青风带你玩蓝牙 nRF52832 系列教程

----------作者：青风

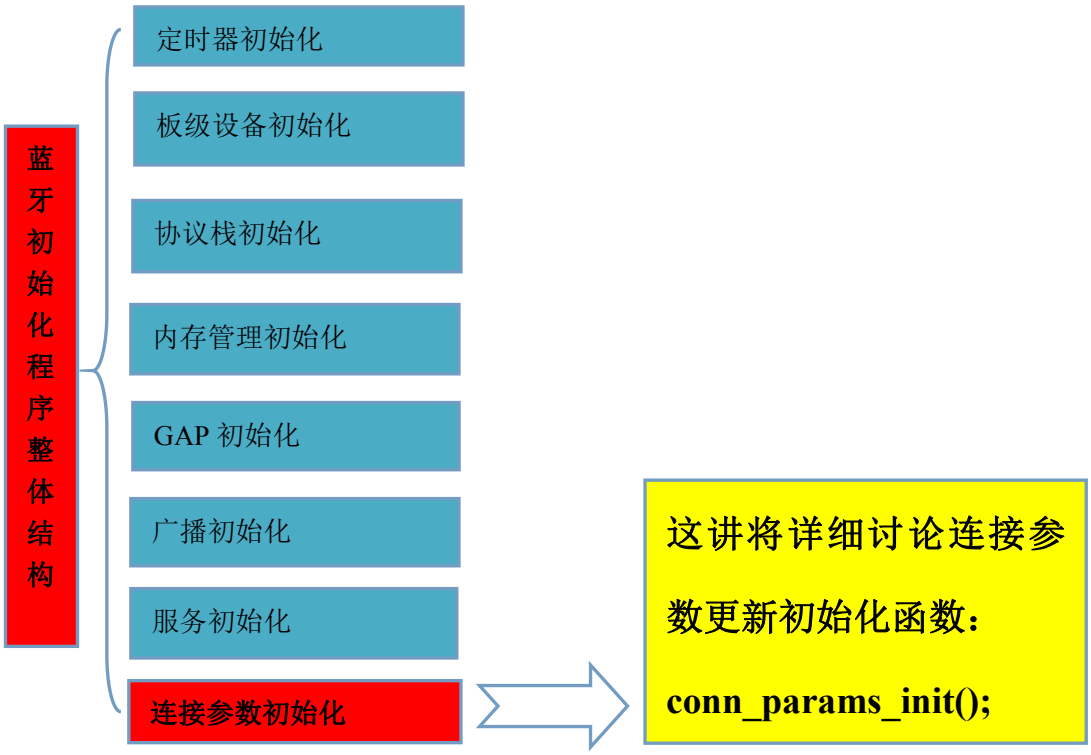
出品论坛：　www.qfv8.com　青风电子社区

作者： 青风
出品论坛: www.qfv8.com
淘宝店：http://qfv5.taobao.com
QQ 技术群：346518370
硬件平台：青云 QY-nRF52832 开发板

2.8 蓝牙链接参数更新详解

在链路层的链接配置过程完成后，GAP 设置中的链接参数 connInterval(连接事件间隔），connSlaceLatency（从机延迟）和 connTimeout(监督超时）参数可以实现更新。关于这三个参数的定义请详细看《青风带你学蓝牙：蓝牙 gap 初步入门》。 并且通过分析基本原理，在匹配的 SDK15.0 的蓝牙样例的例子基础上就行分析与讲解，使用的协议栈为：s132。

```
782  */
783 int main(void)
784 {
785     bool erase_bonds;
786
787     // Initialize.
788     log_init();
789     timers_init();
790     buttons_leds_init(&erase_bonds);
791     power_management_init();
792     ble_stack_init();
793     gap_params_init();
794     gatt_init();
795     advertising_init();
796     services_init();
797     conn_params_init();
798     peer_manager_init();
799
800     // Start execution.
801     NRF_LOG_INFO("Template example started.");
802     application_timers_start();
803
804     advertising_start(erase_bonds);
805
806     // Enter main loop.
807     for (;;)
808     {
809         idle_state_handle();
810     }
811 }
812
813
```



