

V9801Test_IAR_学习板工程说明

本文档为 SD502 工具包套配学习板的说明文档, 主要介绍了如何利用万高科技提供的代码, 在 IAR Embedded Workbench for MCS-51 的环境下建立学习板工程 V9801Test, 并利用该工程验证 SD502 是否正常工作。

V3.0

2014-09-19

本文档也描述了如何使用 IAR Embedded Workbench for MCS-51 的编译器/汇编器和链接器/定位器, 对程序进行代码分页 (Code Banking), 进而完成大于 64KB 的 V9801 应用程序的编译、调试, 并将其烧写入 Flash。如果创建一个不需要进行代码分页的 V9801 应用程序, 请参照 V9811Test_IAR_学习板工程说明.pdf 进行设置。

目前支持的 IAR 版本有 IAR Embedded Workbench for MCS-51 7.51A 和 IAR Embedded Workbench for 8051 8.30。在安装目录下它们命名方式分别是: Embedded Workbench 5.3 和 Embedded Workbench 6.0, 以下我们将以上两个版本简单命名为 7.5 和 8.3。7.5 和 8.3 版本的 dll 不能兼容, 万高公司提供两套 dll, 7.5 版本 IAR 使用的 dll 为 VangoDriver_SD502_IAR75_Vxx.xx.dll, 8.3 版本 IAR 使用的 dll 为 VangoDriver_SD502_IAR83_Vxx.xx.dll。



1. 准备

1. 按照《SD502 用户手册》的说明按装 SD502 的 USB 驱动和万高科技专用 DLL (IAR 版);
2. 硬件: SD502;
3. 芯片: V9801 (8052 MCU 内核, 128KB Flash, 4KB RAM);
4. 源文件: cstartup.s51/iar_banked_code_support.s51/ Ink51ew.xcl/main.c/ delay.c/ bank0.c/bank1.c/bank2.c;
5. 参考资料: V9801 数据手册、EW8051_UserGuide、EW8051_Compiler Reference.pdf

2. 关于代码分页程序

V9801 的片上 Flash 存储器大小为 128KB, 可用于存储大于 64KB 的应用程序。但是, 8051 内核 MCU 的地址总线宽度为 16-bit, 最多只能寻址 64KB 的程序存储空间。所以, 对于超过 64KB 的 V9801 的应用程序, 用户可采用代码分页 (Code Banking) 技术, 将应用程序分成多个小于或等于 64KB 的代码段, 通过硬件额外增加的逻辑扩展地址线, 并结合软件, 将这些代码段烧写入 Flash 中, 供 MCU 访问。

对应用程序进行代码分页时, 应将代码分成两类: 公共代码和段代码。公共代码为所有段代码所共用, 包括一些所有程序都会调用的函数, 比如: 复位向量 (Reset Vector)、中断向量 (Interrupt Vector) 和中断进程 (Interrupt Routine) 等, 而且为了实现不同段代码的跳转, 那些控制段代码跳转的程序也应归为公共代码。段代码一般是各功能模块相应的代码。段代码编号从 0 开始, 即 Bank0……Bankn。不同段代码内的程序互相调用, 必须先进行段代码切换, 增加了程序执行的时间, 所以, 为了避免通过切换不同段代码来实现同一个功能, 同一个功能模块的代码一般被归为一个段代码。

8051 的 64KB 的地址空间被划分成两部分, 公共代码区域 (Common Code Area) 和段代码区域 (Bank Code Area)。在地址空间中, 公共代码区域一般从 0x0000 开始, 至段代码区域开始。CPU 执行程序时, 公共代码映射到公共代码区域。根据程序执行的需要, 不同的段代码将轮流映射到段代码区域。在 V9801 中, 片上 8051 的地址空间的低 32KB (0x0000~0x7FFF) 为公共代码区域, 而高 32KB (0x8000~0xFFFF) 为段代码区域。

3. 硬件支持

V9801 的片上 8051 额外增加了两条逻辑扩展地址线, 即, SFR 0xA0 (CBANK) 的 bit1 和 bit0。这两个位与地址总线的最高位 (A15) 一起, 以 “与” 逻辑的关系, 将 8051 连接到片上 Flash 地址线的高 2 位, 使 8051 可以访问 128KB 的 Flash 存储器地址范围。

只要 A15 为低电平, 8051 即可访问 Flash 存储器的低 32KB 地址 (0x0000~0x7FFF), 该地址范围用于存储公共代码, 即 Root Code。

当 A15 为高电平时，根据寄存器 CBANK SFR 的配置值，8051 访问 Flash 存储器的 3 个区域，即，V9801 支持用户将应用程序分成 3 个段代码，分别被存储于 Flash 的这 3 个区域。用户可通过配置 CBANK SFR 的值进行段代码切换。

在 V9801 中，CBANK SFR 不可被配置为 0x00，而在 8051 IAR C/C++ 编译器中，代码选择寄存器 ?CBANK 默认为 0x00，所以，在使用 8051 IAR C/C++ 编译器进行代码分页时，用户应对 ?CBANK 配置进行修改，使其与 V9801 的段代码选择寄存器 CBANK SFR (0xA0) 相关连，以适应 V9801 的应用。其中，?CBANK 的值代表了各段代码的编号。

表 1 当 A15 为高电平时，CBANK SFR (0xA0) 配置与段代码各种地址的关系

CBANK	?CBANK	映射到地址空间 (0x8000~0xFFFF) 的段代码	物理地址	逻辑地址
0x01	0x00	Bank0	0x08000~0x0FFFF	0x008000~0x00FFFF
0x02	0x01	Bank1	0x10000~0x17FFF	0x018000~0x01FFFF
0x03	0x02	Bank2	0x18000~0x1FFFF	0x028000~0x02FFFF

发生 POR、恢复供电复位、IO/RTC 休眠唤醒复位、RSTn 低电平有效或 WDT 溢出复位时，CBANK SFR 被复位为 0x01，即，此时，根据 A15 的电平，8051 可以访问 Flash 存储器的地址范围 0x0000~0xFFFF。

