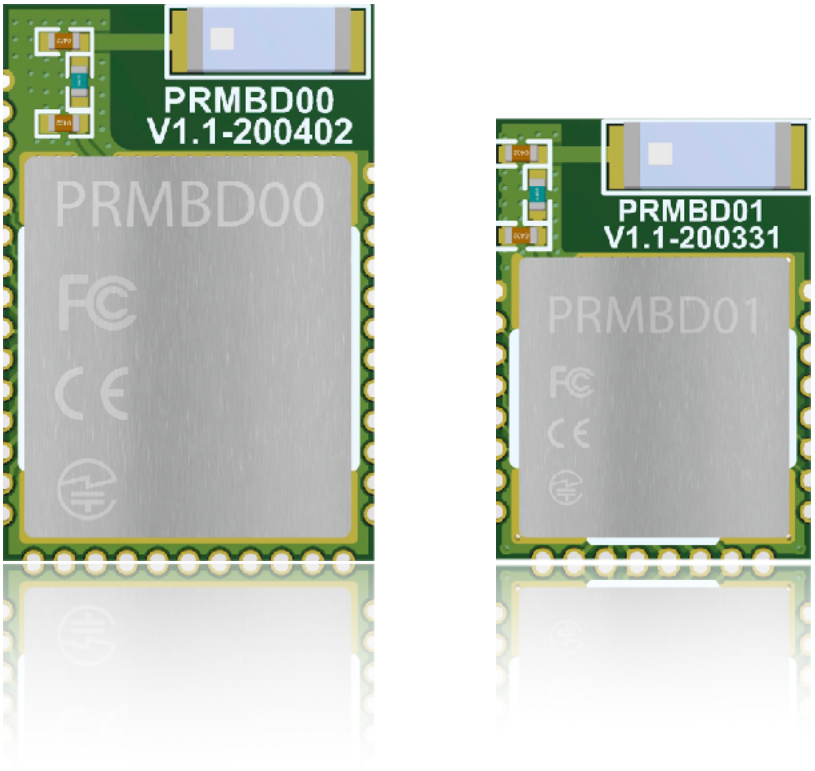
PRBMD0x Application Note

Apple Notification Center Services



Disclaimer

Liability Disclaimer

K-Solution Consulting Co. Ltd reserves the right to make changes without further notice to the product to improve reliability, function or design. K-Solution Consulting Co. Ltd does not assume any liability arising out of the application or use of any product or circuits described herein.

Life Support Applications

K-Solution Consulting Co. Ltd's products are not designed for use in life support appliances, devices, or systems where malfunction of these products can reasonably be expected to result in personal injury. K-Solution Consulting Co. Ltd customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify K-Solution Consulting Co. Ltd for any damages resulting from such improper use or sale.

1. 简介

This document describe the implement of ANCS (Apple Notification Center Service*) on PRBMD0x's SDK.

ANCS is a sub-set of GATT, provide a simple, convenient way to obtain notification from iOS* device.

PRBMD0x's SDK provide an example source code for implement ANCS. The example locates at the following directory:

ANCS Profile	Trunk\components\profiles\ancs
example	Trunk\example\ble_peripheral\ancs

*Apple and ANCS is trade mark of Apple Inc.

2. ANCS Profile

ANCS profile is implement base on Service discovery under GATT Client and data exchange of ANCS application level. ANCS provide a group of API and a notification interface for pushing notification.

2.1. bStatus_t ble_ancs_init(ancs_evt_hdl_t evt_hdl, uint8_t task_ID)

ANCS profile initialisation. Application can initialise and register the notification function.

