SAND REPORT

SAND2002-1600
Unlimited Release
Printed May 2002

PT3660 Destructive Thermal Battery Tester System Software Overview

Joseph E. Garni

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation,
a Lockheed Martin Company, for the United States Department of
Energy under Contract DE-AC04-94AL85000.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government, nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, make any warranty, express or implied, or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represent that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865)576-8401
Facsimile: (865)576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.doe.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800)553-6847
Facsimile: (703)605-6900
E-Mail: orders@ntis.fedworld.gov
Online order: http://www.ntis.gov/ordering.htm

SAND 2002-1600
Unlimited Release
Printed May 2002

PT3660 Destructive Thermal Battery Tester System Software Overview

Joseph E. Garni
Power Source Components Department
Sandia National Laboratories
P. O. Box 5800
Albuquerque, NM 87185-0614

Abstract

The PT3660 Destructive Thermal Battery Tester is used to evaluate the performance of various types of thermal batteries using test profiles to simulate the real-world conditions under which the batteries will operate. The PT3660 will replace the PT3392 tester, which has been used for production thermal battery testing since the mid-1980s. The PT3660, like its predecessor, performs both development and production lot testing. The PT3660 software runs in the MS-DOS operating environment on an IBM-compatible PC. It consists of two major parts: the operator interface software and the data acquisition and load control software. In addition, there are software modules for system configuration, calibration, hardware analysis and troubleshooting, and post-test data analysis.

This document presents an overview of and descriptive paragraphs for each routine within each software module that comprises the PT3660 system software. The software source code files are available by request from the author.

This page intentionally left blank.

TABLE OF CONTENTS

1. PT3660 SYSTEM SOFTWARE

1.1. Overview

The PT3660 Destructive Power Source Tester software runs on an IBM-compatible PC under the MS-DOS operating system (not Windows) and consists of two main executable programs for setting up and executing a destructive power source test. It also includes a program for maintaining the current load module configuration within the system as well as several small programs for testing and verifying hardware components of the system.

The two main testing programs are PT3660.EXE and ACQ_LOOP.EXE. Both reside on the system hard drive in the root directory (C:\). The PT3660.EXE program processes a Test Definition System (TDS) file and accepts user inputs necessary to execute the test. The TDS file contains information needed to control loading of the battery during the test. This information includes loading parameters (expressed in amps, ohms, or watts) and corresponding test times at which to set loads to those parameters, and minimum and maximum voltage specifications and corresponding times. User inputs to PT3660 include (among others) user name, part number, test data storage path names, and filtering parameters. Filtering parameters are used to reduce the amount of data captured by the system during a test, as they define a guardband around a data point that has been recorded in memory. Once a point has been recorded in memory, no other points will be recorded until a data point's voltage or current component differs from the last recorded voltage or current by the specified filter parameter or more. Without filtering, the amount of data recorded would be overwhelming, since data is taken every millisecond during the test.

Once all TDS and user inputs are verified for correctness, PT3660.EXE invokes the DOS system to run ACQ_LOOP.EXE. Before it does, however, PT3660 stores all necessary setup information in a file (TESTPRMS.DAT). When ACQ_LOOP runs, it reads this file. ACQ_LOOP then performs the following tasks:

Establishes global data structures for storage of setup and test data.

Initializes all system devices on the GPIB interface. This includes two power supplies, load modules, and the Yokogawa chart recorder.

Initializes the three data acquisition/load control boards, setting them up to capture data at 10kHz on each of the eight differential input channels, for a total of 80,000 readings per second.

Initializes the Yokogawa chart recorder, setting it up for capture of as many input signals as will be necessary based on the number of test channels defined by the TDS file.

Performs a calibration pulse check for each A/D board and channel accepting inputs from the battery. This is done to verify the accuracy of the input A/Ds and the load programming digital-to-analog (D/A) converters.

Executes the test by repetitively performing each of the following steps every millisecond until all battery channels fall below termination voltage:

Captures data from the A/D boards

Filters the data and records it in memory if necessary.

Checks for channels below termination voltage.

Checks for minimum and maximum voltage violations.

Updates minimum and maximum voltage levels if necessary.

Updates D/A load programming voltage levels.

Sends load programming signals from D/A to load modules.

Updates the real-time graph on the monitor.

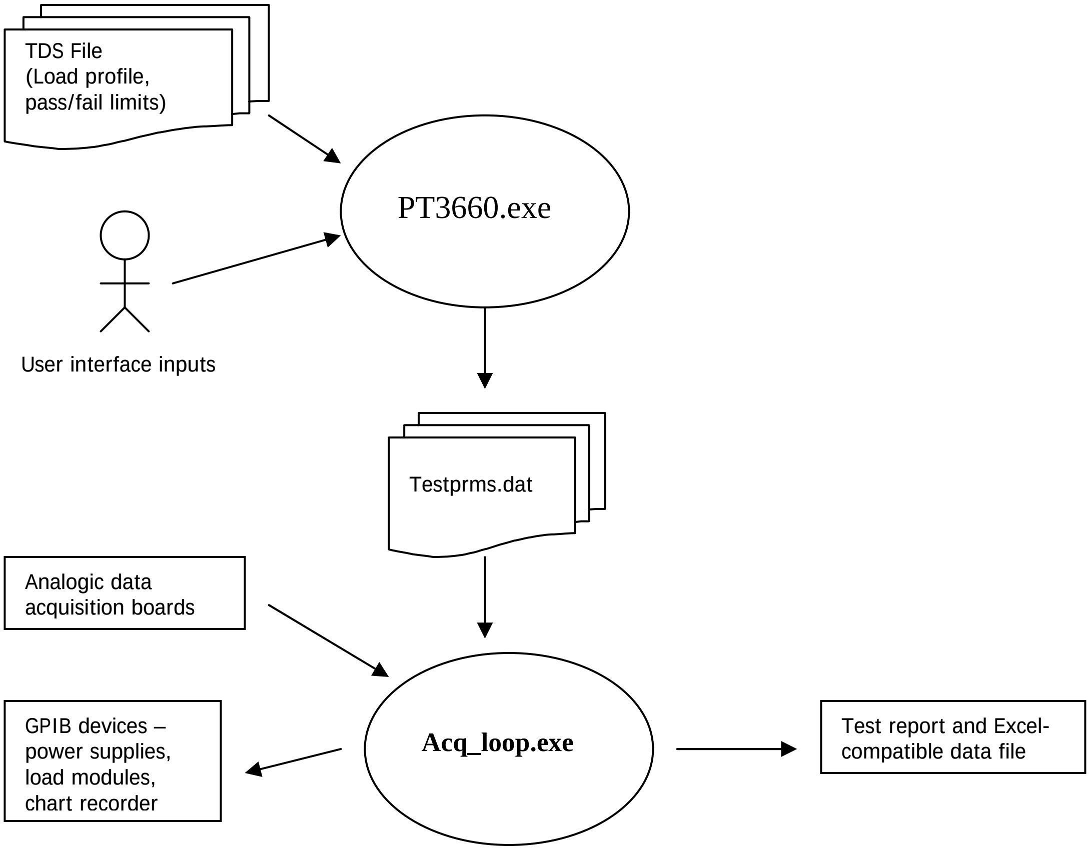

Figure 1 – PT3660 System Software Diagram

In addition to the main testing programs, there are several other programs that are used for system configuration, troubleshooting, and calibration:

SYSCFG.EXE – Allows users to specify the load modules currently resident in the Dynaload MCL488 Multichannel Load. This information is used when calculating values for setting D/A outputs for load programming during testing, troubleshooting, and calibration.

TSTSQUIB.EXE – Performs a test of the battery ignition (squib) and conductance-after-fire circuitry.

