



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: 459B1
 MPI Library Version: 20030620.1.1

Revised 26June2003
 DCR 706

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/2000/XP. The distribution was built using Visual C++ v6.0 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:

	New Version	Previous Version
Firmware	459B1	459A6
MPI Library	20030620.1.1	20030605
Motion Console	03.39.07	03.39.05
Motion Scope	01.21.06	01.21.06



ATTENTION!



This release contains a new Rincon FPGA image. The Rincon FPGA is responsible for Synq-Net communication on the XMP-Series controllers. Version 1.40 and later versions require a minimum FPGA part size, Xilinx XC2S-100 (or larger). Early XMP-SynqNet-PMC controllers (PN: T009-0001) may have a small FPGA, Xilinx XC2S-50. If your controller has the Xilinx XCS-50 (located at U24), you will need to return your controller to MEI for an upgrade to a larger component. Please contact MEI for upgrade details. The MPI software will automatically determine the FPGA size and download the appropriate image. If the MPI does not find the appropriate image, it will return a "File Open Error" message.

Copyright © (26June2003) Motion Engineering, Inc.

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 or in part, without the express written permission of Motion Engineering, Inc.





ATTENTION!



This release supports SynqNet controllers and SynqNet Phase II nodes only. If you are upgrading from a previous software release and you experience problems with Motion Console or Motion Scope, then delete the .INI files for these applications. Some INI files from older releases may not be compatible with Motion Console or Motion Scope. Deleting the .INI files will cause Motion Console and Motion Scope to regenerate new INI files. Your INI configurations will be replaced with default application configurations. The WinNT Motion Console and Motion Scope INI files are named: MC_XMP_NT.ini and MS_XMP_NT.ini.



<http://support.motioneng.com>

INTRODUCING MEI'S UPDATED ON-LINE DOCUMENTATION SYSTEM

Featuring:

- Up-to-Date Information
- Keyword Search Engine
- SynqNet Documentation
- Dynamic Navigational Menu
- Print-friendly PDFs

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

Synqnet Phase II

Introduction

This software release supports SynqNet™ controllers. SynqNet is a high performance synchronous network designed for multi-axis motion control applications. For more information about SynqNet, please contact MEI or visit the SynqNet website at <http://www.synqnet.org>.

SynqNet Phase II Development

The SynqNet development has occurred in two major phases. SynqNet Phase 1 ported internal XMP-Series controller communication protocols to run over 100BaseT network hardware. This allowed MEI to develop SynqNet with minimal software changes. It also allowed our drive partners to design and build SynqNet servo drives. Due to the protocol internals, SynqNet Phase I has several limitations. It only supported 6 nodes, string topologies, non-configurable packet data, and a maximum update rate of 5kHz.

SynqNet Phase 2 is an upgrade to the SynqNet internals. It is a fully scalable network protocol that offers higher performance, reliability, and several additional features including:

- Up to 32 nodes on one network.
- Scalable networking timing.

- Software configurable data packets.
- Network fault recovery when a single cable is broken (ring topologies only).
- Failed Node Tolerance.
- Downloadable FPGA and drive binary code over SynqNet.

Beyond Phase II, more features will be added with future software upgrades.



SynqNet software releases do NOT support XMP-Series Analog, SERCOS, or Pulse controllers.

Features and Limitations

This release supports the SynqNet Phase II protocol. The following are available in this release:

- Up to 32 nodes on one network
- Scalable networking timing - The MPI automatically scales the packet timing based on the node number and type. The maximum controller/network sample rate and servo control latency depends on controller/network load. The minimum sample rate is 1kHz.
- Cable Length - Network cable lengths of up to 100 meters are supported. The network timing calculations automatically compensate for various cable lengths. The MPI also supports configurable cable lengths.
- Software configurable data packets - The MPI automatically configures the packets during network initialization based on the node type. The MPI also supports user configurable packets.
- String (open or closed) and Ring topologies are supported.
- Node FPGA download via SynqNet - RMB-10v (via Outrigger circuit) and all other supported nodes (via Bowsprit circuit). See the list of Supported SynqNet nodes below.
- Time-Based Capture - This is a special technique to derive a captured position from a node's latched time value, the controller's actual position, and the previous sample's actual position. This feature is useful for high accuracy capturing when the controller reads position feedback from a drive (not from the node's FPGA).
- Bad Packet Feedback Compensation - The controller maintains the previous sample's actual position and calculates the actual velocity. If the feedback packet is corrupted, the controller will interpolate the expected actual position. This feature prevents motor jumps due to network data integrity problems.
- Network fault recovery with ring topology - In a ring topology, if any one cable fails, the network will redirect packet data around the break and notify the application with an event. The location of the break can be determined by the application.
- Failed Node Tolerance - If one or more nodes fail, the network will continue to operate in SYNQ mode, communicating with the good nodes.
- Topology Save to Flash - The network topology can be saved to the controller's flash. After a reset, the controller will automatically initialize the network and verify that the number and type of discovered nodes match the expected topology.

The following features are NOT supported in this release:

- Compare
- ADCs
- Stepper (pulse) motor types

Supported SynqNet Controllers

All XMP-Series SynqNet controllers are supported with this release. While XMP-Series SynqNet controllers having local motion blocks are supported, the local motion blocks themselves are not supported. Early XMP-SynqNet-PMC controllers (T009-0001) may have a small FPGA, Xilinx XC25-50. If your controller has the XC25-50 (located at U24), you will need to return your controller to MEI for an upgrade to a larger component. Please contact MEI for details.

Please see the support website (<http://support.motioneng.com>) for a complete list and details about SynqNet controllers.

Supported SynqNet Nodes

The following SynqNet node types are supported:

- MEI RMB-10V-SynqNet
- MEI RMB-10V2-SynqNet
- Yaskawa DMD/IDM
- Yaskawa Sigma SGDS
- Kollmorgen DASA
- Kollmorgen CD
- AMC Digiflex
- Panasonic Minas B
- Sanyo Denki Q-Series
- Glentek Omega

Future Phase II releases will add support for other node types and SynqNet drives as they become available.

Please see the support website (<http://support.motioneng.com>) for a complete list and details about SynqNet nodes.

Upgrading to SynqNet Phase II

If you have SynqNet Phase I hardware/software and would like to upgrade to SynqNet Phase II, please contact MEI and the node manufacturer. SynqNet controllers can be upgraded in the field by downloading new firmware and a new FPGA image. This can be performed with Motion Console or the flash.exe utility. Most SynqNet nodes must be returned to the factory for boot ROM upgrade. Once you have Phase II compatible nodes, then binary FPGA images and drive firmware can be downloaded. Not all nodes support download, so be sure to contact the node manufacturer for the latest information.

After nodes have been upgraded to Phase II, all future upgrades will be possible via software download.

Node FPGA Binary Files

Please see the Node Binary Files: Product and Features Tables on the support website (<http://support.motioneng.com>) for details about binary file to product compatibility and feature set.

Changes in Phase II

We strived to keep the changes to a minimum. But, to take advantage of new features and Phase II protocol, some software changes were required. Here is a summary of the major changes:

MPI

The MPI has been expanded to support SynqNet Phase II. The SynqNet specific methods and structures are located in *synqnet.h* and *sqnode.h*. Please see the support website (<http://support.motioneng.com>) for details.



If you are porting application code from SynqNet Phase I, then the SynqNet portion of your code will need to change to match the new Phase II interface. The motion portions of your application should not require changes. The I/O portions of your application will require changes.

Dedicated I/O Logic

In Phase I, the Amp Fault, Home, and +/- HW Limit default trigger logic was active low. The default configuration was based on Analog controllers. For SynqNet Phase I drives the default trigger logic was backwards (active low, instead of active high). In SynqNet Phase II, the default trigger logic is active high for the Amp Fault, Home, and +/- HW Limits. The default logic is now correct. Make sure to check your Dedicated I/O trigger logic.

Block to sqNode

In Phase I, the block object was responsible for local and remote motion blocks. Each physical block had a fixed set of resources to support 4 motors. In Phase II, local motion blocks are NOT supported. Remote blocks are SynqNet nodes that can support 0 to 8 motors. The block object was replaced with the sqNode object and modified to support Phase II features.

SynqNet to sqNode

In Phase I, the SynqNet module contained methods and structures to read network info and status from the controller and nodes. Also, it provided an interface to send service commands, clear faults, get/set the drive configuration and read drive info. In Phase II, the SynqNet node specific methods and structures have been modified and moved from *synqnet.h*, into *sqnode.h*. Now, only network methods and data are handled through the SynqNet object and SynqNet node methods and data are handled through the sqNode object.

DriveLib to SqNodeLib

In Phase I, the Drives Library handled the Drive specific initialization and configuration. For each drive type, there was a drive specific header file that contained structures and methods to interface to the drive specific features.

In Phase II, the Drives Library was renamed to the SqNodeLib. The basic concept is the same, but has been expanded to include support for all types of SynqNet nodes (not just drives). For each SynqNet node type, there is a node specific header file. The node type and node type name can be determined using `meiSqNodeInfo(...)`. The name of the node specific header file matches the `nodeName` from `MEISqNodeInfo{...}`.

Network Initialization

In Phase I, the network was automatically initialized when `mpiControllnit(...)` was called. The MPI Library handled all network initialization.

In Phase II, the network initialization responsibility is split between the controller and the MPI library. The controller begins the network initialization by checking network integrity, then resets the nodes, and discovers each node. If the controller finds the expected topology it will transition the network to cyclic mode, stream down the FPGA configuration data to each node and return control to the MPI library to complete any node specific initialization. If the controller finds a topology that doesn't match its expected topology, it will return an error. If the controller doesn't have an "expected" topology, then the MPI Library will transition the network to cyclic mode, stream down the default FPGA configuration data (via the controller), and complete any node specific initialization.

Network Verification/Confirmation

In Phase I, the network topology was always discovered by `mpiControlInit(...)`. Any configurations saved in the controller's flash memory would be loaded on a power-up or reset and the FPGA service commands would be sent to the nodes. No matter what network topology was found during discovery, the controller flash configurations would be applied.

In Phase II, the controller determines whether the discovered topology matches the expected topology, does not match the expected topology, or is not specified (default). If the topology information is saved to the controller's flash using `meiSynqNetTopologySave(...)`, then the controller will compare the discovered topology to the expected topology. If the discovered topology matches the expected topology, the controller will send its FPGA configuration data (from flash memory) to the nodes. If the discovered topology does not match the expected topology, then `mpiControlInit(...)` will return a `TOPOLOGY_MISMATCH` error and the FPGA configurations will not be sent to the nodes. To recover from this error, the network topology must be corrected, or the new topology must be saved to flash with `meiSynqNetTopologySave(...)`. The topology can also be cleared with `meiSynqNetTopologyClear(...)`. If the topology information is not saved to the controller's flash, then the MPI library will send the default FPGA configuration data to the nodes.

Packet Errors

In Phase I, the controller and each SynqNet node had one CRC error counter. In Phase II, the controller and each SynqNet node has CRC error counters on each port, a packet error counter and a packet error rate counter.

Capture

In Phase I, the FPGA capture logic was copied from the XMP-Series Analog controller. Each motion block (4 motors) contained 10 configurable capture blocks. The capture object was indexed by number and statically allocated in the controller's memory.

In Phase II, the FPGA capture logic has been redesigned. The capture object association with a motor object is now configurable. The motor object now supports two encoder interfaces. Initially, each encoder interface supports one capture block. In the future, the number of capture blocks per encoder will be variable depending on the FPGA logic.

`mpiCaptureCreate(...)` creates a capture object associated with the specified controller's capture number. Make sure to configure enough captures with `mpiControlConfigSet(...)`.

`MPICaptureConfig{...}` has been changed to support configurable capture trigger sources, edge, type (position or time based), a capture motor number, a feedback motor number, an encoder, a capture index, and a global trigger.

`MPICaptureStatus{...}` has been changed to support a single latched position value per capture.

External E-Stop

In Phase I, the controller and each remote block had an External E-Stop input. This input could be used to trigger an action (Stop, E-Stop, or Abort) on each motor.

In Phase II, the External E-Stop input on each node was renamed "Node Disable." `XEStop` was

replaced with "User Fault." The User Fault has the same capability as the XEStop, plus it adds the ability to command an IoAbort to a node. The MEISqNodeConfig{...} structure contains a userFault configuration. Any controller memory address can be used to generate the userFault. The MEISqNodeConfigIoAbort{...} structure contains a flag to enable/disable the IoAbort command to the node. The MEIMotorConfig{...} structure contains a userFaultAction to configure an action when the userFault occurs. There is one userFault for each node.

IoAbort

In Phase I, the IoAbort signal for remote nodes could be enabled or disabled. When a network fault occurred, the IoAbort (if enabled) would force the remote block outputs to their power-up state.

In Phase II, the IoAbort is configurable. The input conditions for a SynqNet node IoAbort can be configured with meiSqNodeConfigGet/Set(...) using the MEISqNodeConfigIoAbort{...} structure.

MEIMotorConfig{...}

In Phase I, MEIMotorConfig{...} contained support for transceiver I/O configuration. The transceivers were XMP-Series Analog and RMB-10v specific circuits. The features associated with the transceivers were hardware specific.

In Phase II, the transceiver support was removed from MEIMotorConfig{...}. It was replaced with a general purpose MEIMotorIoConfig{...} structure to support various types of configurable I/O. There are 16 bits of configurable I/O per motor. The physical implementation and actual number of configurable motor I/O is node/drive specific. The generic bit masks are enumerated in MEIMotorIo.

A new structure MEIMotorFaultConfig{...} was added to support configuration for the MEIMotorDedicatedInAMP_FAULT bit. The MEIMotorFaultMask can be used to select the trigger conditions for the MEIMotorDedicatedInAMP_FAULT.

MEIMotorInput/Output

In Phase I, these enumerations contained the bits masks for all motor I/O. These enumerations included a mixture of Dedicated, Transceiver, SIM4, User, Capture, and Compare bit masks.

In Phase II, the MEIMotorInput/Output enumerations were replaced with Dedicated I/O and Configurable I/O enumerations. The MEIMotorDedicatedIn/Out enumerations contain bit masks for Dedicated I/O that is common to all SynqNet motor and drive interfaces. The MEIMotorIo contains bit masks for the configurable motor I/O. The configurable motor I/O number and mapping is node specific. You can use the MEIMotorIo bit masks or the node specific bit masks located in the node specific header files (SqNodeLib).

MEIMotorStatus{...}

This structure has been extended to add support for SynqNet MEIMotorFaultStatus bits.

Amp Fault Message/Clear

In Phase I, there were methods to read an Amp Fault message from a drive or clear the Fault message buffer. The meiSynqNetAmpFault(...) method was used to read the drive specific fault message(s) and meiSynqNetAmpFaultClear(...) was used to clear the message buffer.

In Phase II, these methods have been replaced with more generic meiMotorAmpFault(...) and meiMotorAmpFaultClear(...) methods. The MEIMotorAmpFaults{...} structure is used to retrieve the fault codes and string messages. The definitions for the faults codes are drive specific. Also, support has been added to read/clear drive warnings with meiMotorAmpWarning(...) and meiAmpWarningClear(...).

Monitor

The monitor packet field data is now configurable (for drives that support it). `MeiSqNodeDriveMonitorConfigGet/Set(...)` can be used to configure each monitor field based on an index or address.

Node Download

A new method, `meiSqNodeDownload(...)` has been added to support FPGA and drive firmware download.

Utility Programs

The utility programs have been updated to support SynqNet and several have been added to support SynqNet specific features. Documentation for all the utilities can be found at <http://support.motioneng.com> under the Utilities section.

sqTopologySave - Save or clear the network topology to the controller's flash memory.

sqDriveMsg - Displays all warnings and faults for the specified drive.

sqDriveConfig - Configure or display drive parameter configuration for the specified drive.

sqDriveMonitor - Configure or display a drive's monitor fields.

sqDriveParam - Read or write drive parameters.

mtReset - Clear multiturn data on ABS encoders for drives that support them.

sqCmd - Sends a service command to a node/drive.

sqNodeFlash - Downloads FPGA or drive firmware to a node/drive.

sqNodeMemory - Display, read, or write to node memory.

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++ 6.0.

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.



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.

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. To start the set-up process, click on WinNTSetup.exe. 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, XMP\bin\Win2000 for Win2000, and XMP\bin\WinXP for XP under the default installation directory.

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 and follow the instructions.

2 General Changes / New Features

This section lists the changes since the 20020117.1 production release, beginning with the most recent.

 - Denotes modifications that may require changes in code.

Version 20030620.1.1

	New Version	Previous Version
Firmware	459B1	459A6
MPI Library	20030620.1.1	20030605

2.1 New CD Drive Fault Bits

MPI 1145

In version 20030620.1, support for new Kollmorgen CD drive fault bits was added to the kollmorgen_cd.h module. These bits are available in drive firmware version 1.1.0:

CDFaultCodeFEEDBACK_LOSS - logical OR of AB_LINE_BREAK2, ILLEGAL_HALLS2, INDEX_LINE_BREAK2, ENCODER_NOT_INITIALIZED, ENDAT_FAULT, and A_B_OUT_OF_RANGE.

