16 interval_max, uint8 channel_map)</p> <p>【parameter】</p> <p>ble112: the name of instance</p> <p>interval_min: Minimum advertising interval. Value in units of 0.625 ms</p> <ul style="list-style-type: none"> • Range: 0x20 to 0xFFFF • Time range: 20 ms to 40.96 s <p>Default value: 100 ms</p> <p>interval_max:Maximum advertising interval. Value in units of 0.625 ms</p> <ul style="list-style-type: none"> • Range: 0x20 to 0xFFFF • Time range: 20 ms to 40.96 s <p>Default value: 200 ms</p> <p>channel_map: Advertising channel map which determines which of the three channels will be used for advertising. This value is given as a bitmask.</p> <ul style="list-style-type: none"> • 1: Advertise on CH37 • 2: Advertise on CH38 • 3: Advertise on CH37 and CH38 • 4: Advertise on CH39 • 5: Advertise on CH37 and CH39 • 6: Advertise on CH38 and CH39 • 7: Advertise on all channels <p>Default value: 7</p> <p>【return value】</p> <p>0</p>
<p>ble112.ble_cmd_le_gap_discover(mode)</p>	<p>Bluetooth discovery mode setting</p> <p>【statement】</p> <p>ble_cmd_le_gap_discover(uint8 mode)</p> <p>【parameter】</p> <p>ble112: the name of instance</p> <p>mode: discovery mode refer to 'enum_le_gap_discover_mode'</p> <p>【return value】</p> <p>0</p>

ble112.ble_cmd_le_gap_set_adv_data(scan_rsp, adv_data_len, adv_data);	<p>Set advertisement data</p> <p>【statement】</p> <pre>ble_cmd_le_gap_set_adv_data(uint8 scan_rsp, uint8 adv_data_len, const uint8 *adv_data_data)</pre> <p>【parameter】</p> <p>ble112: the name of instance</p> <p>scan_rsp: This value selects if the data is intended for advertising packets, scan response packets or advertising packet in OTA. Values:</p> <ul style="list-style-type: none"> • 0: Advertising packets • 1: Scan response packets • 2: OTA advertising packets • 4: OTA scan response packets <p>adv_data_len: advertise data length, maximum: 31 byte</p> <p>adv_data_data: advertise data</p> <p>【return value】</p> <p>0</p>
ble112.ble_cmd_le_gap_start_advertising(handle, discover, connect)	<p>Start advertising</p> <p>【statement】</p> <pre>ble_cmd_le_gap_start_advertising(uint8 handle, uint8 discover, uint8 connect)</pre> <p>【parameter】</p> <p>ble112: the name of instance</p> <p>handle: BLE leaf handle</p> <p>discover: Discoverable mode refer to 'enum_le_gap_discoverable_mode'</p> <p>connect: Connectable mode refer to 'enum_le_gap_connectable_mode'</p> <p>【return value】</p> <p>0</p>
ble112.ble_cmd_le_gap_stop_advertising(handle)	<p>Stop advertising</p> <p>【statement】</p> <pre>ble_cmd_le_gap_stop_advertising(uint8 handle)</pre> <p>【parameter】</p> <p>ble112: the name of instance</p> <p>【return value】</p> <p>0</p>
ble112.checkActivity(timeout)	<p>Wait for response</p> <p>【statement】</p> <pre>checkActivity(uint16_t timeout)</pre> <p>【parameter】</p> <p>ble112: the name of instance</p> <p>timeout: value of timeout (ms)</p> <p>【return value】</p>

