

MB4/5 Tools

Application Programming Tools for MotionBASIC® 4.x - 5.x

MB4/5-TLSb

Copyright (c) 1999

Ormec Systems Corp.

All rights reserved.

19 Linden Park

Rochester, NY 14625

(716) 385-3520

December 5, 1999

Copyright Notice

Copyright 1997, 1998, 1999 by Ormec Systems Corporation. All rights reserved. This manual and any software that it may describe, remain the exclusive property of Ormec Systems Corporation. No part of either may be reproduced in any form without the prior written permission of ORMEC.

Warranty

ORMEC extends no warranty with respect to the merchantability or fitness of this product for any particular purpose. It is the customer's responsibility to determine whether it is suitable for the specific application and whether it meets performance, reliability, and safety requirements when used in that application. ORMEC reserves the right to make improvements to the product as well as this documentation at any time without notice.

Terms and Conditions of Sale

All software sold or otherwise provided by ORMEC is made available subject to ORMEC's published Standard Terms And Conditions Of Sale.

Table of Contents

1 Welcome	page 1
2 What are MB4 Tools?	page 3
2.1 MB4 Tools Alarm Display Overview	page 4
2.1.1 Alarm Number, Date & Time	page 5
2.1.2 Alarm Description	page 5
2.1.3 Alarm Display Keys/Buttons	page 5
2.2 QuickPanel Time and Date Configuration Panel	page 6
2.3 Alarm Display Procedures	page 7
3 Test Driving MB4 Tools	page 9
3.1 Getting Ready	page 9
3.2 Operating the Demo	page 10
3.2.1 Main Panel	page 10
3.2.2 Jog Mode Panel	page 11
3.2.3 Auto Mode Panel	page 12
4 MB4 Tools File Organization	page 13
4.1 MB4 Tools QuickPanel Project Files	page 13
4.2 MB4 Tools MotionDesk Project Files	page 14
4.2.1 Error Handler File	page 14
4.2.2 Application Specific Subroutines File	page 14
5 Writing Your Program	page 15
5.1 Overview	page 15
5.2 Planning Your Program Structure	page 15
5.3 Avoiding Conflicts	page 17
5.3.1 MotionBASIC Variable Name Prefixes and Suffixes	page 17
5.4 Setting Up Your Development Environment	page 17
5.4.1 Directory and Files	page 18
5.4.2 Opening the MB4 Tools MotionDesk Project	page 18
5.4.3 Loading the MAP and QuickPanel MBXs (QuickPanel Users Only)	page 18
5.4.4 Configuring the Controller and Axes	page 19
5.5 Configuring Your Application Program	page 19
5.5.1 If you are using a QuickPanel...	page 20
5.5.2 If you are using an MMI-840...	page 21
5.5.3 If are not using an MMI-840 or QuickPanel Operator Interface...	page 22
5.6 Viewing the Alarm Display	page 22
5.6.1 Preventing Operator Exit from the Console Alarm Display	page 23
5.7 Viewing the Time & Date Display	page 23
5.8 MB4 Tools Process ID	page 23
5.9 Using the MotionBASIC DATE\$, TIME\$ and TIMER Variables	page 24
5.10 Next Step	page 24

6 Configuring MB4 Tools QuickPanel Display page 25

 6.1 Importing a QuickPanel Project page 25

 6.2 Adding Panels to Your Application page 25

 6.2.1 Panel ID Numbers page 26

 6.2.2 Adding MAP Statements page 26

 6.2.2.1 MAP Reference Number Guidelines page 27

 6.2.3 QuickPanel Color Conventions page 27

 6.2.4 Displayed Panel Control page 29

7 Application Specific Subroutines page 31

 7.1 Application Error Handler (ApplFault Subroutine) page 31

 7.2 Application Specific Error Messages (ApplFltTxt Subroutine) page 32

 7.3 Application Specific Axis and/or Machine Names (ApplNames Subroutine) page 32

8 Useful MB4 Tools Variables & Subroutines page 35

 8.1 Error Handle Debug Flag page 35

 8.2 MAP Statement Disable Flag page 35

 8.3 Error History Variables page 36

 8.4 QuickPanel Related Variables page 37

 8.5 Adding Message Text to the Error History Log page 37

 8.6 Viewing the Error History Log in the MotionDesk Console Window page 38

 8.7 Monitoring QuickPanel Communications Status (QPWatchdog) page 38

 8.7.1 QPWatchdog Initialization page 38

 8.7.2 Using QPOk page 39

Appendix A1 - List of Variables and Labels A-1

Appendix A2 - QPTools Flowcharts A-3

 Error Handler Flow Chart A-4

 Initialization Flow Chart A-5

 Web Tools Initialization Flow Chart A-5

Appendix A3 - Web Tools Description A-7

Appendix A4 - Web Tools Setup A-15

 Optional User Configured Pages A-18

 Optional Custom Pages A-21

Appendix A5 - Tools Initialization Summary A-25

Appendix A6 - Web Tools Communication A-27

Chapter 1

Welcome

1 Welcome

This manual tells you about ORMEC's MB4 Tools for ORION™ controller application programming. The MB4 Tools include many useful subroutines which can be incorporated into an application program, or run separately via the MotionDesk Direct Mode window. The tools also include an alarm display and history log which are written to support the use of a QuickPanel¹ flatpanel touchscreen or MMI-840 operator interface unit, however, neither a QuickPanel or MMI-840 is required to use the MB4 Tools. This manual tells you what the MB4 Tools are and how to develop an application program that uses them.

These programming tools are designed for use with MotionBASIC version 4.0.0 (and higher). If your application uses a QuickPanel, the QuickPanel Communications MotionBASIC Extension (MBX-QP-4) version 2.1.0 (and higher) and MAP MotionBASIC Extension (MBX-MAP-4) version 3.0.0 (and higher) are also required.

This manual is divided into the following chapters:

- | | |
|-----------|---|
| Chapter 1 | Welcome tells you how this manual is organized. |
| Chapter 2 | What Are MB4 Tools? tells you what ORMEC's MB4 Tools are and what they do. |
| Chapter 3 | Test Driving MB4 Tools Alarm Display using a QuickPanel takes you on a test drive of a sample application program using a QuickPanel and the MB4 Tools Alarm Display. The demo helps you to become familiar with the look and feel of the MB4 Tools Alarm Display. |
| Chapter 4 | MB4 Tools File Organization provides descriptions of the various program files that make up MB4 Tools, how they are organized, and how they relate to each other. |

¹ QuickPanel and QuickDesigner are trademarks of Total Control, Inc.

- Chapter 5 **Writing Your Program** tells you what you have to do when writing an application program based on MB4 Tools.
- Chapter 6 **Configuring MB4 Tools QuickPanel Display** explains how to configure the MB4 Tools QuickPanel Display modules to fit your application.
- Chapter 7 **Application Specific Subroutines** tells you about the subroutines contained in the APPLSUBS.BAS file and how you can use them to change the way the MB4 Tools Alarm Display works.
- Chapter 8 **Useful MB4 Tools Variables** tells you about some useful MB4 Tools variables you can use in your program modules.
- Appendix A1 **List of Variables and Labels** includes a list of all variables and labels used by MB4 Tools modules.
- Appendix A2 **Alarm Display and Initialization Flow Charts** includes flow charts describing the MB4 Tools Alarm Display operation and initialization procedures
- Appendix A3 **Webtools Description** explains the latest features provided by the MB Tools package. The tools have been enhanced to provide Orions with the ability to communicate status information via html to a connected web browser.
- Appendix A4 **Webtools Setup** explains the steps necessary to add the webtool feature to an Orion.
- Appendix A5 **Tools Initialization Summary** provides a quick reference of the code required to implement the various optional features of the MB Tools package.
- Appendix A6 **Web Tools Communications** provides information regarding remote communications to an Orion running the web tool server.

Chapter 2

What are MB4 Tools?

2 What are MB4 Tools?

The MB4 Tools are a collection of MotionBASIC program modules and QuickPanel projects for use developing MotionBASIC application programs. These program modules and QuickPanel projects provide the framework for any application that uses an ORION with a QuickPanel flatpanel touchscreen, or for an ORION with an MMI-840 operator interface, or an ORION without a QuickPanel or MMI-840 operator interface.

NOTE:

The MB4 Tools can also be used for applications with an ORION motion controller and a PC running third party operator interface software, such as WonderWare's In Touch, capable of Modbus communications. Before using the MB4 Tools with a third party operator interface software package, verify that the operator interface software string data transfer method in Modbus is compatible with that of the QuickPanel Communications MotionBASIC Extension (MBX-QP-4). Refer to the QuickPanel Communications MBX Help for further information.

MB4 Tools provide a detailed error message display. If a QuickPanel is used, a method for configuring the ORION time & date is also provided.

The MB4 Tools Alarm Display provides the following functionality:

- A precise description of the error, fault or alarm condition
- Customizable axis and machine names
- Customizable user error messages
- Automatic inclusion of the appropriate MBX error codes and messages
- Non-volatile alarm information storage in a configurable length historic program error log, with optional PC Card storage

2.1 MB4 Tools Alarm Display Overview

The MotionDesk Console, QuickPanel, and MMI-840 Error Displays all provide the same information. Refer to Figures 1 through 3 for a representation of each Alarm Display type.

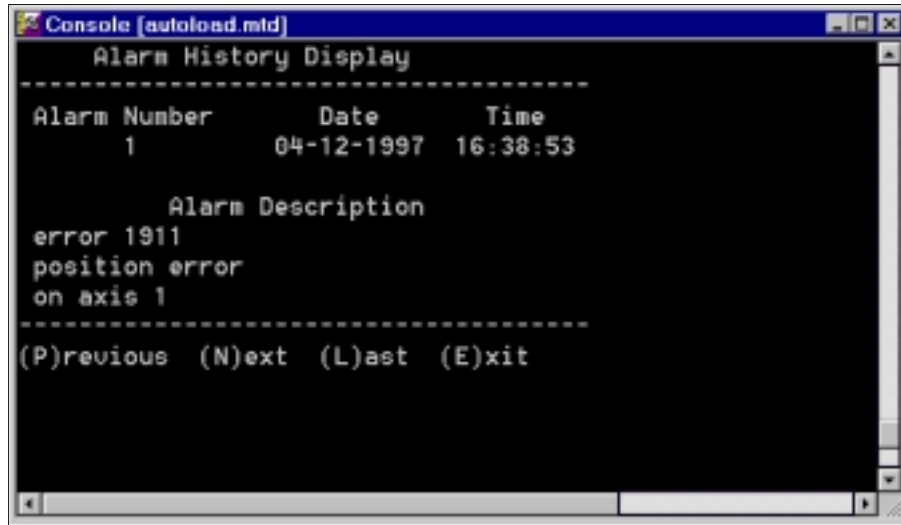


Figure 1, MotionDesk Console Alarm Display

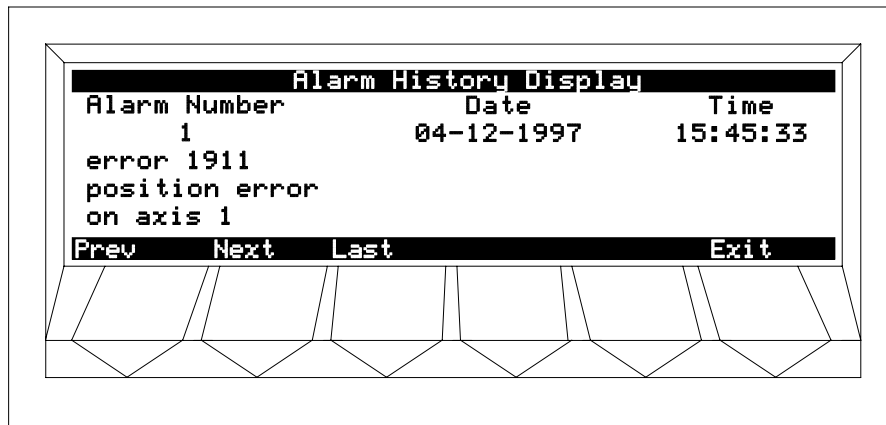


Figure 2, MMI-840 Alarm Display

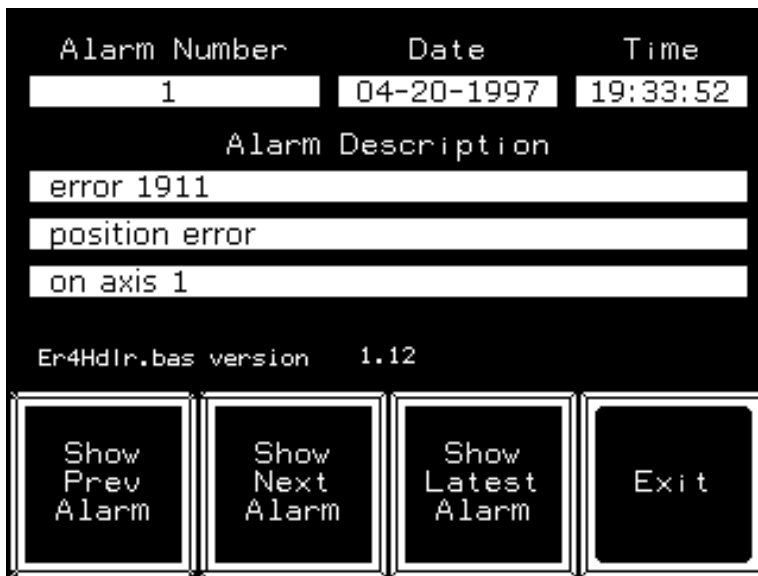


Figure 3, QuickPanel Alarm Display

2.1.1 Alarm Number, Date & Time

Alarm information is stored and displayed according to the Alarm Number. The Alarm Numbers are assigned sequentially, with the most recent alarm having the highest Alarm Number. When an alarm occurs, the date and time, based on the ORION real time clock, is stored with the other alarm information.

2.1.2 Alarm Description

The Alarm Description contains detailed information regarding the alarm. This information can be customized to suit the application (axis names, machine names, user errors, etc.), and non-alarm information can also be displayed.

2.1.3 Alarm Display Keys/Buttons

Each of the Alarm Displays uses "Prev", "Next", and "Last" keys or buttons to scroll through the Alarm History. Refer to Table 1 for an overview of the various keys or buttons used for each type of display.

Alarm Display	Prev	Next	Last	Exit
MotionDesk Console ¹	P	N	L	E
QuickPanel	Show Prev Alarm	Show Next Alarm	Show Latest Alarm	Exit
MMI-840	F1	F2	F3	F6

¹ The MotionDesk Console window must be active to accept keyboard entry.

