

样例程序：TwinCAN 模块使用

1. 简介

本程序实现功能如下：

利用 TwinCAN 模块的两个 CAN 节点 A、B，配置内部自带的 32 消息对象，可修改各个对象的属性，如传输方向（发送、接收），ID，MASK，隶属节点名及传输数据内容。

由于 DAVE 生成的 CAN 相关代码会超过 KEIL uVision 软件评估版本的限制大小、因此需安装完整版本的 KEIL uVision。

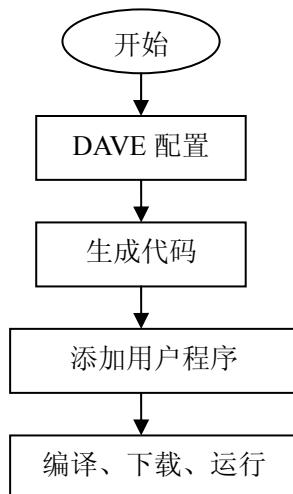
2. Twin-CAN 介绍

TwinCAN 模块内部包括两个 CAN 节点，可以独立工作也可以通过网关功能交换数据。支持 CAN 2.0 B。CAN 节点可以支持 11 位或 29 位 ID 的数据接收和发送。

网关功能支持两个 CAN 系统之间的数据自动交换。可以减轻 CPU 负担并提高整个系统的实时性能。

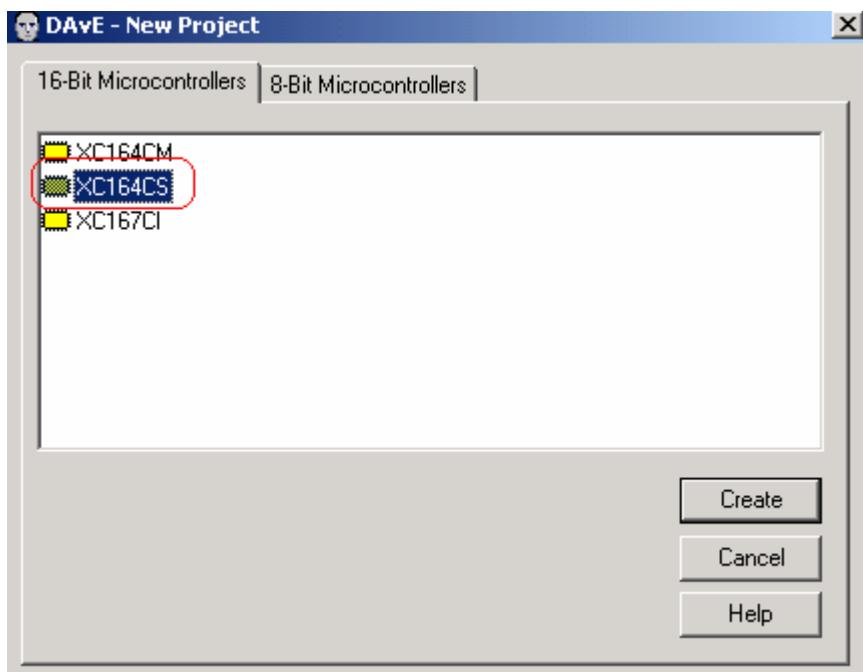
关于 TwinCAN 模块功能的详细介绍，请参照 XC164CS 用户手册。

3. 操作流程：

