



Host Controller EtherNet/IP Communication Protocol User Manual



Allen-Bradley

by ROCKWELL AUTOMATION

User Manual

Original Instructions

Important User Information

Read this document and the documents listed in the additional resources section about installation, configuration, and operation of this equipment before you install, configure, operate, or maintain this product. Users are required to familiarize themselves with installation and wiring instructions in addition to requirements of all applicable codes, laws, and standards.

Activities including installation, adjustments, putting into service, use, assembly, disassembly, and maintenance are required to be carried out by suitably trained personnel in accordance with applicable code of practice.

If this equipment is used in a manner not specified by the manufacturer, the protection provided by the equipment may be impaired.

In no event will Rockwell Automation, Inc. be responsible or liable for indirect or consequential damages resulting from the use or application of this equipment.

The examples and diagrams in this manual are included solely for illustrative purposes. Because of the many variables and requirements associated with any particular installation, Rockwell Automation, Inc. cannot assume responsibility or liability for actual use based on the examples and diagrams.

No patent liability is assumed by Rockwell Automation, Inc. with respect to use of information, circuits, equipment, or software described in this manual.

Reproduction of the contents of this manual, in whole or in part, without written permission of Rockwell Automation, Inc., is prohibited.

The following icon may appear in the text of this document.



Identifies information that is useful and can help to make a process easier to do or easier to understand.



Rockwell Automation recognizes that some of the terms that are currently used in our industry and in this publication are not in alignment with the movement toward inclusive language in technology. We are proactively collaborating with industry peers to find alternatives to such terms and making changes to our products and content. Please excuse the use of such terms in our content while we implement these changes.

Additional Safety Information

Although every effort is made to keep this manual accurate and up-to-date, MagneMotion® and Rockwell Automation® assumes no responsibility for any errors, omissions, or inaccuracies. Information that is provided in this manual is subject to change without notice. Any sample code that is referenced in this manual or included with MagneMotion software is included for illustration only and is, therefore, unsupported.



ATTENTION: For additional safety notices and definitions, see the [Notes, Safety Notices, and Symbols](#) section.

Contents

About This Manual

Overview	9
Purpose	9
Prerequisites	9
Manual Conventions	10
Documentation	10
Notes, Safety Notices, and Symbols	10
Notes	10
Safety Notices	10
Additional Resources	11
Additional Documentation	14
Release Notes	14
Upgrade Procedure	14
Transport System Limits	14
File Maintenance	14
Backup Files	14
Creating Backup Files	14
Restoring from Backup Files	14

1 Introduction

Overview	15
EtherNet/IP Communication Protocol Overview	16
Transport System Components Overview	17
Transport System Software Overview	18
Utilities	19
File Types	19
Getting Started with the EtherNet/IP Communication Protocol	21

2 Transport System Control

Overview	25
Connecting to the Transport System	25
Running the Transport System	25
Transport System Reset, Startup, and Operation	27
Reset	28
Startup	30
Suspend	33

FastStop	35
Resume.....	37
Warm Reset.....	39
Traffic Light Control	41
Motor Controller Control.....	44
Propulsion Power Supply Control	46
Node Controller Digital I/O Control.....	47
Monitoring Transport System Status	48
High-Level Controller.....	48
Node Controllers.....	49
Node Status	50
Path Status.....	50
Vehicle Status	51
Motor Status.....	52
Station Status	54
Traffic Light Status	54
Motor Controller (Inverter) Status	55
Moving Path Node Status	55
Propulsion Power Supply Status.....	56
Node Controller Digital I/O Status	56
Moving Vehicles.....	57
Direction of Motion	57
Move to Position.....	59
Move to Station.....	60
Platooning	61
Shutting Down the Transport System.....	68
 3 Application Notes	
Overview.....	71
Vehicle Motion	71
Brick-wall Headway	71
Vehicle Positioning.....	71
Safe Stopping Distance.....	72
Thrust Limitations.....	73
Inserting and Removing Vehicles.....	74
Vehicle Insertion.....	74
Vehicle Removal.....	74
E-stops	75
Initiating a Host Controller E-stop.....	75
Recovering from a Host Controller E-stop	76
Interlocks	77
Initiating an Interlock.....	77
Recovering from an Interlock	78
Emergency Machine Off.....	78
Initiating an EMO	78
Recovering from an EMO.....	78
Platooning	78

Platooning Operations Overview	80
Creating Platoons	82
Uncoupling Platoons	87
Traffic Lights	91
System Monitoring	92
System Metrics	92
Behavior Patterns	93
Component Types	93
Metric Identifiers	93
Metric Instance	93
Accessing the System Monitoring Interface	94
System Monitoring API	94
Subscribe	94
Unsubscribe	95
Polling	96
Motor Types	97
MagneMover LITE	97
QuickStick (QuickStick 100 and QuickStick 150)	98
QuickStick High Thrust	99
Path Resets and Vehicle Deletion	100
Reset All Paths	100
Reset a Single Path	100
Delete a Vehicle	101
Issues with Partial System Resets and Vehicle Deletes	101
Managing Individual Path Resets	101
Node Type Descriptions and Usage	102
Simple Node	103
Relay Node	104
Terminus Node	105
Handshake for Vehicle Entry onto the Transport System	106
Handshake for Vehicle Exit from the Transport System	110
Example of Two Back to Back Terminus Nodes	113
Gateway Node	116
Moving Downstream Through a Gateway Node	118
Merge and Diverge Nodes	120
Merge Node	121
Diverge Node	123
Merge-Diverge Node	124
Overtravel Node	126
Moving Path Node	127
Path Startup Considerations	129
QuickStick Startup using Moving Paths	130
Path Linking/Unlinking	133
Moving Path Node Configuration	135
Moving Path Switch Configuration	142

4 Protocol Reference

Overview.....	145
Introduction.....	145
Example of Explicit Message Setup	146
Explicit Message Cyclic Send Glitch Workarounds.....	148
Host Controller to Node Controller Compatibility	148
Host Controller to HLC Explicit Message Request Definitions.....	149
MMI_create_traffic_light_cmd	151
MMI_delete_traffic_light_cmd	154
MMI_generic_node_command.....	157
MMI_mgmt_nc_restart_cmd	160
MMI_mgmt_nc_set_config_cmd	162
MMI_motor_inverter_command	165
MMI_mp_link_command.....	169
MMI_mp_unlink_command.....	173
MMI_node_controller_command.....	176
MMI_path_command	180
MMI_set_prop_power_state_cmd	184
MMI_set_traffic_light_cmd.....	188
MMI_sm_poll_command	191
MMI_sm_subscription_command	197
MMI_terminus_node_command	202
MMI_vehicle_command.....	205
MMI_vehicle_delete_order	208
MMI_vehicle_follow_order.....	211
MMI_vehicle_position_order	216
MMI_vehicle_station_order	221
HLC to Host Controller Status/Response Memory Tags.....	226
MMI_extended_hlc_status.....	228
MMI_extended_nc_status.....	231
MMI_extended_vehicle_status	234
MMI_heartbeat	242
MMI_HLC_status.....	244
MMI_mgmt_nc_cmd_status.....	246
MMI_motor_inverter_cmd_status	249
MMI_mp_command_status	251
MMI_mp_path_end_status	254
MMI_node_command_status	261
MMI_node_controller_cmd_status.....	263
MMI_node_controller_dio_status	265
MMI_node_controller_status.....	267
MMI_node_status	269
MMI_path_command_status	273
MMI_path_ml_faults_status.....	276
MMI_path_qs_faults_status.....	280
MMI_path_qs_ht_faults_status	287
MMI_path_status	295

MMI_propulsion_power_cmd_status	298
MMI_propulsion_power_status	301
MMI_qs_ht_sensor_map	303
MMI_sm_command_status	306
MMI_sm_metric_data	309
MMI_station_arrivals	313
MMI_traffic_light_cmd_status	315
MMI_traffic_light_status	318
MMI_vehicle_command_status	321
MMI_vehicle_order_status	324
MMI_vehicle_status	328
HLC Status Codes	333
Embedded udt_MMI_component_type	336
Source/Destination	336
Purpose	336
Support	336
UDT Field Descriptions	337
UDT Field Details	337
See Also	338
5 Troubleshooting	
Overview	339
EtherNet/IP Communications Troubleshooting	339
Motor Fault Troubleshooting	341
Node Fault Troubleshooting	387
Node Controller Fault Troubleshooting	388
Path Fault Troubleshooting	389
Vehicle Fault Troubleshooting	390
6 Communications Protocol	
Communications Format	393
EtherNet/IP	393
Physical Layer	393
Data Link Layer	393
Network Layer	394
Transport Layer	394
Application Layer	395
CIP Standard Objects	395
CIP Identity Object	396
Other CIP Objects	396
Index	397

Changes	
Overview.....	403
Rev. A	403
Rev. B	403
Rev. C	405
Rev. D	406
Rev. E.....	407

About This Manual

Overview

This section provides information about the use of this manual, including the manual structure, additional resources, format conventions, and safety conventions.

Purpose

This manual describes the communication protocol and messages that are used between the high-level controller application and a host controller that is equipped with an EtherNet/IP interface (typically an Allen-Bradley® ControlLogix® controller using Studio 5000 Logix Designer®). This manual also provides basic troubleshooting information. This manual is not intended to provide a design guide for the installation, or a reference for the operation of a transport system.

Use this manual in combination with the other manuals and documentation that accompany the transport system to design, install, configure, test, and operate a transport system.

NOTE: Depending on software version, user access privileges, and application configuration, certain operations that are described in this manual may not be available.

Prerequisites

The information and procedures that are provided in this manual assume the following:

- Familiarity with control system programming. Details on logic creation that uses the message and tag definitions that are provided in this manual for a production system are beyond the scope of this manual.
- The transport system has been installed and configured, and full documentation for the transport system is available.
- All personnel who configure, operate, or service the transport system are properly trained.

Manual Conventions

Numbers – All numbers are assumed to be decimal unless otherwise noted and use the US number format; that is, one thousand = 1,000.00. Non-decimal numbers (binary hexadecimal) are explicitly stated.

- Binary – Followed by ₂, for example, 1100 0001 0101₂, 1111 1111 1111 1111₂.
- Hexadecimal – Preceded by 0x, for example, 0xC15, 0xFFFF. AB₁₆, BA₁₆, and other similar annotations are also considered hexadecimal.

Measurements – All measurements are SI (International System of Units). The format for dual dimensions is SI_units [English_units]; for example, 250 mm [9.8 in].

Documentation

The documentation that is provided with the transport system includes this manual, which provides complete documentation for the use of the EtherNet/IP communication protocol for the transport system. Other manuals in the document set, which are listed in the [Additional Resources](#) section, support installation, configuration, and operation of the transport system.

The examples in this manual are included solely for illustrative purposes. Because of the many variables and requirements that are associated with any LSM system installation, Rockwell Automation cannot assume responsibility or liability for actual use that is based on these examples.

Notes, Safety Notices, and Symbols

Notes, Safety Notices, and Symbols that are used in this manual have specific meanings and formats. Examples of notes, the different types of safety notices and their general meanings, and symbols and their meanings are provided in this section. Adhere to all safety notices provided throughout this manual to help achieve safe installation and use.

Notes

Notes are set apart from other text and provide additional or explanatory information. The text for Notes is in standard type as shown in the following example.

NOTE: A note provides additional or explanatory information.

Safety Notices

Safety Notices are set apart from other text. The symbol on the left of the notice identifies the type of hazard. The text in the message panel identifies the hazard, methods to avoid the hazard, and the consequences of not avoiding the hazard.

Examples of the standard safety notices that are used in this manual are provided in this section. Each example includes a description of the hazard indicated. Labels may also be on or inside the equipment to provide specific precautions.

NOTICE Identifies an informational notice that indicates practices that are not related to personal injury that could result in equipment or property damage.

IMPORTANT Identifies information that is critical for the successful application and understanding of the product.



ATTENTION: Identifies information about practices or circumstances that can lead to personal injury or death, property damage, or economic loss. Attentions help to identify a hazard, avoid a hazard, and recognize the consequence.



SHOCK HAZARD: Identifies information about practices or circumstances where a severe shock hazard is present that could cause personal injury or death.



PINCH/CRUSH HAZARD: Identifies information about practices or circumstances where there are exposed parts that move, which could cause personal injury or death.

Additional Resources

Before configuring or running the transport system, consult the following documentation. You can view or download publications at rok.auto/literature.

Resource	Description
MagneMotion QuickStick and QuickStick HT Design Guide, publication MMI-RM001	This manual explains how to design and configure the track layout and transport system.
MagneMotion Glossary of Terms, publication MMI-RM003	This manual includes definitions of MagneMotion and industry terms.
MagneMotion System Configurator User Manual, publication MMI-UM046	This manual explains how to use the MagneMotion System Configurator to create and modify the Node Controller Configuration File (Configuration File) for the transport systems.
QuickStick Motors Technical Data, publication MMI-TD051	This manual includes technical specifications for the QuickStick 100 and QuickStick 150 motors.

Resource	Description
MagneMotion Node Controller Interface User Manual, publication MMI-UM001	This manual explains how to use the supplied interfaces to configure and administer node controllers that are used with transport systems. This manual also provides basic troubleshooting information.
MagneMotion LSM Synchronization Option User Manual, publication MMI-UM005	This manual explains how to install, operate, and maintain the LSM Synchronization Option for use with transport systems.
MagneMotion NCHost TCP/IP Interface Utility User Manual, publication MMI-UM010	This manual explains how to use the NCHost TCP/IP Interface Utility to run a transport system for testing and debugging. This manual also explains how to develop Demo Scripts to automate vehicle motion for that testing.
MagneMotion Virtual Scope Utility User Manual, publication MMI-UM011	This manual explains how to install and use the MagneMotion Virtual Scope utility. This utility provides real-time feedback of the change in Linear Synchronous Motor (LSM) performance parameters.
MagneMotion Node Controller Hardware User Manual, publication MMI-UM013	This manual explains how to install and maintain the node controllers that are used with transport systems.
MagneMover LITE Ethernet Motor Configuration and Communication, publication MMI-UM031	This manual describes the network topologies for wiring MagneMover LITE Ethernet motors and for combining both RS-422 and Ethernet motors in the same transport system.
MagneMotion Host Controller TCP/IP Communication Protocol User Manual, publication MMI-UM003	These manuals describe the communication protocols between the high level controller and a host controller. These manuals also provide basic troubleshooting information.
MagneMotion Host Controller EtherNet/IP Communication Protocol User Manual, publication MMI-UM004	
Power Supply Reference Manual 1606-XLS960F-3, publication 1606-RM032	The manual provides the specifications for the 1606 power supplies.
MagneMover LITE User Manual, publication MMI-UM002	This manual explains how to install, operate, and maintain the MagneMover LITE transport system. This manual also provides information about basic troubleshooting.
QuickStick 100 User Manual, publication MMI-UM006	This manual explains how to install, operate, and maintain the QuickStick 100 transport system. This manual also provides information about basic troubleshooting.
QuickStick 150 User Manual, publication MMI-UM047	This manual explains how to install, operate, and maintain the QuickStick 150 motors and magnet arrays. This manual also provides information about basic troubleshooting.
QuickStick HT User Manual, publication MMI-UM007	This manual explains how to install, operate, and maintain the QuickStick High Thrust (QSHT) transport system. This manual also provides information about basic troubleshooting.
EtherNet/IP Network Devices User Manual, publication ENET-UM006	Describes how to configure and use EtherNet/IP devices to communicate on the EtherNet/IP network.
Ethernet Reference Manual, publication ENET-RM002	Describes basic Ethernet concepts, infrastructure components, and infrastructure features.

Resource	Description
System Security Design Guidelines Reference Manual, publication SECURE-RM001	Provides guidance on how to conduct security assessments, implement Rockwell Automation products in a secure system, harden the control system, manage user access, and dispose of equipment.
UL Standards Listing for Industrial Control Products, publication CMPNTS-SR002	Assists original equipment manufacturers (OEMs) with construction of panels, to help ensure that they conform to the requirements of Underwriters Laboratories.
American Standards, Configurations, and Ratings: Introduction to Motor Circuit Design, publication IC-AT001	Provides an overview of American motor circuit design based on methods that are outlined in the NEC.
Industrial Components Preventive Maintenance, Enclosures, and Contact Ratings Specifications, publication IC-TD002	Provides a quick reference tool for Allen-Bradley industrial automation controls and assemblies.
Safety Guidelines for the Application, Installation, and Maintenance of Solid-state Control, publication SGI-1.1	Designed to harmonize with NEMA Standards Publication No. ICS 1.1-1987 and provides general guidelines for the application, installation, and maintenance of solid-state control in the form of individual devices or packaged assemblies incorporating solid-state components.
Industrial Automation Wiring and Grounding Guidelines, publication 1770-4.1	Provides general guidelines for installing a Rockwell Automation industrial system.
Product Certifications website, rok.auto/certifications .	Provides declarations of conformity, certificates, and other certification details.

Additional Documentation

Release Notes

The Release Notes that are supplied with Rockwell Automation software include special instructions, identification of software versions, identification of new features and enhancements, and a list of known issues. Reading this file before using the software is recommended.

Upgrade Procedure

The Upgrade Procedures that are supplied with Rockwell Automation software provide instructions for upgrading from one version of Rockwell Automation software to another. They also include the procedures for file and driver upgrades that are associated with the software.

Transport System Limits

For information on transport system limits refer to your motor user manual or the node controller user manuals listed in the [Additional Resources on page 11](#).

File Maintenance

Backup Files

It is recommended that regular backups of all files that have been changed and copies of all original and backup files be kept at a remote location for safety.

Creating Backup Files

Backup files are not created automatically. It is the responsibility of the user to create backups of all files by copying them to a secure location.

Restoring from Backup Files

Damaged files can be restored by copying the backup files into the appropriate locations.

Overview

This chapter provides an overview of the Host Controller EtherNet/IP Communication Protocol that is used to communicate between the high-level controller and the host controller (typically an Allen-Bradley[®] ControlLogix[®] Controller). This chapter also provides a basic overview of the transport system hardware and software. Additionally, the basic set of tasks that are required to use the EtherNet/IP Communication Protocol with a transport system are described.

Use this manual to develop an application using the EtherNet/IP Communication Protocol to monitor and control a transport system. The procedures that are provided in this manual as examples vary in actual use depending on the transport system configuration, communications, and other variables.

This manual supports:

- MagneMover[®] LITE transport systems.
- QuickStick[®] transport systems.
- QuickStick[®] HT[™] transport systems.

EtherNet/IP Communication Protocol Overview

The Host Controller EtherNet/IP Communication Protocol is provided by Rockwell Automation to run the transport system during operation. This protocol supports transport system startup, controlling and monitoring vehicle motion, monitoring transport system module status, and handshaking with external equipment to transfer vehicles on and off the system.

The transport system is a configuration of linear synchronous motors that are placed end-to-end to form long chains or paths. These chains are used to move and position vehicles in a controlled manner at various acceleration/deceleration and velocity profiles while carrying a wide range of payloads with high precision. The transport system consists of the following components at a minimum:

- MagneMover LITE, QuickStick, or QuickStick HT motors.
- Vehicles with magnet arrays.
- Node controllers.
- Power supplies.
- Configuration file defining paths and nodes.
- User-supplied host controller (Allen-Bradley ControlLogix controller).

Each path in the transport system starts at a node and the motor at that node is connected to a node controller. Any path that a vehicle can exit from must also end in a node that is connected to a node controller. Multiple node controllers can be used for systems with many paths. The high-level controller application runs on one node controller, which interfaces the complete transport system with the host controller. This interface allows the host controller to send commands to and receive status from the transport system. This manual specifies the protocol that the high-level controller software uses to communicate with host controllers via EtherNet/IP.

Each of the components of the transport system must be fully defined in the Node Controller Configuration File to achieve proper operation of the transport system. Once the transport system is fully defined, it can be monitored and controlled using the EtherNet/IP Communication Protocol.

NOTE: Changes to the control application may be necessary if any aspect of the transport system changes. Changes could include the number or length of paths, the number or type of nodes, vehicle length, payload weight, or other physical or configuration factors.

All host controller configuration and operation information that is provided in this manual assumes use of an Allen-Bradley ControlLogix controller using Studio 5000 Logix Designer®.

Transport System Components Overview

This section identifies the components of a transport system as shown in [Figure 1-1](#) and described after the figure.

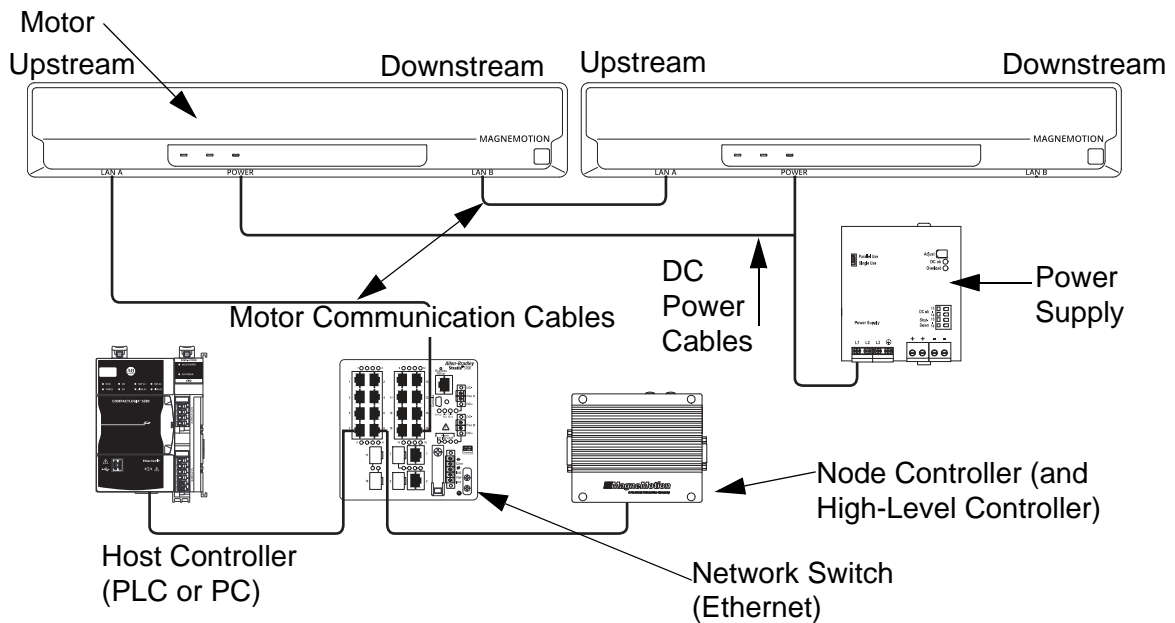


Figure 1-1: Simplified View of the Transport System Components

- **DC Power Cables and Communication Cables** – Distributes DC power to the motors and carries communications, such as RS-422 or Ethernet, between the components of the transport system.
- **High-Level Controller (HLC)** – Software application that is enabled on one node controller. This application handles all communication with the user-supplied host controller and directs communication as appropriate to individual node controllers.
- **Host Controller** – User-supplied controller for control and monitoring of the transport system using EtherNet/IP™ communication.
- **Motor/Stator** – Refers to a linear synchronous motor (LSM).
- **Network** – Ethernet network providing communications (EtherNet/IP) between the host controller, motors, and the HLC (TCP/IP is used between node controllers).
- **Node Controller (NC)** – Coordinates motor operations and communicates with the HLC. For Ethernet enabled motors only. All node controllers support Ethernet communication with the host controller and the motors, and depending on the model, provide RS-422 ports for communication with the motors.
- **Power Supply** – Provides DC power to the motors.
- **Vehicle with Magnet Array** – Carries a payload through the transport system as directed. The magnet array is mounted to the vehicle facing the motors and interacts with the motors, which move each vehicle independently.

Transport System Software Overview

Several software applications are used to configure, test, and administer a transport system as shown in [Figure 1-2](#) and described after the figure. See [Additional Resources on page 11](#) for the reference manuals for these applications.