Table 1, Alarm Display Keys/Buttons

- "Last" - displays the most recent alarm message (not displayed when the latest alarm is displayed)
- "Prev" - displays the next lower number alarm message (not displayed when the oldest alarm in the log is displayed)
- "Next" - displays the next higher number alarm message (not displayed when the latest alarm is displayed)
- "Exit" - allows the program to leave the Alarm Display

2.2 QuickPanel Time and Date Configuration Panel

The Time and Date Configuration panel (Figure 4) allows the operator to set the ORION's real time clock, which is used to set the time and date used for the Error History Log time stamping.



Figure 4, Time & Date Configuration Panel

The Current Date and Current Time displays indicate the current settings for the ORION real time clock. The values in these displays will reflect changes made to the month, day, year, hour, minute, and/or second using the numeric data entry objects on this screen.

The Month, Day, Year, Hour, Minute, and Second numeric data entry objects on this screen are configured for the values of their respective parameters at the time when the panel was first displayed, or their state after a change to the corresponding parameter was made.

2.3 Alarm Display Procedures

The Alarm Display module provides a brief error handler which records the necessary alarm information and branches to your MotionBASIC application program's error handler. This allows you the flexibility to develop an error

handler which meets your application's needs, while still providing an easy to use mechanism for storing and displaying alarm information. The Alarm Display module also provides several useful routines for working with the alarm history and log file.

Chapter 3

Test Driving MB4 Tools Using a QuickPanel

3 Test Driving MB4 Tools

This chapter takes you on a test drive of a sample program based on MB4 Tools to show you how they "look and feel".

To take the test drive you must have the following:

- ORION™ controller with two servo axes (axes 1 and 2).
- MotionKey with at least 800 MotionCredits
- QuickPanel (MMI-QP or MMI-QP2) with cables.
- Release 4.0a (or later) of MotionBASIC®.
- Release 2.0a (or later) of MotionDesk™ (includes the MB4 Tools files)
- Release 2.0a (or later) of the QuickPanel Communication MBX (MBX-QP-4)
- Release 3.0a (or later) of the MAP MBX (MBX-MAP-4).
- A working knowledge of how to use the MotionDesk™ Project Navigator.

3.1 Getting Ready

As part of the MotionDesk 2.0 installation, the MB4 Tools files were installed on your development computer.

Using MotionDesk, open the appropriate demo MotionDesk Project for the QuickPanel unit you are using (File | Open Project). Refer to Table 2 for an overview of the various demo projects.

Cabinet Mounted Units		
QuickPanel Model	QuickPanel 2 Model	MotionDesk Project
MMI-QP/5M	MMI-QP2/5MU	ER4QP5M.MTD
MMI-QP/5C	MMI-QP2/5CU	ER4QP5C.MTD
	MMI-QP2/6MU	ER4QP6M.MTD
	MMI-QP2/6CU	ER4QP6C.MTD
MMI-QP/9E	MMI-QP2/9E_	ER4QP9E.MTD
	MMI-QP2/10M_	ER4QP10M.MTD
MMI-QP/9C	MMI-QP2/10C_	ER4QP9C.MTD
MMI-QP/9T	MMI-QP2/10T_	ER4QP9C.MTD
	MMI-QP2/12T ¹	ER4QP12T.MTD
Hand Held Units		
	MMI-QP2H/6MU ¹	ER4QPH6M.MTD
	MMI-QP2H/6CU ¹	ER4QPH6C.MTD

¹ 12.1" TFT and 6" Hand Held QuickPanels require MBX-QP version 1.2a (or later) for MB 3.x or MBX-QP-4 version 2.2.0 (or later) for MB 4.x.

Table 2, MB4 Tools Demo MotionDesk Projects

Power up the ORION and verify that the servos are properly tuned using the MotionDesk Axis Tune utility. You should leave all of the axis parameters at their default values, except those needed to tune the servos. Refer to the MotionDesk Help for further information regarding using the Axis Tune utility.

When you have the servos properly tuned, exit Axis Tune and run (Debug | Run) the demo program. The demo program will automatically download the QuickPanel.

3.2 Operating the Demo

The demo is a simple indexing application that allows you to command axis motion and generate various errors and faults to demonstrate the MB4 Tools alarm display. In addition to the alarm display panel and the Main Menu panel, are two panels (Jog Mode and Auto Mode) which allow you to command motor motion for two axes, and simulate an actual application.

3.2.1 Main Panel

The Main panel (Figure 5) provides access to all the other Error Demo panels, allows the user enable and disable the axes, and simulate an error by entering a MotionBASIC error code.

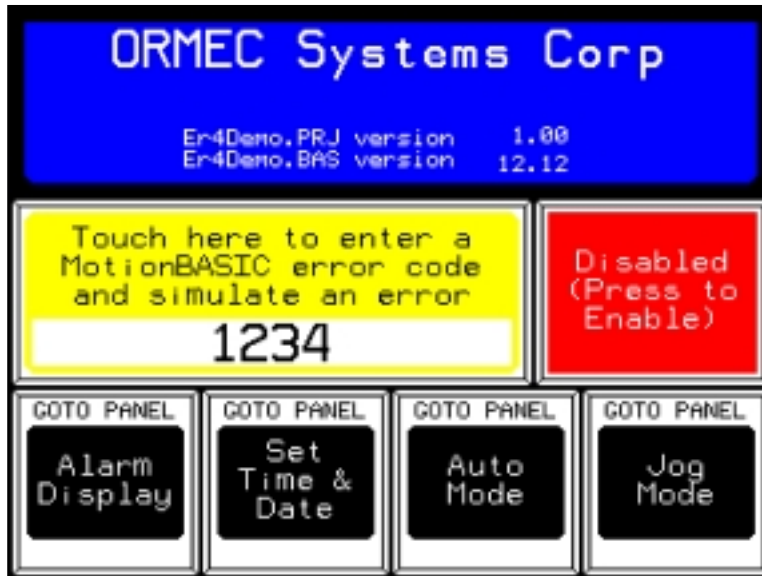


Figure 5, Main Panel

3.2.2 Jog Mode Panel

The Jog Mode panel (Figure 6) provides "Axis FWD" and "Axis REV" push-buttons for jogging two axes, and position displays for each. Both axes have software end of travel limits defined (675 degrees). The axes will jog until the software end of travel position is reached.

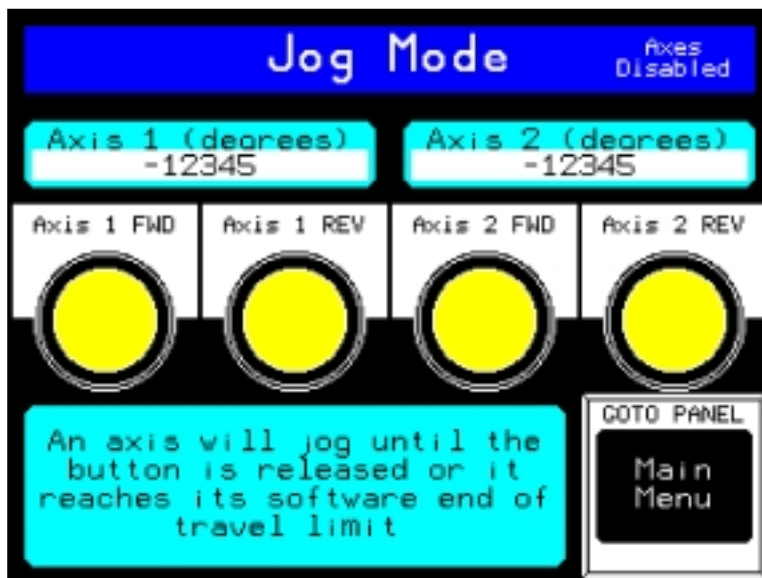


Figure 6, Jog Mode Panel

3.2.3 Auto Mode Panel

The Auto Mode panel (Figure 7) provides "Start" and "Stop" push-buttons for a simple indexing application. Pressing the "Cause EOT Fault" allows the user to generate a software end of travel limit error for Axis 1. Pressing the "Cause User Fault" generates an error 1900 User Defined Fault.

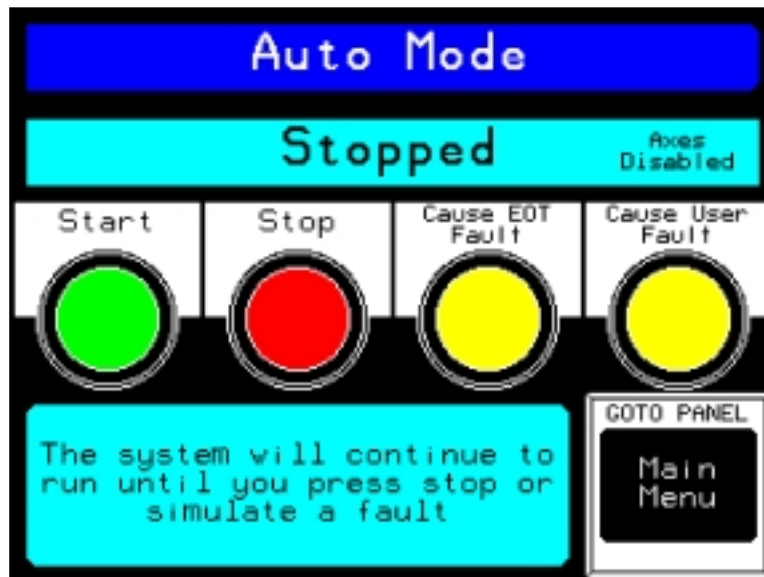


Figure 7, Auto Mode panel

Chapter 4

MB4 Tools File Organization

4 MB4 Tools File Organization

This chapter explains the various program files that make up MB4 Tools, how they are organized, and how they relate to each other. Figure 8 provides an overview of the MB4 Tools directory structure.

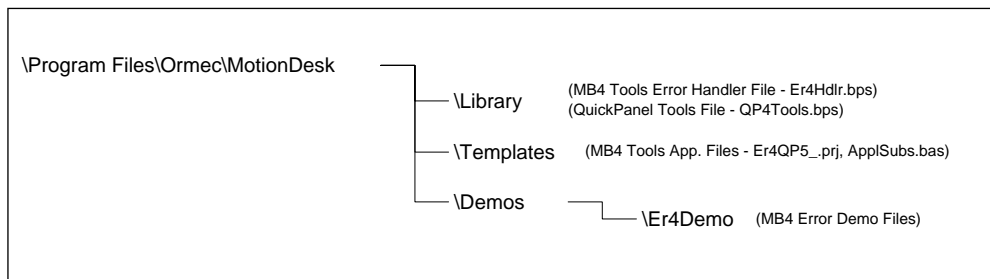


Figure 8. *MB4 Tools Directory Structure*

4.1 MB4 Tools QuickPanel Project Files

The MB4 Tools are supplied with a pair of QuickPanel project files (ER4HQP__.PRJ), one each for the monochrome and color displays. These files, located in the \Program Files\Ormec\MotionDesk\mb4tools directory, contain the MB4 Tools panels and Tag register definitions. Table 3 contains a list of the QuickPanel models and their corresponding MB4 Tools QuickDesigner Project files, which can be imported into QuickDesigner and modified for your application.

Cabinet Mounted Units		
QuickPanel Model	QuickPanel 2 Model	QuickDesigner Project
MMI-QP/5M	MMI-QP2/5MU	ER4QP5M.PRJ
MMI-QP/5C	MMI-QP2/5CU	ER4QP5C.PRJ
	MMI-QP2/6MU	ER4QP6M.PRJ
	MMI-QP2/6CU	ER4QP6C.PRJ
MMI-QP/9E	MMI-QP2/9E_	ER4QP9E.PRJ
	MMI-QP2/10M_	ER4QP10M.PRJ
MMI-QP/9C	MMI-QP2/10C_	ER4QP9C.PRJ
MMI-QP/9T	MMI-QP2/10T_	ER4QP9C.PRJ
	MMI-QP2/12T	ER4QP12T.PRJ
Hand Held Units		
	MMI-QP2H/6MU	ER4QPH6M.PRJ
	MMI-QP2H/6CU	ER4QPH6C.PRJ

Table 3, MB4 Tools QuickPanel Project Files

4.2 MB4 Tools MotionDesk Project Files

The MB4 Tools package consists of a group of MotionBASIC files (ER4HDLR.BPS, QP4TOOLS.BPS and APPLSUBS.BAS) which are to be included in your MotionDesk Project.

4.2.1 Error Handler File

In the event of a program error or fault the Error Handler (ER4HDLR.BPS) stores the necessary error/fault information for later display, and provides a configurable method for dealing with machine shutdown and reset. The Error Handler contains subroutines for the various standard error displays (QuickPanel, MMI-840, and MotionDesk Console), it is configurable by modifying subroutines called in APPLSUBS.BAS. **No changes are required to the MB4 Tools Error Handler itself, which is a write protected file.** The Error Handler program file (ER4HDLR.BPS) is found in the \Program Files\Ormec\MotionDesk\Library\ directory.

4.2.2 Application Specific Subroutines File

The MB4 Tools Application Specific Subroutines file (APPLSUBS.BAS) contains several subroutines which are called by the MB4 Tools Error Handler. This file is modified as needed to fit your application, and is found in the \Program Files\Ormec\MotionDesk\Templates\ directory.

Chapter 5

Writing Your Program

5 Writing Your Program

This chapter tells you what you have to do to write a MotionBASIC application program based on the MB4 Tools. The Application Specific Subroutines chapter also contains several steps related to writing your application program. Also, refer to the Useful MB4 Tools Variables & Subroutines chapter for other MB4 Tools features you can use for your application program.

5.1 Overview

To write a MotionBASIC application using the MB4 Tools, you will need to do the following things, pretty much in the order they are listed.

- Plan the structure of your program, see the Planning Your Program Structure section of this chapter.
- Set up your development environment, see the Setting Up Your Development Environment section of this chapter.
- If you are using a QuickPanel, you will need to add panels to those included in the MB4 Tools QuickPanel Project files (ER4QP5_.PRJ), see the Configuring MB4 Tools QuickPanel Display chapter.
- Modify the application specific subroutine file (APPLSUBS.BAS) as needed for your application, see the Application Specific Subroutines chapter.
- Install the completed program, see the Installing the Finished Program chapter.

5.2 Planning Your Program Structure

If you want to write your application program in the minimum amount of time and have it work right, there is no substitute for a good plan. Our

experience, in writing hundreds of application programs, has shown us that time spent planning at the start of a project pays for itself many times over by the end of the project.

First make sure you have a detailed and complete specification for the application. It should include all of the machine motion sequences, the I/O sequences, operator set-up data, manual operations, etc.

Then allocate all of the machine *procedures* into logical groups or *modes*. Think of a *procedure* as one or more subroutines that allow the machine to perform a particular sequence or operation. A *mode* is a machine state in which there is a specific group of procedures that may be executed.

A subroutine can be used by one particular procedure or by more than one procedure. A typical list of modes and procedures would look like the Figure 9.

