10. Self-diagnostic function
10.1. Self-diagnostic function

10.1.1. Self-diagnostic function summary

The BT-2000 is equipped with a self-diagnostic function to check the status of internal devices. You can check the following devices and content using the self-diagnostic function.

Table 10-1 Confirmation items using the self-diagnostic function

<table>
<thead>
<tr>
<th>Device type</th>
<th>Display name</th>
<th>Diagnostic method</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>CPU</td>
<td>Error check</td>
</tr>
<tr>
<td>Power control IC</td>
<td>PMIC</td>
<td>Check power supply status</td>
</tr>
<tr>
<td>Main memory (RAM)</td>
<td>DDR</td>
<td>Check RAM data</td>
</tr>
<tr>
<td>Data memory (Flash)</td>
<td>eMMC</td>
<td>Reading and writing to data region</td>
</tr>
<tr>
<td>Wireless LAN module</td>
<td>WLAN</td>
<td>Check communication with device</td>
</tr>
<tr>
<td>Bluetooth 3.0</td>
<td>Bluetooth</td>
<td>Check communication with device</td>
</tr>
<tr>
<td>Bluetooth Low Energy</td>
<td>BLE</td>
<td>Check communication with device</td>
</tr>
<tr>
<td>Audio IC</td>
<td>Audio IC</td>
<td>Reading device information characteristics</td>
</tr>
<tr>
<td>Audio commands IC</td>
<td>Voice Command</td>
<td>Reading device information characteristics</td>
</tr>
<tr>
<td>Sensor</td>
<td>Sensor MCU</td>
<td>Communication check</td>
</tr>
<tr>
<td>IMU sensor</td>
<td>IMU</td>
<td>Acquire sensor data</td>
</tr>
<tr>
<td>Sensor Illumination</td>
<td>Sensor Illumination</td>
<td>Acquire sensor data</td>
</tr>
<tr>
<td>Charger IC A</td>
<td>Charger IC A</td>
<td>Check status</td>
</tr>
<tr>
<td>Charger IC B</td>
<td>Charger IC B</td>
<td>Check status</td>
</tr>
<tr>
<td>Battery A</td>
<td>Battery A</td>
<td>Check status</td>
</tr>
<tr>
<td>Battery B</td>
<td>Battery B</td>
<td>Check status</td>
</tr>
</tbody>
</table>

The self-diagnostic function is a basic diagnostic tool. Be aware that it may not be able to locate faults in individual functions in the device. And some functions are not available.
10.1.2. Using the self-diagnostic function
The self-diagnostic function is provided as an app. Start "SelfDiagnosis" from the app list.

When the app starts, click the "Start" button at the top of the screen to start the diagnostic procedure.
When the diagnosis has been performed, the results for each device are displayed. See table 10-1 above for details about the target items and the checking method.

![Self Diagnosis check results](image)

Figure 10-3  SelfDiagnosis check results
11. GPS Assist
11.1. GPS Assist function summary

BT-2000 has GPS assist function to reduce initial positioning time (Time To First Fix: TTFF) of cold start when it takes long GPS positioning time. By downloading GPS satellite orbit prediction data in advance and write in the BT-2000, it will reduce download time of satellite data and reduce TTFF as a result.

Figure 11-1  EPO Data download image
11.2. Precaution and limitation of using /developing GPS Assist function

■ GPS Assist function Effect
To fully demonstrate GPS Assist function effect, it is required to capture 4 GPS satellites. There are some cases that cannot fully demonstrate the effect under condition that cannot capture or hard to capture the GPS satellites.

■ GPS cold start
To start GPS positioning, there are hot start using satellite data of last time, warm start and cold start when you cannot use positioning data of last time.
GPS assist function is a function to demonstrate effect in cold start. It does not have effect to reduce TTFF in hot start and warm start.

■ Setting time
It uses time information to simulate GPS satellite position. So it is necessary to set the system time accurately.

■ Downloading EPO data
EPO data validity period is 30 days. But as the GPS assist function effect is higher by using new data, so please re-download the EPO data by 1 week as a guide.

■ EPO data delivery server
BT-2000 can download EPO data by internet from EPSON EPO data delivery server.
When you cannot use internet or using GPS assist function by local internet, copy EPO data from EPSON data delivery server and place at local internet.
To set EPO data delivery server in BT-2000, write delivery server URL at "internal storage¥agps¥agps.conf".