逻辑地址的 MSB (最高字节) 代表段代码编号；低 2 个字节表示地址空间中的段代码区域 (0x8000~0xFFFF)。

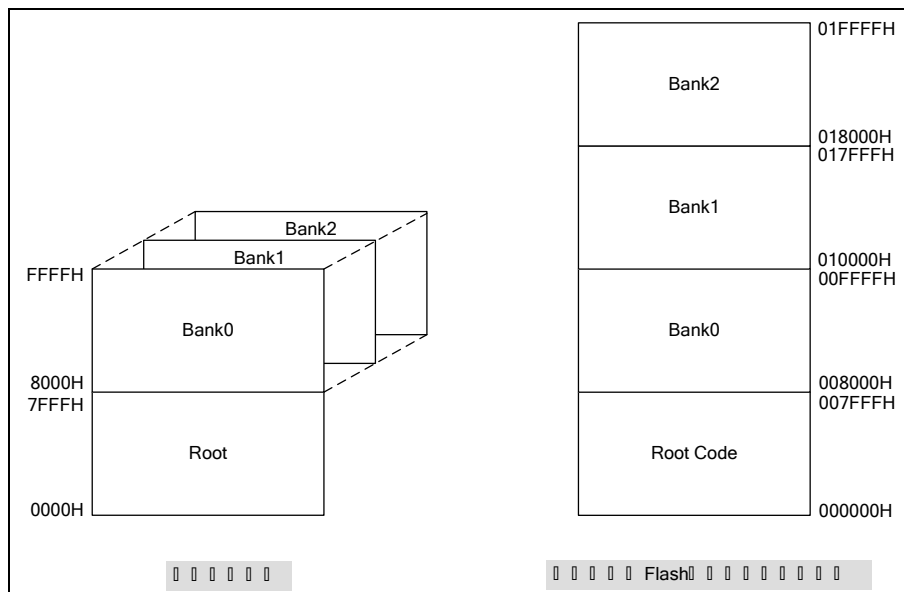


图 1 8051 地址空间划分及各代码段在 Flash 存储器中的物理地址 (V9801)

4. 在 IAR IDE 中建立代码分页的应用程序

1. 点击 **File ->New ->Workspace** 创建一个新 **Workspace** 后，点击 **Project ->Create New Project** 创建一个新的工程，并保存为 V9801Test;

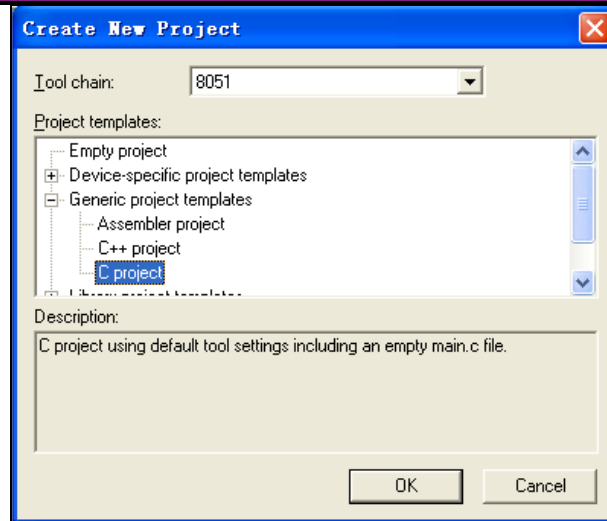


图 2 创建新的 Project

2. 在 Workspace 窗口中，右键点击工程名 **V9801Test-Debug** ->**Add ->Add Group...**；或选择工程名 **V9801Test Debug**，并点击 **Project ->Add Group...**，添加代码组（**Setting** 和 **User**）；
3. 在 Workspace 窗口中，右键点击代码组名 **Setting** ->**Add ->Add Files...**；或选择代码组名 **Setting**，再点击 **Project ->Add Files...**，添加万高科技提供的源文件：

3.1 **cstartup.s51**：万高科技提供的启动代码；

注意：

软件仿真（**Options ->Debugger ->Simulator**）时，应使用 IAR 自带的启动代码 *cstartup.s51*。此时，不需要将该文件添加到工程中。

使用万高科技的硬件（**Options ->Debugger ->Third-Party Driver**）仿真和运行时，应使用万高科技提供的代码分页设置文件 *cstartup.s51*，使用时，仅需将该文件添加到工程中即可。该文件在 IAR 自带的启动代码的基础上做了如下修改：

```
?RESET_CODE_BANK:
MOV    ?CBANK, #0x01
```

3.2 **iar_banked_code_support.s51**、**lnk51ew.xcl**：万高科技提供的代码分页设置文件和链接器命令文件（linker command file），主要用于配置段代码参数，包括段代码个数及段代码选择寄存器的实现方式，可以自行指导编译器实现代码段切换；

注意：**1. 关于代码分页设置文件 *iar_banked_code_support.s51*：**

- 1.1 软件仿真（**Options ->Debugger ->Simulator**）时，应使用 IAR 自带的代码分页设置文件 *iar_banked_code_support.s51*。此时，不需要将该文件添加到工程中。
- 1.2 使用万高科技的硬件（**Options ->Debugger ->Third-Party Driver**）仿真和运行时，应使用万高科技提供的代码分页设置文件 *iar_banked_code_support.s51*，使用时，仅需将该文件添加到工程中即可。该文件在 IAR 自带的代码分页设置文件的基础上做了如下修改：

```
MOV    A, ?CBANK    ; read current PSBANK
CLR C;;add by Vango
SUBB   A, #1
ANL    A, #0xFC     ; mask IFBANK bits
ORL    A, B         ; add new bank number (in B-reg) to PSBANK
```

```
CLR C;  
ADD A, #1 ;add by Vango  
MOV ?CBANK, A ; set new bank
```

2. 关于链接器命令文件.xcl:

万高科技在 IAR 自带的 `lnk51ew.xlc` 的基础上做了如下修改，而且，在文件 `lnk51ew.xcl` 中，将 `link_base.xcl` 的内容页添加进来。

```
-D_CODE_START=0x000000 // First address for code.  
-D_CODE_END=0x7FFF // Last address for code.
```

```
-D_BANK0_START=0x8000  
-D_BANK0_END=0xFFFF
```

```
-D_BANK1_START=0x18000  
-D_BANK1_END=0x1FFFF
```

```
-D_BANK2_START=0x28000  
-D_BANK2_END=0x2FFFF
```

```
-P(CODE)BANK0=_BANK0_START-_BANK0_END  
-P(CODE)BANK1=_BANK1_START-_BANK1_END  
-P(CODE)BANK2=_BANK2_START-_BANK2_END
```

4. 在 Workspace 窗口，右键点击代码组名 **User ->Add ->Add Files...**；或选择代码组名 **User**，再点击 **Project ->Add Files...**，添加万高科技提供的源文件：

