本节介绍 CAN 总线通信的实现过程，首先介绍 CAN 总线通信硬件参考电路，在此基础上给出具体的代码设计过程。

16.3.1 CAN 总线通信硬件电路的设计

STC 官方给出的基于 STC32G 系列单片机和 NXP（中文称恩智浦）TJA1050 收发器的参考电路，如图 16.25 所示。在本章前面已经详细介绍了 STC32G 系列单片机内部 CAN 模块的功能，本节对 CAN 收发器芯片 TJA1050 进行简要介绍，以帮助读者理解 CAN 规范的物理层。

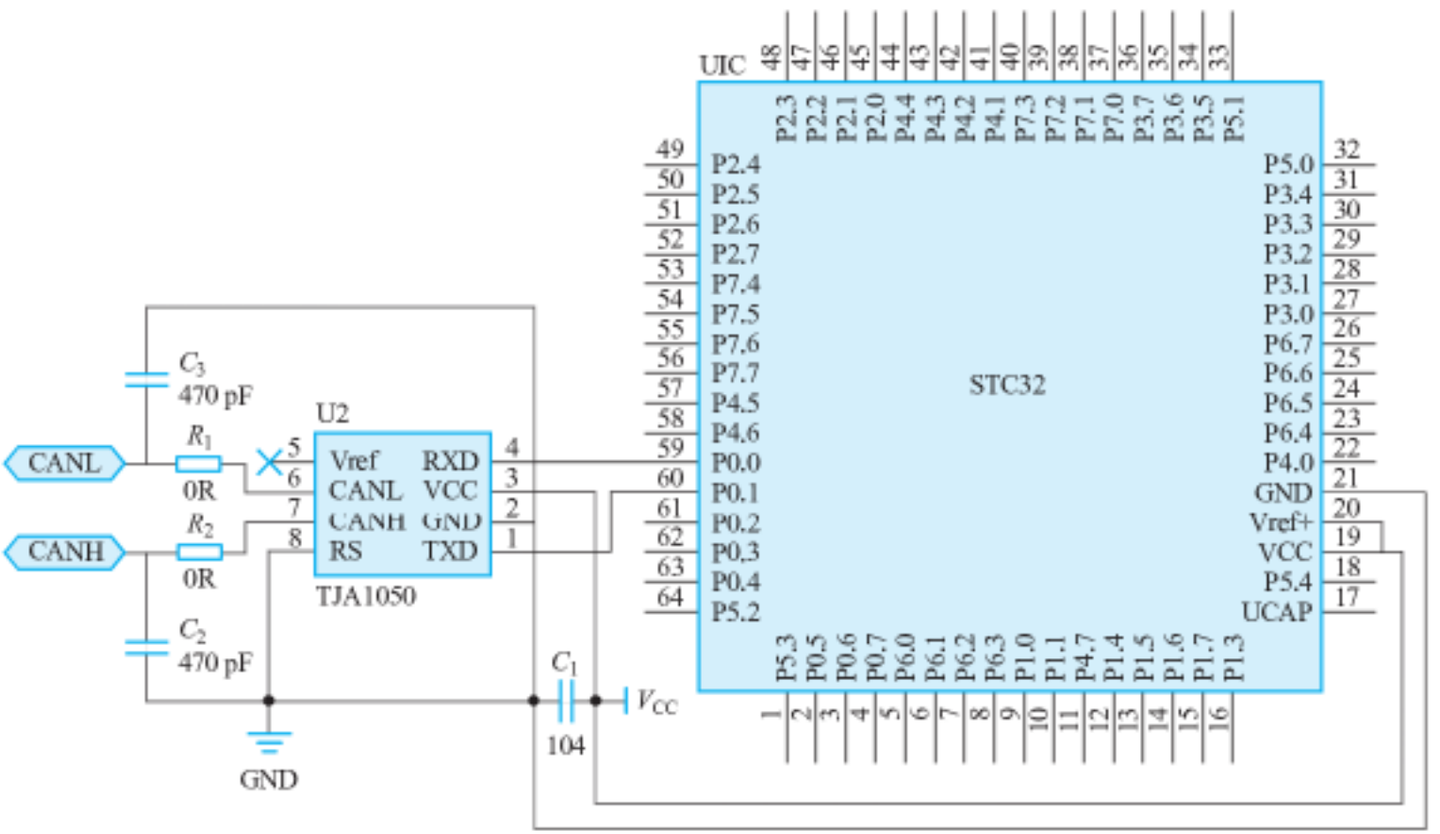

图 16.25 基于 STC32G 系列单片机和 NXP（中文称恩智浦）TJA1050 收发器的参考电路

CAN 收发器芯片 TJA1050 的内部结构，如图 16.26 所示。

注：建议读者使用 STC 官方提供的“屠龙刀”开发板，在该开发板上提供了 STC32G 系列单片机，读者可以自行采购 TJA1050 芯片焊接到该开发板上，并通过下面的软件代码设计来实现基于 CAN 总线的通信。

