TechV-C6416 使用说明书

北京达盛科技有限公司

第一版：2007-4-5, Edited By Lyj

地址：北京市海淀区长春桥路 5 号新起点大厦 2-1501
邮编：100089
电话：010-82564899
Web：http://www.techshine.com/

目　录

第一章　TechV-C6416 介绍

1. 1 CPU 特点

C6000 是 TI 公司于 1997 年推出的 DSP 芯片，C64xx 系列是浮点运算处理器。C6000 片内有 8 个并行处理单元，分为相同的两组，其体系结构采用甚长指令字（VLIW）结构，单指令字长为 32bit，8 个指令组成一个指令包，总字长为 8*32=256bit。芯片内部设置了专门的指令分配模块，可以将每个 256bit 的指令包同时分配到 8 个处理单元，并由 8 个单元同时运行。C6416 具有四个版本，分别是 600MHz、720M、850M 和 1GHz，当 8 个单元同时运行时，对 600MHz 版本而言，其最大处理能力可以达到 600*8=4800MIPS。

新型 C64x DSP 以其 C64x 内核的先进超长指令字(VLIW)结构，获得当前应用所需的最高性能。C64x 内核的 8 个功能单元能够在每个周期内执行 4 组 16 位 MAC 运算或 8 组 8 位 MAC 运算，以便在处理通信和影像算法中获得最大的并行性。单个 C64x DSP 能够同时完成一个通道的 MPEG4 视频编码、一个通道的 MPEG4 视频解码和一个 MPEG2 视频解码，并仍有 50％的余量留给多通道语音和数据编码。

它们的存储器和外设系统具有下面的主要特点：具有 1056KB 片上 SRAM 的实时分级存储系统；64 通道的增强型 DMA 控制器，针对系统存储器的千兆字节/秒数据 I/O 的有效管理，显示出色的并行性；外部双总线提供超过 1.2G 字节的外部存储器带宽；33MHz/32 位 PCI 接口和针对 ATM 的 UtopiaⅡ接口（均仅限于 C6415 和 C6416 两种型号）简化了在 C6415 和 C6416 上主机和网络的连通性；3 个多通道缓冲串行口(McBSP)，均分别支持 128 个 TDM 通道及标准音频接口(包括 AC97 和 IIS)。

C6416 还专门为 3G 算法集成了协处理器。Turbo 协处理器(TCP)可完成无线数据的 Turbo 解码；Vitebi 协处理器(VCP)完成无线语音信道的卷积解码。虽然 C64x 内核为这些算法提供了很好的性能，但是由于它在 3G 无线应用中应泛被采用，故决定采用协处理器提供一种最佳系统解决方案。在协处理器和 C64x 内核的组合中，单个 C6416 器件在 12.2kbps 时支持 300 个 AMR 语音信道的符号速率处理，在 384kbps 时可支持 35 个数据信道，超出目前市场上其它 DSP。

由于 C64x DSP 指令集是 C6x DSP 指令集的超指令集，所以可以直接移植为前一代产品开发的目标代码。利用 C64x DSP C 编译器能够生成新的代码，并快速修改现有代码，对于发展迅速的 3G 无线领域，这是非常重要的功能。

图 1 是其结构框图：

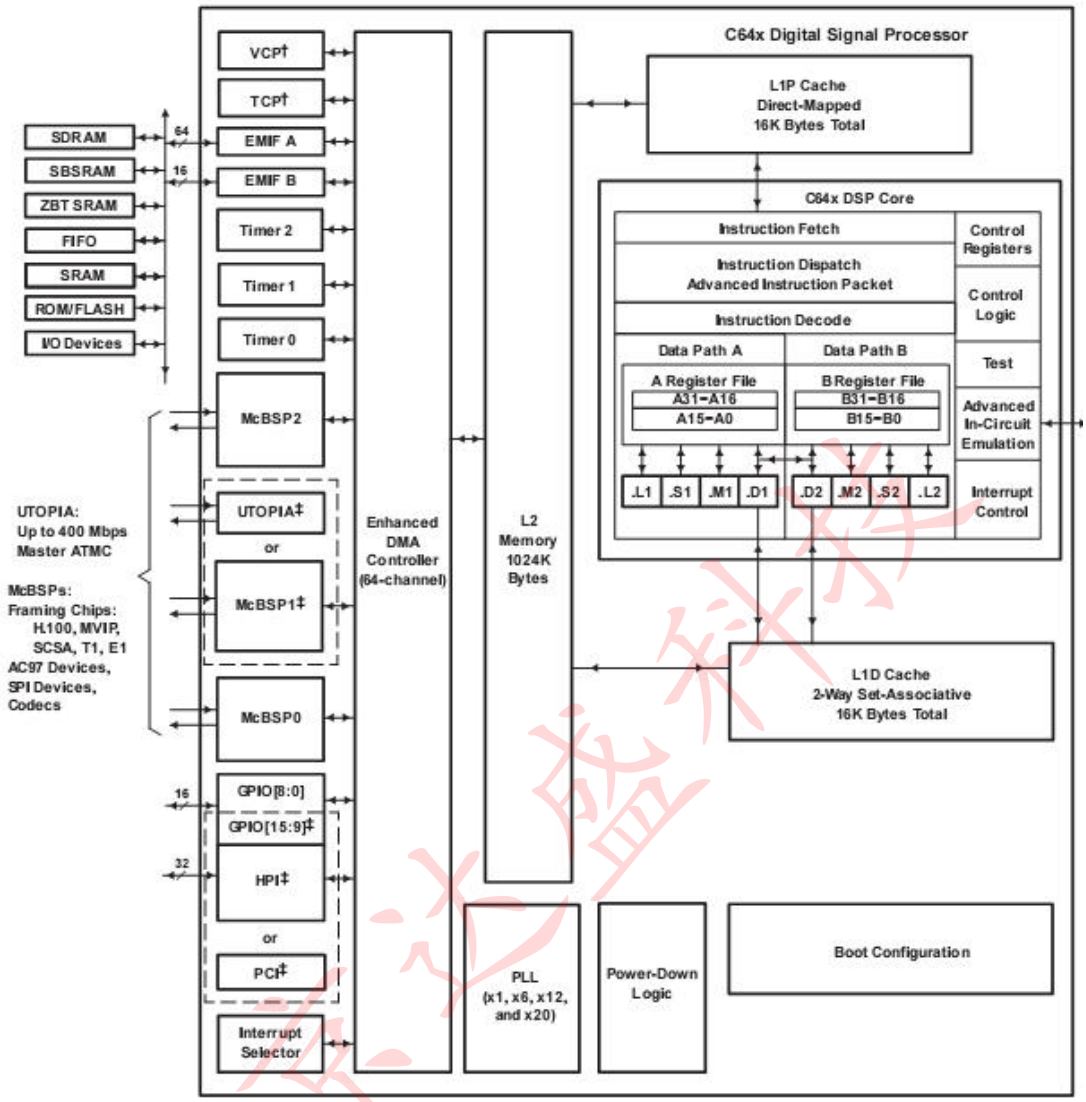


图 1

1.2 板卡特点

TechV-C6416是低成本，高度集成的高性能信号处理开发平台，可以开发仿真TI C64xx系列DSP应用程序，同时也可以将该产品集成到用户的具体应用系统中。方便灵活的接口为用户提供良好的开放平台。采用该系列板卡进行产品开发或系统集成可以大大减少用户的产品开发时间。板卡结构框图如图2所示：

