



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: 430B1
 MPI Library Version: 20021212.1.9

Revised 10Apr2003
 DCR 695

1 Introduction

Welcome to the latest release of Motion Engineering's MPI/XMP Firmware and Motion Programming Interface Library. This distribution has been prepared for Windows® NT 4.0, Windows® 2000 and Windows® 95/98. 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	430B1	430A5
MPI Library	20021212.1.9	20021212.1.8
Motion Console	03.38.22	03.38.22
Motion Scope	01.21.05	01.21.04



ATTENTION!



This release contains a new Rincon FPGA image, version 1.40. See section 4.0, issue MPI 1071 for details. The Rincon FPGA is responsible for SynqNet 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 © (10Apr2003) 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.



INTRODUCING MEI'S NEW and IMPROVED 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)
- 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 depends on controller/network load. The minimum sample rate is 1kHz.
- Cable Length - Network cable lengths must be less than 10 meters. In future releases, the network timing calculations will compensate for longer cable lengths.
- Software configurable data packets - The MPI automatically configures the packets during network initialization based on the node type. In future releases, packets will be user configurable.
- String 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.

The following features are NOT supported in this release:

- Network fault recovery
- 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.

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-III
- 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 `mpiControlInit(...)` 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 `meiControlTopologyToFlash(...)`, 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 `meiControlTopologyToFlash(...)`. 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 feature set is very similar to the older Analog controller based capture. The capture objects are now associated with motor objects. The motor object now supports two encoder interfaces. Initially, each encoder interface supports two capture blocks. In the future, the number of capture blocks per encoder will be variable depending on the FPGA logic.

`mpiCaptureCreate(...)` has new parameters to create a capture object based on a motor handle, an encoder (primary or secondary), and a capture number (0 or 1).

`MPICaptureConfig{...}` has been changed to support configurable capture trigger sources, edge, 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. And the state of each output (in an `IoAbort` condition) can be configured through the `MEIMotorIoConfig{...}` structure with `mpiMotorConfigGet/Set(...)`.

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.

The **sqNodeFlash.exe** utility has been added to support FPGA and drive firmware download. Documentation for this utility can be found at <http://support.motioneng.com>.

A new utility, **sqNodeMemory.exe** has been added to support FPGA and drive firmware download. Documentation for this utility can be found at <http://support.motioneng.com>.

1.1 System Requirements

1.1.1 Operating System

The MPI release is built to operate on Windows® NT 4.0, Windows® 2000, or Windows® 95/98.

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 an InstallShield for Windows Installer program before, the MEI Install Shield will need to install InstallShield installer files before actually installing the MDK. You will have to **reboot** your system after these files are installed. Please shut down all programs before running the InstallShield for the first time. For first-time installations of an MEI controller and accompanying software, please see the *QuickStart Guide* located on the distribution CD.



WARNING! If you are upgrading from a previous MPI/XMP software release, **you will need to remove or archive all previous releases.** This will prevent any conflicts between old and new files. To remove the previous MPI/XMP software release, select **Start -> Control Panel -> Add/Remove Programs**. Select the **MPI/XMP Development Toolkit** entry and click on the **Add/Remove** button.

NOTE: The MPI/XMP software release can also be removed by running the MDK InstallShield and choosing the remove option.

For more information on installing the MPI/XMP software release, please see the *QuickStart Guide*.

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

- device driver (meixmp.sys for WinNT, meixmp.vxd for Win95)
- firmware
- MPI dynamic link library
- utilities
- sample applications


To install the MPI/XMP software release, insert the MDK CD-ROM. The set-up InstallShield will be launched automatically. Follow the InstallShield instructions. The InstallShield will take care of installing the DLL and will also set the PATH environment variable to XMP\bin\WinNT for WinNT and Win2000, or XMP\bin\Windows for Win95/98 under the default installation directory (**C:\MEI**).

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

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

2 General Changes / 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 20021212.1.9

	New Version	Previous Version
Firmware	430B1	430A5
MPI Library	20021212.1.9	20021212.1.8

- There were no general changes or new features in the 20021212.1.9 release.