1：nRF52832 蓝牙协议栈初始化函数结构：

在一个 nrf52832 的工程下，蓝牙连接参数更新初始化函数为 conn_params_init()，其函数基本内容如下图所示：

```
01.     static void conn_params_init(void)
02. {
03.     uint32_t               err_code;
04.     ble_conn_params_init_t cp_init;
05.
06.     memset(&cp_init, 0, sizeof(cp_init));
07.   //设置连接参数更新值
08.     cp_init.p_conn_params                  = NULL;
09.     cp_init.first_conn_params_update_delay = FIRST_CONN_PARAMS_UPDATE_DELAY;
10.     cp_init.next_conn_params_update_delay  = NEXT_CONN_PARAMS_UPDATE_DELAY;
11.     cp_init.max_conn_params_update_count   = MAX_CONN_PARAMS_UPDATE_COUNT;
12.     cp_init.start_on_notify_cccd_handle    = BLE_GATT_HANDLE_INVALID;
13.     cp_init.disconnect_on_fail             = false;
14.     cp_init.evt_handler                    = on_conn_params_evt;
15.     cp_init.error_handler                  = conn_params_error_handler;
16.   //启动蓝牙 BLE 连接参数初始化函数,导入参数
17.     err_code = ble_conn_params_init(&cp_init);
18.     APP_ERROR_CHECK(err_code);
19. }
```

这个函数设置的并没有设置连接参数值，连接参数实际上在 GAP 初始化中已经设置了，这个函数主要关注的是链接参数更新，设置链接更新的方式，下面就结合理论来详细探讨一下。 ble_conn_params_init 函数实际上初始化的一个结构体 cp_init，那么这个结构体的函数包含哪些参数了？详细看下面的中文标注：

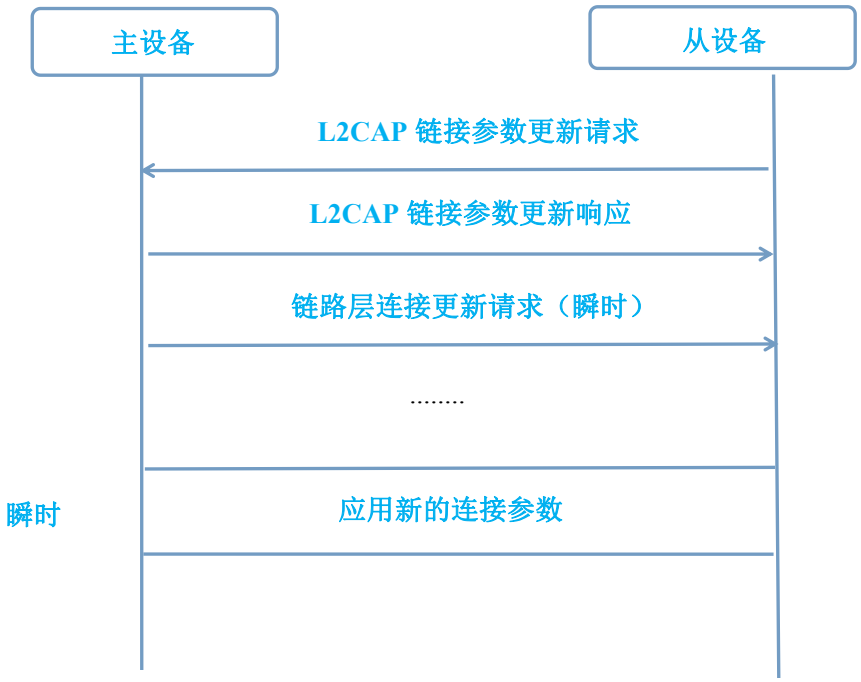
```
20.     typedef struct
21. {
22.     ble_gap_conn_params_t *          p_conn_params;
23.  /**< 指向应用程序中设置的 GAP 连接参数,如果 ble_conn_params_init 中 p_conn_params 参数
    设置为 NULL, 那么连接参数从主机获得（正在使用的链接参数） */
24.     uint32_t                 first_conn_params_update_delay;
25. /**< 初始化事件（连接或者启动通知）到第一次连接参数更新的时间. */
26.     uint32_t                 next_conn_params_update_delay;
27. /**< 第一次更新后，下次发起更新申请的时间间隔。*/
28.     uint8_t              max_conn_params_update_count;
29. /**<放弃协商连接参数前尝试的最大的次数 */
30.     uint16_t                                 start_on_notify_cccd_handle;
31. /**< 如果是当通知开始时启动的这个过程，设置为对应的 CCCD 的句柄. 如果是从连接开始时
    启动的，则设置为 BLE_GATT_HANDLE_INVALID */
32.     bool                                     disconnect_on_fail;
33. /**< 设置为 TRUE 时，当连接参数更新失败，则会自动断开连接. */
34.     ble_conn_params_evt_handler_t evt_handler;
35.  /**< 连续参数更新参数对应的处理事件. */
36.     ble_srv_error_handler_t          error_handler;
37. /**< 发送错误时的错误处理 */
38. } ble_conn_params_init_t;
39.
```

