33 South La Patera Lane
Santa Barbara, CA 93117-3214
ph (805) 681-3300
fax (805) 681-3311
tech@motioneng.com
www.motioneng.com

Release Note MPI/XMP Firmware and Library

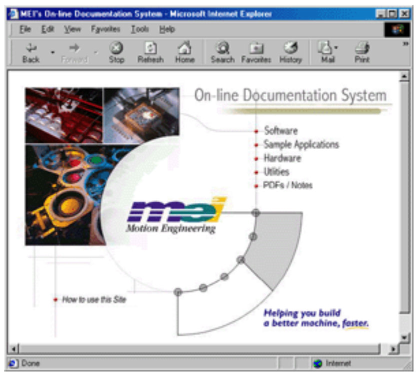
XMP Firmware Version 498A1 MPI Library Version 20020117.1.12 Revised 25Feb2004

DCR 747

1 Introduction

Welcome to the latest release of Motion Engineering's MPI/XMP Firmware and Motion Programming Interface Library. This distribution has been prepared for Windows® NT 4.0/2000/XP. The distribution was built using Visual C++ v4.2 and tested using Visual C++ v6.0. This document provides an overview of the release, and describes the new features, changes and bug fixes between the following versions:

	Previous Version	New Version
Firmware	492A1	498A1
MPI Library	20020117.1.10	20020117.1.12


MEI's On-line Documentation System - Microsoft Internet Explorer
On-line Documentation System
Software
Sample Applications
Hardware
Utilities
PDFs / Notes
Motion Engineering
How to use this Site
Helping you build a better machine, faster.

http://support1.motioneng.com

INTRODUCING MEI'S NEW ON-LINE DOCUMENTATION SYSTEM

Featuring:

- Up-to-Date Documentation
- Dynamic Hyperlinks
- Complete Search Functionality
- Sample Code that you can copy and paste
- Print-friendly PDFs

Guaranteed to help you find the **right** information...***faster!***

This document contains proprietary and confidential information of Motion Engineering, Inc., and is protected by federal copyright law. The contents of the document may not be disclosed to third parties, translated, copied, or duplicated in any form, in whole of in part, without the express written permission of Motion Engineering, Inc.

The XMP-PULSE is now supported by the MPI Library

The XMP-PULSE-PCI motion controller is designed as a cost-effective, XMP-series motion controller specifically designed for pulse (step/direction) control drives having up to 32 axes. For a detailed explanation of its specifications and features, please see ***Application Note 218, Rev. A***. For further information, please contact **Motion Engineering, Inc.**

1.1 System Requirements

1.1.1 Operating System

The MPI release is built to operate on Windows® NT 4.0/2000/XP.

1.1.2 Visual C++ DLLs

The MPI is built using Microsoft Visual C++ 4.2. No Microsoft Visual C++ DLLs are included with this release. These DLLs are required in order to run programs built with the MPI. These DLLs work with Microsoft Visual C++ 4.2 - 6.

1.2 Installing the Distribution

WARNING! You must reboot your system!
If you have not used a InstallShield for Windows Installer program before, the MEI Install Shield will need to install InstallShield installer files before actually installing the MDK. You will have to **reboot** your system after these files are installed. Please shut down all programs before running the InstallShield for the first time. For first-time installations of an MEI controller and accompanying software, please see the *QuickStart Guide* located on the distribution CD.

WARNING! If you are upgrading from a previous MPI/XMP software release, **you will need to remove or archive all previous releases**. This will prevent any conflicts between old and new files. To remove the previous MPI/XMP software release, select **Start -> Control Panel -> Add/Remove Programs**. Select the **MPI/XMP Development Toolkit** entry and click on the **Add/Remove** button.
NOTE: The MPI/XMP software release can also be removed by running the MDK InstallShield and choosing the remove option.
For more information on installing the MPI/XMP software release, please see the *QuickStart Guide*.

The MPI/XMP distribution comes in two parts. The first part is an InstallShield distribution. Key components of the distribution are:

- device driver (meixmp.sys for WinNT, Win2000, WinXP)
- firmware
- MPI dynamic link library
- utilities
- sample applications

To install the MPI/XMP software release, insert the MDK CD-ROM. The set-up InstallShield will be launched automatically. Follow the InstallShield instructions. The InstallShield will take care of installing the DLL and will also set the PATH environment variable to XMP\bin\WinNT for WinNT, Win2000, and WinXP under the default installation directory **(C:\MEI)**.

The second component of this distribution contains customer-specific applications and files. This is provided to you in a separate InstallShield. To install this custom component, click on the InstallShield (default C:\MEI) and follow the instructions.

By default, files will be located in the C:\MEI\Custom directory. To install into an alternate directory, select the custom option during the installation process and change the default directory to the one you prefer.

2 General Changes

This section lists changes since the 20000913 production release, beginning with the most recent.

2.1 PTF and PVTF motion type improvements — MPI 1286

The motion types MPIMotionTypePTF and MPIMotionTypePVTF, support user-specified feedforward values for each point. The following improvements have been made to the PTF and PVTF motion types.

1) The feedforward values are interpolated linearly over the PT or PVT time intervals. The feedfoward values correspond to the P or PV values. (i.e. When the motion reaches a specified position (PTF) or position and velocity (PVTF), the interpolated feedforward value will be equal to what is specified in the motion parameters.)
2) The feedforward values are not set to 0 at the beginning of the move; they retain the last value that is specified in the PTF or PVTF motion parameters.
3) The feedforward values are not changed by non-PTF or PVTF moves. Previous versions of the firmware would set the feedforward value to zero for any move that was not a PTF or PVTF move (i.e. PT, PVT, Spline, S-Curve, etc.).

2.2 PTF and PVTF motion types — MPI 1101

The following PTF and PVTF motion types have been added to the MPI library in motion.h.

```
typedefstruct MPIMotionPTF {
        long            pointCount;
        double          *position;
        double          *feedForward;
        double          *time;
        MPIMotionPoint  point;
} MPIMotionPTF;
```

pointCount - This value specifies the number of points.

position - This array stores the positions for the motion profile. There is one position value per point, per axis. The length of the array must be equal to pointCount multiplied by the number of axes. The positions are interleaved in the array by the axis index.

feedforward - This array stores the feedforward values for the motion profile. There is one feedforward value per point, per axis. The length of the array must be equal to pointCount multiplied by the number of axes. The feedforward values are interleaved in the array by the axis index. The units are raw DAC counts (range -32768 to +32767).

time - This array stores the times for the motion profile. There is one time value per point. The time specifies the number of seconds between the specified position, and the previous position (point). The length of the time array must be equal to pointCount.

point - This structure contains the point configuration. Please see MPIMotionPoint data type for more information.

```
typedefstruct MPIMotionPVTF {
        long              pointCount;
        double            *position;
        double            *velocity;
        double            *feedForward;
        double            *time;
        MPIMotionPoint  point;
} MPIMotionPVTF;
```

pointCount - This value specifies the number of points.

position - This array stores the positions for the motion profile. There is one position value per point, per axis. The length of the array must be equal to pointCount multiplied by the number of axes. The positions are interleaved in the array by the axis index.

velocity - This array stores the times for the motion profile. There is one time value per point. The time specifies the number of seconds between the specified position, and the next position (point). The length of the time array must be equal to pointCount.

feedForward - This array stores the feedforward values for the motion profile. There is one feedforward value per point, per axis. The length of the array must be equal to pointCount multiplied by the number of axes. The feedforward values are interleaved in the array by the axis index. The units are raw DAC counts (range -32768 to +32767).

time - This array stores the times for the motion profile. There is one time value per point. The time specifies the number of seconds between the specified position, and the previous position (point). The length of the time array must be equal to pointCount.

point - This structure contains the point configuration. Please see MPIMotionPoint data type for more information.

The **MPIMotionParams** structure was also modified.

```
typedefstruct MPIMotionParams {
            MPIMotionJog              jog;

            MPIMotionPT               pt;
            MPIMotionPTF              ptf;
            MPIMotionPVT              pvt;
            MPIMotionPVTF             pvtf;
            MPIMotionSPLINE           spline;
            MPIMotionBESSEL           bessel;
            MPIMotionBSPLINE          bspline;

            MPIMotionSCurve           sCurve;
            MPIMotionSCurve           sCurveJerk;
            MPIMotionTrapezoidal      trapezoidal;

            MPIMotionVelocity         velocity;
            MPIMotionVelocity         velocityJerk;

            MPIMotionAttributes       attributes;
            void                      *external;
} MPIMotionParams;
```

2.3 MPI Methods Return *Const* Handles MPI 1054

In previous releases, there were many functions that returned a *const* handle, such as...
mpiMotionCreate(...), mpiMotionAxis(...), mpiMotionControl(...)

If applications declared handles as *const*, it is likely that they would run into compilation errors. Returning *const* handles serves no purpose and actually causes problems with some tools. The *const* was removed from all MPI methods in version 20020117.1.8, 20030120 and later versions.

2.4 Amp Enable Enhancements MPI 1045

In versions 20020117.1.8, 20030120, and later, some safety features were added to the Amp Enable logic. By default, when the Amp is disabled, the controller will:

1) Disable the servo loop output (except for the offset).
2) Set the command position equal to the actual position every sample.
3) Clear the integrator error.

When the Amp is enabled, the controller will operate the servo loop normally. This feature can be disabled or enabled by calling mpiMotorConfigSet(...) with the disableAction element in the MEIMotorConfig structure set to either MEIMotorDisableActionNONE or MEIMotorDisableActionCMD_EQ_ACT.

WARNING!:
Setting the command position equal to the actual position for a stepper can cause unintended results. As a safety precaution, restrictions are placed against setting a stepper motor's amp enable mode to MEIMotorDisableActionCMD_EQ_ACT and against setting a motor to stepper mode when an amp enable is already set to MEIMotorDisableActionCMD_EQ_ACT.

Setting the amp enable to MEIMotorDisableActionCMD_EQ_ACT on a motor configured for steppers will return an error "Motor: cannot set motor type to STEPPER when disable action is CMD_EQ_ACT." Setting a motor to stepper mode with an amp enable already set to MEIMotorDisableActionCMD_EQ_ACT will return "Motor: cannot set disable action to CMD_EQ_ACT when motor type is STEPPER."

2.5 Motion Method Check for Axis ObjectMPI 1037

In previous versions, if a motion method was called without an axis associated with the motion object, the controller would timeout and a TIMEOUT error would be returned. An axis check has been added to the motion methods in versions 20020117.1.8, 20030120, and later. If a motion method is called without an axis associated with the motion object, a NO_AXES_MAPPED error will be returned.

2.6 Filter Object DRate Check MPI 703

The Filter object's DRate (derivative sub-sampling rate) is limited to a range from 0 to 7. Values greater than 7 are not valid. A check was added into the 20020117.1.8, 20030120 and later releases. If the DRate value is out of range, an INVALID_DRATE error message will be returned.

2.7 Addition of Multiple Injection Point Noise Source to xmp firmware MPI 899

In firmware version "371A5," a noise source input was added. This feature can be used to inject noise into the control loop of a specific axis to generate a Bode plot of the physical system. This functionality can be accessed using MEI's Bode Tool Software versions 01.02.01 and later.

2.8 Multi-Point Motion Buffering Improvements MPI 889

Changes have been made to the 20020117.1.3 release to streamline the loading of points lists. Now, only the initial point list is loaded by the host. Now, Motion Modifies only append to the host's point list and does not initiate a load. The eventMgr service thread (as supplied by the service module of apputil) performs all subsequent point list loads upon internal "frame low" events. Frame low events are now generated every sample that the controller has less points than the frame low limit. These changes were implemented in version 20020117.1.4.

2.9 New On-Line Documentation System

MEI would like to introduce its new On-Line Documentation System. This html-based documentation system will dramatically improve your ability to navigate through the technical documentation and find the information that you need. The web site will be accessible and functional through the Internet (http://support.motioneng.com) and through our Install Shield CD-ROM, which will allow any computer to use the web site without a network connection. There are several new features including: conditional pulldown menu bars, keyword search functionality, new sample applications, a PDF/Notes section, and dynamic hyperlinking throughout the site. To learn more about the features of this site and how you can best optimize your time and effort, simply click on the "How to use this Site" link which is on the home page and take a minute to read the page. This section will help you understand how the site is structured and how you can fully take advantage of this new documentation system.

2.10 New Default XMP-Series Controller Configuration MPI 667

The XMP-Series controller's Flash memory is now pre-loaded at the factory with base firmware. This firmware contains a minimal amount of code to boot the DSP and allows the MPI to identify the XMP-Series controller. To operate the controller you will need to download the binary code included in this MPI software distribution. Please see the sections below for a complete description about software binary management and instructions to download firmware to your controller.

Introduction

System designers need to give careful consideration to the software configuration management procedures for their machines. To ensure machine consistency and quality, a process for software installation, configuration, version control, and verification must be implemented. Addressing this issue early in the development cycle will significantly reduce confusion and mistakes. Failure to implement some basic procedures can cause unknown machine configurations, costly field repairs/upgrades, mysterious intermittent problems, broken equipment, and possible injury.

The MPI and XMP-Series controllers contain several features to make configuration management easy. Please take the time to understand and implement these features before you begin development.

Software Components

The MEI software distribution contains several software components, which need to be loaded onto your machine and controller. The MPI DLL, header files, import libraries, device driver, utility programs, controller binaries, sample code, etc. These are all loaded onto your hard drive by the InstallShield distribution. Additionally, the controller contains on-board flash memory to store DSP code, FPGA code, and configuration information. You will also need to load the appropriate DSP (.bin) and FPGA (.fpg) code into your controller's flash memory. The code is loaded into the DSP and FPGA(s) during power-on or when the controller is reset.

Version Control

Each software component has its own version number. These components have been tested together at the factory for interoperability.

The software version numbers have the following format:

Motion Console	NN.NN.nn	Major, Medium, minor
Motion Scope	NN.NN.nn	Major, Medium, minor
MPI DLL	YYYYMMDD.b…r	Year, Month, Day, branch, … rev
XMP Firmware	NNNnn	Major, minor
FPGA Code	RRR	Revision

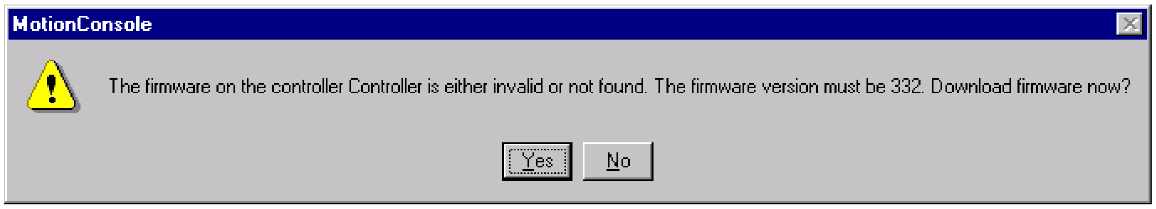
The software has automated compatibility checking. If there is a compatibility problem between software components an error code will be returned.

If an application (Motion Console and Motion Scope are applications too) and MPI DLL are NOT compatible, the error message "Control: application not compatible with MPI DLL" will be returned. To correct this problem, you can recompile your application with the appropriate MPI import library OR install the proper MPI DLL.

If the MPI DLL and firmware versions are NOT compatible, the error message "Control: firmware version mismatch" will be returned. The user or application must download the appropriate firmware to correct the problem. The DLL and firmware versions can be determined with the version.exe utility, Motion Console, or application code.

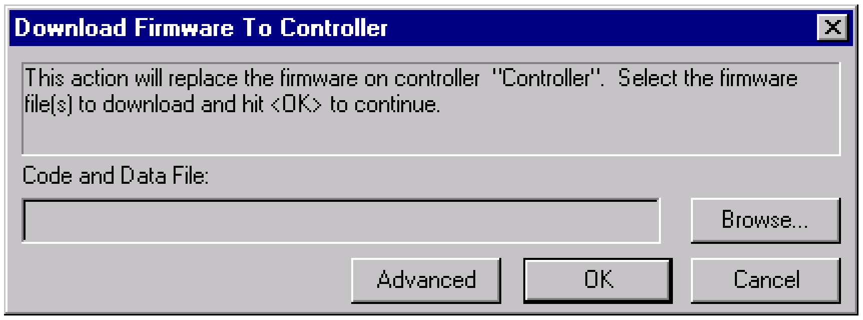
If the controller flash memory has NOT been configured, the error message "Control: no firmware found (factory default)" will be returned. To correct this problem, the user or application must download the appropriate XMP firmware and FPGA images. Firmware and FPGA images can be downloaded from Motion Console, flash.exe, or application code.

For example, if Motion Console detects that the controller is in the factory default configuration (no firmware) or if the MPI DLL is not compatible with the firmware, it will prompt the user to download firmware:


MotionConsole

The firmware on the controller Controller is either invalid or not found. The firmware version must be 332. Download firmware now?

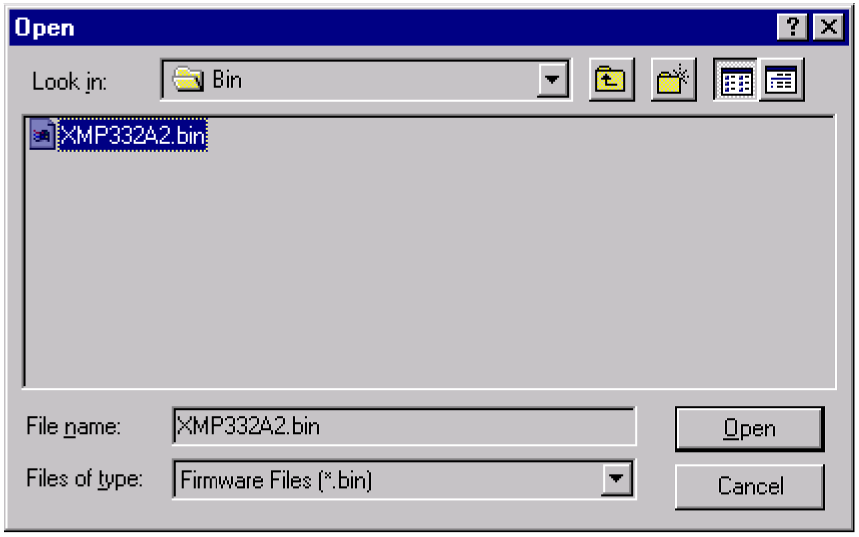
Click “Yes”. Motion Console will then prompt you for the firmware file:


Download Firmware To Controller

This action will replace the firmware on controller "Controller". Select the firmware file(s) to download and hit <OK> to continue.

Code and Data File:

Click “Browse…” and select the appropriate firmware file (.bin). The firmware is stored in the mei\xmp\bin directory (by default):


Open

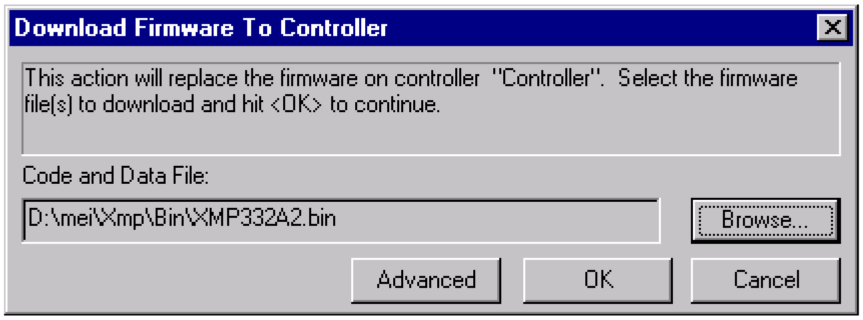
Look in: Bin

XMP332A2.bin

File name: XMP332A2.bin

Files of type: Firmware Files (*.bin)

Select the proper file and click “Open.”



Download Firmware To Controller

This action will replace the firmware on controller "Controller". Select the firmware file(s) to download and hit <OK> to continue.

Code and Data File: D:\mei\Xmp\Bin\XMP332A2.bin

Now download your firmware file by clicking “OK.”

Automated Software Configuration

During your machine development, several different versions of MPI DLLs and/or XMP firmware might be used. You may want to upgrade to new releases to take advantage of new features, MEI may provide custom features or bug patches for previous releases.

During your machine production, you will want to guarantee software configuration consistency. MEI recommends using the InstallShield release package, a third party installer program, or batch scripts to install your application code and MEI’s software onto your machine.

You will also want to guarantee that the controller’s flash memory configuration is consistent. To load the flash, you could automatically download firmware (.bin) and FPGA (.fpg) code during software installation OR during your application initialization. For example, the flash.exe program could be executed from an installer or batch script during software installation. Included in the XMP\Apps directory, is a sample program called “initFlsh.c” that demonstrates how to read the controller’s firmware/FPGA versions, check if they match the desired configuration, and download the correct versions (if needed).

Also, you can create your own custom firmware file by saving configurations to flash, and uploading the firmware file to your hard drive. The firmware contains a “userVersion” field, so you can keep track of your custom configured files. Using this technique, the firmware can be configured with Motion Console, uploaded to a file (myfirm.bin) and downloaded to future machines using Motion Console, flash.exe or your application.

Frequently Asked Questions

Why do I need to download firmware to my controller?
Only the machine developer knows which firmware version and configuration works with their application. By downloading firmware directly, you have complete control over your development and release versions.

Why can't MEI download firmware to my controller at the factory?
MEI can download your firmware, but it is expensive and causes several problems:

1) You would need a custom part number for each controller with a different firmware image. Even if you use the same controller hardware in several machines, each version would need to be ordered, purchased, tracked, and stocked separately.
2) Changing a firmware image (version or configuration) would require a new part number. This causes transition problems between "old" and "new" parts.
3) Repairs and replacements are much more complicated.
4) Field upgrades are not possible. Controllers must be returned to the factory to receive new firmware and a new part number.

What if I need to upgrade software in the field?
If you configure the flash memory as part of your application or installation, then it is very easy to upgrade software and/or firmware in the field. If you do not, then you'll need to manually update the flash memory.

Can I modify the FPGA (.fpg) files?
No. These files are binary and do not contain any configuration data.

2.11 New UserVersion in MPIControlConfig{...} MPI 538

A new element, UserVersion has been added to the MPIControlConfig{...} structure. This feature allows the user to mark a firmware image with a user-defineable version number. This was added in version 20010614.

2.12 Changed DAC level units to volts MPI 535

In version 20010622, the DAC level was changed from DAC units to volts. The MEIMotorDacStatus{...} structure contains the DAC level for cmd and aux DACs. meiMotorStatus(...) is used to read the cmd and aux DAC level from the controller.

2.13 Addition of Branch identification to Firmware/MPI version MPI704

A new field has been added to the XMP's firmware to identify and differentiate between intermediate branch software revisions. The branch value is represented as a hex number between 0x00000000 and 0xFFFFFFFF. Each digit represents an instance of a branch (0x1 to 0xF). A single digit represents a single branch from a specific version, two digits represent a branch of a branch, three digits represent a branch of a branch of a branch, etc.

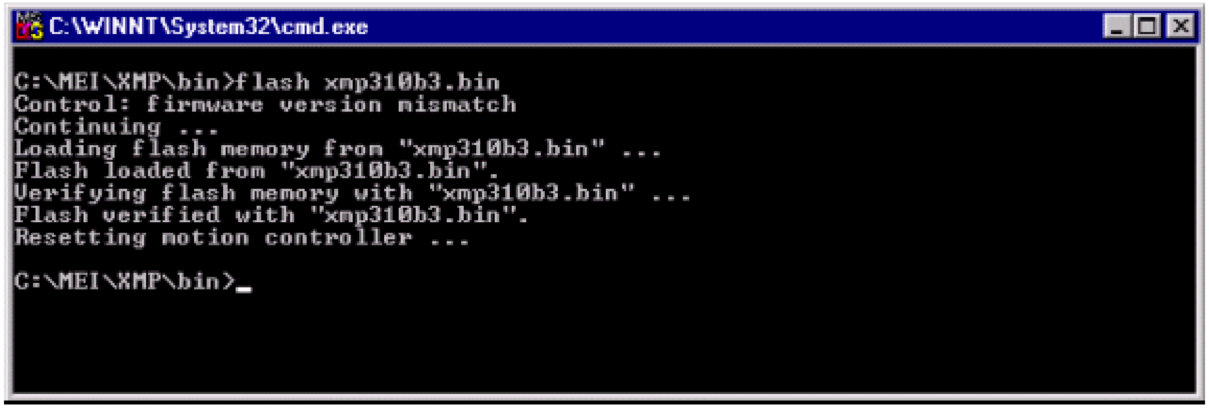
2.14 Path Motion works with all Interpolation Algorithms MPI660

In previous versions, path trajectory generation was only supported by bsplines. Path trajectory generation is now supported by PT, PVT, SPLINE, BESSEL, BSPLINE and BSPLINE2 algorithms. Blending of the corners is only available for the 2 bspline algorithms. Blending of a corner is when the path does not hit the corner but goes through a smooth arc.

2.15 Flash Utility Now Supports Flash from File Interface Changes MPI629

The XMP-Series controllers have on-board flash memory to store code and configuration data for the DSP, and fuse maps for the FPGAs. In previous versions, the DSP and FPGA code were stored in a single flash file (XMPxxxxx.bin). Now, the DSP code/data are stored in one file (XMPxxxxx.bin) and the FPGA code is stored in separate files (xxx_xxxx.fpg). This change was required in order to support several different XMP-Series controller platforms, each with different FPGAs, but all with the same flash memory component. This feature makes it possible to download custom FPGA files.

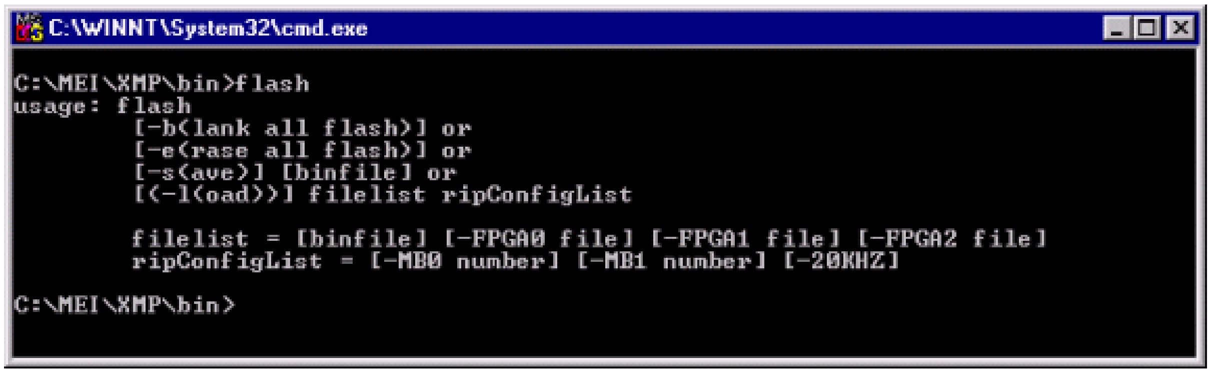
If your application downloads a Flash file (.bin), you will need to update your application code to download the DSP (.bin) and FPGA (.fpg) code. The flash.exe utility sources demonstrate how to implement this into your application. You must make sure the DSP (.bin) and ALL FPGA (.fpg) files are installed on the host computer for downloading. The FPGA and DSP files must be in the same directory or flash must be called with -FPGAx option, specifying the path to the FPGA file.
Here is the output from the **OLD** flash.exe operation.



```
C:\MEI\XMP\bin>flash xmp310b3.bin
Control: firmware version mismatch
Continuing ...
Loading flash memory from "xmp310b3.bin" ...
Flash loaded from "xmp310b3.bin".
Verifying flash memory with "xmp310b3.bin" ...
Flash verified with "xmp310b3.bin".
Resetting motion controller ...

C:\MEI\XMP\bin>_
```

Here is the **NEW** flash.exe command line options and operation. Notice, that flash.exe automatically determines the XMP hardware configuration and downloads the appropriate .fpg files:

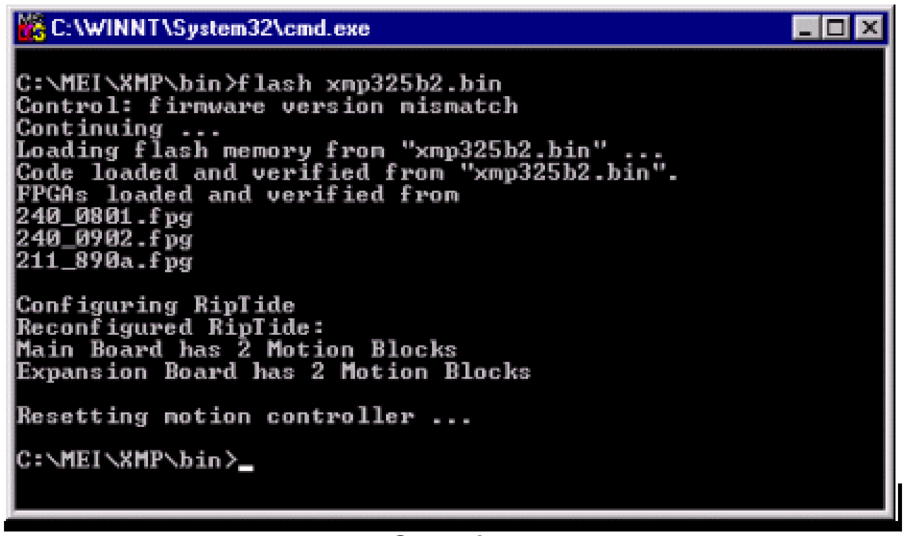


```
C:\MEI\XMP\bin>flash
usage: flash
        [-b(lank all flash)] or
        [-e(rase all flash)] or
        [-s(ave)] [binfile] or
        [(-l(oad))] filelist ripConfigList

        filelist = [binfile] [-FPGA0 file] [-FPGA1 file] [-FPGA2 file]
        ripConfigList = [-MB0 number] [-MB1 number] [-20KHZ]

C:\MEI\XMP\bin>
```

Options



```
C:\WINNT\System32\cmd.exe

C:\MEI\XMP\bin>flash xmp325b2.bin
Control: firmware version mismatch
Continuing ...
Loading flash memory from "xmp325b2.bin" ...
Code loaded and verified from "xmp325b2.bin".
FPGAs loaded and verified from
240_0801.fpg
240_0902.fpg
211_890a.fpg

Configuring RipTide
Reconfigured RipTide:
Main Board has 2 Motion Blocks
Expansion Board has 2 Motion Blocks

Resetting motion controller ...

C:\MEI\XMP\bin>_
```

Operation

2.16 Flash from File Interface Change MPI595

In previous versions, the DSP and FPGA code were stored in a single flash file (XMPxxxxx.bin). Now, the DSP code/data are stored in one file (XMPxxxxx.bin) and the FPGA code is stored in separate files (xxx_xxxx.fpg). This change was required to support several different XMP-Series controller platforms, each with different FPGAs, but all with the same flash memory component.

To support this change, the "Flash Memory" methods were changed in version 20010216. Here are the changes (from stdmei.h):

```
#define MEIFlashFileMaxNameChars(12)/* 8.3 format */
#define MEIFlashFileMaxChars(120)
#define MEIFlashFileMaxPathChars(MEIFlashFileMaxChars - MEIFlashFileMaxNameChars)

typedef struct MEIFlashFiles {
    char binFile[MEIFlashFileMaxChars];
    char FPGAFile[MEIXmpFlashMaxFPGAFiles][MEIFlashFileMaxChars];
} MEIFlashFiles;

MPI_DECL1 long MPI_DECL2
    meiFlashMemoryFromFileAndVerify(MEIFlash           flash,
                                    MEIFlashFiles      *filesIn,
                                    MEIFlashFiles      *filesOut);
```

You provide the filesIn with a valid binFile. If the FPGAFiles are NULL, then the MPI Library will automatically generate the default file names for the FPGAs from the hardware info on the board and the path from the binFile. Otherwise, you can specify FPGA files (up to 3 MEIXmpFlashMaxFPGAFiles), including the path. The file order in MEIFlashFiles.FPGAFile[...] is not important. If the file name is not correct or does not match the controller hardware, an error code is returned and the offending file name will be placed in the filesOut (if it is not NULL) structure.

If meiFlashMemoryFromFile(...) returns MPIMessageOK, then filesOut (if it was not NULL) will contain the files and paths of the bin and FPGA files that were loaded and verified. If an error is returned, then filesOut (if it was not NULL) will contain the bin and FPGA files that either were not found or could not be loaded (the error code tells you the result). Internally, the verify is required to update the table in the data portion of the flash image with valid pointers to the FPGA images.

In flash.h, meiFlashMemoryFromFile(...) has been extended to support both binFile and FPGAFile types. meiFlashVerify(...) was removed, but it's functionality was moved into meiFlashMemoryFromFileAndVerify(...).

2.17 Dac Object Removed MPI628

The Dac object has been removed. The Dac object features have been moved into the Motor object. The Dac numbering has been changed. New methods have been added to write to a Dac's offset and read a Dac's level. The MPIControlConfig structure has been modified to support the new Dac numbering. These changes were required to support new controller models and simplify the MPI.

In previous versions, each Dac object supported a single channel, which was numbered 0, 1, 2, … 31. The XMP controller can have up to two Dacs per axis. One channel is for standard servo control and the other channel is an auxiliary, OR both channels are used for sinusoidal commutation. The number of enabled Dacs (0 to 32) was configured by specifying the dacCount in the MPIControlConfig{…} structure.

The MEIMotorConfig{…} structure has been extended to include a new MEIMotorDacConfig{…} structure. Each MEIMotorDacConfig{…} structure contains a Cmd and an Aux MEIMotorDacChannelConfig{…} structure:

```
typedef struct MEIMotorDacChannelConfig {
        float                           Offset;         /* volts */
        float                           Scale;
        MEIXmpDACInputType              InputType;
        MEIXmpGenericValue              *Input;
} MEIMotorDacChannelConfig;

typedef struct MEIMotorDacConfig {
        MEIXmpDACPhase                  Phase;
        MEIMotorDacChannelConfig        Cmd;
        MEIMotorDacChannelConfig        Aux;
} MEIMotorDacConfig;
```

To read a Dac's offset or write a Dac's offset, use the new methods:

```
meiMotorDacConfigGet(MPIMotor               motor,
                     MEIMotorDacConfig      *dacConfig,
                     MEIFlash               flash);

meiMotorDacConfigSet(MPIMotor               motor,
                     MEIMotorDacConfig      *dacConfig,
                     MEIFlash               flash);
```

The MEIMotorStatus{…} structure has been extended to include a new MEIMotorDacStatus{…} structure. Each MEIMotorDacStatus{…} structure contains a cmd and an aux MEIMotorDacChannelStatus{…} structure:

```
typedef struct MEIMotorDacChannelStatus {
        float          level;        /* volts */
} MEIMotorDacChannelStatus;

typedef struct MEIMotorDacStatus {
        MEIMotorDacChannelStatus                   cmd;
        MEIMotorDacChannelStatus                   aux;
} MEIMotorDacStatus;

typedef struct MEIMotorStatus {
        MEIMotorDacStatus                dac;
} MEIMotorStatus;
```

To read a Dac's level, use meiMotorStatus(…).

