Application Note

TKS-BU 系列仿真器快速入门

AN05220043 V1.00 Date: 2011/02/23 产品应用笔记

类别	内容
关键词	TKS-BU 系列 仿真器 快速入门
摘 要	介绍 TKS-BU 系列仿真器的使用

修订历史

版本	日期	原因
V1.00	2011/02/23	创建文档

目 录

1. TKS-BU系列仿真器的概述

TKS-BU 系列仿真器是 TKScope 旗下的标准 8051 仿真器，在原有老牌产品 TKS-B 系列仿真器的基础上，改进了通讯方式，采用 USB2.0 通讯，速度更快，更加方便用户使用。同时，BU 系列突破传统仿真技术缺陷，采用了先进的 HOOKS 仿真技术，不占用用户资源，仿真更准确，性能更稳定。

TKS-BU仿真器实物图如图 1.1所示。

图 1.1　TKS-BU 仿真器实物图

硬件上采用 NXP 公司 MCU 设计部门的经验，具备高度运行稳定性/芯片兼容性。运行频率突破 HOOKS 技术的极限，达到前所未有的 32MHz。低电压仿真方面性能卓越，可以稳定运行在 2.0V 以下。

软件上支持 TKStudio/Keil 中英文双平台，并首次在 Keil 公司的 uVsion2/uVision3 上稳定实现 64K 超大容量 Trace 接口；4×64K 代码数据覆盖；加彩运行轨迹显示；4×64K 运行断点；超精密运行时间显示等多项激动人心的超级仿真功能。

TKS-BU 系列仿真器在 TKStudio 环境中表现更佳，并能实现数据覆盖分析/加彩显示/高级语法分析等更多附加功能，其中 TKSsitant 智能导向技术将简化用户设计难度。

图 1.2　TKS-BU 系列仿真器支持软件平台

1.1 TKS-BU系列仿真器的特点

- 真正无缝嵌接 Keil/TKStudio 中英文双平台，实现多种高级调试功能。
- 采用先进的 HOOKS 仿真技术，可靠仿真标准 P0/P2 口特性。
- 采用超高稳定 IO/总线设计，仿真性能异常稳定。
- 内置 PLL 频率发生器，可以自动产生用户设置的 20K~100MHz 运行时钟，时钟精确度为 0.001%。
- 高速 USB 通讯，速度快，使用方便。
- 仿真绝对不占用任何用户资源，包括堆栈/内部 RAM/SFR 等。
- 业界领先的超低电压仿真，稳定仿真至 2.0V 以下。
- 超高速仿真设计，实时运行频率 0~32MHz@12Clock/0~26MHz@6Clock。
- 超精密运行时间/周期显示，0~32MHz 范围内最高时间精度为 1clock。
- 内部代码空间/数据空间均支持 1byte 精度的 ReMap 功能，方便用户仿真不同资源的芯片。
- 支持运行仿真器外部/外部 0~64K 的 ROM 和 RAM。
- 支持 64K 全范围的冯诺曼 Von Neumann 结构。
- 64K 的仿真器内部程序存储器/64K 的数据存储器。
- 64K 超大容量实时 Trace 功能，协助用户分析程序运行轨迹。
- 4×64K 用户实时断点，方便用户特殊仿真要求。
- 64K 全地址范围内的代码覆盖分析，加彩运行轨迹显示。
- 64K 全地址范围内的代码读取/数据读写覆盖显示。
- 采用高速信号跟踪技术，真正支持 ALE 静态关闭或动态关闭，不限制对 ALE 信号的非常规切换。
- 自动感知 6Clock/12Clock 时钟，并支持动态切换和静态切换，不限制用户对时钟信号的非常规切换。
- 所有仿真器入出口线 100%保护，避免使用中误操作引起仿真器的损坏，保护用户投资。
- 支持用户目标板的时钟输入和用户目标板的振荡晶体，并进行时钟有效性检查。
- 支持外部复位信号，并在全速运行中有效（可选择为关闭），方便用户调试外部看门狗。
- 丰富的提示信息，自动感知当前仿真任何异常，帮助用户准确了解当前 MCU 运行状态。
- 采用与业界不同的 I/O 口重造技术，再现标准 MCU 的 I/O 特性而无任何限制，用户可以任意在 IO/总线 2 种状态中切换。
- TKSsistan 智能设计导向工具，帮助用户轻松完成设计。
- 单机支持仿真芯片最多，支持全系列标准 8051 芯片仿真（包括最新推出的低电压 V 系列）。

1.2 TKS-BU系列仿真器型号和基本性能

TKS-BU系列仿真器具体型号见表 1.1。

表 1.1 TKS 仿真器 BU 系列型号列表

型号	TKS-52BU	TKS-58BU
仿真技术	HOOKS	HOOKS
仿真程序空间 64K	内外任意配置	内外任意配置
仿真数据空间 64K	内外任意配置	内外任意配置
任意时间发生器	√	√
彩色运行轨迹显示	-	√
4×64K 代码/数据覆盖	√	√
4×64K 任意断点	4×64K	4×64K
冯诺曼结构支持	√	√
64K 实时跟踪支持	-	√
外部电源输入支持	2.0~5.5V	2.0~5.5V
外部复位输入支持	√	√
仿真芯片可更换性	√	√
实时仿真频率	0~32MHz	0~32MHz
支持仿真范围	仿真标准 51	仿真标准 51
6/12CLK 动态切换	√	√
ALE 动态切换	√	√
芯片支持	8xC5x 以下	8xC5x 以下
出厂内置仿真芯片	P87C52X2	P87C52X2

2. 使用TKS-BU系列仿真器

TKS-BU 系列仿真器支持 TKStudio/Keil 中英文双平台，本文主要讲解 TKS-BU 系列仿真器在 Keil 开发环境下的快速入门操作。在使用仿真器之前，用户需要进行以下操作：

- 认识仿真器的硬件结构，正确连接仿真器；
- 正确地安装 Keil 软件和 TKS-BU 系列仿真器的驱动。Keil 软件的安装过程很简单，按照提示信息进行操作即可，在此不做详细描述，下面主要讲解 TKS-BU 系列仿真器的驱动安装过程和方法。

2.1 认识仿真器

仿真器是由两部分组成：仿真器主机和仿真头。仿真器主机顶部俯视图如图 2.1所示。

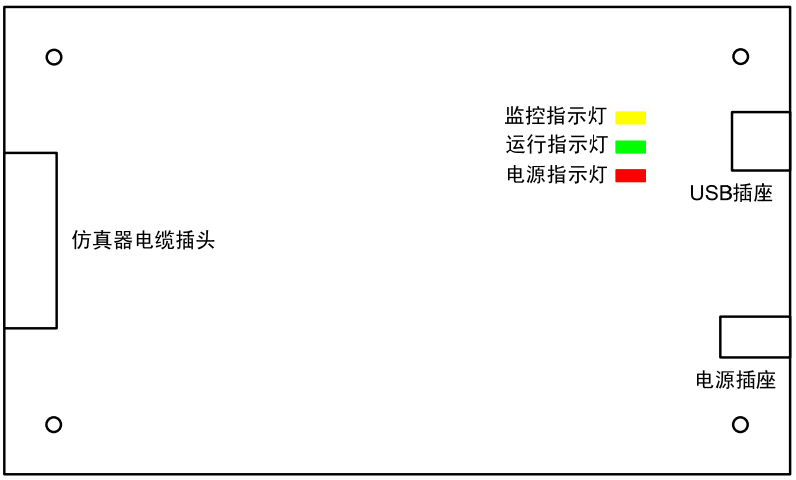

图 2.1 TKS-BU 系列仿真器顶部俯视图

- 监控指示灯： 点亮表示处于监控状态；
- 运行指示灯： 点亮表示进入运行状态；
- 电源指示灯： 点亮表示系统电源正常；
- 仿真电缆插头： 插入仿真电缆连接到仿真头；
- USB 插座： 插入 USB 电缆连接到计算机；
- 电源插座： 输入仿真器主机工作需要的电源，6.3V，内负外正。

用出厂所配的电源适配器给仿真器供电，仿真器的电源指示灯（红灯）呈点亮状态，监控指示灯（黄灯）和运行指示灯（绿灯）会交替闪烁数次，黄灯熄灭，绿灯点亮。此时，仿真器初始化成功，可以正常工作。

2.1.1 仿真头的使用

仿真头是用于连接用户目标板，DIP40 封装。

1. 仿真头

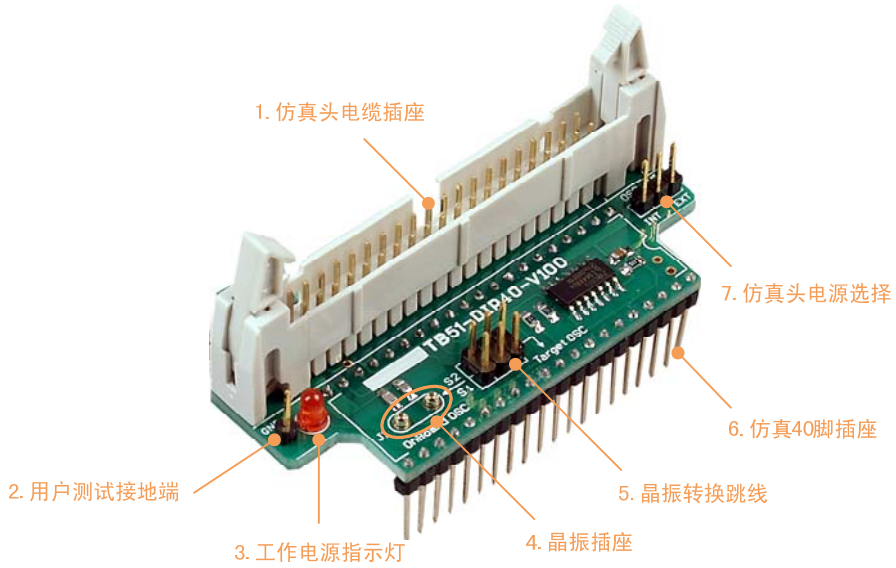

图 2.2 仿真头示意图

1. 仿真头电缆插座

用于插仿真电缆，连接仿真器和仿真头。

2. 用户测试接地端

用户在进行测试时，可以使用该端作为地参考点。

3. 工作电源指示

如果仿真头有电源接入，则该 LED 点亮。

4. 晶振插座

用于插入晶振。

5. 仿真 40 脚插座

用于连接仿真头和用户目标板。

6. 晶振转换跳线

用于转换当前使用的晶振，若使用内部时钟，跳线不起作用。

当该跳线位于右选择（Target OSC）时，选择用户目标板上的晶振；

当该跳线位于左选择（S2 / S1）时，选择仿真头晶振插座的晶振。

用户板上的晶振和仿真头上的晶振可以同时存在，用户板上必须有 10~50P 的补偿电容，并根据用户使用振荡频率而变化。

7. 仿真头电源选择

用于选择仿真头的工作电源。

当向左选择（INT）时，使用仿真器提供的+5V 电源；

当向右选择（EXT）时，使用外部用户板上输入的电源。

当左右都不选择时，仿真头上的电路不工作，具体使用方法如下：

- 若用户使用仿真器内部时钟，可以左右都不选择。由于仿真头上无电源，所以时钟电路不工作，这样可以减少干扰，提高稳定性。
- 若用户使用仿真头上产生的时钟且仿真器使用内部电源，一定向左选择使用仿真器内部电源。
- 若用户使用仿真头上产生的时钟且仿真器使用用户板上提供的电源，在外部电源在 4V~5V 的稳定电压下可以考虑向右选择使用用户板上提供的电源，如果用户板上的电压小于 4V 建议仍然向左选择使用仿真器内部电源。

注意：尽量不要使用用户板板上的电源对仿真头振荡器电路供电。

2. 仿真电缆

仿真电缆一般为 40 芯扁平电缆，要使用标配的电缆线，不能使用硬盘线替代。

3. 仿真头的使用方法和注意事项

- 用户在使用仿真器内部时钟时，仿真头上的时钟电路没有必要使用，因此可将仿真头电源选择的短路跳线取下，这样仿真头上的电路将失去工作电压而不工作。这在仿真一些对干扰敏感的用户系统中是有好处的。

- 如果用户不使用仿真器内部提供的时钟，用户可以选择外部时钟。使用外部时钟用户可以有 3 种选择方案：

  1) 仿真头上晶振产生的时钟

     用户需要在晶振插座位置上插入晶振，晶振转换跳线要左选择（S2 / S1）。

  2) 用户目标板上晶振产生的时钟

     保证用户板上的晶振起振，晶振转换跳线要右选择（Target OSC）。

  3) 用户时钟

     使用用户目标板上提供的时钟（不是由晶振产生的），必须从 XTAL1 接入时钟信号。用户提供的时钟要求有 50%的占空比，如果差异太大在高频时钟仿真时可能达不到规定的最高频率，晶振转换跳线需要向右选择（Target OSC）。

注意：在使用仿真器外部时钟（包括从用户板上直接输入时钟）时都要使用仿真头上的时钟振荡电路，因此仿真头电源选择必须向左或向右选择，不能悬空。

- 在仿真一些用户系统时，如果发生一些很难解释的现象，在某些情况下是由于仿真器和用户目标系统的地线连接不是很理想。用户可以尝试增加另外的地线连接，从

仿真头上的接地端子用户测试接地端上连接另外一根地线到用户系统地上，并且可以尝试用户目标板上不同位置的接地点。

- 如果用户选择了使用外部电源，则仿真芯片工作电源将由用户的目标板提供（电压范围为 2.0V~5.5V），从仿真 40 脚插座的第 40 脚输入。用户提供的电源除了有电压要求外，还要求稳定，不能有电压的瞬间起伏，否则会引起仿真状态的破坏。

2.1.2 TKS-BU系列仿真器的限制

TKS 仿真器 BU 系列不全部支持看门狗的仿真。用户如果需要打开看门狗，只能在程序中运行看门狗启动程序，并且全速运行。如果用户在程序全速运行后打开了看门狗，并又终止程序运行并返回到监控状态，则系统将复位因为看门狗打开后只能作全速运行，返回到仿真器监控状态后看门狗还将继续运作，最终很快溢出并引起复位。

如果在全速运行中用户通过运行程序打开看门狗，并且在看门狗溢出前程序正常喂狗，用户程序将正常的运行；如果用户没能够及时喂狗而发生了溢出，仿真芯片将发生复位。仿真器监控系统能够发现这种现象并复位仿真器的运行时序重新开始运行，在信息输出窗口提示用户。