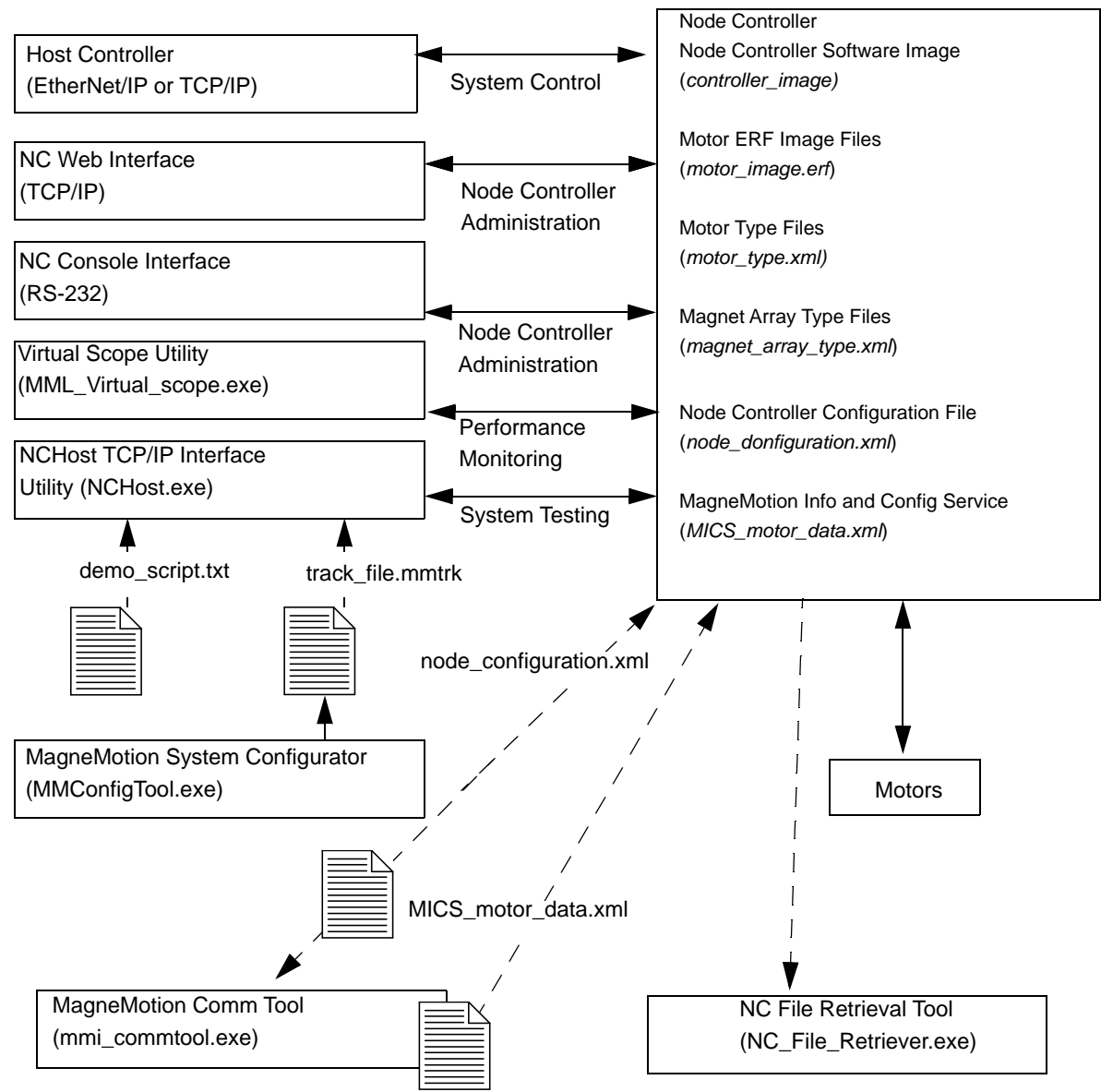


Figure 1-2: Simplified View of Transport System Software Organization

For additional information on utilities or file type information, see *MagneMotion Node Controller Interface User Manual*, publication [MMI-UM001](#).

NOTICE Modifications to the Image or Type files could cause improper operation of the transport system.

Utilities

- **NC Web Interface** – A web-based software application that is supplied by Rockwell Automation and resident on the node controllers, for administration of the transport system components.
- **NC Console Interface** – A serial communication software application that is supplied by Rockwell Automation and resident on the node controllers, for administration of the node controller.
- **Virtual Scope Utility** – A Windows[®] software application that is supplied by Rockwell Automation to monitor and record the change of motor performance parameters. These parameters are displayed as waveforms to analyze the performance of the motors.
- **NCHost TCP Interface Utility** – A Windows software application that is supplied by Rockwell Automation to move vehicles for test or demonstration purposes. This application supports system testing without the host controller to verify that vehicles move correctly before integrating the transport system into a production environment.
- **MagneMotion System Configurator Utility** (Configurator Utility) – A Windows software application that is supplied by Rockwell Automation to create or change the Node Controller Configuration File. This utility is available in both MagneMover LITE and QuickStick versions. The MM LITE[™] version is also used to create or change the Track File and Track Layout File for MagneMover LITE transport systems.
- **Ethernet Motor Commissioning Tool** – A Windows software application that is supplied by Rockwell Automation to create and edit MagneMotion Information and Configuration Service (MICS) files.
- **NC File Retrieval Tool** – A Windows software application that is supplied by Rockwell Automation to download configuration and operation files from the specified HLC and all node controllers controlled by the HLC.

File Types

- **Node Controller Software Image File** (IMG file) – The software file for the node controllers (*controller_image*), includes the node controller and high-level controller applications. The Node Controller Software Image file is uploaded to all node controllers in the transport system.
- **Motor ERF Image Files** (ERF file) – The software files for the motors (*motor_image.erf*). The Motor ERF Image files are uploaded to all node controllers in the transport system and then programmed into all motors.
- **Motor Type Files** – XML files (*motor_type.xml*) that contain basic information about the specific motor types being used. The Motor Type files are uploaded to all node controllers in the transport system.
- **Magnet Array Type File** – An XML file (*magnet_array_type.xml*) that contains basic information about the specific magnet array type that is used on the vehicles in

the transport system. The Magnet Array Type file is uploaded to all node controllers in the transport system.

- **Node Controller Configuration Files** (Configuration file) – An XML file (*node_configuration.xml*) that contains all the parameters for the components in the transport system. Multiple Node Controller Configuration Files can be uploaded to all node controllers in the transport system, but only one is active.
- **MagneMotion Information and Configuration Service (MICS) File** – An XML file (*MICS_motor_data.xml*) that contains the network topology parameters for the transport system when using Ethernet communication with the motors. The file includes the MAC address of each motor and the location of each motor on a path. The MICS file is uploaded to all node controllers in the transport system.
- **Restricted Parameters File** – An XML file (*restricted_parameters.xml*) that provides access to restricted configuration elements for specific transport systems. The Restricted Parameters file is uploaded to the HLC. For the development of a custom Restricted Parameters file for a specific transport system, see [Rockwell Automation Support](#).
- **Demo Script** – A text file (*demo_script.txt*) uploaded to the NCHost TCP Interface Utility to move vehicles on the transport system for test or demonstration purposes.
- **Track File** – A text file (*track_file.mmtrk*) that contains graphical path and motor information about the transport system. The Track file is used by the NCHost TCP Interface Utility to provide a graphical representation of the transport system to monitor system operation. The Track file is created for MagneMover LITE transport systems using the Configurator Utility. See [Rockwell Automation Support](#) for questions regarding the development of Track files for QuickStick transport systems.
- **Track Layout File** – An XML file (*track_layout.ndx*) that contains the parameters for the graphical representation of a MagneMover LITE transport system. The Track Layout file is used by the MagneMover LITE Configurator Utility to generate the Node Controller Configuration File and the Track file for MagneMover LITE systems.

NOTICE	Modifications to the Image or Type files could cause improper operation of the transport system.
---------------	--

Getting Started with the EtherNet/IP Communication Protocol

Use this manual as a guide and reference for application development when using the EtherNet/IP Communication Protocol. Follow the steps in this section to get the entire transport system operational quickly with the aid of the other manuals (see [Additional Resources on page 11](#)).

NOTE: Make sure that all components and complete design specifications, including the physical layout of the transport system, are available before starting to install or test the transport system.

To get started quickly with the transport system refer to the figure and steps below:

1. Download the software for the appropriate motor type in your transport system and the supporting Utilities from rok.auto/pcdc.
NOTE: The minimum requirements for running software applications are a general-purpose computer running Microsoft® Windows® 7 with .NET 4.0 (or later). An Ethernet port (web interface) and an optional RS-232 port (console interface) are required to connect to the node controllers. However, the EtherNet/IP Communication Protocol must be implemented on an Allen-Bradley controller running ControlLogix.
2. Install the components of the transport system as described in the *MagneMotion Node Controller Hardware User Manual*, [MMI-UM013](#), and either the *MagneMover LITE User Manual*, [MMI-UM002](#), the *QuickStick 100 User Manual*, [MMI-UM006](#), *QuickStick 150 User Manual* [MMI-UM047](#), or the *QuickStick HT User Manual*, [MMI-UM007](#).
3. Install the Configurator Utility on a computer for user access (see *MagneMotion System Configurator User Manual*, [MMI-UM046](#)).
 - A. For MM LITE systems, define the motors and paths and their relationships in the transport system graphically to create the Track Layout File (*track_layout.ndx*) and the Track File (*track_file.mmtrk*).
 - B. For all transport systems, create the Node Controller Configuration File (*node_configuration.xml*) to define the components and operating parameters of the transport system.
4. Set the IP address for each node controller and specify the node controller to be used as the high-level controller (see the *MagneMotion Node Controller Interface User Manual*, [MMI-UM001](#)).

5. Upload the configuration, image, and type files to each node controller using the node controller web interface (see the *MagneMotion Node Controller Interface User Manual*, [MMI-UM001](#)).

NOTE: Once configured, the node controllers can be used to simulate the transport system.

6. When using motors with Ethernet communication, create the MICS file and provision the motors (see either the *MagneMover LITE User Manual*, publication [MMI-UM002](#) or the *QuickStick and QuickStick HT Design Guide*, publication [MMI-RM001](#)).

7. Program the motors using the Motor ERF Image files (see the *MagneMotion Node Controller Interface User Manual*, [MMI-UM001](#), and the *MagneMotion NCHost TCP Interface Utility User Manual*, [MMI-UM010](#)).

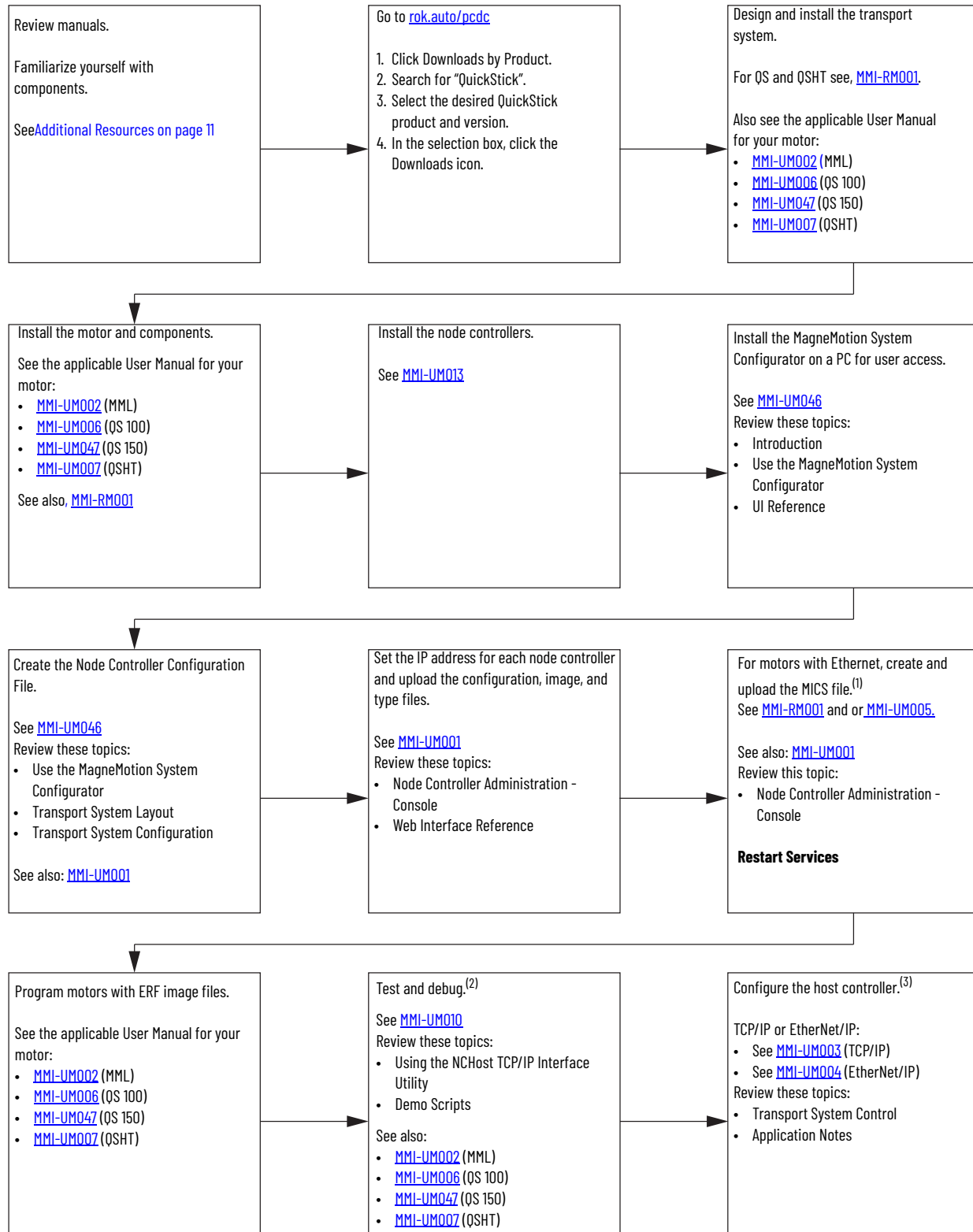
8. Test and debug the transport system by using the NCHost TCP Interface Utility and Demo Scripts (see the *MagneMotion NCHost TCP Interface Utility User Manual*, [MMI-UM010](#)). NCHost provides an easy method to verify proper operation and make adjustments such as refining the control loop tuning.

NOTE: The NCHost TCP Interface Utility is for test and verification trials only. The host controller must be used to control the transport system after verification of functionality.

9. Configure the host controller for transport system control as required to meet the material movement needs of the facility where the system is installed. Use this manual when developing a control application using EtherNet/IP communications with the host controller. See the following sections for examples:

- [Connecting to the Transport System on page 25.](#)
- [Running the Transport System on page 25.](#)
- [Monitoring Transport System Status on page 48.](#)
- [Vehicle Motion on page 71.](#)

When using TCP/IP communications with the host controller, see the *MagneMotion Host Controller TCP/IP Communication Protocol User Manual*, [MMI-UM003](#).



(1) This configuration does not apply to QS 100 motors.

(2) Use the MagneMotion Virtual Scope Utility to confirm PID tuning for all motors, if required. See the MagneMotion Virtual Scope Utility User Manual, publication [MMI-UM011](#).

(3) Configure LSM Synchronization for QS 100 motors, if necessary. See the MagneMotion LSM Synchronization Option User Manual, publication [MMI-UM005](#).

Figure 1-3: Work Flow for QuickStick System Installation and Commissioning

This page intentionally left blank.

Overview

This chapter provides basic information on using the Host Controller EtherNet/IP Communication Protocol to monitor and control a transport system.

See *Application Notes* [on page 71](#) for detailed information on vehicle motion and using the transport system. See *Protocol Reference* [on page 145](#) for detailed information on each tag. This chapter includes tag overviews and provides a representation of the tag memory. For complete message transmission protocol, see *Communications Protocol* [on page 393](#).

Connecting to the Transport System

This section describes how to connect the host controller to the high-level controller (HLC) for the transport system to monitor and control the transport system. Once the connection is established, there is no need for authentication.

1. Make a network connection from the host controller to the HLC using standard 10/100/1000 Base-TX half-duplex or full-duplex twisted-pair Ethernet cable.
2. Use the Ping utility on the host controller to verify the network connection to the HLC.
3. From the host controller, connect to the high-level controller (HLC) at TCP port 44818, see *Communications Protocol* [on page 393](#).

Running the Transport System

This section describes how to start and control the transport system. Startup includes verifying that all components of the transport system have started properly and are ready to move the vehicles and all vehicles on the transport system are identified. Control includes commanding vehicle motion as required, and stopping and restarting all motion on the transport system. Most examples that are provided in this section reference the simplified transport system layout that is shown in [Figure 2-1](#).

1. Power up the various components of the transport system (host controller, node controllers, motors, and any user equipment).
2. Connect to the high-level controller for the transport system (see [Connecting to the Transport System on page 25](#)).
3. Restart each node controller as necessary by issuing a Restart Services command through the node controller web interface (see the *MagneMotion Node Controller Interface User Manual*, [MMI-UM001](#)).

NOTE: Node controller restart is only needed when the system configuration is updated, or other type/image files have been changed. Node controllers typically take longer than any path to reset, so if power to the system is cycled (including the node controllers), simply reset and start the paths.

4. Verify that all node controllers are running (see the *MagneMotion Node Controller Interface User Manual*, [MMI-UM001](#)).

5. Reset all paths in the transport system (see [Reset on page 28](#)).

The HLC updates the [MMI_path_status](#) tag memory in the host controller for each path, which shows either completion or failure of the reset and if it failed, the reason for failure.

6. Start all paths in the transport system (see [Startup on page 30](#)).

The HLC updates the [MMI_path_status](#) tag memory in the host controller for each path, which shows either completion or failure of the startup and if it failed, the reason for failure. The HLC starts continuous updates of [MMI_vehicle_status](#).

7. Send vehicle command messages to the HLC to move the vehicles (see [Moving Vehicles on page 57](#)).

The HLC updates [MMI_path_status](#) and [MMI_node_status](#) tag memory for each path and node as the vehicle moves through them. The HLC continuously updates [MMI_vehicle_status](#) with the vehicle status. The HLC also updates [MMI_vehicle_status](#) when the move command completes (the vehicle is within the arrival tolerance of the destination as defined in the Node Controller Configuration File).

NOTE: The HLC does not update status tags to show that a move command failed to complete. The host controller is required to monitor the status tags and determine that a move command has not completed in an expected amount of time and determine the cause.

8. Monitor the status tag memory to verify the status of the transport system (see [Monitoring Transport System Status on page 48](#)).

The HLC updates status tag memory in the host controller ([MMI_vehicle_status](#) at a rate of 224 vehicle records every 100 milliseconds and [MMI_path_status](#) and [MMI_node_status](#) as there are changes in status).

9. Stop motion on specific paths in the transport system as required (see [Suspend on page 33](#)).
10. Restart motion on suspended paths as required (see [Resume on page 37](#)).
11. Shut down the transport system for service or when not in use (see [Shutting Down the Transport System on page 68](#)).

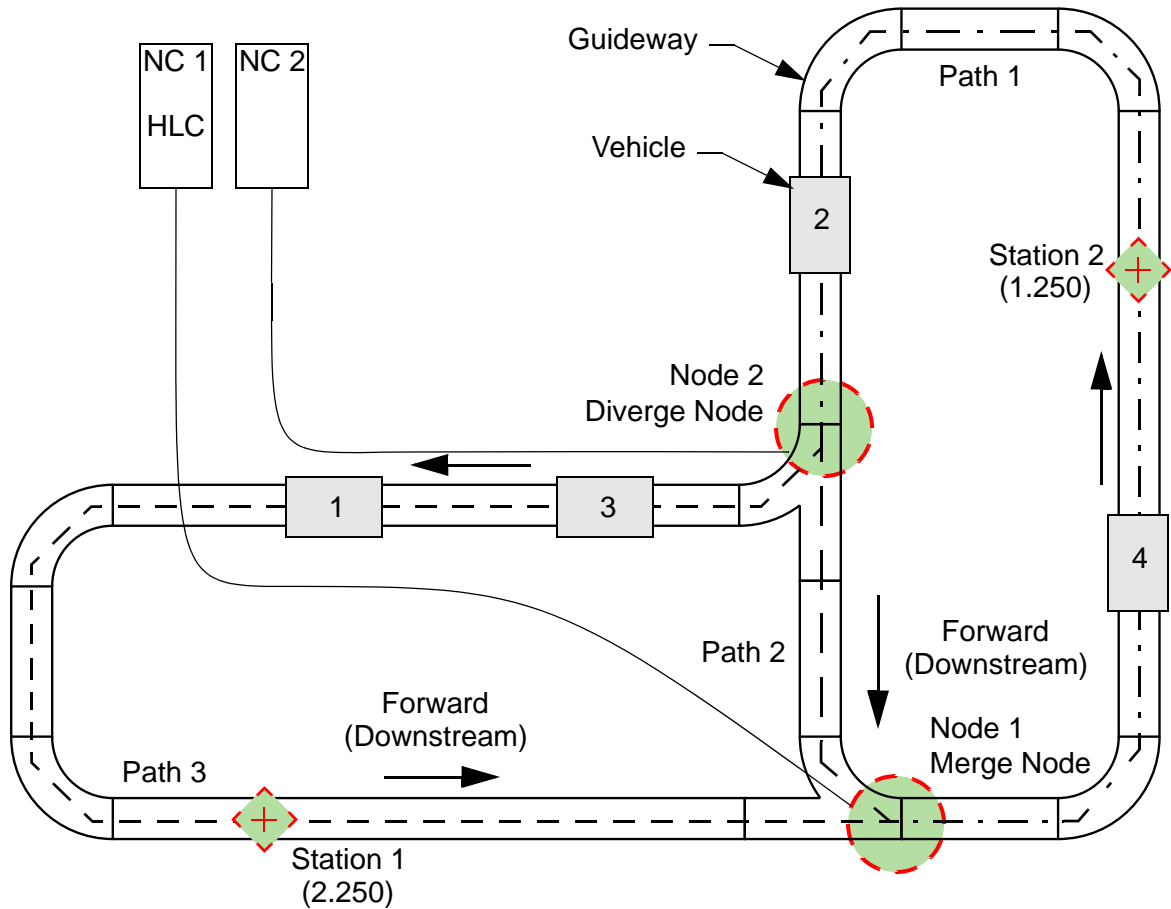


Figure 2-1: Merge-Diverge Transport System for Examples

Transport System Reset, Startup, and Operation

This section describes how to start, or restart the components of the transport system during initial startup or when returning it to service. Vehicle motion cannot be started on a path if the E-stop or Interlock is On for that path. All examples provided reference [Figure 2-1](#).

NOTE: All examples that are provided in this manual use a tabular format to represent the tags and tag memory.

Reset

Use the *MMI_path_command* to issue a Reset command to reset the motors on the specified paths. Reset all motors on all paths when a new Node Controller Configuration File has been uploaded to the node controllers.

All Paths

To reset all paths, use the *MMI_path_command* with **path_id** set to “0”, **command** set to “0xB8”, the **active_flag** set to “1”, and the **command_count** incremented from the previous value. The table represents the *MMI_path_command* message.

path_id	0
command	0xB8
active_flag	1
command_count	63

*The HLC updates the *MMI_path_command_status* array for each path, which provides their command status. The table represents the *MMI_path_command_status* array and each column identifies a path and shows the command status.*

array_index	0	1	2	3
last_command_received_count	—	63	63	63
last_command_accepted_count	—	63	63	63
last_command_received	—	0xB8	0xB8	0xB8
last_command_received_status	—	0x00	0x00	0x00
last_command_accepted	—	0xB8	0xB8	0xB8
last_command_accepted_completion_status	—	0x00	0x00	0x00

*Once the reset is complete, the HLC updates the *MMI_path_command_status* array for each path, which shows the reset was completed.*

array_index	0	1	2	3
last_command_received_count	—	63	63	63
last_command_accepted_count	—	63	63	63
last_command_received	—	0xB8	0xB8	0xB8
last_command_received_status	—	0x00	0x00	0x00

array_index	0	1	2	3
last_command_accepted	—	0xB8	0xB8	0xB8
last_command_accepted_completion_status	—	0x80	0x80	0x80

Specific Path

To reset a specific path (for example, #1), use the [MMI_path_command](#) with **path_id** set to “1”, **command** set to “0xB8”, the **active_flag** set to “1”, and the **command_count** incremented from the previous value. The table represents the [MMI_path_command](#) message.

path_id	1
command	0xB8
active_flag	1
command_count	1521

The HLC updates the [MMI_path_command_status](#) array for the path, which provides their command status. The example shows only path 1 is updated in the array, all other array cells continue to provide data from their last update, which is not shown in these examples.

array_index	0	1	2	3
last_command_received_count	—	1521	—	—
last_command_accepted_count	—	1521	—	—
last_command_received	—	0xB8	—	—
last_command_received_status	—	0x00	—	—
last_command_accepted	—	0xB8	—	—
last_command_accepted_completion_status	—	0x00	—	—

Once the reset is complete, the HLC updates the [MMI_path_command_status](#) array for the path, which shows the reset was completed.

array_index	0	1	2	3
last_command_received_count	—	1521	—	—
last_command_accepted_count	—	1521	—	—
last_command_received	—	0xB8	—	—

array_index	0	1	2	3
last_command_received_status	—	0x00	—	—
last_command_accepted	—	0xB8	—	—
last_command_accepted_completion_status	—	0x80	—	—

Startup

Use the *MMI_path_command* to issue a Startup command to initiate the motor startup sequence on one or more specified paths for the following events:

- When first connecting to a high-level controller.
- After cycling power to the transport system.
- After resetting the HLC.
- After resetting paths.
- To locate all vehicles on the transport system.

NOTE: Paths cannot be started if the E-stop or Interlock is On.

Paths cannot be started if there is a vehicle on the path that is longer than the total length of the path.

Vehicles may not be detected during startup if they straddle a node. Straddling occurs when part of the vehicle is on the downstream end of one path in a node and the rest of the vehicle is on the upstream end of another path in that node.