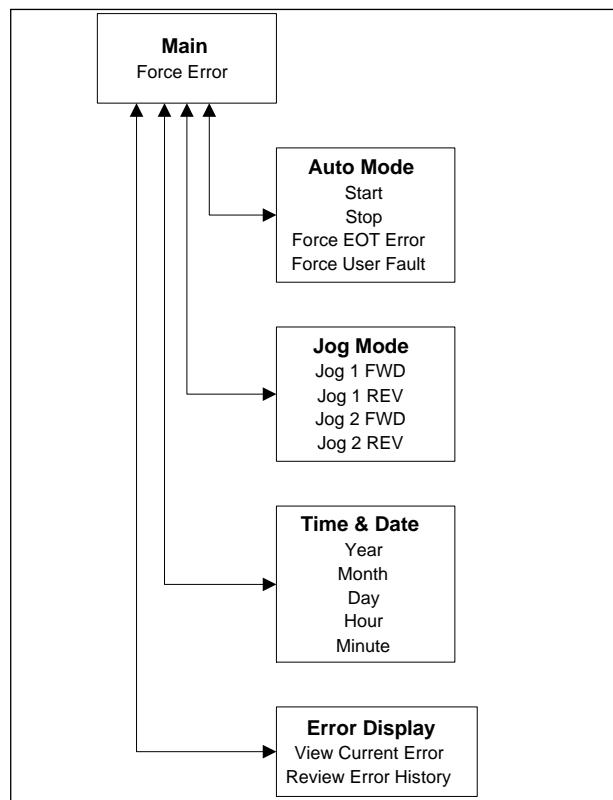


Figure 9. Program structure diagram

From this diagram of modes and procedures you can decide how to divide your program into a logical set of subroutines.

5.3 Avoiding Conflicts

Conflicts between your application modules and MB4 Tool modules will occur when you use a variable or label name in your program that is also used in the MB4 Tools. When done intentionally, you can achieve useful results, when done inadvertently, it can have unpredictable results.

Appendix A1 includes a list of all variable and label names used in the MB4 Tools. You should keep these in mind and avoid inadvertently re-using them in your modules. Refer to the Useful MB4 Tools Variables & Subroutines chapter for further information.

5.3.1 MotionBASIC Variable Name Prefixes and Suffixes

MotionBASIC variables are global variables, a program routine in any thread can access a variable regardless of which thread the variable was originally defined. This global variable accessibility can create particularly difficult to find (and eliminate) programming conflicts in multi-threaded application programs. Management of variable access in multi-threaded programs is critical to proper program execution.

In general, the best policy in a multi-threaded MotionBASIC application program is to avoid using identical variable names in multiple threads, except when needed to pass information between threads.

There will be instances when the same variable name is applicable to routines in different threads. Using the same, or similar, variable names for each instance of a variable with the same functionality helps make a program easier to read and understand. The addition of prefixes and suffixes to duplicate variable names can be used to make the variables unique to a particular thread, without detracting from program readability.

For example, in an application with variables applicable in three different threads, adding a 1, 2 or 3 to the end (e.g. ProductCount1, ProductCount2, and ProductCount3), or an A, B or C to the beginning (AProductCount, BProductCount, and CProductCount) of all the variable names unique to each thread allows each to be individually accessed, without detracting from the programs readability.

All variables defined for use exclusively by the MB4 Tools have an "e" prefix, all those defined for use by the application program have a "g" prefix. To be certain to avoid using a variable intended only for use by the MB4 Tools, avoid using MotionBASIC variable names starting with an "e". Refer to Appendix A1 for a list of all the MB4 Tools variable names.

5.4 Setting Up Your Development Environment

This process consists of creating a directory for your project, copying some files into it, and configuring your ORION. If you have access to the machine that the program will control, this is also a good time to set up the controller and axis configurations.

5.4.1 Directory and Files

Create a sub-directory off of the \Program Files\Ormec\MotionDesk\ directory for your program files. Call it anything you want, we will use \Program Files\Ormec\MotionDesk\Project\ for our examples.

Copy all of the files from the \Program Files\Ormec\MotionDesk\Templates directory into your project directory (\Program Files\Ormec\MotionDesk\Project), this is where all of your project files will be kept.

5.4.2 Opening the MB4 Tools MotionDesk Project

The following steps will create a new MotionDesk project using the MB4 Tools. A working knowledge of MotionDesk 2.0 is required, refer to the MotionDesk 2.0 Help for further information.

1) Run MotionDesk 2.0, and create a new project (File | New Project).

Using the MotionDesk Project Navigator (View | Project Navigator):

- 2) Add the MB4 Tools Error Handler (\MotionDesk\Library\er4hdr.bps) library module to the Project (ProjNav | Insert | Library Module).
- 3) Add the QuickPanel Routines (\MotionDesk\Library\Qp4tools.bps) library module to the Project (ProjNav | Insert | Library Module).
- 4) Add the MotionDesk AxisTune (\MotionDesk\Library\Axistune.bas) library module to the Project (ProjNav | Insert | Library Module).
- 5) Add the MB4 Tools Application Specific Subroutines (\MotionDesk\Project\Applsubs.bas) module to the Project (ProjNav | Insert | Module).
- 6) Add a module to the Project (ProjNav | Insert | Module) which will be your application program, name it whatever you like. MotionDesk will ask if you want to create this program, since it does not yet exist, answer "Yes". Be sure that you save all your application program modules in the Project directory (\MotionDesk\Project\).
- 7) Save your Project (File | Save Project As) in the Project directory (\MotionDesk\Project).

5.4.3 Loading the MAP and QuickPanel MBXs (QuickPanel Users Only)

If you are using a QuickPanel or QuickPanel 2 as your operator interface, you will need QuickDesigner or QuickDesigner 2 (depending on the version QuickPanel) and a working knowledge of how to use it to modify the panels and compile a new QuickPanel program.

The MAP and QuickPanel Communications MotionBASIC Extensions, MBX-MAP and MBX-QP-4 respectively, are required in order to use the MB4 Tools with a QuickPanel. Verify that you have the required MBXs installed in your ORION using the MotionDesk SysInfo utility. Refer to the MotionDesk Help, SysInfo section for further information.

If you do not have the appropriate MBXs (MBX-MAP-4 and MBX-QP-4) installed in your ORION, you can install them using the MotionDesk Upgrade Director. Refer to the Upgrade Director section of the MotionDesk Help for further information.

5.4.4 Configuring the Controller and Axes

Using the MotionDesk Project Navigator Unit and Axis Settings utilities, configure the controller and axes. For more information, refer to the MotionDesk Help Manual and the ORION™ Motion Controller Installation and Operation Manual.

5.5 Configuring Your Application Program

Your application program will need to include the initialization code described in this section. Exactly which initialization code is required depends on which operator interface you are using.

Refer to the Useful MB4 Tools Variables & Subroutines chapter for information regarding MB4 Tools variables which you can configure during your program initialization.

Generally, there are two types of initialization that application programs need. Certain things must be initialized each time the controller is turned on, and some things are initialized only once when the program is first installed.

The following is a list of some common things that should be initialized:

- **String Variables** are not preserved through power-off cycles, they must be initialized every time the program is run.
- **ORMEC Variables** are not preserved through power-off cycles, they must be initialized every time the program is run. The MP.CONFIG command can be used to restore the ORMEC variable values established during unit and axis configuration using MotionDesk.
- **Variable MAP Statements** are not preserved through power-off cycles, they must be executed every time the program is run. This is only required if you are using a QuickPanel.
- **Non-Volatile Variables** remain unchanged by a power-off cycle, they should be initialized once, or every time the program is run, depending on whether or not you want to reset their values.
- **Volatile Variable** values are reset by a power-off cycle, they should be initialized every time the program is run.

5.5.1 If you are using a QuickPanel...

If you are using a QuickPanel in your application, you will need to include the code shown in Figure 10 as part of your program initialization. This code initializes and arms the MB4 Tools Error Handler, and enables the QuickPanel communications through Serial Port 1 (SRL1:).

```

gDebug =false 'enables the MB4 Tools error handler
            'set to true to disable MB4 Tools error handler
gNoMap =false 'enable MBX-MAP statements
gSize =10     'configure the length of the error history log

'----- configure the error history log file only if you are
'----- using it
gLogFile$ ="errlog.bin"
            'config filename for error history log file
gLogFileSize=25 'configure error history log file size (if using)

'----- initialize the error handler (Er4hdlr.bps)
ErrInit

'----- perform your other program initialization here, incl. MAP
'----- statements if you are using a QuickPanel.

'----- enable the QuickPanel
qp.close    'close quickpanel communications
event on    'events must be on for ormec variable access
qp.open 1,"srl1:" 'enable QuickPanel comm. through SRL1 port
qp@=on
qp.dnld "test.qp2" 'download test.qp2 if a change has been made

QPDogTime = 500 'QP watchdog polling interval
QPWatchdogId =Start(QPWatchdog) 'start the QPWatchdog thread

'----- arm the error handler (Er4hdlr.bps)
on error goto ErrHdlr

```

Figure 10. Application program initialization code for using a QuickPanel

5.5.2 If you are using an MMI-840...

If you are using an MMI-840 in your application, you will need to include the code shown in Figure 11 as part of your program initialization. This code initializes and arms the MB4 Tools Error Handler, and enables the MMI-840 communications through Serial Port 1 (SRL1:).

```
gDebug =false 'enables the MB4 Tools error handler
             'set to true to disable MB4 Tools error handler
gNoMap =true  'disable MBX-MAP statements
gSize =10    'configure the length of the error history log

'----- configure the error history log file only if you are
'----- using it
gLogFile$ ="errlog.bin"
             'config filename for error history log file
gLogFileSize=25 'configure error history log file size (if using)

'----- initialize the error handler (Er4hdr.bps)
ErrInit

'----- perform your other program initialization here

'----- enable the MMI-840
close 0
open "r",0,"srl1:mmi"

'----- arm the error handler (Er4hdr.bps)
on error goto ErrHdlr
```

Figure 11, Application program initialization code for using an MMI-840

NOTE: Using an MMI-840 operator interface disables display of error information in the MotionDesk Console window.

5.5.3 If are not using an MMI-840 or QuickPanel Operator Interface...

If you are not using a QuickPanel or an MMI-840 operator interface in your application, you will need to include the code shown in Figure 12 as part of your program initialization. This code initializes and arms the MB4 Tools Error Handler.

```

gDebug =false 'enables the MB4 Tools error handler
              'set to true to disable MB4 Tools error handler
gNoMap =true  'disable MBX-MAP statements
gSize =10    'configure the length of the error history log

'----- configure the error history log file only if you are
'----- using it
gLogFile$ ="errlog.bin"
              'config filename for error history log file
gLogFileSize=25 'configure error history log file size (if using)

'----- initialize the error handler (Er4hdr.bps)
ErrInit

'----- perform your other program initialization here

'----- arm the error handler (Er4hdr.bps)
on error goto ErrHdlr
    
```

Figure 12, Application program initialization code for no operator interface

5.6 Viewing the Alarm Display

The MB4 Tools Error Handler does not automatically switch to the alarm display when an error occurs. It is left to the programmer to write their application program to call the Alarm Display subroutine at the appropriate point during the machine cycle. Refer to Table 4 for a list of the various operator interfaces and the corresponding Alarm Display routines.

Display Type	Alarm Display Routine
MotionDesk Console	ErrDisplayCon
QuickPanel	ErrDisplayQP
MMI-840	ErrDisplayM8

Table 4, MB4 Tools Alarm Display Subroutines

The Alarm Display routines handle all aspects of the alarm display, including the key presses. When the operator presses the Exit key on the operator interface, the Alarm Display routine RETURNS and the application program execution continues.

The MB4 Tools Alarm Display routines for the various operator interfaces are written to operate as separate threads. This allow your application program

to be written such that it will continue machine operation independent of the Alarm Display.

5.6.1 Preventing Operator Exit from the Console Alarm Display

Because of the Console Alarm Display's ability to detect whether or not MotionDesk is connected and the Console Window opened, there is no need to (E)xit the Console Alarm Display. The ability to (E)xit the Console Alarm Display is provided for those user who will be using the Console Window for more than just the Alarm Display.

If you want to be able to connect to your ORION and review the error history using MotionDesk, without stopping machine operation, you must prevent the user from (E)xiting the Console Alarm Display. Refer to the example in Figure 13.

```
'----- included at the end of the application program initialization
:
if thread.state@(ConsoleWindowId) =0 then
  ConsoleWindowID =start(ConsoleWindow, 0)
endif
:
'-----

'----- separate routine in the application program
ConsoleWindow:
  while true
    ErrDisplayCon
  wend
  ConsoleWindowId =0
end thread
```

Figure 13, Preventing Operator from Exiting Console Alarm Display

If the operator presses the (E)xit key, the ErrDisplayCon routine Returns to this While...Wend loop, which in turn executes it again. Using this technique, as the Error Demo does, the (E)xit appears to act similarly to the (L)atest key.

5.7 Viewing the Time & Date Display

If you are using the QuickPanel Time & Date configuration screen, you must call the ErrDateTime subroutine either when the Time & Date configuration screen is displayed, or to force it's display at the appropriate point in the machine cycle. The ErrDateTime routine will continue to execute until the Exit key is pressed, which will cause it to Return.

5.8 MB4 Tools Process ID

The MB4 Tools use 32767 as the Process ID for all error history related threads. To insure proper MB4 Tools and application program operation, **do not use Process ID 32767 when Starting or Creating a thread or**

when Ending a process. Refer to the MotionBASIC Help for further information regarding Process IDs.

5.9 Using the MotionBASIC DATE\$, TIME\$ and TIMER Variables

Setting the DATE\$ and TIME\$ variables sets the ORION's System Clock date and time respectively, which are used as the date and time stamp by the MB4 Tools Error Display.

Do not set TIME\$ ="0" when using the MotionBASIC TIMER variable to measure elapsed time. Refer to the example in Figure 14 for the recommended elapsed time measurement approach using the TIMER variable.

```
TimerExample:  
StartTime = TIMER           'save current time  
wait 100  
ElapsedTime = TIMER -StartTime 'determine the elapsed time  
return
```

Figure 14, TIMER Variable Example

5.10 Next Step

You are now ready to move on to the next step which is either configuring the MB4 Tools QuickPanel Display if you are using a QuickPanel, or configuring the Application Specific Subroutines if you are not.

Chapter 6

Configuring MB 4 Tools QuickPanel Display

6 Configuring MB4 Tools QuickPanel Display

This chapter tells you how to configure the MB4 Tools QuickPanel display for your application. You will need a working knowledge of QuickDesigner in order to make the modifications discussed in this chapter. If your application does not include a QuickPanel, the steps listed in this chapter are not required.

