

# Turbo Debugger 使用简介

英创信息技术有限公司

2005 年 3 月

## 1. 概述

美国 Borland 公司的经典产品 BC3.1 中包含了一个重要的应用程序调试工具 Turbo Debugger, 通常被简称为 TD。所谓应用程序调试工具, 就是让应用程序在调试工具软件 TD 的监控下运行, 从而让程序设计者能即时了解在应用程序在运行过程中的状态或结果, 这些结果或状态包括 CPU 的寄存器、存储器、程序变量、IO 端口状态。TD 可支持 C/C++、汇编程序的源码调试, 对 C 语言程序, 可同时观察到用户编写的 C 源码以及编译生成的汇编代码。当然要实现对程序源码的调试, 应用程序在编译连接时, 必需设置调试信息选项。TD 既可调试本机应用程序, 也可运行于主机, 通过串口对运行于目标机上的应用程序进行调试, 即所谓的远程调试模式。英创公司的嵌入式网络模块均可支持 TD 远程调试模式, 为客户快速开发其应用程序提供了强有力的工具。

本手册将针对 TD 在英创嵌入式网络模块调试使用情况, 对其常用的调试功能进行简要简介, 以帮助用户快速掌握 TD 这一强有力的调试工具的使用, 提高应用程序的开发效率。建议用户在工作盘上以网络模块名建立根目录, 如 NetBox2、ETR232i 等等, 再根据不同的应用或测试建立子目录, 以便于程序代码管理。本手册采用 NetBox2 为例, 但所介绍的 TD 使用方法是通用的。本手册所引用的示例 Step1 和 Step2, 用户可从附带的开发光盘的“使用必读”目录中找到。

## 2. 启动 TD

调试工具软件 TD.EXE 是 BC3.1 集成开发软件的实用工具之一, 存放在 BC\BIN 目录下。本文认为用户已把 BC\BIN 目录加入到了 AUTOEXEC.BAT 中的路径 (PATH) 定义中, 因此用户可在任意工作目录下启动 TD。

在启动 TD 前, 应确保开发主机 PC 的串口已与英创嵌入式网络模块的调试串口正确连接, 且网络模块处于调试模式。

用户通常先打开 MSDOS 窗口, 并转到相关应用程序或测试程序所在目录, 如 D:\NetBox2\Step1>, 然后执行操作: “**td -rp hello**”, 其界面如图 1 所示。在上述命令行操

作中，-rp 为 TD 的远程调试选项，且默认使用 PC 的 COM1 口，若实际使用的是 COM2 口，则远程调试选项应为-rp2。“hello”为编译成功的 EXE 文件名，在采用 PRJ 工程文件进行编译时，则为工程文件名。注意：在 td 的启动命令行中不要带应用程序扩展名。

```
D:\NetBox2\Step1>dir

Volume in drive D is HARDWARE
Volume Serial Number is 16E7-3A67
Directory of D:\NetBox2\Step1

.                <DIR>                03-22-05   13:36  .
..               <DIR>                03-22-05   13:36  ..
HELLO            CPP                347      08-26-02   13:39  HELLO.CPP
HELLO            EXE            13,864    03-22-05   12:51  HELLO.EXE
USER             BAT                 5        03-22-05   13:44  USER.BAT
                3 file(s)                14,216 bytes
                2 dir(s)                4,615.79 MB free

D:\NetBox2\Step1>td -rp hello_
```

图1 启动 TD

TD 启动后，首先检查在目标板（即英创嵌入式网络模块，下同）上是否有 Hello.exe，若存在则比较文件的时间标志，以决定是否需要下载被调试程序到目标板。若需要重新下载程序，TD 将在屏幕上弹出如下窗口信息：