4.1 **main.c**: 主程序，公共代码。编写代码时，用户必须使用 `__near_func` 声明公共代码。

4.2 **delay.c**: 包括两个延时函数，分别延时 100ms 和 1ms，公共代码。

4.3 **bank0.c/bank1.c/bank2.c**: 段代码。

在代码分页模式（Banked code model）下，8051 IAR C/C++ 编译器会将所有的代码自动划归为段代码，用户可使用伪指令 `#pragma location = "BANKx"`，提示指令以下的函数应被定位在哪个段代码地址中。比如：

Bank0.c 中，`#pragma location = "BANK0"` 表示函数应被定位在 Bank0 地址中。

Bank1.c 中，`#pragma location = "BANK1"` 表示函数应被定位在 Bank1 地址中。

Bank2.c 中，`#pragma location = "BANK2"` 表示函数应被定位在 Bank2 地址中。

本例程代码太小，为防 IAR 编译器将所有代码自动划归为公共代码，所以，使用 `__banked_func` 强制声明每个段代码。IAR 软件支持自动分 bank。

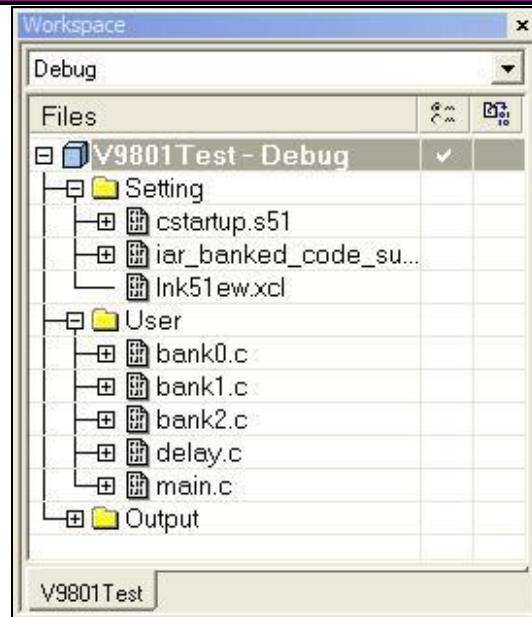


图 3 添加代码组和源文件

5. 右键单击工程名 **V9801Test-Debug** ->**Options**; 或选择工程名 **V9801Test-Debug** 后再点击 **Project -> Options**, 进行工程设置:

5.1 设置 Target 选项卡;

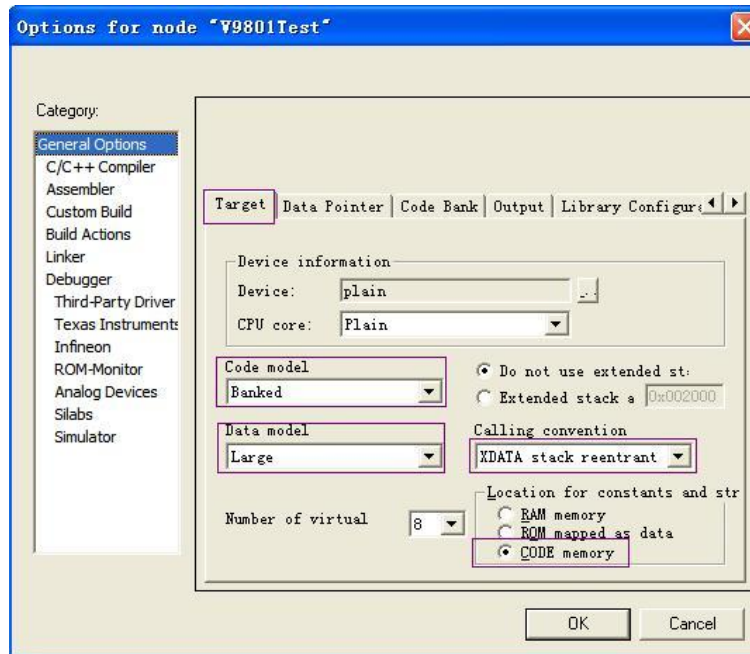


图 4 设置 Target 选项卡

5.2 设置 Code Bank 选项卡;

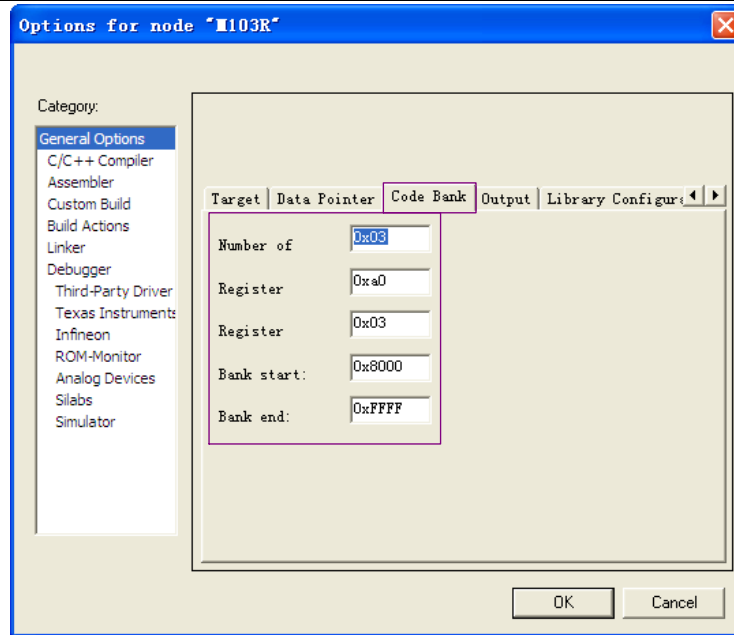


图 5 设置 Code Bank 选项卡

5.3 设置 **Stack/Heap** 选项卡: **Stack sizes** 根据代码使用的栈的情况进行调整; **Heap sizes** 因芯片型号而异。