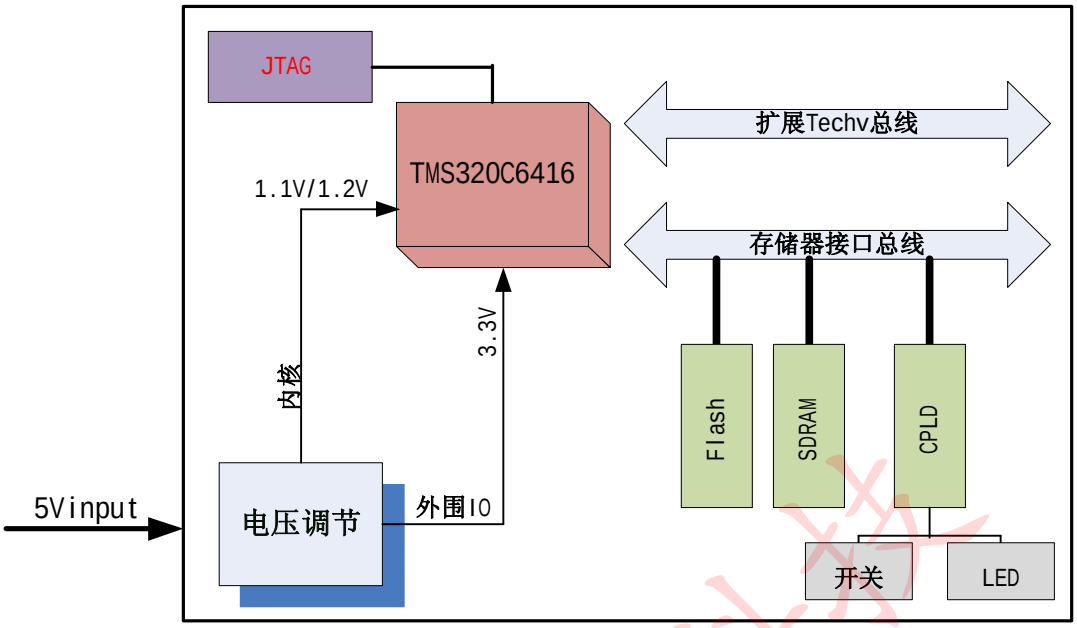


图 2

- **板卡资源包括：**

- TMS320C6416 DSP 可操作在600/720/850/1000 MHz；
- 256 Mbits 的SDRAM存储器；
- 1Mbytes Flash存储器；
- 4个DIP开关，4各状态指示LED；
- 可以通过编程来配置CPLD内部的寄存器配置板卡的功能；
- 8位Flash BOOT模式；
- 10层板制作工艺，稳定可靠；
- 标准外部信号扩展接口；
- JTAG仿真器接口；
- 单电源+5V供电 。

1.3 功能概述

TechV-C6416CPU板卡通过64位外部存储器接口扩展16Mbytes的SDRAM，16位外部存储器接口扩展1Mbytes的Flash，并且所有外部存储器接口信号都连接到子板扩展接口(ExtCE3)上，用户可以通过外部子板扩展接口扩展自己的功能子板。

板上CPLD主要完成系统的控制（DSP通存储器及其他外设的无缝接口）以及状态，DIP和数字量输出/输入的扩展。除基本控制逻辑外包括4位LED、和4位DIP开关量输入。

板卡通过标准电源接口提供电源，采用单5v的供电方式，并且供电电流需要大于1安培，电源调节模块为微处理器及其他外设提供3.3V和1.1/1.2V(对应不同频率CPU)电压。在使用该系列板卡开发过程中可以通过J1接口连接仿真器。

1.4 地址空间映射

C64xx系列数字信号处理器具有较大的寻址空间。处理器采用统一寻址方式，并具有两组外部存储器接口：EMIFA及EMIFB。其中，EMIFA可配置为64、32、16、8位总线，而EMIFB只能配置为16、8位总线接口。程序和数据代码可以存放到任何寻址空间。其芯片的内存映射图如下：

Table 3. TMS320C6414T, C6415T, C6416T Memory Map Summary

MEMORY BLOCK DESCRIPTION	BLOCK SIZE (BYTES)	HEX ADDRESS RANGE
Internal RAM (L2)	1M	0000 0000 – 000F FFFF
Reserved	23M	0010 0000 – 017F FFFF
External Memory Interface A (EMIFA) Registers	256K	0180 0000 – 0183 FFFF
L2 Registers	256K	0184 0000 – 0187 FFFF
HPI Registers	256K	0188 0000 – 018B FFFF
McBSP 0 Registers	256K	018C 0000 – 018F FFFF
McBSP 1 Registers	256K	0190 0000 – 0193 FFFF
Timer 0 Registers	256K	0194 0000 – 0197 FFFF
Timer 1 Registers	256K	0198 0000 – 019B FFFF
Interrupt Selector Registers	256K	019C 0000 – 019F FFFF
EDMA RAM and EDMA Registers	256K	01A0 0000 – 01A3 FFFF
McBSP 2 Registers	256K	01A4 0000 – 01A7 FFFF
EMIFB Registers	256K	01A8 0000 – 01AB FFFF
Timer 2 Registers	256K	01AC 0000 – 01AF FFFF
GPIO Registers	256K	01B0 0000 – 01B3 FFFF
UTOPIA Registers (C6415T and C6416T only)†	256K	01B4 0000 – 01B7 FFFF
TCP/VCP Registers (C6416T only)‡	256K	01B8 0000 – 01BB FFFF
Reserved	256K	01BC 0000 – 01BF FFFF
PCI Registers (C6415T and C6416T only)†	256K	01C0 0000 – 01C3 FFFF
Reserved	4M – 256K	01C4 0000 – 01FF FFFF
QDMA Registers	52	0200 0000 – 0200 0033
Reserved	736M – 52	0200 0034 – 2FFF FFFF
McBSP 0 Data	64M	3000 0000 – 33FF FFFF
McBSP 1 Data	64M	3400 0000 – 37FF FFFF
McBSP 2 Data	64M	3800 0000 – 3BFF FFFF
UTOPIA Queues (C6415T and C6416T only)†	64M	3C00 0000 – 3FFF FFFF
Reserved	256M	4000 0000 – 4FFF FFFF
TCP/VCP (C6416T only)‡	256M	5000 0000 – 5FFF FFFF
EMIFB CE0	64M	6000 0000 – 63FF FFFF
EMIFB CE1	64M	6400 0000 – 67FF FFFF
EMIFB CE2	64M	6800 0000 – 6BFF FFFF
EMIFB CE3	64M	6C00 0000 – 6FFF FFFF
Reserved	256M	7000 0000 – 7FFF FFFF
EMIFA CE0	256M	8000 0000 – 8FFF FFFF
EMIFA CE1	256M	9000 0000 – 9FFF FFFF
EMIFA CE2	256M	A000 0000 – AFFF FFFF
EMIFA CE3	256M	B000 0000 – BFFF FFFF
Reserved	1G	C000 0000 – FFFF FFFF

