# ATM33/e and ATM34/e Series CGM Sensor Example

## Application Note

**SUMMARY:** This application note describes the settings, functionality, and code flow of the CGM_sensor example in the Atmosic SDK to be used in the development of a Continuous Glucose Monitor based on ATM33/e and ATM34/e series.

# Table of Contents



CONFIDENTIAL

# List of Figures



# List of Tables



CONFIDENTIAL

# 1. Overview

This application note describes the settings, functionality, and code flow of the (Continuous Glucose Monitoring) CGM Sensor (CGM_sensor) example code running on an Atmosic ATM33/e and ATM34/e devices. The Atmosic CGM_sensor example is developed as a CGM sensor which exposes glucose measurement and other related data for use in consumer healthcare applications.

Only the ATM33/ATM33e (a.k.a. ATM33/e) series and ATM34/e series of Atmosic devices are supported by this application. The devices are selected through a BOARD variable from the command line. Table 1-1 describes build options of the devices.

Table 1-1 Device Build Options

| Device | Description | Build Option |
| --- | --- | --- |
| ATM33xxx | ATM33/e Series | BOARD=ATMEVK_3330_QN<br>BOARD=ATMEVK_3330e_QN<br>BOARD=ATMEVK_3325_QK |
| ATM34xxx | ATM34/e Series | BOARD=ATMEVK_3405_DQK_2<br>BOARD=ATMEVK_3425_DQK_2<br>BOARD=ATMEVK_3430e_WQN_2 |

## 1.1 Quick Start

- Install Atmosic SDK 6.0.0 or later version.
- Atmosic SDK platform working directories:
  - For ATM33/e devices, use the directory:
    - *platform/atm33/ATM33xx-5*
  - For ATM34/e devices, use the directory:
    - *platform/atm34/ATM34xx-2*

CONFIDENTIAL

- Go to the <working directory>/example/CGM_sensor folder of the Atmosic Software Development Kit (SDK) and enter the commands in Figure 1-1 to program. All the supported CGM features are enabled by default, and will be introduced in the section Bluetooth Parameters.

```
make clean
```

```
make run_all BOARD=
<ATMEVK_3330_QN|ATMEVK_3330e_QN|ATMEVK_3330e_QN|ATMEVK_3425_DQK_2|ATMEVK_3430_WQK_2|ATMEVK_
3430e_WQN_2>
```

*Figure 1-1 Quick Start Program*

CONFIDENTIAL

```
@00000655 [CGM_sensor][N]: cgm_sensor_init:
@000006fe [CGM_sensor][V]: cgm_ble_init:
@000007a0 [  app_cgms][D]: app_cgms_param
@00000841 [   cgm_gap][V]: cgm_gap_init:
@000008ec rwip_init() done
@00000942 Entering main loop
@00000a1c [ble_gap_se][N]: BOND MASK : 0
@00000b64 [   atm_gap][W]: Unhandled GAPM msg 0xda1
@00000c37 [   atm_gap][W]: Unhandled GAPM msg 0xda1
@00000cf9 [  ble_cgms][D]: ble_cgms_start_op:
@00000daf cgms_init: env (0x20006130)
@00000e31 cgms_init: start handle (0x2e)
@00000eb3 cgms_init: env->prf_task(hx) 0xb
@00000f3c cgms_init: prf_task (hx), app_task (hx), api_id       (hx), prf_id
(hx)
@00001031 [   atm_gap][W]: Unhandled GAPM msg 0xda1
@000010f3 [   cgm_gap][N]: GAP_S_IDLE (GAP_OP_INITED)
@000011d7 lock slots of (hibernation, retention, sleep): 0, 0, 0
@000012a2 [   cgm_gap][W]: cgm_gap_s_init_op_inited
@00001361 [CGM_sensor][V]: cgm_ble_init_done:
@00001411 MEAS: cgms_fake_data_generate
@000014ac [   cgm_gap][W]: cgm_gap_create_adv
@0000155b [   cgm_gap][N]: FW: 1.0.0.0(Sep  1 2022 10:13:29)
@00001652 [   cgm_gap][N]: adv_cfm:(0) state:2 status:0
@00001721 [   cgm_gap][D]: cgm_gap_adv_create_cfm: act_idx=0 status=0
adv_param_type=0
@0000184d [   atm_adv][D]: no path to update vaid
@0000191d [   cgm_gap][N]: adv_cfm:(0) state:4 status:0
@000019ec [   cgm_gap][N]: cgm_gap_set_adv_data_cfm(0)
@00001acd [   cgm_gap][N]: adv_cfm:(0) state:6 status:0
@00001b9c [   cgm_gap][W]: cgm_gap_start_adv(0): type: 0
@00001c8e [   atm_adv][D]: Adv0: ON
@00001d22 [   cgm_gap][N]: adv_cfm:(0) state:9 status:0
@00001df0 [   cgm_gap][N]: cgm_gap_adv_start_cfm(0): status = 0
@00001ed6 [   cgm_gap][N]: GAP_S_ADV0ING (GAP_OP_ADV0ING)
@00001fc5 lock slots of (hibernation, retention, sleep): 0x1, 0, 0
```

