应用报告
ZHCA475–September 2011

在 Hercules™ ARM® 安全 MCU 上配置一个 CAN 节点

Pratip Kumar

摘要

德州仪器 (TI) 生产的 Hercules 系列微控制器是系列 32 位 RISC 微控制器，此微控制器具有高级 ARM 架构和一个丰富的外设集。

这个应用报告描述了一个配置 CAN 节点以及 在网络上执行 CAN 通信的典型步骤。

内容

图片列表

1 简介

1.1 控制器局域网络 (CAN)

控制器局域网是一个串行多主控通信协议， 此协议高效支持分布式实时控制，具有高级安全级别，和一个高达 1Mbps 的通信速率。

CAN 总线是嘈杂和恶劣的环境，诸如 汽车和其他要求可靠通信的工业领域应用的理想选择。

外设支持一下 特性

- 协议：
  - 支持 CAN 协议 版本 2.0 部分 A.B
- 时钟和 速度：
  - 高达 1M 位每秒的比特率
  - 双时钟 源
- 消息目标/消息 对话框：
  - 16，32，64 或 128 消息目标（器件 专用）
  - 针对每个消息 目标的独立标识符掩码
  - 针对消息 目标的可编程先进先出 (FIFO) 模式
- 中断：
  - 到中断模块的两条 中断线路
- 低功耗 支持：
  - 全局断电和唤醒支持
  - 本地 断电和唤醒支持
- 突发传输 支持
  - 通过 DMA 的数据传输
- 调试 支持：
  - 针对自检 运行的可编程回路模式
  - 测试 期间到消息 RAM 的直接访问
  - 针对调试 支持的挂起模式
- 其它：
  - 由一个 32 位定时器实现的总线关闭后的自动总线 打开
  - CAN Rx/Tx 引脚 可配置为通用输入输出 (IO) 引脚

