



Data Design Corporation  
Gaithersburg, MD

# **C64DSP CPCI MODULE USER ADAPTABLE DIGITAL SIGNAL PROCESSOR**

**May 2006**

**Data Design Corporation**  
7851-A Beechcraft Avenue  
Gaithersburg, MD 20879  
[WWW.DATADESIGNCORP.NET](http://WWW.DATADESIGNCORP.NET)  
(301) 670-1157

# CONTENTS

1.0 Introduction .....	3
1.1 Specifications Summary .....	3
1.2 Setup and Installation .....	4
1.3 Front Panel And External Interfacing .....	5
2.0 Using C64DSP Software .....	6
2.1 Memory Access Functions.....	7
2.2 Code Loading Functions.....	8
2.3 EEPROM Features .....	9
3.0 C64DSP Software Source Code .....	11
3.1 C64DSP Software Architecture.....	12
3.2 The C64DSP API.....	13
3.2.1 Calibration Functions .....	15
3.2.2 General Functions.....	16
3.2.3 DSP Memory Space Access Functions.....	17

# 1.0 Introduction

The C64DSP provides the digital signal processing power of the high end Texas Instruments (TI) TMS320C6416 DSP chip with a complement of 128Mbyte of SDRAM and a fully outfitted expansion connector, providing a ready platform for rapid integration of DSP applications. The expansion connector includes the entire second external memory interface (EMIFB), both serial ports, both timers, and a complement of digital I/O and power supplies. A provided software application offers the ability to load code, control processor operation, and transfer data to or from any area of DSP memory over the CPCI backplane. A complete schematic of the C64DSP is included on the CD along with driver and application source code. Because there is no need to ever run DSP code to use the features of the DSP chip from the CPCI host, at one level the C64DSP provides a simple CPCI development system. At another level the C64DSP is a powerful DSP coprocessor and configurable I/O module.

It is important to note that the C64DSP product is essentially a repackaging of a TI product on a commonly used industrial computing footprint. There is no existing application associated with the C64DSP module as provided alone and no proprietary components to understand or protect. The value position is associated with the packaging and a driver software suite not provided in this combination by any other vendor. Because the majority of the complexity is associated with the TI product, this manual is kept short to cover only the items associated with the packaging and driver software. The user should become familiar with the architecture of the processor through the TI documentation. Generally from the point of view of the experienced user, the C64xx series of processors is very functional and forgiving, and very well documented. The data sheet for the C6416, available on the TI website, is the starting point. However, TI strategically takes to documenting each subsystem separately and has carefully built a product for which unused subsystems do not interfere with the operation of those of interest. For example, the user should seek out documentation of the EMIF architecture, the serial ports, or the general purpose I/O in separate TI application guides. The C64DSP product is well suited to easily access all of these features from a CPCI host and also allows DSP code to be loaded and run locally for access to features from within the DSP processor. The C64DSP can be used with compiler tools and JTAG debug tools available from TI and other vendors.

## 1.1 Specifications Summary

### Dataway Interface

Compact PCI at 32 Bits and 33 MHz  
Compliant with PICMG 2.0 R3.0  
Single Width 3U Compact PCI Card

### Ambient Temperature Range

0 To 70 C

### Processor

TMS320C6416 at 1GHz

### Memory

128 Megabytes organized 32M x 64

### Interface

EMIFB, Serial Ports, Timers, GPIO  
Voltages including +3.3V,+5V,-5V,+12V,-12V

### Interface Connectors

Samtec FSI-130-10-L-D-M-AD

## 1.2 Setup and Installation

The C64DSP is a single width 3U CPCI module. To insert the C64DSP in the CPCI crate, choose any convenient slot and slide the module into the crate with its top and bottom mated to the guide rails. Be sure that the module is properly aligned with the connector at the back of the crate. Push the module toward the back of the crate with gentle pressure. When the card pushes back, push the locking lever down and press the card into the crate with firm pressure on the top and bottom of the front panel. The locking lever will raise and lock into place. Fasten the retaining screws at the upper end of the front panel and at the bottom of the front panel below the lever. To remove the card, reverse the process by removing the retaining screws and pressing down firmly on the locking lever to disengage the card from the connector at the rear of the crate.

**NOTE: The module should be inserted in the crate only after turning off the power to the crate. Otherwise, damage to the module or system is possible due to momentary misalignment of pins on the connector. CPCI hot swap operation is not supported.**

