

User Manual
For
LM1000

Hardware Version A
Manual Version A.1

© Copyright 2004 LAB Microsystems, LLC

IMPORTANT NOTICES

Unless otherwise specified, all software and documentation © COPYRIGHT 2004 by LAB Microsystems, LLC, Wilton, NH. All Rights Reserved. No part of this document may be reproduced in any form without written permission.

The information in this document is believed to be accurate and reliable. However, no responsibility is assumed for any inaccuracies or for the use of any information contained herein. Furthermore, LAB Microsystems reserves the right to make changes without notice to any product herein to improve reliability, function, or design; LAB Microsystems advises its customers to obtain the latest version of relevant information to verify that the information being relied on is current.

LAB Microsystems assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein.

No license is granted by implication or otherwise under any patent right, copyright, or other intellectual property right of LAB Microsystems or under the rights of others. LAB Microsystems acknowledges all trademarks.

Certain applications using computer boards may involve potential risks of death, personal injury, or severe property or environmental damage. Inclusion of LAB Microsystems products in such applications is understood to be fully at the risk of the customer, and that LAB Microsystems is thereby indemnified against all damages. LAB Microsystems products are not designed, intended, authorized, or warranted to be suitable for use in life-support applications, devices, or systems, or in other critical applications.

You may use the driver and utility software provided by LAB Microsystems Corporation on any one computer; copy the Object Code into any machine-readable form for your use. A reasonable number of copies of this software may be made for BACKUP PURPOSES ONLY, and these copies must be destroyed or transferred to any subsequent purchaser of this product.

You may modify the software provided by LAB Microsystems and/or merge or incorporate it into any general-use software program of your development except for a program of similar nature to the LAB Microsystems product. You may freely reproduce any such program of your development; however, the merged or incorporated part of the LAB Microsystems-provided software will continue to be subject to all other provisions of this agreement. You must reproduce and include the copyright notices on any copy modification or portion thereof merged into another program. All source code is copyrighted under the United States Copyright laws and may not be copied without the express permission of the copyright holder.

Notice to U.S. Government End Users: Restricted Rights Legends

For civilian agencies: This software is licensed only with 'Restricted Rights' and use, reproduction, or disclosure is subject to restrictions set forth in subparagraph (a) through (d) of the Commercial Computer Software--Restricted Rights clause at 52.227-19 of the Federal Acquisition Regulations.

Unpublished -- rights reserved under the copyright laws of the United States.

For units of Department of Defense: Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013.

LIMITED WARRANTY

LAB Microsystems hardware is warranted against defects in materials and workmanship for a period of ONE YEAR from the date of purchase from LAB Microsystems or from an authorized LAB Microsystems distributor. Products that are found to be defective when returned prepaid to LAB Microsystems within the warranty period may be repaired or replaced at LAB Microsystems's option.

LAB Microsystems makes no warranty of merchantability or fitness for a particular purpose. LAB Microsystems shall not be held liable for incidental or consequential damages in connection with or arising out of the furnishing, performance, or use of this product. LAB Microsystems warrants the performance of its products to the specifications applicable at the time of sale. Testing and other quality control techniques are utilized to the extent that LAB Microsystems deems necessary to support this warranty. Specific testing of all functions and all parameters of each product is not necessarily performed, except as mandated by government regulations or requirements. Machine-readable media supplied by LAB Microsystems are guaranteed to load when used with properly adjusted equipment. In case of difficulty, please return any defective media to LAB Microsystems, and it will be replaced at no cost within the first 90 days; after that time, there will be a nominal charge for replacing defective media.

Table of Contents

1. INTRODUCTION.....	6
1.1 Applicable Documents	6
2. QUICK INSTALLATION GUIDE.....	7
2.1 Procedure.....	7
3. FUNCTIONAL DESCRIPTION	8
3.1 Block Diagram.....	8
3.2 Description	8
4. CONNECTORS	8
4.1 32 Bit 33 MHz PCI Edge Connector.....	8
4.2 J1 DSK PCI Interface	10
4.3 J3/J4 DSK Expansion Interface.....	11
5. CONFIGURATION SWITCHES.....	13
6. SROM.....	14
7. PCI DRIVER CODE.....	15
8. PCI DOWNLOAD EXAMPLE	16
8.1 POST Application Modification.....	16
8.2 Process using Hex6x.....	17
8.3 Download Application.....	17
9. SROM PROGRAMMING EXAMPLE	22
10. SCHEMATICS	26
11. SUPPORT	33

Table of Figures

Figure 1 LM1000 Simplified Hardware Architecture Diagram	8
--	---

Table of Tables

Table 1 PCI Edge Connector Pinout.....	9
Table 2 J1 PCI Expansion Connector Pinout.....	10
Table 3 J3 Peripheral Expansion Connector Pinout	11
Table 4 J4 Memory Expansion Connector Pinout	12
Table 5 Configuration Switch Encodings.....	13
Table 6 SROM Memory Map.....	14

1. Introduction

The LM1000 is a low-cost PCI DSK carrier for the Spectrum Digital/TI TMS320C6416 DSK (DSK). By utilizing the LM1000 in conjunction with a DSK, a secondary high speed data path is established which far out paces the standard USB connection. This secondary data path can be used to move real time data between the DSK and the host system and can also be used as a download path for large programs. Since the USB connection is still accessible, full Code Composer Studio (CCS) functionality is maintained.

The LM1000 carrier when coupled with a DSK provides significant value and utility. The additional host-DSP data path, suitable for high-speed real-time data transport, complements the I/O handling features of DSP/BIOS™ and enables a wide range of host-DSP cooperation. The LM1000 provides extreme leverage for your DSK at very low cost, transforming the standalone DSK into a powerful PCI-based development system.

1.1 Applicable Documents

- Spectrum Digital, TMS320C6416 DSK Technical Reference Manual, 505945-0001 Rev A
- Texas Instruments, TMS320C6000 DSP Peripheral Component Interconnect (PCI) Reference Guide, SPRU581A
- Texas Instruments, PCI2050B PCI-to-PCI Bridge Data Manual, June 2003, SCPS076A
- Texas Instruments, PCI2050B Evaluation Module User's Guide, March 2003, SLLU058
- Texas Instruments, TMS Cross-Platform Daughtercard Specification, Revision 0.95

2. Quick Installation Guide

Important Notice

Never connect the DSK power supply when using the LM1000. To do so could cause irreparable damage to the LM1000, DSK or Host system.

