Universal Library

Data Acquisition & Control Programming Tools

Programming Tools Are Important

ComputerBoards research and polls by major magazines show that most modern data acquisition and control applications software is written by you. You have the skills to write clean, efficient code that does exactly what you want. Code you control. Code you own the source to. Code you can freely distribute without paying fees.

Sure, writing software is costly, and so is buying, learning and using someone else's software package. What 80% of data acquisition customers know is that it usually costs less overall to build your applications using modern programming environments like Visual Basic 5, or Visual C++. When you consider the number of Active X components available for the time consuming and difficult tasks of user interface and data presentation, and the ease with which the measurement portion of the program is written using Universal Library, it is no wonder that there is such an overwhelming preference for writing code.

ComputerBoards is proud to introduce release 4 of the Universal Library, a complete set of I/O libraries and drivers for all our boards for all Windows and DOS languages, and graphical programming environments like HP-VEE and LabView.

With Universal Library you can work in:

Win/NT	Win95/98	Win3.x	Win/CE	DOS	Qnx
C	C	C	C	C	C
C++	C++	C++	C++	QC	Call
VB5	VB5	VB5	VB5	QB	
Delphi	Delphi	Delphi		TurboC	

Many others! Call if you don't see your favorite here.

From Leading Edge to Legacy

You want to work with the newest platforms; Windows NT, Windows 95/98, Windows CE, QNX, LINUX and others. You also have to maintain those solid legacy applications running on DOS and Windows 3.x. We know you are concerned about leading edge applications *and* legacy code. Universal Library is constantly expanding, adding leading edge tools *and* improving legacy libraries.

Universal Library comes to you on one CD-ROM, which contains 3 distinct libraries. One 32 bit library for Windows NT 4 and soon 5.0, one 32 bit library for Windows 95/98 and one 16 bit library for Windows 3.x and DOS.

If you have a legacy MetraByte board and no Win98 or WinNT driver available from the manufacturer, we can supply you with one! How is that for long-term support?

Easy to Use, Maintain and Expand

Universal Library is easy to use. It is written from the programmers perspective. Simple data acquisition operations such as making an analog reading, or a series of them, are treated as a single operation.

Universal Library is easy to maintain. The syntax is constant from board to board, and to a great extent from language to language. It is easy to use the same code with different boards and different platforms.

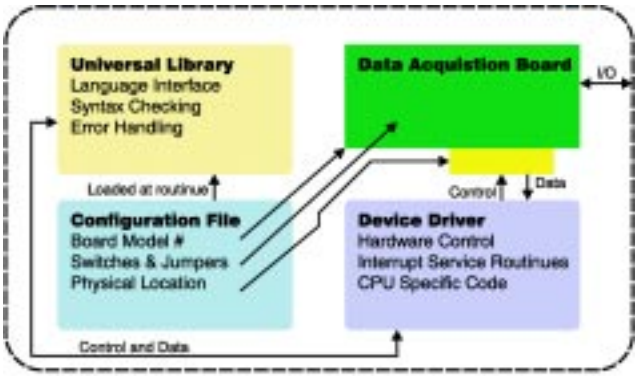
Universal Library is easy to expand. The personal computer is evolving constantly. Today the exiting news is PCI, compact PCI, Windows 98, Windows CE and many other new developments. Here is some good news. Those Universal Library lines of code you wrote for Windows 3.x will run on the newest platforms and support the newest boards without modification! Universal Library was designed to protect your investment in software.

A Logical Structure

Which is the most accepted and logical structure for software which controls I/O hardware? Open the Windows control panel and examine the way your computer is configured. Certain aspects of the hardware are "configured" infrequently. They are assumed to remain relatively constant. In fact, programs depend on it. Other aspects of hardware change as work is done. Your modem is a simple example. Much of it is "configured", then when software uses the modem, those configuration setting allow the software to be written to accept those setting greatly simplifying the code. Data flows through the modem under control of the software.

Universal Library uses the same logical structure. A configuration utility, InstaCal, maintains all the parameters that are not likely or desirable to change under program control. The configuration information is stored in a configuration file that Universal Library reads when loading. It is this logical structure that gives you the freedom to switch boards without having to rewrite code.

Here is a diagram that explains the structure.



Universal means Easy to Learn & Use

Universal means **board** to board the **syntax** for functions, such as an analog input, are the same. From CIO-DAS08 to PCI-DAS1602 the programming syntax is the same. In addition, the Universal-Library is intelligent. It knows about individual boards and their capabilities. Ask for something the board can't do, and a warning message supplies the information you need to correct the program.