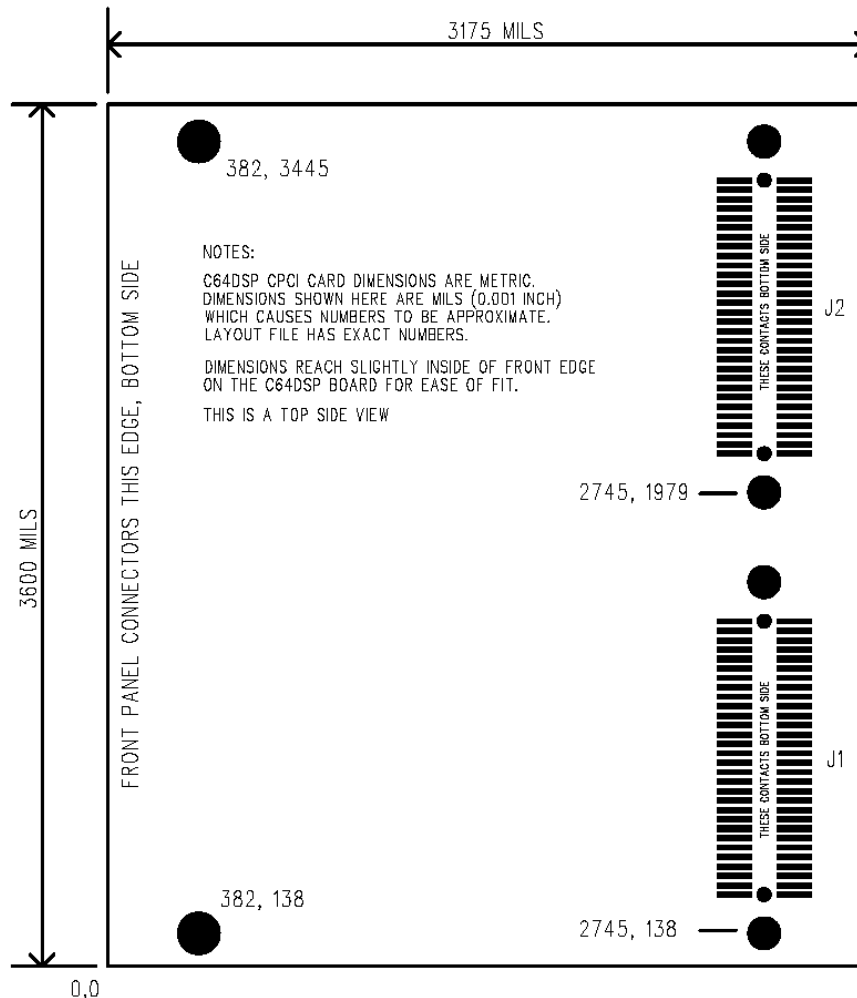
It is recommended that the C64DSP module and a controller be installed alone in the crate until the user is familiar with operation enough to integrate it with other modules in an instrumentation system. When the C64DSP is installed in a slot where it has not been before, the CPCI system will detect it as a new device at which point it will need access to driver software. The software supplied with the C64DSP is designed to operate on platforms with Microsoft Windows 2000 and newer operating systems. To install drivers on these systems, point the installation wizard to the location of the included CD. A CD drive may be located on the crate controller or it may be mounted over a network connection using the wizard's "specify a location" option. The files the installation wizard will need are located in the root directory of the CD. Follow the steps displayed by the installation wizard at the end of which the wizard should indicate that a "C64DSP" has been installed.

To install the C64DSP software, use the Windows explorer or similar tool to locate the software CD. Point to the directory on the CD identified as \C64DSP\Install. Run the program SETUP.EXE and follow the on screen instructions for the familiar software installation process. When any new software has just been installed, it is usually best to reboot the crate controller before attempting to use the software, even when this is not strictly necessary. At this point the installation is complete and the C64DSP and turnkey software are ready to be used.

### 1.3 Front Panel And External Interfacing

The C64DSP front panel is a blank slate with no cutouts or labeling. External interfacing of signal inputs and outputs is intended to be built into an expansion board. A mechanical outline of the expansion board is shown in Figure 1.1 below. A layout template in Mentor Power PCB format is included on the CD.

Connection to the C64DSP expansion connector is a surface contact area as described by Samtec for the mating requirements of the connector specified in section 1.1 above. These connection points are included in the layout template. The definition of each terminal on the interface connectors is contained within the schematic drawing for the C64DSP included on the CD and is not repeated here for the sake of documentation consistency and to encourage the user to examine the schematic. Connectors for external signal termination will generally be mounted on the bottom side of the expansion board at the front edge where the front panel bracket will be modified to accept them. Mounting spacers required on the two front mounting points are ten millimeters in height and are generally specified to accept an M2 screw to match that required by the threaded inserts integral to the connectors.



**Figure 1.1** C64DSP Expansion Board Dimensions

## 2.0 Using C64DSP Software

The C64DSP is provided with a turnkey software package which allows the user to quickly take advantage hardware features in a standalone configuration. This software is designed to coexist with other instruments as well as to allow multiple C64DSP modules to be installed and operated in the same CPCI crate. Installation of the software is discussed in section 1.2 above. The provided software is not intended to serve any instrument function of its own, but to serve as a debug tool allowing DSP hardware to be probed and code to be loaded and executed. The software is therefore essentially a test dialog which provides full access to DSP memory space.

Upon starting the C64DSP software, a splash screen will be displayed while the software searches for a C64DSP module in the system. The software will locate and configure a C64DSP module to default settings and complain if a none can be found. Finally, the test dialog screen shown in Figure 2.1 will appear. This dialog allows the user to access any C64DSP modules in the system. The specific module being accessed is identified by the serial number as shown on the module labeling. To select a different module, click on the *Serial Number* display. If more than one module is in the system, the next module will be found. The *Product ID* display shows the PCI product identifier being used by the module as recorded in the EEPROM memory. This is a secondary indication from the view point of the test dialog and is discussed later in this manual. No more than one application should attempt to control any one C64DSP module in the system at the same time. The C64DSP module is reset through the CPCI bus upon opening or closing the application.

