

控制器局域网络（CAN）控制器

英飞凌XC800系列单片机



Never stop thinking

写在前面

- 本篇内容为英飞凌科技有限公司（Infineon Technologies CO., LTD.）的XC800系列单片机的基础篇之一。
- 本篇所述内容为XC800系列单片机中的XC886/888和XC878子系列提供CAN外设。
- 如无特别说明，所指的产品为上述XC800子系列单片机中的**XC886CLM**单片机。由于后续芯片会有更多的改进 / 增加措施，如需要关注其它产品，需要再结合相应的产品数据手册（Data Sheet）和用户手册（User Manual）！
- 由于版本更新等原因，可能会出现各版本间的资料说法有略微差异，请以英飞凌网站公布的最新英文版本的产品数据手册（Data Sheet）和用户手册（User Manual）为准！

本篇内容

- CAN总线原理
- Infineon MultiCAN
- MultiCAN的组成
- MultiCAN的运用
- 实战练习：
 - LED灯控实验（报文的发送 / 接收）

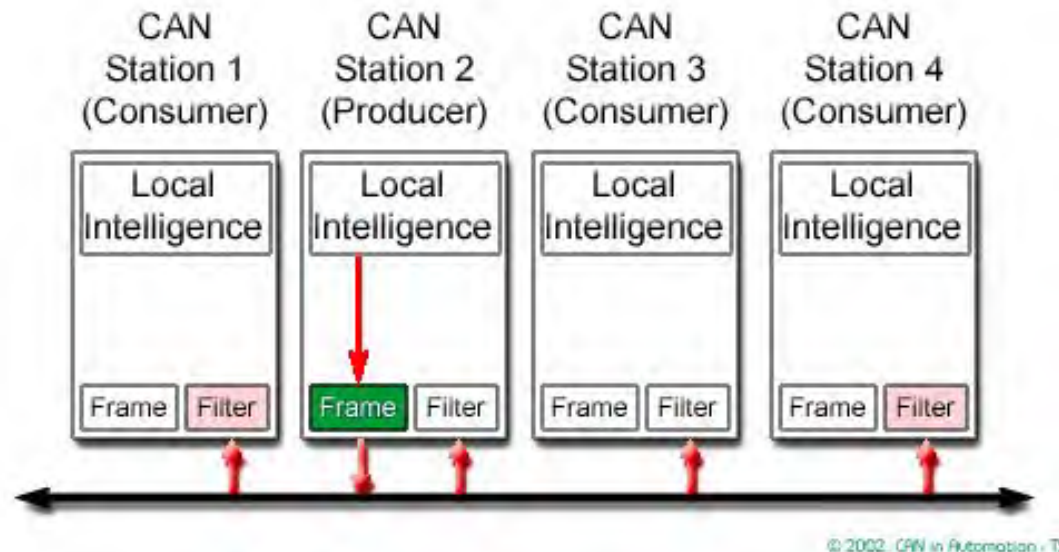
CAN总线原理

■ BOSCH CAN

- CAN（Controller Area Network）为局域网控制总线，符合国际标准ISO11898。
- CAN总线最初是由德国的BOSCH公司为汽车的监测、控制系统设计的，属于总线式通讯网络。CAN总线规范了任意两个CAN节点之间的兼容性，包括电气特性及数据解释协议。CAN协议分为两层：物理层和数据链路层。物理层用于决定实际位传送过程中的电气特性。在同一网络中，所有节点的物理层必须保持一致，但可以采用不同方式的物理层。CAN的数据链路层功能则包括帧组织形式、总线仲裁和检错、错误报告及处理、对要发送信息的确认以及确认接收信息并为应用层提供接口等。
- 其主要特点是：
 - 能够以多主方式工作，网络上的任意节点均可成为主节点，并可向其它节点传送信息。
 - 非破坏性总线仲裁和错误界定，总线冲突的解决和出错界定可由控制器自动完成，且能区分暂时和永久性故障并自动关闭故障节点。
 - CAN节点可被设定为不同的发送优先级。以满足不同的实时要求。
 - 采用差分驱动，可在高噪声干扰环境下使用。

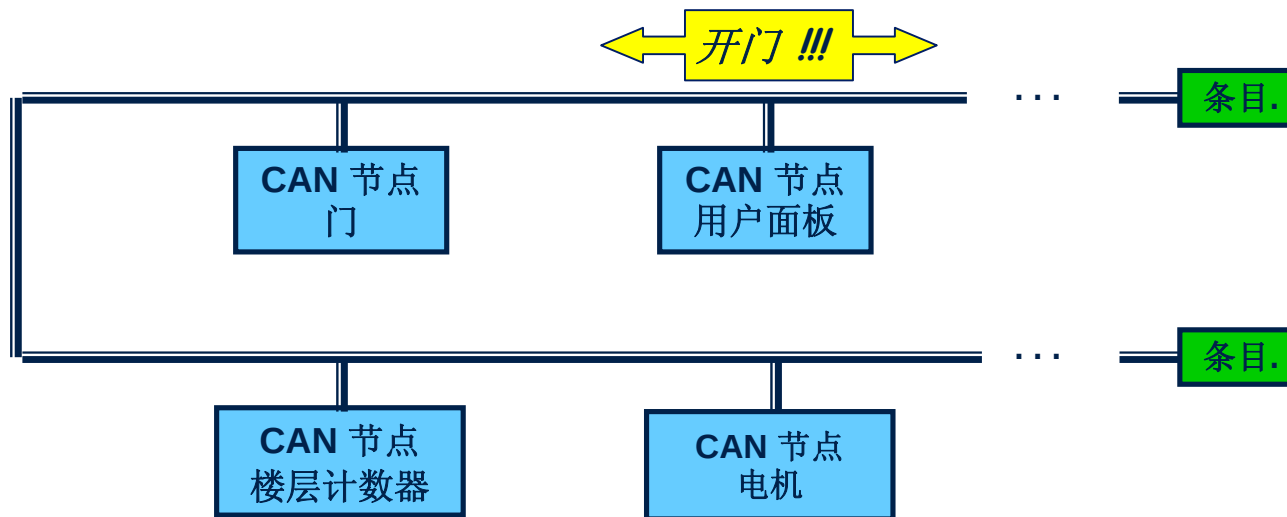
CAN总线原理（续）

- CAN总线是一个面向报文的协议
- 报文的内容是界定的
- 每一个报文都有自己独特的识别标志
- 节点的数目没有限制
- CAN总线网络是容易升级的模块化网络