Version 20021212.1.8

	New Version	Previous Version
Firmware	430A5	430A5
MPI Library	20021212.1.8	20021212.1.7

2.1 Support for the RMB-10V2-SynqNet

MPI 1098

The 20021212.1.8 software release contains support for the RMB-10V2-SynqNet. But, there are some limitations. This release package does not include an FPGA image (C0FE0029) for the new RMB-10V2-SynqNet hardware. Therefore, if you are considering using this release with an RMB-10V2-SynqNet, please contact MEI. Future software releases will include FPGA images for the RMB-10V2-SynqNet.

Version 20021212.1.7

	New Version	Previous Version
Firmware	430A5	430A5
MPI Library	20021212.1.7	20021212.1.6

2.2 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 was added. The CDDivNConfig{...} structure in kollmorgen_ch.h was expanded to support this new feature. The SqN-odeLib support is available in MPI version 20021212.1.7. For more information about Kollmorgen drive configurations, please consult the Kollmorgen drive documentation.

Version 20021212.1.6

	New Version	Previous Version
Firmware	430A5	430A5
MPI Library	20021212.1.6	20021212.1.5

- There were no general changes or new features in the 20021212.1.6 release.

Version 20021212.1.5

	New Version	Previous Version
Firmware	430A5	430A5
MPI Library	20021212.1.5	20021212.1.4

2.3 New Kollmorgen CD drive configurations

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 icon; /* 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 initialization. */
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 thermtype; /* Motor temperature sensor type. */
long izero; /* Sets the ZERO mode current. */

} CDDriveConfig;

```

Version 20021212.1.4

	New Version	Previous Version
Firmware	430A5	430A5
MPI Library	20021212.1.4	20021212.1.3

2.4 Add 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,
    MPIMotorEncoderFaultIILL_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;

```

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.

