**Nb-IOT SDK移植说明**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本号** | **修订日期** | **修订内容** | **修订人** |
| V1.0 | 2017.03.14 | 新增 | XiaoBo |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

1. **SDK目录结构**

nbiot\_sdk

├ include

│ └ platform.h //系统相关的数据类型以及函数定义

├ platforms

│ ├ win //windows系统相关的实现文件

│ ├ posix //posix系统相关的实现文件

├ source

│ └ \*.c //sdk实现

└ sample

└ main.c //测试程序

以上目录结构中，红色字体副本是移植性系统需要实现的部分。具体如下：

* platform.h – 定义sdk所需的size\_t、int8\_t、uint8\_t、int16\_t、uint16\_t、int32\_t、uint32\_t、int64\_t、uint64\_t、float、double、bool、time\_t、clock\_t、DBL\_MAX、PRId64等。
* platforms

├ win

│ ├ environment.c //windows系统环境初始化相关函数实现

│ ├ memory.c //windows系统内存分配相关函数实现

│ ├ socket.c //windows系统的udp socket相关函数实现

│ ├ time.c //windows系统的时间相关函数实现

│ └ utils.c //windows系统对utils.h中定义函数的实现

├ posix

│ ├ environment.c //posix系统环境初始化相关函数实现

│ ├ memory.c //posix系统内存分配相关函数实现

│ ├ socket.c //posix系统的udp socket相关函数实现

│ ├ time.c //posix系统的时间相关函数实现

│ └ utils.c //posix系统对utils.h中定义函数的实现

1. **移植说明（以windows示例）**
   1. 开发步骤说明
2. 在nbiot\_sdk/platforms目录下，创建win目录
3. 在nbiot\_sdk/platforms/win目录下，创建相关“.c”实现文件（可仿照posix目录下的文件结构）
4. 实现nbiot\_sdk/platforms/win目录下的“.c”文件

* environment.c
* memory.c
* socket.c
* time.c
* utils.c

具体的实现细节参见2.2节以及posix的实现文件

* 1. 定义数据类型以及接口实现

1. 初始化sdk运行环境(environment.c)

* 函数
* void nbiot\_init\_environment( int argc, char \*argv[] )
* void nbiot\_clear\_environment( void )

由于windows下socket库需要初始化，所以在nbiot\_init\_environment中显示调用socket库的初始化操作，在nbiot\_clear\_environment中显示调用socket库的清理工作，代码如下：

static bool \_nbiot\_init\_state = false;

void nbiot\_init\_environment( int argc, char \*argv[] )

{

if ( !\_nbiot\_init\_state )

{

WSADATA wsa;

WSAStartup( MAKEWORD(2,2), &wsa );

\_nbiot\_init\_state = true;

}

}

void nbiot\_clear\_environment( void )

{

if ( \_nbiot\_init\_state )

{

WSACleanup();

\_nbiot\_init\_state = false;

}

}

1. 内存管理(memory.c)

* 函数
* void \*nbiot\_malloc( size\_t size )
* void nbiot\_free( void \*ptr )

为了管理sdk运行过程的内存分配与释放，抽离出以上2个接口函数，便于不同环境下采用合适的内存管理策略。

1. udp socket实现(socket.c)

* 类型
* struct nbiot\_socket\_t

{

SOCKET sock;

};

* struct nbiot\_sockaddr\_t

{

struct sockaddr\_in addr;

};

* 函数
* int nbiot\_udp\_create( nbiot\_socket\_t \*\*sock )
* int nbiot\_udp\_close( nbiot\_socket\_t \*sock )
* int nbiot\_udp\_bind( nbiot\_socket\_t \*sock,

const char \*addr,

uint16\_t port )

* int nbiot\_udp\_connect( nbiot\_socket\_t \*sock,

const char \*addr,

uint16\_t port,

nbiot\_sockaddr\_t \*\*dest )

* int nbiot\_udp\_send( nbiot\_socket\_t \*sock,

const void \*buff,

size\_t size,

size\_t \*sent,

const nbiot\_sockaddr\_t \*dest )

* int nbiot\_udp\_recv( nbiot\_socket\_t \*sock,

void \*buff,

size\_t size,

size\_t \*read,

nbiot\_sockaddr\_t \*\*src )

* bool nbiot\_sockaddr\_equal( const nbiot\_sockaddr\_t \*addr1,

const nbiot\_sockaddr\_t \*addr2 )

* void nbiot\_sockaddr\_destroy( nbiot\_sockaddr\_t \*s )

为了方便移植udp socket，抽离出以上8个接口函数。

注：udp socket需实现成非阻塞模式。

1. 时间函数(time.c)

* 函数
* time\_t nbiot\_time( void )
* clock\_t nbiot\_tick( void )
* void nbiot\_sleep( int milliseconds )

此文件实现了3个与时间相关的接口。

1. 工具函数(utils.c)

* 函数
* int nbiot\_strlen( const char \*str )
* int nbiot\_strncpy( char \*dest,

const char \*src,

int size )

* int nbiot\_strncmp( const char \*str1,

const char \*str2,

int size )

* char\* nbiot\_strdup( const char \*str,

int size )

* char\* nbiot\_strrchr( const char \*str,

int size,

char ch )

* int nbiot\_atoi( const char \*str )
* int nbiot\_itoa( int val,

char \*str,

int size )

* void \*nbiot\_memmove( void \*dst,

const void \*src,

size\_t size )

* int nbiot\_memcmp( const void \*mem1,

const void \*mem2,

size\_t size )

* void nbiot\_memzero( void \*mem,

size\_t size )

* int nbiot\_rand( void )
* int nbiot\_isspace( int ch )
* int nbiot\_isprint( int ch )
* void nbiot\_printf( const char \*format, ... )
* void nbiot\_buffer\_printf( const void \*buf,

size\_t len )

此文件实现了字符串操作、内存操作以及随机数等工具函数，大部分是纯c编码，与环境无关。