2.2 驱动安装方法

2.2.1 仿真器驱动安装过程

1. 双击驱动文件【TKScopeSetup_C51】，系统会弹出如图 2.3所示的对话框。

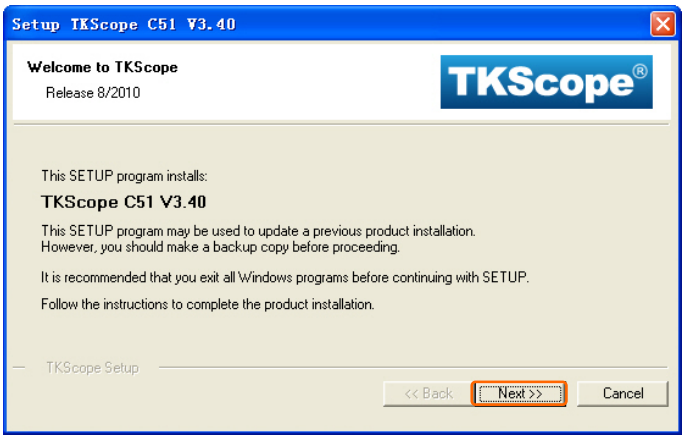

图 2.3 安装驱动提示框

2. 点击图 2.3中【Next】选项，系统会弹出如图 2.4所示的对话框。选中【I agree to…】，然后点击【Next】进行下一步操作。

图 2.4 协议许可提示框

3. 此时，需要指定仿真器驱动程序的安装路径，如图 2.5所示。注意：驱动程序必须安装在Keil软件的安装路径下！

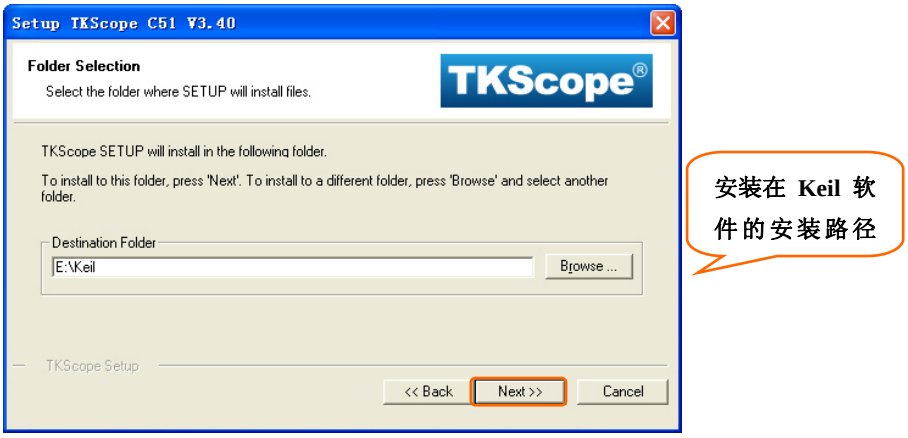

图 2.5　指定驱动安装路径

4. 点击图 2.5中【Next】选项，系统会弹出如图 2.6所示的对话框。用户根据提示填写自己的信息。

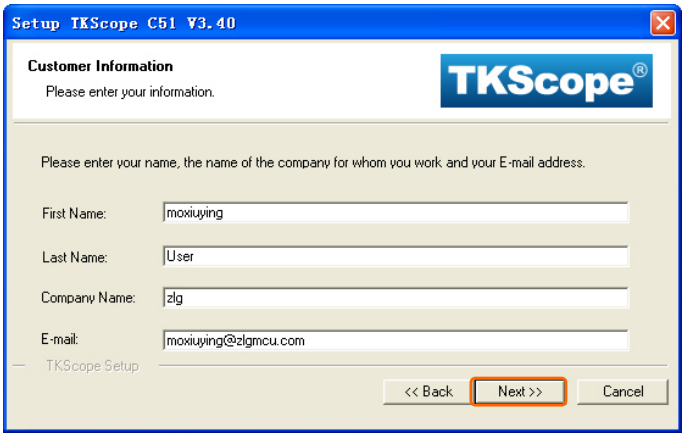

图 2.6　填写用户信息

5. 点击图 2.6中【Next】选项，即可开始仿真器驱动的安装，安装过程如图 2.7所示。

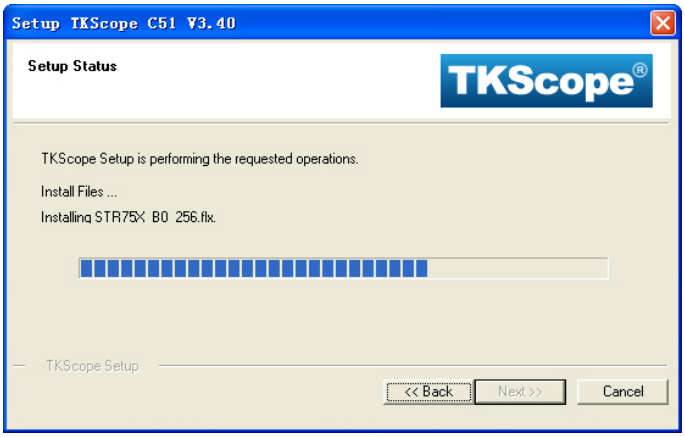

图 2.7　驱动安装进度提示框

6. 仿真器驱动程序正确安装成功的结果，如图 2.8所示，用户点击【Finish】即可结束。

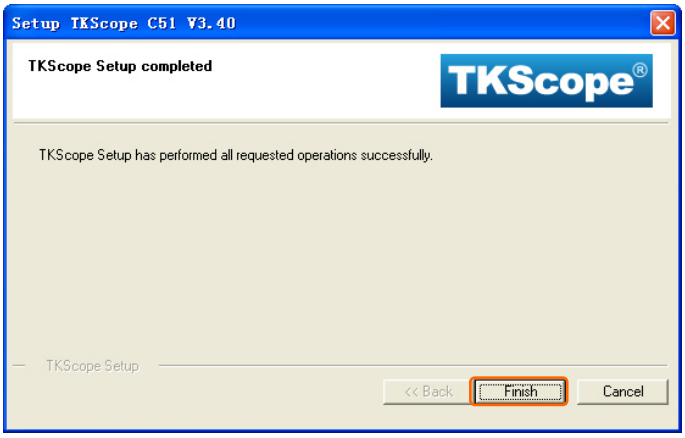

图 2.8　驱动成功安装完成

2.2.2 查看仿真器驱动安装结果

仿真器驱动正确成功安装之后，可以在仿真环境设置界面中，查看仿真器驱动安装的结果。

选择【Project】菜单下【Options for Target】选项，进入仿真环境设置界面。在【Debug】选项里，硬件仿真驱动选择下拉菜单中，可以看到TKScope仿真器的驱动选项【TKScope Debug for 8051】，如图 2.9所示。

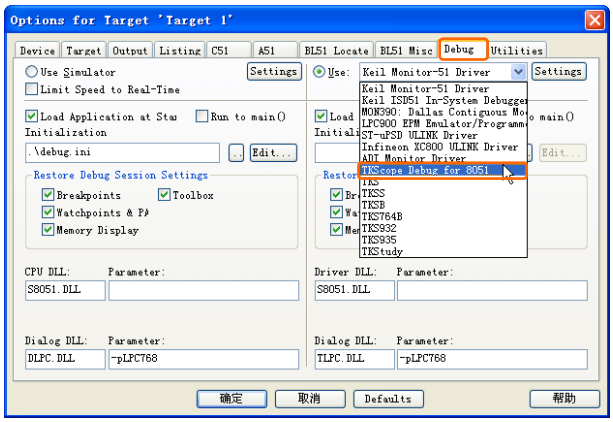

图 2.9　查看仿真器驱动安装结果

2.2.3 USB设备安装过程

驱动程序安装之后，第一次使用仿真器，给其上电时，一般情况下系统会弹出如图 2.10所示的对话框。此时，需要制定USB设备驱动的具体位置。

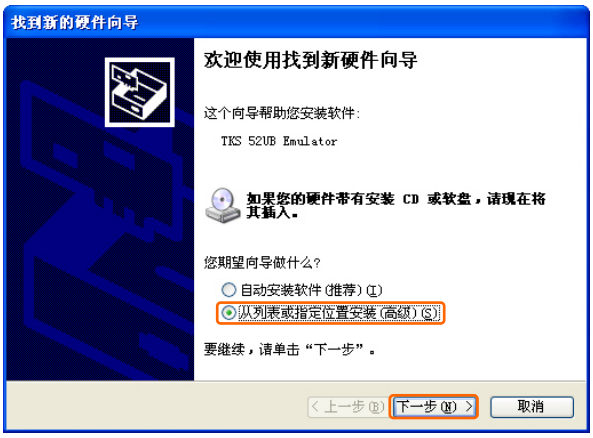

图 2.10　找到新硬件向导

1. 选择图 2.10中的【从列表或指定位置（高级）】选项，然后点击【下一步】，此时系统会弹出如图 2.11所示的对话框。

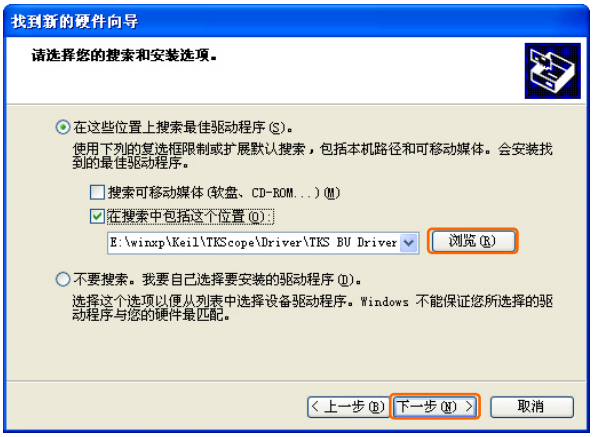

图 2.11　选择驱动提示框

2. 点击图 2.11中【浏览】选项，进入如图 2.12所示的界面。按照TKS-BU仿真器驱动安装的路径，找到驱动文件TKS BU Driver，然后点击【确定】。

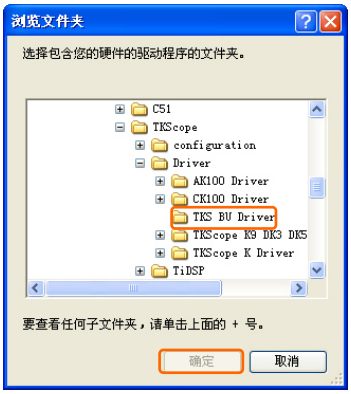

图 2.12　指定具体位置

3. 驱动安装完毕，系统会弹出如图 2.13所示的对话框，提示用户已经完成驱动的安装。此时，点击【完成】即可。至此，驱动程序安装完毕。

图 2.13　新硬件安装完成

2.2.4 USB设备安装结果

系统正确安装驱动后，可以通过查看设备管理器看到当前的硬件设备。使用鼠标右键点击【我的电脑】，选择【属性】，进入如图 2.14所示的界面。

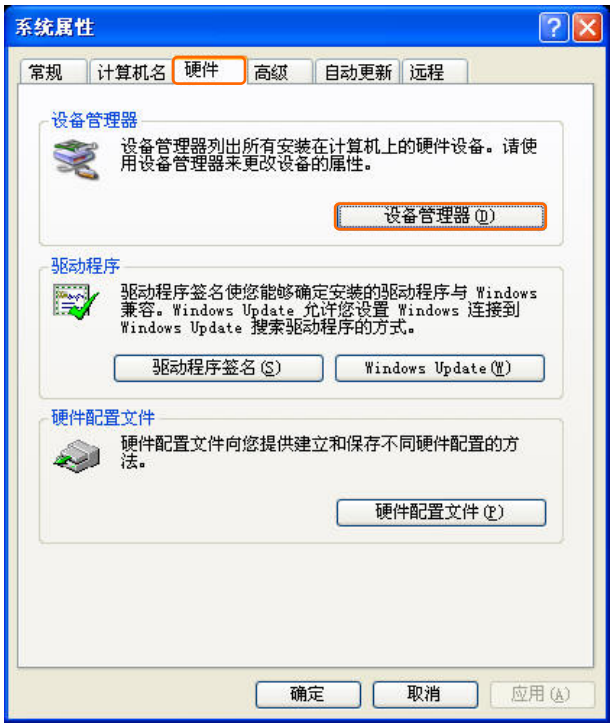

图 2.14　系统属性

在图 2.14中，点击【设备管理器】，进入如图 2.15所示的界面。此时，可以在【通用串行总线控制器】一栏内看到系统识别到的新安装的硬件设备。

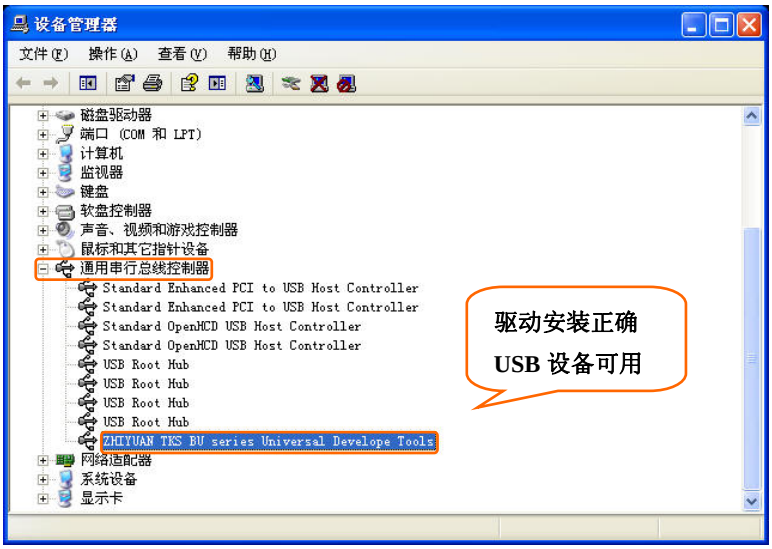

图 2.15　正确安装新硬件结果

如果系统没有安装新硬件的驱动或驱动安装不正确，那么，会看到如图 2.16 所示的界面。系统检测到新的 USB 设备，但是没有找到正确的驱动，无法正常使用。点击鼠标右键，选择【更新驱动程序】选项，如图 2.16 所示。按照上述的过程重新安装驱动程序直到正确为止。

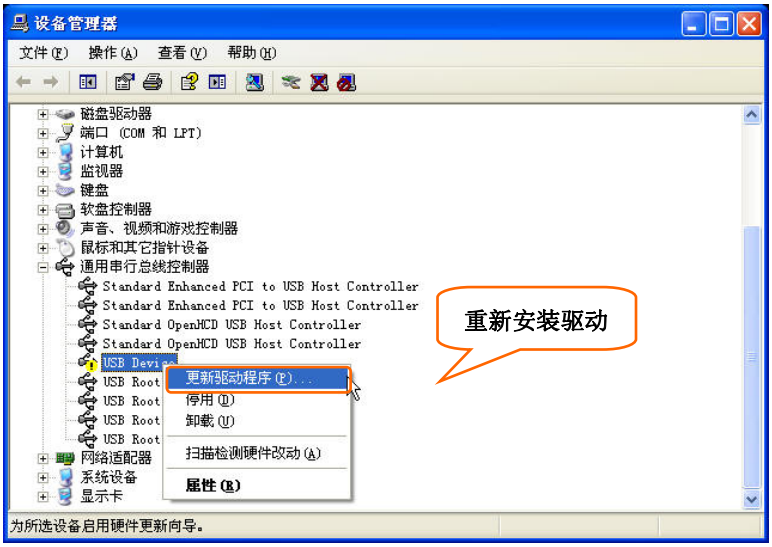

图 2.16　更新驱动程序

2.3 在Keil IDE环境下的入门操作

Keil 软件安装比较简单，在此不详细介绍，按照提示安装即可。安装完毕后，可以编写一个简单的程序来熟悉 Keil 的 IDE 环境，也可以使用 Keil 软件本身提供的范例程序，下面主要结合 Keil 的范例程序 Hello 来讲解 Keil 的具体使用。