*Figure 1-2 CGM_sensor Example Initial Console Message*

- Using 3rd party BLE Mobile apps to find a connectable advertisement with the device name "**Atm_CGM**" and connect to it.

- If the device is not the bonded device, then there is a pop-up window asking for the passkey once connected (or request any read/ write after connected), enter the passkey which is shown over the UART log.
- After pairing successfully,all the  functions are enabled. For more information, refer to Application Demonstration.

## 1.2 Hierarchy and Files

The example was designed to allow users to easily adapt to their final products. Figure 1-3 shows the hierarchy of modules in  the CGM_sensor example.

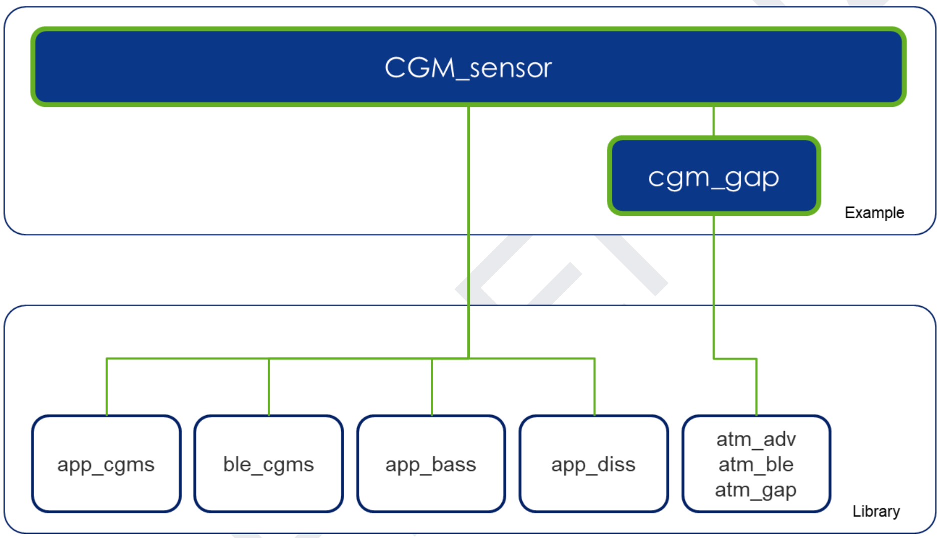


*Figure 1-3 CGM_sensor Example Hierarchy*

Table 1-2 provides description of the modules' functionality.

Table 1-2 Modules' Functionality

| Directory | File Name | Description |
|---|---|---|
| src/ | `CGM_sensor.c`<br>`CGM_sensor.h` | Interacting with all other modules to maintain the flow |
| | `app_config.h` | Bluetooth application parameters such as DIS, BASS and CGMS related definitions |
| src/bt/ | `cgm_gap.c`<br>`cgm_gap.h` | Agent of GAP. It maintains Bluetooth states and informs CGM_sensor while changed |
| | `cgm_param_gap_adv.h` | Advertisement related parameters |
| | `cgm_param_gap.h` | Connection related parameters |

# 2. State Machines

State machines are the core of the application. Users can easily change and optimize design by understanding the state machines. Refer to CGM_sensor State Descriptions and GAP State Descriptions sections. These two sections describe the state transition rules.

## 2.1 CGM_sensor State Descriptions

There are 5 states and 6 operations are defined in the `CGM_sensor.h`. Table 2-1 lists all these states, and Table 2-2 lists these operations.