All Paths

To startup all paths, use the *MMI_path_command* with **path_id** set to “0”, **command** set to “0xB2”, and the **active_flag** set to “1”. The table represents the *MMI_path_command* message.

path_id	0
command	0xB2
active_flag	1
command_count	64

The HLC updates the *MMI_path_command_status* array for each path, which provides their command status. The table represents the *MMI_path_command_status* array and each column identifies a path and shows the command status.

array_index	0	1	2	3
last_command_received_count	—	64	64	64
last_command_accepted_count	—	64	64	64
last_command_received	—	0xB2	0xB2	0xB2
last_command_received_status	—	0x00	0x00	0x00
last_command_accepted	—	0xB2	0xB2	0xB2
last_command_accepted_completion_status	—	0x00	0x00	0x00

Once the startup is complete, the HLC updates the *MMI_path_command_status* array for each path, which shows the startup was completed.

array_index	0	1	2	3
last_command_received_count	—	64	64	64
last_command_accepted_count	—	64	64	64
last_command_received	—	0xB2	0xB2	0xB2
last_command_received_status	—	0x00	0x00	0x00
last_command_accepted	—	0xB2	0xB2	0xB2
last_command_accepted_completion_status	—	0x80	0x80	0x80

Specific Path

To startup a specific path (for example, #1), use the *MMI_path_command* with **path_id** set to “1”, **command** set to “0xB2”, the **active_flag** set to “1”, and the **command_count** incremented from the previous value. The table represents the *MMI_path_command* message).

path_id	1
command	0xB2
active_flag	1
command_count	1522

The HLC updates the [MMI_path_command_status](#) array for the path, which provides their command status. The example shows only path 1 is updated in the array, all other array cells continue to provide data from their last update, which is not shown in these examples.

array_index	0	1	2	3
last_command_received_count	—	1522	—	—
last_command_accepted_count	—	1522	—	—
last_command_received	—	0xB2	—	—
last_command_received_status	—	0x00	—	—
last_command_accepted	—	0xB2	—	—
last_command_accepted_completion_status	—	0x00	—	—

Once the startup is complete, the HLC updates the [MMI_path_command_status](#) array for the path, which shows the startup was completed.

array_index	0	1	2	3
last_command_received_count	—	1522	—	—
last_command_accepted_count	—	1522	—	—
last_command_received	—	0xB2	—	—
last_command_received_status	—	0x00	—	—
last_command_accepted	—	0xB2	—	—
last_command_accepted_completion_status	—	0x80	—	—

Suspend

Use the *MMI_path_command* message to issue a Suspend command to stop all vehicle motion on the specified paths at the closest allowed position in the current direction of motion. Once stopped, all vehicles are held at their locations.

NOTE: Vehicles in motion within a keep-out area in the direction of the keep-out area do not stop until they exit the keep-out area. Vehicles with a destination within the keep-out area stop at their destination.

Stopping motion on one specific path does not affect motion on other paths unless the vehicles on those paths are commanded to the suspended paths. In this case, the vehicle travels as far as it can, then stops and waits for motion on the suspended paths to be resumed.

All Paths

To suspend all paths, use the *MMI_path_command* with **path_id** set to “0”, **command** set to “0xB4”, and the **active_flag** set to “1”. The table represents the *MMI_path_command* message.

path_id	0
command	0xB4
active_flag	1
command_count	65

The HLC updates the *MMI_path_command_status* array for each path, which provides their command status. The table represents the *MMI_path_command_status* array and each column identifies a path and shows the command status.

array_index	0	1	2	3
last_command_received_count	—	65	65	65
last_command_accepted_count	—	65	65	65
last_command_received	—	0xB4	0xB4	0xB4
last_command_received_status	—	0x00	0x00	0x00
last_command_accepted	—	0xB4	0xB4	0xB4
last_command_accepted_completion_status	—	0x00	0x00	0x00

Specific Path

To suspend a specific path (for example, #1), use the *MMI_path_command* with **path_id** set to “1”, **command** set to “0xB4”, the **active_flag** set to “1”, and the **command_count** incremented from the previous value. The table represents the *MMI_path_command* message.

path_id	1
command	0xB4
active_flag	1
command_count	1523

The HLC updates the *MMI_path_command_status* array for the path, which provides their command status. The example shows only path 1 is updated in the array, all other array cells continue to provide data from their last update, which is not shown in these examples.

array_index	0	1	2	3
last_command_received_count	—	1523	—	—
last_command_accepted_count	—	1523	—	—
last_command_received	—	0xB4	—	—
last_command_received_status	—	0x00	—	—
last_command_accepted	—	0xB4	—	—
last_command_accepted_completion_status	—	0x00	—	—

FastStop

Use the *MMI_path_command* message to issue a FastStop command to stop all vehicle motion on the specified paths. The motors immediately apply maximum thrust opposing the direction of motion to all vehicles and hold all stopped vehicles at their locations.

NOTE: Vehicles in motion within a keep-out area stop within the keep-out area.

Stopping motion on one specific path does not affect motion on other paths unless the vehicles on those paths are commanded to the stopped paths. In this case, the vehicle travels as far as it can, then stops and waits for motion on the stopped path to be resumed.

This command is not supported on MagneMover LITE transport systems.

All Paths

To stop motion immediately on all paths, use the *MMI_path_command* with **path_id** set to “0”, **command** set to “0xBC”, and the **active_flag** set to “1”. The table represents the *MMI_path_command* message.

path_id	0
command	0xBC
active_flag	1
command_count	7B5

The HLC updates the *MMI_path_command_status* array for each path, which provides their command status. The table represents the *MMI_path_command_status* array and each column identifies a path and shows the command status.

array_index	0	1	2	3
last_command_received_count	—	7B5	7B5	7B5
last_command_accepted_count	—	7B5	7B5	7B5
last_command_received	—	0xBC	0xBC	0xBC
last_command_received_status	—	0x00	0x00	0x00
last_command_accepted	—	0xBC	0xBC	0xBC
last_command_accepted_completion_status	—	0x00	0x00	0x00

Specific Path

To stop motion immediately on a specific path (for example, #1), use the *MMI_path_command* with **path_id** set to “1”, **command** set to “0xBC”, the **active_flag** set to “1”, and the **command_count** incremented from the previous value. The table represents the *MMI_path_command* message.

path_id	1
command	0xBC
active_flag	1
command_count	7B6

The HLC updates the *MMI_path_command_status* array for the path, which provides their command status. The example shows only path 1 is updated in the array, all other array cells continue to provide data from their last update, which is not shown in these examples.

array_index	0	1	2	3
last_command_received_count	—	7B6	—	—
last_command_accepted_count	—	7B6	—	—
last_command_received	—	0xBC	—	—
last_command_received_status	—	0x00	—	—
last_command_accepted	—	0xBC	—	—
last_command_accepted_completion_status	—	0x00	—	—

Resume

Use the *MMI_path_command* message to issue a Resume command to restart vehicle motion on the specified paths. The Resume command is used after a FastStop, a Suspend command, or after an Emergency Stop (E-stop) was issued (for an E-stop, the E-stop button must be manually reset). Once issued, all vehicles on the specified paths resume motion based on all currently active move commands.



PINCH/CRUSH HAZARD: Moving mechanisms (vehicles) have no obstruction sensors.

Do not operate the transport system without barriers in place or personal injury could result in the squeezing or compression of fingers, hands, or other body parts between moving mechanisms.

All Paths

To resume all paths, use the *MMI_path_command* with **path_id** set to “0”, **command** set to “0xB3”, and the **active_flag** set to “1”. The table represents the *MMI_path_command* message.

path_id	0
command	0xB3
active_flag	1
command_count	66

The HLC updates the *MMI_path_command_status* array for each path, which provides their command status. The table represents the *MMI_path_command_status* array and each column identifies a path and shows the command status.

array_index	0	1	2	3
last_command_received_count	—	66	66	66
last_command_accepted_count	—	66	66	66
last_command_received	—	0xB3	0xB3	0xB3
last_command_received_status	—	0x00	0x00	0x00
last_command_accepted	—	0xB3	0xB3	0xB3
last_command_accepted_completion_status	—	0x00	0x00	0x00

Specific Path

To resume a specific path (for example, #1), use the [MMI_path_command](#) tag with **path_id** set to “1”, **command** set to “0xB3”, the **active_flag** set to “1”, and the **command_count** incremented from the previous value. The table represents the [MMI_path_command](#) message.

path_id	1
command	0xB3
active_flag	1
command_count	1524

The HLC updates the [MMI_path_command_status](#) array for the path, which provides their command status. The example shows only path 1 is updated in the array, all other array cells continue to provide data from their last update, which is not shown in these examples.

array_index	0	1	2	3
last_command_received_count	—	1524	—	—
last_command_accepted_count	—	1524	—	—
last_command_received	—	0xB3	—	—
last_command_received_status	—	0x00	—	—
last_command_accepted	—	0xB3	—	—
last_command_accepted_completion_status	—	0x00	—	—

Warm Reset

Use the [MMI_path_command](#) command with the **Warm Reset** command specified to issue a warm reset to the motors on one or more specified paths without interrupting network communications for the motors. Reset all motors on all paths when a new Node Controller Configuration File has been uploaded to the node controllers. Path status can be monitored as described in [Path Status on page 50](#).

The following actions occur during a warm reset:

- Restarts the runtime code on the motors.
- Deletes the vehicle records for vehicles on the specified paths.

NOTE: This command is not supported on QuickStick® 100 transport systems. If you send a Warm Reset to a QS 100, it will perform a regular Reset Command.

All Paths

To perform a warm reset of all paths, use the [MMI_path_command](#) tag with **path_id** set to “0”, **command** set to “0xBD”, the **active_flag** set to “1”, and the **command_count** incremented from the previous value. The table represents the [MMI_path_command](#) message.

path_id	0
command	0xBD
active_flag	1
command_count	63

The HLC updates the [MMI_path_command_status](#) array for each path, which provides their command status. The table represents the [MMI_path_command_status](#) array and each column identifies a path and shows the command status.

array_index	0	1	2	3
last_command_received_count	—	63	63	63
last_command_accepted_count	—	63	63	63
last_command_received	—	0xBD	0xBD	0xBD
last_command_received_status	—	0x00	0x00	0x00
last_command_accepted	—	0xBD	0xBD	0xBD
last_command_accepted_completion_status	—	0x00	0x00	0x00

Once the reset is complete, the HLC updates the *MMI_path_command_status* array for each path, which shows the reset was completed.

array_index	0	1	2	3
last_command_received_count	—	63	63	63
last_command_accepted_count	—	63	63	63
last_command_received	—	0xBD	0xBD	0xBD
last_command_received_status	—	0x00	0x00	0x00
last_command_accepted	—	0xBD	0xBD	0xBD
last_command_accepted_completion_status	—	0x80	0x80	0x80

Specific Path

To reset a specific path (for example, #1), use the *MMI_path_command* with **path_id** set to “1”, **command** set to “0xB8”, the **active_flag** set to “1”, and the **command_count** incremented from the previous value. The table represents the *MMI_path_command* message.

path_id	1
command	0xBD
active_flag	1
command_count	1521

The HLC updates the *MMI_path_command_status* array for the path, which provides their command status. The example shows only path 1 is updated in the array, all other array cells continue to provide data from their last update, which is not shown in these examples.

array_index	0	1	2	3
last_command_received_count	—	1521	—	—
last_command_accepted_count	—	1521	—	—
last_command_received	—	0xBD	—	—
last_command_received_status	—	0x00	—	—
last_command_accepted	—	0xBD	—	—
last_command_accepted_completion_status	—	0x00	—	—

Once the reset is complete, the HLC updates the [MMI_path_command_status](#) array for the path, which shows the reset was completed.

array_index	0	1	2	3
last_command_received_count	—	1521	—	—
last_command_accepted_count	—	1521	—	—
last_command_received	—	0xBD	—	—
last_command_received_status	—	0x00	—	—
last_command_accepted	—	0xBD	—	—
last_command_accepted_completion_status	—	0x80	—	—

Traffic Light Control

Use the traffic lights tags to create or change traffic light settings and locations. Traffic light status can be monitored as described in [Traffic Light Status on page 54](#).

This section describes the additional XML elements that are used to enable traffic lights and tailor the reporting of traffic light status to a PLC. These elements must be used in the EtherNet/IP section of the Node Controller Configuration File.

NOTE: Dynamically created traffic lights or changes to existing traffic lights are not maintained after an HLC restart.

Create Traffic Lights

To create a traffic light, use the [MMI_create_traffic_light_cmd](#) tag. Traffic lights are assigned the next available Traffic Light ID and set to green when created to allow vehicles to continue to move beyond the traffic light position. The example that is shown creates a traffic light 1.5 m from the beginning of path 3. The table represents the [MMI_create_traffic_light_cmd](#) message.

path_id	3
position	1.5
command_index	1
active_flag	1
command_count	58

The HLC updates the *MMI_traffic_light_cmd_status* array for each traffic light, which provides their command status. The table represents the *MMI_traffic_light_cmd_status* array and each column identifies a traffic light and shows the command status.

array_index	0	1
last_command_received_count	—	58
last_command_accepted_count	—	58
last_command_received	—	0x01
last_command_received_status	—	0x00
last_command_accepted	—	0x01
last_command_accepted_completion_status	—	0x80
last_command_accepted_traffic_light_id	—	1

Set Traffic Lights

To change the color of an existing traffic light, use the *MMI_set_traffic_light_cmd* command and specify the traffic light and its new color. The example that is shown changes traffic light 1 to red. The table represents the *MMI_set_traffic_light_cmd* message.

traffic_light_id	1
color	1
command_index	1
active_flag	1
command_count	0x1521

The HLC updates the *MMI_traffic_light_cmd_status* array for each traffic light, which provides their command status. The table represents the *MMI_traffic_light_cmd_status* array and each column identifies a traffic light and shows the command status.

array_index	0	1
last_command_received_count	—	1521
last_command_accepted_count	—	1521
last_command_received	—	0x02
last_command_received_status	—	0x00
last_command_accepted	—	0x02

array_index	0	1
last_command_accepted_completion_status	—	0x80
last_command_accepted_traffic_light_id	—	1

Delete Traffic Lights

To delete a traffic light, use the *MMI_delete_traffic_light_cmd* command and specify the traffic light to delete. The example that is shown deletes traffic light 3. The table represents the *MMI_delete_traffic_light_cmd* message.

traffic_light_id	1
command_index	3
active_flag	1
command_count	0x1791

The HLC updates the *MMI_traffic_light_cmd_status* array for each traffic light, which provides their command status. The table represents the *MMI_traffic_light_cmd_status* array and each column identifies a traffic light and shows the command status.

array_index	0	1
last_command_received_count	—	1791
last_command_accepted_count	—	1791
last_command_received	—	0x04
last_command_received_status	—	0x00
last_command_accepted	—	0x04
last_command_accepted_completion_status	—	0x80
last_command_accepted_traffic_light_id	—	3

The **Enable Traffic Lights** option must be selected (checked) on the EtherNet/IP Host Controller Settings page in the Configurator when creating the Node Controller Configuration File (see *MagneMotion System Configurator User Manual*, publication [MMI-UM046](#)).

Motor Controller Control

Use the [MMI_motor_inverter_command](#) tag to control the state of the inverters (motor controller) in specific QSHT motors. The motor controller in a motor can be disabled to prevent any commanded motion by that motor until the motor controllers are enabled. The motor controllers in the motor are responsible for controlling the coils in a motor block, which moves the vehicle. Inverter status can be monitored as described in [Motor Status on page 52](#).

NOTE: This command is only supported on QuickStick HT transport systems.

Disable Motor Controllers

To disable a motor controller in a QSHT motor, use the [MMI_motor_inverter_command](#) tag and specify the motor, the inverter within the motor, and its state (disable). The example that is shown disables inverter 1 in motor 1 on path 5. The table represents the [MMI_motor_inverter_command](#) message.

path_id	5
motor_id	1
inverter_id	1
inverter_control	1
active_flag	1
command_count	68

The HLC updates the [MMI_motor_inverter_cmd_status](#) array for each inverter, which provides their command status. The table represents the [MMI_motor_inverter_cmd_status](#) array and each column identifies an inverter and shows the command status.

array_index	1
last_command_received_count	68
last_command_accepted_count	68
last_command_received_status	0x00
last_command_accepted_completion_status	0x80

Enable Motor Controllers

To enable a motor controller in a QSHT motor, use the [MMI_motor_inverter_command](#) tag and specify the motor, the inverter within the motor, and its state

(enable). The example that is shown enables inverter 1 in motor 1 on path 5. The table represents the *MMI_motor_inverter_command* message.

path_id	5
motor_id	1
inverter_id	1
inverter_control	0
active_flag	1
command_count	72

The HLC updates the *MMI_motor_inverter_cmd_status* array for each inverter, which provides their command status. The table represents the *MMI_motor_inverter_cmd_status* array and each column identifies an inverter and shows the command status.

array_index	1
last_command_received_count	72
last_command_accepted_count	72
last_command_received_status	0x00
last_command_accepted_completion_status	0x80

Propulsion Power Supply Control

Use the [MMI_set_prop_power_state_cmd](#) tag to control the state of specific propulsion power supplies connected to QSHT 5700 Motor Controller. Propulsion Power Supply status can be monitored as described in [Propulsion Power Supply Status on page 56](#).

NOTE: This command is only supported on QSHT transport systems using the QSHT 5700 Motor Controller.

Set Propulsion Power Supply

To change the state of an existing Kinetix 2198-Pxxx DC-bus propulsion power supply, use the [MMI_set_prop_power_state_cmd](#) tag. Specify the type of power supply, its ID, and its new state. The example that is shown sets the state of power supply 1, which is a DFE, to running. The table represents the [MMI_set_prop_power_state_cmd](#) message.

supply_id	1
type	1
state	4
active_flag	1
command_count	517

The HLC updates the [MMI_propulsion_power_cmd_status](#) array for each power supply, which provides their command status. The table represents the [MMI_propulsion_power_cmd_status](#) array and each column identifies an inverter and shows the command status.

array_index	1
last_command_received_count	517
last_command_accepted_count	517
last_command_received	2
last_command_received_status	00
last_command_accepted	517
last_command_accepted_completion_status	80

Node Controller Digital I/O Control

Use the [MMI_node_controller_command](#) message to set the digital I/O bits on a specific node controller. Specify an **output_mask** to identify those bits that must be masked (not changed by this command) by entering the mask as a hexadecimal value (mask bit set low). Specify the **output_data** as a hexadecimal value (only those bits that are enabled by the Output Data Mask are output).

NOTE: This command is not supported on MagneMover LITE transport systems.

To set bits 4...7 of the Digital Outputs on an NC-12 node controller (NC #1), use the [MMI_node_controller_command](#) message. Set the **node_controller_id** to “1”, the **output_mask** to 0x0000F0F0, the **output_data** set to 0x000090A0 (the NC-12 provides 16 bits of digital I/O so only 2 bytes are used), and the **active_flag** set to “1”. The table represents the [MMI_node_controller_command](#) message.

node_controller_id	1
output_mask	0x0000F0F0
output_data	0x000090A0
command_count	0x90DF
active_flag	1

The HLC updates the [MMI_node_controller_cmd_status](#) tag, which shows the receipt of the Digital Output command.

array_index	0	1	2
last_command_received_count	—	0x90DF	—
last_command_accepted_count	—	0x90DF	—
last_command_received_status	—	0x00	—

The HLC updates the [MMI_node_controller_dio_status](#) array for the node controller, which provides the current state of the digital I/O where NC #1 is an NC-12 and NC #2 is an NC LITE.

array_index	0	1	2
number_dio_inputs	—	16	0
number_dio_outputs	—	16	0
dio_inputs	—	0x0000F0F0	0x00000000
dio_outputs	—	0x000090A0	0x00000000

Monitoring Transport System Status

This section describes how to monitor the basic components of the transport system. All examples provided reference [Figure 2-1](#).

1. Connect to the high-level controller for the transport system (see [Connecting to the Transport System on page 25](#)).
2. Use the host controller to examine the status array for the type of status being requested.

High-Level Controller

High-Level Controller Status

To check the status of the high-level controller, examine the [MMI_HLC_status](#) tag. This example shows that the HLC is present and operational and the software version is 7.1.20. The table represents the [MMI_HLC_status](#) array.

state	3
software_major_version	7
software_minor_version	1
software_patch_version	20

Extended High-Level Controller Status

To check the extended status of the high-level controller, examine the [MMI_extended_hlc_status](#) tag. This example shows that the HLC is present and operational, the software version is 7.1.20, a static Node Controller Configuration File is being used, and the file is valid. The table represents the [MMI_extended_hlc_status](#) array.

state	3
software_major_version	7
software_minor_version	1
software_patch_version	20
config_id	0
config_valid_flag	1

Node Controllers

Node Controller Status

To check the status of any node controller, request the appropriate index from the *MMI_node_controller_status* array. This example is for node controllers at index 1 and 2, it shows that the node controllers are present and operational and the software version is 7.1.20 on each node controller. The table represents the *MMI_node_controller_status* array.

node_controller_index	0	1	2
state	—	3	3
software_major_version	—	7	7
software_minor_version	—	1	1
software_patch_version	—	20	20

Extended Node Controller Status

To check the extended status of any node controller, request the appropriate index from the *MMI_extended_nc_status* array. This example is for node controllers at index 1 and 2, it shows that the node controllers are present and operational the software version is 7.1.20 on each node controller, managed configurations are not being used, and the Node Controller Configuration File is valid. The table represents the *MMI_extended_nc_status* array.

node_controller_index	0	1	2
state	—	3	3
software_major_version	—	7	7
software_minor_version	—	1	1
software_patch_version	—	20	20
config_id	—	0	0
config_valid_flag	—	1	1

Node Status

To check the status of any node, request the appropriate index from the *MMI_node_status* array.

This example shows that all nodes are present, the node types, that there are no vehicles in the nodes, and that the switches are in their last requested position and operational. The table represents the *MMI_node_status* array.

node_id	0	1	2
vehicle_id	—	0	0
node_type	—	1	2
terminus_signals	—	0000 0000	0000 0000
requested_position	—	1	1
reported_position	—	1	1
device_status	—	3	3

Path Status

To check the status of any path, request the appropriate index from the *MMI_path_status* array. Paths can be controlled as described in *Reset* on page 28, *Startup* on page 30, *Suspend* on page 33, *FastStop* on page 35, *Resume* on page 37, and *Warm Reset* on page 39.

This example shows that all paths are present and operational and that all communications are good. The table represents the *MMI_path_status* array.

path_index	0	1	2	3
path_state	—	2	2	2
path_motion_status	—	0000 0000	0000 0000	0000 0000
upstream_link_status	—	0	0	0
downstream_link_status	—	0	0	0

Vehicle Status

Basic Vehicle Status

To check the status of any vehicle, request the appropriate index from the *MMI_vehicle_status* array. Vehicles can be controlled as described in *Moving Vehicles on page 57*.

This example shows that there are four vehicles, the paths that the vehicles are on, that they are not in motion, their position on the paths, and that they are detected on the motor. The table represents the *MMI_vehicle_status* array.

vehicle_index	0	1	2	3	4
path_id	—	3	1	3	1
dest_path_id	—	0	0	0	1
position	—	0.89	2.69	0.50	0.75
destination_position	—	0	0	0	0
velocity	—	0	0	0	0
dest_station_id	—	0	0	0	0
command	—	0x00	0x00	0x00	0x00
flags	—	0x01	0x01	0x01	0x01

Extended Vehicle Status

The extended vehicle status returns the same status information as the *MMI_vehicle_status* array and also includes additional status information. To check the extended status of any vehicle, request the appropriate index from the *MMI_extended_vehicle_status* array. Vehicles can be controlled as described in *Moving Vehicles on page 57*.