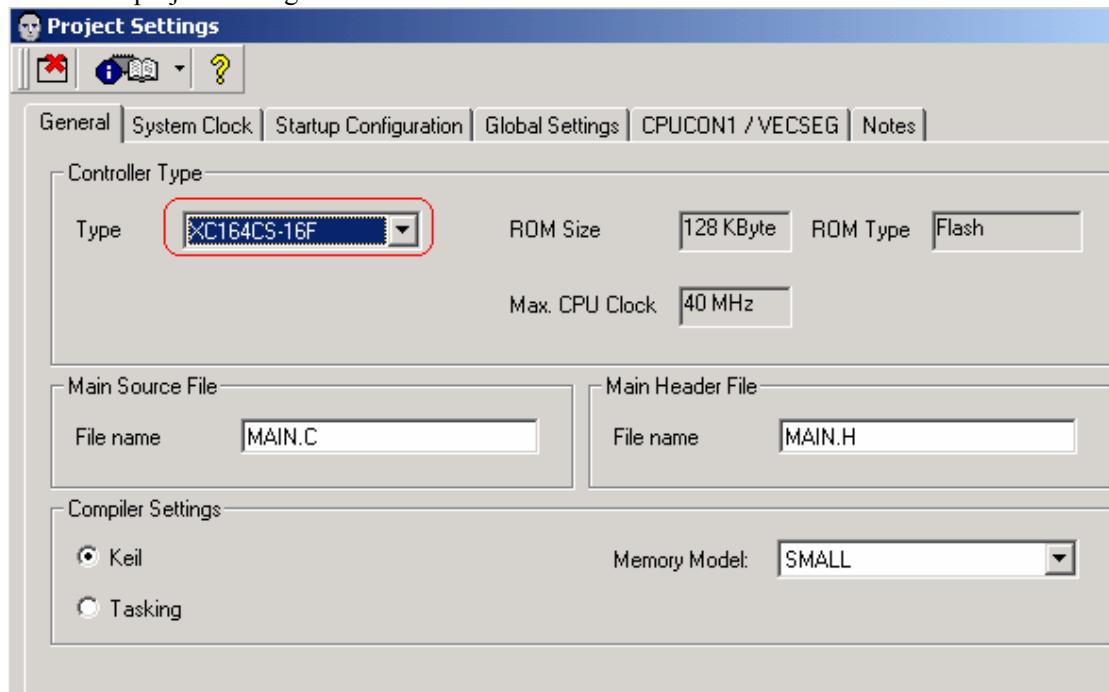


4. DAvE 配置

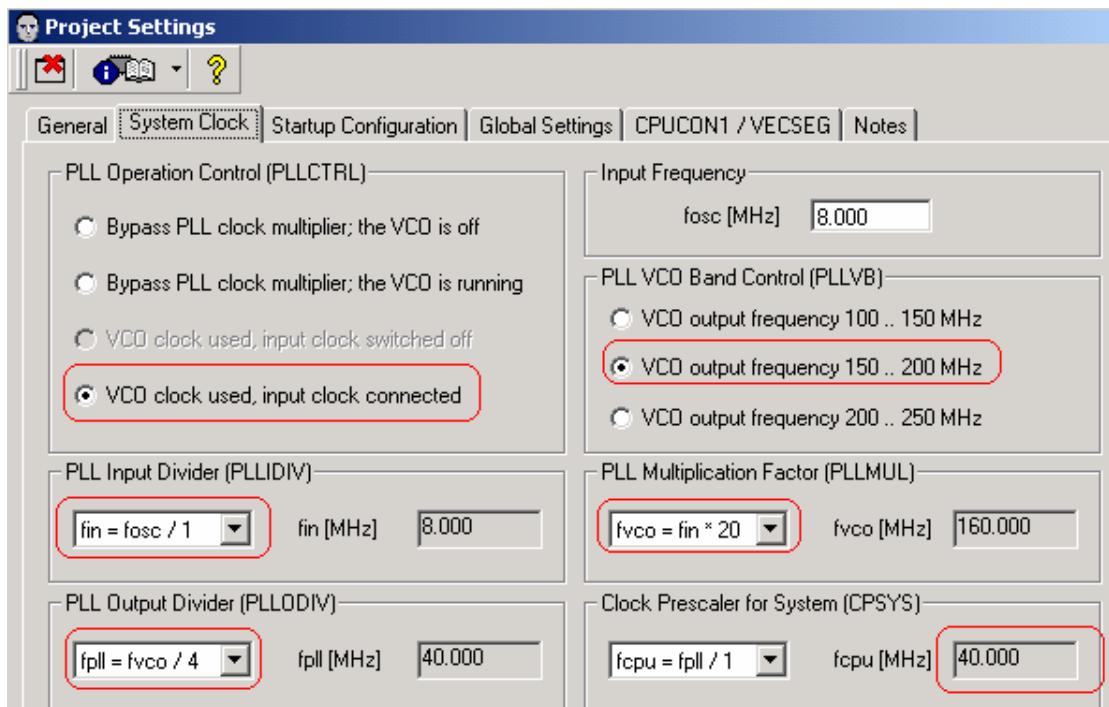
4. 1 New project: select XC164CS



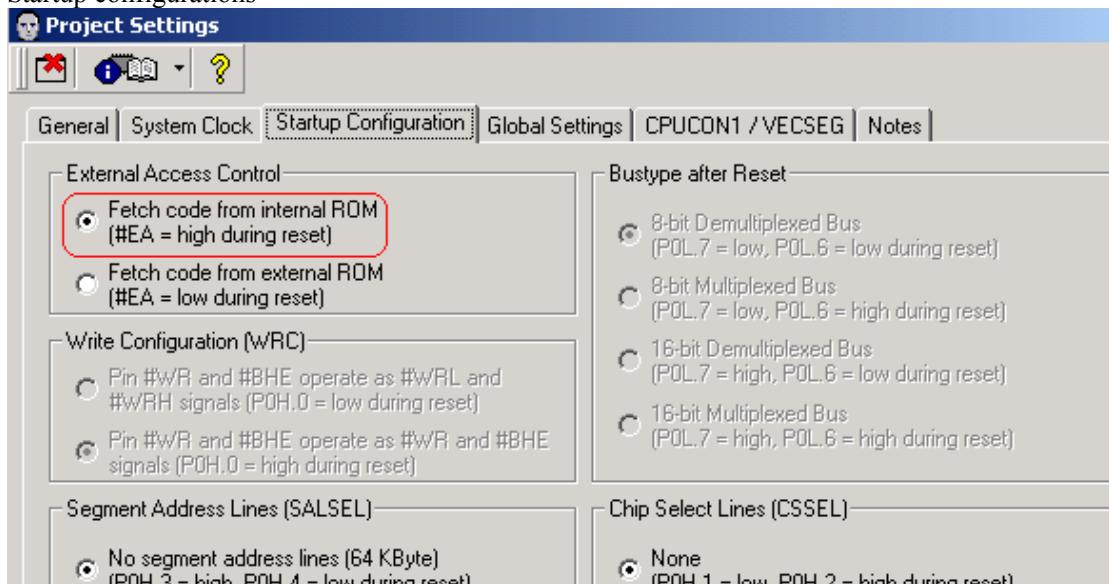
4. 2 The project Settings



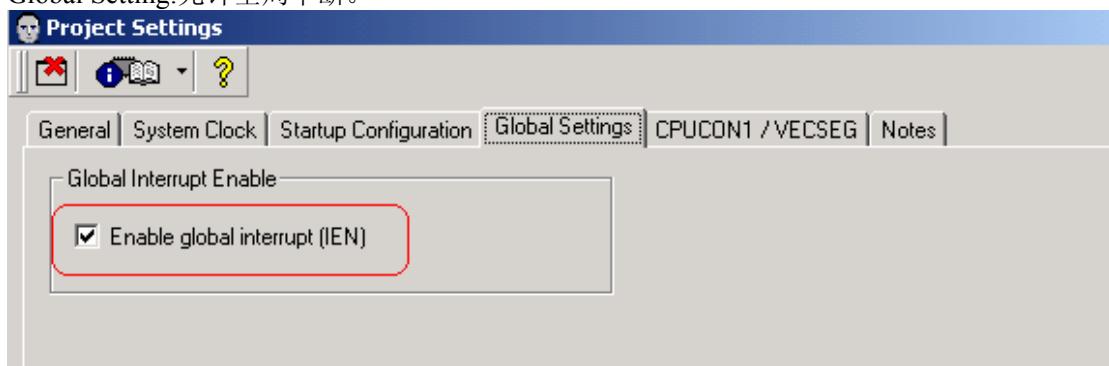
配置系统时钟为 40MHz



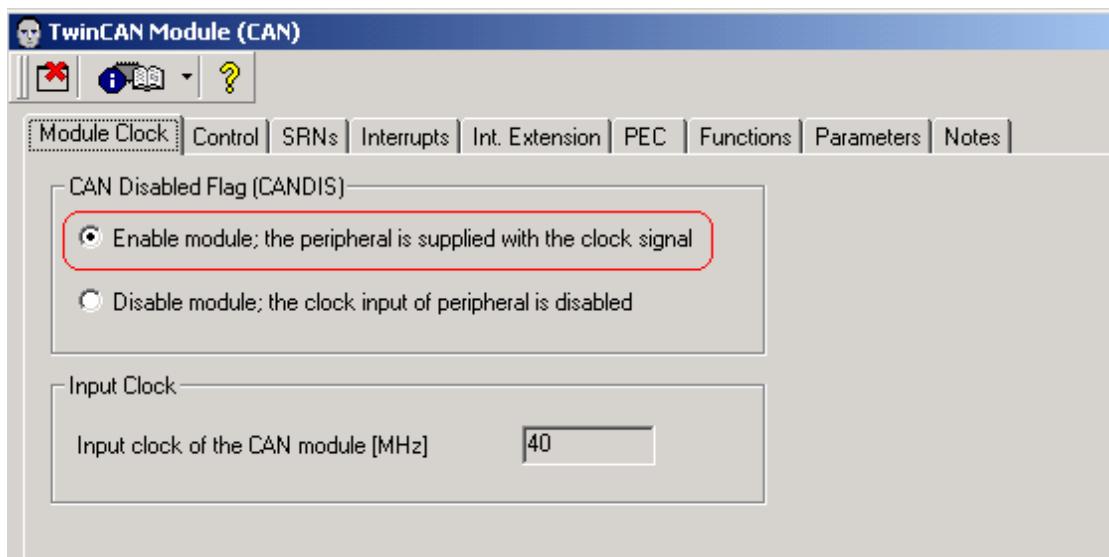
Startup configurations



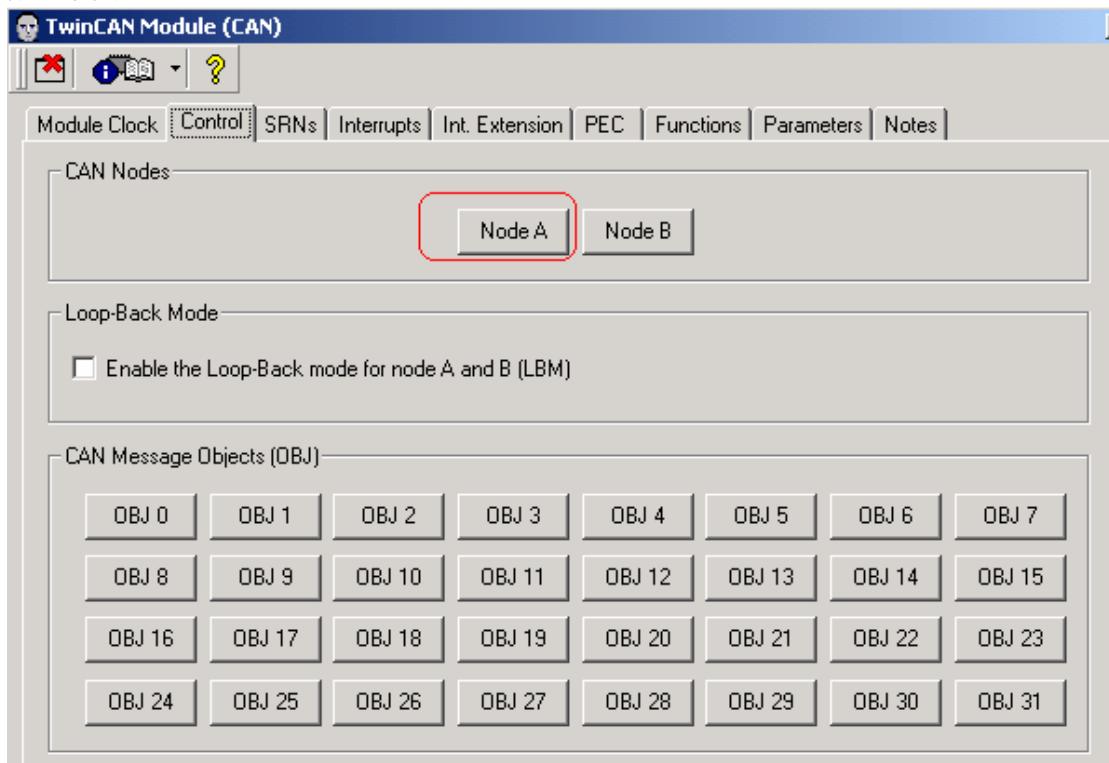
Global Setting: 允许全局中断。



4. 4 配置 Twin CAN 模块 使能模块

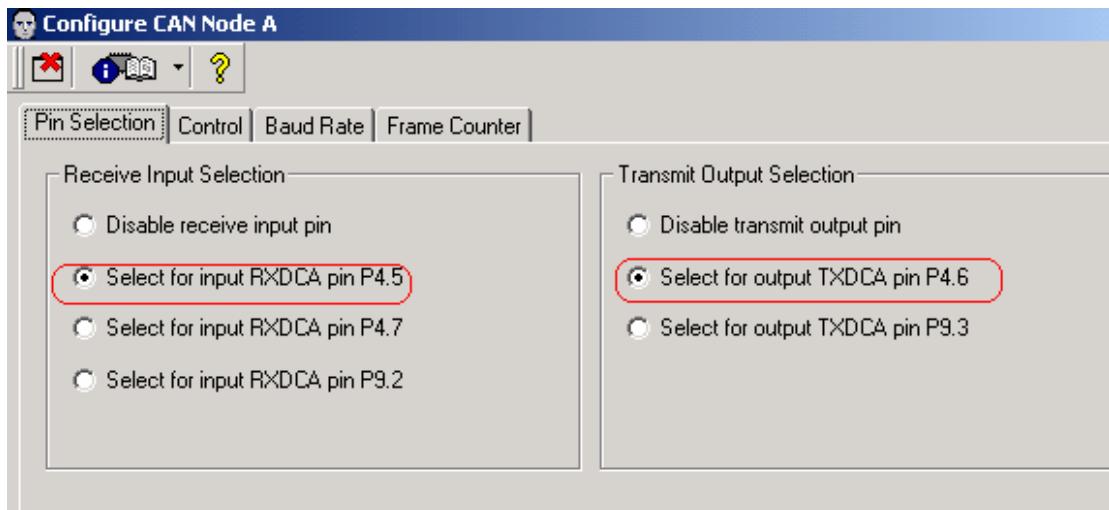


配置节点 A:

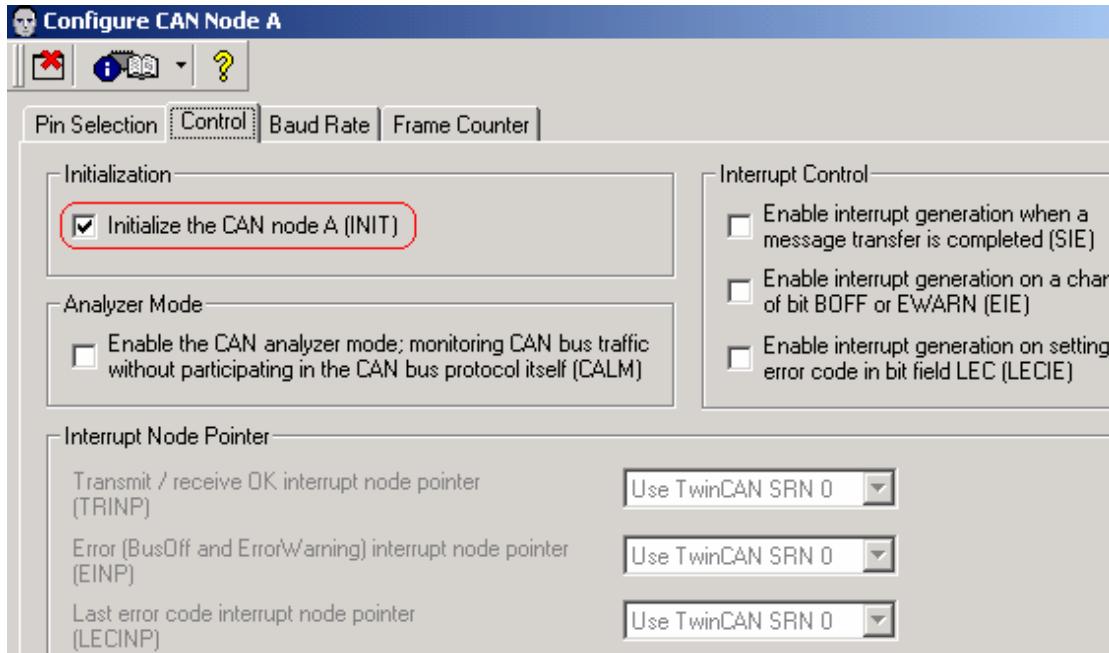


对于节点 A:

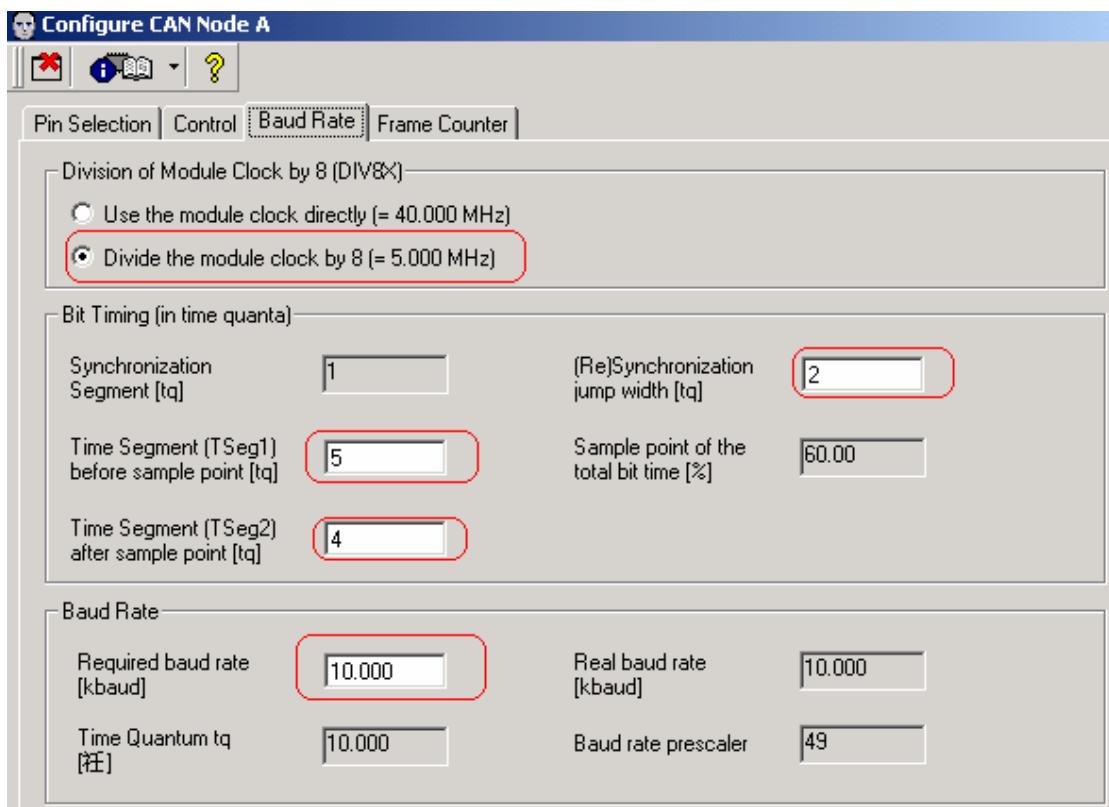
配置引脚:



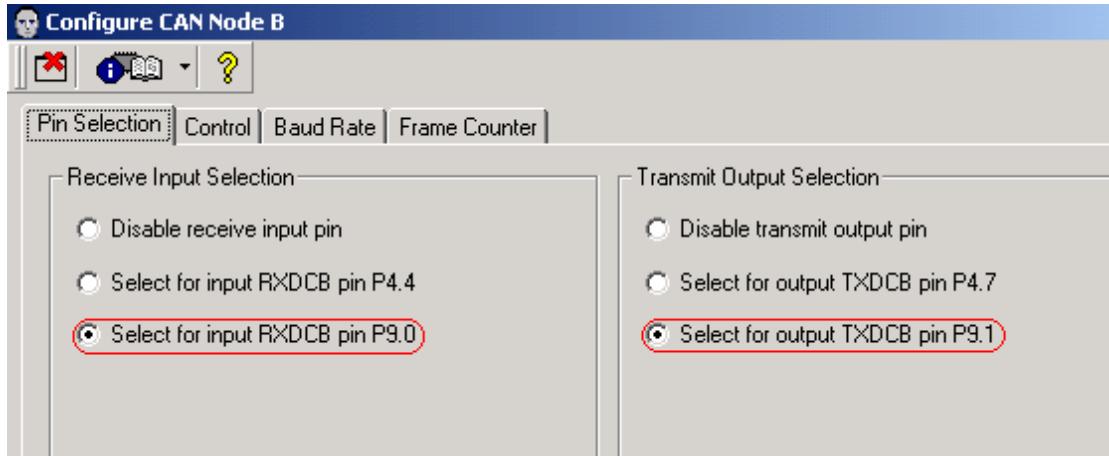
初始化节点 A:



