请注意：本文档不再进行更新。本文档可能包含旧内容和过时的商标。

请参考英文版本以获取最新更新

https://www.intel.com/content/www/us/en/programmable/documentation/lit-index.html

Please take note that this document is no longer being maintained. It may contain legacy content and trademarks which may be outdated.

Please refer to English version for latest update at

https://www.intel.com/content/www/us/en/programmable/documentation/lit-index.html

25. CAN 控制器

cv_54025-1.2

硬核处理器系统 (HPS) 提供两个控制器局域网 (CAN) 控制器，以便通过使用 CAN 协议与 Cortex™-A9 微处理器单元 (MPU) 子系统主机处理器和直接存储访问 (DMA) 控制器进行串行通信。CAN 控制器是 Bosch® D_CAN 控制器的实例并且与 ISO 11898-1 兼容。

CAN 控制器的功能

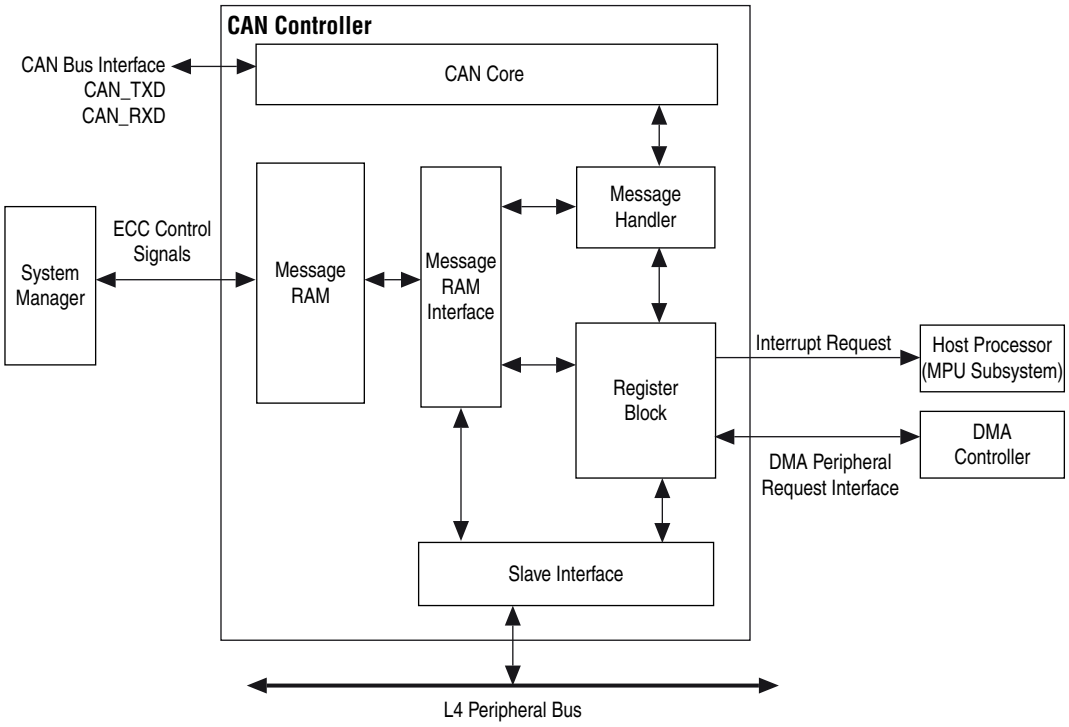
CAN 控制器提供以下功能：

- 与 Bosch 网站 (www.semiconductors.bosch.de) 上的 *CAN Protocol Specification 2.0* 的 A 和 B 兼容。
- 高达 1 Mbps 的可编程通信速率
- 保持高达 128 个消息
- 错误纠正代码 (ECC)
- 11-bit 标准和 29-bit 扩展标识符
- 可编程的环回模式
- 用于较大数据传输的外部直接存储访问 (DMA) 控制器
- 自动重新传输