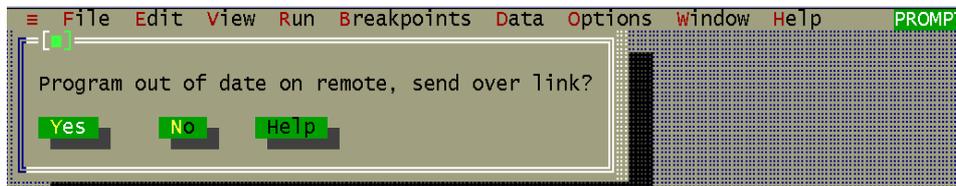


图2 程序下载提示

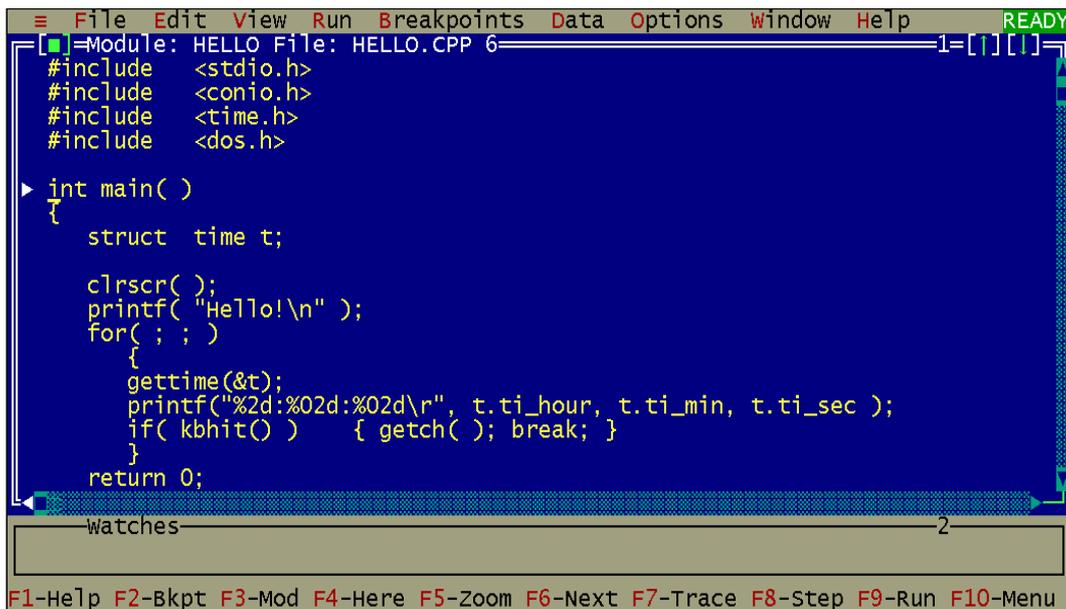


图3 TD 调试主界面

直接按<ENTER>键确认下载程序。这时 TD 通过串口把程序下载到目标板上，下载速度根据不同的电子盘存储介质而有所不同，通常可从目标板的 LCD 上看到下载过程。程序成功下载后，TD 将把程序装载到目标板内存区，并在屏幕上显示源码主界面（图 3）。至此，TD 的启动已正确完成，用户已可以调试运行程序了。

### 3. TD 的常用窗口及快捷键

TD 是一个多窗口调试工具，图 3 是其调试主界面，其中的窗口 1 为源代码，窗口 2 可以随着程序运行同步显示相关的信息，用户可设定显示内容。窗口的上部为下拉菜单，每个菜单名的第一个字母为红色，为关键字，打开下拉菜单操作为 **Alt+<关键字>**。关闭操作则总是按<ESC>键。主界面的下部为常用的调试功能快捷键，也是以红色标注，第 4 节将具体介绍它们的使用。本节主要介绍下拉菜单的相关功能。

最常用的下拉菜单是 **View**，按 **Alt+V** 弹出 **View** 菜单如下：

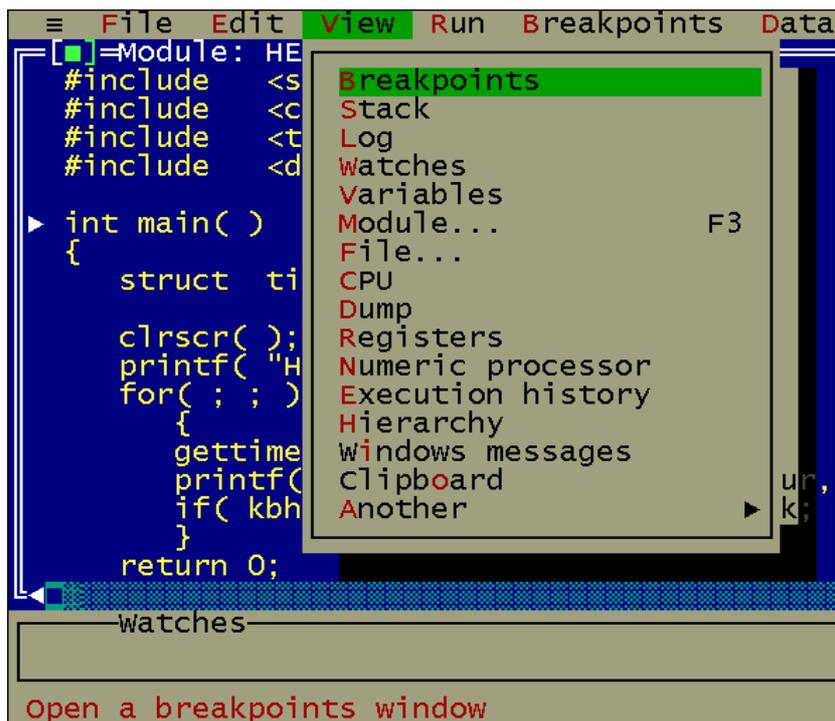


图 4 TD 的 View 下拉菜单

**View** 菜单主要用于选择各种显示窗口，以观察所需信息。菜单中每一选择行中的红色字母为快捷键。最常用的为 **CPU** 和 **Dump** 窗口，**CPU** 窗口用于观察对应的汇编指令、CPU 的寄存器内容，操作 IO 端口（将在第 6 节介绍），而 **Dump** 窗口主要用于查看存储器中的数据。直接按红色的快捷键字母，就可打开相应窗口，如按<C>键打开 **CPU** 窗口如图 4 所示。