Table 2-1 CGM_sensor States

| States | Description |
|---|---|
| CGM_S_INIT | Initial state. |
| CGM_S_INITTING | Initial is ongoing |
| CGM_S_IDLE | No ongoing Bluetooth or other operation activity. Enter hibernate if no further request. |
| CGM_S_ADV | Advertisement is starting. |
| CGM_S_CONNECTED | Bluetooth connected. |

Table 2-2 CGM_sensor Operations

| Operations | Description |
|---|---|
| CGM_OP_INITING | Initial is ongoing. |
| CGM_OP_INIT_DONE | Initial is done. |
| CGM_OP_ADV | Advertisement is starting. |
| CGM_OP_ADV_STOP | Advertisement is stopped normally. |
| CGM_OP_CONNECTED | Bluetooth connected. |
| CGM_OP_DISCONNECTED | Bluetooth link disconnected. |

CONFIDENTIAL

Figure 2-1 shows the transition paths of cgm_sensor states.

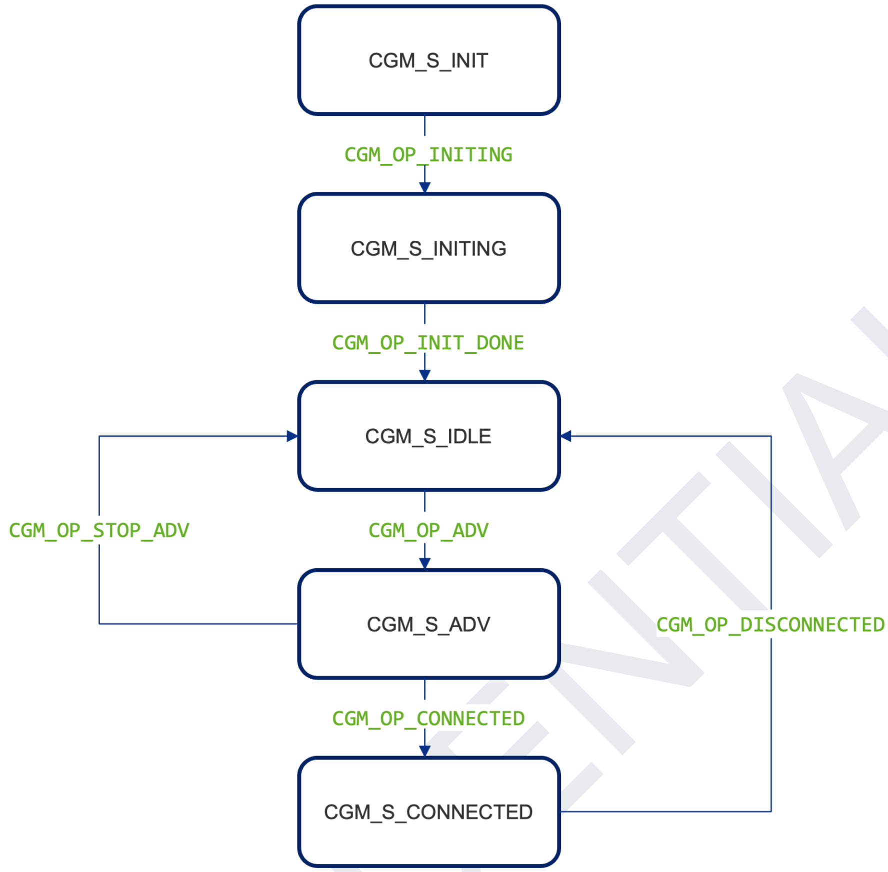


*Figure 2-1 CGM_sensor State Transition*

## 2.2 GAP State Descriptions

The state machine of cgm_gap is created in the application layer to simplify the design. There are 6 states and 9 operations defined in cgm_gap state machines. Table 2-3 describes those states.

Table 2-3 cgm_gap States

| States | Description |
|---|---|
| GAP_S_INIT | Initial state. |
| GAP_S_IDLE | No ongoing Bluetooth activity. Enter hibernate if no further request from CGM_sensor. |
| GAP_S_ADV0ING | Paring advertisement is ongoing. |
| GAP_S_ADV1ING | Reconnecting advertisement is ongoing. |
| GAP_S_ADV_STOPPING | Advertisement is stopping. |

| States | Description |
|---|---|
| GAP_S_CONNECTED | Bluetooth link exists. |

Each operation is triggered by atm_gap modules. Table 2-4 describes the operations and the triggering function.