配置节点 A 的波特率为 10k

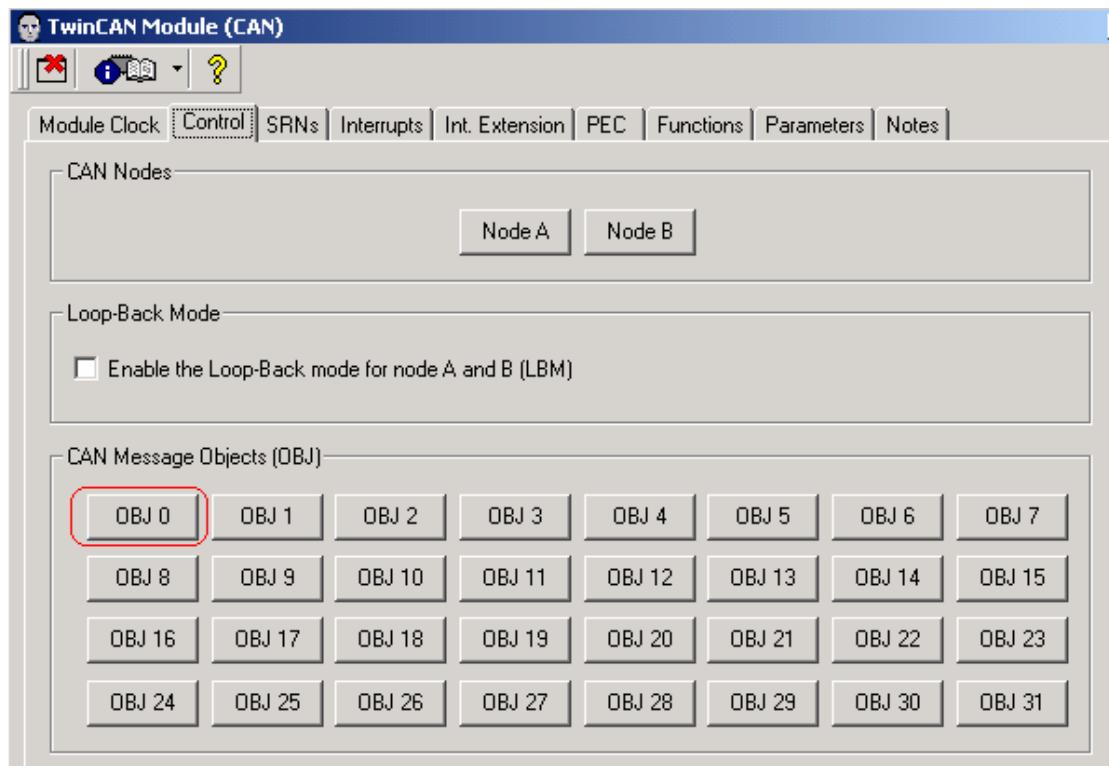


配置节点 B。引脚配置

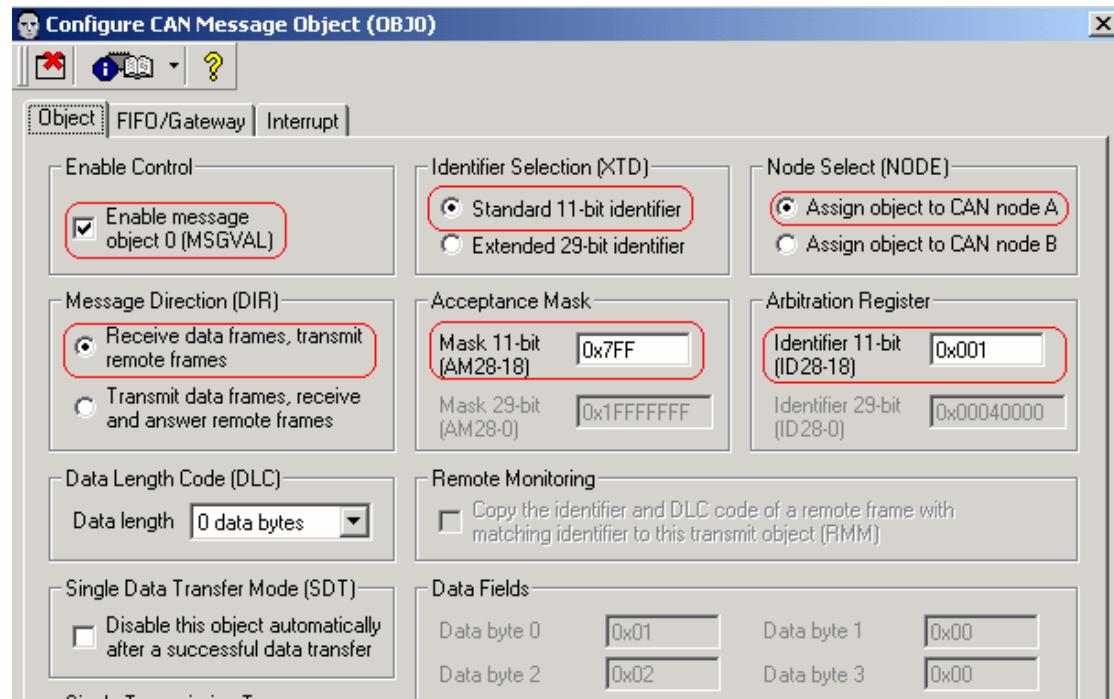


其余配置与节点 A 相同。

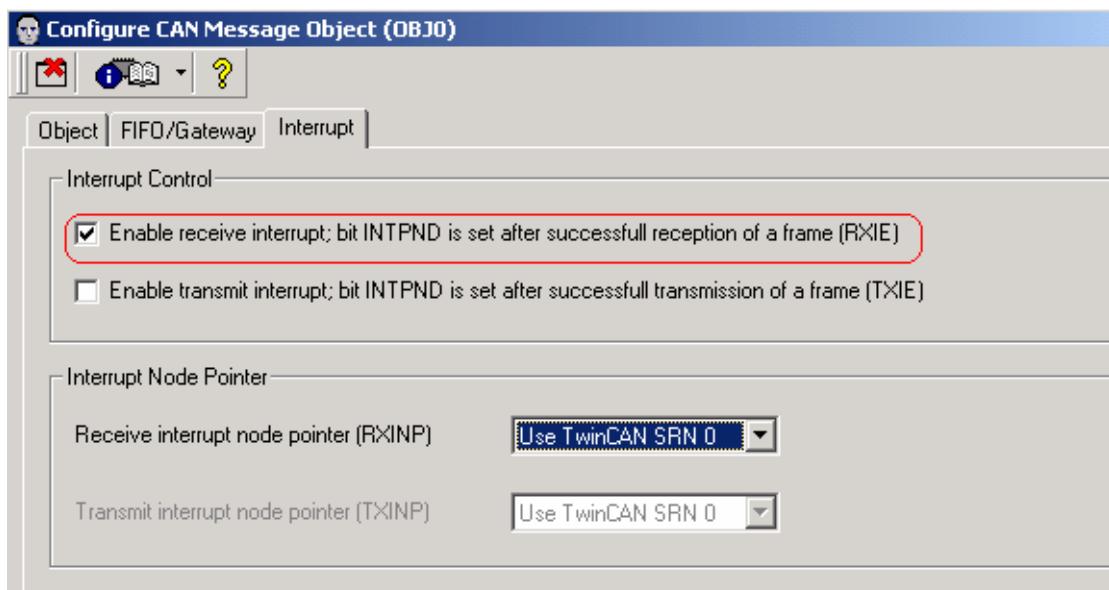
配置消息对象：0-31 配置形式基本相同，仅仅 ID 号依次为 0x001 – 0x020。
以消息对象 0 为例。



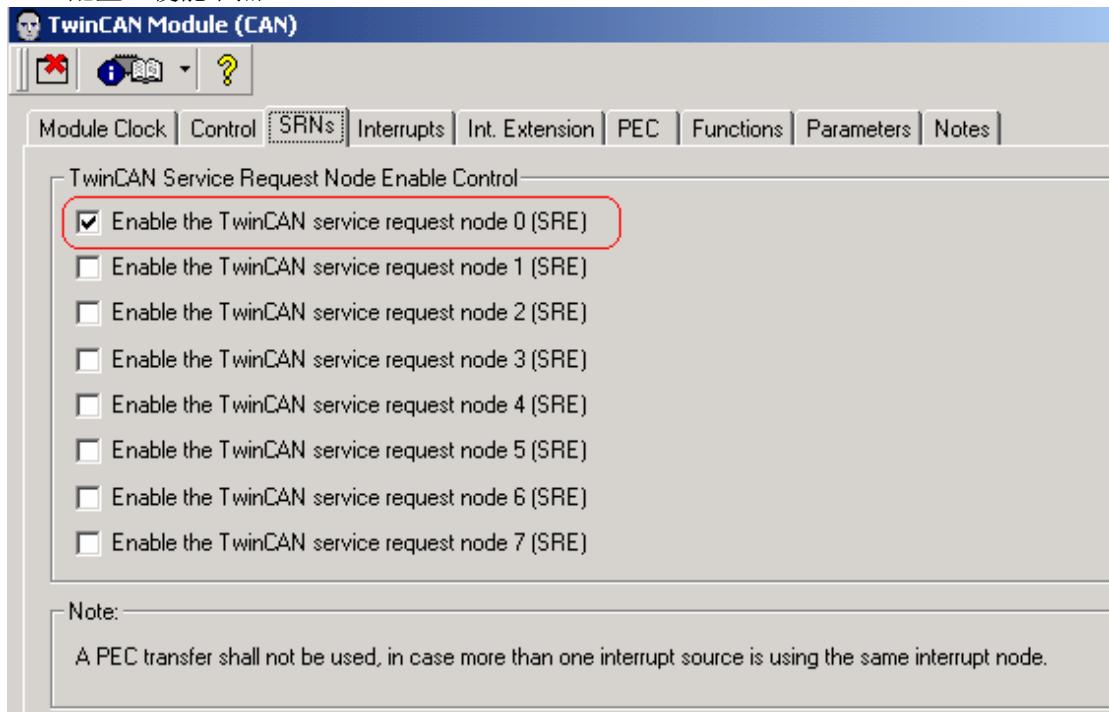
配置对象：使能节点，接收数据，11-bit ID，隶属于节点 A。MASK: 0X7FF。节点号：0x001。



Interrupt: 允许接收中断

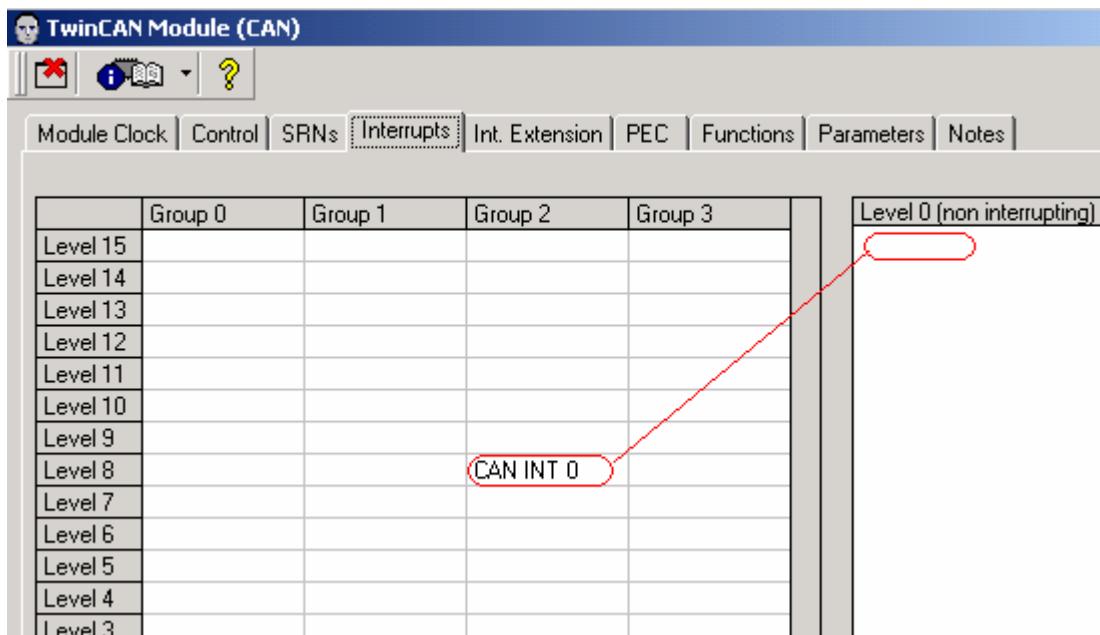


SRN 配置。使能节点 0:



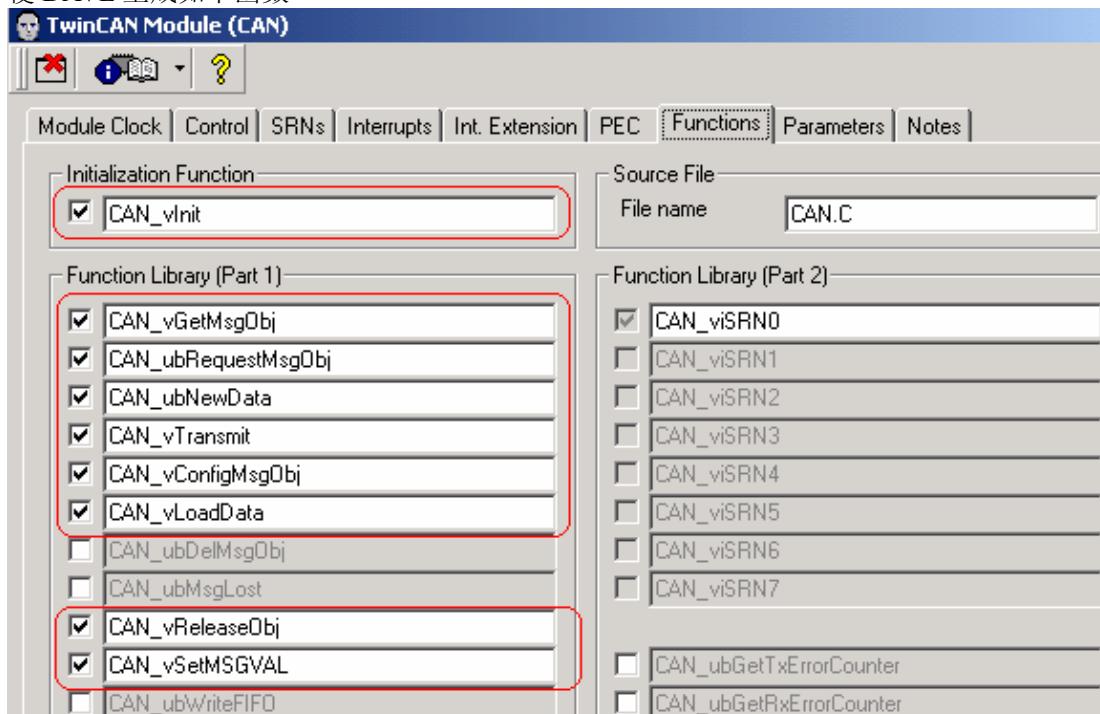
配置 CAN 中断优先级

将 CAN INT0 从右边拖到左边对应的位置，注意优先级、组别。

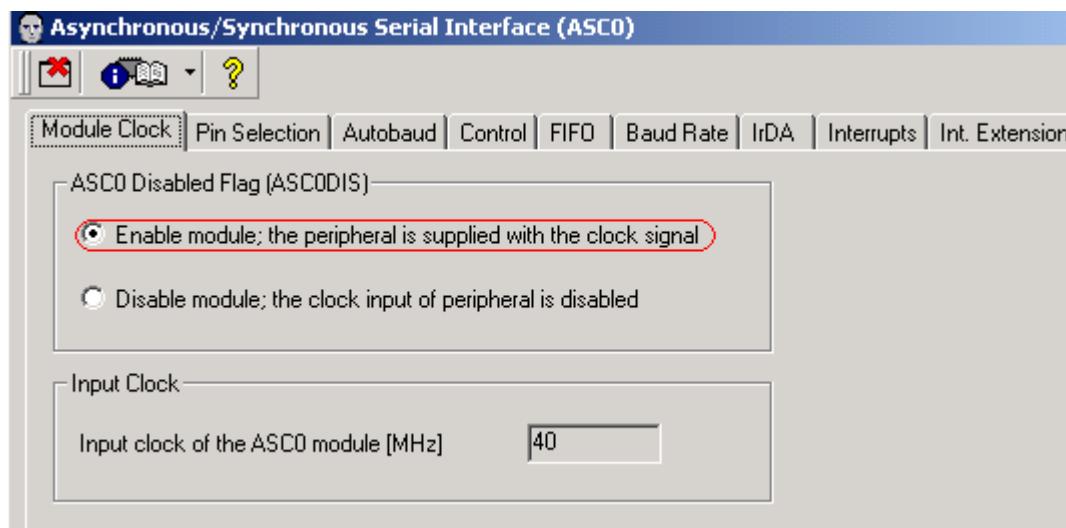


Functions:

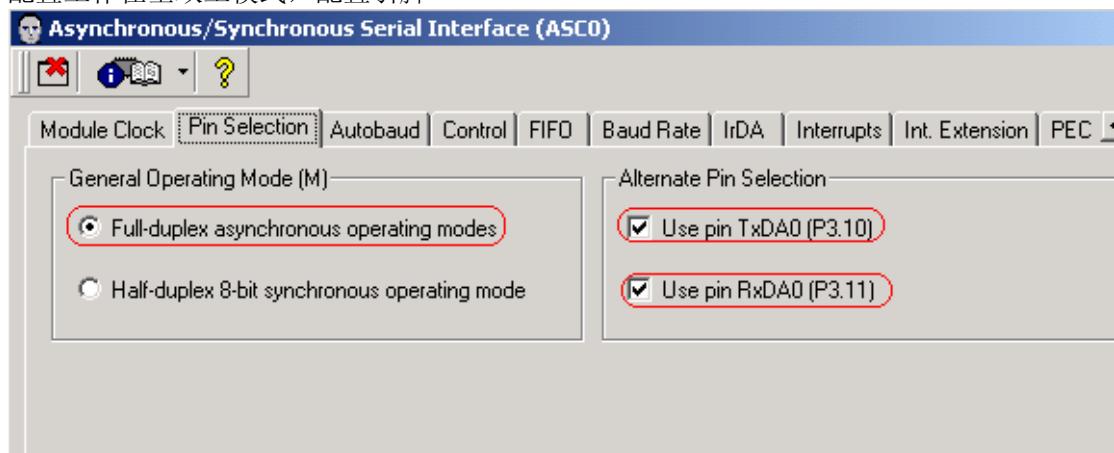
使 DAVE 生成如下函数



4. 5 配置 ASC0 用于将数据通过串口在电脑超级终端显示:

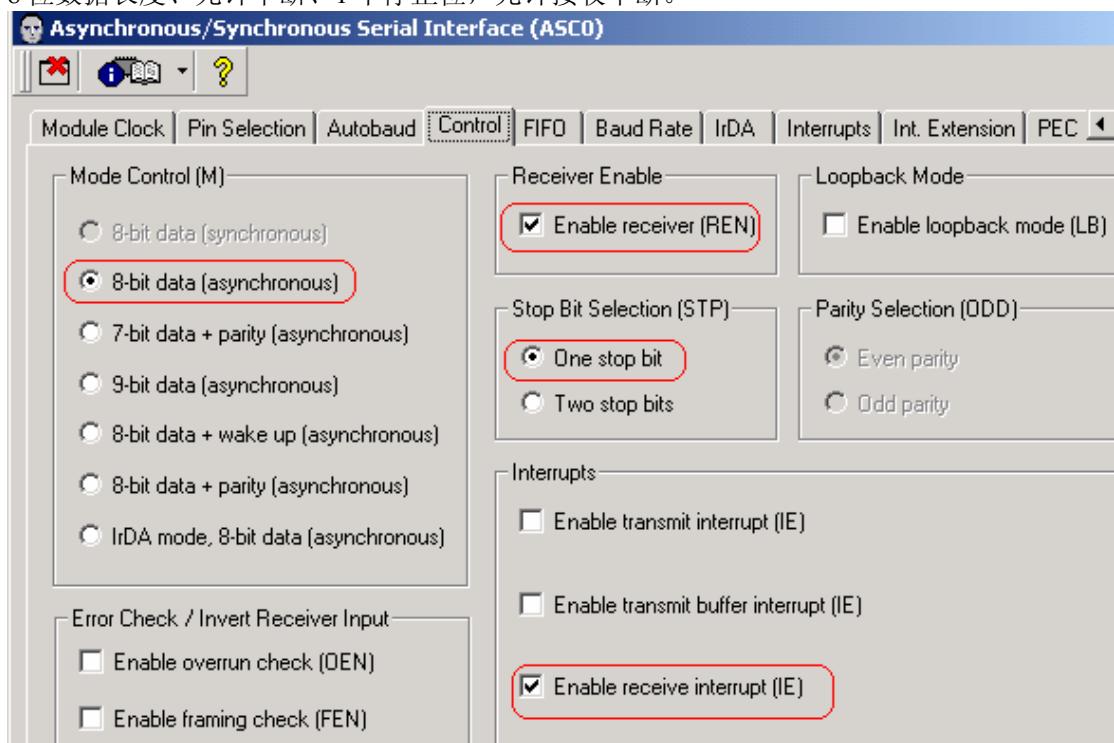


配置工作在全双工模式，配置引脚



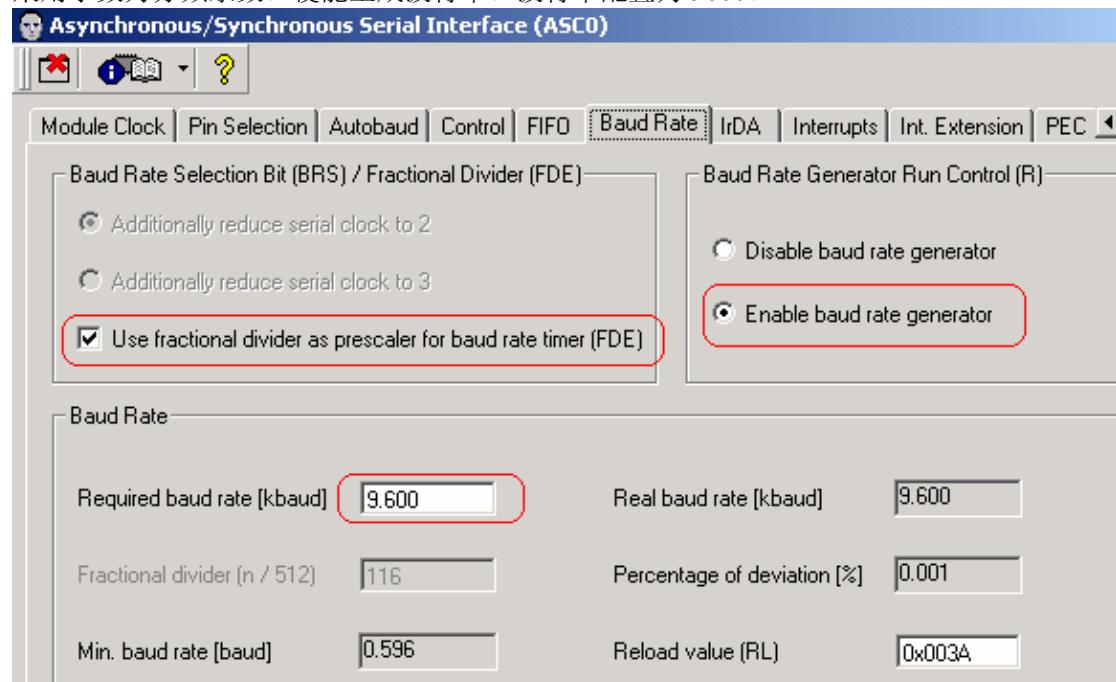
配置控制模式：

8位数据长度、允许中断、1个停止位，允许接收中断。



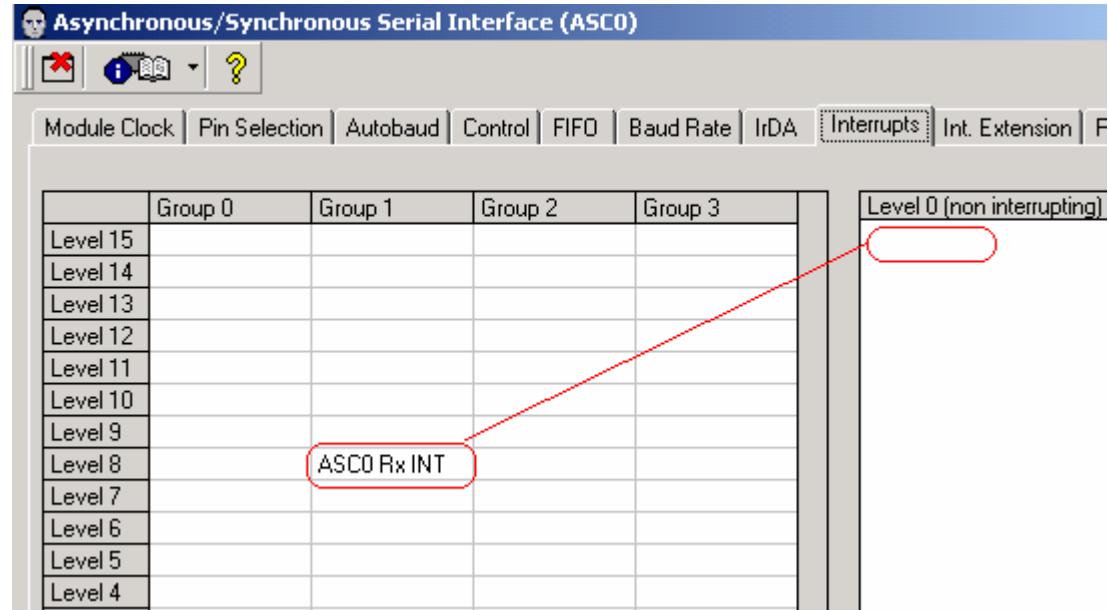
配置波特率

采用小数为分频系数。使能生成波特率。波特率配置为 9600。



配置 ASC0 中断的优先级、组别。

将 ASC0 Rx INT 从右边拖到左边相应的位置。



在 functions 页面选择如下函数

Initialization Function		Source File	
<input checked="" type="checkbox"/>	ASCO_vInit	File name	ASCO.C
Function Library (Part 1)		Function Library (Part 2)	
<input type="checkbox"/>	ASCO_vSendData	<input type="checkbox"/>	ASCO_vReceiverOn
<input checked="" type="checkbox"/>	ASCO_uwGetData	<input type="checkbox"/>	ASCO_vReceiverOff
<input type="checkbox"/>	ASCO_viTx	<input type="checkbox"/>	ASCO_vSendSlaveAdr
<input checked="" type="checkbox"/>	ASCO_vlRx	<input type="checkbox"/>	ASCO_ubOwnAddress
<input type="checkbox"/>	ASCO_viError	<input type="checkbox"/>	ASCO_vWakeUp
<input type="checkbox"/>	ASCO_viTxBuffer	<input type="checkbox"/>	ASCO_vGotoSleep
<input type="checkbox"/>	ASCO_ubTxDataReady	<input type="checkbox"/>	ASCO_vSetBaudrate
<input type="checkbox"/>	ASCO_ubTxBufFree	<input type="checkbox"/>	ASCO_vStopBaudGen

4. 6 配置 GPT1

General Purpose Timer Unit (GPT1)

Module Clock | T2 | T3 | T4 | Interrupts | Int. Extension | PEC | Functions | Parameters | Notes

GPT Disabled Flag (GPTDIS)

- Enable GPT1 AND GPT2 modules; the peripherals are supplied with the clock signal
- Disable GPT1 AND GPT2 modules; the clock input of peripherals is disabled

Input Clock

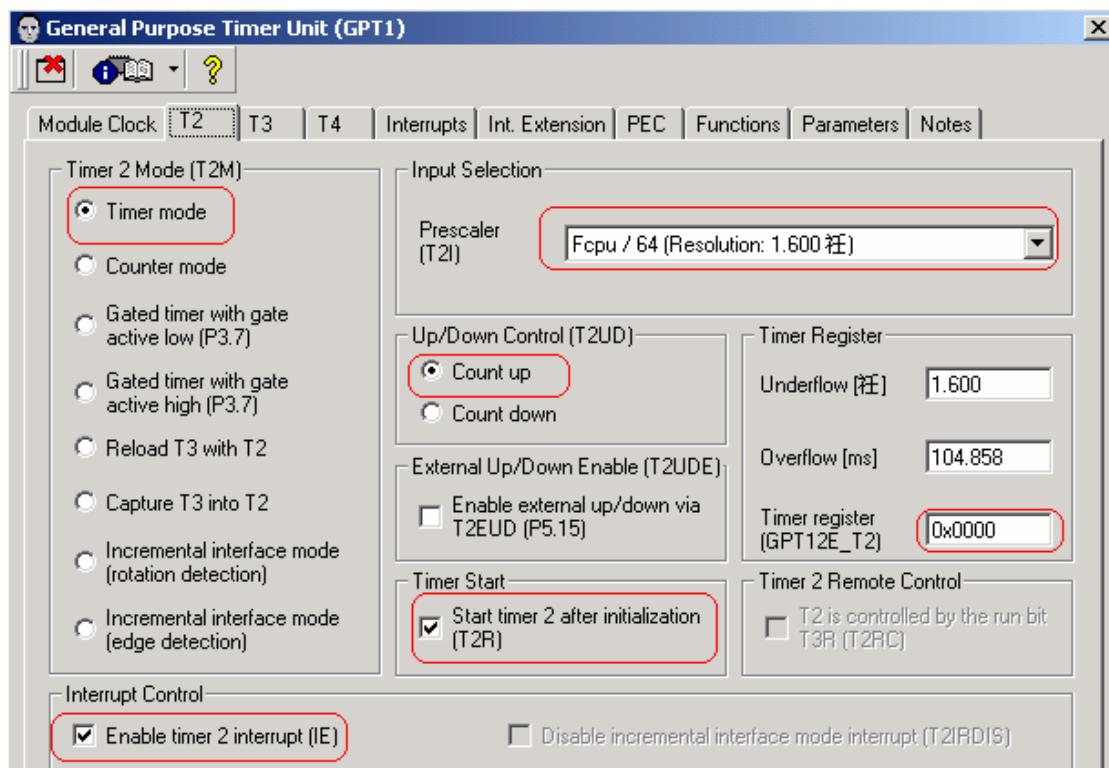
Input clock of the GPT1 module [MHz]

Timer Block Prescaler 1 (BPS1)

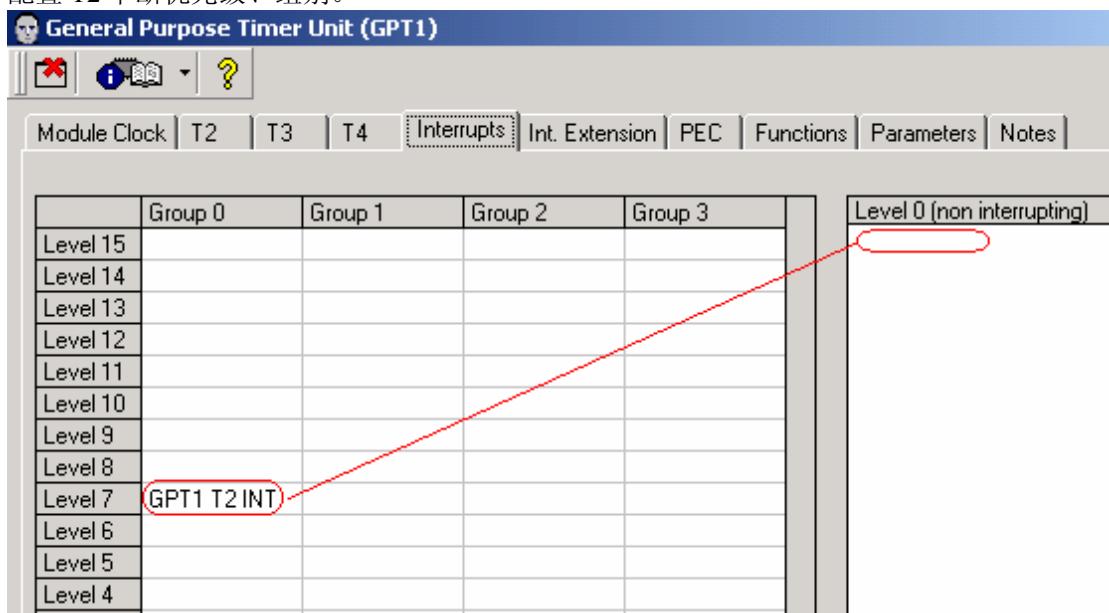
- Prescaler for timer block 1 is 4
- Prescaler for timer block 1 is 8
- Prescaler for timer block 1 is 16
- Prescaler for timer block 1 is 32

Max. input frequency for timer 2/3/4 [MHz]