当安装完毕后，Hello 程序位于…\Keil\C51\EXAMPLES，具体驱动器的位置取决与用户的安装选择，如果用户选择的是缺省安装，Keil 将会安装在 C 盘。

2.3.1 在Keil环境下打开工程

1. 打开Hello工程

点击 Keil 的图标启动 Keil 程序，可以看到 Keil 的主界面。在有的版本中 Hello 工程会自动打开。如果没有看到 Hello 工程或看到的不是 Hello 工程，可以点击 Project->Open Project 按照前面提示的路径找到该工程文件。

Hello工程是一个简单的C语言单模块程序，完成的主要就是通过串口发送“Hello World”字符串。Keil软件编译界面如图 2.17所示。

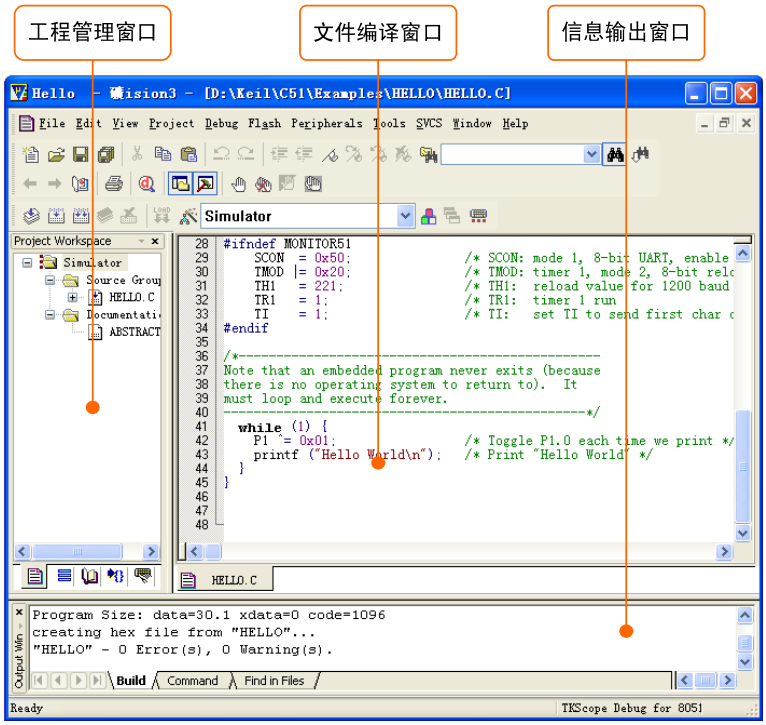

图 2.17 标准编译界面

2. 为工程选择目标器件

在打开工程以后，还应该为您的工程选择合适的目标器件。选择方法为：点击菜单【Project->Select Device for Target】，或者点击快捷图标  出现图 2.18的对话框。

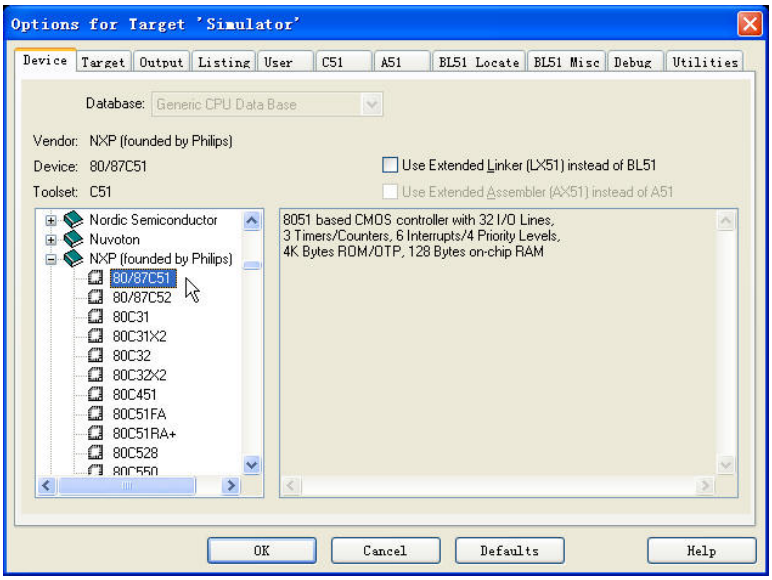

图 2.18　目标器件选择界面

在此对话窗中，左边的 Data base 窗口中把 Keil 所支持的器件按照生产厂商进行分组列表。您可以点开 NXP（founded by Philips）组，找到并选择您想仿真的目标器件。例如图中选择了 NXP 公司的 80C591，按动对话框上的确定按钮。选择正确的仿真器件非常重要，因为仿真时系统将根据所选定的器件为您提供一些特定的功能菜单。例如，如果您选择了 80C591，进入仿真时在 Peripherals 菜单中将会增加很多外设观察选择，例如 I^2C Interface 浮动窗口、PWM unit 等。

3. 编译和链接工程

选择菜单【Project->Rebuilt All Target Files Keil】，或者点击快捷图标  ，将对 Hello 工程进行连接和编译，操作结果的信息显示在输出窗口。如果用户没有对 Hello 工程做任何的改动，编译和连接将会正确完成，编译和连接的结果在信息输出窗口输出。

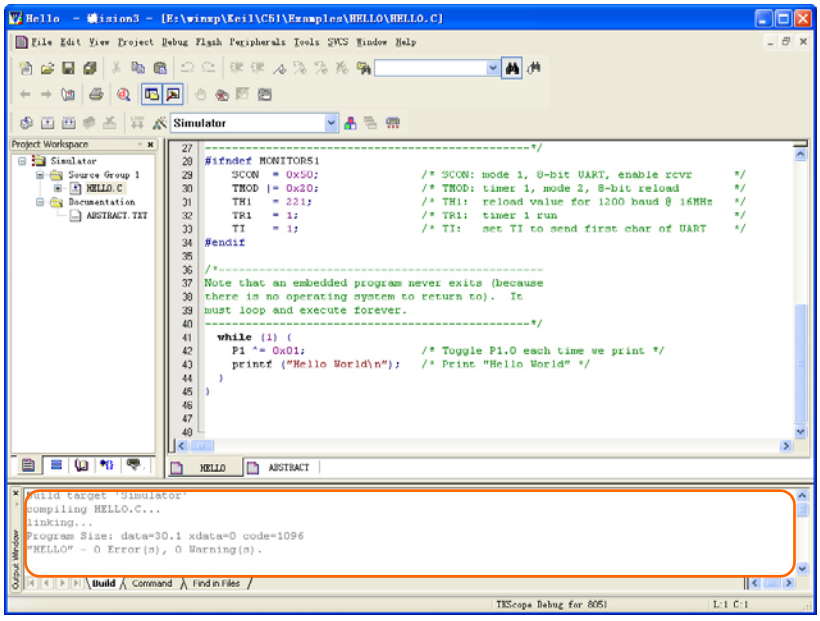

图 2.19　编译输出窗口

2.3.2 仿真Hello工程

Hello 工程经过编译和连接后，可以进行硬件仿真。在仿真前，还必须配置仿真器的参数，否则可能引起仿真失败！

选择菜单【Project->Options for Target ‘Simulator’】，或者点击快捷图标 ，选择【Debug】，进入如图 2.20所示的工程设置界面。

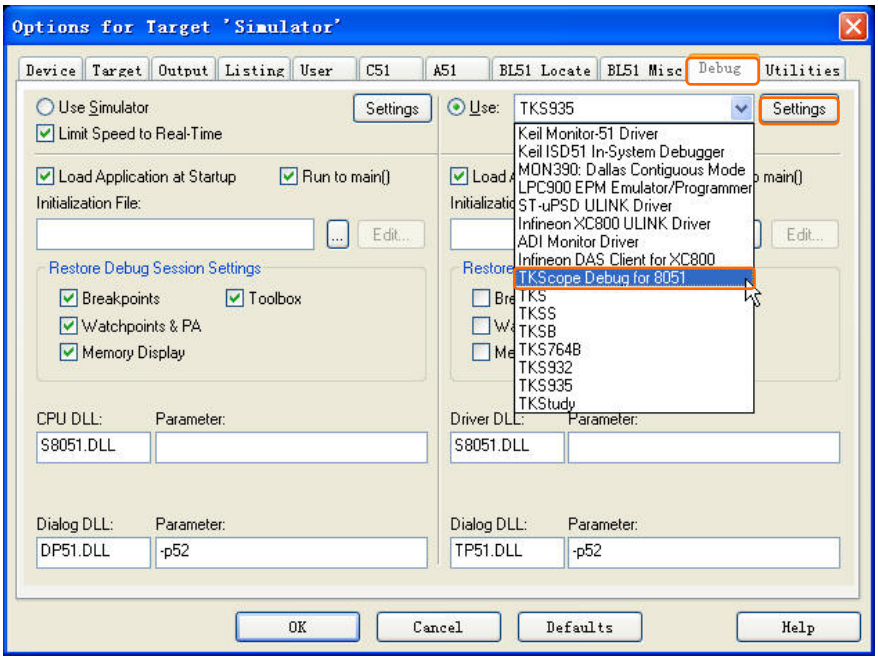

图 2.20　工程设置界面

在图 2.20中，选择硬件仿真，对应的驱动文件选择【TKScope Debug for 8051】。然后，点击【Setting】进入TKScope仿真器设置界面，如图 2.21所示。

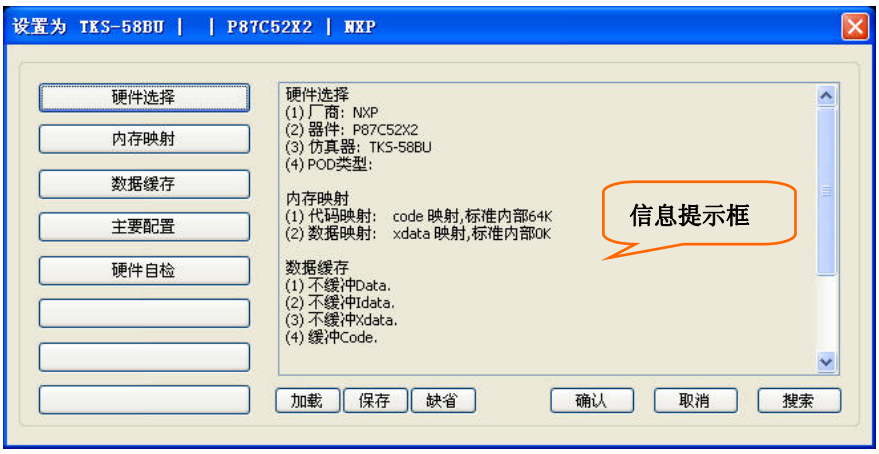

图 2.21　仿真器设置界面

图 2.21中，点击左侧的各个选项，系统会弹出相应的设置界面，同时右侧的信息提示框中会出现各项设置信息的具体含义。下面会详细介绍如何进行仿真器各参数设置。

3. 仿真器参数设置

TKS-BU 系列仿真器工作参数必须正确设置，否则可能会导致仿真错误或失败！

3.1 硬件选择

点击图 2.21中的【硬件选择】，进入如图 3.1所示的界面。

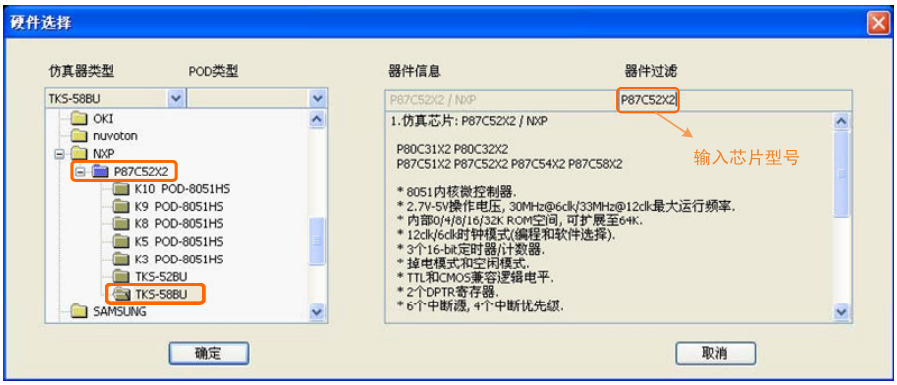

图 3.1　硬件选择界面

TKS-52BU / TKS-58BU 仿真器出厂内置的芯片型号为 P87C52X2，在【器件过滤】窗口中输入该芯片型号，在左边窗口中系统会自动帮助您找出相应的芯片，选择该芯片目录下的仿真器型号，点击【确定】即可。

用户也可以利用系统提供的【搜索】功能来完成。具体方法：用户只需选中P87C52X2芯片，点击【确定】返回到的界面；然后点击图 2.21中的【搜索】，系统会自动搜索出当前使用的仿真器类型，如图 3.2所示。如果电脑连接多个仿真器，【搜索】按钮能把所有的仿真器型号搜索出来，用户只要选中当前使用的那台仿真器即可。

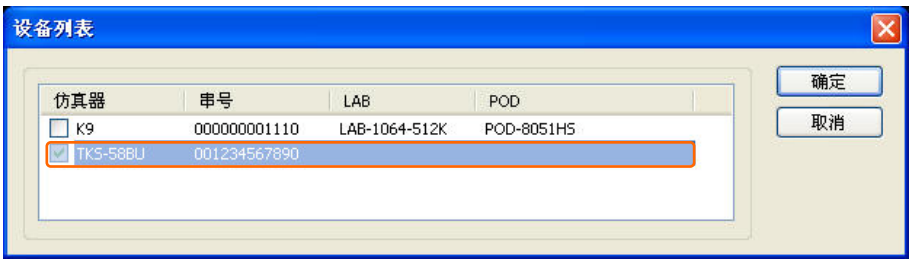

图 3.2　系统搜索结果

3.2 内存映射

点击图 2.21中的【内存映射】，进入如图 3.3所示的界面。图中状态是缺省状态，也是比较常用的状态。用户可以根据实际仿真需要添加xdata空间和code空间映射情况。

图 3.3　内存映射选型

code空间影像

TKS-BU 系列仿真器允许用户设置内部代码空间的影像，代码空间影像 Code Memory Map 窗口已经为用户设置了几个标准配置，分别是内部 64K/32K/16K/8K/4K/0K，则外部代码空间为 0K/32K/48K/56K/60K/64K。如果这些标准设置不能满足您的要求，用户可以添加自己的影像范围。