Since the MPI has changed, the Dac numbering is different:

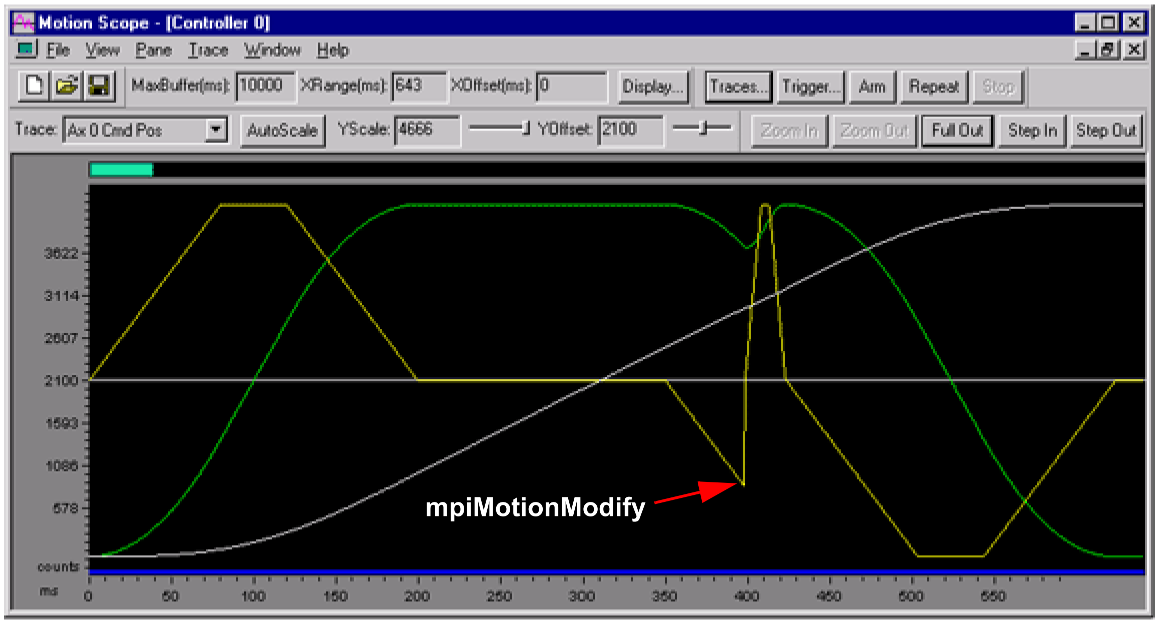
	OLD MPI	***NEW MPI***	***NEW MPI***
Hardware Signal	**Dac Number**	**Motor Number**	**Dac Channel**
Cmd_Dac_Out_0	0	0	Cmd
Aux_Dac_Out_0	16	0	Aux
Cmd_Dac_Out_1	1	1	Cmd
Aux_Dac_Out_1	17	1	Aux
Cmd_Dac_Out_2	2	2	Cmd
Aux_Dac_Out_2	18	2	Aux
Cmd_Dac_Out_3	3	3	Cmd
Aux_Dac_Out_3	19	3	Aux
Cmd_Dac_Out_4	4	4	Cmd
Aux_Dac_Out_4	20	4	Aux
Cmd_Dac_Out_5	5	5	Cmd
Aux_Dac_Out_5	21	5	Aux
Cmd_Dac_Out_6	6	6	Cmd
Aux_Dac_Out_6	22	6	Aux
Cmd_Dac_Out_7	7	7	Cmd
Aux_Dac_Out_7	23	7	Aux
Cmd_Dac_Out_8	8	8	Cmd
Aux_Dac_Out_8	24	8	Aux
Cmd_Dac_Out_9	9	9	Cmd
Aux_Dac_Out_9	25	9	Aux
Cmd_Dac_Out_10	10	10	Cmd
Aux_Dac_Out_10	26	10	Aux
Cmd_Dac_Out_11	11	11	Cmd

	OLD MPI	***NEW MPI***	***NEW MPI***
Aux_Dac_Out_11	27	11	Aux
Cmd_Dac_Out_12	12	12	Cmd
Aux_Dac_Out_12	28	12	Aux
Cmd_Dac_Out_13	13	13	Cmd
Aux_Dac_Out_13	29	13	Aux
Cmd_Dac_Out_14	14	14	Cmd
Aux_Dac_Out_14	30	14	Aux
Cmd_Dac_Out_15	15	15	Cmd
Aux_Dac_Out_15	31	15	Aux

The MPIControlConfig{…} structure has been changed to include cmdDacCount and auxDacCount. The cmdDacCount specifies the number of enabled Cmd Dacs and auxDacCount specifies the number of enabled Aux Dacs. To configure the number of enabled Dacs, use mpiControlConfigGet/Set(…).

2.18 New S-Curve Jerk Algorithm MPI615

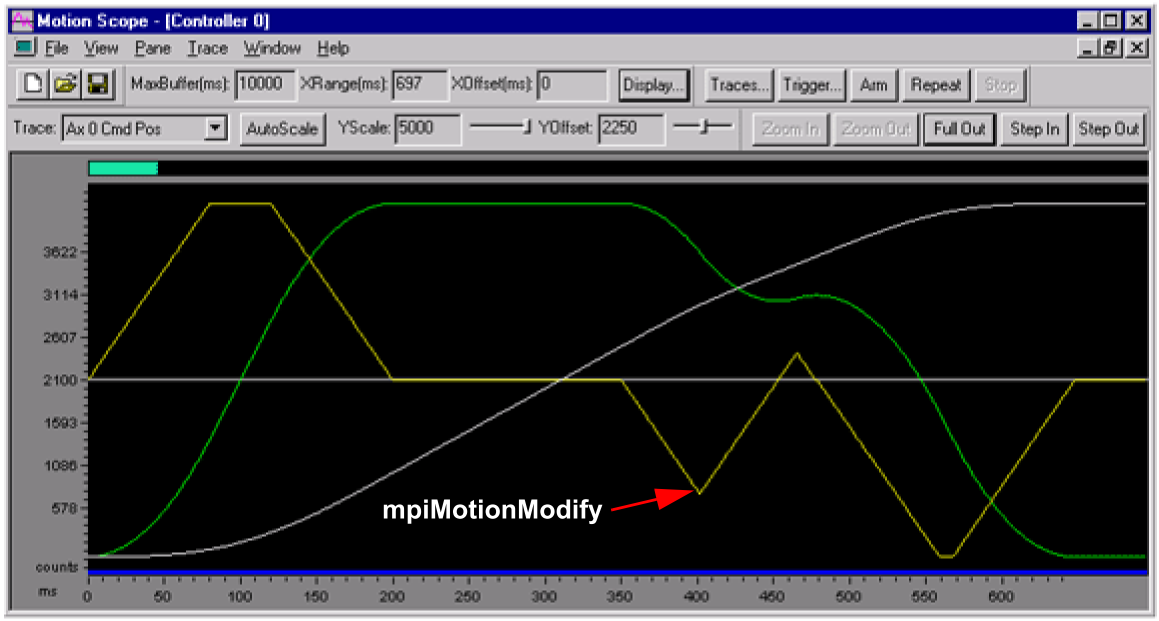
A new move type, MPIMotionTypeS_CURVE_JERK, has been added to support a jerk-specified profile. This replaces the old jerkPercent algorithm. Two added features that the new S-Curve Jerk algorithm provide are the ability to call a motion modify at any time during a path move and the freedom to change jerk, acceleration, and maximum velocity independently. None of these values will be exceeded in the resulting motion. The new S-Curve Jerk algorithm will be ideal for making final adjustments to a move as it draws closer to its final target and for making smoother transitions from one motion to the next.


Motion Scope - [Controller 0]

MaxBuffer(ms): 10000 XRange(ms): 643 XOffset(ms): 0

Trace: Ax 0 Cmd Pos YScale: 4666 YOffset: 2100

Old firmware: Notice that the acceleration is assumed to be zero and that there is a sudden change in the velocity as a result.



New Firmware: Notice that the acceleration changes less abruptly and that the velocity profile is much smoother.

Safe parameters for jerk values should range from a minimum of **amax* amax / vmax** (amax is just reached when accelerating from 0 to vmax) and a maximum of **amax / sample period** (amax is reached in one sample period). In the new firmware, changes to the jerk will also change the time needed to complete a motion. For example, a large value of jerk will have a shorter time, but increase the "jerkiness" of the motion (*see fig 1*). Conversely, a small value of jerk will have a longer time, but a much smoother motion (*see fig 2*).

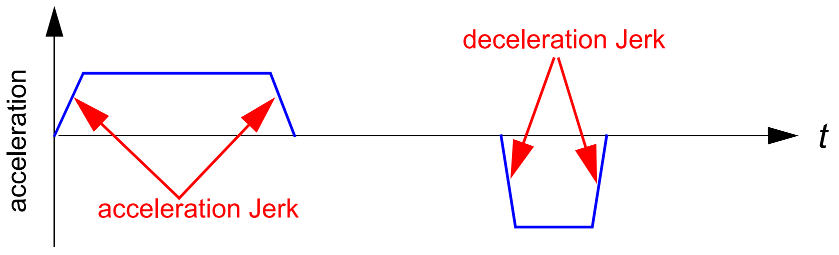


Fig 1. Acceleration profile with **larger** value of jerk.

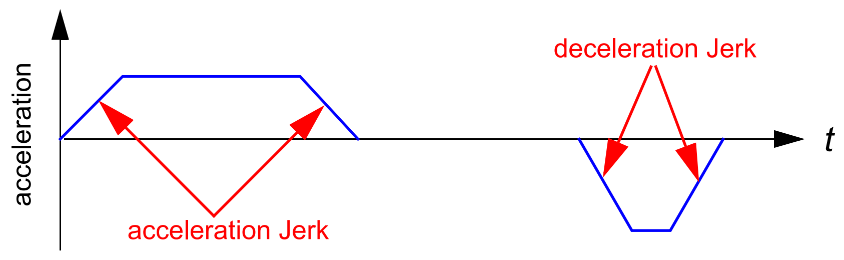


Fig 2. Acceleration profile with **smaller** value of jerk.

Two new parameters, accelerationJerk and decelerationJerk have been added to the MPITrajectory{…} structure. When they are non-zero, the acceleration profile uses the specified jerk and acceleration to ramp an axis(es) to constant velocity and then decelerate to a stop. If accelerationJerk or decelerationJerk is zero, an illegal parameter error is returned.

For the move type, MPIMotionTypeS_CURVE, the MPI calculates an appropriate jerk value based on the specified velocity, acceleration, and jerkPercent. The jerk value is computed according to the following formula:

jerk = amax * amax / (vmax * jp * sp (1 - jp * sp))

jp = jerkPercent
sp = sample period

If jerkPercent is zero, the jerk value is computed so that the maximum acceleration is reached in one sample period. With the previous S-Curve algorithm, the time for a move would not change as jerkPercent value was varied. This is also true for this S-Curve algorithm, as long as the move reaches maximum velocity. In short moves, where maximum velocity is not reached, setting jerkPercent to be small will result in a quicker move than if you were to set jerkPercent to be large.

WARNING! The same jerkPercent values may cause different profiles than the previous S-Curve algorithm.
For S-Curve algorithm attributes, see section 2.5.

2.19 S-Curve Jerk Algorithm Attributes MPI623

The new S_Curve algorithm behaves similarly to the previous algorithm, except for its attributes.
For a general explanation of the new S-Curve algorithm, see section 2.4.

MPIMotionAttrMaskDELAY can now be used with any start motion, but never with motion modify.

MPIMotionAttrMaskAPPEND can be used with any motion, as long as it is not preceded by a motion that had a final velocity.

MEIMotionAttrMaskNO_REVERSAL returns a MPIMotionMessagePROFILE_ERROR if the given specifications would result in a move with a reversal init, thereby preventing the move from being executed.

MPIMotionTypeTRAPEZOIDAL, MPIMotionTypeS_CURVE, and MPIMotionTypeS_CURVE_JERK

MPIMotionAttrMaskRELATIVE, when used with MPIMotionTypeTRAPEZOIDAL, MPIMotionTypeS_CURVE and MPIMotionTypeS_CURVE_JERK means that the final position is relative to the beginning position of the motion.

MEIMotionAttrMaskFINAL_VEL can be used with MPIMotionTypeTRAPEZOIDAL, MPIMotionTypeS_CURVE and MPIMotionTypeS_CURVE_JERK, but should be used with caution as it may not be possible for the controller to compute a trajectory to meet these criteria, which would cause a MPIMotionMessagePROFILE_ERROR to be returned, and the move to be ignored.

--Multi-Axis Motion--

Neither MPIMotionAttrMaskSYNC_START nor MPIMotionAttrMaskSYNC_END
If neither MPIMotionAttrMaskSYNC_START nor MPIMotionAttrMaskSYNC_END are specified, a single MPITrajectory{...} may be specified for the resultant motion of multiple axes on one motion supervisor. The motion of each axis will be synchronized with the others on the motion supervisor. The maximum velocity, acceleration, deceleration, and jerk values of the first MPITrajectory structure will be used for the global vector parameters. It will ignore any other values supplied. This cannot be used with MEIMotionAttrMaskFINAL_VEL.

MPIMotionAttrMaskSYNC_START or MPIMotionAttrMaskSYNC_END but not both
If MPIMotionAttrMaskSYNC_START or MPIMotionAttrMaskSYNC_END (but not both) is specified, each axis will move as fast as possible and either start together, or stop together. If motion is point-to-point and more than one axis on the motion supervisor has a final velocity, MPIMotionAttrMaskSYNC_START or MPIMotionAttrMaskSYNC_END must be used. MPIMotionAttrMaskSYNC_END cannot be used with motion modify.

Both MPIMotionAttrMaskSYNC_START and MPIMotionAttrMaskSYNC_END
With MPIMotionAttrMaskSYNC_START and MPIMotionAttrMaskSYNC_END, the motion for each axis will be scaled so that the motion of all axes will end at approximately the same time. The time for this motion is based on the time for the longest motion, so that the limits are not exceeded. The axes will be scaled to start and stop together, but the scaling may not be exact. Both MPIMotionAttrMaskSYNC_START and MPIMotionAttrMaskSYNC_END cannot be used together with MEIMotionAttrMaskFINAL_VEL.

MPIMotionTypeVELOCITY

MPIMotionTypeVELOCITY moves allow a final velocity to be specified without a final point.

MPIMotionAttrMaskSYNC_START and/or MPIMotionAttrMaskSYNC_END
Neither is supported for this motion type. MPIMotionAttrMaskSYNC_END cannot be used with motion modify.

MEIMotionAttrMaskFINAL_VEL
MEIMotionAttrMaskFINAL_VEL is not supported for this motion type.

MPIMotionAttrMaskRELATIVE
MPIMotionAttrMaskRELATIVE, when used with MPIMotionTypeVELOCITY or MPIMotionTypeVELOCITY_JERK, means that the final velocity is relative to the velocity at the start of the motion.

2.20 Configurable Record Buffer Size MPI577

The Data Recorder buffer size can now be dynamically allocated. The MPIControlConfig{...} structure has a new element, called recordCount. This element allows the application to change the size of the recorder object's data buffer using the mpiControlConfigGet/Set(...) methods. A larger data buffer size can improve the performance of MotionScope running on a slow host or running in Client/Server mode over a congested network.

A new method, meiControlExtMemAvail(...), has been added which will return the size of external memory available for allocation. This value can be added to the current recordCount to expand the record buffer to the maximum possible size.

2.21 Dynamic Allocation of External Memory Buffers MPI575

In previous versions, the XMP external memory was allocated statically at firmware compile time.

In version 20010119 and later, specific buffers of the XMP external memory is dynamically allocated. The dynamic allocation feature allows an application to efficiently use the XMP controller's on-board memory and allows for future expansion. The dynamically allocated buffers currently include the Frame Buffer, Record Buffer and the SERCOS buffer. Each of these buffers sizes are recalculated during a call to mpiControlConfigSet(...) if any of the associated ControlConfig values change.

The Frame Buffer is used for motion on each axis. The Frame Buffer is directly associated with the number of EnabledAxes in the MPIControlConfig structure. The Frame Buffer will be allocated to the minimum size required to support the number of enabled axes. The default number of EnabledAxes is eight (8).

The Record Buffer is used for the on-board data recorder. The Record Buffer is directly associated with the number of EnabledRecord in the MPIControlConfig structure. The Record Buffer will be allocated to the minimum size required to support the number of enabled records. The default number of EnabledRecords is 3064. Each record is the size of one memory word.

The Sercos Buffer is used for motion on each SERCOS ring network. The Sercos Buffer is directly associated with the number of EnabledSercos in the MPIControlConfig structure. The Sercos Buffer will be allocated to the minimum size required to support the number of enabled Sercos rings. The default number of EnabledSercosRings, for a non-sercos controller is zero (0).

A new method has been added to discover how much memory is available on your controller. The method…

```
MPI_DEF1 long MPI_DEF2
        meiControlExtMemAvail(MPIControl       control,
                                   long        *size)
```

…will return the number of memory words available. Since each record size is one memory word, the size returned from the above function can be used to increase the Record Buffer to maximum size possible. This greatly improves client/server operation of Motion Scope and any application used for data collection.

WARNING: Due to the nature of dynamic allocation and the clearing of external memory buffers mpiControlConfigSet(...) should ONLY be called at motion application initialization time and NOT during motion.

2.22 mpiAxisActualVelocity argument changed MPI546

In previous releases, the actual velocity argument in mpiAxisActualVelocity(...) was a float. It has been changed to a double for consistency with other MPI methods. This was changed in version 20001130.

2.23 mpiAxisPositionError(...) added MPI518

A new method has been added:

```
mpiAxisPositionError(MPIAxis      axis)
                     double       *error);
```

This method is used to retrieve the Position Error from a given axis. A valid instance of MPIAxis is passed to this method along with a pointer to a long variable (long* error). The method does appropriate error checking on the MPIAxis varriable as well as the long *. Incorrect arguments could return MPIMessageARG_INVALID or MPIMessageOBJECT_NOT_ENABLED. The method then retreives the PositionError variable from the MEIXmpAxis structure, converts it from a float to a long, and assigns the value to the variable pointed to by the long * argument. A return value of 0 indicates that the procedure executed without error. This new method was added in version 20001103.

3 Incremental Changes

Since the general release of MPI version 20000913, API changes have been made to add features and to fix any bugs encountered. Below is an incremental list of changes that have occurred in later revisions. The incremental changes sections are in reverse order, with the most recent changes first. These changes have been made in a continual effort to provide a better product by incorporating customer feedback and rigorous testing methods. Fixed bugs are treated in Section 5 on page 107. Outstanding bugs and limitations are treated in Section 6 on page 115 of this release note.

Changes and *additions* to existing code are indicated in **bold** characters; in electronic media, they are indicated in **bold blue** characters.
Deletions are indicated in ~~**bold strikethrough**~~ characters; in electronic media, they are indicated in ~~**bold red strikethrough**~~ characters.
Extended lines of unchanged code are indicated with a vertical, bold ellipses along the left margin:

Version 20020117.1.12

In this release, the MPI and firmware versions are:

	Previous Version	New Version
Firmware	492A1	498A1
MPI Library	20020117.1.10	20020117.1.12

- PTF and PVTF improvements were made. Please see Section 2.1 of this general release note for more information.
- PTF and PVTF motion types have been added to the MPI library in motion.h. Please see Section 2.2 of this general release note for more information.

Version 20020117.1.10

In this release, the MPI and firmware versions are:

	Previous Version	New Version
Firmware	434A1	492A1
MPI Library	20020117.1.8	20020117.1.10

- A bug (MPI1240) existed where random, MPIMessageTIMEOUT errors were being returned from the MPI on specific PCs. Please see Section 5 of this release note for more information.
- A bug (MPI1223) existed where if a PVT (or other multi-point motion) move was stopped and then an SCurve (or other point-to-point) move was executed, the Motion Supervisor would enter into an ERROR state. Please see Section 5 of this release note for more information.

Version 20020117.1.8

In this release, the MPI and firmware versions are:

	Previous Version	New Version
Firmware	371A5	434A1
MPI Library	20020117.1.6	20020117.1.8

- The *const* handle was removed from all MPI methods. Returning *const* handles served no purpose and actually caused problems with some tools. Please see Section 2.3 of this general release note for more information.
- Some safety features were added to the Amp Enable logic. Please see Section 2.4 of this general release note for more information.
- An axis check has been added to the motion methods so that an error message will be returned if a motion method is called when no axis is associated with the motion object. Please see Section 2.5 of this general release note for more information.
- A bug (MPI703) where there was no Filter DRate check to make sure that the Filter object's DRate (derivative sub-sampling rate) is limited to a range from 0 to 7 has been fixed. Please see Section 2.6 of this general release note for more information.
- A change was made to correctly perform an "object on list" check, which prevents an application from deleting axis objects that are appended to any motion objects. Please see Section 5 of this release note for more information (MPI1044).
- A bug (MPI1035) where the sequence command MPICommandOperatorNOT_EQUAL would cause a sequencer to continually generate events to the host has been fixed. Please see Section 5 of this release note for more information.
- A bug (MPI1032) where a motion modify command in sequences was not supported has been fixed. Please see Section 5 of this release note for more information.
- A bug (MPI817) where meiMotionParamsValidate() failed with good parameters has been fixed. Please see Section 5 of this release note for more information.

Version 20020117.1.6

In this release, the MPI and firmware versions are:

	Previous Version	New Version
Firmware	364A3	371A5
MPI Library	20020117.1.3	20020117.1.6

- Multiple Injection Point Noise Source has been added to the xmp firmware. Please see Section 2.7 of this general release note for more information.
- Changes have been made to streamline the loading of points lists. Please see Section 2.8 of this general release note for more information.
- (MPI903) Two changes were made to the compensation table calculations to eliminate the immediate toggling of the compensation value on either side of a maximum compensation value in the compensation table. Please see Section 5 of this release note for more information.
- A bug (MPI895) where sim4calc.exe could calculate incorrect look-up tables has been fixed. Please see Section 5 of this release note for more information.
- A bug (MPI888) where PVT moves would produce duplicate points at the transition of Element IDs has been fixed. Please see Section 5 of this release note for more information.
- A bug (MPI887) in multi-point motion (i.e. PVT, PT, spline, etc), which could cause intermitent Memory Access Violation errors has been fixed. Please see Section 5 of this release note for more information.

- A bug (MPI885) where changing sample rates could cause unexpected motor faults has been fixed. Please see Section 5 of this release note for more information.
- A bug (MPI884) where the MPI would misinterpret the axis's state and report a Stopping Error has been fixed. Please see Section 5 of this release note for more information.
- A bug (MPI881) where the meiFlashMemoryVerify(...) would fail if the entire image was compared to a list of host files has been fixed. Please see Section 5 of this release note for more information.
- A bug (MPI879) where mpiFilterConfigSet(...) would improperly return a MPIMessagePARAM_INVALID error if the Algorithm was PIV and the PostFilter.Length was non-zero has been fixed. Please see Section 5 of this release note for more information.
- A typo (MPI843) in the mpiControlInit(...) macro definition, which caused compilation errors has been fixed. Please see Section 5 of this release note for more information.

Version 20020117.1.3

In this release, the MPI and firmware versions are:

	Previous Version	New Version
Firmware	358A2	364A3
MPI Library	20020117.1	20020117.1.3

- A bug (MPI829) where an improperly commented block of code caused a shortening of the step pulse by 1/4 has been fixed. Please see Section 5 of this release note for more information.
- A limitation (MPI836) exists where an overshoot will occur when a motion modify is called under certain conditions. Please see Section 6.2 of this general release note for more information.

Version 20020117.1

In this release, the MPI and firmware versions are:

	Previous Version	New Version
Firmware	358A2	358A2
MPI Library	20020117	20020117.1

- There were no general changes or bug fixes in the 20020117.1 release.

Version 20020117

In this release, the MPI and firmware versions are:

	Previous Version	New Version
Firmware	347B1	358A2
MPI Library	20011213	20020117

- MEI's new On-Line Documentation System is now available. You can access the web site at http://support.motioneng.com or through the Install Shield CD-ROM. Please see Section 2.9 of this general release note for more information.
- There is a new default XMP-Series Controller Configuration. The XMP-Series controller's Flash memory is now pre-loaded at the factory with base firmware. This firmware allows the MPI to identify the XMP-Series controller. Please see Section 2.10 of this general release note for more information.
- A new element, UserVersion has been added to the MPIControlConfig{...} structure. Please see Section 2.13 of this general release note for more information.
- The DAC level has changed from DAC units to volts. Please see Section 2.14 of this general release note for more information.
- A bug (MPI767) that caused a Device Driver port call failure has been fixed. Please see Section 5 of this release note for more information.
- A bug (MPI737) that was caused by incorrect handling of the UPDATE frame has been fixed. Please see Section 5 of this release note for more information.
- A bug (MPI735) where the IN_FINE criteria was only checked in the STOPPED state has been fixed. Please see Section 5 of this release note for more information.
- A bug (MPI734) which caused mpiMotionStart() calls occuring right after a DONE event to cause a MOVING error has been fixed. Please see Section 5 of this release note for more information.
- A bug (MPI691) where an erroneous TIMEOUT return value was returned from mpiMotorEvent-ConfigSet(...) has been fixed. Please see Section 5 of this release note for more information.
- A bug (MPI581) where an incorrect error was returned with mpiMotorStatus(...) and meiMotorSta-tus(...) has been fixed. Please see Section 5 of this release note for more information.
- A bug (MPI528) where mpiAxisCommandPositionSet(...) would not set the command position if the axis was in the Stop condition has been fixed. Please see Section 5 of this release note for more information.
- A bug (23) where mpiControlReset(...) would return too early, causing MPI methods to fail has been fixed. Please see Section 5 of this release note for more information.
- A bug (MPI632) where there was a FrameBuffer referencing error has been fixed. Please see Section 5 of this release note for more information.
- A bug (MPI586) where performing a MPIActionStop after a MPIActionReset caused an error has been fixed. Please see Section 5 of this release note for more information.
- A bug (MPI625) where executing the flash utility with the server option failed has been fixed. Please see Section 5 of this release note for more information.
- A bug (MPI573) where changes made to mpiMotorIoSet(...) could be erased because of Riptide latencies has been fixed. Please see Section 5 of this release note for more information.
- A bug (MPI544) where there was no action synchronization between the MPI and firmware caus-ing the MPI to read old values has been fixed. Please see Section 5 of this release note for more information.

Version 20011213

In this release, the MPI and firmware versions are:

	Previous Version	New Version
Firmware	325B2	347B1
MPI Library	20010417.1	20011213

- A new field has been added to the XMP's firmware to identify and differentiate between intermediate branch software revisions. Please see Section 2.11 of this general release note for more information.
- Path trajectory generation is now supported by PT, PVT, SPLINE, BESSEL, BSPLINE and BSPLINE2 algorithms. Please see Section 2.10 of this general release note for more information.
- A bug (MPI769) where there was a problem caused by an improper error check in the mpiMotionModify(...) has been fixed. Please see Section 5 of this release note for more information.
- A bug (MPI713) where mpiRecorderRecordGet(...) returned corrupted data has been fixed. Please see Section 5 of this general release note for more information.
- A bug (MPI 697) where improper interaction occurred between the mpiMotionModify(...) and command position has now been fixed. Please see Section 5 of this general release note for more information.
- A bug (MPI 688) where mpiMotorConfigGet() would return an error if both the MPIMotorConfig and MEIMotorConfig structures were passed has now been fixed. Please see Section 5 of this general release note for more information.
- A bug (MPI 686) where there was in incorrect motion profile with mpiMotionModify(...) has now been fixed. Please see Section 5 of this general release note for more information.
- A bug (MPI 683) mpiMotionModify(...) would cause a trajectory dicontinuity with velocity type moves has now been fixed. Please see Section 5 of this general release note for more information.
- A bug (MPI 672) where the filter coefficient value for DRate was being incorrectly saved has now been fixed. Please see Section 5 of this general release note for more information.
- A bug (MPI 659) where mpiControlReset(...) could lock up the PCI bus has now been fixed. Please see Section 5 of this general release note for more information.
- A bug (MPI 741) exists where the Win2000 device driver will not allow a host system to go into “Standby” mode. Please see Section 6 for more information.
- A limitation (MPI 703) exists where the Filter object's DRate (derivative sub-sampling rate) is limited to a range from 0 to 7. Please see Section 6.2 for more information.

Version 20010417.1

In this release, the MPI and firmware versions are:

	Previous Version	New Version
Firmware	310B3	325B2
MPI Library	2000091302	20010417.1

Changes to MPI/XMP Header Files

- **Addition of MPIMotionPTF data type ("motion.h" header file)**

```
typedefstruct MPIMotionPTF {
        long                    pointCount;
        double                  *position;
        double                  *feedforward;
        double                  *time;

        MPIMotionPoint          point;
} MPIMotionPTF;
```

- **Addition of MPIMotionPVTF data type ("motion.h" header file)**

```
typedefstruct MPIMotionPVTF {
        long                    pointCount;
        double                  *position;
        double                  *velocity;
        double                  *feedforward;
        double                  *time;

        MPIMotionPoint          point;
} MPIMotionPVTF;
```

- **Changes to MPIMotionParams data type ("motion.h" header file)**

```
typedefstruct MPIMotionParams {
        MPIMotionJog                    jog;

        MPIMotionPT                     pt;
        MPIMotionPTF                    ptf;
        MPIMotionPVT                    pvt;
        MPIMotionPVTF                   pvtf;
        MPIMotionSPLINE                 spline;
        MPIMotionBESSEL                 bessel;
        MPIMotionBSPLINE                bspline;

        MPIMotionSCurve                 sCurve;
        MPIMotionSCurve                 sCurveJerk;
        MPIMotionTrapezoidal            trapezoidal;

        MPIMotionVelocity               velocity;
        MPIMotionVelocity               velocityJerk;

        MPIMotionAttributes             attributes;

        void                            *external;
} MPIMotionParams;
```

- **Change to mpiAdcCreate method ("adc.h" header file)**

MPI_DECL1 ~~const~~ MPIAdc MPI_DECL2

```
        mpiAdcCreate(MPIControl         control,
                     long               number);
```

- **Change to mpiAdcControl method ("adc.h" header file)**

MPI_DECL1 ~~const~~ MPIControl MPI_DECL2

```
        mpiAdcControl(MPIAdc            adc);
```

- **Changes to MPIAxisMessage data type ("axis.h" header file)**

```
typedef enum {
    MPIAxisMessageFIRST = mpiMessageID(MPIModuleIdAXIS, 0),

    MPIAxisMessageAXIS_INVALID,
    MPIAxisMessageCOMMAND_NOT_SET,
   MPIAxisMessageNOT_MAPPED_TO_MS,
    MPIAxisMessageLAST
} MPIAxisMessage;
```

- **Change to <u>mpiAxisCreate</u> method ("axis.h" header file)**

MPI_DECL1 ~~const~~ MPIAxis MPI_DECL2

```
    mpiAxisCreate(MPIControl     control,
                  long           number);
```

- **Change to <u>mpiAxisControl</u> method ("axis.h" header file)**

MPI_DECL1 ~~const~~ MPIControl MPI_DECL2

```
    mpiAxisControl(MPIAxis         axis);
```

- **Change to <u>mpiCaptureCreate</u> method ("capture.h" header file)**

MPI_DECL1 ~~const~~ MPICapture MPI_DECL2

```
    mpiCaptureCreate(MPIControl  control,
                     long        number);
```

- **Change to <u>mpiCaptureControl</u> method ("capture.h" header file)**

MPI_DECL1 ~~const~~ MPIControl MPI_DECL2

```
    mpiCaptureControl(MPICapture            capture);
```

- **Change to <u>mpiCommandCreate</u> method ("command.h" header file)**

MPI_DECL1 ~~const~~ MPICommand MPI_DECL2

```
    mpiCommandCreate(MPICommandType             type,
                     MPICommandParams           *params,
                      const char                *label);
```

- **Change to <u>mpiCompareCreate</u> method ("compare.h" header file)**

MPI_DECL1 ~~const~~ MPICompare MPI_DECL2

```
    mpiCompareCreate(MPIControl             control,
                      long                  number);
```

- **Change to <u>mpiCompareControl</u> method ("compare.h" header file)**

MPI_DECL1 ~~const~~ MPIControl MPI_DECL2

```
    mpiCompareControl(MPICompare      compare);
```

- **Change to <u>mpiControlCreate</u> method ("control.h" header file)**

MPI_DECL1 ~~const~~ MPIControl MPI_DECL2

```
    mpiControlCreate(MPIControlType          type,
                  MPIControlAddress       *address);
```

- **Change to <u>mpiEventCreate</u> method ("event.h" header file)**

MPI_DECL1 ~~const~~ MPIEvent MPI_DECL2

```
    mpiEventCreate(MPIEventStatus             *status);
```

- **Change to <u>mpiEventMgrControl</u> method ("eventmgr.h" header file)**

```
MPI_DECL1 ~~const~~ MPIControl MPI_DECL2
    mpiEventMgrControl(MPIEventMgr      eventMgr,
                       long             index);
```

- **Change to <u>mpiEventMgrControlFirst</u> method ("eventmgr.h" header file)**

```
MPI_DECL1 ~~const~~ MPIControl MPI_DECL2
    mpiEventMgrControlFirst(MPIEventMgr    eventMgr);
```

- **Change to <u>mpiEventMgrControlLast</u> method ("eventmgr.h" header file)**

```
MPI_DECL1 ~~const~~ MPIControl MPI_DECL2
    mpiEventMgrControlLast(MPIEventMgr    eventMgr);
```

- **Change to <u>mpiEventMgrControlNext</u> method ("eventmgr.h" header file)**

```
MPI_DECL1 ~~const~~ MPIControl MPI_DECL2
    mpiEventMgrControlNext(MPIEventMgr eventMgr,
                           MPIControl  control);
```

- **Change to <u>mpiEventMgrControlPrevious</u> method ("eventmgr.h" header file)**

```
MPI_DECL1 ~~const~~ MPIControl MPI_DECL2
    mpiEventMgrControlPrevious(MPIEventMgr       eventMgr,
                               MPIControl        control);
```

- **Change to <u>mpiEventMgrNotify</u> method ("eventmgr.h" header file)**

```
MPI_DECL1 ~~const~~ MPINotify MPI_DECL2
    mpiEventMgrNotify(MPIEventMgr       eventMgr,
                      long              index);
```

- **Change to <u>mpiEventMgrNotifyFirst</u> method ("eventmgr.h" header file)**

```
MPI_DECL1 ~~const~~ MPINotify MPI_DECL2
    mpiEventMgrNotifyFirst(MPIEventMgr    eventMgr);
```

- **Change to <u>mpiEventMgrNotifyLast</u> method ("eventmgr.h" header file)**

```
MPI_DECL1 ~~const~~ MPINotify MPI_DECL2
    mpiEventMgrNotifyLast(MPIEventMgr    eventMgr);
```

- **Change to <u>mpiEventMgrNotifyNext</u> method ("eventmgr.h" header file)**

```
MPI_DECL1 ~~const~~ MPINotify MPI_DECL2
    mpiEventMgrNotifyNext(MPIEventMgreventMgr,
                          MPINotify      notify);
```

- **Change to <u>mpiEventMgrNotifyPrevious</u> method ("eventmgr.h" header file)**

```
MPI_DECL1 ~~const~~ MPINotify MPI_DECL2
    mpiEventMgrNotifyPrevious(MPIEventMgr        eventMgr,
                              MPINotify          notify);
```