在谈上面的这些参数前，先来描述下链接参数更新的过程：

1.1 链接参数更新描述

下图为链路控制和适配协议链接参数更新请求和响应命令。在链接中，如果从设备希望修改当前的链接参数则可以使用该命令。在低功耗蓝牙中，为了实现极低的功耗，低功耗蓝牙协议栈设计为不需要射频时彻底将空中射频关断。所以，尽可能降低从机监听链接事件所需要的频率对提高电池寿命是至关重要的。例如：如果链接事件间隔太快，导致过多的电量浪费。这种情况在从机设备时延很大时没有问题，否则从机设备将会频

繁的监听链路。



链接参数更新请求命令用于从设备向主设备发送，因为主设备随时能够启动链路层链接参数更新控制规程。如果该命令是由主设备发送的，从设备会将其视为一个错误，并返回原因代码为“命令不理解”的指令拒绝命令。

那么在 nrf52832 系列处理器中，会尝试多次 L2CAP 链接参数更新请求。

参数 first_conn_params_update_delay 为初始化后到第一次更新请求的时间间隔。如果没有得到主机的**链接参数更新响应，那么开始申请第二次更新请求，直到到最大的申请次数，如果还是失败，则断开连接。**那么参数 next_conn_params_update_delay 就是后面每次发起申请的时间间隔。而参数 max_conn_params_update_count 就是最大申请次数。

1.2 链接参数更新应答

链接参数更新应答命令用于主设备向从设备发送。

从机设备可以在任何时候发送链接参数更新请求命令。收到该信息的主设备如果可以修改链接参数，则将返回链接参数更应答命令，其中的结果代码设置为接受。然后，主设备将会启动链路层链接参数更新控制规程对链接参数进行更新。

如果主设备不同意从设备的请求命令，它可以发送结果代码为拒绝的链接参数更新应答来拒绝请求，此时，从设备有两种选择：接收主设备希望的正在使用的链接参数或者终止链接。

修改链接参数时，如果要减少主设备拒绝从设备请求的可能性，可以在请求里设置一个可接受的参数范围或者从设备提供一个合理的从设备延迟。主设备可以选择合适的链接事件间隔，从设备可以使用最佳功耗的从设备延迟参数。