注意：添加的影像是内部代码空间，没有涉及到的空间地址则影像到仿真器外部。

添加影像范围的方法很简单，用户只需输入影像范围的起始地址和结束地址即可。例如，用户添加 2K（0x0000～0x07FF）影像范围，只需按照图 3.4所示输入，然后点击【添加】即可。

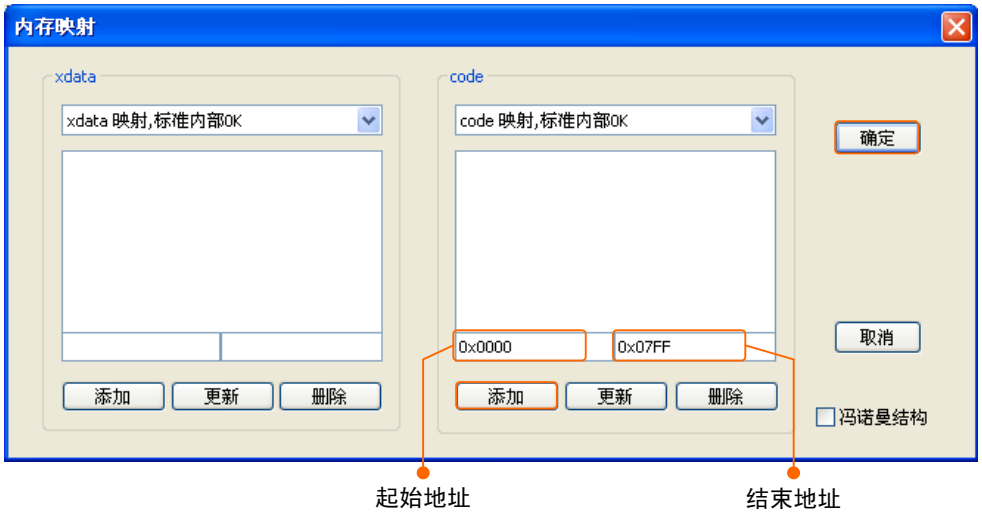

图 3.4　用户更改自定义的影像范围

用户如果要取消添加的影像范围，选中之后点击【删除】即可取消。

xdata空间影像

TKS-BU 系列仿真器允许用户设置内部数据空间的影像，数据空间影像 Xdata Memory Map 窗口已经为用户设置了几个标准配置，分别是内部 64K/32K/16K/8K/4K/0K，则外部数据空间为 0K/32K/48K/56K/60K/64K。如果这些标准设置不能满足您的要求，用户可以添加自己的影像范围。数据空间影像范围的自定义操作方法同代码空间影像范围的自定义操作方法相同，这里不再重复讲解。

注意：添加的影像是内部数据空间，没有涉及到的空间地址则影像到仿真器外部。如果某一个地址的 xdata 影像到仿真器内部，则该地址外部的 xdata 被忽略。如果用户外部有一些 xdata 的外设，不要将该外设地址映射到仿真器内部。

内部数据空间同外部数据空间的区别！内部数据空间同外部数据空间在使用上是完全相同的。但是，如果在 Use No Bus 下使用内部数据空间，由于 P0/P2 口不输出地址/数据，因此内部的 64KB 数据空间类似于片上 xdata，此时 wr/rd 信号线将正常输出信号。

冯•诺曼结构

TKScope 仿真器内部的 64KB 数据空间支持冯•诺曼结构，该结构既可以当作程序运行又可以当作数据空间进行操作。在添加数据影像时，选中冯•诺曼结构添加的项目具有冯•诺曼结构。

3.3 数据缓存

点击图 2.21中的【数据缓存】，进入如图 3.5所示的界面。

- 缓冲 data：直接寻址 data（SFR）缓存
- 缓冲 idata：间接寻址 idata 缓存
- 缓冲 xdata：外部数据 xdata 缓存
- 缓冲 code：代码数据 code 缓存

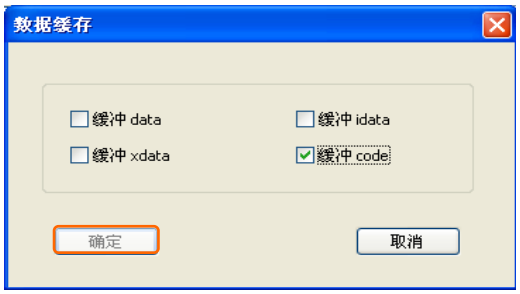

图 3.5 数据缓存选项

数据缓存（Cache Options）配置主要是解决屏幕刷新和仿真速度的矛盾。

如果用户选择 Cache，屏幕显示刷新只在用户程序运行后进行，这样可以加快显示速度，但是如果某一个操作引起的其它数据的变化可能不能及时显示，刷新只能发生运行用户代码以后。

如果用户不选择 Cache，则用户在 PC 端软件的任何操作都将引起显示数据的重新刷新，在查找不稳定硬件时比较理想，但屏幕刷新会影响操作响应速度。

建议用户使用缺省 Cache 设置，仅选择 Cache Code。

3.4 主要配置

点击图 2.21中的【主要配置】，进入如图 3.6所示的界面。

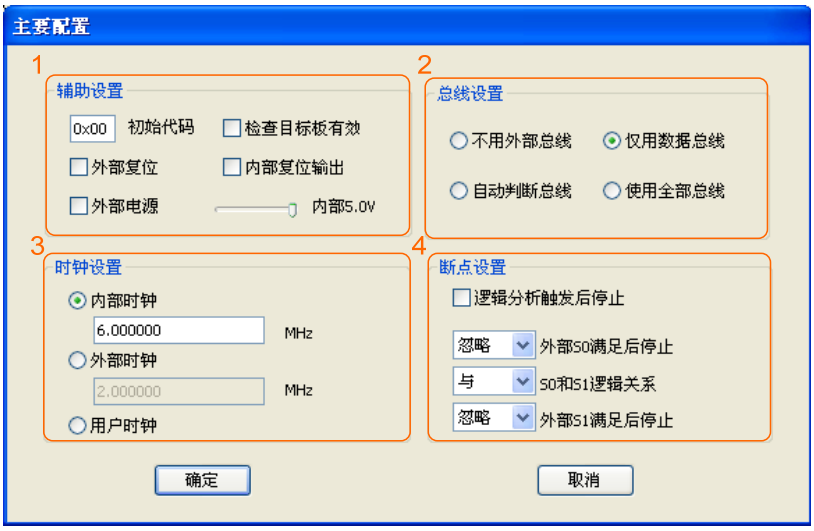

图 3.6　主要配置选项

您在第一次运行仿真器时，系统将采用缺省标准配置。缺省配置可以满足您的一般需求，如果不能满足，您需要手动进行设置，设置后的配置信息系统将予以保存，下次启动时可持续采用。

注意，TKS-BU 系列仿真器在运行前需要进行参数配置，否则可能出现错误的运行结果。

从图 3.6中可以看出，【主要配置】主要分为 4 个方面的设置（辅助设置、总线设置、时钟设置和断点设置），下面详细说明每个配置的含义。

辅助设置

辅助设置为您提供一些特殊的仿真要求，在一般情况下这些选项的缺省设置能保证用户程序的运行。

- 初始化代码

仿真器在调入用户程序前将初始化内部程序数值。代码下载后，没有用户代码的位置仍保留初始化数值，这种功能在一些特殊场合非常有用。

- 使用外部复位

仿真时如果使用外部复位输入，用户板复位脚的信号将影响用户程序的运行。程序运行时，如果发现外部复位信号，程序将复位后继续运行。

- 使用外部电源

一般情况下，仿真器内部的仿真芯片使用的是内部稳定的 5V/3.3V/1.8V 电源，能确保最佳仿真性能。如果用户系统使用的电源和 5V/3.3V/1.8V 电源有差别，可以选择使用外部电源。外部电源从用户板的电源管脚输入，电压范围在 1.8V~5.5V 之间，用户必须保证该电源的稳定性。

总线设置

为了取得最佳的仿真效果，总线分成四种仿真模式，用户可以根据实际情况加以选择，总线模式的选择需要同内存映射（Internal Memory Map）配合使用。

- 不用外部总线

在仿真时，对于操作总线的情况都加以制止，主要针对 P0/P2 口。P0/P2 口在这种情况

下是标准的 I/O 模式，遇到 MOVX/MOVC 指令时，P0/P2 口仍然表现出 I/O 特性而不输出总线信号。

如果用户没有使用总线则选择这种模式较好。另外，如果用户仅使用了 MOVX 指令来操作一些单片机的内部 xdata 空间，则也可以选择这种模式。

- 仅用数据总线

在仿真时，P0/P2 口将根据指令的运行情况来决定工作模式。遇到 MOVX 指令时，P0/P2 口将输出总线地址/数据信号，MOVX 指令结束后将恢复 I/O 模式。

如果用户使用 MOVX@Ri 指令，P2 口仍将表现出 I/O 特性，这种情况下 P2 口仍然可以作为 I/O 口使用。

- 自动判断总线

在自动总线模式下，仿真器将根据用户的程序运行情况自动决定是否打开总线。

该模式在仿真内部/外部 xdata 地址重叠情况下非常有用，用户操作内部 xdata 时总线并不打开，但是在访问外部 xdata 时总线将自动打开。

- 使用全部总线

在仿真时，P0/P2 口完全作为总线使用，用户可以通过 MOVX 指令操作仿真器外部数据空间，也可以通过 MOVC 指令读取外部代码数据，甚至运行外部的程序。在这种模式下，P0/P2 口不能再作为 I/O 口使用。

时钟设置

TKS-BU 系列仿真器允许使用内部提供的时钟，也可以使用外部提供的时钟或晶振。仿真器内部提供的时钟由于进行了专门处理，在稳定性上要超过外部时钟，而且可以运行到更高的仿真频率。外部时钟主要是由用户板上的时钟提供，或通过仿真头上的振荡组件结合仿真头上的晶振或用户板上的晶振产生，由于存在不确定因素，可能会导致外部时钟不稳定，建议用户尽可能使用仿真器内部提供的时钟。

- 内部时钟

TKS-BU 系列仿真器内部提供超高精度的 PLL 连续可调时钟，时钟频率范围为 20KHz~50MHz，用户可以在时钟频率文本框中直接输入时钟频率值(MHz 单位)。

- 外部时钟

TKS-BU 系列仿真器可以使用仿真器外部提供的时钟，时钟范围为 2MHz~24MHz，这里的外部时钟是指通过仿真头上的振荡组件结合仿真头上的晶振或用户板上的晶振产生的时钟，用户在时钟频率文本框中直接输入外部时钟的值（MHz 单位）。

- 用户时钟

TKS-BU 系列仿真器也可以使用用户时钟，时钟范围为 2MHz~24MHz。用户时钟是指用户目标板提供的时钟直接接到 X1 引脚上。

注意：用户时钟用指户目标板上的独立时钟，而不是用户目标板上的晶振产生的时钟。

断点设置

TKS-BU 系列仿真器不支持逻辑分析仪，TKScope 系列仿真器能支持。

3.5 硬件自检

点击图 2.21中的【硬件自检】，进入如图 3.7所示的界面。

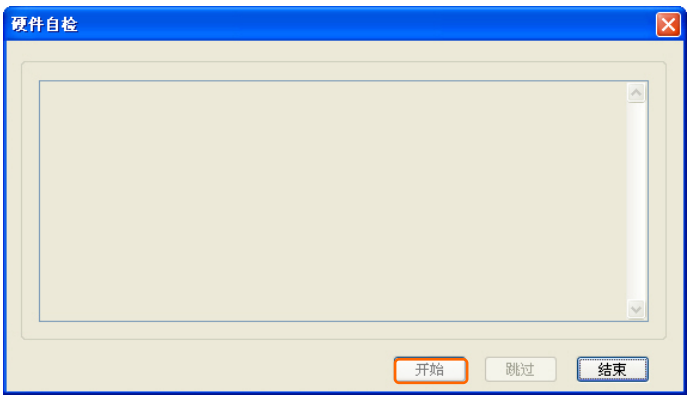

图 3.7 硬件自检选项

硬件自检主要用于TKS-BU系列仿真器自身功能检测以及连接组件故障检测。点击图 3.7中的【开始】选项，即可开始硬件自检。

硬件自检过程中，用户可以点击【跳过】选项，跳过某项功能的自检过程；也可以点击【结束】选项，提前结束自检过程。硬件自检 100％全部正确通过的结果，如图 3.8所示。

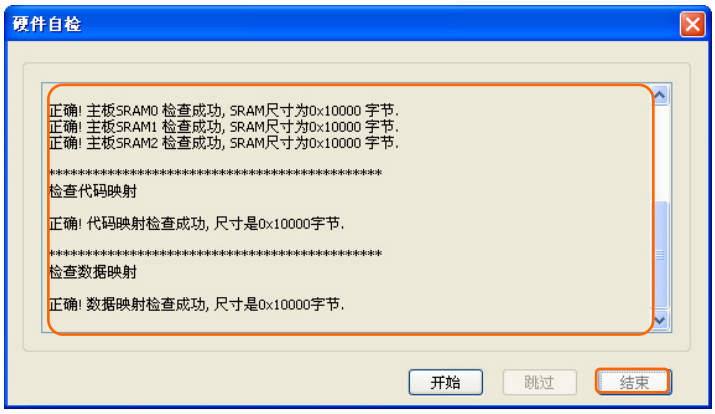

图 3.8 硬件自检全部正确完成

TKS-BU 系列仿真器硬件自检流程：

- 检查硬件初始化。
- 检查通讯速度。
- 检查主板 SRAM。
- 检查代码映射。
- 检查数据映射。

硬件自检过程中，用户在信息提示框中（如图 3.8所示）可以清楚的看到仿真器自检结果的提示信息。

如 USB 通讯速度，用户在提示框中可以看到每秒钟发送的字节数和发送的全部字节数，这样就能够直观的感受到 TKS-BU 系列仿真器的高速通讯速度。

在硬件自检的过程中，用户可以看到仿真器各个功能器件的自检结果，如硬件初始化、复位等功能是否正常，SRAM、XRAM 等器件是否正常工作以及内部是否有损坏，代码映射、数据映射空间是否正常工作以及内部是否有损坏等等。