Universal means **language** to language the **syntax** structure remains constant. The functions and features remain constant. The intelligent capability parser remains constant. Want to change programming languages? The UniversalLibrary requires no re-learning. Moving from DOS to Windows? The UniversalLibrary code moves with you.

What about advanced features?

I/O boards do differ in features. Resolution, maximum speed, transfer methods, channel strategy and gain coding all may change from board to board, affecting price and performance.

Lets take a look at 2 boards which use the function cbAIn Scan(). In each example we will use the maximum A/D rate for the board.

'C' Example

FEATURE	PCI-DAS1602	CIO-DAS16M1
Speed	330KHz	1MHz
XFR Method	RepInSW.	RepInSW
Gain	Programmable	Programmable

For the PCI-DAS1602/12:

```
void main() {
    int BoardNum=0, LowChan = 0, HighChan = 1;
    int Gain = BIP5VOLTS, Options=0;
    long NumPoints = 20, Rate = 110000;
    unsigned DataArray[NumPoints];
    cbAInScan (BoardNum, LowChan, HighChan, NumPoints,
                        &Rate, Gain, DataArray, Options);}
```

For the CIO-DAS16M1

```
void main() {
    int BoardNum=0, LowChan=0, HighChan=1
    int Gain = BIP5VOLTS, Options=0;
    long NumPoints = 20, Rate = 1000000;
    unsigned DataArray[NumPoints];
    cbAInScan (BoardNum, LowChan, HighChan, NumPoints,
                        &Rate, Gain, DataArray, Options);}
```

Notice that the transfer method is not specified. The UniversalLibrary is intelligent and applies board specific strategies which best match the requirements of the acquisition. You may allow the UniversalLibrary to select the transfer method or you may specify it in the Options field.

Quick Basic Example

Before looking at the Quick Basic for DOS code, lets cover a few points skipped in the 'C' example above. First, there are header files for each language which contain descriptions of boards and other system variables. The library reads a configuration file which identifies boards, switch and jumper settings and accessories attached. The configuration file is created by the *InstaCal*™.

```
'$INCLUDE: 'CB.BI'                  'Mandatory include file
'$STATIC
DIM DataBuffer%(Count&)
BoardNum = 0                        'Use board 0
LowChan% = 0                        'First channel
HighChan% = 1                       'Last channel
Count& = 50                         'Number of points to col-
lect
Rate& = 10                          'Rate = 10 samples/sec
Gain% = BIP5VOLTS                   'DAS08 not prog. gain
Options% = CONVERTDATA              'Return 12 bit values

ULStat = cbAInScan% (BoardNum, LowChan%, HighChan%, Count&,
      Rate&, Gain%, DataBuffer%(0), Options%)
```

Note that cbAInScan and variable values are alike for the QuickBasic and 'C' examples. Differences in the example are limited to non-UniversalLibrary statements. Once you learn UniversalLibrary for one language, your knowledge is easily transferred to other languages, *and*, you can communicate effectively with people who program in languages other than your favorite.

Oh, by the way, you do not have to change a line of code to change from one board to the next. Simply run *InstaCal*™ to assign a new board to the board number your program references. *InstaCal*™ modifies the configuration file which is read by the standard header file. The UniversalLibrary will apply only those features to the board which match the capabilities of the board.

FUNCTIONS

The UniversalLibrary is built upon individual functions, each of which programs, triggers, reads from or writes to a boards I/O components.

I/O board functions may be grouped into:

ANALOG I/O

cbAin()	Single analog input.
cbAinScan()	Input from ChLo to ChHi N times at R rate.
cbALoadQueue()	Load channel/gain queue.
cbAOut()	Single analog output.
cbAoutScan()	Output from ChLo to ChHi N times at R rate.
cbAPreTrig()	Set pretrigger buffer and scan values.
cbATrig()	Analog trigger setup.
cbAFileAinScan()	Analog input direct to file.
cbFilePreTrig()	Pre-triggered analog input to a file.
cbAConvertData()	Converts analog input to channel/data format.
cbAConvertPretrigData()	Unload & convert pretrigger data.
cbGetStatus()	Return status of a background operation.
cbStopBackground()	Halt a background process.

THERMOCOUPLE INPUT

cbTIn()	Inputs, Smooths, compensates & linearizes TC
cbTInScan()	Same for a range of Thermocouples.