16.3.2 CAN 总线通信的软件代码设计

本节将在 Keil μVision（C251 版本）IDE 中，设计用于实现 CAN 宗信啊通信的软件应用。

1. 建立新的设计工程

建立新设计工程的主要步骤包括：

（1）启动 μVision5（C251 版本）集成开发环境。

（2）在 Keil μVision5 集成开发环境（下面简称 Keil）主界面主菜单下，选择 Project->New μVision Project...。

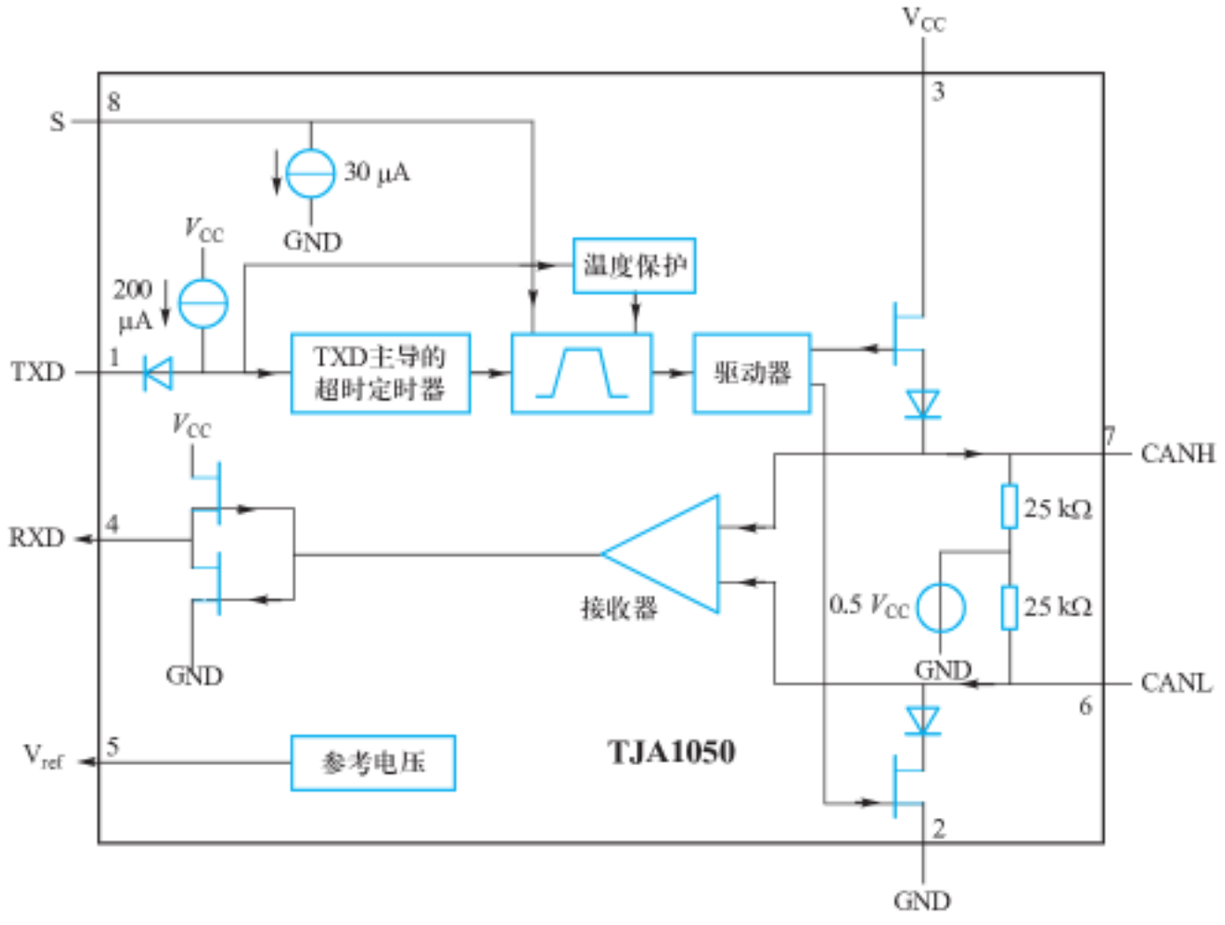

图 16.26　CAN 收发器芯片 TJA1050 内部结构

（3）弹出 Create New Project 对话框界面。在该界面中：

① 将路径定位到 E:\stc32_example\example_16_1。

② 在文件名(N)右侧的文本框中输入 can1tx。

（4）单击保存按钮。

（5）出现 Select a CPU Data Base File 对话框界面。在该界面中的下拉框中，选择 STC MCU Database 选项。

（6）单击 OK 按钮，退出 Select a CPU Data Base File 对话框界面。

（7）出现 Select Device for Target'Target 1'... 对话框界面。在该界面的 Device 标签页面下，通过下拉框选择 STC MCU Database。然后，在下面的左侧窗口的器件列表中，找到并展开 STC 前面的"+"。此时，以列表的形式给出了可用的 STC 单片机型号。在展开项中，找到并选择 STC3232G12K128 Series。

（8）单击 OK 按钮。

（9）选中 Target 1 文件夹。单击鼠标右键，出现浮动菜单。在浮动菜单内，选择 Options for Target'Target 1'... 。