1.2 CAN 收发器接口图

图 1显示了一个典型的 CAN 收发器接口。

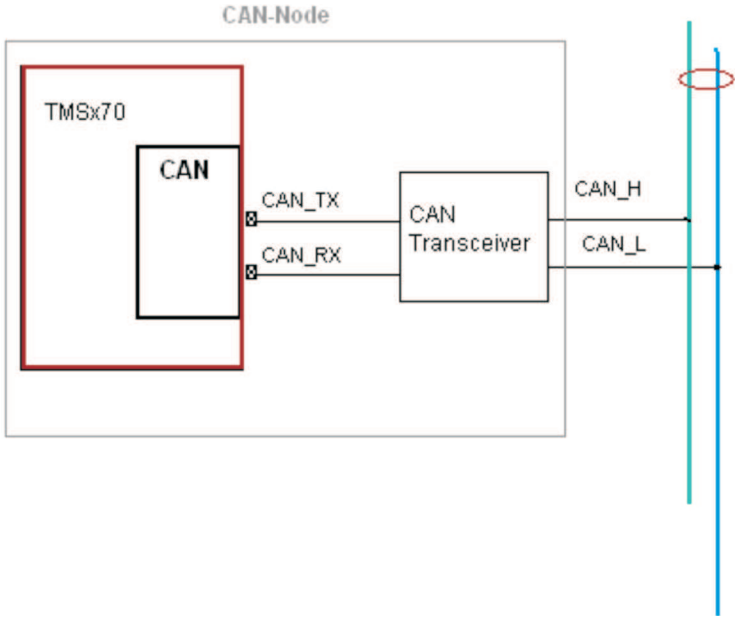

图 1. CAN 收发器接口

1.3 典型 CAN 网络

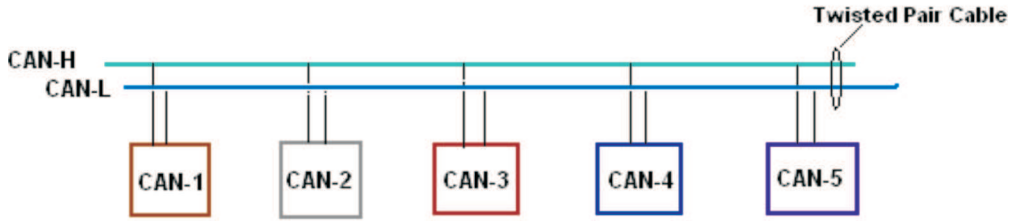

图 2. 一个典型的 CAN 网络

1.4 CAN 方框图

图 3显示了 CAN 控制器的核心内部块。

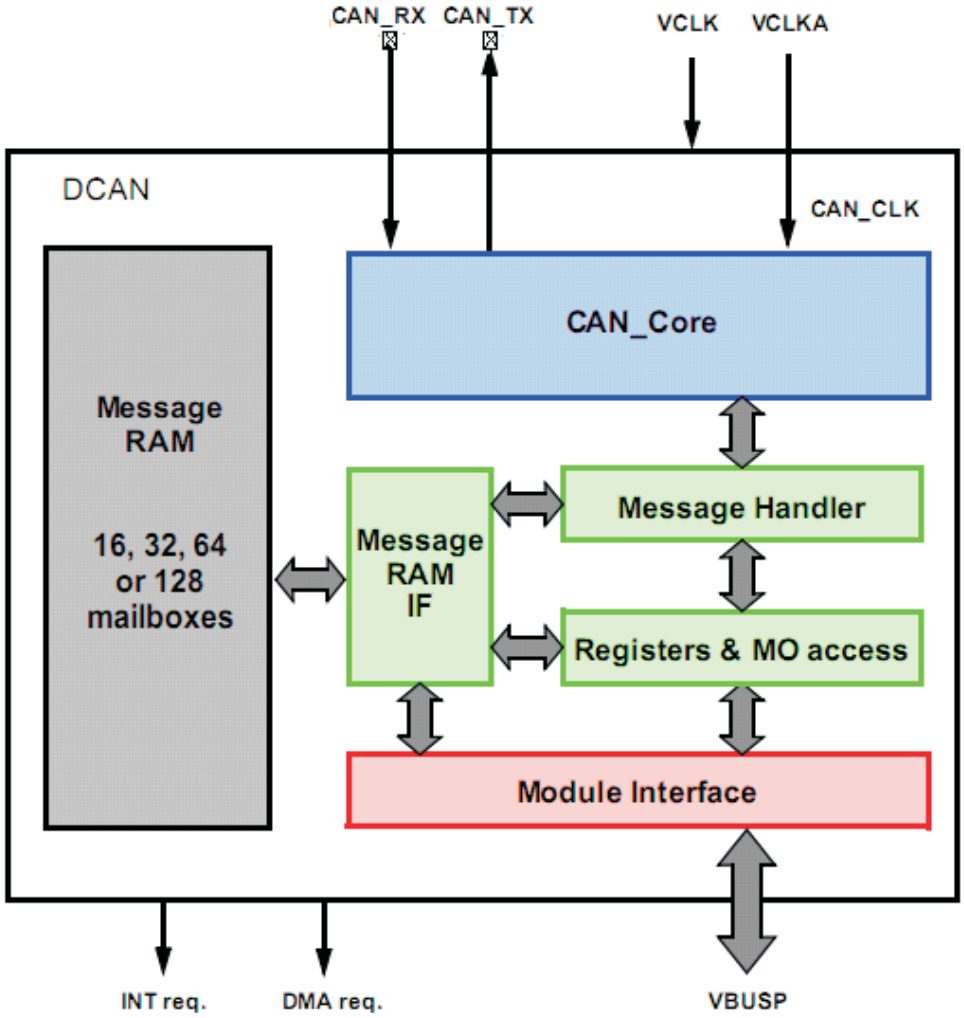

图 3. CAN 模块方框图

2 初始化 CAN 外设

下面常见的步骤涉及初始化 CAN。

- CAN RAM 初始化（建议在引 导期间使用）
- CAN 配置

- CAN 波特率 配置

初始化 CAN 之后，邮箱需要按照 应用的要求配置。

下面的小节 详细介绍了 CAN 初始化过程。

2.1 CAN RAM 初始化

CAN RAM 保存 CAN 消息目标（也被成为邮箱）。 要开始初始化，通过配置系统寄存器，RAM 应该由硬件被初始化为零。

为了初始化 CAN RAM：

1. 切换到存储器初始化模式： MINITGCR.MINITGNA = 0xA
2. 设定 MSINENA。 相对于 DCAN RAM 的存储器初始化使能寄存器 (MSIENAx) 位
3. 通过检查存储器硬件初始化状态寄存器 (MINISTAT)，等待存储器初始化 完成。
4. 从存储器初始化模式中退出。

初始化 RAM 的代码示例：

```
SYS_Ptr->MINITGCR_UN.MINITGCR_UL = 0xA;
SYS_Ptr->MSINENA_UL              = MEM_CH_DCANRAM;
while (SYS_Ptr->MINISTAT_UL == 0);
SYS_Ptr->MINITGCR_UN.MINITGCR_UL = 0x5;
```