2.1 Procedure

This procedure was verified using a Dell computer running Windows XP.

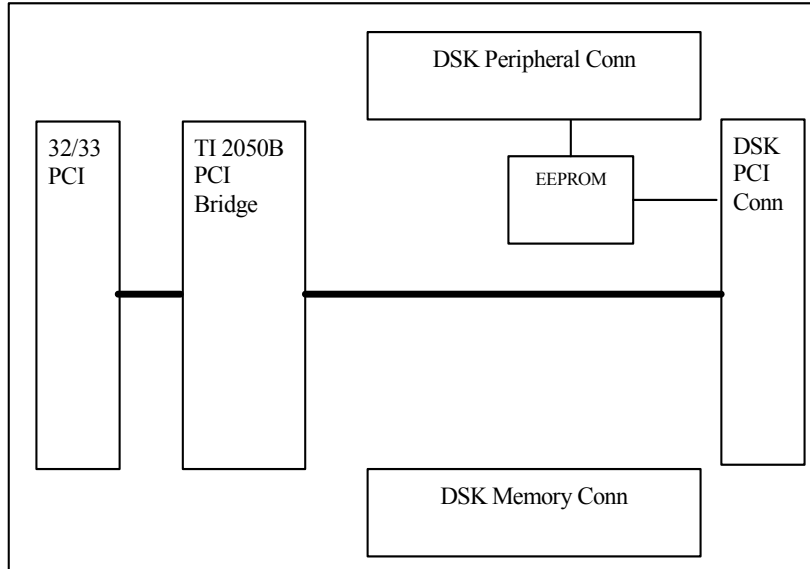
1. Before installing the DSK on to the LM1000, the user should ensure that the DSK functions properly in a stand alone configuration. Follow all the installation instructions provided with the DSK to ensure it functions properly.
2. If the DSK functions properly in the stand alone configuration, proceed to the next step. If it does not function properly, contact TI or Spectrum Digital. Disconnect all power from the DSK and shut down the PC in which the LM1000 is to be installed.
3. Using safe ESD procedures, install the DSK onto the LM1000 using the provided standoffs and screws.
4. Install the LM1000 into any available slot in the PC. The LM1000 exceeds the maximum height specification of PCI cards, so there is a possibility that covers may not fit once the card is installed. Additionally, please ensure that the LM1000 does not contact any other PC cards or metal components in the case. In some cases, a block of nonconductive material may be required to keep the LM1000 isolated from other components. Once the LM1000 is firmly seated in the slot, install a screw to fasten the card bracket to the PC chassis.
5. Install the USB cable between the DSK and the PC.
6. Before powering up the PC, verify that the power supply that came with the DSK is **NOT** connected. The DSK receives all of its power from the LM1000 and does not require a power supply.
7. The PC can now be powered up.
8. During the boot procedure, the PC may identify that new hardware has been found. This hardware will include the PCI bridge chip on the LM1000 and the DSP chip on the DSK. In some systems, the PC will need to download a driver for the PCI bridge. This driver is available from the Microsoft web site. Do not install any driver for the DSP. PCI Driver code will be discussed later in this document. Please note: On some systems, Windows may ask for a driver everytime the system is booted. Again, do not install a driver for DSP.
9. Once the Windows desktop is displayed, launch CCS to access the DSK.

3. Functional Description

3.1 Block Diagram

The functional block diagram of the LM1000 card is as shown in Figure 1.

Figure 1 LM1000 Simplified Hardware Architecture Diagram



3.2 Description

The LM1000 has a very simple hardware architecture which basically consists of a TI PCI2050B Transparent PCI Bridge connected between the host system and the DSK. The bridge provides isolation, buffering and voltage translation between the two PCI domains. There is also a pair of connectors which comply with the TI TMS Cross-Platform Daughtercard Specification, Revision 0.95. These connectors are used to back power the +5V rail on the DSK from the host +5V rail. Finally, the user may install optional connectors on the back of the LM1000 so that other DSK Daughter Cards can be piggybacked on the LM1000.

4. Connectors

4.1 32 Bit 33 MHz PCI Edge Connector

The LM1000 interfaces to the host system through a standard 32 bit 33 MHz PCI edge connector. The edge connector is universal and can be used in 3.3V and 5V signaling environments. The following table provides the pinout of this connector.

Table 1 PCI Edge Connector Pinout

Pin	Description	Pin	Description
A1	-12V	B1	TRST#
A2	TCK	B2	+12V
A3	GND	B3	TMS
A4	TDO	B4	TDI
A5	+5V	B5	+5V
A6	+5V	B6	INTA#
A7	INTB#	B7	INTC#
A8	INTD#	B8	+5V
A9	PRSNT1#	B9	RSVD
A10	RSVD	B10	+VIO
A11	PRSNT2#	B11	RSVD
A12	KEYWAY	B12	KEYWAY
A13	KEYWAY	B13	KEYWAY
A14	RSVD	B14	RSVD
A15	GND	B15	RST#
A16	CLK	B16	+VIO
A17	GND	B17	GNT#
A18	REQ#	B18	GND
A19	+VIO	B19	RSVD
A20	AD31	B20	AD30
A21	AD29	B21	+3.3V
A22	GND	B22	AD28
A23	AD27	B23	AD26
A24	AD25	B24	GND
A25	+3.3V	B25	AD24
A26	C/BE3#	B26	IDSEL
A27	AD23	B27	+3.3V
A28	GND	B28	AD22
A29	AD21	B29	AD20
A30	AD19	B30	GND
A31	+3.3V	B31	AD18
A32	AD17	B32	AD16
A33	C/BE2#	B33	+3.3V
A34	GND	B34	FRAME#
A35	IRDY#	B35	GND
A36	+3.3V	B36	TRDY#
A37	DEVSEL#	B37	GND
A38	GND	B38	STOP#
A39	LOCK#	B39	+3.3V
A40	PERR#	B40	SDONE
A41	+3.3V	B41	SBO#
A42	SERR#	B42	GND
A43	+3.3V	B43	PAR
A44	C/BE1#	B44	AD15
A45	AD14	B45	+3.3V
A46	GND	B46	AD13
A47	AD12	B47	AD11
A48	AD10	B48	GND
A49	M66EN	B49	AD09