Table 2-4 cgm_gap Operations

| Operations | Description | Trigger by |
|---|---|---|
| GAP_OP_INITING | Bluetooth is initializing . | `cgm_gap_init()` |
| GAP_OP_INITED | Bluetooth initialized. | `cgm_gap_init_cfm()` |
| GAP_OP_ADV0ING | ADV 0 is started | `cgm_gap_adv_start_cfm()` |
| GAP_OP_ADV1ING | ADV 1 is started | `cgm_gap_adv_start_cfm()` |
| GAP_OP_ADV_STOP | ADV stopped due to connection. | `cgm_gap_adv_stop_ind()` |
| GAP_OP_ADV_STOP_TOUT | ADV stopped due to timeout | `cgm_gap_adv_stop_ind()` |
| GAP_OP_ADV_STOPPING | ADV is stopped by application. | |
| GAP_OP_CONNECTED | Bluetooth is connected. | `cgm_gap_conn_ind()` |
| GAP_OP_DISCONNED | Bluetooth is disconnected | `cgm_gap_disc_ind()` |

CONFIDENTIAL

Figure 2-2 shows the transition paths of cgm_gap states.

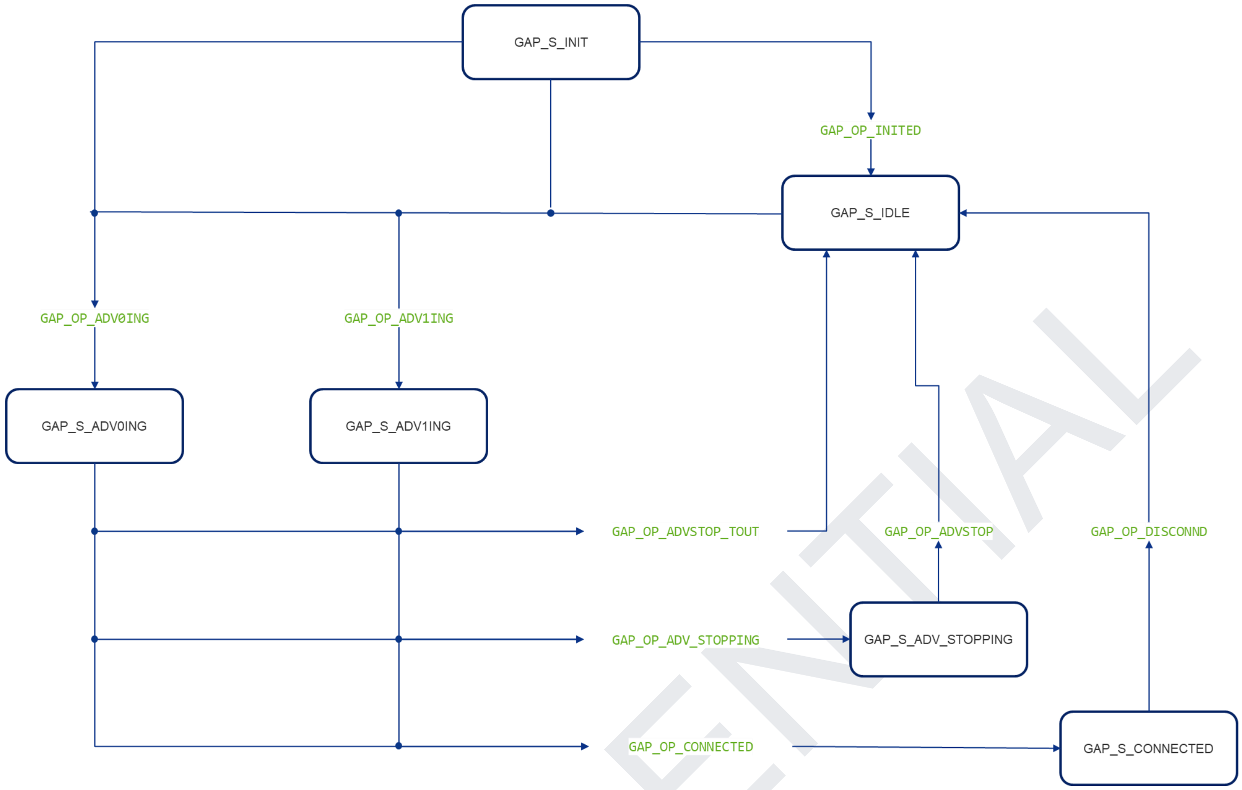


*Figure 2-2 cgm_gap State Transition*

# 3. Power Management

In this example code, the atm_pm module's lock scheme is used to manage whether or not to prevent entering a power state. Refer to SDK Reference Manual for details of the power management states and module details. Below lists all the locks used in this example and descriptions:

| PM_LOCK_HIBERNATION | | |
|---|---|---|
| **Name** | **Modules** | **Description** |
| cgm_gap_lock_hiber | cgm_gap | Locked when cgm_gap state is not GAP_S_IDLE. |

*Table 3-1 Locks used in Power Management*

# 4. Bluetooth Parameters

All the related Bluetooth parameters are defined in `src/app_config.h`, `src/bt/cgms_param_gap_adv.h` and `src/bt/cgms_param_gap.h`.

## 4.1 Timeout Parameters

### 4.1.1 Timeout after initialization done or session start

In the CGM_sensor.c, after the initial process is done or receives the session start command from the peer device, the timer will be set. The `cgm_meas_timer_msg_ind()` handler will be triggered after timeout.
In the `cgm_meas_timer_msg_ind()`, the same timer will be set again, like a periodic timer, and generate a dummy cgm data.

Default timeout value is 1 minute.

```
#define CGMS_MEAS_INTERVAL_CS 6000 // minute
```

Figure 4-1 Timeout after initialization

### 4.1.2 Timeout after session start

In the `app_cgm`, after the session starts the timer will be set. The `app_cgms_session_handler()` will be triggered after timeout, and this timer will be set again. In the `app_cgms_session_handler()`, the variable `time_offset` will be increased one every time. This time_offset is used to be one of the CGM measurement record parameters.

Default timeout value is 1 minute.

```
#define CGMS_SESSION_INTV_CS 6000 // 1 minute, interval for Time Offset
```

*Figure 4-2 Timeout after session start*

### 4.1.3 Timeout after CGM measurement enabled

This section describes timeout after the cgm measurement client characteristic configuration descriptor notification is enabled.

In the `app_cgm`, after the CGM Measurement client characteristic configuration descriptor notification is enabled, the timer will be set. The `app_cgm_meas_send_handler()` will be triggered after timeout, and it calls the `ble_cgms_measurement_send()` API to send cgm measurement data to the peer device.

The default timeout value is followed by the cgm communication interval. The interval can be updated by the peer device. and the default interval is one minute defined in the `src/app_config.h`.

```
#define CFG_CGMS_COM_INTV 1
```

*Figure 4-3 Timeout after CGM measurement enabled*

CONFIDENTIAL

## 4.2 GAP Parameters

### 4.2.1 Advertisement

The advertisement related parameters are defined in `cgms_param_gap_adv.h`.

This example code uses two advertisement sets and the definition is shown below:

```
#define CFG_GAP_ADV_MAX_INST 2
```

*Figure 4-4 Advertisement parameters*

Refer to CFG_ADV0_* for pairing advertisements and CFG_ADV1_* for reconnecting advertisements.

### 4.2.2 Connection Parameters and Negotiation

The connection related parameters are defined in `cgms_param_gap.h`. There are four parameters are related to the connection parameters: CFG_GAP_CONN_INT_MIN, CFG_GAP_CONN_INT_MAX, CFG_GAP_CONN_TIMEOUT, and CFG_GAP_SLAVE_LATENCY.

```
// Minimum value for connection interval
#define CFG_GAP_CONN_INT_MIN 24
// Maximum value for connection interval
#define CFG_GAP_CONN_INT_MAX 200
// Supervision timeout for the LE Link
#define CFG_GAP_CONN_TIMEOUT 1000
// Slave latency for the LE Link
#define CFG_GAP_SLAVE_LATENCY 1
```

*Figure 4-5 Connection Parameters and Negotiation*

These four values would reflect the value of peripheral preferred parameters characteristic in GAP service. Essentially, central will update connection parameters by referring to this characteristic. Except for the updating from central,  peripheral could request itself. Depending on the CFG_SLAVE_PARAM_NEGO compile option,  the device will perform connection parameter update negotiation after connecting with central. In `cgm_gap.c - app_gap_connect_param_nego()`, param_nego is the parameter for connection parameter negotiation. Users can modify the parameter depending on the application.