COUNTER

cbC8254Config()	Select counter operating mode for 82C54 chips.
cb8536Config()	Select operating mode for Z8536 chips.
cb8536Init()	Set options for Z8536 chips.
cbC9513Config()	Select operating mode for 9513 chips.
cbC9513Init()	Set options for 9513 chips.
cbCFreqIn()	Measure frequency using counters.
cbCIn()	Read counter.
cbCLoad()	Load counter value.
cbCStoreOnInt()	Store counter value on interrupt.

DIGITAL I/O

cbDBitIn()	Input a single digital bit..
cbDBitOut()	Output a single digital bit.
cbDConfigPort()	Configure one port for input or output.
cbDIn()	Input a single 8 bit port.
cbDInScan()	Reads N bytes at R rate from one port.
cbDOut()	Output a single 8 bit port.
cbDOutScan()	Outputs N bytes at R rate to one port.

MEGA-FIFO MEMORY INPUT/OUTPUT FUNCTIONS

cbMemSetDTMode()	Set direction of DT-Connect transfer.
cbMemReset()	Reset M-FIFO memory to start address.
cbMemRead()	Read data from M-FIFO to data array.
cbMemWrite()	Write from data array to M-FIFO memory.
cbMemReadPretrig()	Read & organize pre-trigger data from M-F.

STREAMER FILE FUNCTIONS

cbFileAinScan()	Transfer analog input directly to streamer file.
cbFilePreTrig()	Use pretrigger strategy to streamer file.
cbFileGetInfo()	Reads acquisition parameters from streamer file.
cbFileRead()	Reads N data points into array from file.

ERROR HANDLING FUNCTIONS

cbErrHandling()	Selects from several types of error handling.
cbGetErrMsg()	Converts error codes into English messages.

Error handling has several options both for error trapping and error reporting. Reporting may be set to none, warnings only, fatal only or all. Errors may halt program execution or allow it to continue. Error messages are numerical, and may be converted into verbose English statements which provide a clear explanation of the error's cause.

An extensive feature glossary within the Universal Library checks board features against program requests, and traps requests which the boards cannot match. For example, if you are programming a CIO-DAS08 and request a transfer of 50KHz, the Library will trap that error and provide advice that the board is not capable of that rate. The feature glossary will greatly assist in preventing you from writing a program which attempts things the board is not capable of. This will save you hours of debug and possibly prevent you from puzzling over bad data!

EXTENSIVE EXAMPLES INCLUDED

A complete set of example programs is included with Universal Library. Examples for Visual Basic, C and PASCAL for both Windows and DOS languages clarify the use of each Universal Library function.

Comparing Libraries

When you chose a data acquisition board vendor you examine specifications such as speed and accuracy. Once you have the hardware that will do the job reliably, software choices need to be made. Reliability, ease of use, long term commitment are three of the important features to be considered.

Reliability

Universal Library has been in use since 1990. In that time over 250,000 data acquisition boards have been programmed to acquire mission critical data, control factories, experiments and product test stations with Universal Library software. That the Library is only now entering its fifth major revision says a lot for the rigorous testing each upgrade receives before release.

Before you select a library, ask to see what a copy of that library looked like 5 years ago. Ask which hardware available 5 years ago is no longer supported. Ask which new environments and features of current library do not support 5 year old hardware. Ask yourself where you want your application to be in 5 years. Our customers tell us that measurement systems, once constructed and installed, tend to persist for many years.

Ease Of Use

Universal Library was designed with you in mind. Whenever a software package is written, trade-offs are made between ease of use and ease of creation. Universal Library is uncompromising in design; it is always made easier for you to use, which makes it harder for us to write.

Take a look at how many steps are required to make an analog reading. With Universal Library there is one line of code. Other libraries might take many lines of code. The difference lies in the internal structure of the library. Other libraries might make you set up your own timing and transfer operations. Universal Library does all that for you. All the little details are handled by the library internally.

Before you purchase a board and software, look at the code for simplicity and ease of use. It will translate into real savings now, as you write code, and later, as you maintain it.

Long Term Commitment

Universal Library currently supports all the hardware products sold by ComputerBoards, and every platform from DOS to Windows NT. Programs written in the earliest versions will run in revision 5 without changes. We will continue to provide reliable, easy to use software support that protects your investment in measurement and control.

ORDER

Universal Library CD

WinNT/98/95/3x/DOS programming library on CD ROM

Universal Library FD

WinNT/98/95/3x/DOS programming library on 3.5 inch floppies