This example shows that there are four vehicles present, the paths they are on, their current destination, their position on the paths, and they are not in motion. Vehicles 1, 2, and 4 have no current command, they are detected on the motor, they have no current destinations, they are not following another vehicle, and they are using PID set 0. Vehicle 3 has a current command to follow vehicle 4, it is detected on the motor, it has no current destination, and it is using PID set 0. The table represents the *MMI_extended_vehicle_status* array.

vehicle_index	0	1	2	3	4
path_id	—	3	2	1	1
dest_path_id	—	3	0	0	1
position	—	0.89	2.69	0.50	0.75
destination_position	—	0	0	0	0

vehicle_index	0	1	2	3	4
velocity	—	0	0	0	0
dest_station_id	—	0	0	0	0
command	—	0	0	B7	0
flags	—	0001	0001	0601	0001
followed_vehicle_id	—	0	0	4	0
current_target	—	0.89	2.69	0.50	0.75
reported_pid_set_index	—	0	0	0	0
ordered_pid_set_index	—	0	0	0	0
time_since_last_update	—	1858	1858	1858	1858
ordered_acceleration_limit	—	1.0	00000000	1.0	1.0
ordered_velocity_limit	—	0.5	0	0	0
active_move_to_station_offset	—	0	0	0	0

Motor Status

MagneMover LITE

To check the status of any MagneMover LITE motor on any path, request the appropriate index from the [MMI_path_ml_faults_status](#) array (path 2 in [Figure 2-1](#) shown). This example shows that there are three motors on the path, communication is operating properly, and that there are no faults on any of the motors. The table represents the [MMI_path_ml_faults_status](#) array for path 2.

motor_index (Path 2)	0	1	2	3
os_scheduler	—	0x00	0x00	0x00
upstream_comm	—	0x00	0x00	0x00
downstream_comm	—	0x00	0x00	0x00
motor_overall	—	0x00	0x00	0x00
master_board_faults_a	—	0x00	0x00	0x00
master_board_faults_b	—	0x00	0x00	0x00
driver_board_1_faults	—	0x00	0x00	0x00
.
.
driver_board_8_faults	—	0x00	0x00	0x00
unused_pad_1	—	0	0	0
unused_pad_2	—	0	0	0

QuickStick 100 and QuickStick 150 (QS Motors)

To check the status of any QuickStick motor on any path, request the appropriate index from the *MMI_path_qs_faults_status* array (path 2 in Figure 2-1 shown). This example shows that there are three motors on the path, communication is operating properly, and that there are no faults on any of the motors. The table represents the *MMI_path_qs_faults_status* array for path 2.

motor_index (Path 2)	0	1	2	3
os_scheduler	—	0x00	0x00	0x00
upstream_comm	—	0x00	0x00	0x00
downstream_comm	—	0x00	0x00	0x00
motor_overall	—	0x00	0x00	0x00
block_1_faults_a	—	0x00	0x00	0x00
block_1_faults_b	—	0x00	0x00	0x00
.
.
block_10_faults_a	—	0x00	0x00	0x00
block_10_faults_b	—	0x00	0x00	0x00

QuickStick HT and QSHT 5700

To check the status of any QuickStick High Thrust motors on any path, request the appropriate index from the *MMI_path_qs_ht_faults_status* array (path 2 in Figure 2-1 shown). This example shows that there are three motors on the path, communication is operating properly, and that there are no faults on any of the motors. The table represents the *MMI_path_qs_ht_faults_status* array for path 2.

motor_index (Path 2)	0	1	2	3
os_scheduler	—	0x00	0x00	0x00
upstream_comm	—	0x00	0x00	0x00
downstream_comm	—	0x00	0x00	0x00
motor_overall	—	0x00	0x00	0x00
master_board_faults	—	0x0000	0x0000	0x0000
block_1_hes_faults	—	0x0000	0x0000	0x0000
block_2_hes_faults	—	0x0000	0x0000	0x0000
block_1_inverter_faults	—	0x0000	0x0000	0x0000
block_2_inverter_faults	—	0x0000	0x0000	0x0000

motor_index (Path 2)	0	1	2	3
ethernet_comm_faults	—	0x0000 0000	0x0000 0000	0x0000 0000
block_1_safety_faults*	—	0x00	0x00	0x00
block_2_safety_faults*	—	0x00	0x00	0x00
digital_input_status*	—	0x00	0x00	0x00

* Only provided when the QSHT motor is using a QSHT 5700 Motor Controller.

Station Status

To check the status of any station, request the appropriate index from the *MMI_station_arrivals* array. Stations are defined in the Node Controller Configuration File.

This example shows that there are two stations defined (see [Figure 2-1 on page 27](#)), and there are no vehicles in either station. The table represents the *MMI_station_arrivals* array.

Station 1	2
Station 2	0

Traffic Light Status

To check the status of any traffic light, request the appropriate index from the *MMI_traffic_light_status* array. Traffic lights can be controlled as described in [Traffic Light Control on page 41](#).

This example shows that there are two traffic lights defined, traffic light 1 is 1.5 m from the start of path 1 and set to red and traffic light 2 is 1.0 m from the start of path 3 and set to green. The table represents the *MMI_traffic_light_status* array.

traffic_light_index	0	1	2
traffic_light_id	—	1	2
path_id	—	1	3
position	—	1.5	1.0
color	—	1	0
status_change_count	—	36	29

Motor Controller (Inverter) Status

To check the status of any motor inverter, request the appropriate motor index from the [MMI_path_qs_ht_faults_status](#) array. The QSHT Motor Fault data provides the status of the following inverter disable faults; Inverter disabled by command (QSHT and QSHT 5700 Motor Controller), Inverter disabled by power supply not ready (QSHT 5700).

Motor controllers can be managed as described in [Motor Controller Control on page 44](#).

Moving Path Node Status

NOTE: This function is not supported on MagneMover LITE transport systems.

To check the status of the paths in any of the Moving Path nodes, request the appropriate node ID from the [MMI_mp_path_end_status](#) array. Moving Path nodes can be controlled as described in [Moving Path Node on page 127](#).

This example references the Moving Path node that is shown in [Figure 3-45 on page 136](#). *This example shows the array index for node 3, that there are four paths, link information for each path end, vehicle information for each path, and the command counter.* The table represents the [MMI_mp_path_end_status](#) array.

node 3	Paths				
	0	1	2	3	4
path_end_state	—	4	1	6	1
path_end_role	—	2	1	3	1
path_end_type	—	1	1	2	2
peer_node_id	—	0	0	4	4
requested_path_id	—	0	0	0	0
linked_path_id	—	3	0	1	0
last_allowed_vehicle_id	—	8	0	0	0
requesting_vehicle_id	—	0	0	0	0
last_entered_vehicle_id	—	0	0	0	0
owner_vehicle_id	—	0	0	0	0
last_exited_vehicle_id	—	0	0	0	0
alignment_request_count	—	0	0	0	0
status_change_count	—	0	0	0	0

Propulsion Power Supply Status

NOTE: This function is only supported on QSHT 5700 transport systems.

To check the status of any propulsion power supply, request the appropriate index from the [MMI_propulsion_power_status](#) array. Propulsion Power Supplies can be controlled as described in [Propulsion Power Supply Control on page 46](#).

This example shows that there are two power supplies defined, they are each a DFE type, and they are both in the precharge state. The table represents the [MMI_propulsion_power_status](#) array.

power_supply_index	0	1	2
supply_id	—	1	2
type	—	1	1
state	—	1	1
status_change_count	—	7	3

Node Controller Digital I/O Status

NOTE: This function is not supported on MagneMover LITE transport systems.

To check the status of the digital I/O on any node controller, request the appropriate index from the [MMI_node_controller_dio_status](#) array. The digital I/O can be controller as described in [Node Controller Digital I/O Control on page 47](#).

This example is for node controllers at index 1 and 2. It shows node controller 1 (NC-12) has digital I/O and the current input values and output values and that node controller 2 (NC LITE) does not have digital I/O. The table represents the [MMI_node_controller_dio_status](#) array.

vehicle_index	0	1	2
number_dio_inputs	—	16	0
number_dio_outputs	—	16	0
dio_inputs	—	0x0000AAAA	0x00000000
dio_outputs	—	0x00005555	0x00000000

Moving Vehicles

This section describes how to move vehicles in the transport system. To move a vehicle, a destination must be provided as either a position on a path or as a predefined station.

NOTE: If the acceleration or velocity values in a motion command are higher than the limit set in the Node Controller Configuration File, the command is rejected.

If the velocity values are higher than the values for a specific motor, the velocity value for the motor is used while the vehicle is on that motor.

If a motion command is issued to a vehicle already in motion, and the command has a lower acceleration than the previous command to that vehicle the command is rejected.



PINCH/CRUSH HAZARD: Moving mechanisms (vehicles) have no obstruction sensors.

Do not operate the transport system without barriers in place or personal injury could result in the squeezing or compression of fingers, hands, or other body parts between moving mechanisms.

Direction of Motion

Vehicles can be moved either forward or backward on the transport system guideway. The direction of motion for a motion command is typically specified to make sure that the vehicle moves as expected. When bidirectional is specified as the direction for vehicle motion it only applies to the initial selection of direction by the transport system, the vehicle does not change direction during its move.

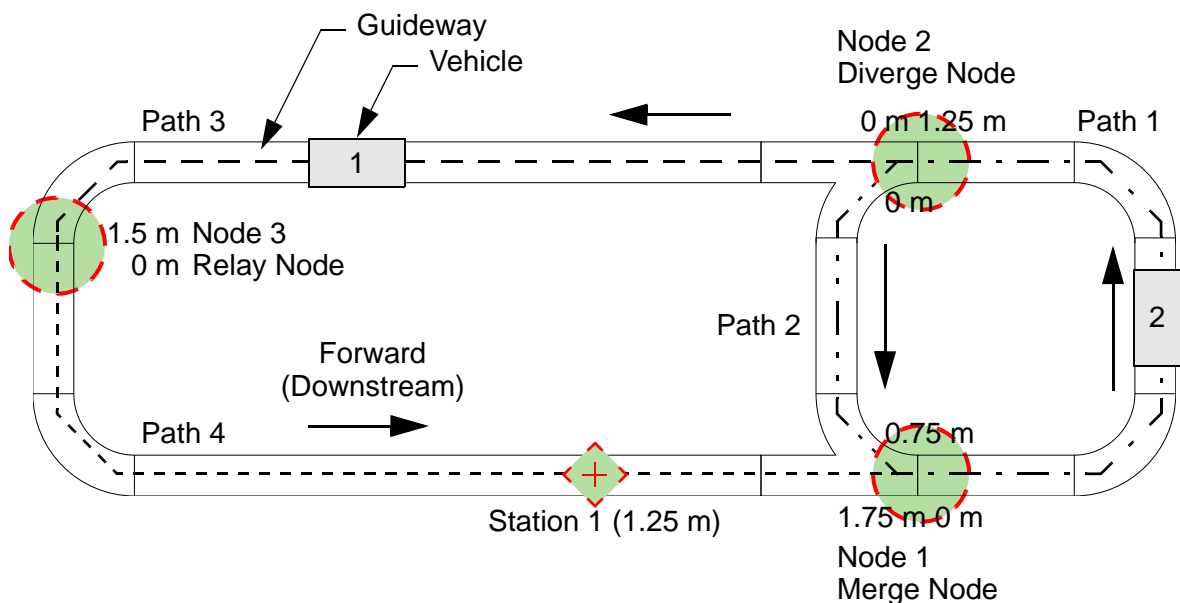


Figure 2-2: Transport System for Motion Examples

Forward – Vehicles move downstream only, useful to implement a unidirectional loop. If the destination is not reachable (for example, the paths do not form a loop), the command is rejected with a command status of 0x41 in the *MMI_vehicle_order_status* array and the vehicle does not move.

Examples:

- A. For Vehicle 1 in [Figure 2-2](#) to reach the beginning of path 3 it moves from path 3 to path 4, from path 4 to path 1, then from path 1 to path 3. Vehicle 2 must move out of the way for Vehicle 1 to complete its motion.
- B. For Vehicle 1 in [Figure 2-2](#) to reach the beginning of path 2 it moves from path 3 to path 4, from path 4 to path 1, then from path 1 to path 2. Vehicle 2 must move out of the way for Vehicle 1 to complete its motion.

Backward – Vehicles move upstream only, useful to implement a unidirectional loop in the backwards direction. If the destination is not reachable (for example, the paths do not form a loop), the command is rejected with a command status of 0x41 and the vehicle does not move.

Examples:

- A. For Vehicle 1 in [Figure 2-2](#) to reach the beginning of path 3 it moves directly to that position.
- B. For Vehicle 1 in [Figure 2-2](#) to reach the beginning of path 2 it moves from path 3 to path 1, from path 1 to path 2, then to the beginning of path 2. Vehicle 2 must move out of the way for Vehicle 1 to complete its motion.

Bidirectional – Vehicles can move in either direction as required to get to the destination in the shortest distance. Once vehicle motion is initiated, the vehicle continues in the initial direction. If the destination is on a path other than the path where the vehicle is located, the forward direction takes precedence for a transport system that is a closed-loop.

Examples:

- A. For Vehicle 1 in [Figure 2-2](#) to reach the beginning of path 3 it moves backwards directly to that position.
 - B. For Vehicle 1 in [Figure 2-2](#) to reach the beginning of path 2 it moves forward from path 3 to path 4, from path 4 to path 1, then from path 1 to path 2. Vehicle 2 must move out of the way for Vehicle 1 to complete its motion.
- Once the vehicle starts moving in a specific direction, it does not change direction.

Move to Position

Move the specified vehicle to a specified position, relative to the start of the specified path. The path must be previously defined in the Node Controller Configuration File.

To move a vehicle to a position on a path, use the *MMI_vehicle_position_order* message with the vehicle ID for the specific vehicle (for example, #2), the path ID, the destination position (measured from the start of the path), the velocity, the acceleration, the order number (required by all move commands for tracking), the flags and direction set for the move (for example, forward using PID Set 0 “unloaded” values), and the active flag to show that the command should be processed. The table represents the *MMI_vehicle_position_order* message.

vehicle_id	2
path_id	2
position	1.25
velocity_limit	1.0
acceleration_limit	1.0
order_number	17
flags_and_direction	0x01
active_flag	1

The HLC updates the *MMI_vehicle_order_status* array for each vehicle, which provides their command status. The table represents the *MMI_vehicle_order_status* array and each column identifies a vehicle and shows the command status.

array_index (vehicle_id)	0	1	2
last_order_number_received	—	17	5
last_order_number_accepted	—	17	5
last_order_type_received	—	0xB1	0xB1
last_order_received_status	—	0x00	0x00
last_order_type_accepted	—	0xB1	0xB1
last_order_accepted_is_complete	—	1	0

Once the move is complete, the HLC updates the *MMI_vehicle_order_status* array for the vehicle, which shows the move was completed.

array_index (vehicle_id)	0	1	2
last_order_number_received	—	17	5
last_order_number_accepted	—	17	5

array_index (vehicle_id)	0	1	2
last_order_type_received	—	0xB1	0xB1
last_order_received_status	—	0x00	0x80
last_order_type_accepted	—	0xB1	0xB1
last_order_accepted_is_complete	—	1	1

Move to Station

Move the specified vehicle to a specified station. The station and the path it is on must be previously defined in the Node Controller Configuration File.

To move a vehicle to a station, use the [MMI_vehicle_station_order](#) message with the vehicle ID set to the specific vehicle (for example, #2), the station ID, the velocity, the acceleration, the order number (required by all move commands for tracking), the flags and direction set for the move (for example, bidirectional using PID Set 0 “unloaded” values), and the active flag to show that the command should be processed. The table represents the [MMI_vehicle_station_order](#) message.

vehicle_id	2
station_id	1
velocity_limit	1.0
acceleration_limit	1.0
order_number	28
flags_and_direction	0x00
active_flag	1

The HLC updates the [MMI_vehicle_order_status](#) array for each vehicle, which provides their command status. The table represents the [MMI_vehicle_order_status](#) array and each column identifies a vehicle and shows its command status.

array_index (vehicle_id)	0	1	2
last_order_number_received	—	28	15
last_order_number_accepted	—	28	15
last_order_type_received	—	0xB1	0xB0
last_order_received_status	—	0x00	0x00
last_order_type_accepted	—	0xB1	0xB0
last_order_accepted_is_complete	—	1	0

Once the move is complete, the HLC updates the *MMI_vehicle_order_status* array for the vehicle, which shows the move was completed.

array_index (vehicle_id)	0	1	2
last_order_number_received	—	28	15
last_order_number_accepted	—	28	15
last_order_type_received	—	0xB1	0xB0
last_order_received_status	—	0x00	0x80
last_order_type_accepted	—	0xB1	0xB0
last_order_accepted_is_complete	—	1	1

Additionally, the HLC updates the *MMI_station_arrivals* array with the station status. The table represents the *MMI_station_arrivals* array where each row identifies a station and shows the number of the vehicle currently present at that station.

Station 1	2
Station 2	0
.	.
.	.
.	.
Station <i>n</i>	0

Platooning

Platooning provides support to move a vehicle to a position relative to another vehicle and couple to it to that vehicle to create a platoon. The lead vehicle in the platoon is moved, which causes all vehicles in the platoon to follow it without brick-wall headway allowing close spacing between the vehicles. Once there is no longer a need for the platoon, the vehicles are decoupled from it. Platoons can also be created where a vehicle within the platoon (either one vehicle or the lead vehicle for a new platoon) can automatically decouple from the platoon at a specified location.

Create a Platoon without Auto-decouple

To create a platoon, move the following vehicle into position behind the lead vehicle using a [MMI_vehicle_position_order](#) tag. The following example (see [Figure 2-2](#) for the system layout) shows the following vehicle being positioned. The command is accepted, and the command is completed once the vehicle stops moving. The table represents the [MMI_vehicle_position_order](#) message.

vehicle_id	2
path_id	3
position	0.8
velocity_limit	0.5
acceleration_limit	1.0
order_number	17
flags_and_direction	0x01
active_flag	1

The HLC updates the [MMI_vehicle_order_status](#) array for each vehicle, which provides their command status. The table represents the [MMI_vehicle_order_status](#) array, note that only vehicle 2 is shown.

array_index (vehicle_id)	2
last_order_number_received	17
last_order_number_accepted	17
last_order_type_received	0xB1
last_order_received_status	0x00
last_order_type_accepted	0xB1
last_order_accepted_is_complete	0

Once the move is complete, the HLC updates the [MMI_vehicle_order_status](#) array for the vehicle, which shows the move was completed.

array_index (vehicle_id)	2
last_order_number_received	17
last_order_number_accepted	17
last_order_type_received	0xB1

array_index (vehicle_id)	2
last_order_received_status	0x80
last_order_type_accepted	0xB1
last_order_accepted_is_complete	1

Couple the following vehicle to the lead vehicle using a [MMI_vehicle_follow_order](#) command. The following example shows the following vehicle being coupled to the lead vehicle without automatic decoupling. The command is accepted, and the command is completed once the vehicles are coupled. The table represents the [MMI_vehicle_follow_order](#) message.

vehicle_id	2
follow_distance	0.2
followed_vehicle_id	1
pid_set_index	00
catchup_acceleration	1.0
catchup_velocity	0.5
destination_path_id	0
destination_position	0
order_number	18
direction	1,2
active_flag	1

The HLC updates the [MMI_vehicle_order_status](#) array for each vehicle, which provides their command status. The table represents the [MMI_vehicle_order_status](#) array.

array_index (vehicle_id)	0	1	2
last_order_number_received	—	10	18
last_order_number_accepted	—	10	18
last_order_type_received	—	0xB1	0xB7
last_order_received_status	—	0x00	0x00
last_order_type_accepted	—	0xB1	0xB7
last_order_accepted_is_complete	—	1	0

Once the move is complete, the HLC updates the *MMI_vehicle_order_status* array for the vehicle, which shows the move was completed.

array_index (vehicle_id)	0	1	2
last_order_number_received	—	10	18
last_order_number_accepted	—	10	18
last_order_type_received	—	0xB1	0xB7
last_order_received_status	—	0x80	0x80
last_order_type_accepted	—	0xB1	0xB7
last_order_accepted_is_complete	—	1	1

Move a Platoon without Auto-decouple

Move the platoon as required using either a *MMI_vehicle_position_order* or a *MMI_vehicle_station_order* command. The following example (see Figure 2-2 for the system layout) shows the lead vehicle in the platoon moving to Station 1, the command is accepted, and the command is completed once the vehicle stops moving. The table represents the *MMI_vehicle_station_order* message.

vehicle_id	1
station_id	1
velocity_limit	1.0
acceleration_limit	1.0
order_number	19
flags_and_direction	0x01
active_flag	1

The HLC updates the *MMI_vehicle_order_status* array for each vehicle, which provides their command status. The table represents the *MMI_vehicle_order_status* array and each column identifies a vehicle and shows the command status.

array_index (vehicle_id)	0	1	2
last_order_number_received	—	19	18
last_order_number_accepted	—	19	18
last_order_type_received	—	0xB1	0xB7
last_order_received_status	—	0x00	0x80

array_index (vehicle_id)	0	1	2
last_order_type_accepted	—	0xB1	0xB7
last_order_accepted_is_complete	—	0	1

Once the move is complete, the HLC updates the *MMI_vehicle_order_status* array for the vehicle, which shows the move was completed.

array_index (vehicle_id)	0	1	2
last_order_number_received	—	19	18
last_order_number_accepted	—	19	18
last_order_type_received	—	0xB1	0xB7
last_order_received_status	—	0x00	0x80
last_order_type_accepted	—	0xB1	0xB7
last_order_accepted_is_complete	—	1	1

Decouple a Platoon

With the platoon not moving, request the appropriate index for the vehicles in the platoon from either the *MMI_vehicle_status* or the *MMI_extended_vehicle_status* array. This example shows that there are two vehicles and the status responses showing their locations. The table represents the *MMI_vehicle_status* array.

vehicle_index	0	1	2
path_id	—	3	3
dest_path_id	—	3	0
position	—	0.6	0.8
position	—	0.6	0.8
velocity	—	0	0
dest_station_id	—	1	
command	—	0xB1	0xB7
flags	—	0x01	0x01

Move the following vehicle to its current position, using a *MMI_vehicle_position_order* command set for bidirectional motion to decouple it from the platoon. The following example shows the following vehicle being positioned at its current location. The command is

accepted, and the command completes, no motion occurs. The table represents the *MMI_vehicle_position_order* message.

NOTE: Bidirectional motion must be specified when decoupling a vehicle from the platoon to make sure the command can complete.

vehicle_id	—	1	2
path_id	—	3	3
position	—	0.6	0.8
velocity_limit	—	1.0	1.0
acceleration_limit	—	1.0	1.0
order_number	—	20	21
flags_and_direction	—	0x00	0x00
active_flag	—	1	1

Create a Platoon with Auto-decouple

To create a platoon, move the following vehicle into position behind the lead vehicle using a *MMI_vehicle_position_order* command. Couple the vehicle to the lead vehicle using a *MMI_vehicle_follow_order* command that is issued to create a platoon with the vehicle following the lead vehicle. The vehicle being coupled is configured to decouple from the platoon automatically and stop at 0.6 m on path 1. The following example shows the following vehicle being coupled to the lead vehicle, the command is accepted, and the command completed. The table represents the *MMI_vehicle_follow_order* message.

vehicle_id	2
follow_distance	0.2
followed_vehicle_id	1
pid_set_index	00
catchup_acceleration	1.0
catchup_velocity	0.5
destination_path_id	1
destination_position	0.6
order_number	18
direction	0
active_flag	1

Move a Platoon with Auto-decouple

The following example (see [Figure 2-2](#) for the system layout) shows a [MMI_vehicle_position_order](#) command that is issued to move the lead vehicle in the platoon to 0.8 m on path 3, the command is accepted, and the command completed. This example shows that once the platoon reaches the correct location on path 1, vehicle 2 automatically decouples from the platoon (a decoupling Command Status is sent) and stops at 0.6 m on path 1 (a vehicle move to position complete command status is sent). The table represents the [MMI_vehicle_position_order](#) message.

vehicle_id	1
path_id	3
position	0.8
velocity_limit	1.0
acceleration_limit	1.0
order_number	21
flags_and_direction	0x01
active_flag	1

The HLC updates the [MMI_vehicle_order_status](#) array for each vehicle, which provides their command status. The table represents the [MMI_vehicle_order_status](#) array.

array_index (vehicle_id)	0	1	2
last_order_number_received	—	21	18
last_order_number_accepted	—	21	18
last_order_type_received	—	0xB1	0xB7
last_order_received_status	—	0x00	0x00
last_order_type_accepted	—	0xB1	0xB7
last_order_accepted_is_complete	—	0	0

The HLC updates the [MMI_vehicle_order_status](#) array for the vehicle, which shows the vehicle is decoupling from the platoon (0x82).

array_index (vehicle_id)	0	1	2
last_order_number_received	—	10	18
last_order_number_accepted	—	10	18

array_index (vehicle_id)	0	1	2
last_order_type_received	—	0xB1	0xB7
last_order_received_status	—	0x00	0x82
last_order_type_accepted	—	0xB1	0xB7
last_order_accepted_is_complete	—	0	0

The HLC updates the *MMI_vehicle_order_status* array for the vehicle, which shows the decoupled vehicle is at its destination.

array_index (vehicle_id)	0	1	2
last_order_number_received	—	10	18
last_order_number_accepted	—	10	18
last_order_type_received	—	0xB1	0xB1
last_order_received_status	—	0x00	0x80
last_order_type_accepted	—	0xB1	0xB1
last_order_accepted_is_complete	—	0	1

Shutting Down the Transport System

Whenever the transport system is being shut down it is important that all motion is stopped and the system is powered down correctly.

