Release Notes for MP2000iec on MP2300 Series Hardware

Release 1.1.3 Build 7

Cumulative for changes from 1.1.2 Build 5

Yaskawa Electric America, Inc.

Important!

Firmware 1.1.1 required to upgrade to Firmware 1.1.3. Upgrading from any prior firmware version requires first upgrading to Firmware 1.1.1 and then upgrading to Firmware 1.1.3.

1 Supported Function Blocks

The following list contains the function blocks supported in this release:

- MC_AbortTrigger
- MC_FinishHoming
- MC_GearIn*
- MC_GearInPos*
- MC_GearOut*
- MC_MoveAbsolute
- MC_MoveRelative
- MC_MoveSuperimposed
- MC_MoveVelocity
- MC_Power*
- MC_ReadActualPosition
- MC_ReadActualTorque
- MC_ReadActualVelocity
- MC_ReadAxisError*
- MC_ReadParameter*
- MC_ReadBoolParameter
- MC_ReadStatus*
- MC_Reset*
- MC_SetPosition*
- MC_StepRefPulse*
- MC_StepLimitSwitch*
- MC_Stop
- MC_TorqueControl
- MC_TouchProbe
- MC_WriteBoolParameter
- MC_WriteParameter
- Y_CamFileSelect
- Y_CamIn*
- Y_CamOut*
- Y_CamScale
- Y_CamShift
- Y_CamStructSelect

- Y_ClearAlarms
- Y_HoldPosition**
- Y_ReadAlarm
- Y_ReadCamTable*
- Y_ReadDriveParameter
- Y_ResetAbsoluteEncoder*
- Y_ResetMechatrolink*
- Y_SlaveOffset
- Y_VerifyParameters*
- Y_WriteCamTable
- Y_WriteDriveParamter
- Y_WriteParameters*

* Indicates that this function block has a known issue.

** Indicates that the function block has been deprecated and will be removed in a future release.

1.1 Unsupported Function Block Inputs and Outputs

The following function block inputs and outputs are not supported and are reserved for future use:

- MC_MoveAbsolute.Jerk
- MC_MoveRelative.Jerk
- MC_MoveAdditive.Jerk
- MC_MoveSuperImposed.Jerk
- MC_MoveVelocity.Jerk
- MC_Stop.Jerk
- MC_Stop.BufferMode (assumed BufferMode is *aborting*)
- MC_Power.BufferMode
- MC_Power.Enable_Positive
- MC_Power.Enable_Negative
- MC_ReadStatus.Busy (always FALSE)
- MC_ReadAxisError.Busy (always FALSE)
- MC_Read[Bool]Parameter.Busy (always FALSE)
- MC_TorqueControl.Acceleration
- MC_TorqueControl.Deceleration
- MC_TorqueControl.Jerk
- MC_Write[Bool]Parameter.Busy (always FALSE)
- MC_ReadActualPosition (always FALSE)
- MC_GearIn.Jerk
- MC_TouchProbe.WindowOnly
- MC_TouchProbe.FirstPosition
- MC_TouchProbe.LastPosition
- MC_SetPosition.Busy (always FALSE)
- MC_ReadActualVelocity.Busy (always FALSE)
- MC_ReadActualTorque.Busy (always FALSE)
- MC_GearInPos.Jerk

2 Important changes from 1.1.2 Release

2.1 Function Blocks

2.1.1 Bug Fixes

- Y_ResetMechatrolink anomilies (SCR 3420)
  Details: Previously, MC_Power would get kernel error 61713 if enabled while Y_ResetMechatrolink is executed. Now, Y_ResetMechtrolink will set Error to TRUE and ErrorID to 0xb116 (ServoNetResetProhibited) if an axis is enabled.
- Gearing cannot be re-enabled after slave hits limit switch (SCR 3496)
  Details: Internally, the gear move was being marked as completed rather than aborted, so all the outputs were off because there is not a Done output that maps to the completed state. However, conditional execution requires at least on output to be active while Execute is high, so the function block missed the falling edge of Execute. The gear move is now aborted properly.
- MC_TouchProbe with LIO card causes critical exception (SCR 3536)
  Details: MC_TouchProbe tried to send a Latch off aux command to the servo node interface, but in this case the external axis doesn't have a node, resulting in dereferencing a NULL pointer. Now, MC_TouchProbe and MC_AbortTrigger use the AxisModule interface instead.
- Need to improve MC_TorqueControl.Ramp functionality (SCR 3537)
  Details: Previously, the initial torque for MC_TorqueControl is either the previous command torque or zero. There are several problems with this implementation. First, while in position mode, the drive may have some initial holding torque, for example it may be compensating for gravity for a Z axis. Starting the torque at zero would be wrong because the axis would drop. Second, while in velocity limited torque mode, the drive will not output the command torque if it's at the velocity limit. A subsequent MC_TorqueControl control might abruptly change the velocity limit, which would abruptly change the actual torque, negating the desired effects of the TorqueRamp input. Now, the initial torque is always the feedback torque.
- External encoder based gearing motion irregularity (SCR 3538)
  Details: The previous procedure for unwrapping the command position used the wrap interval used to calculate the wrapped feedback position, which assumed that no command position will be more than 1/2 the wrap range from the current feedback position. This assumption was incorrect because the combination of the following error and the Mechatrolink communication delay at high speeds might be greater than 1/2 the wrap range.

