Keil 下调试 9200

——by Team Mcuzone

Keil 对很多搞过 8051 的工程师而言都是很熟悉的，Keil 的 C51 编译器可谓是业内公认的标准，经过几年的发展，Keil 的 ARM 编译器也已经很成熟，特别是 Keil 被 ARM 公司收购后，做为 ARM 公司旗下公司，Keil 的认可度更被业界接受。一开始 Keil for ARM 主要支持 ARM7，但是现在 Keil for ARM 也支持了不少型号的 ARM9 了，比如 2410，9200，下面我们就大致的讲述一下用 Keil 调试 9200 的简要步骤。

首先请安装好 Keil，评估版本的软件可以在 www.keil.com 或者 www.mcuzone.com或者www.atarm.com下载到，并连接好ULINK和 9200 目标板。

然后找到一个 AT91RM9200-EK 的路径，如下图：

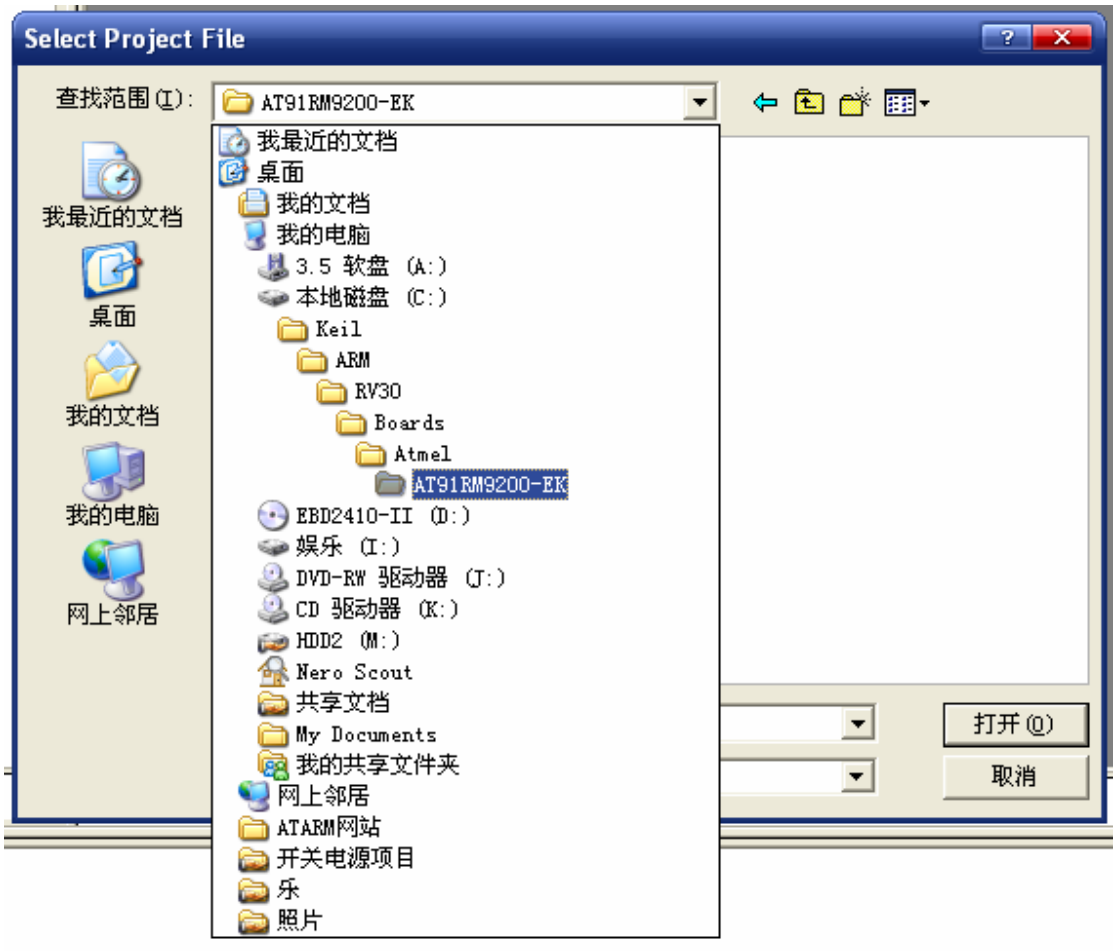

图 1

该目录下有两个范例：

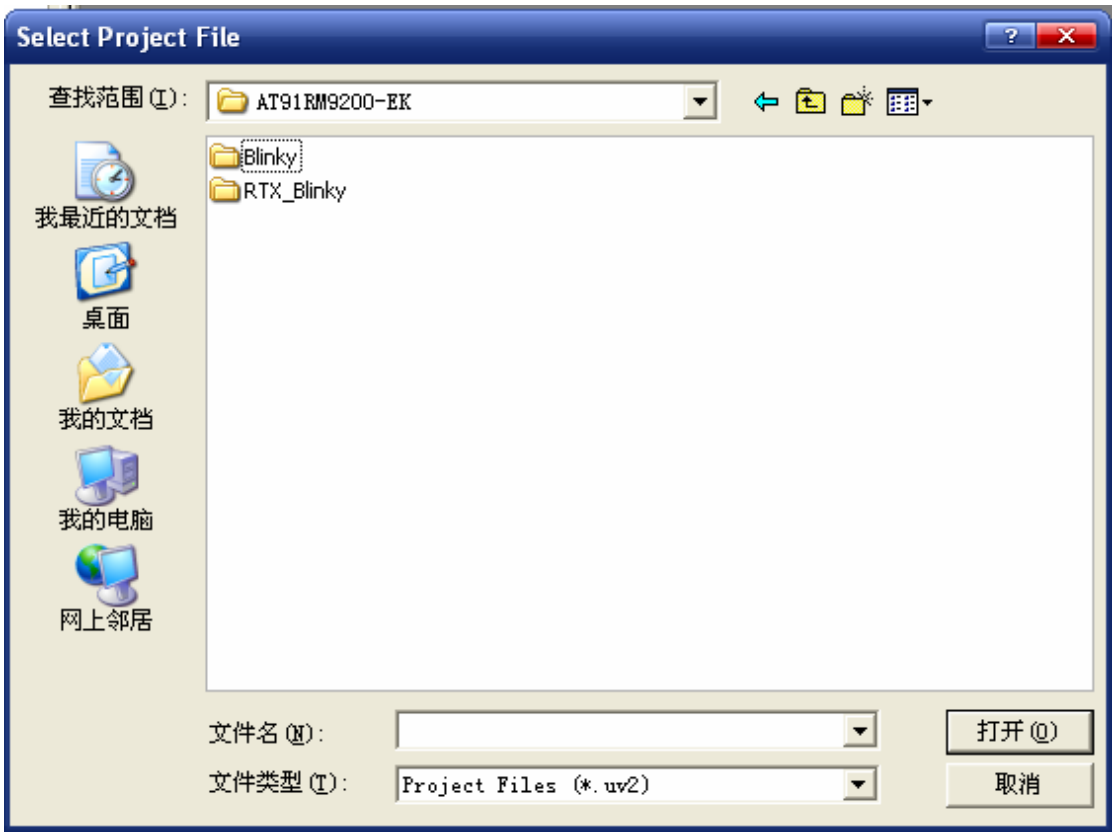

图 2

这里，我们以 Blinky 为例，双击 Blinky.uv2 打开 Blinky 范例。

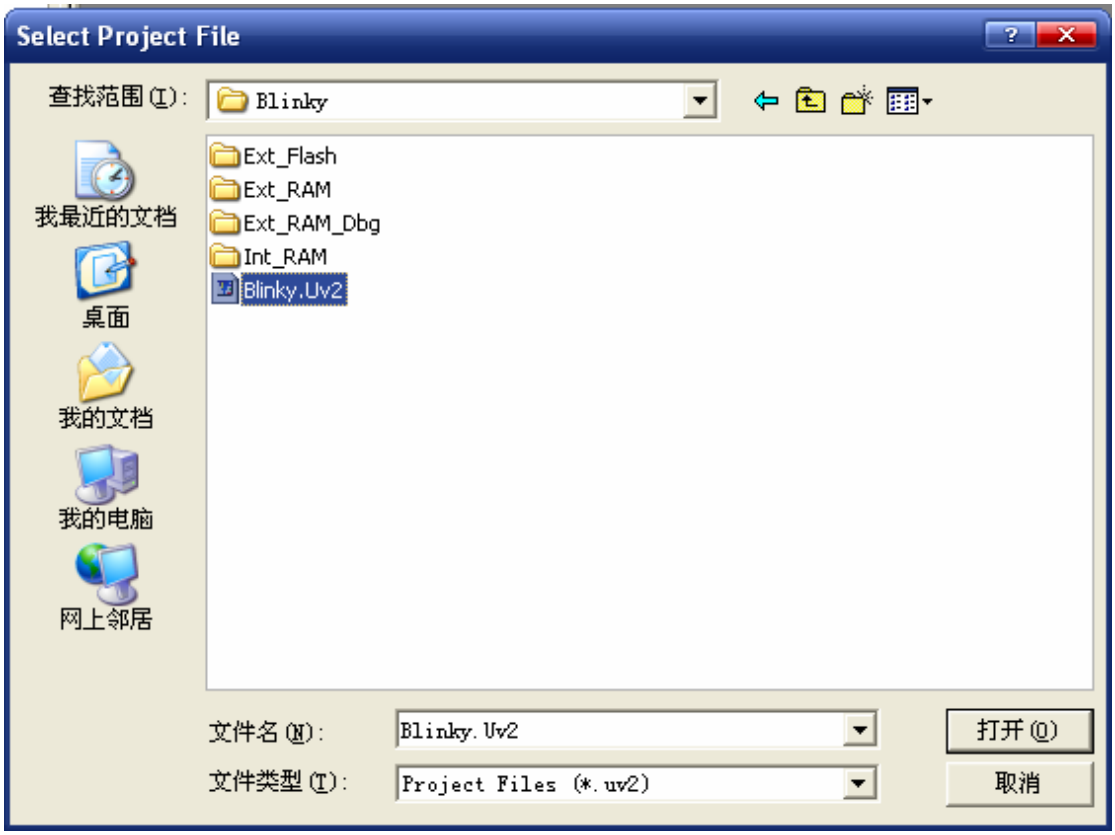

图 3

工程打开后如下所示：

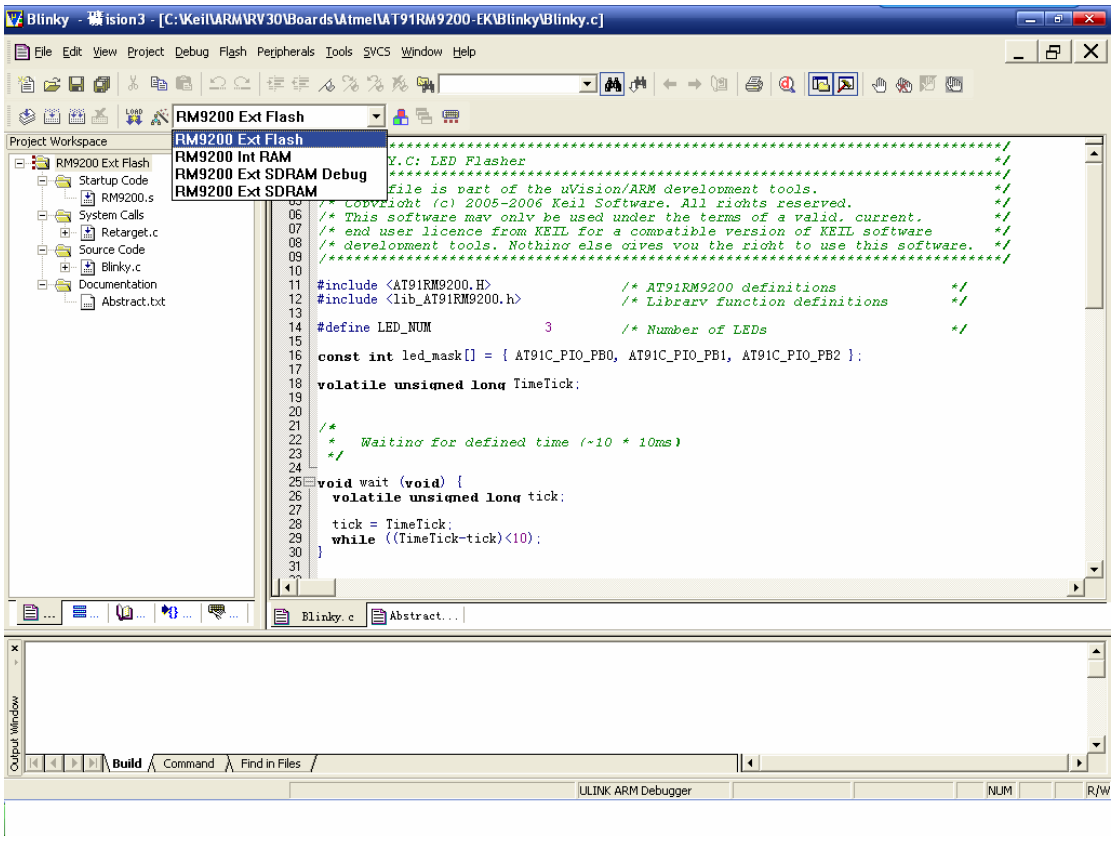

图 4

可以看到有很多种执行方式，我们先选片内 SRAM 进行调试。

“

The Blinky program is available for different targets:

RM9200 Ext Flash: configured for external Flash
(used for production or target debugging)

RM9200 Int RAM: configured for on-chip RAM
(may be used for target debugging)

RM9200 Ext SDRAM Debug: configured for external SDRAM
(may be used for target debugging)

RM9200 Ext SDRAM: configured for code to be in external Flash
and copy itself to external SDRAM and run from
there
(may be used for target debugging or
production)

”