A50	KEYWAY	B50	KEYWAY
A51	KEYWAY	B51	KEYWAY
A52	AD08	B52	C/BE0#
A53	AD07	B53	+3.3V
A54	+3.3V	B54	AD06
A55	AD05	B55	AD04
A56	AD03	B56	GND
A57	GND	B57	AD02
A58	AD01	B58	AD00
A59	+VIO	B59	+VIO
A60	ACK64#	B60	REQ64#
A61	+5V	B61	+5V
A62	+5V	B62	+5V

4.2 J1 DSK PCI Interface

The J1 connector is used to connect the secondary interface of the PCI2050 PCI bridge to the PCI interface of the DSK. There are several additional signals, exclusive of the PCI signals, which are used to configure the DSK. These signals are described in the following paragraph.

PCI_EN is connected to SW1 position 1. MCBSP2_EN is connected to SW1 position 2. The following table provides the pinout of this connector.

Table 2 J1 PCI Expansion Connector Pinout

Pin	Description	Pin	Description
1	PCI_EN	2	MCBSP2_EN
3	GND	4	HPI_RESET
5	XSP_CS	6	TBEA13
7	GND	8	GND
9	PAD1	10	PCBE0
11	PAD3	12	PAD0
13	PAD5	14	PAD2
15	PAD7	16	PAD4
17	GND	18	PAD6
19	PAD8	20	GND
21	PAD10	22	PAD9
23	PAD12	24	PAD11
25	PAD14	26	PAD13
27	PGND	28	PAD15
29	PCBE1	30	GND
31	GND	32	PPAR
33	PSERR	34	GND
35	GND	36	PSTOP
37	PPER	38	GND
39	GND	40	PTRDY
41	PDEVSEL	42	GND
43	GND	44	PFRAME
45	PIRDY	46	GND

47	GND	48	PAD16
49	PCBE2	50	PAD18
51	PAD17	52	PAD20
53	PAD19	54	PAD22
55	PAD21	56	GND
57	PAD23	58	PIDSEL
59	PCBE3	60	PAD24
61	GND	62	PAD26
63	PAD25	64	PAD28
65	PAD27	66	PAD30
67	PAD29	68	PGNT
69	PAD31	70	GND
71	GND	72	PRST
73	PREQ	74	GND
75	GND	76	PINTA
77	PCLK	78	GND
79	GND	80	NC

4.3 J3/J4 DSK Expansion Interface

The J3/J4 connectors are used to back power the +5 volt supply rail of the DSK from the +5 volt rail of the host PCI bus. This eliminates the need for the DSK power supply.

Optional connectors can be installed at locations J6 and J7 on the back of the LM1000. With these connectors installed, DSK compatible daughterboards can be installed on the back side of the LM1000. The following tables provide the pinout of these connectors.

Table 3 J3/J6 Peripheral Expansion Connector Pinout

Pin	Description	Pin	Description
1	12 V	2	-12 V
3	GND	4	GND
5	5 V	6	5 V
7	GND	8	GND
9	5 V	10	5 V
11	NC	12	NC
13	NC	14	NC
15	NC	16	NC
17	NC	18	NC
19	3.3 V	20	3.3 V
21	CLKX0	22	CLKS0
23	FSX0	24	DX0
25	GND	26	GND
27	CLKR0	28	NC
29	FSR0	30	DR0
31	GND	32	GND
33	CLKX1	34	CLKS1
35	FSX1	36	DX1
37	GND	38	GND

39	CLKR1	40	NC
41	FSR1	41	DR1
43	GND	44	GND
45	TOUT0	46	TINP0
47	NC	48	EINT5
49	TOUT1	50	TINP1
51	GND	52	GND
53	EINT4	54	NC
55	NC	56	NC
57	NC	58	NC
59	RST#	60	NC
61	GND	62	GND
63	CNTL1	64	CNTL0
65	STAT1	66	STAT0
67	EINT6	68	EINT7
69	CE3	70	NC
71	NC	72	NC
73	NC	74	NC
75	GND	76	GND
77	GND	78	ECLKOUT
79	GND	80	GND

Table 4 J4/J7 Memory Expansion Connector Pinout

Pin	Description	Pin	Description
1	5 V	2	5 V
3	A21	4	A20
5	A19	6	A18
7	A17	8	A16
9	A15	10	A14
11	GND	12	GND
13	A13	14	A12
15	A11	16	A10
17	A9	18	A8
19	A7	20	A6
21	5 V	22	5 V
23	A5	24	A4
25	A3	26	A2
27	BE3#	28	BE2#
29	BE1#	30	BE0#
31	GND	32	GND
33	D31	34	D30
35	D29	36	D28
37	D27	38	D26
39	D25	40	D24
41	3.3 V	42	3.3 V
43	D23	44	D22
45	D21	46	D20
47	D19	48	D18
49	D17	50	D16
51	GND	52	GND
53	D15	54	D14

55	D13	56	D12
57	D11	58	D10
59	D9	60	D8
61	GND	62	GND
63	D7	64	D6
65	D5	66	D4
67	D3	68	D2
69	D1	70	D0
71	GND	72	GND
73	ARE#	74	AWE#
75	AOE#	76	ARDY#
77	CE3#	78	CE2#
79	GND	80	GND

5. Configuration Switches

The LM1000 provides a single 4 position DIP switch (SW1) to select the configuration of the board. For normal operation, these switches should all be set to the off position. The following table describes the switches and their function.

Table 5 Configuration Switch Encodings

Switch Position	Description
Switch 1	PCI Enable =On PCI Disabled =Off PCI Enabled* (default)
Switch 2	MCBSP2 Enable =On MCBSP2 Disabled =Off MCBSP2 Enabled* (default)
Switch 3	Not Used
Switch 4	Not Used

To turn a switch on, slide its actuator away from the card bracket. To turn a switch off, slide its actuator toward the card bracket.