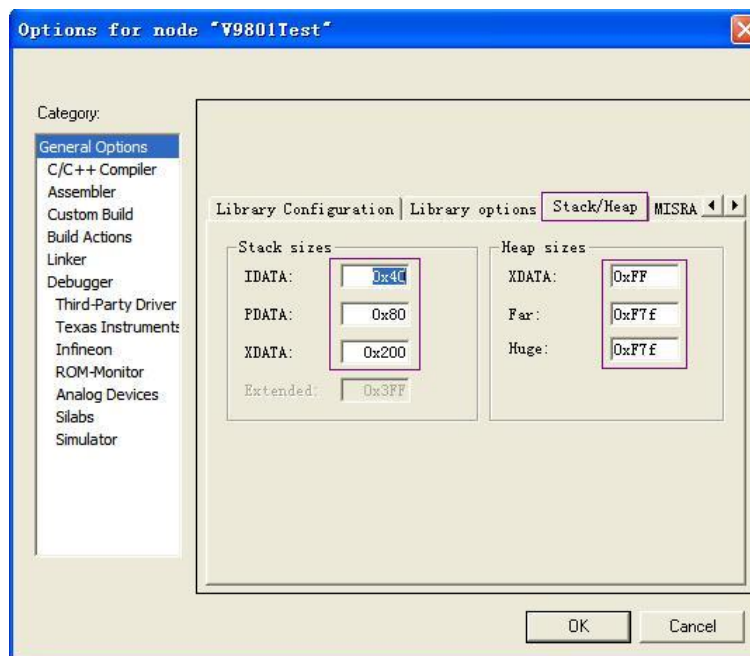


图 6 设置 Stack/Heap 选项卡

5.4 将万高科技提供的 **Inc** 文件夹拷贝至工程所在目录下, 并设置 **C/C++ Compiler**;

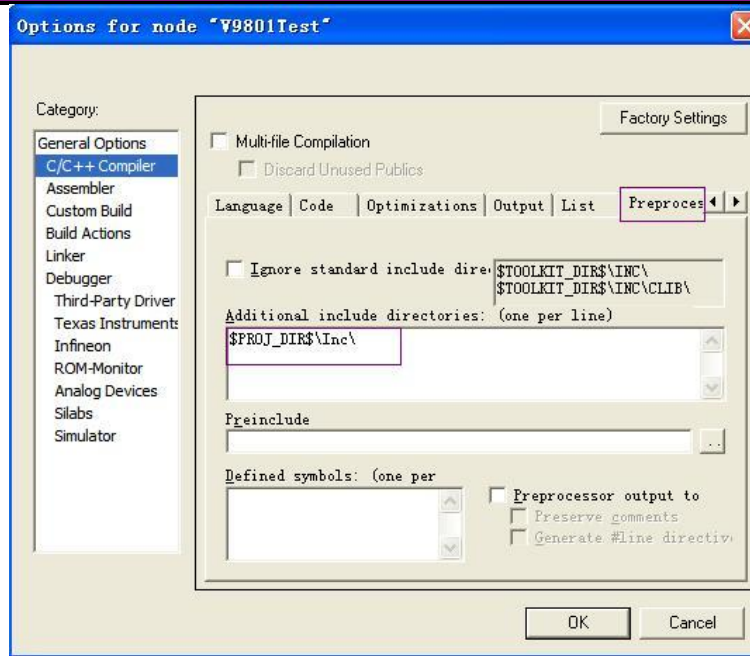


图 7 C/C++ Compiler 选项卡设置

注意:

1. 万高科技提供的 **Inc** 文件夹包含所有与 V9801 相关的头文件;
2. **\$PROJ_DIR\$**: 表示用户建立的工程目录;
3. **\$TOOLKIT_DIR\$**: 安装产生的 MCU 目录, 如 **C:\program files\iar systems\embedded workbench 5.3\8051**。

5.5 如图 8 所示进行 **Assembler->PreProcessor** 选项卡设置;