6.1 Importing a QuickPanel Project

At the start of each development project involving a QuickPanel, you will want to import into QuickDesigner (Files | Import) the appropriate MB4 Tools QuickPanel project (Er4qp5m.prj or Er4qp5c.prj) for the type QuickPanel unit you are using. Refer to the MB4 Tools QuickPanel Project Files section of the MB4 Tools Files Organization chapter for a list of the QuickPanel project files, their location, and corresponding units. Also refer to the QuickDesigner manual for further information regarding importing a project.

The MB4 Tools QuickPanel Projects (Er4qp5m.prj and Er4qp5c.prj) were originally developed on 5" QuickPanels. If you are using a 9", or 10.5" QuickPanel, you will want to resize the text appropriately (no changes should be required if you are using a 6" QuickPanel).

6.2 Adding Panels to Your Application

You can add panels to your application by using QuickDesigner to design the panels and adding the appropriate MAP statements to the program initialization portion of your application program. This section details how the QuickPanel uses MAP statements to access MotionBASIC variable values associated with the panel objects you add to your application. It also provides

recommendations for maintaining a consistent "look and feel" among your application panels. Refer to the QuickDesigner Manual for further information regarding the use of QuickDesigner in designing QuickPanel displays.

6.2.1 Panel ID Numbers

Panel ID numbers 1 and 2 are reserved for use by the MB4 Tools, the maximum panel ID is 32766. We recommend that your application use panels starting at 50 in order to allow for future expansion of the MB4 Tools. Refer to Table 2 for a list of the panels supplied with MB4 Tools and their corresponding panel ID numbers.

Panel Name	Panel ID
Alarm Display	1
Time & Date Config	2

Table 5, MB4 Tools panels and ID numbers

The gQpPanel MotionBASIC variable is used by the MB4 Tools programs to monitor and change the currently displayed panel. Your program can determine the currently displayed panel ID number by reading the value of gQpPanel. Your application can also command the QuickPanel to display a particular panel by setting the value of gQpPanel to the appropriate panel ID. Refer to Displayed Panel Control section of this chapter for further information regarding control of the currently displayed panel.

WARNING: Setting gQpPanel to an invalid panel ID will result in a blank QuickPanel display.

6.2.2 Adding MAP Statements

All QuickPanel objects (pilot lights, push buttons, etc.) which require the QuickPanel to exchange numeric data with an ORION™ have a Tag register value associated with them. This Tag register value corresponds to a MAP Reference Number (MRN) in the ORION™, which acts as a means of identifying a MotionBASIC variable. In this way MotionBASIC variables can be referenced in the QuickPanel, even though the QuickPanel does not allow the use of named variables.

MAP Reference Numbers are between 1 and 9999, with numbers 1 through 199 reserved for use by the MB4 Tools. Refer to Table 5 for a representation of the relationship between Tag register values, MAP Reference Numbers (MRNs), and MotionBASIC variables.

MAP Reference Numbers (MRNs) do not have their own value, They always represent the value of the variable to which they are mapped. If the value of the MotionBASIC variable is changed, the value of the corresponding MRN also changes, and vice versa.

All the MAP statements for your application should be kept together in the initialization section of your program. Refer to Figure 15 for a few MAP statement examples. Refer to the MBX-MAP-4 Windows Help for further

information regarding the MAP functionality, and the QuickDesigner User's Manual for further information regarding QuickPanel Tag Registers.

QuickPanel Tag Register	MAP Ref. Number	MotionBASIC® Variable
OS0201	201	ForceError
OS0202	202	Version
OS0203	203	Jog1.Fwd

Table 6, QP Tags, MAP Reference Numbers, & MotionBASIC Variables

```
'map the QuickPanel registers
MAP 201 TO ForceError      'OS0201
MAP 202 TO Version        'OS0202
MAP 203 TO Jog1Fwd        'OS0203
```

Figure 15, MAP Statements in program initialization routine

6.2.2.1 MAP Reference Number Guidelines

This section provides a list of guidelines which should make your assignment and maintenance of MAP Reference Numbers easier.

- 1) MAP Reference Numbers 1 through 199 are reserved for use by the MB4 Tools, numbers 200 through 9999 may be used by the application.
- 2) You should document the corresponding Tag register value for each MAP statement by adding a comment with the Tag register value to the end of MAP statement. Refer to Figure 15 for an example of how this can be done. This will make working with the MAP References easier and aid in program troubleshooting.
- 3) Group the MAP Reference Numbers either by panel or function so that they are easier to locate in the list, and leave a blank line between each group.

6.2.3 QuickPanel Color Conventions

It is recommended that you maintain a consistent color strategy throughout your application. Consistent use of color conventions to indicate different types of information and functions will make your operator interface easier to understand and use. The following is a recommended color standard.

For dual-scan color and TFT QuickPanels (MMI-QP2/_C & MMI-QP2/_T)

- Panel Titles (legend plate, local message display, etc.) - White text on a blue background with a blue outline. Where practical, the panel title should be centered across the top of the screen.

- Informational Text (numeric data display, legend plate, local message display, etc.) - Black text on a cyan background with a cyan outline. Any read only data (e.g. axis position, machine status, instructions, etc.) is considered informational text.
- Inactive Alarms (alarm window object, not used by MB4 Tools) - White text on a black background.
- Active Alarms (alarm window object, not used by MB4 Tools) - White text on a blinking red background with a red outline.
- Inactive Status Indicator (pilot light, local message display, etc.) - White text on a black background with a yellow outline.
- Active Status Indicator (pilot light, local message display, etc.) - Black text on a yellow background with a yellow outline.
- Numeric Data Entry (numeric data entry) - Black text on a yellow legend plate, with black text on a white data field.
- Action Items (push button, selector switch, etc.) - White outline.
- Switch Panel Buttons (goto panel button, push button, etc.) - White text on a black background with a white outline. Where practical, the switch panel buttons should be across the bottom of the panel.

For monochrome LCD and EL QuickPanels (MMI-QP2/_M & MMI-QP2/_E)

- Panel Titles (legend plate, local message display, etc.) - Black text on a white background with a white outline. Where practical, the panel title should be centered across the top of the screen.
- Informational Text (numeric data display, legend plate, local message display, etc.) - White text on a black background with a white outline. Any read only data (e.g. axis position, machine status, instructions, etc.) is considered informational text.
- Inactive Alarms (alarm window object, not used by MB4 Tools) - White text on a black background with a white outline.
- Active Alarms (alarm window object, not used by MB4 Tools) - Black text on a white background with a white outline.
- Inactive Status Indicator (pilot light, local message display, etc.) - White text on a black background with a white outline.
- Active Status Indicator (pilot light, local message display, etc.) - Black text on a white background with a white outline.
- Numeric Data Entry (numeric data entry) - White text on a black background with white outline.
- Action Items (push button, selector switch, etc.) - White outline.
- Switch Panel Buttons (goto panel button, push button, etc.) - White text on a black background with a white outline and border text (such that "GOTO PANEL" doesn't show). Where practical, the switch panel buttons should be across the bottom of the panel.

6.2.4 Displayed Panel Control

The Goto Panel object allows the operator to change panels by pressing a button on the QuickPanel, without the QuickPanel being commanded to change the displayed panel by the application program. There are instances when this is not desirable.

For example: Suppose your Run Mode panel has a Start/Stop button that initiates machine operation. The operator starts the machine operation and then changes panels several times and the machine suddenly begins jamming. In order for the operator to stop the machine quickly, without using the E-Stop button, they must figure out how to get back to the Run Mode panel and the Stop button. To prevent this situation, you may want to prevent the operator from leaving the Run Mode panel while the machine is in motion.

The following is an example of how you could write your application program to use a pushbutton to change the displayed panel. Using a pushbutton instead of a GoTo Panel button allows the application program to control when the operator from leaves a particular panel.

NOTE: The following example assumes that the pushbutton Action parameter is configured for Momentary.

```

while in MainLoop
  if State =Running then           'if the machine is running
    gQpPanel =RunScrn           'display the run screen
    :
    :
    :
    'run mode programming
  else
    'if the machine is NOT running
    if PanelChangePB then         'if the Panel Change PB is on
      gQpPanel =ManualScrn       'switch to the manual screen
    endif
  endif
wend

```

Figure 16, Program module to control the displayed panel with a pushbutton

Chapter 7

Application Specific Subroutines

7 Application Specific Subroutines

The MB4 Tools package includes a file called APPLSUBS.BAS. This file is a collection of subroutines that are called by the MB4 Tools programs to perform various functions. Many of these functions are things that you will want to customize for your application. They are grouped together in a single file to make them more convenient to modify.

The application specific subroutines perform the following operations:

Subroutine	Functions
ApplFault	Application specific error handler
ApplFltTxt	Allows program to customize error message text
ApplNames	Assigns application specific axis and machine names

7.1 Application Error Handler (ApplFault Subroutine)

The MB4 Tools Error Handler does not trap and service errors, it's only function is to store error information for later display. It is left to the programmer to provide an error handler for the application, which should be placed in the ApplFault subroutine.

With two exceptions, the ApplFault subroutine is automatically called by the ErrHdlr routine whenever an error or fault is detected. The exceptions to this rule are when an 1805 error is generated (<CTRL><BREAK> or the MotionDesk Abort button), and for any error that occurs when the MB4 Tools gDebug variable is TRUE.

NOTE: The ApplFault subroutine ends with a *Resume* statement, do not use a *RESUME <Label>* statement to end the ApplFault routine. Using a

Resume <Label> statement ends all threads, including the error display thread, which results in improper error logging and display

For further information regarding writing an error handler for your application, refer to the MotionBASIC Programming Help.

7.2 Application Specific Error Messages (ApplFltTxt Subroutine)

Your application may have errors for which you want to display customized error information, you can use the ApplFltTxt subroutine to do this. The egTxt\$() array variable is used to store the error display information. Refer to Figure 17 for an example of how to modify the error display text for a user defined error.

```
ApplFltTxt:
if err =1 then
    egTxt$(1) ="Unloader Error"
    egTxt$(2) ="Input Jammed"
    egTxt$(3) ="Clear Jam and Reset"
endif
return
```

'user defined error 1

Figure 17, Application specific error text example

The MB4 Tools are default configured to display 36 characters on each line of the error display panel. This means that only the first 36 characters of each line of text stored in the egTxt\$(1), egTxt\$(2), and egTxt\$(3) variables will be displayed.

7.3 Application Specific Axis and/or Machine Names (ApplNames Subroutine)

The MB4 Tools stores the name for each axis in the gAxname\$() array variable, indexed by the axis ID number. By default, each axis is named using it's axis ID (e.g "Axis 1", "Axis 2", etc.). The ApplNames subroutine can be used to change the default axis names to better fit your application. Refer to Figure 18 for an example of how to use the ApplNames routine to change your application's axis names.

```
ApplNames:
gAxname$(1) = "Pacer encoder"      'rename axis 1
gAxname$(2) = "Unwind Axis"       'rename axis 2
gAxname$(3) = "Nip Roll Axis"     'rename axis 3
gAxname$(4) = "Rewind Axis"       'rename axis 4
return
```

Figure 18, *Application specific axis naming example*

If you are using the ORION's Machine-Stop/No Fault feature, you can use the gMname\$() array variable to customize the machine name displayed by the MB4 Tools error display, similarly to using the gAxname\$() array variable for the axis names. For further information regarding the ORION Machine-Stop/No Fault feature, refer to the ORION I&O Manual and the M-STOP section of the MotionBASIC Help.

Chapter 8

Useful MB4 Tools Variables & Subroutines

8 Useful MB4 Tools Variables & Subroutines

This Chapter describes some useful MB4 Tools variables and subroutines you can use in your program.

8.1 Error Handle Debug Flag

- **gDebug** can be used to disable the MB4 Tools Error Handler. This is useful during debug when you do not want the program to continue operation after a program error or fault. gDebug should be configured as part of your application program initialization code.

Examples:

```
gDebug =TRUE      'disables the MB4 Tools Error Handler
gDebug =FALSE     'enables the MB4 Tools Error Handler
```

8.2 MAP Statement Disable Flag

- **gNoMap** can be used to disable the MB4 Tools Error Handler MAP statements. This is useful for those applications not using a QuickPanel operator interface, setting the gNoMap = true eliminates the need for installing MBX-MAP-4 in these types of applications. gNoMap should be configured as part of your application program initialization code.

Examples:

```
gNoMap =TRUE      'disables the MB4 Tools MAP statements
gDebug =FALSE     'enables the MB4 Tools MAP statements
```

8.3 Error History Variables

- **gSize** configures the length of the non-volatile error history log. You can configure this during the initialization portion of your application program. If gSize is 0 when ErrInit routine is called, gSize will be set to 10.

Example:

```
gSize =20           'configure the length of the error history log
```

- **gLogFile\$** is the filename, including drive and directory path, for the error history log file. If you do not configure this file name, the MB4 Tools will not write the error history log to a file. NOTE: The alarm history information is stored in the ORION's non-volatile memory, use of the error history log file is not required.

Example:

```
gLogFile$ ="e:\logfile.bin"   'configure error history log filename
```

It is recommended that the error history log file be stored on an SRAM PC Card (PCC-SRAM/#####), not on the System Card (PCC-SYS4/#####).

- It is recommended that the error history log file not be written to the System Card because if the ORION loses power while writing a file on the System Card, the System Card may be corrupted which will prevent the ORION from operating properly during powerup.
- It is recommended that an SRAM PC Card (PCC-SRAM/#####) be used for data file storage, instead of a Flash RAM PC Card (PCC-FLASH/#####). Data access is significantly faster with an SRAM PC Card versus a Flash PC Card, especially when the Flash PC Card is performing its utilization leveling function (which can take up to several seconds). Because no other MotionBASIC program execution occurs while a data file is being accessed, it is important to keep this time to a minimum.
- **gLogFileSize** configures the number of Alarms for which information will be stored in the Error Log File.

Example:

```
gLogFileSize =25       'config the length of the error history log file
```

8.4 QuickPanel Related Variables

- **gQpPanel** can be used to determine the currently displayed panel, or to cause the QuickPanel to display a particular panel.

CAUTION: Setting **gQpPanel** to an invalid panel ID will result in a blank QuickPanel display.

Examples:

```
gQpPanel =MainMenu      'displays Main Menu panel
LastPnl=gQpPanel        'sets LastPnl = last displayed panel ID
```

The following parameters are configured by the MB4 Tools and should not be changed:

- **gAlarmPanel** is configured for the Error Display panel ID number [gAlarmPanel =1].
- **gDatePanel** is configured for the Time & Date panel ID number [gDatePanel =2]

8.5 Adding Message Text to the Error History Log

It is often useful to add non-error related message to the error history log, the ErrLog subroutine allows you to do this. The MB4 Tools automatically store the time and date with your message text.