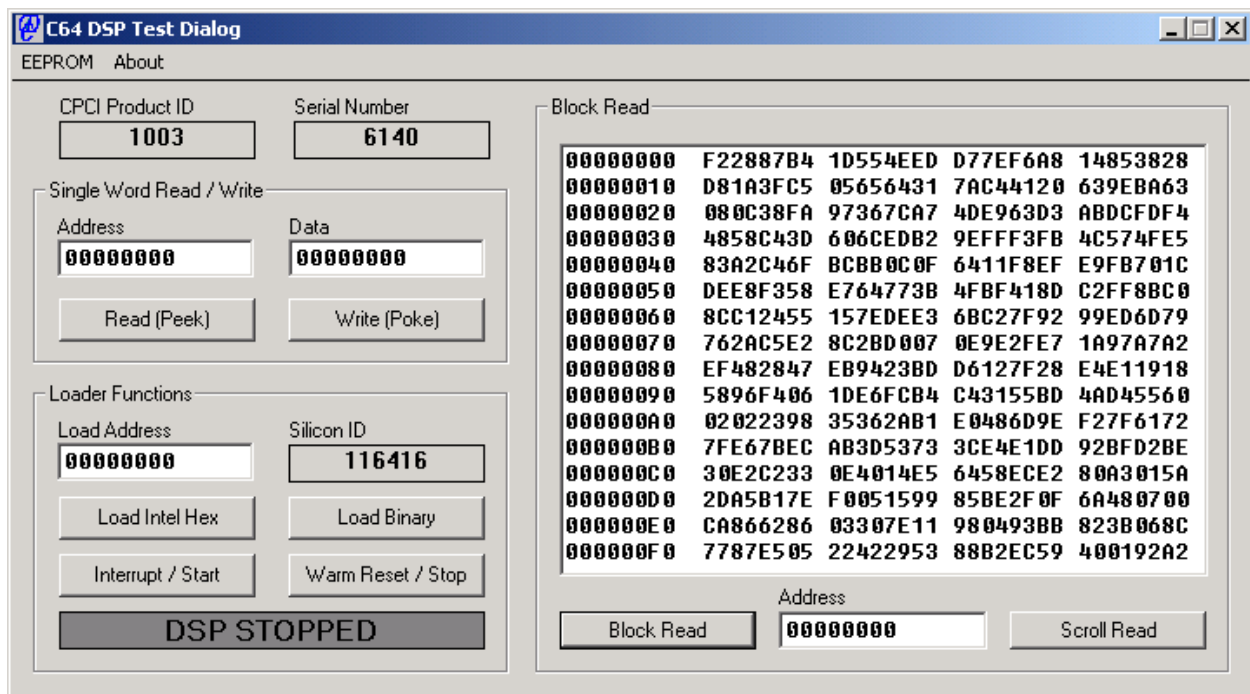


Figure 2.1 C64DSP Test Dialog

## 2.1 Memory Access Functions

One of the most important functions of any test and debug system is access to memory in order to examine and update its contents, and this is the primary function of the test dialog. The test dialog uses the CPCI interface to open the entire memory space of the DSP chip to examination and modification whether or not the DSP is currently running code. These software features take advantage of the TI design, which allows DMA access to memory from the PCI bus. Because all of the registers are memory mapped, this method allows the CPCI host to operate all of the features of the DSP chip directly without the processor ever doing anything.

The *Single Word Read / Write* dialog allows access to any single memory location in DSP memory space for reading or writing. Generally, registers in the DSP chip are 32-bit organized. All memory can be addressed as 32-bit values, even though most memory is byte addressable. As such, this dialog and all the others work with 32-bit addresses and the value entered in the address must be 4 byte aligned (divisible by 4). All values are hexadecimal and must contain all eight characters including leading zeros. An address may be entered in the *Address* field and data in the *Data* field. The *Read* button causes a value read from the DSP chip memory address specified in the *Address* field to be displayed in the *Data* field. The *Write* button causes the value in the *Data* field to be written to the DSP chip memory at the address specified in the *Address* field.

The *Block Read* feature works in a similar manner. The address entered in the *Address* field is the base address of the block to be read. The *Block Read* button causes 32-bit words to be read starting at this address and displayed in a 'memory dump' format where the first column is the start address and the following columns are sequential data words. The *Scroll Read* button causes the read to be repeated starting at the next sequential address from the previous read, each time updating the *Address* field.

## 2.2 Code Loading Functions

The *Loader Functions* dialog essentially provides a set of block write functions for DSP memory access. The *Silicon ID* field is a direct read of a register in the DSP chip which helps demonstrate that memory access is working properly without the need to manually access the chip in any way. The block write functions can be used while the DSP is running code, for example to load data, but the dialog will inquire about this action because these features are normally used to load code while the DSP is not running. The host can not directly tell if the DSP is running code, so the DSP will be reset when the dialog is first opened. The indicator at the bottom of the dialog will indicate the status of the DSP as determined only by history of operations in the dialog. When the DSP is reset it will be in an idle state until it is started by receiving an interrupt from the PCI host. It will then begin executing code at address 00000000, which is the beginning of the internal RAM space. This is therefore the first location to which code should be loaded.