2.5 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;

```

Version 20021212.1.3

	New Version	Previous Version
Firmware	430A5	430A4
MPI Library	20021212.1.3	20021212.1.1

- There were no general changes or new features in the 20021212.1.3 release.

Version 20021212.1.1

	New Version	Previous Version
Firmware	430A4	430A4
MPI Library	20021212.1.1	20021212.1

- There were no general changes or new features in the 20021212.1.1 release.

Version 20021212.1

	New Version	Previous Version
Firmware	430A4	430A2
MPI Library	20021212.1	20021212

- There were no general changes or new features in the 20021212.1 release.

Version 20021212

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

2.6 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.

2.7 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(...)`.

2.8 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.

2.9 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.

2.10 Motor Capture Information

MPI 994

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

2.11 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.

2.12 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.

2.13 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.

2.14 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.

2.15 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

2.16 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.

2.17 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.

2.18 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 processing 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).

2.19 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

2.20 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).

2.21 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.

2.22 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

2.23 Increased User Limits

MPI 956

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

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

MPI 955

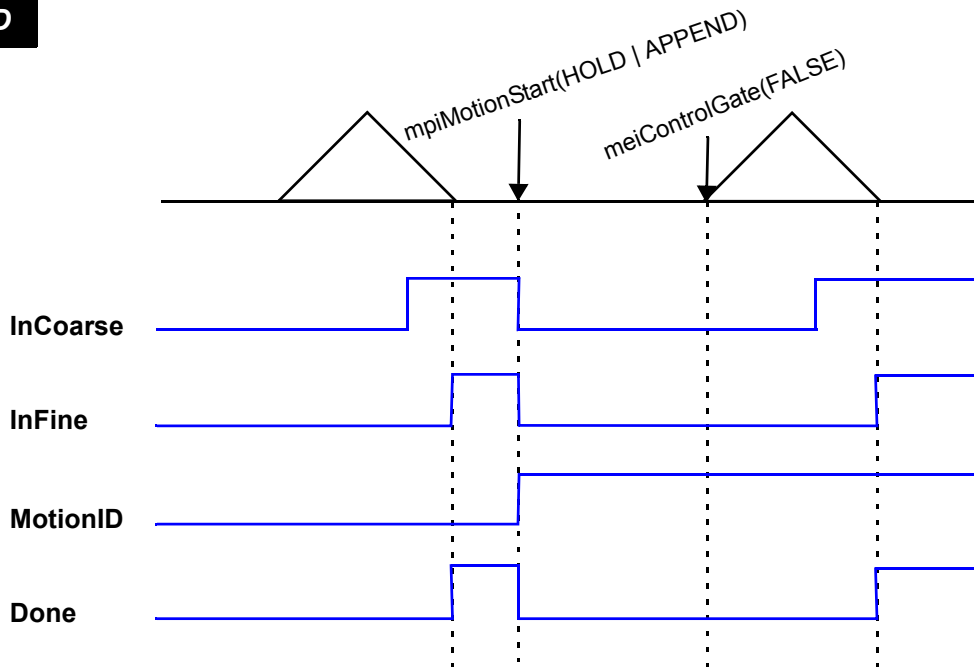
A new "pending" state was added to the firmware motion state machine and the frame sequence was also changed. Previously, the frame sequence was: start frame, hold frame, and then accel frame. The new sequence is: start frame, hold frame, id frame, and then accel frame. The new start frame will trigger 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.

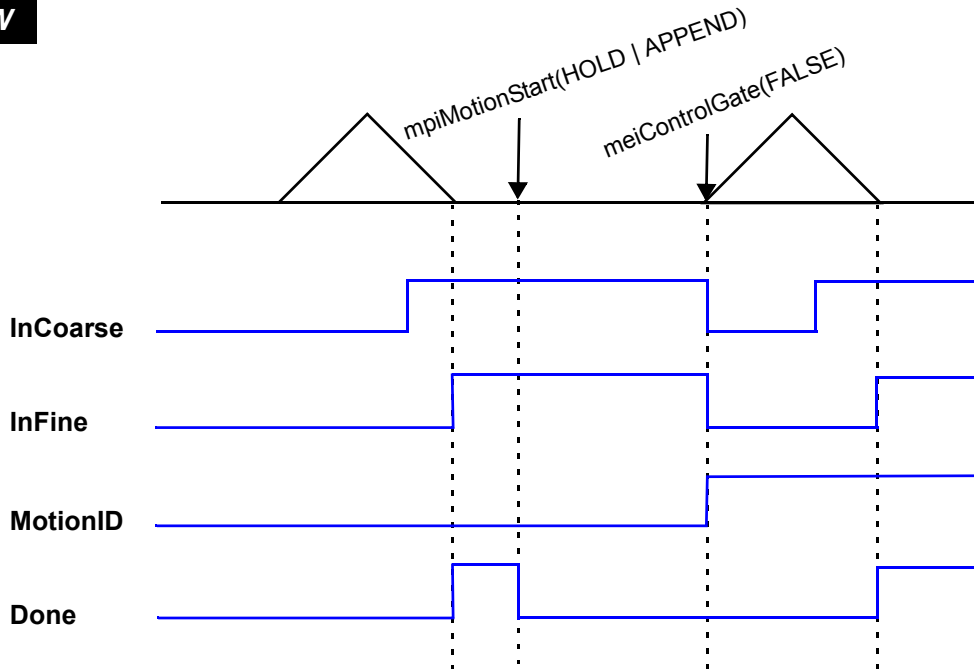
The MotionDone status will be cleared before updating the MotionID.

Please see the diagrams below.

OLD



NEW



2.25 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.

2.26 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;

```

2.27 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(...)`.

2.28 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
meiFilterPostFilterSet
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.

2.29 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>.

2.30 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.