用户在使用仿真器过程中出现工作异常的现象，可以采用硬件自检的方法来排查故障。

如果仿真器硬件自检 100%通过，仿真器肯定是能够正常工作的。

如果硬件自检不能通过，仿真器主机和连接的仿真头组件肯定存在问题，具体问题可以根据自检结果的信息提示来判定。

如仿真器内部器件有损坏，仿真电缆接触不良或内部断路，仿真芯片型号错误或损坏，仿真头接触不良等等一系列问题都可以通过硬件自检排查。

TKS-BU 系列仿真器的硬件自检功能，解决了用户在使用仿真器过程中，出现莫明其妙的问题时束手无策的尴尬局面。

4. 硬件仿真

确保仿真器驱动安装正确和仿真器参数正确设置后，点击工具栏图标 进入硬件仿真。标准仿真界面如图 4.1所示。

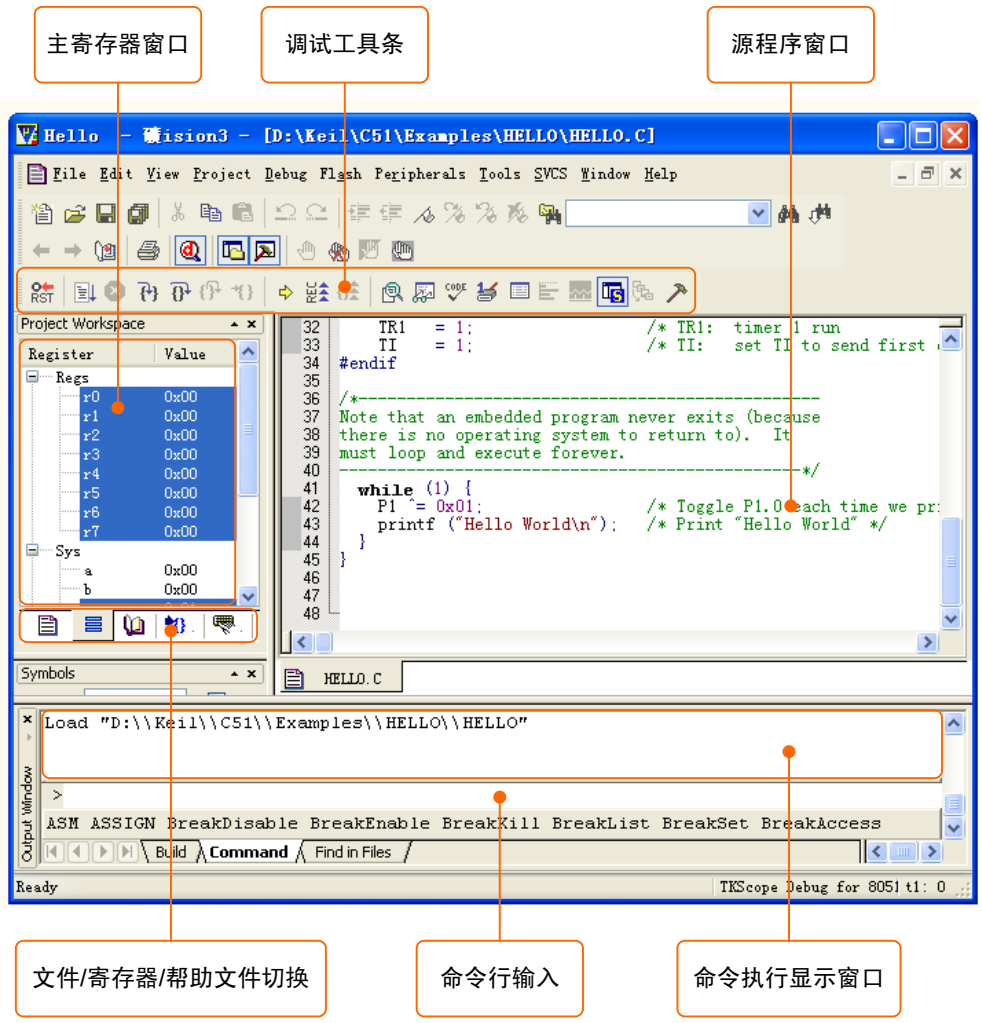

图 4.1　进入仿真界面

4.1　调试工具条

调试工具是用于调试中的快速操作。进入仿真状态之后，用户可以根据实际仿真需要，选择相应的调试工具进行仿真操作。Keil IDE 环境提供快捷的图标工具，快捷图标的定义如下。

进入/退出调试状态（Start/Stop Debug Session）。

复位（Reset CPU）。

全速运行（Go）。

停止运行（Stop）。

单步运行（Step In）。与 Step 命令不同之处在于 Step In 命令将进入该函数。

单步运行（Step）。每次执行一条语句，这时函数调用将被作为一条语句执行。

单步运行（Step Out）。执行完当前被调用的函数，停止在函数调用的下一条语句。

运行到光标（Run To Cursor）。程序运行到当前光标所在行时停止。

设置/取消断点（Insert/Remove Breakpoint）。

取消所有断点（Kill All Breakpoint）。

打开/关闭断点（Enable/Disable Breakpoint）。

关闭所有断点（Disable All Breakpoints）。

打开存储器窗口（Memory Window）。

打开反汇编窗口（Disassembly Window）。

打开变量观察窗口（Watch & Call Stack Window）。

打开代码覆盖窗口（Code Coverage Window）。

4.2 连续单步运行

TKS-BU 系列仿真器有连续单步运行的功能，每一步运行完后将刷新屏幕，然后执行下一步运行。C 语言的源程序窗口下不支持连续单步运行方式，需要打开反汇编窗口观看。

选择【View】菜单下的【Disassembly Window】选项，即打开反汇编窗口，如图 4.2所示。或者点击快捷图标 。

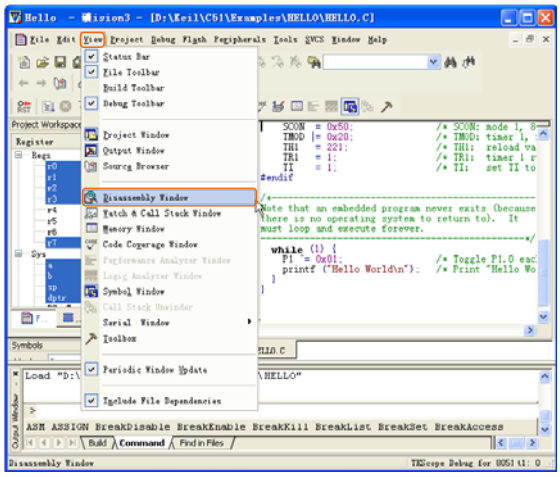

图 4.2 打开反汇编窗口选项

进入反汇编窗口，实现连续单步有以下两种方法。

按着【F11】不松手，PC 指针不断地移动，屏幕上的数据也发生变化，如果想停止运行，只要松开【F11】即可。

如果用户想查看运行 1000 个汇编单步后，程序会运行到哪里。只要在命令窗口输入例 T 1000 的命令（1000 表示运行 1000 个汇编单步），然后就可以看到程序快速的执行单步操作，PC指针不断的移动，屏幕上的数据也会发生变化，当程序执行完 1000 步后就会停止，具体操作见图 4.3。

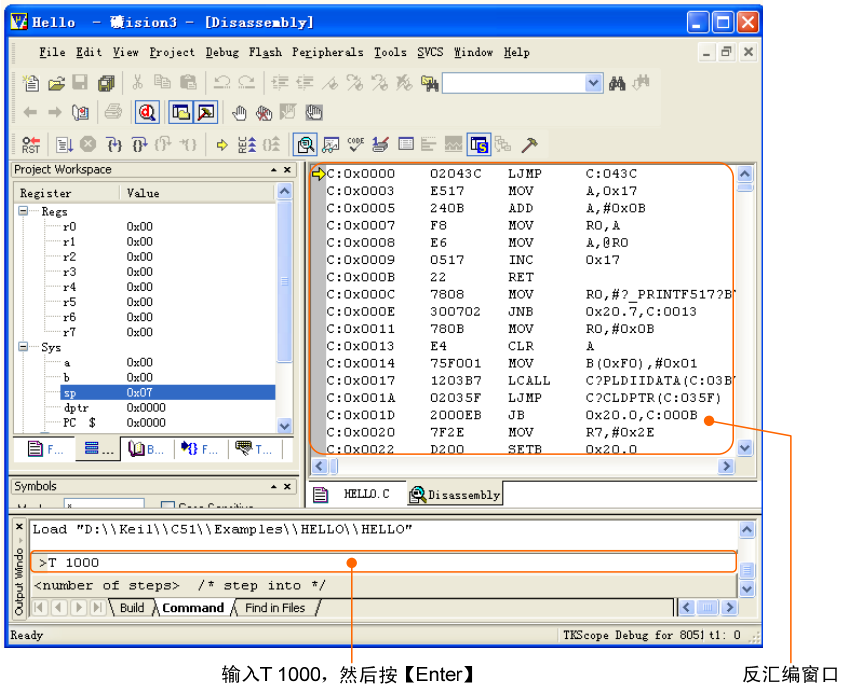

图 4.3　查看 1000 个汇编单步操作

4.3　观察存储空间

程序运行后，可以观察芯片内部的全部寄存器，方法很简单。

- 观察常用寄存器

常用的寄存器，如 A、B、DPTR、SP、R0~R7 等，在【主寄存器窗口】可直接观看到寄存器的变化值，其它的寄存器只需打开相应的寄存器窗口即可看到变化值。

如图 4.4所示，在【Peripherals】菜单下，用户可以根据实际观察进行选择，弹出的窗口中有相关的寄存器。

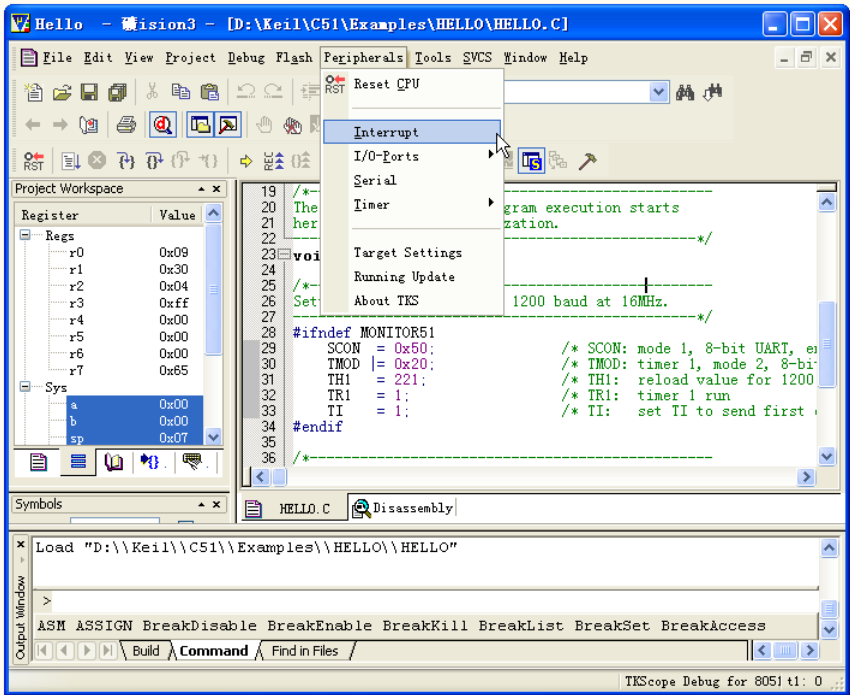

图 4.4　选择需要观察的外设选项

如用户需要观察与中断相关的寄存器，要选择【Interrupt】选项，弹出的观察窗口如图4.5所示，此时用户可以查看相关寄存器的变化值。在【Peripherals】菜单下，用户还可以选择I/O口、串口、定时器、SPI总线等，观察与之相关的寄存器。

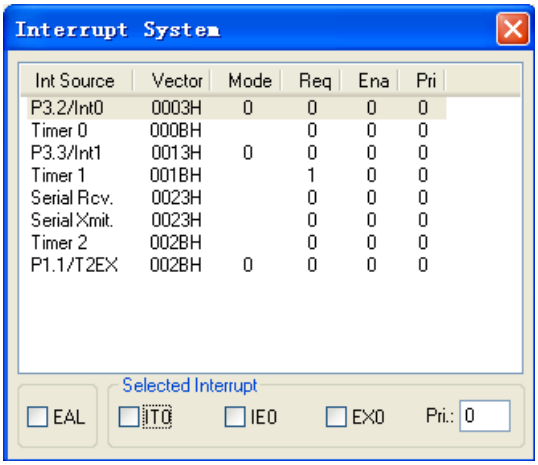

图 4.5　中断的寄存器观察窗口

- 观察 data、idata、xdata、code 空间

选择【View】菜单下【Memory Window】选项，打开存储器窗口，如图 4.6所示，或者点击快捷图标  。

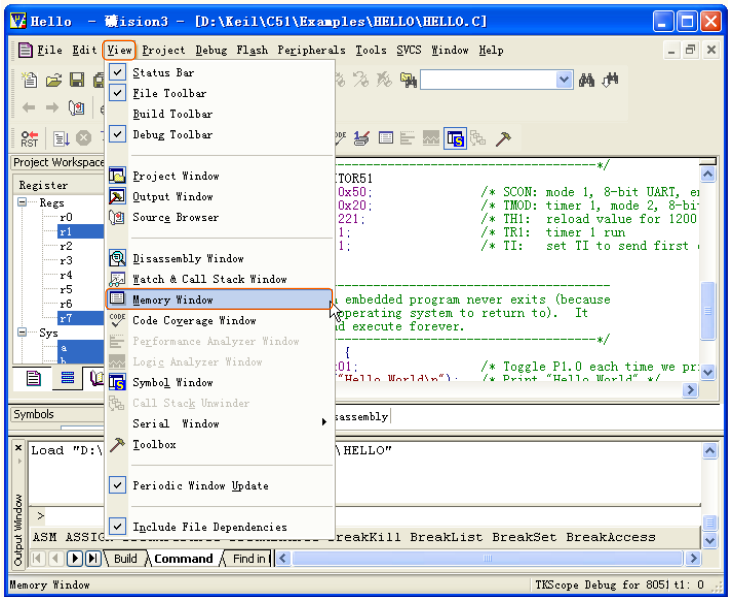

图 4.6　打开存储器窗口

在存储器窗口中输入需要观察的存储空间即可观察该空间的数据。如需要观察Data空间12H地址之后的数据，只需输入在“d:0x12”或“d:12H”，如图 4.7所示。

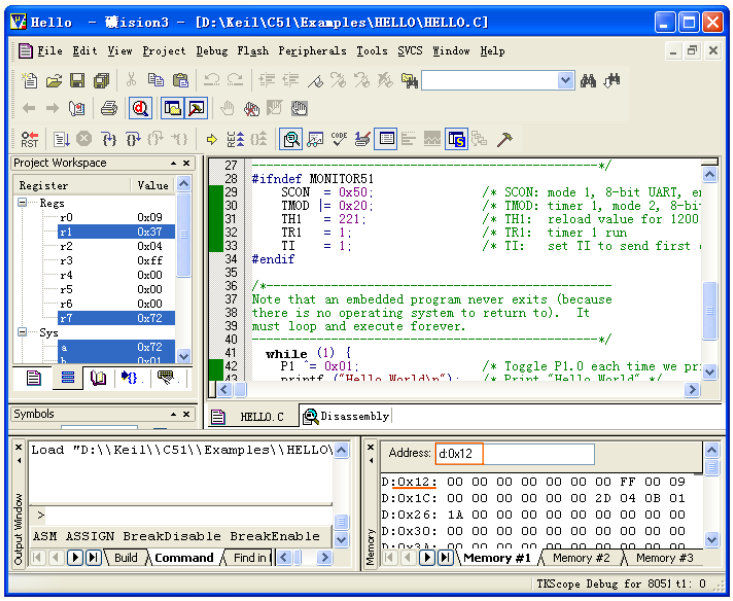

图 4.7　观察 Data 空间

观察存储空间的输入方法规则：

- Data 空间　　d:0xXX 或 d:XXH
- Idata 空间　　i:0xXX 或 i:XXH
- Xdata 空间　　x:0xXXXX 或 x:XXXXH
- Code 空间：　c:0xXXXX 或 c:XXXXH