CPU 窗口由若干子窗口组成，包括汇编、CPU Registers、CPU Flag、堆栈和 Dump 窗口。通过<Tab>键可循环激活各个子窗口，特别的在汇编窗口还可进行 IO 端口的读写。

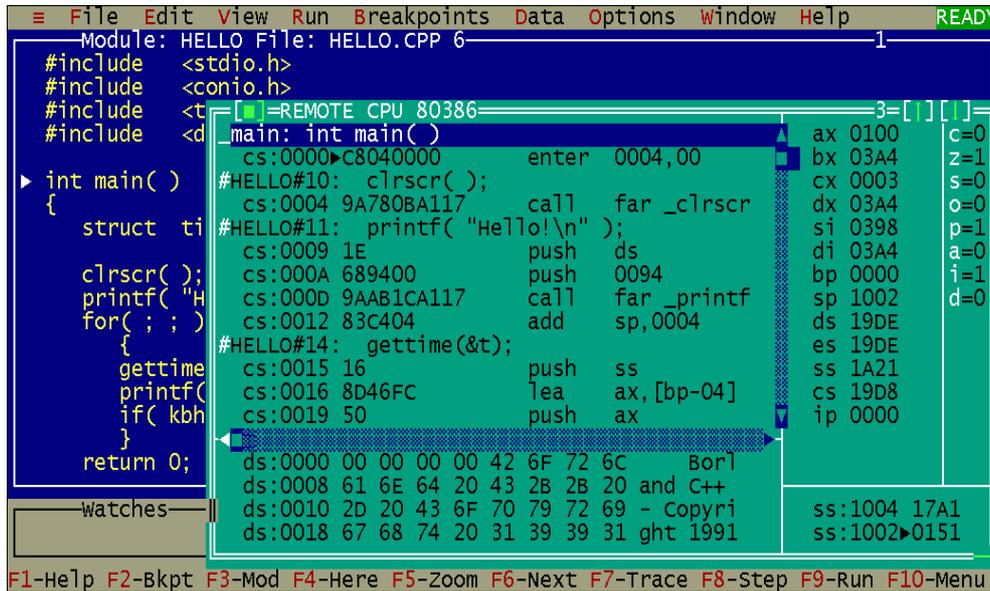


图 5 TD 的 CPU 窗口

对 TD 的多窗口，通常用 **Alt+<数字>**来直接激活所关心的窗口，如图 5 中 **Alt+1** 激活源程序窗口、**Alt+2** 激活 watches 窗口、**Alt+3** 激活 CPU 窗口。按 **Alt+F3** 则关闭当前激活窗口。用户可反复使用这些快捷键来熟悉 TD 窗口的界面操作。

另一个常用功能是包括在 **File** 菜单中的查看系统剩余空间，如图 5 所示：

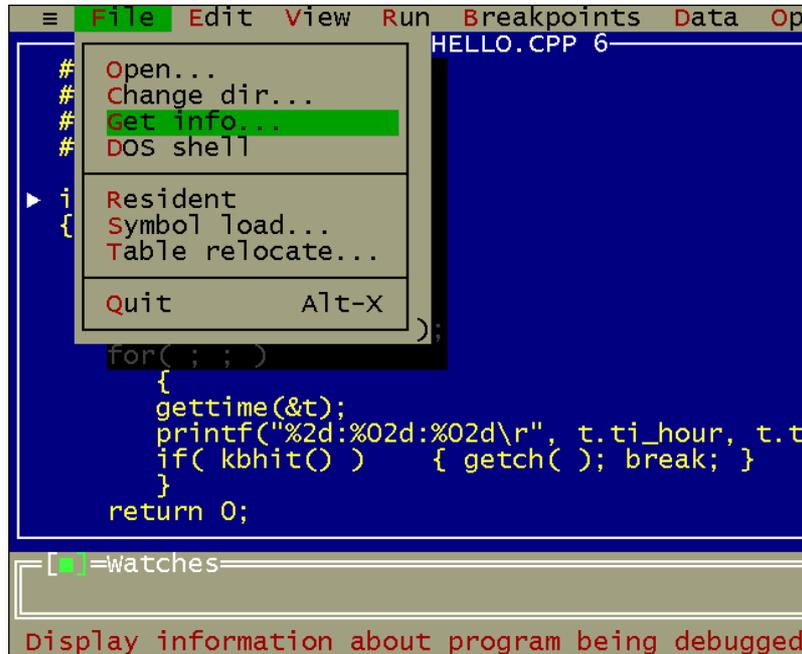


图 6 TD 的 File 下来菜单

选择“**Get info**”，TD 弹出系统信息窗口，其中重要信息包括程序占用存储器空间（目

前为 15k 字节) 以及系统剩余空间 (目前为 496k 字节)。一般较大的应用程序会在 300k 字节的水平。英创各种型号的嵌入式网络模块, 分别针对不同的应用, 在运行空间方面都留有充分的余地, 可满足绝大多数代码的运行需求。