■ GPS assist function API
BT-2000 GPS assist function API is using EPSON own API.
To develop an App using GPS assist function, install SDK provided by EPSON and use EPSON own API.
11.3. How to use

This section describes how to use GPS assist function.

11.3.1. Downloading EPO data

- Turn Wi-Fi function ON to access to EPO delivery server.
  Internet environment is required to access to EPSON EPO data delivery server. When placing EPO data to intranet, connection to intranet is required.

- Open [Setting] App and select [Position information service].
  ![Settings App](image)
  
  - Turn [GPS function] ON and select [AGPS data update].
  ![GPS satellites](image)

11.3.2. Writing EPO data

While writing EPO data to BT-2000, the icon  will be displayed in system bar. Writing is completed after the icon  disappears.

11.3.3. Using EPO data

- Start App using GPS. GPS assist functions automatically operate at cold start.
11.4. Setting method

This section describes about setting for writing EPO data.

Setting file will be located at “internal storage\agps\agps.conf”. Locate created file according to below list.

11.4.1. Setting file agps.conf

File path: /mnt/sdcard/agps/agps.conf
Default: None It is generated with the default value when [Settings]-[AGPS data update] is executed.

File format: Conform to notation of java.util.Properties
Use “\:" to write colon

<table>
<thead>
<tr>
<th>No</th>
<th>Item</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>EPO_URL</td>
<td>Set URL of EPO data delivery server. Default value is “<a href="https://tech.moverio.epson.com/epo/data.bin%E2%80%9D">https://tech.moverio.epson.com/epo/data.bin”</a></td>
</tr>
<tr>
<td>2</td>
<td>EPO_NUM_SEGS</td>
<td>Set EPO writing segment number (1 ~ 28). 1 segment = 6 hours. Default value is 28(6 hours × 28 = 7 days)</td>
</tr>
<tr>
<td>3</td>
<td>LAST_MOD</td>
<td>Shows download time of last time. Not need for setting.</td>
</tr>
</tbody>
</table>
11.5. Application interface function summary

This section describes API of EPO data download and writing to BT-2000.

1) Download EPO data from EPO data delivery server
2) Write downloaded EPO data to BT-2000 GPS module

To use each API, import below module.
com.epson.moverio.bt2000.agpsctl

Table 11-2 Interface list of AGPSControl class

<table>
<thead>
<tr>
<th>NO.</th>
<th>Function name</th>
<th>Function summary</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>downloadEpo</td>
<td>Download EPO data by setting according to agps.conf setting file.</td>
<td>EPSON original API</td>
</tr>
<tr>
<td>2</td>
<td>downloadEpo_url</td>
<td>Download EPO data from URL specified by argument</td>
<td>EPSON original API</td>
</tr>
<tr>
<td>3</td>
<td>writeEpo</td>
<td>Write EPO data to BT-2000 by setting according to agps.conf setting file.</td>
<td>EPSON original API</td>
</tr>
<tr>
<td>4</td>
<td>writeEpo_segs</td>
<td>Write EPO data of specified segment by argument to BT-2000</td>
<td>EPSON original API</td>
</tr>
</tbody>
</table>
11.6. Application interface detail

11.6.1. downloadEpo

- **Function**
  
  Download EPO data according to setting written in Setting file (/sdcatd/agps.conf).
  
  EPO data will be delivered from EPSON data delivery server (Update every day at 0:10 UTC)

- **Format**
  
  ```
  int downloadEpo(void);
  ```

- **Parameter**
  
  None

- **Return value**
  
  Always 0

- **Broadcast Intent**
  
<table>
<thead>
<tr>
<th>Intent</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGPS_DOWNLOAD_COMPLETED</td>
<td>download completed</td>
</tr>
<tr>
<td>AGPS_DOWNLOAD_ERROR</td>
<td>download error</td>
</tr>
<tr>
<td>AGPS_CONF_ERROR</td>
<td>agps.conf error (agps.conf not exist, or defect in EPO_URL)</td>
</tr>
<tr>
<td>AGPS_NETWORK_ERROR</td>
<td>Error by Wi-Fi OFF</td>
</tr>
<tr>
<td>AGPS_FILE_NOT_UPDATED</td>
<td>No update in server file</td>
</tr>
</tbody>
</table>

11.6.2. downloadEpo_url

- **Function**
  
  Download EPO data from the URL of parameter).