1. Use the *MMI_path_command* to suspend motion on all paths, which causes all vehicles to come to a controlled stop at the closest allowed position in the current direction of motion. The table represents the *MMI_path_command* message.

path_id	1
command	0xB4
active_flag	1
command_count	6856

The HLC updates the *MMI_path_command_status* array for each path, which shows the command was accepted for each path. The table represents the *MMI_path_command_status* array.

array_index	0	1	2	3	4
last_command_received_count	—	6856	6856	6856	6856
last_command_accepted_count	—	6856	6856	6856	6856
last_command_received	—	0xB4	0xB4	0xB4	0xB4
last_command_received_status	—	0x00	0x00	0x00	0x00
last_command_accepted	—	0xB4	0xB4	0xB4	0xB4
last_command_accepted_completion_status	—	0x00	0x00	0x00	0x00

Once the Suspend is complete, the HLC updates the *MMI_path_command_status* array for each path, which shows the Suspend was completed and all motion has stopped.

array_index	0	1	2	3	4
last_command_received_count	—	6856	6856	6856	6856
last_command_accepted_count	—	6856	6856	6856	6856
last_command_received	—	0xB4	0xB4	0xB4	0xB4
last_command_received_status	—	0x00	0x00	0x00	0x00
last_command_accepted	—	0xB4	0xB4	0xB4	0xB4
last_command_accepted_completion_status	—	0x80	0x80	0x80	0x80

- Once all motion has stopped, issue a Reset for all paths by sending an *MMI_path_command*, which clears all vehicle records. The table represents the *MMI_path_command* message.

path_id	1
command	0xB8
active_flag	1
command_count	6857

The HLC updates the *MMI_path_command_status* array for each path, which shows the Reset was accepted. The table represents the *MMI_path_command_status* array.

array_index	0	1	2	3	4
last_command_received_count	—	6857	6857	6857	6857
last_command_accepted_count	—	6857	6857	6857	6857
last_command_received	—	0xB8	0xB8	0xB8	0xB8
last_command_received_status	—	0x00	0x00	0x00	0x00
last_command_accepted	—	0xB8	0xB8	0xB8	0xB8
last_command_accepted_completion_status	—	0x00	0x00	0x00	0x00

Once the reset is complete, the HLC updates the *MMI_path_command_status* array for each path, which shows the Reset was completed.

array_index	0	1	2	3	4
last_command_received_count	—	6857	6857	6857	6857
last_command_accepted_count	—	6857	6857	6857	6857
last_command_received	—	0xB8	0xB8	0xB8	0xB8
last_command_received_status	—	0x00	0x00	0x00	0x00
last_command_accepted	—	0xB8	0xB8	0xB8	0xB8
last_command_accepted_completion_status	—	0x80	0x80	0x80	0x80

3. Turn off power to the motors.
4. Turn off power to the node controllers.
5. Turn off power to the host controller.

Overview

This chapter provides examples and step-by-step procedures that are related to specific tasks using motors in a transport system.

Application Notes are provided for specific tasks in the operation of MagneMover[®] LITE and QuickStick[®] transport systems. Simple step-by-step procedures and examples for each application that is covered are provided. These procedures reference detailed information about the steps that is located elsewhere in this manual.

Vehicle Motion

Brick-wall Headway

All vehicles on a transport system are moved with a brick-wall headway between vehicles so that a safe stopping distance between vehicles is always maintained. Brick-wall headway makes sure that if one vehicle stops unexpectedly (for example, an object falls on the guideway) any vehicle following it is able to stop without colliding with it. It is possible to override this move profile if necessary in special circumstances.

Vehicle Positioning

The location a vehicle is commanded to must allow the entire vehicle (as defined in the Node Controller Configuration File) to remain on the transport system (over the motor). The only time any portion of a vehicle can be commanded to a location off a motor is during vehicle insertion or removal through a Terminus Node. If a vehicle is commanded to a position that causes any portion of the vehicle to leave the end of the guideway, the vehicle stops before it reaches that position. This position control makes sure that no part of the vehicle leaves the end of the guideway and since it never reaches its destination it never completes the motion command.

Example: Figure 3-1 shows a simple path that starts at a Simple Node with no connecting path. Commanding the 200 mm vehicle that is shown in the figure to a position of 0 m on the path causes the vehicle to stop at the 100 mm position. The motion command never completes and no Command Status for completion is sent as the command stays pending. In this case, the Obstructed bit in the Vehicle Status for the vehicle is set, which shows that the vehicle cannot move to complete the command.

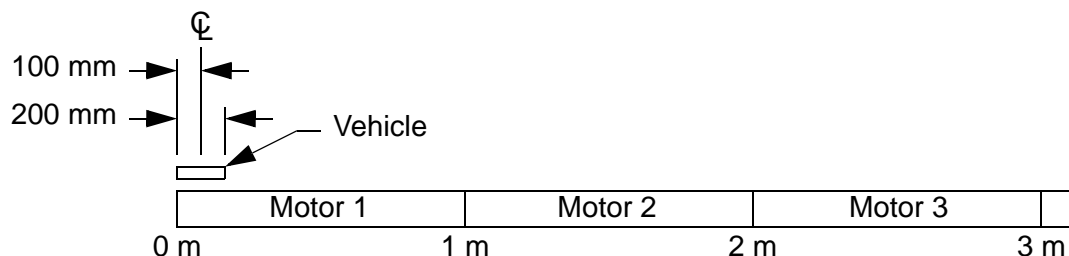


Figure 3-1: Vehicle Positioning Example

Safe Stopping Distance

Standard vehicle control and operation makes sure that vehicles always have a safe stopping distance, referred to as brick-wall headway. Figure 3-2 shows acceleration, velocity, and position versus time for the standard vehicle motion profile. Permission for vehicle motion is granted as required on a block-by-block basis. This permission keeps vehicles on their move profile (solid heavy line) and provides a safe stopping distance (dashed heavy line) based on the current velocity and commanded acceleration of the vehicle. This stopping distance can be found by dividing the square of the current velocity of a vehicle by twice its acceleration ($V^2/2a$).

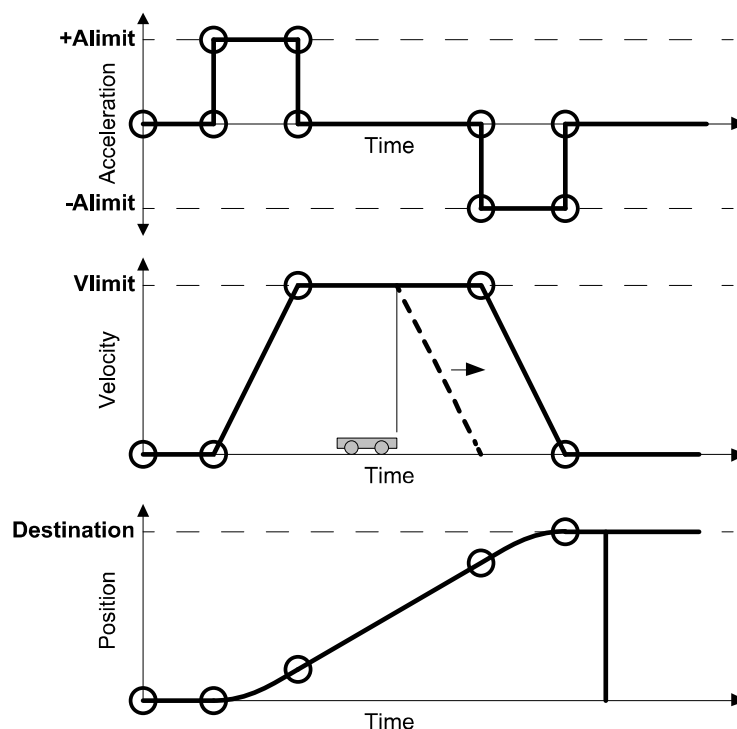


Figure 3-2: Vehicle Motion Profile

Thrust Limitations

When a vehicle is commanded with a higher acceleration rate than the motor can provide, the vehicle falls behind its ideal move profile while accelerating. [Figure 3-3](#) shows both the ideal move profile (solid line) and the degraded move profile (dashed line).

In addition, and more critically, the vehicle is not able to decelerate at the specified rate and overshoots its destination as shown by the dashed line in [Figure 3-3](#). This behavior can result in vehicles colliding with other vehicles or switch components, or loss of control of a vehicle as it exits the area where it has permission to move. Thus, it is important to avoid commanding a move with an acceleration that is higher than the deceleration capability of the system.

The precise deceleration capability depends on vehicle mass (including payload), center of gravity location, speed, and track geometry. Furthermore, the thrust capability of the motors is reduced in proximity to the gaps between motors.

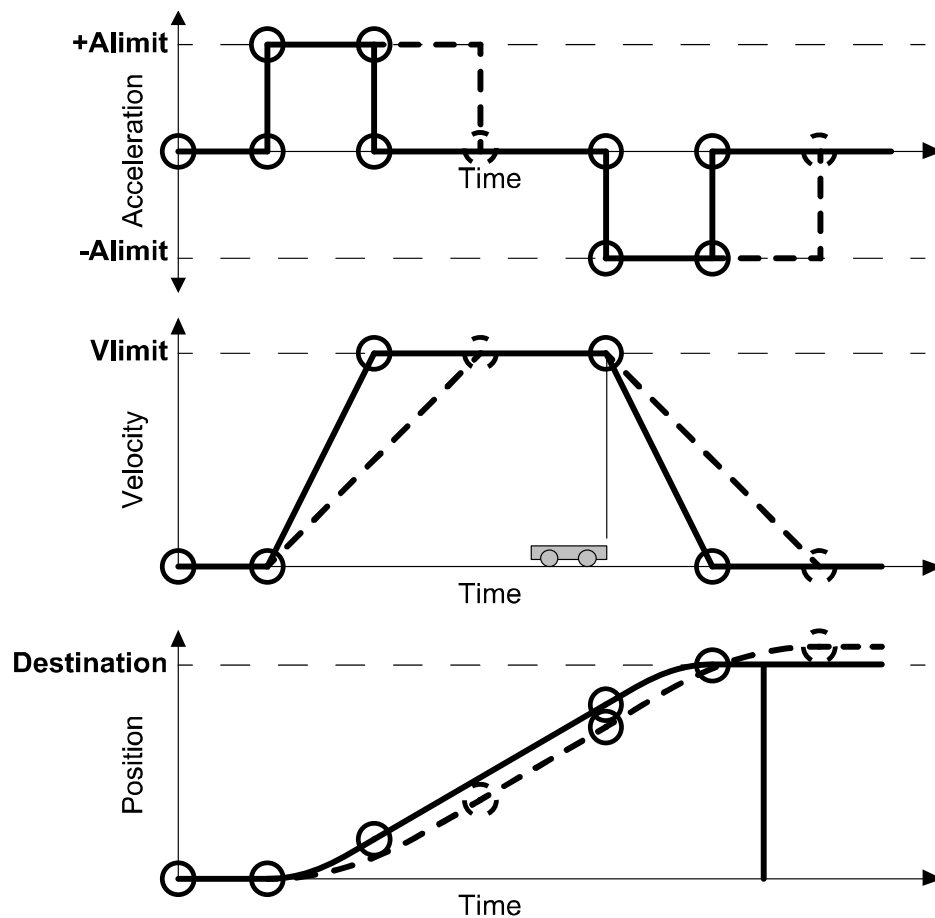


Figure 3-3: Vehicle Motion Profile Showing Thrust Limitations

Inserting and Removing Vehicles

Rockwell Automation recommends that a Terminus Node be used for all insertion and removal of vehicles on the transport system. Terminus Nodes provide all needed handshaking with the remote equipment for smooth transfer of the vehicle and proper identification or deletion of the vehicle on the transport system.

In some situations, it is not possible to move the vehicle fully past the end of a path where a Terminus Node would be located. An alternate method to insert or remove vehicles on the transport system is described in this section.

NOTE: In the following examples, the insert path and the remove path can be the same path if vehicles are both inserted and removed on that path.

When using EtherNet/IP communication, turn off cached connections for the host.

Vehicle Insertion

1. Define a short path (minimum length of one motor and a maximum length of one vehicle length) as the insert path.
2. Issue a Reset command to the insert path.
3. Place the vehicle to be inserted onto the insert path.
4. Issue a Startup command to the insert path.
5. Review the *MMI_vehicle_status* or *MMI_extended_vehicle_status* tag to check the status of all vehicles.

The Vehicle Status tags identify the vehicle ID of the new vehicle now on the insert path.

6. Command the vehicle from the insert path onto the transport system.

Vehicle Removal

1. Define a short path (minimum length of one motor and a maximum length of one vehicle length) as the remove path.
2. Issue a Move Vehicle command to locate the vehicle on the remove path.
3. Issue a Reset command to the remove path.
4. Remove the vehicle from the remove path.
5. Issue a Startup command to the remove path.
6. Review the *MMI_vehicle_status* or *MMI_extended_vehicle_status* tag to check the status of the removed vehicle.

The Vehicle Status tags shows that the vehicle ID of the removed vehicle is no longer being used.

E-stops

An E-stop uses a user-installed locking button that an operator can press if an emergency situation arises to suspend all motion on the specified paths. Once the emergency is resolved, motion can be restored.



SHOCK HAZARD: The E-stop only executes the actions that are described, it is not the same as an EMO (Emergency Off), which removes all power to the transport system.



PINCH/CRUSH HAZARD: Moving mechanisms (vehicles) have no obstruction sensors.

An E-stop does not suspend motion of a vehicle within a keep-out area or motion of a vehicle that has already been granted permission to move through a Terminus Node.

Do not operate the transport system without barriers in place or personal injury could result in the squeezing or compression of fingers, hands, or other body parts between moving mechanisms.

Initiating a Host Controller E-stop

The E-stop can also be configured to operate through the host controller, which requires the host controller to monitor the circuit and initiate the desired action when signaled.

NOTE: If an E-stop is wired to, and monitored by, the host controller, it is the responsibility of the host controller to command the system to perform the desired actions.

Upon receiving an E-stop request, the host controller must use an [MMI_path_command](#) to issue either a Suspend Motion or a FastStop command to the appropriate paths. If a Suspend Motion command is issued, all motors suspend vehicle target requests and permissions. All vehicles then come to a controlled stop at the furthest position they have permission to go to and are held in position by the motors. If a FastStop command is issued, all vehicles immediately decelerate with maximum thrust opposing motion. All motors suspend vehicle target requests and permissions and all vehicles are held in position by the motors. The HLC updates the [MMI_path_status](#) tag for each path, which shows that motion is suspended for each path.

NOTE: When a Suspend Motion command is issued, vehicles in motion within a keep-out area in the direction of the keep-out area do not stop until they exit the keep-out area. Vehicles with a destination within the keep-out area stop at their destination.

When a FastStop command is issued, vehicles in motion within a keep-out area stop within the keep-out area.

Stopping time for each vehicle is dependent on the load, available thrust, and the velocity and acceleration settings.

Power is not removed from the motors.

To check the status of a path, request the appropriate index from the [MMI_path_status](#) tag. The **path_motion_status** shows either “Suspended by Command” or “FastStop Active” for the stopped paths.

Motion cannot resume until the host controller issues a Resume command to the paths associated with the E-stop.

If the host controller-based E-stop is configured so that it shuts off motor power when activated, the host controller must use an [MMI_path_command](#) to issue either a Suspend Motion or a FastStop command before shutting off power. Once all motion has stopped, the host controller can remove power. Time to wait before shutting off power can be determined based on the acceleration and velocity that is configured for the transport system. Make sure that the vehicles have sufficient time to come to a controlled stop.

- If only the motor propulsion power is shut off (motor logic and motor propulsion power must be wired separately), there is an under-voltage fault. When power is restored, the fault is cleared.
 - For MagneMover LITE, [MMI_path_ml_faults_status](#) reports a “Propulsion power not ready” fault.
 - For QuickStick motors, [MMI_path_qs_faults_status](#) reports a “Soft Start not complete” fault.
 - For QuickStick HT motors, [MMI_path_qs_ht_faults_status](#) reports “Soft Start switch off” fault.
- If both motor propulsion power and motor logic power are shut off, the [MMI_path_ml_faults_status](#), [MMI_path_qs_faults_status](#), or [MMI_path_qs_ht_faults_status](#) tag is updated and show a “Motor Not Responding” fault. When power is restored, the fault is cleared.

Recovering from a Host Controller E-stop

Once the events that necessitated the E-stop have been resolved, motion can be restarted. To make sure of proper startup, the E-stop must be cleared by releasing the button that was pressed.

If power was not removed from the motors, once the events that necessitated the E-stop have been resolved the host controller must use an *MMI_path_command* to issue a Resume command to the suspended paths. All vehicles on the suspended paths then resume their motion, which is based on all currently active motion commands.

If power was removed from the motors:

- If only the motor propulsion power was removed, all vehicles on the suspended paths immediately resume their motion once propulsion power is restored. The motion is based on all currently active move commands.
- If both motor propulsion and motor logic power were removed, the motors must be restarted as described in *Startup on page 30* once power is restored.

Interlocks

An interlock is a user-installed circuit that another piece of equipment in the facility activates to suspend all motion on the specified paths temporarily. Once the other equipment completes its task and deactivates the interlock, motion is automatically restored.

NOTE: This is to support legacy node controllers with digital I/O. New systems should control their interlock functionality through the host controller.

Initiating an Interlock

When the node controller detects that the interlock circuit is activated, it commands all paths that are associated with that interlock to suspend vehicle motion. All motors suspend vehicle target requests and permissions. All vehicles then come to a controlled stop at the furthest position they have permission to go to and are held in position by the motors. The HLC updates the *MMI_path_status* tag for each path, which shows the interlock is active for the stopped paths.

NOTE: Vehicles in motion within a keep-out area in the direction of the keep-out area do not stop until they exit the keep-out area. Vehicles with a destination within the keep-out area stop at their destination.

Stopping time for each vehicle is dependent on the load, available thrust, and the velocity and acceleration settings.

Power is not removed from the motors.

To check the status of a path, request the appropriate index from the *MMI_path_status* tag. The **path_motion_status** shows “Interlock Active” for the stopped paths.

Motion cannot resume until the associated equipment releases the interlock.

Recovering from an Interlock

Once the events that necessitated the interlock have been resolved, motion is restarted by deactivating the interlock signal. All vehicles on the suspended paths immediately resume their motion, which is based on all currently active motion commands.

Emergency Machine Off

The equipment that uses a transport system can have an Emergency Machine Off (EMO) circuit that is user-installed and configured. These circuits typically include a locking EMO button configured such that pressing the button removes all power to the transport system. The power that is removed typically includes motor propulsion power and possibly motor logic, node controller, and HLC power. On some transport systems, pneumatic power must also be removed.

Initiating an EMO

Upon loss of propulsion power all vehicles slow to a stop. The amount of time it takes to stop is dependent on vehicle velocity, load, and friction in the vehicle guideway.

Recovering from an EMO

Once the events that necessitated the EMO have been resolved, the transport system can be restarted (see *Startup* [on page 30](#)). To make sure of proper startup, the EMO must be cleared by releasing the button that was pressed before restarting the transport system.

Platooning

Vehicle platooning allows moving multiple vehicles simultaneously without the restriction of brick-wall headway between the vehicles by controlling only the lead vehicle. Brick-wall headway is still maintained before the lead vehicle in the platoon. Platooning is useful for stations that have multiple processes, as it allows a set of vehicles to move into, and out of, position in the station simultaneously, which can minimize change over time. For detailed examples of platooning, see *Platooning* [on page 61](#).

Create platoons by coupling a following vehicle to a leading vehicle. Once a platoon of two or more vehicles is created, additional vehicles are added to it by coupling the additional vehicles to the end of the platoon. Or, by coupling the platoon to a new lead vehicle. Once a platoon is created, control of the vehicle at the front of the platoon (the leader) controls the behavior of all vehicles in the platoon (acceleration, velocity, position).

NOTE: Platooning is only available for MagneMover LITE transport systems on version 15.11.10 and later.



PINCH/CRUSH HAZARD: Moving mechanisms (vehicles) have no obstruction sensors.

Do not operate the transport system without barriers in place or personal injury could result in the squeezing or compression of fingers, hands, or other body parts between moving mechanisms.

This feature currently supports the following functions:

- Vehicles can be coupled into a platoon only when the platoon is not moving.
- Vehicles can be uncoupled from a platoon either when the platoon is not moving or when the vehicle follow order for a specific vehicle is configured for auto-decouple.
- All vehicles in a platoon must be the same size (that is, all single vehicles or all tandem vehicles).
- Motion on any path and motion across a node is supported.
- Coupling vehicles to a platoon that is located across a node is supported.
- Uncoupling a vehicle from a platoon that is located across a node is supported.
- The vehicle follow order to the HLC must include acceleration and velocity values.
- Platoons cannot reverse direction. To change the direction of motion, the platoon must stop and be uncoupled. Once uncoupled, recouple the vehicles in the opposite direction. The last vehicle in the original platoon becomes the lead vehicle in the new platoon.

Platooning Operations Overview

The platooning feature couples a following vehicle to the vehicle in front of it at an ordered position offset to mimic the move profile (acceleration and velocity) of the platoon leader. This feature can be used to create a platoon of N vehicles to move out of a station while a new platoon of N vehicles moves in.



ATTENTION: Vehicles following another vehicle in a platoon do not support collision avoidance (brick-wall headway) between vehicles in the platoon.

When a platoon is moving through a node, the following vehicles in the platoon continue to follow the lead vehicle. Ownership of the node is only released after the entire platoon passes through the node.

The functioning of Single Vehicle areas, Keepout areas, and Traffic Lights is not guaranteed with vehicles that are in a platoon. Only the lead vehicle obeys these functions.

A vehicle ordered to follow another vehicle with a platooning command will obey the specified platoon following distance, and will NOT use wide vehicle spacing. A platoon spacing distance must be specified that is large enough to avoid collisions along the route that the platoon will travel (for example, through curves).

- When using multiple platoons, the distance from the following platoon to the leading platoon must obey brick-wall headway.
- Platoons can be “coupled” and “uncoupled” to move vehicles into and out of a station without the constraint of brick-wall headway between the vehicles in the platoon.
- All vehicles in a platoon must be the same size and use the same type of magnet array.
- When coupling a vehicle into a platoon, the center-to-center distance between vehicles must keep the vehicles from contacting each other when moving through a curve.
- Attempting to couple across a node that another platoon is crossing is rejected.
- For MagneMover LITE, to reduce the effect of competing forces between adjacent magnet arrays and the coils in the motor, the recommended minimum center-to-center space from leading to following vehicles when creating a platoon is:
 - 100 mm for standard 62 x 62 mm single pucks with a 77 mm configured length.
 - 173 mm for standard 62 x 150 mm tandem pucks with a 165 mm configured length.

If the vehicle length is longer than the minimal puck length described above, the minimum gap between vehicles must be the value that is returned from the following formula.

$$\text{MinGap} = 30.0 \text{ mm} + \text{VehicleLength}$$

Where:

MinGap – Minimum gap between vehicles in a platoon (in mm).

VehicleLength – Length of the vehicle (in mm).

The gap that is calculated must be added to the actual vehicle length when deriving minimal vehicle spacing in the platoon, subject to practical minimal distance recommendations.

If the vehicle overhangs the puck, the vehicle edges can be placed closer than the distance that the formula defines. However, if the vehicles are placed too close to each other, tolerances in the vehicle following the profile can cause vehicles to touch.

- For QuickStick100 motors using firmware version 15.9.19, to reduce the effect of competing forces between adjacent magnet arrays and the coils in the motor, the recommended minimum center-to-center space from leading to following vehicles when creating a platoon is:

$$\text{MinGap} = 112 \text{ mm} + \text{VehicleLength} + \text{LargestDownstreamGap}$$

Where:

MinGap – Minimum gap between vehicles in a platoon (in mm).

VehicleLength – Length of the vehicle (in mm).

LargestDownstreamGap – Length of the largest downstream gap (in mm).

The gap that is calculated must be added to the actual vehicle length in deriving minimal vehicle spacing in the platoon, subject to practical minimal distance recommendations.

- To make effective use of platooning, the maximum distance between the vehicles in a platoon should be no greater than:)

$$\text{MaxDistance} = \text{MaxCommandedVelocity}^2 / (\text{CommandedAcceleration} * 2)$$

Where:

MaxDistance – Maximum distance between vehicles in a platoon (in meters).

MaxCommandedVelocity – Maximum velocity defined in the Node Controller Configuration File (in m/s).

CommandedAcceleration – Maximum acceleration being used in a motion command for the platoon (in m/s²).

- When coupling a vehicle into a platoon, the maximum difference between the physical distance and the requested distance between the vehicles (the distance that is specified in the command) must be within 30 mm.

- When coupling a vehicle to a platoon using the vehicle follow order, the vehicle and the platoon must be stopped. Stopped is defined as vehicle motion must be below the **Arrival Velocity Tolerance** defined on the **Global Settings** page in the Configurator Utility when creating the Node Controller Configuration File.
- When uncoupling a vehicle from a platoon using a vehicle position order, the platoon must be stopped (vehicle motion must be below the **Arrival Velocity Tolerance**) and the order must specify bidirectional motion.
- When uncoupling a vehicle from a platoon using a vehicle follow order with a decouple destination, the vehicle decouples from the platoon when the platoon approaches the decouple destination. The vehicle then stops at the decouple destination when the platoon passes that position.
- The leading vehicle in a platoon must be leading in the direction of travel.
- Each vehicle added to the platoon must be added in the direction of travel of the platoon.
- Deleting the lead vehicle from a platoon causes the platoon to stop due to lack of profile data from the leader. The next vehicle in the platoon becomes the new leader. To continue motion of the platoon, send a vehicle motion order to the new leader.

If the vehicle behind the lead vehicle was the end of a two-vehicle platoon, the platoon no longer exists.
- Platoons cannot couple or uncouple vehicles while using Sync.
- Sync can only be used on the lead vehicle of a platoon, and the sync profile must continue in the direction of the platoon. Followers in the platoon ignore all sync messages.

Creating Platoons

Platoons can be created for vehicle motion either downstream or upstream. Once a platoon is created, it is limited to motion in the direction that is specified when it was created. The direction of a platoon can be reversed by decoupling the vehicles in the platoon and creating a platoon moving in the opposite direction (see *Change Platoon Direction* [on page 89](#)).