Code (or data) can be loaded to the DSP memory starting at the address specified in the *Address* field from a file on disk. The file can be a raw binary format loaded 32 bits at a time to DSP memory or it can be an Intel hex file (.hex or .mcs) as generated by a programming tool such as a compiler. A hex file will be verified to have a good format, but the address field will not be used. Instead data will be loaded sequentially from data records in the hex file until the end of file is reached. Special records such as address page change or end records are ignored. Generally, the TI hex converter will be used to generate this file by specifying that an image of memory should be created. See the example project on the CD for a starting point to appropriate TI tool command file settings.

To start the DSP executing code at address 00000000 in internal RAM, click the *Interrupt / Start* button. When the DSP is running code, this button can be used to manually issue an interrupt to the DSP which may be recognized or ignored by DSP code as determined by the programmer. The *Warm Reset / Stop* button returns the DSP to the idle state with status of internal registers as defined by TI for a PCI warm reset event. For most items in the DSP chip, this is the same as any reset event. When started again, the DSP will execute code from the beginning.

## 2.3 EEPROM Features

The menu item *EEPROM* opens a dialog for programming of the EEPROM associated with the DSP chip PCI interface. This small serial memory is intended to hold a few optional parameters for the PCI interface which will be loaded when the power is first applied to the hardware if the EEPROM is present and programmed. The C64DSP has a through hole device serving this function, making it easily removed or socketed. Clicking on the *EEPROM* menu item presents the dialog in Figure 2.2 below. This dialog allows the EEPROM device to be programmed without the need for other hardware, such as a device programmer. Most users will not need to change any of the default values in the EEPROM. Others may require changes to the vendor and product ID fields.

**CAUTION!**  
The settings which appear are those currently established in the configuration EEPROM. Changes to these settings will significantly affect operation of the C64 DSP hardware. If settings are all FFFF then the EEPROM is blank or not present. Use the Set Defaults button to start. Generally, only the Vendor ID, Device ID, and Serial Number fields should be changed.

Vendor ID	Device ID	Class 7-0 / Revision	Class 23-8
101D	1003	0000	0000
Subsystem Vendor ID	Subsystem ID	Max Latency	
0000	0000	0000	
PC_D1 / PC_D0	PC_D3 / PC_D2	PD_D1 / PD_D0	PD_D3 / PD_D2
8080	8080	8080	8080
PC/PD Data Scale	PMC Capabilities	Set Defaults	
0001	0000		
Serial Number (16-Bit Decimal)	Write EEPROM		
6140			

**Figure 2.1** Dialog For Programming PCI Parameters In EEPROM

Because the settings of PCI parameters will substantially affect operation of the DSP chip PCI interface, the EEPROM should be programmed with care. The meaning of each of these settings is carefully documented in the TI C64xx PCI interface manual. One addition is made to the TI-prescribed values in the form of a serial number. The serial number is the value on the module labeling, including on the CPCI ejector handle, and the number reported by the test dialog. Generally, the serial number should not be changed. Data Design will always refer to the serial number on the original product labeling.

If changes are made to the product ID and/or the vendor ID, the driver will no longer be associated with the hardware from the point of view of the operating system. On the next reboot of the CPCI host, the operating system will indicate that new hardware has been found. To direct the Windows installation wizard to use the C64DSP driver, the associated .inf file must be modified. The changes required are fairly straightforward from examining the c64dsp.inf file in a text editor, but the specifics are potentially application dependant and are beyond the scope of this manual. Generally, no modifications will be required to the driver executable. For ease of installation using the Windows hardware installation wizard, collocate the modified c64dsp.inf file with the c64drv.sys driver executable in an installation source directory, similar to the way they are packaged on the original Data Design CPCI products CD.

## 3.0 C64DSP Software Source Code

Hardware such as the C64DSP is only as useful as the software which runs it. While the software provided with the C64DSP makes the hardware functional in a development environment, it is impossible to predict the full range of possible user needs or the software environment in which the device may be installed. Data Design has decided that the best way to meet as many user needs as possible is to provide a window into the workings of the hardware and software. To accomplish this, the source code for the software is published with the software distribution.

This open source approach provides the user with nearly infinite flexibility. It also provides the opportunity for many bad headaches. The inner workings of the software are not for the novice user. While the C64DSP basic functions are not a very demanding, applications will certainly involve complex tricks and traps. It can be a challenge for even the most skilled programmer. The good news is that the base line design is carefully structured and this section of the manual provides a roadmap to the advanced user.

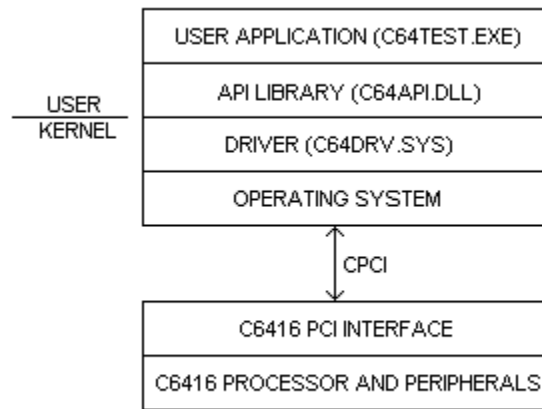
**IMPORTANT NOTICE** Information on source code and the source code itself is provided AS IS and without warranty or support. Data Design is not able to provide technical assistance to those attempting to modify the source code. It simply would not be possible to provide the hardware at reasonable cost if such support were included in the price. Please review the source code license agreement at the end of this manual before using the source code. That said however, Data Design does offer design services for hire. Information can be found on the website at the front of this manual. Also, from time to time, additional applications and source code will be posted on the website for free download.