6. Serial EEPROM

The LM1000 provides a serial EEPROM (U2) which can be used to initialize the DSP's PCI configuration registers with user data. As shipped, this device is disabled but programmed with default values. To enable the device, switch position 2 of SW1 on the LM1000 to the on position.

Should this device be programmed with incorrect data such that the DSK cannot be accessed using the example software, set switch position 2 of SW1 on the LM1000 to the off position and reboot the system. This forces the DSP to use its internal values for initialization of the PCI configuration registers. After the system is booted, set switch position 2 of SW1 on the LM1000 to the on position. This enables the DSP to access the EEPROM.

Table 6 EEPROM Memory Map

Location`	Contents (msb/lbs)	DSP/EEPROM Default Values
0x0	Vendor ID	0x104C
0x1	Device ID	0xA106
0x2	Class Code[7..0] / Revision	0x0001
0x3	Class Code [23:8]	0x0000
0x4	Subsystem Vendor ID	0x0000
0x5	Subsystem ID	0x0000
0x6	Max Latency / Min Grant	0x0000
0x7	PC D1 / PC D0	0x0000
0x8	PC D3 / PC D2	0x0000
0x9	PD D1 / PD D0	0x0000
0xA	PD D3 / PD D2	0x0000
0xB	Data Scale (PD D3..PC D0)	0x0000
0xC	0x00 / PMC [14:9] / PMC [5] / PMC [3]	0x0000
0xD	Checksum	0x1be1

7. PCI Driver Code

The LM1000 is not supplied with a specific driver for Windows or other operating systems. The LM1000 is, however, compatible with generic libraries available from third parties such as Jungo, Tetradyne and ZealSoft, all of which can be found on the internet. These libraries can be used to access PCI hardware without the need for a specific driver.

The following two examples demonstrate how to access the DSK using the PCI interface. Both examples use the ZealSoft Studio MemAccess library to access the hardware. An evaluation version of this library can be found on the CD or can be downloaded from the ZealSoft Studio website. When installed, this library can be accessed through an API and provides access to PCI devices. The MemAccess Library was chosen for its simplicity, ease of use and cost. Both examples were built using Visual Studio Version 6.0.

The first example downloads a DSP application built using Code Composer Studio (CCS). This application downloads to internal SRAM of the 6416 DSP. Once the application is downloaded, the DSP boots and runs the application. The second example demonstrates how to program the serial EEPROM on the LM1000.

8. PCI Download Example

This example program downloads a DSP application to the internal SRAM of the DSP on the DSK. In order to configure the DSK to boot from the PCI interface, positions 2 and 3 of SW3 on the DSK must be set to the on position.

Several steps need to be performed in order for the download to execute properly. First, the POST application needs to be modified and built using CCS. Second, the hex6x utility needs to be run to process the post.out COFF file which was built using CCS, into a format that can be downloaded. And third, the download application needs to be built using Visual Studio 6.0.

8.1 POST Application Modification

In order to prepare the POST application for download via the PCI interface, it needs to be modified. As found on the DSK CD, the POST application is meant to be programmed and booted from the DSK FLASH memory. In our example, the DSP will not be booting from FLASH, but from internal SRAM. To support a direct boot from internal SRAM, the source file boot.asm needs to be modified. Boot.asm is written to copy the POST application from FLASH memory to internal SRAM. Since in our example, the POST application is downloaded directly into internal SRAM, this copy process is not required.

To modify the file, open the POST application project from within CCS. This project is found in the <ti\examples\dsk6416\bsl\post> directory. In the left hand window, expand post.pjt until the source files boot.asm and post.c are shown. Open boot.asm so that it can be edited. Before editing, be sure to save the file as bootold.asm.

Edit boot.asm so that it reads as follows:

```
.title    "LM1000 PCI Boot Code for 6416DSK"
.sect     ".boot_load"
.global  _boot
.ref     _c_int00
_boot:
mvkl    .S2 _c_int00, b0
mvkh    .S2 _c_int00, b0
b       .S2 b0
nop     5
```

The label <_boot:> should start in column1, otherwise it will be interpreted as an illegal instruction. Save the file and then rebuild the application using CCS. After compilation, run the application to ensure that it was built correctly using the <Load Program> option of CCS. If everything is OK, the LEDs on the DSK will cycle as the POST tests are being performed and then flash when the test is complete. If the LEDs do not flash at the end of the test, check to make sure that switch 2 of SW1 on the LM1000 is set to the off position. With switch 2 in the on position, MCBSP2 is disabled which causes the MCBSP2 test of the POST application to fail.

8.2 Process using Hex6x

The next step is to take the post.out file created in \debug directory of the POST project directory and process it using the hex6x utility. This utility can be found in the <ti\c6000\cgtools\bin> This utility is used to create an ASCII file which is more suitable for download by the download example. It should be noted that ASCII files created by hex6x are limited to no more than 64K of code since the file structure only supports 16 bit addressing.

Before running hex6x, a command file needs to be created to pass to parameters to the utility. Create a new file called post.cmd so that it reads as follows:

```
post.out
-a
-memwidth 32
-romwidth 32
-bootsection .boot_load 0x00000000
-map post.map
```

Move a copy of post.out, hex6x.exe and post_lm1k.cmd to a new directory. From a command prompt within this directory, execute the following command:

```
hex6x post.cmd
```

This will create a file called post.a0 which is the file we will download to the target, as well as a file called post.map which is a map file showing where the different sections of the application will be loaded.

8.3 Download Application

The download application was written for a command prompt using Visual Studio 6.0 and the ZealSoft MemAccess Library.

The basic flow of the download application is as follows:

1. The PCI driver library needs to be opened and a link established to the LM1000 DSK Carrier.
2. The physical address mapping (BAR register values) of the DSP need to be retrieved.
3. The physical addresses then need to be mapped to a ring-0 linear addresses.
4. The download file needs to be opened and prepared for processing.
5. The DSP is warm reset which places the DSP core in reset, yet allows the DSP's internal peripherals (PCI interface) to be accessed.
6. The code is then downloaded into internal SRAM.
7. The DSP core is allowed to boot.