† For the C6414T device, these memory address locations are reserved. The C6414T device does ***not*** support the UTOPIA and PCI peripherals.
‡ Only the C6416T device supports the VCP/TCP Coprocessors. For the C6414T and C6415T devices, these memory address locations are reserved.

本板卡具体的存储空间分配如下表，使用对应的地址就可以访问相应的存储器或外围设备。

对应起始地址	C6416CPU	TechvCPU板
0x0000 0000 0x0010 FFFF	内部RAM	内部RAM
0x0180 FFFF	保留	保留
	片内外设寄存器 及保留空间	片内外设寄存器 及保留空间
0x6000 0000	***EMIFB CE0***	CPLD
0x6400 0000	***EMIFB CE1***	Flash
0x6800 0000	***EMIFB CE2***	保留
0x6c00 0000	***EMIFB CE3***	保留
0x8000 0000	EMIFA CE0	SDRAM
0x9000 0000	EMIFA CE1	扩展总线片选ExtCE0
0xA000 0000	EMIFA CE2	扩展总线片选ExtCE1
0xB000 0000	EMIFA CE3	扩展总线片选ExtCE2、3

例如，要访问SDRAM的某一个地址0x04，可以这样操作：

*(int *)0x80000004=0x12345678;

相应译码：

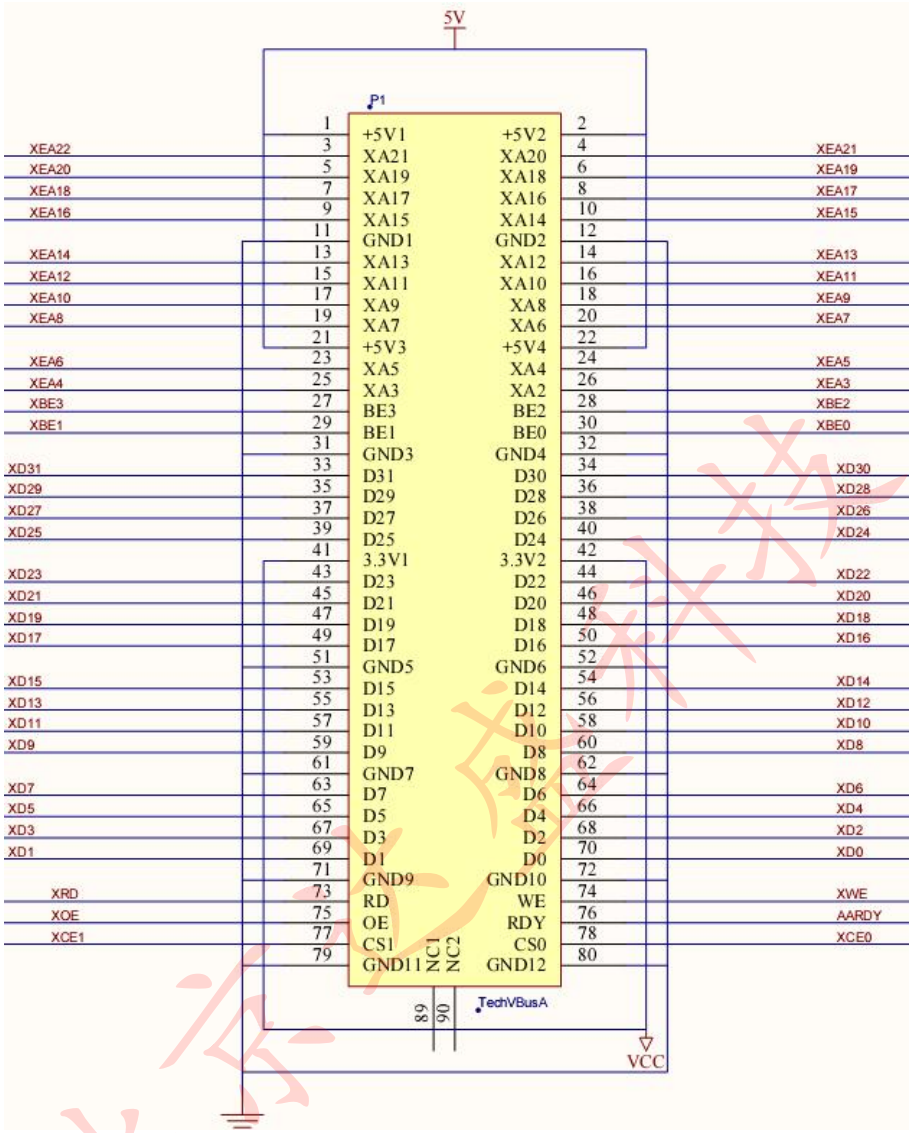
```
XCE(1 downto 0)   <= TACE(2 downto 1);
XCE(2)            <= TACE(3);
XCE(3)            <= TACE(3);
```