CDFaultCodeAB_LINE_BREAK2 - A break in the encoder AB signals was detected.

CDFaultCodeILLEGAL_HALLS2 - An illegal hall state was detected.

CDFaultCodeINDEX_LINE_BREAK2 - A break in the encoder index signal was detected.

CDFaultCodeENCODER_NOT_INITIALIZED - The EnDat encoder was not initialized.

CDFaultCodeENDAT_FAULT - There was a fault with the EnDat encoder.

CDFaultCodeA_B_OUT_OF_RANGE - When detecting line break in sine encoder and resolver we also test the condition $\sin^2 + \cos^2 = 1$ (ideally). If this condition is not met, it means that there is a problem with the feedback signal (line break).

CDFaultCodeMOTOR_OVER_TEMP - The motor over temperature was detected.

After an amp fault occurs, the fault codes can be read from a drive using `meiMotorAmpFault(...)`.

2.2 Drive Parameter Changes

MPI 1144

In version 20030620, a new software module, drivemap.h/c was added to handle the parsing of drive map files. This caused several changes to the sqNode module:

MEISqNodeDriveParamType{...} - Moved to drivemap.h and renamed as MEIDriveMapParamType{...}

MEISqNodeDriveParamAccess{...} - Moved to drivemap.h and renamed as MEIDriveMapParamAc-

cess{...}

MEISqNodeDriveParamInfo{...} - Moved to drivemap.h and renamed as MEIDriveMapParamInfo{...}

MEISqNodeDriveParamValue{...} - Moved to drivemap.h and renamed as MEIDriveMapParamValue{...}

meiSqNodeDriveParamGet/Set(...) and **meiSqNodeDriveParamListGet/Set(...)** - Changed arguments from MEISqNodeDriveParamType to MEIDriveMapParamValue, and MEISqNodeDriveParamValue to MEIDriveMapParamValue.

meiSqNodeDriveParamFileGet/Set(...), meiSqNodeDriveMapParamCount(...), meiSqNodeDriveMapParamList(...), meiSqNodeDriveMapConfigCount(...), and meiSqNodeDriveMapConfigList(...) - Changed several arguments.

See the MPI Software section on <http://support.motioneng.com> for details.

2.3 SynqNet Network Initialization Optimization

MPI 1143

In version 20030620, the SynqNet network initialization was optimized for the case when no nodes are connected to a controller. Previously, if no nodes were connected to the controller, the network initialization routines would not return until several internal timeouts had expired. For example, methods like `mpiControlInit(...)`, `mpiControlReset(...)`, and `mpiControlConfigSet(...)` would require several seconds. A check was added before the network initialization in order to verify that the link status is good for the controller's Out port. If the link status is bad (no node connected), the network initialization routines will exit immediately. This will eliminate the large software initialization delays for applications that initialize before the node hardware has been powered on.

WARNING!

When you power on your node hardware, make sure to wait at least 2 seconds before initializing the network. Some nodes require up to 2 seconds before the hardware completely boots. If you attempt to initialize the network before the nodes have completely booted, some nodes may not be discovered.

2.4 New Multi-Point Motion Types: PTF and PVTF

MPI 1101

The MPI Library now supports two new multi-point motion types: PTF and PVTF.

```
typedef enum {
    MPIMotionTypePT,
    MPIMotionTypePTF,
    MPIMotionTypePVT,
    MPIMotionTypePVTF,
    MPIMotionTypeSPLINE,
    MPIMotionTypeBESSEL,
    MPIMotionTypeBSPLINE,
    MPIMotionTypeBSPLINE2,

    MPIMotionTypeS_CURVE,
    MPIMotionTypeTRAPEZOIDAL,
    MPIMotionTypeS_CURVE_JERK,
```

```

    MPIMotionTypeVELOCITY,
    MPIMotionTypeVELOCITY_JERK,
    MPIMotionTypeMASK = 0xFF,
} MPIMotionType;

```

The PTF and PVT types are identical to the current PT and PVT motion types with the addition of a feed-forward parameter that adds a Torque or Velocity value to the control (command) output.

MPIMotionTypePTF - PTF motion is identical to PT except host-calculated feedforward values can be specified for each PTF motion segment.

MPIMotionTypePVT - PVT motion is identical to PVT except host-calculated feedforward values can be specified for each PVT motion segment.

See the MPI Software section on <http://support.motioneng.com> for more details.

Version 20030605

	New Version	Previous Version
Firmware	459A6	340A2
MPI Library	20030605	20021212

2.5 MEIMotorloConfig{...} Change

MPI 1138

The arguments to `meiCanEventNotifyGet` and `meiCanEventNotifySet` were inconsistent with other MPI objects. In version 20030605, these arguments were changed to be consistent with other MPI objects. These functions now use `MPIEventMask` and a void pointer to external.

2.6 meiCanEventNotifyGet/Set Change

MPI 1137

In version 20030605, the `AbortValue` configuration was removed from `MEIMotorloConfig{...}`. The SynqNet FPGAs no longer support this feature. When the `IoAbort` is active, the node's I/O will return to the power-up state. For fail-safe operation, make sure to design external circuits that are safe when node power is lost.

2.7 MEICanNodeConfig{...} Change

MPI 1134

In version 20030528, the field names of the members of the `MEICanNodeConfig` structure were changed to match MEI's convention of having all members start with a lower case letter.

2.8 MEIFilterDataType{...} Change

MPI 1131

In version 20030522, the `MEIFilterDataType` enumeration was changed to `MEIDataType`. The reason for the change was so the enumeration could be reused by the MPI for other data type declarations.

2.9 Library and Utility Distribution

MPI 1125

In version 20021212.1.7.1, the MPI, SqNodeLib, and AppUtil project build options were updated to include the following preprocessor defines: MEI_ASSERT, MEI_TRACE, and MEI_VALIDATE. This change will enable the RELEASE libraries to have the same settings, features, and code as the DEBUG libraries. The InstallShield distribution will only install the RELEASE DLL and RELEASE utilities. These changes were made to improve the software quality and to eliminate the need for Microsoft's debug libraries during application development. All applications should use the RELEASE libraries. The DEBUG libraries are for MEI use only.

2.10 SynqNet States

MPI 1124

In version 20030429, the MEISynqNetStatus{...} structure was changed to abstract the software SynqNet states from the controller's states.

OLD:

```
typedef struct MEISynqNetStatus {
    MEIXmpSynqNetState          state;
    MEISynqNetStatusCrcError    crcError;
    MPIEventMask                eventMask;
    MEISynqNetFailedNodeMask    failedNodeMask;
} MEISynqNetStatus;
```

NEW:

```
typedef struct MEISynqNetStatus {
    MEISynqNetState             state;
    MEISynqNetStatusCrcError    crcError;
    MPIEventMask                eventMask;
    MEISynqNetFailedNodeMask    failedNodeMask;
} MEISynqNetStatus;
```

The MEISynqNetState network states are:

MEISynqNetStateDISCOVERY - Network boot, reset, and topology discovery.

MEISynqNetStateASYNQ - Data packets are sent/received asynchronously.

MEISynqNetStateSYNQ - Data packets are sent/received synchronously.

MEISynqNetStateSYNQ_RECOVERING - Network fault condition detected. Data packets are redirected around the fault.

2.11 SqNode Drive Parameter Support

MPI 1123

The SqNode object has been extended to include functions that can be used to directly access the drive parameters in SynqNet nodes. The InstallShield has installed a drive map file "drives.dm" under the C:\mei\lmp\bin\Winnt directory that contains all of the information about the drive parameters that each of the various drives support. There are also three new utilities that can be used for accessing drive parame-

ter configuration: sqDriveConfig.exe, sqDriveMonitor.exe, and sqDriveParam.exe.

SqDriveConfig.exe can be used to upload and download drive parameters. Some examples are listed below:

```
sqDriveConfig -get config.dc
sqDriveConfig -set config.dc -node 0
sqDriveConfig -set config.dc -node 0 -drive 1
sqDriveConfig -get config.dc -node 0 -map newDrives.dm
```

For more information about how to use these utilities, please see the Utilities section on <http://support.motioneng.com>.

2.12 SynqNet Error Counters

MPI 1122

In version 20030429, the MEISynqNetStatus{...} structure was expanded to support CRC error counters for each port. The MEISqNodeStatus{...} structure was expanded to support upstream and downstream packet error counters:

OLD:

from synqnet.h:

```
typedef struct MEISynqNetStatus {
    MEIXmpSynqNetState state;
    MPIEventMask      eventMask;
    long              errorCount;
} MEISynqNetStatus;
```

from sqnode.h:

```
typedef struct MEISqNodeStatus {
    MEISqNodeStatusPacketError packetError;
    MEISqNodeStatusCrcError    crcError;
    MPIEventMask               eventMask;
} MEISqNodeStatus;
```

NEW:

from synqnet.h:

```
typedef struct MEISynqNetStatusCrcError {
    long port [MEINetworkPortLAST];
} MEISynqNetStatusCrcError;

typedef struct MEISynqNetStatus {
    MEISynqNetState           state;
    MEISynqNetStatusCrcError  crcError;
    MPIEventMask              eventMask;
    MEISynqNetFailedNodeMask failedNodeMask;
} MEISynqNetStatus;
```

from sqnode.h:

```
typedef struct MEISqNodeStatus {
    MEISqNodeStatusPacketError    upStreamError;
    MEISqNodeStatusPacketError    downStreamError;
    MEISqNodeStatusCrcError        crcError;
    MPIEventMask                   eventMask;
} MEISqNodeStatus;
```

2.13 New Motor I/O Brake Source

MPI 1121

In version 20030429, a new motor I/O source RMBMotorIoTypeBRAKE was added for RMB-10v and RMB-10v2 nodes. When a motor I/O is configured for BRAKE, the MEIMotorDedicatedOutBRAKE_RELEASE bit is also applied to the selected motor I/O. This is useful for nodes that do NOT have hardware support for the dedicated brake.

2.14 Network Topology Type

MPI 1115

In version 20030410, the network type was added to the MEISynqNetInfo{...} structure:

```
typedef struct MEISynqNetInfo {
    MEINetworkType    networkType;
    long               nodeCount;
    long               nodeOffset;
} MEISynqNetInfo;
```

The MEINetworkType enumeration can be found in meidef.h and is currently defined as:

```
typedef enum MEINetworkType {
    MEINetworkTypeINVALID = -1, /* no nodes found */

    MEINetworkTypeSTRING,
    MEINetworkTypeSTRING_TERMINATED,
    MEINetworkTypeRING,

    MEINetworkTypeLAST,
    MEINetworkTypeFIRST = MEINetworkTypeINVALID + 1,
} MEINetworkType;
```

The networkType identifies the physical topology configuration that was discovered during network initialization. Your application should verify that the expected networkType was found at initialization time. The definitions and diagrams for each of the network types can be found on the support site (<http://support.motioneng.com>).

2.15 Cable Integrity after Fault Recovery

MPI 1114

In version 20030507, `meiSynqNetIdleCableListGet(...)` and `MEISynqNetCableList{...}` was added to determine the idle cable(s) in a ring topology:

```
typedef struct MEISynqNetCableList {
    long    count;
    long    cableNumber[MEISynqNetCableHOP_COUNT];
} MEISynqNetCableList;

meiSynqNetIdleCableListGet (MEISynqNet          synqNet,
                           MEISynqNetCableList *idleCable);
```

Also, `meiSynqNetIdleCableStatus(...)` and `MEISynqNetCableStatus{...}` was added to determine the integrity of the idle cable:

```
typedef enum MEISynqNetCableStatus {
    MEISynqNetCableStatusINVALID = -1,

    MEISynqNetCableStatusGOOD,
    MEISynqNetCableStatusBAD_UPSTREAM,
    MEISynqNetCableStatusBAD_DOWNSTREAM,
    MEISynqNetCableStatusBAD,

    MEISynqNetCableStatusLAST,
    MEISynqNetCableStatusFIRST = MEISynqNetCableStatusINVALID + 1,
} MEISynqNetCableStatus;

meiSynqNetIdleCableStatus (MEISynqNet          synqNet,
                           long                cableNumber,
                           MEISynqNetCableStatus *cableStatus);
```

Typically, these methods would be used to locate, replace and verify a cable on a faulted network. For example, suppose a network with a ring topology faulted due to a failed cable between two nodes. The network would automatically go into fault recovery mode, redirecting the network traffic around the failed cable. Using `meiSynqNetIdleCableListGet(...)`, the location of the idle cable could be found. Then, the bad cable could be replaced. To verify that the new cable connection is good, `meiSynqNetIdleCableStatus(...)` sends a test packet across the idle cable in both the upstream and downstream direction. If the test packet is successful, the `linkStatus` will return `MEISynqNetLinkGOOD`.

2.16 Drive Count added to MEISqNodeInfo{...}

MPI 1113

In version 20030410, `driveCount` was added to the `MEISqNodeInfo{...}` structure. The `driveCount` is the number of physical drives on a SynqNet node. To be included in the `driveCount`, the drive must have a digital communication interface with the node's FPGA. For example, an RMB-10v `driveCount` is 0, but the `motorCount` is 4. The `driveCount` is zero because the drives are external to the RMB-10v and are connected via an analog +/- 10 volt signal.

2.17 SynqNet Network Timing Optimized

MPI 1109

In version 20020410, the SynqNet network timing value calculations have been optimized for more efficient usage of network bandwidth. This optimization will increase the maximum packet data and in some cases, may reduce the latency between the controller and drive. Latency improvements are dependent on the controller sample rate, the drive processor's update rate, number of nodes, node types, and packet data sizes.

2.18 Switch ID added to MEISqNodeInfo{...}

MPI 1108

In version 20030414, switchId was added to the MEISqNodeInfo{...} structure. The meiSqNodeInfo(...) method will now return SynqNet drive switch values in the switchId structure member. The value of switchId will be 0xFF for nodes that do not support drive switch. The switchId value is reported for customer application use only. It is NOT used by the MPI for network node addressing.

2.19 Extended Return Value Message Text

MPI 1107

In version 20030331, the message text for specific return values was extended. The improved message reporting was added to mpiMessage(...). If available, mpiMessage(...) will append the extended message information for the last return value to the normal message text. For example, if a service command timeout return value was caused by node 0, the following text information would be displayed:

“SynqNet: service cmd response timeout :: Node 0.

The “::” designates appended extended message information.”

2.20 SSI Encoder Interface

MPI 1104

In version 20030319, support for an SSI encoder interface was added to the MPI for RMB10V2s. The SSI (Synchronous Serial Interface) is supported by the C0FE002C FPGA, and is used as an alternative to the Quadrature feedback offered by the RMB. The SSI is an absolute encoder, provided by a 3rd party vendor. The configuration information can be set through methods in mei_rmb.h.

```
typedef struct {
    RMBssiEncoderInputinput;
    RMBssiParity      parity;
    long              bitCount;
    long              brokenWireEnable;/* True or False */
    long              errorBitEnable;/*True or False */
    long              baudRate;/* units = hertz*/
}RMBssiEncoderConfig;

long rmbSsiEncoderConfigGet (MPIControlcontrol,
                             long              nodeNumber,
                             long              encoderIndex,
                             RMBssiEncoderConfig *config);
```

```

long rmbSsiEncoderConfigSet (MPIControlcontrol,
                             long                nodeNumber,
                             long                encoderIndex,
                             RMBsSiEncoderConfig *config);

```

A sample application that demonstrates how to configure a SSI encoder is provided at `mei\xmp\app\ssiEncConfig.c`.

2.21 Capture Changes

MPI 1103

In version 20030404, `mpiCaptureCreate(...)` and `MPICaptureConfig{...}` were changed to add support for time-based capture. This caused some internal changes which affect the capture operation and interface. The controller firmware processes the capture data before the MPI can read the captured position values.

The `mpiCaptureCreate` method specifies the capture number used by the controller:

```

mpiCaptureCreate (MPIControl    control,
                 long           number);

```

When using captures, the controller must have enough enabled captures to process the specified capture number. The `captureCount` is configured with `mpiControlConfigSet(...)`. The controller will process the enabled captures (`captureCount`) every sample period. Since each capture object is configurable, use the minimum number of captures possible for best firmware performance. For example, if you want to use 2 captures for motor 0 and motor 3, set the capture count to 2 and use capture number 0 and 1.