It should be noted that this section of the manual covers the workings of a particular code base for a particular device. The primary intent of the C64DSP is to provide the user with ready access to the powerful features of the DSP processor. The functionality of the processor hardware is not covered in this manual, but is covered in fine detail and in a very understandable way by Texas Instruments. It would be counter productive to attempt to repeat this information here. The hardware schematic diagrams are provided in full so that the user can see directly how the hardware components are interfaced.

Finally, most users will see the majority of benefit from the hardware when application code is running on the DSP chip. A very simple sample application is provided on the CD so as to provide the context for building and loading an application and the compiler and linker command files that might go with it for the TI compiler tool set. It is important to note that with both a DSP application and the CPCI host accessing DSP memory space there is significant potential for conflicts which can cause some serious side affects, such as freezing operation of the host computer or causing other operation errors on one or the other subsystem. The programmer must carefully consider how access will be shared when and if necessary.

### 3.1 C64DSP Software Architecture

The hardware and software design of the C64DSP is rooted in well accepted, flexible, layered design concepts. The functional software stack is shown in Figure 3.1 below.



**Figure 3.1** C64DSP Software Architecture And Data Flow

There is no software required on the module for the system to function as it takes advantage of the features of the TI DSP chip PCI interface. There source code files for all of the major layers show are provided on the CD. However, this manual discusses only the functions of the API library. The API library is installed in the Windows system directory (\WinNT\System32 or equivalent) when the test dialog application is installed and can be retrieved from there or from the CD when building application installation packages.

The API library provides the primitive operations discussed in the operation of the test dialog. At the lowest level these will generally be all that is required for a potential application of the C64DSP module. The top level user application generally provides only the look and feel to the user. In the case of the test dialog the top level is written in Visual Basic, but it could have been written in C, Delphi, LabVIEW, or some other environment. This definition of the user interface does imply that, for most applications, there will be the need to created another library between the C64API.DLL and the user application to provide functionality which is instrument specific. This is indeed the suggested approach. In this sense, it will generally not be necessary to modify any of the existing source code at the API level and below except for in very advanced applications. The source code is primarily intended for reference and for purposes such as porting to another operating system. Contact the factory for information on support for non-Windows operating systems.

## 3.2 The C64DSP API

The C64DSP application programming interface (API) is provided as a Windows dynamic link library (DLL) which is installed in the Windows system directory (\WinNT\System32 or equivalent). When the API DLL (C64API.DLL) is loaded by the application, or by the operating system on behalf of the application, initialization code is executed which locates a C64DSP module in the system and loads a default configuration.

The API exports functions which provide an easy path to access any item in DSP memory space. Exported functions, data structures, and definitions which will be needed by an application calling the API are defined in the C64API.H header file. All exported functions are defined with a name prefix of C64\_ and will always return an unsigned 32-bit number which will be set to 1 if the function performs as expected and 0 if a difficulty is encountered. Any substantive data returned by a function is returned by reference in the arguments.

Several steps are taken to improve compatibility with a wide variety of programming environments. An export definition file (C64DSPAPI.DEF) is part of the source code so that exported names are explicitly defined for the benefit of environments which may not be aware of Microsoft name mangling standards. All arguments of exported functions are atomic types and pointers to atomic types as opposed to complex data structures which may be complicated to define in some environments. The source code is written in the C programming language with some Microsoft conventions geared towards the Windows programming model for the Intel architecture. The following are some useful relationships between data types for arguments in several software environments.

API Argument Type	Description	VB Type	LabVIEW Type
BYTE	Unsigned 8-bit number	ByVal Byte	u8
DWORD	Unsigned 32-bit number	ByVal Long	u32
long	Signed 32-bit number	ByVal Long	s32
char*	Pointer To Text String	ByVal String	s8*

There are several points to consider regarding range of arguments. Any type may also be presented in an argument as a pointer to data of that type, while the char (strictly a signed 8-bit number) always appears as a pointer in the API and refers to the location of a null terminated text string. Generally, it is an error for null pointers to be passed in arguments. Some environments require more work than others to avoid passing null pointers to strings. The BYTE data type is sometimes used as a number, but is often used as a proxy for a boolean type where 0 is false and anything else is true. The API avoids the use of 16-bit words of any kind because some environments do not support them or do not support an unsigned version. The use of the long integer type implies that the API is expecting a signed number. The DWORD data type implies that the API is expecting an unsigned number. However, in the later case the range will in most cases be less than 31 bits so that environments such as Visual Basic (VB) which do not have an unsigned 32-bit number can use a signed 32-bit number in its place. In the case of the C64DSP, which is heavily dependant on 32-bit numbers, this has a substantial effect on coding as will be obvious in the following sections of this manual.

Most advanced programming environments provide an interface to DLL's. However, all have peculiarities which must be understood for successful use. For example, in the VB environment an argument passed to the API is considered a pointer to a data type unless it is noted to be passed 'ByVal'. The VB type 'String' is a pointer, so the argument would be a pointer to a pointer unless the ByVal qualifier is applied to make it a pointer to a character array. The definitions of the API functions as required by VB can be found in the source code for the C64DSP test dialog application.