1.5 扩展总线 Techv 信号描述

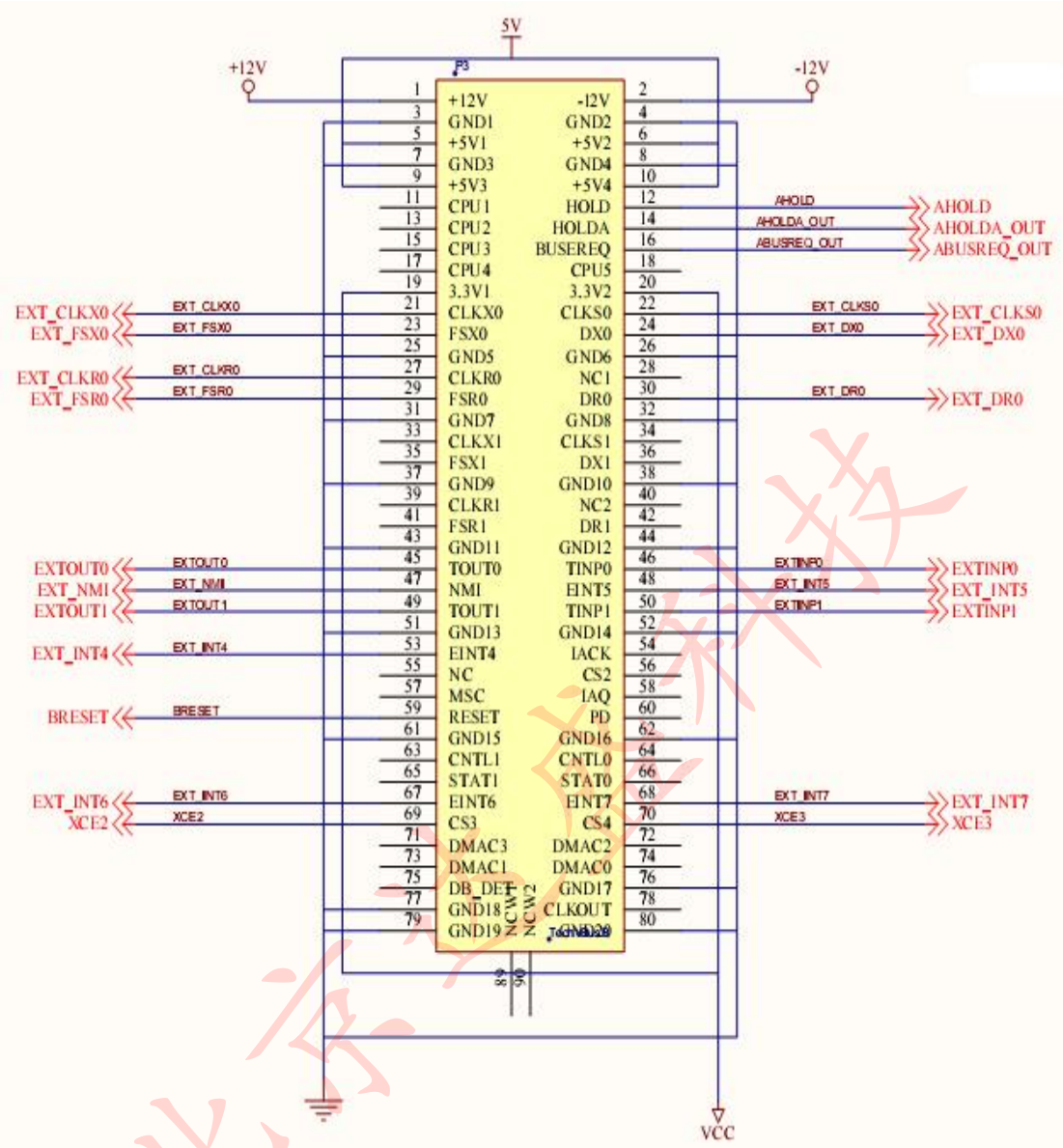
1.5.1 基本描述

- 总线采用 EMIFA 的 32 位访问模式。
- 标准的 Techv 总线，提供中断、定时器脉冲等输入，及定时器、IO 输出。

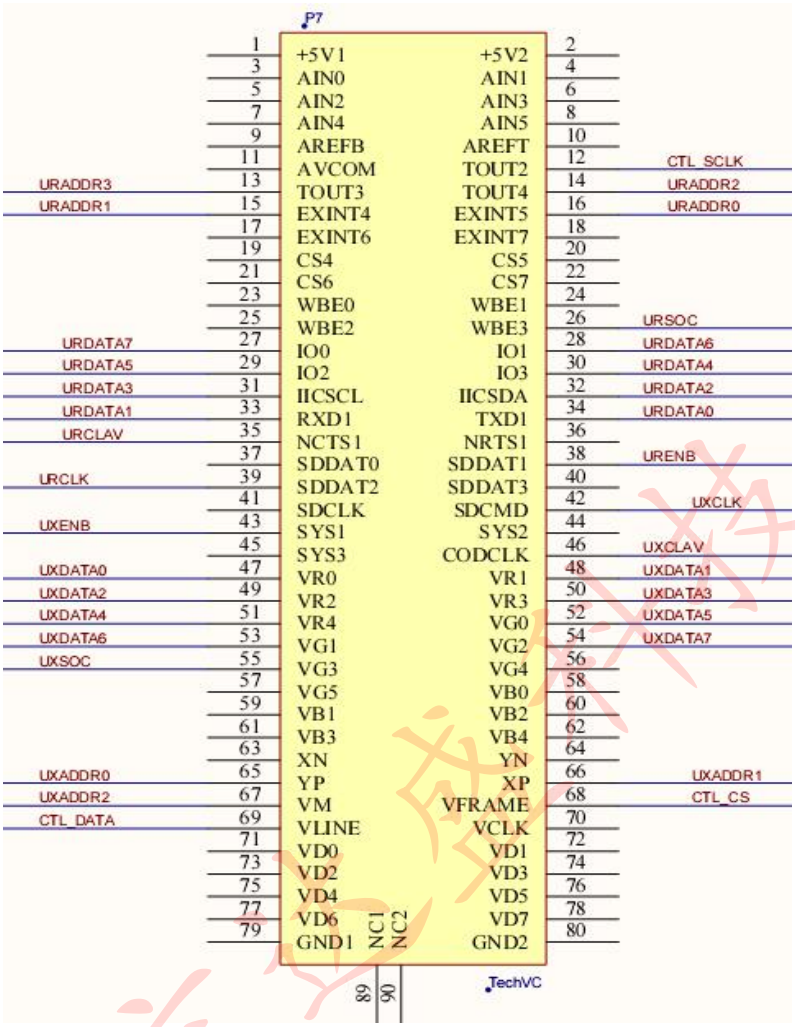
1.5.2 管脚描述



P1 和 P5 各引脚定义相同，P5 在 P1 的背面；



P3 和 P6 各引脚定义相同，P6 在 P3 的背面；



P7 和 P4 各引脚定义相同，P4 在 P7 的背面；

根据预留的 TECH-V 总线接口定义，用户可以根据需要进行二次开发。

1.6 元件布局

板卡元件布局正面视图如图 3 所示：

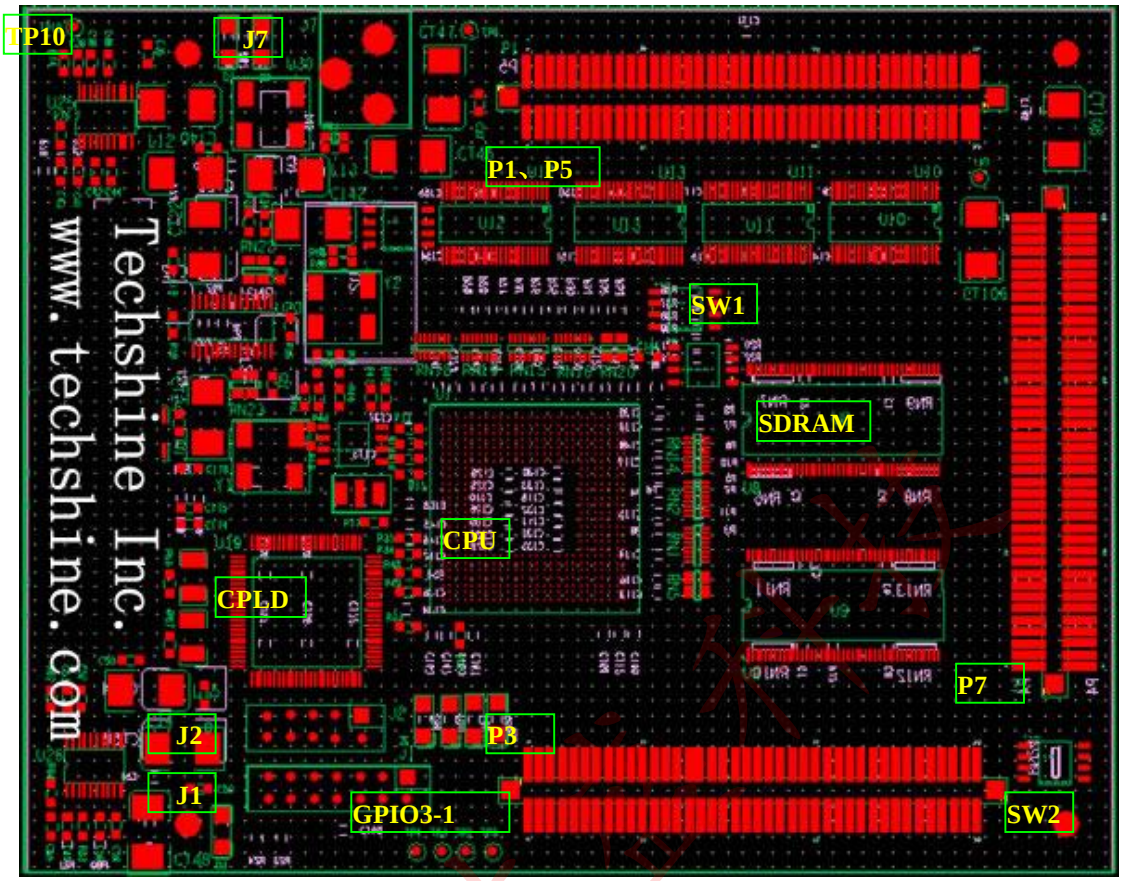


图 3

部分元件说明（按 从左到右、从上到下 顺序）：

- Flash 芯片位于背部。
- TP10 为地 GND 测试点。
- J7 为 5V 电源输入接口接口，如图 4 所示：

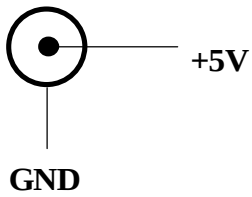
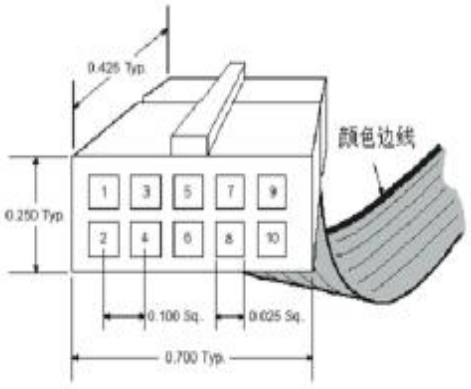


图 4

- TechV 总线 P1～P7 含义如下：
  P1：地址、数据总线，与背后的 P5 相同；
  P3：中断、串行接口等，与背后的 P6 相同；
  P7：HPI/PCI 总线，与背后的 P4 相同。
  具体信号线可以参考文件夹“原理图”目录下的“C6416 原理图.pdf”文件。
- SW1 是芯片配置开关，具体功能下面会有详细说明。
- J2 是 CPLD 的 JTAG，用来下载 CPLD 程序的，端口定义如下：



管脚	含义
1	TCK
3	TMS
5	TDI
7	TDO
4、8	GND