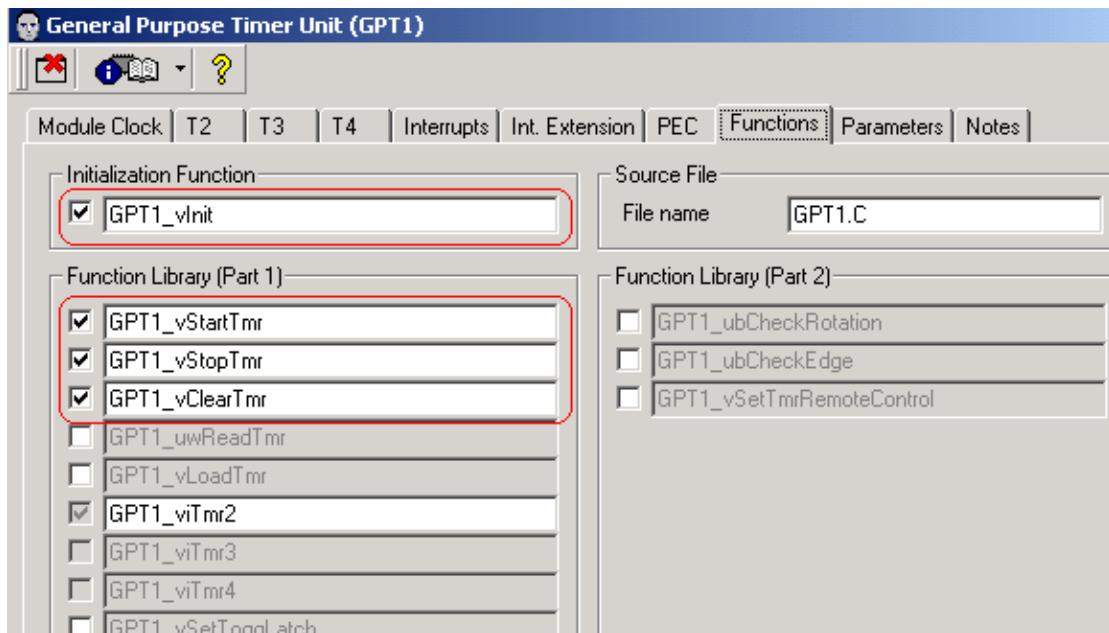
配置 T2



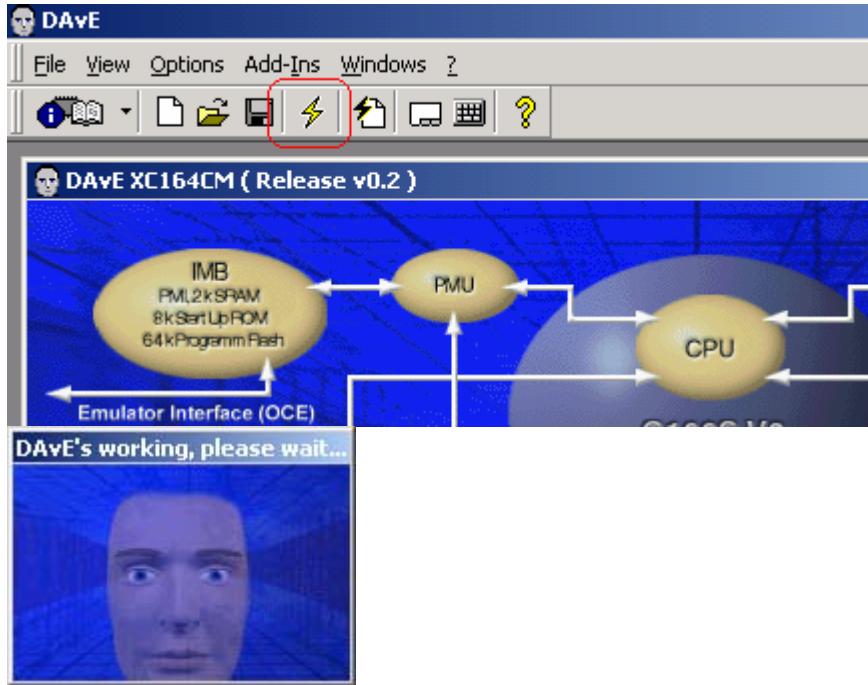
配置 T2 中断优先级、组别。



Functions: 生成如下函数



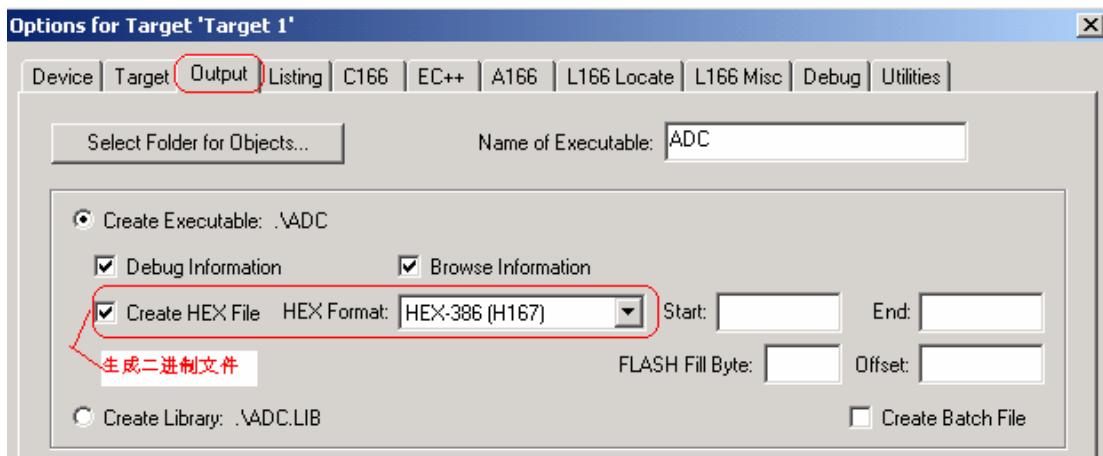
5. 利用 DAVE 生成代码。



6. 修改代码，添加用户自定义代码

6. 1 生成 uVision 工程文件

做完以上步骤之后工程文件夹中会出现 keil 图标的 dpt 文件，双击进入 keil 环境。第一次进入 keil 环境需要设置：project—options for target ‘target 1’。设置如下：



6. 2 修改 main.c

添加引用声明

```
*****  
// @Defines  
*****
```

```
// USER CODE BEGIN (MAIN_General,4)
extern struct stCanObj
{
    ubyte ubData[8]; // Message Data 0..7
    ulong ulCANAR; // Arbitration Register
    ulong ulCANAMR; // Acceptance Mask Register
    dword uwMSGCTR; // Message Control Register
    dword uwCounter; // Frame Counter
    dword uwMSGCFG; // Message Configuration Register
    dword uwINP; // Interrupt Node Pointer
    dword uwCANFCR; // FIFO / Gateway Control Register
    dword uwCANPTR; // FIFO Pointer
    ulong ulReserved; // Reserved
};

// USER CODE END
```

添加全局变量

```
*****  
// @Global Variables  
*****
```

```
// USER CODE BEGIN (MAIN_General,7)
unsigned char KeyBuff[12] = {r', 0x00, 0x00};
unsigned char WriteIndex = 1;
unsigned char ReadIndex = 0;
unsigned int CursorPosX;
unsigned int CursorPosY;
unsigned int MO_Status[32]= {0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01,
                            0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01,
                            0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01,
                            0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01};
TCAN_SWObj SW_MOs[32];
unsigned int MenuItem=0;
// USER CODE END
```

```

void main(void)
{
// USER CODE BEGIN (Main,2)
    unsigned int i;
// USER CODE END
MAIN_vInit();

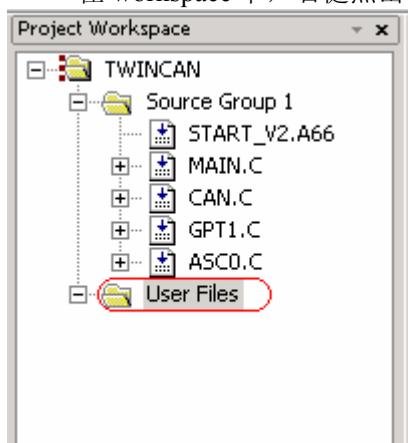
// USER CODE BEGIN (Main,4)
// USER CODE BEGIN (Main,4)
// 添加对各个消息对象的初始化，禁用消息对象。
for (i = 0; i<32; i++)
    CAN_HWOBJ[i].uwMSGCTR = 0xff7f; // disable MO
// 添加 while(1);
while(1)
{
}
// USER CODE END。

```

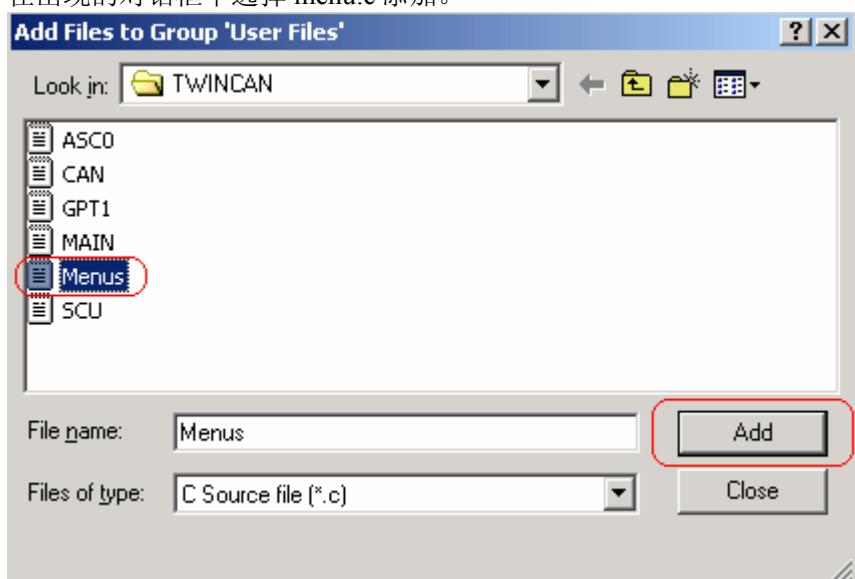
6. 3 添加用户程序

将例程中的 Menu.c 拷贝到项目所在文件夹。

在 workspace 中，右键点击 user Files，选择 add files to User group。



在出现的对话框中选择 menu.c 添加。



关于菜单的具体实现，请参考 menus.c 文件。

6. 4 修改 Main.h

添加对 menus.c 中的函数的引用。

```
*****  
// @Prototypes Of Global Functions  
*****  
void MAIN_vUnlockProtecReg(void);  
  
// USER CODE BEGIN (MAIN_Header,8)  
void PrintMainMenu(void);  
void MoveCursor(unsigned int y, unsigned int x);  
void UpdateStatus(void);  
void MenuHandler(void);  
void PrintPrompt(unsigned int p, unsigned int mo);  
// USER CODE END  
  
// USER CODE BEGIN (MAIN_Header,10)  
#include <STDIO.h>  
extern TCAN_SWObj SW_MOs[];  
// USER CODE END
```

6. 5 修改 gpt1.c

添加定时处理向串口发送菜单数据

```
void GPT1_viTmr2(void) interrupt T2INT  
{  
    // USER CODE BEGIN (Tmr2,2)  
  
    // USER CODE END
```

```
// USER CODE BEGIN (Tmr2,5)  
GPT1_vStopTmr(GPT1_TIMER_2); // 停止 T2  
GPT1_vClearTmr(GPT1_TIMER_2); // 清除计数值  
DP9_P4=!DP9_P4; // 状态显示位  
MenuHandler(); // 处理菜单  
UpdateStatus(); // 更新状态信息。  
GPT1_vStartTmr(GPT1_TIMER_2);  
// USER CODE END
```

```
} // End of function GPT1_viTmr2
```