  The new implementation decouples the wrap interval for the command position and feedback position. The command position is tracked independently, incrementing or decrementing the wrap interval every time the wrapped command position jumps by more than 1/2 the wrap range. When the drive is in torque control, velocity control or hold position mode, the command position and wrap interval track the feedback position and interval. (Previously, the command position tracked the feedback position in these control modes.)

This larger issue was detected because the command velocity for external encoders was always zero due to an initialization bug. Since the command velocity was zero, this configuration had large following errors.

- Non periodic slave runs two cycles if virtual master is run in reverse direction (SCR 3543)
  Details: Previously, the detection for completing a non periodic cam was wrong for negative master directions. As a result, running a non periodic cam with master in the reverse direction caused the slave to run two cycles.

- CamShiftRemainig (Parameter Number 1513) does not update (SCR 3546)
  Details: This parameter was always zero; now it gives the correct value.
- Reconsider POT/NOT alarms (SCR 3548)
  Details: Previously, when an axis reaches a positive or negative over travel limit, the axis would have a controller alarm and the function block would generate an error. The alarm generation was removed alarm generation when POT/NOT is reached, but the function block will still have an error.
- Y_CamShift Y_AdjustMode#WithinRange should account for PhaseShift (SCR 3554)
  Details: Previously, Y_CamShift calculated the master distance over which to shift the cam as: Y_CamShift.EndPosition - MasterPosition. This was not correct because Y_CamShift will not complete with the Cam Master Shifted Position Cyclic -- PN 1502 equal to the final position. Now, the master distance over which to shift is: Y_CamShift.EndPosition - MasterPosition - Y_CamShift.PhaseShift.
- Download Changes causes function block execute to retrigger (SCR 3555)
  Details: Previously, the function block's internal state, including the previous value of execute, was reset if it was modified via download changes. As a result, the function block falsely detected a new rising edge of execute. The Function block's internal state is no longer reset with download changes.
- Y_CamIn internal state not cleared after COLD start (SCR 3563)
  Details: Y_CamIn did not clear all of its internal state (specifically the last value of execute), when the program was stopped. Y_CamIn now clears the last value of execute when the program stops.
- CamIn CommandAbort stays high on cold start (SCR 3570)
  Details: Both Y_CamIn and MC_GearInPos were not properly reset on stop.
- MC_MoveVelocity shows AtVelocity and Error (SCR 3574)
  Details: The AtVelocity output was not set to FALSE when an error occurred.

2.1.2 New issues

- MC_TorqueControl after a steep MC_Stop starts with a twitch (SCR 3540)
  Details: When MC_TorqueControl is started after a steep decelerated MC_Stop, the axis moves in the opposite direction first before starting to move in the commanded direction. The sequence to reproduce this is problem is:
  1) MC_TorqueControl
  2) MC_Stop with a fast decel.
  3) MC_MoveAbsolute back to start position.
  4) MC_TorqueControl.

This appears to be a drive issue related to switching control modes. Using a Mechatrolink packet analyzer, it was confirmed that the controller is always sending out the correct command torque.
Mitigation: This problem does not occur with lower stop decelerations. If higher decelerations are needed the user can increase Pn80D, Pn80E, and Pn80F Additionally, the user can use a MC_TorqueControl function block to ramp down the velocity limit while maintaining the command torque, effectively stopping the axis in a controlled manner.