2.31 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);
```

2.32 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.

2.33 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.

2.34 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.

2.35 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

2.36 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>.

2.37 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.

2.38 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.



2.39 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.



2.40 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).

2.41 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

2.42 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.

2.43 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 {
    MPIMotorBrakeModemode;
    floatenableDelay;
    floatdisableDelay;
} MPIMotorBrake;
```

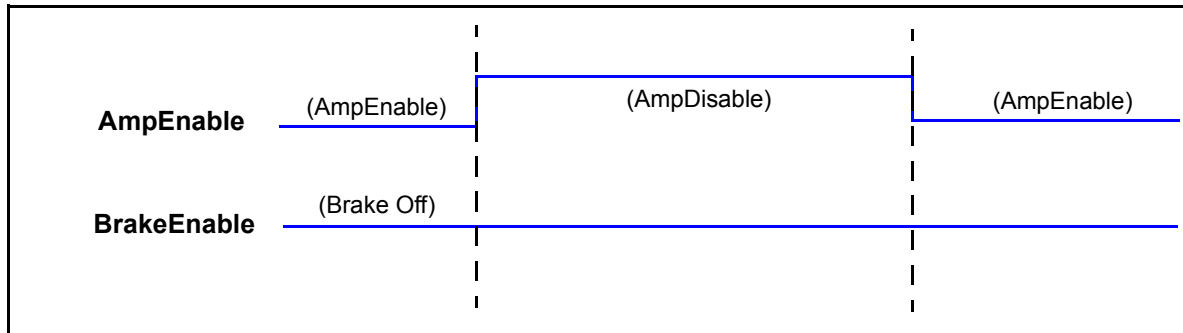
Description of Members

- **mode** - set to MPIMotorBrakeModeNONE for no brake, set to MPIMotorBrakeModeDELAY for use brake feature specified delays.

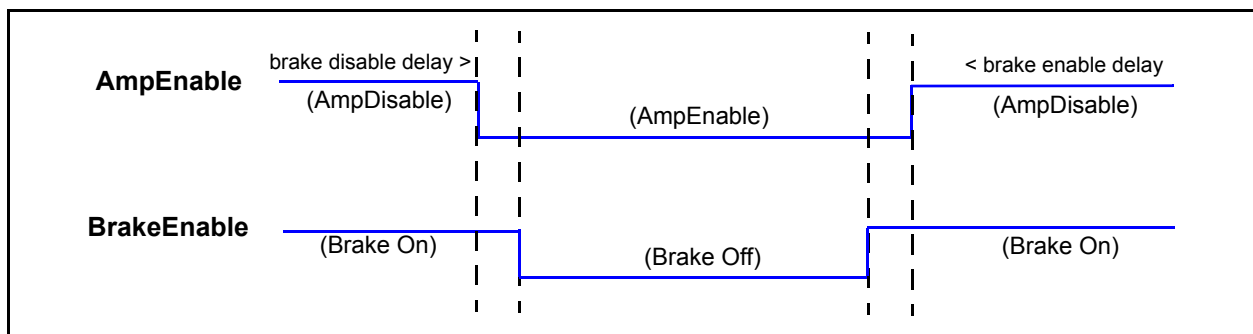
enableDelay - specifies the delay (seconds) between when the brake is active and the amp enable is disabled.

disableDelay - specifies the delay (seconds) between when the amp enable is enabled and the brake is inactive.