2：链接参数更新初始化配置：

上面讲述了链接参数协商的过程，实际的函数配置如下，如果 p_conn_params 参数设置为非 NULL，**则需要我们在 GAP 中调用 sd_ble_gap_ppcp_set 函数设置下三个链接参数（gap 初始化设置过）**。如果设置下为 NULL，则直接从 GAP 的堆栈中直接读取设置，也就是主机默认的三个链接参数。那么这里直接把 p_conn_params 设置为非 NULL，也就是说我们后面要定时更新连接参数：

```
01. uint32_t ble_conn_params_init(const ble_conn_params_init_t * p_init)
02. {
03.     uint32_t err_code;
04.
05.     m_conn_params_config = *p_init;
06.     m_change_param = false;
07.     if (p_init->p_conn_params != NULL)
08.     {
09.         m_preferred_conn_params = *p_init->p_conn_params;
10.         // 在堆栈中设置连接参数
11.         err_code = sd_ble_gap_ppcp_set(&m_preferred_conn_params);
12.         if (err_code != NRF_SUCCESS)
13.         {
14.             return err_code;
15.         }
16.     }
17.     else
18.     {
19.         // 从堆栈 stack 中读取参数
20.         err_code = sd_ble_gap_ppcp_get(&m_preferred_conn_params);
21.         if (err_code != NRF_SUCCESS)
22.         {
23.             return err_code;
24.         }
25.     }
26.  //看从设备数目
27.     for (uint32_t i = 0; i < NRF_BLE_CONN_PARAMS_INSTANCE_COUNT; i++)
28.     {
29.         ble_conn_params_instance_t * p_instance = &m_conn_params_instances[i];
30.
31.         instance_free(p_instance);
32.         p_instance->timer_id = &m_timer_data[i];
33.       //启动定时器开始定时
34.         err_code = app_timer_create(&p_instance->timer_id,
35.                                     APP_TIMER_MODE_SINGLE_SHOT,
36.                                     update_timeout_handler);
37.         if (err_code != NRF_SUCCESS)
```

```
38.             {
39.                 return NRF_ERROR_INTERNAL;
40.             }
41.         }
42.         //lint -restore
43.
44.         return NRF_SUCCESS;
45.
46. }
```

设置完了，就会启动一个软件定时器，关于软件定时器的使用请参考教材《青风带你学蓝牙：定时器的应用》。定时器设置在超时溢出后执行超时操作，这个超时操作就是 update_timeout_handler，也就是参数更新了。下面就是具体的参数更新函数：

```
47. static void update_timeout_handler(void * p_context)
48. {
49.     UNUSED_PARAMETER(p_context);
50.
51.     if (m_conn_handle != BLE_CONN_HANDLE_INVALID)
52.     {
53.         // 如果达到最大的能被接受的数目
54.         m_update_count++;
55.         if (m_update_count <= m_conn_params_config.max_conn_params_update_count)
56.         {
57.             uint32_t err_code;
58.             // 参数没有 ok,发送连接参数更新要求
59.         err_code = sd_ble_gap_conn_param_update(m_conn_handle, &m_preferred_conn_params);
60.             if ((err_code != NRF_SUCCESS) && (m_conn_params_config.error_handler !=
    NULL))
61.             {
62.                 m_conn_params_config.error_handler(err_code);
63.             }
64.         }
65.         else
66.         {
67.             m_update_count = 0;
68.             // 协商失败，如果已经配置了则自动断开
69.             if (m_conn_params_config.disconnect_on_fail)
70.             {
71.                 uint32_t err_code;
72.                 err_code = sd_ble_gap_disconnect(m_conn_handle,
    BLE_HCI_CONN_INTERVAL_UNACCEPTABLE);
73.                 if ((err_code != NRF_SUCCESS) && (m_conn_params_config.error_handler !=
    NULL))
74.                 {
```

```
75.                         m_conn_params_config.error_handler(err_code);
76.                     }
77.                 }
78.                 // 通知应用，上述过程失败
79.                 if (m_conn_params_config.evt_handler != NULL)
80.                 {
81.                     ble_conn_params_evt_t evt;
82.
83.                     evt.evt_type = BLE_CONN_PARAMS_EVT_FAILED;
84.                     m_conn_params_config.evt_handler(&evt);
85.                 }
86.             }
87.         }
88. }
89.
```