- **Format**
  
  ```
  int downloadEpo(String arg0);
  ```

- **Parameter**
  
<table>
<thead>
<tr>
<th>Type name</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>String arg0</td>
<td>URL of EPO data delivery server</td>
</tr>
</tbody>
</table>

- **Return value**
  
  Always 0

- **Broadcast Intent**
  
  Same as downloadEpo
11.6.2.1. writeEpo
- Function
  Write EPO data to BT-2000 GPS module by agps.conf setting.

- Format
  int writeEpo();

- Parameter
  None

- Return value
  Always 0

- Broadcast Intent

<table>
<thead>
<tr>
<th>Intent</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGPS_WRITE_COMPLETED</td>
<td>write the data completed</td>
</tr>
<tr>
<td>AGPS_WRITE_ERROR</td>
<td>Write the data error</td>
</tr>
<tr>
<td>AGPS_CONF_ERROR</td>
<td>agps.conf error (No agps.conf, or the number of EPO_NUM_DAYS is not correct)</td>
</tr>
<tr>
<td>AGPS_FILE_EXPIRED</td>
<td>EPO file is old. (Expired at the present time, there is no data for the specified period.)</td>
</tr>
</tbody>
</table>

11.6.2.2. writeEPO_segs
- Function
  Writes EPO data for the number of segments specified by the argument to the GPS module of BT-2000.
  One segment means the 6 hours.

- Format
  int writeEPO_segs(int arg0);

- Parameter

<table>
<thead>
<tr>
<th>Type name</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>int arg0</td>
<td>Writing segment number</td>
</tr>
</tbody>
</table>

- Broadcast Intent
  Same as writeEPO
11.7. How to use API for GPS assist
11.7.1. Download of EPO data
Create the instance of AGPSControl and call API for download.

■ Implementation example for download

```java
public void execDownload(View view) {
    Log.d(TAG, "exec Download");

    try {
        AGPSControl ctl = new AGPSControl();
        ctl.downloadEpo();
    } catch (Exception ex) {
        ex.printStackTrace();
    }
}
```
11.7.2. Intent reception method

As download and writing process are done asynchronously, return value cannot be referred directly.

So, receive the process result by using Broadcast Intent.

For each Broadcast Intent value issued by each API, refer to the item of Broadcast intent of each API.

This section describes about the intent reception method.

11.7.2.1. At start

Set intent filter and receiver at the start of application.

■ Implementation example

```java
private MyReceiver receiver = new MyReceiver();
private IntentFilter filter = new IntentFilter();

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    filter.addAction(AGPSControl.AGPS_CONF_ERROR);
    filter.addAction(AGPSControl.AGPS_DOWNLOAD_COMPLETED);
    filter.addAction(AGPSControl.AGPS_DOWNLOAD_ERROR);
    filter.addAction(AGPSControl.AGPS_FILE_EXPIRED);
    filter.addAction(AGPSControl.AGPS_FILE_NOT_UPDATED);
    filter.addAction(AGPSControl.AGPS_NETWORK_ERROR);
    filter.addAction(AGPSControl.AGPS_WRITE_COMPLETED);
    filter.addAction(AGPSControl.AGPS_WRITE_ERROR);

    getApplicationContext().registerReceiver(receiver, filter);
}
```
Intent receiver example

```java
class MyReceiver extends BroadcastReceiver {
    public MyReceiver() {
        super();
    }

    @Override
    public void onReceive(Context context, Intent intent) {
        if (intent.getAction().equals(AGPSControl.AGPS_CONF_ERROR)) {
            Toast.makeText(getApplicationContext(), "AGPS_CONF_ERROR",
            Toast.LENGTH_LONG).show();
            Log.d(TAG, "Intent Received | AGPS Conf Error");
        }
        if (intent.getAction().equals(AGPSControl.AGPS_DOWNLOAD_COMPLETED)) {
            Toast.makeText(getApplicationContext(), "AGPS_DOWNLOAD_COMPLETED",
            Toast.LENGTH_LONG).show();
            Log.d(TAG, "Intent Received | AGPS Download Completed");
        }
        ~~~~(Omit other intent)~~~~
    }
}
```

11.7.2.2. Re-setting of receiver due to application situation transition
Cancel intent receiver when application situation transit like becoming background.

Receiver cancellation example

```java
@Override
protected void onPause() {
    super.onPause();
    getApplicationContext().unregisterReceiver(receiver);
}
```

When returning to application again, re-register the intent receiver.

Re-register example

```java
@Override
protected void onResume() {
    super.onResume();
    getApplicationContext().registerReceiver(receiver, filter);
}
```