注： 对于 MEM_CH_DCANRAM，请参见器件专用数据表中的*存储器 初始化*表。

如果您计划使用奇偶校验或 ECC，您需要在 RAM 初始化过程之前启用它。

下面是启用奇偶校验 / ECC 的示例代码：

```
DCAN1_Ptr->CCR_UN.CCR_ST.PMD_B4 = 0xA
```

2.2 CAN 配置

通过软件可配置 CAN 的以下特性：

- 禁用/启用总线活动时的自动唤醒。 [位： CCR.WUBA - 总线活动时的自动唤醒]
- 启用/禁用自动总线打开定时器。 [位：CCR.ABO - 自动总线打开]
- 启用/禁用奇偶校验 / ECC。 [位：CCR.PMD - 奇偶校验模式 位]
- 启用/禁用全局中断线路 0 和 1。[位：CCR.IEx - 中断 使能]
- 启用/禁用错误中断。 [位：CCR.EIE - 错误中断 使能]
- 禁用/启用状态中断。 [位：CCR.SIE - 状态中断 使能]
- 设置消息的自动重传。 [位：DAR - 禁用自动 重传]
- 请求到配置寄存器的写入权限。 [位：CCR.CCE - 配置改变使能]
- 进入调试状态前，设置消息完成。 [位：CCR.IDS]
- 禁用/启用本地断电模式。 [位：CCR.PDR]
- 禁用/启用 DMA 请求线路。 [位：DEx]
- 启用/禁用测试模式。 [位：CCR.Test]

示例 CAN 配置代码：

```
Void CanInit (void
{
     /**@b Initializ @b DCAN1:   */

     canREG1->CTL = 0x00021443U; /* Configure CAN */

     /** - Clear all pending error flags and reset current status */
     canREG1->ES = ox0000031Fu;

     /** - Assign interrupt level for messages */
     canREG1->INTMUXx[0U] = MessageBoxNo;
     canREG1->INTMUXx[1U] = 0x00000000U;

     /** - Setup auto bus on timer period */
     canREG1->ABOTR = 0U;
}
```

2.3 配置 CAN 波特率

通过计算位时序值并且 将这个值编辑进位时序寄存器 (BTR) 来设定 CAN 数据传输波特率。

要初始化 CAN 波特率：

1. 设定 CANCONTROL.init 位。 这将把 CAN 置于初始化 模式中。
2. 设定配置改变使能 (CANCONTROL.CCE) 位。 这将 启用到 BTR 寄存器的写入访问。
3. 将经计算的位时序 元件值写入 BTR 寄存器。
4. 清除 CANCONTROL.init 位之前的 CANCONTROL.CCE 位。

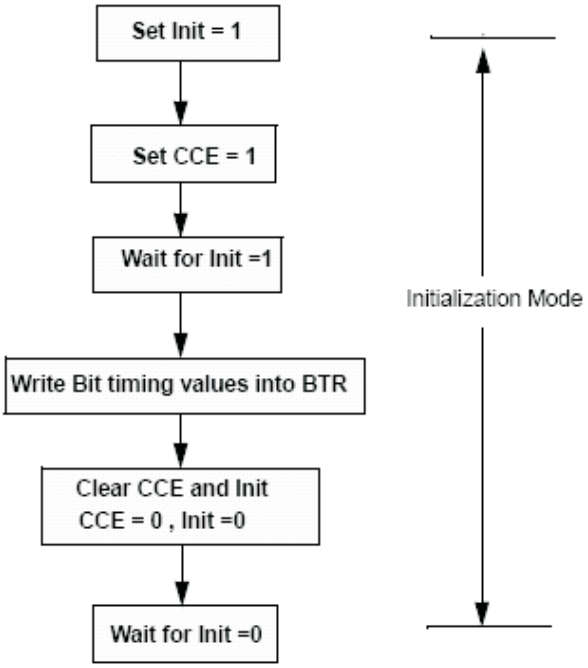

图 **4. CAN** 波特率配置

```
/* Baudrate configuration */
DCAN1_Ptr->CCR_UN.CCR_ST.INIT_B1 = 1 ;         /* Enter Initialization Mode */
DCAN1_Ptr->CCR_UN.CCR_ST.CCE.B1  = 1 ;

while(DCAN1_Ptr->CCR_UN.CCR_ST.INIT_B1 == 0) ;

/* With CAN clock (CAN_CLK) of 8 MHz and BRPE = 0x00,
   the BRT value of 0x2301 configures the CAN
   for a bit rate of 500kBits/s.                                    */

DCAN1_Ptr->CCR_UN.BTR_UL          = 0x2301 ;

DCAN1_Ptr->CCR_UN.CCR_ST.CCE.B1  = 0 ;
DCAN1_Ptr->CCR_UN.CCR_ST.INIT_B1 = 0 ;
while(DCAN1_Ptr->CCR_UN.CCR_ST.INIT_B1 == 1) ; /* Enter Normal Mode         */
```

2.4 CAN 消息目标

表 1中显示了 CAN 消息目标（也称为 CAN 邮箱）的结构。

表 1. CAN 消息目标

消息 目标												
UMask	Msk[28:0]	MXtd	MDir	EoB	未使用	NewDat	MsgLst	RxIE	TxIE	IntPnd	RmtEn	TxRqst
MsgVal	ID[28:0]	Xtd	Dir	DLC[3:0]	数据 0	数据 1	数据 2	数据 3	数据 4	数据 5	数据 6	数据 7

