附件2：CAN参数设置

说明书

说明书版本：V2.03

更新日期：2017.06.30

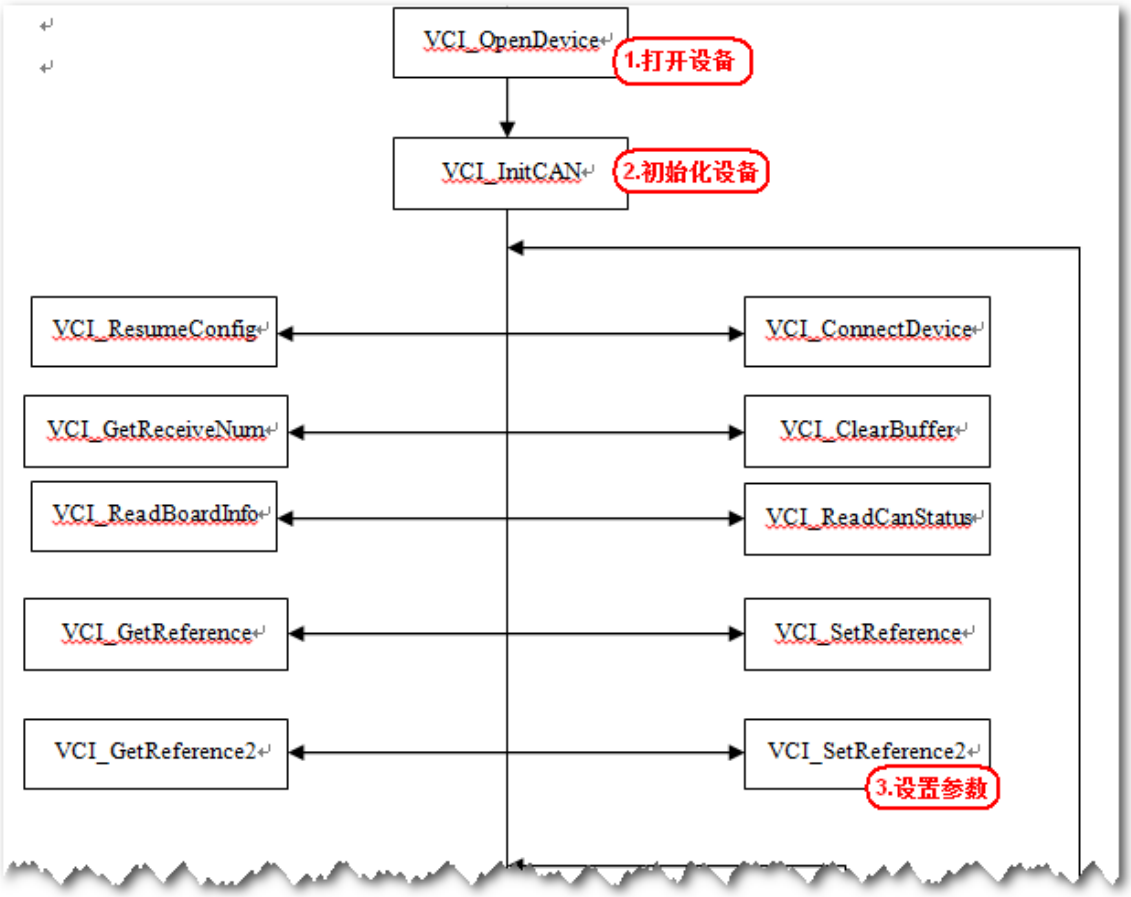
目　录

一、概述

为方便不同用户的需求，USB-CAN 适配器中对参数设置提供了两种模式：1.简洁模式，该模式下提供了波特率、工作模式、滤波模式等的简单选项，在使用随机附带的 USB-CAN Tool 调试工具或用户自定义编程时，仅需进行简单选择设置即可快速配置适配器，适合入门或刚接触 CAN 通信不久的用户使用；2.专业模式，该模式下用户可以通过修改多种寄存器参数来实现自定义波特率，自定义硬件滤波方式等，适合较熟悉 CAN 通信或发烧友等用户使用。