CAN 控制器结构图和系统集成

图 25 – 1 显示了 CAN 控制器结构图。

图 25 – 1. CAN 控制器结构图



CAN 控制器包含以下模块和接口：

- CAN 内核
  - 连接到 CAN 总线接口
  - 处理所有的 ISO 11898-1 协议功能
- 消息处理器
  - 控制消息 RAM 和 CAN 内核之间数据传输的状态机。
  - 处理接受消息过滤和产生中断
- 消息 RAM
  - 存储高达 128 条消息对象
  - 单位错误纠正和双位错误检测
- 消息 RAM 接口
  - 两个单独的接口，IF1 和 IF2
- 寄存器模块
  - 用于模块设置和间接消息对象访问的控制和状态寄存器 (CSR)
  - 所有主机处理器都通过消息 RAM 接口访问消息 RAM。
- 用于 CSR 访问的 Level 4 (L4) 从接口

CAN 控制器的功能说明

CAN 控制器根据 CAN 协议版本 2.0 的 A 和 B 执行通信。CAN 总线上的所有通信都通过消息对象进行。CAN 控制器在其内部消息 RAM 中存储消息对象。主机处理器不可以直接访问消息 RAM，而 IF1 和 IF2 消息接口寄存器组提供主机处理器对消息的访问。消息处理器将消息在消息 RAM 和 CAN 内核之间传递。消息处理器也负责消息层任务，例如接受消息过滤、产生中断和发送请求生成。

消息对象

消息 RAM 可以存储高达 128 个消息对象。要避免主机处理器访问消息 RAM 以及 CAN 消息接收和传输之间的冲突，主机处理器不直接访问消息对象。访问通过 IF1 和 IF2 消息接口寄存器处理。

表 25 – 1 显示了消息对象的结构。第一行包含消息对象控制标记，第二行包含消息对象屏蔽，第三行是 CAN 消息。

表 25 – 1. 消息对象结构

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

消息对象控制标记

该部分介绍消息对象控制标记。

有效消息 (MsgVal)

- 0= 消息对象由消息处理器忽略
- 1= 消息对象被配置并且应由消息处理器考虑

主机处理器复位 CAN 控制寄存器 (CCTRL) 的初始化位 (Init) 以便初始化 CAN 控制器之前，主机处理器必须设置所有未使用的消息对象的 MsgVal 位为 0。当不再使用消息对象时，也必须设置 MsgVal 为 0。

MsgVal 域从消息有效寄存器 (MOVALA、MOVALB、MOVALC、MOVALD 和 MOVALX) 直接可读。然而，要写入指定的消息对象的 MsgVal 域，主机处理器必须写入消息接口寄存器。

新数据 (NewDat)

- 0= 由于上次主机处理器清除该标记，所以信息处理器没有写入新数据到该信息对象的数据部分。
- 1= 信息处理器或主机处理器已经写入新数据到该信息对象的数据部分。

NewDat 域从新数据寄存器 (MONDA、MONDB、MONDC、MONDD 和 MONDX) 直接可读。然而，要写入指定消息对象的 NewDat 域，主机处理器必须写入消息接口寄存器。

消息丢失 (MsgLst)

- 0= 因为该位由主机处理器复位，所以从上次开始没有消息丢失。
- 1=当NewDat位仍然被设置时消息处理器将新消息存储到该对象,表明主机处理器已经丢失消息。

只有当消息方向位 (Dir) 设置为接收时，消息对象中的 MsgLst 才有效。

中断待定 (IntPnd)

- 0= 该消息对象不是中断源。
- 1= 该消息对象是中断源。如果没有其它较高优先权的中断源，那么 CAN 中断寄存器 (CIR) 中的中断标识符域指向该消息对象。

IntPnd 域从中断待定寄存器 (MOIPA、MOIPB、MOIPC、MOIPD 和 MOIPX) 直接可读。然而，要写入指定消息对象的 IntPnd 域，主机处理器必须写入消息接口寄存器。

发送中断使能 (TxIE)

- 0= TxIE 被禁用。成功传输帧后，IntPnd 保持不变。
- 1= TxIE 被使能。成功传输帧后，IntPnd 被设置。

接收中断使能 (RxIE)

- 0= RxIE 被禁用。成功接收帧后，IntPnd 保持不变。
- 1= RxIE 被使能。成功接收帧后，IntPnd 被设置。

远程使能 (RmtEn)

- 0= RmtEn 被禁用。接收到远程帧时，TxRqst 保持不变。
- 1= RmtEn 被使能。接收到远程帧时，TxRqst 被设置。

发送请求 (TxRqst)