表 1 有以下三个关键组件：

- ID：消息 ID（具有扩展的 消息 ID，Xtd）
- DLC：消息长度
- Datax：消息数据 （高达 8 字节）

此外，表 1有下列控制位：

- 消息有效 **(MsgVal)**：表示消息 有效。
- 方向**(Dir)**：表示 邮箱是发送还是接收
- 标识符屏蔽**(Msk)**：表示 ID 中可被屏蔽的位
- 屏蔽扩展标识符**(MXtd)**：表示 扩展 ID Xtd 的屏蔽位
- 屏蔽消息方向**(MDir)**：表示 Dir 是否应该被 屏蔽。
- 使用接受屏蔽**(UMask)**：表示 屏蔽位是否将被使用。
- 块末尾**(EoB)**：表示 FIFO 缓冲器的最后一条消息
- 发送中断使能**(TxIE)**： 提供一个数据传输后的中断。
- 接收中断使能 **(RxIE)**：提供一个 数据接收后的中断。
- 中断等待**(IntPnd)**：表示 中断在等待这个消息目标。
- 新数据 **(NewDat)**：表示新数据在消息 目标中可用。
- 时序请求**(TxRqst)**：请求 数据传输
- 远程使能**(RmtEn)**：启用消息目标接收远程 帧。

2.5 CAN 接口寄存器

消息目标只能通过接口寄存器 (IFx) 访问。 这 是为了避免当 CPU 和 DMA 都尝试访问消息目标时仲裁的发生。 在每个 IFx 访问期间，消息目标中所选的项目可按照需要更新。 一次只能访问一个消息 目标（针对读取/写入）。 总共有三个接口 寄存器。 接口寄存器 (IFx) 运行期间，相应的繁忙位保持 高电平。

注: 要获得支持的 CAN 邮箱数量的细节，请参阅器件专用数据 表。

消息目标 1 的优先级最高，而最后执行的消息目标 具有优先级最低。 如果有多于一个传输请求在等待，它们被按照 优先级被处理。

接口寄存器 IF1 和 IF2 是完全一样的。

Message Interface Register Sets 1 and 2

Address [CAN Base +]	IF1 Register Set 31 16 15 0	Address [CAN Base +]	IF2 Register Set 31 16 15 0
0x100	IF1 Command	0x120	IF2 Command
0x104	IF1 Mask	0x124	IF2 Mask
0x108	IF1 Arbitration	0x128	IF2 Arbitration
0x10C	IF1 Message Control	0x12C	IF2 Message Control
0x110	IF1 Data A	0x130	IF2 Data A
0x114	IF1 Data B	0x134	IF2 Data B

图 **5. IF1/2** 寄存器位

命令寄存器指定数据传输的方向（来自/到消息 目标）以及传输哪部分消息目标并且选择一个消息 RAM 中的消息 目标作为传输的目标或者源。

仲裁寄存器用消息 ID 来启用/禁用消息目标并且 将其配置为发送或接收。

消息控制寄存器定义数据尺寸和其它配置， 这些配置在消息目标初始化期间完成。

数据寄存器保存将被传输或 被接收的数据。

每个消息目标可按照接口 寄存器的要求进行配置。

IF3 寄存器集可使用接收到的消息目标进行自动更新， 而无需初始化来自消息 RAM 的传输。 可针对每一个消息目标 独立编辑自动更新功能。

Message Interface Register Set 3

Address [CAN Base +]	IF1 Register Set 31 16 15 0
0x100	IF3 Observation
0x104	IF3 Mask (Read Only)
0x108	IF3 Arbitration (Read Only)
0x10C	IF3 Message Control (Read Only)
0x110	IF3 Data A (Read Only)
0x114	IF3 Data B (Read Only)

图 **6. IF3** 寄存器

3 配置字消息目标

基本上，一个消息目标可被配置 为：

- 发送消息对象
- 接收消息 目标

图 7是一个用于配置一个 CAN 消息目标的逻辑图。

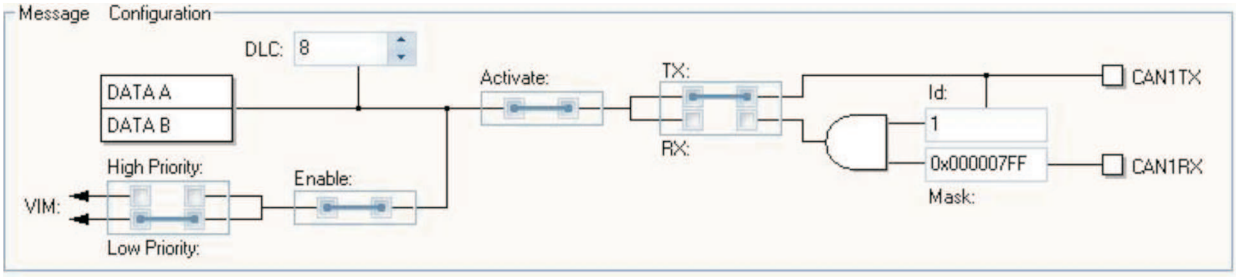

图 7. CAN 消息目标配置-逻辑图

必须通过接口寄存器 (IFx) 来完成消息目标的配置。

配置一个消息目标所需的最少步骤为：

1. 等待 IFx 可被使用。
2. 设定消息 屏蔽。
3. 设定消息仲裁。
4. 设定消息控制 字。
5. 设定 IFx 控制字节。
6. 设定 IFx 消息 数量。