VERIFYAD.EXE – Performs continuous acquisition of voltage readings from one of the system analog-to-digital (A/D) boards and displays them on the monitor. This allows the operator to send known voltage signals (from a power supply or calibrator) into the system sensing inputs to see how accurately the A/Ds are sampling.

TESTDIO.EXE – Allows the operator to set one of the system's 48 digital I/O lines to a specified state (low or high) so that signals can be checked within the interconnect chassis. Each of the three data acquisition boards contains 16 digital I/O lines. Digital I/O lines are used to set switches for signal routing within the interconnect chassis.

DACTEST.EXE – Tests the accuracy of the Dynaload load modules and loading circuitry. Allows the operator to specify a load channel and current level at which to set the load module. DACTEST then reads the corresponding voltage across the channel's current-viewing resistor so that the operator can check the accuracy of the current-sensing circuitry and A/D board.

TSTMON.EXE – Tests the monitor circuit of the PT3660.

1.2. Why DOS and not Windows?

Good question. The answer lies in the timing requirements of the system. Data must be acquired and loads must be changed on millisecond boundaries. Furthermore, load changes can be triggered not only by the passing through of a time value, but by the passing through of a voltage level, especially during a battery's rise towards its nominal voltage after ignition. Further still, load programming voltages (sent out by the D/A converters) must be adjusted as often as possible to coincide with changes in the battery voltage, as load resistance and power values must be converted to current (amps) values in order to program the load module to the correct load level. This conversion is based on Ohm's Law (Voltage = Current * Resistance). All of this means that battery voltage values must be examined every millisecond in order to determine load profile step changes and load programming voltage values. Windows, being a multitasking system, cannot dedicate itself fully to running an application that requires this precise timing, but the DOS system, which only runs one program at a time, is able to do this. In fact, the DOS system clock cannot even be used to perform the timing; its resolution is too coarse. Timing is performed by checking the data-ready bit in the status register of the A/D boards and reading the board's data register to acquire a data reading. When 80 readings have been taken on each A/D board (a reading on each of the eight channels taken every 100 microseconds), this indicates that another millisecond has passed.

The timing of the data acquisition and load control loop is verified by checking the FIFO buffer overflow bit on the data acquisition boards. The board has a 128-sample first-in, first-out (FIFO) to hold data readings as they are captured by the board. Each time a new reading is captured and stored in the FIFO, an internal pointer (let's call it the *storage pointer*) to the next available storage word in the FIFO is incremented or set to the first word in the FIFO, depending upon where in the FIFO the latest reading was placed. At the same time, another internal pointer (called the *read pointer*) is incremented (or wrapped around to the first element in the FIFO) every time a sample is read out of the FIFO using the *_inpw* call (see the *ProcessLoop* source code listing in section 1.52 of the *PT3660 Destructive Thermal Battery Tester System Software Source Code Document*). The read pointer should always lag just a little bit behind the storage pointer. However, if the storage pointer passes the read pointer (is more than 128 samples ahead of the read pointer), this causes an *overflow* condition and means that data collection, processing, load control, and graph updates have taken longer than one millisecond to accomplish. Detection of an overflow condition causes immediate termination of the data acquisition and load control software and therefore the termination of the test.

1.3. Development System Software

The ACQ_LOOP.EXE program was written in Microsoft C/C++ Version 7.0, though only C syntax was used. The file editing and executable file builds are performed using C/C++'s Programmer's Workbench environment. This environment can be run from any release of Windows, although it can produce executable files for both DOS and Windows. See the Microsoft C/C++ manual set, specifically, the "Environment and Tools" manual for further details on the use of Programmer's Workbench.

The PT3660.EXE and SYSCFG.EXE programs were written in Microsoft BASIC Version 7.1 for DOS. The QBX development environment provided by BASIC 7.1 is used for editing source code, but is not used for compiling and linking. A batch file, UI3660.BAT, has been developed to perform this task. Both the QBX environment and the batch file can be accessed from Windows using DOS emulator windows. The BASIC.BAT batch file sets up environment variables (in SETBENV.BAT) needed by QBX, then runs the QBX environment. When modifying BASIC code, you can save a little time by opening the BASIC 7.1 window to get the QBX environment, then opening an MS-DOS window and moving to the directory where source files are located. This way, you can edit the program code in QBX, save the source code file, then move to the DOS window and run UI3660.BAT to compile and link. BASIC was chosen over C/C++ because of its User Interface Toolbox, which makes it easier to develop a Windows-like user interface with buttons, input fields, drop-down menus, and mouse click processing. C/C++ has no such library. For further details, consult the BASIC Language Reference or Programmer's Guide.

1.4. Libraries used in PT3660 Software Development

1.4.1. DAS_LIBM.LIB

This library contains routines used to control the Analogic HSDAS, MSDAS, and LSDAS family of data acquisition and process control boards. The PT3660 system uses three of these boards; each board has 8 differential analog-to-digital (A/D) channels with either 12 or 16 bits of resolution, two digital-to-analog (D/A) converters, and 16 digital I/O (DIO) lines. Board number 1 is used to read voltage signals from all battery channels, and its D/A converters send load programming signals to loads for battery channels 1 and 2. Board number 2 is used to read current (actually voltage across a current-viewing resistor) signals, and its D/As control loads on battery channels 3 and 4. Board number 3 is used to read voltage signals representing three thermocouples set to send a signal representing one millivolt per degree centigrade. Its D/As control loads on battery channels 5 and 6. DIO outputs are used on all three boards to switch on and off various signals within the interconnect chassis and intermediate load. See the Analogic DAS-LIB Programmer's Reference for further details.

1.4.2. XMSIFH.LIB

This library contains routines for setting up extended memory for storage of data during the course of the test. Disk storage is not possible while the data acquisition/load control loop is executing, as the timing of the loop would be compromised. Normally, DOS does not have access to memory past the one-megabyte boundary, but this library makes it possible to get access to as much extended memory as is available in the system. It does this by setting up pointers (handles) to large blocks of extended memory, then providing functions for writing to and reading from specified areas within those blocks. This way, a small (1 KB) buffer can be used to store recorded data; when the 1 KB block is full, it is written to the next available area within the current extended memory block. If an extended memory block becomes full, get the next available memory block handle and begin writing to the next block. The author of this library has provided a small documentation package explaining all available routines and how to call them from your program.

1.4.3. MCIB.LIB

This library was developed by National Instruments for use with the AT-GPIB/TNT family of IEEE-488 interface boards. The boards use the GPIB (General Purpose Interface Bus) protocol to communicate with instruments. In the PT3660, GPIB instruments include the two power supplies (Agilent 6032A and Xantrex XFR 35-35), the Dynaload MCL488 Multi-channel Electronic Load, and the Yokogawa ORM1300 chart recorder. The MCIB library contains routines for initializing instruments, writing commands to and receiving data from GPIB instruments. There are three manuals that explain the use and programming of the AT-GPIB/TNT boards: (1) *Getting Started With Your AT-GPIB/TNT and the NI-488.2 Software for DOS* , Part Number 320718C-01, (2) *NI-488.2 User Manual for DOS,* Part Number 320700-01, and (3) *NI-488.2 Function Reference Manual for DOS/Windows,* Part Number 320702C-01. All of these manuals are available for download from the National Instruments web site www.ni.com. They download in the form of Adobe Acrobat-compatible .pdf files.