用户如果还需要观察其它存储空间的数据，可以选择右侧的【Memory # 2】窗口，输入相应的空间值即可，Keil IDE 软件最多支持 4 个存储窗口同时观察。

4.4 观察消耗时间

程序运行过程中，用户可以观察程序消耗时间的动态变化值。如图 4.8所示，在主界面【主寄存器窗口】，可以看到时间的变化情况。

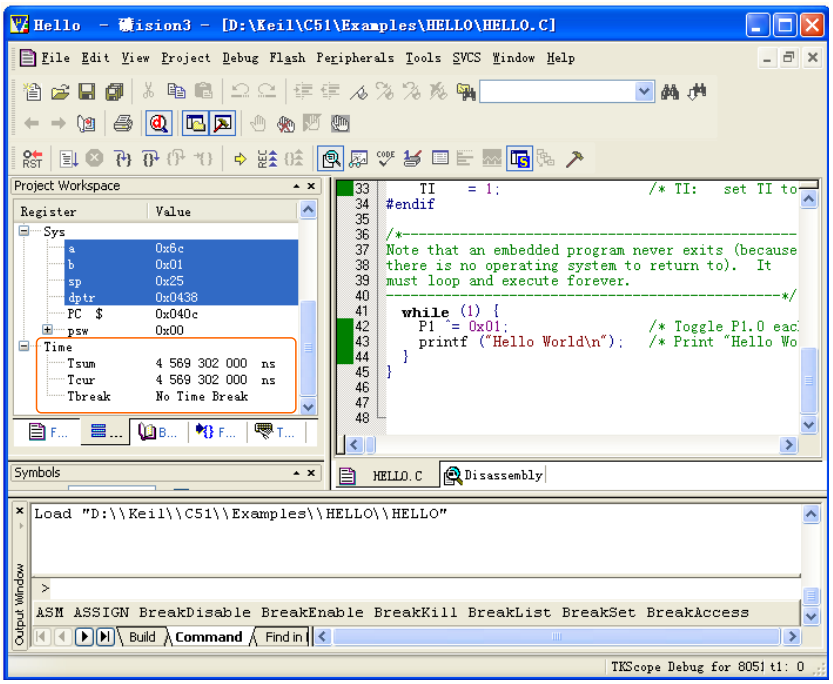

图 4.8 观察程序运行时间

- 总运行时间 Tsum

仿真器从上电或复位后，到当前状态经历的有效运行总时间。Tsum 是有效的运行时间的积累，程序处于仿真状态但停止运行时，时间不积累。

- 当前运行时间 Tcur

记录当前一次有效运行操作经历的时间。例如，运行一个单步经历了 1us，则 Tcur 显示为 1us；再运行一个单步经历了 3us。与总运行时间的不断积累不同，Tcur 是个时间差，便于用户观察本次操作经历的时间。

Tsum/Tcur 的数值是动态显示，即程序在运行过程中，可以查看运行的时间。

- 注意事项

Tsum/Tcur显示的数值跟用户选择的时钟有关。在TKS仿真器BU系列中，用户可以有 3 种时钟选择，见图 3.6。

如果用户选择仿真器内部提供的时钟，仿真器能完全准确的显示 Tsum/Tcur。

如果用户选择外部时钟，仿真器需要知道用户从外部输入时钟的频率。如果用户填写的时钟频率不准确，Tsum/Tcur 显示的数值将是错误的！用户在选择了外部时钟后，需要在时钟设置栏中填入频率数值，注意是以 MHz 为单位的。

复位后 Tsum/Tcur 的数值全部为 0，且不能改动。

4.5 加彩显示

加彩运行轨迹显示是指，系统把执行过的程序指令用彩色标注出来。在 Keil IDE 开发

环境下是在程序行的前端以绿色标注，用户很清楚地看到哪些程序执行过，哪些程序还没有执行。

由图 4.9可以看出，执行过的程序前端变成绿色，指针PC指向到还没执行的程序中。

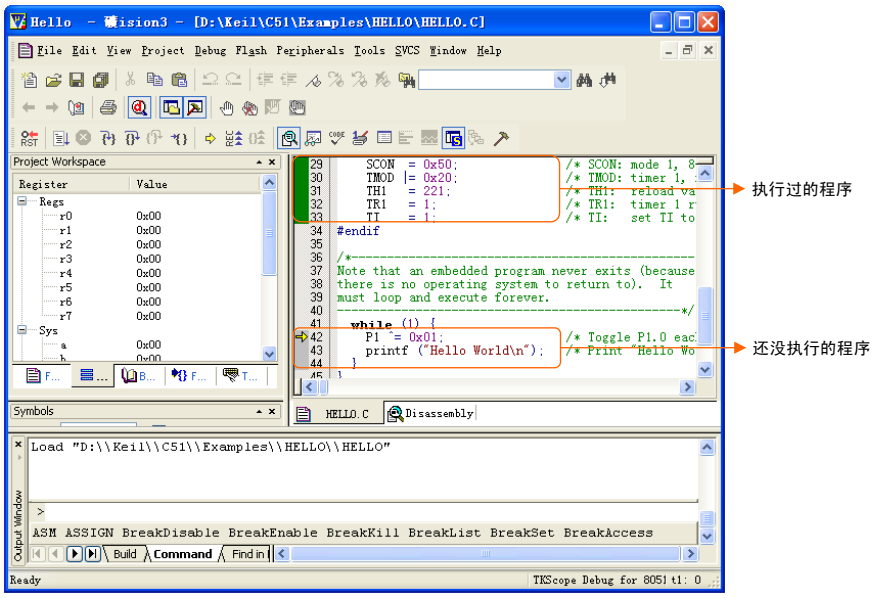

图 4.9　加彩显示

4.6　观察代码分析

代码执行覆盖分析是指，系统把程序中的各个功能函数的执行情况，以百分比的形式显示出来，便于用户直观代码执行的覆盖情况。

仿真器正确设置之后，点击【Debug】进入调试仿真状态，此时可以打开代码执行覆盖分析窗口。选择【View】菜单下的【Code Coverage Window】选项，如图 4.10所示。或者点击快捷图标 

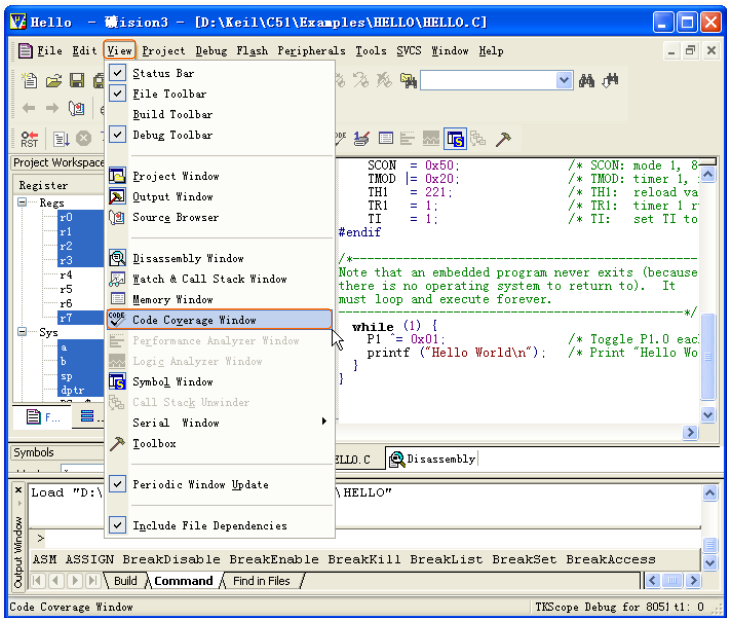

图 4.10　选择代码执行覆盖分析窗口

弹出图 4.11，从代码执行覆盖分析窗口中，可以看到程序中的功能函数系统会自动列举出来，各个函数的执行情况和指令条数在相应的函数名后面有所描述。

如图 4.11中的MAIN函数，100% of 11 instructions，含义是MAIN函数共 11 条指令，全部执行。

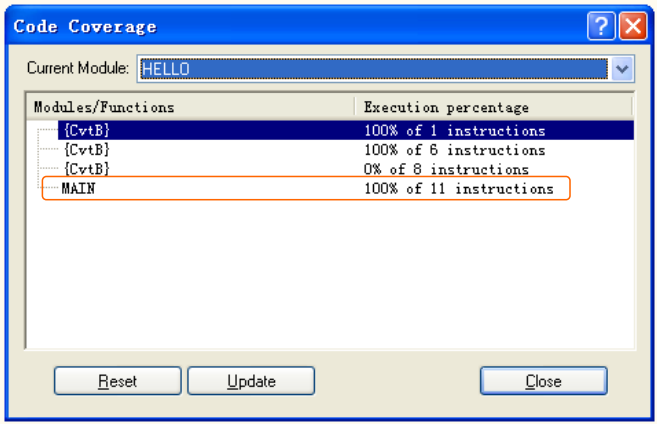

图 4.11　代码执行覆盖分析窗口

这里的指令是指汇编指令，C函数可通过查看反汇编窗口看到汇编指令，用户可以打开反汇编窗口查看。选择【View】菜单下的【Disassembly Window】选项，即打开反汇编窗口，如图 4.12所示，或者点击快捷图标  。

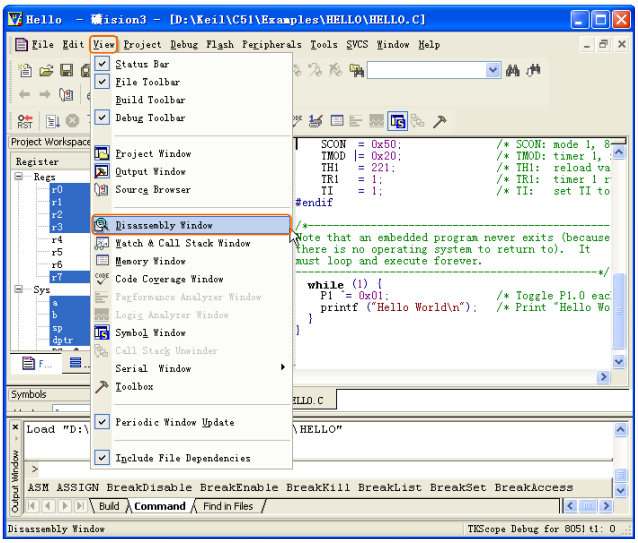

图 4.12　打开反汇编窗口选项

在反汇编窗口中可以看到 MAIN 函数的全部 11 条汇编指令的执行情况。其它函数的汇编指令查看方法同 MAIN 函数是一样的，用户请自行查看。

4.7　断点操作

断点是仿真器非常重要的功能，用户在仿真程序过程中几乎离不开断点。用户通过操作断点可以控制仿真器在指定的位置停止运行，然后分析当前的运行状态，判断程序中可能存在的问题或调试整个系统的硬件。

断点的种类很多，大体分为简单断点和复杂断点两种。不同的仿真器断点种类也不同，一般都支持简单的程序断点，也是用户经常使用的断点。高档仿真器支持的断点种类很多，如数据读/写断点、代码读取断点、组合断点等。

TKS-BU 系列仿真器支持更多的断点种类，供用户根据需要选择不同的断点来调试程序。

- 64K 程序运行断点：程序运行到该位置时停止运行。
- 64K 代码读取断点：MOVC 指令读取该地址数据时，程序在完成操作后停止运行。
- 64K 数据读取断点：MOVX 指令读取该地址数据时，程序在完成操作后停止运行。
- 64K 数据写入断点：MOVX 指令写入该地址数据时，程序在完成操作后停止运行。

用户可以单独使用上述断点，也可以混合使用。其中，程序运行断点是用户最频繁使用的断点。下面将详细讲解上述 4 种断点的操作技巧。

设置断点

断点操作包括设置断点、删除断点和关闭断点，其中删除断点和关闭断点对外表现的仿真效果是一样的，但本质是不同的。删除断点是把断点取消，需要时要再次设置；关闭断点是把断点暂时关闭，需要时简单的启动即可。

使用鼠标操作断点

使用鼠标操作设置、删除断点在程序窗口（包括C语言、汇编和反汇编窗口）中，用鼠标双击需要设置断点的程序行，则在窗口左边的状态条中出现红色的断点标志，如图 4.13所示，完成设置断点操作。

再次用鼠标双击该程序行，窗口左边的红色断点标志消失，则为删除断点操作。

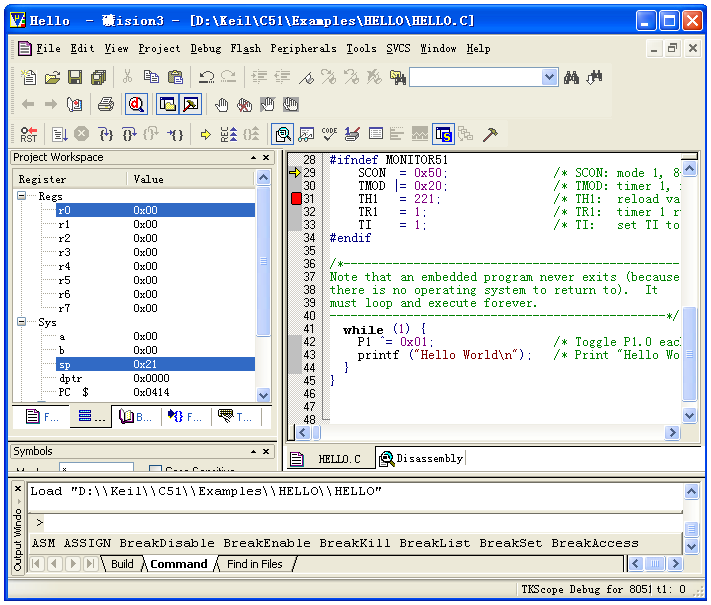

图 4.13　断点标志

使用断点菜单或断点工具条操作断点

在程序窗口（包括 C 语言、汇编和反汇编窗口）中，使用鼠标或键盘的上移/下移键将

光标移动到需要设置断点的程序行，点击快捷图标 或选择【Debug】菜单下的【Insert/Remove Breakpoint】选项，来设置/删除断点。该程序行没有设置断点，则为设置断点操作；该程序行已经设置断点，则为删除断点操作。

- 关闭断点

对于已经定义的断点，用户如果暂时不用，可以删除断点，也可以关闭断点。点击快捷图标或选择【Debug】菜单下的【Enable/Disable Breakpoint】选项，来启动/关闭断点。