- 1) Put the text you want added to the error history log in the gQPTxt\$() array variable.
- 2) Call the ErrLog subroutine

Refer to the example in Figure 19.

```
gQPTxt$(1) = "Opeator has started the machine" 'store event msg
gQPTxt$(2) = "Line Speed =" + str$(LineSpeed) 'store line speed
gQPTxt$(3) = "Product Count =" + str$(ProductCount) 'store product cnt
ErrLog 'add to error history log
```

Figure 19. Adding Messages to Error History Log Example

8.6 Viewing the Error History Log in the MotionDesk Console Window

The ErrDisplayFile subroutine can be used to view the contents of the error history log in the MotionDesk Console window. Caution should be exercised when executing this subroutine from the MotionDesk Direct Mode window as it will suspend normal program execution until the error history file printing is complete.

8.7 Monitoring QuickPanel Communications Status (QPWatchdog)

If a QuickPanel operator interface is switched off, disconnected or fails, it is no longer requesting information from, or sending information to, the motion controller. If the operator has pressed a button on the screen that started a machine operation, jogging for example, the motion controller will not detect that the MMI is no longer connected and the axis would continue jogging. The QPWatchdog routine provides a mechanism for detecting the loss of QuickPanel communications. Without use of the QPWatchdog routine, the only way the operator could stop the machine in this scenario would be with the system Emergency Stop.

The QPWatchdog routine is intended to be initiated as a separate thread which periodically polls the MBX-QP-4 communications message counter QP.CTR(4) to determine the QuickPanel communications status. The technique used in the QPWatchdog routine samples this counter periodically and sets the variable QPOk =TRUE if the number of messages has changed since the last sample or FALSE if it has not changed. The application program can then use QPOk to take appropriate action.

8.7.1 QPWatchdog Initialization

The QPWatchdog polling interval, as defined by the QPDogTime variable, needs to be set long enough to avoid false trips, but short enough to be responsive. It has to be greater than the longest interval between QuickPanel screen updates, and greater than the longest inter-session delay. This is going to vary depending on how many objects are on each screen, what types of objects they are and how they are mapped to program variables. A good starting point would be the default 500 msec. You can then increase or decrease it from there.

The QPDogTime variable is default configured for 500 milliseconds in the ErrInit routine. The QPWatchdog polling interval can be changed in your application program by changing the value of QPDogTime, however, any changes to QPDog Time must be made after running the ErrInit routine. Refer to Figure 10 of the Writing Your Program chapter, and Appendix A2, for further information regarding QPWatchdog initialization.

The QPWatchdog thread is not created by the MB4 Tools, it must be Started by your application program. Refer to Figure 10 of the Writing Your Program chapter, Appendix A2, and the START chapter of the MotionBASIC Help for further information.

8.7.2 Using QPOk

There are two ways your MotionBASIC application program can use QPOk. When QPOk goes FALSE it can simply stop whatever action is taking place, jogging for example, or it can create a fault.

To stop whatever action is in progress, simply include a test for QPOk=TRUE in your action loop. Refer to Figure 20 for an example of how to use QPOk to stop an axis that is jogging (taken from the Error Demo program, Er4demo.bas).

```

if Jog1Fwd =1 then      'is it still pressed
  move ax1~ to stl.fwd@(ax1~-) -45 'yes, move to just before the stl
  ' ----- wait for the button to be released
  wait until Jog1Fwd=0 or QPOk =false
  halt ax1~           'stop the axis
  Jog1Fwd =0
endif

```

Figure 20, Example of using QPOk

If the Jog1Fwd variable (mapped to the Jog button on the QuickPanel) is non-zero, the axis will start jogging. The program uses a "move to stl.fwd@(Ax1~-)-45" so the axis will automatically stop 45 degrees before the forward software end of travel limit. It then waits for the operator to release the Jog button or for the QPOk variable to go false. If either of these happen, it will stop the axis and set the variable mapped to the Jog button to zero. It has to set this variable to zero because if it stopped the axis when QPOk went false, Jog1Fwd would still equal 1 and the Jog would be started again on the next pass through the loop.

NOTE: When the QuickPanel changes from one screen to another, communications stop for a longer time than normal, which can cause QPOk to go false.

Appendix A1

List of Variables and Labels

List of Variables and Labels

The following is a list of all variables and labels used in the MB4 Tools application shell program and associated modules.

Variables Used

afault@()	alarm@()	axis.err1@	axis.list@	axis.merr1@()
console@	create()	ecDate\$	ecKey\$	ecNum
ecOffset	ecPtr	ecStr\$	ecTime\$	ecTmp
ecTxt1\$	ecTxt2\$	ecTxt3\$	edNum	edOffset
edPtr	edTmp	eErr	eFifo\$()	eFifo&()
egAflt	egAlrm	egAx1	egErl&	egErm
egErp&	egErr	eGet	egfNum	egFptr
egFtxt\$	egGet1	egLoc\$	egMerr	egTmp
egTmp1	egTxt\$()	ehAx1	Ehist()	eHist()
eHistProc	ehMach~	ehTmp	ehTmpDate\$	eiTmp
eiTmp\$	eiTmp1	elTmp	emDate\$	emFirst
emKey\$	emLast	emLastKey	emNum	emOffset
emPrevKey	emPtr	eMsg\$	emStr\$	emTime\$
emTmp	emTxt1\$	emTxt2\$	emTxt3\$	eprLast
eprLen	eprNum	eprPtr	eprTmp	eprTmp\$
eprLen	eprNum	eprPtr	eprTmp	eprTmp\$
eprTmp1\$	eprTxt\$	ePtr	ePtr1	ePut
ePutDone	eQpString\$	erm	erp	ErrDisplayConId
ErrGet	ErrGetId	ErrLogTxt	ErrLogTxtId	ErrPrintFile
ErrPrintFileId	ErrSendCon	ErrSendConId	ErrSendM8	ErrSendM8Id
ErrSendQp	ErrSendQpId	ErrSetDate	ErrSetDateId	ErrSetup
etmp\$	etNewDay	etNewHr	etNewMin	etNewMon
etNewSec	etNewYr	etOldDate\$	etOldTime\$	etTmp
fault@	gAlarmNum	gAlarmPanel	gAxname\$()	gDatePanel
gDebug	gDown	gDownKey\$	gEnterEnter\$	gEr4Ver
gErrInit	gEsc	gEsc\$	gExitBtn	gF1\$
gF2\$	gF3\$	gF4\$	gF5\$	gF6\$

gF7\$	gF8\$	gFkey\$()	gFltBtn()	gFltLt()
gKyDly	gKyRpt	gLastBtn	gLastLt	gLeft
gLeftKey\$	gLogFile\$	gLogFileSize	gMname\$()	gNextBtn
gNextLt	gPrevBtn	gPrevLt	gQpPanel	gQPTxt\$()
gQreg()	gRight	gRightKey\$	gSetBtn	gSize
gUp	gUpKey\$	io.mode@()	map	merr.set@
mfault@()	otl.fwd@()	otl.rev@()	thread.state@()	

Labels Used

*ApplFault:	*ApplFltTxt:	*ApplNames:	ErrAxis:
ErrConBanner:	ErrCreateText:	ErrDateTime:	ErrDisplayCon:
ErrDisplayFile:	ErrDisplayM8:	ErrDisplayQP:	ErrEncode:
ErrGet:	ErrHdlr:	ErrInit:	ErrKeysM8:
ErrLog:	ErrLogTxt:	ErrM8Banner:	ErrPrintFile:
ErrPut:	ErrSendCon:	ErrSendM8:	ErrSendQp:
ErrSetDate:	ErrSetup:	ErrStoreText:	ErrWriteText:

Web Tools Addition: Labels Used

ApplHttpPref:	ApplHttpUserLinks:	ApplHttpUserData:	ApplHttpCustomHome:
ApplHttpCustomCall	http.server.thread:	http.init:	http.string.init:
http.get.proj:	http.open.server:	http.server.waiting:	http.server.store.input:
http.respond	http.close.server:	http.user.page:	http.get.usrheader:
http.get.usrdata:	http.update.map:	http.pick.format:	http.home.page:
http.links:	http.objection.page:	http.sysinfo.page:	http.error.info:
http.sysfault.page:	http.sysio.page:	http.axstat.page:	http.axcnfg.page:
http.pgminfo.page:	http.histlog.page:		

Web Tools Addition: Numeric & String Variables

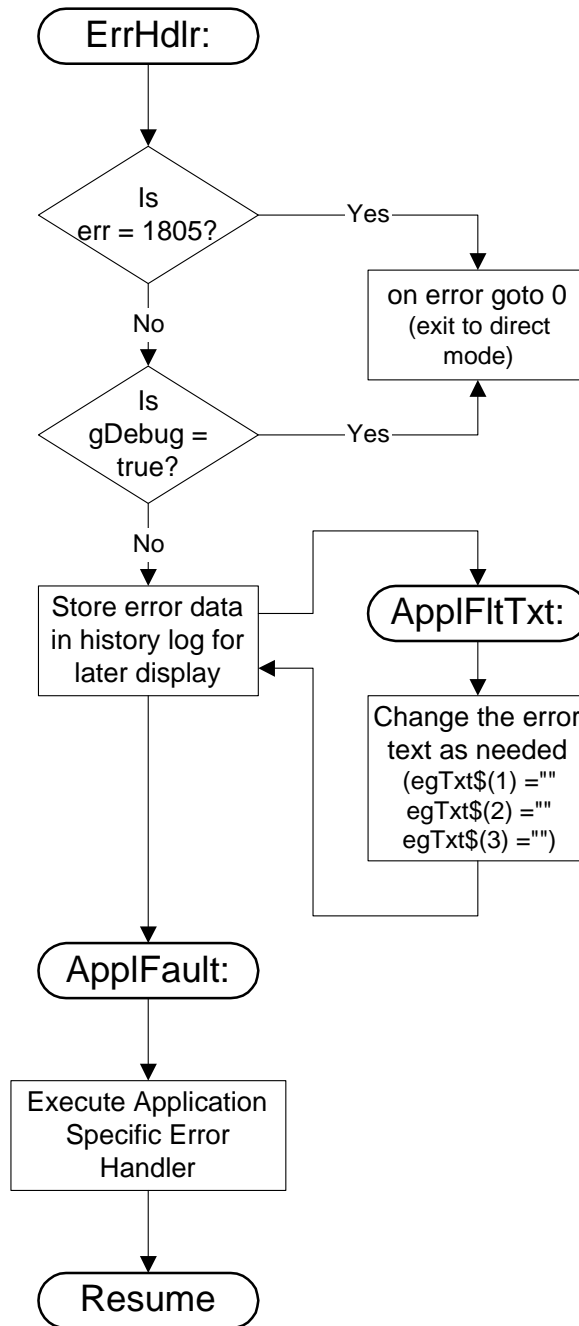
http.lognum%	http.logptr%	http.logtmp%	http.usr.mapref%
http.mapchange.loc%	http.usrpage.req%	http.pgmthrd%	http.mod.cnt%
http.lastfile%	http.filesize&	http.ax.req%	http.ax%
http.data.len%	http.data.end%	http.data.start%	http.endio%
http.startio%	http.maxio%	http.tmp2~	http.tmp%
http.tmp~	http.mach~	http.data.entry.flag%	http.usr.pgnum.data%
http.usr.read.done%	http.server.failed%	http.server.thrd.id%	http.server.thrd.rank%
http.dir.dev%	http.sd%	http.custom.flag%	http.user.flag%
http.disp.format\$	http.usrtype\$	http.tmpstr\$	http.filedate\$
http.filetime\$	http.filename\$	http.iostate\$	http.tmp\$
http.iotype\$	http.usrtext\$	http.usrlink\$	http.data\$
http.data.block\$	http.identity\$	http.server.port\$	http.server.ip.addr\$
http.currentproj\$	http.aflt.text\$()	http.flt.text\$()	http.thrdstate\$()
http.thrdtype\$()			