1.4.4. GRAPHICS.LIB

This library is part of the Microsoft C/C++ Version 7.0 development package and provides both low-level and high-level routines for drawing basic shapes such as lines, circles, and rectangles, and creating complex charts and graphs. The PT3660 software uses mostly low-level routines for creating lines and rectangles and placing text on the screen at specified coordinates. Most of the graphics library functions used in the PT3660 software can be found in the *setup_plots* routine in the *plot_set.c* source file. For further details on the graphics library, consult the Microsoft C/C++ 7.0 manual set, specifically chapter 9 of the *Programming Techniques* manual (titled *Communicating with Graphics*) and the *Run-Time Library Reference* manual.

1.4.5. UI3660.LIB

This is a small library containing miscellaneous low-level routines used by the PT3660.EXE user interface program. It is made up of routines from the Microsoft BASIC 7.1 *dtfmter.lib* library plus the *uiasm.obj* and *intrpt.obj* object files.

1.5. Installing the Operating Software on a PT3660 Computer System

NOTE: It is assumed that the computer system has had Windows 98 and all drivers for the Jaz and CD-ROM drives installed on it.

1.5.1. From the Windows environment

a. Insert the *PT3660 Installation Control Disk* (AM1A0683) into the CD-ROM drive.

b. Run *Windows Explorer* (Start -> Programs -> Windows Explorer). Check to see that the icon for the CD-ROM drive (E:) appears in the left-side window of Explorer. If there is an icon for drive E:, proceed to step j.

c. Run the *Notepad* program (Start -> Programs -> Accessories -> Notepad) and open the *autoexec.bat* file in the C:\ root folder. Locate the line that references the CD-ROM driver *mscdex.exe*. The word “REM” (not necessarily all in capital letters) should be at the start of the line. Move the cursor to the beginning of the line and press the *Delete* key until the word REM disappears.

d. Select the *File* menu option, then select *Open* from the drop-down menu.

e. Click the *Yes* button when asked if the file should be saved.

f. Now type *config.sys* in the file name box and click on the *Open* button.

g. Locate the line containing a reference to *mscd001* and delete the REM at the beginning of the line.

h. Select *File,* then select *Exit*. Click the *Yes* button when asked to save changes.

i. Reboot the computer (Start -> Shut Down, select the *Restart* radio button, then click OK). When the Windows 98 Startup Menu appears, use the up-arrow key to select the *Normal* option, then press the *Enter* key. When the system boots up into Windows, run *Windows Explorer*.

j. Click on the icon for drive E: in the left-side window. Files and folders on the CD-ROM will appear in the right-side window.

k. Select the *Edit* menu option on the menu bar, then choose *Select All* from the drop-down menu.

l. Click on the selected area and drag it to the C: root folder icon in the left side Explorer window. This will initiate a copy of all files and folders from the CD-ROM disk.

m. Run the *Notepad* program (Start -> Programs -> Accessories -> Notepad) and open the *autoexec.bat* file.

n. Insert the letters REM at the very beginning of the line containing the reference to *mscdex.exe*.

o. Make sure that the following four lines appear in the file:

```
path=c:\;c:\at-gpib;d:\;d:\pt3660

c:\ni-pnp

loadhigh c:\tools_95\guest

pt3660.exe
```

If one or more of these lines is not present, type them in as the last four lines of the file.

p. Click on the *File* menu option, then choose the *Open* option from the drop-down menu. Click the *Yes* button when asked to save the file.

q. When the file selection window appears, type *config.sys* in the file name box, then click on the *Open* button.

r. Insert the letters REM at the very beginning of the line containing the reference to the CD-ROM driver *mscd001*.

s. Make sure that the following line appears in the file:

```
Device=c:\at-gpib\gpib.com
```

If this line does not appear in the file, type it in as the last line of the file.

t. Click the *File* menu option,then select the *Exit* option. Click the *Yes* button when the file save window appears.

u. Restart the system (Start -> Shut Down, select the *Restart* radio button, then click the *OK* button.

v. When the system boots up, the PT3660 software will run automatically.

1.5.2. From the DOS environment

a. Insert the *PT3660 Installation Control Disk* (AM1A0683, the latest suffix and issue) into the CD-ROM drive.

b. From then C:\> prompt, type *edit autoexec.bat*.

c. Look for a line that references the CD-ROM driver *mscdex.exe*. If the line has the letters REM (not necessarily case-sensitive) as the first three letters of the line, move the cursor to the beginning of the line and use the *Delete* key to erase the three letters.

d. Press the *Alt* key. Use the down-arrow key to move to the *Open* menu option and press *Return*. When the file selection window appears, type *config.sys* in the input box and press *Return*. Choose the *Yes* option if you are asked to save changes.

e. When the *config.sys* file appears on the screen, examine it and look for a reference to the CD-ROM driver *mscd001*. If the line has remark statement (REM) in the first three letters, remove it.

f. Press the *Alt* key. Use the down-arrow to move to the *Exit* menu option. If you're asked to save the file, choose the *Yes* option.

g. If you had to remove REM statements from either *autoexec.bat* or *config.sys*, press Ctrl-Alt-Delete to reboot the computer. When the computer boots, it will boot into the PT3660 program. Press Alt-E to exit from the program.

h. Type *xcopy e: /s* at the C:\> prompt and press the Enter key. This will install operating software and driver files for the NI 488 board and the Jaz drive.

i. Type *edit autoexec.bat*. Locate the line with the reference to the CD-ROM driver *mscdex.exe* and insert the letters REM at the very beginning of the line.

j. Now make sure that the following four lines appear in the file:

```
path=c:\;c:\at-gpib;d:\;d:\pt3660

c:\ni-pnp

loadhigh c:\tools_95\guest

pt3660.exe
```

If one or more of these lines is not present, type them in as the last four lines of the file.

k. Press the Alt key, then use the down-arrow key to move to the *Open* menu option. When the file selection window appears, type *config.sys* in the file name box, then press the *Enter* key.

l. Locate the line with the reference to the CD-ROM driver *mscd001* and insert the letters REM at the very beginning of the line.

m. Now make sure that the following line appears in the file:

```
Device=c:\at-gpib\gpib.com
```

If the line is not present, type it as the last line of the file.

n. Press the *Alt* key and use the down-arrow key to move to the *Exit* menu option. Press *Return*. When you're asked to save the files, press the *Enter* key to activate the *Yes* option.

o. Press Ctrl-Alt-Delete to reboot the computer.

1.6. Installing the Source Code on Your Computer System

NOTE: Make sure that you have installed the Microsoft C/C++ Version 7.0 and Microsoft BASIC Version 7.1 development environments before proceeding with this section. The installation programs can be found on disk 1 of each system's installation disk set.

a. Insert the PT3660 Source Code CD into the CD drive on your computer.

b. Run the Windows Explorer program and locate the drive letter of your CD drive in the left-side window.

c. Click on the CD drive letter and its description. All root folder names on the CD will appear in the right-side window.

d. Click on the *Edit* option on the menu bar, then select the *Select All* option from the drop-down menu. All files and folder names in the right-side window will be highlighted.

e. Click somewhere in the highlighted area, and while holding the left mouse button down, drag the highlighted area to the left-side window to the folder where you want to install the software. When the target folder name becomes highlighted, release the left mouse button. This will initiate the copy procedure.

f. In the left-side window, locate the name of the folder where you placed the source code and development systems. If it is not expanded so that subfolder names are visible, do so by clicking on the + symbol next to the folder name.

g. Locate the *C700* folder icon and expand its subfolder diagram using the + symbol. Locate the *BIN* folder icon and left-click on it. A listing of all files and folders in the *BIN* folder will appear in the right-side window.

h. Find the *PWB.EXE* file name and right-click on it. A drop-down menu will appear. Select the *Create Shortcut* option from this menu. This will create an item called *Shortcut to PWB* at the bottom of the right-side window.

i. Scroll down to the bottom of the right-side window (if necessary) and left-click on the *Shortcut to PWB* item. Drag it to somewhere on your Windows desktop, and release the mouse button.

j. Locate the *Shortcut to PWB* icon on your Windows desktop. If you wish to rename it, right-click on it and select the *Rename* option from the drop-down menu, then type the new name in the box underneath the desktop icon.

k. Now go back to the Windows Explorer window and click on the *BC7* entry in the left-side window.

l. In the right-side window, locate the *BASIC.BAT* entry. Right-click on it and select *Create Shortcut* from the drop-down menu.

m. Locate the *Shortcut to BASIC.BAT* entry, left-click on it, and drag it to the desktop. If you wish, change its name as you did for the PWB icon in step j.

