

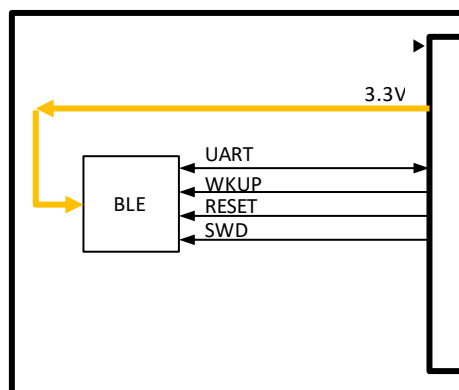
# AC02A BLE Sugar

## 1. 概要

Silicon Labs の技術的認証済み BLE モジュール BGM11S22F256GA-V2 を搭載したリーフである。MCU リーフとは UART で接続される。

## 2. リーフ仕様

### 2-1. ブロック図



### 2-2. 電源仕様

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. 主要部品

部品番号	部品名	型番	ベンダー名	備考
IC200	BLE モジュール	BGM11S22F256GA-V2	Silicon Labs	—

### 2-4. 外観

表面	裏面

## 2-5. ピンアウト

Name	Function
A2	TXD : UART 送信 チップ抵抗の付け替えで D9 に変更可
A1	RXD : UART 受信 チップ抵抗の付け替えで D8 に変更可
D7	WAKEUP : ウェイクアップ H : ウェイクアップ
RESET	RST : リセット
SWCLK	デバッグ I/F クロック
SWDIO	デバッグ I/F データ入出力
3V3	3.3V 入力
GND	GND

## 3. BLE モジュール(BGM11S22F256GA-V2)仕様

### 3-1. 概要

項目	内容
SoC	EFR32BG1 (ARM Cortex-M4)
Bluetooth version	4.2
Frequency range	2400M ~ 2483.5MHz
Internet Security	<ul style="list-style-type: none"> <li>General Purpose CRC</li> <li>Random Number Generator</li> <li>Hardware Cryptographic Acceleration for AES 128/256,SHA-1, SHA-2 (SHA-224 and SHA-256) and ECC</li> </ul>
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. 電気的特性

#### 3-2-1. 最大定格

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

#### 3-2-2. 定格

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 all peripherals disabled	-	1.33mA	1.44mA

	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. データシートリンク先

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

### 3-4. 主な関数とライブラリ

#### 3-4-1. BLE の制御

include file : BGLib.h(Leaf Libraies)

関数	概要
BGLib ble112( HardwareSerial *module, HardwareSerial *output, uint8_t pMode )	<p>BGLib のインスタンスを作成します</p> <p>【構文】 BGLib ble112(HardwareSerial *module, HardwareSerial *output, pMode)</p> <p>【パラメータ】 ble112: インスタンス名 module: BLE リーフと通信するシリアルポートののインスタンス output: BLE リーフが出力するシリアルポートののインスタンス Null 固定 pMode: パケットモード 0 固定</p> <p>【戻り値】 なし</p>

<pre>ble112.ble_cmd_le_gap_set_adv_parameters( interval_min, interval_max, channel_map )</pre>	<p>アドバタイズのパラメータ設定を行います</p> <p>【構文】</p> <pre>ble_cmd_le_gap_set_adv_parameters( uint16 interval_min, uint16 interval_max, uint8 channel_map )</pre> <p>【パラメータ】</p> <p>ble112: インスタンス名</p> <p>interval_min: Minimum advertising interval. Value in units of 0.625 ms</p> <ul style="list-style-type: none"> <li>• Range: 0x20 to 0xFFFF</li> <li>• Time range: 20 ms to 40.96 s</li> </ul> <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"> <li>• Range: 0x20 to 0xFFFF</li> <li>• Time range: 20 ms to 40.96 s</li> </ul> <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"> <li>• 1: Advertise on CH37</li> <li>• 2: Advertise on CH38</li> <li>• 3: Advertise on CH37 and CH38</li> <li>• 4: Advertise on CH39</li> <li>• 5: Advertise on CH37 and CH39</li> <li>• 6: Advertise on CH38 and CH39</li> <li>• 7: Advertise on all channels</li> </ul> <p>Default value: 7</p> <p>【戻り値】</p> <p>0</p>
<pre>ble112.ble_cmd_le_gap_discover( mode )</pre>	<p>Bluetooth discovery mode 設定</p> <p>【構文】</p> <pre>ble_cmd_le_gap_discover( uint8 mode )</pre> <p>【パラメータ】</p> <p>ble112: インスタンス名</p> <p>mode: discovery mode</p> <p>enum_le_gap_discover_mode 参照</p> <p>【戻り値】</p> <p>0</p>