在最大的申请次数下，如果链接参数更新申请被响应，那么就调用协议栈的 API 函数 sd_ble_gap_conn_param_update 进行参数更新。

如果申请失败，这就调用协议栈的 API 函数 sd_ble_gap_disconnect 断开连接。

那么这里关注的焦点就是 API 函数 sd_ble_gap_conn_param_update 了，下面就是改函数的接口介绍，由于协议栈不开源，内部是看不到的，我们只能看他提供的接口了，直接调用改函数可以实现连接参数更新：

```
uint32_t
sd_ble_gap_conn_param_update   (  uint16_t                        conn_handle,
                                  ble_gap_conn_params_t const *   p_conn_params
                                )
```

更新连接参数：

在中央的角色这将发起一个链路层连接参数更新过程 ，否则在外围的角色,这将发送相应的 L2CAP 请求,等待中央执行过程。 在这两种情况下,不管成功或失败,应用程序将被告知 BLE_GAP_EVT_CONN_PARAM_UPDATE 事件的结果。 这个函数可以用来作为中央设备回复 BLE_GAP_EVT_CONN_PARAM_UPDATE_REQUEST 或启动程序被拒绝。

事件生成：

BLE_GAP_EVT_CONN_PARAM_UPDATE	连接参数更新的结果

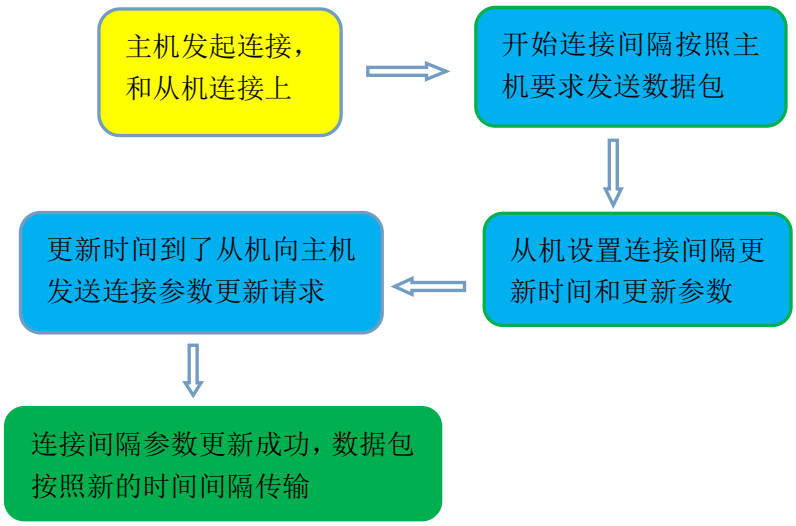
参数

[in]	**conn_handle**	连接句柄
[in]	**p_conn_params**	指针指向需要的连接参数。 如果外部设备设置为 NULL，那么将用 GAP 服务中的 PPCP 特征值取代。如果中央主设备设置为 NULL 同时应答为 BLE_GAP_EVT_CONN_PARAM_UPDATE_REQUEST，外部设备请求会被拒绝。

返回值：

NRF_SUCCESS	连接升级过程成功开始
NRF_ERROR_INVALID_ADDR	提供的是无效的指针
NRF_ERROR_INVALID_PARAM	提供的是无效的参数，检查参数限制
NRF_ERROR_INVALID_STATE	无效的状态来实行操作
NRF_ERROR_BUSY	过程已经在进行或不允许在这个时候,进程等待事件和等待等待程序完成并重试。
BLE_ERROR_INVALID_CONN_HANDLE	提供的是无效的连接句柄
NRF_ERROR_NO_MEM	没有足够的内存去完成操作