The following source code is an example of a very simple download program.

```
/* *****  
pci_boot.c  
  
This program parses an ASCII file generated using hex6x and down-  
loads it to a 6416DSK using the LM1000 carrier and the Memaccess  
Library.  
  
This source code is provided free of charge. NO warranties are made  
either expressed or implied. Use at your own risk.  
  
Copyright 2004 LAB Microsystems, LLC  
  
***** */  
  
#include "stdio.h"  
#include "stdlib.h"  
#include "memacc.h"  
  
// define DSP PCI register offsets  
  
#define HSR ((0x1c1fff0 - 0x1800000) >> 2); // host status reg  
#define HDCR ((0x1c1fff4 - 0x1800000) >> 2); // host-dsp ctrl reg  
#define DSFP ((0x1c1fff8 - 0x1800000) >> 2); // dsp page reg  
  
// function prototypes  
  
int asciitohex(int);  
  
// start of program  
  
int main(int argc, char* argv[])  
{  
    BADDR          paddr[6];  
    HANDLE         hmem0,hmem1;  
    PVOID          pcireg,pcimem;  
    FILE           *fp;  
    BOOL           rval;  
  
    char           fname[81];  
    int            i,c;  
    unsigned int   aoff,anib;  
    unsigned short vid,did;  
    unsigned long  atemp,adata,index;  
    unsigned long  *mptr,*rptr;  
  
    printf("Example Boot Loader");  
    printf("\nCopyright 2004 LAB Microsystems, LLC");  
    printf("\nThis software is provided free of charge with no warranties");  
    printf("\nneither expressed or implied. Use at your own risk.\n\n");  
  
    // check for board using memaccess library  
  
    rval = maOpenLibrary();  
  
    if (0 == rval)  
    {  
        printf("\nError opening Memaccess library...");  
        exit(0);  
    }  
  
    // look for first occurrence of LM1000/6416DSK  
  
    vid = 0x104c; // vendor ID of 6416 DSP  
    did = 0x106; // device ID of 6416 DSP  
    index = 0; // first board occurrence  
  
    rval = maGetDeviceBaseAddress(&vid,&did,index,paddr);  
  
    switch (rval)  
    {  
    case 1: // device found  
  
        printf("\nLM1000/TMS6416 DSK found!");  

```

```
// check all 6 potential BARs
for (i=0;i<6;i++)
{
    if (0x2 == paddr[i].IOType)
        continue;

    printf("\nBAR %d base = %08lx size = %08lx type = %08lx",
        i,paddr[i].BaseAddress,paddr[i].Size,paddr[i].IOType);
}

// map BAR0
pcimem = maMapPhysToLinear(paddr[0].BaseAddress,
                           paddr[0].Size,
                           &hmem0);

if (NULL == pcimem)
{
    printf("\nMapping of BAR 0 failed...");
    exit(0);
}

// map BAR1
pcireg = maMapPhysToLinear(paddr[1].BaseAddress,
                           paddr[1].Size,
                           &hmem1);

if (NULL == pcireg)
{
    printf("\nMapping of BAR 1 failed...");
    exit(0);
}

break;

case 2: // Error
{
    printf("\nError getting base address!");
    exit(0);
}

break;

case -1: // No Device Found
{
    printf("\nNo device found...");
    exit(0);
}

break;
}

printf("\n\nEnter file to download: ");
scanf("%s",fname);

fp = fopen(fname,"r");

if (NULL == fp)
{
    printf("\nError opening file: %s\n",fname);
    exit(0);
}

// reset the LM1000/DSK6416
```

```
printf("\nResetting DSK...");

rptr = (unsigned long *) pcireg + HDCR;
*rptr = 1;

// set the DSP page register to 0
rptr = (unsigned long *) pcireg + DSPP;
*rptr = 0;

// download the code
printf("\nDownloading Code...");

aoff = 0; /* set offset to start */
mptr = (unsigned long *) pcimem;
while (EOF != (c = fgetc(fp)))
{
    switch (c)
    {
        case (0x02): // start character
            break;

        case '\n': // ignore CRLF
            break;

        case ',': // ignore commas
            break;

        case (0x20): // ignore spaces
            break;

        case (0x03): // end character
            break;

        case '$': // address record found
            c = fgetc(fp); /* get the A(a) */
            aoff = 0; // clear the offset
            anib = 0; // clear address nibble

            for (i=0;i<4;)
            {
                c = fgetc(fp);

                if (0x20 == c) /* if white space, then get next character */
                    continue;

                anib = asciitohex(c);
                anib = anib << (4 * (3 - i)); // shift to its proper position
                aoff = aoff + anib;

                i++;
            }

            mptr = (unsigned long *) pcimem + (aoff >> 2);
            break;

        default: // otherwise download the code 32 bits at a time
            adata = 0;
            atemp = 0;
```

```
        // already have first nibble so process it
        atemp = asciitohex(c);
        atemp = atemp << 28;

        adata = adata + atemp;

        // get next 7 nibbles
        for (i=0;i<7;)
        {
            c = fgetc(fp);

            if (0x20 == c) /* if white space, then get next character */
                continue;

            atemp = asciitohex(c);
            atemp = atemp << (4 * (6 - i));

            adata = adata + atemp;

            i++;
        }

        *mptr++ = adata;

        break;
    }
}

// boot the DSP
printf("\nBooting DSK...\n\n");

rptra = (unsigned long *) pcireg + HDCR;
*rptra = 0x2;

// cleanup

fclose (fp);
maUnmapPhysicalMemory(hmem0,pcimem);
maUnmapPhysicalMemory(hmem1,pcireg);
maCloseLibrary();

return 0;
}

// this function converts an ASCII char to a hex nibble, if not
// a valid hex character, it returns the character

int asciitohex(int achar)
{
    if ((achar >= 0x30) && (achar <= 0x39))
        return (achar - 0x30);

    if ((achar >= 0x41) && (achar <= 0x46))
        return (achar - 0x37);

    if ((achar >= 0x61) && (achar <= 0x66))
        return (achar - 0x57);

    return(achar);
}
```

9. SRAM Programming Example

This example program is used to program the serial EEPROM on the LM1000. In this program, there is no DSP application that needs to be downloaded. The program accesses the DSP's EEPROM control registers directly through the PCI interface.