9、10	VCC (5V)

图 5 CPLD 的 JTAG 接口

- J1 是 DSP 的 JTAG，用来调试 DSP。
  注意：仿真器切忌接反，接头红色导线边插在方形针一边。

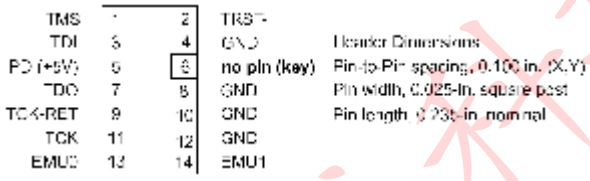


图 6 DSP 的 JTAG 接口

- GPIO3-1 是 3 个通用输入输出管脚，直接接到 CPU 的 GP3、2、1。
- SW2 为 CPLD 接入开关，用户可以自定义。

1.7 配置说明

1.7.1 配置芯片工作模式

Techv6416 CPU 板卡有一个配置拨码开关 SW1，用户可以通过它来配置一系列的 CPU 工作模式。具体含义如下表所示：

<table>
<tr><th>拨码开关对应位</th><th>功能说明</th></tr>
<tr><td>S1</td><td>配置 CPU 的大小端
ON： Big Endian
OFF：Little Endian (默认)</td></tr>
<tr><td>S2</td><td>配置 CPU 启动模式(boot mode)</td></tr>
<tr><td>S3</td><td>
<table>
<tr><th>SW2</th><th>SW3</th><th>Boot Mode</th></tr>
<tr><td>ON</td><td>OFF</td><td>没有启动</td></tr>
<tr><td>ON</td><td>ON</td><td>HPI 启动</td></tr>
<tr><td>OFF</td><td>OFF</td><td>8 位 Flash 启动（默认）</td></tr>
<tr><td>OFF</td><td>ON</td><td>保留</td></tr>
</table>
</td></tr>
<tr><td>S4</td><td>保留</td></tr>
</table>