	0 :nobusy 1 :busy
ble112.ble_cmd_gatt_set_characteristic_notification(connection, characteristic, flags)	Set notification to GATT Server 【statement】 ble_cmd_gatt_set_characteristic_notification(uint8 connection, uint16 characteristic, uint8 flags) 【parameter】 ble112: the name of instance connection: Connection handle characteristic:GATT characteristic handle flags: Characteristic client configuration flags <ul style="list-style-type: none"> • 0: Disable notifications and indications • 1: Notification • 2: Indication 【return value】 0
ble112.ble_cmd_gatt_server_send_characteristic_notification(connection, characteristic, value_len, (const uint8 *)value_data)	Send notification to GATT clients 【statement】 ble_cmd_gatt_server_send_characteristic_notification(uint8 connection, uint16 characteristic, uint8 value_len, const uint8 *value_data) 【parameter】 ble112: the name of instance connection: Connection handle <ul style="list-style-type: none"> • 0xff: Sends notification or indication to all connected devices. • Other: Connection handle characteristic:Characteristic handle refer to 'enum_le_gap_discoverable_mode' value_len: value length value: Value to be notified or indicated 【return value】 0

<pre>ble112.ble_cmd_gatt_write_characteristic_value(connection, characteristic, value_len, *value_data);</pre>	<p>Set notification to GATT Server</p> <p>【statement】</p> <pre>ble_cmd_gatt_write_characteristic_value(uint8 connection, uint16 characteristic, uint8 value_len, const uint8 *value_data)</pre> <p>【parameter】</p> <p>ble112: the name of instance connection: Connection handle characteristic:GATT characteristic handle value_len:Characteristic value length value_data:Characteristic value</p> <p>【return value】</p> <p>0</p>
<pre>ble112.ble_cmd_le_gap_set_scan_parameters(scan_interval, scan_window, active)</pre>	<p>Set scan parameters</p> <p>【statement】</p> <pre>ble_cmd_le_gap_set_scan_parameters(uint16 scan_interval, uint16 scan_window, uint8 active)</pre> <p>【parameter】</p> <p>ble112: the name of instance scan_interval: Scanner interval</p> <ul style="list-style-type: none"> • Time = Value x 0.625 ms • Range: 0x0004 to 0x4000 • Time Range: 2.5 ms to 10.24 s <p>Default value: 10 ms</p> <p>scan_window: Scan window. The duration of the scan.</p> <ul style="list-style-type: none"> • Time = Value x 0.625 ms • Range: 0x0004 to 0x4000 • Time Range: 2.5 ms to 10.24 s <p>Default value: 10 ms Note that packet reception is aborted if it has been started before scan window ends.</p> <p>active : Scan type indicated by a flag</p> <ul style="list-style-type: none"> • 0: Passive scanning • 1: Active scanning <p>Default value: 0</p> <p>【return value】</p> <p>0</p>
<pre>ble112.ble_cmd_le_gap_end_procedure()</pre>	<p>Stop using current GAP procedure</p> <p>【statement】</p> <pre>ble_cmd_le_gap_end_procedure(void)</pre> <p>【parameter】</p> <p>ble112: the name of instance</p> <p>【return value】</p> <p>0</p>

ble112.ble_cmd_le_gap_connect(address, address_type, initiating_phy)	<p>Connect with devices</p> <p>【statement】</p> <pre>ble_cmd_le_gap_connect(bd_addr address, uint8 address_type, uint8 initiating_phy)</pre> <p>【parameter】</p> <p>ble112: the name of instance</p> <p>address: Address of the device to connect to</p> <p>address_type: Address type of the device to connect to refer to 'enum_le_gap_address_types'</p> <p>initiating_phy: The initiating PHY.</p> <ul style="list-style-type: none"> • 1: LE 1M PHY • 4: LE Coded PHY <p>【return value】</p> <p>0</p>
ble112.ble_cmd_le_connection_close(connection)	<p>Disconnect from devices</p> <p>【statement】</p> <pre>ble_cmd_le_connection_close(uint8 connection)</pre> <p>【parameter】</p> <p>ble112: the name of instance</p> <p>connection: Handle of the connection</p> <p>【return value】</p> <p>0</p>
ble112.ble_cmd_system_reset(boot_in_dfu)	<p>Run system reset</p> <p>【statement】</p> <pre>ble_cmd_system_reset(uint8 boot_in_dfu)</pre> <p>【parameter】</p> <p>ble112: the name of instance</p> <p>boot_in_dfu: Boot mode</p> <ul style="list-style-type: none"> • 0: Normal reset • 1: Boot to UART DFU mode • 2: Boot to OTA DFU mode <p>【return value】</p> <p>0</p>
ble112.ble_cmd_system_halt(halt)	<p>Shift into SLEEP mode</p> <p>【statement】</p> <pre>ble_cmd_system_halt(uint8 halt)</pre> <p>【parameter】</p> <p>ble112: the name of instance</p> <p>halt: halt mode</p> <ul style="list-style-type: none"> • 1: halt • 0: resume <p>【return value】</p> <p>0</p>