To couple a vehicle into a platoon, or uncouple a vehicle from a moving platoon, use the *MMI_vehicle_position_order* and the *MMI_vehicle_follow_order* tags as described in the following examples.

Follow A Vehicle Moving Downstream

Create a two-vehicle platoon that moves downstream by specifying a vehicle for another vehicle to follow.

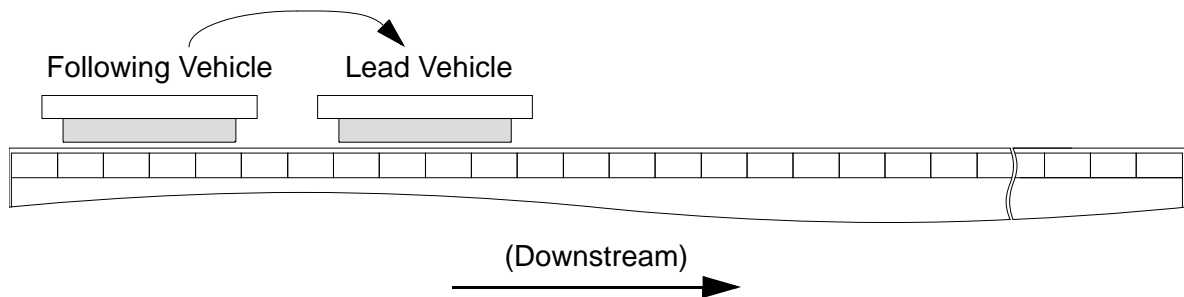


Figure 3-4: Create a Downstream Platoon

1. While the vehicle that is going to be the platoon leader is stopped, command the following vehicle to move downstream into position behind it (upstream of it) using an *MMI_vehicle_position_order* tag. The vehicle follow order must specify this distance between the vehicles as the Follow Distance.

The following vehicle moves to its commanded position.

2. Couple the following vehicle to the lead vehicle and specify the distance (from [Step 1](#)) between the vehicles and the direction of motion (downstream) using an *MMI_vehicle_follow_order* tag.

The vehicles are linked into a platoon at the specified distance between the vehicles.

3. Command the lead vehicle to move downstream as required.

The following vehicle follows behind the lead vehicle at the specified distance.

To add additional vehicles to the platoon, see *Add Additional Vehicles To The End Of A Platoon* [on page 85](#).

Follow A Vehicle Moving Upstream

Create a two-vehicle platoon that moves upstream by specifying a vehicle for another vehicle to follow.

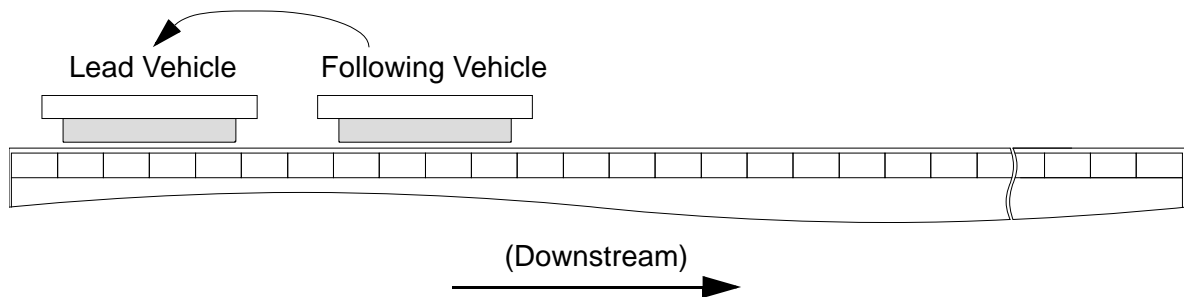


Figure 3-5: Create an Upstream Platoon

1. While the vehicle that is going to be the platoon leader is stopped, command the following vehicle to move upstream into position behind it (downstream of it) using an *MMI_vehicle_position_order* tag. The vehicle follow order must specify this distance between the vehicles as the Follow Distance.

The following vehicle moves to its commanded position.

2. Couple the following vehicle to the lead vehicle and specify the distance (from [Step 1](#)) between the vehicles and the direction of motion (upstream) using an *MMI_vehicle_follow_order* tag.

The vehicles are linked into a platoon at the specified distance between the vehicles.

3. Command the lead vehicle to move upstream as required.

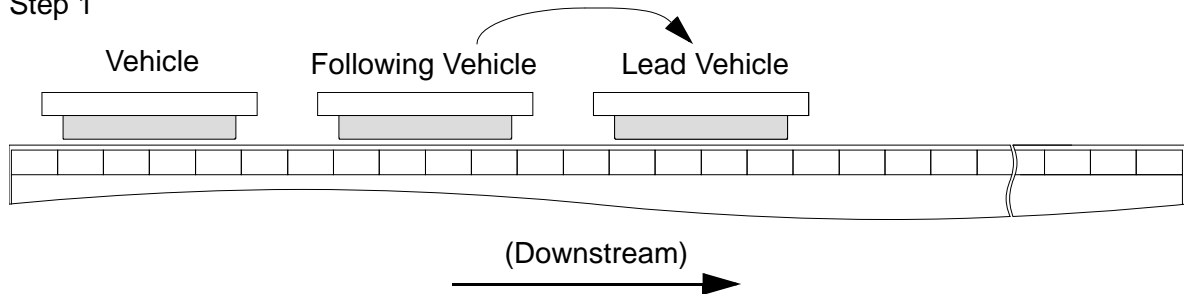
The following vehicle follows behind the lead vehicle at the specified distance.

To add additional vehicles to the platoon, see *Add Additional Vehicles To The End Of A Platoon* [on page 85](#).

Add Additional Vehicles To The End Of A Platoon

Extend a multi-vehicle platoon that moves either downstream or upstream by adding additional vehicles to the end of the platoon. The additional vehicles are added in the direction of motion of the platoon by specifying the last vehicle in the platoon as the vehicle to follow for the vehicle being added. The vehicle being added can be either one vehicle or the lead vehicle in another platoon.

Step 1



Step 2, 3

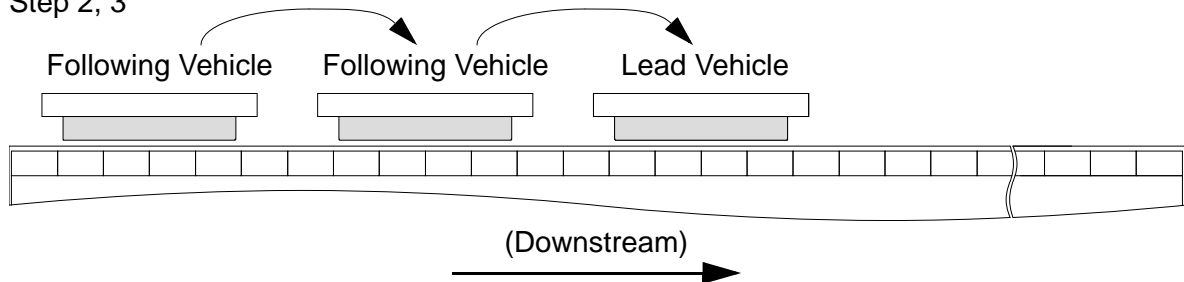


Figure 3-6: Add Vehicles to an Existing Platoon

1. While the platoon is stopped, command the next following vehicle (either one vehicle or the lead vehicle in another platoon) to move into position behind the platoon using an *MMI_vehicle_position_order* tag. The vehicle follow order must specify this distance between the vehicles as the Follow Distance.

The following vehicle moves to its commanded position.

2. Couple the next following vehicle to the platoon by specifying the last vehicle in the platoon as the vehicle to follow. Specify the distance (from [Step 1](#)) between the vehicles and the direction of motion of the platoon using an *MMI_vehicle_follow_order* tag.

The vehicles are linked into a platoon at the specified distance between the vehicles.

3. Command the lead vehicle as required.

All following vehicles follow behind the lead vehicle at their specified distances.

Add A New Leader To A Platoon

Extend a multi-vehicle platoon that moves either downstream or upstream by coupling the lead vehicle in the platoon to a vehicle, which creates a new lead vehicle for the platoon. The platoon is coupled to the vehicle in the direction of motion of the platoon by specifying the vehicle for the current lead vehicle in the platoon to follow. This vehicle can be either one vehicle or the last vehicle in another platoon.

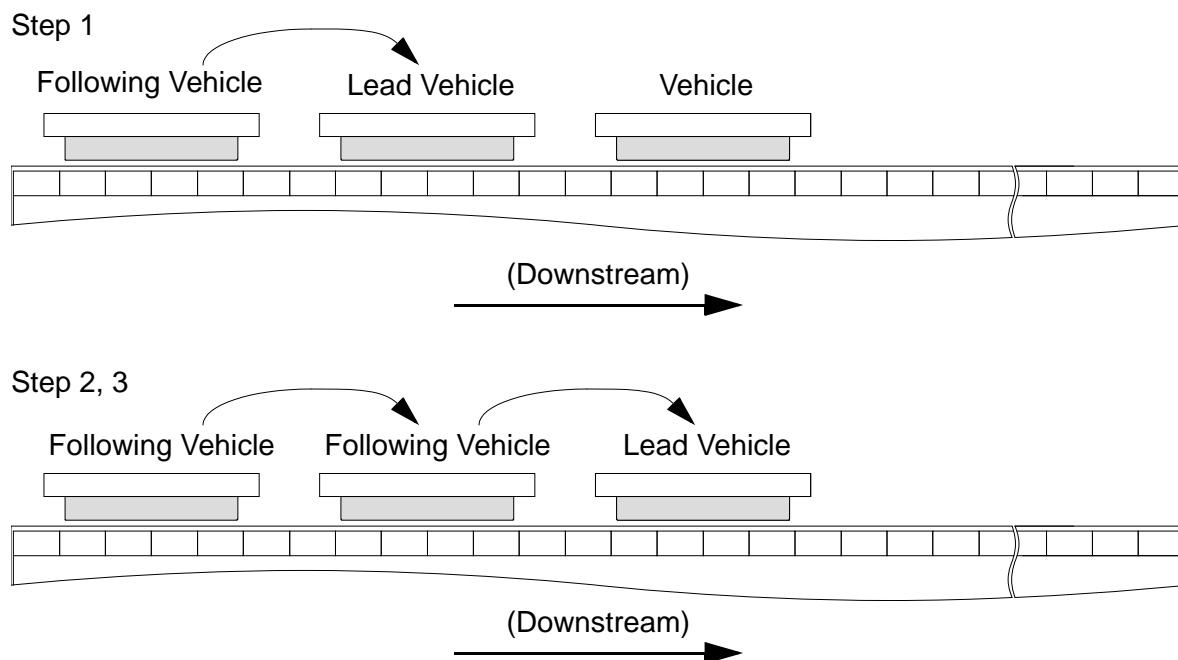


Figure 3-7: Add New Lead Vehicle to a Platoon

1. While the vehicle that becomes the new lead vehicle (or the last vehicle in another platoon) is stopped, command the lead vehicle in the existing platoon to move into position behind that vehicle using an *MMI_vehicle_position_order* tag. The vehicle follow order must specify this distance between the vehicles as the Follow Distance.
The existing platoon follows the lead vehicle as it moves to its commanded position.
2. Couple the platoon by specifying the new lead vehicle (or the last vehicle in a platoon) as the vehicle for the platoon leader to follow. Specify the distance (from [Step 1](#)) between the vehicles and the direction of motion of the platoon using an *MMI_vehicle_follow_order* tag.
The platoon is linked to its new lead vehicle at the specified distance between the vehicles.
3. Command the new lead vehicle as required.
All following vehicles follow behind the lead vehicle at their specified distances.

Uncoupling Platoons

Existing platoons can be uncoupled once the need for platooning is complete. Platoons can be split into their individual vehicles, or they can be split into smaller platoons.

Separate A Platoon Into Vehicles

Once a platoon has been created, it can be separated into uncoupled vehicles when the need for platooning is complete. Remove a vehicle from a platoon by sending it a new vehicle motion command.

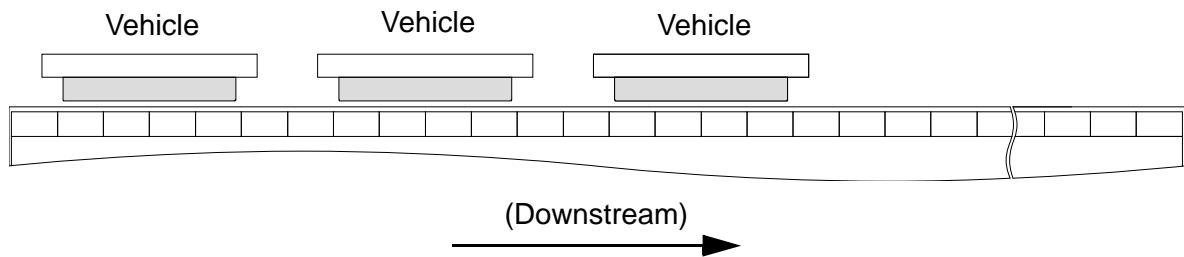


Figure 3-8: Separate A Platoon

1. With the platoon stopped, command vehicles to be uncoupled from the platoon.
 - To separate a vehicle from the vehicle that it is following in the platoon without moving the vehicle, command it to move to its current target as returned in the [MMI_vehicle_status](#) or [MMI_extended_vehicle_status](#) tag. The [MMI_vehicle_position_order](#) tag must specify bidirectional motion.

The vehicle moves to its current target position (no actual motion occurs), which uncouples it from the vehicle that it is following. If there is a vehicle that is coupled to this vehicle, it remains coupled until it is uncoupled.
 - To separate the last vehicle from the platoon and move the vehicle, command it to move to its new location. The [MMI_vehicle_position_order](#) tag must specify motion in the direction away from the platoon.

The vehicle moves to its new position, which uncouples it from the vehicle that it is following.
2. Once the vehicles are uncoupled, command them to their new positions using an [MMI_vehicle_position_order](#) tag.

Each vehicle moves individually as commanded and normal brick-wall headway is maintained between all vehicles.

Separate A Platoon Into Smaller Platoons

Once a platoon has been created, sections can be uncoupled from the original platoon, creating smaller platoons when the need for the larger platoon is complete. To split the platoon, uncouple the lead vehicle of the new smaller platoon from the original platoon by sending it a new vehicle motion command.

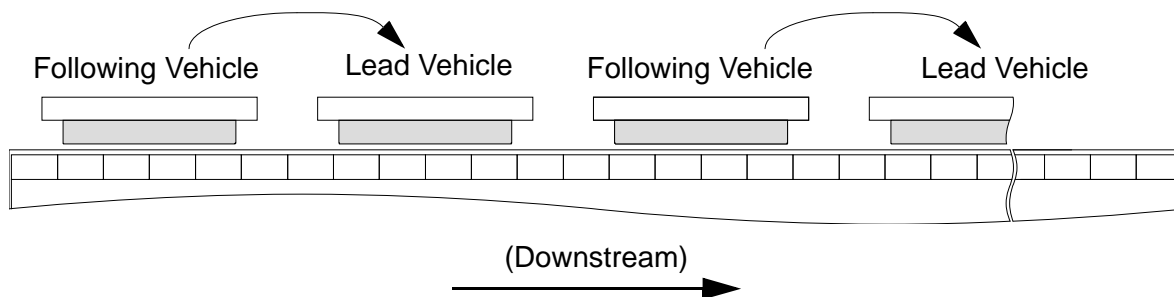


Figure 3-9: Separate A Platoon Into Smaller Platoons

1. With the platoon stopped, determine where to split the platoon into two. Command the vehicle that becomes the lead vehicle for the new platoon to move to its current target as returned in the [MMI_vehicle_status](#) or [MMI_extended_vehicle_status](#) tag. The [MMI_vehicle_position_order](#) tag must specify bidirectional motion.

The vehicle moves to its current position (no actual motion occurs), which uncouples it from the vehicle that it is following. All vehicles that are coupled to this vehicle remain coupled and this vehicle becomes the leader of a new platoon.

2. Once the platoon is separated into smaller platoons, command the lead vehicle of each platoon to its new position using an [MMI_vehicle_position_order](#) tag.

Each platoon moves individually as commanded, normal brick-wall headway is maintained between platoons.

Separate a Vehicle from a Platoon at a Specified Position

When a platoon is created, the follow order for a vehicle can specify a decouple position. When the platoon with the following vehicle with that position specified as part of its follow order approaches that position, the vehicle decouples from the platoon, slows down, and stops at that position. Any vehicles that are following the vehicle that decouples from the platoon continue to follow it after it decouples and stops. The remaining platoon continues to move under its original motion order.

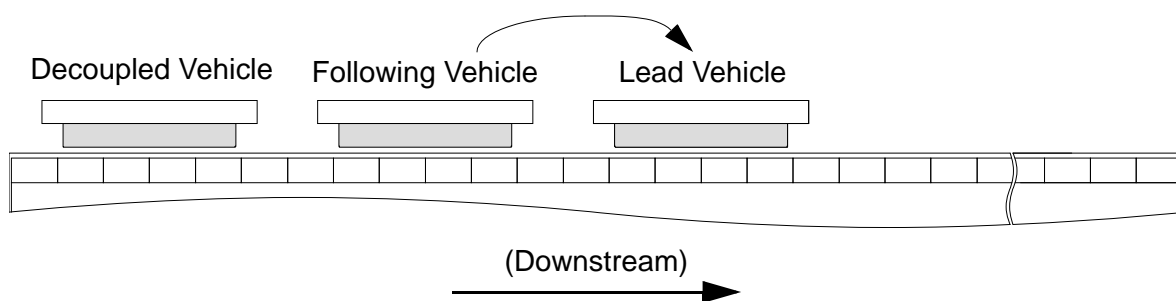


Figure 3-10: Separate a Vehicle from a Moving Platoon

1. While the platoon is stopped, command the following vehicle that is going to decouple from the platoon to move into position behind it (see *Creating Platoons* on page 82). The vehicle follow order must specify this distance between the vehicles as the Follow Distance.
2. Couple the following vehicle to its lead vehicle and specify the distance (from Step 1) between the vehicles and the direction of motion using an *MMI_vehicle_follow_order* tag. This vehicle follow order must also define the decouple path and the final position for this vehicle.
3. Command the lead vehicle to move as required, the following vehicle follows it at the specified distance.

When the platoon approaches the decouple position the vehicle decouples from the platoon, slows down, and stops at that position. Any vehicles that are following the vehicle that decouples from the platoon continue to follow it after it decouples and stops.

Change Platoon Direction

Once a platoon has been created to move in a specific direction, the direction of the platoon can be reversed by stopping it, separating the vehicles in the platoon, and creating a platoon in the opposite direction. The lead vehicle for the new platoon is the vehicle that was the last following vehicle.

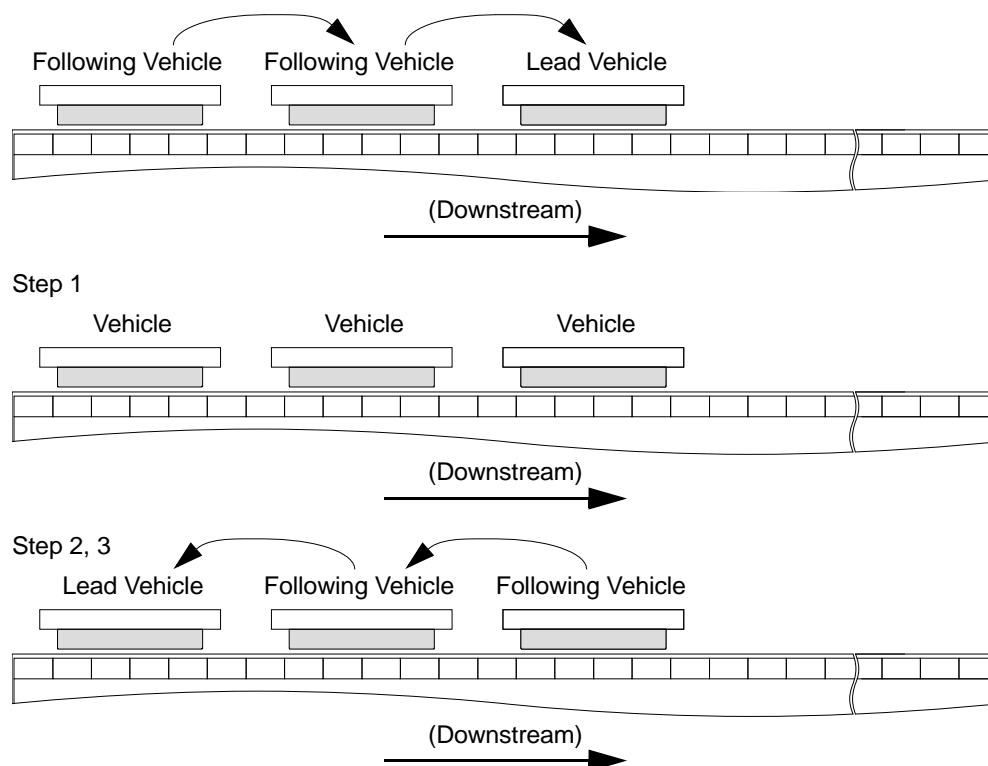


Figure 3-11: Change Platoon Direction

1. With the platoon stopped, command each vehicle to move to its current target as returned in the [MMI_vehicle_status](#) or [MMI_extended_vehicle_status](#) tag. The [MMI_vehicle_position_order](#) tag must specify bidirectional motion.
Each vehicle moves to its current target position (no actual motion occurs), which uncouples it from the vehicle that it is following.
2. Use the last vehicle from the previous platoon as the new platoon leader. Couple the new following vehicle to the new lead vehicle and specify the distance (from [Step 1](#)) between the vehicles and the new direction of motion using an [MMI_vehicle_follow_order](#) tag.
The vehicle is linked into the new platoon as specified.
3. Couple the next following vehicle to the new platoon by specifying the last vehicle in the new platoon as its vehicle to follow. Specify the distance (from [Step 1](#)) between the vehicles and the new direction of motion using an [MMI_vehicle_follow_order](#) tag.
The vehicle is linked into the new platoon as specified.
4. Repeat [Step 3](#) for each additional vehicle in the platoon.
5. Command the new lead vehicle as required, all following vehicles follow it at their specified distances.
The following vehicles follow behind the lead vehicle at their specified distances.

Platooning Across a Node

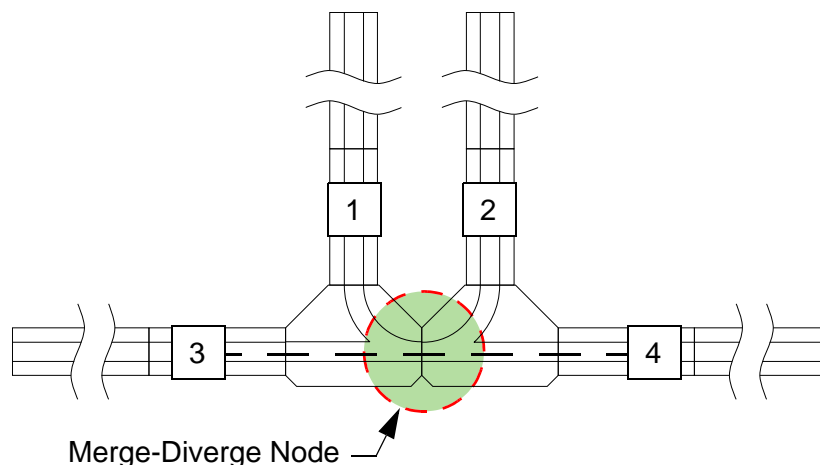


Figure 3-12: Platooning Across a Node

Attempting to couple across a node that is already part of a platoon causes the *MMI_vehicle_follow_order* tag to be rejected.

The example in Figure 3-12 shows a platoon with vehicle 3 coupled to vehicle 4 crossing a node. If a Vehicle Follow Order to couple vehicle 1 to vehicle 2 is sent, the order is rejected since the vehicle 3 - vehicle 4 platoon has locked the Merge-Diverge Node.

Traffic Lights

The Traffic Light feature allows the creation of traffic lights for use during system operation. Traffic lights are used to control vehicle motion at defined positions in the transport system and are associated with specific motor blocks. For examples of traffic light use, see *Traffic Light Control* on page 41 and *Traffic Light Status* on page 54.

When the traffic light is set to green, vehicles are granted permission to move beyond the traffic light location. When the traffic light is set to red, vehicles are not granted permission to enter the motor block where the traffic light is located as shown in Figure 3-13.

There is a limit of one traffic light per motor block and 32 traffic lights per path. The motor block lengths for each motor type are provided in Table 3-1 for reference. Commands to create traffic lights on a motor block where a traffic light is already located or create more than 32 traffic lights on a path are rejected.

IMPORTANT Traffic lights and their settings are not persistent. All traffic lights are created or set using the Traffic Light commands and must be recreated and reset in the following situations.

- Whenever the node controller is restarted or rebooted.
- Whenever any paths where traffic lights are located are reset.