通过两个 方法中的任何一个可执行 CAN 通信：

- 中断方法
- 轮询 方法

3.1 将 CAN 消息目标配置为 发送

下面是一个配置一个传输消息目标的示例代码：

```
while (canREG1->IF1STAT & 0x80) ;

canREG1->IF1MSK  = 0xC0000000U | ((0x000007FFU & 0x1FFFFFFFU0 << 0U) ;
canREG1->IF1ARB  = 0x80000000U | 0x40000000U | 0x20000000U | ((1U & ox1FFFFFFFU) << 0U) ;
canREG1->IF1MCTL = 0x00001080U | 0x00000C00U | 8U ;
canREG1->IF1CMD  = 0x20 ;
canREG1->IF1NO   = 1 ;
```

示例代码解释：

1. 如果 IF1 忙则等待。
2. 配置屏蔽 寄存器。

MXtd	=	1	使用扩展 ID 屏蔽
MDir	=	1	使用消息方向屏蔽
Msk	=	0x7FF	用位 10:0 来过滤

3. 配置仲裁 寄存器

MsgVal	=	1	启用消息目标
Xtd	=	1	扩展的 25 位标识符
Dir	=	1	发送邮箱
ID		1	消息 ID 0x1

4. 配置消息控制 寄存器

UMask	=	1	使用屏蔽来过滤
EoB	=	1	单一消息目标
TxIE	=	1	启用传输中断
RxIE	=	1	启用接收中断
DLC	=	8	将数据长度设定为 8

5. 配置命令 寄存器

消息号	=	0x20	消息号位 0x20

3.2 将 CAN 消息目标配置为 接收

下面是配置一个接收消息目标的示例代码：

```
while (canREG1->IF2STAT & 0x80) ;

canREG1->IF2MSK  = 0xC0000000U | ((0x000007FFU & 0x1FFFFFFFU0 << 0U) ;
canREG1->IF2ARB  = 0x80000000U | 0x40000000U | 0x00000000U | ((2U & ox1FFFFFFFU) << 0U) ;
canREG1->IF2MCTL = 0x00001080U | 0x00000C00U | 8U ;
canREG1->IF2CMD  = 0x22 ;
canREG1->IF2NO   = 2 ;
```

示例代码解释

1. 如果 IF2 忙则等待。
2. 配置屏蔽 寄存器。

MXtd	=	1	使用扩展 ID 屏蔽
MDir	=	1	使用消息方向屏蔽
Msk	=	0x7FF	使用位 10:0 来过滤

3. 配置仲裁 寄存器。

MsgVal	=	1	启用消息目标
Xtd	=	1	扩展 25 位标识符
Dir	=	0	接收邮箱
ID		2	消息 ID 0x2

4. 配置消息控制 寄存器

UMask	=	1	使用屏蔽来过滤
EoB	=	1	单一消息目标
TxIE	=	1	启用发送中断
RxIE	=	1	启用接收中断
DLC	=	8	将数据长度设定为 8

5. 配置命令 寄存器

消息号	=	0x22	消息号为 0x22

3.3 CAN 发送/接收操作

```
unsigned canTransmit (canBASE_t *node, unsigned messageBox, const unsigned char *data)
{
    unsigned I;
    unsigned success  = 0U;
    unsigned regIndex = (messageBox - 1U) >> 5U;
    unsigned bitIndex = 1U << ((messageBox - 1U) & 0x1FU);

    /** - Check for pending message:
    *     - pending message, return 0
    *     - no pending message, start new transmission
    */
    if (node->TXRQx[regIndex] & bitIndex)
    {
       return success;
    }

    /** - Wait until IF1 is ready for use */
    while (node->IF1STAT & 0x80);

    /** - Copy TX data into IF1 */
    for (I = 0U < 8U; I++)
    {
        node->IF1DATx[s_canByteOrder[i]] = *data++;
    }

    /** - Copy TX data into message box */
    node->IF1NO = messageBox;

    success = 1U;

    /**   @note The function canINIT has to be called before this function can be used. \n
    *           The user is responsible to initialize the message box.
    */
    return success;
}
```