1.7. Building New Executable Versions of the Software

1.7.1. ACQ_LOOP.EXE, DACTEST.EXE, VERIFYAD.EXE, TSTSQUIB.EXE, TESTDIO.EXE, TSTMON.EXE

Within the *Programmer's Workbench* environment, open the desired project by selecting the *Project* menu item, selecting *Open Project* from the drop-down menu, then selecting *filename.mak* (where *filename* refers to ACQ_LOOP, DACTEST, VERIFYAD, TSTSQUIB, TESTDIO, or TSTMON) from the *Open Project* window and clicking on the OK button. Open the files you wish to edit by clicking on the *File* menu item, selecting *Open* from the drop-down menu, highlighting the desired file in the *Open File* window, and clicking on the OK button. Modify all source code as needed. When all source file editing is complete and you wish to attempt to compile and link the program, click on the *Project* menu item, then select *Build: filename.exe* from the drop-down menu. All files affected by your changes will be compiled, and if there are no fatal compilation errors, all updated object files will be linked to produce a new version of the executable file. If any compilation or linking errors have occurred, they will be written to the *Build Results* window.

1.7.2. PT3660.EXE

Enter the BASIC 7.1 environment by double-clicking on the BASIC 7.1 icon on the Windows desktop. This will bring up the QBX development environment. Now you'll need to open an MS-DOS window either by double-clicking on the desktop icon (if you have one) or choosing Start->Programs ->Command Prompt from the desktop. Now make the working directory (folder) the one where the BASIC source code is. Type the drive letter of the directory followed by the colon (:) character, then press the Enter key. Now move to the right directory by using the DOS *cd* (change directory) command. You can get help with the use of *cd* by typing *help cd* at the prompt.

Return to the QBX window and open the *ui3660.bas* source code file by clicking on the *File* menu item, selecting *Open Program* from the drop-down menu, highlighting *ui3660.bas* in the *Open Program* window, and clicking on the OK button. You may have to browse through the *Dir/Drives* list in the *Open Program* window to get to the directory where *ui3660.bas* is located. Once *ui3660.bas* is open, you can select functions and subroutines for editing by clicking on the *View* menu item, selecting *SUBs* from the drop-down menu, highlighting the name of the desired routine in the *SUBs* window, then clicking on the *Edit in Active* button to display the routine's source code on the screen. At this point, you can edit the file much as you would a Word document, that is, you can use the mouse to highlight, cut, and paste text, and use the arrow keys for movement of the cursor.

When all editing is complete, select the *Save* option from the *File* menu item's drop-down menu. Move to the DOS Command Prompt window and type *ui3660* followed by the Enter key at the DOS prompt. This will run the BASIC compiler. If all code compiles successfully, the linker will run so that a new version of PT3660.exe can be produced.

1.7.3. SYSCFG.EXE

Follow the steps outlined in section 1.7.2 (PT3660.EXE), substituting *syscfg.bas* for *ui3660.bas* when editing the source file and *makecnfg* for *ui3660* when compiling and linking.

2. DESCRIPTIONS OF SOFTWARE ROUTINES

2.1. ACQ_LOOP.EXE

This section provides a brief description of every routine used in the acq_loop.exe program. The controlling routines are main() (in source file ubt_main.c), which performs most of the setup work, and ProcessLoop() (acq_ctrl.c), which controls loads, takes readings, filters and stores data, and updates real-time screen graphs every millisecond.

2.1.1. all_gpib_devices_offline (gpiboffl.c)

Use the National Instruments GPIB library routines to deinitialize all GPIB devices in the system (power supplies, loads, and the chart recorder.

2.1.2. all_loads_off (loadsoff.c)

Use the Dynaload GPIB command set to switch off all load modules in the Dynaload MCL488 chassis.

2.1.3. allocate_extended_memory (allocxms.c)

Using routines from the XMSIF library, set up pointers to blocks of memory in the extended memory area. This extended memory area is used for storage of data points during the test. Check the amount of memory still available for allocation. Round the number down to the nearest 4K multiple. If the number of bytes available is greater than half a megabyte, allocate another block of extended memory.

2.1.4. apply_and_measure_cal_pulses (calpulse.c)

This routine is performed for each data acquisition channel to be used in an upcoming test. Its purpose is to ensure that the A/D channels are reading voltages, currents, and the monitor channel properly. The Agilent 6032A power supply is set to the nominal voltage of the battery to be tested and its output routed to the specified channels of the voltage and current viewing A/D boards. The voltage is read from the A/D, then multiplied by the channel's voltage divider value. Next, the appropriate load module is programmed to the specified current value, and the current value (actually a voltage across a 0.25-ohm current-viewing resistor) is read back from the current-viewing A/D. Finally, the 10K-ohm monitor calibration resistor is switched into the circuit and the monitor channel is read on the voltage-viewing A/D. All readings are displayed on the CRT along with the expected values and pass/fail indicators.

2.1.5. build_one_data_file (bld1file.c)

For a multiple-channel battery, combines ASCII data files produced for each channel into one data file with a .CSV (comma-separated values) file type compatible with Excel. Delete all individual channel data files used to produce the .CSV file.

2.1.6. cleanup_before_exit (cleanup.c)

This routine runs right before the main program finishes. It stops the D/A converters, switches off all digital I/O (DIO) signals, switches off all load modules in the load box, de-initializes all GPIB devices (load box, power supplies, chart recorder), and deallocates memory used by each data acquisition board. Set working directories on all media to the root directories.

2.1.7. close_volt_violations (clsvtvio.c)

This routine is called at the end of a test and checks to see if a channel has ended a test outside of its minimum or maximum voltage tolerances. If that is the case, set the violation end time to the end-of-test time.

2.1.8. construct_file_name (cnstflnm.c)

Construct a data file name according to the following format:

Bnnnnnnn.00x where nnnnnnn is the battery serial number and x refers to the channel number (1-6). If only one channel is being tested, make the file extension .CSV.

2.1.9. convert_report_values (conv_rpt.c)

Take values gathered during the main data acquisition/control loop and convert them from unsigned short (as acquired by the A/D boards) to floating-point values for presentation on the test report.

2.1.10. convert_test_params (convprms.c)

Convert certain floating-point values to integer representations so that values read from the A/D board don't have to be converted from integer to floating-point for comparison during every iteration of the acquisition/control loop.

2.1.11. convert_to_unsigned (conv_uns.c)

Take a floating-point value and convert it to a 12 or 16-bit unsigned value that is the value's representation on the A/D board. This routine is used primarily to convert values used for comparison (termination voltages, voltage filters, graph axis limits) before the data acquisition/load control loop is executed to avoid having to do a lot of time-consuming conversions each loop iteration.

2.1.12. copy_file (copyfile.c)

Copy a file from one directory to another. Open the source file and read bytes 1 KB at a time and write the bytes to the destination file. This routine is necessary because there is not enough system memory available to invoke the "system" library routine to run the DOS "copy" command while the PT3660 and ACQ_LOOP programs are loaded in memory.