图 7 系统信息弹出窗口

TD 的退出与 BC 是一样的, 按 **Alt+X** 即可。

#### 4. 应用程序的调试运行

本节内容涉及 C 语言的一些基本概念, 要求读者具有相应的背景知识。

一般说来, 按照第 2 节方法启动 TD, 应用程序已加载到内存可以运行, 图 3 左上角的 **READY** 是系统准备好的标志。画面上的白色三角标志为当前程序执行的位置。

按快捷键 **F9**, 应用程序进入全速运行, 这时闪烁的光标消失, 用户可在目标机的 LCD 上看到 **printf(..)** 显示的结果。在程序运行的任意时候, 都可按 **Ctrl+<Break>** 暂停程序的运行, 这时 TD 通常会弹出 CPU 窗口, 以显示程序当前执行的位置, 与图 5 显示类似。

用户可按 **Ctrl+F2** 来重载程序, 以便反复执行同一程序。

按快捷键 **F8**, 应用程序按单步执行, 注意白色三角标志的移动以及 LCD 上的显示:

- 按 **F8** 执行 **clrscr()**, LCD 将清屏;
- 按 **F8** 执行 **printf("Hello\n")**, LCD 将显示 Hello 字样;

断点是在程序调试中常用的方法, 在 TD 中, 断点的设置非常简单: (1) 把光标移到需要设置断点的所在行; (2) 按快捷键 **F2** 即设置断点, 该行变为红色, 如图 8 所示; (3) 若在断点所在行再按快捷键 **F2** 则取消断点。

```

int main( )
{
    struct  time t;

    clrscr( );
    printf( "Hello!\n" );
    for( ; ; )
    {
        gettime(&t);
        printf("%2d:%02d:%02d\r", t.ti_hour, t.ti_min, t.ti_sec );
        if( kbhit( ) ) { getch( ); break; }
    }
    return 0;
}

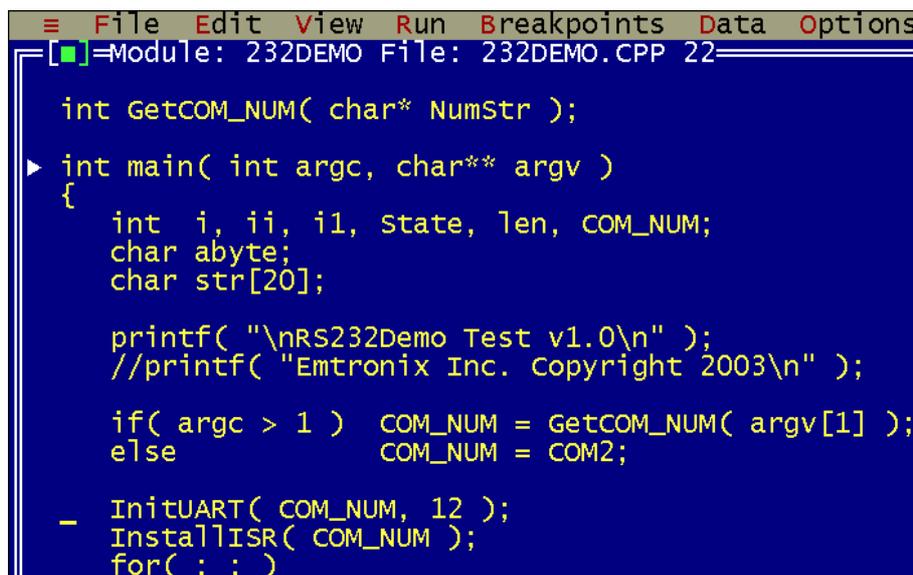
```

图 8 TD 的断点设置

另外一种常用的运行方式是运行到光标所在位置，具体操作为把光标移至希望暂停的所在行，按快捷键 **F4**，程序将运行到光标所在行而暂停。注意在使用快捷键 **F4** 时，所选光标位置应当是程序在一定条件下或无条件定会执行到的位置。

在实际的程序调试中，应用程序通常都是由多个 CPP 代码模块，通过 PRJ 工程文件联编而成的。在用 TD 调试时就需要打开不同的代码窗口，我们以资料光盘中 STEP2 的例子来说明相关的操作。