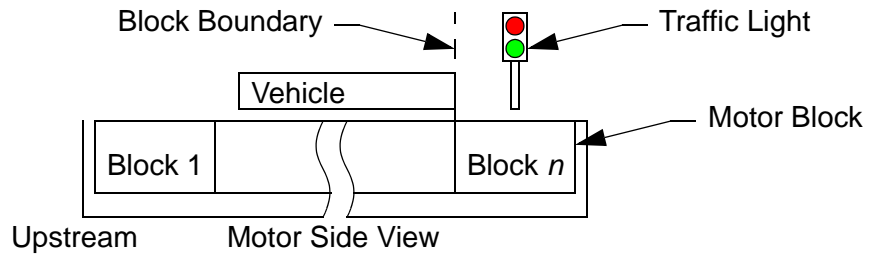


Figure 3-13: Traffic Lights and Motor Blocks

Table 3-1: Block Length

Motor Type	Block Length
MagneMover LITE	16.47 mm
QuickStick	96 mm
QuickStick HT	480 mm

System Monitoring

System monitoring describes a mechanism for retrieving system metrics that can be used to characterize the health of various components within a transport system.

When collecting system metrics on a larger scale and over longer periods of time, such metrics can be used to discover meaningful patterns for improving system operation and early fault detection. By aggregating such data, system metrics facilitate predictive analytics, factory optimization, and efficiency analysis decisions.

The goal of system monitoring is to provide the user with the information to discover, diagnose, and solve installation issues and monitor the operation of running systems.

System Metrics

During normal operation, a host controller can survey the health of system components within a transport system. For example, each motor can be directed to report key system metrics such as board temperature and propulsion voltage continually.

System metrics are enumerated using a unique identifier and organized into simple behavior patterns.

Behavior Patterns

System Metrics are organized into the following behavioral patterns:

1. The Variable pattern represents a simple value, such as constants, configuration, or other internal parameter information. Variables only return the last updated value when read.
2. The Counter pattern represents a positive integer value that increases monotonically until it reaches its maximum value, when it wraps around and starts increasing again from zero. Counters only return the last updated value when read.
3. The Gauge pattern represents a dynamic value that increases or decreases and contains the following properties: last value, minimum value, maximum value, and average value for the requested sampling interval (see [Table 3-2](#)).

Table 3-2: Gauge Pattern Properties

Gauge Property	Description
Last Value	The last value that updated this metric.
Minimum Value	The lowest value that this metric experienced.
Maximum Value	The highest value that this metric experienced.
Average Value	The statistical mean that is calculated by dividing the sum of the updated values by the number of updates for this metric.

When the software updates a Gauge metric, it updates all four properties to maintain statistical consistency. Gauges return all four values when read.

Component Types

System metrics can be collected from any of the primary components comprising a transport system. Each component is grouped according to its assigned component type.

Available component types are described in [Table 4-29 on page 193](#).

Metric Identifiers

A unique, 16-bit number that is called a Metric Identifier (see [Table 4-29 on page 193](#)) is used to identify each metric. Metric identifiers uniquely identify the behavior and characteristics of the metric it references and are consistent across system components.

Metric Instance

Some metrics (for example, communication counters) have multiple unique instances that are present for each of the communication ports. These kinds of metrics use the instance field to

select which port the metric is referencing. For these metrics, the instance field is mandatory, and is used to select the correct occurrence of the value. When a metric consists of a single value and does not require an instance, the instance field value must be zero.

Accessing the System Monitoring Interface

To access System Monitoring commands, the host controller must connect to the high-level controller using TCP port 801 using industry standard socket communication. This port has been designated for System Monitoring commands only and cannot be used for any other purpose.

System Monitoring API

The System Monitoring API is a subscription-based mechanism that allows host controllers to request specific metric data at a custom interval or on demand. The examples that are provided show monitoring of motors.

The host controller can subscribe to a specific system metric to report at the specified interval continuously until stopped with an unsubscribe command. The time interval between samples can range from 1 second to over 18 hours. A command option in the *MMI_sm_subscription_command* tag can be used to instruct the motor to clear the metric data before sampling starts.

Each motor maintains an internal list of active metric subscriptions, up to a maximum of four entries as represented in *Figure 3-14*. Each subscription slot, along with tracking data, maintains its own interval time and subscription options. On reset or power cycle, the motor starts with all subscription slots empty.





	0	--empty--
	0	--empty--
	0	--empty--
	0	--empty--

Figure 3-14: Empty Metric Subscription Slots

Subscribe

The host controller subscribes to a system metric for a motor using its Path ID and Motor ID by sending an SM Subscription Command message to the HLC, which passes it to the appropriate Node Controller. The subscription command also specifies the metric identifier and metric instance, its sampling interval in seconds, and command options.

Each subscription command consumes a subscription slot in the motor. If a subscription command is received when there are no available subscription slots, the motor responds with an SM Subscription Response message with a status of No record available.

When the subscription command is received, the motor stores subscription parameters in one of its available subscription slots. If the subscription command specifies the “Clear Metric Data on Sampling Start” command option, the targeted metric data is cleared at the beginning of each sampling interval.

An SM Subscription Response message is sent to the host controller to signal command failure or command completion. In addition, metric data updates are continuously reported to the host controller using SM Subscription Data Response messages as specified.

The sequence diagram in [Figure 3-15](#) shows a subscription operation to sample motor temperature every 300 seconds (5 minutes). After 300 seconds, a metric data update is sent and repeated every 300 seconds thereafter until an unsubscribe command message is sent to stop it.

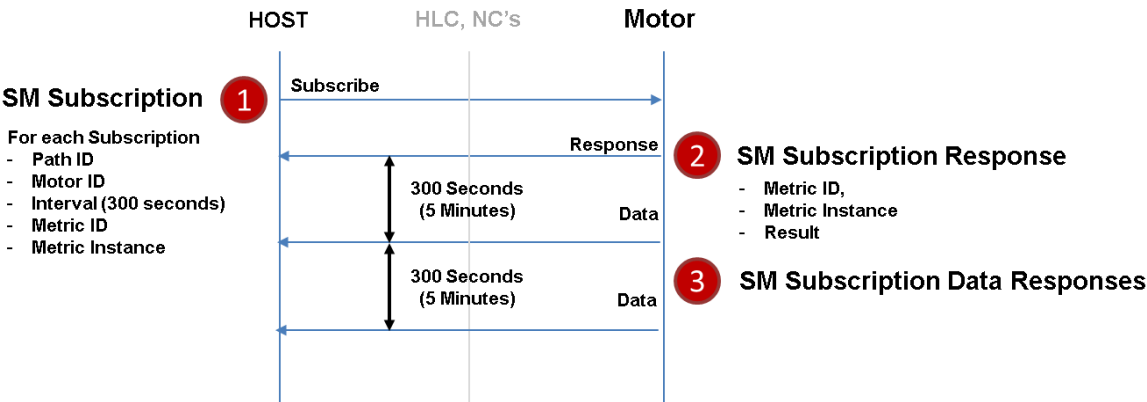


Figure 3-15: Subscribe Sequence Diagram

When the subscription operation completes, the motor has consumed the first subscription slot to monitor board temperature as shown in [Figure 3-16](#).





	300	Temperature
	0	--empty--
	0	--empty--
	0	--empty--

Figure 3-16: Metric Subscription Slot Allocated for Temperature Monitoring

Unsubscribe

When a subscription is no longer needed, an SM Subscription Command message specifying an unsubscribe operation (Subscription Interval = 0) stops metric data updates and clears the motor’s subscription slot for reuse. When completed, the motor responds with an SM Subscription Response message.

The sequence diagram in [Figure 3-17](#) shows an unsubscribe operation.

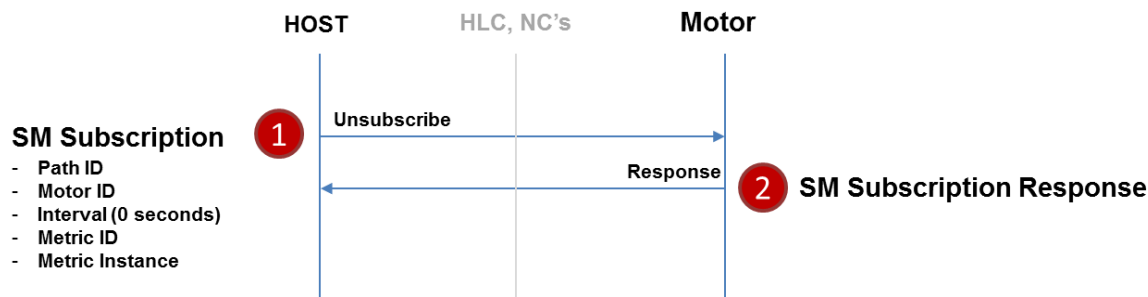


Figure 3-17: Unsubscribe Sequence Diagram

When the unsubscribe operation completes, the motor frees the specified subscription slot for reuse as shown in [Figure 3-18](#).





	0	--empty--
	0	--empty--
	0	--empty--
	0	--empty--

Figure 3-18: Metric Subscription Slot Freed for Reuse

Polling

The host controller can poll metric data on demand using the [MMI_sm_poll_command](#) tag. Requesting data on demand is useful when the interval desired is longer than the maximum interval supported in the subscription command. The SM Poll Command when combined with the “Clear Metric Data on Sampling Start” command option, provides the same statistical consistency as subscription-based monitoring.

Care must be used when using the “Clear Metric Data on Sampling Start” command option when polling metric data with active subscriptions. Clearing metric data using the SM Poll Command affects the consistency of the data being reported in the subscription. When polling, specify that metric data is not cleared, which makes sure that the subscription-based metric data remains consistent.

The sequence diagram in [Figure 3-19](#) shows a poll operation.

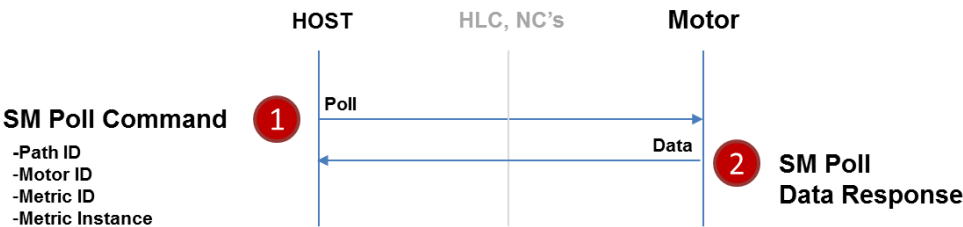


Figure 3-19: Poll Sequence Diagram

Motor Types

The transport systems are a configuration of linear synchronous motors that are placed end-to-end to form long chains or paths. These chains are used to move and position material carriers (vehicles) in a controlled manner at various acceleration/deceleration and velocity profiles while carrying various payloads with high precision. The Host Controller EtherNet/IP Communication Protocol supports the following types of motors.

NOTE: Motors from different product lines cannot be mixed in the same transport system.

MagneMover LITE

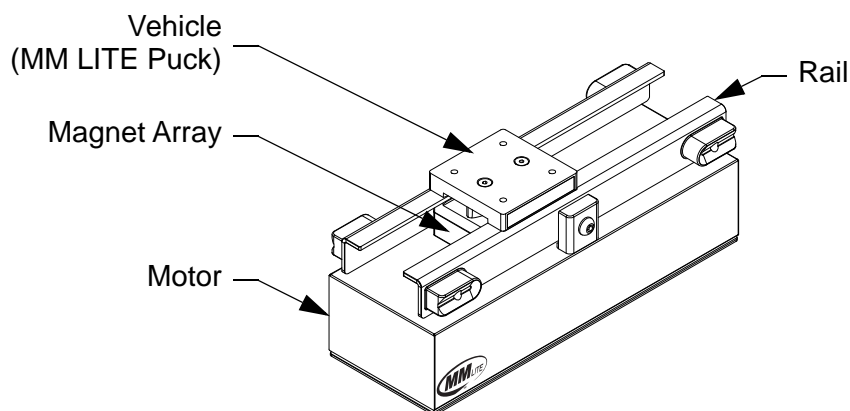


Figure 3-20: MagneMover LITE Motor

The MagneMover LITE motor consists of the motor with internal control electronics and an integrated rail (the 250 mm motor is shown in [Figure 3-20](#)). Motors are mounted end-to-end on a frame with the rails on the motors connected together to create the guideway.

The motor controller for each MagneMover LITE motor is located inside the MM LITE motor. The motor controller monitors the motor and controls vehicle position. The controller also communicates vehicle position and other information to the other motors in the path and to the node controller.

The vehicle that is used with the MagneMover LITE motors is a preconfigured puck with an integrated magnet array that slides on the rails. The motors interact with the magnet array to move and track the puck.

QuickStick (QuickStick 100 and QuickStick 150)

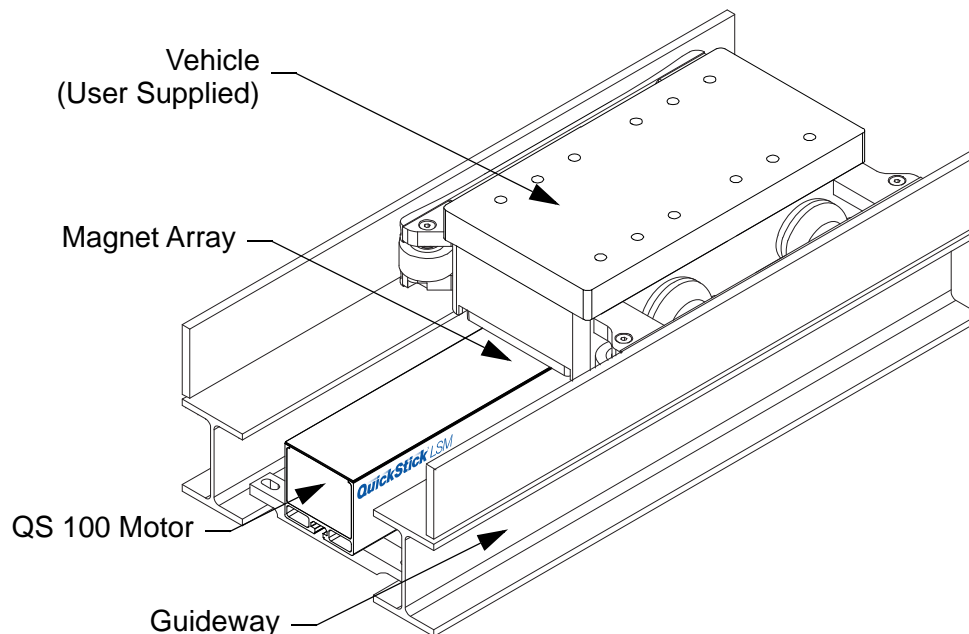


Figure 3-21: QuickStick 100 Motor

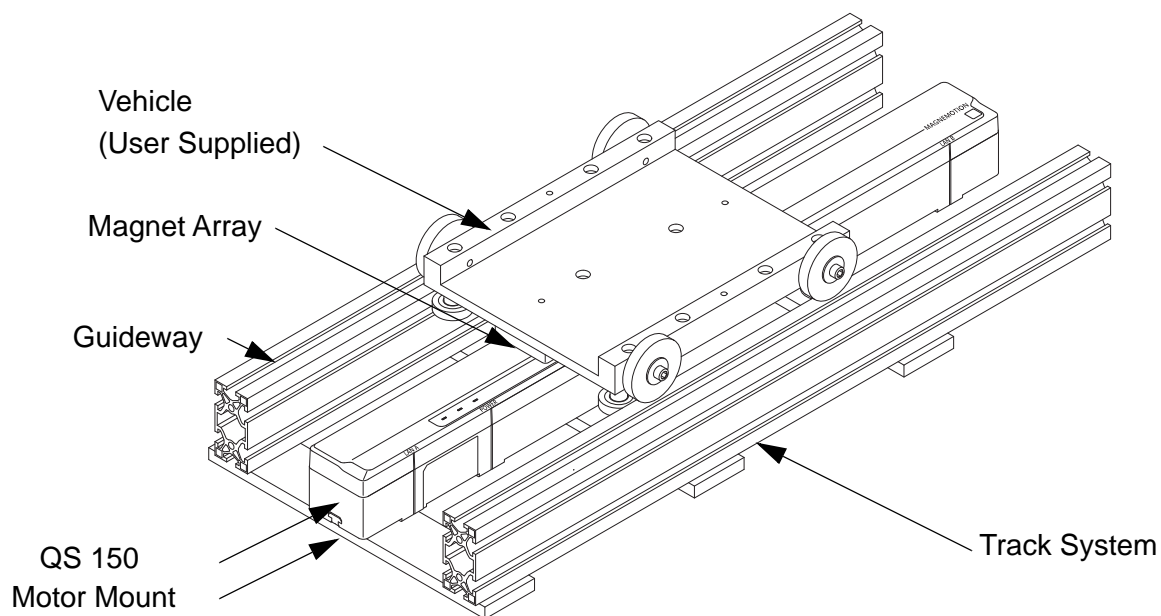


Figure 3-22: QuickStick 150 Motor

The QuickStick motors consist of a motor with internal control electronics. Motors are mounted end-to-end on a user-supplied frame with a user-supplied guideway.

The motor controller for each QuickStick motor is located inside the motor. The motor controller monitors the motor and controls vehicle position. The controller also communicates

vehicle position and other information to the other motors in the path and to the node controller.

The vehicle that is used with the QuickStick motors is user-defined with a magnet array that is mounted on the surface closest to the motors. The vehicle rides on the guideway, which is typically mounted to the frame the motors are mounted on. The motors interact with the magnet array to move and track the vehicle.

QuickStick High Thrust

For definitions of motors, motor controllers, or inverters, see MagneMotion Glossary of Terms, publication [MMI-RM003](#).

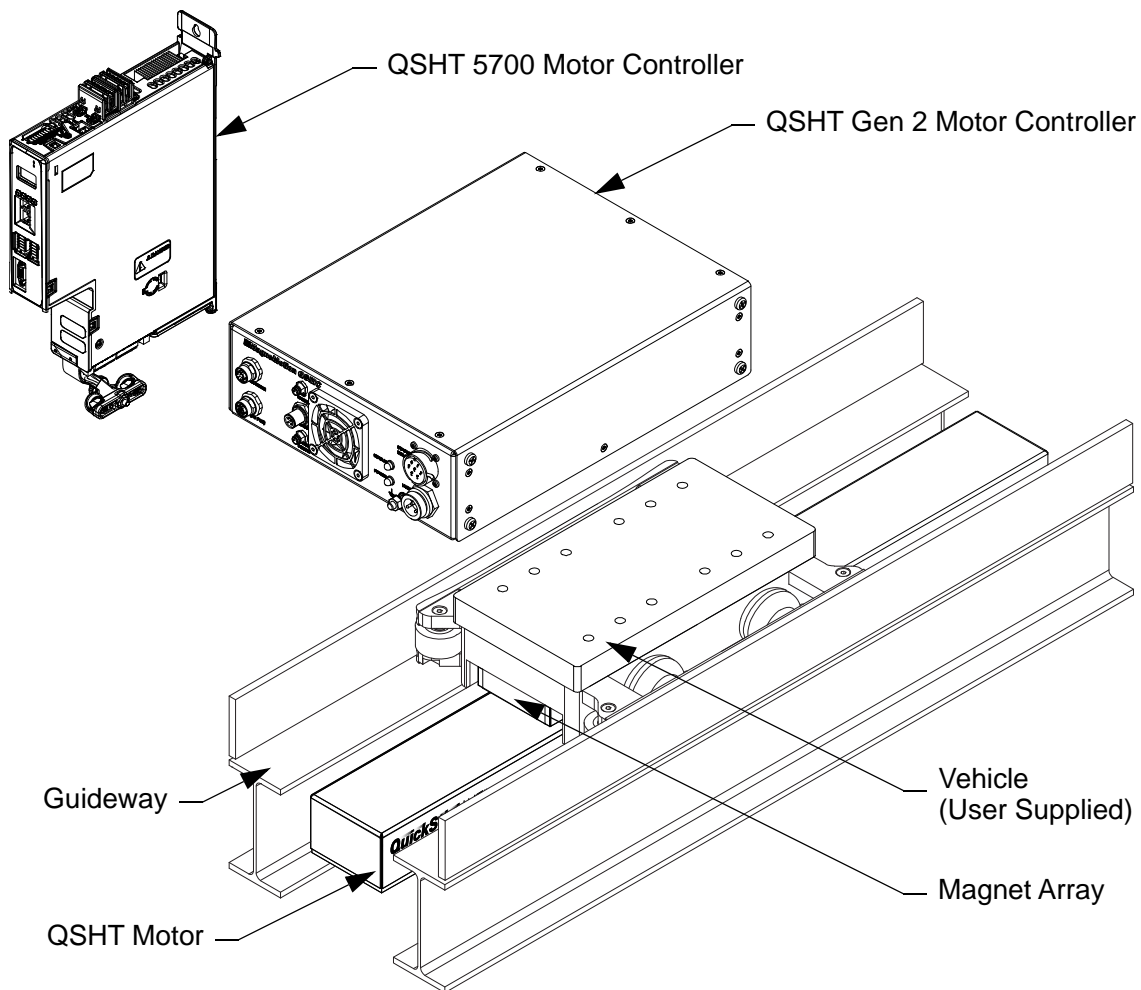


Figure 3-23: QuickStick High Thrust Motor

The QuickStick High Thrust motor consists of the motor with external control electronics (the 1 m motor is shown in [Figure 3-23](#)). Motors are mounted end-to-end on a user-supplied frame with a user-supplied guideway.

The motor controller for each QuickStick HT motor is located external to the QSHT motor. The motor controller monitors the motor and controls vehicle position. The external control electronics for the QSHT motors are the QSHT Gen 2 motor controller and QSHT 5700 motor controller. The motor controllers monitor and power the QSHT motors and control vehicle position. The motor controller also communicate vehicle position and other information to the other controllers in the path and to the node controller.

The vehicle that is used with the QuickStick HT motors is user-defined with a magnet array that is mounted on the surface closest to the motors. The vehicle rides on the guideway, which is typically mounted to the frame the motors are mounted on. The motors interact with the magnet array to move and track the vehicle.

Path Resets and Vehicle Deletion

This section covers reset path and vehicle delete operations. It describes the actions taken by the MMI LSM Transport Control System on path reset and delete vehicle commands. If an application requires path resets on a portion of an transport system, information on issues to be aware of and recommended reset procedures are discussed.

Reset All Paths

When the host controller sends a path reset with the path ID set to zero (0):

- The HLC sends a `reset_path_request` for all paths, causing the Node Controllers to put all paths in reset state.
- During each path's reset process, vehicle records in the motors for that path are deleted.
- Additionally, the HLC detects that all paths are in reset state at the same time. Subsequently, when the HLC detects that all paths are no longer in reset, it sends a `reset_node_request` to all Node Controllers clearing any vehicle node ownership record. Thus, records for all vehicles on the system are deleted.

Reset a Single Path

When the host controller sends a path reset to a specific path ID:

- The HLC sends a `reset_path_request` to the Node Controller responsible for the specified path.
- During the path's reset process, vehicle records in the motors for that path are deleted. Vehicles on the target path that are not involved in a node will be reliably deleted.

Delete a Vehicle

When the host controller sends a delete vehicle command:

- The vehicle records are purged from all motors, nodes, and paths.

Issues with Partial System Resets and Vehicle Deletes

1. If a vehicle is spanning adjacent paths at a node, its records will be deleted from the path being reset, but retained on the adjacent paths if they are not reset as well. This can cause startup to fail to properly locate the offending vehicle, possibly without a startup failure indication.
2. If a vehicle owns a node when the path is reset, the node will continue to be owned by the vehicle even after the vehicle records are deleted from the motors in the path.
3. Relay nodes are not owned but issue 1 above applies at relay nodes.
4. When a vehicle is deleted, permission to occupy the blocks that it owned can be granted to another vehicle. As a result a vehicle under command can collide with an unlocated vehicle. It is important that paths be suspended prior to deleting any vehicles on the path. It is also important that paths reporting unlocated vehicle faults be suspended.

Managing Individual Path Resets

To manage individual path resets, the host controller must use one of the following procedures.

Option 1:

1. Suspend the path that will be reset, termed the target path in this discussion.
2. Suspend any paths that share membership in the target path's upstream and downstream nodes.
3. Delete any vehicles that own the nodes at the end of the target path.
4. If there is a relay node at an end of the target path, delete any vehicle that is located on the opposite side of the relay node within $\frac{1}{2}$ a vehicle length of the relay node.
5. Reset the target path.
6. Resume all suspended paths and ensure that they are all operational.
7. Startup the path that was reset and confirm the deleted vehicles are recovered prior to putting the system back in production.

Option 2:

1. Suspend the path that will be reset, termed the target path in this discussion.
2. Suspend any paths that share membership in the target path's upstream and downstream nodes.
3. If the target path's end is a relay node, also reset the path on the other side of the relay node. This process will create a string of target paths linked by relay nodes but terminated by switch, simple, or terminus nodes.
4. Delete any vehicles that own the nodes at either end of the string of target paths.
5. Reset the string of target paths.
6. Resume the suspended paths and ensure they are operational.
7. Startup the paths that were reset and confirm that the deleted vehicles are recovered prior putting the system back into production.