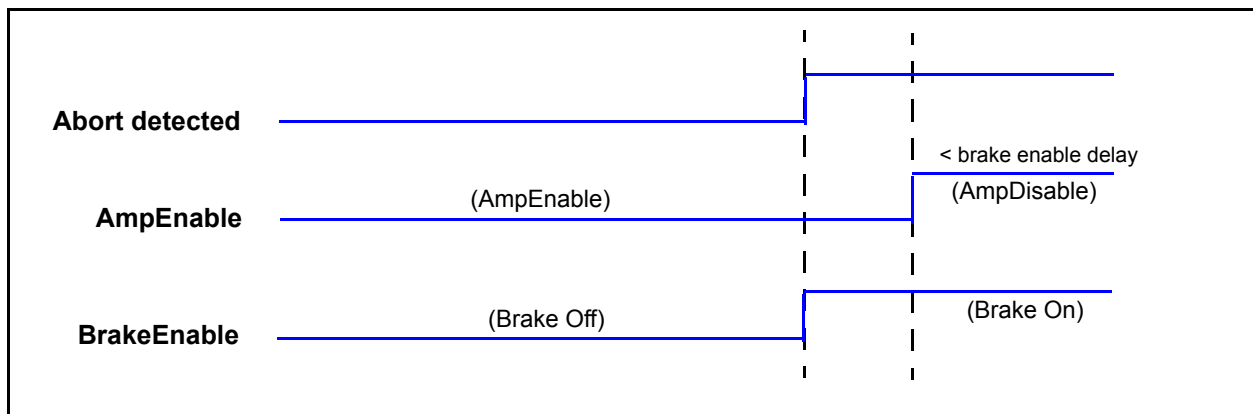
The following diagrams further explain how the Brake Enable/Disable Delay feature works:



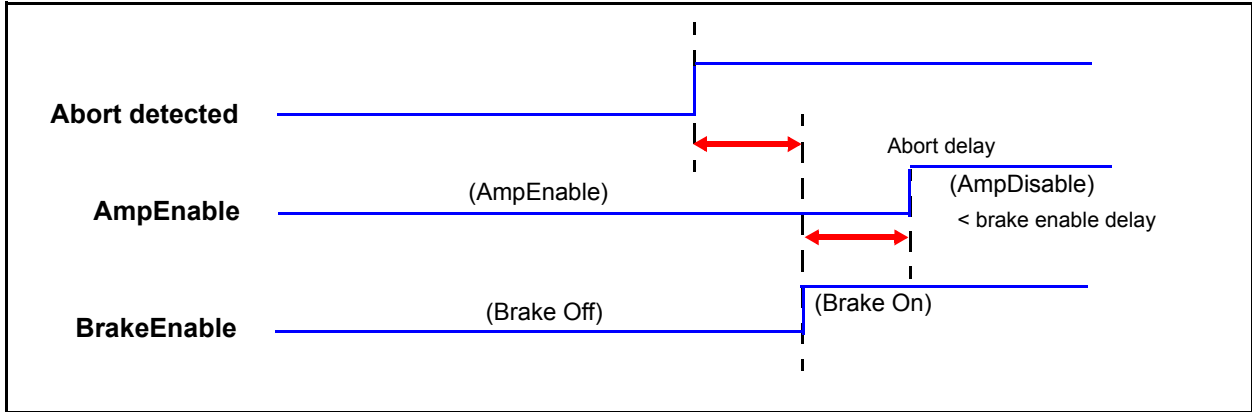
Case 1. Amp On/Off with NO Brake



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
20021212.1.9	20021212.1.8

4 Fixed Bugs of MPI/MEI Libraries:

Version 20021212.1.9

	New Version	Previous Version
Firmware	430B1	430A5
MPI Library	20021212.1.9	20021212.1.8

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 20021212.1.8

	New Version	Previous Version
Firmware	430A5	430A5
MPI Library	20021212.1.8	20021212.1.7

Glentek Omega Drive Initialization Problem **MPI 1097**

In previous versions, the Glentek Omega drive initialization would fail making it impossible to enable the amplifier. This problem was caused by a sequencing issue with the drive reset, PLL clock configuration, and an improper value for the PLL period. This problem was fixed in version 20021212.1.8.

Version 20021212.1.7

	New Version	Previous Version
Firmware	430A5	430A5
MPI Library	20021212.1.7	20021212.1.6

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.

Kollmorgen CD Drive Feedback Source **MPI 1087**

In version 20021212.1.5, the default feedback source was configured for the drive's feedback. The default feedback source was changed in version 20021212.1.7 to the quadrature encoder feedback. In future releases, this configuration will be configurable via the motor object.