2.1.13. create_data_file_directories (creatdir.c)

If the user has specified directory paths for storage of test data on the Jaz and/or floppy drives, call the create_directory_path routine to create the directories if they don't already exist.

2.1.14. create_directory_path (dir_path.c)

Parse a directory path designation string (such as D:\MC3323A\LOT1) and use library routines _chdir and _mkdir to create each subdirectory if it does not exist.

2.1.15. create_test_status_file (statfile.c)

This routine is called right before the end of the program and stores the test status indicator (complete or aborted) in a file so that the PT3660 user interface program can read the file and display the status for the user.

2.1.16. dac_02_voltage_out (dac02.c)

Outputs the voltage level corresponding to the desired ignitor system current from the Keithley D/A converter board. This current is used to ignite the battery. Typically, the voltage value will be the same as the current value, i.e., 4.5 volts sent from the Keithley board will correspond to an ignitor amplitude of 4.5 amps.

2.1.17. DasBirth (analogic.c)

Call routines from the Analogic DAS_LIBM library to initialize a data acquisition board. DasBirth sets up the A/D boards to run at the desired sample rate (10 kHz per channel) and sets up parameters used before, during, and after the test to convert unsigned-integer readings from the A/D board to floating-point voltage values. DasBirth also configures the digital-to-analog (D/A) systems used for load programming and configures the board's 16 digital I/O (DIO) lines as output-only lines. DasBirth establishes a unique handle to be used to access the board in subsequent calls to DAS_LIBM routines.

2.1.18. DasDeath (analogic.c)

Deletes the board handle established by DasBirth, in effect cutting off access to the board from software.

2.1.19. DasDisable (analogic.c)

Use DAS_LIBM routines to stop a board's A/D and D/A systems at the end of a test, whether the termination was normal or abnormal.

2.1.20. DasEnable (analogic.c)

Use DAS_LIBM routines to start a board's A/D and D/A systems after initial configuration in DasBirth.

2.1.21. DasOutput (analogic.c)

Set the specified board's D/A converters 0 and 1 to the values passed in the *dac0* and *dac1* variables. Set the digital I/O lines to the value specified in the *dio* variable.

2.1.22. DasRead (analogic.c)

For the board specified in the *handle* variable, read the contents of the register specified in *code*. Read the register as many times as is specified by the *trys* variable.

2.1.23. DasWrite (analogic.c)

For the board specified in the *handle* variable, write the value specified in *data* to the register specified in *code*. Write to the register as many times as is specified by the *trys* variable.

2.1.24. deallocate_extended_memory (dealloc.c)

Free up blocks of extended memory used for data storage during the test.

2.1.25. display_battery_connect_message (cnct_msg.c)

Display a screen message instructing the operator to connect the battery to the test cable, then press a key to proceed with the test. Use graphics library routines for color and centering on the monitor.

2.1.26. display_error_msg (disp_err.c)

This routine is called in the event that a failure in some hardware or software component has been detected. This includes failures in the GPIB instruments (power supplies, loads, chart recorder), data acquisition boards, extended memory access, and file I/O.

2.1.27. final_acq_board_setup (final.c)

Perform the necessary steps to do a final setup of the Analogic data acquisition boards before the test begins. Set all D/A (load programming voltage) outputs to zero, set all digital output lines to high (default state), start the two D/As on each board (with the DASEnable routine), issue a software trigger to each A/D board, and finally start each A/D system as the last thing immediately before the data acquisition loop begins.

2.1.28. final_data_posting (finlpost.c)

At the end of the test, write whatever data points are in the temporary storage buffer (data_buffer) to a block of extended memory.

2.1.29. find_current_axis_limits (curraxis.c)

Find the upper axis limit for the real-time graph's right-hand y axis, which indicates current level. Take the maximum current value and add 10 percent. The find_y_tic_spacing_val routine will then take this value and round it upward to the closest multiple of 5, 10, 20, 50, or 100.

2.1.30. find_max_currents (max_curr.c)

For each battery channel defined for the test, search through the load profile records (from the TDS file) to determine the maximum current the channel will experience during the test. If a load profile record calls out a resistance or power value, use the channel's nominal voltage value to convert it to a current value.

2.1.31. find_time_scale_factor (scl_fctr.c)

Find the value by which to scale the x-axis tic label values for each real-time graph to be displayed during the test run.

2.1.32. find_volt_range (vltrange.c)

Based on a channel's maximum current value, set the volt range for the channel's corresponding input on the chart recorder.

2.1.33. find_voltage_axis_limits (voltaxis.c)

Find the upper axis limit for the real-time graph's left y axis, which indicates voltage level. Take the nominal voltage value and add 10 percent. The find_y_tic_spacing_val routine will then take this value and round it upward to the closest multiple of 5, 10, 20, etc.

2.1.34. find_voltage_divider_values (volt_div.c)

Find the value of the voltage divider in each battery channel's voltage sensing circuit. Set the Agilent 6032A power supply to 10 volts, then set the DIO lines to route the power supply's output to each input channel of the voltage sensing A/D board. Measure the voltage at each input, divide 10 by the voltage, and round to the nearest whole number.

2.1.35. find_x_tic_spacing_val (x_tics.c)

Determine the number of divisions for the x-axis of the real-time graphs based on the max_x_axis_val variable. If necessary, round max_x_axis_val upward to a nice, round number.

2.1.36. find_y_tic_spacing_val (y_tics2.c)

Determines the tic spacing value for one of the two y-axes displayed on the PT3660 real-time graph. If necessary, round max_y_axis_val upward to the the nearest multiple of 5, 10, 20, 50, or 100.

2.1.37. get_dac_conversion_values (dac_conv.c)

Read the tester configuration file, which contains information about each load module in the system (min and max voltage, current, and power). Set the load current to programming voltage ratio based on the maximum current value.

2.1.38. get_disk_free_space (diskfree.c)

Find the number of bytes available for storage on a disk drive.

2.1.39. get_file_size (filesize.c)

Determine the byte count of a file.

2.1.40. init_all_gpib_devices (init_all.c)

Initialize each GPIB device in the system. GPIB devices are the Yokogawa chart recorder, the Agilent 6032A power supply, the Xantrex XFR35-35 power supply, and the Dynaload MCL488 load module.

2.1.41. init_gpib_device (initgpdv.c)

Establish a "handle" for a GPIB device. This handle value will be used in subsequent GPIB library calls to the device. Clear the device.

2.1.42. init_power_supplies (init_ps.c)

Initialize each power supply in the system.

2.1.43. input_gpib_str (gpibrdst.c)

Read a string from a GPIB device.

2.1.44. main (ubt_main.c)

- Establish global data structures
- Set up GPIB devices (power supplies, loads, chart recorder)
- Initialize data acquisition boards and corresponding data structures
- Set up chart recorder and load modules
- Perform calibration pulse check for each battery channel
- Call ProcessLoop to perform test
- Deinitialize data acquisition boards and A/D boards.

2.1.45. midstring (substr.c)

Return a string of characters i1 through i2 of str.

2.1.46. output_current_to_DAC (out_dac.c)

Take a current value and determine its programming voltage value. Send the programming voltage out the appropriate Digital to Analog (D/A) converter.

2.1.47. output_gpib (out_gpib.c)

Send a character string command to a GPIB device.

2.1.48. pre_fire_setup (pre_fire.c)

Switch on digital I/O bits for test indicator, squib/sonalert enable, and conductance after fire. Program the Keithley D/A board for the ignitor amplitude value, then set the digital I/O bits to switch load module outputs to each active channel. Set the Xantrex power supply to 28 volts out, and start the chart recorder and set it to high-speed mode for the first three seconds of the test.

2.1.49. print_file (prntfile.c)

Open a file, read it line-by-line, make each line's linefeed (10) character a null, then send it to the printer. It was necessary to write this routine because there is not enough system memory available to call the "system" library routine with the DOS "print" command.