当然，这两种模式是根据您的使用情况自由选择的。

通过调用 ControlCAN.dll 中的“VCI_SetReference2”函数进行参数设置，该函数的使用流程如下图，关于函数的详细介绍请参见《接口函数库（二次开发库）使用说明书.pdf》



注意：

CAN 总线在正常收发数据的时候，尽量不要通过 USBCAN 适配器修改 CAN 总线参数或关闭 CAN 总线，应等数据收发停止或将 USBCAN 适配器脱离 CAN 总线再进行相应操作。

二、波特率设置

2.1 兼容 SJA1000 模式

为了方便（SJA1000）CAN 控制器的用户，USB-CAN 适配器中做了相应的兼容性处理，用户只需配置相应的 Timing0（BTR0）、Timing1（BTR1）寄存器的值，即可配置得到相应的波特率。而不必接触相对复杂、专业的多个寄存器参数。

通过该模式设置波特率时，SetReference2 函数的 RefType 参数须传递 10，具体应用可参考本小节后的示例代码。常规波特率索引值对照表如下：

CAN波特率	Timing0(BTR0)	Timing1(BTR1)
10 Kbps	0x31	0x1C
20 Kbps	0x18	0x1C
40 Kbps	0x87	0xFF
50 Kbps	0x09	0x1C
80 Kbps	0x83	0xFF
100 Kbps	0x04	0x1C
125 Kbps	0x03	0x1C
200 Kbps	0x81	0xFA
250 Kbps	0x01	0x1C
400 Kbps	0x80	0xFA
500 Kbps	0x00	0x1C
666 Kbps	0x80	0xB6
800 Kbps	0x00	0x16
1000 Kbps	0x00	0x14
33.33 Kbps	0x09	0x6F
66.66 Kbps	0x04	0x6F
83.33 Kbps	0x03	0x6F

注：

1.配置波特率时，用户只需要按照 SJA1000（16MHz）给的波特率参数进行设置即可。

2.常规波特率直接按照上表的值配置即可。其它非常规波特率，可以使用附带的波特率侦测工具进行侦测，并到得相应的波特率参数。或是使用USB_CAN TOOL安装目录下的波特率计算工具计算。（参考《6.插件2：波特率侦测工具使用说明书.pdf》）

3.本适配器暂时不支持 10K 以下波特率。

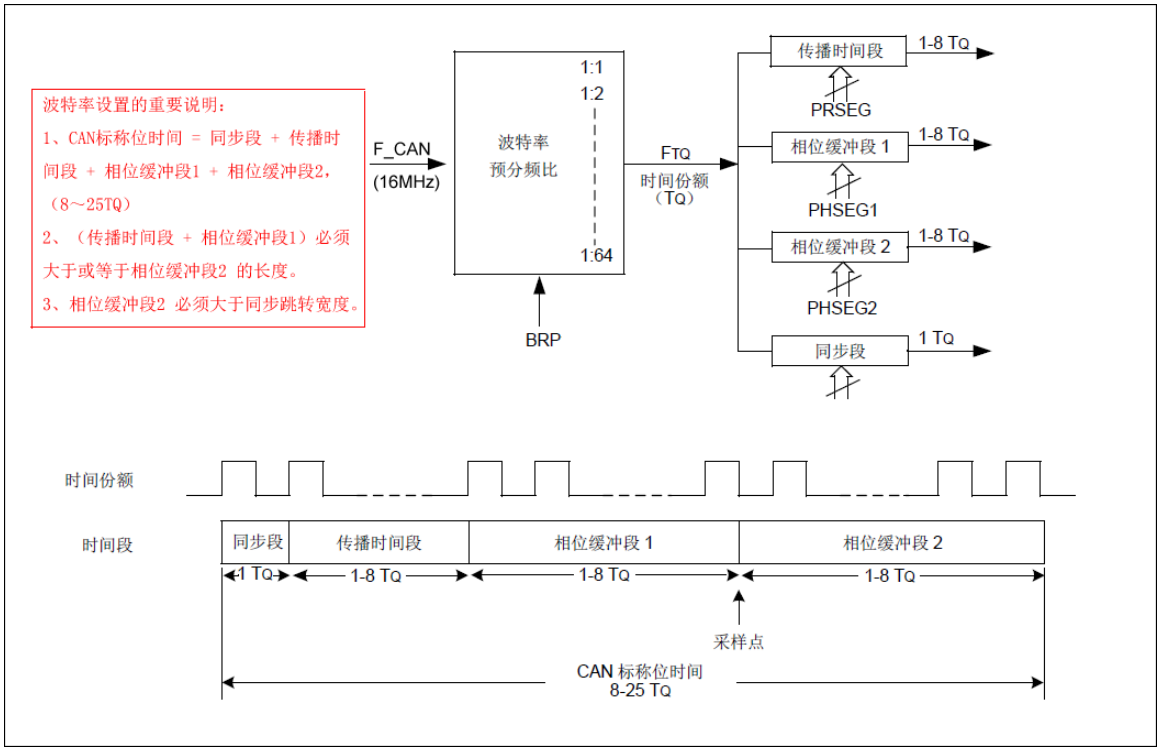
示例 1_兼容 SJA1000 模式设置波特率的调用示例：

```
#include "ControlCan.h"

int   nDeviceType = 4;          /*USB-CAN2.0 */

int   nDeviceInd = 0;           /*  第0个设备  */

int   nCANInd = 0;

BYTE refType = 10;              /*  使用兼容SJA1000的模式进行波特率设置  */

DWORD dwRel;

BYTE pData[3] ={0};

pData[0] = 0;                   /*  保留参数，填0即可  */

pData[1] = 0x03;                /* Timing0（BTR0）的值*/

pData[2] = 0x1C;                /* Timing1（BTR1）的值---BTR0和BTR1组合设置波特率=125kbps*/

bRel = VCI_SetReference2(nDeviceType, nDeviceInd, nCANInd, refType,pData);

if(dwRel != 1)

{

   MessageBox(_T("设置波特率失败!"), _T("警告"), MB_OK|MB_ICONQUESTION);

   return FALSE;

}
```