- **Changes to <u>MPIFilterMessage</u> data type ("filter.h" header file)**

```
typedef enum {
    MPIFilterMessageFIRST = mpiMessageID(MPIModuleIdFILTER, 0),

    MPIFilterMessageFILTER_INVALID,
    MPIFilterMessageINVALID_ALGORITHM,
    MPIFilterMessageINVALID_DRATE,

    MPIFilterMessageLAST
} MPIFilterMessage;
```

- **Change to <u>mpiFilterCreate</u> method ("filter.h" header file)**

MPI_DECL1 ~~const~~ MPIFilter MPI_DECL2
```
        mpiFilterCreate(MPIControl     control,
                        long           number);
```

- **Change to <u>mpiFilterControl</u> method ("filter.h" header file)**

MPI_DECL1 ~~const~~ MPIControl MPI_DECL2
```
        mpiFilterControl(MPIFilter      filter);
```

- **Change to <u>mpiIdnCreate</u> method ("idn.h" header file)**

MPI_DECL1 ~~const~~ MPIIdn MPI_DECL2
```
        mpiIdnCreate(MPIIdnNumber     number);
```

- **Change to <u>mpiIdnListCreate</u> method ("idnlist.h" header file)**

MPI_DECL1 ~~const~~ MPIIdnList MPI_DECL2
```
        mpiIdnListCreate(MPIIdn      idn);
```

- **Change to <u>mpiMotionCreate</u> method ("motion.h" header file)**

MPI_DECL1 ~~const~~ MPIMotion MPI_DECL2
```
        mpiMotionCreate(MPIControl  control,
                        long        number,
                        MPIAxis     axis);
```

- **Change to <u>mpiMotionControl</u> method ("motion.h" header file)**

MPI_DECL1 ~~const~~ MPIControl MPI_DECL2
```
        mpiMotionControl(MPIMotion      motion);
```

- **Change to <u>mpiMotionAxis</u> method ("motion.h" header file)**

MPI_DECL1 ~~const~~ MPIAxis MPI_DECL2
```
        mpiMotionAxis(MPIMotion      motion,
                      long           index);
```

- **Change to <u>mpiMotionAxisFirst</u> method ("motion.h" header file)**

MPI_DECL1 ~~const~~ MPIAxis MPI_DECL2
```
        mpiMotionAxisFirst(MPIMotion      motion);
```

- **Change to <u>mpiMotionAxisLast</u> method ("motion.h" header file)**

MPI_DECL1 ~~const~~ MPIAxis MPI_DECL2
```
        mpiMotionAxisLast(MPIMotion      motion);
```

- **Change to <u>mpiMotionAxisNext</u> method ("motion.h" header file)**

MPI_DECL1 ~~const~~ MPIAxis MPI_DECL2
```
        mpiMotionAxisNext(MPIMotion   motion,
                          MPIAxis     axis);
```

- **Change to mpiMotionAxisPrevious method ("motion.h" header file)**

```
MPI_DECL1 ~~const~~ MPIAxis MPI_DECL2
    mpiMotionAxisPrevious(MPIMotion      motion,
                          MPIAxis        axis);
```

- **Change to mpiMotorCreate method ("motor.h" header file)**

```
MPI_DECL1 ~~const~~ MPIMotor MPI_DECL2
    mpiMotorCreate(MPIControl   control,
                   long         number);
```

- **Change to mpiMotorControl method ("motor.h" header file)**

```
MPI_DECL1 ~~const~~ MPIControl MPI_DECL2
    mpiMotorControl(MPIMotor     motor);
```

- **Change to mpiNodeCreate method ("node.h" header file)**

```
MPI_DECL1 ~~const~~ MPINode MPI_DECL2
    mpiNodeCreate(MPISercos    sercos,
                  long         number);
```

- **Change to mpiNodeSercos method ("node.h" header file)**

```
MPI_DECL1 ~~const~~ MPISercos MPI_DECL2
    mpiNodeSercos(MPINode    node);
```

- **Change to mpiNotifyCreate method ("notify.h" header file)**

```
MPI_DECL1 ~~const~~ MPINotify MPI_DECL2
    mpiNotifyCreate(MPIEventMask     mask,
                    void             *source);
```

- **Change to mpiPathCreate method ("path.h" header file)**

```
MPI_DECL1 ~~const~~ MPIPath MPI_DECL2
    mpiPathCreate();
```

- **Change to mpiProgramCreate method ("program.h" header file)**

```
MPI_DECL1 ~~const~~ MPIProgram MPI_DECL2
    mpiProgramCreate();
```

- **Change to mpiProgramCommand method ("program.h" header file)**

```
MPI_DECL1 ~~const~~ MPICommand MPI_DECL2
    mpiProgramCommand(MPIProgram      program,
                      long            index);
```

- **Change to mpiProgramCommandFirst method ("program.h" header file)**

```
MPI_DECL1 ~~const~~ MPICommand MPI_DECL2
    mpiProgramCommandFirst(MPIProgram     program);
```

- **Change to mpiProgramCommandLast method ("program.h" header file)**

```
MPI_DECL1 ~~const~~ MPICommand MPI_DECL2
    mpiProgramCommandLast(MPIProgram     program);
```

- **Change to mpiProgramCommandNext method ("program.h" header file)**

```
MPI_DECL1 ~~const~~ MPICommand MPI_DECL2
```

```
        mpiProgramCommandNext(MPIProgram            program,
                              MPICommand            command);
```

- **Change to <u>mpiProgramCommandPrevious</u> method ("program.h" header file)**

```
MPI_DECL1 ~~const~~ MPICommand MPI_DECL2
        mpiProgramCommandPrevious(MPIProgram        program,
                                  MPICommand        command);
```

- **Change to <u>mpiRecorderCreate</u> method ("recorder.h" header file)**

```
MPI_DECL1 ~~const~~ MPIRecorder MPI_DECL2
        mpiRecorderCreate(MPIControl      control);
```

- **Change to <u>mpiRecorderControl</u> method ("recorder.h" header file)**

```
MPI_DECL1 ~~const~~ MPIControl MPI_DECL2
        mpiRecorderControl(MPIRecorder     recorder);
```

- **Change to <u>mpiSequenceCreate</u> method ("sequence.h" header file)**

```
MPI_DECL1 ~~const~~ MPISequence MPI_DECL2
        mpiSequenceCreate(MPIControl          control,
                          long                number,
                          long                pageSize);
```

- **Change to <u>mpiSequenceControl</u> method ("sequence.h" header file)**

```
MPI_DECL1 ~~const~~ MPIControl MPI_DECL2
        mpiSequenceControl(MPISequence     sequence);
```

- **Change to <u>mpiSequenceCommand</u> method ("sequence.h" header file)**

```
MPI_DECL1 ~~const~~ MPICommand MPI_DECL2
        mpiSequenceCommand(MPISequence              sequence,
                           long                     index);
```

- **Change to <u>mpiSequenceCommandFirst</u> method ("sequence.h" header file)**

```
MPI_DECL1 ~~const~~ MPICommand MPI_DECL2
        mpiSequenceCommandFirst(MPISequence     sequence);
```

- **Change to <u>mpiSequenceCommandLast</u> method ("sequence.h" header file)**

```
MPI_DECL1 ~~const~~ MPICommand MPI_DECL2
        mpiSequenceCommandLast(MPISequence      sequence);
```

- **Change to <u>mpiSequenceCommandNext</u> method ("sequence.h" header file)**

```
MPI_DECL1 ~~const~~ MPICommand MPI_DECL2
        mpiSequenceCommandNext(MPISequence    sequence,
                               MPICommand     command);
```

- **Change to <u>mpiSequenceCommandPrevious</u> method ("sequence.h" header file)**

```
MPI_DECL1 ~~const~~ MPICommand MPI_DECL2
        mpiSequenceCommandPrevious(MPISequence    sequence,
                                   MPICommand     command);
```

- **Change to <u>mpiSercosCreate</u> method ("sercos.h" header file)**

```
MPI_DECL1 ~~const~~ MPISercos MPI_DECL2
        mpiSercosCreate(MPIControl                control,
                    long                    number);
```

- **Change to <u>mpiSercosControl</u> method ("sercos.h" header file)**

```
MPI_DECL1 ~~const~~ MPIControl MPI_DECL2
        mpiSercosControl(MPISercos        sercos);
```

- **Change to <u>mpiSercosNode</u> method ("sercos.h" header file)**

```
MPI_DECL1 ~~const~~ MPINode MPI_DECL2
        mpiSercosNode(MPISercos    sercos,
                      long           index);
```

- **Change to <u>mpiSercosNodeFirst</u> method ("sercos.h" header file)**

```
MPI_DECL1 ~~const~~ MPINode MPI_DECL2
        mpiSercosNodeFirst(MPISercos       sercos);
```

- **Change to <u>mpiSercosNodeLast</u> method ("sercos.h" header file)**

```
MPI_DECL1 ~~const~~ MPINode MPI_DECL2
        mpiSercosNodeLast(MPISercos      sercos);
```

- **Change to <u>mpiSercosNodeNext</u> method ("sercos.h" header file)**

```
MPI_DECL1 ~~const~~ MPINode MPI_DECL2
        mpiSercosNodeNext(MPISercos            sercos,
                          MPINode              node);
```

- **Change to <u>mpiSercosNodePrevious</u> method ("sercos.h" header file)**

```
MPI_DECL1 ~~const~~ MPINode MPI_DECL2
        mpiSercosNodePrevious(MPISercos     sercos,
                              MPINode         node);
```

- **Changes to <u># defines</u> ("control.h" header file)**

```
#define MPI_VERSION     "20020117"

#define MPIControlUserIoSizeINPUT       (2)
#define MPIControlUserIoSizeOUTPUT      (2)
#define MPIControlUserIoSizeCONFIG      (2)
```

- **Changes to <u>MPIAxisMessage</u> data type ("axis.h" header file)**

```
typedef enum {
        MPIAxisMessageFIRST = mpiMessageID(MPIModuleIdAXIS, 0),

        MPIAxisMessageAXIS_INVALID,
        MPIAxisMessageCOMMAND_NOT_SET,

        MPIAxisMessageLAST
} MPIAxisMessage;
```

- **Changes to <u>MPICommandMessage</u> data type ("command.h" header file)**

```
typedef enum {
        MPICommandMessageFIRST = mpiMessageID(MPIModuleIdCOMMAND, 0),

        MPICommandMessageCOMMAND_INVALID,
        MPICommandMessageTYPE_INVALID,
        MPICommandMessagePARAM_INVALID,

        MPICommandMessageLAST
} MPICommandMessage;
```

- **Changes to #defines ("control.h" header file)**

```
#define MPI_VERSION     "20010828"

#define MPIControlUserIoSizeINPUT      (2)
#define MPIControlUserIoSizeOUTPUT   (2)
#define MPIControlUserIoSizeCONFIG    (2)
```

Changes to MPIControlConfig ("control.h" header file)

```
typedef struct MPIControlConfig {
        long    adcCount;
        long    axisCount;
        long    captureCount;
        long    compareCount;
        long    cmdDacCount;
        long    auxDacCount;
        long    filterCount;
        long    motionCount;
        long    motorCount;
        long    recordCount;
        long    sequenceCount;
        long    sercosCount;
        long    userVersion;
        long    sampleRate;
        ~~MPIControlUserIoConfiguserIoConfig;~~
        ~~MPIControlUserIo    userIo;~~
} MPIControlConfig;
```

- **Changes to MPIControlMessage data type ("control.h" header file)**

```
typedef enum {
        MPIControlMessageFIRST = mpiMessageID(MPIModuleIdCONTROL, 0),

        MPIControlMessageLIBRARY_VERSION,       /* Keep as first control message */
        MPIControlMessageADDRESS_INVALID,
        MPIControlMessageCONTROL_INVALID,
        MPIControlMessageTYPE_INVALID,
        MPIControlMessageINTERRUPTS_DISABLED,
        MPIControlMessageEXTERNAL_MEMORY_OVERFLOW,
        MPIControlMessageADC_COUNT_INVALID,
        MPIControlMessageAXIS_COUNT_INVALID,
        MPIControlMessageCAPTURE_COUNT_INVALID,
        MPIControlMessageCOMPARE_COUNT_INVALID,
        MPIControlMessageCMDDAC_COUNT_INVALID,
        MPIControlMessageAUXDAC_COUNT_INVALID,
        MPIControlMessageFILTER_COUNT_INVALID,
        MPIControlMessageMOTION_COUNT_INVALID,
        MPIControlMessageMOTOR_COUNT_INVALID,
        MPIControlMessageLAST
} MPIControlMessage;
```

- **Deletion of mpiControlUserIoConfigGet method ("control.h" header file)**

```
~~MPI_DECL1 long MPI_DECL2~~
        ~~mpiControlUserIoConfigGet(MPIControl                    control,~~
                                    ~~MPIControlUserIoConfig      *userIoConfig);~~
```

- **Deletion of mpiControlUserIoConfigSet method ("control.h" header file)**

```
~~MPI_DECL1 long MPI_DECL2~~
    ~~mpiControlUserIoConfigSet(MPIControl             control,~~
                              ~~MPIControlUserIoConfig    *userIoConfig);~~
```

- **Deletion of mpiControlUserIoGet method ("control.h" header file)**

```
~~MPI_DECL1 long MPI_DECL2~~
    ~~mpiControlUserIoGet(MPIControl            control,~~
                        ~~MPIControlUserIo      *userIo);~~
```

- **Deletion of mpiControlUserIoSet method ("control.h" header file)**

```
~~MPI_DECL1 long MPI_DECL2~~
    ~~mpiControlUserIoSet(MPIControl            control,~~
                        ~~MPIControlUserIo      *userIo);~~
```

- **Changes to MPIMotionMessage data type ("motion.h" header file)**

```
typedef enum {
    MPIMotionMessageFIRST = mpiMessageID(MPIModuleIdMOTION, 0),

    MPIMotionMessageMOTION_INVALID,
    MPIMotionMessageAXIS_NOT_FOUND,
    MPIMotionMessageAXIS_COUNT,
    MPIMotionMessageTYPE_INVALID,
    MPIMotionMessageATTRIBUTE_INVALID,
    MPIMotionMessageNOT_READY,
    MPIMotionMessageIDLE,
    MPIMotionMessageMOVING,
    MPIMotionMessageSTOPPING,
    MPIMotionMessageSTOPPING_ERROR,
    MPIMotionMessageERROR,
    MPIMotionMessageAUTO_START,
    MPIMotionMessagePROFILE_ERROR,
    MPIMotionMessagePATH_ERROR,
    MPIMotionMessageFRAMES_LOW,
    MPIMotionMessageFRAMES_EMPTY,

    MPIMotionMessageLAST
} MPIMotionMessage;
```

- **Changes to MPIPathElementType data type ("path.h" header file)**

```
typedef enum {
    MPIPathElementTypeINVALID = -1,

    MPIPathElementTypeARC,             /* only 2D */
    MPIPathElementTypeARC_CENTER,      /* only 2D */
    MPIPathElementTypeARC_END_POINT, /* both 2D and 3D */
    MPIPathElementTypeHELIX,           /* not currently supported */
    MPIPathElementTypeIO,              /* not currently supported */
    MPIPathElementTypeLINE,            /* both 2D and 3D */

    MPIPathElementTypeLAST,
    MPIPathElementTypeFIRST= MPIPathElementTypeINVALID + 1,
    MPIPathElementTypeMASK= 0xFF,
} MPIPathElementType;
```

- **Changes to MPIPathElementAttrMask data type ("path.h" header file)**

```
typedefenum {
        MPIPathElementAttrMaskRELATIVE      = mpiPathElementAttrMaskBIT(MPIPathElementAttrRELATIVE),
        MPIPathElementAttrMaskID            = mpiPathElementAttrMaskBIT(MPIPathElementAttrID),
        MPIPathElementAttrMaskVELOCITY      = mpiPathElementAttrMaskBIT(MPIPathElementAttrVELOCITY),
        MPIPathElementAttrMaskACCEL         = mpiPathElementAttrMaskBIT(MPIPathElementAttrACCEL),
        MPIPathElementAttrMaskTIMESLICE     = mpiPathElementAttrMaskBIT(MPIPathElementAttrTIMESLICE),

        MPIPathElementAttrMaskALL           = -1 << MPIPathElementAttrFIRST,
} MPIPathElementAttrMask;
```

- **Changes to <u>MPIPathElementAttributes</u> data type (“path.h” header file)**

```
typedef struct MPIPathElementAttributes {
        long        id;               /* MPIPathAttrID*/
        double      velocity;         /* MPIPathAttrVELOCITY*/
        double      acceleration;     /* MPIPathAttrACCELERATION*/
        double      timeSlice;        /* MPIPathAttrTIMESLICE*/
} MPIPathElementAttributes;
```

- **Changes to <u>MPIPathMessage</u> data type (“path.h” header file)**

```
typedef enum {
        MPIPathMessageFIRST = mpiMessageID(MPIModuleIdPATH, 0),

        MPIPathMessagePATH_INVALID,
        MPIPathMessageILLEGAL_DIMENSION,
        MPIPathMessageILLEGAL_ELEMENT,
        MPIPathMessageARC_ILLEGAL_DIMENSION,
        MPIPathMessageHELIX_ILLEGAL_DIMENSION,
        MPIPathMessageILLEGAL_RADIUS,
        MPIPathMessagePATH_TOO_LONG,
        MPIPathMessageILLEGAL_VELOCITY,
        MPIPathMessageILLEGAL_ACCELERATION ,
        MPIPathMessageILLEGAL_TIMESLICE ,
        MPIPathMessageINVALID_BLENDING,

        MPIPathMessageLAST
} MPIPathMessage;
```

- **Change to <u>mpiAxisActualVelocity</u> method (“axis.h” header file)**

```
MPI_DECL1 long MPI_DECL2
        mpiAxisActualVelocity(MPIAxis       axis,
                              double        *actual);
```

- **Addition of <u>mpiAxisPositionError</u> method (“axis.h” header file)**

```
MPI_DECL1 long MPI_DECL2
        mpiAxisPositionError(MPIAxis        axis,
                             double         *error);
```

- **Change to <u>MPICommandParams</u> data type (“command.h” header file)**

```
typedef union {
        struct {    /* *'dst' = 'value' */
                    MPICommandAddress       dst;
                    MPICommandConstant      value;
                    MPIControl              control;            /* Ignored by Sequence */
        } assign;
```

- **Change to MPIControlConfig data type ("control.h" header file)**

```
typedef struct MPIControlConfig {
	long	adcCount;
	long	axisCount;
	long	captureCount;
	long	compareCount;
	long	cmdDacCount;
	long	auxDacCount;
	long	filterCount;
	long	motionCount;
	long	motorCount;
	long	recordCount;
	long	sequenceCount;
	long	sercosCount;

	long	sampleRate;

	MPIControlUserIoConfiguserIoConfig;
	MPIControlUserIo	userIo;
} MPIControlConfig;
```

- **Change to MPIControlMessage data type ("control.h" header file)**

```
typedef enum {
	MPIControlMessageFIRST = mpiMessageID(MPIModuleIdCONTROL, 0),
	MPIControlMessageLIBRARY_VERSION,		/* Keep as first control message */
	MPIControlMessageADDRESS_INVALID,
	MPIControlMessageCONTROL_INVALID,
	MPIControlMessageTYPE_INVALID,
	MPIControlMessageINTERRUPTS_DISABLED,
	MPIControlMessageEXTERNAL_MEMORY_OVERFLOW,

	MPIControlMessageLAST
} MPIControlMessage;
```

- **Change to MPIFilterMessage data type ("filter.h" header file)**

```
typedef enum {
	MPIFilterMessageFIRST = mpiMessageID(MPIModuleIdFILTER, 0),

	MPIFilterMessageFILTER_INVALID,
	MPIFilterMessageINVALID_ALGORITHM,

	MPIFilterMessageLAST
} MPIFilterMessage;
```

- **Addition of mpiFilterIntegratorReset method ("filter.h" header file)**

```
MPI_DECL1 long MPI_DECL2
	mpiFilterIntegratorReset(MPIFilter filter);
```

- **Change to MPIMotionType data type ("motion.h" header file)**

```
typedef enum {
	MPIMotionTypeINVALID = -1,

	MPIMotionTypeJOG,

	MPIMotionTypePT,
	MPIMotionTypePVT,
	MPIMotionTypeSPLINE,
	MPIMotionTypeBESSEL,
	MPIMotionTypeBSPLINE,
	MPIMotionTypeBSPLINE2,

	MPIMotionTypeS_CURVE,
	MPIMotionTypeTRAPEZOIDAL,
	MPIMotionTypeS_CURVE_JERK,

	MPIMotionTypeVELOCITY,
	MPIMotionTypeVELOCITY_JERK,

#if 0
	/* Reserved for future use */
	MPIMotionTypeCOORD_ARC,
	MPIMotionTypeCOORD_ARC_FINAL_RADIUS,
	MPIMotionTypeCOORD_HELICAL,
	MPIMotionTypeCOORD_LINEAR,

	MPIMotionTypePARABOLIC,

#endif

	MPIMotionTypeLAST,
	MPIMotionTypeFIRST= MPIMotionTypeINVALID + 1,
	MPIMotionTypeMASK= 0xFF,
} MPIMotionType;
```

- **Addition of MPIMotionSCurveJerk data type ("motion.h" header file)**

```
typedef MPIMotionSCurve          MPIMotionSCurveJerk;
```

- **Change to MPIMotionParams data type ("motion.h" header file)**

```
typedefstruct MPIMotionParams {
	MPIMotionJog			jog;

	MPIMotionPT			pt;
	MPIMotionPVT			pvt;
	MPIMotionSPLINE			spline;
	MPIMotionBESSEL			bessel;
	MPIMotionBSPLINE		bspline;

	MPIMotionSCurve			sCurve;
	MPIMotionSCurve			sCurveJerk;
	MPIMotionTrapezoidal trapezoidal;

	MPIMotionVelocity   velocity;
	MPIMotionVelocity velocityJerk;

	MPIMotionAttributesattributes;

	void		*external;
} MPIMotionParams;
```

- **Deletion of MPIMotorDac data type ("motor.h" header file)**

```
~~#defineMPIMotorDacCountMAX(sizeof(MPIObjectMap) * 8)~~

~~typedef struct MPIMotorDac {~~
    ~~long~~      ~~count;~~
    ~~long~~      ~~number[MPIMotorDacCountMAX];~~
~~} MPIMotorDac;~~
```

- **Change to MPIMotorConfig data type ("motor.h" header file)**

```
typedef struct MPIMotorConfig {
    MPIMotorType        type;

    /* Event configuration, ordered by MPIEventType */
    MPIMotorEventConfigevent[MPIEventTypeMOTOR_LAST];

    long      ampEnablePolarity;          /* FALSE => active lo, else active hi */
    long      encoderPhase;               /* 0 => normal, else reversed */
    long      captureOnChange;            /* 0 => normal, else enabled */

    float     abortDelay;
    float     brakeDelay;
    float     enableDelay;

    MPIObjectMap        filterMap;
    MPIObjectMap        adcMap;
    ~~MPIMotorDac~~         ~~dac;~~

    MPIMotorIo          io;
} MPIMotorConfig;
```

- **Deletion of mpiMotorDacGet method ("motor.h" header file)**

```
~~MPI_DECL1 long MPI_DECL2~~
    ~~mpiMotorDacGet(MPIMotor~~  ~~motor,~~
                  ~~long~~      ~~*count,~~
                  ~~long~~      ~~*number);~~
```

- **Deletion of mpiMotorDacMapGet method ("motor.h" header file)**

```
~~MPI_DECL1 long MPI_DECL2~~
    ~~mpiMotorDacMapGet(MPIMotor~~        ~~motor,~~
                      ~~MPIObjectMap~~    ~~*map);~~
```

- **Deletion of mpiMotorDacSet method ("motor.h" header file)**

```
~~MPI_DECL1 long MPI_DECL2~~
    ~~mpiMotorDacSet(MPIMotor~~  ~~motor,~~
                  ~~long~~      ~~count,~~
                  ~~long~~      ~~*number)~~
```

- **Change to MPIModuleID data type ("mpidef.h" header file)**

```
typedef enum {
	MPIModuleIdINVALID = -1,

	MPIModuleIdMESSAGE,
	MPIModuleIdADC,
	MPIModuleIdAXIS,
	MPIModuleIdCAPTURE,
	MPIModuleIdCOMMAND,
	MPIModuleIdCOMPARE,
	MPIModuleIdCONTROL,
	~~MPIModuleIdDAC,~~
	MPIModuleIdEVENT,
	MPIModuleIdEVENTMGR,
	MPIModuleIdFILTER,
	MPIModuleIdIDN,
	MPIModuleIdIDNLIST,
	MPIModuleIdMOTION,
	MPIModuleIdMOTOR,
	MPIModuleIdNODE,
	MPIModuleIdNOTIFY,
	MPIModuleIdPATH,
	MPIModuleIdPROGRAM,
	MPIModuleIdRECORDER,
	MPIModuleIdSEQUENCE,
	MPIModuleIdSERCOS,

	MPIModuleIdLAST,
	MPIModuleIdFIRST = MPIModuleIdINVALID + 1,

	MPIModuleIdEXTERNAL = 0x80,

	MPIModuleIdMAX = 0xFF
} MPIModuleId;
```

- **Change to MPITrajectory data type ("mpidef.h" header file)**

```
typedef struct MPITrajectory {
	double	velocity;
	double	acceleration;
	double	deceleration;
	double	jerkPercent;
	double	accelerationJerk;
	double	decelerationJerk;
} MPITrajectory;
```

- **Addition of MPIPathParams data type ("path.h" header file)**

```
#if 1
/*
 *     PathConfig deprecated in favor of PathParams.
 *     WARNING: These definitions will eventually be removed.
 */
#defineMPIPathConfig  MPIPathParams
#definempiPathConfigGetmpiPathParamsGet
#definempiPathConfigSetmpiPathParamsSet
#endif

typedef struct MPIPathParams {
       long                          dimension;
       MPIPathPointstart;
       double                        velocity;
       double                        acceleration;
       double                        deceleration;
       MPIMotionTypeinterpolation;
       double                        timeSlice;
       double                        conversion[MPIPathPointDIMENSION_MAX][MPIPathPointDIMENSION_MAX];
} MPIPathParams;
```

- **Deletion of mpiPathConfigGet method ("path.h" header file)**

~~MPI_DECL1 long MPI_DECL2~~
~~mpiPathParamsGet(MPIPath path,~~
~~MPIPathParams *params,~~
~~void *external);~~

- **Deletion of mpiPathConfigSet method ("path.h" header file)**

~~MPI_DECL1 long MPI_DECL2~~
~~mpiPathConfigSet(MPIPath path,~~
~~MPIPathConfig *config,~~
~~void *external);~~

- **Addition of mpiPathParamsGet method ("path.h" header file)**

```
MPI_DECL1 long MPI_DECL2
       mpiPathParamsGet(MPIPath              path,
                        MPIPathParams        *params,
                        void                 *external);
```

- **Addition of mpiPathParamsSet method ("path.h" header file)**

```
MPI_DECL1 long MPI_DECL2
       mpiPathParamsSet(MPIPath              path,
                        MPIPathParams        *params,
                        void                 *external);
```

- **Change to mpiPathParamsSet method ("path.h" header file)**

```
MPI_DECL1 long MPI_DECL2
       mpiPathParamsSet(MPIPath              path,
                        MPIPathParams        *params,
                        void                 *external);
```

- **Change to MPIRecorderADDRESS_COUNT_MAX constant ("recorder.h" header file)**

```
#defineMPIRecorderADDRESS_COUNT_MAX       (128)
```

Changes to MEI/XMP Header Files

- **Change to meiCanCreate method ("can.h" header file)**

```
MPI_DECL1 ~~const~~ MEICan MPI_DECL2
    meiCanCreate(MPIControl    control);
```

- **Change to meiClientCreate method ("client.h" header file)**

```
MPI_DECL1 ~~const~~ MEIClient MPI_DECL2
    meiClientCreate(MPIControl    control);
```

- **Change to meiClientControl method ("client.h" header file)**

```
MPI_DECL1 ~~const~~ MPIControl MPI_DECL2
    meiClientControl(MEIClient    client);
```

- **Change to meiClientPacket method ("client.h" header file)**

```
MPI_DECL1 ~~const~~ MEIPacket MPI_DECL2
    meiClientPacket(MEIClient    client);
```

- **Change to meiElementCreate method ("element.h" header file)**

```
MPI_DECL1 ~~const~~ MEIElement MPI_DECL2
    meiElementCreate(MEIElement   element,
                     void         *object);
```

- **Change to meiFlashCreate method ("flash.h" header file)**

```
MPI_DECL1 ~~const~~ MEIFlash MPI_DECL2
    meiFlashCreate(MPIControl    control);
```

- **Change to meiFlashControl method ("flash.h" header file)**

```
MPI_DECL1 ~~const~~ MPIControl MPI_DECL2
    meiFlashControl(MEIFlash    flash);
```

- **Change to meiListCreate method ("list.h" header file)**

```
MPI_DECL1 ~~const~~ MEIList MPI_DECL2
    meiListCreate(MEIElement    element);
```

- **Change to meiMapCreate method ("map.h" header file)**

```
MPI_DECL1 ~~const~~ MEIMap MPI_DECL2
    meiMapCreate(MPIControl    control,
                 char          *fileName);
```

- **Change to meiMapControl method ("map.h" header file)**

```
MPI_DECL1 ~~const~~ MPIControl MPI_DECL2
    meiMapControl(MEIMap    map);
```

- **Change to meiPacketCreate method ("packet.h" header file)**

```
MPI_DECL1 ~~const~~ MEIPacket MPI_DECL2
    meiPacketCreate(void);
```

- **Change to meiPlatformCreate method ("platform.h" header file)**

```
MPI_DECL1 ~~const~~ MEIPlatform MPI_DECL2
    meiPlatformCreate(MPIControl    control);
```

- **Change to <u>meiServerCreate</u> method ("server.h" header file)**

```
MPI_DECL1 ~~const~~ MEIServer MPI_DECL2
        meiServerCreate(MPIControl              control,
                        MEIPacket               packet,
                        long                    port);
```

- **Change to <u>meiServerControl</u> method ("server.h" header file)**

```
MPI_DECL1 ~~const~~ MPIControl MPI_DECL2
        meiServerControl(MEIServer     server);
```

- **Change to <u>meiServerPacket</u> method ("server.h" header file)**

```
MPI_DECL1 ~~const~~ MEIPacket MPI_DECL2
        meiServerPacket(MEIServer     server);
```

- **Change to <u>meiServerClientPacket</u> method ("server.h" header file)**

```
MPI_DECL1 ~~const~~ MEIPacket MPI_DECL2
        meiServerClientPacket(MEIServer      server);
```

- **Changes to <u>MEIMotorMessage</u> data type ("stdmei.h" header file)**

```
typedef enum {
        MEIMotorMessageABS_ENCODER_FAULT = MPIMotorMessageLAST,
        MEIMotorMessageABS_ENCODER_TIMEOUT,
        MEIMotorMessageMOTOR_NOT_ENABLED,
        MEIMotorMessageSTEPPER_INVALID,
        MEIMotorMessageDISABLE_ACTION_INVALID,

        MEIMotorMessageLAST
} MEIMotorMessage;
```

- **Addition of <u>MEIMotorDisableAction</u> data type ("stdmei.h" header file)**

```
typedef enum MEIMotorDisableAction {
        MEIMotorDisableActionINVALID = -1,

        MEIMotorDisableActionNONE,
        MEIMotorDisableActionCMD_EQ_ACT,

        MEIMotorDisableActionLAST,
        MEIMotorDisableActionFIRST = MEIMotorDisableActionINVALID + 1,
} MEIMotorDisableAction;
```

- **Changes to <u>MEIMotorConfig</u> data type ("stdmei.h" header file)**

```
typedef struct MEIMotorConfig {
        MEIMotorEncoder             Encoder[MEIXmpMotorEncoders];
        MEIXmpIO                    StatusOutput[MEIXmpMotorStatusOutputs];


        MEIMotorTransceiver          Transceiver[MEIXmpMotorTransceivers];
        MEIMotorTransceiver          TransceiverExtended[MEIXmpMotorTransceiversExtended];
        long                        UserOutInvert;        /* Opto Polarity */
        MEIMotorStepper             Stepper;
        long                        EncoderTermination;
        long                        SIM4;
        MEIMotorDacConfig            Dac;

        long                         pulseEnable; /* 0 => normal, else pulse output */
        long                                    pulseWidth;  /* 0.1 to 25.5 microseconds */
        ~~* Commutation is read only from field Theta to end*/~~
        ~~MEIXmpCommutationBlockCommutation;~~
        ~~MEIXmpLimitDataLimit[MEIXmpLimitLAST];~~
        ~~MEIXmpMotorTorqueLimitConfig TorqueLimitConfig;~~
        ~~long AmpDisableWithLSR;/* TRUE => XMP disables amp when LSR is active */~~

        MEIXmpCommutationBlock         Commutation; /* read-only from field Theta to end*/
        MEIXmpLimitData                Limit[MEIXmpLimitLAST];
        MEIXmpMotorTorqueLimitConfig TorqueLimitConfig;
        MEIMotorDisableAction          disableAction;
        long                           AmpDisableWithLSR;/* TRUE= XMP disables amp when LSR is active */
        MEIMotorFilterInput            FilterInput[MEIXmpMotorFilterInputs];
} MEIMotorConfig;
```

- **Addition of <u>meiPlatformAssertSet</u> method ("platform.h" header file)**

```
MPI_DEF1 void MPI_DEF2
        meiPlatformAssertSet(void (MPI_DECL2 *func) (const char *file, long line));
```

- **Changes to <u>MEIRemoteMethod</u> data type ("remote.h" header file)**

```
typedef enum {
        MEIRemoteMethodINVALID = -1,

        MEIRemoteMethodBOARD_TYPE,
        MEIRemoteMethodBOARD_INFO_GET,
        MEIRemoteMethodBOARD_INFO_SET,

        MEIRemoteMethodFLASH_MEMORY_GET,
        MEIRemoteMethodFLASH_MEMORY_SET,

        MEIRemoteMethodINTERRUPT_ENABLE,
        MEIRemoteMethodINTERRUPT_WAIT,
        MEIRemoteMethodINTERRUPT_WAKE,

        MEIRemoteMethodMEMORY,

        MEIRemoteMethodMEMORY_GET,
        MEIRemoteMethodMEMORY_SET,

        MEIRemoteMethodPORT_IN_CHAR,
        MEIRemoteMethodPORT_OUT_CHAR,


        MEIRemoteMethodOBJECT_LOCK_GIVE,
        MEIRemoteMethodOBJECT_LOCK_TAKE,

        MEIRemoteMethodRESET,

        MEIRemoteMethodLAST,
        MEIRemoteMethodFIRST = MEIRemoteMethodINVALID + 1
} MEIRemoteMethod;
```

- **Addition of <u>MEIRemoteMethodPortInChar</u> data type ("remote.h" header file)**

```
typedef struct MEIRemoteMethodPortInChar {
        unsigned     short   port;
        unsigned     char    *value;
} MEIRemoteMethodPortInChar;
```

- **Addition of <u>MEIRemoteMethodPortOutChar</u> data type ("remote.h" header file)**

```
typedef struct MEIRemoteMethodPortOutChar {
	unsigned	short	port;
	unsigned	char	value;
} MEIRemoteMethodPortOutChar;
```