参照第 2 节的介绍，转到目录 D:\NetBox2\Step2 下，执行“**td -rp 232demo**”，程序自动下载后，屏幕显示如图 9 所示：



```

File Edit View Run Breakpoints Data Options
Module: 232DEMO File: 232DEMO.CPP 22
int GetCOM_NUM( char* NumStr );
▶ int main( int argc, char** argv )
{
    int i, ii, i1, State, len, COM_NUM;
    char abyte;
    char str[20];

    printf( "\nRS232Demo Test v1.0\n" );
    //printf( "Emtronix Inc. Copyright 2003\n" );

    if( argc > 1 ) COM_NUM = GetCOM_NUM( argv[1] );
    else          COM_NUM = COM2;

    - InitUART( COM_NUM, 12 );
    - InstallISR( COM_NUM );
    for( ; ; )

```

图 9 RS232 演示程序主界面

232DEMO.EXE 是由底层驱动模块 RS232X4.CPP 和上层主模块 232DEMO.CPP 构成，用户可按快捷键 **F3** 打开模块窗口进行选择，如图 10 所示。请用上下箭头键选择模块。

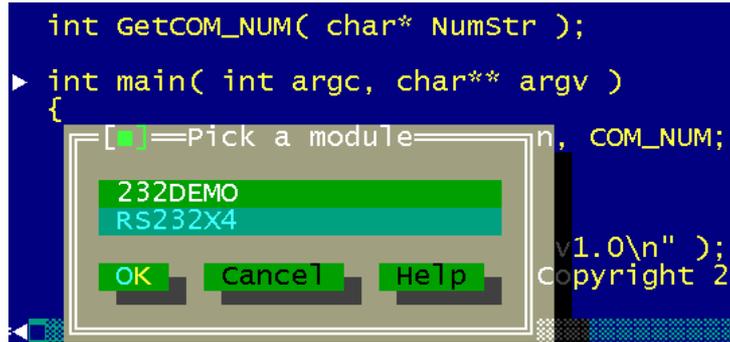


图 10 代码模块选择窗口

若选择 RS232X4，则主界面将显示 RS232X4.CPP，如图 11 所示：

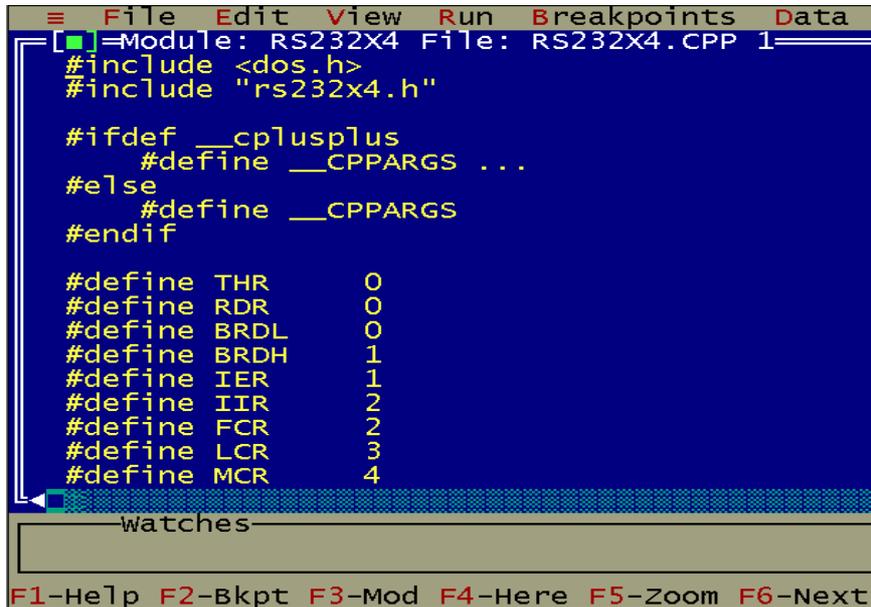


图 11 打开子模块的主界面窗口

注意源代码窗口只能打开一个。

在 C/C++ 程序中，存在着大量的子函数调用情形，如图 9 中的串口初始化函数 InitUART，中断安装子函数 InstallISR 等等，以 InitUART 为例，进入子函数进行调试的步骤为：

- 把光标移至 InitUART 子函数所在行，按快捷键 **F4** 运行到此行；

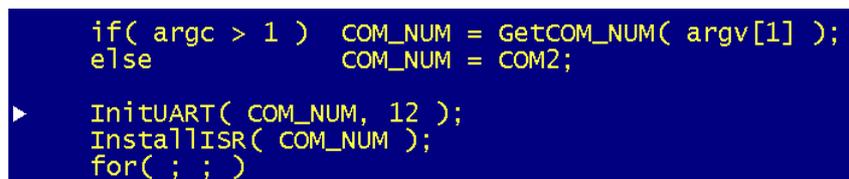


图 12 按 F4 运行到光标所在处

- 按快捷键 **F7** 进入 (TRACE) 子函数，如图 13 所示。

```

// BaudIdx = 12: 9600bps
// BaudIdx = ...
int InitUART( int ComIdx, int BaudIdx, int Parity )
{
    int i1;
    int tport;
    char Mode;

    switch( Parity )
    {
        case 1:
            Mode = COM_CHR8 | COM_STOP1 | COM_ODDPARITY;
            break;
    }
}
    
```