Appendix A2

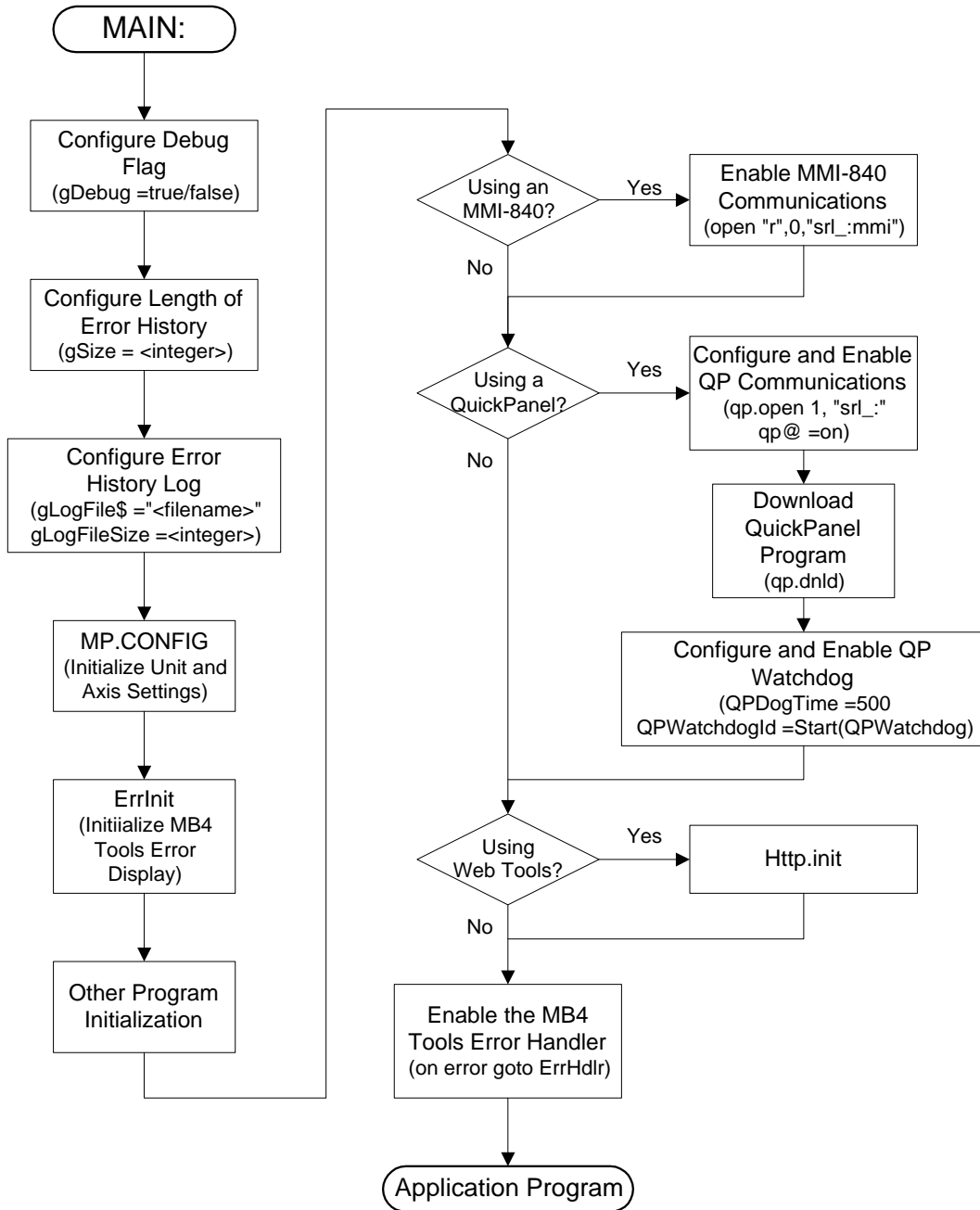
MB4 Tools Flow Charts

MB4 Tools Flow Charts

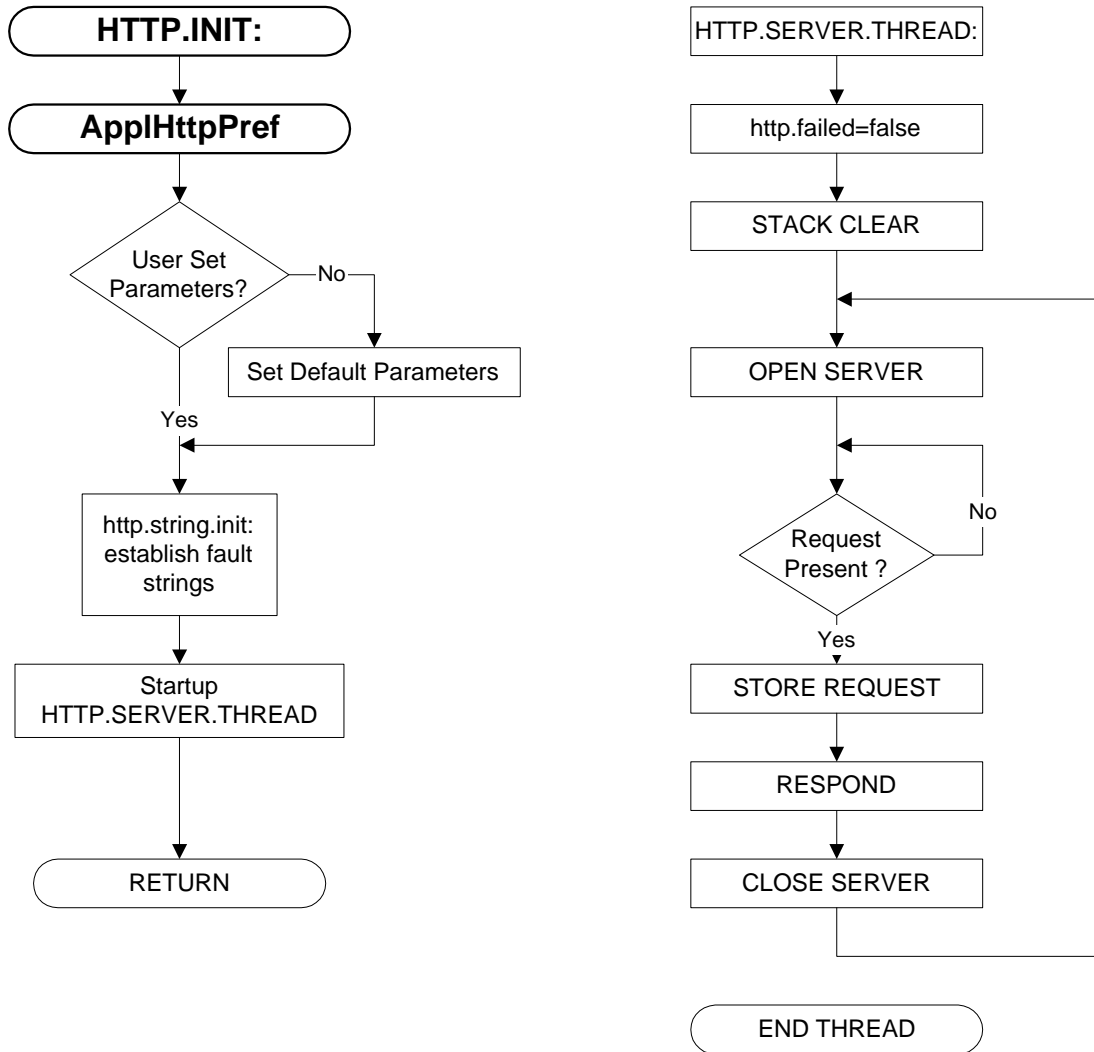
The following flow charts document the MB4 Tools shell program Error Handler, program initialization, and the Web Tools Server.



MB4 Tools Response to Faults and Errors



MB4 Tools Initialization



**MB4/5 Web Tools
 Initialization & Server Thread**

Appendix A3

Web Tools Description

Web Tools Addition

The purpose of the Web Tools addition to the MB Tools package is to provide the user with a means to remotely communicate with their Ormec Orion Controllers. Through this communication, the user then acquires the ability to have the Orion Controller report status information about its hardware, software, faults and general history of application alarms and events that have occurred.

The means by which this is done is really quite simple. A single program thread is started which one could call the "server thread". This server thread listens to a TCP/IP port, really doing nothing until such time as a valid external request for data is recognized at this TCP/IP port. When a valid request is identified, the Orion Controller responds by printing HTML (HyperText Markup Language) characters and commands to the client requesting information. The assumption is that a Web Browser package is the client which will then accept the HTML code sent by the Orion and present to the user a "web page".

The standard Web Tools provides for 9 pre-defined Web Pages. They are:

- Orion Home Page
- System Information Page
- System Faults Page
- System I/O Page
- Axis Status Page
- Axis Configuration Page
- Program Information Page
- History Log Page (dependent upon use of MB Tools Error Handling)
- Objection Page

Orion Home Page

The Orion Home Page provides a starting point from which to access the informational pages available through the use of the web tool software.

Orion Home Page

Unit Identification: Unidentified Orion Unit, ip_address:port = 200.200.200.200:7500
 Unit Time & Date: 16:01:47 07-07-1999

[HOME](#) [SYS.INFO](#) [SYS.FAILTS](#) [SYS.LCM](#) [AXIS.STATUS](#) [AXIS.CONFIG](#) [PROGRAM.INFO](#) [HISTORY](#)

Standard Maintenance Tool Pages

- [System Information Page](#)
 Displays Installed Software Version Information, System Hardware Information, Machine Setup Information, and Most Recent Error data.
- [System Faults Page](#)
 Displays the Most Recent Error data, E-Stop and Fault Information, Failed Machine Information, and Failed Axis data.
- [System I/O Page](#)
 Displays the current states & configurations of 16 System IO Points
 User entry field provided to specify the desired IO point number at which to begin the display.
- [Axis Status Page](#)
 Displays information on the Operation Status, Motion Status, and IO Status of an Axis.
 User entry field provided to specify the desired axis number.
- [Axis Configuration Page](#)
 Displays the current settings of axis configuration variables.
 Data includes: Motor Unit settings, MotionDATA settings, Motion parameter settings, Axis Inputs & Outputs, and Loop parameters.
 User entry field provided to specify the desired axis number.
- [Program Information Page](#)
 Displays the active Project Name, and lists the installed Program Source Modules.
 Program thread data is commented, as is the Most Recent Error data, and Memory Available information.
- [History Log Page](#)
 Displays an entry from the memory resident history log queue. This history log queue is provided via MotionBASIC tools software.
 User buttons allow scrolling through the queue to view entries.

Web Tools, Ver 0.99+J. Copyright © 1999 Ormer Systems Corp. All rights reserved

System Information Page

The System Information Page provides information on the Orion Software, Hardware, Machine Setups, and most recent error information.

System Information

Unit Identification: Unidentified Orion Unit, ip_address:port = 200.200.200.200:7500
 Unit Time & Date: 16:03:55 07-07-1999

[HOME](#) [SYS.INFO](#) [SYS.FAILTS](#) [SYS.LCM](#) [AXIS.STATUS](#) [AXIS.CONFIG](#) [PROGRAM.INFO](#) [HISTORY](#)

System Software Version Information

MotionBASIC	MotionBASIC Extensions
5.0.0	QF3.0.0 + QE2.0.0 + MAP4.0.0

System Hardware Information

CPU Speed Reference	1032
DSP Loop Rate	3000
Hardware Revision Codes	S4001
System E-Stop Input	NOT OK
System Faults (Bits Set)	()
Axes Installed	()
Axes Failed	()

Most Recent Error Information

Error	Error Message	Module	At Line	Thread ID	Thread Starting Label
1805	ABORT or Ctrl-Break detected	mainwin.BAS	9	5	***Thread Ended**

System Faults Page

The System Faults Page provides information about active faults with information on Errors, E-Stops, Machine Faults, and Axis Fault data.

System Faults

Unit Identification: Unidentified Orion Unit, ip_address:port = 200.200.200.200:7500
 Unit Time & Date: 16.06.01 07-07-1999

[\[HOME\]](#)
[\[SYS.INFO\]](#)
[\[SYS.FAULTS\]](#)
[\[SYS.I/O\]](#)
[\[AXIS.STATUS\]](#)
[\[AXIS.CONFIG\]](#)
[\[PROGRAM.INFO\]](#)
[\[HISTORY\]](#)

Most Recent Error Information

Error	Error Message	Module	At Line	Thread ID	Thread Starting Label
1910	E-Stop OK input open	mainfram.BAS	9	9	debug.main

System E-Stop & Fault Information

System E-Stop Input	NOT OK
System Faults (Bits Set)	{6}
Fault Bit {6}	E or M-Stop Input Not OK
All Axes Faulted	{}
1st Axis to Fault	{}

System I/O Page

The System I/O Page provides information on the state and configuration of 16 I/O points, with an input data field to specify the I/O point to begin the display.

System I/O

Unit Identification: Unidentified Orion Unit, ip_address:port = 200.200.200.200:7500
 Unit Time & Date: 16.07.19 07-07-1999

[\[HOME\]](#)
[\[SYS.INFO\]](#)
[\[SYS.FAULTS\]](#)
[\[SYS.I/O\]](#)
[\[AXIS.STATUS\]](#)
[\[AXIS.CONFIG\]](#)
[\[PROGRAM.INFO\]](#)
[\[HISTORY\]](#)

Valid I/O range 1 to 40. Starting I/O Point Number?

Type	I/O #	State
Input	34	OFF
Input	35	OFF
Input	36	OFF
Input	37	OFF
Input	38	OFF
Input	39	OFF
Input	40	OFF

Axis Status Page

Displays information on the Operation Status, Motion Status, and I/O Status of an Axis. Data Entry Field allows selection of other axes.

Axis Status

Unit Identification: Unidentified Orion Unit, ip_address:port = 200.200.200.200:7500
 Unit Time & Date: 10/20/29 07:08-1999

[\[HOME\]](#)
[\[SYS.INFO\]](#)
[\[SYS.FAULTS\]](#)
[\[SYS.I/O\]](#)
[\[AXIS.STATUS\]](#)
[\[AXIS.CONFIG\]](#)
[\[PROGRAM.INFO\]](#)
[\[HISTORY\]](#)

Axis 1 of Axes: {1,2}

Enter Axis # to view:

Axis 1 Operation Status

Operating Mode	0
Axis Fault Status	OK, No Faults
Drive Alarm Code	0
MotionData Input	OFF
MotionData Output	PASS-THRU

Axis 1 Motion Status

	Commanded	Actual	Error
Position	0	0	0
Velocity	0	0	
Drive Command	0		
Gear Ratio	0		

Axis 1 I/O Status

Input Status		Trapped Position
A-Sensor	-1	0
B-Sensor	-1	0
Z-Reference	-1	0
Hdw OTL Forward	0	0
Hdw OTL Reverse	0	
Analog Input 1	-874	
Analog Input 2	83	

Programmable Limit Switch Status

PLS 1	0
PLS 2	0
PLS 3	0

Axis Configuration Data Page

Displays the current settings of axis configuration variables. Data includes: Motor Unit Settings, MotionDATA settings, Motion Parameter settings, Axis Inputs & Outputs, and Loop Tuning Parameter settings. User entry field allows selection of other axes.

Axis Configuration Data

Unit Identification: Unidentified Orion Unit, ip_address port = 200.200.200.200:7500
 Unit Time & Date: 10:29:41 07-08-1999

[\[HOME\]](#) [\[SYS.INFO\]](#) [\[SYS.FAULTS\]](#) [\[SYS.I/O\]](#) [\[AXIS.STATUS\]](#) [\[AXIS.CONFIG\]](#) [\[PROGRAM.INFO\]](#) [\[HISTORY\]](#)

Axis 1 of Axes: (1,2)

Enter Axis # to view

Axis 1 Motor Units

Resolution and Total Inertia

Encoder Resolution: 6000 counts/rev
 Total Inertia: 0.0001 in-lb-sec²
 [1.10769E-05 Kg-m²]

Limitations and Unit Conversion Factors

<u>Motor Units</u>	<u>User Units</u>
Position: 6000 counts	360 position units
Speed Limit: 4000 RPM	4000 speed units
Accel Limit: 796 rev/sec ²	5000 accel units

Default Maximum User Parameters

Max Speed: 4000 speed units	
Max Accel: 5000 accel units	
Max Decel: 5000 accel units	
Max Drive Output: 1	0.00 in-lbs [0.00 N-m]
Error Decel Rate: 1000 accel units	

Program Information Page

Displays the active project name, lists the program source modules, program thread data is summarized, as is most recent error information, and memory available information.

User Program Information

Unit Identification: Unidentified Orion Unit, ip_address:port = 200.200.200.7500
 Unit Time & Date: 16:07:57 07-07-1999

[\[HOME\]](#) [\[SYS INFO\]](#) [\[SYS FAULTS\]](#) [\[SYS LOG\]](#) [\[AXIS STATUS\]](#) [\[AXIS CONFIG\]](#) [\[PROGRAM INFO\]](#) [\[HISTORY\]](#)

User Project: **MINIMUM.MTP**

Module #	Program Module Name	Size	Time	Date
1	webtools.bas	63292	17:55:23	07:07:1999
2	applruba.bas	10301	16:48:52	06:15:1999
3	minimum.BAS	1079	10:37:36	06:28:1999
4	ErrHdlr.bas	40485	17:55:16	06:23:1999

Program Thread Information

ID	Type	State	Rank	Process ID	Now in Module	At Line	Thread Starting Label
12	ERROR	STOP	100	0	applruba.bas	13	ErrHdlr
11	RBO	RUN	1000	0	webtools.bas	1556	http server thread
10	RBO	STOP	100	32767	ErrHdlr.bas	168	ErrGet
9	RBO	ELIG	100	0	minimum.BAS	9	debug.main

Most Recent Error Information

Error	Error Message	Module	At Line	Thread ID	Thread Starting Label
1910	E-Stop OK input open	minimum.BAS	9	9	debug.main

Memory Available Information

Numeric Data	String Space	Non-Volatile
60183	3853	30304

History Log Page

Displays an entry from the memory resident log queue. This history log queue is provided via the Error Handling MB Tools software. If MB Tools Error Handling was not used then this display is not available. User buttons are provided to scroll through the queue to view entries.