The `app_gap_connect_param_nego()` and related definitions are defined in `lib/app_gap/app_gap.c`.

```c
#define APP_PARAM_NEGO_RETRY 3
#define APP_PARAM_NEGO_EACH_TIMEOUT 300
#define APP_PARAM_NEGO_DELAY 300 // unit: 10ms

#ifndef CFG_GAP_CONN_INT_MIN // in 1.25 ms
#define APP_GAP_CONN_INT_MIN 9
#else
#define APP_GAP_CONN_INT_MIN CFG_GAP_CONN_INT_MIN
#endif

#ifndef CFG_GAP_CONN_INT_MAX // in 1.25 ms
#define APP_GAP_CONN_INT_MAX 9
#else
#define APP_GAP_CONN_INT_MAX CFG_GAP_CONN_INT_MAX
#endif

#ifndef CFG_GAP_PERIPH_LATENCY // in number of connection events
#define APP_GAP_PERIPH_LATENCY 29
#else
#define APP_GAP_PERIPH_LATENCY CFG_GAP_PERIPH_LATENCY
#endif

#ifndef CFG_GAP_CONN_TIMEOUT // in the unit of 10ms. Range: 100ms - 32s
#define APP_GAP_CONN_TIMEOUT 500
#else
#define APP_GAP_CONN_TIMEOUT CFG_GAP_CONN_TIMEOUT
#endif
static atm_gap_param_nego_t const param_nego = {
    .param_nego_cfm = app_param_nego_cfm,
    .delay = APP_PARAM_NEGO_DELAY,
    .force_retry = false,
    .retry_times = APP_PARAM_NEGO_RETRY,
    .check_result = APP_PARAM_NEGO_EACH_TIMEOUT,
    .target = &(ble_gap_conn_param_t const) {
        .intv_min = APP_GAP_CONN_INT_MIN,
        .intv_max = APP_GAP_CONN_INT_MAX,
        .latency = APP_GAP_PERIPH_LATENCY,
        .time_out = APP_GAP_CONN_TIMEOUT,
    }
};
```

*Figure 4-6 List of Definitions*

## 4.3 Device Information Service Parameters

The strings of device information service such as manufacture name, model name, firmware revision, software revision, etc,.. are defined with APP_DIS_*. They can be modified by users if needed in `app_config.h`.

```
#define APP_DIS_MANUFACTURER_NAME "Atmosic Tech."
#define APP_DIS_MODEL_NB_STR "CGM Sensor"
#define APP_DIS_SERIAL_NB_STR "1.0.0.0"
#define APP_DIS_FIRM_REV_STR "1.0.0.0"
#define APP_DIS_HARD_REV_STR "1.0.0"
#define APP_DIS_SW_REV_STR APP_VERSION
#define APP_DIS_SYSTEM_ID "\x12\x34\x56\xFF\xFE\x9A\xBC\xDE"
#define APP_DIS_IEEE "\xFF\xEE\xDD\xCC\xBB\xAA"
#define APP_DIS_PNP_ID "\x01\x45\x75\x21\x00\x10\x01"
```

*Figure 4-7 Device Information Service Parameters*

## 4.4 Continuous Glucose Monitoring Service (CGMS) Parameters

The parameters of continuous glucose monitoring service (CGMS) such as CGM feature, CGM measurement flag, CGM default communication interval, etc,.. are defined with CFG_CGMS_*. They can be modified by users if needed in `app_config.h`.

CONFIDENTIAL

```
#define CFG_CGMS_MEAS_FLAG (CGMS_MEAS_SENSOR_ANNU_WARN_PRESENT | \
CGMS_MEAS_SENSOR_ANNU_STS_PRESENT)

#define CFG_CGMS_FEATURE (CGMS_FEAT_SENSOR_MALFUNC_DET_SUPP | \
CGMS_FEAT_SENSOR_RESULT_DET_SUPP | CGMS_FEAT_SENSOR_TYPE_DET_SUPP | \
CGMS_FEAT_MULTI_BOND_SUPP | CGMS_FEAT_MULTI_SESS_SUPP | \
CGMS_FEAT_CGM_TREND_INFO_SUPP | CGMS_FEAT_CGM_QUALITY_SUPP)

#define CFG_CGMS_SENSOR_TYPE_LOCATION (CGMS_TYPE_ISF |
CGMS_LOC_SUBCUTANEOUS)

// Supports the fattest communication interval in minutes
#define CFG_CGMS_MAX_COM_INTV 1

// Default communication interval
#define CFG_CGMS_COM_INTV 1
```