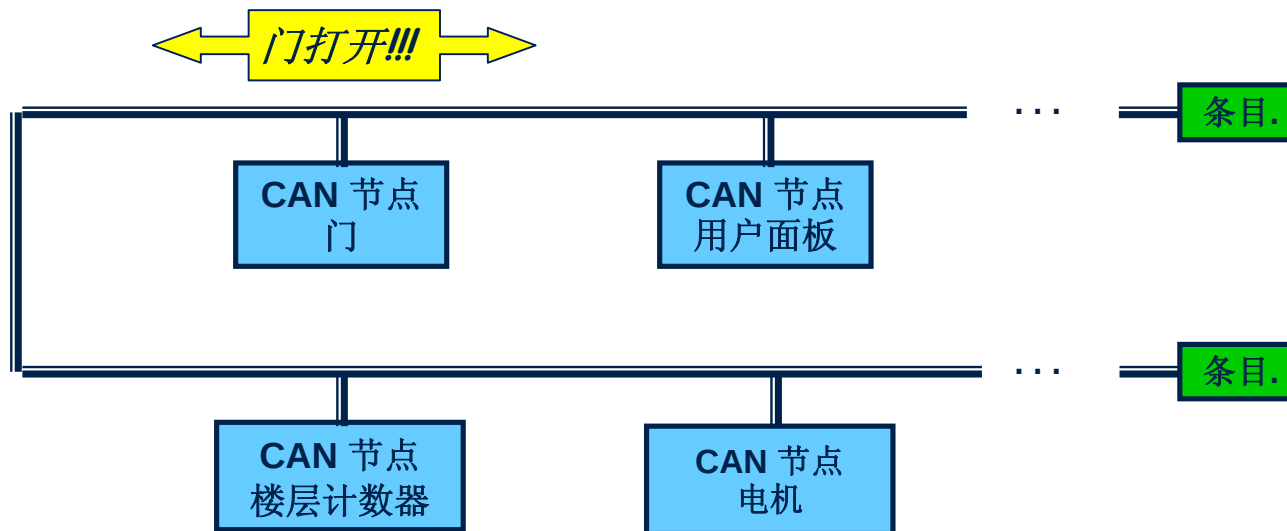
入门例子-电梯（1）

- 4个节点
- 10个不同的报文
- 门命令



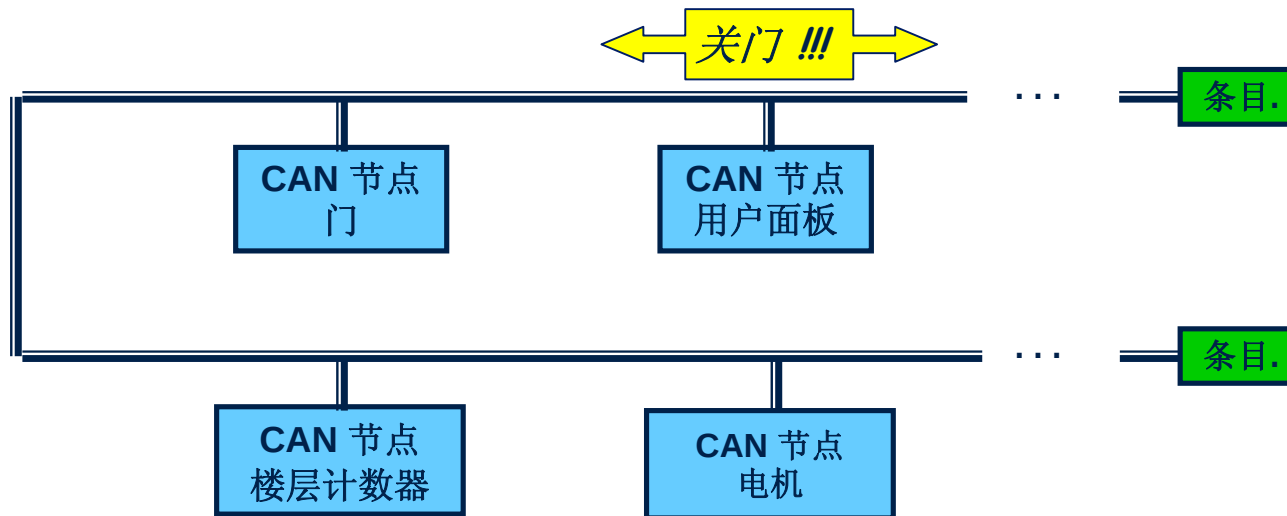
入门例子-电梯（2）

- 4个节点
- 10个不同的报文
- 门命令



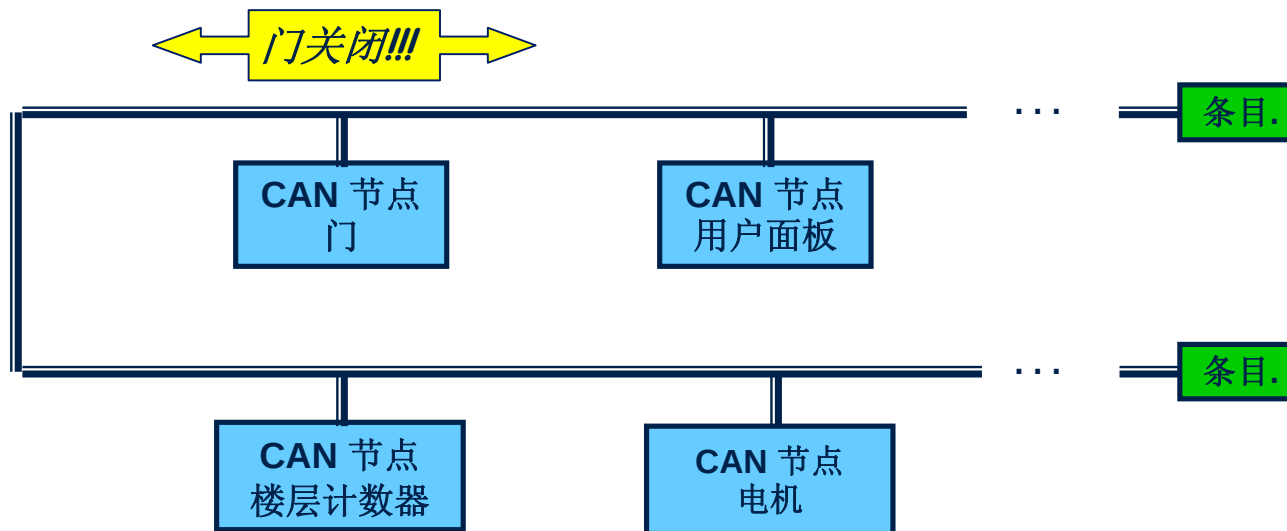
入门例子-电梯（3）

- 4个节点
- 10个不同的报文
- 门命令



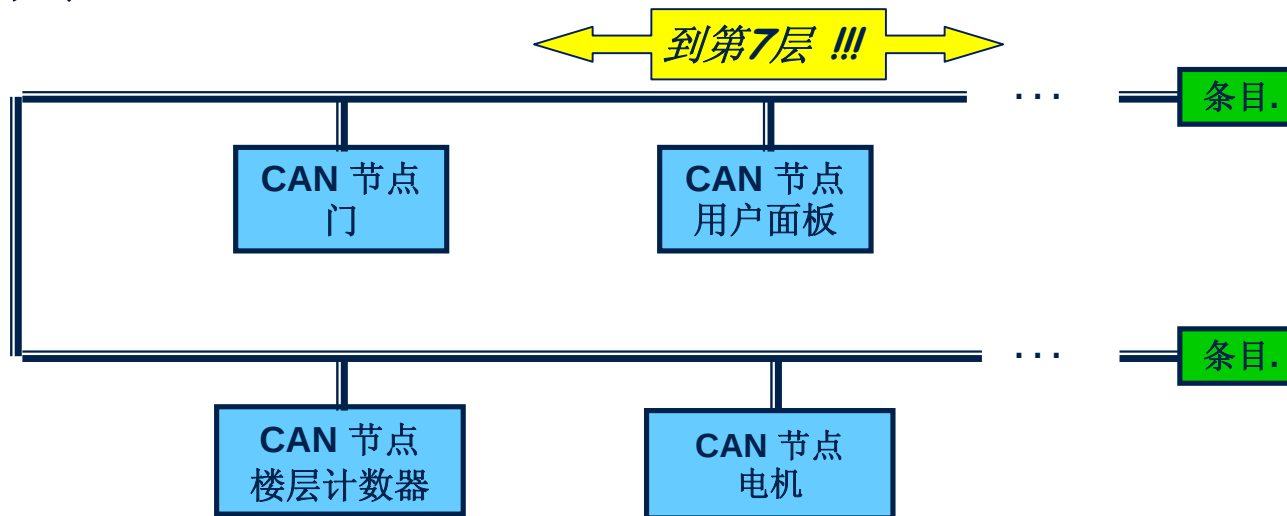
入门例子-电梯（4）

- 4个节点
- 10个不同的报文
- 门命令



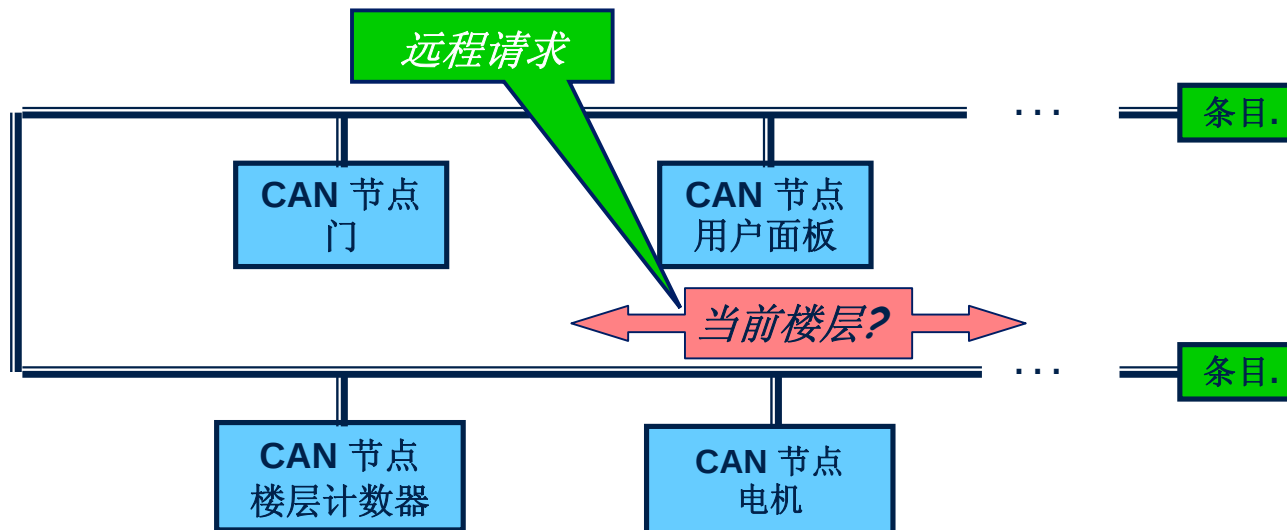
入门例子-电梯（5）

- 4个节点
- 10个不同的报文
 - 门命令
 - 楼层命令



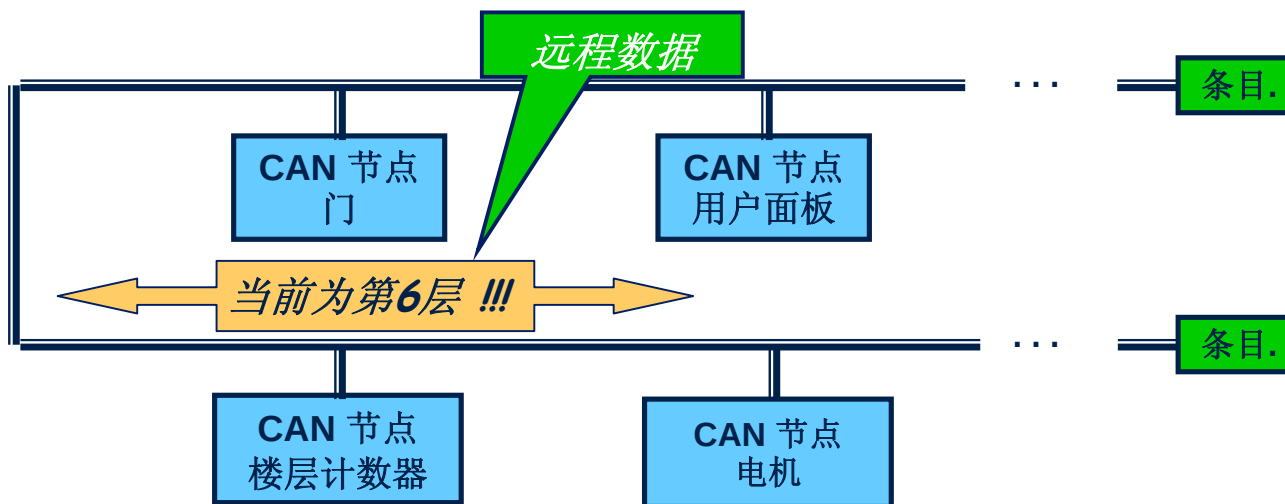
入门例子-电梯（6）

- 4个节点
- 10个不同的报文
 - 门命令
 - 楼层命令
- 一个远程请求/响应



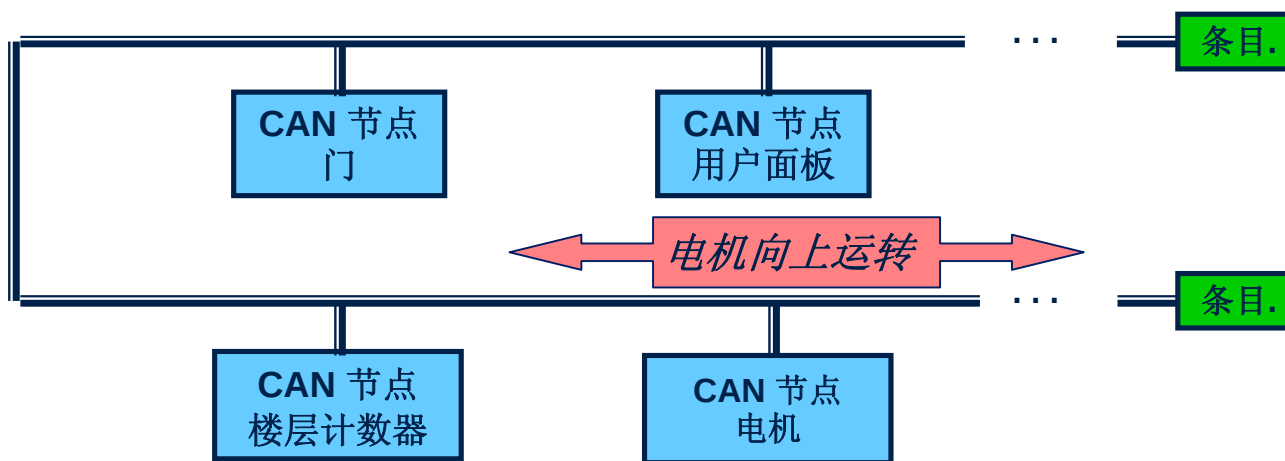
入门例子-电梯（7）

- 4个节点
- 10个不同的报文
 - 门命令
 - 楼层命令
- 一个远程请求/响应



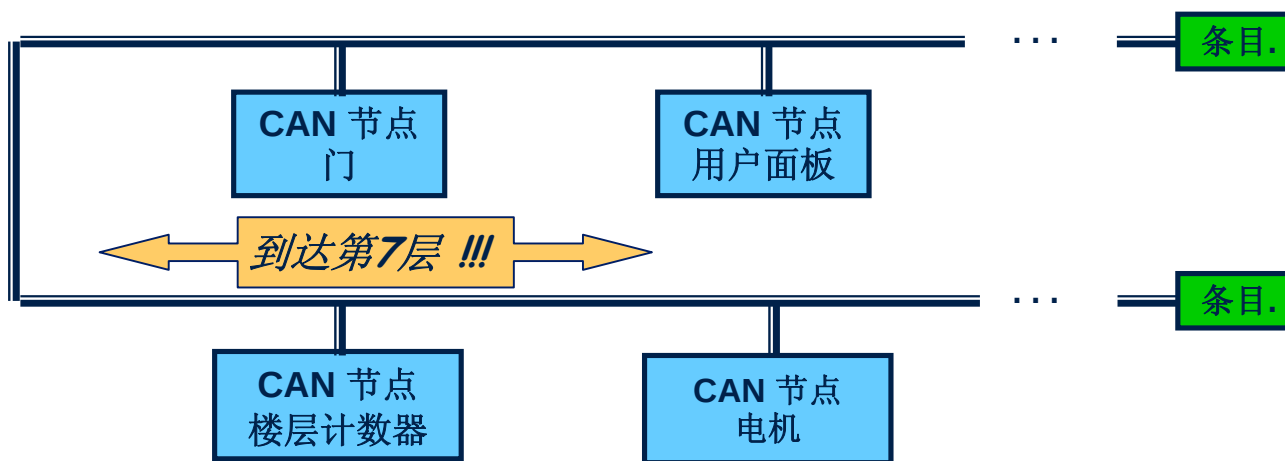
入门例子-电梯（8）

- 4个节点
- 10个不同的报文
 - 门命令
 - 楼层命令
- 一个远程请求/响应



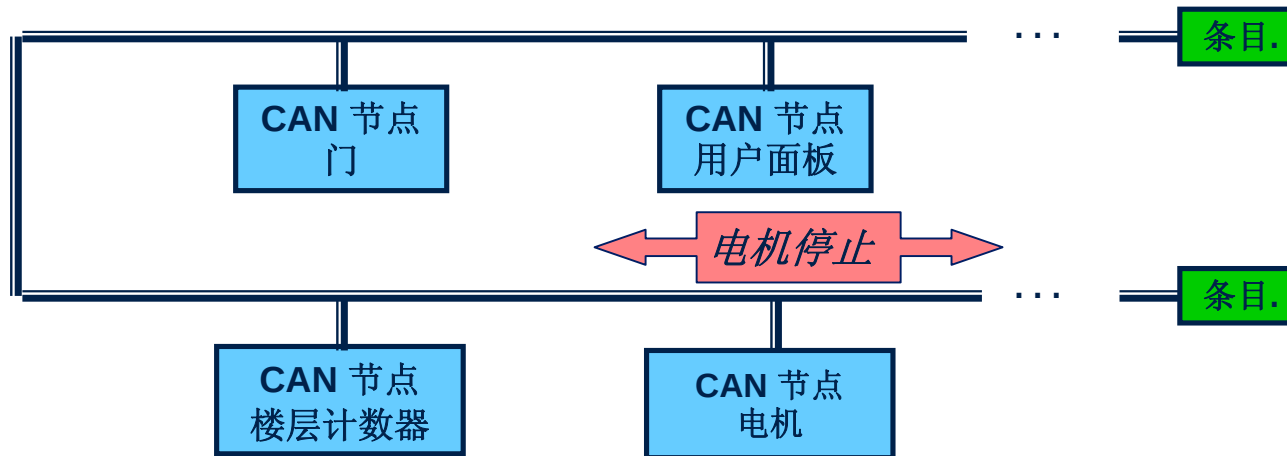
入门例子-电梯（9）

- 4个节点
- 10个不同的报文
 - 门命令
 - 楼层命令
- 一个远程请求/响应



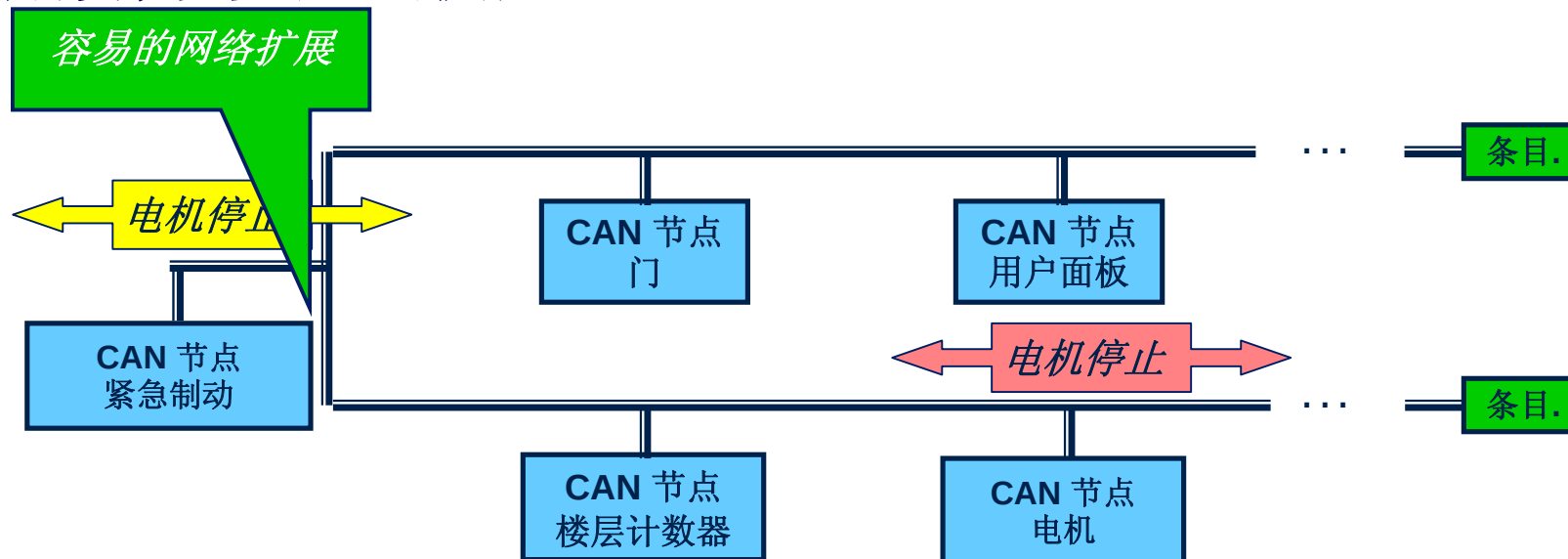
入门例子-电梯（10）

- 4个节点
- 10个不同的报文
 - 门命令
 - 楼层命令
- 一个远程请求/响应



入门例子-电梯（11）

- 4个节点
- 10个不同的报文
 - 门命令
 - 楼层命令
- 一个远程请求/响应
- 容易实现更多节点的扩展



■ Infineon XC800单片机所带的CAN称为MultiCAN

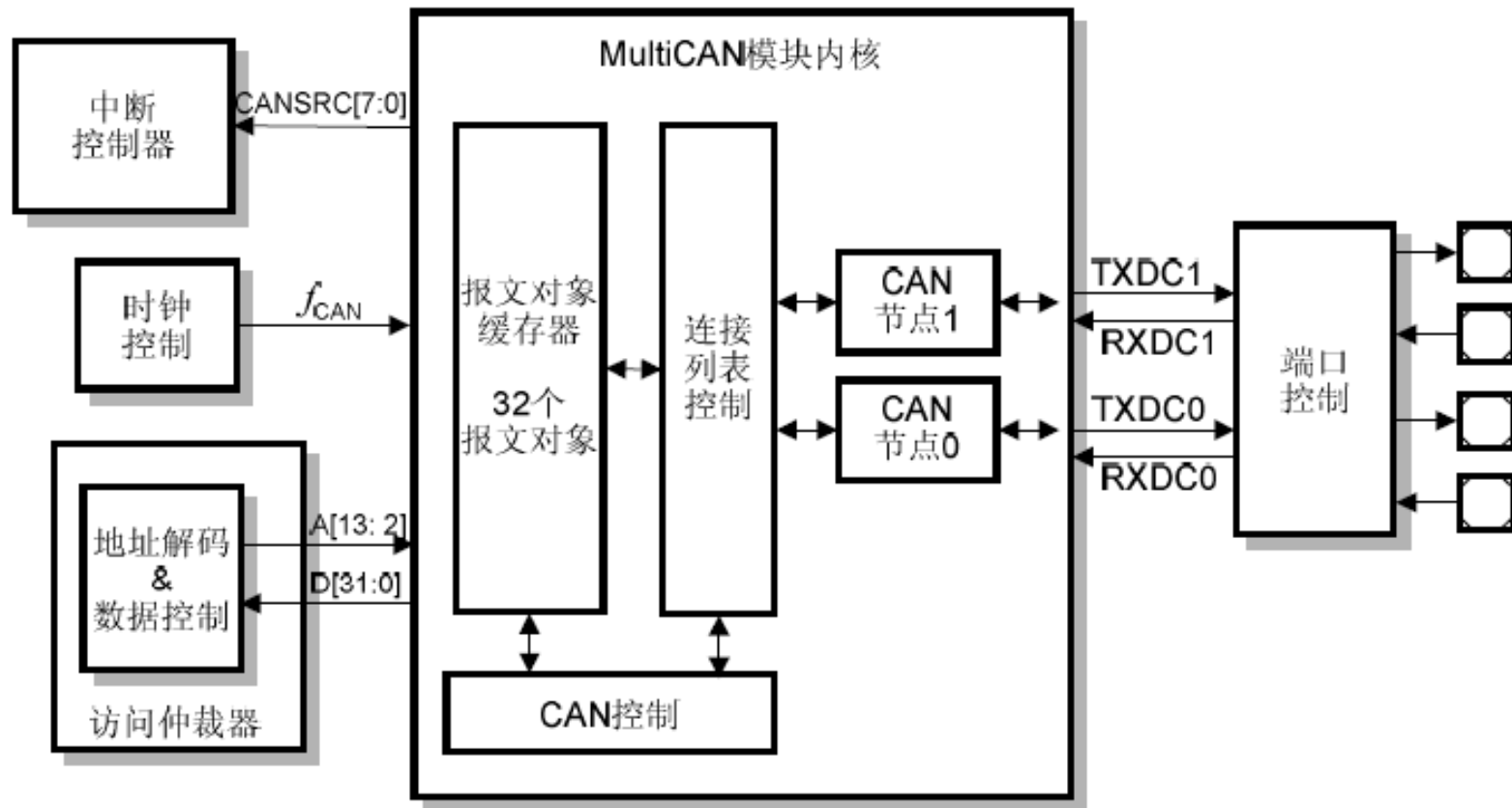
- Infineon的MultiCAN为BOSCH CAN的超集，并在此基础上扩充了一些特殊功能。
- MultiCAN模块包含大于两个的全CAN功能节点（XC800系列的产品含两个CAN节点），每个节点可独立工作或者通过网关功能交换数据和远程帧。所有的CAN节点共用一套报文对象，每个报文对象可被独立分配给任意一个有效的CAN节点之一。每个CAN节点都可以接收和发送带11位标识符的标准帧和带29位标识符的扩展帧。
- MultiCAN节点仅将帧存储到分配给该节点列表的报文对象中，且仅发送属于该报文对象列表的报文。功能强大，由命令驱动列表控制器执行所以报文对象列表操作。

- 与ISO11898标准兼容
- 根据CAN V2.0B Active技术规范确定CAN功能
- 每个CAN节点都有专用控制寄存器
- 数据传送速率高达1Mbit/s
- CAN总线位时序分析和由帧计数器实现的波特率检测功能
- 先进的验收滤波功能
- 先进的报文对象功能
- 先进的数据管理
- 先进的中断处理

Infineon MultiCAN

模块组成

■ MultiCAN模块



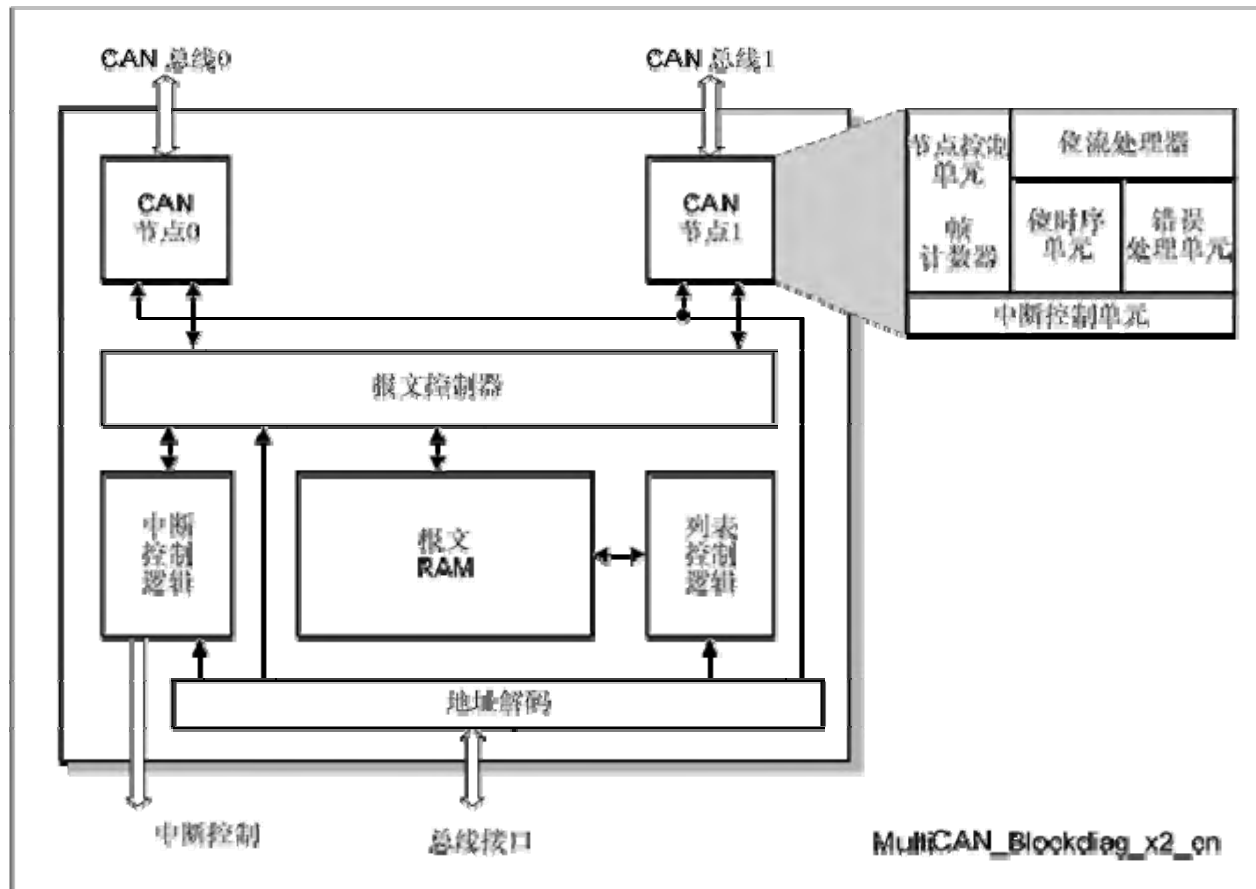
MultiCAN_XC8_overview_cn

Infineon MultiCAN

结构框图

■ MultiCAN

- 每个CAN节点可单独配制并运行，不依赖于其他CAN节点。每个CAN节点都配有一套独立的SFR寄存器，对该节点进行控制和监控。



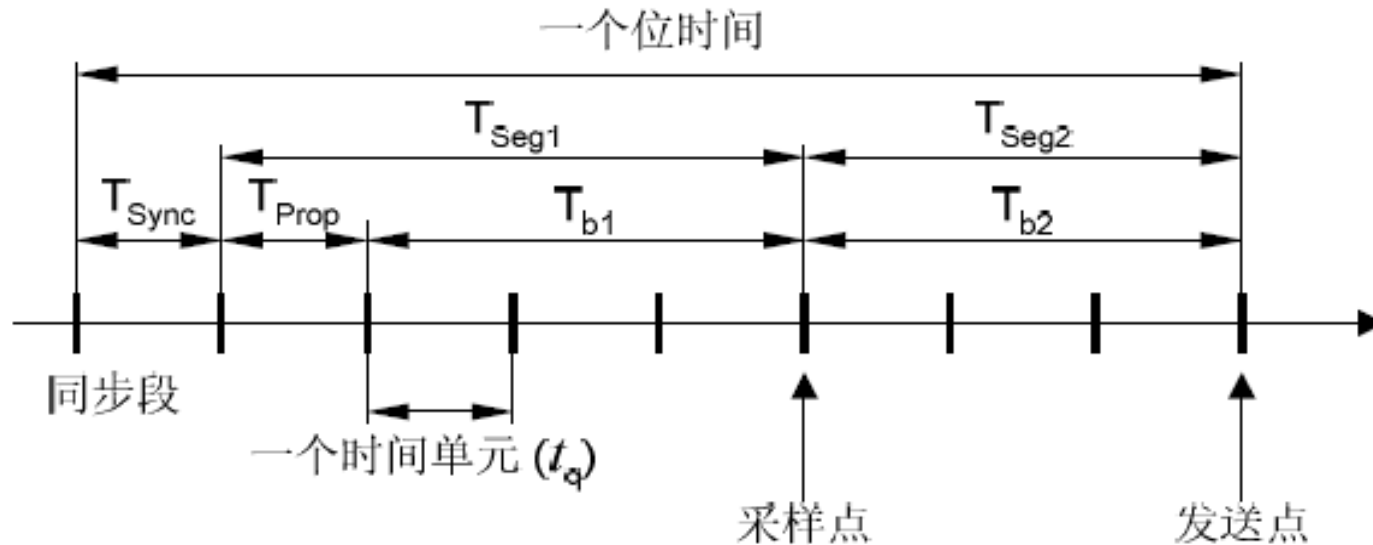
- 当有效CAN节点的波特率为1Mbit/s时，需要注意有最小工作频率 f_{CAN} 的限制，见下表。
- 如果需要的波特率小于1Mbit/s，所需的频率值依照线性比例确定（例如：需要的最大波特率为500kbit/s时，频率值为下表中所给出的50%）。

表：最小工作频率 [MHz]

被分配的报文对象个数 ¹⁾	只有1个CAN节点有效	2个CAN节点都有效
16个报文对象	12	19
32个报文对象	15	23

- ¹⁾ 只考虑那些已经分配给CAN节点的报文对象，未分配的报文对象对最小工作频率没有影响。

- 在同步段 (T_{Sync}) 进行发射和接收时间基准之间的相位同步。同步段长度总为一个 t_q 。传播段 (T_{Prop}) 考虑CAN总线上的发射输出驱动器和收发电路的物理传播延迟。对于工作冲突检测机制来讲, T_{Prop} 必须等于所有的传播延迟量的总和舍入到 t_q 的整数倍之后, 再乘以2所得的值。采样点之前和之后的相位缓冲段1和2 (T_{b1} , T_{b2}) 用于补偿同步段中检测到的发射和接收之间的时钟相位失配。



MultiCAN的组成

位时间的计算

- 重新同步阶段所允许的时间单元的最大值由位域NBTRx.SJW定义。传播段和相位缓冲段1结合成参数 T_{Seg1} ，由NBTRx.TSEG1中的值定义，ISO标准要求其最小值为3个时间单元。参数 T_{Seg2} ，由NBTRx.TSEG2 的值定义，包含了相位缓冲段2，ISO标准要求其最小值为2个时间单元。根据ISO标准，CAN位时间，是 T_{Sync} ， T_{Seg1} 和 T_{Seg2} 的总和，必须不少于8个时间单元。

$$t_q = (\text{BRP} + 1) / f_{\text{CAN}}$$

如果 $\text{DIV8} = 0$

$$= 8 \times (\text{BRP} + 1) / f_{\text{CAN}}$$

如果 $\text{DIV8} = 1$

$$T_{\text{Sync}} = 1 \times t_q$$

$$T_{\text{Seg1}} = (\text{TSEG1} + 1) \times t_q \quad (\text{最小: } 3t_q)$$

$$T_{\text{Seg2}} = (\text{TSEG2} + 1) \times t_q \quad (\text{最小: } 2t_q)$$

$$\text{位时间} = T_{\text{Sync}} + T_{\text{Seg1}} + T_{\text{Seg2}} \quad (\text{最小: } 8t_q)$$

MultiCAN的组成

位时间的计算（续）

- 为了补偿不同 CAN控制器之间的时钟相移，CAN控制器必须在从隐性到显性总线电平的任意边沿进行同步。如果硬同步被使能（在一帧的开始），在同步段重新开始位时间。否则，重新同步跳转宽度TSJW定义最大数目的时间单元，重新同步操作可能缩短或拉长位时间。SJW的值由位域NBTRx.SJW定义。

$$T_{SJW} = (SJW + 1) \times t_q$$

$$T_{Seg1} \geq T_{SJW} + T_{Prop}$$

$$T_{Seg2} \geq T_{SJW}$$

f_{CAN} 相对容差由相位缓冲段和重新同步跳转宽度决定。

$$df_{CAN} \leq \min(T_{b1}, T_{b2}) / 2 \times (13 \times \text{位时间} - T_{b2}) \text{ 且}$$

$$df_{CAN} \leq T_{SJW} / 20 \times \text{位时间}$$

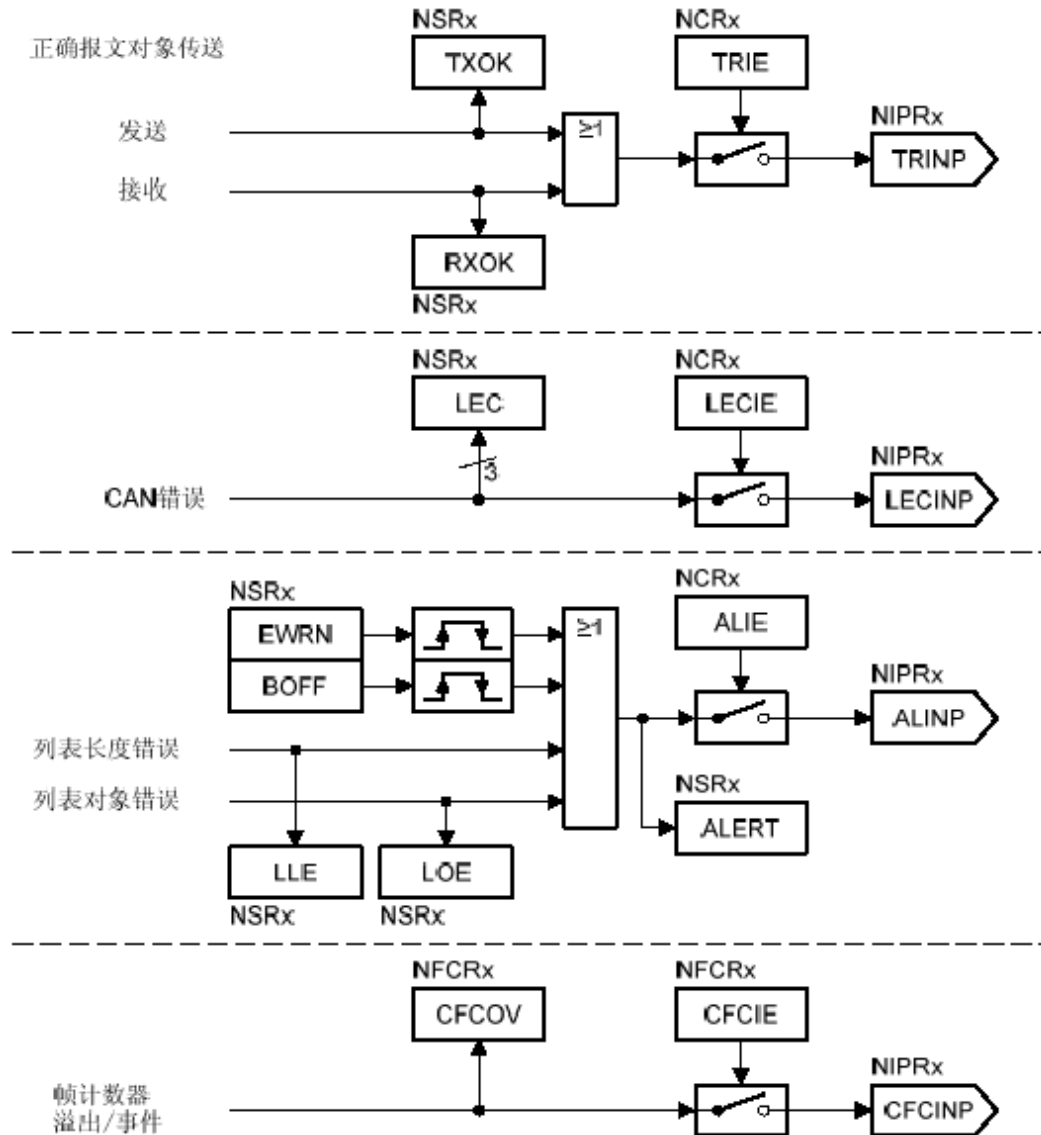
- 必须在复位 NCRx.INIT之前，将有效的位时序写入到寄存器NBTR中，在位 NCRx.CCE置位（使能配置更改）的情况下，才可以对寄存器NBTRx进行写操作。

- **CAN节点x的错误处理单元负责对CAN器件进行故障界定。该单元有2个计数器，接收错误计数器NECNTx.REC和发送错误计数器 NECNTx.TEC，这两个计数器根据来自位流处理器的命令相应地增加和减少。如果在发送操作的同时，位流处理器自身检测到了一个错误，发送错误计数器增加 8。当一个外部CAN节点因为错误帧产生而报告一个错误情况，节点计数器增1。对于错误分析、出错的报文传送方向和识别到传送错误的节点，由相关的CAN节点x的寄存器NECNTx指示。根据错误计数器中的值，CAN节点被设置为如下状态：“错误激活”“错误认可”和“总线关闭”。**
- 如果两个错误计数器的值都低于“错误认可”界限128，CAN处于错误激活状态。如果至少两者之一等于或者大于128，CAN节点处于错误认可状态。
- 如果发送错误计数器的值等于或者大于“总线关闭”界限256，“总线关闭”状态被激活，并由标志符NSRx.BOFF报告该状态。器件一直保持在该状态，直到完成“总线关闭”恢复过程为止。此外，当至少一个错误计数器等于或大于位域NECNTx.EWRNLVL中定义的错误警告值时，NSRx.EWRN被置位。如果两个错误计数器的值再次降到错误警告值以下，NSRx.EWRN被复位。

- 每个 CAN 节点都配有 4 个中断源来产生中断请求：
 - 成功发送/接收一帧
 - 带最近错误码的 CAN 协议错误
 - 出现警告条件：发送/接收错误计数器达到警告界限，总线关闭状态改变，出现列表长度错误，或者出现列表对象错误
 - 帧计数器溢出
- 除了由硬件产生中断，也可通过寄存器 MITR 由软件触发中断。向位域 MITR.IT 的位 n 写 1，将在相应的中断输出线 CANSRCm 上产生中断请求信号。当写访问位域 MITR.IT 时，如果多于一位被置位，则导致相应的多条中断输出线 CANSRCm 被同时激活。

MultiCAN的组成

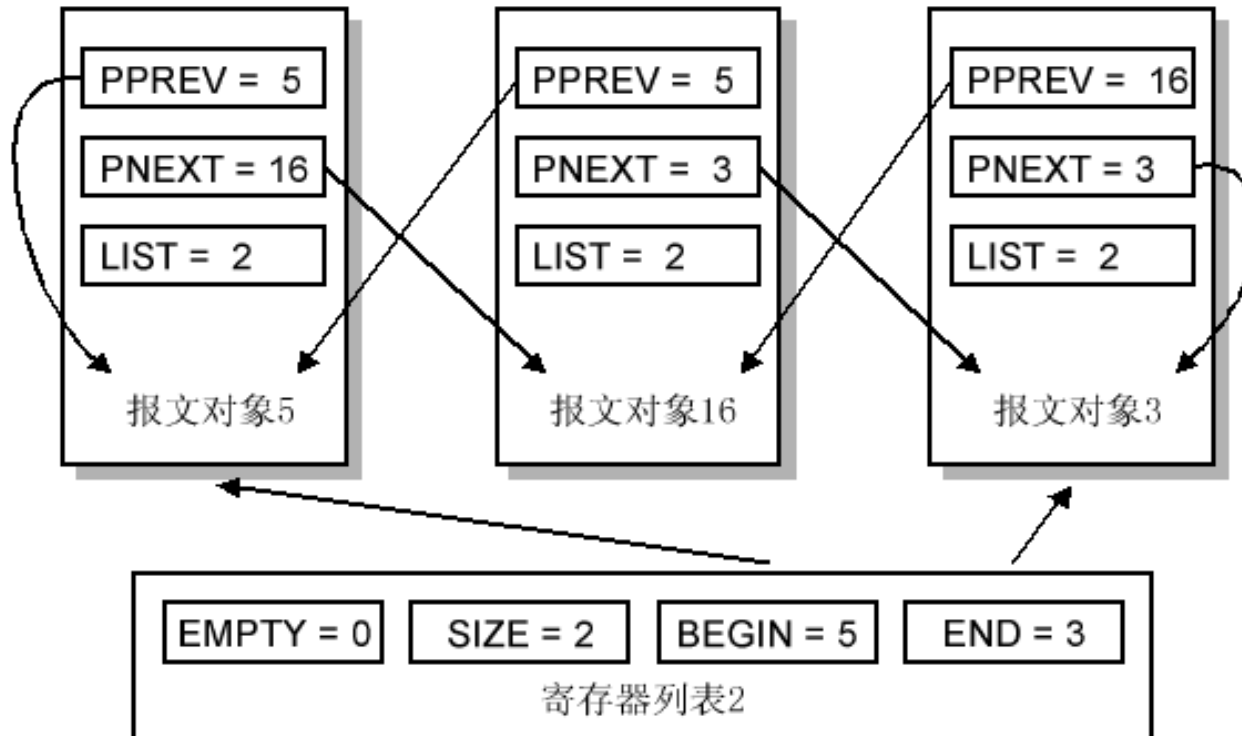
CAN节点中断（续）



MultiCAN的组成

报文的列表结构

- **MultiCAN**模块的报文对象被组织为双链列表，在该列表结构中，每个报文对象都有两个指针，一个指向列表中前一个报文对象，另一个指向列表中的后一个报文对象。**MultiCAN**模块提供**8**个列表，每个报文对象分配给其中之一。



MultiCAN的组成

报文的列表结构（续）

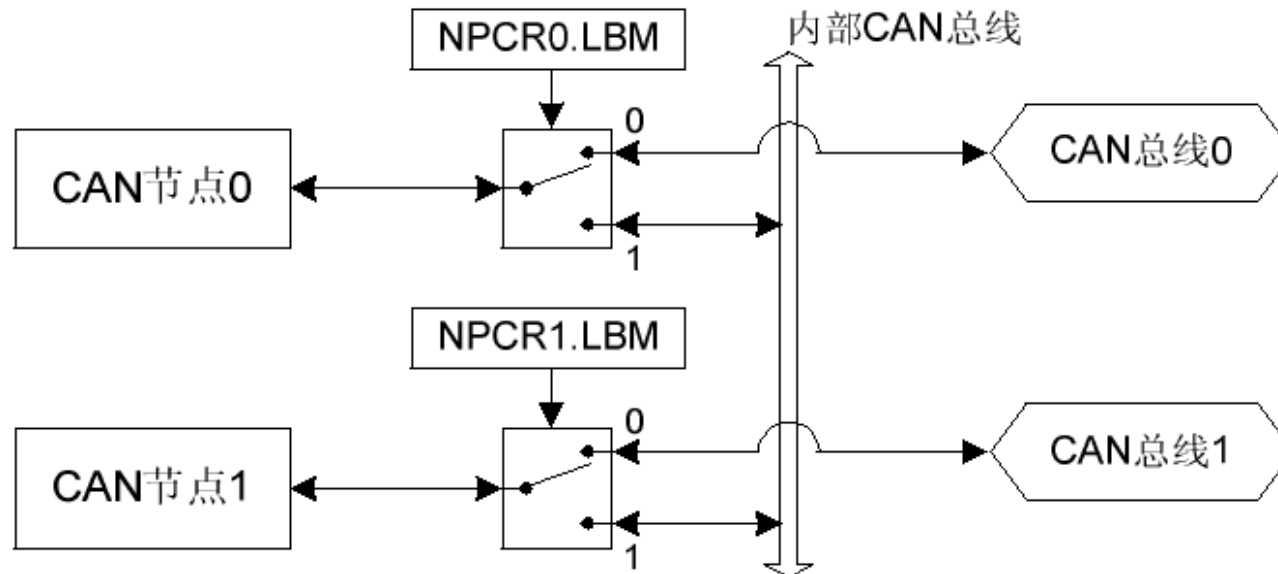
- 位域LIST.BEGIN指向列表中的第一个元素（例中的报文对象5），位域LIST.END指向列表中的最后一个元素（例中的报文对象3）。列表中的元素个数由位域LIST.SIZE给出（ $SIZE = \text{列表中的元素数} - 1$ ，因此例子中元素数为3， $SIZE = 2$ ）。位LIST.EMPTY指示该列表是否为空（因为列表2不为空，所以例中EMPTY = 0）
- 每个报文对象n有一个指针MOCTRn.PNEXT，用来指向列表中的下一个报文对象，同时还有一个指针MOCTRn.PPREV，用来指向列表中的前一个报文对象。第一个报文对象的PPREV指向其本身，因为第一个报文对象之前没有报文对象（例中的报文对象5为第一个报文对象，由PPREV = 5表示）。最后一个报文对象的PNEXT也指向其本身，因为最后一个报文对象之后没有报文对象（例中报文对象3为列表的最后一个报文对象，由PNEXT = 3表示）
- 位域 MOCTRn.LIST指示报文对象当前所属的列表编号，例中的报文对象都分配给了列表2。因此，所有的分配给列表2的报文对象的LIST位域都设置为LIST = 2。

- 分析模式：对CAN 通信进行监控，而不影响 CAN 总线逻辑状态。置位 NCRx.CALM，选择CAN分析模式。
- 在 CAN 分析模式下，CAN 节点的发送引脚永久保持为隐性电平。CAN 节点可以接收帧（数据，远程，和错误帧），但是不允许发送。接收到的数据/远程帧不会被应答（也就是说，应答间隙为隐性）但是只要其它任何一个节点应答了该帧，该帧将被接收和保存在匹配报文对象中。分析模式提供完整的报文对象功能，但是将不执行发送请求。
- 在分析模式下，节点只接受数据，不发送数据。在不影响总线使用的情况下，监控CAN总线的逻辑状态，分析在CAN总线上数据的传输。也同样用于数据的传输记录保存。

MultiCAN的组成

CAN的回环模式

- CAN的回环模式可以使用户无需访问外部 CAN 总线就可以进行 MultiCAN 模块的在系统测试和 CAN 驱动软件的开发。
- 置位 NPCRx.LBM 来选择回环模式。所有处于回环模式的 CAN 节点可以用内部总线进行通信，而不影响其它不处于回环模式的 CAN 节点的正常工作。



- 每个CAN节点都通过CAN帧计数器分析模式执行详细的位时序分析。帧计数器的位时序分析功能用于CAN波特率自动检测和CAN网络的时序分析。
- 通过设置 $\text{NFCRx.CFMOD} = 10_{\text{B}}$ ，选择位时序分析功能位时序分析不影响CAN节点的操作。位时序的测量结果被写入到位域NFCRx.CFC。在位时序分析模式下，每当位域NFCRx.CFC被更新，位NFCRx.CFCOV都被置位，用来指示 CFC 更新事件。如果NFCRx.CFCIE被置位，会产生一个中断请求。