Please Note: Before running this program, the user should check to ensure that switch position 2 of SW1 on the LM1000 is set to the on position. This disables MCBSP2 and enables access to the serial EEPROM.

```

/* *****
srom.c

This example program programs and verifies the PCI eeprom on the
LM1000.

This source code is provided free of charge. NO warranties are made
either expressed or implied. Use at your own risk.

Copyright 2004 LAB Microsystems, LLC

***** */

#include "stdio.h"
#include "stdlib.h"
#include "memacc.h"

// define DSP internal register offsets

#define HDCR ((0x1c1fff4 - 0x1800000) >> 2); // host-dsp ctrl reg
#define EEADD ((0x1c20000 - 0x1800000) >> 2); // srom addr reg
#define EEDAT ((0x1c20004 - 0x1800000) >> 2); // srom data reg
#define EECTL ((0x1c20008 - 0x1800000) >> 2); // srom control reg

// start of program

int main(int argc, char* argv[])
{
    BADDR          paddr[6];
    HANDLE         hmem0,hmem1;
    PVOID          pcireg,pcimem;
    BOOL          rval;

    int            i;
    unsigned int   cksum;
    unsigned int   index;
    unsigned short vid,did;
    unsigned long  *rptr;

    // table to program into SRAM

    unsigned int   srom_tab[14] = {0x104c,0xa106,0x0001,0x0000,
                                   0x0000,0x0000,0x0000,0x0000,
                                   0x0000,0x0000,0x0000,0x0000,
                                   0x0000,0x0000};

    printf("Example EEPROM Programmer");
    printf("\nCopyright 2004 LAB Microsystems, LLC");
    printf("\nThis software is provided free of charge with no warranties");
    printf("\neither expressed or implied. Use at your own risk.\n");

    // check for board using memaccess library

    rval = maOpenLibrary();

```

```
if (0 == rval)
{
    printf("\nError opening Memaccess library...");
    exit(0);
}

// look for first occurrence of LM1000/6416DSK

vid = 0x104c; // vendor ID of 6416 DSP
did = 0xa106; // device ID of 6416 DSP
index = 0;    // first board occurrence

rval = maGetDeviceBaseAddress(&vid,&did,index,paddr);

switch (rval)
{
case 1: // device found

    printf("\nLM1000/TMS6416 DSK found!");

    // check all 6 potential BARs

    for (i=0;i<6;i++)
    {
        if (0x2 == paddr[i].IOType)
            continue;

        printf("\nBAR %d base = %08lx size = %08lx type = %08lx",
            i,paddr[i].BaseAddress,paddr[i].Size,paddr[i].IOType);
    }

    // map BAR0 to linear address

    pcimem = maMapPhysToLinear(paddr[0].BaseAddress,
                                paddr[0].Size,
                                &hmem0);

    if (NULL == pcimem)
    {
        printf("\nMapping of BAR 0 failed...");
        exit(0);
    }

    // map BAR1 to a linear address

    pcireg = maMapPhysToLinear(paddr[1].BaseAddress,
                                paddr[1].Size,
                                &hmem1);

    if (NULL == pcireg)
    {
        printf("\nMapping of BAR 1 failed...");
        exit(0);
    }

    break;

case 2: // Error
    {
        printf("\nError getting base address!");
        exit(0);
    }

    break;

case -1: // No Device Found
    {
        printf("\nNo device found...");
        exit(0);
    }
}
```

```
        break;
    }

    // reset the dsk
    printf("\n\nResetting DSK...");

    rptr = (unsigned long *) pcireg + HDCR;
    *rptr = 1;

    printf("\n\nProgramming srom...");
    // do a pseudo read to wake it up
    rptr = (unsigned long *) pcireg + EECTL;
    *rptr = 2;

    // write enable srom so we can program it
    printf("\n\nUnlocking device...");

    rptr = (unsigned long *) pcireg + EEADD;
    *rptr = (unsigned long) 0xc0;

    rptr = (unsigned long *) pcireg + EECTL;
    while (0 == (*rptr & 0x4));
    *rptr = 0x00;

    // write the table
    printf("\n\nProgramming device");

    cksum = 0xaaaa;

    for (i=0;i<0xe;i++)
    {
        rptr = (unsigned long *) pcireg + EEADD;
        *rptr = (unsigned long) i;

        rptr = (unsigned long *) pcireg + EEDAT;
        if (0xd == i)
            *rptr = cksum; // last word is check sum
        else
            *rptr = srom_tab[i];

        rptr = (unsigned long *) pcireg + EECTL;
        *rptr = 0x01;
        while (0 == (*rptr & 0x4));
        cksum = cksum ^ srom_tab[i];
        printf(".");
    }

    // verify device
    printf("\n\nVerifying device");

    cksum = 0xaaaa;

    for (i=0;i<0xe;i++)
    {
        rptr = (unsigned long *) pcireg + EEADD;
        *rptr = (unsigned long) i;
```

```
rptr = (unsigned long *) pcireg + EECTL;
*rptr = 0x02;
while (0 == (*rptr & 0x4)); // wait for command to complete
rptr = (unsigned long *) pcireg + EEDAT;
if (0xd != i)
{
    if (srom_tab[i] != *rptr)
    {
        printf("\nVerify Error Addr %02x Read %04x Wrote %04x",
            i, *rptr, srom_tab[i]);
    }
}
else
{
    if (cksum != *rptr)
    {
        printf("\nCheck sum error Read %04x Computed %04x",
            *rptr, cksum);
    }
}
cksum = cksum ^ (unsigned int) *rptr;
printf(".");
}

printf("\nDone...\n");
maUnmapPhysicalMemory(hmem0, pcimem);
maUnmapPhysicalMemory(hmem1, pcireg);
maCloseLibrary();

return 0;
}
```