<pre>ble112.ble_cmd_le_gap_set_adv_data( scan_rsp, adv_data_len, adv_data );</pre>	<p>アドバタイズデータの設定を行います</p> <p>【構文】</p> <pre>ble_cmd_le_gap_set_adv_data( uint8 scan_rsp, uint8 adv_data_len, const uint8 *adv_data_data )</pre> <p>【パラメータ】</p> <p>ble112: インスタンス名</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"> <li>• 0: Advertising packets</li> <li>• 1: Scan response packets</li> <li>• 2: OTA advertising packets</li> <li>• 4: OTA scan response packets</li> </ul> <p>adv_data_len:設定するアドバタイズデータ長 最大 31 バイト</p> <p>adv_data_data:アドバタイズデータ</p> <p>【戻り値】</p> <p>0</p>
<pre>ble112.ble_cmd_le_gap_start_advertising(handle, discover, connect)</pre>	<p>アドバタイズを開始します</p> <p>【構文】</p> <pre>ble_cmd_le_gap_start_advertising( uint8 handle, uint8 discover, uint8 connect )</pre> <p>【パラメータ】</p> <p>ble112: インスタンス名</p> <p>handle: BLE leaf handle</p> <p>discover:Discoverable mode enum_le_gap_discoverable_mode 参照</p> <p>connect: Connectable mode enum_le_gap_connectable_mode 参照</p> <p>【戻り値】</p> <p>0</p>
<pre>ble112.ble_cmd_le_gap_stop_advertising( handle )</pre>	<p>アドバタイズを終了します</p> <p>【構文】</p> <pre>ble_cmd_le_gap_stop_advertising( uint8 handle )</pre> <p>【パラメータ】</p> <p>ble112: インスタンス名</p> <p>【戻り値】</p> <p>0</p>

ble112.checkActivity( timeout )	<p>応答があるまで待ちます</p> <p>【構文】</p> <pre>checkActivity( uint16_t timeout )</pre> <p>【パラメータ】</p> <p>ble112: インスタンス名</p> <p>timeout: タイムアウト値 ms</p> <p>【戻り値】</p> <p>0 :nobusy</p> <p>1 :busy</p>
ble112.ble_cmd_gatt_set_characteristic_notification(connection, characteristic, flags)	<p>GATT Server に notification を設定します</p> <p>【構文】</p> <pre>ble_cmd_gatt_set_characteristic_notification( uint8 connection, uint16 characteristic, uint8 flags )</pre> <p>【パラメータ】</p> <p>ble112: インスタンス名</p> <p>connection: Connection handle</p> <p>characteristic:GATT characteristic handle</p> <p>flags: Characteristic client configuration flags</p> <ul style="list-style-type: none"> <li>• 0: Disable notifications and indications</li> <li>• 1: Notification</li> <li>• 2: Indication</li> </ul> <p>【戻り値】</p> <p>0</p>
ble112.ble_cmd_gatt_server_send_characteristic_notification( connection, characteristic, value_len, (const uint8 *)value_data )	<p>GATT clients に notification を送信します</p> <p>【構文】</p> <pre>ble_cmd_gatt_server_send_characteristic_notification( uint8 connection, uint16 characteristic, uint8 value_len, const uint8 *value_data )</pre> <p>【パラメータ】</p> <p>ble112: インスタンス名</p> <p>connection: Connection handle</p> <ul style="list-style-type: none"> <li>• 0xff: Sends notification or indication to all connected devices.</li> <li>• Other: Connection handle</li> </ul> <p>characteristic:Characteristic handle</p> <p>enum_le_gap_discoverable_mode 参照</p> <p>value_len: value length</p> <p>value: Value to be notified or indicated</p> <p>【戻り値】</p> <p>0</p>

<pre>ble112.ble_cmd_gatt_write_characteristic_value(connection, characteristic, value_len, *value_data);</pre>	<p>GATT Server に notification を設定します</p> <p>【構文】</p> <pre>ble_cmd_gatt_write_characteristic_value( uint8 connection, uint16 characteristic, uint8 value_len, const uint8 *value_data )</pre> <p>【パラメータ】</p> <p>ble112: インスタンス名</p> <p>connection: Connection handle</p> <p>characteristic: GATT characteristic handle</p> <p>value_len: Characteristic value length</p> <p>value_data: Characteristic value</p> <p>【戻り値】</p> <p>0</p>
<pre>ble112.ble_cmd_le_gap_set_scan_parameters(scan_interval, scan_window, active)</pre>	<p>スキャンパラメータを設定します</p> <p>【構文】</p> <pre>ble_cmd_le_gap_set_scan_parameters( uint16 scan_interval, uint16 scan_window, uint8 active )</pre> <p>【パラメータ】</p> <p>ble112: インスタンス名</p> <p>scan_interval: Scanner interval</p> <ul style="list-style-type: none"> <li>• Time = Value x 0.625 ms</li> <li>• Range: 0x0004 to 0x4000</li> <li>• Time Range: 2.5 ms to 10.24 s</li> </ul> <p>Default value: 10 ms</p> <p>scan_window: Scan window. The duration of the scan.</p> <ul style="list-style-type: none"> <li>• Time = Value x 0.625 ms</li> <li>• Range: 0x0004 to 0x4000</li> <li>• Time Range: 2.5 ms to 10.24 s</li> </ul> <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"> <li>• 0: Passive scanning</li> <li>• 1: Active scanning</li> </ul> <p>Default value: 0</p> <p>【戻り値】</p> <p>0</p>
<pre>ble112.ble_cmd_le_gap_end_procedure()</pre>	<p>current GAP procedure の使用を停止します</p> <p>【構文】</p> <pre>ble_cmd_le_gap_end_procedure( void )</pre> <p>【パラメータ】</p> <p>ble112: インスタンス名</p> <p>【戻り値】</p> <p>0</p>