The exported function list contains several classes of functions. These include abstract functions which contain identifiable hardware and user features while another set of functions deals with items like calibration and other house keeping tasks. In addition to exported items and others enumerated in the header file, the source code contains a number of private functions and data variables. For the most part the private data are unique to a particular process under Windows. A process is an instance of the top level application, such as the test dialog. The software is designed with the intent that only one application at a time will access a given C64DSP module.

Even most advanced users will use the source code as a reference to build additional higher level applications. It would be unusual to actually modify the source code. However, the source code can be a valuable reference in understanding how each of the functions behave. There is extensive documentation in the source code itself. This manual will provide a roadmap of exported functions, an overview of their intended use, and how they fit in to the overall software design.

### 3.2.1 Calibration Functions

There are no calibration points in the C64DSP hardware. However, as a matter of consistency, items which are set by the factory are established as calibration functions. For the C64DSP, the contents of the PCI configuration EEPROM are the only such permanently configurable data points. These contents are accessed using functions declared as follows.

```
C64API C64_CAL_WriteConfig(DWORD *PCI_Config, DWORD *Serial_Number);  
C64API C64_CAL_ReadConfig(DWORD *PCI_Config, DWORD *Serial_Number);
```

Because the test dialog application is available, it is hard to imagine a situation where a programmer would need to use current or future calibration functions in an alternative environment. Probably such things should in fact be avoided. The exception is the use of the read function to retrieve the serial number, by which the programmer can identify the particular module being accessed in the case of a system with multiple C64DSP modules.

**Note: Generally, the user should not set the serial number, but in the tradition of open source nothing is held back for better or worse. However, the serial number from the point of view of the factory will always be that originally applied to the unit labeling.**

### 3.2.2 General Functions

Several functions are provided to allow the application to locate, configure, and command the hardware. These functions are presented below with a brief description of their place in the design. They may be ordered differently than in the source code for clarity of discussion.

#### **C64API C64\_OpenNext(void);**

When the library is loaded, the API will automatically open a software path to the first C64DSP instrument it finds in the system and all functionality of the API will be directed to that instrument. Calling this function will cause the API to look again and address the next C64DSP it finds. If there is only one C64DSP in the system, this function will not have much affect as the same module will still be addressed.

#### **C64API C64\_RestoreHardwareDefaults(void);**

When access to a C64DSP module is first established, it is useful to return its hardware to a known state. This will usually also be necessary after any other form of reset is applied. This function performs that task by setting registers in the DSP chip where they need to be different than the reset defaults. In the case of the C64DSP this includes configuring the EMIFA controller to support the SDRAM array.

#### **C64API C64\_WarmReset(void);**

This function performs the warm reset through the DSP chip PCI interface. The DSP will return to an idle state with the register and program counter values as specified by TI documentation.

#### **C64API C64\_InterruptDsp(void);**

This function sends an interrupt to the DSP chip through the PCI interface. If the DSP is still in an idle state, this operation will start execution of code at address 00000000, the beginning of internal RAM. If the DSP is already running code, this function will set the interrupt flag for the PCI source. The effect of this interrupt will depend on the DSP application particulars, such as if the interrupt is enabled and its service routine in place.

### 3.2.3 DSP Memory Space Access Functions

The primary function of the API is to provide host program access to the DSP memory space. This provides host access to all DSP chip features and the ability to load code and share data. The memory window is available at all times through a DMA function built into the DSP chip, but it is still very possible to cause major problems for the host or a DSP application by writing to the wrong space. The API simply provides the access and does not enforce its correct use. The functions for this purpose are described here.

**C64API C64\_Write(DWORD DspAddress, DWORD Length, DWORD \*Data);**

**C64API C64\_Read(DWORD DspAddress, DWORD Length, DWORD \*Data);**

These functions provide the core of the memory access functionality, allowing read and write of one or more 32-bit words at sequential locations starting at the given 32-bit address. All data and addresses are 32-bits based on the architecture of the DSP. The address must therefore be four byte aligned. It is also important to note that the arguments do use the full unsigned 32-bit numbers.

**C64API C64\_LoadHex(char \*DspAddress, char \*Filename);**

**C64API C64\_LoadBinary(char \*DspAddress, char \*Filename);**

These functions provide the ability to load code or data from a file as described in the discussion of the test dialog. The starting address is passed as a string for the benefit of the test dialog and similar applications which do not handle large unsigned numbers well. However, other applications will typically have known target locations making the string definition of a location such as "00000000" not as cumbersome as it first seems.

**C64API C64\_GetSiliconId(DWORD \*SiliconId);**

This function serves the purpose required by the test application, but it is also a useful example of how to access specific DSP registers using the generic read and write routines from the C programming language.

**C64API C64\_Poke(char \*DspAddress, char \*DspData);**

**C64API C64\_Peek(char \*DspAddress, char \*DspData);**

**C64API C64\_MemoryDump(char \*DspAddress, char \*DspMemoryDump);**

These functions are entirely for the benefit of the test dialog and similar interfaces. The addresses and data are passed as strings. The return data from the memory dump function is a completely constructed text screen as presented by the test dialog. These functions are generally not expected to be used by target system applications.

*C64DSP* and *C64DSP Software* are products of Data Design Corporation. No portion of this manual or any accompanying software may be reproduced by any means for any purpose other than for user archives and application of the *C64DSP* products without the express written consent of the copyright owner.