- ErrorID 4371 on MC_Power for drive power issue does not go off if enable is low (SCR 3542)
  Details: If MC_Power.Enable=TRUE and the drive can not be enabled, then MC_Power.ErrorID = 4371. However, setting MC_Power.Enable=FALSE, does not clear the error.
  Mitigation: If MC_Power is subsequently enabled, then the ErrorID is cleared, and normal motion will be possible.
- Y_CamShift gets ErrorID 4370 even though axis can move (SCR 3545)
  Details: Sigma5 linear axis had a A.91 (overload warning) on it. Axis could enable and home, but the Y_CamShift reports 4370.
  Mitigation: This issue is being investigated.
- MC_Stop and MC_TorqueControl (SCR 3565)
  Details: When MC_Stop occurs after MC_TorqueControl, both the command torque and the velocity limit are ramped down to zero according to MC_Stop.Deceleration. However, as the command torque is ramped down, the drive may not output the torque required to stop, so the axis might still be moving after the deceleration phase. MC_Stop then uses a HOLD position command to transition the axis back to position mode. If the axis is still moving, then drive will stop the axis according to Pn80D, Pn80E, and Pn80F.
  Mitigation: There are two options for handling this issue: 1) the user can increase the MC_Stop.Deceleration and Pn80D, Pn80E, and Pn80F so that the axis will stop within the desired range, or 2) the user can smoothly decelerate to a stop by first using a second MC_TorqueControl that leaves the command torque unchanged but ramps down the velocity limit to zero and then using MC_Stop to place the axis in the stop state.

2.2 EtherNet/IP

2.2.1 Bug fixes

- Ethernet/IP clock rollover causes disconnects every 75 minutes (SCR #3598)
  Details: The system timestamp clock was being used to determine the elapsed ticks since the Ethernet/IP and Modbus/TCP last ran. However, when the time stamp clock rolled over, the elapsed time calculation gave an erroneous result causing the controller to drop the connection and then reconnect. Now, the system timestamp is no longer being used since the Ethernet I/P and Modbus TCP scan tasks are triggered off of the Mechatrolink update rate and not prone to drifting.

2.3 Mechatrolink

2.3.1 New features

- Make feed forward velocity optional (SCR 3531)
  Details: For very tightly tuned systems with low inertias, the step changes in controller feed forward velocity at the Mechatrolink update rate causes audible noise. For these situations, it's preferable for the drive to calculate and filter the feed forward velocity according to Pn109 and Pn10A, respectively, instead of using the controller feed forward velocities. Consequently, the controller feed forward velocity is now optional. It can be enabled or disabled via configuration files and changed at run time using MC_WriteBoolParameter with ParameterNumber=1310.

- Support Pn 812 Moving Average Filter (SCR 3566)
  Details: Pn 812 (Moving Average Filter) filters the command position calculated by the drive not feedforward velocity. The effect of Pn 812 can be seen by logging Position Reference Speed in Sigma Win. With Pn 812=0, Position Reference Speed is a stair step at the Mlink update. Pn 812 = Mlink update, the drive interpolates and the stair step is now at 4 kHz, drive position loop frequency. Note, there is still a stair step seen because the data logging is being done at 8kHz.

  New configuration feature property: "1311: Drive Motion Command OPTION ACCFIL Value (0,1,2)"
  New Parameter: 1311

  By default, for backward compatibility, it is off. Also, it makes no attempt to set Pn810, Pn811, or Pn812.

3 Known issues

3.1 Function Blocks

3.1.1 Bugs

- Y_CamFileSelect
  - Downloading application while executing Y_CamFileSelect crashes controller (SCR 3490)
    Details: When Y_CamFileSelect executes it spawns a new thread to load the cam file in the back ground. When a new program is downloaded, the original function block is destroyed, and the background thread will be executing deleted or partially deleted objects.
    Mitigation: Do not download a new program while Y_CamFileSelect is executing.
- MC_StepRefPulse and MC_StepRefLimit
  - Downloading application while executing MC_StepRefPulse and MC_StepRefLimit crashes controller (SCR 3499)

Details: When MC_StepRefPulse and MC_StepRefLimit execute, they spawns a new thread to perform the homing sequence. When a new program is downloaded, the original function block is destroyed, and the background thread will be executing deleted or partially deleted objects.
Mitigation: Do not download a new program while executing MC_StepRefPulse and MC_StepRefLimit.

- Y_CamShift
  - Y_CamShift gets ErrorID 4370 even though axis can move (SCR 3545)
    Details: Sigma5 linear axis had a A.91 (overload warning) on it. Axis could enable and home, but the Y_CamShift reports 4370.
    Mitigation: This issue is being investigated.