整个更新连接参数的过程就讲完了，下面我们有个图表详细的描述一下整个过程：



3：参数更新过程演示：

看下应用实例，我们采用同一个手机做主机，连接从设备。从设备里分别下载了蓝牙样例历程和蓝牙串口历程，我们通过抓包进行对比，要更新的连接参数值在《GAP 详解》里讲过在 GAP 初始化函数中设置：

	串口蓝牙例子	蓝牙样例
更新的 Max 时间	75ms	200ms
更新的 Min 时间	20ms	100ms

蓝牙样例抓包演示：

① 连接的时候采用主机默认的连接参数,一个连接间隔大概为 44ms:

P.nbr.	Time (us)	Channel	Access Address	Direction	ACK Status	Data Type	Data Header					CRC	RSSI (dBm)	FCS
							LLID	NESN	SN	MD	PDU-Length			
694	+44770 =27408608	0x08	0x66A1D3A3	M->S	OK	Empty PDU	1	0	0	0	0	0x6395F3	-44	OK
695	+230 =27408838	0x08	0x66A1D3A3	S->M	OK	Empty PDU	1	1	0	0	0	0x639320	-46	OK
696	+44771 =27453609	0x22	0x66A1D3A3	M->S	OK	Empty PDU	1	1	1	0	0	0x639E86	-54	OK
697	+230 =27453839	0x22	0x66A1D3A3	S->M	OK	Empty PDU	1	0	1	0	0	0x639855	-46	OK

② **从机发起更新，**首先是 S->M，即从设备发送连接参数更新请求，请求中带有申请的连接参数，然后 M->S，即主设备返回连接参数更新响应，Result 为 0，表示同意修改更新。蓝色部分 M->S 发送 Data type 为 Control 的链路层连接参数更新控制规程，携带同意的连接参数，这样，新的连接参数就会投入使用。图中连接参数为 min 为 0x50，max 为 0xA0，延迟为 0。单位为 1.25ms,正好验证了我们的设置更新的 Max 时间为 200ms，Min 时间为 100ms。

P.nbr.	Time (us)	Channel	Access Address	Direction	ACK Status	Data Type	Data Header					L2CAP Header		SIG Pkt Header			SIG_Connection_Param_Update_Req			
							LLID	NESN	SN	MD	PDU-Length	L2CAP-Length	ChanId	Code	Id	Data-Length	IntervalMin	IntervalMax	SlaveLatency	TimeoutMultiplier
1944	+230 =15086063	0x10	0x8E744A70	S->M	OK	L2CAP-S	2	0	1	0	16	0x000C	0x0005	0x12	0x03	0x0008	0x0050	0x00A0	0x0000	0x0190
1945	+44772 =15130835	0x15	0x8E744A70	M->S	OK	Control	3	0	0	0	12	LL_Opcode		LL_Connect_Update_Req						
												Connection_Update_Req(0x00)		WinSize 0x02	WinOffset 0x006B	Interval 0x009C	Latency 0x0000	Timeout 0x0190	Instant 0x008A	CRC 0x7F99A0
1946	+325 =15131160	0x15	0x8E744A70	S->M	OK	Empty PDU	1	1	0	0	0	CRC 0xCE6857	RSSI (dBm) -46	FCS OK						
1947	+44676 =15175836	0x1A	0x8E744A70	?	OK	L2CAP-S	2	1	1	0	10	0x0006	0x0005	0x13	0x03	0x0002	0x0000	CRC 0xD50799	RSSI (dBm) -50	FCS OK

P.nbr.	Time (us)	Channel	Access Address	Adv PDU Type	Adv PDU Header				InitA	AdvA	LLData (Part 1)				
					Type	TxAdd	RxAdd	PDU-Length			AccessAddr	CRCInit	WinSize	WinOffset	Interval	Latency
576	+406 =18259483	0x25	0x8E89BED6	ADV_CONNECT_REQ	5	1	1	34	0x4A5F864F8870	0xEDB84F1FC83A	0x3396BB26	06 A2 75	02	0x0014	0x0024	0x0000