Each of these pages provides for links to all other pages. In addition to the "standard pages" the Web Tools structure allows for user enhancements and the addition of user pages. There are two classes of User Page Additions: User Configured Pages & Custom Pages.

User Configured Pages

The implementer need not know any HTML to add user pages. The user provides data which instructs our server thread as to what pieces of information and text the user would like to see on the User Page(s). Simply stated, to create a User Page, the implementer adds data statements to an Ormec Supplied routine, where the user specifies:

- 1) The User Page Number
- 2) The Link Text to call that page
- 3) The User Page Title

Once a User Page exists, it is then the task of the user to provide data which instructs the Server thread as to what exactly should be presented upon each user page. There are three different types of information that can be displayed upon a configured User Page. These types are:

- 1) Formatted Text
- 2) Read Only Variable Field (w/ description)
- 3) Read/Write Variable Field (w/ description)

The variable fields mentioned above refer to variables within the Orion that the user has mapped to an MRV (Mapped reference value). These are numeric variables from within the Orion that are accessible by a mapped reference number.

Custom Pages

The implementer would need to have knowledge of HTML as the code he will provide will essentially consist of MotionBASIC PRINT statements. These PRINT statements will be printing a sequence of characters which represents legal HTML code to the client Browser of the Orion Server.

Objection Page

When the Orion receives a request, but that request contains unexpected or unrecognized information, then the Web Tools server will present an objection page.

Objection Page

Unit Identification: Unidentified Orion Unit, ip_address:port = 200.200.200.200:7500
Unit Time & Date: 09:14:02 09-09-1999

[\[HOME\]](#) [\[SYS.INFO\]](#) [\[SYS.FAULTS\]](#) [\[SYS.I/O\]](#) [\[AXIS.STATUS\]](#) [\[AXIS.CONFIG\]](#) [\[PROGRAM.INFO\]](#) [\[HISTORY\]](#)
[\[USR1.LINK\]](#) [\[USR2.LINK\]](#) [\[USR5.LINK\]](#)

SERVER OBJECTION!

The last request was not recognized by the Orion server.

Appendix A4

Web Tools Setup

Web Tools Setup

This section contains the setup instructions to add web tools functionality to an Orion controller.

New Version Software Modules

The Web Tools addition is meant to be compatible with Orion Controllers running MotionBASIC software version 4 and higher. The newer software modules relevant to the web server enhanced tools are:

- | | |
|---------------------|--|
| APPLSUBS.BAS | Same name, but a new version (supplied w/ tools revision 1.4.0) is required. |
| WEBTOOLS.BPS | Introduction of new file, (not supplied with older MB Tools packages, but with tools package revision 1.4.0) |
| ER4HDLR.BPS | Same name, but rev 6 required (supplied w/ tools revision 1.4.0) |

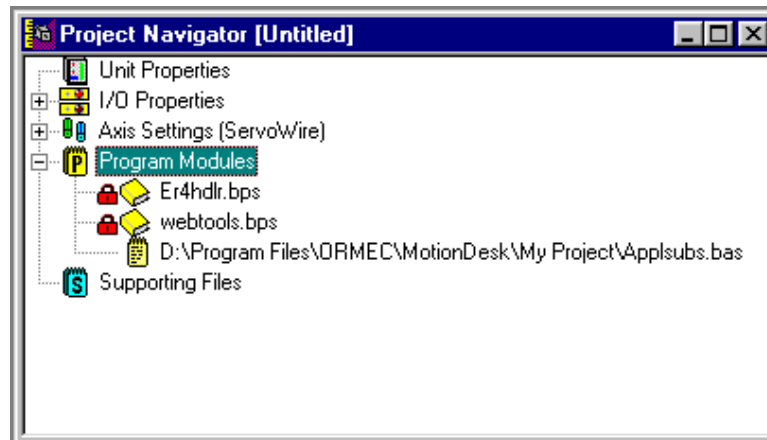
The extension, BPS (Binary Protected Source), means that the supplied files are encrypted. This encryption was performed to avoid the possibility of inadvertent editing of the supplied files. It is not necessary to edit the files, WEBTOOLS and ER4HDLR.

Adding Program Modules

You will need to add new versions of specific program modules to implement the Web Tools features. The program modules are:

- \ORMEC\MotionDesk\Templates\applsups.bas
(standard location of original. Make a copy and relocate it for your project, as it will be necessary to modify this file for each project that uses Tool features.)
- \ORMEC\MotionDesk\Library\webtools.bps (encrypted)
- \ORMEC\MotionDesk\Library\Er4hdr.bps (encrypted)

Each of the above modules would be added to your project as shown below.



Modifying Program Modules

To support the use of the Web Tool feature addition to MB Tools, application subroutines were added to the program module APPLSUBS.BAS. User modifications to these supplied routines may be necessary.

ApplHttpPref: Mandatory subroutine that must be present as it is called by the web tools initialization code. This routine is supplied so that the user can supercede default values for: Orion Identity, Device numbers used for open ports and files, IP address and port number for the web tool server, and rank of the web server thread. This routine is also where the user may turn on as many as three flag variables to enable the additional features of: User Configured Pages, Remote Data Entry, and Custom Pages. If these flags remain unused, (commented and thus not set on) then this is the only application subroutine that is relevant to the standard web tools feature set.

```

'----- Module: APPLSUBS.BAS -----
'Routine name:  ApplHttpPref
'
'Abstract: This subroutine is called during Web Tools initialization.
'         This is provided so that the user can turn on Web Tool Features,
'         and establish non-default settings for device numbers
'         & processing parameters
'-----
ApplHttpPref:
'http.identity$="Unidentified Orion Unit" 'User text to identify the Orion

'http.user.flag=ON      'Set to ON if you wish to use configured user pages
'http.data.entry.flag=ON 'Set to ON to allow remote entry of data from user pages
'http.custom.flag=ON   'Set to ON IF you wish to support custom pages

'http.sd=14            'Valid range 1 to 14, must be unique and not conflict w/ user device #'s
'http.dir.dev=4        'Valid range 1 to 4, must be unique and not conflict w/ user device #'s
'http.server.ip.addr$="200.200.200.200" 'IP address octets, Default:Orion Development Port
'http.server.port$="7500" 'Socket port, Default:7500, recommend > 6000, range 1 - 65535
'http.server.thrd.rank=1000 'Valid range 100 - 32767, default: 1000
RETURN

```

As supplied, the above routine consists only of comments, and therefore really does nothing more than document the default settings for variables used by the webtools initialization code.

The user should uncomment specific lines within this routine to take advantage of the available feature. (To uncomment a line, delete the leading single quote.) For example, each standard page has in its heading information identifying the Orion Unit. The identity string variable, **http.identity\$** defaults to "Unidentified Orion Unit". It is expected that every user will want to have a descriptive and unique identity for each of their Orion Units so they would un-comment the first line in the above routine and replace the default text string with the desired identifying description.

The variables **http.server.ip.addr\$** and **http.server.port\$** are used for the Internet Protocol address and port number at which the web tools server is established. It is expected that many users will wish to obtain information from their Orion via the serial development port whose address is 200.200.200.200, however others may wish to obtain this information from other means such as an ethernet card installed in the Orion. Those wishing to use an Ethernet card would uncomment the address string variable setting statement and replace the default IP address string, "200.200.200.200", with the IP address that they had established for their ethernet card. (e.g. "192.168.18.1") the socket number is also a number used when establishing the server address, and is fairly arbitrary in the range 1 to 65535 with the recommendation that a number above 6000 be

used, 7500 being the default. To change, uncomment the line and replace "7500" with the desired socket number.

The variable, **http.server.thrd.rank**, is used to establish the rank of the server thread. It is expected that user program threads will be ranked at or near the default of 100 and thus the default server thread rank was chosen to be 1000, or ten times slower than user program thread ranks. To adjust, uncomment the line and replace 1000 with the desired rank value for the web tool server thread.

The variable **http.sd** is used to store the device number of the server thread. It is anticipated that the user has not already used the default device #14, however if the user has, the web tool server thread can be reassigned to a different device number from this routine. Uncomment and change the value. Valid range is device #'s 1 - 14.

The variable **http.dir.dev** is used to store the device number used to read the system card directory information. It is anticipated that the user has not already used the default device #4, however if the user has, the value can be reassigned to a different device number in this routine. Valid settings are 1, 2, 3, or 4. (Must not use the same device number as http.sd above)

OPTIONAL USER CONFIGURED PAGES

The flag variable, **http.user.flag** in ApplHttpPref subroutine, is used to switch ON the addition of User Configured Web Tool Pages. Uncommenting this line such that the flag variable is set to ON will require the user to perform configuration work that defines their user pages.

ApplHttpUserLinks: This label tags the location where the web server expects data that defines overhead for all User Configured Pages. The overhead data required for each User Page is minimal as it only specifies: 1) the page number, 2) the page link, and 3) the page title. This data section concludes with DATA -1, which defines the end of the overhead data bank. This is not a subroutine, but rather a location of data. There is no limitation on the number of User Pages except the page number must fit within an integer variable. The following shows the supplied example data section.

```

|----- Module: APPLSUBS.BAS -----
| Label (DATA Locator):  ApplHttpUserLinks
|
| Abstract: This is the location where users define their pages and links.
|           Three data items:
|           Page number(integer), Link text(string), Page Title(String)
|           DATA -1 (must be included to indicate end of user DATA)
|-----
ApplHttpUserLinks:
|   pg#, link,                page title
|
|----- Beginning of Example Data -----
|   pg#, link,                page title
| data 1, "USR1LINK",        "User Page 1 Title"
| data 2, "USR2LINK",        "User Page 2 Title"
|
| data 5, "USR5LINK",        "User Page 5 Title"
|-----End of Example Data -----
|
| data -1 'signifies the end of the link/page definitions
| STOP

```

ApplHttpUserData: This label tags the location where the web server expects data that defines the contents for all User Configured Pages. The user is defining what information is to be presented on a particular page. A user page can be empty, however it is expected that the user wished to display something on the page beside the supplied header. The user can specify particular text and specific variable information be placed upon a user page. To accomplish this the user provides data detailing each item including target page number and the type of item to be presented. Then depending upon the type of item specified, variable field or text string, the user provides additional details. In the case of a text string, the user provides the text, and a number which serves to format the text. In the case of a variable field, the user specifies a text descriptor for the field, and then a number specifying the map reference of the variable to be displayed. This data section concludes with DATA -1, which defines the end of the this data bank. This is not a subroutine, but rather a location of data.

There are also two types of variable data fields that can be displayed, a read only variable field, and a data entry variable field from which the user could actually write numeric data to the Orion. The data entry variable fields are displayed exactly like the read only fields when the flag variable, **http.data.entry.flag** is set to OFF, however when the flag is ON, the data entry field includes a data entry window and "Change" button to send entered numeric data to the Orion.

The following figures show first an example User Page, and secondly the ApplHttpUserData that was used to implement this user display:

User Page 2 Title

Unit Identification: Unidentified Orion Unit, ip_address:port = 200.200.200.200:7500
 Unit Time & Date: 14:00:25 09-08-1999

[\[HOME\]](#) [\[SYS.INFO\]](#) [\[SYS.FAULTS\]](#) [\[SYS.I/O\]](#) [\[AXIS.STATUS\]](#) [\[AXIS.CONFIG\]](#) [\[PROGRAM.INFO\]](#) [\[HISTORY\]](#)
[\[USR1.LINK\]](#) [\[USR2.LINK\]](#) [\[USR5.LINK\]](#)

	<u>Formatted User Explanatory Text</u>	
Read Only Display of Map Ref Var 200	2345	
Conditional Entry of Map Ref Var 200	2345	<input style="width: 40px; height: 15px;" type="text"/> <input style="border: none; background-color: #cccccc; padding: 2px 5px;" type="button" value="Change"/>

```

'----- Module: APPLSUBS.BAS -----
' Label (DATA Locator):  ApplHttpUserData
'
' Abstract: This is the location where users define items on a user page.
'
' The format of the data statements is as follows:
'   data pg,type$,text$,modifier
'
' Valid values for pg (an integer constant):
'   includes the same values for user page numbers as defined under ApplHttpUserlinks:
'
' Valid values for type$ (a string constant):
'   "display" - a read-only display of a mapped variable
'   "entry"   - display the current value of a mapped variable and provide an option
'               to submit a new value
'   "string"  - display the string in the text$ field
'
' Valid values for text$ (a string constant):
'   "any user string whatsoever"
'
' Valid values for modifier
'   if type$="display" or "entry" then value can be any valid MAP reference number
'   if type$="string" then the modifier formats the string. Valid values are 0-7, with:
'     0 = left
'     1 = Center
'     2 = Bold
'     3 = Center & Bold
'     4 = Underline
'     5 = Center & Underline
'     6 = Bold & Underline
'     7 = Center, Bold & Underline
'
'     DATA -1 (must be included to indicate end of user DATA)
'-----
ApplHttpUserData:
'   pg$, "Type$ ", "Accompanying Text for the item",      format or map$
'
'-----Beginning of Example Data -----
'   pg$, "Type$ ", "Accompanying Text for the item",      format or map$
data 1, "string", "Formatted User Explanatory Text",      4 '4 = Underline
data 1, "display", "Read Only Display of Map Ref Var 200",200 'map$ #
data 1, "entry",  "Conditional Entry of Map Ref Var 200",200 'map$ #
'
data 2, "string", "Formatted User Explanatory Text",      5 '5 = Center & Underline
data 2, "display", "Read Only Display of Map Ref Var 200",200 'map$ #
data 2, "entry",  "Conditional Entry of Map Ref Var 200",200 'map$ #
'
data 5, "string", "Formatted User Explanatory Text",      6 '6 = Bold & Underline
data 5, "display", "Read Only Display of Map Ref Var 200",200 'map$ #
data 5, "entry",  "Conditional Entry of Map Ref Var 200",200 'map$ #
'-----End of Example Data -----

data -1 'signifies the end of all user page display definitions
STOP

```

OPTIONAL CUSTOM PAGES

The flag variable, **http.custom.flag** in ApplHttpPref subroutine, is used to switch ON the addition of Custom Pages. Uncommenting this line such that the flag variable is set to ON will then require the user to perform programming work to implement custom pages. The implementer would need to have knowledge of HTML as the code needed will essentially consist of MotionBASIC PRINT # statements. These PRINT # statements will be printing a sequence of characters out from the Web Server Device port. The print statements should be sending characters which represent legal HTML code to the client Browser of the Orion Server. (In the code, PRINT # is abbreviated with ? #, an equivalent).