- **Changes to <u>MEIRemoteMethodArgs</u> data type ("remote.h" header file)**

```
typedef union {
	MEIPlatformBoardType*boardType;
	MEIRemoteMethodBoardInfoGetboardInfoGet;
	MEIRemoteMethodBoardInfoSetboardInfoSet;
	union {
			long					enable;
			MEIRemoteMethodInterruptWait		wait;
	} interruptArgs;
	MEIRemoteMethodMemory			memory;
	MEIRemoteMethodMemoryGet		memoryGet;
	MEIRemoteMethodMemorySet		memorySet;
	MEIRemoteMethodPortInChar		portIn;
	MEIRemoteMethodPortOutChar		portOut;
	MEIRemoteMethodObjectLockGive		objectLockGive;
	MEIRemoteMethodObjectLockTake		objectLockTake;
} MEIRemoteMethodArgs;
```

- **Changes to <u>MEIRemoteHeader</u> data type ("remote.h" header file)**

```
typedef struct MEIRemoteHeader {
	unsigned longsize;	/* bytes */
	unsigned longsequence;

	MEIRemoteTypetype;
	union {
			struct {
					MEIRemoteMethod		method;
					MEIRemoteMethodArgs	args;
			} query;
			struct {
					long		returnValue;
					union {
							MEIPlatformBoardType	boardType;	/* meiPlatformBoardType() */
							unsigned		char		portValue;
							long			interrupted;	/* mpiControlInterruptWait() */
							struct {
									void	*firmware;
									void	*external;
							} memory;		/* mpiControlMemory() */
					} output;
			} reply;
	} asa;

	unsigned longmemory;/* mpiControlMemory[GS]et(count == 4) */
} MEIRemoteHeader;
```

- **Changes to <u># defines</u> ("xmp.h" header file)**

```
/* #defines and enums */

#define MEIXmpVERSION			358
			/* version, 200 = 2.00 */
#define MEIXmpOPTION			0
			/* FPGA Revision Number */
#define MEIXmpFPGAMBREV			242
#define MEIXmpFPGASIM4REV		211
```

• Changes to MEIXmpEvent data type ("xmp.h" header file)

```
typedef enum {
	MEIXmpEventIN_COARSE_POSITION		= 0,
	MEIXmpEventAT_TARGET,
	MEIXmpEventAT_VELOCITY,
	MEIXmpEventIN_FINE_POSITION,
	MEIXmpEventDONE,
	MEIXmpEventPPI,
	MEIXmpEventPS_FAULT,
	MEIXmpEventMS_FAULT,
	MEIXmpEventOUT_OF_FRAMES,
	MEIXmpEventEXTERNAL,
	MEIXmpEventFRAME,
	MEIXmpEventRESET,
	MEIXmpEventRESUME,
	MEIXmpEventPAUSE,
	MEIXmpEventSTOP,
	MEIXmpEventESTOP,
	MEIXmpEventESTOP_ABORT,
	MEIXmpEventABORT,
	MEIXmpEventHOST,
	MEIXmpEventRESERVED0,
	MEIXmpEventRESERVED1,
	MEIXmpEventRESERVED2,
	MEIXmpEventREC_IDLE,
	MEIXmpEventREC_FULL,
	MEIXmpEventREC_RUNNING,
	MEIXmpEventLIMIT			= 31,

} MEIXmpEvent;
```

• Changes to MEIXmpStatus data type ("xmp.h" header file)

```
typedef enum {
	MEIXmpStatusIN_COARSE_POSITION		= (1 << MEIXmpEventIN_COARSE_POSITION),	/* 0x00000001 */
	MEIXmpStatusAT_TARGET			= (1 << MEIXmpEventAT_TARGET),		/* 0x00000002 */
	MEIXmpStatusAT_VELOCITY			= (1 << MEIXmpEventAT_VELOCITY),		/* 0x00000004 */
	MEIXmpStatusIN_FINE_POSITION		= (1 << MEIXmpEventIN_FINE_POSITION),	/* 0x00000008 */

	MEIXmpStatusSETTLED			= MEIXmpStatusIN_FINE_POSITION,		/* 0x00000008 */
	MEIXmpStatusDONE_MASK			= (MEIXmpStatusSETTLED |			/* 0x0000000A */
						  MEIXmpStatusAT_TARGET),

	MEIXmpStatusDONE			= (1 << MEIXmpEventDONE),			/* 0x00000010 */
	MEIXmpStatusIN_FINE_POSITION_LATCHED	= MEIXmpStatusDONE,			/* 0x00000010 */
	MEIXmpStatusPPI				= (1 <<    MEIXmpEventPPI),			/* 0x00000020 */
	MEIXmpStatusPS_FAULT			= (1 << MEIXmpEventPS_FAULT),		/* 0x00000040 */
	MEIXmpStatusMS_FAULT			= (1 << MEIXmpEventMS_FAULT),		/* 0x00000080 */
	MEIXmpStatusOUT_OF_FRAMES		= (1 << MEIXmpEventOUT_OF_FRAMES),	/* 0x00000100 */
	MEIXmpStatusEXTERNAL			= (1 << MEIXmpEventEXTERNAL),		/* 0x00000200 */
	MEIXmpStatusFRAME			= (1 << MEIXmpEventFRAME),		/* 0x00000400 */
	MEIXmpStatusRESET			= (1 << MEIXmpEventRESET),		/* 0x00000800 */
	MEIXmpStatusRESUME			= (1 << MEIXmpEventRESUME),		/* 0x00001000 */

	MEIXmpStatusPAUSE			= (1 << MEIXmpEventPAUSE),		/* 0x00002000 */
	MEIXmpStatusSTOP			= (1 << MEIXmpEventSTOP),			/* 0x00004000 */
	MEIXmpStatusESTOP			= (1 << MEIXmpEventESTOP),		/* 0x00008000 */
	MEIXmpStatusESTOP_ABORT			= (1 << MEIXmpEventESTOP_ABORT),		/* 0x00010000 */
	MEIXmpStatusABORT			= (1 << MEIXmpEventABORT),		/* 0x00020000 */

	MEIXmpStatusERROR_MASK			= (MEIXmpStatusESTOP |			/* 0x00038000 */
                                                          MEIXmpStatusESTOP_ABORT |
                                                          MEIXmpStatusABORT),

        MEIXmpStatusHOST                        = (1 <<  MEIXmpEventHOST),              /* 0x00040000 */

        MEIXmpStatusRESERVED0                   = (1 << MEIXmpEventRESERVED0),          /* 0x00080000 */
        MEIXmpStatusRESERVED1                   = (1 << MEIXmpEventRESERVED1),          /* 0x00100000 */
        MEIXmpStatusRESERVED2                   = (1 << MEIXmpEventRESERVED2),          /* 0x00200000 */

        MEIXmpStatusRESERVED_MASK               = (MEIXmpStatusRESERVED0 |              /* 0x00380000 */
                                                   MEIXmpStatusRESERVED1 |
                                                  MEIXmpStatusRESERVED2),

        MEIXmpStatusREC_IDLE            = (1 << MEIXmpEventREC_IDLE),                 /* 0x00400000 Recorder Status */
        MEIXmpStatusREC_FULL            = (1 << MEIXmpEventREC_FULL),                  /* 0x00800000 */
        MEIXmpStatusREC_RUNNING         = (1 << MEIXmpEventREC_RUNNING),               /* 0x01000000 */
        MEIXmpStatusLIMIT               = ((long)((unsigned long)1 << MEIXmpEventLIMIT)),  /* 0x80000000 */

        /* for backward compatibility */
        MEIXmpStatusID_USER0            = MEIXmpStatusLIMIT,                        /* 0x80000000 */
        MEIXmpStatusID_USER1            = MEIXmpStatusLIMIT,                        /* 0x80000000 */
        MEIXmpStatusID_USER2            = MEIXmpStatusLIMIT,                        /* 0x80000000 */
        MEIXmpStatusID_USER3            = MEIXmpStatusLIMIT,                        /* 0x80000000 */
        MEIXmpStatusID_USER4            = MEIXmpStatusLIMIT,                        /* 0x80000000 */
        MEIXmpStatusID_USER5            = MEIXmpStatusLIMIT,                        /* 0x80000000 */
        MEIXmpStatusID_USER6            = MEIXmpStatusLIMIT,                        /* 0x80000000 */
        MEIXmpStatusID_USER7            = MEIXmpStatusLIMIT,                        /* 0x80000000 */

        MEIXmpAxisSTATUS_LATCH          = (MEIXmpStatusERROR_MASK |                 /* 0x803FC010 */
                                          MEIXmpStatusLIMIT |
                                          MEIXmpStatusHOST |
                                          MEIXmpStatusRESERVED_MASK |
                                          MEIXmpStatusSTOP |
                                          MEIXmpStatusDONE),

        MEIXmpStatusMOTION              = (MEIXmpStatusDONE_MASK |                  /* 0x0020001F */
        ~~MEIXmpAxisLATCH_MASK~~            ~~= (MEIXmpAxisSTATE_LATCH |~~
                                        ~~MEIXmpAxisSTATUS_LATCH),~~

        ~~MEIXmpStatusMOTION~~              ~~= (MEIXmpStatusDONE_MASK |~~                  ~~/* 0x0010001D */~~

MEIXmpStatusIN_COARSE_POSITION |
                                        MEIXmpStatusAT_VELOCITY |
                                        MEIXmpStatusRESERVED2 |
                                        MEIXmpStatusDONE),

        MEIXmpMS_OR_MASK                = (MEIXmpStatusERROR_MASK |                 /* 0x801FEC20 */
                                          MEIXmpStatusLIMIT |
                                          MEIXmpStatusHOST |
                                          MEIXmpStatusRESERVED0 |
                                          MEIXmpStatusRESERVED1 |
                                          MEIXmpStatusPAUSE |
                                          MEIXmpStatusSTOP |
                                          MEIXmpStatusPPI |
                                          MEIXmpStatusFRAME |
                                          MEIXmpStatusRESET),

        MEIXmpMS_AND_MASK               =  MEIXmpStatusMOTION,                      /* 0x0020001F */

        MEIXmpMSAxisMASK                = (MEIXmpAxisSTATUS_LATCH |
                                          MEIXmpStatusPAUSE),                       /* 0x802FE010 */

} MEIXmpStatus;
```

- **Deletion of # defines (“xmp.h” header file)**

```
~~/* flags used to modify state machine */~~
~~#define MEIXmpStateFlags~~                ~~MEIXmpStatus~~

~~#define MEIXmpFlagDONE~~                  ~~MEIXmpStatusDONE~~
~~#define MEIXmpFlagRESET~~                 ~~MEIXmpStatusRESET~~
~~#define MEIXmpFlagSTOP~~                  ~~MEIXmpStatusSTOP~~
~~#define MEIXmpFlagESTOP~~                 ~~MEIXmpStatusESTOP~~
~~#define MEIXmpFlagESTOP_ABORT~~           ~~MEIXmpStatusESTOP_ABORT~~
~~#define MEIXmpFlagABORT~~                 ~~MEIXmpStatusABORT~~
~~#defineMEIXmpFlagERROR_MASK~~            ~~MEIXmpStatusERROR_MASK~~
~~#define MEIXmpFlagHOST~~                  ~~MEIXmpStatusHOST~~
```

- **Changes to MEIXmpMotionType data type (“xmp.h” header file)**

```
typedef enum {
    MEIXmpMotionTypeINVALID                 = -1,

    MEIXmpMotionTypeNONE,
    ~~MEIXmpMotionTypeUPDATE,~~

    ~~MEIXmpMotionTypeSTART,~~
    ~~MEIXmpMotionTypeMODIFY_ID,~~
    ~~MEIXmpMotionTypeID,~~

    MEIXmpMotionTypeHOLD,
    MEIXmpMotionTypeOUTPUT,
    MEIXmpMotionTypeJOG,

    MEIXmpMotionTypeVELOCITY,
    MEIXmpMotionTypeVELOCITY_JERK,
    MEIXmpMotionTypeS_CURVE,
    MEIXmpMotionTypeS_CURVE_JERK,

    MEIXmpMotionTypePATH_END                = MEIXmpMotionTypeS_CURVE,
    MEIXmpMotionTypePATH_OPEN,

    MEIXmpMotionTypeLAST,
    MEIXmpMotionTypeFIRST                   = MEIXmpMotionTypeINVALID + 1,

} MEIXmpMotionType;
```

- **Addition of MEIXmpFrameType data type (“xmp.h” header file)**

```
typedef enum {
    MEIXmpFrameTypeINVALID                  = -1,

    MEIXmpFrameTypeNONE,
    MEIXmpFrameTypeUPDATE,

    MEIXmpFrameTypeSTART,
    MEIXmpFrameTypeMODIFY_ID,
    MEIXmpFrameTypeID,

    MEIXmpFrameTypeHOLD,
    MEIXmpFrameTypeOUTPUT,

    MEIXmpFrameTypeVELOCITY,
    MEIXmpFrameTypeS_CURVE,

    MEIXmpFrameTypeLAST,
    MEIXmpFrameTypeFIRST                = MEIXmpFrameTypeINVALID + 1,

} MEIXmpFrameType;
```

- **Changes to MEIXmpFrameStatus data type ("xmp.h" header file)**

```
typedef enum {
    MEIXmpFrameStatusDIRECTION          = 0x00000001,
    MEIXmpFrameStatusTARGET             = 0x00000002,
    MEIXmpFrameStatusVELOCITY_MOVE      = 0x00010000,    /* assumed S_CURVE move if not set */
    MEIXmpFrameStatusMOVE_IN_PROGRESS   = 0x80000000,
} MEIXmpFrameStatus;
```

- **Changes to MEIXmpFrame data type ("xmp.h" header file)**

```
typedef struct MEIXmpFrame {
    MEIXmpFrameType         Type;
    float                   t;
    long                    Control;
    long                    Position;
    float                   Velocity;
    float                   Accel;
    float                   Jerk;
    long                    Reserved;
} MEIXmpFrame;
```

- **Changes to MEIXmpStartFrame data type ("xmp.h" header file)**

```
typedef struct MEIXmpStartFrame {
    MEIXmpFrameType         Type;
    float                   t;
    long                    Control;
    long                    Position;
    long                    MoveID;
    long                    ElementID;
    float                   MoveTime;
    MEIXmpFrameStatus       Status;
} MEIXmpStartFrame;
```

- **Changes to MEIXmpIOFrame data type ("xmp.h" header file)**

```
typedef struct MEIXmpIOFrame {
    MEIXmpFrameType         Type;
    float                   t;
    long                    Control;
    long                    *Ptr;
    long                    Mask;
    long                    Pattern;
} MEIXmpIOFrame;
```

- **Changes to MEIXmpIDFrame data type ("xmp.h" header file)**

```
typedef struct MEIXmpIDFrame {
    MEIXmpFrameType         Type;
    long                    Sample;
    long                    Control;
    long                    Position;
    long                    MoveID;
    long                    ElementID;
    float                   MoveTime;
    MEIXmpFrameStatus       Status;
} MEIXmpIDFrame;
```

- **Changes to MEIXmpAxis data type ("xmp.h" header file)**

```
typedef struct {
	MEIXmpLink			*Link;
	MEIXmpMotionSupervisor		*MS;
	MEIXmpPosInput			APos[MEIXmpAxisPosInputs];
	long				ActualPosition;
	long				CommandPosition;
	long				TargetPosition;
	float				TargetVelocity;
	long				Origin;
	long				Compensation;
	float				ActualVelocity;
	float				CommandVelocity;
	float				PositionError;
	float				FinePosTolerance;
	long				CoarsePosTolerance;
	float				VelTolerance;
	long				SettlingTime;
	long				SettlingCount;
	MEIXmpStatus			SettlingMask;
	MEIXmpStatus			Status;
	~~MEIXmpStatus			StateFlags;~~
	MEIXmpState			State;
	long				MoveStatus;
	MEIXmpMetrics			Metric;
	long				ModifyIndex;
	float				ModifyTime;
	MEIXmpTrajectoryCalculator	TC;
	MEIXmpAxisGear			Gear;
	long				MoveID;
	long				ElementID;
	MEIXmpHostSignal		Signal;
} MEIXmpAxis;
```

- **Addition of can.h header file** Not Supported (reserved for future use).
- **Addition of xmpcan.h header file** Not Supported (reserved for future use).
- **Deletion of eeprom.h header file**
- **Deletion of pci.h header file**
- **Deletion of pciUtil.h header file**

- **Changes to MEIMapGroup data type ("map.h" header file)**

```
typedef enum  {
	MEIMapGroupINVALID = -1,
	MEIMapGroupMPI_ADC_CONFIG,		/* MEIMapGroupCONFIG_FIRST */
	MEIMapGroupMEI_ADC_CONFIG,
	MEIMapGroupMPI_AXIS_CONFIG,
	MEIMapGroupMEI_AXIS_CONFIG,
	MEIMapGroupMEI_CAN_CONFIG,
	MEIMapGroupMPI_CAPTURE_CONFIG,
	MEIMapGroupMEI_CAPTURE_CONFIG,
	MEIMapGroupMPI_CONTROL_CONFIG,
	MEIMapGroupMEI_CONTROL_CONFIG,
	MEIMapGroupMPI_EVENTMGR_CONFIG,
	MEIMapGroupMEI_EVENTMGR_CONFIG,
	MEIMapGroupMPI_FILTER_CONFIG,
	MEIMapGroupMEI_FILTER_CONFIG,
        MEIMapGroupMPI_MOTION_CONFIG,
        MEIMapGroupMEI_MOTION_CONFIG,
        MEIMapGroupMPI_MOTOR_CONFIG,
        MEIMapGroupMEI_MOTOR_CONFIG,
        MEIMapGroupMPI_NODE_CONFIG,
        MEIMapGroupMEI_NODE_CONFIG,
        MEIMapGroupMPI_RECORDER_CONFIG,
        MEIMapGroupMEI_RECORDER_CONFIG,
        MEIMapGroupMPI_SEQUENCE_CONFIG,
        MEIMapGroupMEI_SEQUENCE_CONFIG,
        MEIMapGroupMPI_SERCOS_CONFIG,
        MEIMapGroupMEI_SERCOS_CONFIG,           /* MEIMapGroupCONFIG_LAST */
        MEIMapGroupMEI_XMP_BUFFER_DATA,         /* MEIMapGroupXMP_FIRST */
        MEIMapGroupMEI_XMP_DATA,
        MEIMapGroupMEI_XMP_RIPTIDE_DATA,
        MEIMapGroupMEI_XMP_PLD,
        MEIMapGroupMEI_XMP_SERCON,
        MEIMapGroupMEI_XMP_CAN,
        MEIMapGroupMEI_XMP_FRAME_BUFFER,
        MEIMapGroupMEI_XMP_RECORD_BUFFER,
        MEIMapGroupMEI_XMP_SERCOS_BUFFER,
                                                /* MEIMapGroupXMP_LAST */
        MEIMapGroupLAST,
        MEIMapGroupFIRST = MEIMapGroupINVALID + 1,

        MEIMapGroupCONFIG_FIRST = MEIMapGroupMPI_ADC_CONFIG,
        MEIMapGroupCONFIG_LAST= MEIMapGroupMEI_SERCOS_CONFIG + 1,

        MEIMapGroupXMP_FIRST= MEIMapGroupMEI_XMP_BUFFER_DATA,
        MEIMapGroupXMP_LAST= MEIMapGroupMEI_XMP_SERCON + 1
} MEIMapGroup;
```

- **Changes to MEIModuleId data type ("meidef.h" header file)**

```
typedef enum {
        MEIModuleIdPLATFORM = MPIModuleIdEXTERNAL,

        MEIModuleIdCAN,
        MEIModuleIdCLIENT,
        MEIModuleIdELEMENT,
        MEIModuleIdFLASH,
        MEIModuleIdLIST,
        MEIModuleIdMAP,
        MEIModuleIdPACKET,
        MEIModuleIdSERVER,

        MEIModuleIdLAST,
        MEIModuleIdFIRST = MPIModuleIdEXTERNAL
} MEIModuleId;
```

- **Changes to MEIPlatformBoardInfo data type ("platform.h" header file)**

```
typedef struct MEIPlatformBoardInfo {
        long        reserved;
        struct {
                long        revision;
                long        lower;
                long        upper;
        } manufacturer;
        long        serialNumber;
        long        userId;
        long        boardId;
        long        socketInfo[5];
} MEIPlatformBoardInfo;
```

- **Changes to MEIPlatformBoardInfo data type ("platform.h" header file)**

```
typedef struct MEIPlatformBoardInfo {
	long		reserved;
	struct {
			long		revision;
			long		lower;
			long		upper;
	} manufacturer;
```

- **Changes to # includes ("stdmei.h" header file)**

```
#include "meidef.h"

#include "can.h"
#include "client.h"
#include "element.h"
#include "firmware.h"
#include "flash.h"
#include "list.h"
#include "map.h"
#include "packet.h"
#include "platform.h"
#include "remote.h"
#include "server.h"
#include "xmpcan.h"

#if defined(__cplusplus)
extern "C" {
#endif
```

- **Changes to MEIControlVersion data type ("stdmei.h" header file)**

```
typedef struct MEIControlVersion {
	struct {	/* control.c */
			char		*version;			/* MEIControlVersionMPI (YYYYMMDD) */

			struct {	/* xmp.h */
					long		version;	/* MEIXmpVERSION */
					long		option;		/* MEIXmpOPTION */
			} firmware;
	} mpi;

	struct {
			long		version;			/* hardware version */

			struct {	/* MEIXmpData.SystemData{} */
					long		version;			/* MEIXmpVERSION_EXTRACT(SoftwareID) */
					char		revision;			/* ('A' - 1) + MEIXmpREVISION_EXTRACT(SoftwareID) */
					long		subRevision;		/* MEIXmpSUB_REV_EXTRACT(Option) */
					long		developmentId;		/* MEIXmpDEVELOPMENT_ID_EXTRACT(Option) */
					long		option;				/* MEIXmpOPTION_EXTRACT(Option) */
					long		userVersion;
					long		branchId;
			} firmware;

			struct		{
					long		FPGA[MEIXmpFPGAsPerBlock];
			} motionBlock[MEIXmpMaxMotionBlocks];

			struct {
```

```
                        struct      {
                                    long        version;
                                    long        option;
                        } busInterface;
                } board[MEIXmpMaxBoards];
        } xmp;
} MEIControlVersion;
```

- **Changes to <u>MEIControlMessage</u> data type ("stdmei.h" header file)**

```
typedef enum {
        MEIControlMessageFIRMWARE_INVALID = MPIControlMessageLAST,
        MEIControlMessageFIRMWARE_VERSION_NONE,
        MEIControlMessageFIRMWARE_VERSION,
        MEIControlMessageSOCKETS,
        MEIControlMessageBAD_SOCKET_DATA,
        MEIControlMessageNO_SOCKET,

        MEIControlMessageLAST
} MEIControlMessage;
```

- **Addition of <u>meiControlMemoryToFile</u> method ("stdmei.h" header file)**

```
MPI_DECL1 long MPI_DECL2
        meiControlMemoryToFile(MPIControl       control,
                                char            *fileName);
```

- **Addition of <u>meiControlSampleRate</u> method ("stdmei.h" header file)**

```
/* Returns the controller's sampleRate */
MPI_DECL1 long MPI_DECL2
         meiControlSampleRate(MPIControl        control,
                               double           *sampleRate);
```

- **Addition of <u>meiMapFileLoadVerify</u> method ("stdmei.h" header file)**

```
MPI_DECL1 long MPI_DECL2
        meiMapFileLoadVerify(MEIMap             map);
```

- **Deletion of <u>MEIFilterGainStep</u> data type ("stdmei.h" header file)**

~~typedefstruct MEIFilterGainStep {~~
~~long stepCommand;~~
~~} MEIFilterGainStep;~~

- **Deletion of <u>MEIFilterGainStepCoeff</u> data type ("stdmei.h" header file)**

~~typedefenum {~~
~~MEIFilterGainStepCoeffINVALID = -1,~~

~~MEIFilterGainStepCoeffSTEP_COMMAND,/* Step Command */~~

~~MEIFilterGainStepCoeffLAST,~~
~~MEIFilterGainStepCoeffFIRST = MEIFilterGainStepCoeffINVALID + 1~~
~~} MEIFilterGainStepCoeff;~~

- **Addition of # defines ("stdmei.h" header file)**

```
/* Filter -- Postfilter */

#define MEIPi           (3.14159265358979323846264338327950288 4)
#defineMEI2Pi           (6.28318530717958647692528676655900576 8)
#define MEIRoot2        (1.41421356237309504880168872420 97)
#define MEIe            (2.71828182845904523536028747135 27)
```

- **Addition of <u>MEIFilterType</u> data type ("stdmei.h" header file)**

```
typedef enum {
	MEIFilterTypeINVALID = -1,

	MEIFilterTypeUNITY_GAIN, /* B0 = 1    B1=B2=A1=A2 = 0    (effectively acting as no filter) */
	MEIFilterTypeBIQUAD,
	MEIFilterTypeSINGLE_ORDER,
	MEIFilterTypeLOW_PASS,
	MEIFilterTypeHIGH_PASS,
	MEIFilterTypeNOTCH,
	MEIFilterTypeRESONATOR,
	MEIFilterTypeLEAD_LAG,

	MEIFilterTypeZERO_GAIN,/* b0=b1=b2=a1=a2 = 0   (this does act as a filter.... zeroing the output) */

	MEIFilterTypeIIR,

	MEIFilterTypeLAST,
	MEIFilterTypeFIRST = MEIFilterTypeINVALID + 1,

} MEIFilterType;
```

- **Addition of # defines ("stdmei.h" header file)**

```
#define MEIMaxBiQuadSections     (6)
```

- **Addition of <u>MEIPostfilterSection</u> data type ("stdmei.h" header file)**

```
typedef struct MEIPostfilterSection {

	MEIFilterTypetype;

	union {

		struct {
			double breakPoint;			/* Hz */
		} lowPass;

		struct {
			double breakPoint;			/* Hz */
		} highPass;

		struct {
			double centerFrequency;		/* Hz */
			double bandwidth;			/* Hz */
		} notch;

		struct {
			double centerFrequency;		/* Hz */
			double bandwidth;			/* Hz */
			double gain;				/* dB */
		} resonator;

		struct {
			double lowFrequencyGain;	/* dB */
			double highFrequencyGain;	/* dB */
			double centerFrequency;		/* Hz */
		} leadLag;
```

```
                /* Analog coefficients */
                struct {
                        double a1;
                        double a2;
                        double b0;
                        double b1;
                        double b2;
                } biquad;

                struct {
                        long numberOfCoefficients;
                        double coeff[MEIXmpMAX_PostFilterSize];
                } iir;

        } data;

} MEIPostfilterSection;
```

- **Addition of <u>meiFilterPostfilterGet</u> method ("stdmei.h" header file)**

```
/* Get all sections of a postfilter */
MPI_DECL1 long MPI_DECL2
    meiFilterPostfilterGet(MPIFilter                filter,
                           long                     *sectionCount,
                           MEIPostfilterSection     *sections);
```

- **Addition of <u>meiFilterPostfilterSet</u> method ("stdmei.h" header file)**

```
/* Set multiple sections of a postfilter.  Sections 0 through (numberOfSections-1) */
MPI_DECL1 long MPI_DECL2
    meiFilterPostfilterSet(MPIFilter                filter,
                           long                     sectionCount,
                           MEIPostfilterSection     *sections);
```

- **Addition of <u>meiFilterPostfilterSectionGet</u> method ("stdmei.h" header file)**

```
/* Get a section of a postfilter */
MPI_DECL1 long MPI_DECL2
    meiFilterPostfilterSectionGet(MPIFilter                filter,
                                  long                     sectionNumber,
                                  MEIPostfilterSection     *section);
```

- **Addition of <u>meiFilterPostfilterSectionSet</u> method ("stdmei.h" header file)**

```
/* Set a section of a postfilter */
MPI_DECL1 long MPI_DECL2
    meiFilterPostfilterSectionSet(MPIFilter                filter,
                                  long                     sectionNumber,
                                  MEIPostfilterSection     *section);
```

- **Changes to <u>MEIMotionMessage</u> data type ("stdmei.h" header file)**

```
typedef enum {
    MEIMotionMessageRESERVED0 = MPIMotionMessageLAST,
    MEIMotionMessageRESERVED1,
    MEIMotionMessageRESERVED2,
    MEIMotionMessageNO_AXES_MAPPED,

    MEIMotionMessageLAST
} MEIMotionMessage;
```

- **Changes to meiMotionFrameBufferLoad method ("stdmei.h" header file)**

```
MPI_DECL1 long MPI_DECL2
    meiMotionFrameBufferLoad(MPIMotionmotion,
                             long    initial,        /* TRUE/FALSE */
                             long    lock,           /* TRUE/FALSE */
                             long    frameLowEvent); /* TRUE/FALSE */
```

- **Changes to MEIMotorConfig data type ("stdmei.h" header file)**

```
typedef struct MEIMotorConfig {
    MEIMotorEncoder         Encoder[MEIXmpMotorEncoders];
    MEIXmpIO                StatusOutput[MEIXmpMotorStatusOutputs];

    MEIMotorTransceiver     Transceiver[MEIXmpMotorTransceivers];
    MEIMotorTransceiver     TransceiverExtended[MEIXmpMotorTransceiversExtended];
    long                    UserOutInvert;    /* Opto Polarity */
    MEIMotorStepper         Stepper;
    long                    EncoderTermination;
    long                    SIM4;
    MEIMotorDacConfig       Dac;

    long    pulseEnable;    /* 0 => normal, else pulse output */
    long    pulseWidth;     /* 0.1 to 25.5 microseconds */

    /* Commutation is read-only from field Theta to end*/
    MEIXmpCommutationBlockCommutation;

    MEIXmpLimitDataLimit[MEIXmpLimitLAST];

    MEIXmpMotorTorqueLimitConfig TorqueLimitConfig;

    long AmpDisableWithLSR;/* TRUE => XMP disables amp when LSR is active */

    MEIMotorFilterInputFilterInput[MEIXmpMotorFilterInputs];
} MEIMotorConfig;
```

- **Deletion of MEITraceParams data type ("trace.h" header file)**

~~typedef enum {~~
~~MEITraceParamsINVALID = -1,~~

~~MEITraceParamsMOTION_START = 2,~~
~~MEITraceParamsMOTION_MODIFY = 3,~~
~~MEITraceParamsEVENT = 4,~~

~~MEITraceParamsLAST,~~
~~MEITraceParamsFIRST = MEITraceParamsINVALID - 1,~~
~~} MEITraceParams;~~

- **Changes to MEITraceMaskGLOBAL data type ("trace.h" header file)**

```
typedef long (*MEITraceFunction)(const char *buffer);
extern MEITraceMask
    MEITraceMaskGLOBAL;
```

- **Changes to #defines ("xmp.h" header file)**

```
/* #defines and enums */

#define MEIXmpVERSION           347
                /* version, 200 = 2.00 */
#define MEIXmpOPTION            0
```

```
               /* FPGA Revision Number */
#define MEIXmpFPGAMBREV       241
#define MEIXmpFPGASIM4REV     211


#define MEIXmpMAXCompDimensions          (2)
#define MEIXmpCompTableSize              (512)
#defineMEIXmpMAX_COORD_AXES              (16)
#define MEIXmpMAX_MESSAGES               (128)
#define MEIXmpMaxLatches                 (10)
#define MEIXmpMaxComparePositions        (10)


#define MEIXmpMaxRecSize                 (32)
#define MEIXmpMaxCollectionSize          (8)
#define MEIXmpPointBufferSize            (MEIXmpMAX_MSs * MEIXmpMAX_Axes)
#define MEIXmpDFilterSize                (8)
#define MEIXmpMaxGainTables              (4)
```

- **Changes to MEIXmpStatus data type ("xmp.h" header file)**

```
typedef enum {
    MEIXmpStatusIN_COARSE_POSITION= (1 <<MEIXmpEventIN_COARSE_POSITION),/* 0x00000001 */
    MEIXmpStatusAT_TARGET           = (1 <<    MEIXmpEventAT_TARGET),   /* 0x00000002 */
    MEIXmpStatusAT_VELOCITY         = (1 <<    MEIXmpEventAT_VELOCITY),/* 0x00000004 */
    MEIXmpStatusIN_FINE_POSITION    = (1 <<    MEIXmpEventIN_FINE_POSITION),/* 0x00000008 */

    MEIXmpStatusSETTLED             = MEIXmpStatusIN_FINE_POSITION,  /* 0x00000008 */
    MEIXmpStatusDONE_MASK                   = (MEIXmpStatusSETTLED |                /* 0x0000000A */
                                             MEIXmpStatusAT_TARGET),

    MEIXmpStatusDONE                        = (1 <<    MEIXmpEventDONE),            /* 0x00000010 */
    MEIXmpStatusIN_FINE_POSITION_LATCHED    = MEIXmpStatusDONE,                     /* 0x00000010 */
    MEIXmpStatusPPI                         = (1 <<    MEIXmpEventPPI),             /* 0x00000020 */
    MEIXmpStatusPS_FAULT                    = (1 << MEIXmpEventPS_FAULT),           /* 0x00000040 */
    MEIXmpStatusMS_FAULT                    = (1 << MEIXmpEventMS_FAULT),           /* 0x00000080 */
    MEIXmpStatusOUT_OF_FRAMES               = (1 << MEIXmpEventOUT_OF_FRAMES),      /* 0x00000100 */
    MEIXmpStatusEXTERNAL                    = (1 << MEIXmpEventEXTERNAL),           /* 0x00000200 */
    MEIXmpStatusFRAME                       = (1 << MEIXmpEventFRAME),              /* 0x00000400 */
    MEIXmpStatusRESET                       = (1 << MEIXmpEventRESET),              /* 0x00000800 */

    MEIXmpStatusPAUSE                       = (1 <<    MEIXmpEventPAUSE),           /* 0x00001000 */
    MEIXmpStatusSTOP                        = (1 <<    MEIXmpEventSTOP),            /* 0x00002000 */
    MEIXmpStatusESTOP                       = (1 << MEIXmpEventESTOP),              /* 0x00004000 */
    MEIXmpStatusESTOP_ABORT                 = (1 << MEIXmpEventESTOP_ABORT),        /* 0x00008000 */
    MEIXmpStatusABORT                       = (1 <<    MEIXmpEventABORT),           /* 0x00010000 */

    MEIXmpStatusERROR_MASK                  = (MEIXmpStatusESTOP |                  /* 0x0001C000 */
                                            MEIXmpStatusESTOP_ABORT |
                                            MEIXmpStatusABORT),

    MEIXmpStatusHOST                = (1 <<    MEIXmpEventHOST),                /* 0x00020000 */

    MEIXmpStatusRESERVED0           = (1 <<    MEIXmpEventRESERVED0),           /* 0x00040000 */
    MEIXmpStatusRESERVED1           = (1 <<    MEIXmpEventRESERVED1),           /* 0x00080000 */
    MEIXmpStatusRESERVED2           = (1 <<    MEIXmpEventRESERVED2),           /* 0x00100000 */

    MEIXmpStatusRESERVED_MASK       = (MEIXmpStatusRESERVED0 |          /* 0x001C0000 */
                                    MEIXmpStatusRESERVED1 |
                                    MEIXmpStatusRESERVED2),

    MEIXmpStatusREC_IDLE            = (1 << MEIXmpEventREC_IDLE),                /* 0x00200000 Recorder Status */
    MEIXmpStatusREC_FULL            = (1 << MEIXmpEventREC_FULL),                 /* 0x00400000 */
    MEIXmpStatusREC_RUNNING         = (1 << MEIXmpEventREC_RUNNING),              /* 0x00800000 */
    MEIXmpStatusLIMIT                = ((long)((unsigned long)1 << MEIXmpEventLIMIT)),  /* 0x80000000 */

    /* for backward compatibility */
    MEIXmpStatusID_USER0             = MEIXmpStatusLIMIT,        /* 0x80000000 */
    MEIXmpStatusID_USER1             = MEIXmpStatusLIMIT,        /* 0x80000000 */
    MEIXmpStatusID_USER2             = MEIXmpStatusLIMIT,        /* 0x80000000 */
    MEIXmpStatusID_USER3             = MEIXmpStatusLIMIT,        /* 0x80000000 */
    MEIXmpStatusID_USER4             = MEIXmpStatusLIMIT,        /* 0x80000000 */
    MEIXmpStatusID_USER5             = MEIXmpStatusLIMIT,        /* 0x80000000 */
    MEIXmpStatusID_USER6             = MEIXmpStatusLIMIT,        /* 0x80000000 */
    MEIXmpStatusID_USER7             = MEIXmpStatusLIMIT,        /* 0x80000000 */

    MEIXmpAxisSTATE_LATCH            = (MEIXmpStatusERROR_MASK |     /* 0x800FF013 */
                                       MEIXmpStatusLIMIT |
                                       MEIXmpStatusHOST |
                                       MEIXmpStatusRESERVED_MASK |
                                       MEIXmpStatusSTOP |
                                       ~~MEIXmpStatusAT_TARGET |~~
                                       ~~MEIXmpStatusIN_COARSE_POSITION |~~
                                       MEIXmpStatusDONE),

    MEIXmpAxisSTATUS_LATCH           = (MEIXmpStatusERROR_MASK |     /* 0x800FF013 */
                                       MEIXmpStatusLIMIT |
                                       MEIXmpStatusHOST |
                                       MEIXmpStatusRESERVED_MASK |
                                       MEIXmpStatusDONE |
                                       MEIXmpStatusAT_TARGET |
                                       MEIXmpStatusIN_COARSE_POSITION),

    MEIXmpAxisLATCH_MASK             = (MEIXmpAxisSTATE_LATCH |
                                       MEIXmpAxisSTATUS_LATCH),

    MEIXmpStatusMOTION               = (MEIXmpStatusDONE_MASK |      /* 0x0010001D */
                                       MEIXmpStatusIN_COARSE_POSITION |
                                       MEIXmpStatusAT_VELOCITY |
                                       MEIXmpStatusRESERVED2 |
                                       MEIXmpStatusDONE),

    MEIXmpMS_OR_MASK                 = (MEIXmpStatusERROR_MASK |     /* 0x8007FE30 */
                                       MEIXmpStatusLIMIT |
                                       MEIXmpStatusHOST |
                                       MEIXmpStatusRESERVED0 |
                                       MEIXmpStatusRESERVED1 |
                                       MEIXmpStatusPAUSE |
                                       MEIXmpStatusSTOP |
                                       MEIXmpStatusPPI |
                                       MEIXmpStatusFRAME |
                                       MEIXmpStatusRESET),

    MEIXmpMS_AND_MASK                =  MEIXmpStatusMOTION,

    MEIXmpMSAxisMASK                 = (MEIXmpAxisSTATE_LATCH |
                                       MEIXmpAxisSTATUS_LATCH |
                                       MEIXmpStatusPAUSE),           /* 0x800FF810 */
} MEIXmpStatus;
```