图 13 按 F7 进入子函数

至此我们已介绍程序运行中的常用快捷键，小节如下表：

快捷键	英文提示	执行功能
<b>F2</b>	Bkpt	断点设置或取消
<b>F3</b>	Mod	打开代码模块选择窗口
<b>F4</b>	Here	运行到光标所在处
<b>F7</b>	Trace	跟踪进入子函数
<b>F8</b>	Step	单步执行程序
<b>F9</b>	Run	全速运行程序

### 5. 观察变量与状态

本节内容涉及 C 语言变量类型和数据结构的概念，要求读者具有相应的背景知识。

在程序调试中，常常需要运行到程序的某一个地方暂停下来观察程序中的相关变量。这时一般需要在相关的代码行设置断点，然后按 **F9** 运行程序到所设的断点处，注意此时 CPU 还没有执行红色的断点代码。我们以 STEP1 目录中的 `hello.exe` 来进行说明，设置断点如图 8 所示，按 **F9** 运行到断点处，此时通常会做的事包括：

- (1) 观察相关的变量，如此时的结构变量 `t`，把光标移至需观察的变量下面如图 14 所示，按快捷键 **Ctrl+I**，TD 将弹出窗口显示结构变量 `t` 的全部信息如图 15 所示。

```

gettime(&t);
printf("%2d:%02
if( kbhit() )
    
```

图 14 光标移至变量 t 的下面

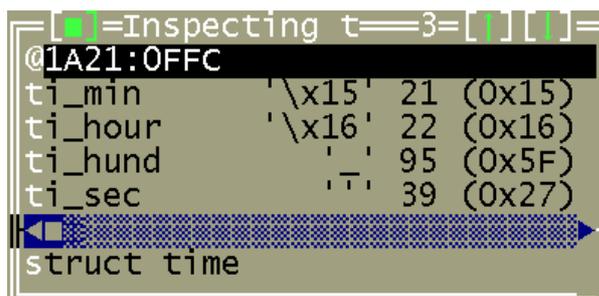


图 15 结构变量 t 的当前值

(2) 快捷键 **F8** 单步执行断点行代码，并在 LCD 上观察输出结果，还可与 TD 看到的结果进行比较。注意用户在实际操作时，变量 **t** 的当前值与图 15 会不一样。

(3) 把某变量放到观察（watches）窗口中，以便程序运行时同步观察该变量。这里把 **t.ti\_sec** 放入观察窗口：

- 把光标移至 **t.ti\_sec** 下：**t.ti\_sec** ); (图 16)
- 按<Insert>键，标注变量 **t.ti\_sec** ); (图 17)
- 按左右箭头键，标注整个变量：**t.ti\_sec** (图 18)
- 按快捷键 **Ctrl+F7** 弹出变量选择确认窗口：

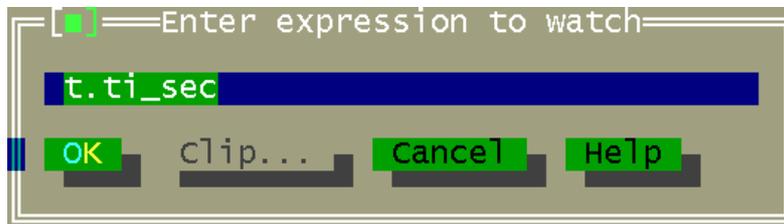


图 19 按 Ctrl+F7 弹出变量输入窗口

- 确认后，变量 **t.ti\_sec** 出现在观察窗口：

```

int main( )
{
    struct time t;

    clrscr( );
    printf( "Hello!\n" );
    for( ; ; )
    {
        gettime(&t);
        printf("%2d:%02d:%02d\r", t.ti_hour, t.ti_min, t.ti_sec );
        if( kbhit( ) ) { getch( ); break; }
    }
    return 0;
}

```

watches  
 t.ti\_sec                      unsigned char '\x1c' 28 (0x1c)

F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

图 20 变量放入观察窗口

此时用户可用 **F8** 单步执行循环中的代码，并观察 **t.ti\_sec** 的变化以及 LCD 屏幕显示的更新。

TD 观察的变量，即可以是全局变量（globe）也可以是局部变量（local）。全局变量可在程序运行的全过程进行观察，而局部变量只能是该变量所在的函数正处于运行之中才是有效的。

## 6. 利用 TD 进行硬件接口调试

本节内容涉及 x86 CPU 体系结构及汇编语言，要求读者具有相应的背景知识。

TD 除了主要用于应用软件程序的调试外，还可以用于调试硬件接口电路，其基本原理是在 TD 控制下让 CPU 循环运行一些简单的汇编指令，在扩展总线上产生特定的准周期性总线读写时序，然后用示波器观察相关的总线信号，如 CS1#、WE#、RD#、SA[0..4]、D[0..7] 等，从而来判断接口电路工作的正确性。由于 TD 可支持直接写入汇编，因此无需事先编写任何程序，可直接启动进行操作。举例说明如下。