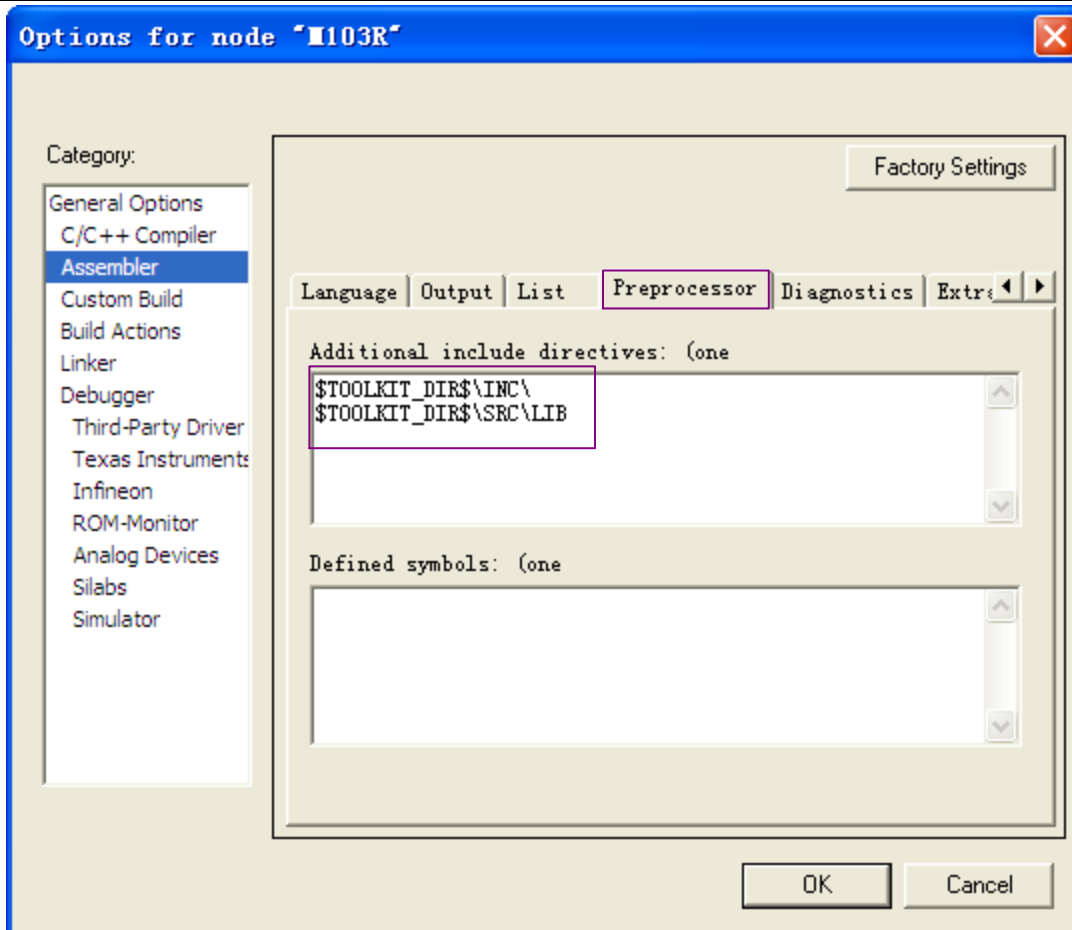


图 8 Assembler->Preprocessor 选项卡设置

如果没有设置，编译会报错，报错内容，如图 9 所示：

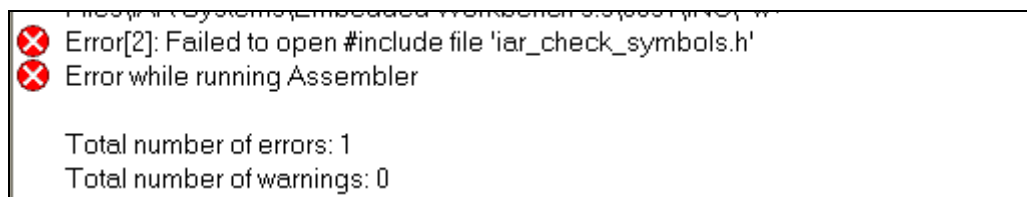


图 9 Assembler->Preprocessor 选项卡未设置，编辑报错内容

5.6 如图 10~12 所示设置 Linker 选项卡；

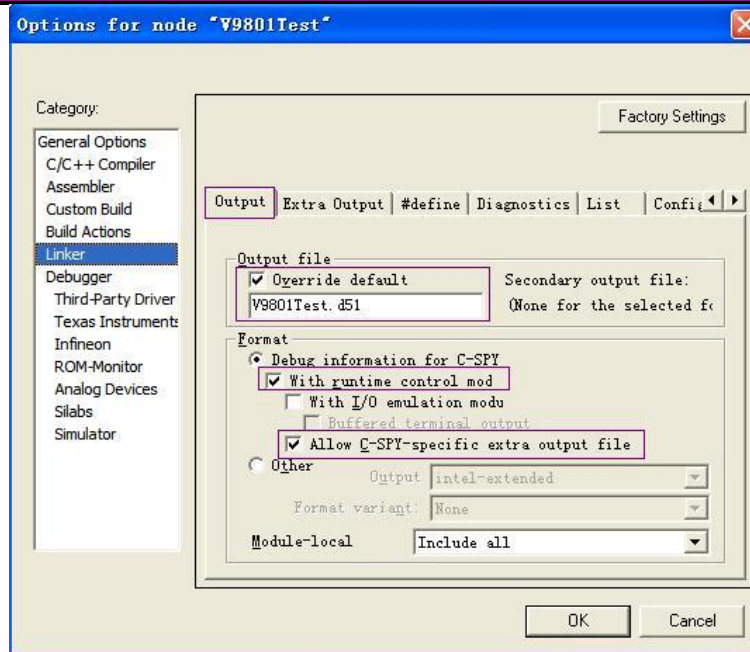


图 3 Linker ->Output 选项卡设置

注意:

1. 勾选 **Debug information for C-SPY** 和 **With runtime control module** 生成一个**.d51** 文件;
2. 勾选 **Allow C-SPY-specific extra output file** 激活 **Extra Output** 选项卡中的选项;
3. 如果不做上述两项选择, 则在调试时, 无法在.c 等工程文件中设置断点。