■ 自动波特率检测

- 对于自动波特率检测，必须测量在CAN总线上观察到的连续显性沿之间的时间。如果位域NFCRx.CFSEL = 000_B，则自动执行该测量。监测CAN接收输入线上每一个显性沿，该边沿和最近的显性沿之间的时间（测量以f_{CAN}时钟周期数为单位）被储存在位域NFCRx.CFC中。

■ 同步分析

- 如果NFCRx.CFSEL = 010_B，则进行位时间同步监测。测量第一个显性沿和采样点之间的时间，并将其存储在位域NFCRx.CFC中。位时序同步偏移可以从该时间得出，因为采样点之后的第一个跳变沿触发同步，且在连续的采样点之间仅有一个同步。

■ 驱动延迟测量

- 当NFCRx.CFSEL = 011_B且NFCRx.CFSEL = 100_B时，测量发送沿和相应的接收沿之间的延迟。这些延迟指示将一个新的位值送到CAN总线上进行物理实现所需要的时间。

- 接收验收滤波：当从CAN节点接收CAN帧时，在成功接收之后，接收到的帧存储在唯一的一个报文对象中。只有满足下面六个条件的报文对象才具备接收CAN帧的资格：
 - 报文对象被分配给了要接收该帧的 CAN 节点的报文列表中。
 - MOSTATn.MSGVAL被置位。
 - MOSTATn.RXEN被置位。
 - 位MOSTATn.DIR等于接收帧的位 RTR。
 - 如果位MOSTATn.DIR = 1（发送对象），报文对象仅验收远程帧。如果位MOSTATn.DIR = 0（接收对象），报文对象仅验收数据帧。
 - 如果位 MOAMRn.MIDE = 1，用下列方法评估接收帧 IDE 位：如果 MOARn.IDE = 1，接收帧 IDE 位必须置位（表示扩展标识符）；如果 MOARn.IDE = 0，接收帧 IDE 必须清零（表示标准标识符）。如果 MOAMRn.MIDE = 0，不需要注意接收帧IDE位。在这种情况下，带有标准和扩展帧的报文对象都可以通过验收。
- 满足所有这六条评判标准的报文中，具有最高接收优先级的报文对象赢得接收验收滤波并被选中用来储存接收帧。所有其它报文对象失去接收验收滤波。

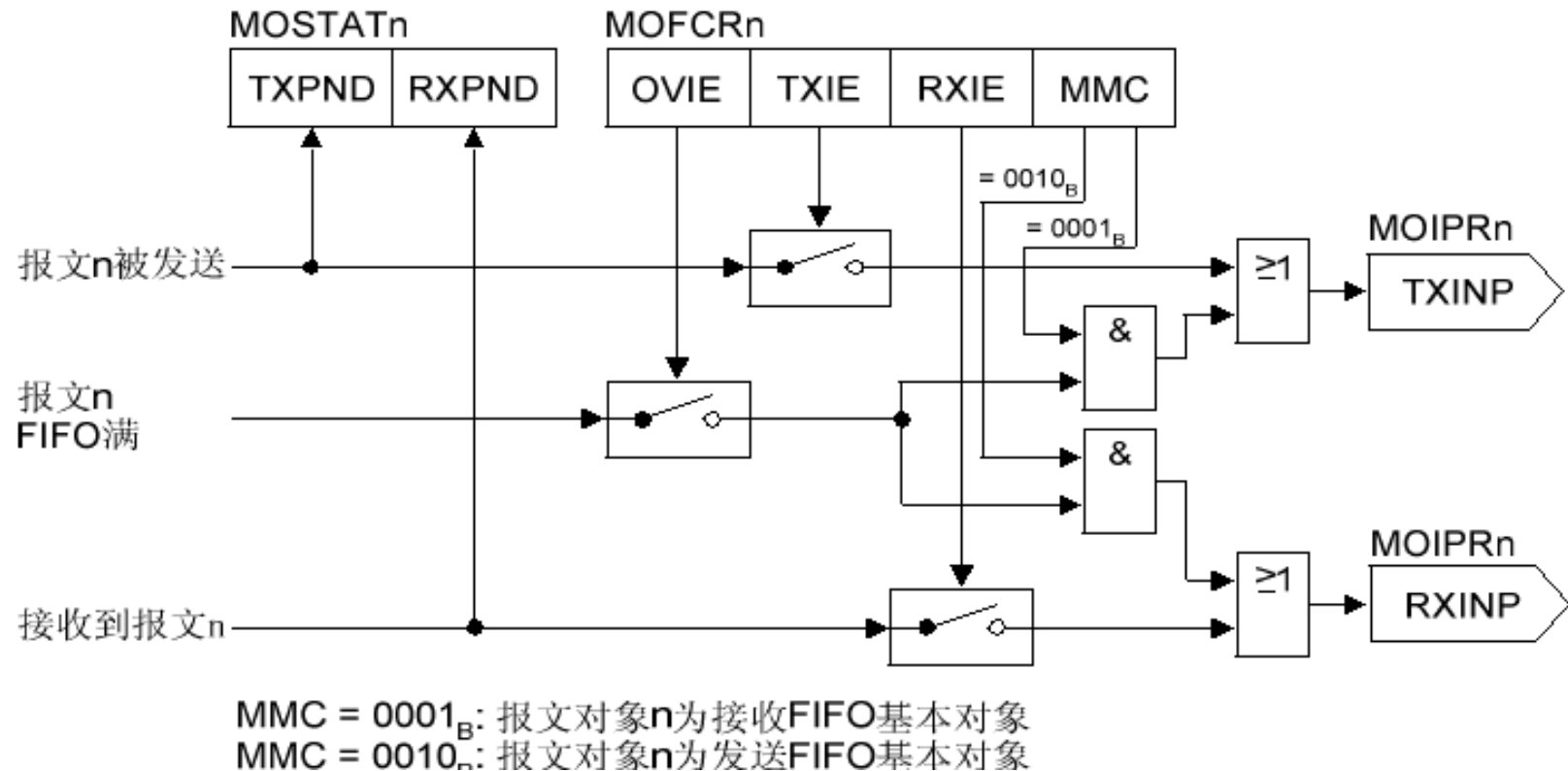
- 在CAN节点上，满足下述四个条件的报文对象才具备发送的资格：
 - 报文对象分配给了CAN节点的报文对象列表。
 - 位MOSTATn.MSGVAL被置位。
 - 位MOSTATn.TXRQ被置位。
 - 位MOSTATn.TXEN0和MOSTATn.TXEN1被置位。
- 由优先级机制判定最先发送哪一个报文对象。有发送资格的报文中发送优先级最高的报文对象赢得发送验收滤波，将最先被发送。所有其它的报文对象失去当前一轮的发送验收滤波机会。在下面几轮的验收滤波中他们都将得到新的机会，再次进行验收滤波。

- 在一个报文对象成功发送或者接收一帧后，可通知**CPU**在该报文对象上进行报文后处理。MultiCAN模块的后处理包括两个单元：
 - 报文中断触发报文后处理
 - 报文挂起寄存器将报文挂起中断收集到一个公共结构中，用于报文后处理。

MultiCAN的组成

报文中断触发报文后处理

- 当接收到的帧存储到一个报文对象中或者成功发送了一帧，可以发出报文中断。在一次成功的发送/接收之后，状态位MOSTATn.TXPND和MOSTATn.RXPND总是被置位。

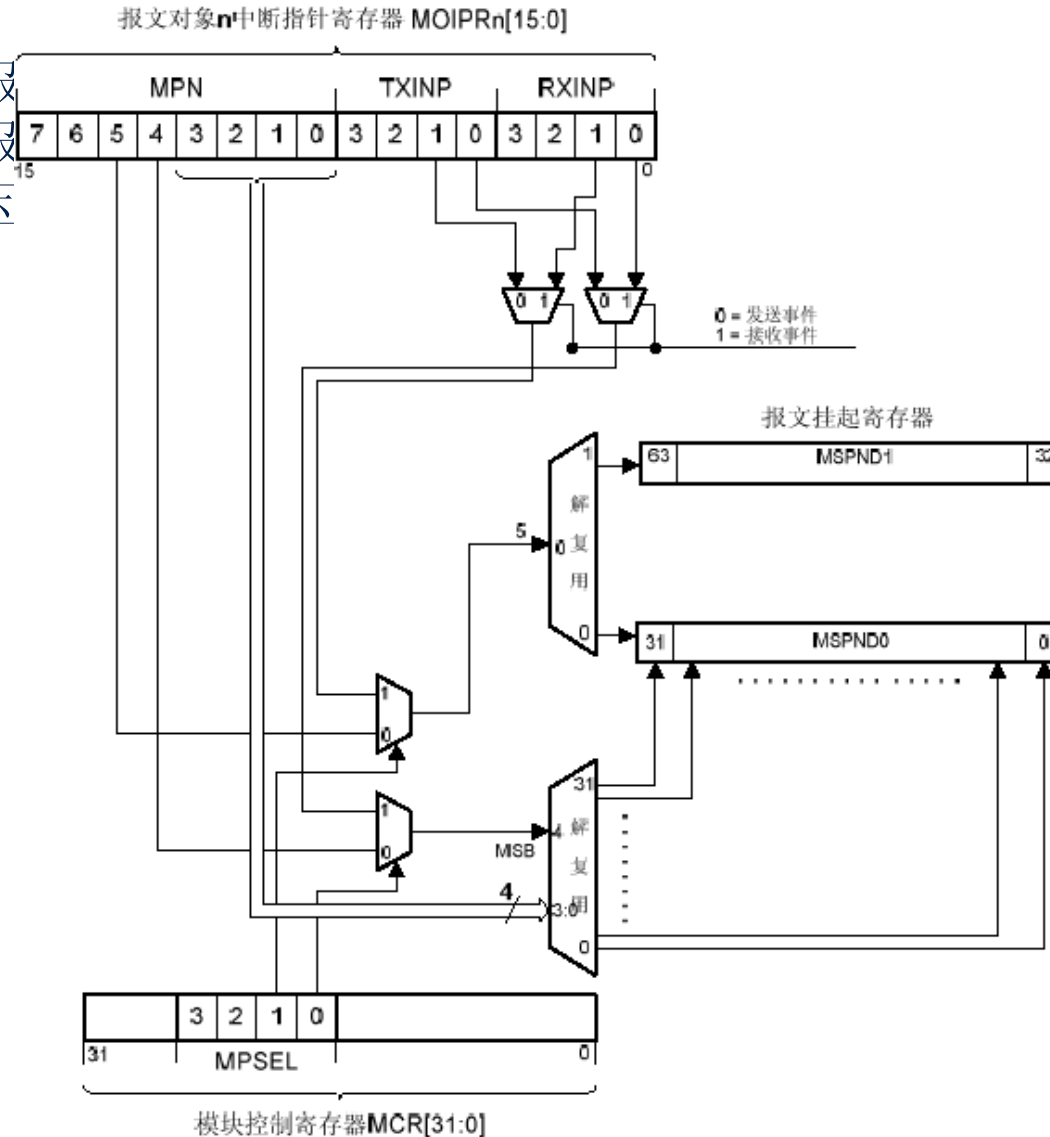


MultiCAN的组成

报文挂起

- 产生一个报
- 共有 2个报
- 挂起位，因

一个报文挂起位。
器有 32 个可用

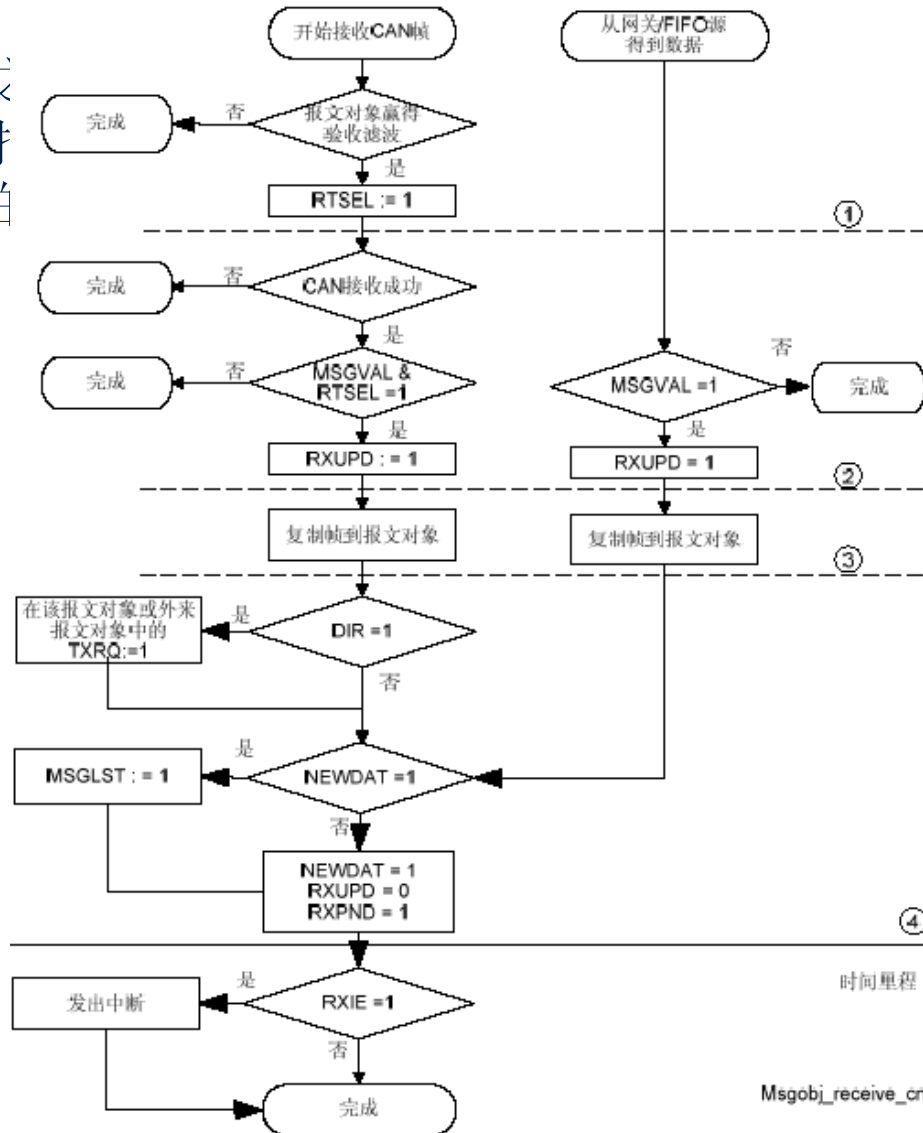


MultiCAN的组成

报文对象的接收

- 接收一个报文之
的数据复制到报
CPU之间一致的

模块不仅将接收到
实现MultiCAN和



■ 报文对象接收进程中各相关标志位的变化：

□ MSGVAL

- ✎ 帧接收期间，仅当MOSTATn.MSGVAL = 1，才在报文对象中存储信息。如果位MSGVAL被CPU复位，MultiCAN模块将停止所有正在进行的对该报文对象的写访问。

□ RTSEL

- ✎ 当一个报文对象赢得了接收验收滤波，MultiCAN模块置位RTSEL位，用来指示即将进行帧传送。MultiCAN模块检查RTSEL是否因为成功接收一帧而置位，以验证这个报文对象是否仍然准备就绪，可以接收帧。只有在RTSEL = 1时，接收到的帧被保存在报文对象中。

□ RXEN

- ✎ 位MOSTATn.RXEN用于使能报文对象的帧接收操作。仅当RXEN = 1时，报文对象才从CAN总线上接收CAN报文。只有在接收验收滤波期间，MultiCAN模块才评估RXEN的值。

□ RXUPD

- ─ 位MOSTATn.RXUPD（“接收更新”）用来指示正在进行的帧储存过程。对象更新开始时RXUPD置位，结束时清零RXUPD（包含帧存储和标志符更新）。

□ NEWDAT与MSGLST

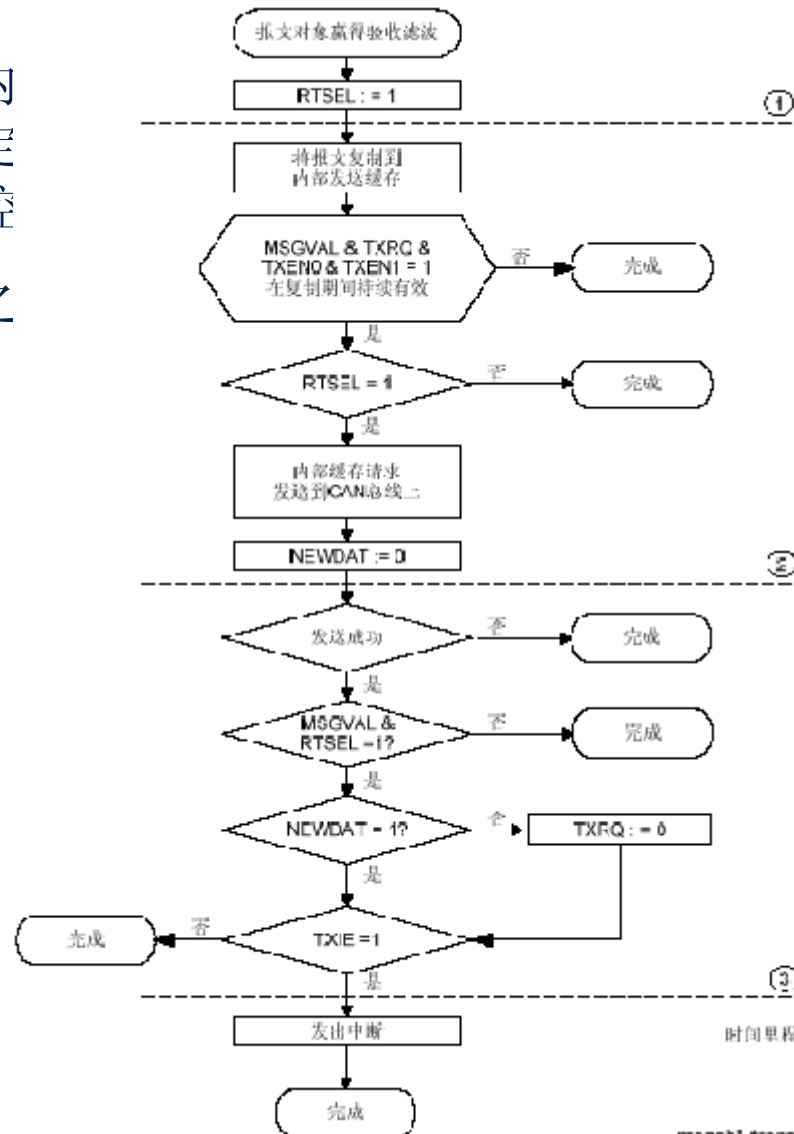
- ─ 在储存接收到的帧（标识符，IDE 位，DLC和数据帧的数据位域）之后，置位MOSTATn.NEWDAT（“新数据”）。如果再次置位之前NEWDAT已经被置位，置位MOSTATn.MSGLST（“报文丢失”）指示数据的丢失情况。

- 对于接收数据和远程帧，位 RXUPD、NEWDAT和MSGLST行为特征相同。

MultiCAN的组成

报文对象的发送

- 将要发送的报文内据域) 复制到指定态标志符, 用来控
- 在发送验收滤波之同。



- 仅在寄存器MOSTATn的下面4位:MSGVAL (“报文有效”), TXRQ (“发送请求”), TXEN0 (“发送使能 0”), TXEN1 (“发送使能1”)都置位的情况下，才能发送报文。
 - MSGVAL: 报文有效 该位是报文对象的主控制位
 - TXRQ: 发送请求 为标准发送请求位。只要一个报文对象要进行发送，就必须置位该位。在一次成功发送结束时，除非该报文还有新数据（由 NEWDAT = 1 指示）要发送，否则 TXRQ 被硬件清零。
 - TEXN0: 发送使能0 当 TXEN0 = 0 时，仍接受远程请求，但是数据帧的发送被挂起，直到软件再次置位发送使能。
 - TEXN1: 发送使能1 发送 FIFO 中，用该位选择发送有效的报文对象。

■ 报文对象发送进程中各相关标志位的变化：

□ RTSEL

- 在发送验收滤波之后，当已经验证一个报文对象就是下一个要被发送的报文对象时，置位MOCTRn.RTSEL。当报文对象被复制到内部发送缓存中时，检查位RTSEL，仅在RTSEL = 1时，才发送报文。在成功发送报文之后，再次检查位RTSEL，只有RTSEL = 1，才会执行报文后处理。

□ NEWDAT

- 当报文对象的内容被传送到CAN节点的内部发送缓存中之后硬件清零位MOSTATn.NEWDAT（新数据），指示发送报文对象中的数据不再是新值。当帧发送成功且NEWDAT仍然为0（也就是说，其间没有新数据被复制到报文对象中），硬件自动清零TXRQ（发送请求）。然而，如果软件再次置位NEWDAT（因为要发送新的一帧），不清零TXRQ，以使能新数据发送。

- 标准报文对象：当位域MOFCRn.MMC = 0000_B，选择标准报文对象。
- 单一数据传送模式：在CAN总线上广播数据为了避免发送重复信息，采用单一数据传送模式。通过位MOFCRn.SDT选择该模式。
- 报文对象FIFO结构：CPU高负荷的情况下，及时的处理一串CAN帧可能比较困难。该结构可以用于在短时间内发送或者接收多个报文。
- 网关模式：网关模式在两个独立CAN总线之间建立自动信息传送，而不需要和CPU相互作用。

■ 报文接收：

- 当存储在报文对象中的接收报文被新接收到的报文覆盖，第一个报文的内容丢失，且被接收到的新值取代（由 **MSGLST = 1** 指示）。
- 单一数据传送模式（**SDT = 1**），在存储了接收到的数据帧之后，硬件自动清零位**MSGVAL**。该操作阻止了后续的报文接收。
- 在接收远程帧之后，位 **MSGVAL** 不会被自动清零。

■ 报文发送：

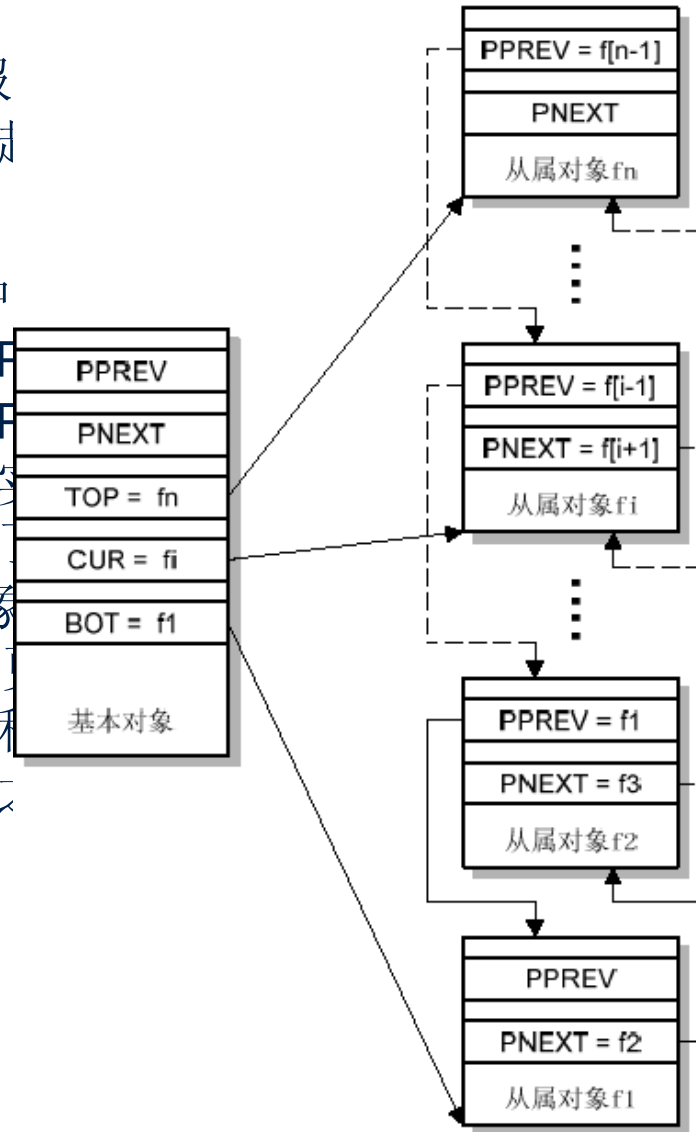
- 当接收到一系列远程请求时，报文对象发送几个数据帧来回应远程请求。如果报文对象中的数据在两次发送期间未被更新，同样的数据在 **CAN** 总线上发送次数可以多于一次。
- 在单一数据传送模式（**SDT = 1**），因为在成功发送一个数据帧之后，自动清零**MSGVAL**从而避免了数据被再次发送。
- 在发送远程帧之后，位 **MSGVAL** 不会被自动清零。

MultiCAN的组成

报文对象的FIFO结构

- FIFO由一个基本报文列表结构链接在一个列表。

- 在FIFO基本对象中
向（包括编号）FIFO
指向（包括编号）FIFO
指向（包括编号）FIFO
当在该对象中出现
（CUR = 当前对象
部），CUR的下次
这种机制代表了一个
个和最后一个元素



。从属报文对象由一个本对象可以分配给任意

位域MOFGPRn.BOT指
象。位域MOFGPRn.TOP
象。位域MOFGPRn.CUR
于报文传送的从属对象。
中下一个从属对象。
OP（到达了FIFO顶
FO顶部绕回到底部）。
OP建立了FIFO中第一

MultiCAN的组成

报文对象的FIFO结构-接收FIFO

- 接收FIFO结构用于缓存接收到的远程或数据帧。
- 在FIFO基本对象中设置MOFCRn.MMC = 0001_B，选择接收FIFO。MMC码自动指定该报文对象为FIFO基本对象FIFO从属报文对象的报文模式和接收FIFO操作无关。
- 当FIFO基本对象从其所属的CAN节点接收一帧时，该帧并不存储在基本对象中，而是存储在基本对象MOFGPRn.CUR指针所选择的从属报文对象中，就像是由从属报文对象直接接收该帧一样。然而，MOCFRn.MMC = 0000_B隐含地假设FIFO从属对象执行的是标准报文传送。忽略FIFO从属对象实际的报文模式（MMC设置）。从属对象不对接收到的帧进行验收滤波，即不检查匹配标识符，IDE位和DIR位。
- 接收一个CAN帧之后，基本对象的当前指针CUR被设置为FIFO结构中的下一个报文对象的编号。这个报文对象将用来储存下一个接收报文。