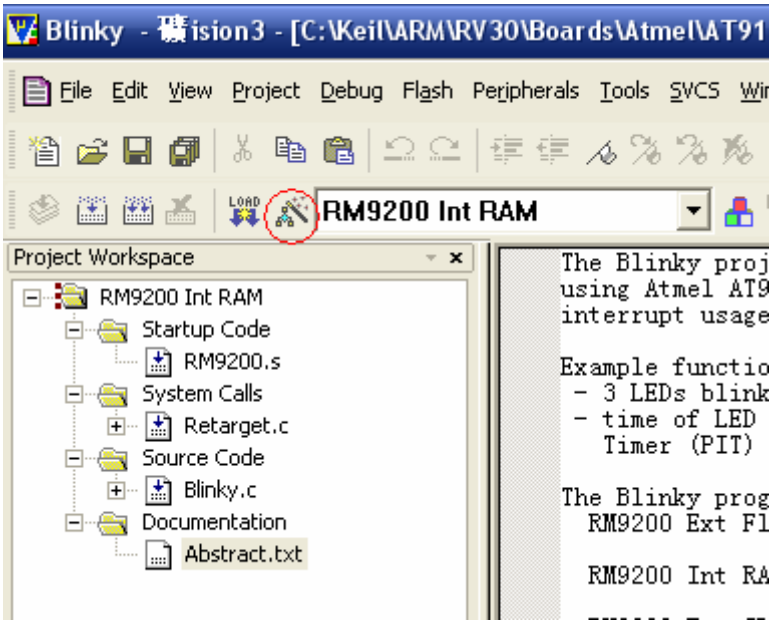

图 5

点击图 5 做红色圈内的按钮（Options for Project）进行设置。

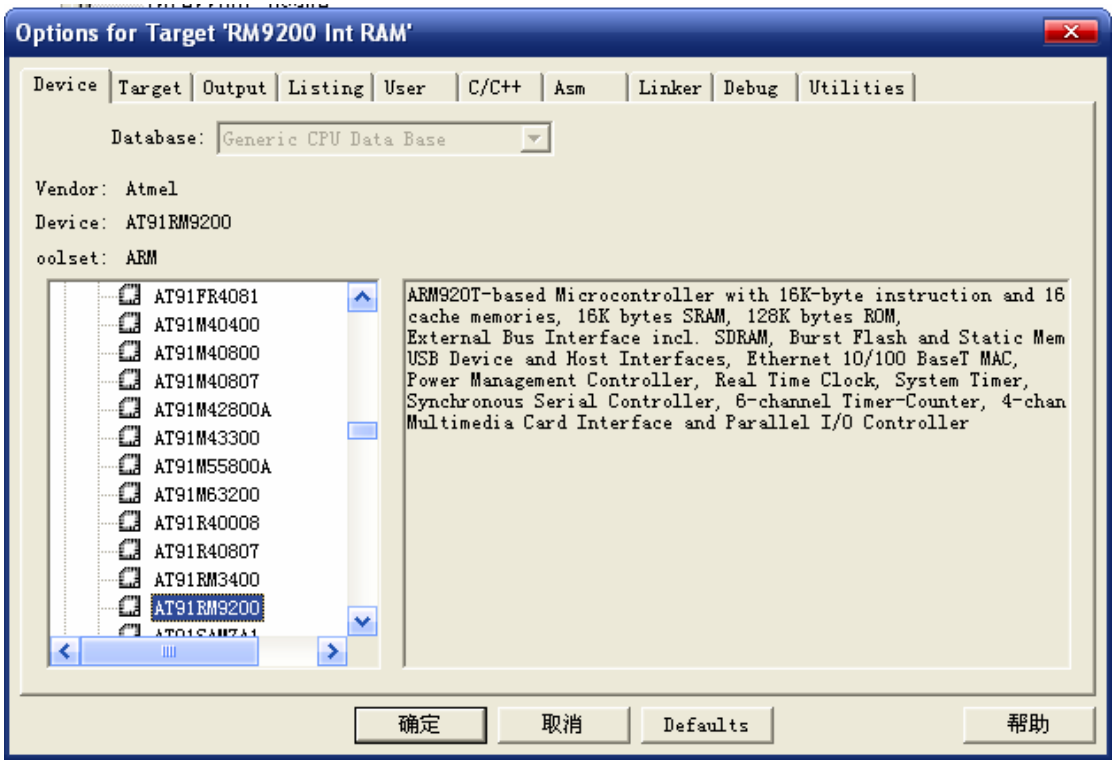

图 6

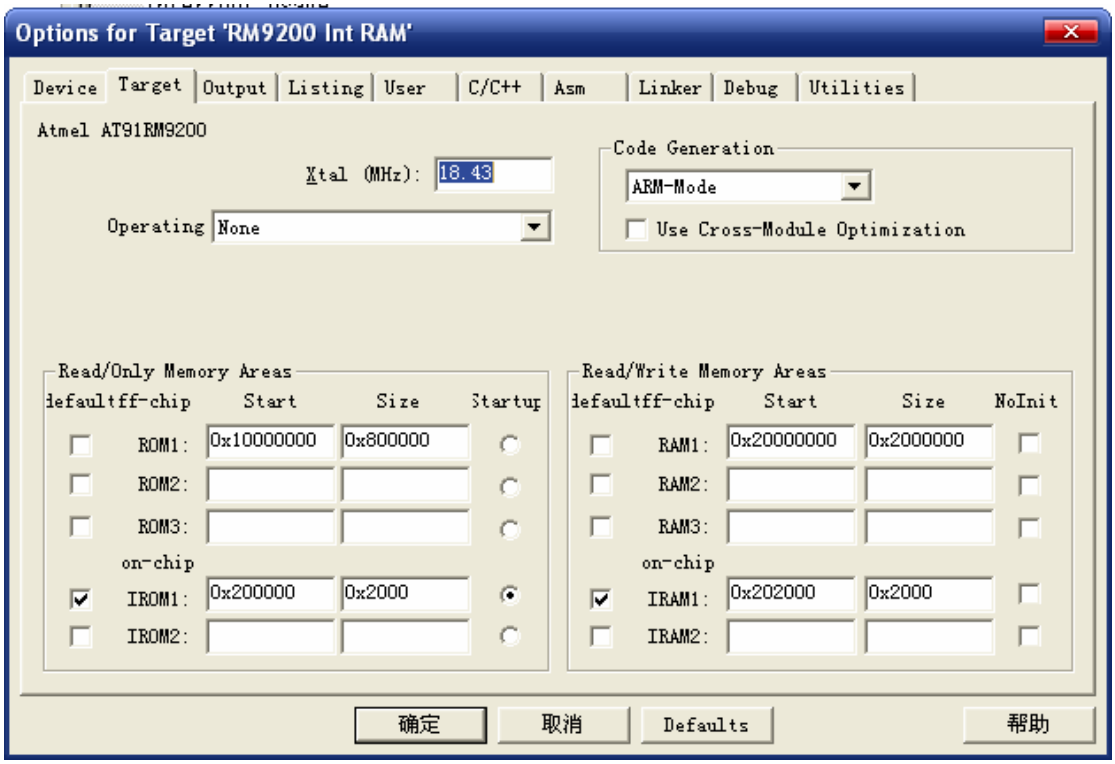
Options for Target 'RM9200 Int RAM'
Device
Target
Output
Listing
User
C/C++
Asm
Linker
Debug
Utilities
Atmel AT91RM9200
Xtal (MHz): 18.43
Operating None
Code Generation
ARM-Mode
Use Cross-Module Optimization
Read/Only Memory Areas
Read/Write Memory Areas
Start
Size
Startup
NoInit
ROM1: 0x10000000 0x800000
ROM2:
ROM3:
on-chip
IROM1: 0x200000 0x2000
IROM2:
RAM1: 0x20000000 0x2000000
RAM2:
RAM3:
IRAM1: 0x202000 0x2000
IRAM2:
确定
取消
Defaults
帮助

图 7

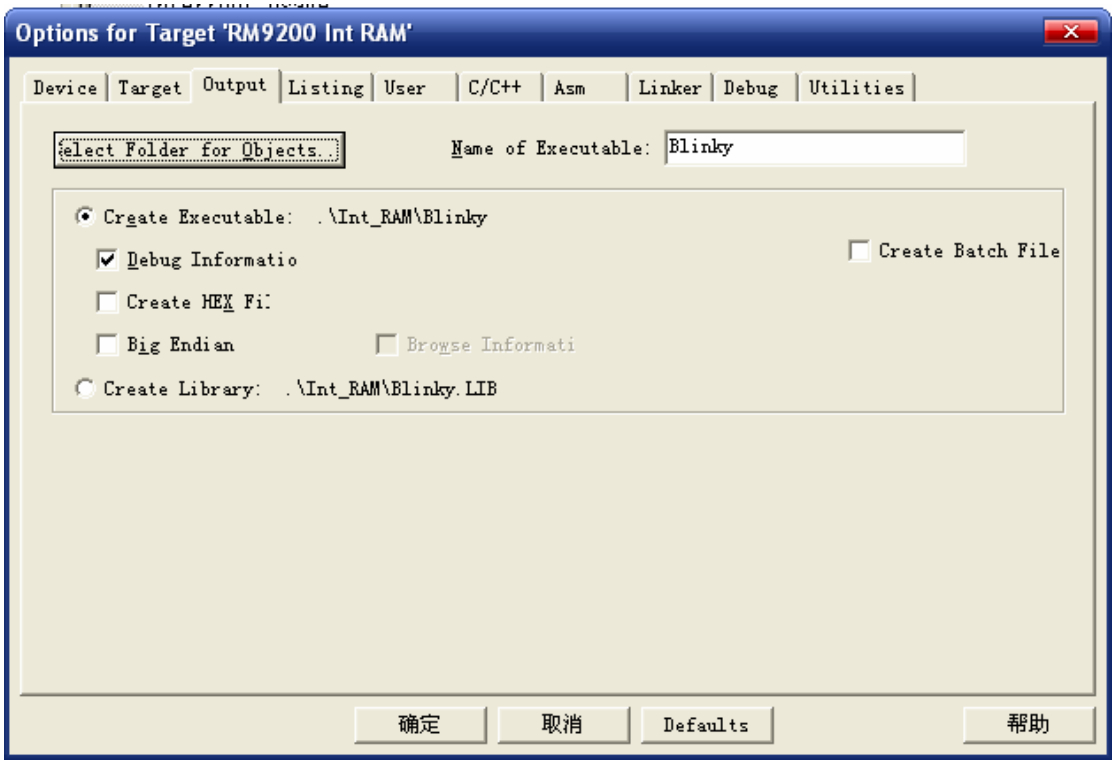
Options for Target 'RM9200 Int RAM'
Device
Target
Output