The `MPICaptureConfig{...}` structure has been expanded to support several new features:

```

typedef struct MPICaptureConfig {
    MPICaptureTrigger    source[MPICaptureSourceCOUNT]; /* use
                                                           MPICaptureSource to index */
    MPICaptureEdge       edge;
    MPICaptureTriggerGlobal global;
    MPICaptureType       type;
    long                 captureMotorNumber;
    long                 feedbackMotorNumber; /* the same as
                                               captureMotorNumber for POSITION capture */
    MPIMotorEncoder      encoder;
    long                 captureIndex; /* 0,1,... */
} MPICaptureConfig;

```

The capture type specifies whether to use the FPGA's position latching to capture position (`MPICaptureTypePOSITION`) or to use the FPGA's time latching to capture position (`MPICaptureTypeTIME`). For the `POSITION` type, the position counters are latched in the FPGA and sent to the controller. This methodology works well for incremental, quadrature encoders. For the `TIME` type, the FPGA latches the clock and sends the clock value and the position value for the sample to the controller. The controller interpolates the position value from the previous sample's position, the present sample's position, and the clock data. This methodology works very well for cyclic feedback data that is digitally transmitted from the drive to the FPGA. Many drives have a proprietary serial encoder that decodes the encoder position and sends the

position information to the FPGA once per sample. In these cases, time-based capture is more accurate than position-based capture.

The `captureMotorNumber` specifies which motor's digital inputs (source) to use to trigger the capture. The `feedbackMotorNumber` specifies which motor's actual position should be captured. For the POSITION type, `captureMotorNumber` and `feedbackMotorNumber` must be the same. For the TIME type, the `captureMotorNumber` and `feedbackMotorNumber` can be different.

2.22 Support for the RMB-10v2

MPI 1098

In version 20030331, support for the RMB-10v2-SynqNet node was added. The normal FPGA for the RMB-10v2 is C0FE0029. For details about the RMB-10v2 or the C0FE0029 image, please see the support site: <http://support.motioneng.com>.

NOTE: the RMB-10v2 is not backwards compatible with previous software releases.

2.23 MEIMotorConfig Changes

MPI 1093

In version 20030319, the `MEIMotorConfig{...}` structure was changed to remove some unsupported configurations. The `EncoderTermination` was removed from `MEIMotorConfig{...}` and `Invert` was removed from `MEIMotorIoConfig{...}`. Please see the Software > MPI > Documentation section on <http://support.motioneng.com> for the updated structures.

2.24 Motor Feedback Method

MPI 1092

In version 20030402, a new method was added to replace `mpiMotorFeedbackGet(...)`. The new method `mpiMotorFeedback(...)` reads the feedback values for both primary and secondary encoders. Not all hardware nodes have secondary feedback devices.

The old method, `mpiMotorFeedbackGet(...)`, could only read feedback values for the primary encoder. (For backwards compatibility, the old method will still be supported with the same name.) It is recommended that the new method be used instead of `mpiMotorFeedbackGet(...)`, since it might be removed (deprecated) in future MPI releases. All deprecated methods and structures can be found in the `XMP\include\meiDeprecated.h` header file.

2.25 Dedicated Input and Motor Fault Changes

MPI 1091

In version 20030417, several new dedicated inputs were added to the `MEIMotorDedicatedIn{...}` enumeration:

MEIMotorDedicatedInCAPTURED - Drive-based captured position state. This input is valid when a drive is configured to capture position by a service command.

MEIMotorDedicatedInHALL_A

MEIMotorDedicatedInHALL_B

MEIMotorDedicatedInHALL_C - Hall sensor states for inputs A, B, and C. These inputs are generated from a drive.

MEIMotorDedicatedInAMP_ACTIVE - Amplifier state.

1 = Amplifier is closing the current loop and the motor windings are energized.

0 = Amplifier is not closing the current loop and the motor windings are not energized.

MEIMotorDedicatedInWARNING - Drive warning state.

1 = Drive warning status bit is active and warning message is available.

0 = Drive warning status bit is not active.

MEIMotorDedicatedInDRIVE_STATUS_9 - Drive specific status bit. Reserved for future use.

MEIMotorDedicatedInDRIVE_STATUS_10 - Drive specific status bit. Reserved for future use.

The old enums `MEIMotorDedicatedInDRIVE_STATUS_0`, `MEIMotorDedicatedInDRIVE_STATUS_1`, and `MEIMotorDedicatedInDRIVE_STATUS_2` were replaced with `MEIMotorDedicatedInHALL_B`, `MEIMotorDedicatedInHALL_C`, and `MEIMotorDedicatedInAMP_ACTIVE`.

In version 20030417, new motor fault bits were added to the `MEIMotorFaultBits{...}` enumeration:

MEIMotorFaultBitAMP_NOT_POWERED - The SynqNet drive amplifier stage does not have sufficient voltage. The motor windings cannot be energized properly until this bit is clear.

MEIMotorFaultBitDRIVE_NOT_READY - The SynqNet drive is not ready to receive commands or servo motors. This fault indicates the drive has not completed its processor initialization or there is some other serious drive processor problem.

2.26 New Node Alarm Configurations

MPI 1090

In version 20030319, the `notCyclicEnable` and `ioAbortEnable` configurations were added to the `MEISqNodeConfigAlarm{...}` structure.

If `notCyclicEnable = TRUE`,
the node's digital alarm output signal will be active when the node is not in cyclic (SYNQ) mode.

If `notCyclicEnable = FALSE`,
the node's digital alarm output signal will not be affected by the node's state.

If `ioAbortEnable = TRUE`,
the node's digital alarm output signal will be active when the node's `ioAbort` is active.

If `ioAbortEnable = FALSE`,
the node's digital alarm output signal will not be affected by the node's `ioAbort` state.

For more information about the node alarm and its configurations, see the node faults diagram under Technology > SynqNet on the Technical Support site (<http://support.motioneng.com>).

2.27 New Kollmorgen CD Drive Configuration

MPI 1089

In Kollmorgen CD drive firmware version 0.1.5, a new configuration for the Divide-by-N Speed feature was added to the CDDivNConfig{...} structure in kollmorgen_ch.h. This feature is supported in the 20021212.1.7 release. For more information about Kollmorgen drive configurations, please consult the Kollmorgen drive documentation.

2.28 MEIMotorEncoder{...} Change

MPI 1083

In version 20030310, the MEIMotorEncoder{...} structure was changed to add support for configurable encoder feedback. MEIMotorEncoderType{...} has replaced MEIXmpEncoderType{...} in the MEIMotorEncoder{...} structure. This enumeration is used to specify the type of feedback (ie. Quadrature, Drive interface, SSI) being used by a specific motor, and is used to configure the SynqNet Node FPGA. During SynqNet network initialization, the default encoder type will be configured by the sqNodeLib. Applications may change the encoderType if the node/drive supports the specified configuration.

```
typedef enum MEIMotorEncoderType{
    MEIMotorEncoderTypeINVALID = -1,

    MEIMotorEncoderTypeQUAD_AB,
    MEIMotorEncoderTypeDRIVE,
    MEIMotorEncoderTypeSSI,

    MEIMotorEncoderTypeLAST,
    MEIMotorEncoderTypeFIRST = MEIMotorEncoderTypeINVALID + 1,
} MEIMotorEncoderType;

typedef struct MEIMotorEncoder {
    MEIMotorEncoderType      type;
    long                     countsPerRev;
    MEIMotorEncoderRatio     ratio;
    MEIMotorEncoderReverseModulo  reverseModulo;
} MEIMotorEncoder;
```

2.29 SynqNet Timing Information

MPI 1081

In version 20030410, a new SynqNet method, meiSynqNetTiming(...) was added. This method returns SynqNet timing information to allow system designers to fine tune network performance. Please see the Technical Support website (<http://support.motioneng.com>) for details about the timing values.

2.30 New Kollmorgen CD Drive Configuration Features

MPI 1079

In Kollmorgen CD drive firmware version 0.1.2, several new configurations were added (highlighted in blue). The CDDriveConfig{...} structure in kollmorgen_cd.h was expanded to support these new features. The SqNodeLib support is available in MPI version 20021212.1.5.

```
/*
 * Drive Config Structure
```

```

*/

typedef struct CDDriveConfig {
    long mpitch;      /* Defines the pole-pitch of the motor and allow the
                       drive to be able to calculate other values. */
    long motortype; /*
                       */
    long mipeak;     /* Sets the motor's peak rated current */
    long micont;     /* Sets the motor's continuous rated current */
    long mspeed;     /* Defines the maximum recommended velocity of
                       the Motor */
    long mkt;        /* Motor torque constant */
    long mencre;     /* Displays the resolution of the motor encoder in number
                       of lines per revolution of the motor */
    long menctype;  /* Motor encoder type */
    long mencoff;   /* Sets the encoder index position */
    long mlmin;     /* Sets the motor's minimum line-to-line inductance */
    long mphase;    /* Defines the encoder phase relative to the "standard"
                       commutation table */
    long mpoles;    /* Sets the number of motor poles */
    long mbemfcomp; /* Sets a back EMF compensation percentage value */
    long mlgainc;   /* Sets the current loop adaptive gain value at
                       continuous motor current */
    long mlgainp;   /* Sets the current loop adaptive gain value at peak
                       motor current */
    long mtanglc;   /* Sets the value of the torque-related commutation
                       angle advance at the motor's continuous current
                       rating */
    long mtanglp;   /* Sets the value of the torque-related commutation
                       angle advance at the motor's peak current */
    long mvanglf;   /* Sets the value of the velocity-rated commutation
                       angle advance for when the motor is operating at motor
                       max speed */
    long mvanglh;   /* Sets the value of vel-rated commutation angle advance
                       for when the motor is operating at half of the motor
                       max speed */
    long mhinva;    /* HALL A inversion */
    long mhinvb;    /* HALL B inversion */
    long mhinvc;    /* HALL C inversion */
    long vbus;      /* Sets the drive bus voltage */
    long ilim;      /* Sets the application current limit */
    long icont;     /* Sets the system continuous current */
    long vlim;      /* Sets the application velocity limit. */
    long mfbdir;    /* Motor & feedback direction. */
    long anoff1;    /* Analog offset 1. */
    long anoff2;    /* Analog offset 2. */
    long initgain;  /* gain for encoder initalization. */
    long iencstart; /* Maximum current for wake no shake. */
    long uvmode;    /* Action in in response to under-voltage. */
    long uvtime;    /* Under voltage warning time. */
    long uvrecover; /* Recovery from under-voltage fault. */
    long thermmode; /* Action when motor thermostat opens. */
    long thermtpe;  /* Motor temperature sensor type. */
}

```



```

        long izero;        /* Sets the ZERO mode current. */
    } CDDriveConfig;

```

2.31 EncoderFault support for SynqNet Encoders

MPI 1076

In previous versions, the encoder fault limit supported local quadrature encoder fault conditions for a motor's encoder. The possible conditions were broken wire, illegal state, and ABS errors (for absolute encoders), which were specified by a fault mask in the MPIMotorEventTrigger structure. In SynqNet, the motor object can support one or two encoders (primary and secondary). Each quadrature encoder has a single fault bit. The MPIMotorEventTrigger structure was changed to support the new encoder fault configuration.

OLD:

```

typedef enum {
    MPIMotorEncoderFaultINVALID = -1,

    MPIMotorEncoderFaultBW_DET,
    MPIMotorEncoderFaultILL_DET,
    MPIMotorEncoderFaultABS_ERR,

    MPIMotorEncoderFaultLAST,
    MPIMotorEncoderFaultFIRST = MPIMotorEncoderFaultINVALID + 1
} MPIMotorEncoderFault;

typedef union {
    long    polarity;    /* 0 => active low, else active high */
    long    position;    /* MPIEventTypeLIMIT_SW_[POS|NEG] */
    float   error;       /* MPIEventTypeLIMIT_ERROR */
    long    mask;        /* MPIEventTypeENCODER_FAULT */
} MPIMotorEventTrigger;

```

NEW:

```

typedef enum {
    MPIMotorEncoderFaultINVALID = -1,

    MPIMotorEncoderFaultPRIMARY,
    MPIMotorEncoderFaultSECONDARY,
    MPIMotorEncoderFaultPRIMARY_OR_SECONDARY,

    MPIMotorEncoderFaultLAST,
    MPIMotorEncoderFaultFIRST = MPIMotorEncoderFaultINVALID + 1
} MPIMotorEncoderFault;

typedef union {
    long    polarity;    /* 0 => active low, else active high */
    long    position;    /* MPIEventTypeLIMIT_SW_[POS|NEG] */
    float   error;       /* MPIEventTypeLIMIT_ERROR */
    long    encoder;     /* MPIEventTypeENCODER_FAULT */
} MPIMotorEventTrigger;

```

```
    } MPIMotorEventTrigger;
```

Use MPIMotorEventTrigger.encoder to set the encoder fault limit to trigger from the primary, secondary, or either of the encoder's faults. To determine the cause of the encoder failure (broken wire, illegal state, etc), use meiMotorStatus(...).

Note: SynqNet drives that support position feedback via a proprietary serial interface (not quadrature encoder) do not support the encoder fault bit. They use the dedicated amp fault and amp fault message to notify the application when a position feedback device failure occurs. These changes were implemented in versions 20021212.1.4 and 20030120.

2.32 MEIMotorInfo Structure Change

MPI 1075

In version 20021212.1.4 and 20021219, the MEIMotorInfo structure was changed. The captureCount and encoderCount elements were moved outside of the sqNode struct since they really belong to the motor.

OLD:

```
typedef struct MEIMotorInfo {
    MEIMotorInfoNodeType    nodeType;
    struct {
        long networkNumber;
        long nodeNumber;
        long driveIndex;
        long captureCount;
        long encoderCount;
    } sqNode;
} MEIMotorInfo;
```

NEW:

```
typedef struct MEIMotorInfo {
    MEIMotorInfoNodeType    nodeType;
    struct {
        long networkNumber;
        long nodeNumber;
        long driveIndex;
    } sqNode;
    long captureCount;
    long encoderCount;
} MEIMotorInfo;
```

2.33 meiSqNodeDriveInfo(...) Change

MPI 1072

In version 20030410, the meiSqNodeDriveInfo(...) method was changed to add support for a new MEISqNodeDriveInfo{...} structure. This expansion makes it possible to support common drive information and drive specific information.

NOTE: The drive specific info passed through the "void *info" argument in the old method is the same as

the "void *external" argument in the new method.

OLD:

```
meiSqNodeDriveInfo (MEISqNode      node,
                    long            driveIndex, /* relative to the node */
                    void            *info);     /* node specific */
```

NEW:

```
typedef struct MEISqNodeDriveInfo {
    char        firmwareVersion[MEISqNodeID_CHAR_MAX];
} MEISqNodeDriveInfo;

meiSqNodeDriveInfo (MEISqNode      node,
                    long            driveIndex, /* relative to the node */
                    MEISqNodeDriveInfo *info,
                    void            *external); /* node specific */
```

2.34 User Configurable Packets

MPI 1065

In version 20030331, meiSynqNetPacketConfigGet/Set(...) methods have been added to allow user configuration of network packets. Most applications do NOT need to configure the network packets. The MPI automatically configures the packets to match the resources supported by each node type. In some special cases, an application may need to disable some features in the packets to optimize network bandwidth. The user configurable packets features are intended for advanced SynqNet system designers.

WARNING!

If you are using configurable packets, make sure to configure the packets BEFORE making other object configurations (control, motion, axis, filter, and motor objects). The packets define the interface between the controller's objects and the network nodes. Adding or removing packet resources will require the MPI to clear some object configurations, re-initialize pointers, and re-initialize the network using the default object configurations. Any previous object configurations will be ignored.

2.35 Motor Brake Name Change

MPI 1055

In previous versions, the brake logic was supported via a User I/O bit. In SynqNet Phase II, the brake logic is supported via a dedicated brake release output bit and a dedicated brake applied input bit. The MEIMotorDedicatedIn{...} enumeration has a MEIMotorDedicatedInBRAKE_APPLIED bit and the MEIMotorDedicatedOut{...} enumeration has a MEIMotorDedicatedOutBRAKE_RELEASE bit. Also, the MPIMotorBrake{...} structure was updated to use the new "apply" and "release" terminology:

```
typedef struct MPIMotorBrake {
    MPIMotorBrakeMode    mode;
    float                applyDelay;
    float                releaseDelay;
} MPIMotorBrake;
```

The functionality of the brake feature has not changed from previous releases. The brake structure name

changes were made in version 20030207.

2.36 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.

2.37 SynqNet Network Cable Length Compensation

MPI 1053

Automatic Cable Length Compensation has been added in this release. Upon network initialization, the controller performs a quick measurement of the cable lengths. These cable length measurements are used by the network timing equations for improved network bandwidth optimization. The measured cable lengths can be viewed by calling the meiSynqNetInfo(...) method. Three cable length values are used by the network timing equations: Nominal (or measured), Minimum, and Maximum lengths. To allow more system control and timing consistency, the MPI allows a Get/Set of these three values which are the actual values in the SynqNet timing equations. By using meiSynqNetConfigGet/Set(...), these values can be set to dynamic memory or saved to flash memory.

WARNING: Changing a cable length value will cause the network to be reset and timing values to be recalculated.

ReturnValues:

MEISynqNetMessageCABLE_LENGTH_INVAILD_NOMINAL - The *nominal* cable length value that you are attempting to set is out of range. Any numbers greater than 65534 (0xFFFF) meters will cause this value to be returned. However, the physical specification for cable lengths is much less than this (approximately 100 meters).