2.1.50. print_report (report.c)

Generate a post-test report. Store the report in a file, then call the *print_file* routine to send the file to the printer.

2.1.51. process_data_into_files (datafile.c)

Extract raw test data from the extended memory area and format it into comma-delimited ASCII data ready for use by spreadsheet software such as Excel. First create separate files for each channel's data, then combine all files (if necessary) into one file with the build_one_data_file routine. If specified by the user, copy the file to the Jaz and/or floppy drives.

2.1.52. ProcessLoop (acq_ctrl.c)

This is the main data acquisition/process control loop for the PT3660.

- Disable loads
- Determine maximum test length
- Determine which channels are to be tested
- Set up the real-time CRT graph
- Allocate extended memory area for data storage
- Pre-fire setup and final A/D board setup
- Main acquisition/control loop (1 millisecond duration)
    - Fire squib or CAF if necessary
    - Read data from A/Ds (10kHz per channel)
    - Average 10 readings for each channel
    - For each defined channel

```
            - If latest reading outside guardband (filter) value
                - Record latest readings for all channels in extended memory along with readings from last loop iteration
            - If voltage is below termination voltage
                - Flag it, if all other channels are below termination voltage, bail out of acq/control loop
            - Check for minimum and maximum voltage violations
            - Update minimum and maximum voltage levels if necessary
            - Check for activated life and rise time reached
            - If within half a second of a load change on channel 1, set chart recorder to high speed
            - Check for time to go to the next load profile step
            - Calculate load programming voltage based on latest voltage reading and load requirement (amps, ohms, or watts)
            - Send load programming voltage from D/A to the load module
            - Switch external loads on or off if necessary
        - Check for max CAF, record latest monitor reading
        - Update real-time graph (1 channel per loop iteration)
    - End main data acq/control loop
    - Do final posting of test data from buffer to extended memory
    - Set D/A load programming outputs to 0, switch off all DIO lines
    - Check for A/D timing glitches
    - Generate report and data files
```

2.1.53. read_chans (rd_chans.c)

Take a set of 80 readings (10 readings on all 8 channels) from the A/D board at 10 kHz per channel. Average the 10 readings for each channel. This routine is used when a set of readings is needed in a non-time-critical situation. It uses routines from the Analogic DAS board library.

2.1.54. read_chart_recorder_data (rdchtrec.c)

Command the Yokogawa chart recorder to send back readings for all of its initialized channels. The readings come back in the form of a large character string with the reading value and measurement units occupying 21 characters for each active channel. For example, if three channels are active, the string returned by the chart recorder will be 63 (3 * 21) characters in length.

2.1.55. read_load_file (readload.c)

Read definition parameters for the load module specified in load_name. These parameters include minimum and maximum voltage, current, and power. These parameters are used to calculate the ratio of the load programming voltage to the current value set by the load. For example, if a load definition calls out a maximum current of 60 amps, the programming voltage to load current ratio is 10 volts to 60 amps, as 10 volts is the maximum output of the digital-to-analog (D/A) converter on the Analogic DAS board. Since the D/A is a 12-bit converter, the resolution is 2.44 millivolts per count (10/0xFFFF), or 14.65 mA per count (2.44 * (60/10)).

2.1.56. read_test_parameters_file (testprms.c)

Read test parameters file created by the PT3660 user interface program. Parameters come from the Test Definition System (TDS) file or from user inputs.

2.1.57. read_tester_config_file (tstrcnfg.c)

Read tester configuration data from a data file. Configuration data contains information about which load modules currently exist in the Dynaload. This information is in turn used to calculate programming voltage values to program the load.

2.1.58. read_tester_info_file (tstrinfo.c)

Read the file containing information such as tester software AM number, suffix, and issue, tester ID, suffix, and issue, tester serial number, and most recent calibration date.

2.1.59. report_gpib_error (rptgperr.c)

This routine is called in the event that an error has occurred in writing to, reading from, initializing, or clearing a GPIB device.

2.1.60. reset_all_dio_lines (rsalldio.c)

Set all digital I/O lines on all three DAS boards to high, the default state.

2.1.61. round (round.c)

Round a floating-point number to the nearest whole number.

2.1.62. round_up_to_nearest (nearest.c)

Take *num_to_round* and round it up to the nearest multiple of *multiple*. This routine is used when calculating the upper limit for one of the y-axes for the real-time graph.

2.1.63. rtrim (rtrim.c)

Pass in a string and return a string that is a copy of the original string with trailing blanks removed.

2.1.64. set_all_loads_on (loads_on.c)

Set all 6 load modules of the Dynaload to external programming mode simultaneously. Send Dynaload command set commands across the GP-IB.

2.1.65. set_chart_recorder_for_long_tests (cr_long.c)

Call this routine in the event that a maximum test length is greater than 20 minutes. Normally, the chart recorder is set to print at 5 mm per second. Set it to 1 mm/sec here so as to avoid having too much paper being put out by the chart recorder.

2.1.66. set_digital_out (dig_out.c)

Set the digital I/O bits on *board_num* specified in *bit_pattern* to the state (HIGH or LOW) specified in *bit_state*.

2.1.67. set_power_supply (set_ps.c)

Send commands over the GPIB to first set the specified power supply's limiting current (IMAX), then set the output voltage and current levels. This routine can be called for both the Agilent and Xantrex power supplies.

2.1.68. set_profile_end_records (prof_end.c)

Set the last records of each channel's load, max volt, and min volt profiles to the maximum test length plus 1 second. As the test progresses, it is necessary to look at the time element of the next profile record to determine when to transition to that load or min max voltage step. Setting the last record past the end-of-test time will ensure that no unintended profile transitions are made.

2.1.69. set_working_directory_to_root (wd_root.c)

For the drive specified, use system routines to change the drive's current working directory to the root directory.

2.1.70. setup_chart_recorder (chartrec.c)

Issue GPIB commands to the Yokogawa chart recorder to initialize chart recorder inputs for each channel to be monitored. Initialization steps include setting the channel's input range, upper and lower scale limits, position on the paper, and filter value. For more information on the chart recorder commands, refer to the Yokogawa OR1400 ORP/ORM Series GPIB/RS-232-C Interface User's Manual.

2.1.71. setup_plot_parameters (plotprms.c)

Determine various parameters necessary for setting up on-screen graphs used for real-time data display during the test. These parameters include axis limits for the x axis and the two y-axes (expressed in terms of A/D board count values), and tic spacing values for all axes.

2.1.72. setup_plots (plot_set.c)

Use the Microsoft C/C++ Version 7.0 graphics library routines to draw a real-time graph area on the CRT for each battery channel to be tested. Place the voltage scale on the leftmost y-axis and the current scale on the rightmost y-axis. Test time will appear on the x-axis. Also draw lines to indicate the minimum and maximum voltage limit profile as outlined in the TDS file.

2.1.73. sleep (sleep.c)

This function takes as input a number of seconds and delays for that length of time.

2.1.74. strichr (strichr.c)

Return the array index of the first occurrence of *char_to_find* in string *st*.

2.1.75. wait_for_char (waitchar.c)

Monitor user keyboard input until the user enters a character that is in the *charset* string.

2.2. TSTSQUIB.EXE

2.2.1. cal_setup_chart_recorder (calctrec.c)

Send commands across the GPIB to the Yokogawa chart recorder to initialize it for the squib and conductance after fire tests. Set channels 13 and 14 to accept inputs of 0 to 10 volts DC with 4 kHz filtering. Specify where on the paper to print the traces from each channel and specify the chart speed for high-speed mode and the source (external) for triggering the recorder to operate in high-speed mode.