Listing
User
C/C++
Asm
Linker
Debug
Utilities
elect Folder for Objects...
Name of Executable: Blinky
Create Executable: .\Int_RAM\Blinky
Debug Informatio
Create HEX Fi:
Big Endian
Browse Informati
Create Batch File
Create Library: .\Int_RAM\Blinky.LIB
确定
取消
Defaults
帮助

图 8

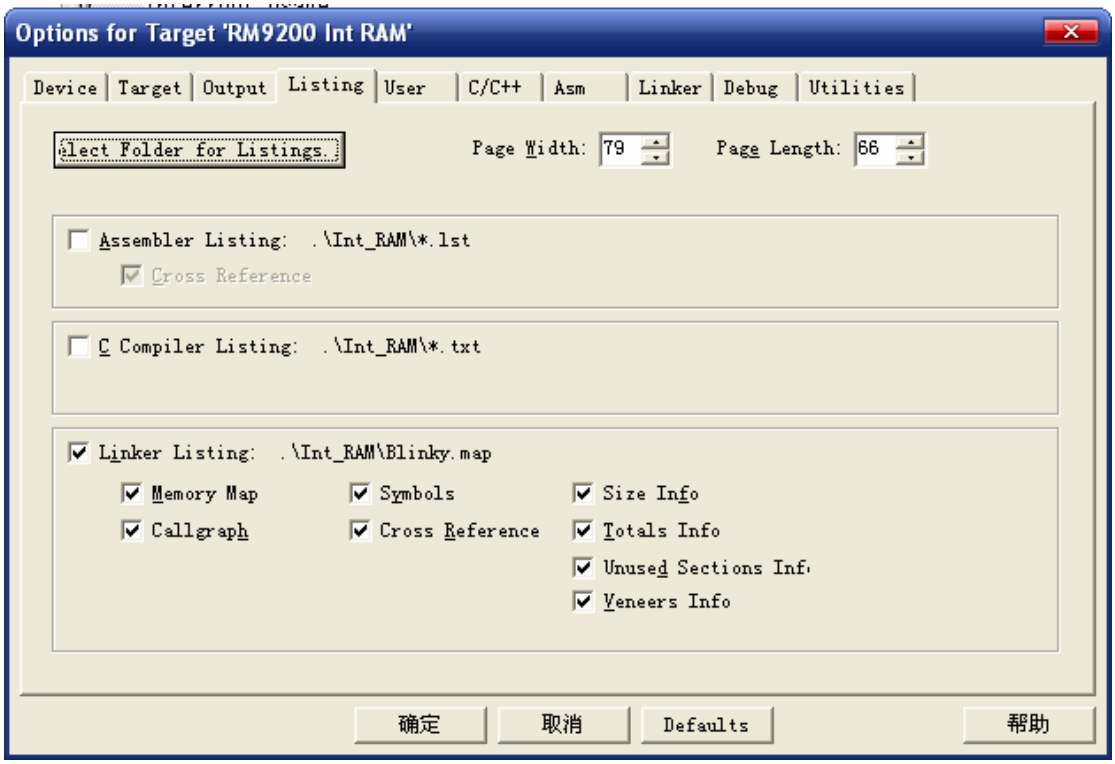
Options for Target 'RM9200 Int RAM'
Device
Target
Output
Listing
User
C/C++
Asm
Linker
Debug
Utilities
elect Folder for Listings...
Page Width: 79
Page Length: 66
Assembler Listing: .\Int_RAM*.lst
Cross Reference
C Compiler Listing: .\Int_RAM*.txt
Linker Listing: .\Int_RAM\Blinky.map
Memory Map
Symbols
Size Info
Callgraph
Cross Reference
Totals Info
Unused Sections Inf
Veneers Info
确定
取消
Defaults
帮助