MEISynqNetMessageCABLE_LENGTH_INVAILD_MIN - The *minimum* cable length value that you are attempting to set is out of range. This value must be less than the nominal cable length value. Any numbers greater than 65534 (0xFFFF) meters will cause this value to be returned.

MEISynqNetMessageCABLE_LENGTH_INVAILD_MAX - The *maximum* cable length value that you are attempting to set is out of range. This value must be greater than the nominal cable length value. Any numbers greater than 65534 (0xFFFF) meters will cause this value to be returned.

If cable lengths are saved to flash, the controller will verify that the measured cable length is within range of the Minimum and Maximum cable length values. This happens as part of the network topology verification. If the measured value is NOT within range, mpiControllnit(...) and/or meiSynqNetInit(...) will return the following error message:

MEISynqNetMessageCABLE_LENGTH_MISMATCH - the discovered cable length does not fall in

between the min and max cable ranges that were saved to flash.

2.38 Network Fault Recovery and Node Failure Tolerance MPI 1051

In version 20030501, SynqNet networks support network fault recovery with a ring topology and tolerate node failures. To support these features, you must have SqNode FPGA runtime images, version 0x0207 or later installed on your nodes.

Network fault recovery is only supported with ring topologies. If any single network cable fails, the network traffic will be automatically re-routed around the faulty connection. When the packet error rate counter exceeds the packet error fault limit, the connection is faulted. The node hardware and/or controller will detect the location of the fault and switch the direction of network traffic for the nodes downstream from the faulted cable. The controller will generate a `MEIEventTypeSYNQNET_RECOVERY` status/event. If a network fault recovery occurs, an application should notify the user about the fault and fix the broken cable as soon as possible. An application can determine the location of the fault using `meiSynqNetIdleCableList-Get(...)`. A ring topology has one idle cable, which has no data traffic.

The network fault recovery mode can be configured with `meiSynqNetConfigGet/Set(...)`, using the `MEISynqNetRecoveryMode` enumeration:

MEISynqNetRecoveryModeDISABLED (default for string topology) - Network does not attempt to redirect network traffic around a fault.

MEISynqNetRecoveryModeSINGLE_SHOT - Network will redirect network traffic around a fault one time. A second fault will cause all nodes downstream from the fault to fail.

MEISynqNetRecoveryModeAUTO_ARM (default for ring topology) - Network will redirect network traffic around a fault each time a fault occurs. After the network traffic redirection, the controller will wait for the node(s) upstream and downstream packet error rate counters to decrement to zero until the recovery is re-armed. Then, the network will be able to respond to another fault.

Node failure tolerance is supported with string or ring topologies. If one or more nodes fail, the network will continue to operate in SYNQ mode, sending and receiving data from the good nodes. The controller will generate a `MEIEventTypeSYNQNET_NODE_FAILURE` status/event for the network when any node fails and a `MEIEventTypeSQNODE_NODE_FAILURE` status/event for each node that fails. If a node failure occurs, an application should safely shutdown the motors on the operational nodes and notify the user of the failed nodes, so the hardware can be repaired. An application can determine which nodes failed by reading the `failedNodeMask` with `meiSynqNetStatus(...)` or the `MEIEventTypeSQNODE_NODE_FAILURE` status mask with `meiSqNodeStatus(...)`.

The motor configuration also has a node failure action. If any node fails, the controller can be configured to generate an action (none, stop, e-stop, abort, etc.) for the motors on the operational nodes. The `nodeFailureAction` configuration is in the `MEIMotorConfig{...}` structure.

2.39 mpiMotorConfigSet(...) without available hardware MPI 1047

In versions 20030106 and earlier, `mpiMotorConfigSet(...)` allowed configuration of a motor object that had no physical hardware on the network. This problem allowed for a potential mismatch of hardware and host

application configurations. A check for node hardware has been added to version 20030107 and later to avoid this problem. If an application attempts to configure a motor object that does not have physical node hardware associated with it, the `mpiMotorConfigSet(...)` will return the following error code:

```
MEIMotorMessageHARDWARE_NOT_FOUND ("Motor: hardware not found")
```

In version 20030423, the hardware check was optimized. The MPI determines if the configuration requires service commands to be sent to node hardware. The following configurations require node hardware:

```
MPIMotorConfig.encoderPhase  
MPIMotorConfig.secondaryEncoderPhase  
MEIMotorConfig.faultConfig.faultMask  
MEIMotorConfig.Encoder[].type  
MEIMotorConfig.Io[].Type  
MEIMotorConfig.Io[].AbortValue
```

If node hardware is not available and any of these configurations are modified, `mpiMotorConfigSet(...)` will return `MEIMotorMessageHARDWARE_NOT_FOUND`.

2.40 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."

2.41 Motion Method Check for Axis Object

MPI 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.

2.42 Bad Packet Feedback Compensation

MPI 1026

In previous versions, if a bad feedback packet occurred the controller would use the feedback position value from the previous sample. This would cause the servo control loop to experience zero actual velocity input, usually causing a large change in the torque command output for the sample. In versions 20030410 and later, the controller firmware compensates for bad or missing feedback packets by interpolating an actual position between the last good position and the last good actual velocity. This will prevent large torque command output changes in the case of a missing or corrupted feedback packet. If your system is experiencing several bad packets, then check your wiring, power, controller, and node hardware integrity. The bad packet feedback compensation is intended to minimize the effects of a bad packet. It does NOT solve network integrity issues.

2.43 sqNodeMemory Utility

MPI 1015

The sqNodeMemory utility allows a user to read or write to a SynqNet node's memory. This utility is similar to VM3. Command line arguments can be used to specify the controller number, client/server operation, and the node number. For more information about the sqNodeMemory utility, please see the Utilities section on the Technical Support website (<http://support.motioneng.com>).

2.44 Data Recorder Overflow

MPI 901

In previous versions, the MPI attempted to detect and recover from a data recorder buffer overflow. The MPI kept track of records read from the buffer (RecsOut). And the firmware kept track of the total number of records added to the buffer (RecsIn). If $\text{RecsIn} - \text{RecsOut} > \text{BufferSize}$, then the buffer overflowed. The MPI would flush the buffer completely and re-start data collection. This can be a problem when the overflow occurs after the RecsIn has been read, causing the overflow to go undetected and corrupted data to be collected.

This problem was corrected by redesigning the MPI and firmware data collection. Now, when the buffer is full, the controller will set a "Full" bit and stop collecting data. The MPI reads all of the available data and then clears the "Full" bit. After the "Full" bit is cleared, the firmware continues collecting data into the buffer. This design guarantees that data will not be corrupted. Also, it simplifies the MPI and firmware code.

During the redesign, a firmware bug was found and corrected. The bug occurred when the Recln pointer value wraps around. In the old code, the Recln pointer was updated incorrectly and could cause the firmware to write past its allocated buffer. This would cause the MPI to incorrectly calculate the number of records to be read, thereby causing older data to be re-read. This problem was corrected in the MPI and the firmware.

The "Flash" methods were removed from the MPI since they do not make sense for the recorder.

The functionality of the "fullCount" in the MPIRecorderConfig{...} structure and in mpiRecorderStart(...) has

changed. In previous versions, setting fullCount to 0 would automatically disable the status bit that generates MPIEventRECORDER_FULL events. This is no longer the case. The RECORDER_FULL status bit will be set whenever the "Full" handshake bit is set by the firmware. RECORDER_FULL events can be enabled or disabled by mpiEventConfigSet(...). If the "fullCount" is set to 0, the MPI will automatically adjust the firmware's value so that it reflects the largest buffer size that will exactly fit an integer number of records within the allocated buffer (fullCount = allocatedBufferSize - (allocatedBufferSize % recordSize)).

All of these changes were made in release version 20030120 and later.

2.45 MPIState Changes

MPI 828

In version 20030605, the MPIState enumeration was changed. MPIStateINIT was removed and MPIStateSTOPPED was added. The INIT state was never supported. The STOPPED state occurs after the motion is done due to a STOP action. From the STOPPED state, a motion profile can be resumed using the RESUME action. While in a STOPPED state, the application cannot set the axis's command position.

2.46 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 20020117.1.7, 20030120, and later releases. If the DRate value is out of range, an INVALID_DRATE error message will be returned.

2.47 Gain Table Switching

463

In firmware version 431A3, standard support for gain table switching was added. The controller's memory has 5 gain tables, each containing a set of filter parameters. The controller's filter can be configured to switch filter parameters from the gain tables. The criteria for gain switching is based on the firmware's internal switching algorithm and the configurable parameters in MEIFilterConfig{...}. The standard switching criteria is based on the controller's trajectory calculator, in which the states correspond to the following gain table indexes:

MEIFilterGainIndexNO_MOTION - No commanded motion. Trajectory parameters Velocity, Acceleration, and Jerk equal zero.

MEIFilterGainIndexACCEL - Acceleration portion of the commanded move.

MEIFilterGainIndexDECEL - Deceleration portion of the commanded move.

MEIFilterGainIndexVELOCITY - Constant velocity portion of the commanded move.

Gain switching is configured by setting the GainSwitchType, GainDelay, and GainWindow in the MEIFilterConfig{...} structure and calling mpiFilterConfigGet/Set(...). The GainSwitchType has the following options:

MEIXmpSwitchTypeNONE - Gain switching is disabled.

MEIXmpSwitchTypeMOTION_ONLY - Switch gains based on controller's switching algorithm.

MEIXmpSwitchTypeWINDOW - Switch gains based on position window. Requires custom firmware.

MEIXmpSwitchTypeUSER - Switch gains based on user criteria. Requires custom firmware.

The **GainDelay** is only available with custom firmware. It specifies the time (seconds) between the controller's calculated gain switch and the applied gain switch.

The **GainWindow** is only available with custom firmware. It specifies the position window (counts) to apply

the gain switch.

Version 20021212

	New Version	Previous Version
Firmware	340A2	422A3
MPI Library	20021212	20021101

2.48 Yaskawa Sigma-III DP_RESET Compatibility

MPI 1036

The Yaskawa SynqNet Sigma-III drive contains a DP_RESET signal that can be wired to a pull-up or pull-down resistor. In previous versions, the SynqNet initialization required a 3 sec delay to wait for a pull-down configured drive to complete its reset. In versions 20021101.1 and 20021212 (and later), the 3 sec delay has been removed. This release is ONLY compatible with Sigma-III drives that have a DP_RESET pull-up resistor. Yaskawa is upgrading all drives to the pull-up configuration. If you have an old drive with the pull-down resistor, it is not compatible with this release. Please contact Yaskawa for an upgrade.

2.49 Encoder Fault for Secondary Encoders

MPI 1019

In version 20021212, the MPI and Firmware were extended to support encoder faults (broken wire and illegal state) for secondary encoders. Each motor object has two encoder inputs (primary and secondary). Depending on the SynqNet node type, the hardware may or may not support secondary encoders. By default, during network initialization, the MPI maps each secondary encoder to each motor sequentially.

For example, a node with 4 primary encoders and 1 secondary encoder will have the secondary encoder mapped to motor 0. The encoder fault inputs for the primary encoder and the secondary encoder are mapped to the same motor object.

To configure the encoder fault conditions and action, use `mpiMotorEventConfigSet(...)`.

To clear encoder faults, use `meiMotorEncoderReset(...)`.

To determine encoder fault status, use `meiMotorStatus(...)`.

2.50 SqNodeMemory Utility

MPI 1015

The SqNodeMemory utility allows a user to read or write to a SynqNet node's memory. This utility is similar to VM3. Command line arguments can be used to specify the controller number and client/server operation.

Usage:

Ctrl + Pg Up/Down - cycles through the different nodes.

Pg Up/Down - cycles through the different drives on the node if drive memory is available.

2.51 mpiCaptureCreate(...) Check

MPI 995

In version 20021212, a capture availability check was added to mpiCaptureCreate(...). If a given node does not support capture, then mpiCaptureValidate(...) will return MPIMessageUNSUPPORTED.

2.52 Motor Capture Information

MPI 994

In version 20021212, a captureCount and encoderCount were added to the MEIMotorInfo{...} structure to be retrieved by meiMotorInfo(...) method.

2.53 Change to Post Filter Methods

MPI 944

Calls to the following methods:

```
meiFilterPostfilterGet
meiFilterPostfilterSet
meiFilterPostfilterSectionSet
meiFilterPostfilterSectionGet
```

now require that you specify the form of the digital filter, as well as its type in the MEIPostfilterSection structure before passing this structure to the methods.

Valid values for the MeiFilterType are:

```
typedef enum {
    MEIFilterTypeINVALID = -1,

    MEIFilterTypeUNITY_GAIN,
    MEIFilterTypeSINGLE_ORDER,
    MEIFilterTypeLOW_PASS,
    MEIFilterTypeHIGH_PASS,
    MEIFilterTypeNOTCH,
    MEIFilterTypeRESONATOR,
    MEIFilterTypeLEAD_LAG,
    MEIFilterTypeZERO_GAIN,

    MEIFilterTypeUNKNOWN,
    MEIFilterTypeLAST,
    MEIFilterTypeFIRST = MEIFilterTypeINVALID + 1,

} MEIFilterType;
```

Valid values for the MeiFilterForm are:

```
typedef enum{
    MEIFilterFormINVALID = -1,
```

```

    MEIFilterFormIIR,
    MEIFilterFormBIQUAD,
    MEIFilterFormSS_BIQUAD,
    MEIFilterFormINT_BIQUAD,
    MEIFilterFormINT_SS_BIQUAD,

    MEIFilterFormLAST,
    MEIFilterFormFIRST = MEIFilterTypeINVALID + 1,

} MEIfilterForm;

```

In addition to the canonical floating point biquad form there are three additional new biquad types: A 32 bit fixed point biquad, and implementations of the fixed and floating point biquads in state space form.

2.54 SynqNet Initialization from controller firmware

MPI 940

In version 447A2, the controller firmware now supports SynqNet network initialization after an mpiControl-Reset(...) if the topology has been saved to flash. Please see the meiSynqNetTopologySave(...) and meiSynqNetTopologyClear(...) documentation for more details.

2.55 MEIEventTypeSETTLED

MPI 918

In version 20021212, an enumerated value MEIEventTypeSETTLED was added to match the status bit MEIXmpStatusSETTLED. MEIEventTypeSETTLED is equivalent to MEIEventTypeIN_POSITION_FINE.

2.56 Motion Type Error Message

MPI 857

In version 20021212, an error code was added for unsupported motion types. If the firmware does not support a motion type, the MPI will return MPIMotionMessagePROFILE_NOT_SUPPORTED. Presently, the firmware does not support S_CURVE_JERK and VELOCITY_JERK motion types due to space limitations.

2.57 Filter Data Types

MPI 839

In version 20021212, the static arrays MEIFilterGainTypePIV and MEIFilterGainTypePID were added in order to be used in conjunction with the MEIFilterGainPIVCoeff and MEIFilterGainPIDCoeff enumerations. Based on filter algorithms, an application can index the proper coefficients using the MEIFilterGainPIVCoeff and MEIFilterGainPIDCoeff enumerations. The application can determine the data type of the value stored by indexing these new static arrays.

2.58 Map.h Renamed

MPI 790

Map.h is now a template in the Standard Template Library. To avoid compilation complications, MEI has renamed the map module (map.h) in the MPI to meimap.h. Customers that include the MPI map module (map.h) in their application will need to modify their include statement to include meimap.h instead.

Version 20021101

	New Version	Previous Version
Firmware	422A3	420A6
MPI Library	20021101	20021030

- There were no general changes or new features in the 20021101 release.

Version 20021030

	New Version	Previous Version
Firmware	420A6	420A2
MPI Library	20021030	20021025

- There were no general changes or new features in the 20021030 release.

Version 20021025

	New Version	Previous Version
Firmware	420A2	415A6
MPI Library	20021025	20021021

2.59 SynqNet Network Initialization Improved

MPI 1002

In version 20021024, the SynqNet network initialization sequence was improved. In previous versions, the nodes were reset sequentially. For efficiency, the nodes are now reset simultaneously and then the software waits for all the nodes to complete their resets.

2.60 Config Utility Updated

MPI 993

The 20021021 release did not support the config.exe utility. The config.exe utility has now been ported to run with the new SynqNet Phase II MPI interface.

2.61 Tx Failure Status/Event

MPI 992

In version 20021024, a new SynqNet TX_FAILURE status and event was added. If the controller sample rate is too high or the foreground processing time is too long, the DSP may not be able to update the Tx buffer before the data is sent to the nodes. If the DSP updates the Tx data too late (in two successive samples), the DSP will shutdown the network and generate an TX_FAILURE status and event. If TX_FAILURE events occur, you will need to reduce the sample rate and/or reduce the DSP's foreground cycle process-

ing load or increase the Tx Time (percentage). Be sure to eliminate any unused resources to reduce the foreground cycle load. Motion Console has a "Stats" tab in the Controller summary screen to measure the foreground cycle time. Also, a good rule of thumb is to keep the foreground cycle time less than (sample time) * (Tx Time). For example, if the sample time is 1/2000 (default) and the Tx Time is 75% (default), then the foreground cycle time should be less than ~375 microseconds (plus some margin).