- 0= 该消息对象不等待传输。
- 1= 该消息对象的传输被请求并且没有完成。

TxRqst 域从传输请求寄存器 (MOTRA、MOTRB、MOTRC、MOTRD 和 MOTRX) 直接可读。然而，要写入指定消息对象的 TxRqst 域，那么主机处理器必须写入消息接口寄存器。

块结束 (EoB)

- 0= 消息对象属于一个FIFO缓冲器模块并且不是该FIFO缓冲器模块的最后消息对象。
- 1= FIFO 缓冲器模块的单消息对象或最后消息对象。

该位用于连结两个或多个消息对象（高达 128）以创建一个 FIFO 缓冲器。对于单消息对象（不属于一个 FIFO 缓冲器），该位必须总被设置为 1。

消息对象屏蔽位

消息对象屏蔽位，以及仲裁位，都用于输入消息的接受消息过滤。

使用接收屏蔽 (UMask)

- 0= 忽略屏蔽。Msk[28:0]、MXtd 和 Mdir 对接收消息过滤没有影响。对于一个要接受的输入消息，必须满足所有以下条件：
  - 接收到的消息是消息方向设置为0(接收)的数据帧或消息方向设置为1(发送)的远程帧。
  - 接收到的消息标识符匹配消息对象的消息标识符 (ID[28:0])。
  - 接收到的标识符扩展位匹配消息对象的标识符扩展位 (Xtd)。

- 1= 如果对接收消息过滤设置了各自的屏蔽位，那么将屏蔽 (Msk[28:0]、MXtd 和 MDir) 用于接收消息过滤。对于一个要接受的输入消息，必须满足所有以下条件：
  - 接收到的消息是消息方向设置为0(接收)的数据帧或是消息方向设置为1(发送)的远程帧，其中 MDir 屏蔽位使能。
  - 接收到的消息标识符匹配消息对象的消息标识符 (ID[28:0])，其中 Msk[28:0] 屏蔽位使能。
  - 接收到的标识符扩展位匹配消息对象的标识符扩展位 (Xtd)，其中 MXtd 屏蔽位使能。

如果UMask位被设置为1，那么在MsgVal设置为1之前，消息对象的屏蔽位必须在消息对象的初始化期间被配置。

标识符屏蔽 (Msk[28:0])

标识符屏蔽过滤 ID[28:0] 中相应的位。

- 0= 相应的标识符位对接受消息过滤器没有影响
- 1= 相应的标识符位用于接受消息过滤。

扩展标识符屏蔽 (MXtd)

- 0= 扩展帧标识符位 (Xtd) 对接收消息过滤没有影响
- 1= 扩展帧标识符位 (Xtd) 用于接收消息过滤

当 11-bit(标准)标识符用于消息对象时，接收到的数据帧的标识符被写入位 ID28 到 ID18。对于接受消息过滤，仅使用这些位以及屏蔽位 Msk28 到 Msk18。

屏蔽消息方向 (MDir)

- 0= 消息方向位 (Dir) 对接受消息过滤没有影响。
- 1= 消息方向位 (Dir) 用于接受消息过滤。

(1) Altera 建议总是设置 MDir 为 1。忽略消息方向位是一项高级技术，必须谨慎处理。

CAN 消息位

仲裁域 ID28-0、Xtd 和 Dir 用于定义标识符以及输出消息的类型并且(与屏蔽域 Msk28-0、MXtd 和 MDir) 一起用于输入消息的接收消息过滤。当方向被设置为接收数据帧或发送远程帧时，接收到的消息被存储到标识符匹配的有效消息对象中。扩展帧仅可以存储在 Xtd 设置为 1 的消息对象中，标准帧存储在 Xtd 设置为 0 的消息对象中。如果一个接收到的消息(数据帧或远程帧)匹配多个有效消息对象，那么它被存储在具有最低消息数的对象中。

消息标识符 (ID[28:0])

- ID28-ID0: 29-bit 标识符(扩展帧)
- ID28-ID18: 11-bit 标识符(标准帧)

扩展帧标识符 (Xtd)

- 0= 11-bit (标准)标识符用于该消息对象。
- 1= 29-bit (扩展)标识符用于该消息对象。

消息方向 (Dir)