- parameter

Type	Parameter	Description
ancs_evt_hdl_t	evt_hdl	ANCS profile return function, for pushing notification after registration.
uint8_t	task_ID	OSAL task ID。

- Return value

SUCCESS	initialisation success
other value	refers to <comdef.h>

2.2. bStatus_t ble_ancs_attr_add(const ancs_notif_attr_id id, uint8_t * p_data, const uint16_t len)

ANCS attribute configuration. It need to be configured before ANCS Discovery，and after implementing ble_ancs_init(), 0~7 are standard notification attribute, 8~255 are reserved value. Details are illustrated as below：

NotificationAttributeIDAppIdentifier	= 0,
NotificationAttributeIDTitle	= 1, (Needs to be followed by a 2-bytes max length parameter)
NotificationAttributeIDSubtitle	= 2, (Needs to be followed by a 2-bytes max length parameter)
NotificationAttributeIDMessage	= 3, (Needs to be followed by a 2-bytes max length parameter)
NotificationAttributeIDMessageSize	= 4,
NotificationAttributeIDDate	= 5,
NotificationAttributeIDPositiveActionLabel	= 6,
NotificationAttributeIDNegativeActionLabel	= 7,
Reserved NotificationAttributeID values	= 8-255

Notification attribute will be valid only after configuration. When application call ble_ancs_get_notif_attrs() to obtain notification attribute, notification event will return configured notification parameter value.

- parameter

type	Parameter	Description
const ancs_notif_attr	Id	GPIO pin。

uint8_t *	p_data	Allocated memory, The allocated memory is used to globally store the parameter values of the corresponding attributes（usually UTF-8 string）。
const uint16_t	Len	size of p_data

- return value

SUCCESS	initialisation success
other value	refers to <comdef.h>

2.3. bStatus_t ble_ancs_get_notif_attrs(const uint8_t * pNotificationUID)

After receiving ANCS Notify message, Application can ask for notification attribute through Notification UID. After calling this function, valid requested data will be returned in notification format.

- Parameter

type	Parameter	description
const uint8_t *	pNotificationUID	Notification UID，this ID will be in the UTF-8 string format, and end with 0

- return value

SUCCESS	initialisation success
other value	refers to <comdef.h>

2.4. bStatus_t ble_ancs_get_app_attrs(const uint8_t * p_app_id, uint8_t app_id_len)

If the “attr_id” in return notification attribute is BLE_ANCS_NOTIF_ATTR_ID_APP_IDENTIFIER, then it is possible to request APP attribute content by calling this function.

- Parameter

Type	Parameter	description
const uint8_t *	p_app_id	App ID，from notification attribute return event
uint8_t	app_id_len	App ID length

- Return value

SUCCESS	initialisation success
other value	refers to <comdef.h>

2.5. bStatus_t ble_ancs_start_descovery(uint16_t conn_handle)

After ANCS service from Discovery host started, this function can be execute after SMP established. After Discovery completed, BLE_ANCS_EVT_DISCOVERY_COMPLETE event will be returned; if failed, BLE_ANCS_EVT_DISCOVERY_FAILED will be returned.

- Parameter

Type	Parameter	description
uint16	conn_handle	Connection handle

- Return value

SUCCESS	initialisation success。
other value	refers to <comdef.h>

2.6. bStatus_t ble_ancs_handle_gatt_event(gattMsgEvent_t* pMsg)

Responds to GATT event. ANCS profile needs to responds to host GATT event under ANCS service. Function will ignore any un-related event and will keep input data remain unchanged.

- Parameter

Type	Parameter名	description
gattMsgEvent_t*	pMsg	GATT event

- Return value

SUCCESS	initialisation success。
other value	refers to <comdef.h>

3. ANCS application example

The following description is base on the example in SDK (\example\ble_peripheral\ancs)

3.1. OSAL task initialisation

Refer to the code in **bold** in the text box below, the initialisation process requires：