ApplHttpCustomHome: Subroutine called by the Orion Home page generation code when the user has turned on the Custom Page variable flag. This allows the user to add customized information and links on the home page. The user would be required to have knowledge of HTML since it is expected that the code needed here is simply print statements which literally send HTML code to the browser client. This routine is called after the home page header is presented, but prior to the standard home page body information, thus presenting the users custom information at the top of the body of the home page.

Example Code:

```

'----- Module: APPLSUBS.BAS -----
'Routine name:  ApplHttpCustomHome
'
'Abstract: This subroutine is called when http.custom.flag is set ON
'          It allows the user to add their own text, links, etc.
'          to the upper portion of the ORION Home page, just below
'          the standard header section.  Example code provided.
'-----
'
ApplHttpCustomHome:
'the text string "/custom" must precede any unique text transmitted by a custom link so that
'the server will recognize and pass execution back to the user at routine ApplHttpCustomCall

'-----Beginning of Example Code-----
? #http.sd, "<h3>Custom Application Pages</h3>";
? #http.sd, "<ul>"
? #http.sd, "<li><b><a href='CHR$(34)'/custom'CHR$(34)'>Custom 1 Link</a><br>"
? #http.sd, "</b><small>Custom Page 1 is supported with example code.</small><br>"
? #http.sd, "</li>"
? #http.sd, "<li><b><a href='CHR$(34)'/custom2'CHR$(34)'>Custom 2 Link</a><br>"
? #http.sd, "</b><small>No example code was provided for Custom 2.</small><br>"
? #http.sd, "</li>"
? #http.sd, "<li><b><a href='CHR$(34)'/custom3'CHR$(34)'>Custom 3 Link</a><br>"
? #http.sd, "</b><small>No example code was provided for Custom 3.</small><br>"
? #http.sd, "</li>"
? #http.sd, "</ul>"
'-----End of Example Code-----

RETURN

```

Resultant "Customized" Home Page:

Orion Home Page

Unit Identification: Unidentified Orion Unit, ip_address:port = 200.200.200.200:7500
 Unit Time & Date: 14:38:03 09-08-1999

[\[HOME\]](#) [\[SYS INFO\]](#) [\[SYS FAULTS\]](#) [\[SYS I/O\]](#) [\[AXIS STATUS\]](#) [\[AXIS CONFIG\]](#) [\[PROGRAM INFO\]](#) [\[HISTORY\]](#)
[\[USR1LINK\]](#) [\[USR2LINK\]](#) [\[USR5LINK\]](#)

Custom Application Pages

- [Custom 1 Link](#)
 Custom Page 1 is supported with example code.
- [Custom 2 Link](#)
 No example code was provided for Custom 2.
- [Custom 3 Link](#)
 No example code was provided for Custom 3.

User Configured Pages

- **User Configured Links & Displays**
 Displays of User Text and User Mapped Reference Values. Page access is via the second row of user links in the above header.
 NOTE: The ability to write to specific Map Values is currently turned ON.

Standard Maintenance Tool Pages

In the example above, the link items presented in the Custom Application Pages section of the Home Page are designed to send specific text when a user clicks upon a link. The custom links above will respectively send the text: "/custom1", "/custom2", and "/custom3". The web server thread checks for text relevant to the standard pages, and having not found any standard page request text, checks for the presence of the text "/custom". Whenever this is found, the execution is passed back to the user at the routine, ApplHttpCustomCall.

ApplHttpCustomCall: Subroutine called by the webserver when an incoming link has sent within it the text substring "/custom". This means that the webserver thread will take no further action, and it is now up to the user to decide what is to be sent out to the browser client. The incoming request information is contained within the string variable, **http.data.block\$**, and it is expected that the user will parse this to decide upon a course of action to take. When this routine ultimately executes its RETURN, execution is passed back to the standard Web Server loop.

It is expected that the user will call their own routine from this application subroutine. Their routine would then proceed to print the desired HTML code out of the server socket via:

```
PRINT #http.sd, " custom html code "
```

or the equivalent:

```
? #http.sd, " custom html code "
```

```

----- Module: APPLSUBS.BAS -----
|Routine name:  ApplHttpCustomCall
|
|
|Abstract: This subroutine is called when a link has been selected
|          and the text, "/custom" is found within the link text sent.
|          It is up to the user to further parse the link text data
|          and print an appropriate web page via HTML code.
|          Example Code is provided.
|-----
|
|
ApplHttpCustomCall:
|Parse the link data received and decide how to respond.
|The link that got us here sent characters now stored in http.data.block%
|We know that at least "/custom" is in http.data.block% for the server to
|have called this routine
|
|-----Beginning of Example Custom Call Code-----
IF INSTR(http.data.block%,"/custom1")THEN
  GOSUB CUSTOM1.LABEL 'an example routine that prints a page
ELSEIF INSTR(http.data.block%,"/custom2") THEN
  GOSUB CUSTOM2.LABEL 'Not Provided
ELSEIF INSTR(http.data.block%,"/custom3") THEN
  GOSUB CUSTOM3.LABEL 'Not Provided
ELSE
  GOSUB http.objection.page 'Provided! Prints the Objection Web Page
ENDIF
|-----End of Example Custom Call Code-----

RETURN

```

```

-----Beginning of Example Custom Page Code-----
----- Module: APPLSUBS.BAS -----
|Routine name:  custom1.label
|
|
|Abstract: This subroutine is provided as an example of a custom web page.
|          Example Code is provided.
|-----
|
|
custom1.label:
? #http.sd,"<HTML>";
? #http.sd,"<HEAD>";
? #http.sd,"<meta http-equiv="CHR$(34)"Content-Type"CHR$(34)";
? #http.sd," content="CHR$(34)"text/html; charset=iso-8859-1"CHR$(34)">";
? #http.sd,"<TITLE>Custom Page 1 Title</TITLE></HEAD>";
? #http.sd,"<body bgcolor="CHR$(34)"#dddddd"CHR$(34)">";
? #http.sd,"<h2>Custom Page 1 Header</h2>";

http.links 'A useful web tool routine that prints identity information,
           'date, time, standard links, user links, and a horizontal line.

? #http.sd,"<ul>"
? #http.sd,"<li><b><a href="CHR$(34)"/>custom2"CHR$(34)">Custom 2 Link</a><br>"
? #http.sd,"</b><small>No example code was provided for Custom 2.</small><br>"
? #http.sd,"</li>"
? #http.sd,"<li><b><a href="CHR$(34)"/>custom3"CHR$(34)">Custom 3 Link</a><br>"
? #http.sd,"</b><small>No example code was provided for Custom 3.</small><br>"
? #http.sd,"</li>"
? #http.sd,"</ul>"

? #http.sd,"</body>"
? #http.sd,"</HTML>"
RETURN
|-----End of Example Custom Page Code-----

```

Resultant Custom Page:

Custom Page 1 Header

Unit Identification: Unidentified Orion Unit, ip_address:port = 200.200.200.200:7500
Unit Time & Date: 10:22:56 09-09-1999

[\[HOME\]](#) [\[SYS.INFO\]](#) [\[SYS.FAULTS\]](#) [\[SYS.I/O\]](#) [\[AXIS.STATUS\]](#) [\[AXIS.CONFIG\]](#) [\[PROGRAM.INFO\]](#) [\[HISTORY\]](#)
[\[USR1.LINK\]](#) [\[USR2.LINK\]](#) [\[USR5.LINK\]](#)

- [Custom 2 Link](#)
No example code was provided for Custom 2.
- [Custom 3 Link](#)
No example code was provided for Custom 3.

The above custom page example is purposely an uncomplicated example. The user should restrict their page design to fairly simple HTML constructs: Text, linked lists, tables. Graphics are to be avoided.

Appendix A5

Tools Initialization Summary

Tools Initialization Summary

A summary of useful and/or required statements to run the MB Tools.

Initialization Code

The User's Application Program will setup flags, variables, and perform subroutine calls to enable the operation of the tool features. The relevant statements are documented below. Optional Statements are shown in brackets. Statements considered **essential** are **bolded**.

'The following statements are relevant to the setup of the tools error handler.

'These would be appropriately located as part of the User's Program Initialization

[gDebug = TRUE]	'flag to turn off tools error handling. Only useful during debugging.
[gNoMap = TRUE]	'flag to turn off Quick Panel mappings. With no QuickPanel, use it!
[gSize = 11]	'sets error log memory queue size. Default: 10 errors (when omitted)
[gLogFile\$ = "E:\logfile.bin"]	'name for a PC Card resident error logfile. Default: none (no file).
[gLogFileSize = 26]	'sets size of PC Card resident error logfile. Default: 25 errors.
ErrInit	'subroutine call to initialize the Tools Error Handler.

'The following statements are relevant to the tools support of a QuickPanel Operator Display

'They are indicated as optional, [], since they are only **essential** when a QuickPanel is used

'These statements also require the installation of a QP MBX (QuickPanel MotionBASIC Extension)

[QP.CLOSE]	'close QuickPanel
[QP.OPEN 1 , "SRL1:"]	'open QuickPanel Device, "SRL2:", is also valid, and typical
[QP@ = ON]	'turn ON QuickPanel Communications
[QP.DNLD "cfgfilename"]	'load the QP with a configuration file (supporting file on PC Card)

'The following statements require the additional program code of the Library Module: QP4Tools.bps

[QPDogTime = 1000]	'setup ms timeout for watchdog feature of Library Module: QP4Tools.bps
[QPWatchDogID = START(QPWatchdog)]	'startup QP Watchdog Thread from QP4Tools.bps

'This next statement is relevant to Web Tools. Only **essential** if web tool features are desired.

Http.init	'subroutine call to initialize and start the Web Tools Server
------------------	---

' The following statement needs to be located at the point in the user's program where error trapping is to be armed and enabled. *This statement should be located after all initialization tasks.*

ON ERROR GOTO Errhdlr	'arm the Tools Error Handler to trap errors.
------------------------------	--

Advantages of the Tools Error Handler

The MB Tools error handler which is initialized with the subroutine call, **ErrInit**, and established as the Error Handler thread with the statement, **ON ERROR GOTO ErrHdlr**, is required for:

- 1) Error / Event Logging and thus
 - QuickPanel Alarm/Log Display
 - Console Window Alarm/Log Display
 - Web Tools History Page
- 2) Error /Event Logging Thread recovery
- 3) Web Tools Server Thread recovery

Running Web Tools without the Tools Error Handler

*It is possible to run the Web Tool Features without the Tools supplied error handler, however there will be no History Page feature, and there will be no server thread recovery code. **Web Tool Server Thread Recovery functionality must be added to the users error handler!!***

The user who wishes to add web tools to their application and continue to use their own error handler would be required to add to their error handler the following web tool server thread recovery block:

```
IF ASYNC.ERR@ = FALSE AND ERT =http.server.thrd.id THEN
  STOP THREAD http.server.thrd.id
  CONT THREAD http.server.thrd.id, http.server.thread
  RESUME
ENDIF
```

The code above establishes that whenever the web tools thread experiences a synchronous error, the thread is reset. The web tool server thread will experience synchronous errors if communication collisions occur at the server. These collisions are caused when the browser client is sending new requests while the server is still responding to old requests.

The above functionality is supplied with ER4HDLR.BPS (rev 6) and only needs to be explicitly added to application code when the user has elected to not utilize the Tools Error Handler.

Appendix A6

Web Tools Communication

Web Tools Communications

This purpose of this appendix is to aid the user in establishing communications to an Orion running the web tool server.

Browser Connection

The users web browser may require configuration of its connection properties depending upon how it is planned to connect to the Orion web tool server from this browser. For example it may be appropriate to setup the browser to connect to the internet "via a modem" (that being the Orion Serial Connection) when attempting communication across the direct serial cable.

There commonly are also means in web browsers to bypass normal connection methodologies for specific IP Addresses. As the variations in networks and browser package versions are numerous, it is recommended to refer to your network administrator to assist in the browser setup if difficulties arise in directing messages at the networked Orion unit.

From the configured web browser, the address one must enter to communicate to the Orion web tool server is of the form:

http://ipaddress:portnumber/

Where:

ipaddress the actual IP Address established within ApplHttpPref routine (or an alias established in the computers hosts file)

portnumber the actual port number established within ApplHttpPref routine. (Port 80 is the default used by most web browsers)

Using the tools default IP Address of 200.200.200.200 and default port number of 7500, the proper address entered at the browser would be:

http://200.200.200.200:7500/

When the Orion web tool server receives this request it responds with the Orion Home Page.

Example: The user has implemented an Ethernet network, and has established the network card in the Orion to have IP address 192.168.18.1 (as directed by their network administrator). Within their application subroutine, ApplHttpPref, the user established:

```
http.server.ip.addr$="192.168.18.1"
```

```
http.server.port$="7500"
```

The proper address entered at the browser to address the Orion web tool server at the Ethernet Card would be:

```
http://192.168.18.1:7500/
```

It is still possible to connect to the web tool server with the above address via a direct serial cable connection at the serial port, and in that case the proper address reflects the IP address of the Orion Development Port:

```
http://200.200.200.200:7500/
```

It is a feature of TCP/IP which allows connections to one server opened at a port with multiple IP addresses.

Using Computers without MotionDesk Software

After Ormec's MotionDESK software is installed on a computer, the user must perform various network setup tasks before that computer can successfully dial-up network to the Orion development port via a serial cable. On those systems, a web browser can also take advantage of that dial-up network connection and use it to communicate to the Orion over the serial cable.

Computers which are not intended to have MotionDesk installed on them, but would benefit from the information provided by the web tool server, could be setup to communicate over a dial-up network connection via a serial cable. A help file and a modem information file would need to be transferred to the target computer in order to set it up for this dial-up networking connection.

The modem information file can be found on a computer where MotionDesk has been installed. Its normal location and name is:

\Program Files\Ormec\MotionDesk\mdmormec.inf

Copy this file and transfer it to the target computer to be setup.

The Network Setup help file required is one of two possible files, depending upon the operating system. The Network Setup help file can be found on a computer where MotionDesk has been installed. If the operating system is Windows 95, then the normal location and name of the file is:

\Program Files\Ormec\MotionDesk\netset95.hlp

If the operating system is Windows NT, then the normal location and name of the file is:

\Program Files\Ormec\MotionDesk\netsetnt.hlp

Copy the appropriate file and transfer it to the target computer to be setup.

Once this is accomplished it remains to perform the setup on the target computer. This is done by starting up the help file which has detailed, step-by-step instructions. The result of this exercise will be a valid dial-up networking connection which will support communications to an Orion via a serial cable to its development port.