MultiCAN的组成

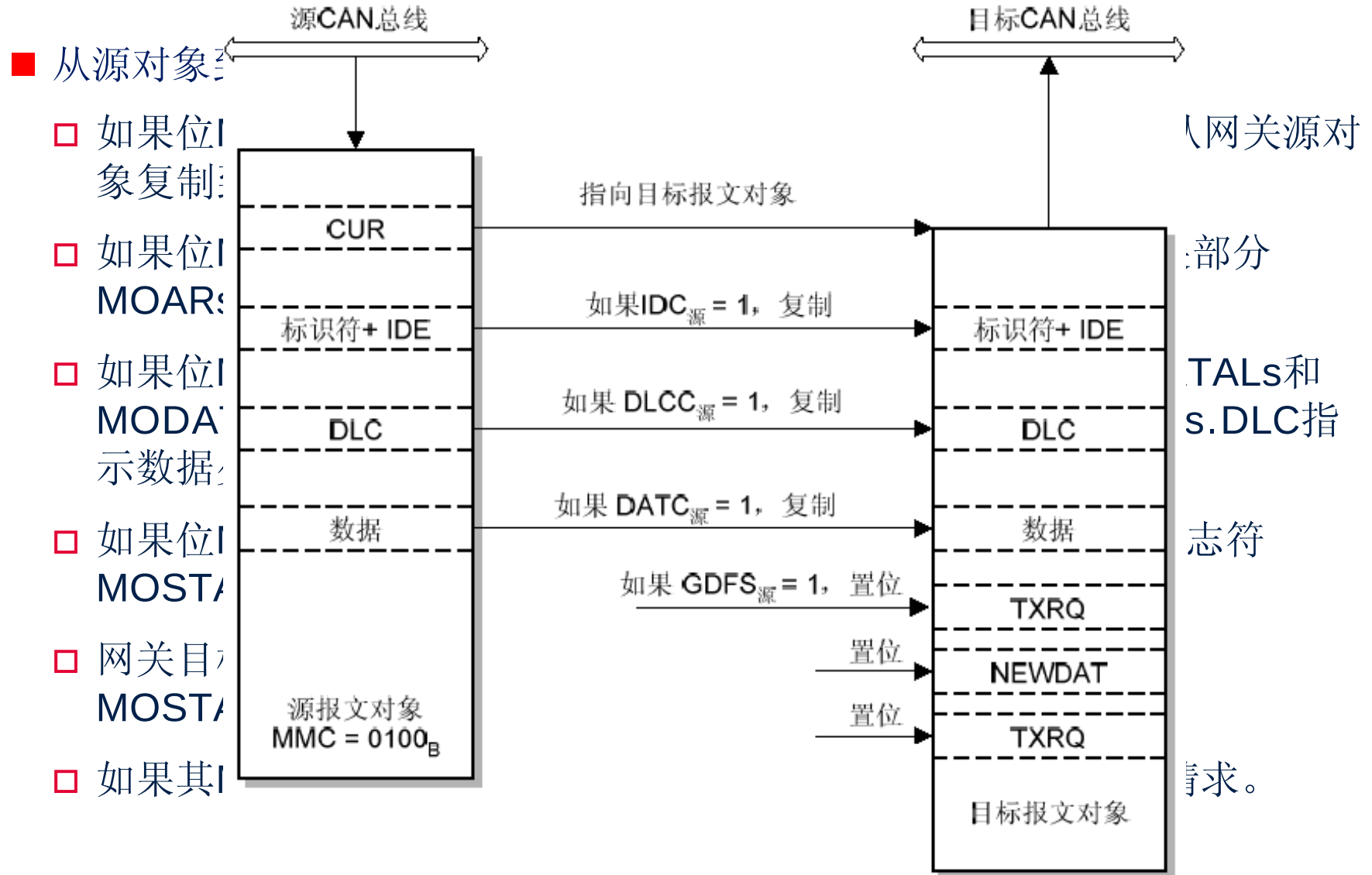
报文对象的FIFO结构-发送FIFO

- 在FIFO基本对象中，通过设置MOFCRn.MMC = 0010_B，选择发送FIFO。与接收FIFO不同，分配给发送FIFO的从属对象需要明确地设置它们的位域MOFCRn.MMC = 0011_B。所有从属对象中的CUR指针都必须指回到发送FIFO基本对象（由软件初始化）。
- 除了基本对象的CUR指针选择的报文对象，其它所有报文对象的MOSTATn.TXEN1（“发送使能 1”）都必须用软件清零。CUR选择的报文对象（从属对象）的TXEN1必须被置位。
- 发送FIFO用所有的FIFO元素的MOCTRn.TXEN1位选择实际要发送的报文。发送验收滤波评估每个报文对象的TXEN1，且只有其TXEN1置位的报文对象可以赢得发送验收滤波。当FIFO报文对象已经发送一帧报文，硬件除了执行标准的发送后处理（清零TXRQ，发送中断等等），还对TXEN1位清零，移动CUR指针使其指向下一个要发送的报文对象。硬件自动置位下一个报文对象的TXEN1。

- 网关模式在报文对象一级上工作。在网关模式下，信息的传送是在两个报文对象之间进行，结果这两个报文对象所属的**CAN**节点之间也进行了信息传送。网关可建立在任意一对**CAN**节点之间，可以建立网关的个数和可用于构造网关结构的报文对象的个数一样多。
- 通过设置网关源对象s的位域 $\text{MOCFCRs.MMC} = 0100_{\text{B}}$ ，选择网关模式。由源报文对象s的 MOFGPRd.CUR 指针选择网关目标报文对象d。目标对象只需要设为有效（其 $\text{MSGVAL} = 1$ ）。所有其它设置都与源和目标对象之间的信息传送无关。

MultiCAN的组成

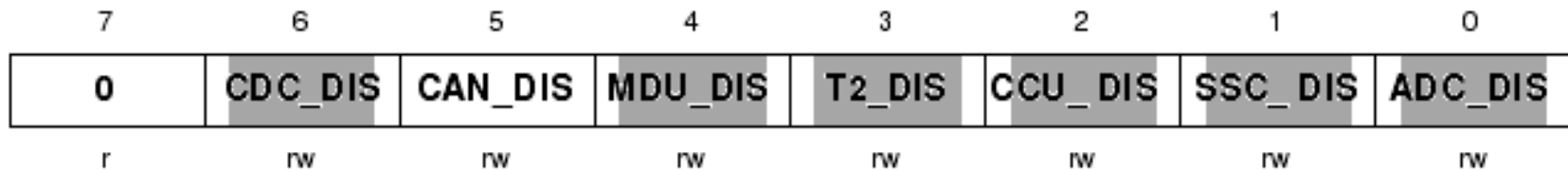
网关模式（续）



MultiCAN的组成

CAN的低功耗模式

- 如果完全不需要MultiCAN功能，可关闭其输入时钟最大限度的降低功耗。该模式通过置位寄存器PMCON1中的位CAN_DIS来实现。



PMCON1

功率模式控制寄存器1

CAN_DIS

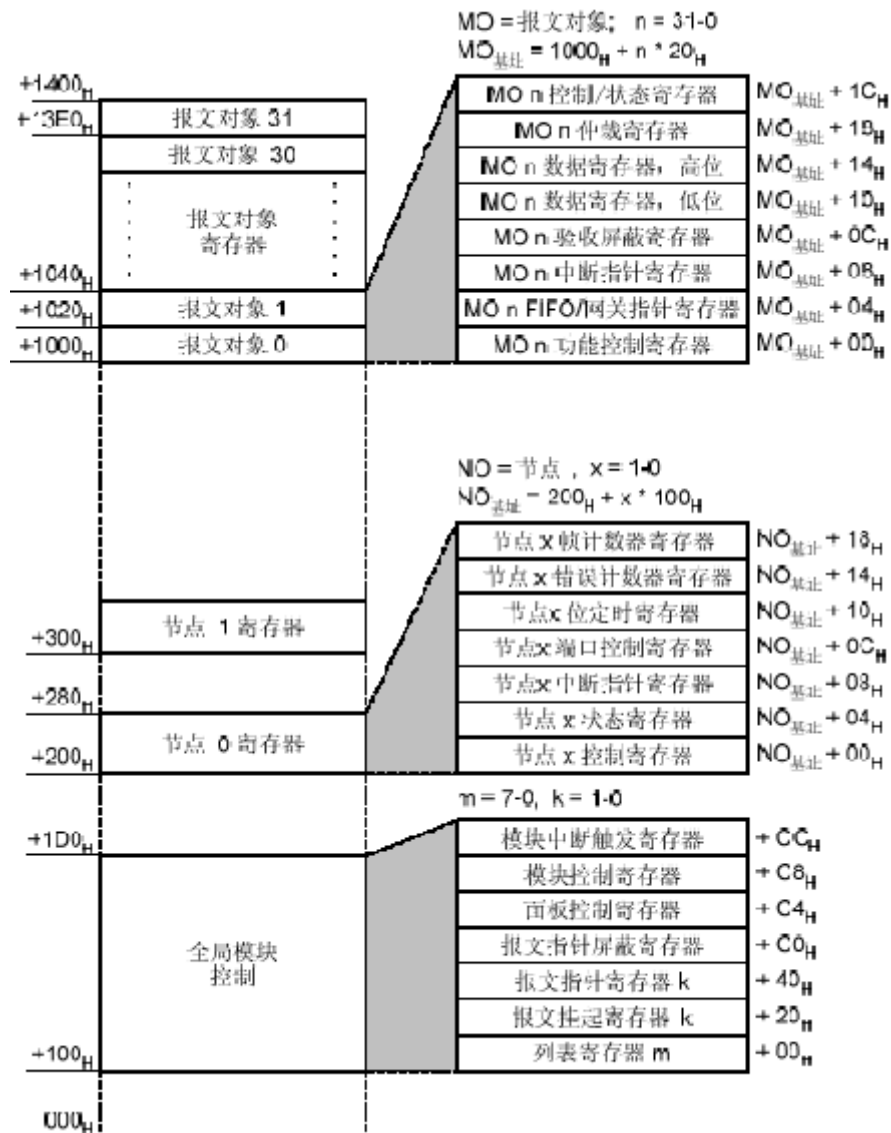
CAN禁止请求位，高有效

0 CAN正常工作（缺省状态）

1 请求禁止CAN

MultiCAN的组成

内核寄存器地址映射图



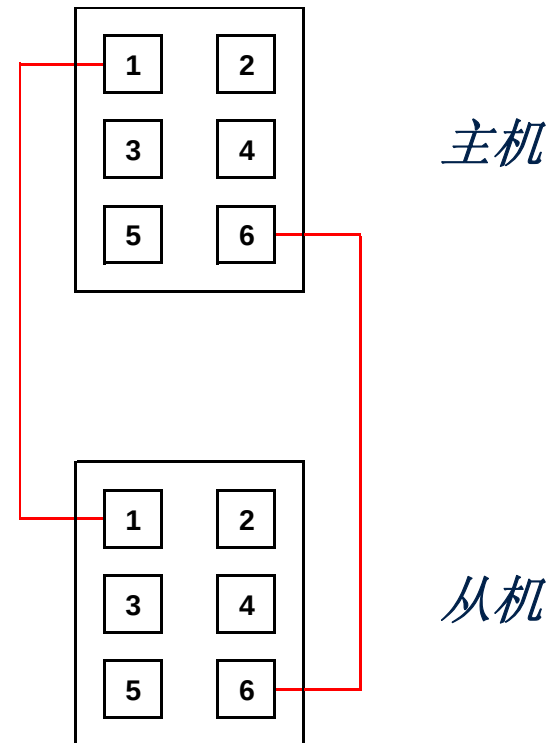
MultiCAN的运用

实战练习：LED灯控实验

- 为了便于了解MultiCAN的工作流程，下面介绍利用MultiCAN控制LED灯的亮灭程序。
- 在该程序中，选择使用XC888CLM单片机，在主机PORT3口输出LED亮灭信息，再利用100Kbps的MultiCAN传输信息到从机器，然后在从机输出LED亮灭信息。
- 硬件连接图：

Pin Assignment:

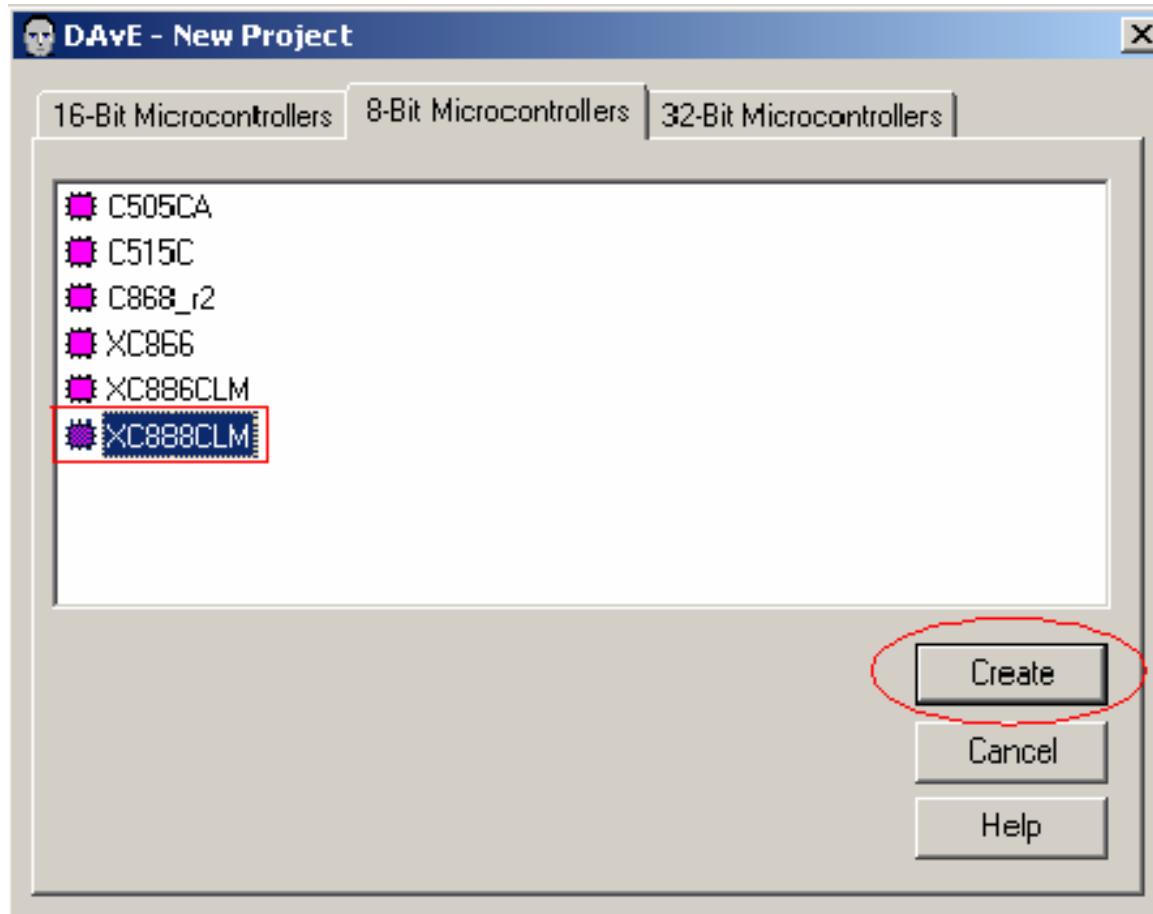
Pin	Signal name	Comment
1	CANL1	
2	GND	Ground
3	CANL0	
4	CANH0	
5	GND	Ground
6	CANH1	



MultiCAN的运用

实战练习：报文的发送

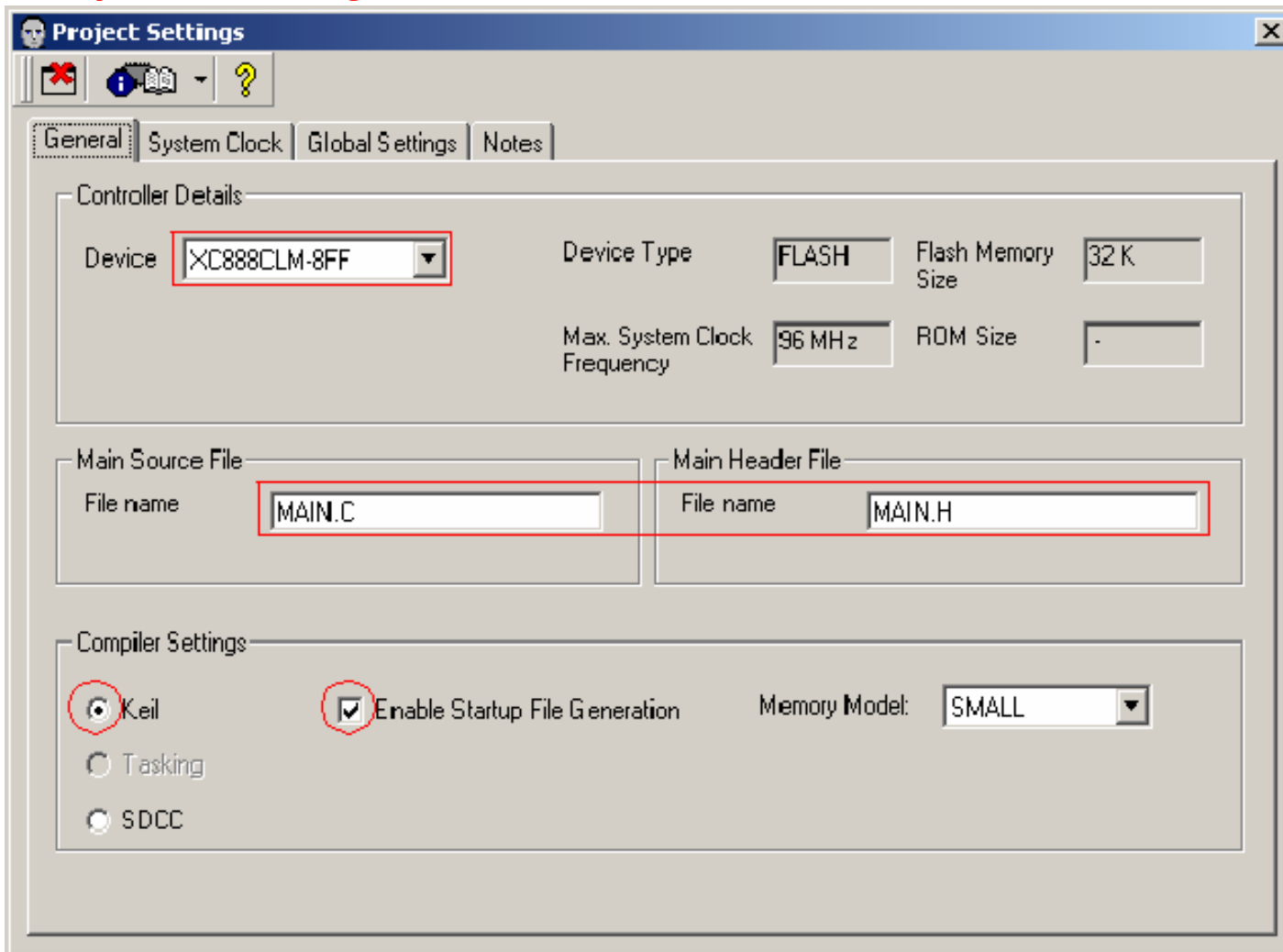
- 首先，使用DAVE软件创建一个工程。



MultiCAN的运用

实战练习：报文的发送（续）

- 进入“**Project Settings - General**”页，进行项目设定。



Project Settings

General | System Clock | Global Settings | Notes

Controller Details

Device: **XC888CLM-8FF** Device Type: FLASH Flash Memory Size: 32 K

Max. System Clock Frequency: 96 MHz ROM Size: .

Main Source File

File name: **MAIN.C**

Main Header File

File name: **MAIN.H**

Compiler Settings

☒ Keil ☒ Enable Startup File Generation Memory Model: SMALL

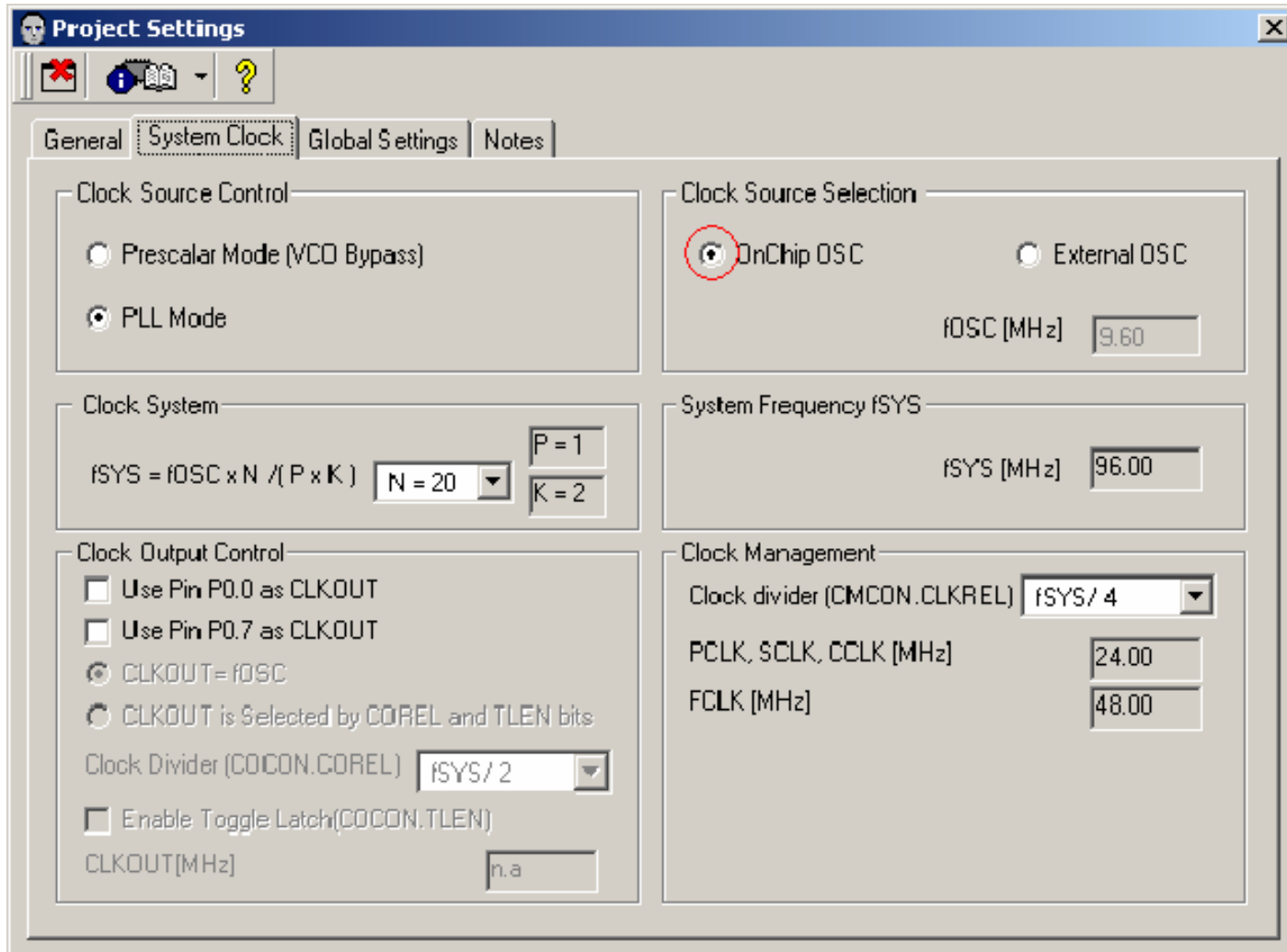
☐ Tasking

☐ SDCC

MultiCAN的运用

实战练习：报文的发送（续）

- 进入“**System Clock**”页，配置时钟。



The screenshot shows the 'Project Settings' dialog box with the 'System Clock' tab selected. The dialog is divided into several sections for configuring the clock system.

General | **System Clock** | Global Settings | Notes

Clock Source Control

- ☐ Prescaler Mode (VCO Bypass)
- ☒ PLL Mode

Clock Source Selection

- ☒ OnChip OSC
- ☐ External OSC

fOSC [MHz] 9.60

Clock System

fSYS = fOSC x N / (P x K) P = 1 N = 20 K = 2

System Frequency fSYS

fSYS [MHz] 96.00

Clock Output Control

- ☐ Use Pin P0.0 as CLKOUT
- ☐ Use Pin P0.7 as CLKOUT
- ☒ CLKOUT = fOSC
- ☐ CLKOUT is Selected by COREL and TLEN bits

Clock Divider (COCON.COREL) fSYS / 2

☐ Enable Toggle Latch(COCON.TLEN)