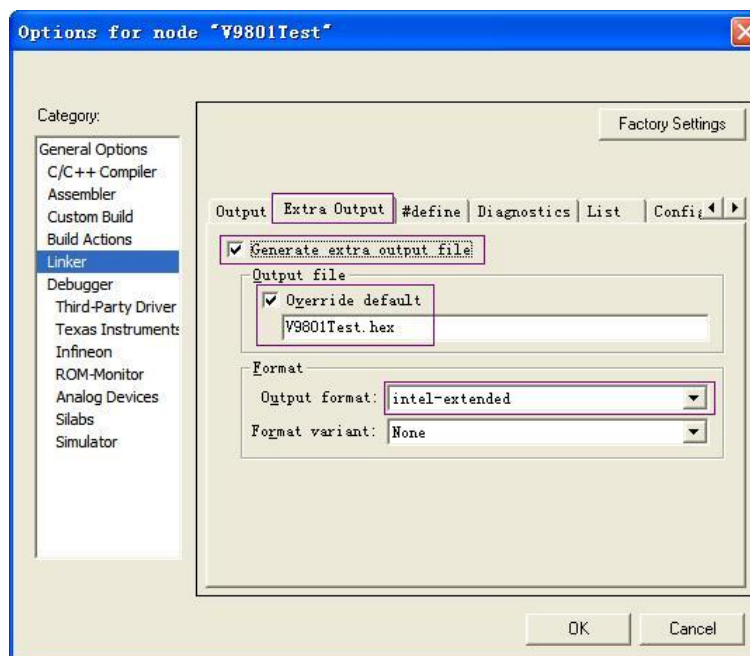


图 4 设置 Linker ->Extra Output 生成 HEX 文件

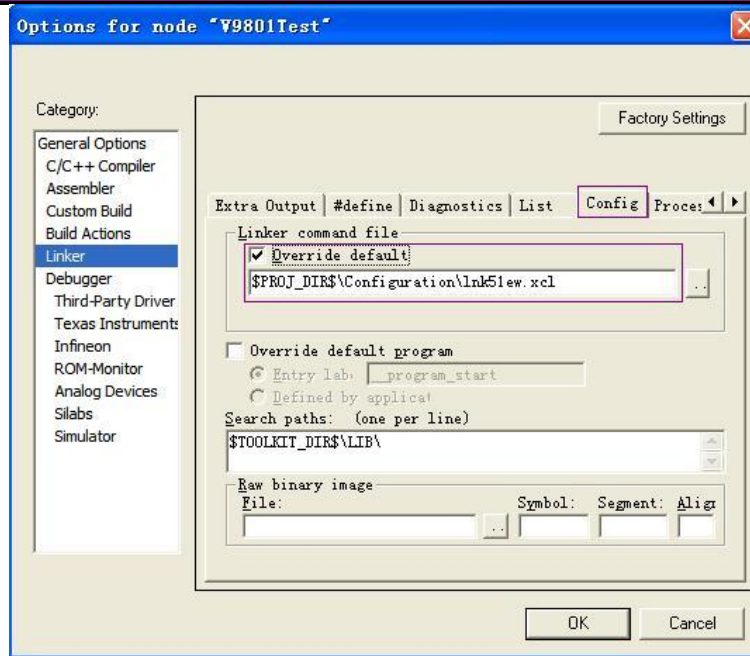


图 5 设置 Linker ->Config 选项卡，选择万高科技提供的链接器命令文件（.xcl）

5.7 按照以下描述进行 **Debugger** 选项卡的设置：

5.7.1 设置 **Setup** 选项卡：

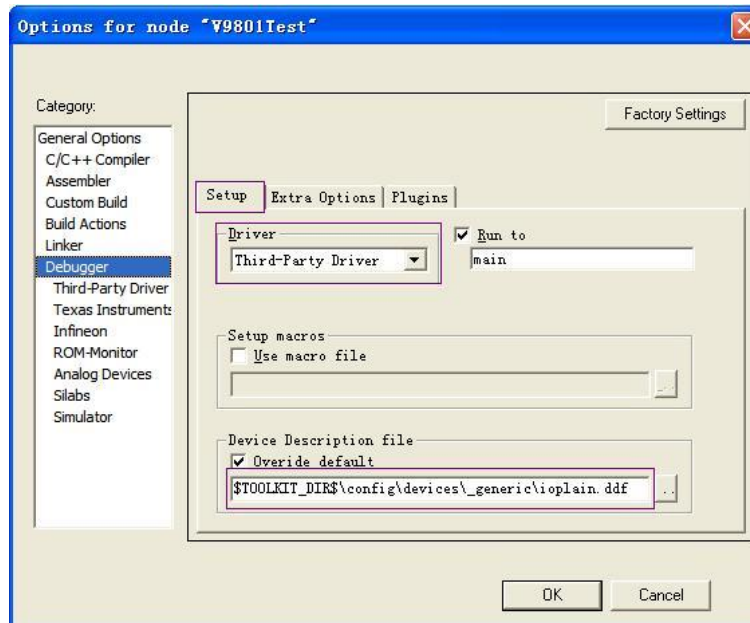


图 6 如果使用 SD502 进行硬件调试/运行，设置 Debugger ->Setup ->Third-Party Driver

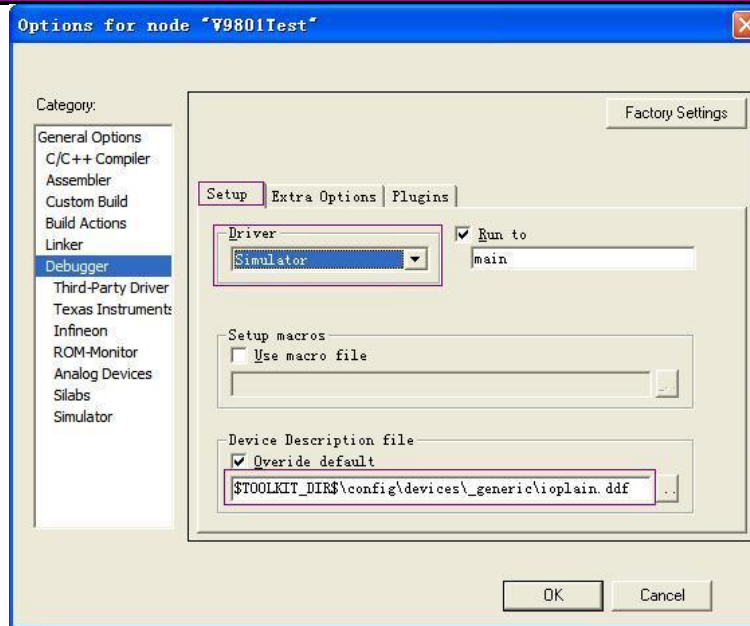


图 7 如果进行软件调试，设置 Debugger ->Setup ->Simulator

注意：在图 7 中，**Device** 设置为 **plain**，所以在图 13 和图 14 所示的 **Device Description file** 中，应选择 **plain** 专用的描述文件 **ioplain.ddf**。

5.7.2 如果使用万高科技提供的工具进行硬件调试/运行，则设置 **Third-Party Driver** 选项卡：应使用万高科技提供的专用 DLL 文件，该 dll 名称见具体的发布文档，该文件必须存放于 IAR 安装目录下的子目录 **\8051\bin** 中。

如果进行软件仿真，那么在 **Simulator** 选项卡中采用默认值。

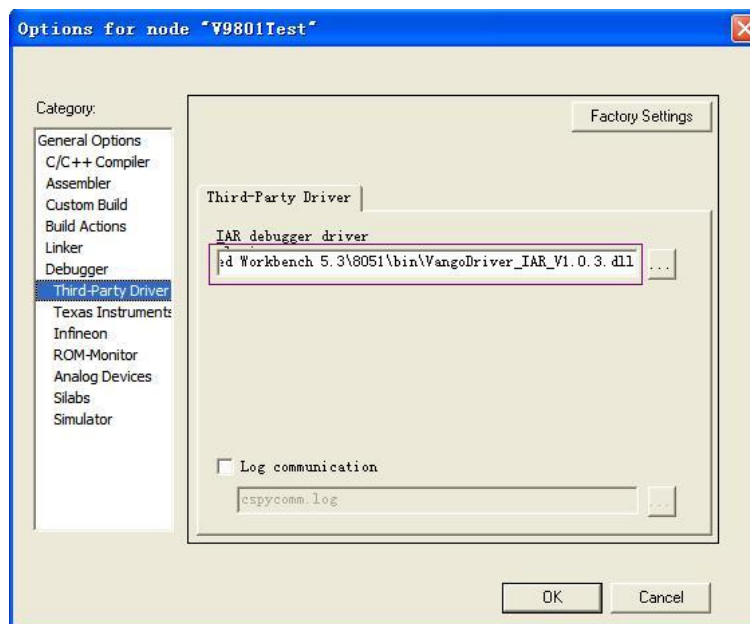



图 8 设置 Third-Party Driver

至此，已完成对工程 **V9801Test-Debug** 的配置。

6. 点击  按钮，对程序进行编译和链接。下方的 **Build** 窗口会显示编译和链接过程。如果错误与警告数量均为 0，则表明工程建立成功，并在工程所在目录的 `\Debug\Exe` 子目录中生成了一个 Hex 文件 **V9801Test.hex**。

或者，在 **Workspace** 窗口，选择工程名 **V9801Test-Debug** 后，再点击 **Project ->Make**，或按 **F7** 键；又或者，右键点击工程名 **V9801Test-Debug ->Make**，完成上述程序编译和链接。