（10）弹出 Options for Target'Target 1'对话框界面。按如下设置参数：

① CPU Mode:Source(251 native)。

② Memory Model:XSmall:near vars,for const,ptr-4。

③ Code Rom Size:Large:64K program。

单击 Output 标签。在该标签界面下，选中 Create HEX File 前面的复选框。

（11）单击 OK 按钮，退出 Options for Target'Target 1'对话框界面。

2. 添加新的设计文件

本节将在当前工程中添加新的 C 语言文件，主要步骤包括：

（1）在 Project 窗口界面下，选择 Source Group 1，单击右键，出现浮动菜单。在浮动菜单内，选择 Add New Item to Group‘Source Group 1’选项。

（2）出现 Add New Item to Group’Source Group 1’对话框界面，按下面设置参数：

① Type：选择 C File(.c)

② Name：cantx

③ location：E:\stc32_example\example_16_1

（3）单击 Add 按钮，退出 Add New Item to Group‘Source Group 1’对话框界面。

（4）在的 Project 窗口中，在 Source Group 1 子目录下添加了名字为 main.c 的 C 语言文件。

（5）在 Keil μVision5 主界面左侧的 Project 窗口中，找到并双击 can1tx.c 文件。在该文件中添加设计代码，如代码清单 16-1 所示。