注：具体配置方式可以参考 6416 的数据手册 Page37。

1.7.2 用户开关功能

SW2 是用户定义的拨码开关，出厂时为 LED_D4 的指示选择，其功能如下：

拨码开关对应位				功能说明
S1	S2	S3	S4	
ON	ON	ON	ON	保留
ON	ON	ON	OFF	内核电压 1.1、1.2V 稳定指示
ON	ON	OFF	ON	I/O 电压 3.3V 稳定指示
ON	OFF	OFF	ON	数据 D0 与 D1 的“或”输出
ON	OFF	OFF	OFF	数据 D2 与 D3 的“或”输出
X	X	X	X	数据 D4、D5、D6 及 D7 的“或”输出

注：x 表示任意其他组合。

U30 按钮：系统的复位按键，每按一下，系统进行一次复位。D7 亮时表示系统复位结束；D7 灭时表示系统正在复位。

1.7.3 使用板上的LED

LED 对应编号	功能说明
LED_D4	见上面表格
LED_D5	可编程，对应数据线 D0 地址：0x6000 0000 数据：0：LED 灭 1：LED 亮
LED_D6	可编程，对应数据线 D1 地址：0x6000 0000 数据：0：LED 灭 1：LED 亮
LED_D7	系统电源复位指示： LED 灭：系统正在复位 LED 亮：系统复位完毕

注：用户可以参照原理图，自己编写 CPLD 程序，重定义 LED 功能。

LED_D1:5V 电压指示灯；供电正常亮。

LED_D2:外设电压指示灯；供电正常亮。

LED_D3:DSP 核电压指示灯；此 DSP 核电压不能将其驱动点亮。

1.7.4 TECH-V总线上中断分配

TECH-V 总线上外扩的中断 EXT_INT4/ EXT_INT5/EXT_INT6/ EXT_INT7/EXTINP0/ EXTINP1 与 DSP 的中断 EXT_INT4/ EXT_INT5/EXT_INT6/ EXT_INT7/EXTINP0/

EXTINP1 一一对应，可供用户扩展使用。详见 TECH-V 总线的各引脚定义。

1.7.5 TECH-V总线外扩片选地址空间分配

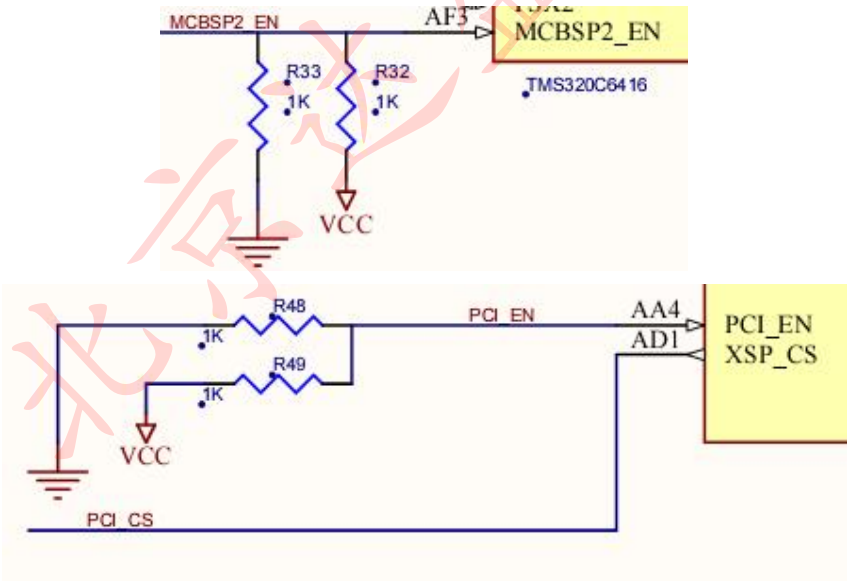
XCE0：映射空间为 0x90000000～0x9fffffff；
XCE1：映射空间为 0xa0000000～0xafffffff；
XCE2：映射空间为 0xb0000000～0xbfffffff；
XCE3：映射空间为 0xb0000000～0xbfffffff；
相应译码：
XCE(1 downto 0)　<= TACE(2 downto 1);
XCE(2)　　<= TACE(3);
XCE(3)　　<= TACE(3);
其中 TACE(3 downto 1)分别对应 DSP 的 EMIFA CE3~ EMIFA CE1 片选信号。

1.8 其他几个配置引脚

由于 C6416 的许多引脚具有复用功能，出厂设置的功能不一定符合用户所期望，所以用户可以更改一些电阻焊接实现更多的功能，具体参考原理图的 Page4。

1.8.1 HPI、PCI、MCBSP2 使能

连接关系如图：



出厂时 R32、R48 已焊接，具体功能指示请参考 C6416 手册 Page38 或下图。

Table 27. PCI_EN and MCBSP2_EN Peripheral Selection (HPI, GP[15:9], PCI, and McBSP2)

PERIPHERAL SELECTION†		PERIPHERALS SELECTED				
PCI_EN Pin [AA4]	MCBSP2_EN Pin [AF3]	HPI	GP[15:9]	PCI	EEPROM (Internal to PCI)	McBSP2
0	0	√	√			√
0	1	√	√			√
1	0			√	√	‡
1	1			√		√

† The PCI_EN pin *must* be driven valid at all times and the user *must not* switch values throughout device operation. The MCBSP2_EN pin *must* be driven valid at all times and the user *can* switch values throughout device operation.

‡ The only time McBSP2 is disabled is when both PCI_EN = 1 and MCBSP2_EN = 0. This configuration enables, at reset, the auto-initialization of the PCI peripheral through the PCI internal EEPROM [provided the PCI EEPROM Auto-Initialization pin (BEA13) is pulled up (EEAI = 1)]. The user can then enable the McBSP2 peripheral (disabling EEPROM) by dynamically changing MCBSP2_EN to a "1" after the device is initialized (out of reset).

- If the PCI is disabled (PCI_EN = 0), the HPI peripheral is enabled and GP[15:9] pins can be programmed as GPIO, provided the GPxEN and GPxDIR bits are properly configured. [Note: The PCI_EN pin *must* be driven valid at all times and the user *must not* switch values throughout device operation.]