- **Addition of # defines ("xmp.h" header file)**

```
/* flags used to modify state machine */
#define MEIXmpStateFlags                    MEIXmpStatus

#define MEIXmpFlagDONE                      MEIXmpStatusDONE
#define MEIXmpFlagRESET                     MEIXmpStatusRESET
#define MEIXmpFlagSTOP                      MEIXmpStatusSTOP
#define MEIXmpFlagESTOP                     MEIXmpStatusESTOP
#define MEIXmpFlagESTOP_ABORT               MEIXmpStatusESTOP_ABORT
#define MEIXmpFlagABORT                     MEIXmpStatusABORT
#defineMEIXmpFlagERROR_MASK                 MEIXmpStatusERROR_MASK
#define MEIXmpFlagHOST                      MEIXmpStatusHOST
```

- **Changes to <u>MEIXmpMotionType</u> data type ("xmp.h" header file)**

```
typedef enum {
    MEIXmpMotionTypeINVALID             = -1,

    MEIXmpMotionTypeNONE,
    MEIXmpMotionTypeUPDATE,

    MEIXmpMotionTypeSTART,
    MEIXmpMotionTypeMODIFY_ID,
    MEIXmpMotionTypeID,

    MEIXmpMotionTypeHOLD,
    MEIXmpMotionTypeOUTPUT,
    MEIXmpMotionTypeJOG,

    MEIXmpMotionTypeVELOCITY,
    MEIXmpMotionTypeVELOCITY_JERK,
    MEIXmpMotionTypeS_CURVE,
    MEIXmpMotionTypeS_CURVE_JERK,

    MEIXmpMotionTypePATH_END            = MEIXmpMotionTypeS_CURVE,
    MEIXmpMotionTypePATH_OPEN,

    MEIXmpMotionTypeLAST,
    MEIXmpMotionTypeFIRST               = MEIXmpMotionTypeINVALID + 1,

} MEIXmpMotionType;
```

- **Changes to <u>MEIXmpMotionType</u> data type ("xmp.h" header file)**

```
typedef union {
    float                       f[MEIXmpFilterDataSize];
    struct      {
                /* ErrorSum needs to be in the same position as PIV.PosErrorSum for Reset Integrator function to work */
                float           ErrorSum;
                float           ErrorDelta;
                float           OldError;
                float           OldVelocity;
                /* fft variables need to be in same position in both PID and PIV structures for system analysis tools to work */
                float           fftCh1;
                float           fftCh2;
                float           fftCh3;
                float           PIDOutput;
                float           DerivFilter[MEIXmpDFilterSize];
    } PID;
    struct      {
                /* PosErrorSum needs to be in the same position as PID.ErrorSum for Reset Integrator function to work */
                float           PosErrorSum;
                float           VelErrorSum;
                float           OldVelocity;
```

```
		float			OldY;
		/* fft variables need to be in same position in both PID and PIV structures for system analysis tools to work */
		float			fftCh1;
		float			fftCh2;
		float			fftCh3;
		float			PIVOutput;
	} PIV;
	struct {
		long			FF;
		float			OldVelocity;
	} SERCOS_DRIVE;
} MEIXmpFilterData;
```

• Changes to MEIXmpAxis data type ("xmp.h" header file)

```
typedef struct {
	MEIXmpLink			*Link;
	MEIXmpMotionSupervisor		*MS;
	MEIXmpPosInput			APos[MEIXmpAxisPosInputs];
	long				ActualPosition;
	long				CommandPosition;
	long				TargetPosition;
	float				TargetVelocity;
	long				Origin;
	long				Compensation;
	float				ActualVelocity;
	float				CommandVelocity;
	float				PositionError;
	float				FinePosTolerance;
	long				CoarsePosTolerance;
	float				VelTolerance;
	long				SettlingTime;
	long				SettlingCount;
	MEIXmpStatus			SettlingMask;
	MEIXmpStatus			Status;
	MEIXmpStatus			StateFlags;
	MEIXmpState			State;
	long				TargetValid;
	MEIXmpMetrics			Metric;
	long				ModifyIndex;
	float				ModifyTime;
	MEIXmpTrajectoryCalculator	TC;
	MEIXmpAxisGear			Gear;
	long				MoveID;
	long				ElementID;
	MEIXmpHostSignal		Signal;
} MEIXmpAxis;
```

• Changes to MEIXmpServiceCmdMotor data type ("xmp.h" header file)

```
typedef struct {
	MEIXmpServiceCmd		Config;
	MEIXmpServiceCmd		StepConfig0;
	MEIXmpServiceCmd		StepConfig1;
	MEIXmpServiceCmd		OutputA[MEIXmpLookupCmds];
	MEIXmpServiceCmd		OutputB[MEIXmpLookupCmds];
	MEIXmpServiceCmd		OutputC[MEIXmpLookupCmds];
	MEIXmpServiceCmd		OutputD[MEIXmpLookupCmds];
	MEIXmpServiceCmd		OutputE[MEIXmpLookupCmds];
	MEIXmpServiceCmd		OutputF[MEIXmpLookupCmds];
	MEIXmpServiceCmd		OutputAMP_EN[MEIXmpLookupCmds];
```

```
        MEIXmpServiceCmd            UserOut[MEIXmpLookupCmds];
        MEIXmpServiceCmd            Event[MEIXmpLookupCmds];
        MEIXmpServiceCmd            Compare[MEIXmpLookupCmds];
        MEIXmpServiceCmd            AbsSource[MEIXmpLookupCmds];
        MEIXmpServiceCmd            AbsConfig;
        MEIXmpServiceCmd            SIM4Config;      /* in CAPTURE_MODE */
        ~~MEIXmpServiceCmd~~            ~~Clear;~~
        MEIXmpServiceCmd            PulseWidth;
        MEIXmpServiceCmd            Clear;           /* Must be last in ServiceCmdMotor structure */
} MEIXmpServiceCmdMotor;
```

• Changes to MEIXmpSoftware data type (“xmp.h” header file)

```
typedef struct {
        long        ID;
        long        BranchID;
        long        Option;
        long        UserVersion;
} MEIXmpSoftware;
```

• Addition of MEIFlashSection data type (“flash.h” header file)

```
typedef struct MEIFlashSection {
        unsigned char       *address;
        long                size;
        long                sectorIndex;
} MEIFlashSection;
```

• Change to MEIFlashConfig data type (“flash.h” header file)

```
typedef struct MEIFlashConfig {
        long                        wordSize;
        ~~unsigned char~~               ~~*address;~~
        ~~long~~                        ~~size;~~
        ~~unsigned char~~               ~~*addressCode;~~
        ~~long~~                        ~~sizeCode;~~
        ~~unsigned char~~               ~~*addressData;~~
        ~~long~~                        ~~sizeData;~~
        ~~unsigned char~~               ~~*addressExternal;~~
        ~~long~~                        ~~sizeExternal;~~
        ~~long~~                        ~~sectorSize;~~
        long                        sectorSize;
        MEIFlashSectionall;
        MEIFlashSectioncode;
        MEIFlashSectiondata;
        MEIFlashSectiondataExt;
        MEIFlashSectionFPGA0;
        MEIFlashSectionFPGA1;
        MEIFlashSectionFPGA2;
} MEIFlashConfig;
```

- **Addition of MEIFlashFileType data type ("flash.h" header file)**

```
typedef enum {
	MEIFlashFileTypeNONE = 0,
	MEIFlashFileTypeCode,
	MEIFlashFileTypeDataInt,
	MEIFlashFileTypeDataExt,
	MEIFlashFileTypeCodeAndData,
	MEIFlashFileTypeFPGA0,
	MEIFlashFileTypeFPGA1,
	MEIFlashFileTypeFPGA2,
	MEIFlashFileTypeALL		/* Loads Code and all FPGAs (for .bin files that include the FPGA images) */
} MEIFlashFileType;
```

- **Change to MEIFlashMessage data type ("flash.h" header file)**

```
typedef enum {
	MEIFlashMessageFIRST = mpiMessageID(MEIModuleIdFLASH, 0),

	MEIFlashMessageFLASH_INVALID,
	MEIFlashMessageFLASH_READ_ERROR,
	MEIFlashMessageFLASH_WRITE_ERROR,
	MEIFlashMessagePATH,

	MEIFlashMessageLAST
} MEIFlashMessage;
```

- **Change to meiFlashMemoryFromFileType method ("flash.h" header file)**

```
MPI_DECL1 long MPI_DECL2
	meiFlashMemoryFromFileType(MEIFlash		flash,
				   const char		*fileName,
				   MEIFlashFileType	fileType);
```

- **Change to meiFlashMemoryToFile method ("flash.h" header file)**

```
PI_DECL1 long MPI_DECL2
	meiFlashMemoryToFile(MEIFlash			flash,
			     const char			*fileName,
			     MEIFlashFileType		fileType);
```

- **Change to MEIMapGroup data type ("map.h" header file)**

```
typedef enum  {
	MEIMapGroupINVALID = -1,

	MEIMapGroupMPI_ADC_CONFIG,		/* MEIMapGroupCONFIG_FIRST */
	MEIMapGroupMEI_ADC_CONFIG,
	MEIMapGroupMPI_AXIS_CONFIG,
	MEIMapGroupMEI_AXIS_CONFIG,
	MEIMapGroupMPI_CAPTURE_CONFIG,
	MEIMapGroupMEI_CAPTURE_CONFIG,
	MEIMapGroupMPI_CONTROL_CONFIG,
	MEIMapGroupMEI_CONTROL_CONFIG,
	~~MEIMapGroupMPI_DAC_CONFIG,~~
	~~MEIMapGroupMEI_DAC_CONFIG,~~
	MEIMapGroupMPI_EVENTMGR_CONFIG,
	MEIMapGroupMEI_EVENTMGR_CONFIG,
	MEIMapGroupMPI_FILTER_CONFIG,
	MEIMapGroupMEI_FILTER_CONFIG,
	MEIMapGroupMPI_MOTION_CONFIG,
	MEIMapGroupMEI_MOTION_CONFIG,
	MEIMapGroupMPI_MOTOR_CONFIG,
    MEIMapGroupMEI_MOTOR_CONFIG,
    MEIMapGroupMPI_NODE_CONFIG,
    MEIMapGroupMEI_NODE_CONFIG,
    MEIMapGroupMPI_RECORDER_CONFIG,
    MEIMapGroupMEI_RECORDER_CONFIG,
    MEIMapGroupMPI_SEQUENCE_CONFIG,
    MEIMapGroupMEI_SEQUENCE_CONFIG,
    MEIMapGroupMPI_SERCOS_CONFIG,
    MEIMapGroupMEI_SERCOS_CONFIG,
                                        /* MEIMapGroupCONFIG_LAST */
    MEIMapGroupMEI_XMP_BUFFER_DATA,     /* MEIMapGroupXMP_FIRST */
    MEIMapGroupMEI_XMP_DATA,
    MEIMapGroupMEI_XMP_RIPTIDE_DATA,
    MEIMapGroupMEI_XMP_PLD,
    MEIMapGroupMEI_XMP_SERCON,

    MEIMapGroupMEI_XMP_FRAME_BUFFER,
    MEIMapGroupMEI_XMP_RECORD_BUFFER,
    MEIMapGroupMEI_XMP_SERCOS_BUFFER,

                                        /* MEIMapGroupXMP_LAST */
    MEIMapGroupLAST,
    MEIMapGroupFIRST                = MEIMapGroupINVALID + 1,

    MEIMapGroupCONFIG_FIRST         = MEIMapGroupMPI_ADC_CONFIG,
    MEIMapGroupCONFIG_LAST          = MEIMapGroupMEI_SERCOS_CONFIG + 1,

    MEIMapGroupXMP_FIRST            = MEIMapGroupMEI_XMP_BUFFER_DATA,
    MEIMapGroupXMP_LAST             = MEIMapGroupMEI_XMP_SERCON + 1
} MEIMapGroup
```

- **Addition of meiPacketClose method ("packet.h" header file)**

```
MPI_DECL1 long MPI_DECL2
    meiPacketClose(MEIPacket  packet);
```

- **Addition of MEIPlatformEEPromTableType data type ("platform.h" header file)**

```
#define  MEIPlatformSocketInfoNA  (0xFFFFFFFF)

typedef enum {/* These are 1 byte wide */
    MEIPlatformEEPromTableTypeNONE          = 0x0,
    MEIPlatformEEPromTableTypeSOCKET        = 0x1,
    MEIPlatformEEPromTableTypeNA            = 0xFF,
} MEIPlatformEEPromTableType;
```

- **Change to MEIPlatformBoardInfo data type ("platform.h" header file)**

```
typedef struct MEIPlatformBoardInfo {
    long        boardId;
    long        userId;
    long        serialNumber;
    struct {
                long        number;
                long        revision;
    } manufacturer;
    long        reserved[2];
    long        socketInfo[5];
} MEIPlatformBoardInfo;
```

• Change to <u>MEIPlatformBoardType</u> data type ("platform.h" header file)

```
typedef enum {
	MEIPlatformBoardTypeINVALID = -1,

	MEIPlatformBoardTypeUNKNOWN,
	MEIPlatformBoardTypeHAMMERHEAD,
	MEIPlatformBoardTypeXMP,
	MEIPlatformBoardTypeXMP_EXPANSION,

	MEIPlatformBoardTypeLAST,
	MEIPlatformBoardTypeFIRST = MEIPlatformBoardTypeINVALID + 1
} MEIPlatformBoardType;
```

• Change to <u>meiPlatformBoardInfoGet</u> method ("platform.h" header file)

```
MPI_DECL1 long MPI_DECL2
	meiPlatformBoardInfoGet(MEIPlatform			platform,
				MEIPlatformBoardType		boardType,
				MEIPlatformBoardInfo		*boardInfo);
```

• Change to <u>meiPlatformBoardInfoSet</u> method ("platform.h" header file)

```
MPI_DECL1 long MPI_DECL2
	meiPlatformBoardInfoSet(MEIPlatform			platform,
				MEIPlatformBoardType		boardType,
				MEIPlatformBoardInfo		*boardInfo);
```

• Change to <u>MEIRemoteMethod</u> data type ("remote.h" header file)

```
typedef enum {
	MEIRemoteMethodINVALID = -1,

	MEIRemoteMethodBOARD_TYPE,
	MEIRemoteMethodBOARD_INFO_GET,
	MEIRemoteMethodBOARD_INFO_SET,

	MEIRemoteMethodFLASH_MEMORY_GET,
	MEIRemoteMethodFLASH_MEMORY_SET,

	MEIRemoteMethodINTERRUPT_ENABLE,
	MEIRemoteMethodINTERRUPT_WAIT,
	MEIRemoteMethodINTERRUPT_WAKE,

	MEIRemoteMethodMEMORY,

	MEIRemoteMethodMEMORY_GET,
	MEIRemoteMethodMEMORY_SET,

	MEIRemoteMethodOBJECT_LOCK_GIVE,
	MEIRemoteMethodOBJECT_LOCK_TAKE,

	MEIRemoteMethodRESET,

	MEIRemoteMethodLAST,
	MEIRemoteMethodFIRST = MEIRemoteMethodINVALID + 1
} MEIRemoteMethod;
```

• Addition of <u>MEIRemoteMethodBoardInfoGet</u> data type ("remote.h" header file)

```
typedef struct MEIRemoteMethodBoardInfoGet {
	MEIPlatformBoardType		boardType;
	MEIPlatformBoardInfo		*boardInfo;
} MEIRemoteMethodBoardInfoGet;
```

- **Addition of MEIRemoteMethodBoardInfoSet data type ("remote.h" header file)**

```
typedef MEIRemoteMethodBoardInfoGet MEIRemoteMethodBoardInfoSet;
```

- **Change to MEIRemoteMethodArgs data type ("remote.h" header file)**

```
typedef union {
        MEIPlatformBoardType                *boardType;
        MEIRemoteMethodBoardInfoGet         boardInfoGet;
        MEIRemoteMethodBoardInfoSet         boardInfoSet;
        union {
                long                                enable;
                MEIRemoteMethodInterruptWait        wait;
        } interruptArgs;
        MEIRemoteMethodMemory               memory;
        MEIRemoteMethodMemoryGet            memoryGet;
        MEIRemoteMethodMemorySet            memorySet;
        MEIRemoteMethodObjectLockGive       objectLockGive;
        MEIRemoteMethodObjectLockTake       objectLockTake;
} MEIRemoteMethodArgs;
```

- **Change to MEIXmpRipTideTxBuff data type ("riptide.h" header file)**

```
typedef struct MEIXmpRipTideTxBuff {
        MEIXmpRipTidePosCmd        Position[MEIXmpMaxMotionBlocks];
        MEIXmpRipTideSample        Sample[MEIXmpMaxMotionBlocks];
        MEIXmpRipTideIoCmd         Io[MEIXmpMaxMotionBlocks];
        MEIXmpRipTideRemoraTx      Remora[MEIXmpMaxSports];
} MEIXmpRipTideTxBuff;
```

- **Addition of MEIXmpRipTideMotor data type ("riptide.h" header file)**

```
typedef struct MEIXmpRipTideMotor {
        long        DlyCmd;
        long        MaxInc;
} MEIXmpRipTideMotor;
```

- **Change to MEIXmpRipTideStep data type ("riptide.h" header file)**

```
typedef struct MEIXmpRipTideStep {
        MEIXmpRipTideMotor Motor[MEIXmpMotorsPerBlock];
} MEIXmpRipTideStep;
```

- **Change to MEIXmpRipTidePosCmd data type ("riptide.h" header file)**

```
typedef struct MEIXmpRipTidePosCmd {
        long                                Header;
        long                                DAC[MEIXmpMotorsPerBlock];
        MEIXmpRipTideStep                   Step[MEIXmpStepEnginesPerMotorBlock];
        long                                Trailer;
} MEIXmpRipTidePosCmd;
```

- **Change to MEIXmpRipTideRxBuff data type ("riptide.h" header file)**

```
typedef struct MEIXmpRipTideRxBuff {
        MEIXmpRipTideIoStatus      Io[MEIXmpMaxMotionBlocks];
        MEIXmpRipTideRemoraRx      Remora[MEIXmpMaxSports];
        MEIXmpRipTidePosStatus     Position[MEIXmpMaxMotionBlocks];
} MEIXmpRipTideRxBuff;
```

- **Addition of MEIXmpRipTidePSIOSendWait data type ("riptide.h" header file)**

```
typedef struct MEIXmpRipTidePSIOSendWait {
      long value[MEIXmpMaxMotionBlocksPerSport];
} MEIXmpRipTidePSIOSendWait;
```

- **Addition of <u>MEIXmpRipTideSPSendWait</u> data type ("riptide.h" header file)**

```
typedef struct MEIXmpRipTideSPSendWait {
      long value[MEIXmpMaxMotionBlocksPerSport];
} MEIXmpRipTideSPSendWait;
```

- **Change to <u>MEIXMPRipTideData</u> data type ("riptide.h" header file)**

```
typedef struct MEIXmpRipTideData {
      MEIXmpRipTideTxBuff         TxBuff;
      MEIXmpRipTideRxBuff         RxBuff;
      MEIXmpRipTideTCB            TxBuffPositionTcb[MEIXmpMaxSports];
      MEIXmpRipTideTCB            TxBuffSampleTcb[MEIXmpMaxSports];
      MEIXmpRipTideTCB            TxBuffIoTcb[MEIXmpMaxSports];
      MEIXmpRipTideTCB            TxBuffRemoraTcb[MEIXmpMaxSports];
      MEIXmpRipTideTCB            RxBuffIoTcb[MEIXmpMaxSports];
      MEIXmpRipTideTCB            RxBuffRemoraTcb[MEIXmpMaxSports];
      MEIXmpRipTideTCB            RxBuffPositionTcb[MEIXmpMaxSports];

      long        ErrorFlag;
      long        FpgaDownloadError;
      ~~long        NewWait;~~
      ~~long        WaitConstant;~~
      long        Update20khz;
      long        BlocksPerSport[MEIXmpMaxSports];
      long        NewWait[MEIXmpMaxSports];
      long        Step;
      long        CurrentWaitConstant[MEIXmpMaxSports];
      long        AvailableWaitConstant[6];
      long        AvailableSampleWait[5];
      MEIXmpRipTidePSIOSendWait AvailablePosSampleIoSendWait[5];
      MEIXmpRipTideSPSendWait AvailableStepPosSendWait[5];
} MEIXmpRipTideData;
```

- **Change to MEIXMPMoveData data type ("riptide.h" header file)**

```
typedef struct MEIXmpMoveData {
	long		x0;
	float		v0;
	long		x5;
	float		v5;
	float		a[5];
	float		t[5];
	float		tmax;
	float		amax;
	float		dmax;
	float		vmax;
	float		dx;
	long		n_frames;
	long		frame_start;
	long		current_frame;
	float		new_time;
	long		origin;
	float		bl;
	MEIXmpAxis*axis;
} MEIXmpMoveData;
```

- **Addition of MEICaptureSIMConfig data type ("stdmei.h" header file)**

```
typedef struct MEICaptureSIMConfig {
	long enable;
} MEICaptureSIMConfig;
```

- **Change to MEICaptureConfig data type ("stdmei.h" header file)**

```
typedef struct MEICaptureConfig {
	MEICaptureSIMConfig SIM;
} MEICaptureConfig;
```

- **Addition of MEICompareDivideByNMode data type ("stdmei.h" header file)**

```
typedef enum {
	MEICompareDivideByNModeINVALID = -1,

	MEICompareDivideByNModeRANGE,
	MEICompareDivideByNModeFREE,

	MEICompareDivideByNModeLAST,
	MEICompareDivideByNModeFIRST = MEICompareDivideByNModeINVALID + 1
} MEICompareDivideByNMode;
```

- **Addition of MEICompareDivByNConfig data type ("stdmei.h" header file)**

```
typedef struct MEICompareDivByNConfig {
	long		enable;
	long		n;
} MEICompareDivByNConfig;
```

- **Change to MEICompareConfig data type ("stdmei.h" header file)**

```
typedef struct MEICompareConfig {
	long		continuous;
	MEICompareDivByNConfig divByN;
} MEICompareConfig;
```

- **Addition of <u>MEICompareDivByNParams</u> data type ("stdmei.h" header file)**

```
typedef struct MEICompareDivByNParams {
    MPIComparestopCompare;

    double   startPosition;
    double   stopPosition;

    long     arm;
    long     dir;
    long     mode;
} MEICompareDivByNParams;
```

- **Addition of <u>meiCompareDivideByNArm</u> method ("stdmei.h" header file)**

```
MPI_DECL1 long MPI_DECL2
    meiCompareDivideByNArm(MPICompare                compare,
                           MEICompareDivByNParams    *params);
```

- **Addition of <u>MEIFlashFiles</u> data type ("stdmei.h" header file)**

```
typedef struct MEIFlashFiles {
    char     binFile[MEIFlashFileMaxChars];
    char     FPGAFile[MEIXmpFlashMaxFPGAFiles][MEIFlashFileMaxChars];
} MEIFlashFiles;
```

- **Addition of <u>meiFlashMemoryFromFile</u> method ("stdmei.h" header file)**

```
MPI_DECL1 long MPI_DECL2
    meiFlashMemoryFromFile(MEIFlash           flash,
                           MEIFlashFiles      *filesIn,
                           MEIFlashFiles      *filesOut);
```

- **Addition of <u>MEIControlHardwareSocketType</u> data type ("stdmei.h" header file)**

```
typedef enum {
    MEIControlHardwareSocketTypeNONE          = 0x0,
    MEIControlHardwareSocketTypeANALOG        = 0x1,
    MEIControlHardwareSocketTypePULSE         = 0x2,
    MEIControlHardwareSocketTypeSIM4          = 0x11,
    MEIControlHardwareSocketTypeNonMBMask     = 0x10,
} MEIControlHardwareSocketType;
```

- **Addition of <u>MEIControlHardwareICType</u> data type ("stdmei.h" header file)**

```
typedef enum {
    MEIControlHardwareICType4044XL    = 0x0,
    MEIControlHardwareICType4044XLA   = 0x1,
    MEIControlHardwareICType4062XLA   = 0x3,
    MEIControlHardwareICType4085XLA   = 0x5,
    MEIControlHardwareICTypeASIC      = 0x7,
} MEIControlHardwareICType;
```

- **Addition of <u>MEIControlFPGAOptionID</u> data type ("stdmei.h" header file)**

```
typedef enum {
    MEIControlFPGAOptionID_ANALOGXL   = 0x1,
    MEIControlFPGAOptionID_ANALOGXLA = 0x2,
    MEIControlFPGAOptionID_PULSEXLA   = 0x3,
    MEIControlFPGAOptionID_SIM4XL     = 0x9,
    MEIControlFPGAOptionID_SIM4XLA    = 0xA,
} MEIControlFPGAOptionID;
```

- **Addition of MEIControlSocketInfo data type ("stdmei.h" header file)**

```
typedef struct MEIControlSocketInfo {
      long                                   sockets;
      MEIControlHardwareSocketType           socketType[MEIXmpMaxFPGAs];
      MEIControlHardwareICType               ICType[MEIXmpMaxFPGAs];
} MEIControlSocketInfo
```

- **Addition of MEIControlRipTideConfig data type ("stdmei.h" header file)**

```
typedef struct MEIControlRipTideConfig {
      long motionBlocks[MEIXmpMaxSports];
      long update20khz;
} MEIControlRipTideConfig;
```

- **Addition of MEIControlFPGA data type ("stdmei.h" header file)**

```
typedef struct MEIControlFPGA {
      char      FileName[MEIXmpFlashMaxFPGAFiles][MEIFlashFileMaxChars];
      long      *CodeAddress[MEIXmpMaxFPGAs];
} MEIControlFPGA;
```

- **Change to MEIControlConfig data type ("stdmei.h" header file)**

```
typedef struct MEIControlConfig {
      long                              preFilterCount;
      long                              compensatorCount;
      ~~long                              singleMotionBlock;~~
      MEIXmpPreFilter                   PreFilter[MEIXmpMAX_PreFilters];
      MEIXmpCompensator                 Compensator[MEIXmpMAX_Compensators];
      long                              CompensationTable[MEIXmpCompTableSize];
      MEIXmpUserBuffer                  UserBuffer;
} MEIControlConfig;
```

- **Change to MEIControlVersion data type ("stdmei.h" header file)**

```
typedef struct MEIControlVersion {
      struct {     /* control.c */
            char       *version; /* MEIControlVersionMPI (YYYYMMDD) */

            struct {     /* xmp.h */
                  long       version;    /* MEIXmpVERSION */
                  long       option;     /* MEIXmpOPTION */
            } firmware;
      } mpi;

      struct {
            long       version;               /* hardware version */

            struct {     /* MEIXmpData.SystemData{} */
                  long       version;          /* MEIXmpVERSION_EXTRACT(SoftwareID) */
                  char       revision;         /* ('A' - 1) + MEIXmpREVISION_EXTRACT(SoftwareID) */
                  long       subRevision;      /* MEIXmpSUB_REV_EXTRACT(Option) */
                  long       developmentId;    /* MEIXmpDEVELOPMENT_ID_EXTRACT(Option) */
                  long       option;           /* MEIXmpOPTION_EXTRACT(Option) */
            } firmware;

            struct     {
                  long       FPGA[MEIXmpFPGAsPerBlock];
            } motionBlock[MEIXmpMaxMotionBlocks];

            struct {
                  struct     {
                        ~~long       FPGA[MEIXmpFPGAsPerBlock];~~
                  ~~} motionBlock[MEIXmpBlocksPerBoard];~~
```

```
                        ~~struct~~      ~~{~~
                                long        version;
                                long        option;
                        } busInterface;
                } board[MEIXmpMaxBoards];
        } xmp;
} MEIControlVersion;
```

- **Change to MEIControlMessage data type ("stdmei.h" header file)**

```
typedef enum {
        MEIControlMessageFIRMWARE_INVALID = MPIControlMessageLAST,
        MEIControlMessageFIRMWARE_VERSION,
        MEIControlMessageSOCKETS,
        MEIControlMessageBAD_SOCKET_DATA,
        MEIControlMessageNO_SOCKET,

        MEIControlMessageLAST
} MEIControlMessage;
```

- **Addition of meiControlExtMemAvail method ("stdmei.h" header file)**

```
MPI_DECL1 long MPI_DECL2
        meiControlExtMemAvail(MPIControl    control,
                              long          *size);
```

- **Addition of meiControlSocketInfoGet method ("stdmei.h" header file)**

```
MPI_DECL1 long MPI_DECL2
        meiControlSocketInfoGet(MPIControl              control,
                                MEIControlSocketInfo    *socketInfo);
```

- **Addition of meiControlSocketInfoSet method ("stdmei.h" header file)**

```
MPI_DECL1 long MPI_DECL2
        meiControlSocketInfoSet(MPIControl              control,
                                MEIControlSocketInfo    *socketInfo);
```

- **Addition of meiControlFPGADefaultGet method ("stdmei.h" header file)**

```
MPI_DECL1 long MPI_DECL2
        meiControlFPGADefaultGet(MPIControl             control,
                                 MEIControlSocketInfo   *socketInfo,
                                 MEIControlFPGA         *fpga);
```

- **Addition of meiControlFlashRipTideConfigSet method ("stdmei.h" header file)**

```
MPI_DECL1 long MPI_DECL2
        meiControlFlashRipTideConfigSet(MPIControl              control,
                                        MEIFlash                flash,
                                        MEIControlRipTideConfig *config);
```

- **Deletion of MEIDacConfig data type ("stdmei.h" header file)**

```
~~typedef struct MEIDacConfig {~~
        ~~float~~                       ~~Scale;~~
        ~~MEIXmpDACInputType~~          ~~InputType;~~
        ~~MEIXmpGenericValue~~          ~~*Input;~~
~~} MEIDacConfig;~~
```

- **Deletion of MEIDacTrace data type ("stdmei.h" header file)**

```
~~typedef enum {~~
	~~MEIDacTraceFIRST = MEITraceLAST << 1,~~

	~~MEIDacTraceLAST  = MEIDacTraceFIRST << 15~~
~~} MEIDacTrace;~~
```

- **Change to MEIFilterGainPIV macro ("xmp.h" header file)**

```
typedefstruct MEIFilterGainPIV {
	struct {
		float	proportional;	/* Kpp */
		float	integral;	/* Kip */
	} gainPosition;
	struct {
		float	proportional;	/* Kpv */
	} gainVelocity1;
	struct {
		float	position;	/* Kpff */
		float	velocity;	/* Kvff */
		float	acceleration;	/* Kaff */
		float	friction;	/* Kfff */
	} feedForward;
	struct {
		float	moving;	/* MovingIMax */
		float	rest;	/* RestIMax */
	} integrationMax;
	struct {
		float	feedback;	/* Kdv */
	} gainVelocity2;
	struct {
		float	limit;	/* OutputLimit */
		float	limitHigh;	/* OutputLimitHigh */
		float	limitLow;	/* OutputLimitLow */
		float	offset;	/* OutputOffset */
	} output;
	struct {
		float	integral;	/* Kiv */
		float	integrationMax;	/* VintMax */
	} gainVelocity3;
	struct {
		float	positionFFT;	/* Ka0 */
		float	filterFFT;	/* Ka1 */
	} noise;
} MEIFilterGainPIV;
```

- **Change to MEIMotionAttrOutput data type ("stdmei.h" header file)**

```
typedefstruct MEIMotionAttrOutput {
	MEIMotionAttrOutputTypetype;
	union {
		long	*output;
		long	motor;
	} as;
	long	mask;
	long	pattern;
	long	pointIndex;	/* MEIMotionAttrMaskOUTPUT for path motion - point index for turning on output - used
			with point lists */
} MEIMotionAttrOutput;
```

- **Change to <u>MEIMotionAttributes</u> data type ("stdmei.h" header file)**

```
typedef struct MEIMotionAttributes {
        MPIEventMask            eventMask;      /* MEIMotionAttrMaskEVENT */
        double                   *finalVelocity; /* MEIMotionAttrMaskFINAL_VEL */
        MEIMotionAttrHold       *hold;          /* MEIMotionAttrMaskHOLD */
        long                    *outputCount;   /* MEIMotionAttrMaskOUTPUT for path motion - number of outputs - per axis */
        MEIMotionAttrOutput      *output;       /* MEIMotionAttrMaskOUTPUT for path and non path motion - outputs - per axis */
} MEIMotionAttributes;
```