- 0= 接收方向。当 TxRqst 被设置为 1 时，具有该消息对象标识符的远程帧被发送。接收到具有匹配标识符的数据帧时，那个消息被存储在该消息对象中。
- 1= 发送方向。当 TxRqst 被设置为 1 时，各自的消息对象被发送为一个数据帧。接收到具有匹配标识符的远程帧时，该消息对象的 TxRqst 位被设置为 1（如果 RmtEn = 1）。

数据长度代码 (DLC[3:0])

DLC 指定数据帧中数据字节数。最高数是 8。

消息对象的数据长度代码 (DLC) 的定义必须与所有 CAN 器件的具有相同标识符的所有相应对象中的相同。当消息处理器存储一个数据帧时，它将 DLC 域设置为接收到的消息中提供的值。

数据字节 0-7 (Data 0[7:0] - Data 7[7:0])

数据字节在 CAN 数据帧中。

消息接口寄存器

存在两组消息接口寄存器，IF1 和 IF2，可以对主机处理器或 DMA 控制器提供一种方式，使其间接地访问所有消息对象。消息对象在消息 RAM 和消息缓冲寄存器之间传输（作为保持消息中数据一致性的单一、原子操作）。

表 25–2 列出了每个消息接口寄存器组的结构，其中 *x* 代表组 1 或组 2。

表 25–2. 消息接口寄存器组

寄存器类型	寄存器	名称	说明
命令	IFxCMR	IFx 命令寄存器	指定传输方向以及选择要传输的消息对象部分。
消息缓冲器	IFxMSK	IFx 屏蔽寄存器	提供对消息对象的 Msk、 MDir 和 MXtd 屏蔽域的访问
	IFxARB	IFx 仲裁寄存器	提供对消息对象的 ID、Dir、Xtd 和 MsgVal 仲裁域的访问
	IFxMCTR	IFx 消息控制寄存器	提供对消息对象的 DLC、EoB、 TxRqst、RmtEn、RxIE、TxIE、UMask、 IntPnd、MsgLst 和 NewDat 域的访问
	IFxDA	IFx 数据 A 寄存器	提供对消息对象的数据字节 0-3 的访问
	IFxDB	IFx 数据 B 寄存器	提供对消息对象的数据字节 4-7 的访问

DMA 模式

CAN 控制器，如图 25–1 中所示，可以发出 DMA 控制器请求，以便在一个或两个消息接口寄存器和系统存储器之间传输数据。CAN 控制器具有两个 DMA 请求接口，称为 can_if1dma 和 can_if2dma。CAN 外设请求接口与 FPGA DMA 外设请求接口共享。要使用 DMA 外设请求接口，主机处理器必须访问协议组 (protogrp) 中的 CAN 控制寄存器 (CCTRL)。外设请求接口通过系统管理器而被选择。

要了解关于选择CAN DMA外设请求接口的更多信息，请参考 *Cyclone V器件手册*第3卷的系统管理器章节。

要激活 DMA 支持功能并且启动一个传输，写入 1 到消息接口组 (msgifgrp) 的相应 IF 命令寄存器 (IFxCMR) 中的 DMAactive 位。消息对象传输完成后，CAN 控制器发出一个 DMA 外设请求以便执行下一个消息对象传输。请求保持激活直到消息接口寄存器的第一个读取或写入发生。

要了解更多信息，请参考 *Cyclone V 器件手册*第 3 卷的 *DMA Controller* 章节。

自动再传输

CAN 控制器对在传输期间丢失仲裁或有错误的帧的自动再传输提供一种方式。再传输自动发生，而无需用户干涉或提示。当传输成功完成时，就会给予正常确认。

测试模式

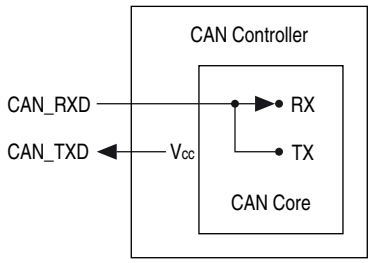
要使能测试模式，请设置 CCTRL 寄存器中测试模式使能位 (Test) 为 1。这一操作会激活 CAN 测试寄存器 (CTR) 的写访问。以下部分介绍可用测试模式。