CLKOUT [MHz] n.a

Clock Management

Clock divider (CMCON.CLKREL) fSYS / 4

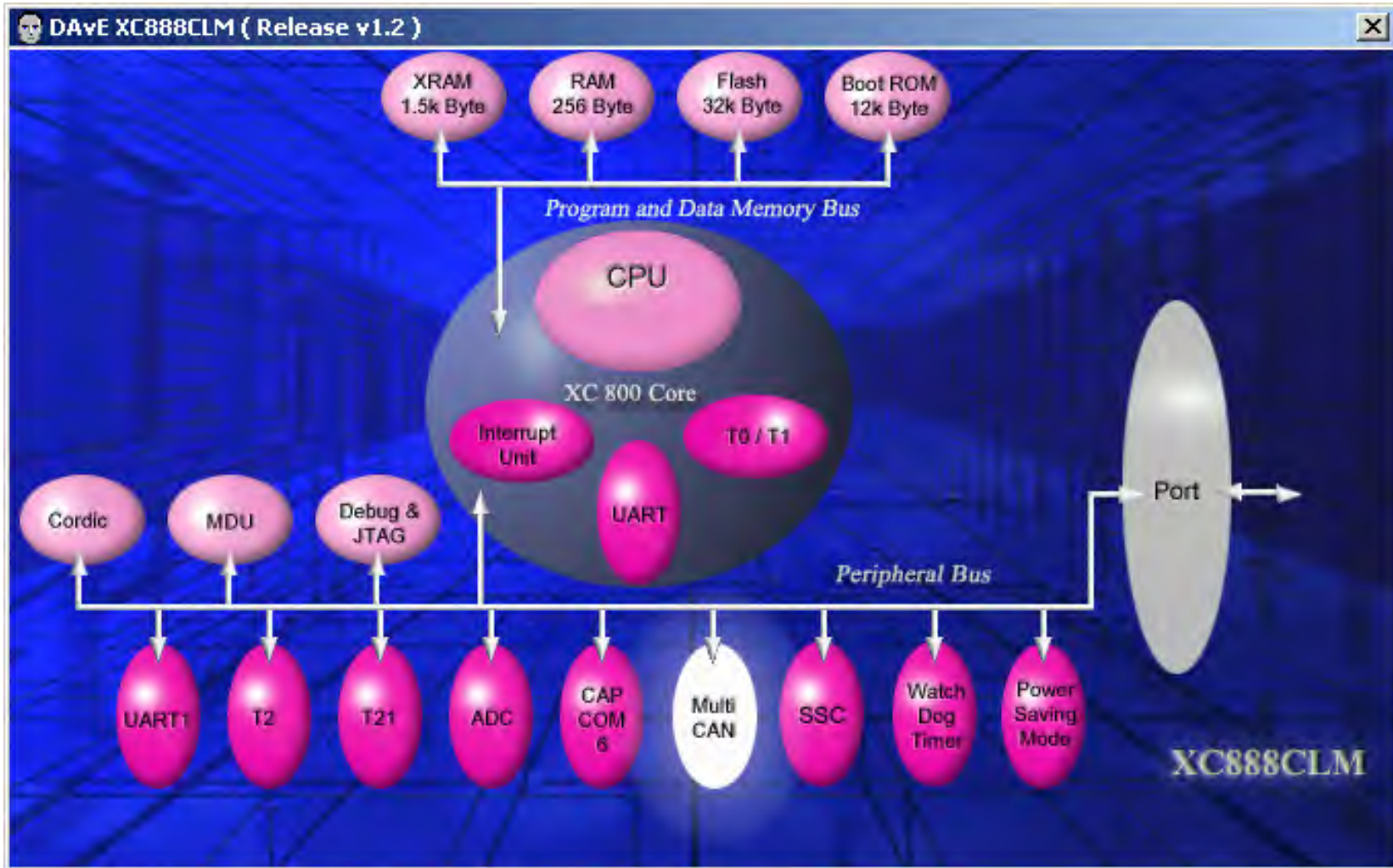
PCLK, SCLK, CCLK [MHz] 24.00

FCLK [MHz] 48.00

MultiCAN的运用

实战练习：报文的发送（续）

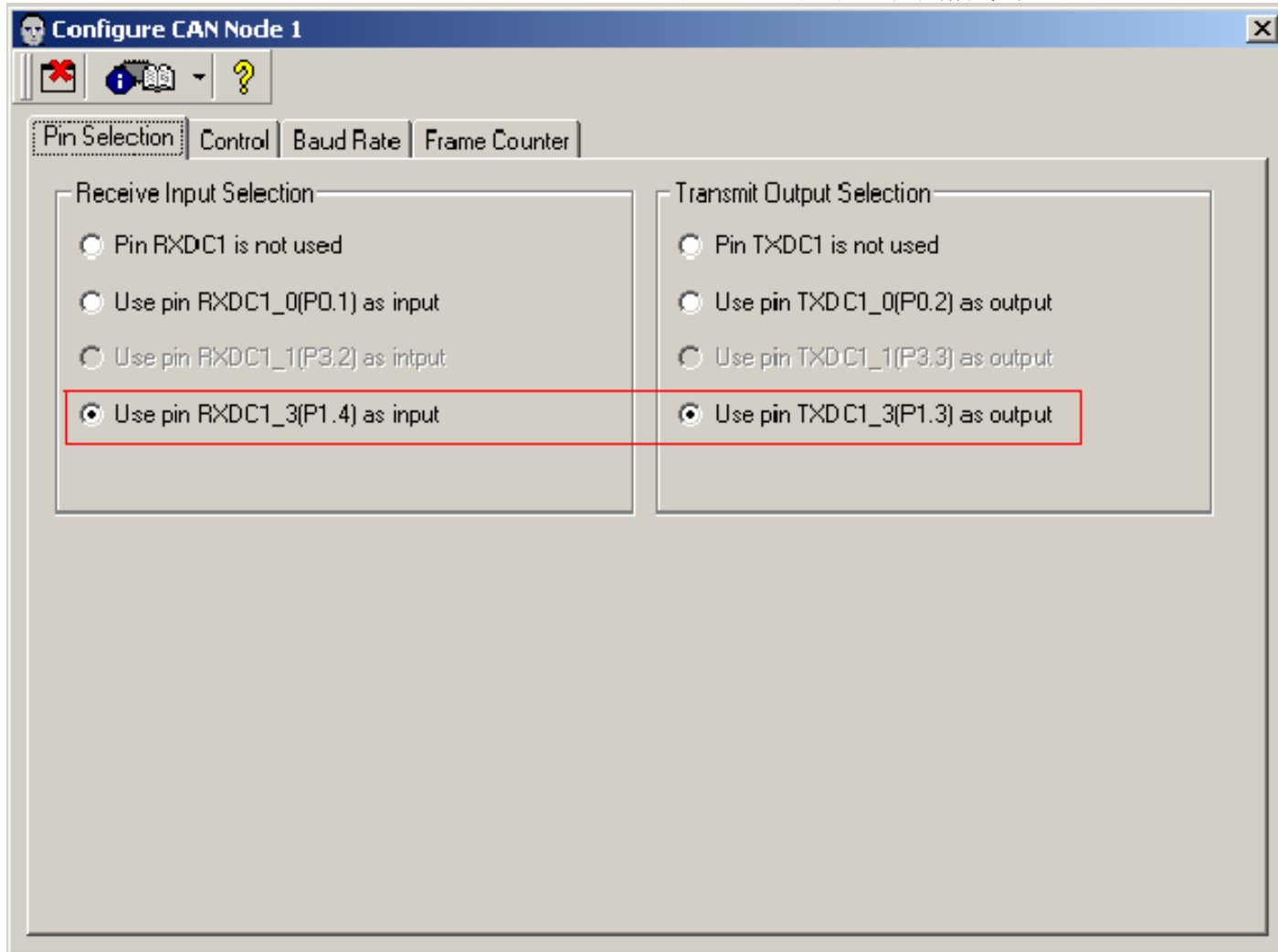
- 选择MultiCAN单元，进行一些简单设定：



MultiCAN的运用

实战练习：报文的发送（续）

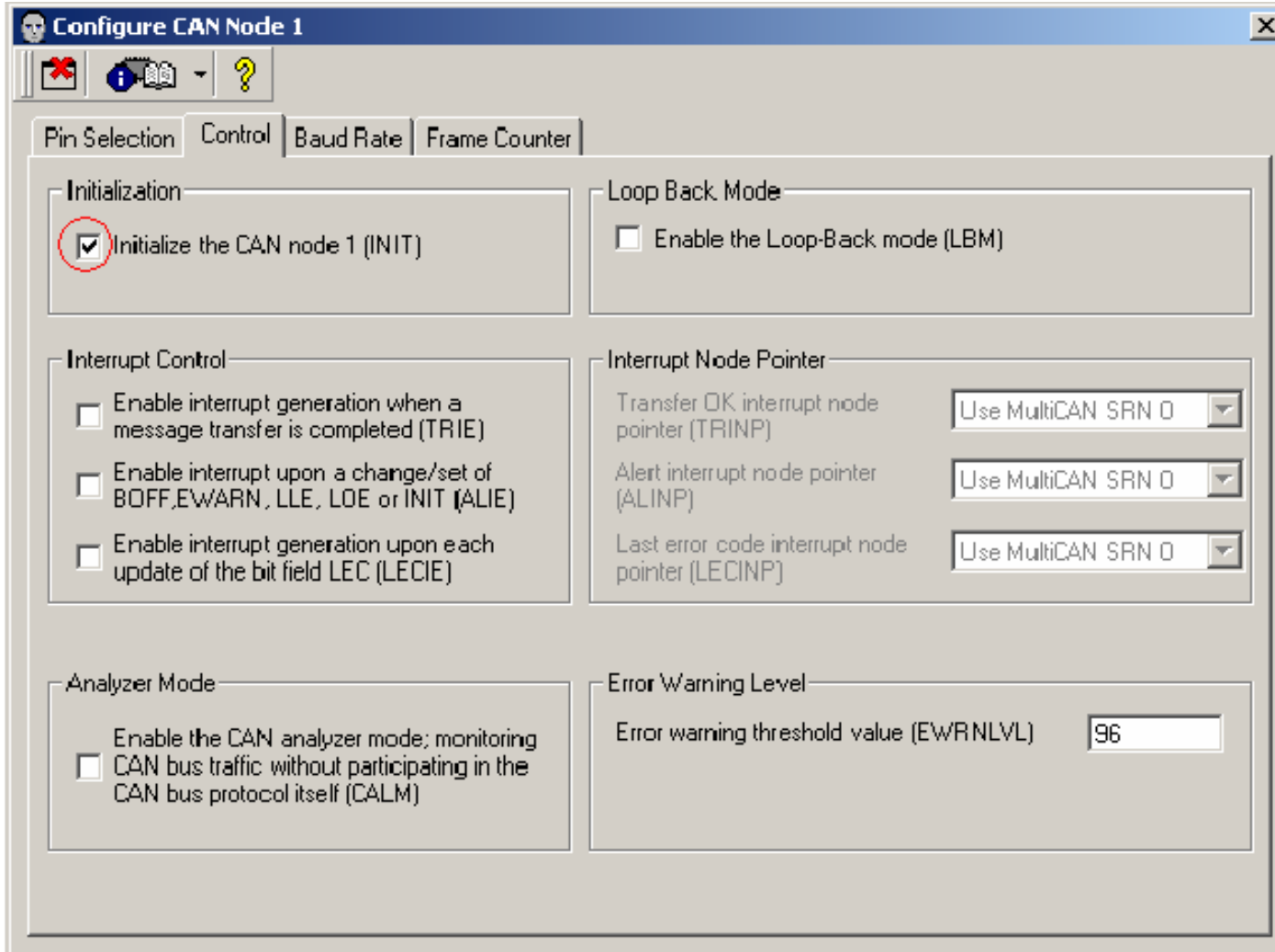
- 选择“Node 1”，进入“Pin Selection”页，设定传输使用IO口：



MultiCAN的运用

实战练习：报文的发送（续）

- 进入“Control”页，设定CAN模式。



Configure CAN Node 1

Pin Selection | **Control** | Baud Rate | Frame Counter

Initialization

- ☒ Initialize the CAN node 1 (INIT)

Loop Back Mode

- ☐ Enable the Loop-Back mode (LBM)

Interrupt Control

- ☐ Enable interrupt generation when a message transfer is completed (TRIE)
- ☐ Enable interrupt upon a change/set of BOFF, EWARN, LLE, LOE or INIT (ALIE)
- ☐ Enable interrupt generation upon each update of the bit field LEC (LECIE)

Interrupt Node Pointer

- Transfer OK interrupt node pointer (TRINP): Use MultiCAN SRN 0
- Alert interrupt node pointer (ALINP): Use MultiCAN SRN 0
- Last error code interrupt node pointer (LECINP): Use MultiCAN SRN 0

Analyzer Mode

- ☐ Enable the CAN analyzer mode; monitoring CAN bus traffic without participating in the CAN bus protocol itself (CALM)

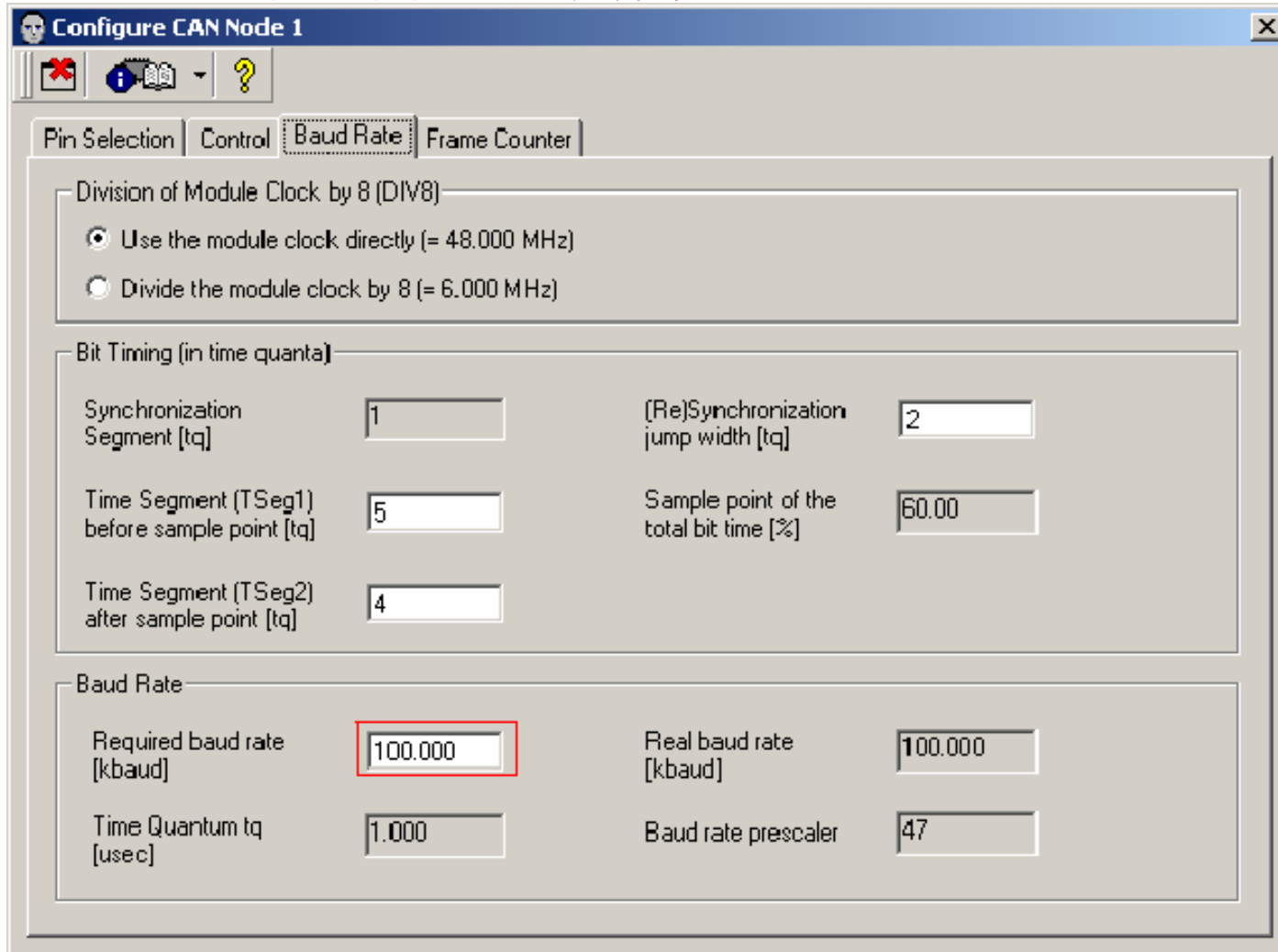
Error Warning Level

- Error warning threshold value (EWRNLVL): 96

MultiCAN的运用

实战练习：报文的发送（续）

- 进入“**Baud Rate**”页，设定CAN波特率。



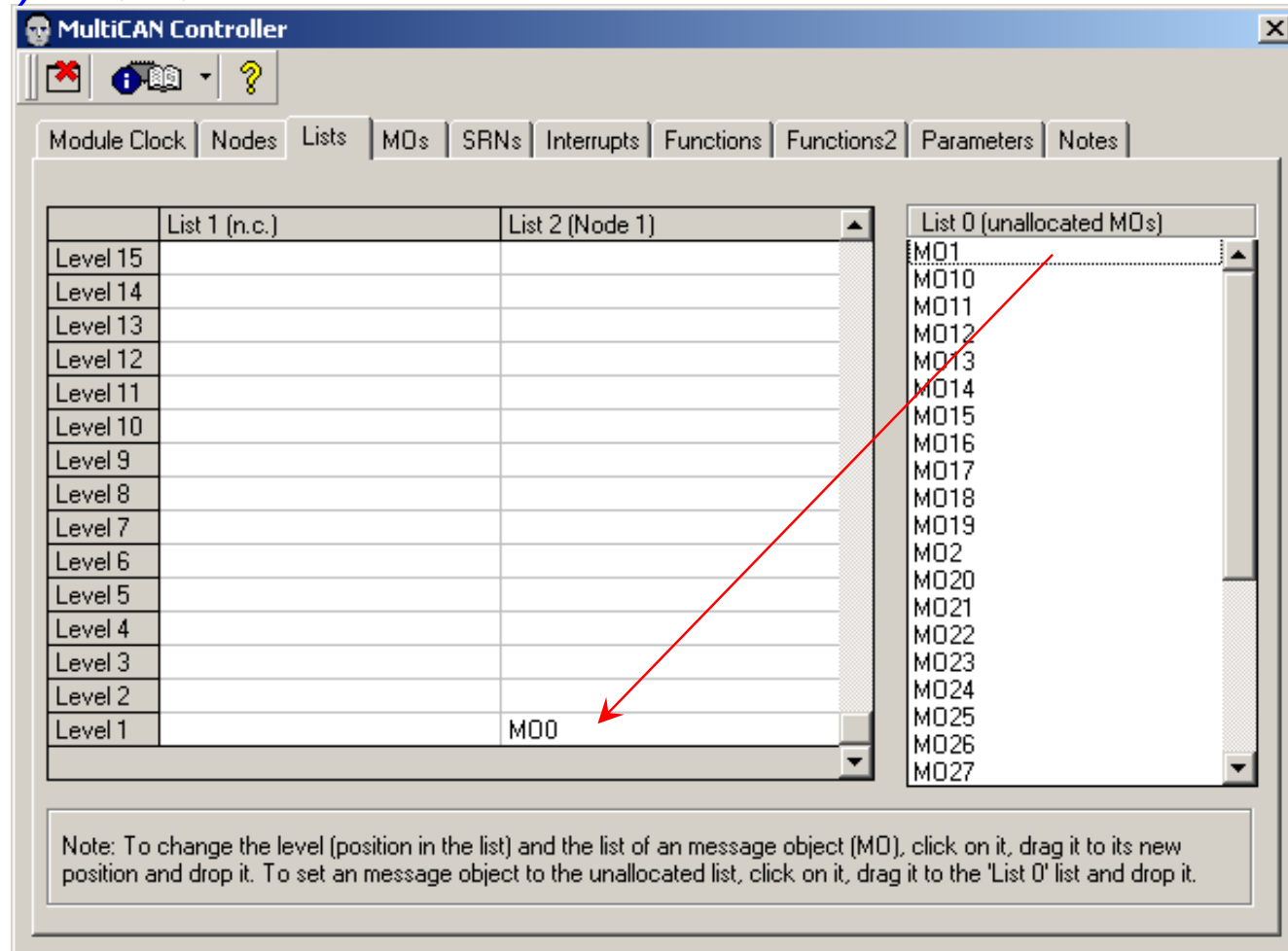
The screenshot shows the 'Configure CAN Node 1' dialog box with the 'Baud Rate' tab selected. The dialog has a title bar with a close button and a toolbar with icons for help, save, and cancel. Below the tabs, there are three sections: 'Division of Module Clock by 8 (DIV8)', 'Bit Timing (in time quanta)', and 'Baud Rate'. The 'Baud Rate' section contains four input fields: 'Required baud rate [kbaud]' (100.000), 'Real baud rate [kbaud]' (100.000), 'Time Quantum tq [usec]' (1.000), and 'Baud rate prescaler' (47). The 'Required baud rate' field is highlighted with a red rectangle.

Section	Parameter	Value
Division of Module Clock by 8 (DIV8)	Use the module clock directly (= 48.000 MHz)	☒
	Divide the module clock by 8 (= 6.000 MHz)	☐
Bit Timing (in time quanta)	Synchronization Segment [tq]	1
	(Re)Synchronization jump width [tq]	2
	Time Segment (TSeg1) before sample point [tq]	5
	Time Segment (TSeg2) after sample point [tq]	4
Baud Rate	Required baud rate [kbaud]	100.000
	Real baud rate [kbaud]	100.000
	Time Quantum tq [usec]	1.000
	Baud rate prescaler	47

MultiCAN的运用

实战练习：报文的发送（续）

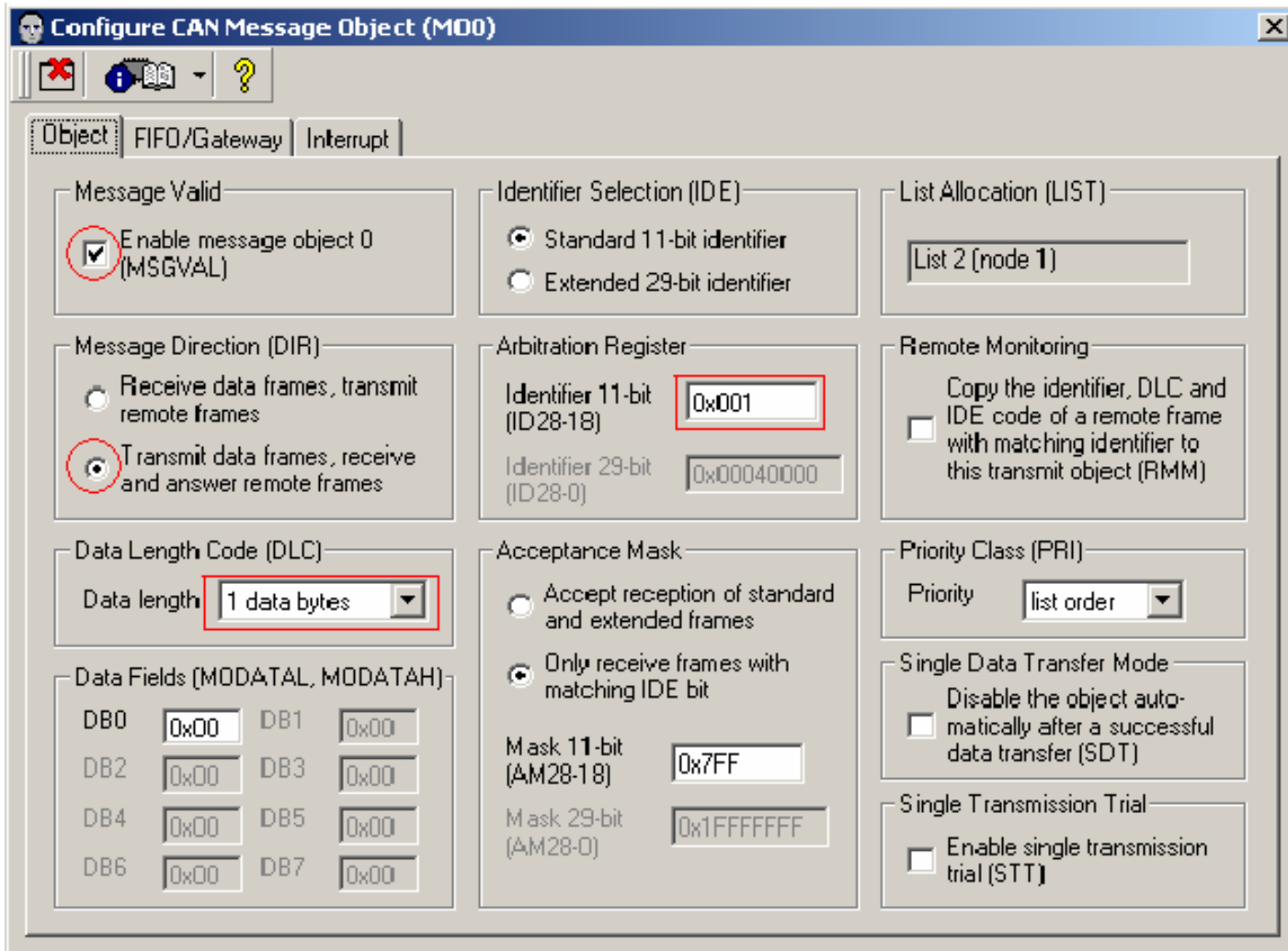
- 在“Lists”页中，将MO0从“List 0 (unallocated-MOs)”一栏中拖至“List 2 (Node 1)”一栏中。



MultiCAN的运用

实战练习：报文的发送（续）

- 进入“**MOs**”页，选择**MO0**，进行信息传输的设定。



Configure CAN Message Object (MO0)

Object FIFO/Gateway Interrupt

Message Valid

- ☒ Enable message object 0 (MSGVAL)

Message Direction (DIR)

- ☐ Receive data frames, transmit remote frames
- ☒ Transmit data frames, receive and answer remote frames

Data Length Code (DLC)

Data length 1 data bytes

Data Fields (MODAT0L, MODATAH)

DB0	0x00	DB1	0x00
DB2	0x00	DB3	0x00
DB4	0x00	DB5	0x00
DB6	0x00	DB7	0x00

Identifier Selection (IDE)

- ☒ Standard 11-bit identifier
- ☐ Extended 29-bit identifier

Arbitration Register

Identifier 11-bit (ID28-18) 0x001

Identifier 29-bit (ID28-0) 0x00040000

Acceptance Mask

- ☐ Accept reception of standard and extended frames
- ☒ Only receive frames with matching IDE bit

Mask 11-bit (AM28-18) 0x7FF

Mask 29-bit (AM28-0) 0x1FFFFFFF

List Allocation (LIST)

List 2 (node 1)

Remote Monitoring

- ☐ Copy the identifier, DLC and IDE code of a remote frame with matching identifier to this transmit object (RMM)

Priority Class (PRI)

Priority list order

Single Data Transfer Mode

- ☐ Disable the object automatically after a successful data transfer (SDT)

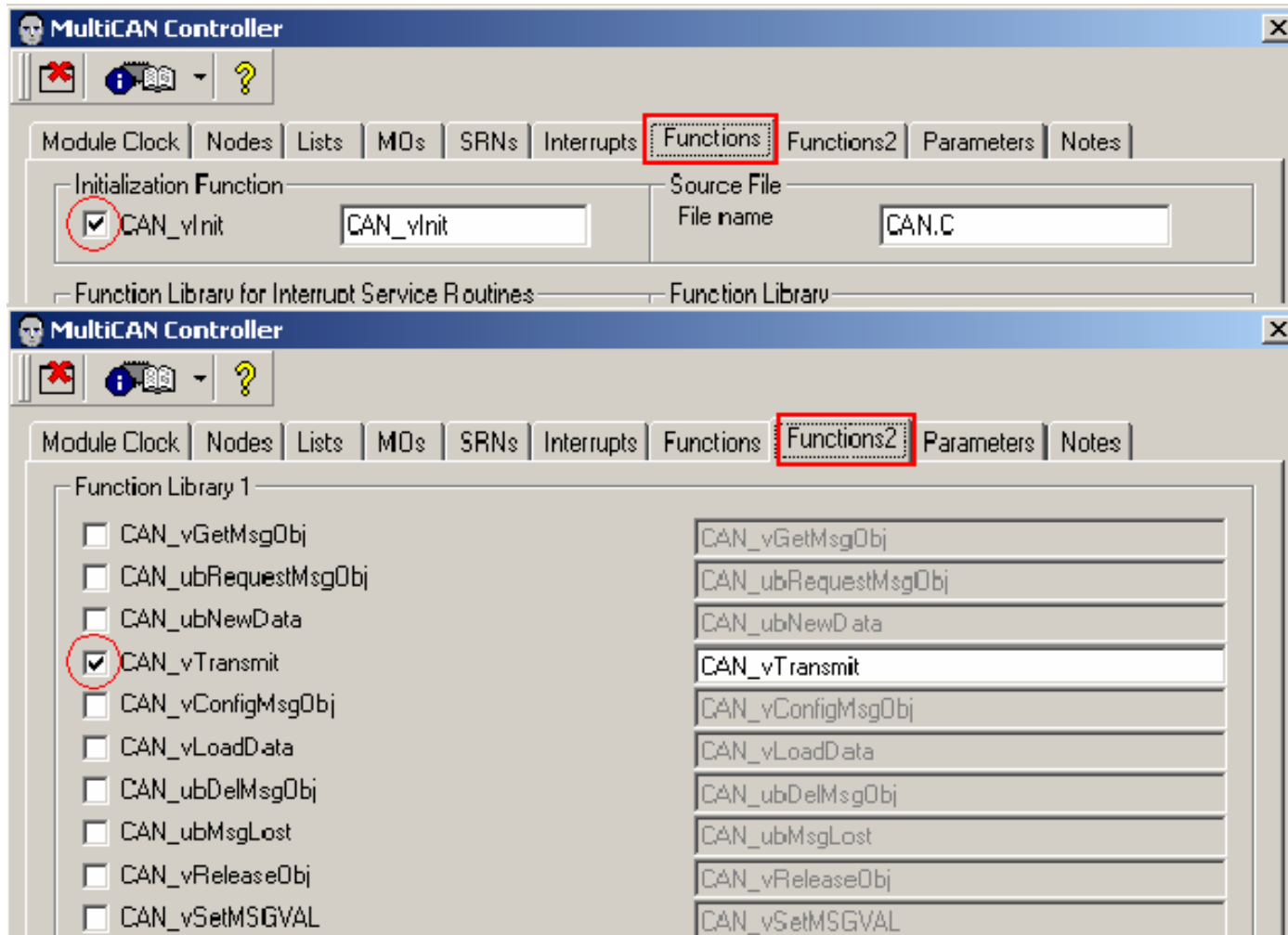
Single Transmission Trial

- ☐ Enable single transmission trial (STT)