  This means all multiplexed HPI/PCI pins function as HPI and all standalone PCI pins ($\overline{\text{PCBE0}}$ and XSP_CS) are tied-off (Hi-Z). Also, the multiplexed GPIO/PCI pins can be used as GPIO with the proper software configuration of the GPIO enable and direction registers (for more details, see Table 29).

- If the PCI is enabled (PCI_EN = 1), the HPI peripheral is disabled. [Note: The PCI_EN pin *must* be driven valid at all times and the user *must not* switch values throughout device operation.]

  This means all multiplexed HPI/PCI pins function as PCI. Also, the multiplexed GPIO/PCI pins function as PCI pins (for more details, see Table 29).

- The MCBSP2_EN pin, in combination with the PCI_EN pin, controls the selection of the McBSP2 peripheral and the PCI internal EEPROM (for more details, see Table 27 and its footnotes). [Note: The MCBSP2_EN pin *must* be driven valid at all times and the user *can* switch values throughout device operation.]

1.8.2 UTOPIA 和 McBSP1

已配置为 McBSP1，并通过 TECHV 总线的 P3、P6 输出。

1.8.3 EMIFA 、EMIFB 时钟输入选择

均已配置为 CPU 时钟的 1/6 。

第二章　CPU 板的使用演示

2.1 程序演示

2.1.1 LED灯测试程序

本板卡的程序均在 CCS3.1+Windows XP 平台测试通过，下面以实现 LED_D5 闪烁为例说明之。

1、连接好仿真器和 CPU 板卡，上电。

2、打开 CCS3.1，连接 CPU，并加载程序：

图 7 连接上标志

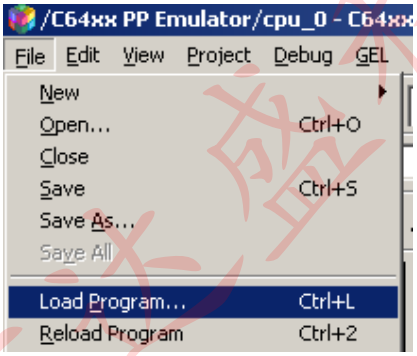


图 8

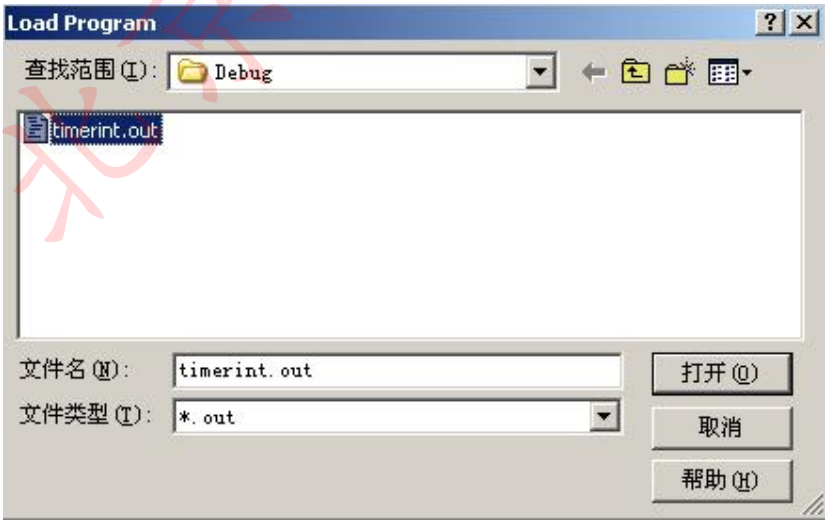


图 9

3、点击 或者按 F5 运行程序，可以看到板卡的用户 LED-D5~LED-D6 间隔闪烁，PCB

上对应 D5－D6。

2.1.2 外扩RAM测试程序

本板卡的程序均在 CCS3.1+Windows XP 平台测试通过，下面以实现 LED_D5 闪烁为例说明之。

1、连接好仿真器和 CPU 板卡，上电。

2、打开 CCS3.1，连接 CPU，并加载程序：

图 10 连接上标志

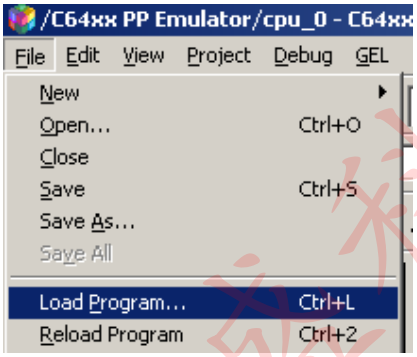


图 11

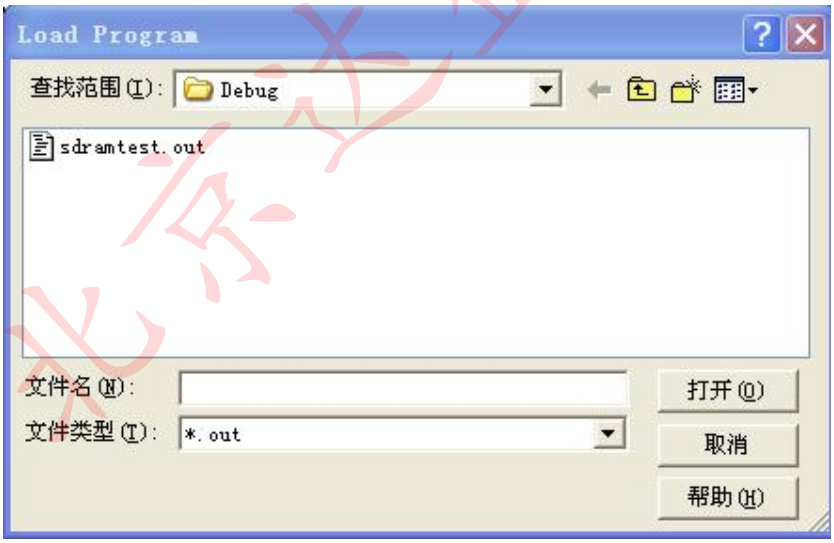


图 12

3、用下图所示快捷键打开内存观察窗口；

图 13

进行如下设置，单击 OK 确定：

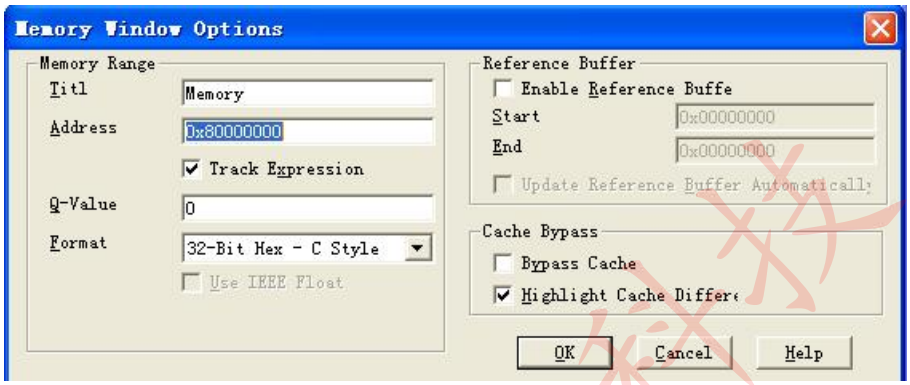
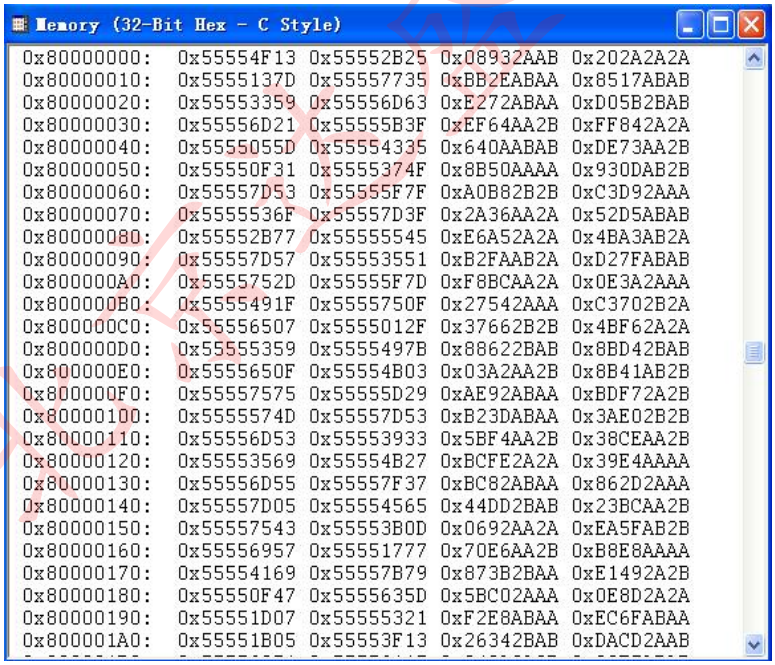