- **Change to <u>MEIMotorMessage</u> data type ("stdmei.h" header file)**

```
typedef enum {
        MEIMotorMessageABS_ENCODER_FAULT = MPIMotorMessageLAST,
        MEIMotorMessageABS_ENCODER_TIMEOUT,
        MEIMotorMessageMOTOR_NOT_ENABLED,

        MEIMotorMessageLAST
} MEIMotorMessage;
```

- **Addition of <u>MEIMotorEventOpto</u> data type ("stdmei.h" header file)**

```
/* used to define MPIMotorEventConfig.custom.io */
typedef enum {
        MEIMotorEventMaskOPTOA_IN  = MEIXmpMotorIOMaskUSER,
        MEIMotorEventMaskOPTOA_OUT = MEIXmpMotorIOMaskUSER,
        MEIMotorEventMaskOPTOB_IN  = MEIXmpMotorIOMaskPOS_LIMIT,
        MEIMotorEventMaskOPTOB_OUT = MEIXmpMotorIOMaskAMP_ENABLE,
        MEIMotorEventMaskOPTOC_IN  = MEIXmpMotorIOMaskNEG_LIMIT,
        MEIMotorEventMaskOPTOD_IN  = MEIXmpMotorIOMaskHOME,
} MEIMotorEventOpto;
```

- **Addition of <u>MEIMotorEventMotorBlock</u> data type ("stdmei.h" header file)**

```
/* used to define MPIMotorEventConfig.custom.motorNumber */
typedef enum {
        MEIMotorEventMotorBlockINVALID = -1,

        MEIMotorEventMotorBlock0,
        MEIMotorEventMotorBlock1,
        MEIMotorEventMotorBlock2,
        MEIMotorEventMotorBlock3,

        MEIMotorEventMotorBlockFIRST = MEIMotorEventMotorBlock0,
        MEIMotorEventMotorBlockLAST  = MEIMotorEventMotorBlock3,
} MEIMotorEventMotorBlock;
```

- **Addition of <u>meiMotorDedicatedIOAddrDecode</u> method ("stdmei.h" header file)**

```
MPI_DECL1 long MPI_DECL2
        meiMotorDedicatedIOAddrDecode(MPIMotor     motor,
                                      long         addr,
                                      long*        motorNumber);
```

- **Addition of <u>meiMotorDedicatedInAddrGet</u> method ("stdmei.h" header file)**

```
MPI_DECL1 long MPI_DECL2
        meiMotorDedicatedInAddrGet(MPIMotormotor,
                                   long        motorNumber);
```

- **Addition of meiMotorDedicatedOutAddrGet method ("stdmei.h" header file)**

```
MPI_DECL1 long MPI_DECL2
    meiMotorDedicatedOutAddrGet(MPIMotor        motor,
                                long            motorNumber);
```

- **Change to MEIMotorTransceiverConfig data type ("stdmei.h" header file)**

```
typedef enum {
    MEIMotorTransceiverConfigINVALID = -1,

    MEIMotorTransceiverConfigINPUT,      /* 0 */
    MEIMotorTransceiverConfigOUTPUT,     /* 1 */
    MEIMotorTransceiverConfigSTEP,       /* 2 */
    MEIMotorTransceiverConfigDIR,        /* 3 */
    MEIMotorTransceiverConfigCW,         /* 4 */
    MEIMotorTransceiverConfigCCW,        /* 5 */
    MEIMotorTransceiverConfigQUAD_A,     /* 6 */
    MEIMotorTransceiverConfigQUAD_B,     /* 7 */
    MEIMotorTransceiverConfigCOMPARE,    /* 8 */
    MEIMotorTransceiverConfigDIAG,       /* 9 */
    MEIMotorTransceiverConfigNOT_AVAILABLE,


    MEIMotorTransceiverConfigLAST,
    MEIMotorTransceiverConfigFIRST = MEIMotorTransceiverConfigINVALID + 1,
} MEIMotorTransceiverConfig;
```

- **Addition of MEIMotorResourceNumber data type ("stdmei.h" header file)**

```
typedef enum {
    MEIMotorResourceNumberINVALID = -1,
    MEIMotorResourceNumber0,
    MEIMotorResourceNumber1,
    MEIMotorResourceNumber2,
    MEIMotorResourceNumber3,
    MEIMotorResourceNumber4,
    MEIMotorResourceNumber5,
    MEIMotorResourceNumber6,
    MEIMotorResourceNumber7,
    MEIMotorResourceNumber8,
    MEIMotorResourceNumber9,
    MEIMotorResourceNumber10,
    MEIMotorResourceNumber11,
    MEIMotorResourceNumber12,
    MEIMotorResourceNumber13,
    MEIMotorResourceNumber14,
    MEIMotorResourceNumber15,
    MEIMotorResourceNumber16,
    MEIMotorResourceNumber17,
    MEIMotorResourceNumber18,
    MEIMotorResourceNumber19,
    MEIMotorResourceNumber20,
    MEIMotorResourceNumber21,
    MEIMotorResourceNumber22,
    MEIMotorResourceNumber23,
    MEIMotorResourceNumber24,
    MEIMotorResourceNumber25,
    MEIMotorResourceNumber26,
    MEIMotorResourceNumber27,
    MEIMotorResourceNumber28,
    MEIMotorResourceNumber29,
    MEIMotorResourceNumber30,
    MEIMotorResourceNumber31,
    MEIMotorResourceNumberLAST,
    MEIMotorResourceNumberFIRST= MEIMotorResourceNumber0,
}MEIMotorResourceNumber;
```

- **Change to MEIMotorStepper data type ("stdmei.h" header file)**

```
typedef struct MEIMotorStepper {
        float                     PulseWidth;/* output pulse width  (sec) */
        long                      Loopback;/* TRUE = count step pulses in encoder reg. */
        MEIMotorResourceNumber    ResourceNumber;
} MEIMotorStepper;
```

- **Addition of MEIMotorDacChannelConfig data type ("stdmei.h" header file)**

```
typedef struct MEIMotorDacChannelConfig {
        float                        Offset;          /* volts */
        float                        Scale;
        MEIXmpDACInputType           InputType;
        MEIXmpGenericValue           *Input;
} MEIMotorDacChannelConfig;
```

- **Addition of MEIMotorDacConfig data type ("stdmei.h" header file)**

```
typedef struct MEIMotorDacConfig {
        MEIXmpDACPhase               Phase;
        MEIMotorDacChannelConfig     Cmd;
        MEIMotorDacChannelConfig     Aux;
} MEIMotorDacConfig;
```

- **Change to MEIXmpConfig data type ("stdmei.h" header file)**

```
typedef struct MEIMotorConfig {
        MEIMotorEncoder          Encoder[MEIXmpMotorEncoders];
        MEIXmpIO                 StatusOutput[MEIXmpMotorStatusOutputs];

        MEIMotorTransceiver      Transceiver[MEIXmpMotorTransceivers];
        MEIMotorTransceiver      TransceiverExtended[MEIXmpMotorTransceiversExtended];
        long                     UserOutInvert;       /* Opto Polarity */
        MEIMotorStepper          Stepper;
        long                     EncoderTermination;
        long                     SIM4;
        MEIMotorDacConfigDac;

        long      pulseEnable;   /* 0 => normal, else pulse output */
        long      pulseWidth;    /* 1 to 255 microseconds */

        /* Commutation is read-only from field Theta to end*/
        MEIXmpCommutationBlockCommutation;

        MEIXmpLimitDataLimit[MEIXmpLimitLAST];

        MEIXmpMotorTorqueLimitConfig TorqueLimitConfig;

        long AmpDisableWithLSR;/* TRUE => XMP disables amp when LSR is active */

        MEIMotorFilterInputFilterInput[MEIXmpMotorFilterInputs];
} MEIMotorConfig;
```

- **Addition of MEIMotorDacChannelStatus data type ("stdmei.h" header file)**

```
typedef struct MEIMotorDacChannelStatus {
	float		level;		/* volts */
} MEIMotorDacChannelStatus;
```

- **Addition of MEIMotorDacChannelStatus data type ("stdmei.h" header file)**

```
typedef struct MEIMotorDacStatus {
	MEIMotorDacChannelStatus		cmd;
	MEIMotorDacChannelStatus		aux;
} MEIMotorDacStatus;
```

- **Addition of MEIMotorDacChannelStatus data type ("stdmei.h" header file)**

```
typedef struct MEIMotorStatus {
	MEIMotorDacStatus		dac;
} MEIMotorStatus;
```

- **Addition of meiMotorRelatedStepMotorGet method ("stdmei.h" header file)**

```
MPI_DECL1 long MPI_DECL2
	meiMotorRelatedStepMotorGet(MPIMotor		motor,
					long		*motorNumber);
```

- **Addition of meiMotorCompareListGet method ("stdmei.h" header file)**

```
MPI_DECL1 long MPI_DECL2
	meiMotorCompareListGet(MPIMotor	motor,
				long		*compareCount,
				long		*compareList);
```

- **Addition of meiMotorDacConfigGet method ("stdmei.h" header file)**

```
MPI_DECL1 long MPI_DECL2
	meiMotorDacConfigGet(MPIMotor			motor,
			MEIMotorDacConfig		*dacConfig,
			MEIFlash			flash);
```

- **Addition of meiMotorDacConfigSet method ("stdmei.h" header file)**

```
MPI_DECL1 long MPI_DECL2
	meiMotorDacConfigSet(MPIMotor			motor,
			MEIMotorDacConfig		*dacConfig,
			MEIFlash			flash);
```

- **Addition of MEINodeMessage date type ("stdmei.h" header file)**

```
typedef enum {
	MEINodeMessageBUFFER_SIZE_ERROR = MPINodeMessageLAST,

	MEINodeMessageLAST
} MEINodeMessage;
```

- **Addition of MEISercosMessage data type ("stdmei.h" header file)**

```
typedef enum {
	MEISercosMessageBUFFER_SIZE_ERROR = MPISercosMessageLAST,

	MEISercosMessageLAST
} MEISercosMessage;
```

- **Change to MEIXmpMotorFPGA data type ("xmp.h" header file)**

```
typedef enum {
        MEIXmpMotorFPGA_XCVR_A_OUT                 = 0x00000001,
        MEIXmpMotorFPGA_XCVR_B_OUT                 = 0x00000002,
        MEIXmpMotorFPGA_XCVR_C_OUT                 = 0x00000004,
        MEIXmpMotorFPGA_XCVR_D_OUT                 = 0x00000010,
        MEIXmpMotorFPGA_XCVR_E_OUT                 = 0x00000020,
        MEIXmpMotorFPGA_XCVR_F_OUT                 = 0x00000040,
        MEIXmpMotorFPGA_SIM4                       = 0x02000000,
        MEIXmpMotorFPGA_CAPMODE_SIM4               = 0x00000080,
        MEIXmpMotorFPGA_ENCODER_TERM               = 0x04000000,
        MEIXmpMotorFPGA_STEP_LOOPBACK1             = 0x08000000,
        MEIXmpMotorFPGA_STEP_LOOPBACK0             = 0x40000000,
        MEIXmpMotorFPGA_REVERSE_ENCODER            = 0x80000000,
        MEIXmpMotorFPGA_QUAD_OUT                   = 0x00000001,
        MEIXmpMotorFPGA_CAPTURE_ON_CHANGE          = 0x10000000,
        MEIXmpMotorFPGA_COMPARE_CONTINUOUS         = 0x00000010,
        MEIXmpMotorFPGA_PULSE_ENABLE               = 0x00000008,
} MEIXmpMotorFPGA;
```

- **Change to MEIXmpMotorLookup data type ("xmp.h" header file)**

```
typedef enum {
        /* Motor Output Configuration */
        MEIXmpMotorLookupXCVR_A0_OUT               = 0x00000010,
        MEIXmpMotorLookupXCVR_A1_OUT               = 0x00000020,
        MEIXmpMotorLookupXCVR_B0_OUT               = 0x00000010,
        MEIXmpMotorLookupXCVR_B1_OUT               = 0x00000020,
        MEIXmpMotorLookupXCVR_C_OUT                = 0x00000010,
        MEIXmpMotorLookupXCVR_D_OUT                = 0x00000010,
        MEIXmpMotorLookupXCVR_DEF_OUT              = 0x00000010,
        MEIXmpMotorLookupXCVR_E_OUT                = 0x00000010,
        MEIXmpMotorLookupXCVR_F_OUT                = 0x00000010,
        MEIXmpMotorLookupAMP_EN_OUT                = 0x00000010,


        MEIXmpMotorLookupXCVR_CASCADE              = 0x00000001,
        MEIXmpMotorLookupRESET_IN                  = 0x00000001,
        MEIXmpMotorLookupDIR_IN                    = 0x00000002,
        MEIXmpMotorLookupDIAG_IN                   = 0x00000002,
        MEIXmpMotorLookupSTEP_IN                   = 0x00000004,
        MEIXmpMotorLookupCOMPARE_INC               = 0x00000004,
        MEIXmpMotorLookupCOMPARE_IN                = 0x00000008,
        MEIXmpMotorLookupXCVR_IN                   = 0x00000008,
        MEIXmpMotorLookupAMP_EN_IN                 = 0x00000008,


        /* Motor UserOut Configuration */
        MEIXmpMotorLookupUSER_OUT                  = 0x00000010,
        MEIXmpMotorLookupUSER_IN                   = 0x00000008,
        /* Motor Event Configuration */
        MEIXmpMotorLookupCASCADE_IN                = 0x00000008,
        MEIXmpMotorLookupCASCADE_OUT               = 0x00000010,
        MEIXmpMotorLookupEVENT_OUT                 = 0x00000020,
        MEIXmpMotorLookupOVERTRAVEL_POS            = 0x00000001,
        MEIXmpMotorLookupOVERTRAVEL_NEG            = 0x00000002,
        MEIXmpMotorLookupHOME                      = 0x00000004,
        MEIXmpMotorLookupINDEX                     = 0x00000008,
        MEIXmpMotorLookupXCVR_A                    = 0x00000001,
        MEIXmpMotorLookupXCVR_B                    = 0x00000002,
        MEIXmpMotorLookupXCVR_C                    = 0x00000004,
        MEIXmpMotorLookupSIM4_INDEX                = 0x00000001,
        MEIXmpMotorLookupSIM4_ENCB                 = 0x00000002,
        MEIXmpMotorLookupSIM4_ENCA                 = 0x00000004,
        MEIXmpMotorLookupSIM4_EVENT_IN          = 0x00000008,
        MEIXmpMotorLookupSIM4_EVENT_OUT         = 0x00000080,
} MEIXmpMotorLookup;
```

- **Change to <u>MEIXmpMotionType</u> data type ("xmp.h" header file)**

```
typedef enum {
    MEIXmpMotionTypeINVALID             = -1,

    MEIXmpMotionTypeNONE,

    MEIXmpMotionTypeSTART,
    MEIXmpMotionTypeMODIFY_ID,
    MEIXmpMotionTypeID,

    MEIXmpMotionTypeHOLD,
    MEIXmpMotionTypeOUTPUT,
    MEIXmpMotionTypeJOG,

    MEIXmpMotionTypeVELOCITY,
    MEIXmpMotionTypeVELOCITY_JERK,
    MEIXmpMotionTypeS_CURVE,
    MEIXmpMotionTypeS_CURVE_JERK,

    MEIXmpMotionTypePATH_END            = MEIXmpMotionTypeS_CURVE,
    MEIXmpMotionTypePATH_OPEN,

    MEIXmpMotionTypeLAST,
    MEIXmpMotionTypeFIRST               = MEIXmpMotionTypeINVALID + 1,

} MEIXmpMotionType;
```

- **Addition of <u>MEIXmpDACPhase</u> data type ("xmp.h" header file)**

```
typedef enum {
    MEIXmpDACPhaseNORMAL = 0,
    MEIXmpDACPhaseINVERTED = 1,
} MEIXmpDACPhase;
```

- **Addition of <u>MEIXmpDAC</u> data type ("xmp.h" header file)**

```
typedef struct {
    MEIXmpDACPhase          Phase;
    MEIXmpDACChannel        Cmd;
    MEIXmpDACChannel        Aux;
} MEIXmpDAC;
```

- **Addition of <u>MEIXmpLimitDataModifyState</u> data type ("xmp.h" header file)**

```
typedef enum {
    MEIXmpLimitDataModifyStateIDLE     = 0x0,
    MEIXmpLimitDataModifyStateMODIFY= 0x1,
    MEIXmpLimitDataModifyStateDONE     = 0x2,
} MEIXmpLimitDataModifyState;
```

• Change to MEIXmpLimitData data type (“xmp.h” header file)

```
typedef struct {
        MEIXmpLimitCondition            Condition[MEIXmpLimitConditions];
        MEIXmpStatus                    Status;
        MEIXmpLogic                     Logic;
        MEIXmpLimitOutput               Output;
        /* These variables are used internally.
        They should not be changed by the Host */
        long                            Count;
        long                            State;
        MEIXmpLimitDataModifyState      ModifyState;
} MEIXmpLimitData;
```

• Change to MEIXmpPoint data type (“xmp.h” header file)

```
typedef union {
        struct {
                long    Position;
                float   MaxVelocity;
                float   MaxAccel;
                float   MaxDecel;
                float   JerkPercent;
                float   AccelJerk;
                float   DecelJerk;
                float   EndVelocity;
                float   Delay;
                long    Control;
                long    *InPtr;
                long    InMask;
                long    InPattern;
                float   InputTimeout;
                long    *OutPtr;
                long    OutMask;
                long    OutPattern;
        } POS;
        struct {
                long    ADCChannel;
                float   Center;
                float   Deadband;
                float   M1;
                float   M3;
        } JOG;
} MEIXmpPoint;
```

• Deletion of MEIXmpMSAxis data type (“xmp.h” header file)

~~typedef struct {~~
~~long~~ ~~AxisNumber;~~
~~MEIXmpPoint~~ ~~Point;~~
~~} MEIXmpMSAxis;~~

• Change to MEIXmpMSLink data type (“xmp.h” header file)

typedef struct {
MEIXmpLink Link;
~~long~~ ~~DACNumber[MEIXmpCommDACs];~~
~~long~~ ~~DACs;~~
MEIXmpObjectMap ADCMap;
~~MEIXmpObjectMap~~ ~~DACMap;~~
long SercosNumber;
long NodeNumber;

- **Change to <u>MEIXmpMSLink</u> data type ("xmp.h" header file)**

```
typedef struct {
	~~MEIXmpMSAxis	Axis[MEIXmpMAX_COORD_AXES];~~
	long		Axes;
	long		AxisNumber[MEIXmpMAX_COORD_AXES];
} MEIXmpMSLink
```

- **Change to <u>MEIXmpMotorLink</u> data type ("xmp.h" header file)**

```
typedef struct {
	MEIXmpLink		Link;
	~~long			DACNumber[MEIXmpCommDACs];~~
	~~long			DACs;~~
	MEIXmpObjectMap		ADCMap;
	~~MEIXmpObjectMap	DACMap;~~
	long			SercosNumber;
	long			NodeNumber;
} MEIXmpMotorLink;
```

- **Change to <u>MEIXmpMotor</u> data type ("xmp.h" header file)**

```
typedef struct {
	MEIXmpSampleConfig		SampleConfig;
	MEIXmpMotorLink			*Link;
	MEIXmpMotorType			Type;
	MEIXmpStatus			Status;
	MEIXmpMotorIO			IO;
	float				CommandOutput;
	long				AbortDelay;
	long				AbortDelayCount;
	long				EnableDelay;
	long				EnableDelayCount;
	long				BrakeDelay;
	long				BrakeDelayCount;
	MEIXmpMotorTorqueLimitConfig	TorqueLimitConfig;
	long				TorqueLimitState;
	MEIXmpDAC			DAC;
	MEIXmpCommutationBlock		Commutation;
	MEIXmpHostSignal		Signal;
} MEIXmpMotor;
```

- **Change to <u>MEIXmpDataRecorder</u> data type ("xmp.h" header file)**

```
typedef struct {
	long			*RecAddress[MEIXmpMaxRecSize];
	long			RecSize;
	long			CollectionSize;
	long			Period;
	long			BufferLimit;
	long			RecsOut;
	long			RecsIn;
	long			RecIn;
	long			Index;
	long			Sample;
	long			Count;
	long			Enable;
	MEIXmpStatus		Status;
	MEIXmpHostSignal	Signal;
	~~long			*Record;~~
	~~long			BufferSize;~~
} MEIXmpDataRecorder;
```

- **Change to <u>MEIXmpLinkBuffer</u> data type ("xmp.h" header file)**

```
typedef struct {
	MEIXmpDomainMap		DomainMap[MEIXmpMAX_Domains];
	MEIXmpObjectMap		ADCMap[MEIXmpMAX_ADCs];
	~~MEIXmpObjectMap~~		~~DACMap[MEIXmpMAX_DACs];~~
	MEIXmpMotorLink		MotorLink[MEIXmpMAX_Motors];
	MEIXmpLink			FilterLink[MEIXmpMAX_Filters];
	MEIXmpLink			AxisLink[MEIXmpMAX_Axes];
	MEIXmpMSLink		MSLink[MEIXmpMAX_MSs];
} MEIXmpLinkBuffer;
```

- **Change to <u>MEIXmpServiceCmdMotor</u> data type ("xmp.h" header file)**

```
typedef struct {
	MEIXmpServiceCmdConfig;
	MEIXmpServiceCmd		StepConfig0;
	MEIXmpServiceCmd		StepConfig1;
	MEIXmpServiceCmd		OutputA[MEIXmpLookupCmds];
	MEIXmpServiceCmd		OutputB[MEIXmpLookupCmds];
	MEIXmpServiceCmd		OutputC[MEIXmpLookupCmds];
	MEIXmpServiceCmd		OutputD[MEIXmpLookupCmds];
	MEIXmpServiceCmd		OutputE[MEIXmpLookupCmds];
	MEIXmpServiceCmd		OutputF[MEIXmpLookupCmds];
	MEIXmpServiceCmd		OutputAMP_EN[MEIXmpLookupCmds];

	MEIXmpServiceCmd		UserOut[MEIXmpLookupCmds];
	MEIXmpServiceCmd		Event[MEIXmpLookupCmds];
	MEIXmpServiceCmd		Compare[MEIXmpLookupCmds];
	~~MEIXmpServiceCmd~~		~~Clear;~~
	MEIXmpServiceCmd		AbsSource[MEIXmpLookupCmds];
	MEIXmpServiceCmd		AbsConfig;
	MEIXmpServiceCmd		SIM4Config;/* in CAPTURE_MODE */
	MEIXmpServiceCmd		Clear;
	MEIXmpServiceCmd		PulseWidth;
} MEIXmpServiceCmdMotor;
```

- **Change to <u>MEIXmpServiceCmdAux</u> data type ("xmp.h" header file)**

```
typedef struct {
	MEIXmpServiceCmd		Config;
	~~MEIXmpServiceCmd~~		~~Clear;~~
	MEIXmpServiceCmd		AbsSource[MEIXmpLookupCmds];
	MEIXmpServiceCmd		AbsConfig;
	MEIXmpServiceCmd		Clear;
} MEIXmpServiceCmdAux;
```

- **Change to <u>MEIXmpServiceCmdCompare</u> data type ("xmp.h" header file)**

```
typedef struct {
	MEIXmpServiceCmd		ValueSelect[MEIXmpLookupCmds];
	MEIXmpServiceCmd		Previous[MEIXmpLookupCmds];
	MEIXmpServiceCmd		CompareMode[MEIXmpLookupCmds];
	MEIXmpServiceCmd		DivByNControl;
	MEIXmpServiceCmd		DivByNValue; /* sets "N" */
} MEIXmpServiceCmdCompare;
```

- **Change to <u>MEIXmpServiceCmdBuffer</u> data type ("xmp.h" header file)**

```
typedef struct {
	MEIXmpServiceCmdBlock	Block[MEIXmpMaxMotionBlocks];
} MEIXmpServiceCmdBuffer;
```

- **Change to MEIXmpBufferData data type ("xmp.h" header file)**

```
typedef struct {
    long                    ExtMemSize;
    MEIXmpCommandBuffer     CommandBuffer;
    float                   CommutationTable[MEIXmpCOMM_TABLE_SIZE];
    long                    CompensationTable[MEIXmpCompTableSize];
    MEIXmpDomain            Domain[MEIXmpMAX_Domains];
    MEIXmpLinkBuffer        LinkBuffer;
    MEIXmpPoint             PointBuffer[MEIXmpMAX_Axes];
    MEIXmpMessage           HostMessageBuffer[MEIXmpMAX_MESSAGES];
    MEIXmpUserBuffer        UserBuffer;
    MEIXmpServiceCmdBuffer  ServiceCmdBuffer;
    ~~MEIXmpSercos            Sercos[MEIXmpSercosCountMAX];~~
    MEIXmpCaptureCompare    Capture[MEIXmpMaxCaptures];
    MEIXmpCaptureCompare    Compare[MEIXmpMaxCompares];
    MEIXmpMotorLimit        MotorLimit[MEIXmpMAX_Motors];
    long                    LimitTable[MEIXmpLimitLookupSize];
    MEIXmpPreFilter         PreFilter[MEIXmpMAX_PreFilters];
    MEIXmpCustomBuffer      CustomBuffer;
    ~~/* Note: Frame Buffer will not be initialized or stored in flash */~~
    ~~MEIXmpFrame             FrameBuffer[MEIXmpMAX_Axes * MEIXmpFrameBufferSize];~~
    ~~long                    RecordBuffer[MEIXmpRecordBufferSize];~~
} MEIXmpBufferData;
```

- **Change to MEIXmpHWVersion data type ("xmp.h" header file)**

```
typedef struct {
    struct  {
            long    Version;
            long    Option;
    } BusIF;
    ~~long    Block[MEIXmpBlocksPerBoard];~~
} MEIXmpHWVersion;
```

- **Addition of MEIXmpFrameBuffer data type ("xmp.h" header file)**

```
typedef struct {
    MEIXmpFrame             *Ptr;
    long                    Size;           /* number of words */
} MEIXmpFrameBuffer;
```

- **Addition of MEIXmpDataRecord data type ("xmp.h" header file)**

```
typedef struct {
    long                Record;
} MEIXmpDataRecord;   /* This structure needed by map module */
```

- **Addition of MEIXmpSercosBuffer data type ("xmp.h" header file)**

```
typedef struct {
    MEIXmpSercos    *Ptr;
    long            Size;       /* number of words */
} MEIXmpSercosBuffer;
```

- **Addition of <u>MEIXmpExtAlloc</u> data type ("xmp.h" header file)**

```
typedef struct {                 /* Structures in allocated external memory */
    MEIXmpFrameBuffer            FrameBuffer;      /* This should allways be first in the structure so that reallocation will
                                                      not require the MPI to rewrite all of the Axis' Frame pointers */
    MEIXmpDataRecordBuffer       DataRecordBuffer;
    MEIXmpSercosBuffer           SercosBuffer;     /* This has a size of 0 if no Sercos hardware exists */
} MEIXmpExtAlloc;
```

- **Addition of <u>MEIXmpSoftware</u> data type ("xmp.h" header file)**

```
typedef struct {
    long        ID;
    long        Option;
    long        UserVersion;
} MEIXmpSoftware;
```

- **Change to <u>MEIXmpSystemData</u> data type ("xmp.h" header file)**

typedef struct {
    long   Signature;
    long   Disable;
    MEIXmpSoftware   Software;
    ~~long   SoftwareID;~~
    ~~long   Option;~~
    long   EnabledMotors;
    long   EnabledFilters;
    long   EnabledAxes;
    long   EnabledMSs;
    long   EnabledPSs;
    long   EnabledPreFilters;
    long   EnabledCompensators;
    long   EnabledCmdDACs;
    long   EnabledAuxDACs;
    long   EnabledADCs;
    long   EnabledSercosRings;
    ~~long   SingleMotionBlock;~~
    long   EnabledRecords;
    long   SamplePeriod;
    long   SampleCounter;
    long   CountDelta;
    long   BackgroundCycle;
    long   MaxForegroundTime;
    long   MaxBackgroundTime;
    long   Gate;
    long   HostGate;
    MEIXmpHWVersion   HWVersion[MEIXmpMaxBoards];
    long   MotionBlockVersion[MEIXmpMaxMotionBlocks];
    long   *FPGACode[MEIXmpMaxFPGAs];
    MEIXmpExtAlloc   ExtAlloc;
} MEIXmpSystemData;

- **Change to <u>MEIXmpData</u> data type ("xmp.h" header file)**

```
typedef struct {
    MEIXmpSystemData             SystemData;
    MEIXmpMotor                  Motor[MEIXmpMAX_Motors];
    MEIXmpFilter                 Filter[MEIXmpMAX_Filters];
    MEIXmpAxis                   Axis[MEIXmpMAX_Axes];
    MEIXmpMotionSupervisor       MS[MEIXmpMAX_MSs];
    MEIXmpProgramSequencer       PS[MEIXmpMAX_PSs];
    MEIXmpHostMessage            HostMessage;
    MEIXmpDataRecorder           Recorder;
```

```
        MEIXmpCompensator         Compensator[MEIXmpMAX_Compensators];
        ~~MEIXmpDAC~~                 ~~DAC[MEIXmpMAX_DACs];~~
        MEIXmpADC                 ADC[MEIXmpMAX_ADCs + 1];
        MEIXmpBlock               Block[MEIXmpMaxMotionBlocks];
        MEIXmpInternalData        Internal;
} MEIXmpData;
```

4 Motion Console and Motion Scope

4.1 Utilties: Closed Issues

4.1.1 XMP Motion Console

Modified in Version: **03.37.21**

Modification Type: ***DR (Discrepancy Report)***

Number **Name**

1013 **Add Numeric IDN treated as read-only**

Under certain conditions, the data of an IDN will be treated as read-only.

1014 **Invalid SERCOS Ring Baud Rate displayed**

Invalid text is displayed for any baud rate greater than MPISercosBaud4MBIT.

1015 **Unsigned Hex IDN data displayed as decimal**

Under some situations, IDN data that should be displayed in HEX format is displayed in decimal format.

Modified in Version: **03.37.20**

Modification Type: ***NF (New Feature)***

Number **Name**

905 **Add "Amp Disable Action" attribute to Motor Summary**

The "Amp Disable Action" attribute has been added to the Motor Summary general configuration tab. It corresponds to the disableAction attribute of the MEIMotorConfig structure.

Modification Type: ***MI (Minor Improvement)***

Number **Name**

899 **[Dup. of 883] Display all digits for hex values**

For numeric values displayed in hex format, all digits will now be displayed. The value is padded on the left with zeros.

901 **Performance issues in Controller Summary**

When a controller attribute was modified, the profile of every object was being saved. This did not actually serve any purpose and has been suppressed because it caused a noticeable pause.

Also, summary windows were being updated twice after a controller was reset or during a refresh. This has been corrected.

Modification Type: ***DR (Discrepancy Report)***

Number **Name**

856 **[Dup. of 855] Summary Configuration Gets Confused when Configured For Many Objects**

A bug in the way the Summary configuration is saved to the .INI file causes a Summary window to get confused when it is programmed to display a large number of objects.

858 **[Dup. of 857] Help/About Doesn't Display Version Info if a User's Locale is Not Supported**

If the user’s locale was set to be a non-supported locale (e.g. English, Canada), then the Help/About dialog would not show the correct application version or copyright date.

863 **[Dup. of 862] Motion Console Crashes if Active Tab Page >= Tab Page Count**

When Motion Console was executed with an .INI file that was created with a different version of Motion Console, there was a chance that it would crash. The crash would occur if the Active Tab Page of a Summary was set to be higher than the current number of tabs in the Summary.

900 **Default column width is miscalculated**

The default column width in a summary window may be wider than necessary if the control that requires the widest width is a combo box.

Modification Type: ***CR (Change Request)***

Number	Name

845 **Object List Config window Application Error**

Clicking the Add or Set buttons in any of the Object List Configuration windows when no controllers had been added would cause an NT Application Error.

904 **Status Output choices modified to reflect changes to MEIXmpEvent**

The choices for Motor Status Out Config are the events defined by the enum MEIXmpEvent. The choices have been modified to reflect changes made to MEIXmpEvent in the 220020117.1.8 version of the MPI.

Modified in Version: **03.37.19**

Modification Type: ***MI (Minor Improvement)***

Number	Name

839 **[Dup. of 799] NULL firmware message**

The following error messages will now be displayed when mpiControlInit fails with the corresponding error code:

MEIControlMessageFIRMWARE_VERSION_NONE: No firmware is on the controller.
MEIControlMessageFIRMWARE_VERSION: The version of firmware on the controller is invalid.
MEIControlMessageFIRMWARE_INVALID: The firmware on the controller is invalid.

Modification Type: ***DR (Discrepancy Report)***

Number	Name

834 **Motion Console crashes when browse button in download firmware window is pressed**

If the MEI_INSTALL_DIR environment variable was not set, then Motion Console would crash when browsing for a firmware file to upload or download.

835 **[Dup. of 807] Panic action does not stick**

The panic action setting was not being saved when minimizing and restoring Motion Console. This has been fixed.

836 **[Dup. of 774] Two ampersands (&&) in a tooltip are displayed as an underline**

Two ampersands that should be displayed in a tooltip were instead being displayed as an underline.

841 **[Dup. of 750] Sine Comm error message limits ability to fix problem**

There is no longer a minimum number of Aux DACs required to switch to no commutation mode.

Modification Type: ***CR (Change Request)***

Number	Name

837 **[Dup. of 810] Motion Console doesn't display the state of the index input under the motor summary's bottom I/O tab**

The Index motor input bit (MEIXmpMotorIOBitINDEX) is now displayed in the Motor I/O Status tab.

Modified in Version: 03.37.18

Modification Type: DR (Discrepancy Report)

Number	Name
822	[Dup. of 821] Motion Console dies when MEI_INSTALL_DIR does not exist

MotionConsole uses the environment variable MEI_INSTALL_DIR to determine where the default .INI file is to be located. If MEI_INSTALL_DIR was set to a directory that didn't exist, Motion Console would crash. It now displays an appropriate error message and exits.

Number	Name
823	[Dup. of 816] Initial directory for firmware download/upload should be MEI_DIR

When the user downloads or uploads firmware for the first time, the directory where the browser is set is obtained from the environment variable MEI_DIR. There is a bug that erroneously sets this directory to C:\,MEI. Since thi directory normally does not exist, the initial directory for the browser is set to whatever the default .

Modified in Version: 03.37.17

Modification Type: MI (Minor Improvement)

Number	Name
788	Merge Japanese Translations Intro Production Release Version

Merge the current Japanese translations into the Production Release version.

Modified in Version: 03.37.16

Modification Type: MI (Minor Improvement)

Number	Name
780	Display MPI Library Assertion Violations

MPI library assertion violations are now displayed in a dialog box before the application exits. The user is given the choice of ignoring the error.

Modified in Version: 03.37.15

Modification Type: MI (Minor Improvement)

Number	Name
551	Default firmware download directory should be ...\xmp\bin

When one tries to load new firmware, the default directory should be ...\xmp\bin. Or, it should remember the last directory that was used.