静音模式

通过编程测试寄存器 (CTR) 中静音模式 (Silent) 位为 1，CAN 控制器被设置为静音模式。在静音模式中，CAN 控制器能够接收有效数据帧和有效远程帧，但是它保持 CAN_TXD 管脚为高电平，对 CAN 总线不发送数据。静音模式可用于分析 CAN 总线的流量，而不会通过关键位（确认位、错误帧）的发送对其产生影响。在 ISO 11898-1 中，静音模式被称为*总线监控模式*。

图 25 – 2 显示了静音模式中的 CAN 内核。

图 25 – 2. 静音模式中的 CAN 内核



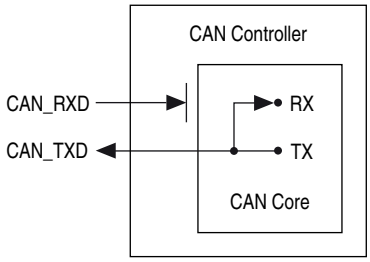
环回模式

通过编程测试寄存器 (CTR) 中环回模式 (LBack) 位为 1，CAN 控制器被设置为环回模式。在环回模式中，CAN 控制器将自己发送的消息作为接收的消息并且将它们（如果它们通过接收消息过滤）存储到接收缓冲器中。

为了独立于外部仿真，CAN 控制器忽略环回模式中的确认错误。在该模式中，CAN 控制器提供从其发送 (TX) 输出到其接收 (RX) 输入的内部反馈。输入管脚的实际值被 CAN 控制器忽略。

图 25 – 3 显示了环回模式中的 CAN 内核。

图 25 – 3. 环回模式中的 CAN 内核

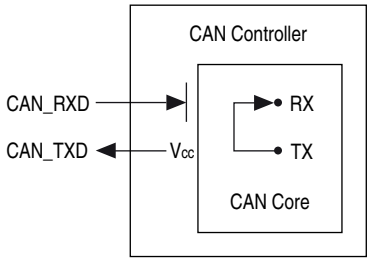


组合模式

通过编程测试寄存器 (CTR) 中的静音模式 (Silent) 和环回模式 (LBack) 位为 1，CAN 控制器被设置为组合的环回和静音模式。组合模式可用于测试 CAN 硬件而不会影响连接到 CAN 总线的其它器件。在该模式中，CAN_RXD 管脚从 CAN 内核断开并且 CAN_TXD 管脚保持高电平。

图 25 – 4 显示了组合模式中的 CAN 内核。

图 25 – 4. 组合模式中的 CAN 内核



L4 从接口

主机处理器通过 L4 从接口访问 CAN 控制器的数据、控制和状态信息。从接口仅支持 32-bit 访问。

该接口不支持错误响应。

时钟

CAN 控制器在 l4_sp_clk 和 can_clk 时钟输入上进行操作。l4_sp_clk 时钟由 L4 从接口使用而 can_clk 用于操作 CAN 内核。

can_clk clock 必须被编程至少为 CAN 总线接口速度的 8 倍。例如，对于在 1 Mbps baud 速率上操作的 CAN 总线接口，can_clk 时钟必须被设置为至少 8 MHz。l4_sp_clk 时钟可以在等于或大于 can_clk 频率的时钟频率上操作。

要了解关于l4_sp_clk和can_clk时钟的更多信息，请参考*Cyclone V器件手册*第3卷的*时钟管理器*章节。

复位

每个 CAN 控制器都可以由软件或硬件进行复位。

软件复位

软件初始化由设置 CAN 控制器寄存器 map 的协议组 (protogrp) 的 CAN 控制寄存器 (CCTRL) 中的 Init 位而完成。当总线关断情况出现在 CAN 链路时，该位通过 CAN 协议设置。该位也通过 “ 硬件复位 ” 中介绍的硬件复位输入设置。

由于两个时钟域之间的同步机制，可能存在延迟直到可以读回写入Init位的值。要确保之前写入的值已经被接受，那么在设置 Init 位一个新值之前读取该位。

总线关断恢复序列不可以通过设置或复位 Init 位而被缩短。要了解关于总线关断的更多信息，请参考 Bosch 网站 (www.semiconductors.bosch.de) 的 *CAN Protocol Specification 2.0* 的 A 和 B 部分。