在当前工作目录下直接执行“td -rp”，TD 将弹出 CPU 窗口：

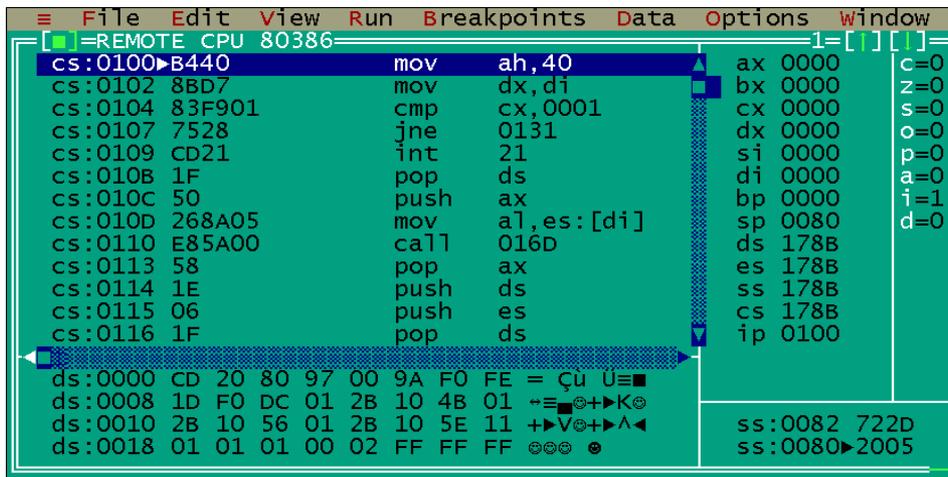


图 21 CPU 窗口作为主界面

为了检查精简 ISA 总线上扩展的接口电路的正确性，可以直接构造一个 IO 端口输出操作循环，从而在总线上生成周期性波形以便于示波器观察。这里 CS1#的片选地址区域为 300h 至 31fh，我们以端口 300h 为例，在汇编子窗口上直接按键输入汇编指令“out dx, al”，这时 TD 会自动弹出指令输入窗口，以让你输入完整的汇编指令：

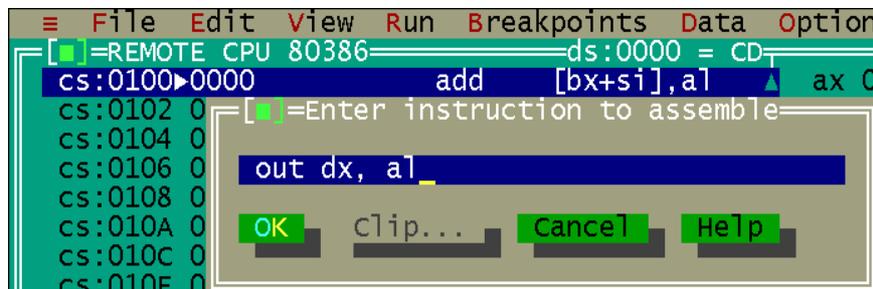


图 22 直接输入汇编指令

紧跟第一条汇编指令，再输入后续 2 条指令“not al”和“jmp 100h”，注意这里的“100h”是 16 进制的地址偏移量。

```

[ ] =REMOTE CPU 80386
cs:0100▶EE          out    dx,a1
cs:0101 F6D0        not    a1
cs:0103 EBF8        jmp    0100
cs:0105 90          nop

```

图 23 3 条汇编指令构成的循环

按<Tab>跳转到寄存器子窗口，光标移到 DX 栏，按键直接输入 300h，TD 也会自动弹输入窗口（见图 24），以让你输入完整的参数。

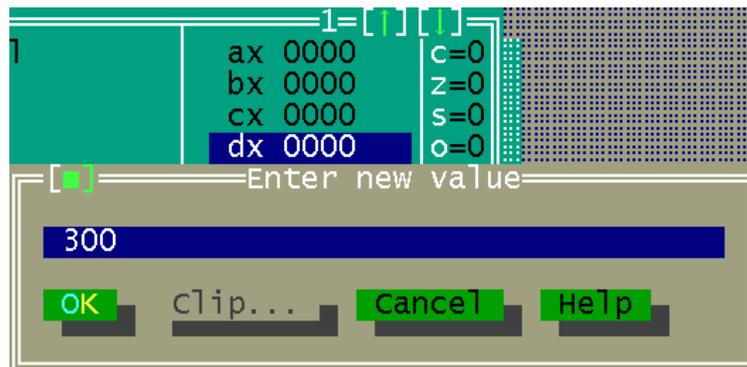


图 24 设置 IO 口地址到 DX

再跳转回汇编子窗口，先按 **F8** 单步执行这个循环，用户可以看到寄存器 AL 的变化如图 25 所示，并确认整个循环工作正常。

```

[ ] =REMOTE CPU 80386
cs:0100 EE          out    dx,a1    ax 00FF
cs:0101 F6D0        not    a1       bx 0000
cs:0103▶EBF8        jmp    0100 ↑   cx 0000
cs:0105 90          nop          dx 0300