2.62 Scale Interpolation Module Support Removed

MPI 986

In version 20021017, support for SIM (Scale Interpolation Module) was removed. The SIM is only available with XMP-Series Analog controllers, it is NOT compatible with SynqNet controllers or software.

Version 20021021

	New Version	Previous Version
Firmware	415A6	415A5
MPI Library	20021021	20021011

- There were no general changes or new features in the 20021021 release.

Version 20021011

	New Version	Previous Version
Firmware	415A5	363A6
MPI Library	20021011	20020403.1.3

2.63 EStop with Command = Actual

MPI 970

In version 20021009, a new EStop Action was added, MPIActionE_STOP_CMD_EQ_ACT. When the E_STOP_CMD_EQ_ACT is triggered, the output of the axis trajectory calculator is replaced by setting the command position equal to last sample's actual position. After the E-Stop time expires, the AmpEnable is disabled. The new action can be commanded by the user via the MPI or by configuring a motor limit to set the new status bit. Settling mode is also supported for this new action. It can be configured with settleOnEstopCmdEqAct (similar to settleOnStop/settleOnEStop).

2.64 Abort Action with Pending Moves

MPI 967

In firmware version 397A1, if a move was pre-loaded using the HOLD attribute and an ABORT action was commanded, the Motion Supervisor state would get stuck in a STOPPING condition. This was caused by an internal state machine problem in the firmware. It was fixed in version 397A2, 398A5 and later versions.

2.65 Auxiliary Encoders

MPI 965

SynqNet nodes that have auxiliary encoders will have those encoders mapped as the secondary encoder

on the first motor on the node (Motor 0). The second auxiliary encoder will be mapped to Motor 1, so on and so forth.

Auxillary encoders are currently fixed at a maximum of 32bit of resolution

2.66 Increased User Limits

MPI 956

In version 20020916, the number of user limits per motor has been increased from 8 to 16.

2.67 Changed Motion/Axis status to support pre-loaded move triggering

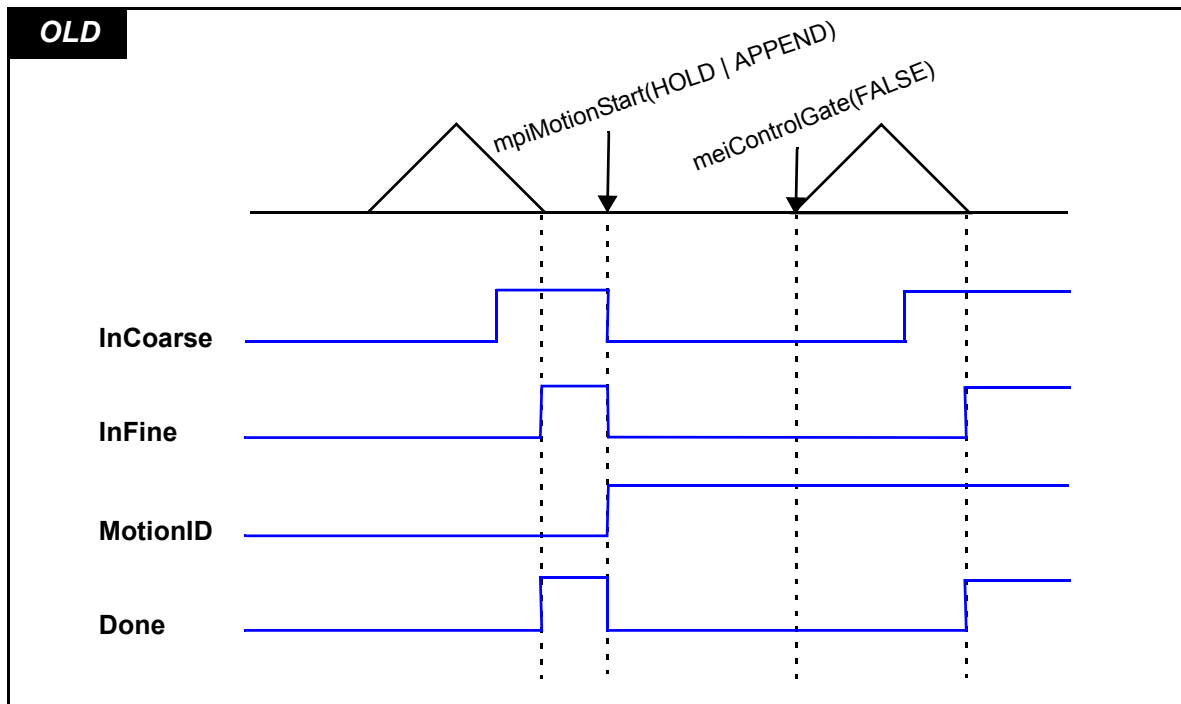
MPI 955

A new "pending" state was added to the firmware motion state machine and the frame sequence was also changed. Previously, the frame sequence was: start frame, hold frame, and then accel frame. The new sequence is: start frame, hold frame, id frame, and then accel frame. The new start frame will trigger the new "pending" state. The new id frame will then load the target position and clear the InCoarse, InFine, AtTarget status, load the moveID, and update the motion state to "moving."

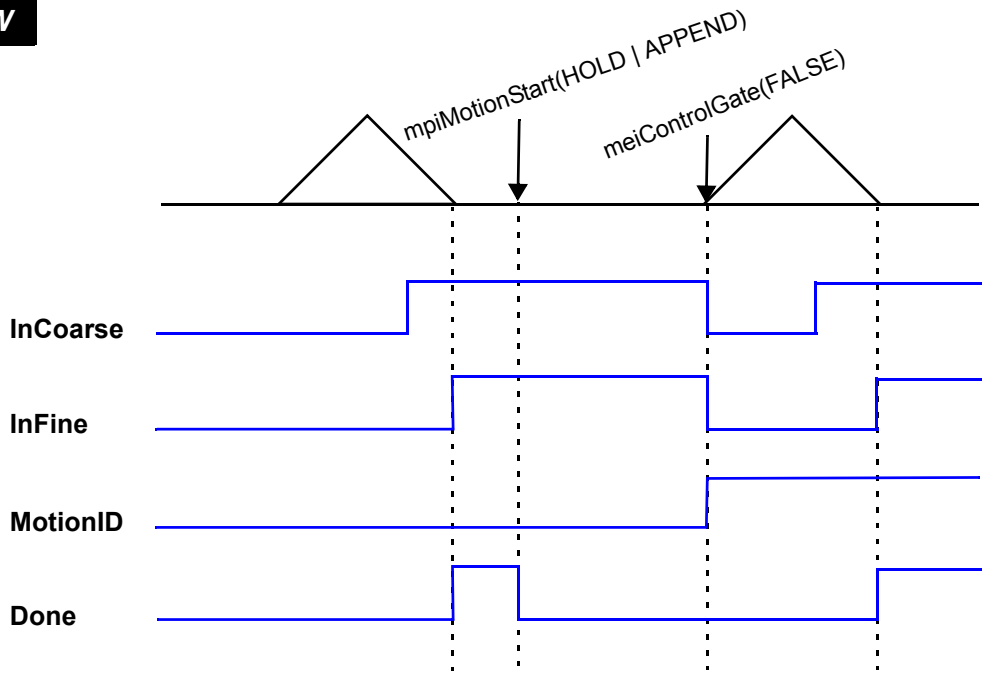
The InCoarse, InFine, and AtTarget status will not be cleared until the APPENDED motion profile has been triggered. The MotionID will not be updated until the APPENDED motion profile has been triggered.

The MotionDone status will be cleared before updating the MotionID.

Please see the diagrams below.



NEW



2.68 Addition of MEIMotorFaultStatus and MEIMotorFaultConfig to stdmei.h.

MPI 948

MEIMotorFaultStatus is a member of MEIMotorStatus and is used to report what conditions caused an AMP Fault. MEIMotorFaultConfig is a member of MEIMotorConfig and is used to configure the fault conditions that will cause an AMP Fault. Both variables are *long's* and are *bit masks*. They both use MEIMotorFaultMask to decipher and manipulate these variables.

2.69 Support for Motor I/O

MPI 947

SynqNet Phase 2 supports 16 bits of configurable motor I/O. The 16 bits of general Motor I/O that can be used for a variety of purposes depending on the hardware. Depending on the node, the number of I/O, type, and signal naming will differ.

Each FPGA I/O pin has many configuration possibilities. The limitations on the configuration options will be based on the hardware attached to the pins (per node manufacturer). Each pin has the capability of being configured as an Input or Output. If the pin is an Output, the source of this output also needs to be configured (output register, step, direction, compare, etc.). The Output can also be inverted. In the case of an I/O Abort, the state of the Output can be configured to a 0 or 1.

To configure these I/O pins the following structures have been added to the MEIMotorConfig structure in stdmei.h.

```
typedef struct MEIMotorIoConfig {
    MEIMotorIoType    Type;
    long              Invert;
    long              AbortValue;
} MEIMotorIoConfig;

typedef enum MEIMotorIoType {
    MEIMotorIoTypeINPUT = -1,
    MEIMotorIoTypeOUTPUT,
    MEIMotorIoTypeSOURCE1,
    MEIMotorIoTypeSOURCE2,
    MEIMotorIoTypeSOURCE3,

    MEIMotorIoTypeLAST,
    MEIMotorIoTypeFIRST = MEIMotorIoTypeINPUT,
} MEIMotorIoType;
```

In order for the software programmer to resolve which IO pins are associated to the hardware present, a drive specific enumeration will be used which will be located in the sqNodeLib library. The following is an example of an enumeration that would be found in mei_rmb.h. This IO is a direct reflection of what was present on our Xmp Analog boards.

```

/* RMBMotorIoConfig is used to index an array of MEIMotorConfigIo
   found in MEIMotorConfig */

typedef enum {
    RMBMotorIoConfigXCVR_A    = MEIMotorIoConfigIndex0,
    RMBMotorIoConfigXCVR_B    = MEIMotorIoConfigIndex1,
    RMBMotorIoConfigXCVR_C    = MEIMotorIoConfigIndex2,
    RMBMotorIoConfigXCVR_D    = MEIMotorIoConfigIndex3,
    RMBMotorIoConfigXCVR_E    = MEIMotorIoConfigIndex4,
    RMBMotorIoConfigXCVR_F    = MEIMotorIoConfigIndex5,
    RMBMotorIoConfigUSER_0_IN = MEIMotorIoConfigIndex6,
    RMBMotorIoConfigUSER_0_OUT = MEIMotorIoConfigIndex7,

} RMBMotorIoConfig;

```

The individual drive modules will be responsible for doing error checking on the configuration of a given bit. meiMotorConfigSet will fail and an error will be reported if an illegal bit configuration is specified.

Sample code:

```

/***** sample code *****/

MEIMotorConfig    xmpConfig;

/* old way */
xmpConfig.Transceiver[MEIMotorTransceiverIdA].Invert = 1;
xmpConfig.Transceiver[MEIMotorTransceiverIdA].Config =
    MEIMotorTransceiverConfigOUTPUT;

xmpConfig.UserOutInvert[0] = 1;

/* new way */
xmpConfig.Io[RMBMotorIoConfigXCVR_A].Source = RMBMotorIoTypeOUTPUT;
xmpConfig.Io[RMBMotorIoConfigXCVR_A].Invert = 1;

xmpConfig.Io[RMBMotorIoConfigUSER_0_OUT].Invert = 1;

```

The following is the definition of MEIMotorConfig:

```

typedef struct MEIMotorConfig {
    MEIMotorEncoder    Encoder[MEIXmpMotorEncoders];
    MEIMotorStatusOutput    StatusOutput;

    long                EncoderTermination;
    long                SIM4;
    MEIMotorDacConfig    Dac;

    /***** NEW STRUCTURE *****/

```

```

MEIMotorIoConfig Io[MEIMotorIoConfigIndexLAST];

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;

MEIXmpLimitData          Limit[MEIXmpLimitLAST];

MEIXmpMotorTorqueLimitConfig TorqueLimitConfig;

long AmpDisableWithLSR;
MEIXmpStatus XEStopAction;

} MEIMotorConfig;

```

2.70 SynqNet Network Init Method

MPI 945

The meiSynqNetInit(...) function is the same as the network initialization performed by mpiControlInit(...). This is useful for applications that want to re-initialize a network that has faulted (shutdown). It will not rediscover a changed network topology. If you want to "Re-Initialize" a network and re-discover the topology, use mpiControlReset(...).

2.71 Changes to meiFilterPostfilter(...)

MPI 944

The following methods are used to set and get post filter coefficients. They have updated in order to include new biquad types.

```

meiFilterPostFilterGet
meiFilterPostfFilterSet
meiFilterPostfilterSectionGet
meiFilterPostfilterSectionSet

```

Each method has been updated to include the use of three new biquad filter types. In addition to the standard floating point biquad filter, there is also:

- a fixed point biquad filter type
- a floating point state space biquad filter type
- a fixed point state space biquad filter type

Definition changes in the meiFilterPostfilter() methods:

The resonator section has been redefined. New equations are in the document postfilter.pdf.

The old resonator's bandwidth asymptotically agreed with the notch filter as the dB gain approached negative infinity. However, two old resonators with the same center frequency and bandwidth, but opposite gains (example: +10dB and -10dB) would not produce unity gain together. This made it difficult to cancel resonances effectively.

Two new resonators with the same center frequency and bandwidth, but opposite gains do produce unity gain together. The new definition of bandwidth is the full width, half max. deflection on a dB -vs- log(frequency) graph. It is now much easier to cancel resonances. However, the new resonator's bandwidth no longer asymptotically agrees with the notch filter.

Bug fixes to the meiFilterPostfilter() methods:

- 1) The lead and lag filters produced a constant gain regardless of frequency.
- 2) Wrong identification of lead and lag filters.
- 3) Specifying a zero gain filter created a bilinear filter on the XMP. This caused problems when trying to remove postfilter sections.
- 4) meiFilterPostfilterSectionSet() could not specify a filter length of zero on the XMP. This method could not remove all postfilter sections.
- 5) High pass, lead, and lag filters had 180 degree phase lag at zero frequency. This caused the lead and lag filters to be unstable in closed loop systems.
- 6) In the documentation postfilter.pdf, there was an incorrect sign for the equation of parameter a1 for the notch, resonator, lead, and lag filters.
NOTE: The MPI methods always had the correct sign.

2.72 New sqNodeFlash Utility

MPI 942

The sqNodeFlash Utility loads, saves, or erases sqNodeflash memory on a SynqNet node/drive. For more information about how to use this utility, please go to <http://support.motioneng.com/util/sqnode/home.htm>.

2.73 meiFlashMemoryFromFile(...) no longer supports local motion blocks

MPI 938

Since SynqNet controllers do not have local motion blocks, the local motion block FPGA download capability has been removed from MPI library. In version 20020924 and later, meiFlashMemoryFromFile(...) only supports FPGA download for the SynqNet network FPGA.

2.74 Position Triggered Motion

MPI 931

In version 20020905, pre-loaded motion triggering was extended to support triggers from command position, actual position and a specified address. Also, the comparison logic was extended to support "less than or equal" and "greater than or equal" comparisons. The new motion attribute masks are:

MEIMotionAttrMaskHOLD_LESS	Less than or equal to logic.
MEIMotionAttrMaskHOLD_GREATER	Greater than or equal to logic.

The new motion hold types are:

MEIMotionAttrHoldTypeAXIS_POSITION_ACTUAL	Input comparison from an axis' actual position.
MEIMotionAttrHoldTypeAXIS_POSITION_COMMAND	Input comparison from an axis' command position.
MEIMotionAttrHoldTypeUSER_ADDRESS	Input comparison from a controller address.

MEIMotionAttrMaskHOLD_LESS and MEIMotionAttrMaskHOLD_GREATER

Can be used with or MEIMotionAttrHoldTypeAXIS_POSITION_COMMAND to configure the pre-loaded move to trigger off of any axis' actual or command position. The LESS or GREATER attributes can be used with the MEIMotionAttrHoldTypeUSER_ADDRESS type to trigger the pre-loaded motion from any value in the XMP memory. When using AXIS_POSITION_ACTUAL or AXIS_POSITION_COMMAND, the MEIMotionAttrHoldSource axis.number and axis.position must be specified. When using USER_ADDRESS, the MEIMotionAttrHoldSource user.address, user.mask, and user.pattern must be specified.