2.2 高级模式

波特率设置的高级模式中，用户通过设置 CAN 寄存器值来改变需要的波特率。与波特率相关的寄存器共有 7 个，分别是：同步跳转宽度、预分频、采样点、相位缓冲段 2 选择位、同步段、传播时间段、相位缓冲段 1 和相位缓冲段 2。下列的位时序图详细说明了这些寄存

器之间的关系：



关于CAN自定义波特率设置的几点重要说明：

1)、CAN标称位时间 = *同步段* + *传播时间段* + *相位缓冲段1* + *相位缓冲段2*，(8～25TQ)。

2）、(*传播时间段* + *相位缓冲段1*)必须大于或等于*相位缓冲段2*的长度。

3）、*相位缓冲段2*必须大于同步跳转宽度。

波特率计算公式=16000000/(同步段+传播时间段+相位缓冲段 1+相位缓冲段 2)/预分频。

各寄存器理论取值范围如下表所示：

简写	描述	取值范围	备注
SJW	同步跳转宽度	1-4	
BRP	预分频	1-64	
SAM	采样点	0-1	*0*-采样一次 *1*-采样三次
SYN	同步段	1	该值强制为1，用户无需设置
PHSEG2_SEL	相位缓冲段2选择位	0-1	*0*-由相位缓冲段1时间决定，此时相位缓冲段2的值强制与相位缓冲段1相等 *1*-可编程
PRSEG	传播时间段	1-8	