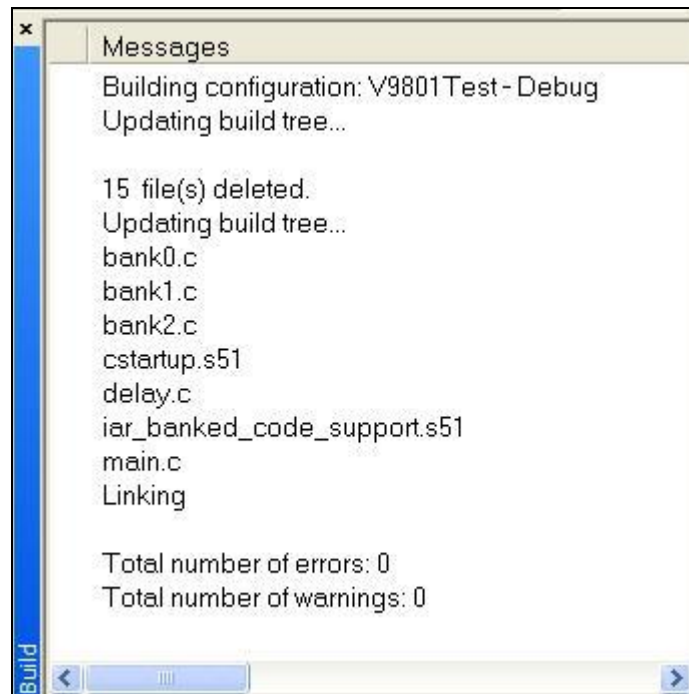


图 9 Build 窗口显示编译和链接过程与结果

5. 调试程序/烧写 Flash

如果使用 SD502 进行硬件调试/运行，则正确连接 PC、SD502 和学习板，然后，点击  按钮，或点击 **Project ->Debug**，或按 **Ctrl+D** 键，将程序烧写入学习板 Flash 中。

如果进行软件仿真，则点击  按钮，或点击 **Project ->Debug**，或按 **Ctrl+D** 键，开始仿真。

在调试过程中，**调试工具栏**被激活，而且，用户也可通过点击 **View** 打开各种有用的调试用窗口。

在**调试状态**下，用户可通过点击 **View ->Disassembly**，**View ->Register**，和 **View ->Memory**，分别调出 **Disassembly**（反汇编）、**Register**（寄存器）和 **Memory**（存储器）察看窗口，通过输入逻辑地址（LogicalCode）或 ROM 地址（Code）访问各代码段，如：

1. 通过逻辑地址（LogicalCode）访问 Bank1：先在 **Toggle memory zone** 处选择 **LogicalCode**，再在 **Go to** 处输入逻辑地址 0x018000；

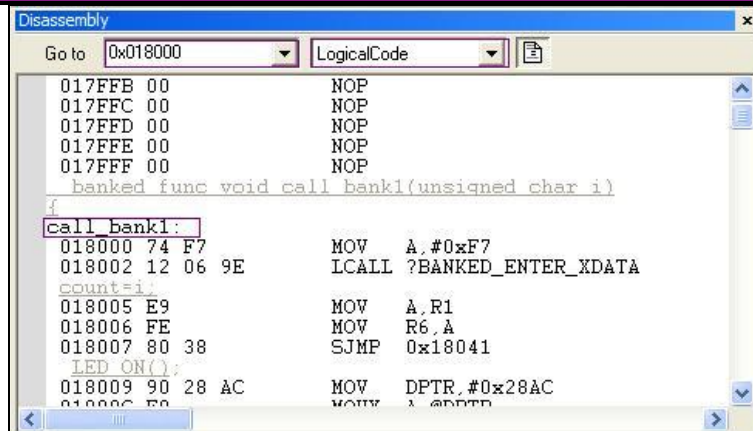


图 10 在 Disassembly 窗口通过逻辑地址访问 Bank1

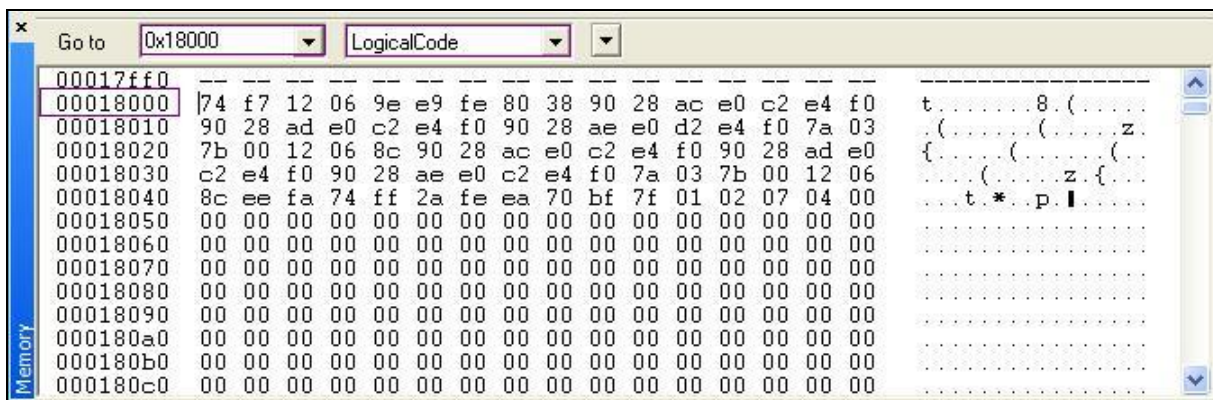


图 11 在 Memory 窗口通过 LogicalCode 访问 Bank1

2. 通过 ROM 地址 (Code) 访问 Bank1 时, 用户应:

2.1 在 **Disassembly** 窗口或 **Memory** 窗口的 **Toggle memory zone** 处选择 **Code**;

2.2 在上述两个窗口的 **Go to** 处输入 ROM 地址 0x8000;

2.3 在 **Register** 窗口配置 ?CBANK, 从而实现各代码段之间的跳转。如表 1 所示, 当 ?CBANK=0x01 时, 选择 Bank1。