图 9

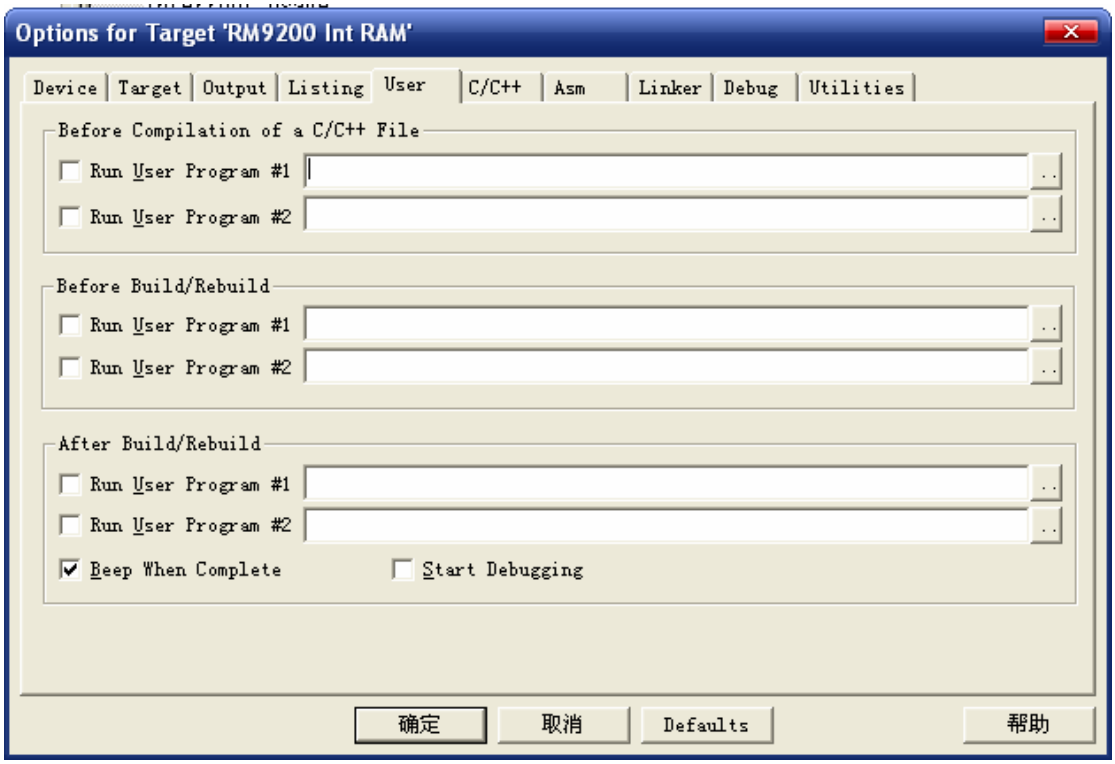
Options for Target 'RM9200 Int RAM'
Device
Target
Output
Listing
User
C/C++
Asm
Linker
Debug
Utilities
Before Compilation of a C/C++ File
Run User Program #1
Run User Program #2
Before Build/Rebuild
Run User Program #1
Run User Program #2
After Build/Rebuild
Run User Program #1
Run User Program #2
Beep When Complete
Start Debugging
确定
取消
Defaults
帮助

图 10

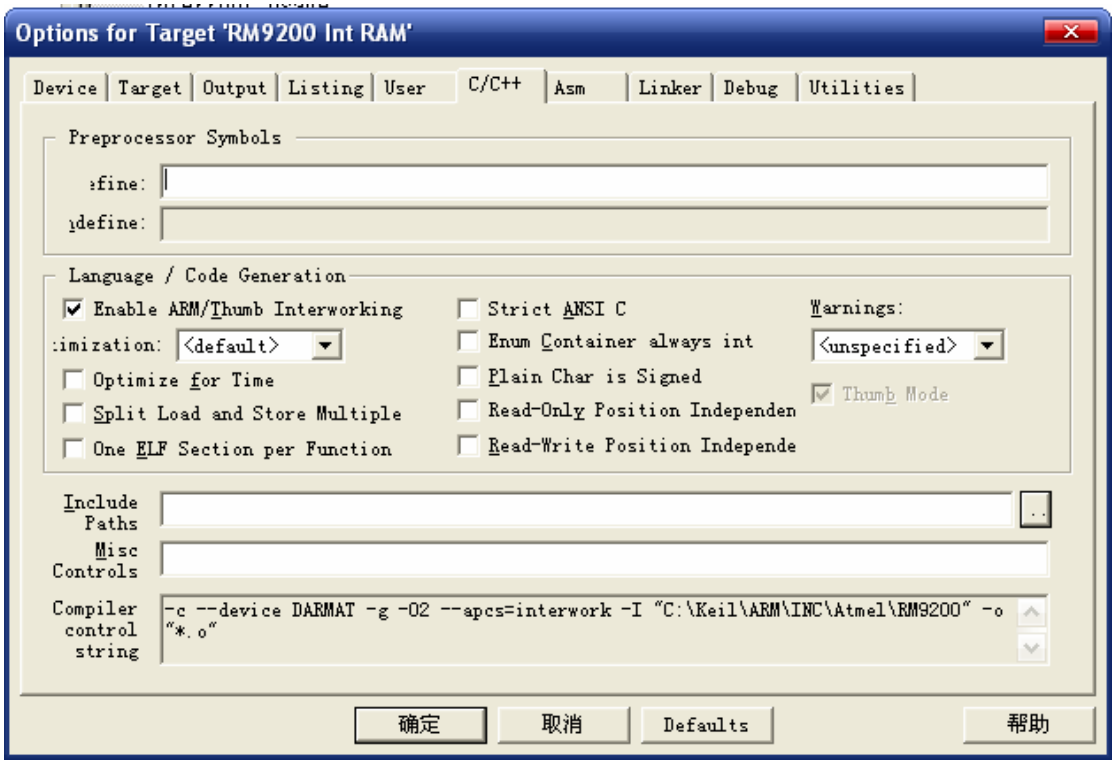
Options for Target 'RM9200 Int RAM'
Device
Target
Output
Listing
User
C/C++
Asm
Linker
Debug
Utilities
Preprocessor Symbols
Define:
Undefine:
Language / Code Generation
Enable ARM/Thumb Interworking
Strict ANSI C
Warnings:
Optimization:
<default>
Enum Container always int
<unspecified>
Optimize for Time
Plain Char is Signed
Thumb Mode
Split Load and Store Multiple
Read-Only Position Independen
One ELF Section per Function
Read-Write Position Independe
Include Paths
Misc Controls
Compiler control string
-c --device DARMAT -g -O2 --apcs=interwork -I "C:\Keil\ARM\INC\Atmel\RM9200" -o "*.o"
确定
取消
Defaults
帮助

图 11

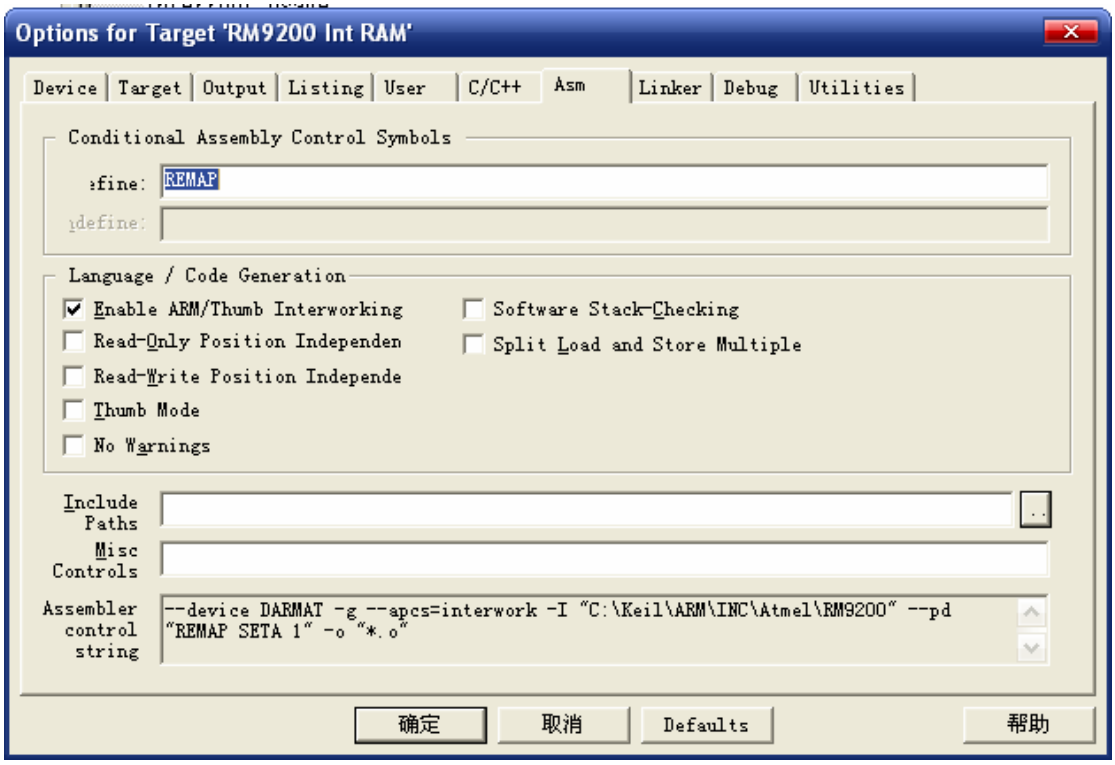
Options for Target 'RM9200 Int RAM'
Device
Target
Output
Listing
User
C/C++
Asm
Linker
Debug
Utilities
Conditional Assembly Control Symbols
Define:
REMAP
Undefine:
Language / Code Generation
Enable ARM/Thumb Interworking
Software Stack-Checking
Read-Only Position Independen
Split Load and Store Multiple
Read-Write Position Independe
Thumb Mode
No Warnings
Include Paths
Misc Controls
Assembler control string
--device DARMAT -g --apcs=interwork -I "C:\Keil\ARM\INC\Atmel\RM9200" --pd "REMAP SETA 1" -o "*.o"
确定
取消
Defaults
帮助