简写	描述	取值范围	备注
PHSEG1	相位缓冲段1	1-8	
PHSEG2	相位缓冲段2	1-8	

通过寄存器设置波特率时，SetReference2函数的RefType参数须传递16，具体应用可参考本小节后的示例代码。

示例2_波特率设置的高级模式VC调用示例：

```
#include "ControlCan.h"

int   nDeviceType = 4;        /*USB-CAN2.0 */

int   nDeviceInd = 0;         /* 第0个设备 */

int   nCANInd = 0;

BYTE refType = 16;            /* 选择通过寄存器设置波特率 */

DWORD dwRel;

BYTE pData[11] ={0};

for(int i=0;i<4;i++)

   pData[i] = 0;              /* 字节0-3用于在GetReference2时保存波特率实际数据，设置时保留 */

/* 设置波特率为125kbps =16000000/(1+7+4+4)/8   */

pData[4] = 1;                 /* 同步跳转宽度SJW=1 */

pData[5] = 8;                 /* 预分频BRP=8 */

pData[6] = 0;                 /* 采样点SAM=0 ： 采样一次*/

pData[7] = 1;                 /* 相位缓冲段2选择PHSEG2_SEL=1 ： 可编程*/

pData[8] = 7;                 /* 传播时间段PRSEG= 7 */

pData[9] = 4;                 /* 相位缓冲段1 PHSEG1=4 */

pData[10] = 4;                /* 相位缓冲段2 PHSEG1=4 */

bRel = VCI_SetReference2(nDeviceType, nDeviceInd, nCANInd, refType,pData);

if(dwRel != 1)

{

   MessageBox(_T("设置波特率失败!"), _T("警告"), MB_OK|MB_ICONQUESTION);

   return FALSE;

}
```

三、工作模式

USB-CAN 适配器支持 3 种工作模式：正常工作、仅侦听模式和自测模式（环回模式）。这三种工作模式的取值对照表如下所示：

工作模式	取值	备注
正常工作	0	CAN模块会出现在CAN总线上，可以发送和接收CAN 报文。
仅侦听模式	1	如果仅监听模式被激活，则模块会出现在CAN总线上，但处于被动状态。它会接收报文，但不会发送报文，也不会应答信号。该模式可用作总线监视器，因为CAN总线不会影响数据通信。
自测模式（环回模式）	2	自测（环回）模式用于适配器进行自测试，让CAN模块接收它自己的报文。在该模式下，CAN模块发送路径在内部与接收路径相连接。该模式下会提供“假”应答，从而不需要另一个节点来提供应答位。CAN报文不会实际发送到CAN总线上。适配器发出的CAN帧将能被适配器接收回来。

设置工作模式时，SetReference2函数的RefType参数须传递0，具体应用可参考本小节后的示例代码。

示例3_设置工作模式：

```
#include "ControlCan.h"

int   nDeviceType = 4;        /*USB-CAN2.0 */

int   nDeviceInd = 0;         /* 第0个设备 */

int   nCANInd = 0;

BYTE refType = 0;             /* 设置工作模式 */

DWORD dwRel;

BYTE pData[1] ={0};

pData[0] = 2;                 /* 设置“自测模式（环回模式）”*/

bRel = VCI_SetReference2(nDeviceType, nDeviceInd, nCANInd, refType,pData);

if(dwRel != 1)

{
```

```
    MessageBox(_T("设置工作模式失败!"), _T("警告"), MB_OK|MB_ICONQUESTION);
    return FALSE;
}
```

四、滤波设置

4.1 滤波寄存器

滤波寄存器包括 2 个 32 位的寄存器：过滤验收滤波器（ACR）和过滤屏蔽寄存器（AMR），通过软件设置这两个寄存器的值，实现对 CAN 接收报文的过滤功能。

在 CAN 模块接收到报文时，会将报文标识符与过滤器中的相应位进行比较。如果标识符与用户配置的过滤器匹配，报文会被存储到 CAN 控制器相应的接收缓存队列中。

接收屏蔽器可用于在接收时忽略标识符的选定位。在接收报文时，这些位将不与过滤器中的位进行比较。例如，如果用户希望接收带有标识符 0、1、2 和 3 的所有报文，用户需要屏蔽掉标识符的低 2 位。屏蔽寄存器的某一位等于 1 时，表示忽略对该位对应 ID 位的滤波，如屏蔽寄存器值=FFFFFFFF，则可接收所有消息。

验收滤波器ACR，验收屏蔽器AMR都是32bits（4bytes）。对于需要验收滤波的ID值，ID的最高位（标准帧ID最高位为Bit10，扩展帧为Bit28）与ACR/AMR的最高位（Bit31）位对齐，即左对齐方式。