For example, to trigger a move when Axis 1's command position exceeds 10000:

```
MPIMotionParams      params;      /* motion parameters */
MEIMotionAttributes  attributes;  /* motion attributes */
MEIMotionAttrHold    hold;        /* hold attribute configuration */

hold.type = MEIMotionAttrHoldTypeAXIS_POSITION_ACTUAL;
hold.source.axis.number = 1;
hold.source.axis.position = 10000;

attributes.hold      = &hold;
params.external      = &attributes;

/* Load motion profile with HOLD attribute */
returnValue =
    mpiMotionStart(motion,
        (MPIMotionType) (MPIMotionTypeS_CURVE |
            MEIMotionAttrMaskHOLD_GREATER),
        &params);
```

2.75 New Drivers for WinNT and Win2000

MPI 858

In version 20021010, the XMP device driver driver was updated. The new drivers for WinNT and Win2000 fix an intermittent bug check condition experienced on multiprocessor systems when enabling interrupts.

2.76 MPI Trace Utility

MPI 854

The MPI Trace utility (trace.exe) displays all supported MPI/MEI trace bit masks. These bits masks are useful for debugging code operation. All sample applications and utilities support the '-trace 0xNNNNN' command-line argument. Use any combination of the trace bits displayed by this utility in the hexadecimal number passed along with the '-trace 0xNNNNN' command-line option.

2.77 Thread Options Parameter in AppUtil Library

MPI 834

In version 20020907, the AppUtil library was modified to add the ThreadOptions parameter to the threadCreate() function. The ThreadOptions structure and the new threadCreate() prototype are:

```
typedef struct ThreadOptions {
    long stackSize; /* Bytes (0 => DEFAULT) */
} ThreadOptions;

/* Create/Delete/Validate */
const Thread
    threadCreate(ThreadOptions *options);
```

The options parameter was added to allow the calling application to specify different thread creation options (i.e. thread stack size). Passing in NULL for the *options pointer will cause the thread to be created with default options. Default stack size for Win32 operating systems is 2MB.

WARNING: This change is NOT backwards compatible. If your application uses the threadCreate(), and you choose to recompile, you will need to make a modification to your source code.

2.78 Change Software Limits to Trigger from Actual Position

MPI 830

In previous firmware releases, the software positive and negative position limits were based on the command positions. The limits were changed to trigger based on the actual position in firmware version 385A1. This change was implemented to improve safety.

Version 20020403.1.3

	New Version	Previous Version
Firmware	363A6	358A2
MPI Library	20020403.1.3	20020117.1.3

2.79 Support for CAN

A CAN interface is now supported by the MPI library and is available on several MEI XMP Motion Controllers. CANOpen is a serial bus topology used to connect a wide variety of I/O nodes together onto one network.

The interface provides a network master interface conforming to CANOpen DS301 version 4.01.

- The interface provides diagnostic, as well as error detection features, which allow the user's program to detect and react to network failures.
- The interface can support networks of up to the CANOpen maximum of 127 nodes, with each node supporting 64 digital inputs, 64 digital outputs, 8 analog inputs and 8 analog outputs.
- The interface provides a configurable system that allows the user to configure quantities such as the network bit rate and cyclic rate.

For more information about the implementation and programming interface, please see the CAN Application Note (9D00-0162) by going to the PDFs section at <http://support.motioneng.com>.

2.80 SynqNet Support - Phase I

This software release supports SynqNet™ controllers. SynqNet is a high performance synchronous network designed for multi-axis motion control applications. For more information about SynqNet please contact MEI or visit the SynqNet website: www.synqnet.org

The SynqNet Phase I software supports up to 6 nodes with a maximum sample rate of 5kHz. Each node can support up to 4 motors. SynqNet compatible drives are available from several manufactures. Plus, MEI offers an RMB-10v to bridge between the SynqNet network and standard analog drives. Analog controller and SynqNet Phase I software support similar features.

Future SynqNet software (Phase II) will support additional features:

- Up to 32 nodes on one network.
- Scalable network timing.
- Software configurable data packets.
- Network fault recovery when a single cable is broken.
- Software tools to analyze and optimize network performance.
- Downloadable FPGA and drive binary code over SynqNet.

2.81 SynqNet Node Status LED

MPI 865

The status LED associated with each motion block or SynqNet node is commanded ON and OFF by the firmware at a rate of once every 4096 samples (2048 on, 2048 off). Thus a flashing LED indicates that the motion block or SynqNet node is receiving data. This feature was implemented in firmware version 358A2.



2.82 MPI DLL Name Change

MPI 792

The debug version of the MPI dll has been renamed. It is now named **mpivc60d.dll** (formerly mpivc40.dll). The release version of the MPI dll has been renamed. It is now named **mpivc60.dll** (formerly mpivc40.dll). The MPI dlls are now compiled with Microsoft Visual C/C++ 6.0.



2.83 MSVC60 Makefiles

MPI 792

The MPI makefiles have been converted from the MSVC 4.2 format to the MSVC 6.0 format. Previously, a single makefile (.mak) was included for each workspace. These have been replaced with MSVC 6.0 makefile types (.dsp and .dsw).

2.84 New Rincon FPGA Image (version 1.2.6)

MPI 776

The Rincon FPGA binary image has been updated from version 1.2.3 to 1.2.6. Your SynqNet™ controller hardware is automatically updated to this new image when downloading the new firmware. The Rincon

binary image filenames are: 125_9501.fpg (PMC) and 125_9101.fpg (CPCI). It contains the following new features:

- Support for XMP-SynqNet-PMC Rev 2 local Opto I/O.
- Support for an activity LED per port:
 - ON** : both transmit and receive packets detected (normal operation)
 - BLINK** : transmit packets detected (discovery with no reply)
 - OFF** : no transmit packets (network shut down)

NOTE: the previous FPGA versions toggled the LED for each transmit

2.85 Support for local PMC I/O

MPI 745

New controller I/O methods have been added to access additional controller I/O bits that exist on some SynqNet™ controllers.

2.86 Brake enable/disable delay

MPI 533

The Brake feature sets the User Output (one per motor) to an active state when the Amp Enable output is disabled and sets the User Output to an inactive state when the Amp Enable is enabled. The Brake feature also supports specifiable delay times between the Amp Enable/Disable and User Output logic.

The following data structure has been added to the **motor.h** header file to support the Brake feature.

Data Structure

```
typedef enum{
    MPIMotorBrakeModeINVALID = -1,

    MPIMotorBrakeModeNONE,
    MPIMotorBrakeModeDELAY,

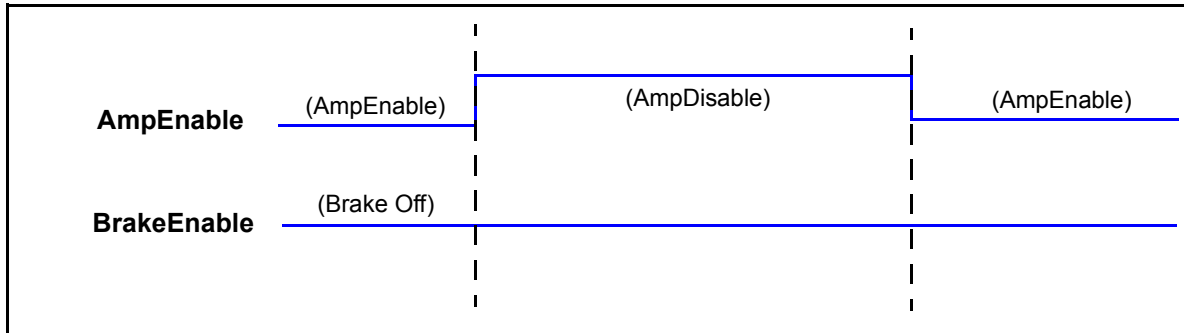
    MPIMotorBrakeModeLAST,
    MPIMotorBrakeModeFIRST = MPIMotorBrakeModeINVALID + 1
} MPIMotorBrakeMode;

typedef struct MPIMotorBrake {
    MPIMotorBrakeMode mode;
    float enableDelay;
    float disableDelay;
} MPIMotorBrake;
```

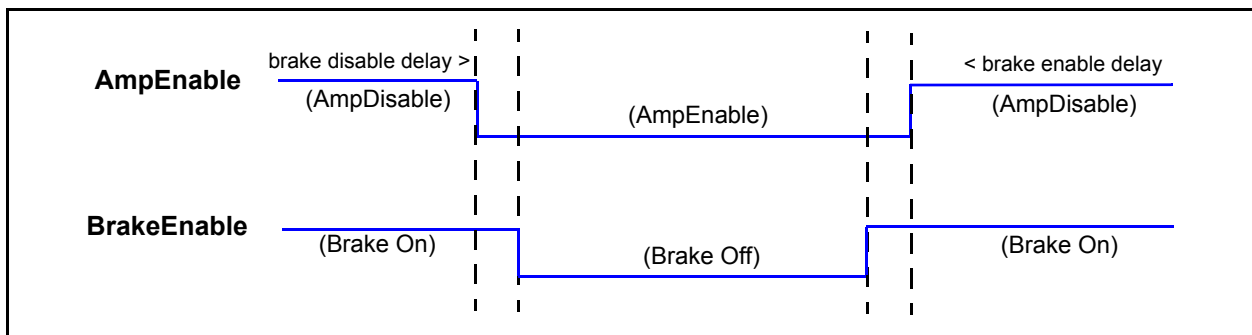
Description of Members

- mode** - set to MPIMotorBrakeModeNONE for no brake, set to MPIMotorBrakeModeDELAY for use brake feature specified delays.
- enableDelay** - specifies the delay (seconds) between when the brake is active and the amp enable is disabled.
- disableDelay** - specifies the delay (seconds) between when the amp enable is enabled and the brake is inactive.

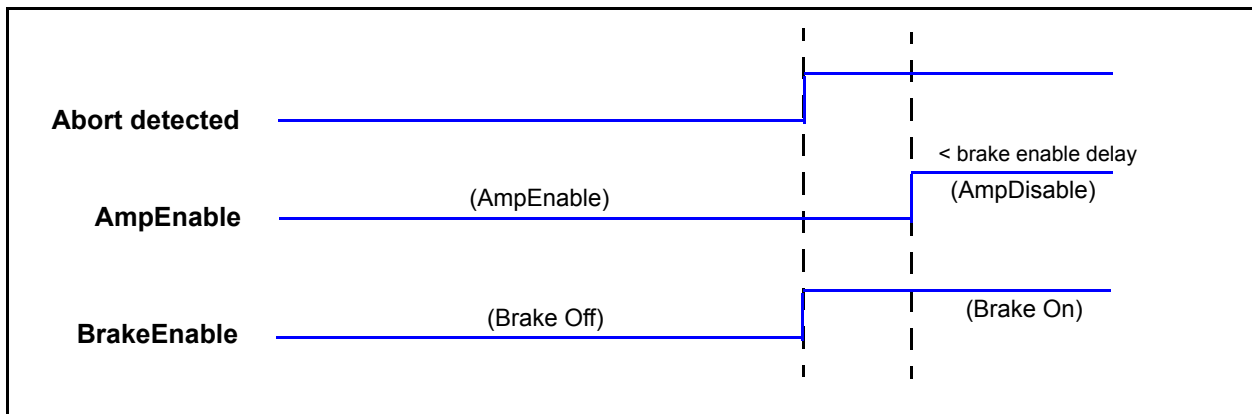
The following diagrams further explain how the Brake Enable/Disable Delay feature works:



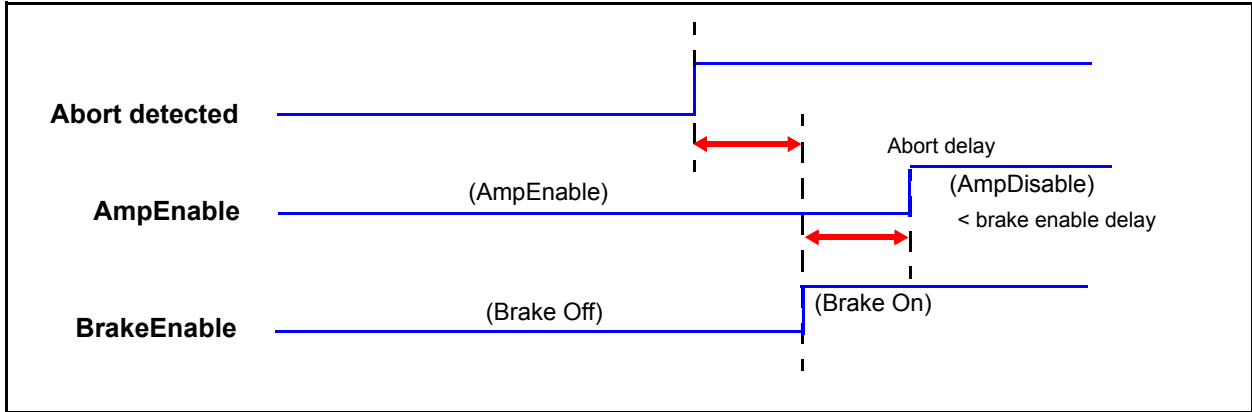
Case 1. Amp On/Off with NO Brake



Case 2. Amp On/Off with Brake



Case 3. Amp Off after ABORT Detection (in system with NO abort delay)



Case 4. Amp Off after ABORT Detection (in system WITH abort delay)

3 Incremental Changes

Since the last Production Release of the MPI, version 20020117.1.3, API changes have been made to the header files that add features and fix various software and firmware bugs. Please see the **\XMP\doc\header_diff.doc** file to see a compiled list of the incremental changes that have taken place between the following two versions of the MPI library.

New Version	Previous Version
20030620.1.1	20030605

4 Fixed Bugs of MPI/MEI Libraries:

Version 20030620.1.1

	New Version	Previous Version
Firmware	459A6	459A6
MPI Library	20030620.1.1	20030605

RMB-10V and RMB-10V2 Brake Polarity

MPI 1149

In previous versions, the polarity for the motor I/O brake source was incorrect for fail-safe operation. In version 20030620.1.1, the brake source polarity was inverted to correct the problem.

Motor I/O and Dedicated Brake Conflict

MPI 1148

In previous versions, the methods `mpiMotorIoGet(...)` and `mpiMotorIoSet(...)` masked off the dedicated Amp Enable bit, but did not mask off the dedicated brake bit. This could cause the dedicated brake signal to fail if `mpiMotorIoSet(...)` was called by an application. To fix the problem, the Amp Enable mask was removed from the `mpiMotorIoGet(...)` method, so an application can read the Amp Enable state. `mpiMotorIoSet(...)` now masks off of the Amp Enable and Brake Release bits. This problem was corrected in version 20030620.1.1.

Version 20030605

	New Version	Previous Version
Firmware	459A6	430A2
MPI Library	20030605	20021212

Frame Buffering Fails with Multi-Point Motion

MPI 1139

In versions since 20010620, any multi-point motion using more than 64 frames (points) for axis maps that did not include Axis 0 would incorrectly handle a "frames low" event. This would cause an E-Stop action when the controller ran out of frames. An internal method, `meiMotionFramesLow(...)` was only using Axis 0 to determine the frame buffer status. This problem was corrected in version 20030605.

PVT Moves Do Not Start

MPI 1136

In previous versions, sometimes a Motion Supervisor with multiple axes would not execute a PT, PVT, or other spline motion start. The problem was caused by one axis moving prematurely. Normally, a Motion Supervisor with more than one axis will wait for all axes to complete their motion before starting a new move. If one axis started before the others, the Motion Supervisor would enter a deadlock, since it would not start the other axes until the first axis had completed its motion. Also, the first axis would never finish the motion because the Motion Supervisor was holding the feedrate at zero (for the other axes).

The premature start of motion occurred when an axis had a NULL pointer to its Motion Supervisor and the host loaded a start frame (for PVT moves, etc.). Another cause of premature motion starting occurred if the axis' Motion Supervisor pointer was pointing to a different (single-axis) motion supervisor than when the start frame was loaded.

This problem was corrected in version 20030603, by adding a check in the firmware to determine if the

Motion Supervisor pointer was NULL and by having the host write the correct Motion Supervisor pointers to each axis before loading any frames.

CANOpen Heartbeat Problem

MPI 1132

In CAN firmware version 002A3, configuring the heartbeat consumer field of the object dictionary generated an error. The object dictionary subindex for writing to the heartbeat consumer field of the object dictionary should have been a 1, not a 0. This problem was corrected in CAN firmware version 002A4.

Controller Firmware Lock-up

MPI 1126

In previous firmware versions, if the Motion Supervisor/Axis map was modified by setting the number of axes associated with an Motion Supervisor to zero, there was an occasional firmware lockup. The problem was caused by a compiler issue with a for(...) loop. If the number of axes was set to zero during the time when the for(...) loop was executing, it would continuously loop until the maximum value (65536) was reached. This problem was corrected by using a temporary variable for the number of axes, which can only be updated outside the for(...) loop. This problem was corrected in versions 430B4, 453A4, and later).

Glentek Omega Drive Initialization Problem

MPI 1097

In previous versions, the Glentek Omega drive initialization would fail, making it impossible to enable the amplifier. The problem was caused by an incorrect sequence of the PLL configuration. This problem was corrected in versions 20021212.1.8 and 20021219.

meiSqNodeDriveParameterSave(...) Timeout

MPI 1088

In previous versions, the meiSqNodeDriveParameterSave(...) method would return a timeout error when the controller was configured for an 8kHz sample rate (or higher). This problem was caused by a missing conversion from seconds to controller samples. This problem was corrected in version 20021212.1.7 and later.