图 12

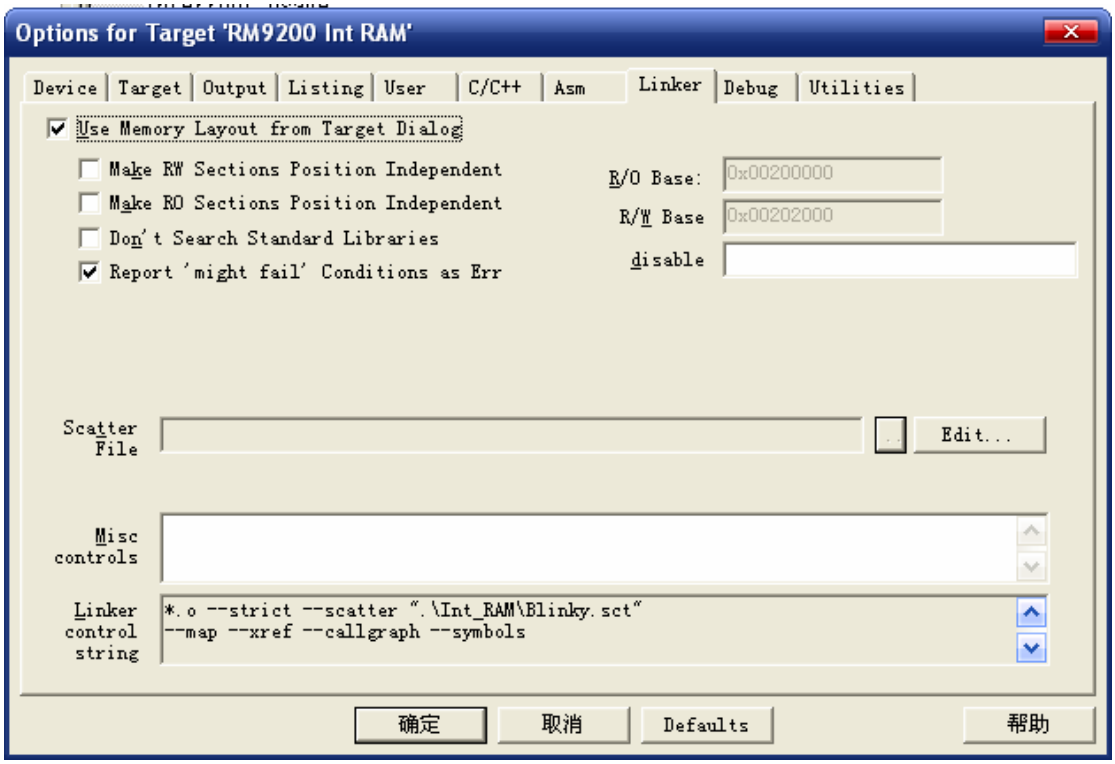
Options for Target 'RM9200 Int RAM'
Device
Target
Output
Listing
User
C/C++
Asm
Linker
Debug
Utilities
Use Memory Layout from Target Dialog
Make RW Sections Position Independent
Make RO Sections Position Independent
Don't Search Standard Libraries
Report 'might fail' Conditions as Err
R/O Base:
0x00200000
R/W Base
0x00202000
disable
Scatter File
Edit...
Misc controls
Linker control string
*.o --strict --scatter ".\Int_RAM\Blinky.sct"
--map --xref --callgraph --symbols
确定
取消
Defaults
帮助

图 13

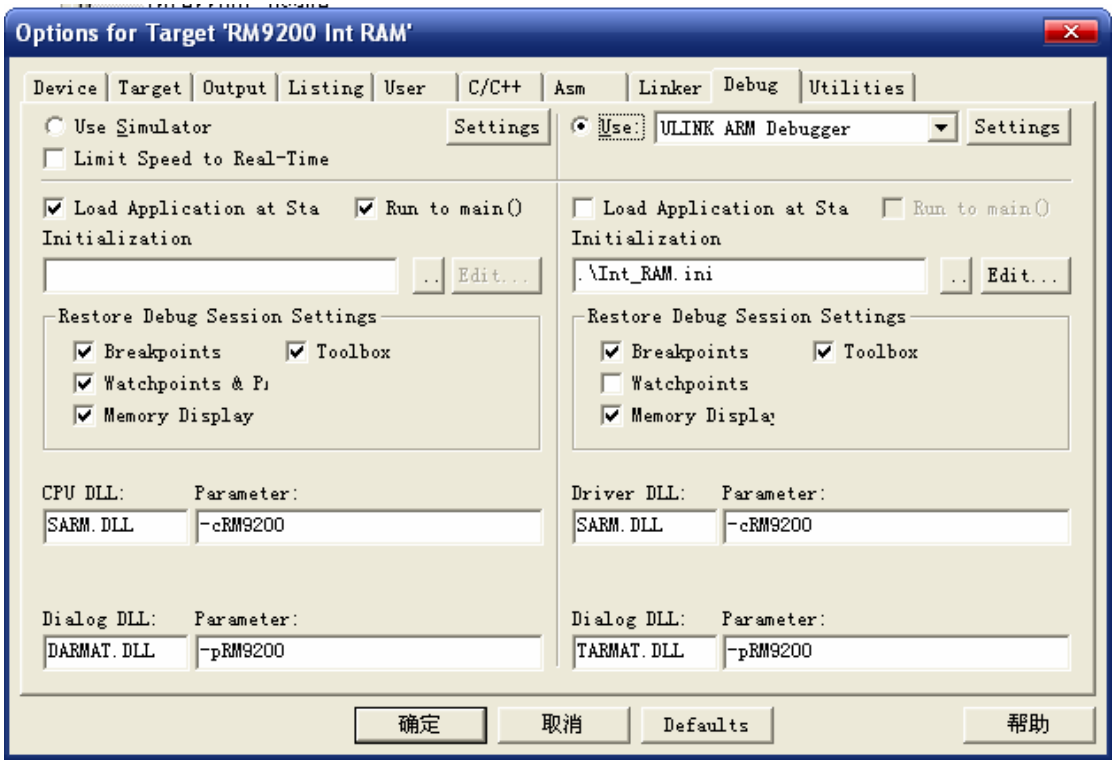
Options for Target 'RM9200 Int RAM'
Device
Target
Output
Listing
User
C/C++
Asm
Linker
Debug
Utilities
Use Simulator
Settings
Limit Speed to Real-Time
Use:
ULINK ARM Debugger
Settings
Load Application at Sta
Run to main()
Initialization
Edit...
Restore Debug Session Settings
Breakpoints
Toolbox
Watchpoints & P
Memory Display
Load Application at Sta
Run to main()
Initialization
.\Int_RAM.ini
Edit...
Restore Debug Session Settings
Breakpoints
Toolbox
Watchpoints
Memory Displa
CPU DLL:
Parameter:
SARM.DLL
-cRM9200
Driver DLL:
Parameter:
SARM.DLL
-cRM9200
Dialog DLL:
Parameter:
DARMAT.DLL
-pRM9200
Dialog DLL:
Parameter:
TARMAT.DLL
-pRM9200
确定
取消
Defaults
帮助

图 14

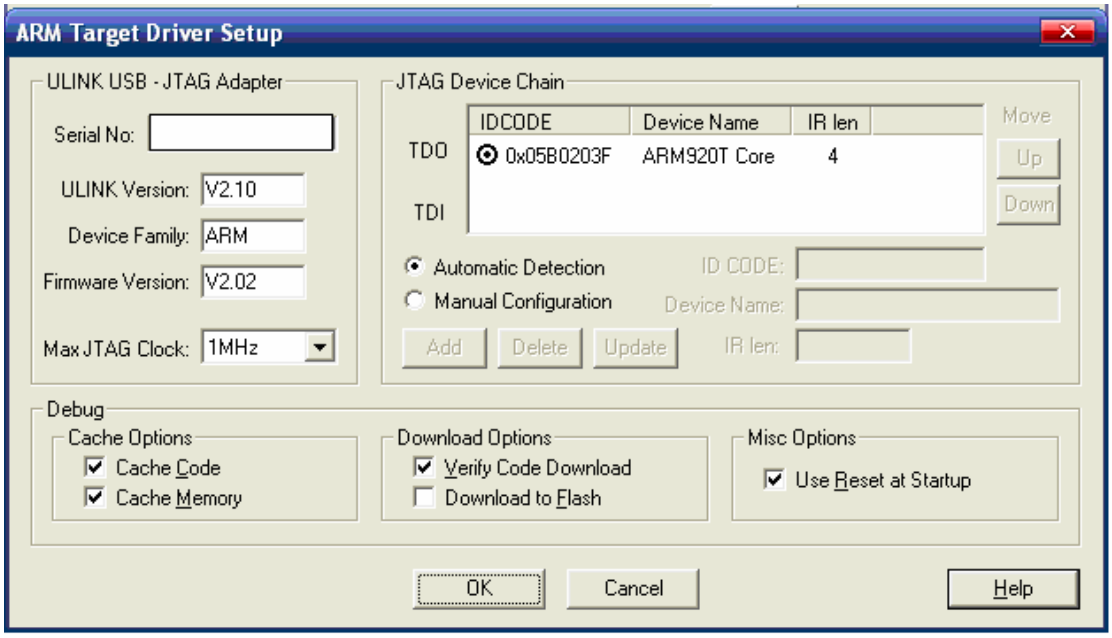
ARM Target Driver Setup
ULINK USB - JTAG Adapter
Serial No:
ULINK Version: V2.10
Device Family: ARM
Firmware Version: V2.02
Max JTAG Clock: 1MHz
JTAG Device Chain
TDO
TDI
IDCODE
Device Name
IR len
0x05B0203F
ARM920T Core
4
Move
Up
Down
Automatic Detection
Manual Configuration
ID CODE:
Device Name:
IR len:
Add
Delete
Update
Debug
Cache Options
Cache Code
Cache Memory
Download Options
Verify Code Download
Download to Flash
Misc Options
Use Reset at Startup
OK
Cancel
Help