下面是一个接收 CAN 数据的示例代码：

```
unsigned canGetData (canBASE_t *node, unsigned messageBox, unsigned char * const data)
{
    unsigned        I;
    unsigned        size;
    unsigned char   *pData   = (unsigned char *) data;
    unsigned        success  = 0U;
    unsigned        regIndex = (messageBox - 1U) >> 5U;
    unsigned        bitIndex = 1U << ((messageBox - 1U) & 0x1FU);

    /** - Check if new data has arrived:
    *     - no new data, return 0
    *     - new data, get received message
    */
    if (!(node->NWDATx[regIndex] & bitIndex))
    {
       return success;
    }

    /** - Wait until IF2 is ready for use */
    while (node->IF2STAT & 0x80);

    /** - Copy data into IF2 */
    node->IF2NO = messageBox

    /** - Wait until data are copied into IF2 */
    while (node->IF2STAT & 0x80);

    /** - Get number of received bytes */
    size = node->IF2MCTL & 0xFU;


    /** - Copy RX data into destination buffer */
    for (I = 0U; I < size; I++)
    {
         *pData++ = node->IF2DATx[s_canByteOrder[i]];
    }

    success = 1U;

    /** - Check if data has been lost:
    *     - no data lost, return 1
    *     - data lost, return 3
    */
    if (node->IF2MCTL & 0x4000U)
    {
        success = 3U;
    }

    return success;
}
```

图 8说明了处理接收中断的流程。

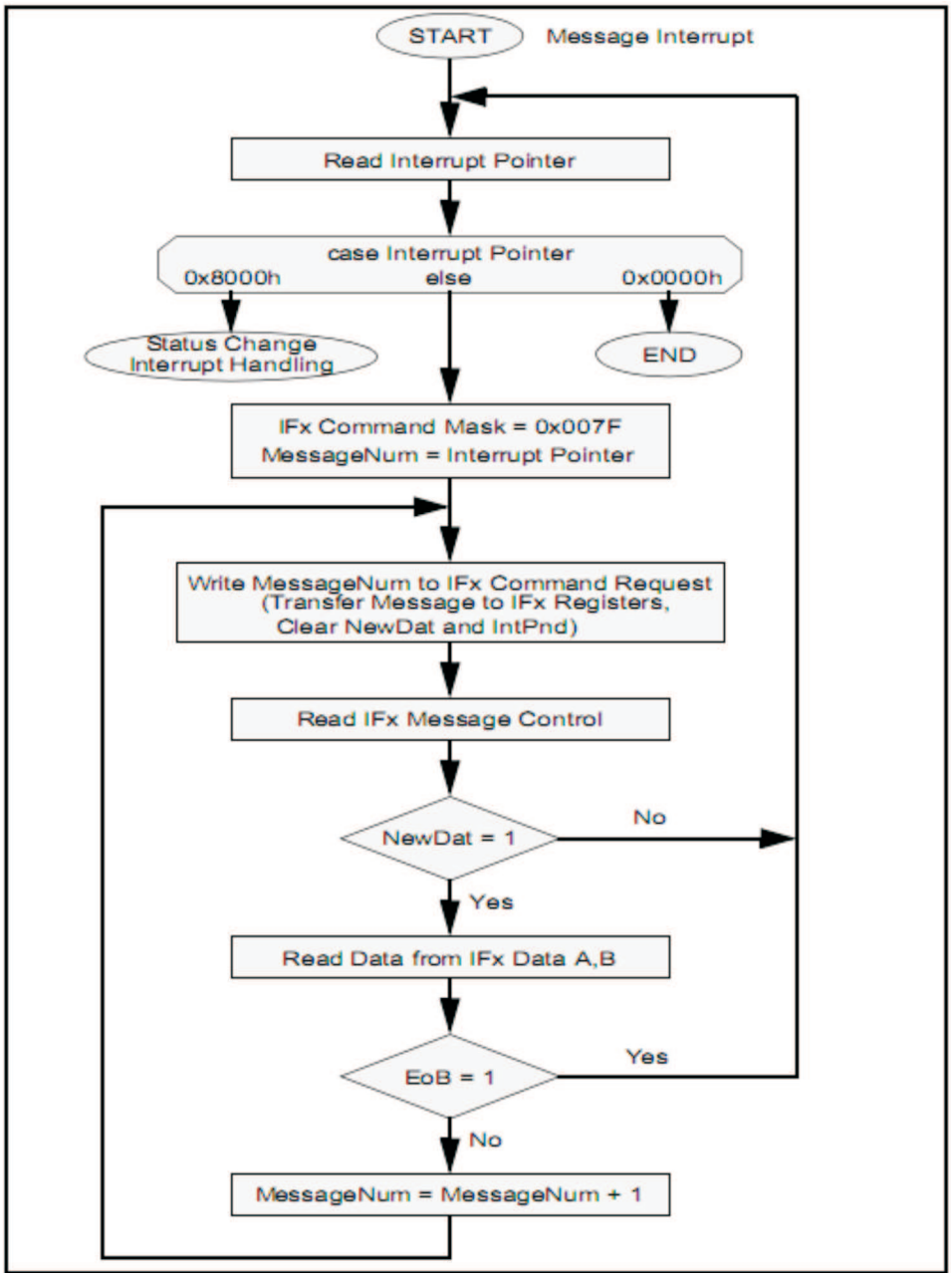

图 **8.** 处理 **CAN** 消息接收

4 软件流

这一小节显示了针对一个 CAN 节点的软件流汇总。

1. 针对消息 目标来初始化 CAN RAM 空间。
2. 配置 CAN 通用参数（DMA，中断，自动给唤醒， 等）。
3. 配置所需的 CAN 波特率。
4. 用所需的 ID 和屏蔽来配置 需要的消息目标。
5. 根据中断模式或轮询 模式提供 发送/接收例程。