代码清单 16-1　can1tx.c 文件

```
#include "..\..\comm\STC32G.h"
#include "intrins.h"

typedef  unsigned char    u8;
typedef  unsigned int     u16;
typedef  unsigned long    u32;
#define MAIN_Fosc        24000000UL

/************************* 用户定义宏 *******************************/
//CAN 总线波特率=Fclk/((1+(TSG1+1)+(TSG2+1))*(BRP+1)*2)
#define TSG1    2     //0~15
#define TSG2    1     //1~7 (TSG2 不能设置为 0)
#define BRP     3     //0~63
//24000000/((1+3+2)*4*2)=500KHz

#define SJW     0     //重新同步跳跃宽度

//总线波特率 100KHz 以上设置为 0; 100KHz 以下设置为 1
#define SAM     0     //总线电平采样次数: 0:采样 1 次; 1:采样 3 次

/*************************  本地常量声明    ****************************/
/******************定时器 0 中断频率设置为 1000 次/秒 **************/
#define Timer0_Reload   (65536UL -(MAIN_Fosc / 1000))

/*************************  本地变量声明    ************************/
u16 CAN_ID;
u8 RX_BUF[8];
u8 TX_BUF[8];
```

```
bit B_CanRead;                  //CAN 收到数据标志
bit B_1ms;                      //1ms 标志
u16 msecond;

/**************  本地函数声明    ***********************/
void CANInit();
void CanSendMsg(u16 canid, u8 *pdat);
u8 CanReadReg(u8 addr);
u16 CanReadMsg(u8 *pdat);
/**********************主函数  ****************************/
void main(void)
{
    u8 sr;
/**设置程序指令延时参数,赋值为 0 可将 CPU 执行指令的速度设置为最快 **/
    WTST = 0;
    EAXFR = 1;                  // 扩展寄存器(XFR)访问使能
    CKCON = 0;                  // 提高访问 XRAM 速度
/****设置 P0.4、P0.5 为漏极开路(实验箱加了上拉电阻到 3.3V) *****/
    P0M1 = 0x30;
    P0M0 = 0x30;
/****设置 P1.1、P1.4、P1.5 为漏极开路(实验箱加了上拉电阻到 3.3V)***/
/****P1.1 在 PWM 当 DAC 电路通过电阻串联到 P2.3 *****/
    P1M1 = 0x32;
    P1M0 = 0x32;
/****设置 P2.2~P2.5 为漏极开路(实验箱加了上拉电阻到 3.3V) ******/
    P2M1 = 0x3c;
    P2M0 = 0x3c;
/****设置 P3.4、P3.6 为漏极开路(实验箱加了上拉电阻到 3.3V) *****/
    P3M1 = 0x50;
    P3M0 = 0x50;
/****设置 P4.2~P4.5 为漏极开路(实验箱加了上拉电阻到 3.3V) *****/
    P4M1 = 0x3c;
    P4M0 = 0x3c;
/*****设置 P5.2、P5.3 为漏极开路(实验箱加了上拉电阻到 3.3V) *****/
    P5M1 = 0x0c;
    P5M0 = 0x0c;
/*****设置为漏极开路(实验箱加了上拉电阻到 3.3V) **********/
    P6M1 = 0xff;
    P6M0 = 0xff;
/******设置为准双向口 **************/
    P7M1 = 0x00;
    P7M0 = 0x00;
    AUXR = 0x80;                // 定时器 0 设置为 1T, 16 位定时器自动重加载
    TH0 = (u8)(Timer0_Reload / 256);
```

```
    TL0 = (u8)(Timer0_Reload %256);
    ET0 = 1;                                    // 定时器 0 中断使能
    TR0 = 1;                                    // 定时器 0 运行
    CANInit();                                  // 初始化 CAN 控制器
    EA = 1;                                     // 打开总中断

    CAN_ID = 0x0345;
    TX_BUF[0] = 0x11;
    TX_BUF[1] = 0x22;
    TX_BUF[2] = 0x33;
    TX_BUF[3] = 0x44;
    TX_BUF[4] = 0x55;
    TX_BUF[5] = 0x66;
    TX_BUF[6] = 0x77;
    TX_BUF[7] = 0x88;

    while(1)
    {
        if(B_1ms)                               //1ms 定时时间到
        {
            B_1ms = 0;
            if(++msecond >= 1000)               //1 秒定时时间到
            {
                msecond = 0;
                sr = CanReadReg(SR);
                if(sr & 0x01)                   //判断是否有总线关闭状态
                {