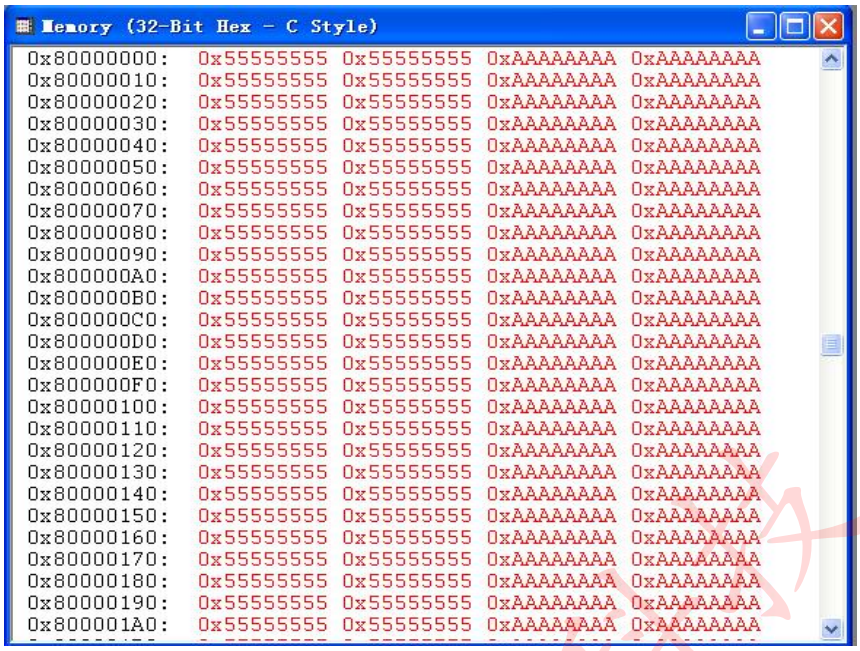


图 14



4、点击  或者按 F5 运行程序，观察内存单元内容的变化；写入内存的长度为 0xfff。



写入数据正确，表明外扩 RAM 工作正常。

2.1.3 FLASH烧写

测试 FLASH 存储器的烧写，可按程序中的说明进行操作；在程序中注明断点的地方设置断点，进行 FLASH 的擦除和烧写。

详见程序中注释。

2.2 附部分程序说明

```
//              北京精仪达盛科技有限公司
//                    Techshine
//                 www.techshine.com
#define CHIP_6416 //需要定义使用的芯片，因为 CSL 函数是通过此句区分 CPU 的。
#include <csl.h>
#include <csl_irq.h>
#include <csl_timer.h>
#include "dbgm.h"
#include "c6x.h"
void SetupInterrupts(void);
void ConfigureAllTimers(void);
extern void SetupEmifb();
#define LED_D5 *(Uint8*)0x60000000
void intConfig(void);
void main(void)
{
    volatile int KeepRunning = 1,j;
    CSL_init(); //CSL 函数初始化，建议采用 CSL，方便一点。
```

```
    SetupEmifb(); //设置 EMIFB
    ConfigureAllTimers();
    SetupInterrupts(); //设置中断
    while ( KeepRunning )
    {
        j++;
    }
}
interrupt void Timer0Isr(void)
{
    static Uint8 i=0;
    LED_D5=(i++)%2?0:1;//每次中断更新 LED 状态
}
void SetupInterrupts(void)
{
    IRQ_resetAll();                         /* Reset all maskable interrupts   */
    IRQ_enable(IRQ_EVT_TINT0);                 /* Enable timer0 -> CPU interrupt */
    IRQ_nmiEnable();                        /* Enable non-maskable interrupt   */
    IRQ_globalEnable();                     /* Globally enable all interrupts */
}
void ConfigureAllTimers(void)
{
    TIMER_Handle timer0;
    TIMER_Handle timer1;
    TIMER_Handle timer2;
    timer0 = TIMER_open(TIMER_DEV0,0);
    TIMER_configArgs( timer0, 0x000002c0, 0x1ffffff, 0x00000000);
    timer1 = TIMER_open(TIMER_DEV1,0);
    TIMER_configArgs( timer1, 0x000002c0, 0x00100, 0x00000000);
    timer2 = TIMER_open(TIMER_DEV2,0);
    TIMER_configArgs( timer2, 0x000002c0, 0x10000, 0x00000000);
}
```