附录 A　　采样位 时序

图 9和图 10说明了针对一个 500 和 1000 比特率计算的采样位时序， 此时序被编辑进位时序寄存器。

针对位速率**= 500**的采样位时序计算

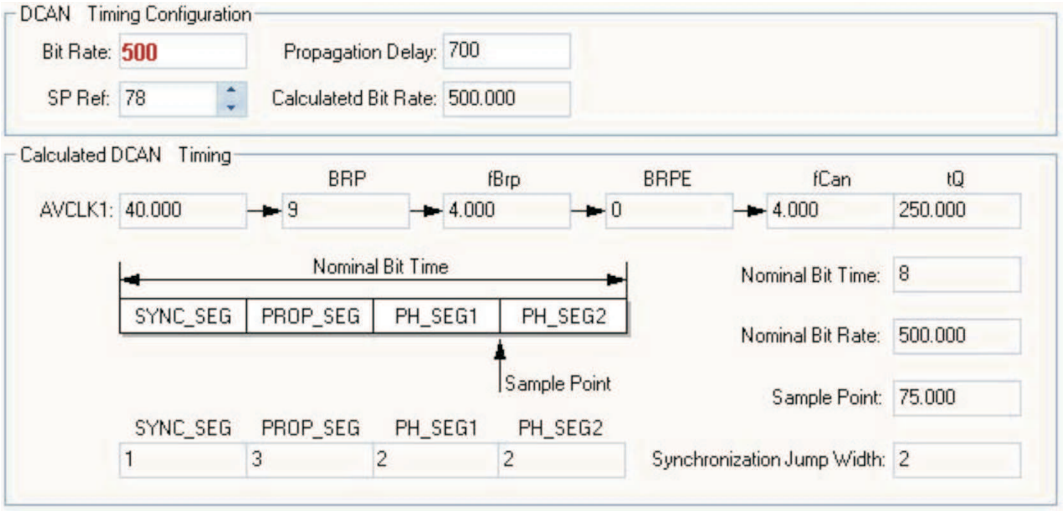

图 **9.** 针对 **500kb** 的采样位时序计算

位时序寄存器值：

BRPE=(0U<<16U)

TSEG2：PHASE SEG2=((2U-1U)<<12U)

TSEG1：PROP SEG+PHASE SEG1=(((3U+2U)-1U)<<8U)

SJW：((2U-1U)<<6U)

BRP=9U；

针对位速率**= 1000**的采样位时序计算

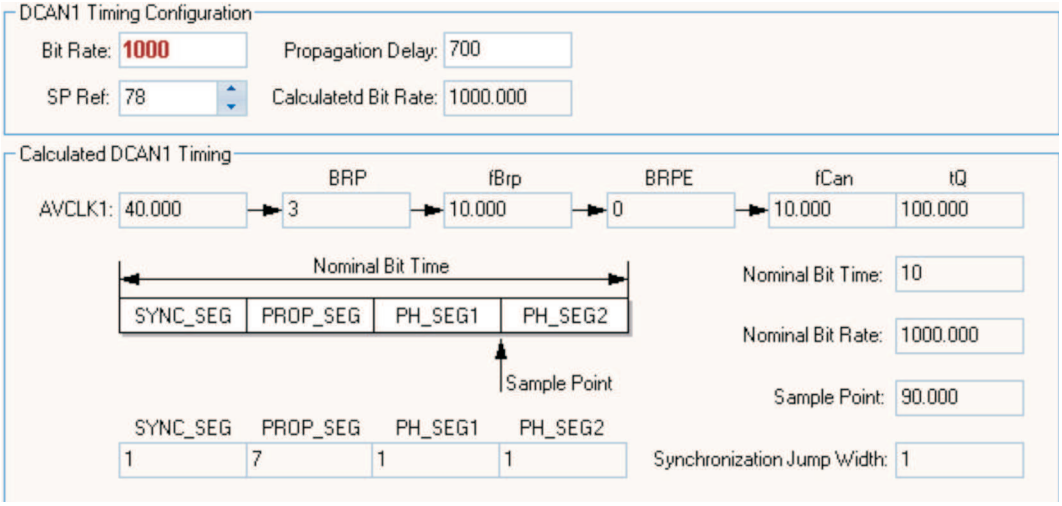

图 **10.** 针对 **1000kb** 的采样位时序计算

位时序寄存器值：

BRPE=(0U<<16U)