图 15

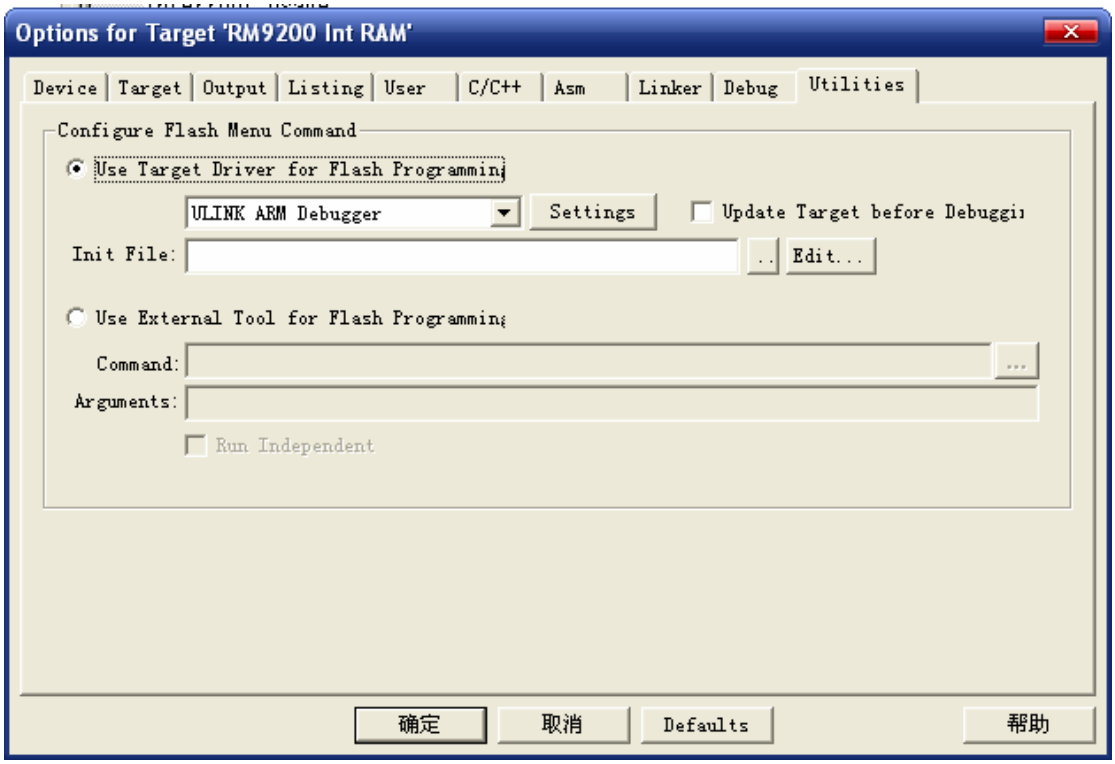
Options for Target 'RM9200 Int RAM'
Device
Target
Output
Listing
User
C/C++
Asm
Linker
Debug
Utilities
Configure Flash Menu Command
Use Target Driver for Flash Programmin
ULINK ARM Debugger
Settings
Update Target before Debuggi
Init File:
Edit...
Use External Tool for Flash Programmin
Command:
Arguments:
Run Independent
确定
取消
Defaults
帮助