MultiCAN的运用

实战练习：报文的发送（续）

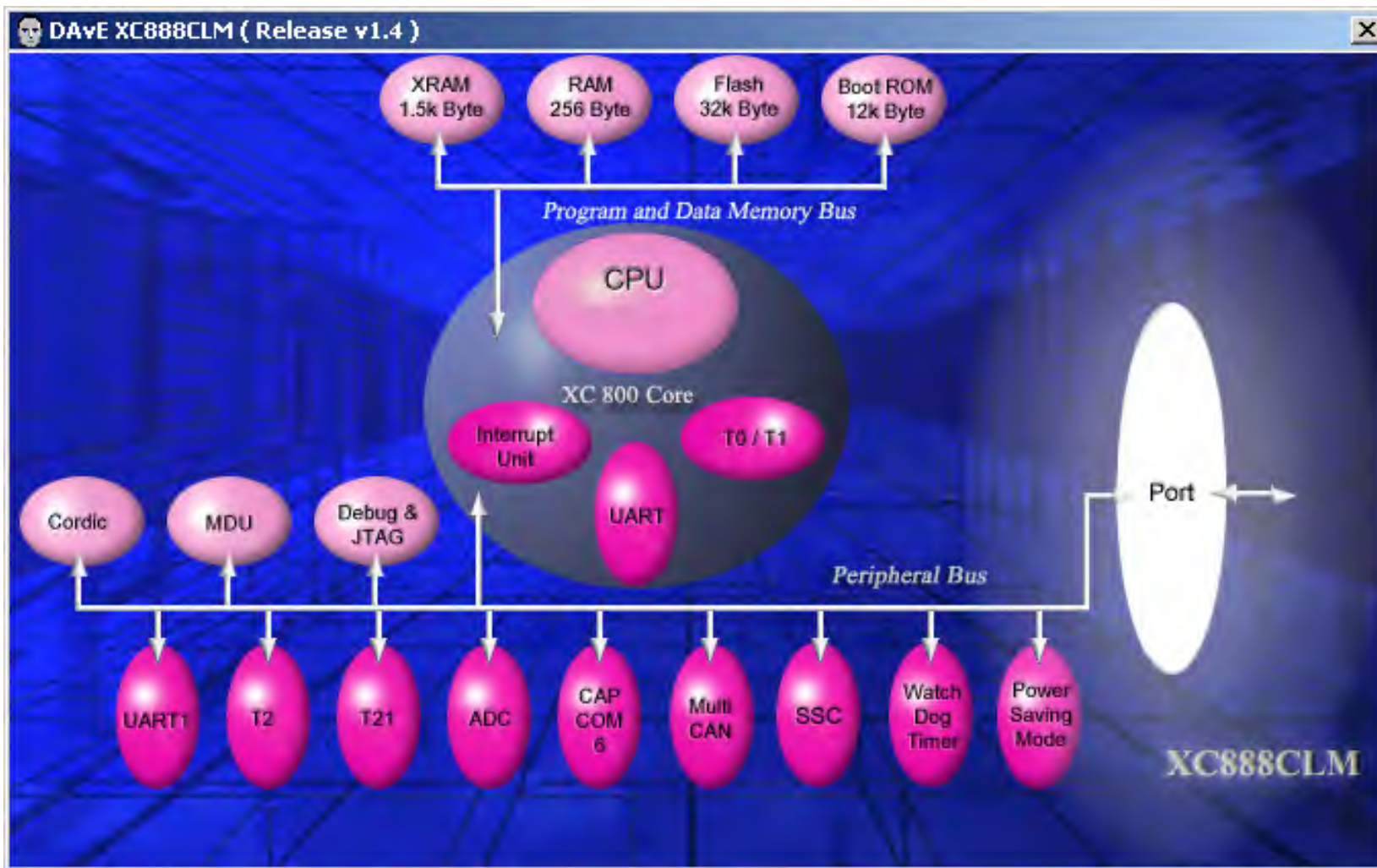
- 进入“**Functions**”页，进行函数的选用。



MultiCAN的运用

实战练习：报文的发送（续）

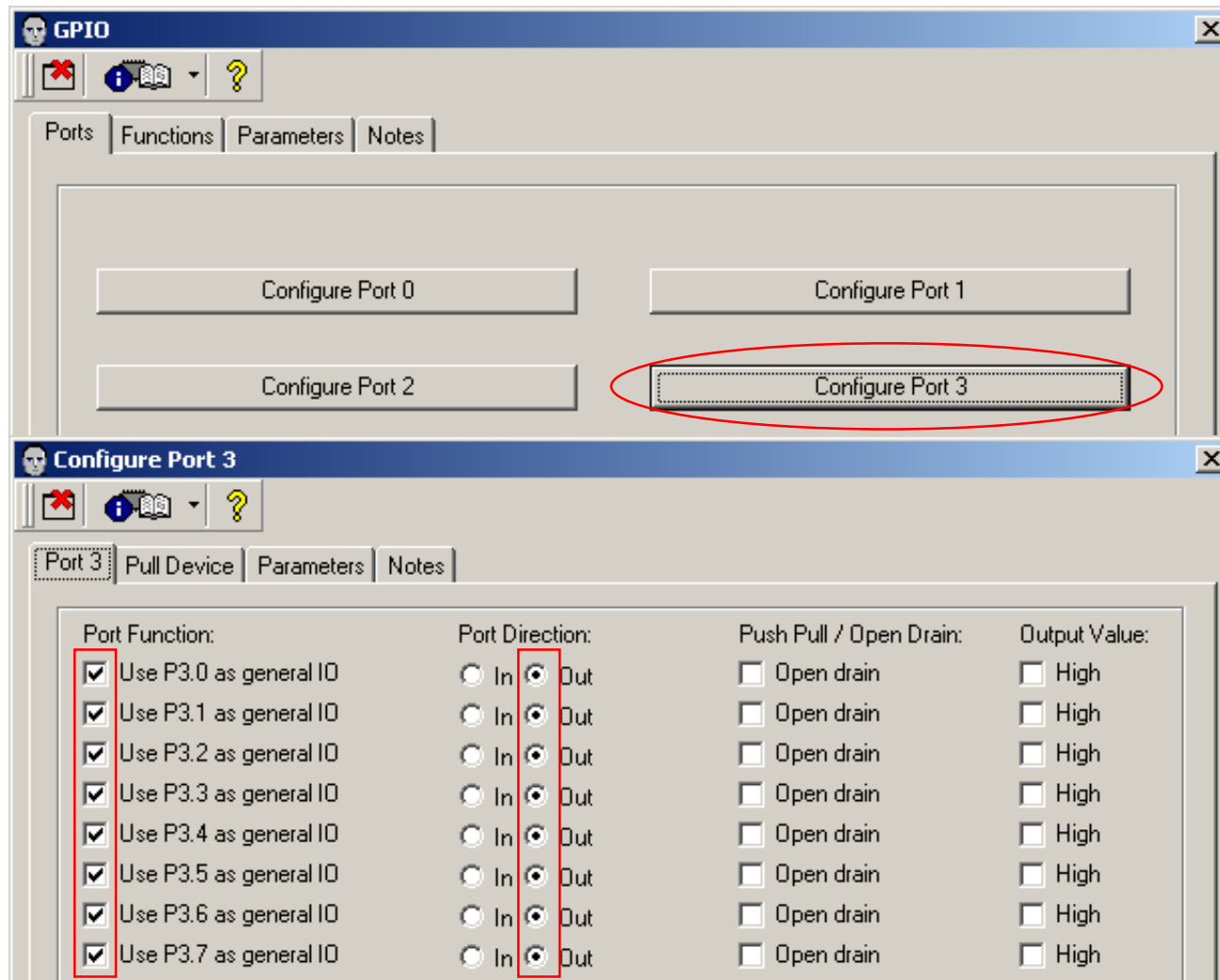
- 进入“Port”，进行端口输出设定。



MultiCAN的运用

实战练习：报文的发送（续）

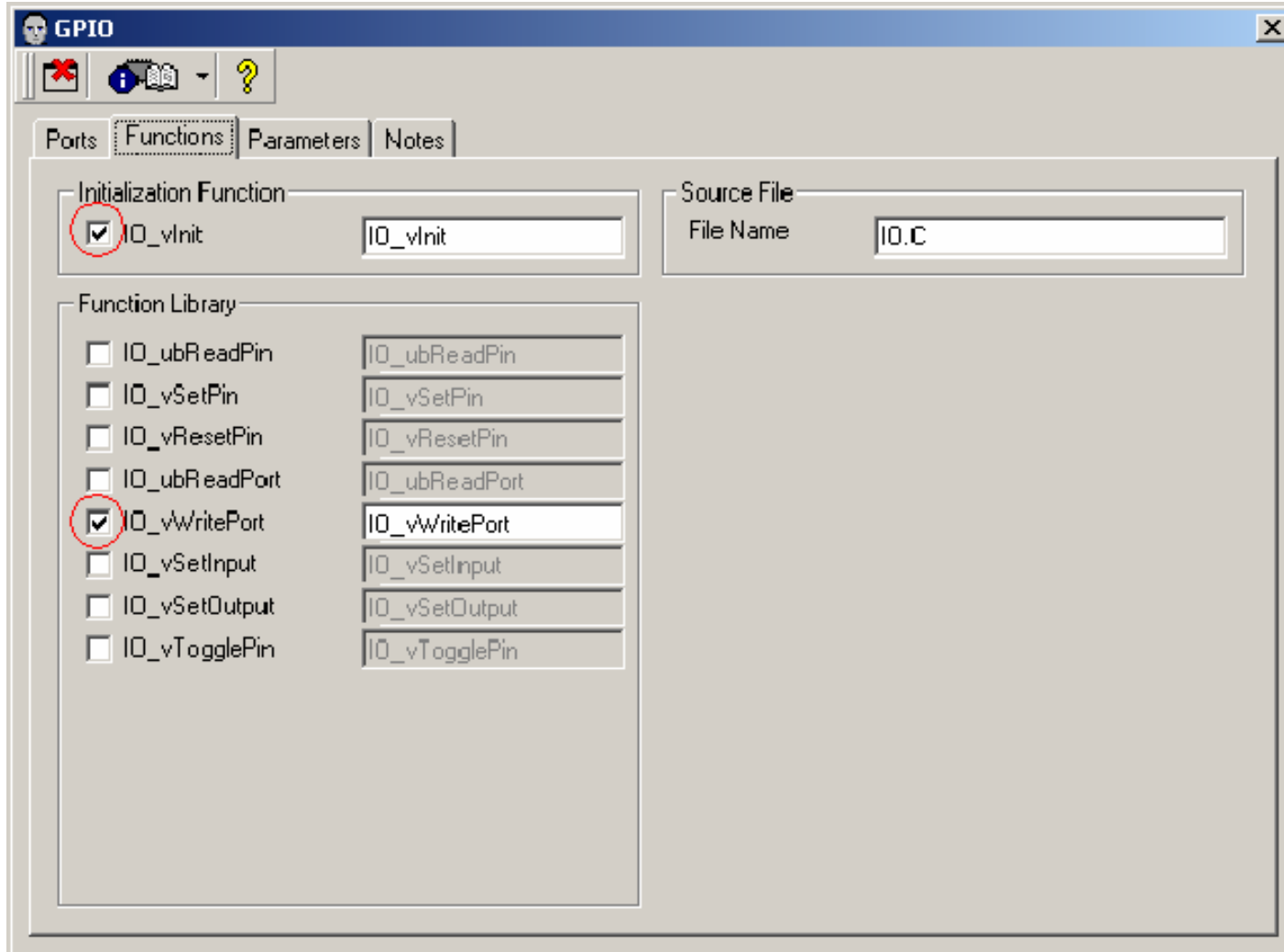
- 在“Port 3”页，设置P3.0~P3.7为输出。



MultiCAN的运用

实战练习：报文的发送（续）

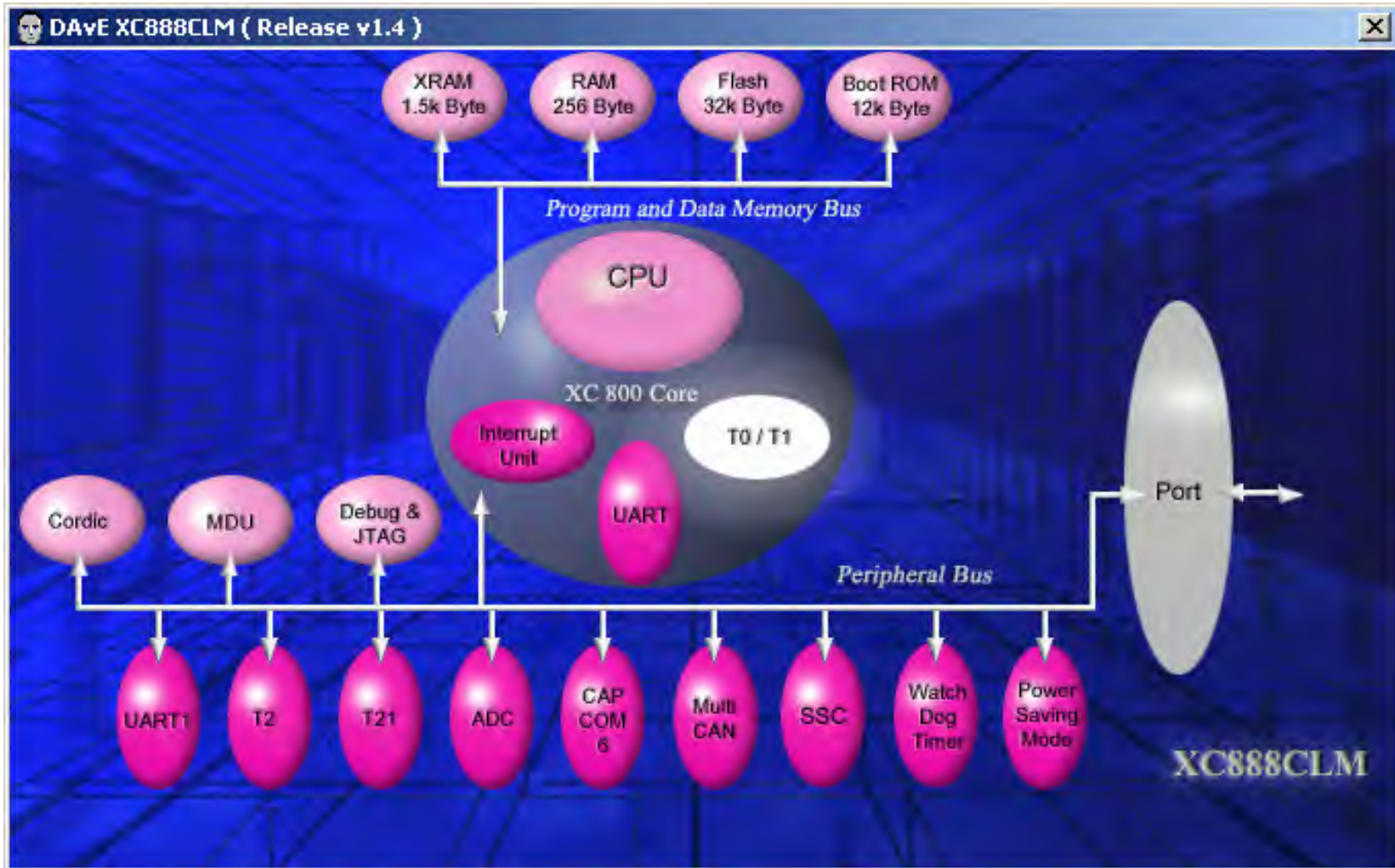
- 在“Functions”页中选用可用的函数。



MultiCAN的运用

实战练习：报文的发送（续）

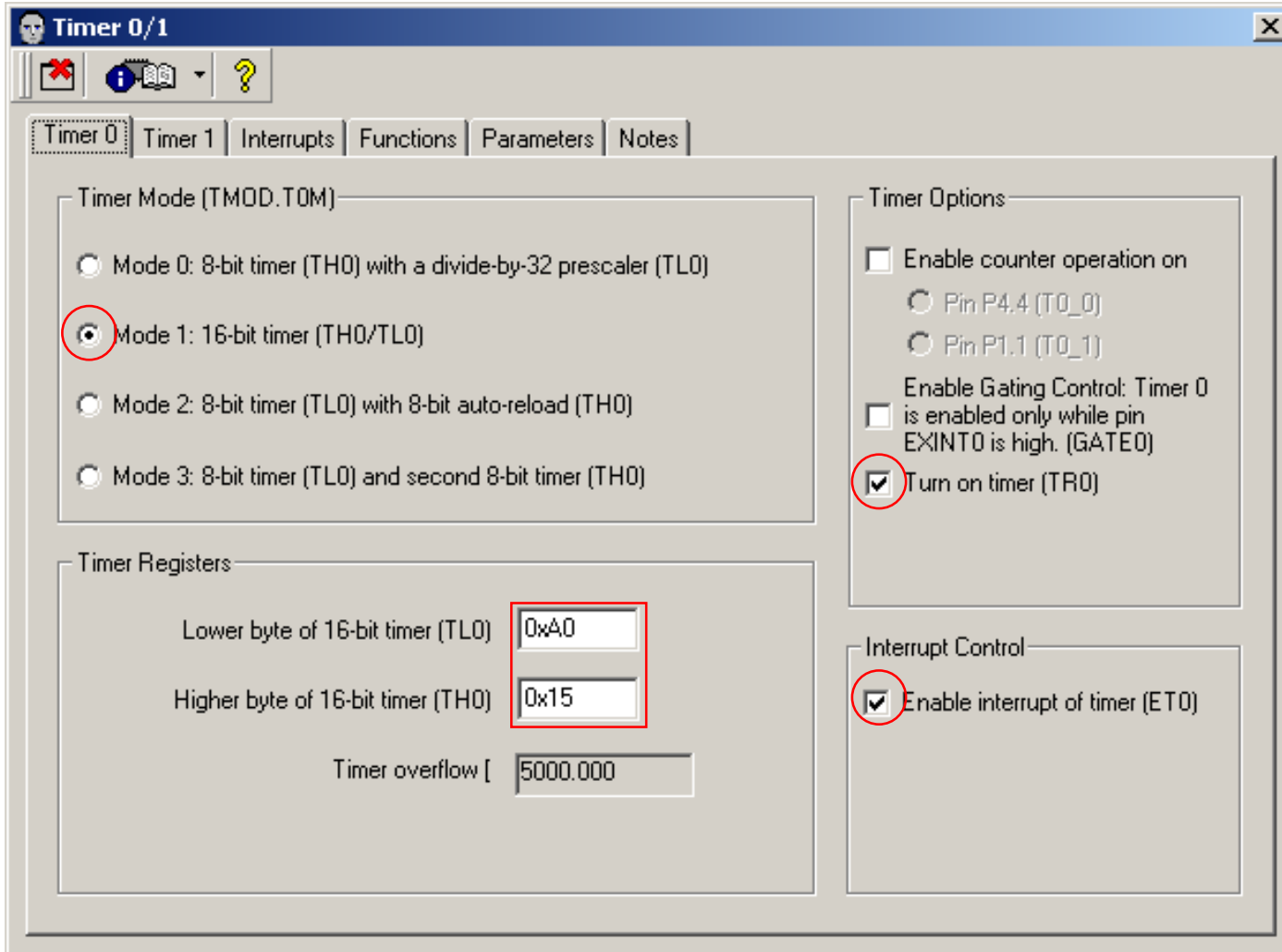
- 进入“T0 / T1”，进行定时器设定。



MultiCAN的运用

实战练习：报文的发送（续）

■ 设置“Timer 0”。



Timer 0/1

Timer 0 Timer 1 Interrupts Functions Parameters Notes

Timer Mode (TMOD.T0M)

- ☐ Mode 0: 8-bit timer (TH0) with a divide-by-32 prescaler (TL0)
- ☒ Mode 1: 16-bit timer (TH0/TL0)
- ☐ Mode 2: 8-bit timer (TL0) with 8-bit auto-reload (TH0)
- ☐ Mode 3: 8-bit timer (TL0) and second 8-bit timer (TH0)

Timer Options

- ☐ Enable counter operation on
 - ☐ Pin P4.4 (T0_0)
 - ☐ Pin P1.1 (T0_1)
- ☐ Enable Gating Control: Timer 0 is enabled only while pin EXINT0 is high. (GATE0)
- ☒ Turn on timer (TR0)

Timer Registers

Lower byte of 16-bit timer (TL0)

Higher byte of 16-bit timer (TH0)

Timer overflow []

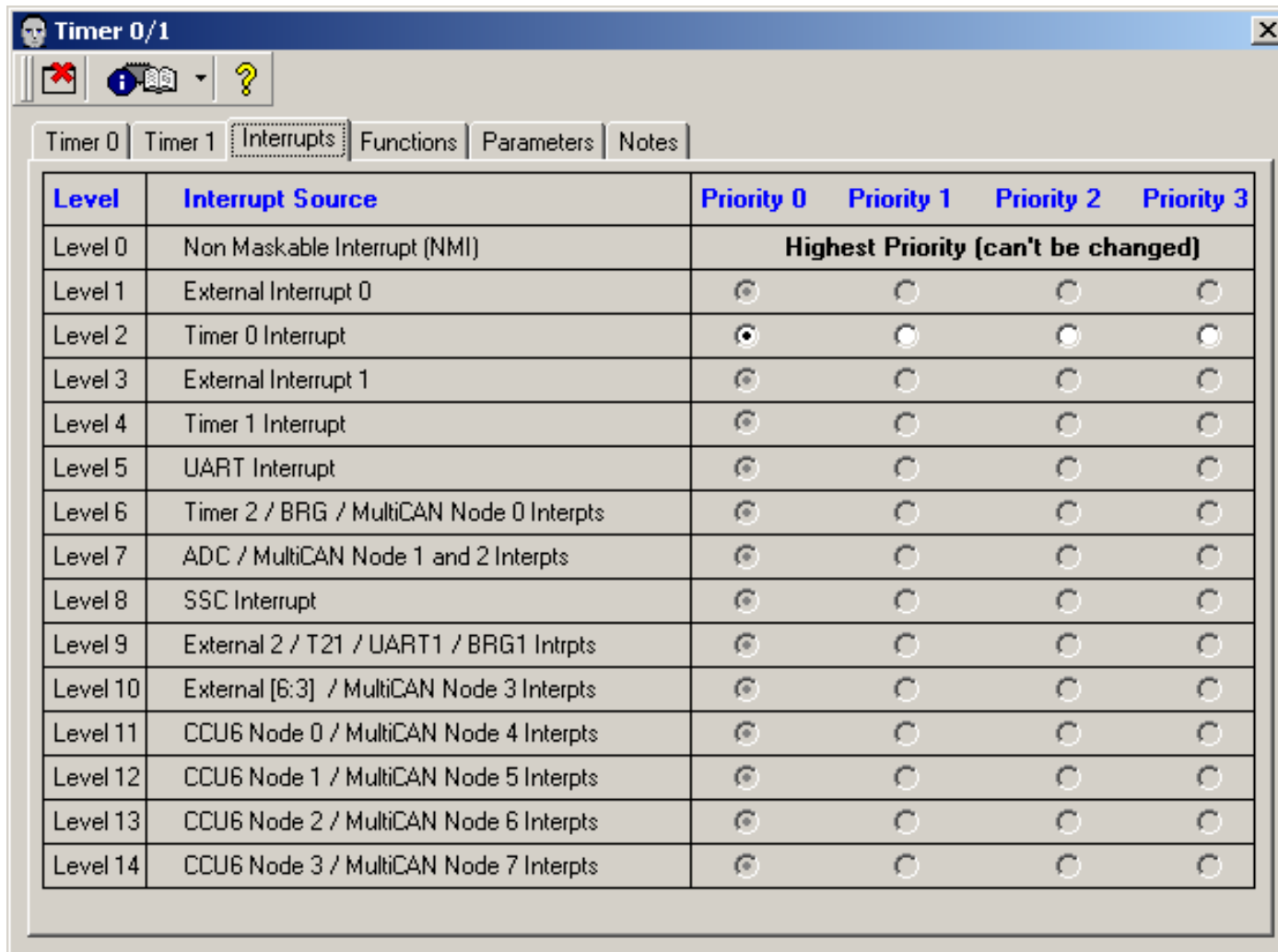
Interrupt Control

- ☒ Enable interrupt of timer (ET0)

MultiCAN的运用

实战练习：报文的发送（续）

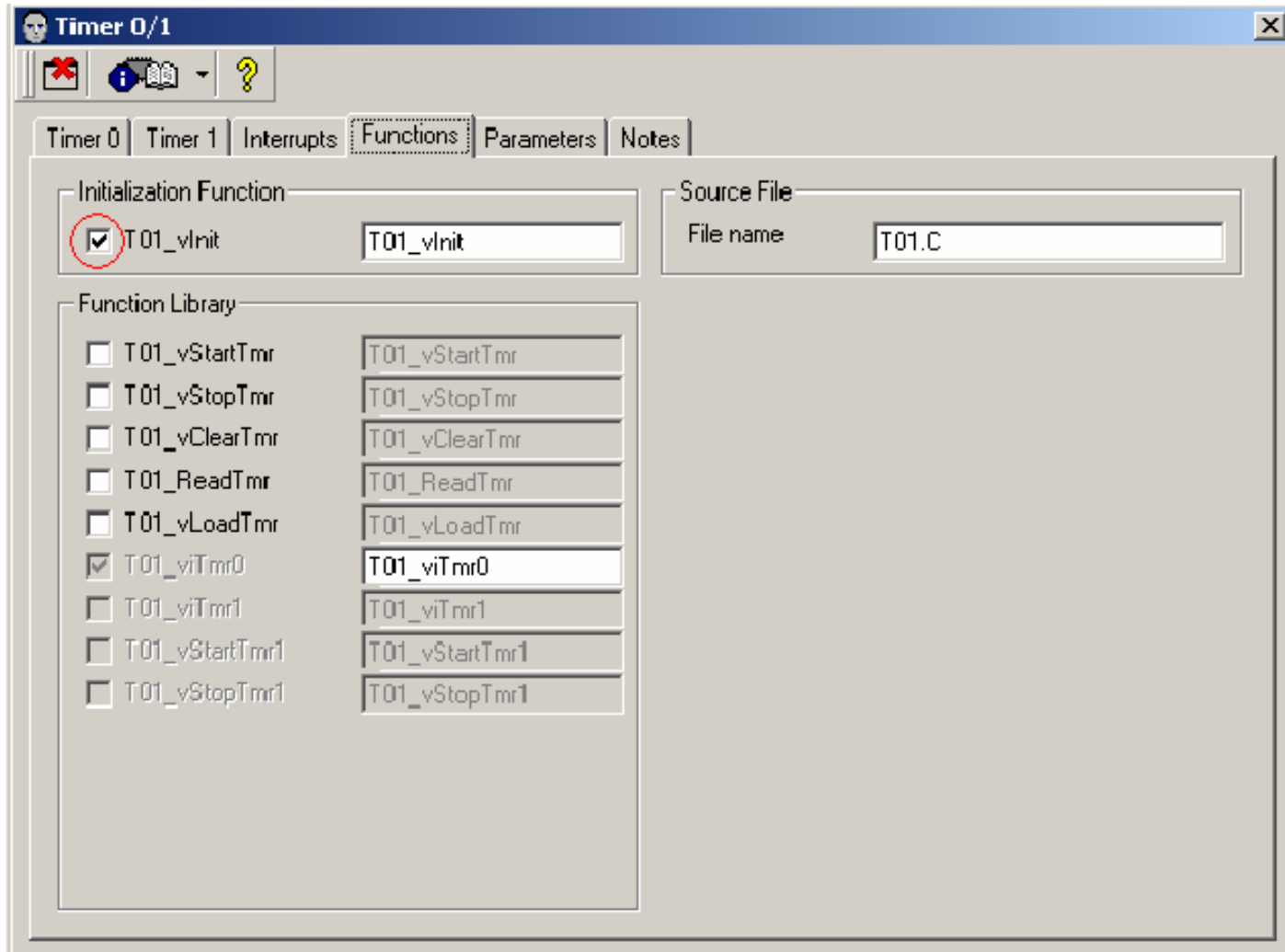
- 中断优先级使用缺省设置。



MultiCAN的运用

实战练习：报文的发送（续）

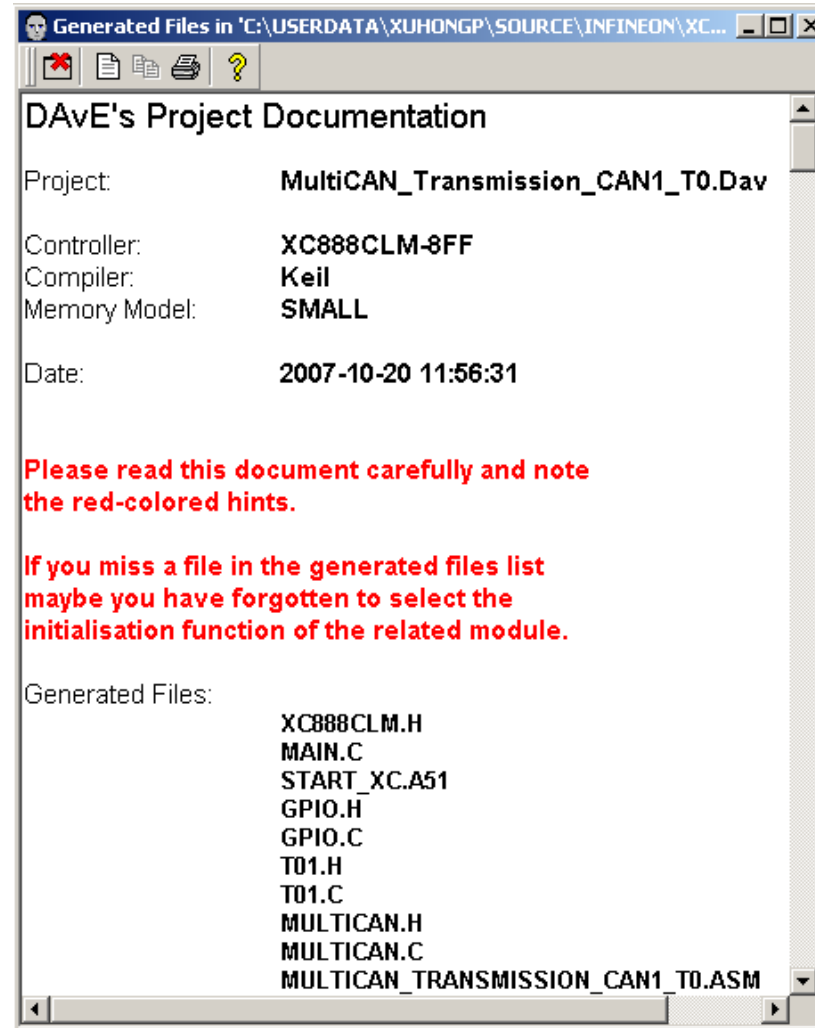
- 在“**Functions**”页进行函数选用。这里选用函数**T0_vlnit**。



MultiCAN的运用

实战练习：报文的发送（续）

- 完成了基本设置，保存并生成项目代码。



MultiCAN的运用

实战练习：报文的发送（续）

■ 在生成代码后，需要对代码进行一些修改：

□ 在Main.C文件中，“MAIN_General,7”中插入代码；

```
// USER CODE BEGIN (MAIN_General,7)
```

```
ubyte ubiCounter;
```

```
ubyte ubSecond;
```

```
// USER CODE END
```

□ 在“MAIN_Main,3”中，插入代码

```
// USER CODE BEGIN (MAIN_Main,3)
```

```
ubiCounter = 0;
```

```
ubSecond = 0;
```

```
// USER CODE END
```

MultiCAN的运用

实战练习：报文的发送（续）

- 在T01.C文件中 “T01_General,6”插入代码

```
// USER CODE BEGIN (T01_General,6)
extern ubyte ubiCounter;
extern ubyte ubSecond;
// USER CODE END
```

- 在“T01_IsrTmr0,2”中，插入代码

```
// USER CODE BEGIN (T01_IsrTmr0,2)
TL0 = 0xA0;
TH0 = 0x15;
ubiCounter ++;
if(ubiCounter == 200)
{
    ubiCounter = 0;
    ubSecond ++;
    IO_vWritePort(P3, ubSecond);
    CAN_vWriteCANAddress(CAN_MODATAL0);
    CAN_DATA0 = ubSecond;
    CAN_vWriteEN(D0_VALID);
    CAN_vTransmit(0);}
// USER CODE END
```