10. Schematics

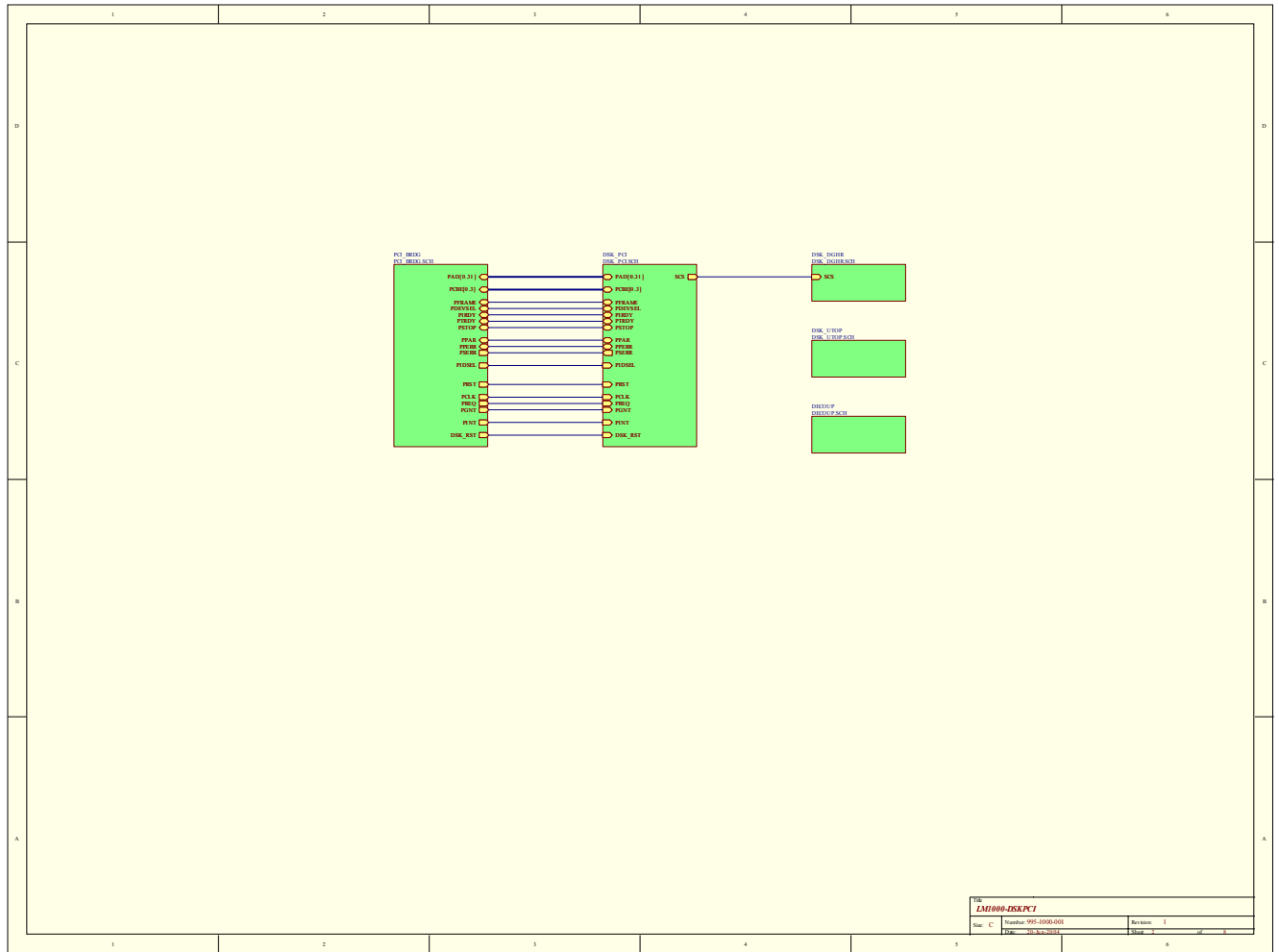
Revision			
Rev	Description	Date	Approved

Notes:
 1. Resistance values are ohms.
 2. Capacitance values are microfarads unless otherwise noted.
 3. Inductance values are microhenries unless otherwise noted.
 4.

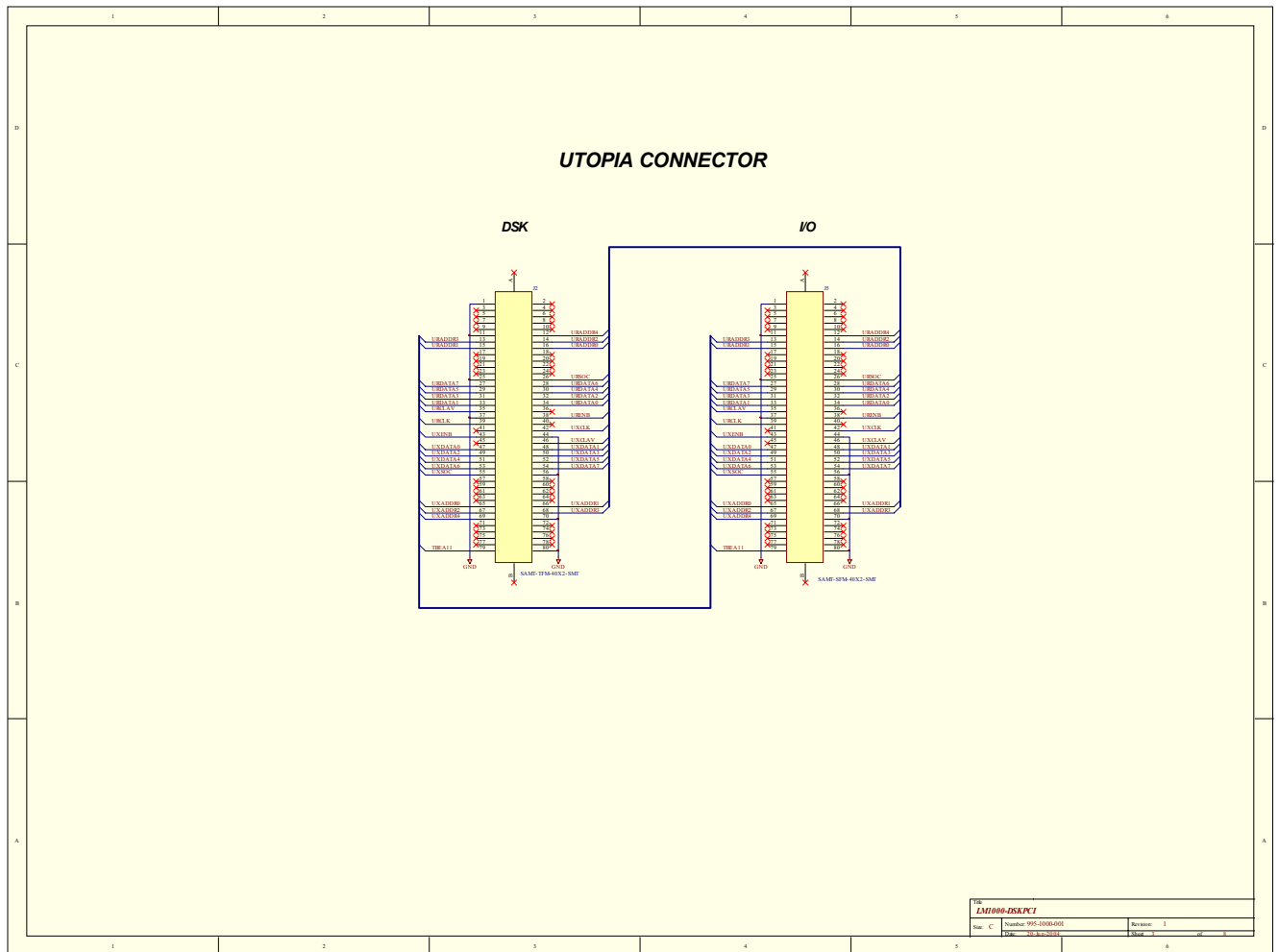
Copyright 2003 LAB Microsystems, LLC.
Confidential - Do not disclose without permission of LAB Microsystems.