6. 6 修改 ASC0.c

接收从电脑超级终端传来的用户输入的信息。

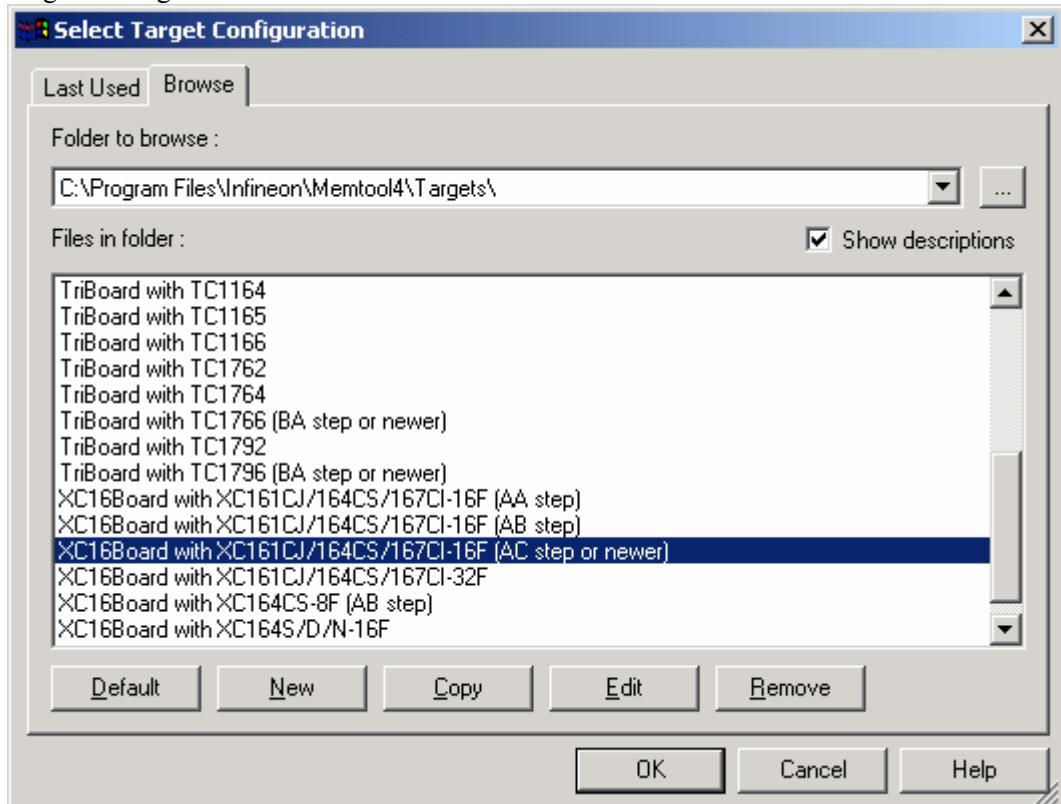
```
// USER CODE BEGIN (ASC0_Function,3)  
ASC0_TIC_IR = 1;  
// USER CODE END  
  
void ASC0_viRx(void) interrupt ASC0_RINT  
{  
  
    // USER CODE BEGIN (Rx,2)  
    KeyBuf[WriteIndex++] = (unsigned char)ASC0_uwGetData();  
    if (WriteIndex == 12)  
        WriteIndex = 0;  
    // USER CODE END  
  
} // End of function ASC0_viRx
```

7. 编译

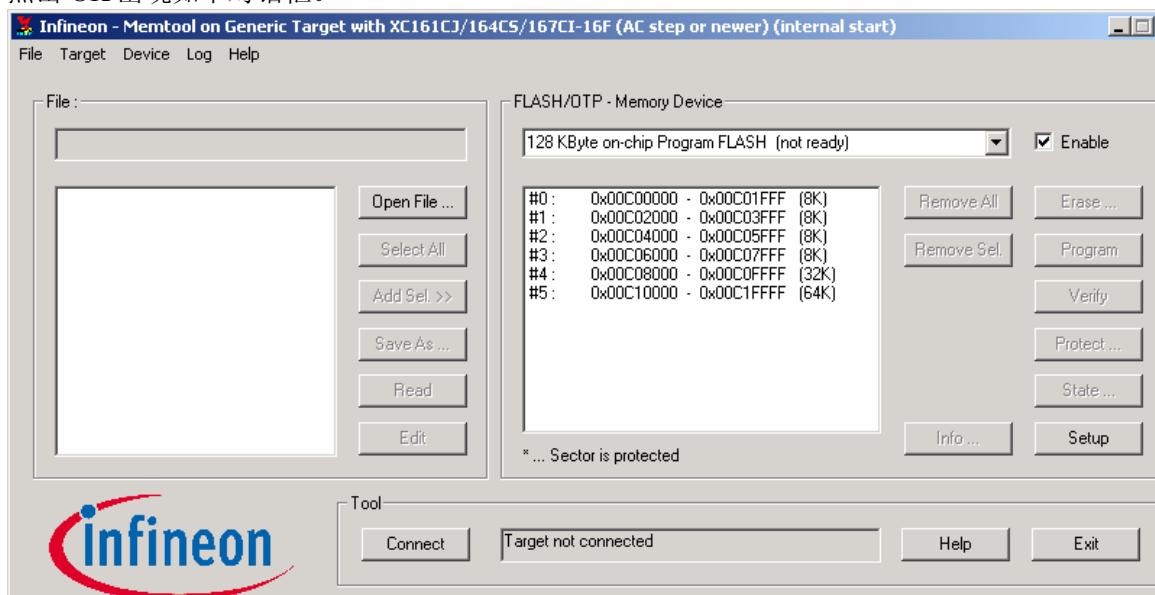
点击  图标进行编译连接。如有错误进行更改，直到出现‘0 Errors found.’。

8. 下载

利用 memtool 软件将上面生成的 h86 文件下载到单片机。打开 memtool 软件，点击菜单 Target—Change，选择 XC164CS。界面如下：



点击 OK 出现如下对话框。

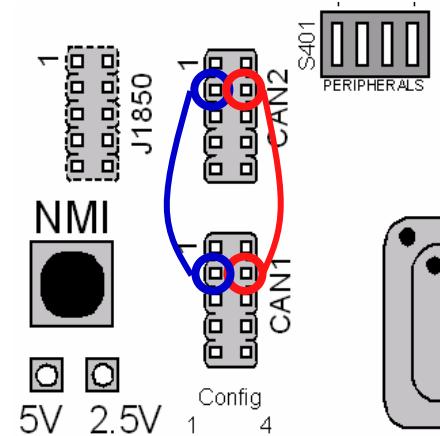


点击‘connect’进行通讯连接。通讯成功之后，按照顺序 open file...—select all—add sel.>>将 h86 文件添加到右边框中，然后选择‘Erase...’和‘Program’进行擦除、编程。如有必要可点击‘Verify’进行校验。

9. 运行

将串口线连接至电脑，开启超级终端，使用 9600 8-N-1。

配置 XC164CS starter kit。连接 CAN1_L 与 CAN2_L，连接 CAN1_H 与 CAN2_H，使节点 A 与节点 B 相连。



StarterKit 上电，电脑超级终端，显示如下：

MO:	Node:	DIR:	ID:	Mask:	Data:	MO:	Node:	DIR:	ID:	Mask:	Data:
00	-	--	-----	-----	-----	01	-	--	-----	-----	-----
02	-	--	-----	-----	-----	03	-	--	-----	-----	-----
04	-	--	-----	-----	-----	05	-	--	-----	-----	-----
06	-	--	-----	-----	-----	07	-	--	-----	-----	-----
08	-	--	-----	-----	-----	09	-	--	-----	-----	-----
10	-	--	-----	-----	-----	11	-	--	-----	-----	-----
12	-	--	-----	-----	-----	13	-	--	-----	-----	-----
14	-	--	-----	-----	-----	15	-	--	-----	-----	-----
16	-	--	-----	-----	-----	17	-	--	-----	-----	-----
18	-	--	-----	-----	-----	19	-	--	-----	-----	-----
20	-	--	-----	-----	-----	21	-	--	-----	-----	-----
22	-	--	-----	-----	-----	23	-	--	-----	-----	-----
24	-	--	-----	-----	-----	25	-	--	-----	-----	-----
26	-	--	-----	-----	-----	27	-	--	-----	-----	-----
28	-	--	-----	-----	-----	29	-	--	-----	-----	-----
30	-	--	-----	-----	-----	31	-	--	-----	-----	-----

Enter 'E' to Edit MO, Enter 'T' to Transmit MO, Enter 'R' to Refresh:

选择 ‘E’ 配置消息对象 0，根据提示输入。

Enter Field to Edit: N=Node, R=RX, T=TX, I=ID, M=Mask, D=Data, V=Enable\Disable

配置如下：隶属于节点 A，TX，ID：0x001，Mask：0x7ff，Data：Infineon，V(enable)。

点击 ‘Esc’ 键推到上级目录。接着输入 31，配置消息对象 31：

隶属于节点 B，RX，ID：0x001（需要与发送的消息对象相同），Mask：0x7ff，V(enable)。

配置完成后如下所示：

MO:	Node:	DIR:	ID:	Mask:	Data:	MO:	Node:	DIR:	ID:	Mask:	Data:
00	H	TX	0x001	0x7ff	Infineon	01	-	--	--	--	--
02	-	--	--	--	--	03	-	--	--	--	--
04	-	--	--	--	--	05	-	--	--	--	--
06	-	--	--	--	--	07	-	--	--	--	--
08	-	--	--	--	--	09	-	--	--	--	--
10	-	--	--	--	--	11	-	--	--	--	--
12	-	--	--	--	--	13	-	--	--	--	--
14	-	--	--	--	--	15	-	--	--	--	--
16	-	--	--	--	--	17	-	--	--	--	--
18	-	--	--	--	--	19	-	--	--	--	--
20	-	--	--	--	--	21	-	--	--	--	--
22	-	--	--	--	--	23	-	--	--	--	--
24	-	--	--	--	--	25	-	--	--	--	--
26	-	--	--	--	--	27	-	--	--	--	--
28	-	--	--	--	--	29	-	--	--	--	--
30	-	--	--	--	--	31	B	RX	0x001	0x7ff	--

Enter 'E' to Edit MO, Enter 'T' to Transmit MO, Enter 'R' to Refresh:
-

按 Esc 退到上图所示界面。按 T 进行发送，输入传输 MO 号：00，回车。
结果如下。如果消息对象 31 数据没有发生变化，按 ‘R’ 键进行刷新。

MO:	Node:	DIR:	ID:	Mask:	Data:	MO:	Node:	DIR:	ID:	Mask:	Data:
00	A	TX	0x001	0x7ff	Infineon	01	-	--	--	--	--
02	-	--	--	--	--	03	-	--	--	--	--
04	-	--	--	--	--	05	-	--	--	--	--
06	-	--	--	--	--	07	-	--	--	--	--
08	-	--	--	--	--	09	-	--	--	--	--
10	-	--	--	--	--	11	-	--	--	--	--
12	-	--	--	--	--	13	-	--	--	--	--
14	-	--	--	--	--	15	-	--	--	--	--
16	-	--	--	--	--	17	-	--	--	--	--
18	-	--	--	--	--	19	-	--	--	--	--
20	-	--	--	--	--	21	-	--	--	--	--
22	-	--	--	--	--	23	-	--	--	--	--
24	-	--	--	--	--	25	-	--	--	--	--
26	-	--	--	--	--	27	-	--	--	--	--
28	-	--	--	--	--	29	-	--	--	--	--
30	-	--	--	--	--	31	B	RX	0x001	0x7ff	Infineon

Enter 'E' to Edit MO, Enter 'T' to Transmit MO, Enter 'R' to Refresh:
-