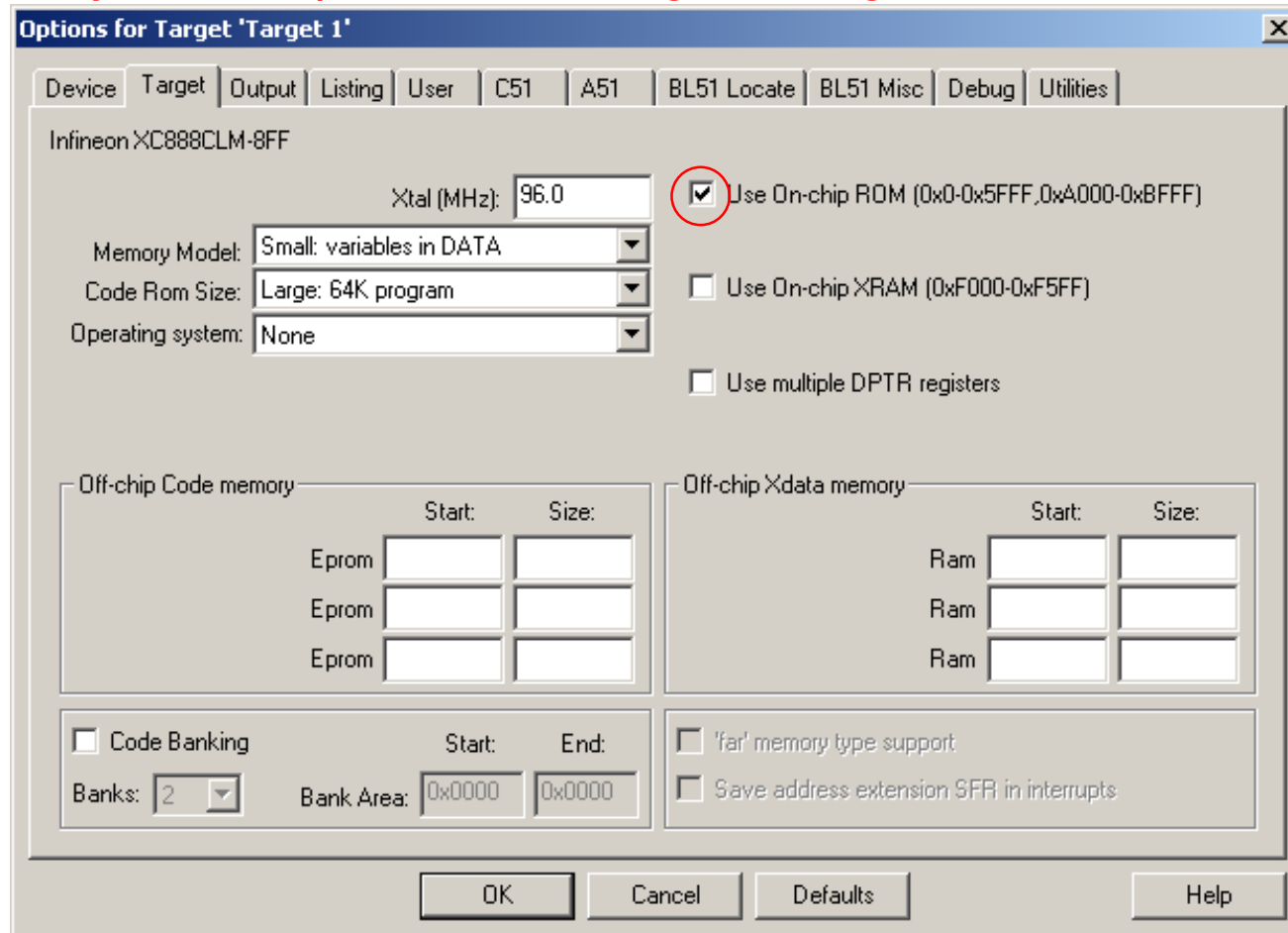
由于Timer 0的中断时间只有5ms，这里需要中断200次，得到1s

MultiCAN的运用

实战练习：报文的发送（续）

■ 使用Keil打开DAvE建立的工程：

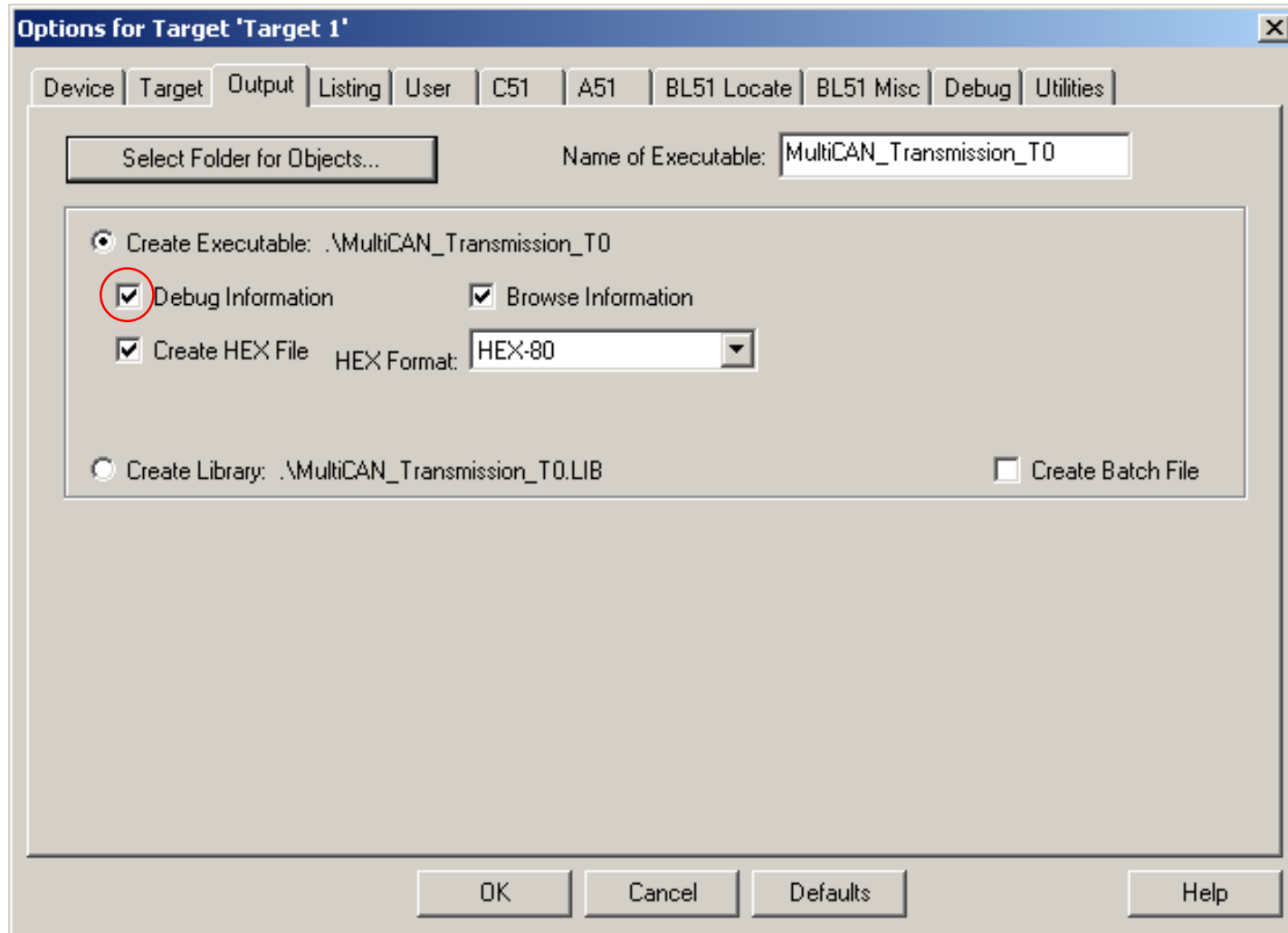
□ 选择“**Project** → **Options for Target 'Target 1'**”，修改项目设定。



MultiCAN的运用

实战练习：报文的发送（续）

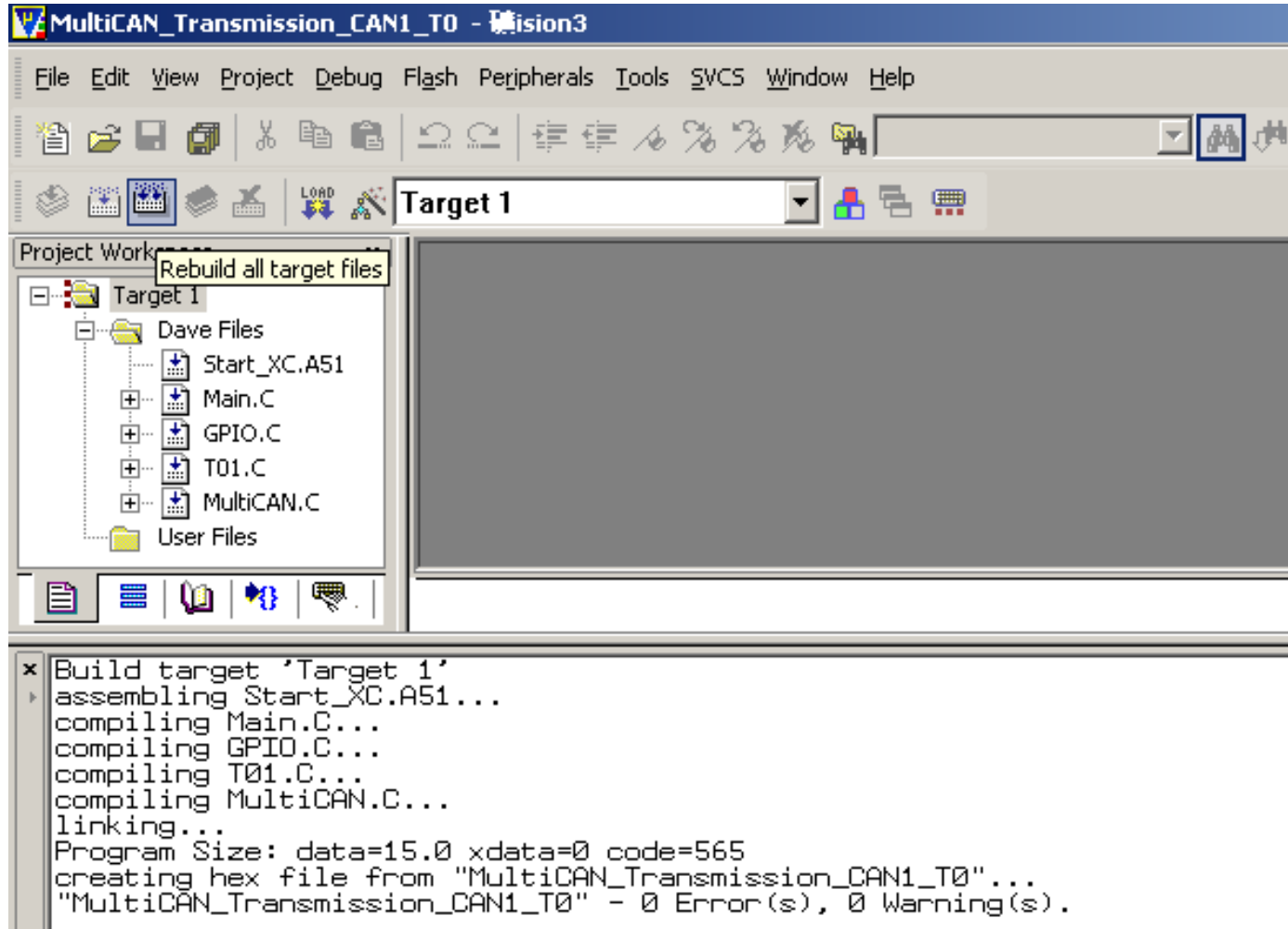
- 在“Output”页，选择“Create HEX File”。



MultiCAN的运用

实战练习：报文的发送（续）

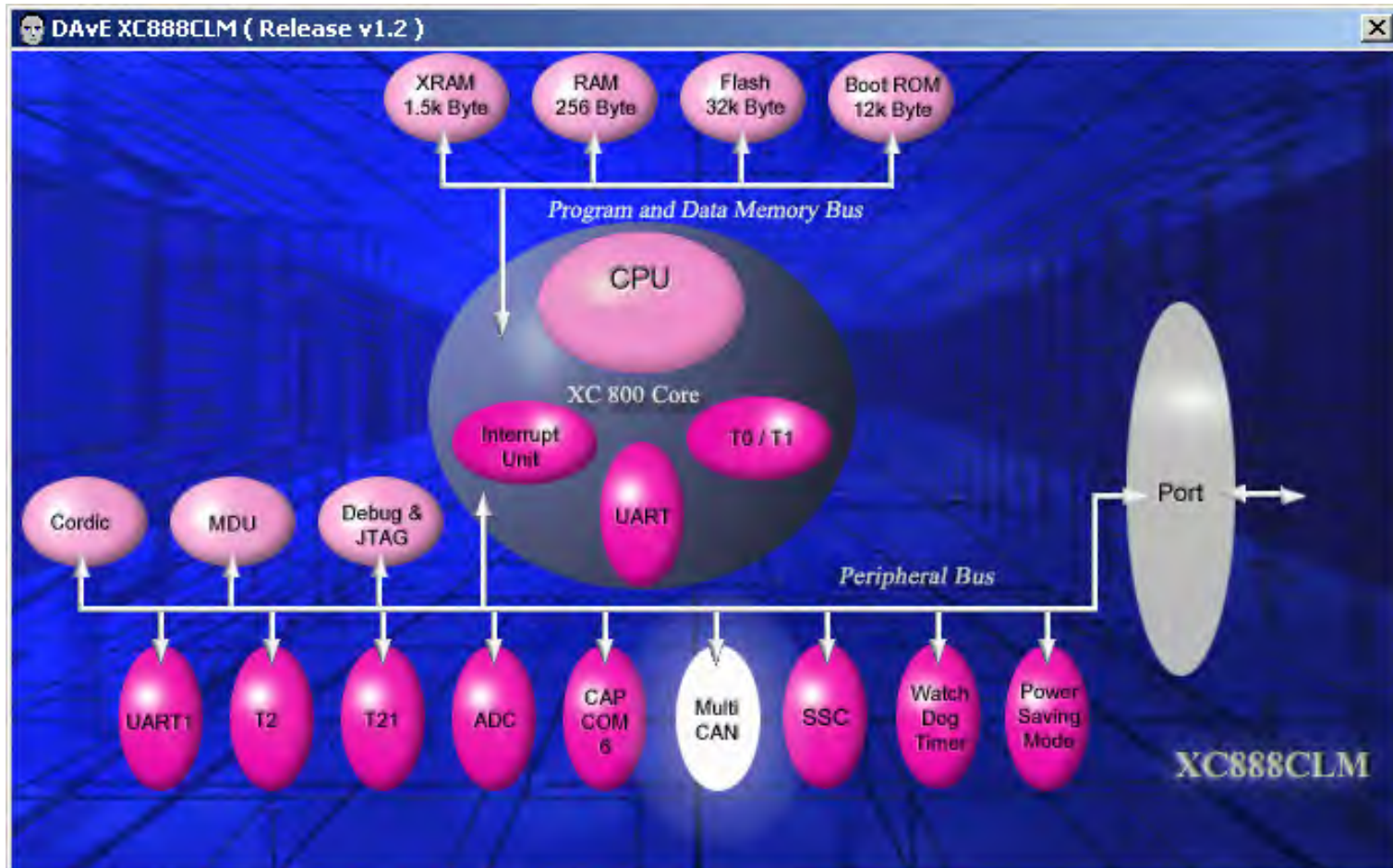
- 修改设置后，编译通过，生成HEX文件，可以下载到目标芯片了。



MultiCAN的运用

实战练习：报文的接收（续）

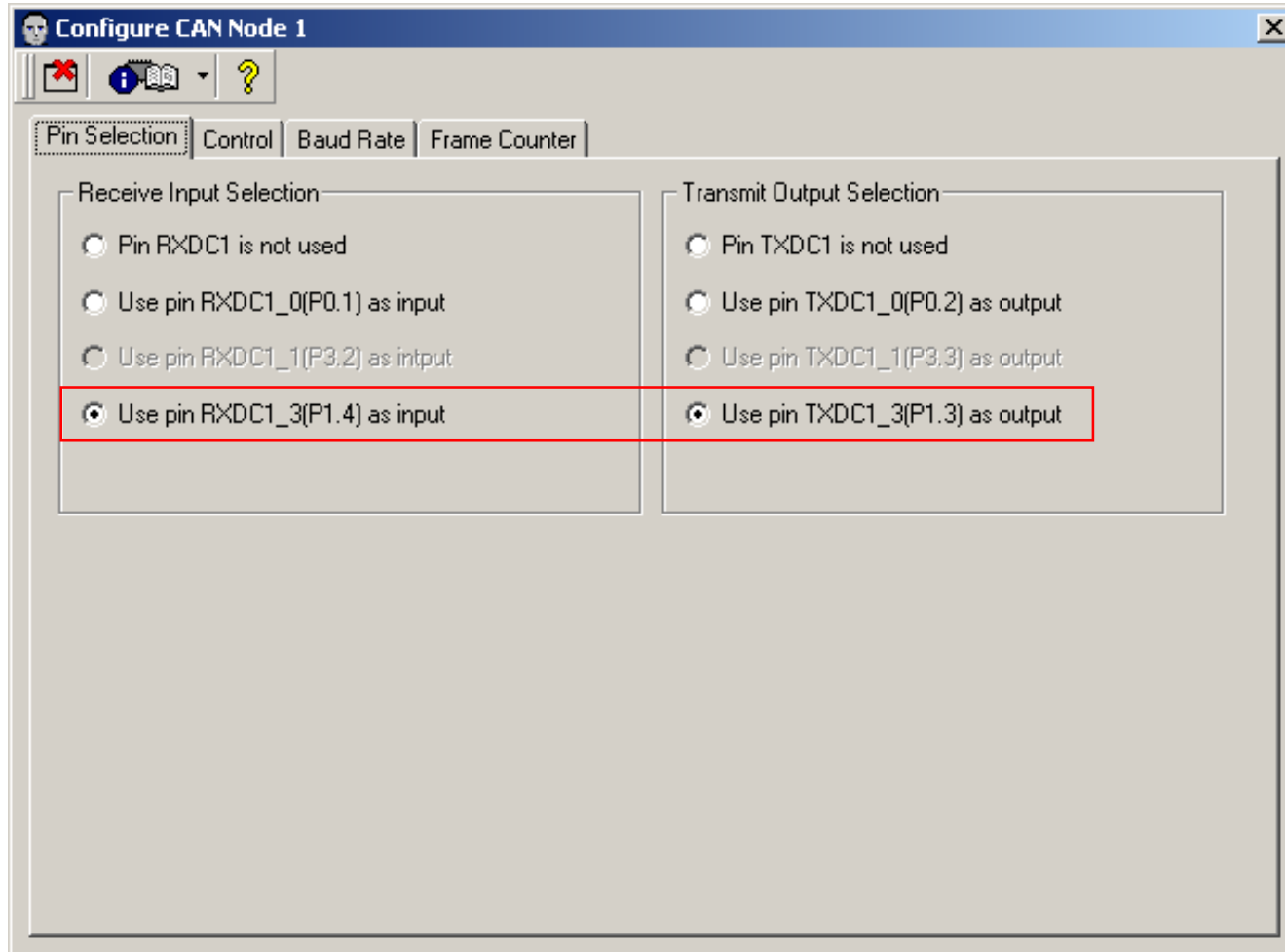
- 下面将接收MultiCAN传输来的LED的信息，然后在端口3上显示输出。
- 使用DAvE建立一个新工程，点击“MultiCAN”修改设定。



MultiCAN的运用

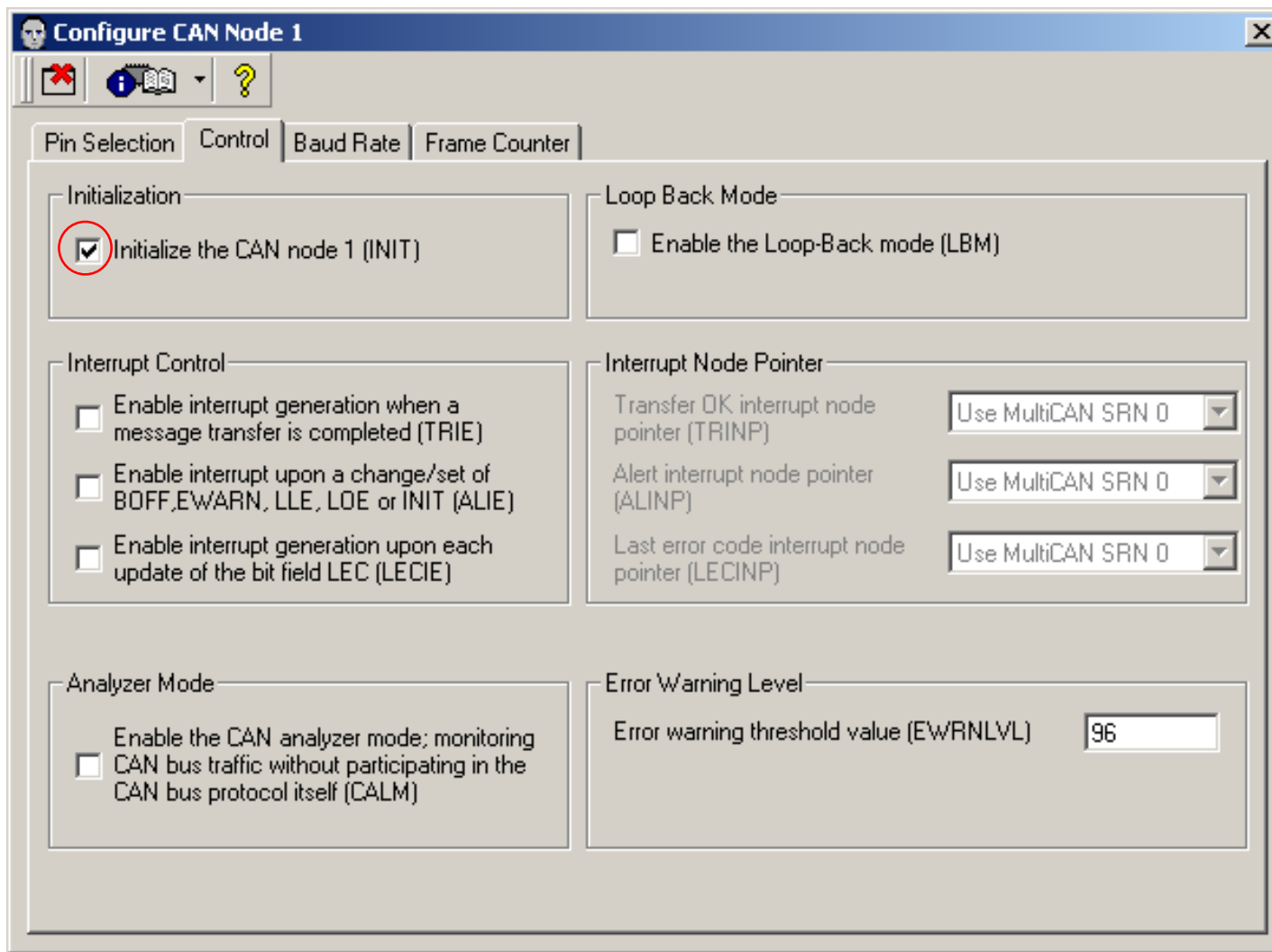
实战练习：报文的接收（续）

- 在从机程序中，同样也需要设置传输脚，传输模式，以及传输信息，可以参考“发送”部分。



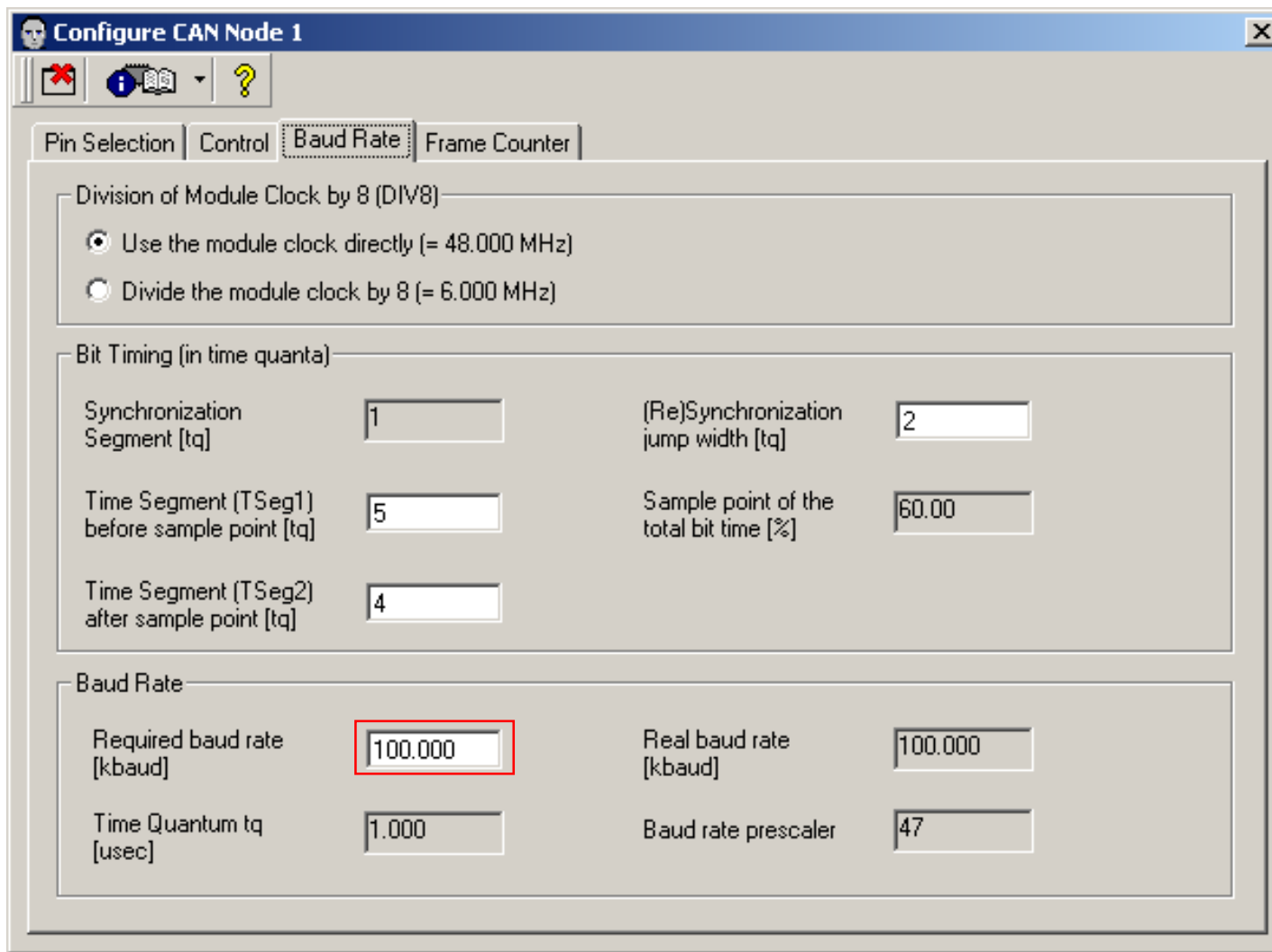
MultiCAN的运用

实战练习：报文的接收（续）



MultiCAN的运用

实战练习：报文的接收（续）



Configure CAN Node 1

Pin Selection | Control | **Baud Rate** | Frame Counter

Division of Module Clock by 8 (DIV8)

- ☒ Use the module clock directly (= 48.000 MHz)
- ☐ Divide the module clock by 8 (= 6.000 MHz)

Bit Timing (in time quanta)