Copyright © 2006 Data Design Corporation  
All rights reserved.

#### PRODUCT WARRANTY:

**1. Product.** The "Product" referred to herein is the instrumentation product *C64DSP* and the software product *C64DSP Software*. The product as defined does not include the *C64DSP Software Source Code* or documentation of such source code.

**2. Limited Warranty.** Data Design Corporation warrants that the Product will perform substantially in accordance with the accompanying documentation for a period of one year from the date of receipt. Any implied warranties on the Product are limited to one year. SOME STATES DO NOT ALLOW LIMITATIONS ON DURATION OF AN IMPLIED WARRANTY, SO THE ABOVE LIMITATION MAY NOT APPLY TO YOU.

**3. Remedies.** The entire liability of Data Design Corporations and/or its suppliers and your exclusive remedy shall be, at the option of Data Design Corporation, either (a) return of the purchase price you paid or (b) repair or replacement of the Product that does not meet the Limited Warranty and that is returned to the place of purchase with a copy of your receipt. The Limited Warranty is void if failure of the Product has resulted from accident, abuse, or misapplication. Any replacement Product will be warranted for the remainder of the original warranty period or thirty (30) days, whichever is longer. THE FOREGOING CONSTITUTES YOUR SOLE AND EXCLUSIVE REMEDY UNDER THIS WARRANTY. EXCEPT FOR THE WARRANTIES SET FORTH ABOVE, THE PRODUCT AND DOCUMENTATION ARE PROVIDED "AS IS," AND DATA DESIGN CORPORATION AND/OR ITS SUPPLIERS DISCLAIM ANY AND ALL OTHER WARRANTIES, WHETHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

**4. Limitations of Liability.** The cumulative liability of Data Design Corporation and/or its suppliers to you or any other party for any loss or damages resulting from any claims, demands, or actions arising out of or relating to this warranty shall not exceed the purchase price you paid for the Product and documentation. In no event shall Data Design Corporation and/or its suppliers be liable for any indirect, incidental, consequential, special, or exemplary damages, loss of business profits, business interruption, or loss of business information, even if advised of the possibility of such damages. SOME STATES DO NOT ALLOW THE LIMITATION OR EXCLUSION OF LIABILITY FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THE ABOVE LIMITATION OR EXCLUSION MAY NOT APPLY TO YOU.

#### TRADEMARKS:

Commercial names and trademarks are used for identification purposes only and are recognized as the property of the respective trademark owners.

#### REVISION LEVEL:

Firmware Date: N/A  
Software Version: 1.10  
Manual Release Date: May 17, 2005

TERMS AND CONDITIONS OF LICENSE REGARDING C64DSP SOFTWARE SOURCE CODE:

PLEASE REVIEW THE FOLLOWING TERMS AND CONDITIONS CAREFULLY BEFORE DOWNLOADING OR USING ANY SOURCE CODE OR SOURCE CODE DOCUMENTATION RELATED TO THE C64DSP PRODUCT (the "SOURCE CODE"). BY USING THE SOURCE CODE, YOU INDICATE YOUR ACCEPTANCE OF THESE TERMS AND CONDITIONS, WHICH CONSTITUTE A LICENSE AGREEMENT (the "AGREEMENT") BETWEEN YOU AND DATA DESIGN CORPORATION ("DATA DESIGN"). IN THE EVENT THAT YOU DO NOT AGREE WITH ANY OF THESE TERMS AND CONDITIONS, DO NOT DOWNLOAD, COPY, OR USE THE SOURCE CODE. THE SOURCE CODE IS GOVERNED SOLELY BY THIS AGREEMENT AND NOT BY ANY OTHER AGREEMENT OR LICENSE THAT YOU MAY HAVE WITH DATA DESIGN, UNLESS PROVIDED IN WRITING. BY DOWNLOADING OR USING THE SOURCE CODE, LICENSEE ACKNOWLEDGES THAT HE/SHE HAS READ THIS AGREEMENT, UNDERSTANDS IT, AND AGREES TO BE BOUND BY ITS TERMS AND CONDITIONS

Under this Agreement, Data Design is providing source code for software related to the C64DSP product as delivered and as may be amended or augmented via various mediums from time to time. LICENSEE MAY NOT USE, COPY, MODIFY, DISTRIBUTE, SUBLICENSE OR TRANSFER THE SOURCE CODE, OR MERGED OR COMBINED PORTION THEREOF, IN WHOLE OR IN PART, EXCEPT AS EXPRESSLY PROVIDED FOR IN THIS AGREEMENT. THE SOURCE CODE MAY NOT BE USED FOR ANY PURPOSE OTHER THAN THE DEVELOPMENT OF SOFTWARE APPLICATIONS FOR THE C64DSP PRODUCT.

"Source Code" means one or more hardware or software design files in source format.

"Licensee" means YOU, the user of the Source Code, under this Agreement.

**License** Subject to the terms and conditions of this Agreement, Data Design grants to Licensee, a non-transferable, non-exclusive, perpetual license to use the Source Code. The Source Code, and the algorithms, concepts, techniques, methods and processes embodied therein, are proprietary to Data Design. Data Design retains all rights with respect to the Source Code, including any copyright, patent, and other proprietary rights, not expressly granted herein.