*Figure 4-8 CGMS Parameters*

## 4.5 Record Access Control Point (RACP) Parameters

The records of continuous glucose monitoring service (CGMS) are stored in RRAM or flash and the maximum size of records are defined in `app_cgms.h`. Users could retrieve CGM records by RACP service.

Define the maximum size of user RRAM is in `makefile`.

```
# User Data: stored RACP records
USER_SIZE := 0x1000
CFLAGS += \
    -DCFG_BLE_RACP_DATA_SIZE=$(USER_SIZE) \
    -DCFG_EMU_WVT \
```

*Figure 4-9 RACP Parameters*

The maximum size of RACP records are defined in `app_cgms.h`.

```
/// CGMS default the maximum count of CGM RACP records
#ifndef CFG_CGMS_RACP_CNT_MAX
#define APP_CGMS_RACP_CNT_MAX 240
#else
#define APP_CGMS_RACP_CNT_MAX CFG_CGMS_RACP_CNT_MAX
#endif
```

*Figure 4-10 Maximum count RACP records*

CONFIDENTIAL

# 5. Application API

This section will introduce the application API that can be used to get the cgms state or update the measured data in `CGM_sensor.c`. About the code flow, the user can referer to the  Application State Machine Application Note in the customer portal.

## 5.1 cgms_fake_data_generate

This API is used to generate the fake cgm data and load it into the app_cgm library. It is called from `cgm_meas_enable()`. The user can modify and update cgm measurement data here. The function name also can be renamed.

## 5.2 cgm_meas_enable

This API is called when the initialization is done or a new session starts. And it is  used to generate a fake cgm data and start a timer to periodically do the same thing. This timer is introduced at Timeout after initialization done or session start and the timer handler `cgms_meas_timer_msg_ind()` can reference the next section.

## 5.3 cgms_meas_timer_msg_ind

This is the timer handler and the Timeout after initialization done or session start explains the trigger time.
It used to call the `cgms_fake_data_generate()`.

## 5.4 app_cgms_start_session_handler

This is a callback function that is registered with the app_cgms module, and it is called after receiving the peer device start session command.

## 5.5 app_cgms_stop_session_handler

This is a callback function that is registered with the app_cgms module, and it is called after receiving the peer device stop session command.

CONFIDENTIAL

## 5.6 app_cgms_session_run_time_handler

This is a callback function that is registered with the app_cgms module, and it is called after receiving the peer device read session read time command. For the session run time, reference the CGMS_1.0.2 or later specification.

CONFIDENTIAL

# 6. Application Demonstration

This example showcases the Continuous Glucose Monitoring (CGM) profile as a server role. This CGM sensor is meant to pair with a BLE CGM collector. Once connected and the Client Characteristic Configuration descriptor is enabled, the collector will receive the fake CGM measurement data from the device periodically.

Follow the following steps to test the application:

1) Using the client (smartphone app or PC tool) to find a connectable advertisement with the device name "**Atm_CGM**" and connect to it.
2) If the device is not the bonded device, then there is a pop-up window asking for the passkey once connected (or request any read/ write after connected), enter the passkey which is shown over the UART log below.

   Example: Enter "*034105*" to the popup window.

```
@00054b34 [   cgm_gap][V]: cgm_gap_pair_req_ind
@00054bfe [   app_gap][N]: app_pair_rsp: MITM
@00054ca7 [   app_gap][N]: app_pair_rsp: Bonding
@00054d57 [   app_gap][N]: app_pair_rsp: Both using Bonding
@00054e3d PASSKEY: 034105
```

*Figure 6-1 Test Application*

The log shows "app_gap_pair_ind: Pair success" if bonded successfully, otherwise the link is disconnected.

```
@00054e3d PASSKEY: 034105
@0006433a [   atm_gap][V]: target (200 29)
@000643de [   atm_gap][V]: Param NOT good.(24, 0)
@000644b8 [   cgm_gap][N]: param nego = 0, 0X12
@0007c56d [   atm_gap][V]: target (200 29)
@0007c611 [   atm_gap][V]: Param NOT good.(24, 0)
@00122eec [   cgm_gap][V]: cgm_gap_pair_ind
@00122f92 [   app_gap][N]: app_gap_pair_ind: Pair success
```