P.nbr.	Time (us)	Channel	Access Address	Direction	ACK Status	Data Type	Data Header					LL_Opcode	LL_Feature_Req		RSSI (dBm)	FCS
							LLID	NESN	SN	MD	PDU-Length		FeatureSet	CRC		
577	+27425 =18286908	0x16	0x3396BB26	M->S	OK	Control	3	0	0	0	9	Feature_Req(0x08)	00 00 00 00 00 00 00 FF	0xAF2431	-49	OK

计算下第一次更新时间大概为 **23-18=5s**，和我们设置第一次更新时间 first_conn_params_update_delay **的值相符合。这个时间长度就是由初始化里的软件定时器实现的。**

③连接间隔按照更新的时间运行，主机根据情况的运行在 gap 设置的范围内，180ms：

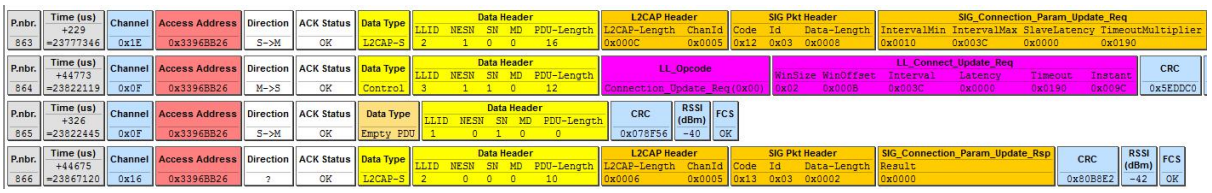
P.nbr.	Time (us)	Channel	Access Address	Direction	ACK Status	Data Type	Data Header					CRC	RSSI (dBm)	FCS
							LLID	NESN	SN	MD	PDU-Length			
1961	+44771 =15490847	0x18	0x8E744A70	?	OK	Empty PDU	1	0	0	0	0	0xCE6E84	-48	OK
1962	+230 =15491077	0x18	0x8E744A70	?	OK	Empty PDU	1	1	0	0	0	0xCE6857	-47	OK
1963	+179777 =15670854	0x1D	0x8E744A70	?	OK	Empty PDU	1	1	1	0	0	0xCE65F1	-48	OK
1964	+230 =15671084	0x1D	0x8E744A70	?	OK	Empty PDU	1	0	1	0	0	0xCE6322	-48	OK

蓝牙串口抓包演示：

①连接的时候采用主机默认的连接参数,一个连接间隔大概为 44ms:

P.nbr.	Time (us)	Channel	Access Address	Direction	ACK Status	Data Type	Data Header					CRC	RSSI (dBm)	FCS
							LLID	NESN	SN	MD	PDU-Length			
694	+44770 =27408608	0x08	0x66A1D3A3	M->S	OK	Empty PDU	1	0	0	0	0	0x6395F3	-44	OK
695	+230 =27408838	0x08	0x66A1D3A3	S->M	OK	Empty PDU	1	1	0	0	0	0x639320	-46	OK
696	+44771 =27453609	0x22	0x66A1D3A3	M->S	OK	Empty PDU	1	1	1	0	0	0x639E86	-54	OK
697	+230 =27453839	0x22	0x66A1D3A3	S->M	OK	Empty PDU	1	0	1	0	0	0x639855	-46	OK

②从机发起更新，首先是 S->M，即从设备发送连接参数更新请求，请求中带有申请的连接参数，然后 M->S，即主设备返回连接参数更新响应，Result 为 0，表示同意修改更新。蓝色部分 M->S 发送 Data type 为 Control 的链路层连接参数更新控制规程，携带同意的连接参数，这样，新的连接参数就会投入使用。图中连接参数为 min 为 0x10，max 为 0x3c，延迟为 0。单位为 1.25ms，正好验证了我们的设置更新的 Max 时间为 20ms，Min 时间为 75ms。