2.2.2. main (tstsquib.c)

```
- Initialize data acquisition boards
- Initialize all GPIB devices (loads, power supplies, chart recorder)
- Set the Xantrex power supply for 28-volt output
- Initialize the Yokogawa chart recorder to display data from
  squib/CAF and monitor channels.
- Loop until user wants to quit
        - Get user choice of squib or CAF for test, or quit.
        - If CAF, set conductance after fire DIO bit
        - Prompt user for time length for fire pulse
        - Prompt user for squib current level for fire pulse
        - If squib current selected and current level < 1, set
```

```
            conductance after fire DIO bit.
          - Set squib/sonalert DIO bit.
          - Use the Keithley D/A board (DAC02) send the squib current.
          - Set the chart recorder to high-speed mode.
          - Switch the squib fire DIO bit.
          - Wait the user-specified number of milliseconds while acquiring
            data from the conductance-after-fire input channel.
          - Switch off all DIO lines.
          - Display results for user.
- End Loop
- Reset all DIO lines.
- Deinitialize all GPIB devices.
- Set Keithley D/A output to zero volts.
```

2.2.3. milliseconds_wait (msecwait.c)

This routine uses the A/D sample rate as a timer to pause a specified number of milliseconds. This is because the resolution of the system clock is too coarse to depend on for this precise timing. Start the A/D system, then complete as many scans of the A/D channels that correspond to the specified number of milliseconds to wait. The number of A/D scans is calculated based on the sampling rate of the A/D board and is equal to the specified number of milliseconds multiplied by the sampling rate divided by 1000.

2.2.4. milliseconds_wait_with_acq (mswtacq.c)

This routine uses the A/D sample rate as a timer to pause a specified number of milliseconds. This is because the resolution of the system clock is too coarse to depend on for this precise timing. Start the A/D system, then complete as many scans of the A/D channels that correspond to the specified number of milliseconds to wait. The number of A/D scans is calculated based on the sampling rate of the A/D board and is equal to the specified number of milliseconds multiplied by the sampling rate divided by 1000. Calculate the average of ten consecutive readings taken on the specified A/D channel.

2.3. DACTEST.EXE

2.3.1. main (dactest.c)

Program to test the battery loading system digital-to-analog (D/A) converters and Dynaload MCL488 load modules.

```
- Read tester config file with info for converting currents to
  load programming voltage values.
- Initialize data acq boards and GPIB devices.
```

```
   - Set up loads to accept external programming voltages.
```

2.3.2. read_chans (rd_chans.c)

Use routines from the Analogic DAS board library (DAS_LIBM) to initialize one board and take a set of readings on each of its A/D channels. Convert the readings to floating-point voltage values.

2.3.3. setup_dynaload (setup_ld.c)

Send commands over the GPIB to the Dynaload MCL488 to setup one of its load channels for external programming. When in external programming mode, the load accepts voltage inputs from 0 to 10 volts at its programming terminals then converts this voltage to a load level.

2.3.4. test_dac_outputs (tstdaout.c)

Test the digital-to-analog (D/A) conversion systems on one of the three data acquisition boards. The D/A systems are used to send programming voltage signals to the load modules in the Dynaload MCL488. Each data acquisition board has two D/A converters; board 1's D/As control loads for battery channels 1 and 2, board 2's D/As control loads for channels 3 and 4, and board 3's D/As control loads for channels 5 and 6.

This routine takes as input the handle to one of the data acquisition boards as well as the board's offset and gain values. It also takes in the number of the D/A converter (0 or 1), the battery channel, and the conversion value necessary for converting a current value to a programming voltage value.

```
- Establish gain and offset values for the specified D/A.
- Set up the Dynaload to accept programming voltage values.
- Start the D/As.
- Set the initial load programming voltage to a very low value.
- Set the DIO line to connect the D/A to the load.
- Loop until user wants to quit
        - Get load current value from user.
        - Determine load programming voltage value based on current.
        - Send voltage to load using _outpw register write.
        - Loop until user wants to quit.
           - Read channel's current value (actually voltage across CVR)
           - Ask user for repeat measurement or quit.
        - End Loop
```

- End Loop
- Stop D/As and reset DIO outputs.

2.4. TESTDIO.EXE

2.4.1. main (testdio.c)

Allows users to switch on or off one of the digital I/O (DIO) lines on one of the three data acqusition boards. The user is asked to specify board number (1-3), DIO line number (1-16), and line state (LOW or HIGH).

2.5. VERIFYAD.EXE

2.5.1. analog_input_verification (inp_ver.c)

Display a screen for presenting voltage readings being taken continuously by the specified A/D board. Take and display A/D readings until the user presses any key to quit.

2.5.2. main (verifyad.c)

Initialize the A/D boards, prompt the user for the number of the A/D board to test (1-3), then call *analog_input_verification* to continuously take data from all channels of the board until the user wishes to quit. Deinitialize the A/D boards.

2.6. TSTMON.EXE

2.6.1. main (tstmon.c)

Program for testing the monitor circuit of the PT3660. Set the Xantrex power supply to put out 28 volts, switch on the monitor connect relay to route the monitor signal to the assigned A/D channel, and read the A/D channel. The voltage should be close to7 volts. Allow a 5 percent tolerance band around the expected value.

2.7. PT3660.EXE

NOTE: All routines can be found in the UI3660.BAS source file.

2.7.1. BigListBox

Present the user with a list box from which to select one item from a group of items in the *Filearray$* array. The maximum box size is ten items. If there are more than ten items in *Filearray$*, the user can use the scroll bar at the side of the list box to view the next or previous section of the list. If there are fewer than ten items in *Filearray$*, the box size will equal the number of items in *Filearray$*.

2.7.2. CalcTime

Produce a single-precision number representing a time passed in as a string in DDHHMMSSmmm form.

2.7.3. CheckDiskOK

Attempt to read a disk in the user-specified drive. First find out how many files on the disk are TDS files, that is, the have the .tds extension. If there is a problem with the disk or drive or there are no TDS files on the disk, alert the user. Otherwise, gather the filenames with no extension characters (*.) into the *Filearray$* array.

2.7.4. CheckDiskOKForWrite

Check to see if a disk is OK to receive test data. Check for disk write-protected, disk full, and general failures.

2.7.5. CheckTestSpecs

Check the validity of user-specified items from the test specification input screen.

2.7.6. ClearBackground

Clear the blue background screen. This background screen is used as a backdrop for gray informative message windows such as error message or instructions

2.7.7. ClearTBScreen1Inputs

Set data storage for user inputs from the test specification screen to default values.

2.7.8. CrossCheckError

Display an error message based on the error detected in the CrossCheckTDS routine.

2.7.9. CrossCheckTDS

For each relevant record of the TDS file's load profile, compare user-specified parameters (voltage, current, power, resistance) against maximums and minimums found in the configuration files for the load modules currently in the Dynaload MCL488.

2.7.10. DisableUIToolbox

Set the monitor back to default mode.

2.7.11. DisplayDiskFreeSpace

Use the *Interrupt* DOS system call to retrieve free space information about a specified disk drive (usually the Jaz drive). Calculate the maximum data file size for the impending test. Display free space information and maximum data file size and allow the user to either leave the disk in the drive or change it.

2.7.12. DisplayMainMenu

Display the program main menu and accept the user's selection of the function he/she wishes to perform. If a TDS file has not yet been selected by the user, make the Select TDS File option the default choice (chosen by pressing the ENTER key). If a TDS file has already been selected, make Start Test the default choice.

2.7.13. DisplayMessage

Put up or take down a generic message in or from the center of the screen.

2.7.14. EnableUIToolbox

Initialize the monitor, making it ready for use by routines from the BASIC 7.1 User Interface Toolbox library.