                    CANAR = MR;
                    CANDR &= ~0x04;             //清除 Reset Mode, 从总线关闭状态退出
                }
                else
                {
                    CanSendMsg(CAN_ID,TX_BUF);
                }
            }
        }

        if(B_CanRead)
        {
            B_CanRead = 0;

            CAN_ID = CanReadMsg(RX_BUF); //接收 CAN 总线数据
            CanSendMsg(CAN_ID+1,RX_BUF); //发送 CAN 总线数据
        }
    }
}
```

```
/************************Timer0 1ms 中断函数 ************************/
void timer0 (void) interrupt 1
{
    B_1ms = 1;              //1ms 标志
}

// 函数: u8 CanReadReg(u8 addr)为 CAN 功能寄存器读取函数
// 参数: CAN 功能寄存器地址,返回: CAN 功能寄存器数据
u8 CanReadReg(u8 addr)
{
    u8 dat;
    CANAR = addr;
    dat = CANDR;
    return dat;
}

// 函数: void CanWriteReg(u8 addr, u8 dat)为 CAN 功能寄存器配置函数
// 参数: CAN 功能寄存器地址, CAN 功能寄存器数据。返回: none.
void CanWriteReg(u8 addr, u8 dat)
{
    CANAR = addr;
    CANDR = dat;
}

//=========================================================
// 函数 void CanReadFifo(u8 *pdat)读取 CAN 缓冲区数据函数
// 参数: *pdat: 存放 CAN 缓冲区数据。返回: none.
void CanReadFifo(u8 *pdat)
{
    pdat[0]  = CanReadReg(RX_BUF0);
    pdat[1]  = CanReadReg(RX_BUF1);
    pdat[2]  = CanReadReg(RX_BUF2);
    pdat[3]  = CanReadReg(RX_BUF3);

    pdat[4]  = CanReadReg(RX_BUF0);
    pdat[5]  = CanReadReg(RX_BUF1);
    pdat[6]  = CanReadReg(RX_BUF2);
    pdat[7]  = CanReadReg(RX_BUF3);

    pdat[8]  = CanReadReg(RX_BUF0);
    pdat[9]  = CanReadReg(RX_BUF1);
    pdat[10] = CanReadReg(RX_BUF2);
    pdat[11] = CanReadReg(RX_BUF3);
```

```
    pdat[12]  = CanReadReg(RX_BUF0);
    pdat[13]  = CanReadReg(RX_BUF1);
    pdat[14]  = CanReadReg(RX_BUF2);
    pdat[15]  = CanReadReg(RX_BUF3);
}

//=============================================================
// 函数:u16 CanReadMsg(u8 *pdat)为 CAN 接收数据函数
// 参数: *pdat: 接收数据缓冲区。返回: CAN ID.
u16 CanReadMsg(u8 *pdat)
{
    u8 i;
    u16 CanID;
    u8 buffer[16];

    CanReadFifo(buffer);
    CanID = ((buffer[1] << 8) + buffer[2]) >> 5;
    for(i=0;i<8;i++)
    {
        pdat[i] = buffer[i+3];
    }
    return CanID;
}

//=============================================================
// 函数: void CanSendMsg(u16 canid, u8 *pdat)为 CAN 发送数据函数
// 参数: canid: CAN ID; *pdat: 发送数据缓冲区.。返回: none.
void CanSendMsg(u16 canid, u8 *pdat)
{
    u16 CanID;
    CanID = canid << 5;
/*bit7: 标准帧(0)/扩展帧(1), bit6: 数据帧(0)/远程帧(1), bit3~bit0: 数据长度(DLC) */
    CanWriteReg(TX_BUF0,0x08);
    CanWriteReg(TX_BUF1,(u8)(CanID>>8));
    CanWriteReg(TX_BUF2,(u8)CanID);
    CanWriteReg(TX_BUF3,pdat[0]);

    CanWriteReg(TX_BUF0,pdat[1]);
    CanWriteReg(TX_BUF1,pdat[2]);
    CanWriteReg(TX_BUF2,pdat[3]);
    CanWriteReg(TX_BUF3,pdat[4]);

    CanWriteReg(TX_BUF0,pdat[5]);
    CanWriteReg(TX_BUF1,pdat[6]);
```

```
        CanWriteReg(TX_BUF2,pdat[7]);

        CanWriteReg(TX_BUF3,0x00);
        CanWriteReg(CMR ,0x04);          //发起一次帧传输
    }

    //-------------------------------------------------------------
    // 函数:void CANSetBaudrate()为 CAN 总线波特率设置函数