P.nbr.	Time (us)	Channel	Access Address	Direction	ACK Status	Data Type	Data Header	L2CAP Header	SIG Pkt Header	SIG_Connection_Param_Update_Req	CRC	RSSI (dBm)	FCS
863	+229 =23777346	0x1E	0x3396BB26	S->M	OK	L2CAP-S	LLID NESN SN MD PDU-Length: 2 1 0 0 16	L2CAP-Length ChanId: 0x000C 0x0005	Code Id Data-Length: 0x12 0x03 0x0008	IntervalMin IntervalMax SlaveLatency TimeoutMultiplier: 0x0010 0x003C 0x0000 0x0190			
864	+44773 =23822119	0x0F	0x3396BB26	M->S	OK	Control	LLID NESN SN MD PDU-Length: 3 1 1 0 12	LL_Opcode: Connection_Update_Req(0x00)		WinSize WinOffset Interval Latency Timeout Instant: 0x02 0x000B 0x003C 0x0000 0x0190 0x009C	0x5EDDC0		
865	+326 =23822445	0x0F	0x3396BB26	S->M	OK	Empty PDU	LLID NESN SN MD PDU-Length: 1 0 1 0 0	0x078F56	-40	OK			
866	+44675 =23867120	0x16	0x3396BB26	?	OK	L2CAP-S	LLID NESN SN MD PDU-Length: 2 0 0 0 10	L2CAP-Length ChanId: 0x0006 0x0005	Code Id Data-Length: 0x13 0x03 0x0002	Result: 0x0000	0x80B8E2	-42	OK

③连接间隔按照更新的时间运行，主机根据时间情况运行在设置范围内，75ms：

P.nbr.	Time (us)	Channel	Access Address	Direction	ACK Status	Data Type	Data Header	CRC	RSSI (dBm)	FCS
1432	+44772 =15338898	0x0B	0xBD4840D2	?	OK	Empty PDU	LLID NESN SN MD PDU-Length: 1 1 1 0 0	0x36208D	-36	OK
1433	+229 =15339127	0x0B	0xBD4840D2	?	OK	Empty PDU	LLID NESN SN MD PDU-Length: 1 0 1 0 0	0x36265E	-39	OK
1434	+44773 =15383900	0x11	0xBD4840D2	?	OK	Empty PDU	LLID NESN SN MD PDU-Length: 1 0 0 0 0	0x362BF8	-35	OK
1435	+229 =15384129	0x11	0xBD4840D2	?	OK	Empty PDU	LLID NESN SN MD PDU-Length: 1 1 0 0 0	0x362D2B	-39	OK
1436	+59773 =15443902	0x17	0xBD4840D2	?	OK	Empty PDU	LLID NESN SN MD PDU-Length: 1 1 1 0 0	0x36208D	-36	OK
1437	+230 =15444132	0x17	0xBD4840D2	?	OK	Empty PDU	LLID NESN SN MD PDU-Length: 1 0 1 0 0	0x36265E	-38	OK
1438	+74773 =15518905	0x04	0xBD4840D2	?	OK	Empty PDU	LLID NESN SN MD PDU-Length: 1 0 0 0 0	0x362BF8	-36	OK
1439	+230 =15519135	0x04	0xBD4840D2	?	OK	Empty PDU	LLID NESN SN MD PDU-Length: 1 1 0 0 0	0x362D2B	-38	OK

4．本章总结：

根据这一章的学习，你可以再程序中自由的设置和更改连接参数，提高数据流量或者降低功耗。在链路层的连接配置过程完成后，链接参数 connInterval，connSlaceLatency 和 connTimeout 参数可以实现更新。主设备通过发送 CONN_PARAM_UPDATE_REQ 数据包去完成更新，而从设备则通过 Host 层的处理去影响这些参数。本讲主要是探讨如何设置从机更新请求过程，因为整个更新必须从机发起申请，再由主机发起更新。