3.1.2 Usage Notes

- MC_GearOut
  - MC_GearOut holds current velocity even if not gearing. (SCR 2808)
    Details: For example, executing MC_GearOut while a MC_MoveAbsolute function block is active will abort the MC_MoveAbsolute function and hold the current velocity.
    Mitigation: Only call MC_GearOut when gearing. If holding the current velocity is not desired, then use MC_Stop.
- MC_Power
  - A 95 being issued when MC_Power disabled (SCR 2810, 3065)
    Mitigation: User programs can clear this alarm.
  - MC_Power shows Status High even after Mechatrolink is down (SCR 3493)
    Details: When the Mechatrolink cable is disconnected, the controller can not communicate with the drive, and it generates an controller watch dog alarm (0x23010001). The drives response to this same event it to generate an AE50 alarm and disable. Since communication has been lost, the controller does not detect that the drive has disabled, so MC_Power.Status is still high.
    Mitigation: The axis is in the ErrorStop state as it should be, so motion is prohibited, and MC_ReadAxisError shows the communication error. If the communication cable is inserted again, the MC_Power.Status will be FALSE..
  - ErrorID 4371 on MC_Power for drive power issue does not go off if enable is low (SCR 3542)
    Details: If MC_Power.Enable=TRUE and the drive can not be enabled, then MC_Power.ErrorID = 4371. However, setting MC_Power.Enable=FALSE, does not clear the error.
    Mitigation: If MC_Power is subsequently enabled, then the ErrorID is cleared, and normal motion will be possible.
- MC_ReadAxisError
  - Alarm does not match alarm shown on drive (SCR 2792)
    Mitigation: The drive may have multiple alarms, and one of these is returned by MC_ReadAxisError
- MC_ReadStatus (Axis State Machine):

- No transition from ErrorStop to Disabled when MC_Power.Enable=False. (SCR 2822)
  Mitigation: Technically this is not part of the PLCopen specification; the specification does not indicate any transitions to Disabled state.
- No transition from Disabled to ErrorStop when MC_Power.Enable=True and there is an error on the axis. (SCR 3450)
  Mitigation: Customers should use MC_ReadAxisError to determine when the axis has an error.

- MC_Reset
  - MC_Reset does not clear A.ED on Sigma II (SCR 2729)
    Details: A.ED alarm requires the servo network to be reset.
- MC_StepRefPulse & MC_StepLimitSwitch
  - MC_StepRefPulse behaves incorrectly at high command velocity (SCR 2879)
    Details: When the velocity is set at 50 rev/s the motor spins for several seconds before the Done output is TRUE. The issue is that the torque limited velocity move is immediately followed by a hold position, and the hold position takes several revolutions to stop the axis. Suggested maximum speed is 1 rev/s.
    Mitigation: This issue does not occur with slower velocities (less than 1 rev/s) which are more typical.
  - MC_StepLimitSwitch only supports one LimitSwitchMode: MC_EdgeOn (3131)
    Details: MC_StepLimitSwitch only works when detecting the rising edge of an input.
    Mitigation: Application use MC_EdgeOn only.
- MC_TorqueControl
  - MC_TorqueControl requires MC_Stop before using any other motion function block. (SCR 3051)
    Details: MC_TorqueControl can not be aborted by a 'position mode' motion block such as MC_MoveAbsolute.
    Mitigation: Changing control modes while moving has not been a requirement for a customer
  - MC_TorqueControl after a steep MC_Stop starts with a twitch (SCR 3540)
    Details: When MC_TorqueControl is started after a steep decelerated MC_Stop, the axis moves to the opposite direction first before starting to move in the commanded direction. The sequence to reproduce this is problem is:
    1) MC_TorqueControl
    2) MC_Stop with a fast decel.
    3) MC_MoveAbsolute back to start position.
    4) MC_TorqueControl.

    This appears to be a drive issue related to switching control modes. Using a Mechatrolink packet analyzer, it was confirmed that the controller is always sending out the correct command torque.
    Mitigation: This problem does not occur with lower stop decelerations. If higher decelerations are needed the user can increase Pn80D, Pn80E, and Pn80F Additionally, the user can use a MC_TorqueControl function block to ramp down the velocity limit while maintaining the command torque, effectively stopping the axis in a controlled manner.