    // 参数: none。返回: none.
    void CANSetBaudrate()
    {
        CanWriteReg(BTR0,(SJW << 6) + BRP);
        CanWriteReg(BTR1,(SAM << 7) + (TSG2 << 4) + TSG1);
    }

    //-------------------------------------------------------------
    // 函数 void CANInit()为 CAN 初始化函数
    // 参数: none。返回: none
    void CANInit()
    {
        CANEN = 1;                       //CAN1 模块使能
        CANSEL = 0;                      //选择 CAN1 模块
    /***端口切换(CAN_Rx,CAN_Tx) 0x00:P0.0,P0.1  0x10:P5.0,P5.1  0x20:P4.2,P4.5
0x30:P7.0,P7.1 *****/
        P_SW1 = (P_SW1 & ~(3<<4)) | (0<<4);
    //CAN2EN = 1;                      //CAN2 模块使能
    //CANSEL = 1;                      //选择 CAN2 模块
    /****端口切换(CAN_Rx,CAN_Tx) 0x00:P0.2,P0.3  0x01:P5.2,P5.3  0x02:P4.6,P4.7
  0x03:P7.2,P7.3 ***/
    //P_SW3 = (P_SW3 & ~(3)) | (0);

        CanWriteReg(MR ,0x04);           //使能 Reset Mode
        CANSetBaudrate();                //设置波特率

    //使用单滤波过滤器,只接收 ID=0x07fe 的报文
    //CanWriteReg(ACR0,0xff);          //总线验收代码寄存器
    //CanWriteReg(ACR1,0xc0);
    //CanWriteReg(ACR2,0x00);
    //CanWriteReg(ACR3,0x00);
    //CanWriteReg(AMR0,0x00);          //总线验收屏蔽寄存器
    //CanWriteReg(AMR1,0x0F);
    //CanWriteReg(AMR2,0xFF);
```

```
//CanWriteReg(AMR3,0xFF);

  CanWriteReg(ACR0,0x00);          //总线验收代码寄存器
  CanWriteReg(ACR1,0x00);
  CanWriteReg(ACR2,0x00);
  CanWriteReg(ACR3,0x00);
  CanWriteReg(AMR0,0xFF);          //总线验收屏蔽寄存器
  CanWriteReg(AMR1,0xFF);
  CanWriteReg(AMR2,0xFF);
  CanWriteReg(AMR3,0xFF);

  CanWriteReg(IMR ,0xff);          //中断寄存器
  CanWriteReg(ISR ,0xff);          //清中断标志
/***退出 Reset Mode, 采用单滤波设置(设置过滤器后注意选择滤波模式) ***/

    CanWriteReg(MR ,0x01);
    CANICR = 0x02;                 //CAN 中断使能
}

//=========================================================
// 函数: void CANBUS_Interrupt(void) interrupt CAN_VECTOR 为 CAN 总线中断函数
// 参数: none。返回: none
void CANBUS_Interrupt(void) interrupt CAN1_VECTOR
{
    u8 isr;
    u8 arTemp;
/***CANAR 现场保存,避免主循环里写完 CANAR 后产生中断,在中断里修改了 CANAR 内容 ****/
    arTemp = CANAR;
    isr = CanReadReg(ISR);
    if((isr & 0x04) == 0x04)       //TI
    {
        CANAR = ISR;
        CANDR |= 0x04;             //CLR FLAG
    }
    if((isr & 0x08) == 0x08)       //RI
    {
        CANAR = ISR;
        CANDR |= 0x08;             //CLR FLAG
        B_CanRead = 1;
    }

    if((isr & 0x40) == 0x40)       //ALI
    {
        CANAR = ISR;
```

```
        CANDR |= 0x40;          //CLR FLAG
    }

    if((isr & 0x20) == 0x20)    //EWI
    {
        CANAR = ISR;
        CANDR |= 0x20;          //CLR FLAG
    }

    if((isr & 0x10) == 0x10)    //EPI
    {
        CANAR = ISR;
        CANDR |= 0x10;          //CLR FLAG
    }

    if((isr & 0x02) == 0x02)    //BEI
    {
        CANAR = ISR;
        CANDR |= 0x02;          //CLR FLAG
    }

    if((isr & 0x01) == 0x01)    //DOI
    {
        CANAR = ISR;
        CANDR |= 0x01;          //CLR FLAG
    }

        CANAR = arTemp;         //CANAR 现场恢复
    }
```

（6）按 Ctrl+S 按键，保存设计代码。