*Figure 6-2 Successful log*

3) Get CGM related information by reading the defined characteristics.
    - CGM Feature (UUID is 2AA8)
    - CGMS Status (UUID is 2AA9)
    - CGM Session Run Time (UUID is 2AAB)
4) Configure CGM Session Start Time by writing CGM Session Start Time characteristic which UUID is 2AAA. The CGM session start time could be read as well.

    Example:

    - Data time: 2022/07/05 09:05:00, UTC 0, DST-Offset: Standard Time.
    - Write 'E6 07 07 05 09 05 00 00 00' to CGM Session Start Time in byte array.

    The current time offset and the updated CGM session start time are shown over the UART log.

```
@0049cd7a [   app_cgms][D]: app_cgms_start_time_write
@0049ce3c Offset: 2
@0049ce96 StartTime: 7/5/2022 9:5:0
@0049cf24 Updated: 7/5/2022 9:3:0
```

*Figure 6-3 UART Log*

5) Enable Client Characteristic Configuration (CCCD) of CGM Measurement which UUID is 2AA7 to receive periodic CGM measurement notification every one minute.

6) Test Specific Operations Control Point (SOCP) related functions. Enable CCCD of SOCP which UUID is 2AAC to receive the SOCP indications.
    - Set Communication Interval Configuration

        a. Write ‘0x01 0x02’ to SOCP to set the communication interval to 2 minutes.
        b. Observe the indication response data.
        '1C 01 01': Response code, Set communication interval, Success.

- Get Communication Interval Configuration

    a. Write '0x02' to SOCP.
    b. Observe the indication response data.
    '03 xx': Communication interval response, the communication interval is xx in minutes.
- Start CGM Session: (UUID 0x2AAC)
    a. Write '0x1A' to SOCP.
    *NOTE: The sensor deletes all CGM measurement databases and clears Session Start Time characteristic value.*
    b. Observe the indication response data.
    '1C 1A 01': Response code, Session start, Success
    c. Observe the notification (2AA7) to receive the fake CGM measurement data.
- Stop CGM Session: (UUID 0x2AAC)
    a. Write '0x1B' to SOCP.
    b. Observe the indication response data.
    '1C 1B 01': Response code, Session stop, Success
    c. Measurement is stopped.

CONFIDENTIAL

# Revision History

| Date | Version | Description |
| --- | --- | --- |
| January 26, 2024 | 0.1 | Initial version created. |

CONFIDENTIAL

ATMOSIC TECHNOLOGIES – DISCLAIMER

This product document is intended to be a general informational aid and not a substitute for any literature or labeling accompanying your purchase of the Atmosic product. Atmosic reserves the right to amend its product literature at any time without notice and for any reason, including to improve product design or function. While Atmosic strives to make its documents accurate and current, Atmosic makes no warranty or representation that the information contained in this document is completely accurate, and Atmosic hereby disclaims (i) any and all liability for any errors or inaccuracies contained in any document or in any other product literature and any damages or lost profits resulting therefrom; (ii) any and all liability and responsibility for any action you take or fail to take based on the information contained in this document; and (iii) any and all implied warranties which may attach to this document, including warranties of fitness for particular purpose, non-infringement and merchantability. Consequently, you assume all risk in your use of this document, the Atmosic product, and in any action you take or fail to take based upon the information in this document. Any statements in this document in regard to the suitability of an Atmosic product for certain types of applications are based on Atmosic's general knowledge of typical requirements in generic applications and are not binding statements about the suitability of Atmosic products for any particular application. It is your responsibility as the customer to validate that a particular Atmosic product is suitable for use in a particular application. All content in this document is proprietary, copyrighted, and owned or licensed by Atmosic, and any unauthorized use of content or trademarks contained herein is strictly prohibited.

Copyright ©2022 by Atmosic Technologies. All rights reserved. Atmosic logo is a registered trademark of Atmosic Technologies Inc. All other trademarks are the properties of their respective holders.

Atmosic Technologies | 2105 S. Bascom Ave. | Campbell CA, 95008
**www.atmosic.com**