ble112.ble_cmd_le_gap_connect(address, address_type, initiating_phy)	<p>デバイスと接続します</p> <p>【構文】</p> <pre>ble_cmd_le_gap_connect( bd_addr address, uint8 address_type, uint8 initiating_phy )</pre> <p>【パラメータ】</p> <p>ble112: インスタンス名</p> <p>address: Address of the device to connect to</p> <p>address_type: Address type of the device to connect to enum_le_gap_address_types 参照</p> <p>initiating_phy: The initiating PHY.</p> <ul style="list-style-type: none"> <li>• 1: LE 1M PHY</li> <li>• 4: LE Coded PHY</li> </ul> <p>【戻り値】</p> <p>0</p>
ble112.ble_cmd_le_connection_close(connection)	<p>デバイスを切断します</p> <p>【構文】</p> <pre>ble_cmd_le_connection_close(uint8 connection)</pre> <p>【パラメータ】</p> <p>ble112: インスタンス名</p> <p>connection: Handle of the connection</p> <p>【戻り値】</p> <p>0</p>
ble112.ble_cmd_system_reset(boot_in_dfu)	<p>システムリセットを実行します</p> <p>【構文】</p> <pre>ble_cmd_system_reset(uint8 boot_in_dfu)</pre> <p>【パラメータ】</p> <p>ble112: インスタンス名</p> <p>boot_in_dfu: Boot mode</p> <ul style="list-style-type: none"> <li>• 0: Normal reset</li> <li>• 1: Boot to UART DFU mode</li> <li>• 2: Boot to OTA DFU mode</li> </ul> <p>【戻り値】</p> <p>0</p>
ble112.ble_cmd_system_halt(halt)	<p>SLEEP モードへ移行します</p> <p>【構文】</p> <pre>ble_cmd_system_halt(uint8 halt)</pre> <p>【パラメータ】</p> <p>ble112: インスタンス名</p> <p>halt: halt mode</p> <ul style="list-style-type: none"> <li>• 1: halt</li> <li>• 0: resume</li> </ul> <p>【戻り値】</p> <p>0</p>



ble112.getLastEvent()	<p>最後に受信したイベントを返します</p> <p>【構文】</p> <pre>getLastEvent()</pre> <p>【パラメータ】</p> <p>ble112: インスタンス名</p> <p>【戻り値】</p> <p>lastEvent[0] : Message class: System</p> <p>lastEvent[1] : Message ID</p>
-----------------------	---

#### 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. イベントコールバック関数

イベントコールバック関数	概要
ble_evt_gatt_server_attribute_value	ローカル GATT データベース内のアトリビュート値が、リモート GATT クライアントによって変更されたときに呼ばれるコールバック関数ポインタ 【構文】 name.ble_evt_gatt_server_attribute_value = my_evt_gatt_server_attribute_value;
ble_evt_le_connection_opend	セントラルと接続したときに呼ばれるコールバック関数ポインタ 【構文】 name..ble_evt_le_connection_opend = my_evt_le_connection_opend;
ble_evt_le_connection_closed	セントラルと接続が終了したときに呼ばれるコールバック関数ポインタ 【構文】 name.ble_evt_le_connection_closed = my_evt_le_connection_closed;
ble_evt_system_boot	システムが起動したときに呼ばれるコールバック関数ポインタ 【構文】 name.ble_evt_system_boot = my_evt_system_boot;
ble_evt_system_aware	システムがスリープモードから復帰したときに呼ばれるコールバック関数ポインタ 【構文】 name.ble_evt_system_aware = my_evt_system_aware;
ble_evt_le_gap_scan_response	スキャン応答を受信した時に呼ばれるコールバック関数ポインタ 【構文】 name.ble_evt_le_gap_scan_response = my_evt_le_gap_scan_response;

### 3-6. 省電力

BLE の省電力は下記関数により Sleep モードに移行する。

関数 : ble112.ble\_cmd\_system\_halt(1)

D7 の WAKEUP 信号を High にすることにより Wakeup する。