硬件复位

每个 CAN 控制器都有一个单独的复位信号。复位管理器驱动冷或暖复位时的信号。复位信号同步到两个时钟域并且应用于 CAN 控制器内的相应逻辑。

要了解更多信息，请参考 *Cyclone V 器件手册*第 3 卷的*复位管理器*章节。

中断

每个 CAN 控制器都生成两个中断信号。一个信号表示错误和状态中断，另一个信号表示消息对象中断。两个中断信号连接到全局中断控制器 (GIC)。中断在协议组 (protogrp) 的 CAN 控制寄存器 (CCTRL) 中使能。协议组 (protogrp) 中的 CAN 中断寄存器 (CIR) 指出待定的最高优先权中断。

错误中断

以下错误情况生成中断：

- 总线关断 — 当发送错误数等于或大于 256 时，协议组 (protogrp) 的 CAN 状态寄存器 (CSTS) 中的总线关断 (BOff) 位被设置为 1。
- 错误警告 — 当发送错误计数器或接收错误计数器达到96时，协议组 (protogrp) 的CAN 状态寄存器 (CSTS) 中的错误警告状态 (EWarn) 位被设置为 1。

状态中断

以下状态情况生成中断：

- 接收OK — 当CAN控制器成功接收消息时，协议组 (protogrp) 的CAN状态寄存器 (CSTS) 中的 RxOK 位被设置为 1。
- 发送OK — 当CAN控制器成功发送消息时，协议组 (protogrp) 的CAN状态寄存器 (CSTS) 中的 TxOK 位被设置为 1。
- 最后错误代码 — 当接收的或发送的消息具有错误时，协议组 (protogrp) 的CAN状态寄存器 (CSTS) 中的 LEC 位根据错误类型被设置。

消息对象中断

当相应的消息对象 TxIE 位或 RxIE 位设置为 1 时，消息对象的 IntPnd 位可生成中断。表 25 – 2 列出了中断待定寄存器中消息对象中断信息的位置。中断待定寄存器位于消息处理器组 (msghandgrp) 中。

表 25 – 3. 消息对象中断寄存器

寄存器	名称	消息对象
MOIPA	中断待定 A 寄存器	1 到 32
MOIPB	中断待定 B 寄存器	33 到 64
MOIPC	中断待定 C 寄存器	65 到 96
MOIPD	中断待定 D 寄存器	97 到 128

MOIPX 寄存器使软件可以快速检测到哪个消息对象组具有待定中断。

CAN 控制器编程模型

该部分介绍如何操作 CAN 控制器。

软件初始化

通过设置 CAN 控制寄存器 (CCTRL) 中的 Init 位为 1 开始软件初始化。当 Init 位是 1 时，消息不会发送到 CAN 总线或从 CAN 总线发送，并且 CAN_TXD CAN 总线发送保持高电平状态。设置 Init 位不会更改任何的配置寄存器。

要初始化 CAN 控制器，主机处理器必须编程用于 CAN 通信的 CAN 位时序 (CBT) 寄存器和消息对象。如果不需要消息对象，那么将消息对象的 MsgVal 位设置为无效 (0) 已足够，这在 RAM 初始化之后是默认的。设置 MsgVal 位为有效 (1) 之前必须设置整个消息对象。消息对象通过其中一个消息接口寄存器组设置。

只有当 CAN 控制寄存器 (CCTRL) 中的配置更改使能 (CCE) 和 Init 位设置为 1 时，才使能对 CAN 位时序 (CBT) 寄存器的访问。

设置 Init 位为 0 会完成软件初始化。CAN 内核参加总线活动和消息传输之前，通过等待总线达到闲置状态，它将本身同步到 CAN 总线的数据传输。

消息对象的初始化与 CAN 控制器初始化无关并且可以在随时完成，但是消息发送开始之前，消息对象应该全部被配置为特定的标识符或设置为无效。

上电时，消息 RAM 必须被初始化。要初始化 RAM，设置 Init 位为 1，然后设置协议组 (protogrp) 的 CAN 功能寄存器 (CFR) 中的 RAMInit 位为 1。当 RAM 初始化完成时，RAMInit 位返回 0。在 RAM 初始化期间，所有消息对象被清零并且 RAM ECC 位被初始化。RAM 初始化之前或期间不允许访问 RAM。