Modified in Version: 03.37.14

Modification Type: MI (Minor Improvement)

Number	Name
737	Make grid cell tooltips multi-line

The implementation of tooltips has been modified to support multiple lines of text. If the tooltip exceeds the width of the screen, or if there are newline characters in the text, then the tooltip will be displayed with multiple lines.

Modification Type: DR (Discrepancy Report)

Number	Name
743	DAC units wrong

The DAC Level units in the motor summary are in Volts. The tool tips have been modified to reflect this change.

Modified in Version: 03.37.13

Modification Type: DR (Discrepancy Report)

Number **Name**

727 **Motion state icon changes when attribute is edited**

While an Axis is moving towards position 2, the right arrow state icon is displayed. Prior to this fix, if the user edited an Axis or Motion Supervisor attribute, the state icon would erroneously switch to the left arrow. The motion itself did not change, just the icon.

Modified in Version: 03.37.12

Modification Type: DR (Discrepancy Report)

Number **Name**

715 **Downloading firmware with fewer MS objects causes library errors**

When firmware was downloaded onto a controller that resulted in there being fewer Motion Supervisors, a library error was displayed for every MS object that was no longer enabled.

716 **Modifying controller attribute while motor is in motion causes strange behavior**

If a motor was in motion and the user modified a controller attribute and then clicked on another grid, then an error message was displayed. After the message was displayed, Motion Console behaved as if the left mouse button was being held down.

717 **Object status always updated, even when not being displayed**

In general, an object status should not be updated unless it is being displayed. There was a bug that caused the object status to always be updated after the object was initially displayed.

720 **Suspect firmware files dialog box displayed at inappropriate times**

The "Suspect Firmware Files" dialog box should only be displayed when meiFlashMemoryFromFile() fails with an error code of MPIMessageFILE_OPEN_ERROR. It was being displayed for any error code and sometimes after downloading a good firmware file.

Modified in Version: 03.37.11

Modification Type: DR (Discrepancy Report)

Number **Name**

702 **Object Explorer context menus are out of sync**

When the user right-clicked on any folder icon in the Object Explorer, an inappropriate context menu for that item was displayed.

706 **Controller Summary column headers should display the controller name**

The column headers in the Controller Summary should display controller names instead of the controller index. The name is assigned to the controller when it is created by the user. The controller index is meaningless to the user and can actually conflict with the name.

707 **Summary profiles not saved before switching profile**

The Summary window position and object list configuration were not being saved to the profile before switching to a different profile.

708 **Ctrl + S opens Save dialog box, but nothing is saved**

Typing Ctrl + S opens a Save dialog box. The shortcut should not be defined at all.

Modified in Version: 03.37.10

Modification Type: NF (New Feature)

Number **Name**

690 **Add a command line option to disable the splash screen**

The option "-s X" has been added to the command line to allow the user to disable the splash screen. If the value of X is 0, then the splash screen will be disabled.

Modification Type: MI (Minor Improvement)

Number **Name**

692 **Object Configuration Refreshed When Modified Externally**

Motion Console will now refresh the configuration of an object immediately prior to the user editing an attribute of that object. This makes it possible for the correct configuration to be set when the configuration of an object has been modified outside of Motion Console.

Modification Type: CR (Change Request)

Number **Name**

687 **No CellTip displayed for View I/O button when controller is uninitialized**

The View I/O button on the Config tab page of the Controller Summary did not display a CellTip error when the controller was unable to initialize.

Modified in Version: 03.37.08

Modification Type: DR (Discrepancy Report)

Number **Name**

585 **Order of sub-objects can be changed when order is irrelevant**

When the Object List Configuration Dialog Box is used to modify the sub-object list of an object, the order of the sub-objects should not be modifiable if the order is irrelevant. This should be done by hiding the "Up" and "Down" buttons. This was not being done.

597 **Invalid object count displayed in Controller Summary window**

If an object count was changed by entering a new number and hitting <Enter> on the Config page of the Controller Summary window, and then the controller was reset, an invalid object count was being displayed when entering edit mode on the object count cell.

658 **Incorrect error message**

In the Motor Summary window, under the SinComm tab, attempting to modify any of the fields used to give the error message: "There is an insufficient number of DACs mapped to this motor for commutation. Make sure that there are enough DACs enabled in the controller and then map two DACs to this motor object."
Since DACs can no longer be mapped to motors, the error message has been changed to "There is an insufficient number of enabled DACs or auxiliary DACs to enable commutation for this motor. The number of DACs and auxiliary DACs enabled in the controller must be at least %d", where %d is the motor number + 1.

683 **Transceiver Status Reporting**

On the I/O status page of the Motor Summary, levels for transceivers D - F and and User I/O were mixed up. Transceiver D was displaying User I/O, transceiver E was displaying transceiver D, transceiver F was displaying transceiver E, and User I/O was displaying transceiver F.

Modified in Version: 03.37.07

Modification Type: CR (Change Request)

Number **Name**

652 **Add support for MPI version string**

The text in the "About Motion Console" dialog has been modified to correctly display the MPI version. The version of the MPI that Motion Console was compiled with has also been added.

653 **Ignore MPIControlMessageLIBRARY_VERSION error from mpiControlInit( )**

While the controller is being initialized, if an MPIControlMessageLIBRARY_VERSION error occurs, then the error will be displayed, but otherwise it is ignored.

Modified in Version: 03.37.06

Modification Type: *NF (New Feature)*

Number	Name
623	**Refresh hotkey**

The F5 key now behaves as a refresh key, as in other applications. When F5 is clicked, the display will be refreshed to reflect any changes that may have been made to the controller configuration outside of Motion Console. The behavior is exactly the same as selecting all controllers in the Object Explorer or the Controller Summary and then clicking the Refresh Display button in that window. A display refresh can also be triggered by clicking on a new icon on the main toolbar (next to the Panic Button), or by selecting the "Refresh Display" menu item under the View menu.

Modification Type: *NF (New Feature)*

Number	Name
631	**Add shortcuts to menu items**

Many of the menu items in Motion Console have keyboard shortcuts. Unfortunately, the user must use a mouse to find out what the shortcut is. The shortcut should be added to the menu item.

Modification Type: *DR (Discrepancy Report)*

Number	Name
639	**Flashing Pulse Controller with Motion Console doesn't work**

When downloading firmware, Motion Console was not configuring RipTide like the flash utility.

650	**Motion Console dies when controller is deleted while being displayed in Controller Summary**

Motion Console was dying when a controller was removed while being displayed in the Controller Summary.

Modified in Version: 03.37.05

Modification Type: *NF (New Feature)*

Number	Name
591	**Add Splash Screen**

An MEI splash screen is now displayed as Motion Console is loading.

649	**Add support for new motion types: S-Curve Jerk, Velocity Jerk**

To the Axis Summary, add the following rows to the Motion tab: AccelJerk, DecelJerk. To the Motion Type combo on the Motion Supervisor Summary, add the following new types: S-Curve Jerk, Velocity Jerk.

Modification Type: *DR (Discrepancy Report)*

Number	Name
632	**Controller information not removed from .INI file after controller is removed**

After a controller has been removed, all data for that controller should be removed from the .INI file. Data for axes and motion supervisors was not being removed.

636	**Incorrect error message in Motion Console**

When in closed loop sine comm mode, an attempt was made to change the DAC phasing, it returned an error. The message, "DAC phasing cannot be changed in open loop mode" was being displayed. The correct message should have been "... closed loop mode".

Modified in Version: 03.37.04

Modification Type: ***CR (Change Request)***

Number	Name
638	**New dialog box for downloading firmware**

A new dialog box for downloading firmware has been added. With the new dialog box, FPGA files can be specified as well as a .BIN file.

Modified in Version: 03.37.03

Modification Type: ***NF (New Feature)***

Number	Name
594	**Add Motor Stepper Configuration Attributes for XMP Pulse**

The following motor configuration attribute has been added to the Motor General Configuration tab page:

Stepper Resource Number: MEIMotorConfig.Stepper.ResourceNumber

595	**Add new Transceiver Configuration Options for XMP Pulse**

The transceiver configuration options for transceivers A and B have been expanded to reflect the new definition of the MEIMotorTransceiverConfig enumeration.

Modification Type: ***MI (Minor Improvement)***

Number	Name
629	**Summary Tabs To Use System Menu Font**

The font used for the Summary window tabs has been changed from "Arial" to being based on the system menu font. The height of the tab beam is adjusted according to the height of the font. The system menu font is modified in the Display Properties system dialog box, under the "Appearance" tab.

Modification Type: ***DR (Discrepancy Report)***

Number	Name
622	**Japanese Font Size On All Tabs Displayed Bigger Than Expected**

Japanese fonts on the all tabs of the all summary windows were being displayed taller than the tab beam. The minor improvement implemented for issue #629 fixed this problem.

625	**ToolTips for Main Frame ToolBar Look Strange**

The ToolTips for the buttons on the Main Frame ToolBar were being displayed with a strange character in the middle of them.

627	**Clicking on another cell causes selection to change before value is set in multiple cell**

When several cells are selected, and the user modifies the value of the current cell, then the value should be set in all the cells in the selection. This wasn't happening when the user clicked on a cell that was not in the set of selected cells.

Modification Type: ***CR (Change Request)***

Number	Name
626	**Remove DAC object entirely**

The DAC object has been removed from the MPI, so it has also been removed from Motion Console. The functionality that was formerly implemented in the DAC Summary window has been moved to the Motor Summary window. The DAC configuration has been moved to the "Config" tab page and the DAC status has been moved to the "Status" tab page.

Modified in Version: 03.37.02

Modification Type: ***MI (Minor Improvement)***

Number	Name
607	**Change Summary window style to include maximize, minimize, and restore buttons**

The maximize, minimize, and restore buttons have been reinstated to the top right corner.

613 **Move tab scroll buttons to the right of the tabs**

The tab scroll buttons have been moved to the right of the tabs. Also, the tab scroll buttons will now automatically hide when the window is large enough to display all the tabs.

614 **Add Solid Border to Top of Status Grids**

A solid border was added to the top of the status grids, just like in the configuration grids.

615 **Improved Tab Appearance**

The appearance of the tabs on the Summary windows has been made to look more tab-like.

Modification Type: ***DR (Discrepancy Report)***

Number	Name
288	**Grid ToolTip covers up CellTip for last row**

The CellTip for the row header in the last row was obscured by the ToolTip for the grid. The ToolTip for the grid is now displayed when the user hovers over the tab for the grid. Therefore, the two ToolTips are not displayed at the same time.

562 **Columns disappear after resetting all columns to default width**

It is possible to get a summary window into a state where different columns are displayed in the status vs. the configuration grids. In this state, there are "hidden" columns in the configuration grid, because they are actually scrolled to the left of the grid, but there is no horizontal scrollbar allowing the user to access the hidden columns. They can be forced into the viewable area by using the arrow keys. To recreate the problem, follow these steps:

1) Open the axis summary window and program it to display four axes.
2) Select the entire grid by clicking on the top, left-most cell.
3) Resize all the columns by dragging right-most tracking handle.
4) Restore the width of every column to the default width by double-clicking on the last tracking handle.
5) Compare the status and configuration grids. They should be displaying the same columns.

602 **Tab control doesn't recognize the arrow keys**

Prior to this fix, the tab controls used in the new grid controls didn't capture the focus and didn't respond to the arrow keys. A mouse was necessary to change the tab page.

604 **Add Controller button on main menu bar doesn't work**

The Add Controller button on the main frame did nothing unless the Object Explorer was open and it was the active window.

606 **Object sub-object lists not updated**

If Filter F is mapped to Supervisor S, then changing the mapping of motors to Filter F should automatically update the motor sub-object list for Supervisor S. This wasn't happening when the mapping was done using the Motor Map button on the Filter Summary. The problem was best demonstrated by configuring the Motor Summary to display motors for Supervisor S and then modifying the motors mapped to Filter F from the Filter Summary.

608 **Menus not displayed properly**

The menus at the top of the main frame were not being displayed properly.

610 **Wrong tab page activated when Summary is initially opened**

When a summary window is opened, its active page is restored from the .INI file. But, there was a bug that caused the active view for the window to be set to the first page. This caused all keyboard events to go to the first page, even when a different page was being displayed.

611 **Summary Window Doesn't Display ToolTips**

Prior to this fix, the Summary window wouldn't display ToolTips. Now, ToolTips are displayed when the user hovers over a tab or any control on a tab page. Note that a different ToolTip will be displayed for each tab that the user hovers over. Also note that a ToolTip is no longer displayed when the user hovers over the grid, as in the 03.36 series. The ToolTip that used to be displayed in this situation is now displayed when the user hovers over the active tab.

616 **Bottom Tab Minimized when Windows Settings Change**

When a Window's setting was changed, such as the scroll bar width, while Motion Console was running with the Motor Summary window open, the bottom part of the splitter became minimized.

618 **Escaping from Grid Combo List Box Causes Memory Corruption**

When the <Esc> key was pressed while a grid combo list box was dropped down, memory was being corrupted. The problem only manifested itself in the Debug version of MotionConsole.

619 **Drop Row and Column always set to first Cell**

When a selection of cells were dragged to another grid, the selection could only be dropped at the top, left-most cell.

620 **Focus not set to grid when current cell is clicked on**

If the focus was on a tab, and the user clicked on the current grid cell, then the focus would not switch to the grid. The following procedure recreates the problem:

1) Open a summary window
2) Click on a tab
3) click on the grid cell located at coordinates 1, 1 (this cell is the default current cell)
4) note that the tab doesn't lose the focus, as indicated by the focus

621 **Vertical Scrollbar On Filter Coefficients Tab Page Not Calibrated Correctly**

When the Coefficients tab page on the Filter Summary window was selected the first time, the vertical ScrollBar was never calibrated such that it could be used to scroll the view.

Modification Type: ***CR (Change Request)***

Number	Name
612	**Remove the ability to edit tabs**

The feature that allowed the user to change the text displayed on the summary tabs has been removed.

Modified in Version: 03.37.00

Modification Type: ***NF (New Feature)***

Number	Name
600	**Generalize Configuration Object/Grid**

The method used for displaying object attributes in a grid control has been generalized so that it can be used for any set of objects. The resulting Object/Grid framework can be used in other applications, such as MotionScope. It can also be used in MotionConsole to display a set of objects that are completely separate from the objects currently displayed. For example, the framework can be used to add a grid control that can be used to configure the other grid controls.

The following changes are evident to the user when compared to the previous version:

1. The title bar of the Summary windows is thinner
2. The minimize and maximize buttons and the icon have been removed from the Summary title bar
3. The tabs look different. Some aesthetic improvements can still be made to them.
4. The tab labels can be edited by double-clicking on them. The modified name is saved in the .INI file.
5. The splitter bar that divides the status and configuration tab controls can be used to resize their heights.

The changes made were sufficiently extensive to warrant stepping the minor revision number from 03.36 to 03.37. All the settings saved in the .INI file are labeled differently from those used in the 03.36 series. Therefore, when the 03.37 version is run for the first time, it will behave as if the .INI file had been removed, i.e. no controllers will be displayed in the Object

Explorer. In fact, both versions of Motion Console share the same .INI file, but changes made while running one version will not affect the other version. Settings saved in the .INI file include: 1) the number of controllers and their names, 2) the position of the main window and each child window, 3) the axis trajectory values and 4) the default grid column widths.

Modified in Version: 03.36.16

Modification Type: *NF (New Feature)*

Number	Name
596	**Refresh related motor on Motor ConfigSet**

Modifying the configuration of a motor may cause the configuration of a related motor to also be modified. This adds the requirement that Motion Console refresh the configuration of the related motor whenever the configuration of a motor is set. The library function "meiMotorRelatedStepMotorGet()" is used to find the related motor.

Modified in Version: 03.36.14

Modification Type: *NF (New Feature)*

Number	Name
590	**Add support for transceivers D, E and F**

Configuration and status attributes for transceivers D, E and F has been added to the Motor I/O status and configuration pages. These transceivers are treated in the same way as transceivers A and B.

4.1.2 XMP Motion Scope

Modified in Version: **01.20.27**

Modification Type: ***DR (Discrepancy Report)***

Number	Name
932	**[Dup. of 931] Floating Point Data is Rounded to 3 Digits after Decimal Point**

When a pane is saved or exported, floating point data is rounded to 3 digits after the decimal point.

Modified in Version: **01.20.26**

Modification Type: ***CR (Change Report)***

Number	Name
933	**Bring 20020117 Branch into Sync with Main Branch**

Many features and bug fixes have been added to the main branch version of Motion Scope. These changes have been merged into the 20020117 branch.

Modified in Version: **01.20.25**

Modification Type: ***DR (Discrepancy Report)***

Number	Name
826	**Cannot save pane data when controller is a client type**

When the user attempts to save pane data for a pane that is connected to a client controller, nothing happens.

Modified in Version: **01.20.24**

Modification Type: ***DR (Discrepancy Report)***

Number	Name
803	**Up the Max Motion Supervisor Number to 33**

In the Trigger dialog box, the maximum number for the Motion Supervisor has been increased from 17 to 33.
In the Traces dialog box, the maximum "banded" axis number has been increased to 31.

Modified in Version: **01.20.23**

Modification Type: ***MI (Minor Improvement)***

Number	Name
790	**[Dup. of 789] Merge Japanese Translations Into Production Release Version**

Merge the current Japanese translations into the Production Release version.

Modification Type: ***DR (Discrepancy Report)***

Number	Name
796	**[Dup. of 262] Missing Help File**

Clicking on the menu item "Help/Help Topics" will now open an appropriate help document.

Modified in Version: 01.20.22

Modification Type: MI (Minor Improvement)

Number	Name
782	**[Dup. of 780] Display MPI Library Assertion Violations**

MPI library assertion violations are now displayed in a dialog box before the application exits. The user is given the choice of ignoring the error.

Modified in Version: 01.20.21

Modification Type: FE (Future Enhancement)

Number	Name
765	**Add vertical lines to MoScope indicating when motion done, in fine position, at velocity, ... status changes**

Stock "status" Traces have been added for each MotionSupervisor in the Traces dialog for AtTarget, AtVelocity, Done, InCoarsePosition and InFinePosition. These have been set up as "binary" Traces (with values of zero or one) and are the result of masking and shifting the Status value for a given MotionSupervisor. These approximate the use of vertical lines to reflect status changes.

766 **Increase Trace list to include 32 Axes**

Trace list has been extended to include 32 Axes.

Modification Type: DR (Discrepancy Report)

Number	Name
770	**Cannot open .PAN file that has no data section**

When opening a .PAN file that has no data saved in it, the "Include data" checkbox is enabled only if there is a valid data header and at least one line of seemingly valid data. If the "Include data" is not checked, then the "Read Only" checkbox is cleared and disabled (as an acquire must be done to display any data). If the "Read Only" checkbox is checked, then the "Include data" checkbox is checked and disabled (as an acquire is not allowed, so the data in the file must be used).

771 **Hex display should only be allowed for ULONG type in Edit Trace dialog.**

Hex display is only allowed for ULONG type in the Edit Trace dialog.

772 **Trigger display cutoff when "polling" motion detection method is used.**

The beginning of the data acquire where the Trigger point would normally be displayed is no longer cutoff when the "polling" motion detection method is used.

773 **Trigger detection by Polling fails with "Bad sample number" upon MotionStart.**

Trigger detection by the Polling method no longer fails with a "Bad sample number" message upon MotionStart.

Modification Type: CR (Change Request)

Number	Name
777	**The default trace list to support up 32 axes (per one SynqNet controller).**

The list of axis traces has been increased to match a default of 32 in MT766.

Modified in Version: 01.20.20

Modification Type: FE (Future Enhancement)

Number	Name
459	**MoScope Controller Reset**

A button has been added to the Pane Mode dialog to Reset the MPI Controller.

Modification Type: ***DR (Discrepancy Report)***

Number	Name
731	**Shift + LMB on Y Range Bar Slider Edge moves traces instead of scaling them**

Dragging an edge of the Range Bar with the Left Mouse Button while holding the Shift key down now rescales all the traces.

732 **MoScope needs an IP address to "talk" to a server**

MoScope failed to initialize a controller over the network when the same controller number was used as some other Pane using a controller locally. This problem was solved by not checking the usage of controller numbers for remote/server ("Client") connections. Note that erroneous data will be reported if multiple Panes refer to the same Controller by using the server to connect to the local machine, due to the limit of one application per data Recorder and one date Recorder per Controller.

747 **Pane with no Traces gets Config error on .INI file open.**

A Pane with no Traces used to cause an error on .INI file open, but it is now handled correctly.

748 **Assert occurs if YRangeBar clicked when changing focus to Pane with no Traces.**

Assert used to occur in Debug version when YRangeBar clicked when changing focus to Pane with no Traces.

764 **Pane draw loops when no data from triggered acquire.**

A bug where Pane draw loops when there was no data acquired on a trigger event has been fixed.

Modification Type: ***CR (Change Request)***

Number	Name
744	**Merge File Import with File Open**

Merged File Import with File Open. Placed "Read-only" check box in File Open dialog to replace the File Import functionality. A file opened "read-only" is not attached to a controller or data source and is not allowed to perform acquires. In this manner, multiple Panes can be opened displaying previously acquired data without regard to Controller-to-Pane limitations.

749 **Handle drop in sample numbers from Recorder (unresolved MPI error).**

If there is a drop in sample numbers from the Data Recorder due to any MPI errors, then an error message pops up and data acquisition is halted.

Modified in Version: **01.20.19**

Modification Type: ***CR (Change Request)***

Number	Name
728	**Merge Pane Open/Save with File Open/Save**

Merged Pane Open/Save with File Open/Save. Now has only one file type for settings/data files as *.pan. Presence of data is optional in the files. Removed Pane Open/Save from the menus.

Modified in Version: **01.20.18**

Modification Type: ***FE (Future Enhancement)***

Number	Name
508	**Save trace data along with pane data**

The ability to save the current trace data in a pane along with the pane configuration parameters has been added. Opening a pane file with trace data embedded in it with either File Open or File Import causes the data to be displayed.

Modification Type: ***DR (Discrepancy Report)***

Number	Name
698	**MoScope dies when zooming in**

After data is acquired and traces are displayed, MoScope dies when the user selects a zoom region and clicks on the Zoom

In button.

699 **Y-Axis labels and grid lines not redrawn after data acquisition**
After data is acquired and graphed, the Y-axis labels and grid lines disappear. They are redrawn if the window is covered up and then uncovered.

709 **File Import data with zero as first x point yields error.**
File Import data with zero as first point in the first data column (X axis) is not called an error UNLESS the scaling for the X-axis is log10. In this case, zero is an illegal value.

711 **XRange edit box not range checking entry against MaxBuffer.**
XRange edit box was not being range checked against MaxBuffer, causing an Assert in the Debug version and possible invalid settings in the Release version. Now, the maximum value of XRange is MaxBuffer.

714 **MoScope custom traces not displayed**
For a Trace whose data value stays constant for the entire XRange of the Pane (i.e., its YRange is zero), the problem of no data being displayed (and no YRangeBar) has been rectified. Now, for those cases where the data is a flat horizontal line, the YRange for that Trace is set arbitrarily to 10.

Modification Type: ***CR (Change Request)***

Number **Name**

722 **Rename File Import to File Import FFT**
Renamed "File Import" to "File Import FFT" for importing FFT files.

723 **Change command line parameter for FFT files.**
Changed command line parameter to File Import FFT files from "i" to "f". "i" is used for normal File Import files.

724 **Change File Open/Save functionality to include data in addition to parameters.**
Changed File Open/Save functionality to include data in addition to parameters using the *.mos file extension. Pane Open/Save still implements the old functionality of using settings only (and still uses the *.pan extension).

725 **File Import to read settings and data in read-only mode.**
File Import reads settings and data in read-only mode. The same files can be used as for File Open/Save (.mos files), but the Pane will not be allowed to acquire more data from any source (other than Pane Import). This will allow multiple Panes of previously acquired data to be displayed (which had been previously restricted due to the one-Pane-per-Controller limitation).

726 **Pane Import to read in data from a file in read-only mode.**
Pane Import reads in data from a file in read-only mode (and uses the *.txt extension). This is similar to File Import, except that Pane settings will not be changed by the file. Note that the number of Traces and Trace attributes need to match those when the data was created in order to produce meaningful results.

Modified in Version: **01.20.17**

Modification Type: ***MI (Minor Improvement)***

Number **Name**

685 **Use Shift Key Consistently on YRangeBar**
The following functionality has been added to the YRangeBar:

YRangeBar Slider:
Shift + Drag LMB - Move all traces

YRangeBar Edge:
Shift + Drag LMB = Modify range for all traces

YRangeBar Span:
Shift + Click LMB - Nudge all traces
Ctrl + Shift + Click LMB Extend all traces

Modification Type: FE (Future Enhancement)

Number	Name
701	**Add Shift key to LMB Ctrl functions for YRangeBar.**

To the YRangeBar the following functionality has been added:

Hi Span LMB Shift Ctrl: Extend up for all Traces.
Li Span LMB Shift Ctrl: Extend down for all Traces.
Slider LMB Shift Ctrl: Truncate for all Traces.

Modification Type: DR (Discrepancy Report)

Number	Name
467	**Pane not redrawn correctly when partially covered by another window**

If a portion of a pane was covered by a window, that portion of the pane was not redrawn when the window covering it was moved to a new location. This bug is no longer reproducible.

468 **MoScope dies after armed pane is closed**

This bug involved creating a second pane, which currently requires a second controller. This bug was reproduced by following this procedure: 1) open Motion Scope and create pane 0 and configure it to trigger on Motion Start and Buffer full; 2) create pane 1 and configure it for triggering on Motion Start and Buffer full; 3) verify that both panes trigger when motion is commanded on the appropriate Motion Supervisors; 4) save both panes to a file so that you do not have to keep recreating them after Motion Scope crashes; 5) close both panes and exit Motion Scope for a clean start; 6) restart MoScope and open the file for pane 0; 7) open the file for pane 1; 8) arm pane 0 and then close it; 9) arm pane 1 and trigger graphing on this pane by commanding motion on the Motion Supervisor associated with pane 1; 10) at this point MoScope crashed. This bug is no longer reproducible.

496 **Trace colors are reset after Traces... dialog is run**

The trace colors were being reset after the Traces dialog was run.

648 **MoScope crashes after a sequence of opening and importing.**

The following series of actions used to cause MoScope to crash.

1. Open a new pane and start some data acquisition on it (you'll need Motion Console running, of course).
2. While the pane is still graphing, import a .fft file. When it asks to close other panes, click "yes."
3. Immediately, try importing again (without doing ANYTHING else) and pick the same .fft file. Again select "yes" when asked to close other panes. MoScope should crash. This has been rectified.

670 **YOffset edit box not being updated via the Y-RangeBar.**

YOffset edit box was not being updated via the Y-RangeBar. This was a result of the removal of the slider controls.

675 **File Input hangs on second invocation while closing existing Panes.**

File Input hanging on second invocation while closing existing Panes has been fixed. A race condition was eliminated.

678 **Pane Export not supporting "hex" display format.**

Pane Export now supports "hex" display format.

700 **Missing Trace min/max info in .INI gives Assert**

Missing Trace min/max info in .INI file (for example, if there is no .INI file) gave an unnecessary Assert in Debug version. This has been fixed.

Modification Type: CR (Change Request)

Number	Name
674	**Change File Import input file Data format to not use keys per sample.**

Changed File Import input file Data format to not use keys per sample (i.e., "SAMPLE#=").

Modified in Version: 01.20.15

Modification Type: NF (New Feature)

Number	Name
592	Add Splash Screen

An MEI splash screen is now displayed as Motion Scope is coming up. The option "-s X" has been added to the command line to allow the user to disable the splash screen. If the value of X is 0, then the splash screen will be disabled.

691	Make the .INI file a command line option

The option "-p filename" has been added to the command line to allow the user to specify an alternate .INI file. The file name can be either an absolute path, or it can be a simple file name. In the latter case, the .INI file will be created in the same directory as the default .INI file, i.e., the windows directory.

Modified in Version: 01.20.14

Modification Type: MI (Minor Improvement)

Number	Name
572	"Counts" scale on MoScope displays fixed number of scale markings, rounding each to nearest integer.

Number of "Counts" scale markings for a Pane are no longer fixed but are now dependent on given window size and the range of counts in the Pane. Only exact labels are displayed (no rounding of labels is done).

664	Link Panes for Ctrl-LMB cursors operation.

For Panes created via the File Import command the ability to Link Panes for Ctrl-LMB cursors operation has been added. The File Import FFT file can specify per Plot another Pane to be linked so that cross-hair cursors are displayed in the linked Pane when Ctrl-LMB is used in the first Pane. Note that the values displayed reflect those at the same relative cursor position in each Pane. This means that the values probably make the most sense when the XRange and XOffset are the same for each Pane, as well as for the Y-axis parameters. The Panes do not have to be the same exact size, as proportions are preserved in the transformation between relative positions.

665	Y-RangeBar StepIn/Out option for application to all Traces simultaneously.

Y-RangeBar StepIn/Out option has been implemented for application to all Traces simultaneously. Applies the proportionate increase/reduction in YRange for each Trace when this option is invoked. StepInAll is invoked via LMB-Shift-DoubleClick on the Y-RangeBar slider, and StepOutAll is invoked via RMB-Shift-DoubleClick on the slider.

666	Launch MoScope with FFT file as input via command line.

MoScope can be lauched with an FFT file as input via command line. Use "-i" as the command line flag followed by the FFT import file name. Must be a legal file acceptable to the File Import menu command.

667	"Full Out" button to apply to Y-axis also.

If Pane is created from a File Import menu command, then the "Full Out" button also applies to the Y-axis. Each Trace is restored to its original YRange and YOffset settings based upon the MinRange and MaxRange parameters specified in the file Imported.

Modification Type: FE (Future Enhancement)

Number	Name
458	ZoomIn/ZoomOut with left/right mouse double click

After selecting a ZoomIn area, it is now possible to shift-double click the left mouse button within the ZoomIn area to ZoomIn. If outside the ZoomBox or if there is no ZoomBox and the Pane has been zoomed in, the same key combination will apply ZoomOut.

Modification Type: DR (Discrepancy Report)

Number	Name
642	Win98 will not import a .fft file

Errors on File Import involving messages with "Bad or missing Data SAMPLE_%u" have been eliminated.

659 **Pane Export data not properly scaled for some Traces.**
Pane Export data is not having the internal scale factor (usually a sample rate and ms/sec conversion) applied to the ActVel, TC.Velocity and Accel Traces.

673 **Ctrl-LMB on deactivated Pane leaves a little strip.**
Use of the Ctrl-LMB no longer leaves a little "strip."

Modification Type:CR (Change Request)

Number	Name
660	**File Import FFT files should support tab delimiters**

File Import FFT files requires tab delimiters instead of commas for separating columns of data.

Modified in Version: 01.20.12

Modification Type:MI (Minor Improvement)

Number	Name
453	**Difficult to line up cross-hairs at high magnification**

When zoomed-in at high magnification, it is now much easier to line up the cross-hairs with a desired point. The cross-hairs will snap to the nearest point.

647 **The icons for new, open, & save in the toolbar have white dots in their graphic**
White dots in the graphics for the New, Open, & Save icons in the toolbar have been cosmetically removed.

651 **FFT MoScope - Allow Y-RangeBar to slide (change offset) for ALL traces**
Enable all Traces to have Offset altered via the Y-RangeBar with RMB-Drag. Single (selected) Trace Offset implementation remains the same via Y-RangeBar LMB-Drag.

Modification Type:FE (Future Enhancement)

Number	Name
456	**StepOut with Right mouse double click**

Double clicking with the left mouse button on the RangeBar causes a "StepIn." It is now possible to “StepOut” using double clicking on the right mouse button. This has been implemented for both the X and Y-RangeBars.

501 **Enforce a maximum value for MaxBuffer and Range**
A maximum value for MaxBuffer and Range is now enforced. The maximum limit for MaxBuffer is now 500,000 samples. The default values for MaxBuffer and Range are now 10,000 and 5,000 samples, respectively.

502 **Print Pane feature**
The ability to print the currently active Pane has been added to MoScope via the Pane Print menu command.

503 **Print Screen feature**
The ability to print the entire Motion Scope screen has been added via the File Print menu command and the Print button on the standard ToolBar.

505 **Display Trace Name in ToolTip**
MoScope now displays the names of the closest Trace at a given location on a Pane when the cursor hovers near the Trace for an extended period of time.

506 **Add a RangeBar for the X Axis**
A RangeBar for the Y-axis has been added, similar to the RangeBar on the X-axis.

593 **Add Accel to trace list**
"Accel" has been added to the stock list of Traces.

Modification Type: DR (Discrepancy Report)

Number	Name

457 **RangeBar nudge size**
Left-clicking the mouse in a non-highlighted area of the RangeBar caused the XOffset value to increase arbitrarily. This increment has now been fixed to be half of the current XRange value.

466 **Cannot create pane for controller when Controller # is typed**
Bug fix for the following scenario: When the user opens the Pane Mode dialog box and manually enters a number into the "Controller #" combo box (rather than selecting a number), the number is not accepted after the "OK" button is clicked. The error message "Controller not available - choose again." is displayed. This issue has now been resolved.

497 **Zoombox drawn outside of pane**
Zoom boxes are now limited to be completely within the Pane area.

498 **Traces with binary values drawn at edge of pane**
Traces with "binary" values are now AutoScaled better so that they are drawn within the pane, instead of at the edges.

580 **MoScope crash with 28 traces**
When 28 or more traces are set, Motion Scope would crash. The number of Traces allowed has now been extended to 30, and the user is not allowed to enter more, so MoScope no longer crashes in this scenario.

586 **ToolTip for AutoScale is no longer valid**
After the new AutoScale features were added, the functionality of the AutoScale button changed, but the tooltip did not.This condition has been rectified. It now reads, "AutoScale all Traces to their current min and max values in current Range."

587 **Tooltip for "Step Out" is wrong**
The tooltip for the "Step Out" button should have read "increase range," as opposed to "decrease range." This has now been rectified.

588 **Holes in data when graphing continuously on Win2K**
Bug fix for the following scenario: When trigger conditions are set to "Go Button" and "Stop Button," large holes appear in the data during data acquisition. This was only noticed in Win2K (due to its slowness). This was an optimization issue and has now been resolved.

589 **Y-Scale and Y-Offset not updated when new data is acquired**
After new data is acquired, the display of Y-Scale and Y-Offset did not change, even though the actual values may have been different. Selecting a new trace forced the current values of Y-Scale and Y-Offset to be displayed. This issue has now been resolved.

633 **MoScope fails with 20000913 when ini file has MODE=2**
The condition where a crash occured when the MoScope .INI file had MODE=2 is now handled gracefully.

635 **Y axis scale in MoScope / trouble with small numbers**
Bug fix for the following scenario: The number spacing on the Y axis is uniform regardless of zoom rate. When the user zooms in really close, numbers are repeated (i.e. -1,-1,-1,0,0,0,1,1,1). This issue has now been resolved.

641 **MoScope crashes when trying to import .txt, .xls, or any other type of file than .fft types**
Bug fix of crash when File Import done on a file without FFT format (this is determined by the encoding within the file itself and not by the filename suffix).

644 **Radio Button in MoScope gets stuck**
Bug fix for the following scenario: In the "Edit Trace Properties" Dialog box, under the "Units" category, the radio button titled "counts/sec**2" gets stuck if you select it. In other words, once it's selected, you can't unselect it by selecting another radio button. Instead it actually selects both buttons. This has been resolved.