- Call ancs initialization function
- Call the function ble_ancs_attr_add () to add the attributes required for configuration notification.

```
void AncsApp_init( uint8 task_id )
{
  AncsApp_TaskID = task_id;

  //OSAL Task 初始化配置

  // Initialize GATT attributes
  GGS_AddService(GATT_ALL_SERVICES);          // GAP GATT Service
  GATTServApp_AddService(GATT_ALL_SERVICES);  // GATT Service
  DevInfo_AddService();                       // Device Information Service

// For ANCS, the device must register an a GATT client, whereas the
// iPhone acts as a GATT server.
ble_ancs_init(on_ancs_evt, AncsApp_TaskID);
ble_ancs_attr_add(BLE_ANCS_NOTIF_ATTR_ID_APP_IDENTIFIER,m_attr_appid,ANCS_ATTR_DATA
_MAX);
//
ble_ancs_attr_add(BLE_ANCS_APP_ATTR_ID_DISPLAY_NAME,m_attr_disp_name,sizeof(m_attr_d
isp_name));
ble_ancs_attr_add(BLE_ANCS_NOTIF_ATTR_ID_TITLE,m_attr_title,ANCS_ATTR_DATA_MAX);
ble_ancs_attr_add(BLE_ANCS_NOTIF_ATTR_ID_MESSAGE,m_attr_message,ANCS_ATTR_DATA_M
AX);
ble_ancs_attr_add(BLE_ANCS_NOTIF_ATTR_ID_SUBTITLE,m_attr_subtitle,ANCS_ATTR_DATA_MA
X);
ble_ancs_attr_add(BLE_ANCS_NOTIF_ATTR_ID_MESSAGE_SIZE,m_attr_message_size,ANCS_ATTR
_DATA_MAX);
ble_ancs_attr_add(BLE_ANCS_NOTIF_ATTR_ID_DATE,m_attr_date,ANCS_ATTR_DATA_MAX);
ble_ancs_attr_add(BLE_ANCS_NOTIF_ATTR_ID_POSITIVE_ACTION_LABEL,m_attr_posaction,ANC
S_ATTR_DATA_MAX);
ble_ancs_attr_add(BLE_ANCS_NOTIF_ATTR_ID_NEGATIVE_ACTION_LABEL,m_attr_negaction,ANC
S_ATTR_DATA_MAX);
osal_set_event( AncsApp_TaskID, START_DEVICE_EVT );
}
```

3.2. Starting Discovery

After the device is connected to the iOS device, if the SMP process has been completed, you can call the function ble_ancs_start_descovery () to start Discovery, refer to the **bold** text in the following text box to call.

```
static void AncsApp_processPairState(uint8_t state, uint8_t status)
{
 if (state == GAPBOND_PAIRING_STATE_STARTED){
  LOG("Pairing started\n");
 }
 else if (state == GAPBOND_PAIRING_STATE_COMPLETE){
  if (status == SUCCESS){
   LOG("Pairing Successful\n");
   // Now that the device has successfully paired to the iPhone,
   // the subscription will not fail due to insufficient authentication.
   ble_ancs_start_descovery(gapConnHandle);
  }
  else{
   LOG("Pairing fail: %d\n", status);
  }
 }
 else if (state == GAPBOND_PAIRING_STATE_BONDED){
  if (status == SUCCESS){
   LOG("Bonding Successful\n");
   ble_ancs_start_descovery(gapConnHandle);
  }
 }
}
```

3.3. Responding to GATT event

Please refer to the call of the bold part of the following text box.

```
static uint8_t AncsApp_processGATTMsg(gattMsgEvent_t *pMsg)
{
 ble_ancs_handle_gatt_event(pMsg);

 //ANCS requires authentication, if the NP attempts to read/write chars on the
 //NP without proper authentication, the NP will respond with insufficent_athen
 //error to which we must respond with a slave security request
 //The following code is used for the application itself to respond to GATT events, skipped here
 return (TRUE);
}
```

3.4. ANCS application layer event

In this example, the ANCS application layer events are handled by the function on_ancs_evt (ancs_evt_t * p_evt) ◦

4. Reference

This document is base on Application note PHY62XX_ANCS_Application_Note from PHYPlus semiconductors.