Licensee may implement the Source Code in a larger work, modify the Source Code, and create derivative works thereof. However, no copy or derivative work may be presented for sale in any venue without the express written consent of Data Design. All such works must be applicable only to use of the C64DSP product. Any copy or portion of the Source Code, including any derivative works thereof, including any portion merged into a design and any design or product that incorporates any portion of the Source Code, will continue to be subject to the terms and conditions of this Agreement.

Any direct copies of the Source Code made by Licensee under this Agreement shall include all intellectual property notices, including copyright and proprietary rights notices, appearing on the Source Code as provided by Data Design.

**No License To Trademarks** Data Design reserves exclusive use of any and all of its trademarks.

**Term** This Agreement is effective until terminated. Licensee may terminate this agreement at any time by destroying the Source Code together with all copies, derivative works and portions thereof in any form (including any portions merged into a design). This Agreement will also terminate immediately upon Licensee's material breach or if Licensee fails to comply with any term or condition of this Agreement. Upon any termination of this Agreement, the license and rights of Licensee under this Agreement shall terminate, and Licensee shall destroy the Source Code, including all copies, derivative works and portions thereof in any form (including any portions thereof merged into a design).

**Payment** The Source Code is being provided to Licensee at no cost, except to the extent that the Source Code may only be used in conjunction with a product which Data Design offers for sale. Data Design shall be compensated for any customization of the Source Code performed by Data Design as agreed upon in writing by the parties.

**No Warranties, Support, or Maintenance** THE SOURCE CODE IS PROVIDED TO LICENSEE "AS-IS". LICENSEE ALSO AGREES THAT DATA DESIGN DOES NOT PROVIDE MAINTENANCE OR SUPPORT FOR THE SOURCE CODE. NO WARRANTIES, REPRESENTATIONS, OR GUARANTIES, EITHER EXPRESS OR IMPLIED, ARE MADE WITH RESPECT TO THE SOURCE CODE, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE OR NON-INFRINGEMENT. LICENSEE ASSUMES THE ENTIRE RISK AS TO THE QUALITY, COMPLETENESS AND PERFORMANCE OF THE SOURCE CODE AND ANY DESIGN OR PRODUCT IN WHICH THE SOURCE CODE MAY BE USED. SHOULD THE SOURCE CODE PROVE DEFECTIVE, DATA DESIGN ASSUMES NO LIABILITY FOR ANY COST OF ANY NECESSARY SERVICING, REPAIR, OR CORRECTION. Some jurisdictions do not allow the exclusion of implied warranties, so the above exclusion may not apply to you in full, but shall be interpreted to apply to the maximum extent permissible under applicable law. DATA DESIGN DOES NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE SOURCE CODE WILL MEET ANY REQUIREMENTS, OR THAT THE OPERATION OF THE SOURCE CODE WILL BE UNINTERRUPTED OR ERROR-FREE. LICENSEE ALSO ASSUMES RESPONSIBILITY FOR THE SELECTION OF THE SOURCE CODE TO ACHIEVE INTENDED RESULTS AND FOR THE INSTALLATION, USE, AND RESULTS OBTAINED FROM THE SOURCE CODE.

**Limitations of Liability** In no event shall the aggregate liability of Data Design relating to this Agreement or the subject matter hereof under any legal theory (whether in tort, contract, or otherwise), including any liability for any loss or damages directly or indirectly suffered by Licensee relating to the Source Code, exceed One Dollar (\$1.00). IN NO EVENT SHALL DATA DESIGN BE LIABLE UNDER ANY LEGAL THEORY, WHETHER IN TORT, CONTRACT OR OTHERWISE (a) FOR ANY LOST PROFITS, LOST REVENUE OR LOST BUSINESS, (b) FOR ANY LOSS OF OR DAMAGES TO OTHER SOFTWARE OR DATA, OR (c) FOR ANY INCIDENTAL, INDIRECT, CONSEQUENTIAL, PUNITIVE OR SPECIAL DAMAGES RELATING TO THIS AGREEMENT OR THE SUBJECT MATTER HEREOF, INCLUDING BUT NOT LIMITED TO THE DELIVERY, USE, SUPPORT, OPERATION OR FAILURE OF THE SOURCE CODE, EVEN IF DATA DESIGN HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH LIABILITY AND NOTWITHSTANDING ANY FAILURE OF THE ESSENTIAL PURPOSE OF ANY LIMITED REMEDY STATED HEREIN. LICENSEE ACKNOWLEDGES THAT DATA DESIGN HAS NO RESPONSIBILITY OR DUTY TO DEFEND, INDEMNIFY OR HOLD LICENSEE HARMLESS FROM AND AGAINST ANY CLAIMS, SUITS, PROCEEDINGS, DAMAGES, LOSSES, COSTS AND EXPENSES, BASED ON PATENT OR OTHER INTELLECTUAL PROPERTY CLAIMS.

**U.S. Government Restricted Rights** If Licensee is an agency or instrumentality of the United States Government, the Source Code and related documentation are "commercial computer software" and "commercial computer software documentation", and pursuant to FAR 12.212 or DFARS 227.7202, and their successors, as applicable, use, reproduction and disclosure of the Source Code and related documentation are governed by the terms of this Agreement.

**Severability** If any provision of this agreement is held by a court of competent jurisdiction to be legally ineffective or unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable and the validity of the remaining provisions shall not be affected.