```

图 25 单步执行汇编指令

一旦确认无误，就可按 **F9** 全速运行这段代码，同时可以用示波器来观察总线时序。通常是选用 **CS1#** 作为触发通道（下降沿触发），而用示波器的另一通道去观察其它的相关信号。

上述方法也可在加载了 C 语言环境下使用，只是需要在输入常数时注意书写的格式：当没有加载 C 程序时，TD 默认的常数是 16 进制的，当 A-F 开始时，需要在前面冠以 0，建议总是以 h 结尾以表明是 16 进制数据，如 0aah；当加载了 C 程序时，TD 则遵循 C 语言的规则，即 16 进制常数以 0x 开始，如 0xaa。

通过 TD 还可以读写 IO 端口寄存器，这在许多硬件调试中是很有用的手段。首先如图 21 所示进入 TD，在汇编子窗口激活条件下，按 **Ctrl+I** 弹出 IO 操作选择菜单如图 26 所示。菜单中的 4 个选项为：从 IO 端口读一个字节 (**In byte**)；写一个字节到 IO 端口 (**Out byte**)；从 IO 端口读一个 16-bit 字 (**Read word**)；写一个字节到 IO 端口 (**Write word**)。现以 NetBox-II 为例，读写 COM2 口的线路控制寄存器 LCR (baseport+3)。

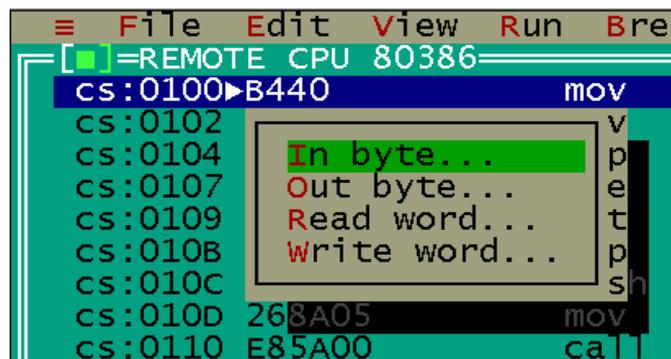


图 26 Ctrl+I 弹出 IO 操作窗口

当选择字节写时，TD 弹出窗口图 27 所示窗口，按提示格式写入 LCR 的口地址（2f8h+3）和写入值 0aah，回车确认；

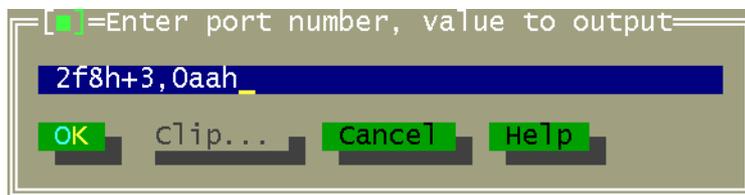


图 27 在 TD 中执行 outp (port, value)

当选择读字节时，TD 弹出窗口图 28 所示窗口，按提示格式写入 LCR 的口地址（2f8h+3）

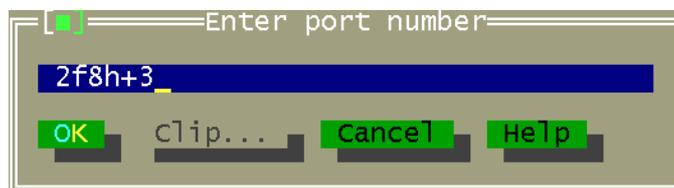


图 28 在 TD 中执行 inp (port)

回车确认，此时 TD 弹出窗口显示读取的当前值（图 29），按<ESC>关闭弹出窗口。

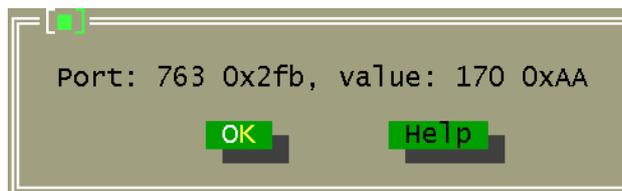


图 29 在 TD 中执行 inp (port) 的返回值

用户可用完全相同的方法 16-bit 的 IO 端口。

## 7. 结束语

包括 TD 在内的 BC31 是当时一群世界顶尖高手的杰作。我们希望通过本文的介绍以及在英创嵌入式网络模块上的实战操作，用户能感受体验 TD 的快捷与高效、分享大师们的智慧，因为智慧比知识更有力量。