Kollmorgen CD Drive: "C1" Error after Reset

MPI 1078

In previous versions, the Kollmorgen CD-Series drive would report a "C1" error after a controller reset or network initialization. The error was caused by service commands being sent to the drive before it completed its firmware boot sequence. This problem was corrected by increasing the SqNodeLib delay to wait for the drive firmware to complete its boot sequence. This problem was fixed in version 20021212.1.5.

Controller Reset Causes Drive Errors

MPI 1074

In previous versions, an mpiControlReset(...) would immediately shutdown the SynqNet network and reset the controller. Some SynqNet drives logged this as an error condition and after initialization it would display an error code. To correct this problem, the mpiControlReset(...) was modified to perform a safe shutdown to notify the SynqNet nodes that the network would be shutdown. This was corrected in versions 20021212.1.4 and 20030207.

Position Jump During Move After Position Clear

MPI 1073

In previous firmware versions, if an axis's actual position was changed (not by a commanded move), then the command and actual positions are cleared, the next move will be calculated improperly. This problem was caused by an initial position and origin compensation for the first move frame. This problem was corrected in firmware version 433A4.

Motion Supervisor Produces TIMEOUTS on Motion Methods **MPI 1070**

A bug existed that produced a timeout return value from a mpiMotorStart(...) call. This return value only occurred when attempting to start a trapezoidal move after an MPIMotionActionRESET had been performed, which was preceded by a long PT/PVT move (that had been stopped) with more than the 128 points. The PT/PVT move could be stopped by performing a MPIMotionActionSTOP but, calling MPIMotionActionRESET action would not completely clean-up the stopped PVT move, thereby leaving the MPI in an unstable condition. This problem was fixed in the 20021212.1.4, as well as any releases after 20030207.

Platform Reset with Client Server **MPI 1064**

A bug existed that produced an mpiControlReset instead of an meiPlatformReset on the server when requested by the client. When meiPlatformReset was called with platform--> type == MPIControlTypeCLIENT, MEIRemoteMethodRESET was passed to meiClientMethod. The Server then executed an mpiControlReset instead of an meiPlatformReset. This bug has been fixed in the 20021212.1.4 release. MEIPlatformReset is now called by the server program when requested by the client.

No software support for Capture2 on DASA **MPI 1050**

In previous versions, during SynqNet initialization, the DASA drive module did not allocate space for capture data in the SynqNet message structure (MEISynqNetMessage). This problem was fixed in the 20021212.1 release.

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.

CANFlash does not work with eXMP **MPI 1041**

In version 20021122, when the CANFlash utility was executed via client/server, it returned a "Control: firmware version mismatch" error. This problem was corrected in version 20030513.

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.

Problem with the Last Node's Repeater in a Ring Topology **MPI 1110**

A bug existed in the 20021212.1.8 release, where the last node in a ring topology enabled its repeater during the Discovery phase. If the repeater is enabled, the SynqNet message will be corrupted. This problem has been fixed in the 20021212.1.9 release. The repeater is no longer enabled.

Version 20021212

	New Version	Previous Version
Firmware	430A2	422A3
MPI Library	20021212	20021101

Sequencer Continuously Generates Events

MPI 1032

In previous versions, the sequence command MPICommandOperatorNOT_EQUAL would cause a sequencer to continually generate events to the host. This problem was caused by an improper definition for the MPICommandOperatorNOT_EQUAL. This has been corrected in version 20021212.

VM3.EXE Cannot Save or Load a Dump File

MPI 1013

In version 20021101, the VM3 utility could not save or load a controller memory dump file. This problem was corrected in version 20021121.

Version 20021101

	New Version	Previous Version
Firmware	422A3	420A6
MPI Library	20021101	20021030

- There were no bug fixes in the 20021101 release.

Version 20021030

	New Version	Previous Version
Firmware	420A6	420A2
MPI Library	20021030	20021025

- There were no bug fixes in the 20021030 release.

Version 20021025

	New Version	Previous Version
Firmware	420A2	415A6
MPI Library	20021025	20021021

- There were no bug fixes in the 20021025 release.

Version 20021021

	New Version	Previous Version
Firmware	415A6	415A5
MPI Library	20021021	20021014

- There were no bug fixes in the 20021021 release.

Version 20021014

	New Version	Previous Version
Firmware	415A5	415A4
MPI Library	20021014	20021011

Topology Flash problem

MPI 983

In the 20021011 release, a "SynqNet: node 0 : service cmd unsupported" error message would be returned in Motion Console after saving the controller topology to flash. This problem was fixed in the 20021014 release.

Processing problem with velocity moves

MPI 982

Axis did not check for ESTOP_CMD_EQ_ACT when processing velocity moves, so fault processing did not occur. This problem was fixed in the 20021014 release.

Version 20021011

	New Version	Previous Version
Firmware	415A4	363A6
MPI Library	20021011	20020403.1.3

Program Sequencer Fails

MPI 911

In firmware versions before 380A1, program sequencers other than number 0 would fail. This was due to an Analog Devices compiler bug. A compiler patch has been implemented. This problem was fixed in versions 380A1 and later.

meiFlashMemoryVerify(...) missing from flash.h

MPI 881

In the 20000913 release, the FPGA and SHARC code were 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 and during flash download the flash memory was modified to initialize 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. As a result, the prototype was removed from flash.h. In version 20020117.1.4, the prototype was added back to flash.h. To use meiFlashMemoryVerify(...) successfully, save the existing flash image to a file (.img) using meiFlashMemoryToFileType(..., MEIFlashFileTypeALL). After it has been saved, it can be compared to the controller's flash image using meiFlashMemoryVerify(..., MEIFlashFileTypeALL). See the sample program, checkFlash.c for more details.

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 structure when neither MPIMotionAttrMaskSYNC_START nor MPIMotionAttrMaskSYNC_END are specified. This bug was fixed in the 20011011 release.

Version 20020403.1.3

	New Version	Previous Version
Firmware	363A6	358A2
MPI Library	20020403.1.3	20020117.1.3

ADC problem

In the course of verification testing on the RMB-10V, there was a problem with the ADC interface. Under certain operating conditions, the RMB-10V would return incorrect ADC values. This problem is caused by a timing error in the FPGA code, C0FE0001_20011105.sff. This problem also exists in FPGA code C0FE0003_20011105.sff.

A patch for C0FE0001_20011105.sff has been developed and verified. The patch version is C0FE0001_20020430.sff.

No patch has been developed for C0FE0003_20011105.sff.

Config utility changes Capture configuration

MPI 835

In MPI versions prior to 20020222, a bug existed where the *config.exe* utility could incorrectly set the Capture configurations for unmapped capture objects. Since the objects are not mapped to motors, no performance change would be experienced, however differences in the config.exe output files could be noticed. This has been fixed in releases after the 20020305 version.

MS/State variable toggling between IDLE and STOPPED

MPI 831

In the 362A4 (20011220.1) and 363A3 (main tree) firmware versions, it is possible to get the axis state machine in a situation where the state will cycle continuously between IDLE and STOPPING. This can only happen if a STOP has occurred at the same time that a start frame has been cued and is waiting to be executed. When this happens, the axis state machine continuously cycles through its various states reacting to the STOP.

To get into this situation, a startMotion needs to be called while any motor limit that is configured for a STOP event is triggering. When this happens, the STOP is cleared by the Motion Supervisor when the start frame is loaded. The motor will then evaluate the limit as triggered, and sets the STOP event in the

axis status. When the axis state machine executes, it sees a STOP at the same time a start frame is cued and waiting to be executed.

This problem was fixed by adding code to the axis state machine that prevents the axis from changing the state from IDLE when a STOP or any error has occurred. The start frame will no longer execute and the state will remain IDLE.

Reversed functionality of mpiEventMgrServiceConfigGet/Set() MPI 808

The functionality of mpiEventMgrServiceConfigGet/Set() was reversed in releases prior to version 20020212. Before the fix, mpiEventMgrServiceConfigGet() would set the configuration and mpiEventMgrServiceConfigSet() would get the configuration. This has been fixed in the 20020212 MPI release.

5 Existing Bugs of MPI/MEI Libraries:

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.

6 Limitations of MPI/MEI Libraries:

WinNT Driver Invalid Board Number Bug

MPI 568

The MEIXMP device driver can support a maximum of 8 XMP-Series controllers.

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.

mpiCommandCreate(..) fails with MotionModify

CRN 399

MPIMotionModify is not supported in sequences.

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.

7 Motion Console and Motion Scope

7.1 Closed Issues: Motion Console

	New Version	Previous Version
Motion Console	03.39.07	03.39.05

Modified in Version: **03.39.07**

Modification Type: **CR (Change Request)**

Number **Name**

959 **MPIMotionTypeP and MPIMotionTypePF removed from MPI**

The MPIMotionTypeP and MPIMotionTypePF motion types were removed from the MPI and so were also removed from MotionConsole.

Modified in Version: **03.39.06**

Modification Type: **DR (Discrepancy Report)**

Number **Name**

958 **FPGA Type Erroneously Interpreted as BOOT**

If meiSqNodeInfo() fails before setting MEISqNodeInfo.fpga.type, Motion Console will mistakenly interpret the FPGA type as BOOT and prompt the user to download a runtime binary image to the node. It is also possible that Motion Console will misinterpret the MotorCount and MotorOffset in this situation.

Modified in Version: **03.39.05**

Modification Type: **DR (Discrepancy Report)**

Number **Name**

955 **CRC Error Counts In/Out are Reversed**

The CRC Err. IN 0 and OUT 0 status attributes on the SqNode Summary were in the wrong order.

957 **Motion Console hangs when there is bad firmware on the CAN**

Motion Console was hanging for a very long time when adding a controller that had invalid firmware on the CAN chip.

Modification Type: **CR (Change Request)**

Number **Name**

956 **MEIMotorConfig.io.AbortValue was removed From MPI**

Since MEIMotorConfig.io.AbortValue was removed from the MPI, it was also removed from Motion Console.

Modified in Version: **03.39.04**

Modification Type: **NF (New Feature)**

Number **Name**

936 **Add new status bits to Motor Summary**

The "Amp Not Powered" and "Drive Not Ready" fault bits have been added to Motor Summary Status tab.

952 **Add check for valid MAC rev for SynqNet nodes**

When Motion Console is coming up or when a controller is refreshed, a check is made that each SynqNet node has a

valid MAC fpga version.

Modification Type: FE (Future Enhancement)

Number Name

928 Window Focus on View Sub-Objects

When the object selection of an Object Summary window is set by the user, that Summary window will be brought to the top. The focus will be set to the Summary window except when it is programmed from the Object Explorer, in which case the focus is left on the Object Explorer. This allows multiple Summary windows to be programmed from the Object Explorer without having the Object Explorer being obscured by the Summary windows.

Modification Type: DR (Discrepancy Report)

Number Name

953 Wrong Motor Fault Bits Displayed

The Motor Fault bits AMP_NOT_POWERED and DRIVE_NOT_READY were being displayed as ON when they were really OFF.

Modification Type: CR (Change Request)

Number Name

924 Alternative Add Controller Dialogue

The Add Controller dialog box has been changed. The controller type is now selected by clicking on a radio button. Fields that are not relevant to the selected controller type are disabled.

Modified in Version: **03.39.03**

Modification Type: NF (New Feature)

Number Name

891 Added TxFailure Event Status to SynqNet Summary

Added TxFailure Event Status to SynqNet Summary.

Number Name

934 Update dedicated I/O bits displayed in the Motor Summary

The dedicated I/O bits in the Motor Summary window has been updated to display the bits defined in stdmei.h. The following is a list of the presently displayed bits:

MEIMotorDedicatedInAMP_FAULT
MEIMotorDedicatedInBRAKE_APPLIED
MEIMotorDedicatedInHOME
MEIMotorDedicatedInLIMIT_HW_POS
MEIMotorDedicatedInLIMIT_HW_NEG
MEIMotorDedicatedInINDEX
MEIMotorDedicatedInFEEDBACK_FAULT
MEIMotorDedicatedInCAPTURED
MEIMotorDedicatedInHALL_A
MEIMotorDedicatedInHALL_B
MEIMotorDedicatedInHALL_C
MEIMotorDedicatedInAMP_ACTIVE
MEIMotorDedicatedInWARNING

MEIMotorDedicatedOutAMP_ENABLE
MEIMotorDedicatedOutBRAKE_RELEASE

935 Add masks for new nodeAlarm configuration features

Extended SqNode Summary Config tab to support two new nodeAlarm configurations that have been added to sqNode.h:

notCyclicEnable

ioAbortEnable

939 Add Capture Count to Controller Summary

Add access to MPIControlConfig.captureCount to the Controller Summary Config tab.

941 Add Motor Feedback to Motor Summary

The encoder feedback for both the primary and secondary encoders is now displayed on the Status tab of the Motor.

942 Add new fault bits to Motor Summary

The following status bits have been added to the Status tab and to the Fault Config attribute on the Config tab of Motor Summary:

MEIMotorFaultMaskAMP_NOT_POWERED

MEIMotorFaultMaskDRIVE_NOT_READY

943 Comm Failure Event Status Added to SqNode Summary

MEIEventTypeSQNODE_COMM_FAILURE has been added to the Status tab of the SqNode Summary.

944 Motion Type Should Be Saved as Text

In the .INI file, Motion types should be saved as text strings rather than as their actual enum values. This will allow obsolete motion types to be recognized and interpreted to be something safe.

945 Add Drive Count to SqNode Summary

The drive count is now displayed in the SqNode Summary.

947 Add Shutdown Button To SynqNet Summary

A button was added to the SynqNet Summary that shuts down the SynqNet network when clicked.

*Modification Type: **MI (Minor Improvement)***

Number **Name**

895 Update the Motor Event Encoder Fault Trigger configuration to match the MPI

Motor encoder fault trigger conditions changed in MPI. Therefore, the interface in Motion Console changed as well.

*Modification Type: **FE (Future Enhancement)***

Number **Name**

927 Add Clear Topology Button to Controller Summary

Added a button to the SynqNet Summary that, when clicked on, will cause meiSynqNetTopologyClear() to be called.

*Modification Type: **CR (Change Request)***

Number **Name**

937 Move "Save Topology to Flash" Button to SynqNet Summary

Saving topology to flash has become a SynqNet operation in the MPI and is now reflected in Motion Console.

946 Display missing events in SynqNet Summary Status Tab

The following events were not being displayed in the SynqNet Summary:

MEIEventTypeSYNQNET_TX_FAILURE

MEIEventTypeSYNQNET_NODE_FAILURE

MEIEventTypeSYNQNET_RECOVERY

950 Build Motion Console Release configuration with MPI Release configuration

Prior to this change, the Release configuration of Motion Console was linked to the Debug configuration of the MPI. The Release configuration of Motion Console is now linked to the Release configuration of the MPI.

Modified in Version: **03.39.02**

Modification Type: **DR (Discrepancy Report)**

Number **Name**

930 Remove Unsupported MotionTypes from Motion Supervisor Summary

All MPIMotionTypes that Motion Console does not support have been removed from the choice list in the Motion Supervisor Summary.

Modified in Version: **03.39.01**

Modification Type: **NF (New Feature)**

Number **Name**

892 Add "Out of Frames" fault bit to Motion Supervisor Summary

Added the "Out of Frames" fault bit to the Motion Supervisor Summary.

Modification Type: **MI (Minor Improvement)**

Number **Name**

915 Display 5th gain table

A 5th gain table was added to the filter configuration. It is currently not accessible in Motion Console.

Modification Type: **CR (Change Request)**

Number **Name**

917 Changes in MEIMotorEncoderType enums

Changes to the MEIMotorEncoderType enums resulted in changes to the encoder type display in the Motor Summary window.

918 MPIStateINIT dropped, MPIStateSTOPPED added

Changes to the MPIState enum in the MPI resulted in changes in Motion Console.

919 MPIMotorBrake structure changed

MPIMotorBrake.enableDelay and disableDelay were changed to applyDelay and releaseDelay, resulted in changes in Motion Console.

920 Motor I/O Text and Count Should Match Drive Specs

The text and count for the Motor I/O types now come from the SqNodeLib module, so they match the actual drive specifications.

921 Reset Button in Error Window Renamed

At the bottom of the function error window there was a reset button. Users could accidentally misinterpret this as the controller reset button. This button has been renamed "Clear Error List."

925 Motor Encoder Termination Removed

Motor encoder termination was removed from the MPI and was also removed from Motion Console.

926 Repeat Mode Disabled Unnecessarily

Under the following conditions, Motion Supervisor Repeat Mode will be disabled with no GUI feedback:

- Repeat Mode is enabled on an Motion Supervisor
- The Motion Supervisor is in an error state
- The user commands motion

There is actually no reason to disable Repeat Mode in this situation.

Modified in Version: **03.39.00**

Modification Type: **NF (New Feature)**

Number **Name**

911 **[Dup. of 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.