2.7.15. FileSelect

Invoke the BigListBox function to determine which TDS file, if any, was selected by the user. Pass the file name back in the *FileInput$* string.

2.7.16. FindDocumentReference

Peruse the 18 lines of comments in the TDS header section looking for specifications for the TDS file disk AM number and the battery's product specification. The AM number will be in the form of AMnnnnnn-sss-i, where nnnnnn is the six-character AM number, sss is the three-digit suffix, and i is the one-character issue specifier. The product specification will show up in the form PSnnnnnn-sss-i.

2.7.17. GetTestStatus

Read the test status indicator file set by the data acquisition and control loop software (acq_loop.exe) in its most recent run.

2.7.18. InitAndRunTest

Invoke the routines to accept user inputs of test parameters, then call the routine to perform the tests specified in the user's TDS file. If a TDS file has not yet been selected or an input screen has not been properly filled in, alert the user and prevent commencement of the test.

2.7.19. InsertTDSFloppy

Instruct the user to select the disk drive from which the TDS file is to be read. This is done using option buttons. Once a drive has been selected and the OK button has been pressed, the CheckDiskOK subroutine is called to search the disk for TDS files.

2.7.20. LoadTDSFile

Read the contents of the TDS file specified by the user. Place the file contents into the relevant data structures. Once the file is read, cross check it against the capabilities of the tester instruments and produce a test profile that is ready to send to the data acquisition and control loop program (acq_loop.exe) to execute the test.

2.7.21. OpenLoadFile

Open the configuration file for the load specified in the tester configuration file. Read the items into numeric values.

2.7.22. PaintBackground

Put up a blue background window as a backdrop for error windows and informative windows.

2.7.23. ReadConfigurationFiles

Read into memory the contents of the tester configuration file and the contents of the configuration files for the power supply and load specified in the tester configuration file.

2.7.24. ReadCurrentConfig

Read the current tester configuration file, which contains information about the load modules currently installed in the system. This information will be used by *acq_loop.exe* for calculating load programming voltages during the test.

2.7.25. RecordNotification

Instruct the user to enter information about the impending test in the tester's log book. A log book should be kept somewhere in the tester equipment rack and should be updated each time a test is run. Information such as test date, battery ID, lot number, serial number, and test type should be entered, and the entry should be signed by the test operator.

2.7.26. SelectTDSFile

Call the routines necessary for the user to select the TDS file he/she wants to define the load profile of the upcoming test.

2.7.27. SelectTDSFile

Call the routines necessary for the user to select the TDS file he/she wants to define the load profile of the upcoming test.

2.7.28. SetupTestParamFile

Gather all important information from the TDS file and user inputs and store it in a file for use by the data acquisition/load control program (acq_loop.exe).

2.7.29. TBTestSpecs

Accept user input of parameters from the first parameter input screen encountered during the course of setting up a test. Check the parameters for errors, then store them in the proper data structures.

2.8. SYSCFG.EXE

NOTE: All routines can be found in the SYSCFG.BAS source file

2.8.1. AlterTesterConfig

Change the configuration file containing setup information about the modules currently installed in the Dynaload MCL488 Multichannel Load. For each load channel (1-6), the user has the option of selecting the name of a load module from the list of available modules for which a configuration file exists. Information for load module configuration files is input in the GetLoadInputs routine.

2.8.2. AlterTesterInfo

Present an input window that allows the user to change certain information about the tester. This information includes tester ID, suffix, and issue, tester serial number, tester software AM number, suffix, and issue, and the most recent calibration date.

2.8.3. BigListBox

Present the user with a list box from which to select one item from a group of items in the Filearray$ array. The maximum box size is ten items. If there are more than ten items in Filearray$, the user can use the scroll bar at the side of the list box to view the next or previous section of the list. If there are fewer than ten items in Filearray$, the box size will equal the number of items in Filearray$.

2.8.4. CheckDiskOK

Search a disk for occurrences of specified file types. Store all matching file names in the Filearray$ string array. Trap any file access errors and report them.

2.8.5. CheckLoadInputs

Check for errors in the user's input of load module configuration information from the GetLoadInputs routine.

2.8.6. FileSelect

This routine is called after a list of files of a certain type has been gathered into the FileArray array. A call is made to the BigListBox routine to display them on the screen so that the user can select one of them.

2.8.7. GetLoadInputs

Get user input of information about a load module to go into the Dynaload MCL488 Multichannel Load. This information includes minimum and maximum voltage, current, resistance, and power and is used in calculating voltage values from a digital-to-analog (D/A) converter to the load when programming it during a test.

2.8.8. GetUserSelection

Present the main menu screen and get the user's choice of changing the load module configuration, defining a new load module, altering an existing load module definition, or altering tester information.

2.8.9. IdleTime

Pause for the number of seconds specified by the *WaitTime* argument.

2.8.10. InitLoadRecord

Set members of the load configuration record to default values.

2.8.11. ModLoad

Process screen inputs that allow the user to modify information in an existing load module definition and configuration file.

2.8.12. OpenLoadFile

Read a load module definition/configuration file.

2.8.13. OpenTesterInfo

Read the tester information file.

2.8.14. ReadCurrentConfig

Read the tester configuration file containing the load module configuration.

2.8.15. ReadLoadTypes

Build a list of all available load module definition/configuration files so that the user can choose one when building the tester configuration.

2.8.16. StoreCurrentConfig

Write the current tester configuration record to a file.

2.8.17. StoreTesterInfo

Write the tester information record to a file.

2.8.18. WriteRecToDisk

Write a record of a specified type to a file.

2.9. QACHKSUM.OBJ

2.9.1. qachksum

Perform a quality assurance check on a TDS file that was quality-assurance (QA) stamped at the time it was built with the TDS program. If a user has used an editor of some sort to alter any information in the header, load profile, or min/max voltage profile sections of a QA-stamped file, this routine will detect it and not allow the user to use the TDS file to control the test.

In a QA-stamped file, the last nine lines are generated by performing exclusive-or operations on each line of the file. The first of these nine lines is generated by performing an exclusive-or on all header records of the file. Each of the next eight lines is generated by performing exclusive-or operations on all load profile and min/max records for each of the eight potential battery channels.

Each time an 80-character line is read from the TDS file, its equivalent 40-word unsigned integer buffer is used in an exclusive-or operation with a previously-generated 40-word buffer to produce a third 40-word buffer. Each Nth word of the first buffer is XORed with the N+1th word of the second buffer for the first 39 words of buffer 1 and the result placed in the N+1th word of the third buffer. The first word of the third buffer is generated by XORing the first word of buffer 1 with the last word of buffer 2.

2.9.2. do_exclusive_or

Perform an exclusive-or operation on two 40-word buffers to produce a third 40-word buffer. See the synopsis of the *qachksum* routine for further explanation.

2.9.3. compare_checksums

Compare each Nth element of two 40-word buffers and return the CHECKSUM_MISMATCH value if two elements don't match.

2.9.4. midstring

Return a string of characters i1 through i2 of *str*.

DISTRIBUTION:

MS-0451/06515	Brian Murphy-Dye
MS-0512/02500	James K. Rice
MS-0521/02501	J. Thomas Cutchen
MS-0613/02521	Daniel H. Doughty
MS-0613/02525	Rudolph Jungst
MS-0614/02523	Robert Bickes, Jr.
MS-0614/02522	Paul Butler
MS-0614/02523	Roy Morgan
MS-0614/02523	James Norton
MS-0614/02523	David Sawayda
MS-9018/08945-1	Central Technical Files
MS-0612/09612	Review & Approval For DOE/OSTI (1)
MS-0899/09616	Technical Library (2)