图 16

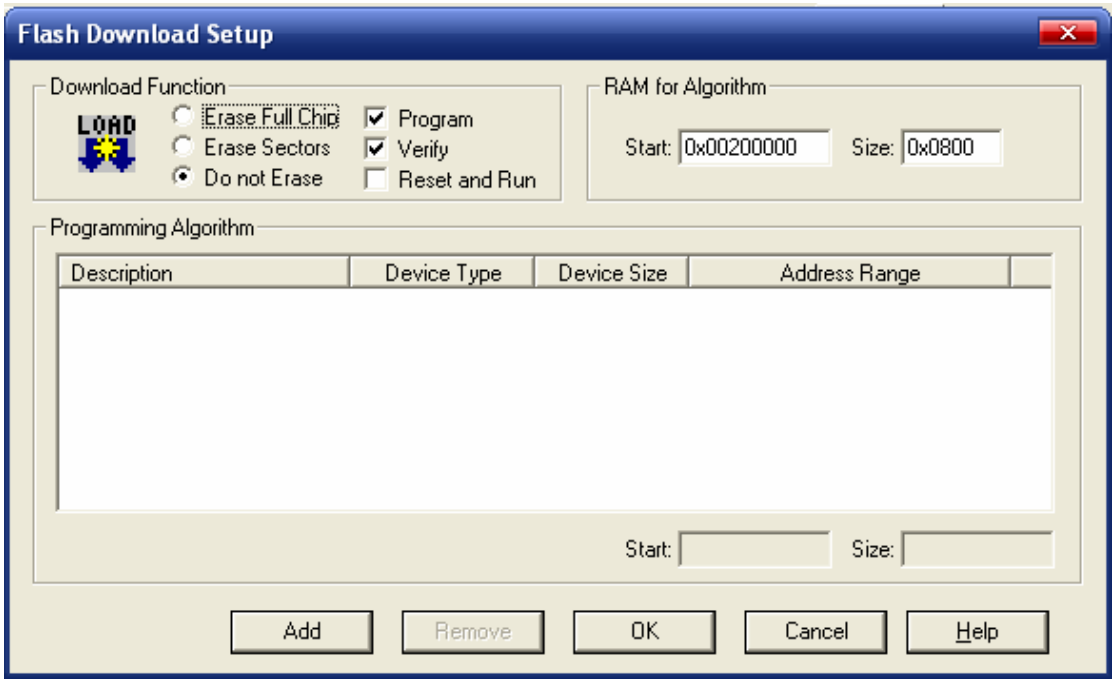

图 17

下面正式开始调试，点击下图中 d 字符按钮进入调试状态：

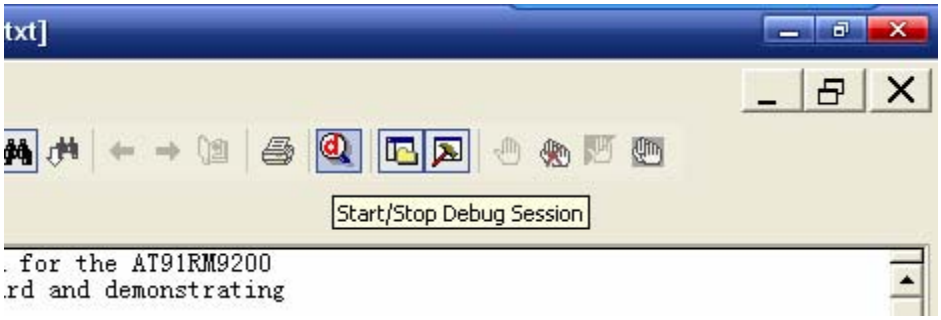

图 18

按下 debug 按钮后，Keil 很快编译和下载完成，进入 debug 状态，并停在 main 函数。

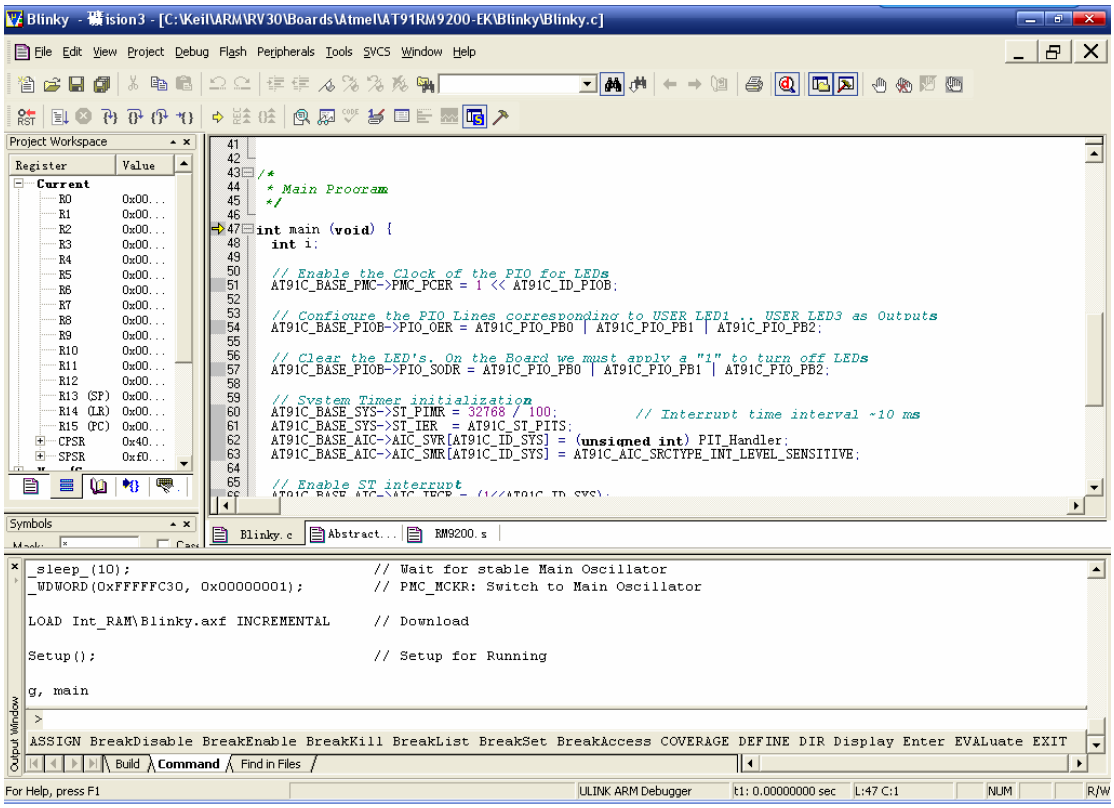

图 19

看看 output 窗口的内容：

“

Include "C:\\Keil\\ARM\\RV30\\Boards\\Atmel\\AT91RM9200-EK\\Blinky\\Int_RAM.ini"

/***/

/* Int_RAM.INI: Internal RAM Initialization File */

/***/

// <<< Use Configuration Wizard in Context Menu >>> //

/***/

/* This file is part of the uVision/ARM development tools. */

/* Copyright (c) 2005-2006 Keil Software. All rights reserved. */

/* This software may only be used under the terms of a valid, current, */

/* end user licence from KEIL for a compatible version of KEIL software */

/* development tools. Nothing else gives you the right to use this software. */

/***/

FUNC void Setup (void) {

// <o> Program Entry Point

PC = 0x200000;

}

```
// Switching from Slow Clock to Main Oscillator for faster Download
_WDWORD(0xFFFFFC20, 0x00000601);            // PMC_MOR: Enable Main Oscillator
_sleep_(10);                                // Wait for stable Main Oscillator
_WDWORD(0xFFFFFC30, 0x00000001);            // PMC_MCKR: Switch to Main Oscillator

LOAD Int_RAM\Blinky.axf INCREMENTAL         // Download

Setup();                                    // Setup for Running

g, main
”
```

主程序的内容：

“

```
/*----------------------------------------------------------------------------*/
/* BLINKY.C: LED Flasher                                                      */
/*----------------------------------------------------------------------------*/
/* This file is part of the uVision/ARM development tools.                    */
/* Copyright (c) 2005-2006 Keil Software. All rights reserved.                */
/* This software may only be used under the terms of a valid, current,        */
/* end user licence from KEIL for a compatible version of KEIL software       */
/* development tools. Nothing else gives you the right to use this software.  */
/*----------------------------------------------------------------------------*/

#include <AT91RM9200.H>                     /* AT91RM9200 definitions         */
#include <lib_AT91RM9200.h>                 /* Library function definitions   */

#define LED_NUM                     3       /* Number of LEDs                 */

const int led_mask[] = { AT91C_PIO_PB0, AT91C_PIO_PB1, AT91C_PIO_PB2 };

volatile unsigned long TimeTick;


/*
 *     Waiting for defined time (~10 * 10ms)
 */

void wait (void) {
   volatile unsigned long tick;

   tick = TimeTick;
   while ((TimeTick-tick)<10);
```

```
}


/*
 * Period Interrupt Timer (PIT) - interrupt function (every ~10 ms)
 */

__irq void PIT_Handler (void)    {
   TimeTick++;
   *AT91C_AIC_EOICR = *AT91C_ST_SR;
}


/*
 * Main Program
 */

int main (void) {
   int i;

   // Enable the Clock of the PIO for LEDs
   AT91C_BASE_PMC->PMC_PCER = 1 << AT91C_ID_PIOB;

   // Configure the PIO Lines corresponding to USER LED1 .. USER LED3 as Outputs
   AT91C_BASE_PIOB->PIO_OER = AT91C_PIO_PB0 | AT91C_PIO_PB1 | AT91C_PIO_PB2;

   // Clear the LED's. On the Board we must apply a "1" to turn off LEDs
   AT91C_BASE_PIOB->PIO_SODR = AT91C_PIO_PB0 | AT91C_PIO_PB1 | AT91C_PIO_PB2;

   // System Timer initialization
   AT91C_BASE_SYS->ST_PIMR = 32768 / 100;                  // Interrupt time interval ~10 ms
   AT91C_BASE_SYS->ST_IER    = AT91C_ST_PITS;
   AT91C_BASE_AIC->AIC_SVR[AT91C_ID_SYS] = (unsigned int) PIT_Handler;
   AT91C_BASE_AIC->AIC_SMR[AT91C_ID_SYS] = AT91C_AIC_SRCTYPE_INT_LEVEL_SENSITIVE;

   // Enable ST interrupt
   AT91C_BASE_AIC->AIC_IECR = (1<<AT91C_ID_SYS);

   // Loop forever
   for (;;) {
      for (i = 0; i < LED_NUM; i++) {
         AT91C_BASE_PIOB->PIO_CODR = led_mask[i];
         wait();
         AT91C_BASE_PIOB->PIO_SODR = led_mask[i];
```

```
        wait();
    }
    for (i = (LED_NUM - 1); i >= 0; i--) {
        AT91C_BASE_PIOB->PIO_CODR = led_mask[i];
        wait();
        AT91C_BASE_PIOB->PIO_SODR = led_mask[i];
        wait();
    }
  }
}

”
```