- 关闭所有断点

用户可以点击快捷图标或选择【Debug】菜单下的【Disable All Breakpoints】选项，来关闭程序中设置的所有断点。

使用断点管理器

选择【Debug】菜单下的【Breakpoints】选项，即可打开断点管理器，如图 4.14所示。

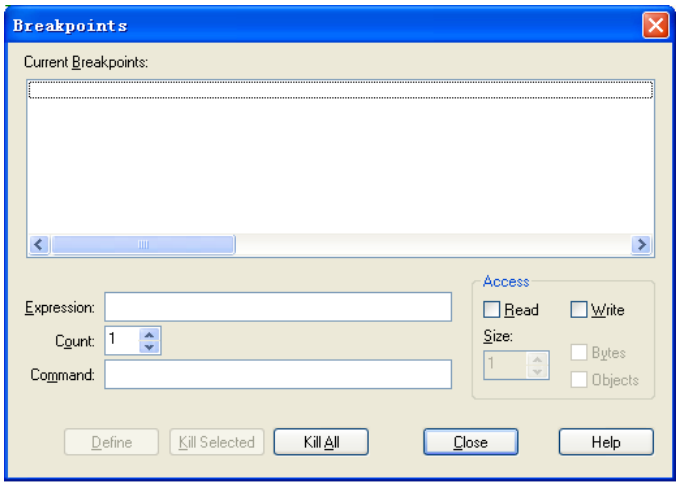

图 4.14　断点管理器

断点管理器窗口的主要功能模块含义如下：

表 4.1　断点管理器窗口的功能模块

功能模块	意义
Current Breakpoints	当前所有断点的显示窗口
Expression	断点表达式
Count	触发断点的次数（BU 系列不支持）
Command	断点命令输入（BU 系列不支持）
Read	断点操作属性为读
Write	断点操作属性为写
Size	断点的尺寸，以 Byte 为单位，BU 系列支持 8bits 尺寸（Size＝1）
Byte	BU 系列不支持选择
Objects	BU 系列不支持选择
Define	定义一个断点
Kill Selectd	删除选择的断点
Kill All	删除全部的断点

具体实例

断点的操作涉及到程序中具体的函数名、变量名等，所以需要结合下面实例来讲解，见程序清单 4.1。

程序清单 4.1　断点操作例程

```
#include<reg51.h>
#include<intrins.h>
#define uchar unsigned char
code char tempc[5] = {0x00,0x01,0x02,0x03,0x04};
xdata char tempx[5] = {0x00};
void delay(void)                    //延时子函数
{
    uchar x,y;
    for(x = 0;x < 100;x++)
        for(y = 0;y < 200;y++);
}

void main(void)
{
    uchar dat,t;
    t = 0;
    while(1)
    {
        dat = tempc[t];             //读代码段数据，执行 MOVC 指令
        delay();
        tempx[t] = dat;             //写数据到 xdata 空间，执行 MOVX 指令
        delay();
        dat = tempx[t];             //读 xdata 空间数据，执行 MOVX 指令
        delay();
        if(t >= 4) t = 0;
        else t++;
    }
}
```

1.　定义程序运行断点

程序运行断点可以采用前面介绍的简单方法操作，也可以在断点管理器中进行操作。在断点管理器中的操作有两种方法。

- 在【Expression】中写入函数名称。

例如，在上面实例中的main函数入口处设置断点，只需在【Expression】中写入“main”，【Access】中的【Read】和【Write】属性均不选择，如图 4.15所示，然后点击【Define】即可。

此时，在【Current Breakpoints】窗口可以看到定义的断点列表，在 C 程序窗口和反汇编窗口对应的程序行可以看到断点的红色标志。

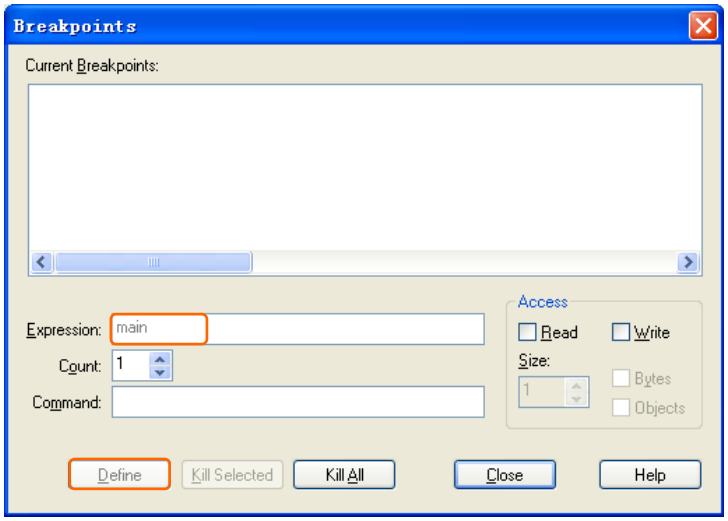

图 4.15　定义程序运行断点

使用函数名称定义断点的方法，用户无需知道断点的具体地址，只需输入函数名称，方便用户记忆。即使程序重新编译，函数地址发生变化，系统能够自动重新调整断点的位置。

- 在【Expression】中写入函数地址

例如，同样的main函数入口处设置断点，此时需要知道main函数的具体，通过查看反汇编窗口可以获得地址，如图 4.16所示。在【Expression】中写入“C：0x008F”，其它设置同方法一。其中，C表示程序属性，0x008F是main函数的地址（不同的编译环境，函数地址值可能不同）。使用函数地址定义断点的方法，用户必须知道函数准确地址。

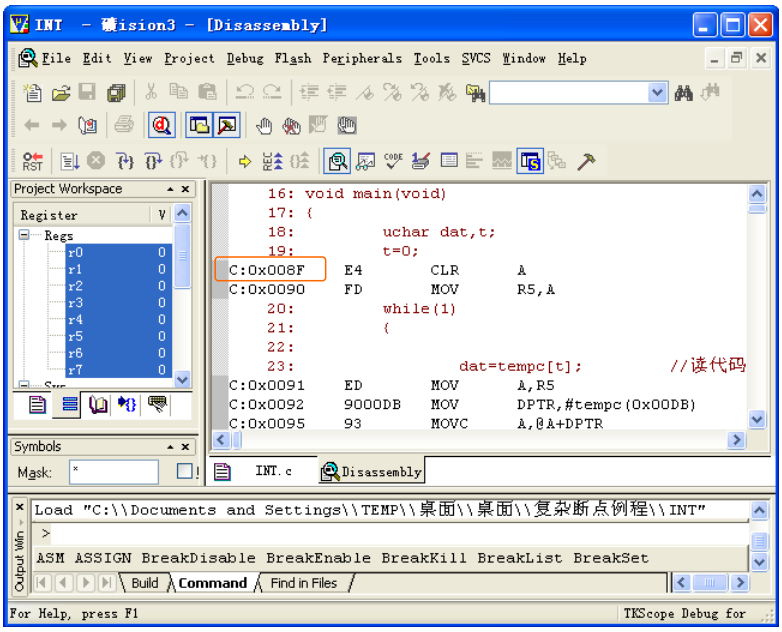

图 4.16　查看 main 函数的具体地址

2.　定义MOVC代码读取断点

MOVC 代码读取断点操作属于复杂断点操作，只能在断点管理器中进行。在断点管理器中的操作有两种方法。

- 在【Expression】中写入代码变量名称。

例如，上面实例中定义了“code char temp[5]”这样一个数组，用户如果要查看读取tempc[3]后的程序运行状态，可以在 tempc[3]后的程序运行状态，可以在 tempc[3]位置添加MOVC 读取断点。

在【Expression】中写入tempc[3]，【Access】中的【Read】属性选中【Write】属性不选中，如图 4.17所示，然后点击【Define】即可。

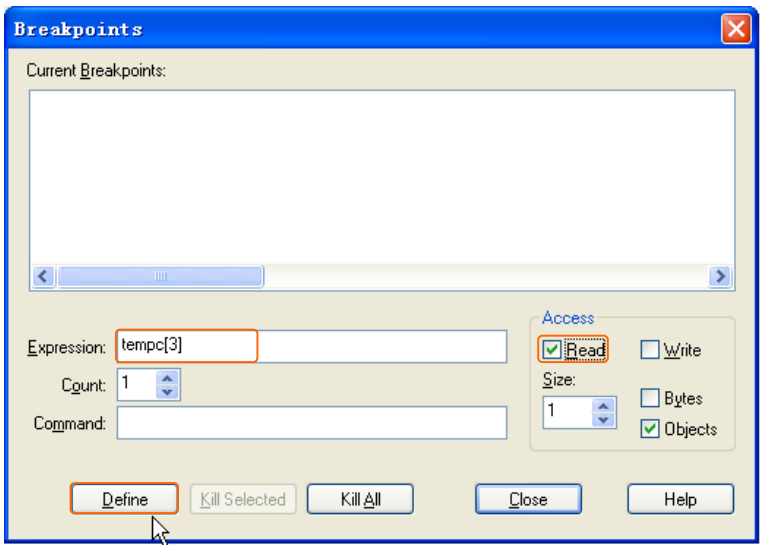

图 4.17　定义 MOVC 读取断点

点击全速运行的快捷图标，程序会停止在【C:0x0095　93　MOVC　A,@A+DPTR】，如图 4.18所示。

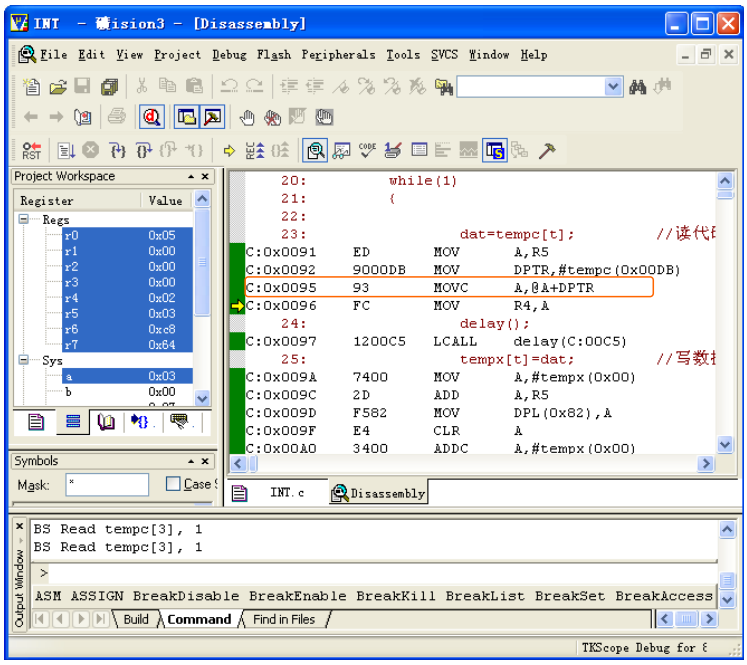

图 4.18　读取完 MOVC 指令后停止

- 在【Expression】中写入代码变量地址

例如，同样的在 tempc[3]位置添加 MOVC 读取断点，此时需要知道 tempc[3]的具体地址，通过查看反汇编窗口可以获得地址。tempc 的地址是 0x00DB，则 tempc[3]的地址是 0x00DE。在【Expression】中写入“C：0x00DE”其它设置同方法一。

使用函数地址定义断点的方法，用户必须知道函数的准确地址。

注意！在定义 MOVC 读取断点时，【Read】属性一定要选中，否则将成为程序运行断点。【Write】属性不能选中，因为程序代码无法通过指令进行写操作。

3. 定义MOVX数据读取断点

MOVX 数据读取断点操作属于复杂断点操作，只能在断点管理器中进行。在断点管理器中的操作有两种方法。

- 在【Expression】中写入外部数据变量名称。

例如，上面实例中定义了“xdata char tempx[5]”这样一个外部数据空间数组，用户如果要查看读取 tempx[2]后的程序运行状态，可以在 tempx[2]位置添加 MOVX 读取断点。

在【Expression】中写入 tempx[2]，【Access】中的【Read】属性选中【Write】属性不选中，如图 XX 所示，然后点击【Define】即可。

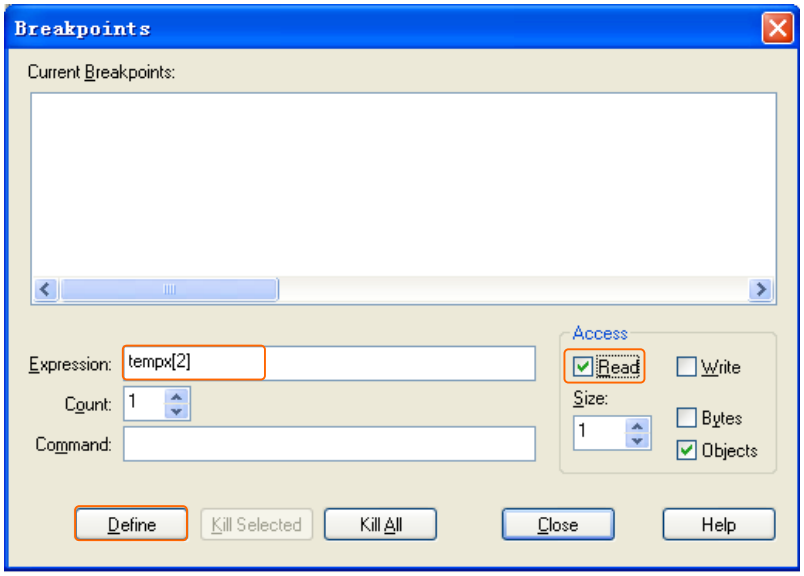

图 4.19 定义 MOVX 读取断点

点击全速运行的快捷图标，程序会执行完语句【C:0x00B3 E0 MOVX A，@DPTR】后停止，如图 4.20所示。

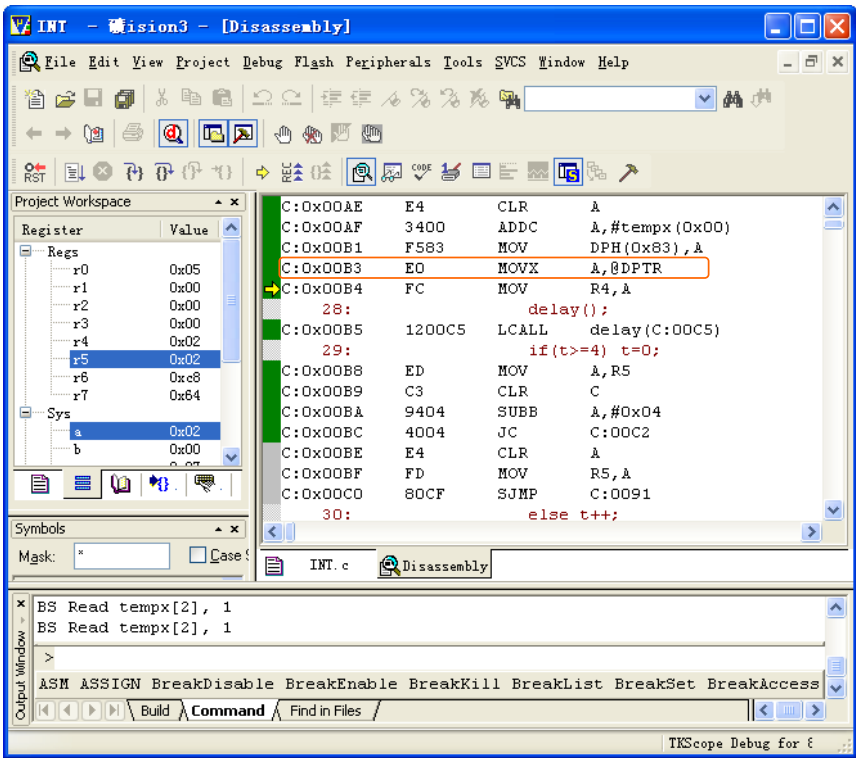

图 4.20　执行完 MOVX 指令后停止

- 在【Expression】中写入外部数据变量地址

例如，同样的在 tempx[2]位置添加 MOVX 读取断点，此时需要知道 tempx[2]的具体地址，通过查看反汇编窗口可以获得地址。Tempx 的地址是 0x00，则 tempx[2]的地址是 0x02。在【Expression】中写入“X: 0x000002”其它设置同方法一。其中，X 表示是外部数据空间属性，0x000002 是 tempx[2]的数据地址。