TSEG2：PHASE SEG2=((1U-1U)<<12U)

TSEG1: PROP SEG+PHASE SEG1=(((7U+1U)-1U) << 8U)

SJW：((1U-1U)<<6U)

BRP=3U；

重要声明

德州仪器(TI) 及其下属子公司有权在不事先通知的情况下, 随时对所提供的产品和服务进行更正、修改、增强、改进或其它更改，并有权随时中止提供任何产品和服务。客户在下订单前应获取最新的相关信息，并验证这些信息是否完整且是最新的。所有产品的销售都遵循在订单确认时所提供的TI 销售条款与条件。

TI 保证其所销售的硬件产品的性能符合TI 标准保修的适用规范。仅在TI 保证的范围内，且TI 认为有必要时才会使用测试或其它质量控制技术。除非政府做出了硬性规定，否则没有必要对每种产品的所有参数进行测试。

TI 对应用帮助或客户产品设计不承担任何义务。客户应对其使用TI 组件的产品和应用自行负责。为尽量减小与客户产品和应用相关的风险，客户应提供充分的设计与操作安全措施。

TI 不对任何TI 专利权、版权、屏蔽作品权或其它与使用了TI 产品或服务的组合设备、机器、流程相关的TI 知识产权中授予的直接或隐含权限作出任何保证或解释。TI 所发布的与第三方产品或服务有关的信息，不能构成从TI 获得使用这些产品或服务的许可、授权、或认可。使用此类信息可能需要获得第三方的专利权或其它知识产权方面的许可，或是TI 的专利权或其它知识产权方面的许可。

对于TI 的产品手册或数据表，仅在没有对内容进行任何篡改且带有相关授权、条件、限制和声明的情况下才允许进行复制。在复制信息的过程中对内容的篡改属于非法的、欺诈性商业行为。TI 对此类篡改过的文件不承担任何责任。

在转售TI 产品或服务时，如果存在对产品或服务参数的虚假陈述，则会失去相关TI 产品或服务的明示或暗示授权，且这是非法的、欺诈性商业行为。TI 对此类虚假陈述不承担任何责任。

TI 产品未获得用于关键的安全应用中的授权，例如生命支持应用（在该类应用中一旦TI 产品故障将预计造成重大的人员伤亡），除非各方官员已经达成了专门管控此类使用的协议。购买者的购买行为即表示，他们具备有关其应用安全以及规章衍生所需的所有专业技术和知识，并且认可和同意，尽管任何应用相关信息或支持仍可能由TI 提供，但他们将独力负责满足在关键安全应用中使用其产 品及TI 产品所需的所有法律、法规和安全相关要求。此外，购买者必须全额赔偿因在此类关键安全应用中使用TI 产品而对TI 及其 代表造成的损失。

TI 产品并非设计或专门用于军事/航空应用，以及环境方面的产品，除非TI 特别注明该产品属于“军用”或“增强型塑料”产品。只 有TI 指定的军用产品才满足军用规格。购买者认可并同意，对TI 未指定军用的产品进行军事方面的应用，风险由购买者单独承担，并且独力负责在此类相关使用中满足所有法律和法规要求。

TI 产品并非设计或专门用于汽车应用以及环境方面的产品，除非TI 特别注明该产品符合ISO/TS 16949 要求。购买者认可并同意，如果他们在汽车应用中使用任何未被指定的产品，TI 对未能满足应用所需要求不承担任何责任。

可访问以下URL 地址以获取有关其它TI 产品和应用解决方案的信息：

	产品		应用
数字音频	www.ti.com.cn/audio	通信与电信	www.ti.com.cn/telecom
放大器和线性器件	www.ti.com.cn/amplifiers	计算机及周边	www.ti.com.cn/computer
数据转换器	www.ti.com.cn/dataconverters	消费电子	www.ti.com/consumer-apps
DLP® 产品	www.dlp.com	能源	www.ti.com/energy
DSP - 数字信号处理器	www.ti.com.cn/dsp	工业应用	www.ti.com.cn/industrial
时钟和计时器	www.ti.com.cn/clockandtimers	医疗电子	www.ti.com.cn/medical
接口	www.ti.com.cn/interface	安防应用	www.ti.com.cn/security
逻辑	www.ti.com.cn/logic	汽车电子	www.ti.com.cn/automotive
电源管理	www.ti.com.cn/power	视频和影像	www.ti.com.cn/video
微控制器 (MCU)	www.ti.com.cn/microcontrollers		
RFID 系统	www.ti.com.cn/rfidsys		
OMAP 机动性处理器	www.ti.com/omap		
无线连通性	www.ti.com.cn/wirelessconnectivity		
	德州仪器在线技术支持社区	www.deyisupport.com	

邮寄地址： 上海市浦东新区世纪大道 1568 号，中建大厦 32 楼 邮政编码： 200122