CAN总线验收滤波器和验收屏蔽器均对于CAN接收而言。注：当AMR为FF FF FF FF 时，表示屏蔽ACR的所有滤波位，即可以接收所有的信息。

注：关于ID格式的详细说明请参见：《*8.附件1：ID对齐方式.pdf*》说明文档。

示例：

(1).标准帧举例：若要接收标准帧，则滤波方式需要选择为“接收所有类型”或“只接收标准帧”，ACR=任意值，AMR= 0xFFFFFFFF，适配器能接收任意ID的CAN消息；ACR=0x6F000000,AMR=0x00FFFFFF,适配器可接收ID为0x378到0x37F的帧。

(2).扩展帧举例：若要接收扩展帧，则滤波方式需要选择为“接收所有类型”或“只接收扩展帧”，ACR=任意值，AMR= 0xFFFFFFFF，适配器能接收任意ID的CAN消息；ACR=0x00001BC0，AMR=0x0000003F，则可接收ID=0x00000378到0x0000037F的CAN消息帧。

设置过滤验收滤波器时，VCI_SetReference2函数的RefType参数须传递2；设置过滤屏蔽寄存器时，VCI_SetReference2函数的RefType参数须传递6，具体应用可参考本小节后的示例

代码。

示例4_设置滤波寄存器：

```
#include "ControlCan.h"

int   nDeviceType = 4;        /*USB-CAN2.0 */

int   nDeviceInd = 0;         /* 第0个设备 */

int   nCANInd = 0;

DWORD dwRel;

BYTE refType = 2;             /* 设置过滤验收寄存器 */

BYTE pData[4] ={0};

/* 将验收滤波器分成4个字节设置，设置其值为：0x00001BC0 */

pData[0] = 0x00;              /* 验收滤波器最高字节 */

pData[1] = 0x00;              /* 验收滤波器次高字节 */

pData[2] = 0x1B;              /* 验收滤波器次低字节 */

pData[3] = 0XC0;              /* 验收滤波器最低字节 */

bRel = VCI_SetReference2(nDeviceType, nDeviceInd, nCANInd, refType,pData);

if(dwRel != 1)

{

   MessageBox(_T("设置验收滤波器失败!"), _T("警告"), MB_OK|MB_ICONQUESTION);

   return FALSE;

}

refType = 6;     /* 设置过滤屏蔽寄存器 */

/* 将屏蔽滤波器分成4个字节设置，设置其值为：0x0000003F*/

pData[0] = 0x00;              /* 屏蔽滤波器最高字节 */

pData[1] = 0x00;              /* 屏蔽滤波器次高字节 */

pData[2] = 0x00;              /* 屏蔽滤波器次低字节 */

pData[3] = 0x3F;              /* 屏蔽滤波器最低字节 */

bRel = VCI_SetReference2(nDeviceType, nDeviceInd, nCANInd, refType,pData);

if(dwRel != 1)

{
```

```
    MessageBox(_T("设置屏蔽滤波器失败!"), _T("警告"), MB_OK|MB_ICONQUESTION);
    return FALSE;
}
```

4.2 滤波模式

USB-CAN适配器提供3种滤波模式：接收所有类型、只接收标准帧、只接收扩展帧。下表为滤波模式取值对照表：

值	名称	说明
1	接收所有类型	滤波器允许接收标准帧和扩展帧。
2	只接收标准帧	滤波器允许接收标准帧，扩展帧将不能通过。
3	只接收扩展帧	滤波器允许接收扩展帧，标准帧将不能通过。

在设置滤波模式时，SetReference2函数的RefType参数须传递1，具体应用可参考本小节后的示例代码。

示例5_设置滤波模式：

```
#include "ControlCan.h"
int   nDeviceType = 4;        /*USB-CAN2.0 */
int   nDeviceInd = 0;         /*  第0个设备  */
int   nCANInd = 0;
BYTE refType = 1;             /*  设置滤波模式  */
DWORD dwRel;
BYTE pData[1] ={0};
pData[0] = 1;                 /*  接收所有类型  */
bRel = VCI_SetReference2(nDeviceType, nDeviceInd, nCANInd, refType,pData);
if(dwRel != 1)
{
    MessageBox(_T("设置滤波模式失败!"), _T("警告"), MB_OK|MB_ICONQUESTION);
    return FALSE;
}
```

注：关于更为详细的滤波设置相关说明，请参照：《*附件4：CAN滤波设置*》说明文档。