CAN 消息传输

一旦 CAN 控制器被初始化，CAN 控制器将其本身同步到 CAN 总线并且开始传输消息。

如果接收的消息通过消息处理器的接受消息过滤，那么它们被存储到相应的消息对象中。整个消息，包括所有仲裁位 Xtd、Dir、DLC、8 个数据字节，屏蔽和控制位 UMask、MXtd、MDir、EoB、MsgLst、RxIE、TxIE 和 RmtEn，存储在消息对象中。当接收到的消息被存储时，屏蔽的仲裁位可能在消息对象中更改。

主机处理器可能使用消息接口寄存器随时读取或更新每个消息。当主机处理器在消息被传输到 RAM 或从 RAM 传输的同时访问消息对象时，消息处理器保证数据一致性。

发送的消息由主机处理器更新。如果对于消息出现一个永久消息对象（对于多个 CAN 传输，配置期间设置的仲裁和控制位不更改），那么只有数据字节需要被更新。如果几个发送消息被分配到相同的消息对象（当消息对象数不足够时），那么整个消息对象必须在该消息被要求传输之前被配置。

根据消息对象的内部优先权，消息对象被发送的同时消息对象的任意数的发送会被请求。消息对象数是 1 到 128，1 为最低内部优先权，128 为最高优先权。甚至当消息的请求发送仍然待定时，消息也会被随时更新或设置为无效 (MsgVal=0)。当消息在其待定发送开始之前被更新时，旧数据被忽略。

根据消息对象的配置，通过接收具有匹配标识符的远程帧，消息的发送会被自动请求。远程帧用于请求一个 CAN 网络的特定消息。

☞ 为了简化编程，Altera建议将一个IF消息接口用于所有接收方向活动和另一个IF消息接口用于所有发送方向活动。

帧接收的消息对象重配置

要配置消息对象以接收数据帧，请设置 Dir 域为 0。

要配置消息对象以接收远程帧，请设置 Dir 域为 1、设置 UMask 为 1 并且设置 RmtEn 为 0。

要避免在对象发送时对其进行修改，必须在更改以下任意一个配置和控制位之前将 MsgVal 设置为 0:

- ID[28:0]
- Xtd
- DLC[3:0]
- RxIE
- TxIE
- RmtEn
- EoB
- UMask
- Msk[28:0]
- MXtd
- MDir

以下的消息对象域可以被更改而无需将 MsgVal 清零：

- Data0[7:0] 到 Data7[7:0]
- TxRqst
- NewDat
- MsgLst
- IntPnd

帧发送的消息对象重配置

要配置消息对象以发送数据帧，设置 Dir 域为 1，并设置 UMask 为 0 或设置 RmtEn 为 1。

更改以下任意一个配置和控制位之前，必须设置 MsgVal 为 0：

- Dir
- RxIE
- TxIE
- RmtEn
- EoB
- UMask
- Msk[28:0]
- MXtd
- MDir

以下消息对象的域可以被更改而无需将 MsgVal 清零：

- ID[28:0]
- Xtd
- DLC[3:0]
- Data0[7:0] to Data7[7:0]
- TxRqst
- NewDat
- MsgLst
- IntPnd

CAN 控制器地址映射和寄存器定义

地址映射和寄存器定义位于该手册卷附带的 **hps.html** 文件中。点击链接以打开文件。

要查看模块说明和基地址，找到并且点击以下其中一个模块实例的链接：

- **can0**
- **can1**

然后要查看寄存器和域说明，找到并且点击寄存器名称。寄存器地址是相对于每个模块实例的基地址偏移。

所有模块的基地址也在 *Cyclone V 器件手册*第 3 卷的*硬核处理器系统的简介*章节中列出。

文档修订历史

表 25 – 4 显示了该文档的修订历史。

表 25 – 4. 文档修订历史

日期	版本	修订内容
2012 年 11 月	1.2	■ 少量文本编辑。 ■ 扩展了复位部分。 ■ 扩展了中断部分。
2012 年 5 月	1.1	添加了结构图和系统集成、功能说明、编程模型、地址映射和寄存器定义部分。
2012 年 1 月	1.0	首次发布。