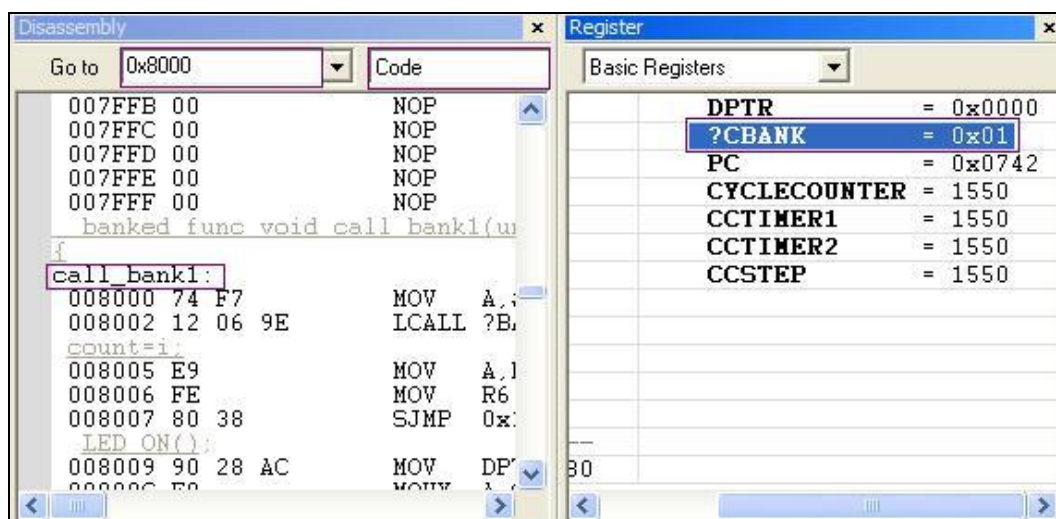


图 12 通过 Code 访问 Bank1 (Disassembly 窗口显示)

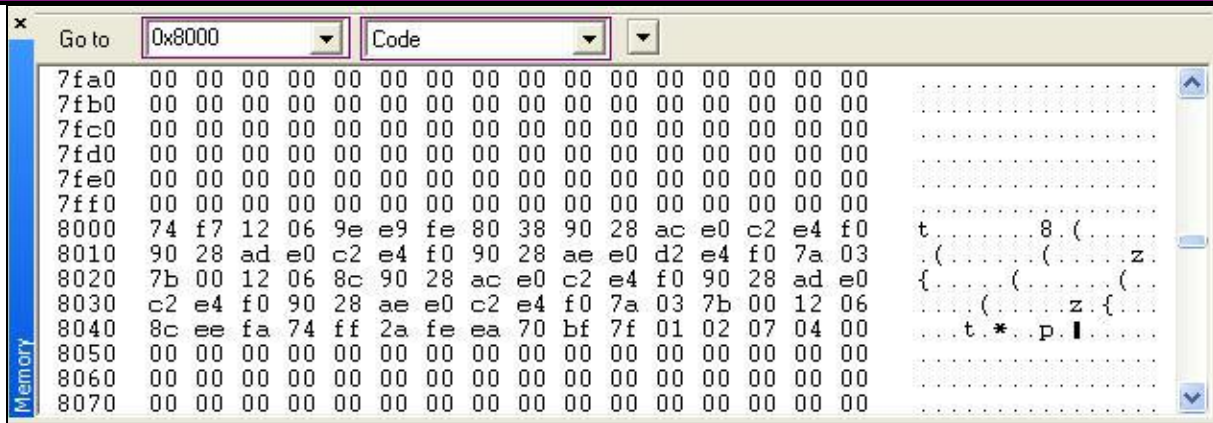


图 13 通过 Code 访问 Bank1（Memory 窗口显示）

6. 验证

程序烧写完成后，复位学习板，并观察学习板上的 LED 指示灯。如果观察到如图 21 所示的现象，表明程序下载成功。

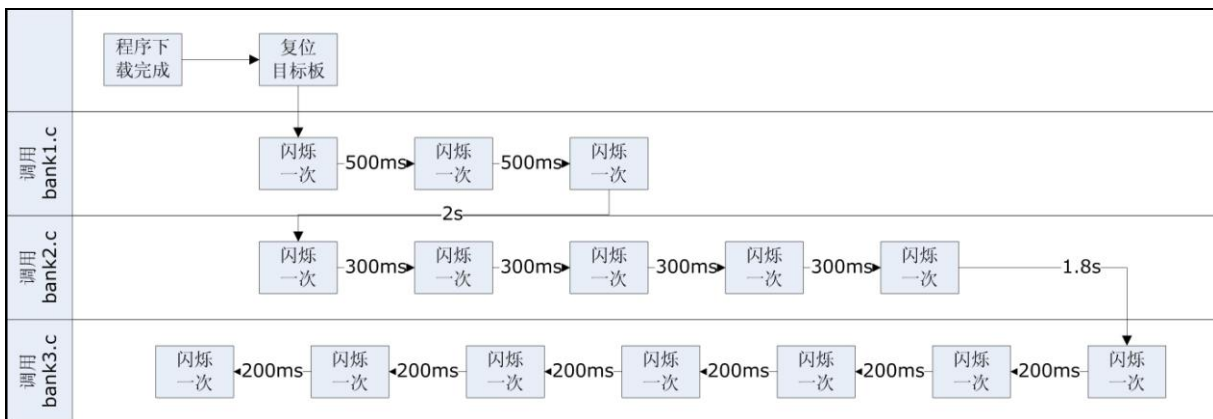



图 14 验证现象

7. 注意

- 调试代码时，仅能在 V9801 中设置最多 3 个硬件断点，所以必须时刻注意断点的占用量。以下几种设置都会占用一个硬件断点：
 - 1.1 选择 **Project ->Options ->Debugger ->Setup -> Run to;**
 - 1.2 选择 **Project ->Options ->Debugger -> Plugins -> Stack;**
 - 1.3 选择 **Project ->Options ->Linker ->Output ->Debug information for C-SPY ->With runtime control module ->With I/O emulation module;**
 - 1.4 点击  或  按钮，或按 **F10** 键或 **F11** 键。
- 开始调试代码时，应该先启动 C-SPY，再设置断点。调试结束后，应先清除所有断点，再退出 C-SPY，以免下次启动 C-SPY 时报错“没有足够可用的断点”。

3. 在调试过程中，当遇见函数 `SetPLL(SETPLL_6_4M);` 时，不可点击  按钮，或按 **F11** 键进入此函数调试。因为在切换 PLL 过程中，芯片速度会下降，此时若使用单步调试会失去调试连接。
4. 如果创建一个不需要进行代码分页的 V9801 应用程序，请参照 **V9811Test_IAR_学习板工程说明.pdf** 进行设置。

版本修订记录

日期	版本	说明
2013-09-19	V3.0	将仿真器改为 SD502
2012-08-31	V2.0	更新文档结构/格式
2012-06-29	V1.0	

杭州万高科技有限公司保留对本文档所涉及的产品及相关的技术信息进行补正或更新的权利。使用本文档时，请您从我们的销售渠道或登录公司网站 <http://www.vangotech.com> 获取最新信息。