ble112.getLastEvent()	Get last event 【statement】 getLastEvent() 【parameter】 ble112: the name of instance 【return value】 lastEvent[0] : Message class: System lastEvent[1] :Message ID
-----------------------	--

enum_le_gap_connectable_mode

Value	Name	Description
0	le_gap_non_connectable	Non-connectable non-scannable.
1	le_gap_directed_connectable	Directed connectable (RESERVED, DO NOT USE)
2	le_gap_undirected_connectable	Undirected connectable scannable. Deprecated, replaced by enum le_gap_connectable_scannable. This mode can only be used in legacy advertising PDUs.
2	le_gap_connectable_scannable	Undirected connectable scannable. This mode can only be used in legacy advertising PDUs.
3	le_gap_scannable_non_connectable	Undirected scannable (Non-connectable but responds to scan requests)
4	le_gap_connectable_non_scannable	Undirected connectable non-scannable. This mode can only be used in extended advertising PDUs.

enum_le_gap_discoverable_mode

Value	Name	Description
0	le_gap_non_discoverable	Not discoverable
1	le_gap_limited_discoverable	Discoverable using both limited and general discovery procedures
2	le_gap_general_discoverable	Discoverable using general discovery procedure
3	le_gap_broadcast	Device is not discoverable in either limited or generic discovery procedure, but may be discovered by using the Observation procedure
4	le_gap_user_data	Send advertising and/or scan response data defined by the user using le_gap_bt5_set_adv_data. The limited/general discoverable flags are defined by the user.

enum_le_gap_discover_mode

Value	Name	Description
0	le_gap_discover_limited	Discover only limited discoverable devices
1	le_gap_discover_generic	Discover limited and generic discoverable devices
2	le_gap_discover_observation	Discover all devices

enum_le_gap_address_type

Value	Name	Description
0	le_gap_address_type_public	Public address
1	le_gap_address_type_random	Random address
2	le_gap_address_type_public_identity	Public identity address resolved by stack
3	le_gap_address_type_random_identity	Random identity address resolved by stack

3-5. Event callback function

Event callback function	Description
ble_evt_gatt_server_attribute_value	The callback function pointer is being called when the attribute value in local GATT database is changed by remoted GATT client [statement] name.ble_evt_gatt_server_attribute_value = my_evt_gatt_server_attribute_value;
ble_evt_le_connection_open	The callback function pointer is being called when center is connected [statement] name..ble_evt_le_connection_open = my_evt_le_connection_open;
ble_evt_le_connection_closed	The callback function pointer is being called when center is disconnected [statement] name.ble_evt_le_connection_closed = my_evt_le_connection_closed;
ble_evt_system_boot	The callback function pointer is being called when system is launched [statement] name.ble_evt_system_boot = my_evt_system_boot;
ble_evt_system_awake	The callback function pointer is being called when system has returned from sleep mode [statement] name.ble_evt_system_aware = my_evt_system_aware;
ble_evt_le_gap_scan_response	The callback function pointer is being called when receiving scan response [statement] name.ble_evt_le_gap_scan_response = my_evt_le_gap_scan_response;

3-6. Power saving

Sleep mode saves power and can be done by the function below.

function : ble112.ble_cmd_system_halt(1)

Wakeup if the WAKEUP signal of D7 is high.