很明显，这是一个 LED 跑马程序，通过测量 PB0-2 的电平变化情况可以验证程序运行的正确性。

再次按照 d 字符的 debug 按钮，退出调试状态。

下面测试一下片外 FLASH 调试，请按照图 4 所示，选择 EXT FLASH 调试，然后按照图 5 进入设置界面：

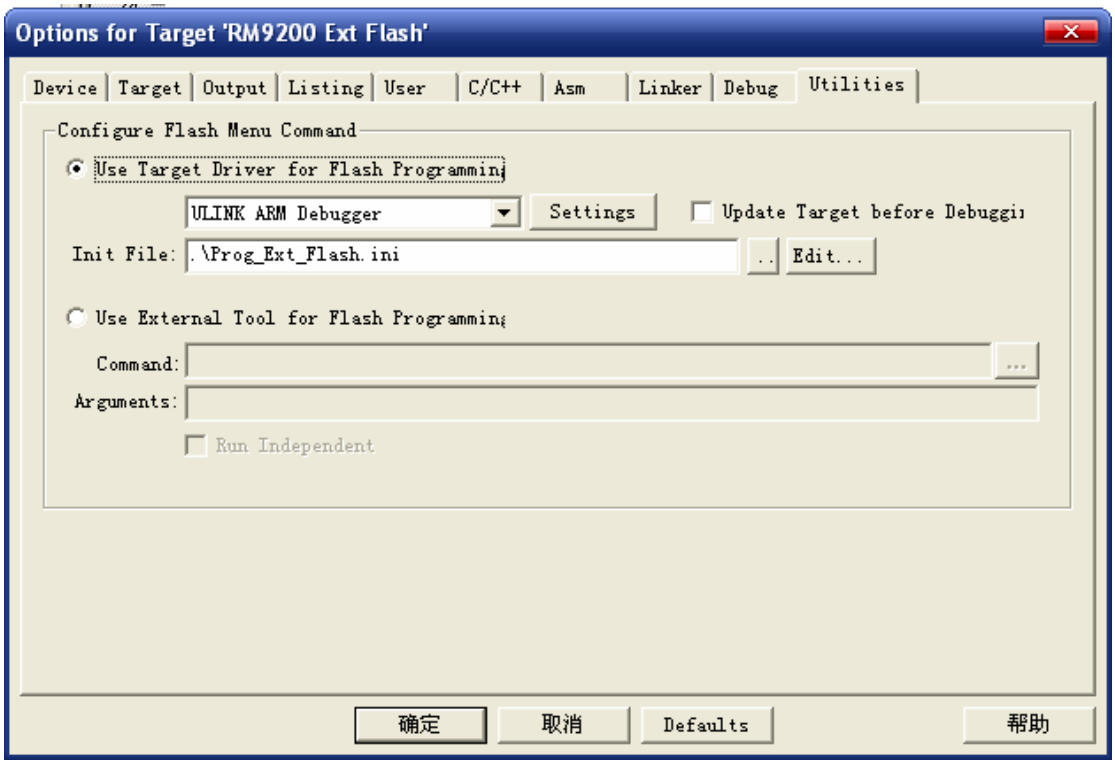

图 20

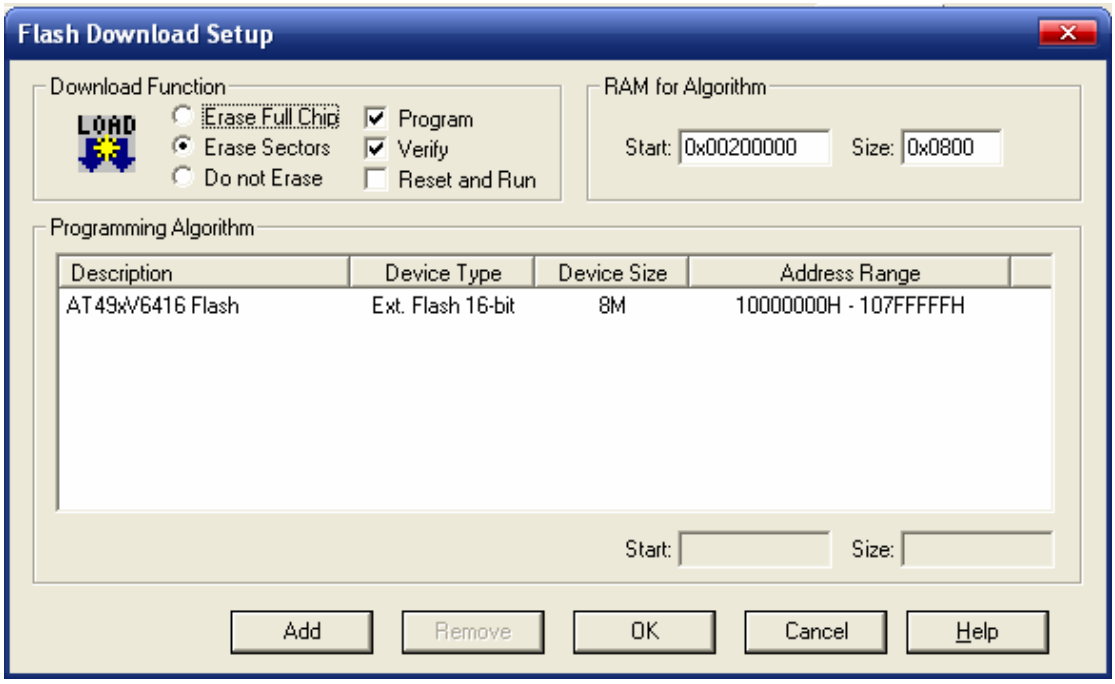

图 21

可以看到 9200EK 上使用的是 6416，而我们实际可能使用的是 163，由于结构类似，所以可以直接替换。另外，Keil 似乎并没有对 FLASH 的 ID 进行检测，因为实际编程中并没有出现错误提示。

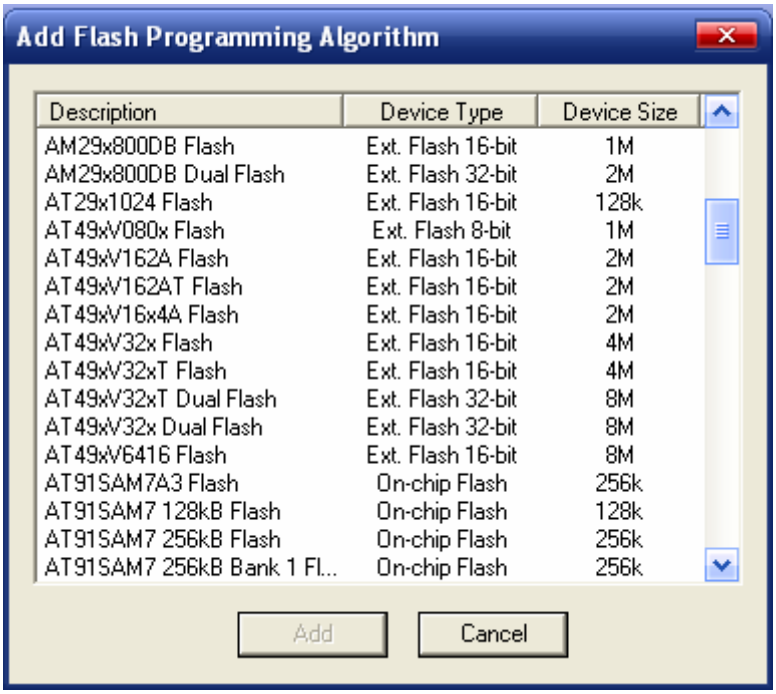

图 22

点击图 21 的 add 按钮，可以看到目前 Keil 支持的内片和片外 FLASH 的种类。这里并没有看到 AT49BV163D 的型号，我们只能用 AT49XV6416 来代替。

我们可以通过点击图 5 中设置按钮左边的 load 按钮来进行 FLASH 编程，以下是编程输出信息：

“

```
Load "C:\\Keil\\ARM\\RV30\\Boards\\Atmel\\AT91RM9200-EK\\Blinky\\Ext_Flash\\Blinky.AXF"
Include "C:\\Keil\\ARM\\RV30\\Boards\\Atmel\\AT91RM9200-EK\\Blinky\\Prog_Ext_Flash.ini"
/*****************************************************************************/
/* Prog_Ext_Flash.INI: Initialization File for Programming of External Flash  */
/*****************************************************************************/
// <<< Use Configuration Wizard in Context Menu >>>                           //
/*****************************************************************************/
/* This file is part of the uVision/ARM development tools.                   */
/* Copyright (c) 2005-2006 Keil Software. All rights reserved.               */
/* This software may only be used under the terms of a valid, current,       */
/* end user licence from KEIL for a compatible version of KEIL software      */
/* development tools. Nothing else gives you the right to use this software. */
/*****************************************************************************/
// Switching from Slow Clock to Main Oscillator for faster Download
_WDWORD(0xFFFFFC20, 0x00000601);        // PMC_MOR: Enable Main Oscillator
_sleep_(10);                            // Wait for stable Main Oscillator
_WDWORD(0xFFFFFC30, 0x00000001);        // PMC_MCKR: Switch to Main Oscillator
_WDWORD(0xFFFFFF70, 0x00003284);        // SMC_CSR0: Setup SCM for Ext Flash
Erase Done.
Programming Done.
Verify OK.
```

”

可以看到编程成功完成。

点击 debug 按钮进入调试状态：

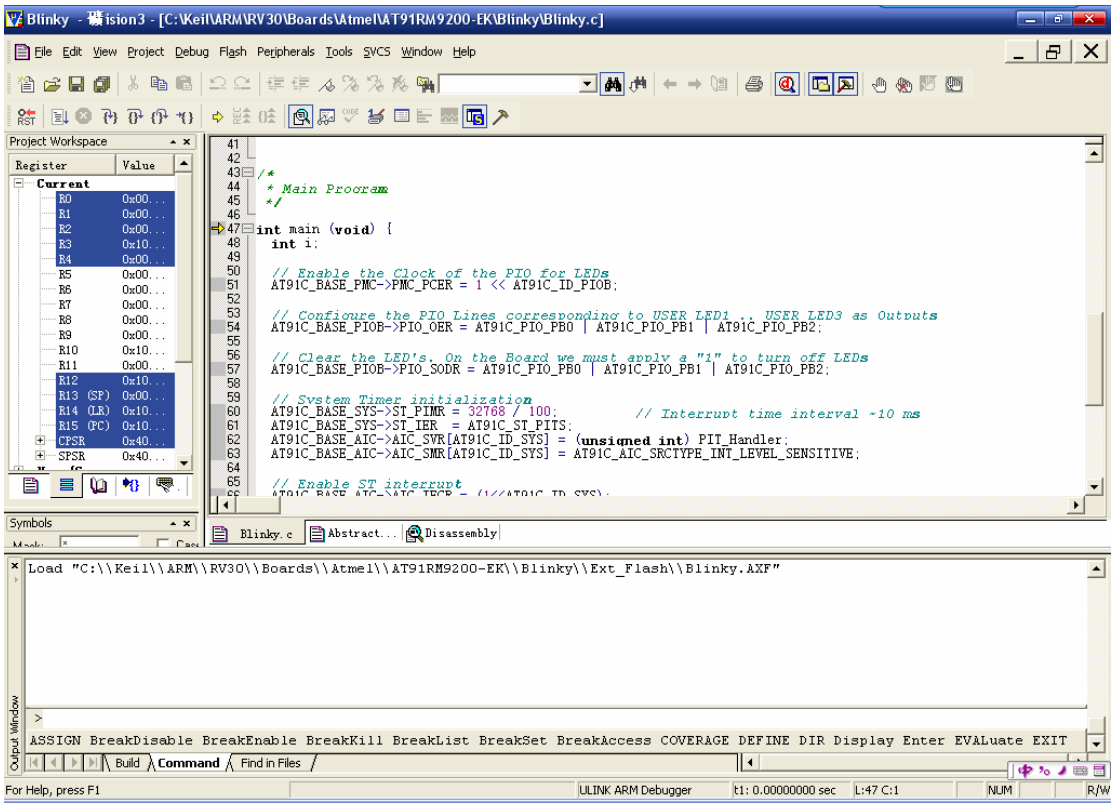

图 23

通过检测 PB0-2 的电平，可以验证程序已经写入并且在运行。

注意：如果要编程 FLASH，请一定要跳上从 FLASH 运行的跳线，不然就算 KEIL 提示编程完成，也可能出现进入 DEBUG 后无法停止，或者提示“Memory Mismatch”。

到此，Keil 下调试 9200 的步骤基本完成，可以按照实际情况进行 SRAM，SDRAM，FLASH 调试。具体关于编译器的使用细节问题可以参考相关资料，如 Keil 的帮助文档。