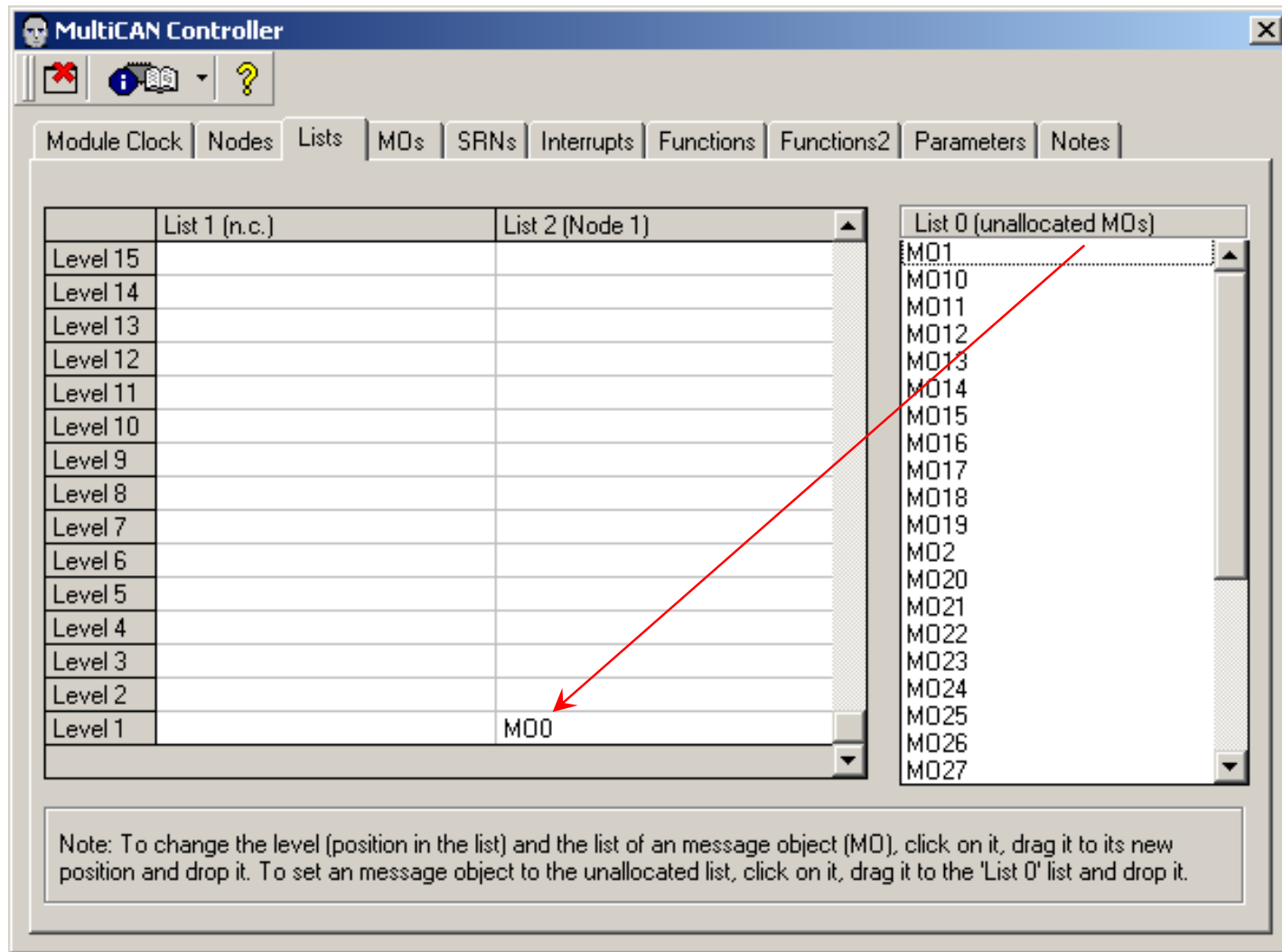
Synchronization Segment [tq]	1	(Re)Synchronization jump width [tq]	2
Time Segment (TSeg1) before sample point [tq]	5	Sample point of the total bit time [%]	60.00
Time Segment (TSeg2) after sample point [tq]	4		

Baud Rate

Required baud rate [kbaud]	100.000	Real baud rate [kbaud]	100.000
Time Quantum tq [usec]	1.000	Baud rate prescaler	47

MultiCAN的运用

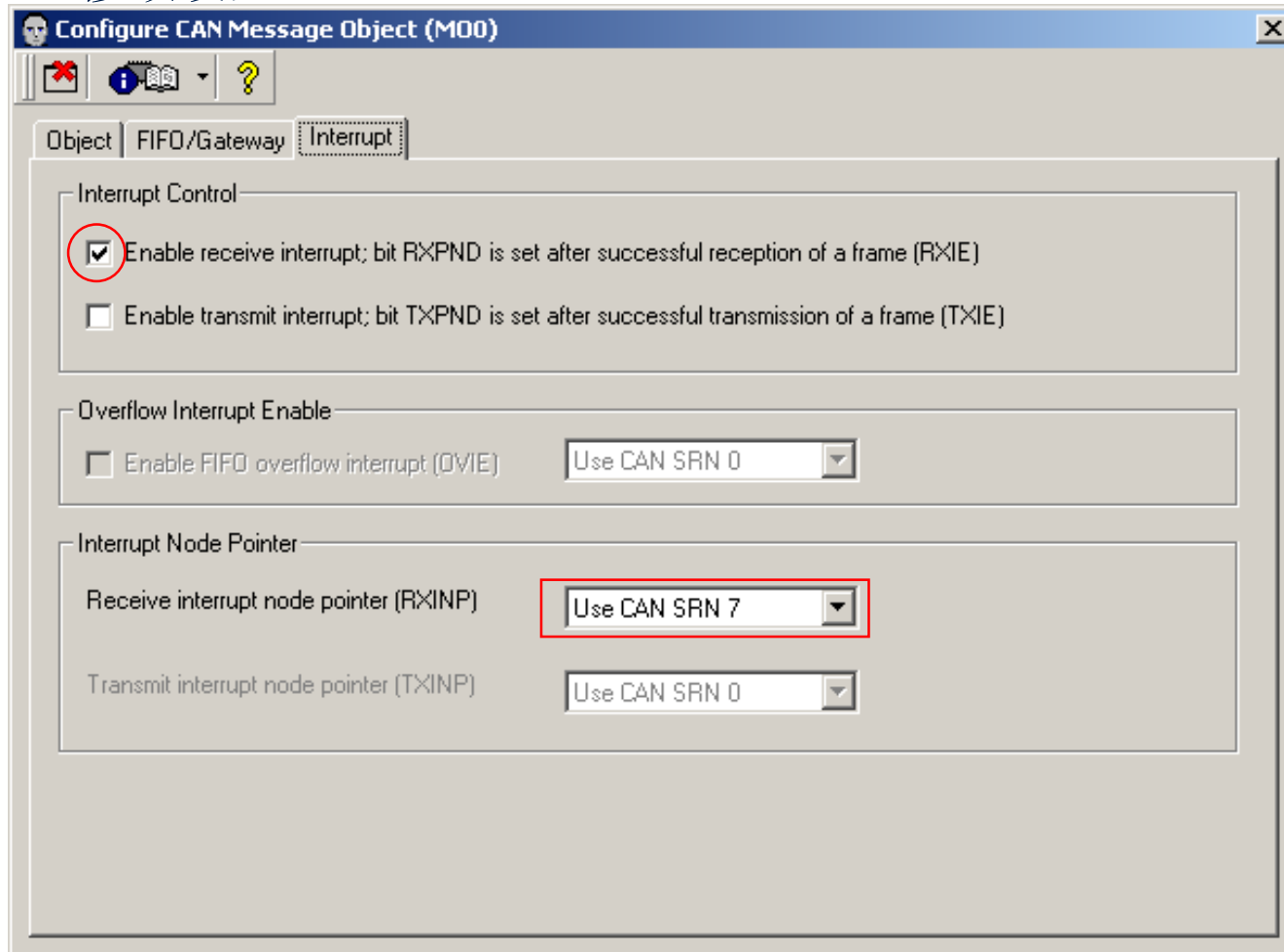
实战练习：报文的接收（续）



MultiCAN的运用

实战练习：报文的接收（续）

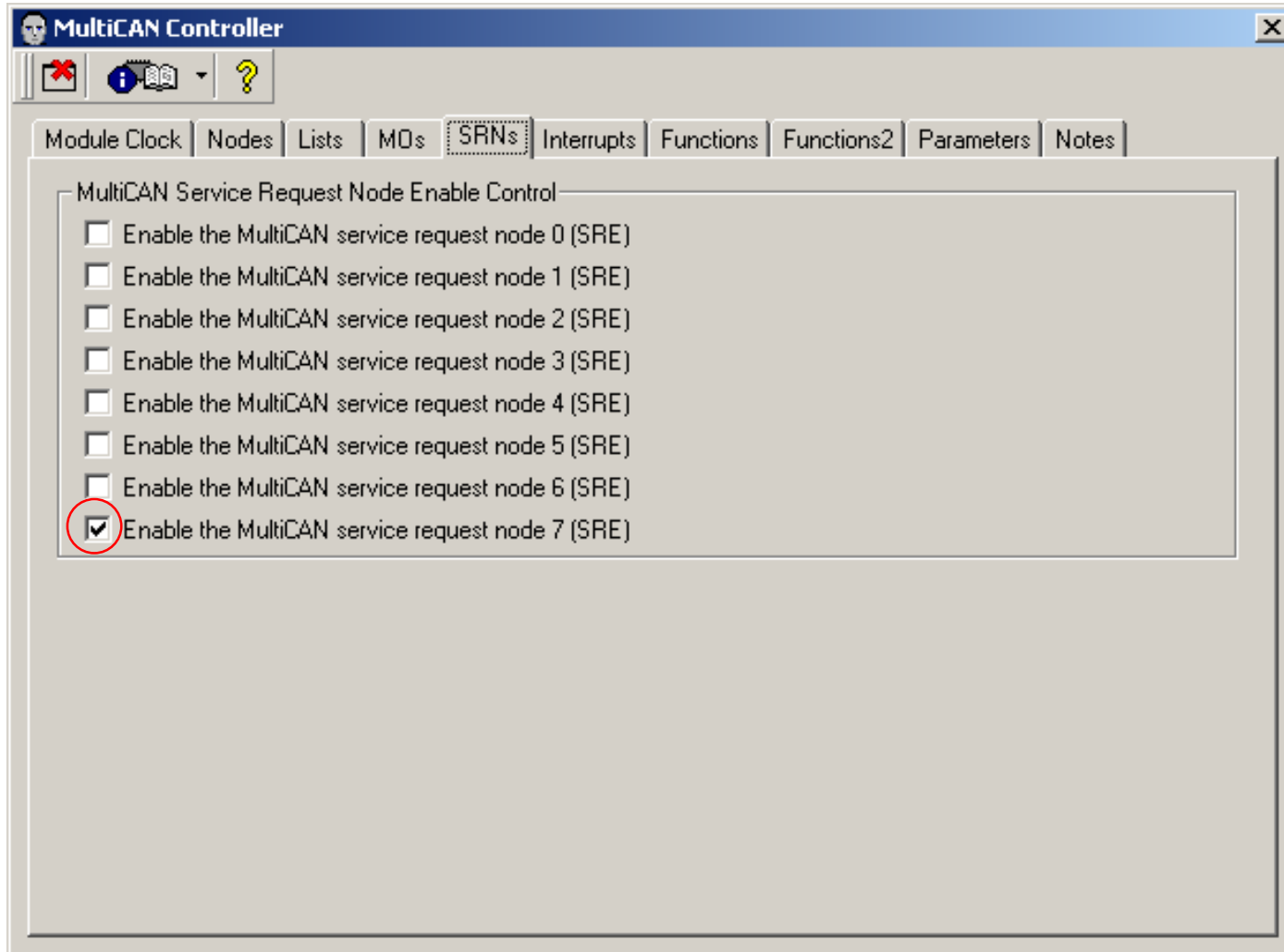
- 和“发送”部分不同的是，需要使用中断来接收MultiCAN传输的信息，因此需要做一些修改设置。



MultiCAN的运用

实战练习：报文的接收（续）

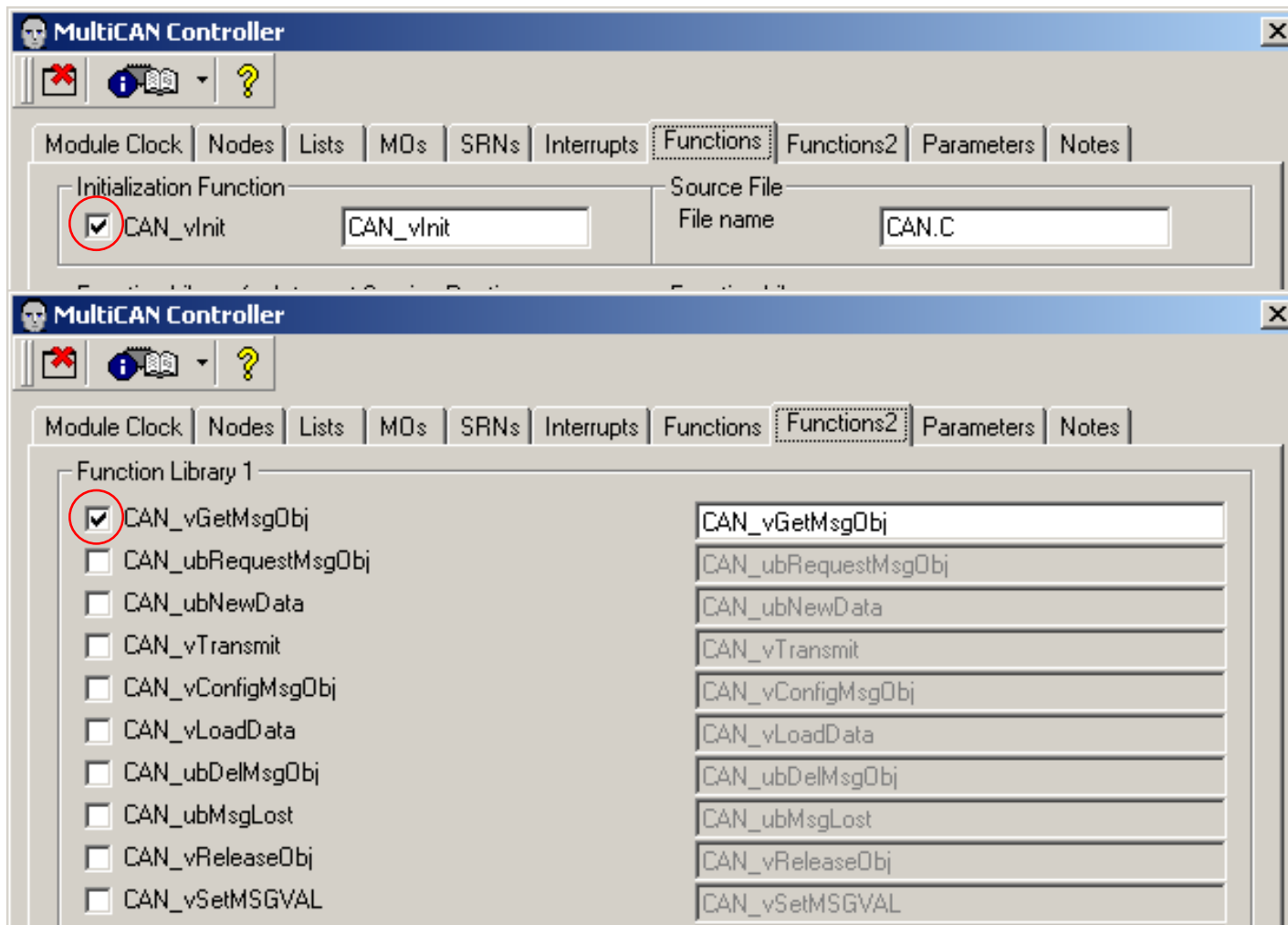
- 选用中断后，还要指定中断节点。



MultiCAN的运用

实战练习：报文的接收（续）

- 在“Functions”页，选用需要使用的函数。



MultiCAN的运用

实战练习：报文的接收（续）

- 设置完成后，生成工程代码，然后进行一些修改：

- 在文件“Shared_INT.C”中的“NodeI3,2”处，添加代码

```
// USER CODE BEGIN (NodeI3,2)
```

```
stCAN_SWObj SW_MOs;
```

```
// USER CODE BEGIN (NodeI3,2)
```

- 在“SRN7_OBJ,1”处，添加代码：

```
// USER CODE BEGIN (SRN7_OBJ,1)
```

```
if(ubTempVarObjHandler & MOSTAT_RXPND)
```

```
    // if RXPND is set
```

```
{ if(ubTempVarObjHandler & MOSTAT_NEWDAT)
```

```
    // if NEWDAT is set
```

```
{ if(ubTempVarObjHandler & MOSTAT_MSGLST)
```

```
    // if MSGLST is set
```

```
    { // Indicates that the CAN controller has stored a new  
      // message into this object, while NEWDAT was still set,  
      // ie. the previously stored message is lost. }
```

```
    else
```

MultiCAN的运用

实战练习：报文的接收（续）

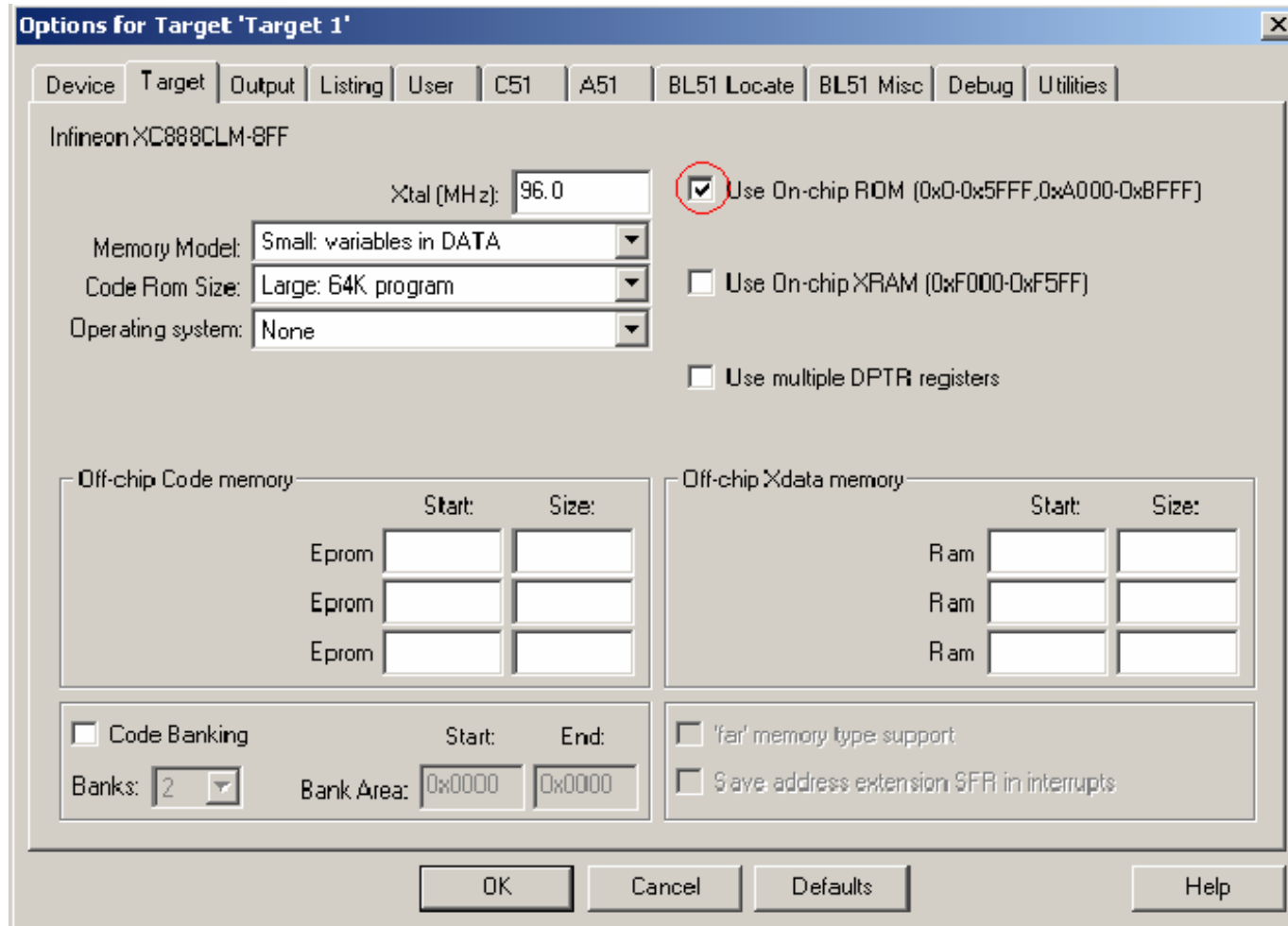


```
{// The CAN controller has stored a new message into this object.  
    CAN_vGetMsgObj(ubTempMsgID, &SW_MOs);  
    IO_vWritePort(P3, SW_MOs.ulDATA1.ubDB[3]);  
        // Display First Byte Of The Message  
    }  
  
        // Reset RXPND, NEWDAT, MSGLST (if set)  
    CAN_vWriteCANAddress(CAN_MOCTR(ubTempMsgID));  
    CAN_DATA0=(ubTempVarObjHandler&MOSTAT_RST_MNR);  
    CAN_DATA1 = 0x00;  
    CAN_DATA2 = 0x00;  
    CAN_DATA3 = 0x00;  
    CAN_vWriteEN(ALL_DATA_VALID); // Writemode is Enabled  
}  
  
// USER CODE END
```

MultiCAN的运用

实战练习：报文的接收（续）

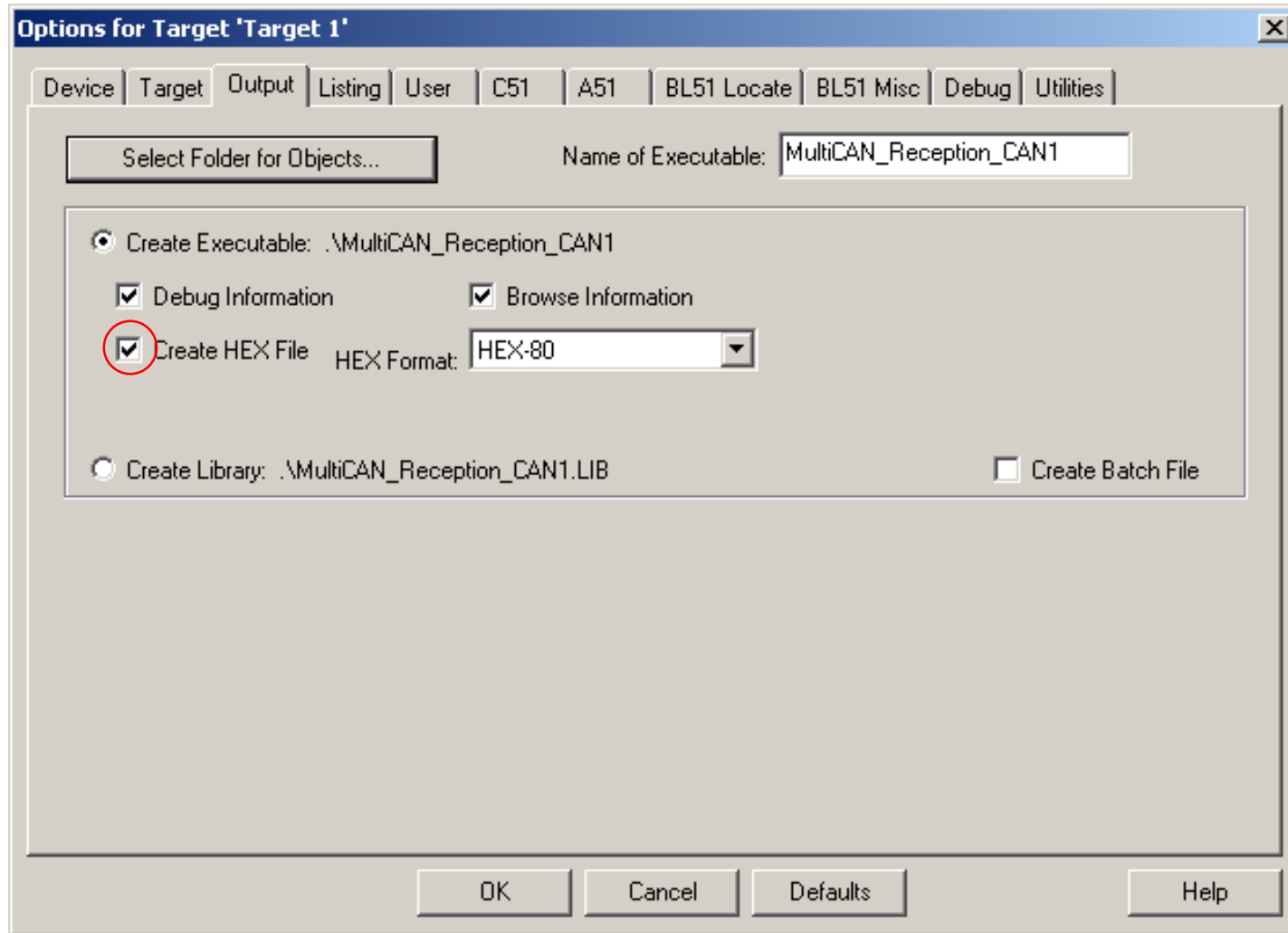
- 使用Keil打开工程，修改设置“Project → Options for Target 'Target 1'”。



MultiCAN的运用

实战练习：报文的接收（续）

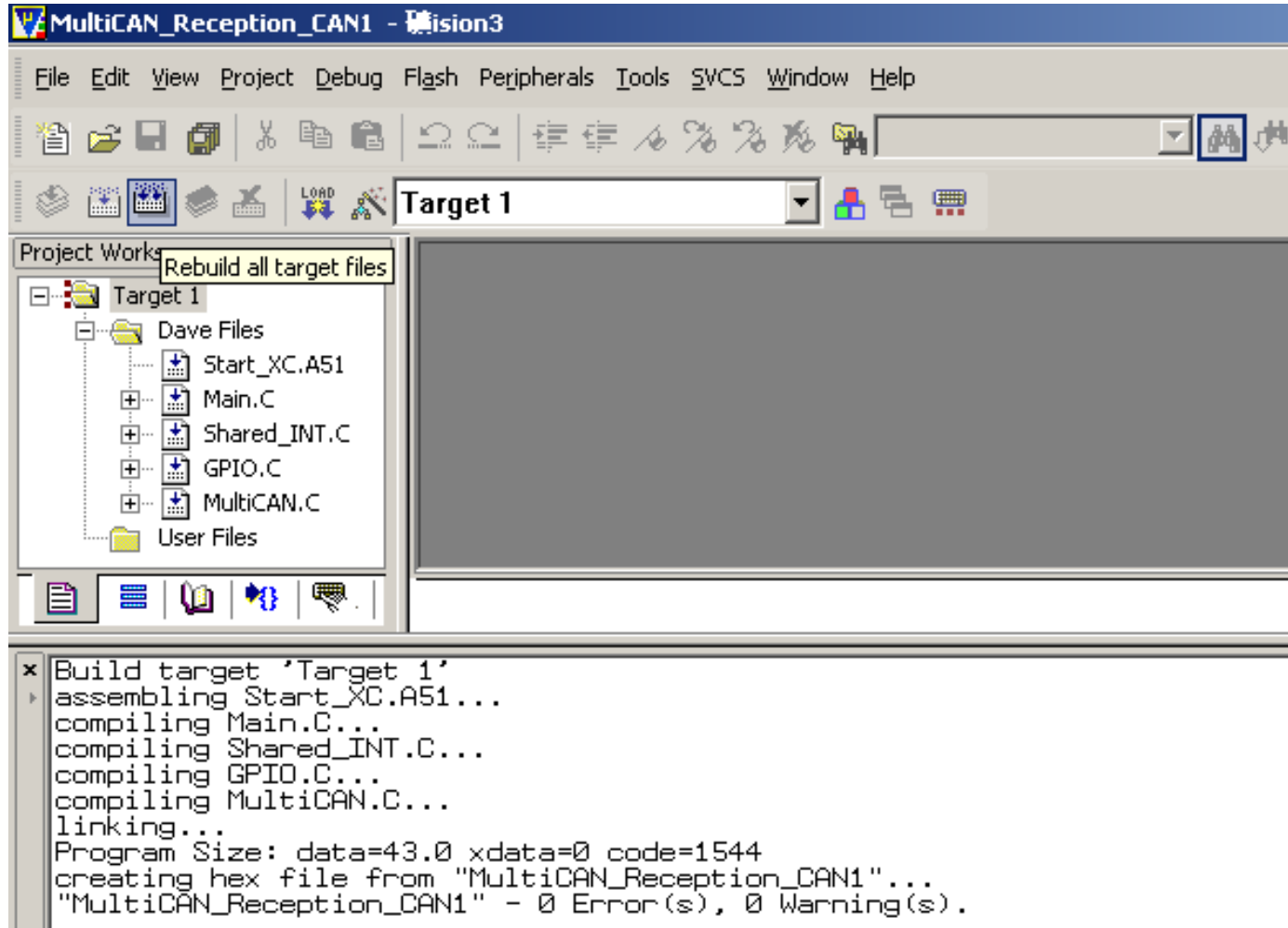
- 在“Output”页，选择“Create HEX File”。



MultiCAN的运用

实战练习：报文的接收（续）

- 编译并生成HEX文件。



MultiCAN的运用

实战练习：报文的接收/接收



- 发送和接收的程序都正确生成**HEX**文件后，可以使用下载工具烧写到目标芯片中，已验证程序的正确与否。
- 利用**FLOAD**或者**MEMTOOL**下载程序到主机和从机，然后运行主机和从机。可以看到，从机的**LED**将会显示从主机传输来的秒的信息。



We commit.
We innovate.
We partner.
We create value.



Never stop thinking