Modified in Version: **03.38.21**

Modification Type: **NF (New Feature)**

Number **Name**

908 **[Dup. of 906] The user should be able to pause scrolling errors**

A "Pause" button has been added to the Library Function Errors window. Clicking on the button will pause the display of future error messages. This is useful when errors are scrolling by too fast to be read.

910 **[Dup. of 736] Add a "Copy to Clipboard" button to the Library Function Errors' window**

A button has been added to the Library Function Errors' window that, when clicked, will copy the text of the current set of errors into the clipboard. The user can also select text in the error list and copy it to the clipboard using the Ctrl+C shortcut or the Edit/Copy menu item.

Modification Type: **DR (Discrepancy Report)**

Number **Name**

909 **[Dup. of 907] Long library errors can now be completely seen**

A scrollbar added to the Library Errors window now allows the entire text of library errors to be seen, even when it does not completely fit into the space provided.

Modification Type: **CR (Change Request)**

Number **Name**

912 **Update Help File to SynqNet Phase 2**

The help file have been updated to the SynqNet Phase 2 documentation.

Modified in Version: **03.38.20**

Modification Type: **CR (Change Request)**

Number **Name**

903 **Change to motor encoder event configuration in MPI**

Motor encoder event configuration in the MPI, MPIMotorConfig.event[MPIEventTypeENCODER_FAULT], was changed from a bit-mask to an enum. MotionConsole has been updated to reflect this change.

Modified in Version: **03.38.19**

Modification Type: **NF (New Feature)**

Number **Name**

889 **Add sqNode info.id.exactMatch to the sqNode Info Summary screen**

The "Exact Match" attribute is now displayed in the SqNode Summary. It tells the user whether or not the SqNodeLib was able to identify the node.

890 **Add Broken Wire and Illegal State for Secondary encoders in the Motor Status screen**

Broken Wire and Illegal State status flags for the secondary encoder are now displayed in the Status tab of the Motor Summary.

Modification Type: **MI (Minor Improvement)**

Number **Name**

893 Temporarily remove the "Invert" option in the Motor I/O screen

The "Invert" option in the Motor I/O tab of the I/O Summary has been removed due to a lack of support from the FPGA.

Modified in Version: **03.38.18**

Modification Type: **MI (Minor Improvement)**

Number **Name**

888 Add "FPGA" in SqNode Binary Download dialog box

Changed "Node" to "Node FPGA" in the list control. Also changed "Drive #" to "Drive Processor Firmware."

Modified in Version: **03.38.17**

Modification Type: **NF (New Feature)**

Number **Name**

879 Add User Fault Action support to Motor Config

The User Fault Action configuration was added to the Motor Summary screen under the config tab.

885 Add Out of Frames Status to Motion Console

Out of Frames event status (MEIEventTypeMOTION_OUT_OF_FRAMES), has been added to the Motion Summary.

886 Add User Fault status to SqNode Summary

User Fault event status (MEIEventTypeSQNODE_USER_FAULT) has been added to the SqNode Summary.

Modification Type: **MI (Minor Improvement)**

Number **Name**

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.

Modified in Version: **03.38.15**

Modification Type: **MI (Minor Improvement)**

Number **Name**

875 Transmission is misspelled

Transmission was misspelled as "transmition" in several places.

876 Scrolling errors generated from SqNode Summary

The error window was scrolling library errors caused by an error returned by meiSqNodeStatus(). The error is now reported only once.

Modification Type: **CR (Change Request)**

Number **Name**

874 [Dup. of 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.

Modified in Version: **03.38.14**

Modification Type: **NF (New Feature)**

Number **Name**

849 **Add sqNodeFlash download support to MoCon**

For SynqNet Phase II, add support to download sqNodeFlash images. This includes FPGA and drive firmware images. All of the binary images will be stored in the XMP\bin directory, so MoCon should open the file dialog box in this directory. Some download processes can be quite lengthy, so a progress status bar and an abort button would be very helpful. After SynqNet initialization, nodes will require "working" FPGA images to operate. From the factory, nodes will be shipped with "boot" FPGA images. MoCon should detect sqNodes with "boot" FPGA images and query the user to download "working" FPGA images. Note: MoCon uses a similar technique with XMP controller firmware.

851 **Save SynqNet Topology to Flash**

Add a button to save the SynqNet topology to the Controller's flash using meiControlTopologyToFlash(...)

Modification Type: **DR (Discrepancy Report)**

Number **Name**

854 **Need to refresh SqNode and Motor objects after downloading flash**

After downloading firmware to a node, the object configuration is not being refreshed in the SqNode Summary. The Motor Summary will also need to be refreshed.

Modification Type: **CR (Change Request)**

Number **Name**

848 **Add phase 2 SynqNet support**

Make necessary changes to accommodate the SynqNet Phase 2 changes in the MPI.

Modified in Version: **03.38.13**

Issue Type: **NF (New Feature)**

Number **Name**

860 **Move Configurable Motor I/O to the I/O Summary**

For SynqNet Phase 2, display the configurable motor I/O in the I/O Summary. Remove XCVR and USER I/O from the Motor Summary.

Issue Type: **DR (Discrepancy Report)**

Number **Name**

855 **Summary Configuration Gets Confused When Configured for Many Objects**

A bug in the way 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.

857 **Help/About Doesn't Display Version Info if User's Locale is not Supported**

If the user's locale is set to be a non-supported locale (e.g. English, Canada), then the Help/About dialog will not show the correct application version or copyright date.

862 **Motion Console Crashes if Active Tab Page >= Tab Page Count**

When Motion Console is executed with the .INI file that was created with a different version of Motion Console, there is a chance that it will crash. The crash will occur if the Active Tab Page of a Summary is set to be higher than the current number of tabs in the Summary.

Modified in Version: **03.38.12**

Modification Type: **NF (New Feature)**

Number **Name**

861 **Add Motor Fault Config and Status to Motor Summary**

The following changes have been made to the Motor Summary:

- 1) Motor fault configuration has been added to the Config tab
- 2) Motor fault status has been added to the Status tab.

Modified in Version: **03.38.10**

Issue Type: **DR (Discrepancy Report)**

Number **Name**

846 **mpiControlReset inadvertently called when downloading flash**

A bug in the "Download to Flash" algorithm was causing errors after downloading to flash.

Modified in Version: **03.38.09**

Issue Type: **CR (Change Request)**

Number **Name**

842 **Add text for new SynqNet states**

Text strings have been added for the following new SynqNet states:

MEIXmpSynqNetStateBOOT

MEIXmpSynqNetStateSHUTDOWN

Modified in Version: **03.38.08**

Issue Type: **MI (Minor Improvement)**

Number **Name**

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.

Issue Type: **DR (Discrepancy Report)**

Number **Name**

750 **Sine Comm error message limits ability to fix problem**

There is no longer a minimum number of Aux DACs required in order to switch to no commutation mode.

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.

807 **Panic action does not stick**

The panic action setting was not being saved when minimizing and restoring Motion Console.

829 **[Dup. of 828] SynqNet Block Count display not updated**

The Block Count displayed in the SynqNet Summary may display an invalid value after refreshing or resetting the controller. Clicking on the Block Count cell will force the correct value to be displayed.

Issue Type: **CR (Change Request)**

Number Name

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: **03.38.07**

Issue Type: **DR (Discrepancy Report)**

Number Name

824 **[Dup. of 262] Missing Help Feature**

Clicking on the menu item "Help/Help Topics" will now open an appropriate help document.

Modified in Version: **03.38.06**

Issue Type: **DR (Discrepancy Report)**

Number Name

820 **Invalid Title on the Download Firmware dialog box**

The title on the Download Firmware dialog box was: "Download Firmware To Controller %s." This has been changed to: "Download Firmware To Controller."

821 **Motion Console dies when MEI_INSTALL_DIR does not exist**

Motion Console 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.

Modified in Version: **03.38.05**

Modification Type: **NF (New Feature)**

Number Name

817 **Add Down/Up-load Firmware feature to the CAN Network object**

Added the ability to download and upload firmware to and from the CAN Network. The new buttons for this feature can be seen in the CAN Network Summary.

Issue Type: **DR (Discrepancy Report)**

Number Name

815 **meiCanNodeDigitalInputGet returns unknown error message**

- a) With Motion Console running and the network alive, reset the controller (in Motion Console)
- b) Note the error messages in the Library Function Error window. The error message is blank.

Within meiCanNodeDigitalInputGet(), meiCanVALID returns 6285196 (0x005fe78c). mpiMessage returns an empty string for this error message.

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 this directory normally does not exist, the initial directory for the browser is set to whatever the default is.

Modified in Version: **03.38.04**



Issue Type: **CR (Change Request)**

Number **Name**

808 **Make controller Local Motion Block Count read-only**

The Local Motion Block Count, displayed in the Controller Summary, has been made into a read-only attribute.

809 **SERCOS Count removed from Controller Summary**

The SERCOS Count has been removed from the Controller Summary.

Modified in Version: **03.38.03**

Modification Type: **NF (New Feature)**

Number **Name**

801 **Add support for CAN objects**

The following objects have been added to support CAN I/O: CAN Network, CAN Node, I/O. Summary windows associated with each of these objects can be opened and configured from the Object Explorer and by clicking on the associated buttons on the main frame toolbar.

Issue Type: **CR (Change Request)**

Number **Name**

800 **Remove SERCOS support from Motion Console**

All support for SERCOS objects have been removed. This includes the Sercos Ring, Sercos Node, and Sercos IDN objects. The objects no longer appear in the Object Explorer and the Summary windows are no longer available. The SERCOS tab in the Filter Summary has also been removed.

Modified in Version: **03.38.02**

Modification Type: **NF (New Feature)**

Number **Name**

787 **Add SynqNet TxTime to Control Config**

Add the SynqNet transmission time to the Controller Summary, Config tab page.

Issue Type: **MI (Minor Improvement)**

Number **Name**

793 **[Dup. of 788] Merge Japanese Translations Into Production Release Version**

Merge the current Japanese translations from the Production Release version.

Modified in Version: **03.38.01**

Issue Type: **CR (Change Request)**

Number **Name**

792 **[Dup. of 791] Made changes necessary to link with mpivc60d.lib**

Made all changes necessary to link with mpivc60d.lib.

7.2 Closed Issues: Motion Scope

	New Version	Previous Version
Motion Scope	01.21.06	01.21.06

Modified in Version: **01.21.06**

Modification Type: **CR (Change Request)**

Number **Name**

951 **Build Motion Scope Release configuration with MPI Release configuration**

Prior to this change, the Release configuration of Motion Scope was linked to the Debug configuration of the MPI. The Release configuration of Motion Scope is now linked to the Release configuration of the MPI.

Modified in Version: **01.21.05**

Modification Type: **DR (Discrepancy Report)**

Number **Name**

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: **01.21.04**

Modification Type: **MI (Minor Improvement)**

Number **Name**

813 **Export Motion Scope data with masking**

A check box has been added in the Edit trace properties (or Save data window) to allow saving/exporting data with mask enabled. This allows users to easily graph I/O bits or other masked data with other programs.

818 **Extend Motion Scope Redraw Time parameter**

Pane Mode dialog has an additional checkbox labeled "Redraw Pane on Done/Full Only" so that Pane is not drawn until the full data acquisition is completed. This feature is not available in "continuous mode" (where Trigger End is the Stop button). Additionally, when this feature has not been activated, the Pane Redraw time has been extended/limited to the current XRange value in milliseconds.

830 **Stock "status" Traces should be masked automatically on Pane Export.**

Stock "status" Traces are now masked and shifted automatically on Pane Export. The data value displayed on screen is always the same as the data sent to a file.

831 **User defined Traces to have no unit type ("None") by default.**

User defined Traces have no unit type ("None") by default (instead of "Counts").

832 **Line up columns in exported data files (for when font is fixed size).**

Columns in exported data files (Pane Export and File Save) are lined up for all data items when font is fixed size.

Modification Type: **FE (Future Enhancement)**

Number **Name**

812 **Include units in Motion Scope exported data**

Customer request to include the units at the head of each column of data in the text file. Either include in the row with Trace name or add second row to include units.

Modification Type: **DR (Discrepancy Report)**

Number **Name**

802 **The full out button should zoom to the XRange size**

The Zoom Full Out button now restores the X and Y Range settings to the original outermost view. Any existing zoom boxes are also preserved and the user may use Zoom In to return again to a previous Zoom state.

859 **[Dup. of 857] Help/About Doesn't Display Version Info if User's Locale is not Supported**

If the users locale is set to be a non-supported locale (e.g. English, Canada), then the Help/About dialog will not show the correct application version or copyright date.

878 **Text is wrong when opening a .PAN fails**

When MoScope failed to open a .PAN file because a pane was already for the controller, the error message was: Controller # already taken by another Pane. It was changed to include the controller number.

Modification Type: **CR (Change Request)**

Number **Name**

833 **More precision in Motion Scope exported floating point data**

Precision for float data saved to a file via the File|SaveAs or Pane|Export menu commands can now be specified via the dialog boxes for those commands. A maximum of 12 digits is allowed.

Modified in Version: **01.21.03**

Issue Type: **DR (Discrepancy Report)**

Number **Name**

827 **[Dup. of 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: **01.21.02**

Modification Type: **DR (Discrepancy Report)**

Number **Name**

825 **[Dup. of 262] Missing Help Feature**

Clicking on the menu item "Help/Help Topics" will now open an appropriate help document.

Modified in Version: **01.21.01**

Modification Type: **DR (Discrepancy Report)**

Number **Name**

804 **[Dup. of 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: **01.21.00**

Modification Type: **MI (Minor Improvement)**

Number **Name**

789 **Merge Japanese Translations Into Production Release Version**

Merge the current Japanese translations into the Production Release version.

Issue Type: **CR (Change Request)**

Number

Name

791

Made changes necessary to link with mpivc60d.lib

Made all changes necessary to link with mpivc60d.lib.

Modified in Version: **01.20.24**

Issue 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.

7.3 Open Issues: 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 doesnot 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), 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 Motion Supervisor summary is clicked unless the "Clear Fault" button is clicked first.

761 **Pulldown 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 function on the primary monitor.

949 **Motion Console hogs CPU when finding errors.**

954 **Scaling in Summary Windows Partially Implemented**

The size of the grids in a Summary window can be scaled using Ctrl+MouseWheel. The following needs to happen for the feature to be fully implemented:

- 1) Scale all grids in the summary window, not just the one with the focus
- 2) Adjust max size of Summary window after scaling
- 3) Adjust max splitter position after scaling
- 4) Provide some more obvious method of setting the scale

960 **[Dup. of 913] Column width is forgotten after Motion Console restarts**

When a column width has been modified by the user, the new width should be associated with the object and saved in the .INI file. This allows the column width to be restored to the user's preference when the summary window is re-opened.

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.

7.4 Open Issues: Motion Scope

Issue Type: **DR (Discrepancy Report)**

Number **Name**

542 Motion Scope 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 Motion Scope).
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 Motion Scope 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 Motion Scope 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 displays 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 see that Motion Scope really didn't. Perhaps there is some rounding error in the calculation of elapsed time during the motion?

806 Motion Scope loses all traces after a SynqNet node disappears

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. It is easy to remove a pane but it takes a lot of work to recover settings if they were needed.

852 Time scale on Motion Scope does not refresh 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, then 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.

896 [From MPI Libraries and Firmware :] moScope "Save Pane" Corrupts timebase.

The following problem was reported using MPI 20021219:

1. Open Motion Scope and configure it to record a measured position, then capture some data.
2. Save the Motion Scope pane using File -> Save As.
3. Close the view using File -> Close.

4. Load the previously saved pane using File -> Open.

897 Motion Scope bit masking does not work with long data types

Bit masking and normalize works with unsigned long data types, but when selecting "long" types, the bit masking and normalize features are disabled. But, if the masking and normalize is configured, and the data type is later changed from unsigned to long, the trace properties screen is grayed out, but the masking/normalizing is applied to the data.

916 Erroneous error when importing data

The following error message will sometimes be displayed when importing pane data:

"No Traces have been chosen and allocated for data"

The text of the error appears to be garbled. The error occurs when the number of columns in the imported data does not match the number of traces for which memory has been allocated.

923 [Dup. of 922] Cannot open .PAN when banding was enabled

If banding is enabled and the pane is saved to a file, including data, the resulting file cannot be opened by Motion Scope.

When opened as "read-only," the following error is displayed:

"No Traces have been chosen and allocated for data"

When opened as writeable, the following error is displayed:

"Data does not match expected number of traces"

938 Full out goes full in

When the full out button is pressed, the time axis should zoom all the way into a 1 msec wide window. It should set the XRange to MaxBuf.

940 Motion Scope mask disabled for long

When configuring a new trace in Motion Scope, if you select a data type of long the Bit masking options is disabled.

Issue Type: **MI (Minor Improvement)**

Number Name

473 Dialog boxes missing ToolTips

None of the dialog boxes display ToolTips.

662 Parameter precision (number of digits to right of decimal point) for X and Y axis labels.

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.