- MC_Stop and MC_TorqueControl (SCR 3565)
  Details: When MC_Stop occurs after MC_TorqueControl, both the command torque and the velocity limit are ramped down to zero according to MC_Stop.Deceleration. However, as the command torque is ramped down, the drive may not output the torque required to stop, so the axis might still be moving after the deceleration phase. MC_Stop then use a HOLD position command to transition the axis back to position mode. If the axis is still moving, then drive will stop the axis according to Pn80D, Pn80E, and Pn80F.
  Mitigation: There are two options for handling this issue: 1) the user can increase the MC_Stop.Deceleration and Pn80D, Pn80E, and Pn80F so that the axis will stop within the desired range, or 2) the user can smoothly decelerate to a stop by first using a second MC_TorqueControl that leaves the command torque unchanged but ramps down the velocity limit to zero and then using MC_Stop to place the axis in the stop state.

- Y_CamFileSelect
  - Y_CamFileSelect becomes unresponsive (SCR 3393)
    Details: If the Execute input is toggled off and on while the function block is busy loading a file, then the function blocks output will never turn on.
    Mitigation: The Execute input on the Y_CamFileSelect block should be interlocked with the busy output so that the Execute input will not "see" a rising edge while the busy output is set.
- Y_CamIn
  - Y_CamIn does not detect if the engage window is too small (SCR 3356)
    Details: If the window is too small, Y_CamIn will not engage.
  - Y_CamIn engages after a cycle of engage position (SCR 3413)
    Details: When the axis crosses the engage position, the controller prepares the axis for cam, but the actual cam processing does not occur until one Mechatrolink cycle later, when all the axes are processed at the same time.
    Mitigation: To ensure a smooth transition into a caming, the user can use a "pick-up" cam table where the axis is idle with in the position window.
- Y_CamOut
  - No automatic adjustment if command position does not match disengage position (SCR 3358)
    Details: The slave is commanded to the disengage position with in one Mechatrolink cycle, perhaps causing too large a control effort for the drives.
    Mitigation: Applications should disengage in a portion of the profile while the slave is in a stopping portion of the profile or use a "drop off" cam table.
- Y_CamScale
  - CamScale with one way cam is broken (SCR 3461)
    Details: Using Y_CamScale with a one way cam can cause large changes in position since the cumulative cyclic offset is scaled too.
    Mitigation: Do not use Y_CamScale with one way cams.
- Y_CamStructSelect
  - Output Exclusivity of Y_CamStructSelect (SCR 3503)
    Details: If Y_CamStructSelect is busy when it's retriggered, both the Busy and Error outputs will be on. In this case, Y_CamStructSelect is still loading the

original structure, and when finished, then the Done output will be TRUE and both the Error and Busy will be off. The motivation for this implementation is to prevent the resource leak associated with loosing track of a CamID.
Mitigation: Retriggering Y_CamStructSelect while busy is a programming error, and Error output will come on.

- Y_VerifyParameters
  - Y_VerifyParamters has InvalidParameter set when Matches is TRUE (SCR 3504)
    Details: When Y_VerifyParamters.Matches=TRUE, Y_VerifyParamters.InvalidParameter is set to the last parameter.
    Mitigation: Error and Error ID outputs are not set, and Expected and Actual are both zero.

3.2 Modbus/TCP

- Function code 15: write multiple coils is not supported (SCR 2739)
  Details: Write multiple coils is not supported, so each coil has to be written using a separate transaction. As a result, writing multiple coils is not recommended. Use registers instead.

3.3 MECHATROLINK

- Controller reboots if gearing 16 axes with a 2ms Mechatrolink update rate (SCR 2919)
  Details: MC_GearIn.Execute=TRUE for all axes, the controller reboots.
  Mitigation: Increase the Mechatrolink update rate.

3.4 SGDV Servo Drive

- A94B alarm generated after Relative, Absolute, or Geared move. (SCR 3083)
  Details: With SGDV drives previously tuned with Sigma Win +, executing MC_MoveRelative, MC_MoveAbsolute or MC_GearIn(Pos) at roughly half the rated speed causes an A94B warning. This is caused by "model following control" being enable in Pn140.
  Mitigation: Set SGDV drive Pn140 to the default value of 0x0100.

3.5 Outputs

- Outputs can not be retained (SCR 3475)
  Details: In the IDE, the Global Variables table contains a column marked "Retain." Selecting a check box within this column causes the corresponding variable to be allocated in SRAM. However, for outputs, this feature is not working. At this point, it is not clear if the problem is with the IDE or the firmware.
  Mitigation: This issue is fairly easy to workaround by first storing the output value in a global variable that is retained and then copying the global variable to the output.