使用外部数据变量地址定义断点的方法，用户必须知道数据变量的准确地址。

4.　定义MOVX数据写入断点

MOVX 数据写入断点操作属于复杂断点操作，只能在断点管理器中进行。在断点管理器中的操作有两种方法。

- 在【Expression】中写入外部数据变量名称。

例如，上面实例中定义了“xdata char tempx[5]”这样一个外部数据空间数组，用户如果要查看写入数据到 tempx[4]后的程序运行状态，可以在 tempx[4]位置添加 MOVX 写入断点。

在【Expression】中写入tempx[4]，【Access】中的【Write】属性选中【Read】属性不选中，如图 4.21所示，然后点击【Define】即可。

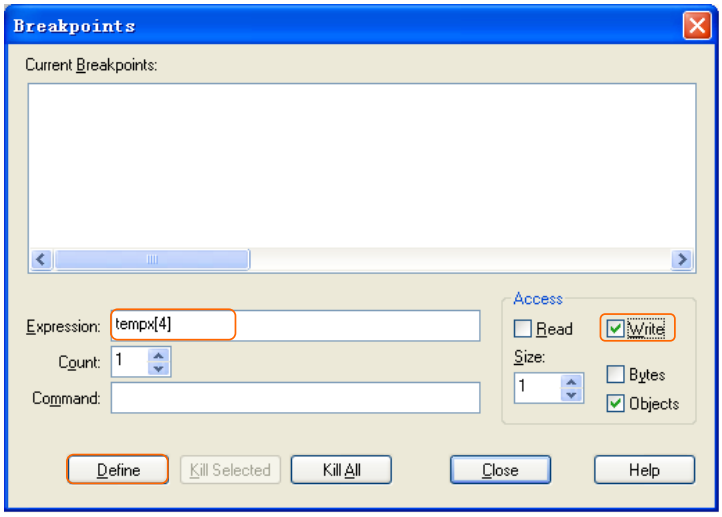

图 4.21　定义 MOVX 数据写入断点

点击全速运行的快捷图标，程序会执行完语句【C:0x007F　F0　MOVX　@DPTR，A】后停止，如图 4.22所示。

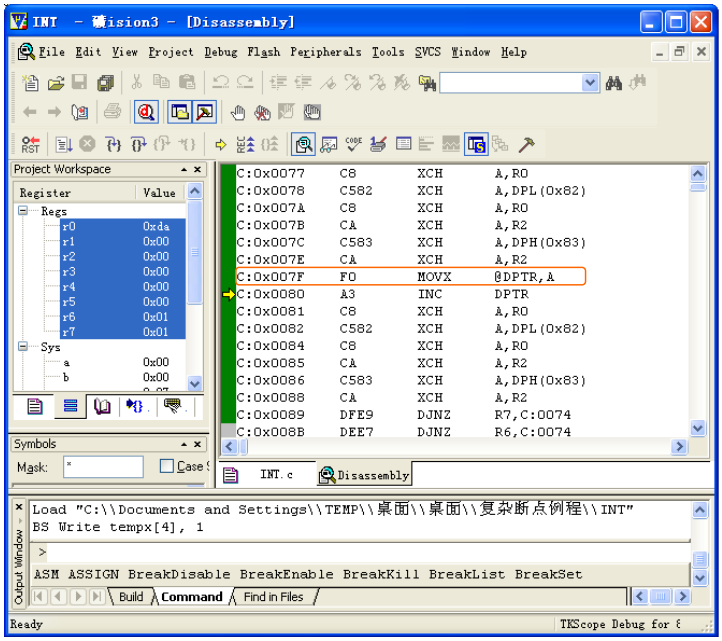

图 4.22　执行完 MOVX 指令后停止

注意！在定义 MOVX 读取/写入断点时，【Read】/【Write】至少必须选中一项，否则将不能形成有效的 MOVX 操作断点。

如果【Read】/【Write】全部选中，则该断点在读取/写入任何操作发生时都会形成有效的断点。

对于【Current Breakpoints】窗口中存在的各个断点，用户可以暂时关闭，使其断点作用失效。方法是点击断点表达式前面的打勾框，使之取消打勾。由于失效的断点仍然存在于【Current Breakpoints】窗口中，因此用户可以在需要的时候重新启用，这样比删除断点然后再添加断点方便得多。如图 4.23所示，就是关闭main函数入口处设置的断点。

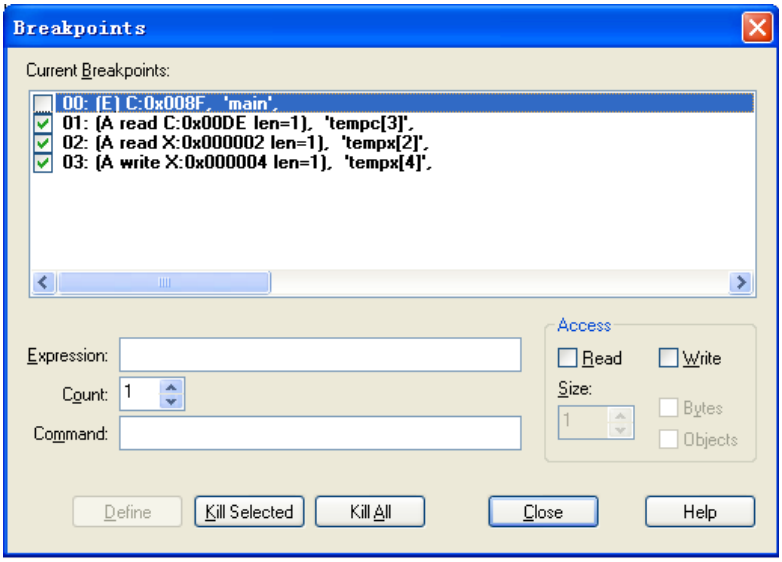

图 4.23　关闭 main 函数处的断点

5. TKS-BU系列仿真器技术支持

5.1 常见问题

1. 仿真器无法联机，不能正常使用

如果出现通讯失败将无法进入硬件仿真，出现这种现象的原因一般是：

- 仿真器同 PC 的通讯电缆连接不良。
- 仿真器没有接电源。
- 错误的选择了仿真器硬件驱动程序。
- 没有正确安装驱动，系统识别不到新的 USB 硬件设备。

2. 使用仿真器内部提供的电源仿真正常，而使用用户电源则不正常

仿真器可以使用外部用户板提供的电源，但在以下情况下无法使用外部电源：

- 外部电源没有从仿真头的电源输入脚引入或因为接触不好而无法引入。
- 用户外部输入的电源电压没有在 2.0V~5.5V 之间或不稳定。
- 在仿真器的设置中没有选择使用外部电源。
- 仿真头电源选择不正确。

3. 无法使用外部时钟

- 仿真头/用户板上没有晶振，或者晶振损坏不能正常工作。
- 仿真头上的晶振选择跳线没有正确设置。
- 仿真电缆同仿真器主机和仿真头之间接触不良。
- 用户没有选择使用外部时钟。

4. 单步运行后仿真器会进入死机状态，且运行和监控指示灯全部点亮。

这种现象一般出现在 C 语言调试中，或者 C 语言和汇编语言混合调试中。如果当用户当前的源程序窗是 C 源程序，但是当前 PC 的位置却并没有在 C 源程序的有效范围内，单步运行实际上是要求仿真器单步运行一条 C 程序语句，到下一条 C 程序语句位置停止。由于当前的 PC 位置并没有处于 C 的有效位置，因此，仿真器只能单步运行每一条汇编程序到下一条 C 程序语句位置。如果这期间的汇编语句较多，可能会耗费较多的时间。仿真器连续的单步运行，不再接收其它命令，好象进入死机状态。连续快速的在单步运行和监控不断切换所以监控和运行指示灯会全亮。

例如，如果用户当前的 PC=0，这时在 C 源程序中不会有程序运行位置指示，但是在汇编窗口中运行指示在 PC=0000H 位置。如果用户把当前的窗口点击到 C 程序窗口，则仿真器认为下一个单步将运行到 main()函数位置，在用户要求单步运行后将单步运行 PC=0000H 到 main()函数期间的所有汇编指令，可能要耗费较多的时间。

采用全速运行加断点的方式可以解决这个问题。

5.2 联系我们

感谢您选用 TKS-BU 系列仿真器，我们将竭诚为您提供技术服务，帮助您解决在使用仿真器开发过程中遇到的问题。为了尽快解决问题，不影响您的开发进度，我们建议您通过下面的方式联系我们，获得技术支持。

电话（传真）

使用电话沟通交流时最快的解决问题的途径。请您在拨通电话之前，整理好您的问题，将仿真器放置在计算机和电话机旁边并处于联机状态，我们的技术工程师会根据您的操作来分析您提出的问题。

技术支持电话：020-22644360

电子邮箱

使用电子邮件您可以详细的说明描述您遇到的问题，也可以提供测试程序或工程文件等，我们更有针对性的帮您分析解决问题。

技术支持Email：TKS@zlgmcu.com。

BBS

欢迎访问周立功单片机论坛 http://www.zlgmcu.com.cn/index.asp，我们为仿真器开设了专门的 BBS，并安排工程师轮流值日制度，对于用户提出的全部问题，可以得到我们详尽的技术解答，同时也可以得到其他有经验的网友朋友的回答。

维修服务

如果您在使用过程中遇到仿真器损坏的现象，请寄给广州致远电子有限公司维修部，由生产厂商直接快捷地为您服务，详细的联系方式如下：

公司：广州致远电子有限公司

地址：广州市天河区车陂路黄洲工业区 3 栋 2 楼

电话：020-22644245

注意！由于仿真器属于特殊产品，需要用户在开发环境里正确的设置才能正常使用，所以多数情况下仿真器出现问题可能是设置不正确引起的，而不是仿真器损坏，用户不需要返修而解决问题。如果您认为仿真器出现问题，我们建议您在决定返修前，与我们的技术支持工程师进行联系，确认仿真器是否真正的出现硬件上故障要维修。另外，您需要将仿真器的故障现象和您的详细联系方式写成文字资料，与仿真器一同返回，这样我们的维修人员能根据您提供的资料快速的进行处理，节省您的维修时间。

5.3 结束语

TKS-BU 系列仿真器是广州致远电子有限公司在 NXP 和 Keil 公司支持下推出业界领先的系列仿真器，采用当前最先进的 HOOKS 仿真技术，设计独到的仿真性能处于全球的全面领先水准。

再次感谢您选用TKS-BU系列仿真器，希望能够带给您帮助和惊喜，给您不一样的开发感觉，让您如虎添翼，项目早日成功！如果您遇到问题，请联系我们的技术支持工程师或访问我们的主页http://www.zlgmcu.com。

销售与服务网络（一）

广州周立功单片机发展有限公司

地址：广州市天河北路 689 号光大银行大厦 12 楼 F4 邮编：510630
电话：(020)38730916 38730917 38730972 38730976 38730977
传真：(020)38730925
网址：www.zlgmcu.com

广州专卖店

地址：广州市天河区新赛格电子城 203-204 室
电话：(020)87578634 87569917
传真：(020)87578842

南京周立功

地址：南京市珠江路 280 号珠江大厦 2006 室
电话：(025)83613221 83613271 83603500
传真：(025)83613271

北京周立功

地址：北京市海淀区知春路 113 号银网中心 A 座 1207-1208 室 （中发电子市场斜对面）
电话：(010)62536178 62536179 82628073
传真：(010)82614433

重庆周立功

地址：重庆市石桥铺科园一路二号大西洋国际大厦（赛格电子市场）1611 室
电话：(023)68796438 68796439
传真：(023)68796439

杭州周立功

地址：杭州市天目山路 217 号江南电子大厦 502 室
电话：(0571) 28139611 28139612 28139613
传真：(0571) 28139621

成都周立功

地址：成都市一环路南二段 1 号数码同人港 401 室（磨子桥立交西北角）
电话：(028)85439836 85437446
传真：(028)85437896

深圳周立功

地址：深圳市深南中路 2070 号电子科技大厦 C 座 4 楼 D 室
电话：(0755)83781788（5 线）
传真：(0755)83793285

武汉周立功

地址：武汉市洪山区广埠屯珞瑜路 158 号 12128 室（华中电脑数码市场）
电话：(027)87168497 87168297 87168397
传真：(027)87163755

上海周立功

地址：上海市北京东路 668 号科技京城东座 7E 室
电话：(021)53083452 53083453 53083496
传真：(021)53083491

西安办事处

地址：西安市长安北路 54 号太平洋大厦 1201 室
电话：(029)87881296 83063000 87881295
传真：(029)87880865

销售与服务网络（二）

广州致远电子有限公司

地址：广州市天河区车陂路黄洲工业区 3 栋 2 楼
邮编：510660
传真：(020)38601859
网址：www.embedtools.com （嵌入式系统事业部）
www.embedcontrol.com （工控网络事业部）
www.ecardsys.com （楼宇自动化事业部）

技术支持：

CAN-bus：

电话：(020)22644381 22644382 22644253
邮箱：can.support@embedcontrol.com

iCAN 及模块：

电话：(020)28872344 22644373
邮箱：ican@embedcontrol.com

MiniARM：

电话：(020)28872684 28267813
邮箱：miniarm.support@embedtools.com

以太网及无线：

电话：(020)22644380 22644385 22644386
邮箱：wireless@embedcontrol.com
ethernet.support@embedcontrol.com

编程器：

电话：(020)22644371
邮箱：programmer@embedtools.com

分析仪器：

电话：(020)22644375 28872624 28872345
邮箱：tools@embedtools.com

ARM 嵌入式系统：

电话：(020)28872347 28872377 22644383 22644384
邮箱：arm.support@zlgmcu.com

楼宇自动化：

电话：(020)22644376 22644389 28267806
邮箱：mjs.support@ecardsys.com
mifare.support@zlgmcu.com

销售：

电话：(020)22644249 22644399 22644372 22644261 28872524
28872342 28872349 28872569 28872573 38601786

维修：

电话：(020)22644245