Config.exe Utility Error

MPI 1086

In version 20021212.1.5, typing "config set file.txt" returned the following error message, "Exiting: motor: Motor: hardware not found (0xb04)." This was caused by hardware not being available for the configuration. This problem was corrected by changing the config.exe utility to ignore hardware "not found" errors.

Version 20021212.1.6

	New Version	Previous Version
Firmware	430A5	430A5
MPI Library	20021212.1.6	20021212.1.5

- There were no bug fixes in the 20021212.1.6 release.

Version 20021212.1.5

	New Version	Previous Version
Firmware	430A5	430A5
MPI Library	20021212.1.5	20021212.1.4

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.

Version 20021212.1.4

	New Version	Previous Version
Firmware	430A5	430A5
MPI Library	20021212.1.4	20021212.1.3

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.

A missing packet causes the network to shut down

MPI 1071

In Rincon FPGA image versions 1.2.9 through 1.3.5, a bug existed where a single missing packet would shut down the network. The network is designed to tolerate one packet error and shut down when two

packet errors are received. But in previous versions, two packet errors were generated when only one missing packet error was received.

NOTE: Corrupted packets were correctly recognized as a single error; only missing packets caused the fault.

This bug was corrected in the new Rincon FPGA image 1.4.0. The upgrade to Rincon 1.4.0 is recommended for all existing SynqNet Phase1 and Phase2 systems. This upgrade will allow the network to tolerate any single packet error, and shut down when two consecutive packet errors are received. Without this upgrade, a single missing packet error could potentially shut down the entire network.

Version 20021212.1.3

	New Version	Previous Version
Firmware	430A5	430A4
MPI Library	20021212.1.3	20021212.1.1

Timeout errors with 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 performing an `MPIMotionActionRESET`, which was preceded by a long PT/PVT move with more than 128 points which were stopped. Performing an `MPIMotionActionSTOP` would have stopped the PT/PVT move. Calling `MPIMotionActionRESET` action would not completely clean-up the stopped PVT move, thereby leaving the MPI in an unstable condition. This problem has been fixed in the 20021212.1.3, as well as in any releases after 20030207.

Version 20021212.1.1

	New Version	Previous Version
Firmware	430A4	430A4
MPI Library	20021212.1.1	20021212.1

Position shift after `mpiMotorConfigSet(...)`

MPI 1063

A bug existed in earlier releases where a position shift was experienced after calling `mpiMotorConfigSet(...)` where `external != NULL`. This occurred ONLY when using a non-zero `MEIMotorConfig.ReverseModulo.Value` parameter. This has been fixed in the 20021212.1.1 release.

Version 20021212.1

	New Version	Previous Version
Firmware	430A4	430A2
MPI Library	20021212.1	20021212

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.

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:

meiControlTopologyToFlash(...) Not Supported

MPI 1003

In version 20021014, meiControlTopologyToFlash(...) was disabled due to internal problems. This method will return an "Unsupported" error. This will be fixed in a future release.

Encoder termination cannot be disabled

MPI 957

The RMB-10v motor encoder termination cannot be disabled. The MPI and FPGA does not support motor encoder termination enable/disable. The default configuration is enabled. This feature will be added at a later date.

DRate limited in firmware

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.

WinNT Driver Invalid Board Number Bug

MPI 568

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

Firmware support for jogging

MPI 554

The MPI has a motion type for jogging (MPIMotionTypeJOG), but presently the firmware does not support it.

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.

mpiCommandPositionSet(...) failure **461**

mpiCommandPositionSet(...) does not set the command position

- when the axis is in a MOVING or ERROR state
- after a STOP, E_STOP, or ABORT event has occurred

The mpiCommandPositionSet(...) does not return an error code when called during these conditions.

NOTE: a call to mpiMotionAction(..., RESET) makes it possible to set the command position.

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.

Default Commutation OutputLevel is non-zero **433**

The default open-loop commutation output level is set to 8192, representing 2.5 V at the DACs. This level can damage a motor winding and drive. Since a drive gain and motor safe current levels are unknown, the correct value should be 0.0, so that the open-loop current level may be intentionally set by the user.