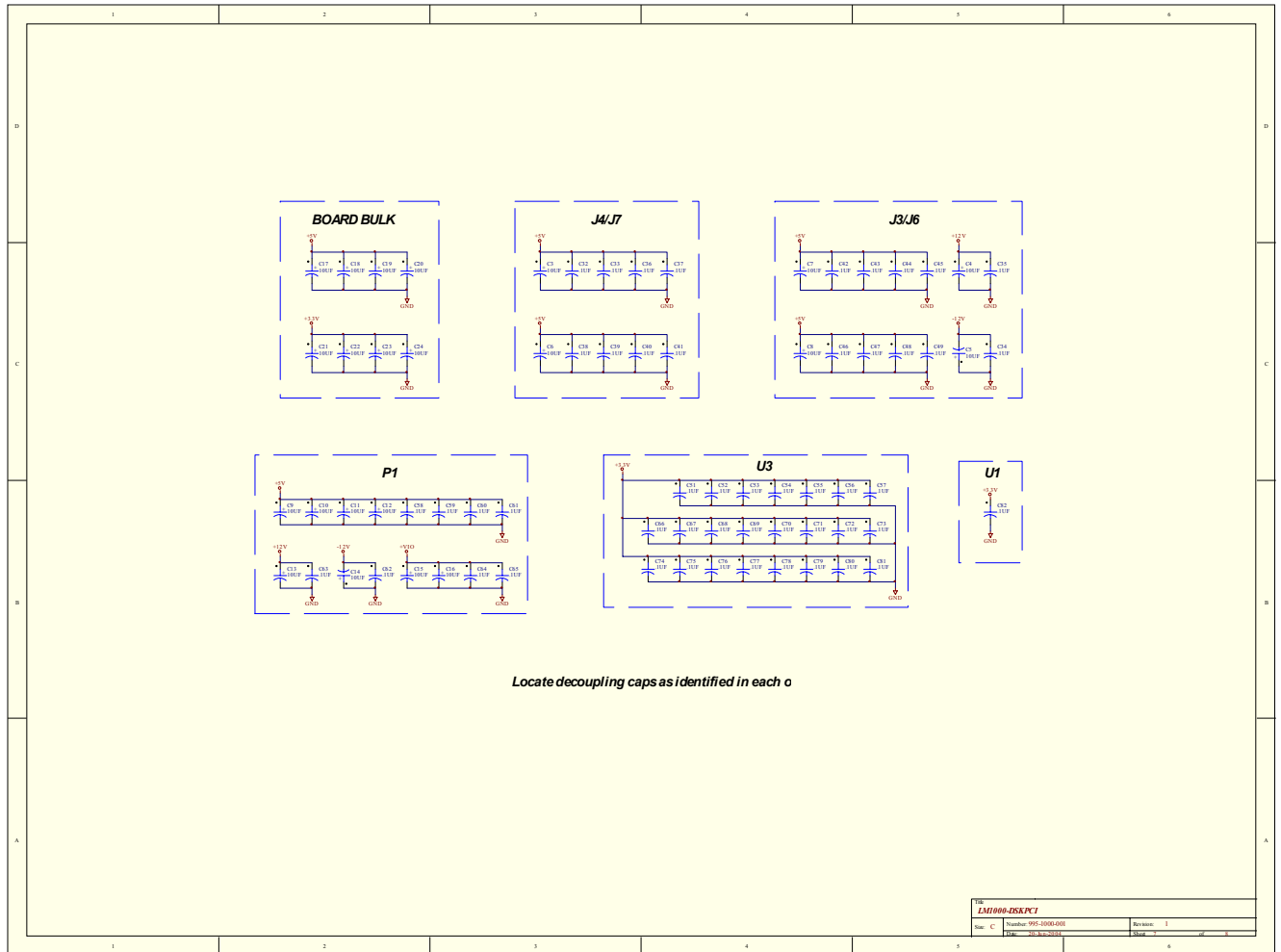
LAB Microsystems, LLC
 87 Siles Farm Road
 Williston, NH 03096 USA
 603-654-6899
 info@labmicrosystems.com

Design	Date	Part: LM1000-DBKPC1 Number: 995-000-000 Date: 28-Nov-2003	
Approved	Date	Spec: C	Revision: 1



Title		Revision	
LM1000-DSK/PC1		1	
Sheet	Number	Page	of
C	95-1000-001	1	1





11. Support

The best way to contact LAB Microsystems, LLC for support is via email. Please send support requests to support@labmicrosystems.com. We try to respond to all support requests within 24 hours during normal business hours.