645 **Strange empty window behind new pane dialog box doesn't refresh**
Bug fix for the following scenario: This works best if no other pane windows are open in MoScope. Select the "New Pane Window" icon in the toolbar and while the dialog box is open, there is an empty window behind it. What's actually strange is that the empty window does not refresh properly if you move something over it (like the dialog box) and you get a tracer effect. This has now been resolved.

646 **Numbers along the x-axis overlap when they get above 100,000**
Labels on the X-Axis are no longer allowed to run into each other. This could happen previously when they were six or more digits long and the Pane was horizontally sized smaller.

Modification Type: *CR (Change Request)*

Number	Name
654	**Ignore MPIControlMessageLIBRARY_VERSION error from mpiControlInit()**

If, while the controller is being initialized, an MPIControlMessageLIBRARY_VERSION error occurs, then the error will be ignored.

655 **Add support for MPI version string**
The text in the "About Motion Scope" dialog has been modified to correctly display the MPI version. The version of the MPI that Motion Scope was compiled with has also been added.

Modified in Version:01.20.08

Modification Type: *CR (Change Report)*

Number	Name
654	**Ignore MPIControlMessageLIBRARY_VERSION error from mpiControlInit( )**

While the controller is being initialized, if an MPIControlMessageLIBRARY_VERSION error occurs, then the error will be ignored.

655 **Add support for MPI version string**

The text in the "About Motion Scope" dialog has been modified to correctly display the MPI version. The version of the MPI that Motion Scope was compiled with has also been added.

Modification Type: *DR (Discrepancy Report)*

Number	Name
536	**Misleading line drawn at top and bottom of pane**

A line is drawn from the point where the value of a trace goes outside the limit of what is being graphed in the pane, and where the trace value returns to within the limit of what is being graphed. This gives the appearance that the value remains flat when in fact the value is off the graph.

540 **Zoom in on traces displays nothing**
When zooming in on data that crossed in and out of the zoom region many times, often nothing was being displayed. Other times only a single horizontal line was being displayed.

577 **ZoomBox drawn to left of Pane gets "thrown" to the right**
When the ZoomBox being drawn crossed over into the area left of the Pane, instead of truncating the ZoomBox at the left edge of the Pane, the ZoomBox was immediately mirrored to the right across the right-hand edge of itself.

578 **Crash upon ZoomIn when no data**
When a ZoomBox was drawn when there was no data and the ZoomIn button subsequently pushed, MoScope would crash.

579 **Upon startup, no Traces listed in TraceBox leading to crash upon Go**
Upon startup and loading of MoScope, the active Pane saved in the initialization file would come up with no Traces listed in the TraceList box of the PaneBar. When the Go button was pushed, a crash would occur.

4.2 Utilities: Open Issues

4.2.1 Motion Console

Issue Type: ***DR (Discrepancy Report)***

Number	Name
393	**CellTips don't work for checkboxes**
	If the text of a cell does not fit within the cell of an Object Attribute Grid, then the CellTip should display the complete text of the cell. This feature does not work for cells containing checkboxes.
427	**Grid Not Always Drawn Correctly When Selection Changes**
	Sometimes, selected cells are not being drawn as selected (i.e. with the colors inverted) until some window event occurs. One way to reproduce this bug is to select the entire table by clicking on the top, leftmost cell of the grid. When this is done, some cells in the grid are sometimes not drawn as inverted, but then drawn correctly when the user clicks on the grid or hovers over a control, causing a tooltip to be displayed.
561	**Last column cannot be sized to the edge of the grid**
	The width of the last grid column cannot be moved to the edge of the grid. If the vertical scroll bar is present, then attempting to resize the last column will cause the width to snap to a distance of 4 pixels to the left of the right edge of the grid.
569	**Gray button drawn in origin cell when 1st column is minimized**
	A faulty button is drawn in the origin cell when the following procedure is followed: 1) select the entire first column of the Motion Supervisor Actions tab grid; 2) slide the column width to the narrowest possible width. This results in the gray button appearing to be a combination of all the buttons in the column.
628	**Horizontal Scroll Bar Behaves Strangely When Large Numbers of Objects are Displayed**
	When some summary windows are programmed to display a large number of objects (more than 20), then the scroll bar will behave strangely.
637	**Creative position zero behavior**
	If the controller is in open loop sine comm mode, the command position doesn't zero when the "Zero Position" button in the MS summary is clicked unless the "Clear Fault" button is clicked first.
657	**"(Not Available)" listed as an option in pull down menu**
	In the Motor Summary window, under the I/O configuration tab, all XCVR Config pull-down menus list "(Not Available)" as an option.
741	**User In bit not reported**
	The User In bit is not reported, when bit is toggled.
761	**Pull down boxes only work on primary monitor with a multiple monitor setup on win2k**
	When using Motion Console on a Win2k system with multiple monitors, the pull down boxes don't function on the secondary monitor, but work on the primary monitor.
847	**Flickering could appear on several status windows**
	Some flickering could appear on the Axis, Motion, and Motor Status pages because of a bug in how event status flags are compared.
881	**ASynq mode makes Motion Console take 100% of CPU time**
	When SynqNet is in the ASynq mode, Motion Console takes 100% of the CPU time. This make the system very sluggish.
887	**Object settings not saved prior to opening a new .INI file**
	Changes to object attributes that are stored in the .INI file are not saved when another .INI file is opened or created.

Issue Type: ***MI (Minor Improvement)***

Number	Name
739	**Add more detail to tooltips for disabled controller buttons**

When a button on the Controller Summary is disabled because the controller is not initialized, some clues can be added to help the user rectify the situation.

740	**Add greater detail to toolbar button tooltips concerning various modes of operation**

The action that is executed when a toolbar button is clicked can sometimes be modified by holding down the Ctrl or Shift keys. The nature of this modified behavior should be described in detail in the tooltip for each button.

775	**Remove Broken Wire and Illegal State for Motor I/O page**

Broken Wire and Illegal State statuses are displayed both in the Motor I/O and Status tab pages. They are somewhat inappropriate for the I/O window because there are no I/O pins associated with them.

895	**Update the Motor Event Encoder Fault Trigger configuration to match the MPI**

Motor encoder fault trigger conditions will change in MPI, so the interface in Motion Console will have to change as well.

4.2.2 Motion Scope

Issue Type: **DR (Discrepancy Report)**

Number	Name
542	**MoScope fails to draw data on Windows 98**

With triggering set to "Go Button" and "Stop Button," data will accumulate (as seen by the XOffset value changing), but no traces will be drawn. Changing the status of "View/Status Bar" will cause the pane to draw the traces. This problem occurs frequently, but irregularly. We have not found a way to reliably reproduce the problem. We have also not seen this problem on Windows NT.

643 **Odd behavior when opening a .pan file**

Here are the steps to reproduce the bug:

1. Open up a .pan file (previously created with File Save from MoScope).
2. Immediately hit the “Go” button.
3. While the plots are being generated, right-click somewhere on the pane and the graphing will mysteriously disappear.

Now, if you use the “Stop” button to halt data acquisition, click "Traces" to bring up the Traces list dialog and then hit the “OK” button, the problem will be solved and any graphing you do after this will not have this behavior.

679 **Ctrl-LMB value display hides Y-units label.**

Pane Export not supporting "hex" display format.

713 **MoScope Data not aligned with scale lines**

When collecting/displaying data, sometimes the data points don't align properly with the scale markers on the X axis. This is easiest to see by turning on the "sample band" in the Pane Display configuration and Displaying in Units of Samples. The problem can be corrected by forcing a re-draw of the data: sliding the data on/off the screen, minimizing/maximizing, or zooming in/out.

769 **MoScope hangs when opening file multiple times**

Motion Scope will sometimes hang when opening a .PAN file. This can be recreated by opening a .PAN file and then closing the pane. Repeat until the hang occurs: usually after the 4th or 5th time.

776 **AutoScale occasionally fails to utilize last portion of data in Range for selected Trace.**

AutoScale occasionally fails to utilize last portion of data in Range for selected Trace.

781 **Motion Scope displayes graph as if it missed a sample when it really didn't**

While using Motion Scope to record the sample counter while I was testing motion modify code, Motion Scope displayed some data as if it missed a sample, but while investigating the sample counter I saw that this was not the case. Perhaps there is some rounding error in the calculation of elapsed time during the motion?
The sample counter is in white. Even if MoScope missed a sample.

806 **Motion Scope looses all traces after a SynqNet node disapears**

When a SynqNet node disappears when using Motion Scope, Motion Scope displays some error messages and then the pane being used vanishes. This can be reproduced by plotting some information with Motion Scope and then pulling the SynqNet cable on the first node. This can be particularly troublesome if special traces have been set up and not saved.

852 **Time scale on Motion Scope is not refreshed upon sample rate change**

When the sample rate on the XMP is changed, Motion Scope is not aware of it.

877 **Shift key inhibits dragging of YRangeBar slider edge**

When the cursor is placed on the YRangeBar slider edge and a shift-drag is attempted, and the tooltip window is open, the tooltip window is dragged instead of the slider edge. Without the shift key down, the tooltip window closes automatically and the drag works fine.

897 **Motion Scope bit masking does not work with long data types**

Issue Type: ***MI (Minor Improvement)***

Number	Name
473	**Dialog boxes missing ToolTips**

None of the dialog boxes display ToolTips.

662	**Parameters precision (number of digits to right of decimal point) for X and Y axis**

Add parameters that provide the ability to modify the precision (number of digits to right of decimal point) for X and Y axis data labels. Add a separate parameter for the X-axis and parameters per Trace on the Y-axis.

663	**Groups to be supported in File Import input FFT files.**

Groups to be supported in File Import input FFT files.

5 MPI/MEI Libraries: Fixed Bugs

Random TIMEOUT Errors *MPI 1240*

In previous releases, occasionally there are random, MPIMessageTIMEOUT errors being returned from the MPI on specific PCs. Mostly, the errors occur from mpiControlReset(...), which internally calls meiControlSampleWait(...). The failure was caused by a bug in certain chipsets that cause the Win32 QueryPerformanceCounter(...) to randomly jump ahead by 4 to 5 seconds. The MPI uses this counter to check how much time has elapsed when waiting for a specific number of controller samples. For more details, see the report from Microsoft at http://support.microsoft.com/default.aspx?scid=kb;en-us;274323.
This problem was corrected by using a lower resolution millisecond "tick" timer from the host.

Multi-Point Motion Problem *MPI 1223*

In previous releases, if a PVT (or other multi-point motion) move was stopped and then an SCurve (or other point-to-point) move was executed, the Motion Supervisor could enter into an ERROR state. The problem was caused by the point buffer's low and empty limits not being disabled by the mpiMotionStart/Modify(...) for the second move, which triggered an E-Stop. This problem has been corrected.

mpiAxisDelete(...) Object Check *MPI 1044*

In previous releases, mpiAxisDelete(...) did not check to see if the axis object was a member of another object list. This check is needed in order to prevent an application from deleting axis objects that are appended to any motion objects. A change was made to correctly perform the "object on list" check in version 20020117.1.8, 20030120 and later. Applications which attempt to delete an axis object without first deleting its parent object or removing the axis from the motion object's axis list, will experience an OBJECT_ON_LIST error code.

Motion Modify fails in a Sequence *MPI 1035*

Previous releases did not support motion modify command in sequences. Attempting to use a command object with a motion command of MPICommandMotionMODIFY would result in a return value from mpiSequenceStart(...) of MPIMessageARG_INVALID. Support for motion modify commands is supported in release versions 20020117.1.8, 20030120 and later.

Sequencer Continuously Generates Events *MPI 1032*

In version 20011220.1.15.1 and 20020117.1.6, the sequence command MPICommandOperatorNOT_EQUAL would cause a sequencer to continually generate events to the host. The problem was caused by an improper definition for MEIXmpStatusEXTERNAL. This has been corrected in version 20011220.1.15.3, 20020117.1.8 and future releases.

meiMotionParamsValidate() fails with good motion parameters *MPI 817*

meiMotionParamsValidate() had a bug which required the same number of valid MPITrajectory structures to be specified as the number of axes associated with a motion supervisor regardless of whether or not MPIMotionAttrMaskSYNC_START or MPIMotionAttrMaskSYNC_END were specified. However, when neither MPIMotionAttrMaskSYNC_START nor MPIMotionAttrMaskSYNC_END are specified, the MPI only uses one MPITrajectory structure. meiMotionParamsValidate() has now been corrected to look for only one valid MPITrajectory stucture when neither MPIMotionAttrMaskSYNC_START nor MPIMotionAttrMaskSYNC_END are specified. This bug was fixed in the 20011011, 20020117.1.8 and future releases.

Changes to the Compensation Table Calculations *MPI 903*

Two changes were made to the compensation table calculations. The first change recast a portion of the compensation value calculation equation from a *float* to a *long*. This change was required to eliminate the immediate toggling of the compensation value on either side of a maximum compensation value in the table.

The second change removes the compensation value from the actual position that is used by the compensation table to calculate the compensation value. Now the actual position used to calculate the compensation value is calculated using only the raw encoder count and the origin. This will remove inappropriate changes in the compensation value caused by changes to the compensation value.

These two changes were made to the 371A4 firmware and the 20020117.1.5 MPI.

sim4calc.exe calibration problem — *MPI 895*

In version 20020117.1.3, sim4calc.exe could calculate incorrect look-up tables. This would cause incorrect interpolated position values with sinusoidal scales. The problem was caused by a rollver in the calculations to compensate for scale offsets in the look-up table. This was corrected in version 20020117.1.5.

Memory Access Violation with multi-point motion — *MPI 887*

In releases 20011220.1.3 and 20020117.1.3, a bug in multi-point motion (i.e. PVT, PT, spline, etc) exists, which can cause intermitent Memory Access Violation errors. The problem was caused by an invalid pointer dereference in the frame buffer handling routines. This was corrected in version 20020117.1.5.

Sample Rate change causes motor faults — *MPI 885*

In firmware version 364A3, changing sample rates could cause unexpected motor faults. This was caused by an internal timing problem between the DSP's interrupt and the serial data communication (Riptide) with local motion blocks. For a small period of time (1~2 samples), the serial data is not valid. This was corrected by waiting until the Riptide data is stable (when the sample rate is changed) before updating the background task's status. This was corrected in version 371A4.

Motion Supervisor 23 Initialization — *MPI 884*

In firmware version 364A3, the internal Axis State variable for axis 23 was accidentally initialized to an incorrect value (0x1A). This caused the MPI to misinterpret the axis's state and a Stopping Error was reported. This variable has been correctly initialized to a value of 0 in version 371A2.

mpiFilterConfigSet(...) returns PARAM_INVALID error — *MPI 879*

In version 20020117.1.3, mpiFilterConfigSet(...) would improperly return a MPIMessagePARAM_INVALID error if the Algorithm was PIV and the PostFilter.Length was non-zero. The error was caused by an internal library check to prevent the Bi-quad blocks from conflicting with the PIV algorithm. The check was not required. This has been corrected in version 20020117.1.5.

mpiControlInit(...) macro definition — *MPI 843*

In previous versions, the mpiControlInit(...) macro definition had a semicolon at the end, which caused compilation errors. This typo was corrected in version 20020117.1.5.

meiFlashMemoryVerify(...) missing from flash.h — *MPI 881*

In the 20000913 release, the FPGA and SHARC code was stored in a single flash image. Flash memory verification was possible using meiFlashMemoryVerify(...). In 20020117.1.3, the FPGA files were separated from the SHARC code. During flash download, the flash memory was modified to intialize an index table for the FPGA images. Thus, the meiFlashMemoryVerify(...) would fail if the entire image (code and data) was compared to a list of host files. So, the prototype was removed from flash.h. In version

20020117.1.5, the prototype was added back to flash.h. To use meiFlashMemoryVerify(...) successfully, you will need to first save the existing flash image to a file (.img) using meiFlashMemoryToFileType(..., MEIFlashFileTypeALL). Then, the flash image file can be compared to the controller's flash image using meiFlashMemoryVerify(..., MEIFlashFileTypeALL). See the sample program, checkFlash.c for more details. This error was fixed in the 20020117.1.5 release.

Incorrect FPGA step pulse width — *MPI 829*

In version 242 FPGA for XMP Analog controllers, there was an improperly commented block of code, which resulted in a shortening of the step pulse by 1/4. This problem was caused by a divide-by-4 piece of logic that was removed from the pulse stretch time logic. The code was changed to reinstate the divide-by-4 logic and the problem was fixed.

Device Driver port call failure — *MPI 767*

A new device driver is included in the 20020117 release to fix an existing bug (MPI723) that caused intermittent EEPROM corruption and Motion Console to crash.

This has been fixed in both WinNT and Win2000 and requires installation from the installShield release for driver replacement. To verify that the new device driver has been installed, check the date of (c:)\WinNT\System32\drivers\meixmp.sys. The date of the Win2000 driver file should be 01/21/2002. The date of the WinNT driver file should be 01/21/2002.

307C2 Frame problem — *MPI 737*

This problem was caused by incorrect handling of the UPDATE frame. This problem has only been reported in the 307 versions of the firmware. The 307D1 (307C5_307D1 branch) version was created to correct the problem (307D3 from the 307C6_307D2 branch, does NOT correct the problem). 310 versions of the firmware were tested and the problem did not occur.

IN_FINE_POSITION is incorrectly calculated — *MPI 735*

This problem was due to a change in 340A1 where the IN_FINE criteria was only checked in the STOPPED state. This was corrected in firmware 341B3.

Motion DONE occurs before State = IDLE — *MPI 734*

Motion DONE events occur at the beginning of the execution of UPDATE frames, which can be up to 2 samples before the Motion Supervisor state becomes IDLE. This will cause mpiMotionStart() calls occuring right after a DONE event to cause a MOVING error. This problem has been corrected in the latest release.

EventConfig timeout — *MPI 691*

Under specific conditions an erroneous TIMEOUT return value was returned from mpiMotorEventConfigSet(...). This only occured if a limit was disabled using a previous call to EventConfigSet(...) with Condition Logic set to MEIXmpLogicNEVER. The next call to mpiMotorEventConfigSet(...) would return a timeout. This bug has been corrected in the latest release.

mpiMotorStatus(...) and meiMotorStatus(...) error — *MPI 581*

In previous versions, mpiMotorStatus(...) and meiMotorStatus(...) returned MPIMessageARG_INVALID if the external argument was not-NULL. This was corrected in version 20010125.

mpiAxisCommandPositionSet(...) — *MPI 528*

In previous versions, mpiAxisCommandPositionSet(...) would not set the command position if the axis was in a Stop condition. No error code was returned. Note, setting the command position worked after an E-Stop or Abort action. During a Stop Action, the XMP continually calculates new command positions, but uses a feedrate of 0 (this allows us to either resume the motion or back up on path). Because the XMP is calculating the command position, the MPI is not allowed to write to the command position. In version 20010710, a check was added to the MPI to verify that the command position was successfully set. If the command position cannot be set, mpiCommandPositionSet(...) will return an error code.

mpiControlReset(...) returns too early — *23*

In previous releases, mpiControlReset(...) waited a fixed time period for the controller hardware to complete its reset. Occasionally, mpiControlReset(...) would return too early, causing MPI methods to fail. mpiControlReset(...) has been changed to monitor the controller hardware during reset, returning only after the reset completion. This was corrected in MPI version 20010130.

FrameBuffer referencing error — *MPI 632*

In version 20010403, when the motion parameter point.retain is set to FALSE with mpiMotionStart(...) , the internal method meiMotionFrameBufferLoad(...) deletes its frameBuffer after use. Later, when meiMotionFrameBufferLoad(...) is called again, the function assumes the frameBuffer is still valid, and attempts to access the members Count and Index, which results in a memory referencing error. This was corrected in version 20010807.

Executing flash utility with server option fails — *MPI 625*

In version 20010403, the utility program flash.exe did not work properly with the "-server" option. This was corrected in version 20010522.

Reset after Stop — *MPI 586*

In previous versions, performing a mpiMotionAction(MPIActionSTOP), polling mpiMotionStatus(...) for the axis to be IDLE, and then performing mpiMotionAction(MPIActionRESET) can cause the following error:

```
ERROR 0xd09: Motion: MPIStateSTOPPING
```

This occurred because the method mpiMotionStatus(...) reads directly from the XMP axis status. mpiMotionAction(MPIActionRESET) checks the status of the XMP motion supervisor status. The firmware updates the MS status from the Axis status in the background cycle. It takes at least one sample for the MS status to reflect the IDLE state from the Axis status. Therefore a MPIActionRESET immediately after the axis status changes to IDLE can produce MPIStateSTOPPING errors because the DSP has not yet updated the MS status. This was corrected in version 20010524.

mpiMotorIoGet(...) and mpiMotorIoSet(...) access different parts of memory — *MPI 573*

In previous versions, if the following MPI calls are made consecutively with the same MPIMotor object as an argument, the changes made by the first call to mpiMotorIoSet(...) could be erased because of Riptide latencies.

mpiMotorIoGet(...), mpiMotorIoSet(...), mpiMotorIoGet(...), mpiMotorIoSet(...)

These were corrected in version 20010614.

Action synchronization between the MPI and Firmware — *MPI 544*

In many cases the MPI writes a value to the XMP, the XMP processes the data, calculating a new value (foreground and/or background cycle) and then the MPI reads this new value. In some specific cases, the MPI is not protected from reading the data before the XMP has completed its processing. In these cases, the MPI will read the old value. Here are the unprotected dependencies:

1) Motor->IO.DedicatedOUT.IO (read by mpiMotorIoGet(...)) depends on Motor->IO.HostOutput (set by mpiMotorIoSet(...))
2) SystemData->Gate (read by meiControlGateGet(...) depends on SystemData->HostGate (set by meiControlGateSet(...))

This was corrected in version 20010614.

Motion supervisor pointer problem — *MPI 516*

When changing the motion supervisor to axes mapping, the motion supervisor pointer in the axes objects continues to point to a motion supervisor after the motion supervisor stops pointing to the axis. This can cause strange behavior, such as axes that are in an error state resuming motion when another axis is set in motion. This problem only arises when an application changes the mapping of axes to motion supervisors in the middle of an application and does not reassign the old axes to new motion supervisors.

Motion Modify Problem — *MPI 769*

In version 2000072803, if mpiMotionModify(...) was called during a Stop action, it could have incorrectly returned an MPIMessageOK without having modified the move. This problem was caused by an improper error check in the mpiMotionModify(...). At the end of a Stop action, if the command velocity was zero and the Done status bit had not been set, the MPI would incorrectly consider the Stop action complete. This problem was corrected in versions 2000072804 and 20011213. Now, the mpiMotionModify(...) routine correctly checks to see if the Stop action is complete. If it is not complete, it will return a STOPPING error code.

mpiRecorderRecordGet() returns corrupted data — *MPI 713*

A recorder overflow can occur whenever the XMP fills the record buffer faster than the MPI can remove the data. Since the record buffer is circular, an overflow can cause new records to overwrite older records. In version 20000913, if the MPI read records when an overflow occured, the data might have a mix of new and old records. The overflow recovery has been improved in version 20011213. When an overflow occurs, the MPI only reads the new records and omits the old records, preserving data integrity.

Motion Modify does not work when command position is reached — *MPI 697*

The firmware motion supervisor code would not change the command position when a call to mpiMotionModify(...) was made after the command position had reached the target position even though the motion had not been completed (met the settling criteria). This problem has been fixed in the 341B2 firmware. A call to mpiMotionModify(...) can now be made before or after the command position has reached the target position.

mpiMotorConfigGet() error — *MPI 688*

In version 20010417.1, mpiMotorConfigGet(...) would return a MEIMotorTransceiverConfigNOT_AVAILABLE error if both the MPIMotorConfig and MEIMotorConfig structures were passed. This was caused by the MPI not reading the Stepper ResourceNumber from the controller. This bug was corrected in version 20011213.

Incorrect Motion Profile with mpiMotionModify() — *MPI 686*

In Version 325B2, if mpiMotionModify(...) was called with the same motion parameters as the executing move while moving in the negative direction, the resultant motion profile would have a discontinuity. This problem was fixed in firmware version 347B1.

Motion Modification Bug for Velocity Moves — *MPI 683*
In version 325B2, mpiMotionModify(...) would cause a trajectory discontinuity with velocity type moves. This was fixed in version 347B1.

Config Utility does not save DRate coefficient — *MPI 672*
In previous versions of the config.exe utility, the filter coefficient value for DRate was being incorrectly saved. Upgrading XMP configurations using a config utility output file from a previous version will restore incorrect values for DRate. MEI suggests setting the configuration using the config utility and then manually modifying the DRate in Motion Console. This suggestion only applies to customers using the PID control algorithm and the config.exe utility for XMP configuration. This discrepency has been resolved in the 20011213 release.

mpiControlReset(...) locks up PCI bus — *MPI 659*
In firmware version 310B (and older), under certain, very rare timing conditions, an mpiControlReset(...) could lock-up the PCI bus for half a second. This was caused by an internal controller flag that was monitored by the firmware during a reset. This potential problem was corrected in firmware version 347B1.

Modification of Velocity Integrator term in PIV loop — *MPI 639*
In firmware version 310, Kiv was not multiplied optimally. This resulted in the contribution of the integrator actually exceeding the limit, because the limit was applied before multiplying by Kiv. This caused nonlinear behavior and integrator windup. This problem was corrected in firmware version 310J1. The integration limit is now applied after summation, which limits the output of the integration portion of the velocity loop.

Kfff bug in PIV algorithm — *MPI 637*
Friction Feed Forward (Kfff) was incorrectly applied to the filter output in firmware versions 295A3 to 310B3, when using the PIV algorithm. Kfff is applied correctly with the PID algorithm, but in the PIV algorithm, Kfff was added in two places: in the velocity loop and after the velocity loop. As a result of Kfff being incorrectly applied with the PIV algorithm, there is potential for an unexpectedly high output in the direction of commanded motion, which could cause instability. This problem only occurs if Kfff is set to a non-zero value. The default value for Kfff is zero. In the fixed version, Kfff is added only after the velocity loop. This was corrected in firmware version 307C4 and 310J1.

MPIMotionTypeSPLINE motion error — *MPI 633*
Calling mpiMotionStart(...) with motion type MPIMotionTypeSPLINE generated a profile with a large command position jump near the end of the move. This was caused by an error in the algorithm that looked one point beyond the end of the positions list. As this value was uninitialized, the resultant profile included a potentially large command position jump. This was corrected in the 2000091303 release.

No AT_TARGET with path motion — *MPI 630*
In the 2000072802 release AT_TARGET status and event were not issued during path motion when each axis had different target values. This problem was corrected in the 2000072803 release.

PIV parameter structure mismatch — *MPI 603*
In the 20000913 release, a mismatch between the DSP PIV Algorithm and the MEIFilterGainPIV structure existed. This mismatch caused the last 15 PIV filter coeffecients to be incorrectly labeled. This mismatch was fixed in the 20001103 release.

meiFrameBufferLoad(...) empty limit disable bug — *MPI 584*

In previous versions, a boundary condition existed in the path motion algorithm, which occasionally caused the controller to stop an axis with an unexplained error state. The problem occured when the last frame buffer load was exactly 64 frames. The load algorithm did not check to see if the 64th frame was the last frame. This caused a buffer low event, stopping the axes, and triggering a false EMPTY_LIMIT. This was corrected in version 20010131.

Position Error requires two ActionRESETS — *MPI 574*

In previous versions, when an axis was ABORTed and there is a position error, mpiMotionAction(..., MPI-ActionRESET) did not always clear the position error properly. Sometimes, mpiMotionAction(...) needed to be called twice. This problem was corrected in firmware version 320A1.

User Limit race condition — *MPI 565*

In previous versions, a possible race condition exsited between the MPI and the XMP firmware with user limits. The problem occurs when the XMP executes a limit condition while the MPI is writting to the limit structure. If the MPI sets the outputPtr to zero while disabling an enabled limit there is the posibility of dia-bling the execution of the XMP's DSP.

This problem was corrected by adding a handshake state variable between the DSP and the MPI. The possible states are IDLE, MODIFY and DONE. In the MPI, when meiMotorEventLimitSet(...) is called, it waits for the syncronization word (modifyState) to be IDLE. If the state is IDLE, then the MPI sets the state to MODIFY, writes the limit structure to XMP memory, then sets the state to DONE. The firmware will not process the data if the state is not IDLE. This problem was corrected in version 20010105.

DAC limit and OutputOffset Changes — *MPI 562*

In previous firmware versions, when configured for PID or PIV filter mode, the OutputOffset filter coefficient was added to the output value after the bipolar output limits were checked. For example, if the PID or PIV filter output was 32767 (10 volts) and the OutputOffset is 0, the DAC value is 32767. If OutputOffset was 1, the DAC value is 32768 (-10.0) volts. This problem was corrected by bounding the filter output value to 16 bits, before writing to the DAC. Also, the OutputOffset is now applied before the output limits. This problem was corrected in firmware version 320A1 and 310J1.

Encoder termination always set — *MPI 551*

In version 200000913, encoder termination was always active and could not be cleared. Encoder termina-tion can now be turned on or off by using the encoder termination field in the MEIMotorConfig{...} data structure when calling the MEIMotorConfigSet(...) method. This bug was fixed in version 2000091302 soft-ware.

Multi-axis motion modify during acceleration — *MPI 525*

In version 310B3 firmware, if two axes were accelerating towards a target position (in the same Motion Supervisor) and the motion was modified to reach a further target position, the acceleration changed to 0.0 and after a while, the acceleration resumed back to its original commanded value. This problem was caused by the point-to-point algorithm for multiple axes. This problem was corrected in the new S-Curve/ Trapezoidal algorithm, which was implemented in firmware version 325B1.

mpiMotionAction(RESUME) does not resume motion — *MPI 524*

In firmware version 310B3, when a STOP occured, the firmware decellerated the axis by decreasing the feedrate. Once the feedrate became 0, the axis settled. Once settled, the done bit was set and the axis entered the IDLE state. A MPIActionRESUME cleared the STOP bit, but did not change the state of the axis. Since the axis was still in the IDLE state, the MS set the feedrate to 0, so that the axis would not resume motion. This problem was fixed in version 310B5 and 325A1.

Zero time value in S_CURVE Frame from MotionModify *MPI 519*

In previous versions, if mpiMotionModify(...) is called during the first sample of the previous motion profile, the firmware would incorrectly calculate zero for the frame execution time. This causes the motion profile to halt execution. This was corrected in firmware version 311A1.

mpiCaptureConfigGet/Set Bug *MPI 482*

In previous MPI versions, mpiCaptureConfigGet(...) did not return proper values for mask and pattern. This was corrected in version 20010119.

6 MPI/MEI Libraries: Outstanding Bugs, Limitations

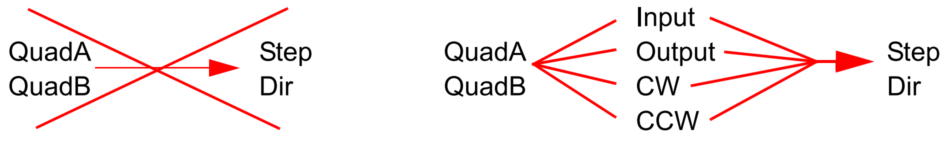
6.1 Known Bugs

Win2000 Device Driver System Stand by Error *MPI 741*

The XMP Windows 2000 device driver will not allow a host system go into "Standby" or "Hibernation" mode. This bug will be corrected in a subsequent release.

Motor XCVR configuration QUADA dependence *MPI 508*

Changing XCVRA or XCVRB from QuadA or QuadB, to Step or Dir will result in XCVRA or XCVRB remaining configured for QuadA or QuadB. In order to change XCVRA or XCVRB from QuadA or QuadB to Step or Dir, first change QuadA or QuadB to any other value except Step and Dir, such as Input, Output, CW, or CCW. Once it has been changed to one of these other values, it is possible to configure the XCVR's for Step and/or Dir. (*See 564 in section 4.21 Motion Console: Open Issues*)



6.2 Known Limitations

Motion Modify Overshoots — *MPI 836*

An overshoot will occur when mpiMotionModify(...) is called:
- with the same motion parameters that were used with the mpiMotionStart(...) method.
- with a deceleration greater than the acceleration.
- during the deceleration part of the move.

WinNT Driver Invalid Board Number Bug — *MPI 568*

The MEIXMP device driver can support a maximum of 8 XMP-Series controllers.

Firmware support for jogging — *MPI 554*

The MPI has a motion type for jogging (MPIMotionTypeJOG), but presently the firmware does not support it.

Brake Enable/Disable Delay — *MPI 533*

The Brake feature sets the User Output to an Active state when an Abort Event occurs. The "Brake Delay" specifies the amount of time to delay between the Abort Event and setting the User Output bit. Presently, the only way to clear the Brake is with a Controller Reset.

Frame buffer overwritten by Start/Modify append — *MPI 532*

Each axis has a 128 frame buffer (FIFO). Motion Start and Motion Modify calls will load up to 10 frames. No provision has been made to check if the new frames will overwrite currently executing frames. This could happen after about 12 Start/Modify calls are made with the APPEND attribute.

Gear Ratio with Stepper Axes — *MPI 522*

The MEIXmpAxisGear firmware feature only supports servo motor types. The axis gear feature does not support step motor types.

MEI Motion Attribute limitations in Sequences — *MPI 488*

The following MEI motion attributes are supported in motion sequences:
- MEIMotionAttrMaskFINAL_VEL
- MEIMotionAttrEVENT

Other motion attributes will be available in future releases.

MPI motion attribute limitations in Sequences — *MPI 487*

The following MPI motion attributes are supported in motion sequences:
- MPIMotionAttrMaskID
- MPIMotionAttrMaskDELAY

Other motion attributes will be available in future releases.

PT/PVT Motion Types currently unsupported in Motion Sequences — *MPI 486*

These motion types will be available in future releases.

BSpline motion — *MPI 470*

AUTO_START is not yet supported for BSpline motion.

Non-integer relative moves **442**
When successive non-integer length relative motions are commanded, the fractional portion is truncated and discarded. This may cause problems if the fractional value needs to be summed over multiple moves.

Axis jumps on frame buffer underflow **435**
If E-stop deceleration rates are not set high enough to stop within the number of frames specified by the empty frame limit, the axis jumps on a frame underflow. The axis will E-stop along the path of the last frames in the buffer, then continue onto the next frames (which are the frames from 128 frames ago). This can potentially cause a dangerous condition.

MS/Axis Mapping Error Code **CRN 353**
Misleading time-out errors are returned when trying to manipulate improperly mapped motion supervisors.

Motion Modify with Delay **CRN 289**
MPI motion with Modify is not supported with the Delay attribute.

Motion Events with Motion Supervisors sharing axes **CRN 243**
When using multiple Motion Supervisors that share axes, Motion events (Done, AtVelocity) are sent to both Motion objects, no matter which Motion Supervisor commanded the motion. This occurs, because the Motion events are derived from the Motion Supervisor status, which is derived from each axis' status.

Software Position Limit can produce both Positive and Negative limit events **CRN 13**
When the distance between the positive and negative limit configurations exceed 32 bits (4,294,967,296 counts), both limits are triggered. The distance between the positive and negative software position limits must be less than 32 bits (4,294,967,296 counts).

Long point-to-point moves **06**
The XMP firmware velocity frame execution time cannot exceed 16,384,000 samples. With the sample rate configured for 2000 (default), the maximum velocity time is 2.27 hours. If the commanded motion exceeds the maximum frame time, the axes will stop abruptly after 16,384,000 samples. The motors will still maintain servo control and no errors are reported.