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	03.38.22	03.38.22

Modified in Version: **03.38.21**

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

Number **Name**

908 [Dup. of 906] **Pause Scrolling Errors**

A "Pause" button has been added to the Library Function Errors window. Clicking 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. Clicking the button 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 read**

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 has 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*. Motion Console 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 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 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.

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: **03.38.10**

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)**

<u>Number</u>	<u>Name</u>
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)**

<u>Number</u>	<u>Name</u>
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)**

<u>Number</u>	<u>Name</u>
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**

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

<u>Number</u>	<u>Name</u>
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)**

<u>Number</u>	<u>Name</u>
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**

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**

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	01.21.05	01.21.04

Modified in Version: **01.21.05**

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

Number **Name**

931 **Floating Point Data is Rounded to 3 Digits after the Decimal Point**

When a pane is saved or exported, floating point data is rounded to 3 digits after the decimal point.

7.3 Open Issues: Motion Console

Issue Type: CR (Change Request)

Number **Name**

924 **Alternative Add Controller Dialogue**

Having multiple tab pages in the Add Controller dialog box is confusing to users. Some think that they have to fill in information on every tab. In reality, the tab control is used solely to select the controller type.

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.

657 **"(Not Available)" listed as an option in pull down menu**

In the Motor Summary window, under the I/O configuration tab, all XCVR Config pull-down menus list "(Not Available)" as an option.

741 **User In bit not reported**

The User In bit is not reported, when bit is toggled.

761 **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.

847 **Flickering could appear on several status windows**

Some flickering could appear on the Axis, Motion, and Motor Status pages because of a bug in how event status flags are compared.

881 **ASynq mode makes Motion Console use 100% of CPU time**

When SynqNet is in the ASynq mode, **Motion Console** takes 100% of the CPU time. This makes the system very sluggish.

887 Object settings not saved prior to opening a new .INI file

Changes to object attributes that are stored in the .INI file are not saved when another .INI file is opened or created.

913 Column width is forgotten after a Restart of Motion Console

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, which will allow the column width to be restored to the users preference when the summary window is reopened.

914 Motion Console crashes on phase 2 to phase 1 conversion

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

Number Name

739 Add more detail to tooltips for disabled controller buttons

When a button on the Controller Summary is disabled because the controller is not initialized, some clues can be added to help the user rectify the situation.

740 Add greater detail to toolbar button tooltips concerning various modes of operation

The action that is executed when a toolbar button is clicked can sometimes be modified by holding down the Ctrl or Shift keys. The nature of this modified behavior should be described in detail in the tooltip for each button.

775 Remove Broken Wire and Illegal State for Motor I/O page

Broken Wire and Illegal State statuses are displayed both in the Motor I/O and Status tab pages. They are somewhat inappropriate for the I/O window because there are no I/O pins associated with them.

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

Motor encoder fault trigger conditions will change in the MPI, so the interface in Motion Console will have to change as well.

7.4 Open Issues: Motion Scope

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

Number **Name**

833 **More precision in Motion Scope's 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.

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 :] Motion Scope "Save Pane" Corrupts timebase.

897 Motion Scope bit masking doesn't 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 are configured, then 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 doesn't match the number of traces for which memory has been allocated.

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 a *read-only*, the following error is displayed:

"No Traces have been chosen and allocated for data"

When opened as *writable*, the following error is displayed:

"Data does not match expected number of traces"

938 Full out goes full in

940 Motion Scope mask disabled for long

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

Number Name

473 Dialog boxes missing ToolTips

None of the dialog boxes display ToolTips.

662 Support for parameter precision (number of digits to right of decimal point) for X and Y axis

Add parameters that provide the ability to modify the precision (number of digits to right of decimal point) for X and Y axis data labels. Add a separate parameter for the X-axis and parameters per Trace on the Y-axis.

663 Groups to be supported in File Import input FFT files.

Groups to be supported in File Import input FFT files.