Node Type Descriptions and Usage

Nodes define the beginning of all paths and the connections between paths. See *MagneMotion System Configurator User Manual*, [MMI-UM046](#), for a detailed description of nodes, node types, and node parameters. There are several behaviors that are related to nodes that must be considered when moving vehicles.

- A vehicle is said to be navigating the node (or owning the node) when the node is within the motor permissions that are required for brick-wall headway for the vehicle (see *Safe Stopping Distance* [on page 72](#)).
- Vehicles are considered to have left the node (released ownership) once the center of the vehicle is 1/2 vehicle length plus 1/4 motor cycle past the node (end of the motor) in the direction of travel for the vehicle. These clearance distances are shown in [Table 3-1](#) as the distance from the end of the motor to the end of the vehicle.

Table 3-1: Node Clearances

Motor Type	Clearance Distance
MagneMover LITE	12.35 mm
QuickStick	12 mm
QuickStick HT	30 mm

- Nodes create different loads on the node controllers depending on the type of node. Multi-port nodes types (Relay, moving path, merge, or diverge nodes) are nodes that

connect multiple paths and require multiple connections to the node controller in serial systems. In Ethernet systems, these connections are managed via the network and do not require a physical connection. These types of nodes result in the same amount of load on the node controller even if they have different numbers of paths that are connected to the node controller. Single-port node types (Simple, Overtravel, Terminus, and Gateway) present a negligible load. When using two Terminus Nodes to pass vehicles between paths, they count as one multi-port node due to the increased communication load. See the *MagneMotion Node Controller Hardware User Manual*, [MMI-UM013](#), for additional information about loading.

- Node clearance distances and entry gate distances are used with switching nodes. These configurable anti-collision parameters define the clearance for vehicles entering and exiting the nodes. These clearances are used to avoid collisions with other vehicles on adjoining paths or with any mechanisms that are related to the node.

Relay nodes allow a vehicle to move from the end of a path to the beginning of another path. This motion can either be on one path creating a loop, or from one path to another extending the length of the string of motors.

Switching nodes allow a vehicle to move from the end of the path it is on to the beginning of another path through a switch mechanism. Switches in MagneMover LITE systems are preconfigured motor modules that provide either a diverge from one path to two paths or a merge from two paths to one path. The MM LITE™ switch modules use an integrated switch mechanism that does not require any user action as described in *Merge and Diverge Nodes* [on page 120](#). Switches in QuickStick systems require an external switch mechanism that the host controller controls as described in *Moving Path Node* [on page 127](#).

The transport systems support the following types of nodes. All node types support bidirectional motion through the node. Not all systems support all node types. Node types define their use and are presented in this section in order from least to most complex.

Simple Node

A Simple Node is used to begin a standalone path. This type of path is not connected to anything else at the upstream end and no vehicles enter or exit through the node. The upstream end of the path where the node is located requires one connection to the node controller. See [Figure 3-24](#), where the shaded circle represents the Simple Node. Paths can also start from other node types.

Support for this node type is provided in:

- MagneMover LITE transport systems.
- QuickStick transport systems.
- QuickStick HT transport systems.

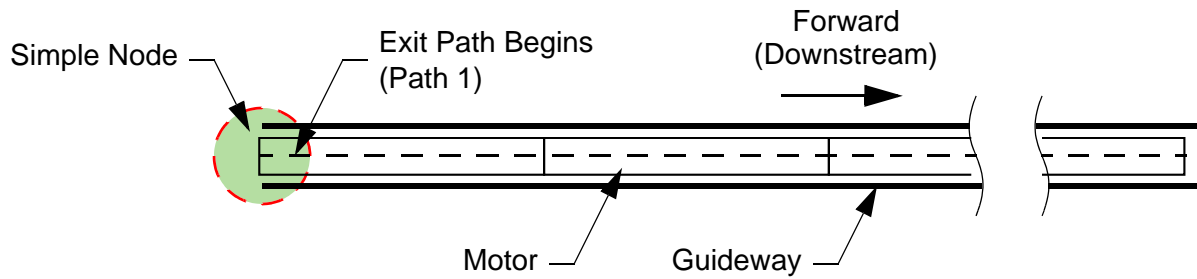


Figure 3-24: Simple Node

Node Operation

- Normal vehicle clearances apply.

Relay Node

A Relay Node is used to provide a simple connection where the end of one path connects to the start of another path. Relay Nodes are typically used to create a loop or a long continuous path. The two path ends require two connections to the same node controller. This type of node is used to break up large paths, create a simple loop, or separate E-stop/Interlock zones. See [Figure 3-25](#), where the shaded circle represents the Relay Node.

Support for this node type is provided in:

- MagneMover LITE transport systems.
- QuickStick transport systems.
- QuickStick HT transport systems.

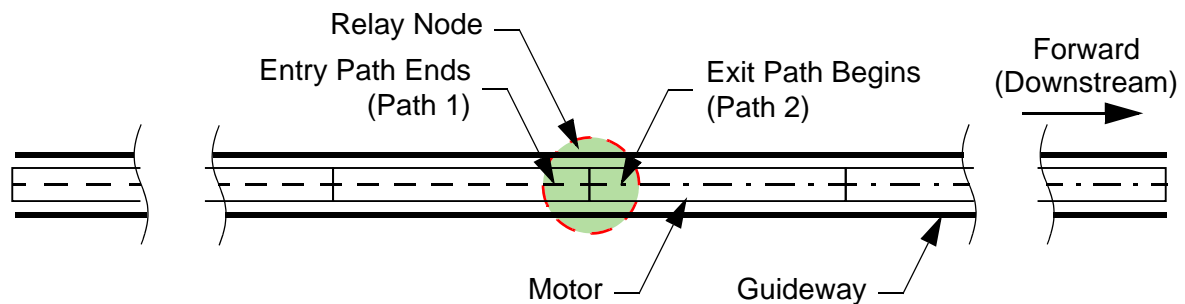


Figure 3-25: Relay Node

Node Operation

- Normal vehicle clearances apply across the node.
- Vehicles moving in the same direction can queue across the node.
- The move profile for the vehicle is maintained across the node so the vehicle crosses the node at a consistent velocity and acceleration.

Terminus Node

A Terminus Node is used on a path where vehicles want to enter or exit a path through a blind handoff. Terminus Nodes can be placed at either the upstream or downstream end of the path. See [Figure 3-26](#), where the shaded circle represents the Terminus Node at the downstream end of the path. This section describes how to move a vehicle on or off the transport system through a Terminus Node, including all handshaking.

Handshaking between the HLC and the host controller is used to insert and extract vehicles on a path from equipment separate from the transport system. The host controller sets and clears signals that are affiliated with a Terminus Node using the [MMI_terminus_node_command](#) message. The last recorded signal state is viewed by examining the [MMI_node_status](#) entry for the appropriate Node ID. The [MMI_node_status](#) tag provides the current state of the signals.

NOTE: When a Terminus Node output state changes, the node controller automatically updates the appropriate [MMI_node_status](#) entry.



PINCH/CRUSH HAZARD: Moving mechanisms (vehicles) have no obstruction sensors.

Do not operate the transport system without barriers in place or personal injury could result in the squeezing or compression of fingers, hands, or other body parts between moving mechanisms.

Support for this node type is provided in:

- MagneMover LITE transport systems.
- QuickStick transport systems.
- QuickStick HT transport systems.

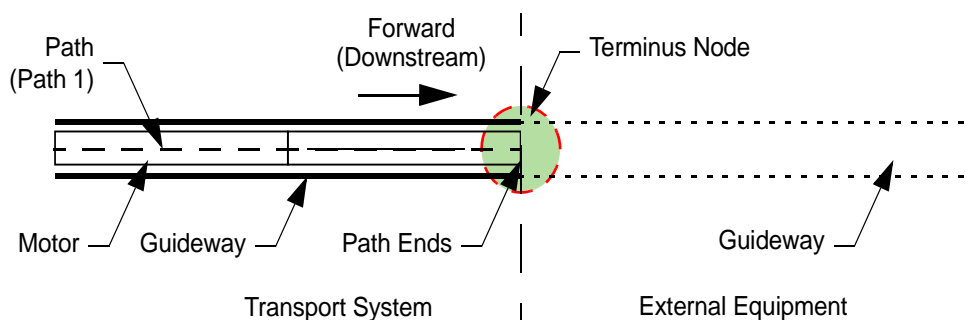


Figure 3-26: Terminus Node

Node Operation

- Normal vehicle clearances apply on the transport system side of the node.
- Vehicles moving in the same direction can queue up to the node.

- Vehicles are not allowed to enter the node unless there is a full vehicle length of space free at the end of the path where the node is located.
- The move profile for the vehicle is maintained across the node so the vehicle transfers from the node to the user-supplied equipment at a consistent velocity and acceleration. Once the vehicle exits the transport system, the motors no longer provide any motive force or tracking of the vehicle.

Handshake for Vehicle Entry onto the Transport System

This handshake is used for inserting a vehicle onto a transport system path at a Terminus Node. The handshake is designed to let only one vehicle at a time be inserted at a Terminus Node without risk of collision (simultaneous insertions at other termini are allowed). Once the vehicle enters the Terminus Node, it stops once half the vehicle is on the motor unless there is a new vehicle command. Only issue the new command after the **ENTRY_ALLOWED** signal goes High.

Signal Descriptions

Signal Name	Input/Output	Description
ENTRY_CLEAR	Output	Signal from the high-level controller that, last time it was checked, sufficient room to insert a vehicle at the Terminus Node exists. When the ENTRY_CLEAR signal is high, entry permission has not been granted, it is only a signal that the entry space was clear the last time it was checked. If a vehicle moves into this space in the meantime, or an error occurs, ENTRY_ALLOWED does not go high when entry is requested.
ENTRY_REQUESTED	Input	Signal from the host controller that a vehicle on the external equipment is ready for entry.
ENTRY_ALLOWED	Output	Signal from the high-level controller that motion permission has been acquired for the vehicle and the vehicle can enter.

Entry Handshake

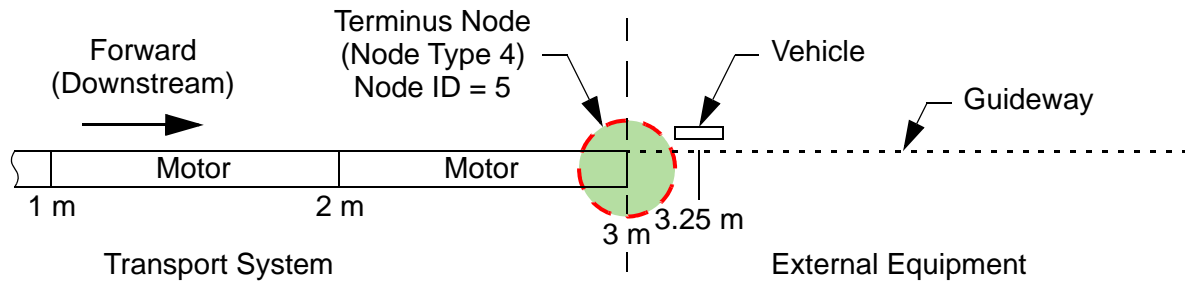


Figure 3-27: Terminus Node Entry Example

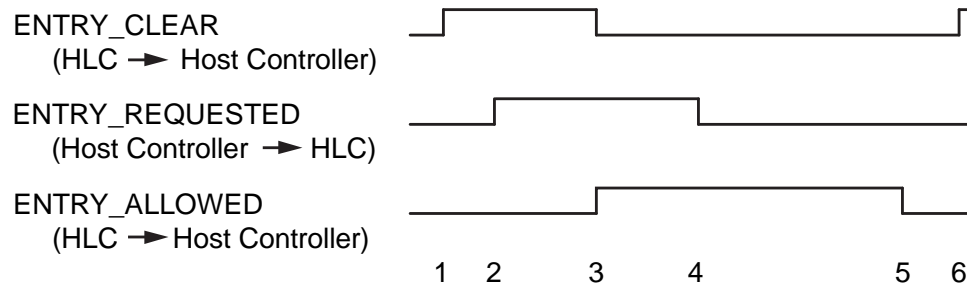


Figure 3-28: Entry Handshake Timing

1. Whenever the entry area is clear the high-level controller sets the **ENTRY_CLEAR** signal High (see [Figure 3-27](#)).

NOTE: If the status of the entry area changes, the HLC changes the **ENTRY_CLEAR** signal.

The host controller checks the status of the Terminus Node (see [Figure 3-27](#)) by referencing the appropriate index from the [MMI_node_status](#) tag. The status shows that the node is clear for vehicle insertion (**ENTRY_CLEAR** = High, all other bits = Low).

vehicle_id	0
node_type	4
terminus_signals	0001 0000
requested_position	0
reported_position	0
device_status	0

2. The host controller sets the **ENTRY_REQUESTED** signal High using the [MMI_terminus_node_command](#) to indicate that it has a vehicle ready to enter.

NOTE: The command is rejected if a vehicle ID that is already in use is assigned. Therefore, Rockwell Automation recommends not specifying a vehicle ID and using the ID assigned by the system.

node_id	5
signal_number	0
signal_level	1
command_count	5483
vehicle_id	0
active_flag	0x01

3. The HLC sets the **ENTRY_CLEAR** signal Low and sets the **ENTRY_ALLOWED** signal High to indicate permission for the vehicle to enter the path at the Terminus Node. This status shows that space has been reserved on the motor at the Terminus Node for the vehicle.

The host controller checks the status of the Terminus Node by referencing the appropriate index from the [MMI_node_status](#) tag. The status shows that the node is clear and a vehicle ID has been assigned for the incoming vehicle (**ENTRY_CLEAR** = Low, **ENTRY_ALLOWED** = High, and **ENTRY_REQUESTED** = High).

vehicle_id	3
node_type	4
terminus_signals	0010 0001
requested_position	0
reported_position	0
device_status	0

The HLC is now ready to accept a vehicle move command for the vehicle.

4. The host controller acknowledges entry permission has been granted by setting the **ENTRY_REQUESTED** signal Low using the [MMI_terminus_node_command](#).

node_id	5
signal_number	0
signal_level	0
command_count	5483
vehicle_id	3
active_flag	0x01

If desired, the host controller sends an *MMI_vehicle_position_order* to move the vehicle to a position on the path and out of the Terminus Node. If a vehicle position

command is not sent, the vehicle stops as soon as it fully enters the transport system at the Terminus Node.

vehicle_id	3
path_id	3
position	0.125
velocity_limit	1.0
acceleration_limit	1.0
order_number	42
flags_and_direction	02
active_flag	0x01

The vehicle is allowed to enter (the host controller commands the remote equipment to insert the vehicle onto the transport system).

NOTE: If a vehicle move command is sent before the vehicle is inserted, the vehicle starts moving under that command as soon as it is inserted. To have a move command queued this way, requires specifying the vehicle ID when the *MMI_terminus_node_command* is sent in [Step 2](#).

While the vehicle is in motion through the node, the host controller can check the status of the Terminus Node by referencing the appropriate index from the *MMI_node_status* tag. The status shows that the node is clear (**ENTRY_CLEAR** = Low, **ENTRY_ALLOWED** = High, and **ENTRY_REQUESTED** = Low).

NOTE: In simulation mode, when **ENTRY_REQUESTED** is changed to Low by the Host Controller, the receiving Path takes control of the mover; the mover is considered to have arrived. The **ENTRY_ALLOWED** bit is changed to Low by the HLC; the mover is received.
On a physical track, when the Host Controller sets **ENTRY_REQUESTED** to Low, the **ENTRY_ALLOWED** remains High until the mover is received.

vehicle_id	3
node_type	4
terminus_signals	0010 0000
requested_position	0
reported_position	0
device_status	0

- Once the vehicle clears the node, the HLC confirms that the entry request is over (**ENTRY_REQUESTED** is Low) and the vehicle has entered by setting the **ENTRY_ALLOWED** signal Low.

Upon completion of the vehicle command, the HLC updates the [MMI_vehicle_status](#) tag, which shows the vehicle has completed its move.

path_id	3
dest_path_id	0
position	0.125
commanded_position	0.125
velocity	0
dest_station_id	0
command	0x00
flags	0000 0001

6. The host controller must wait for the **ENTRY_ALLOWED** signal to be Low and the **ENTRY_CLEAR** signal to be High before sending another entry request.

Handshake for Vehicle Exit from the Transport System

This handshake is used for removing a vehicle from a transport system path at a Terminus Node. The handshake is designed to let only one vehicle at a time leave a path through a Terminus Node without risk of collision (simultaneous exits at other termini are allowed). The vehicle exiting at the Terminus Node must have been given a command to move beyond the end of the path in order for the vehicle to exit. The vehicle then waits on the motor for the appropriate handshake to take place before exiting. The destination in the command must place the entire vehicle (not just the center of the vehicle) beyond the end of the motor.

Signal Descriptions

Signal Name	Input/Output	Description
EXIT_ALLOWED	Input	Signal from the host controller that the external equipment is ready to receive a vehicle from the Terminus Node (the motor is allowed to push the vehicle off the path).
EXITING	Output	Signal from the high-level controller that motion permission has been acquired for the vehicle and the vehicle is exiting.

Exit Handshake

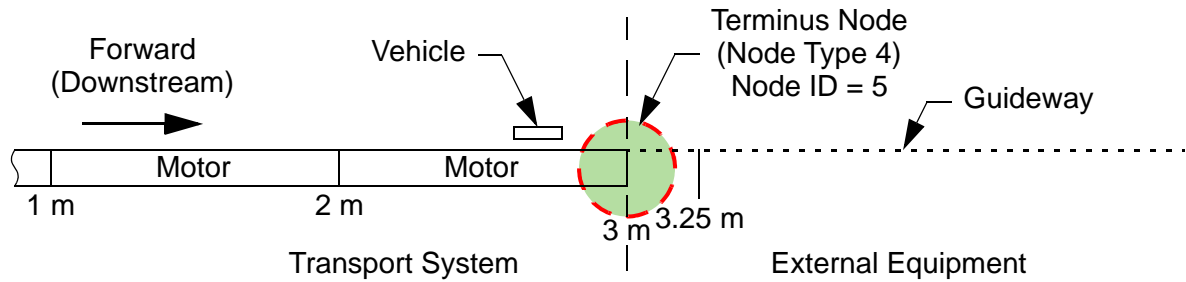


Figure 3-29: Terminus Node Exit Example

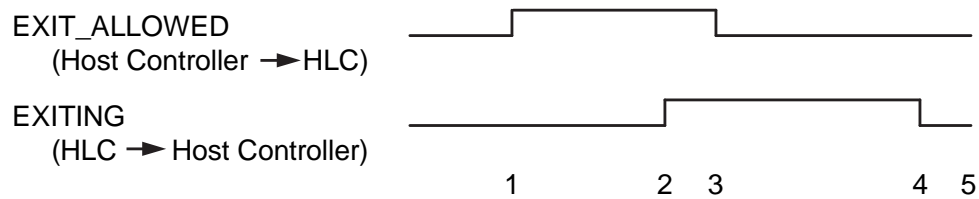


Figure 3-30: Exit Handshake Timing

1. The host controller sets **EXIT_ALLOWED** High for the Terminus Node (see [Figure 3-29](#)) using the [MMI_terminus_node_command](#) to signal that it is ready for a vehicle to exit. This signal specifies space is available for a vehicle on the remote equipment beyond the Terminus Node of a transport system.

node_id	5
signal_number	1
signal_level	1
command_count	5483
vehicle_id	0
active_flag	0x01

The host controller sends an *MMI_vehicle_position_order* to move the vehicle past the end of the path. The minimum commanded distance past the end of the motor must be at least one vehicle length.

vehicle_id	3
path_id	3
position	-1.0
velocity_limit	1.0
acceleration_limit	1.0

order_number	42
flags_and_direction	01
active_flag	0x01

NOTE: This vehicle command never completes (*MMI_vehicle_order_status* will not return a "Command Complete" status) as the vehicle has left the transport system.

2. The HLC notes that the **EXIT_ALLOWED** signal is High and has a vehicle that has begun exiting and sets the **EXITING** signal High.

The host controller checks the status of the Terminus Node by referencing the appropriate index from the *MMI_node_status* tag. The status shows vehicle #3 is in the node, **EXIT_ALLOWED** = High, and **EXITING** = High).

node_id	5
vehicle_id	3
node_type	4
terminus_signals	0100 0010
requested_position	0
reported_position	0
device_status	3

3. The host controller acknowledges that a vehicle is leaving (**EXITING** signal has gone High) by setting the **EXIT_ALLOWED** signal Low using the *MMI_terminus_node_command*. The host controller commands the remote equipment to remove the vehicle from the transport system.

node_id	5
signal_number	1
signal_level	0
command_count	5485
vehicle_id	4
active_flag	0x01

4. The HLC confirms that the **EXIT_ALLOWED** signal is Low and the vehicle has left by setting the **EXITING** signal Low.
5. Once the vehicle has been moved out of the node and onto the remote equipment, the host controller checks the Terminus Node status using the *MMI_node_status* tag. The

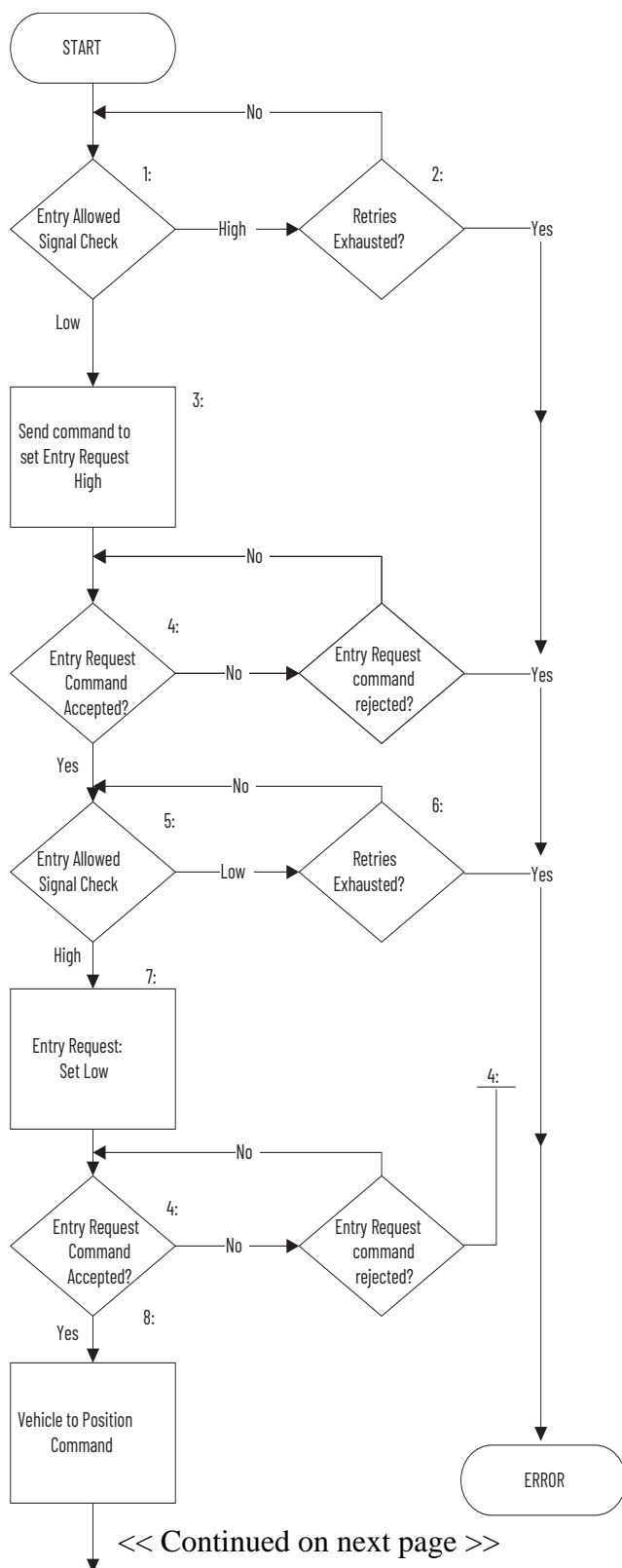
status shows no vehicle in the node, **EXIT_ALLOWED** = Low, and **EXITING** = Low).

node_id	5
vehicle_id	0
node_type	4
terminus_signals	0000 0000
requested_position	0
reported_position	0
device_status	3

6. The host controller can now set **EXIT_ALLOWED** High again when the user-supplied equipment has space for another vehicle.

Example of Two Back to Back Terminus Nodes

One terminus node is receiving a vehicle from another terminus node.



Explanation of steps in the flow chart:

1: Entry Allowed (high) indicated that permission has been given for a vehicle may enter the node.

2: For Retries Exhausted you need to determine how long to wait for the node to be clear. Entry allowed will be high when a vehicle has been given permission to enter and while the vehicle is in the process of entering. Entry Allowed will be low when a vehicle has completed the entry into the node.

3: This is the command from the Host to the HLC to set the Entry Request bit high for the terminus node.

4: This is the standard response for command accepted. This should always be received. If there is an error in the format of the command then it will be rejected.

5: Entry Allowed (high) indicated that permission has been given for a vehicle may enter the node.

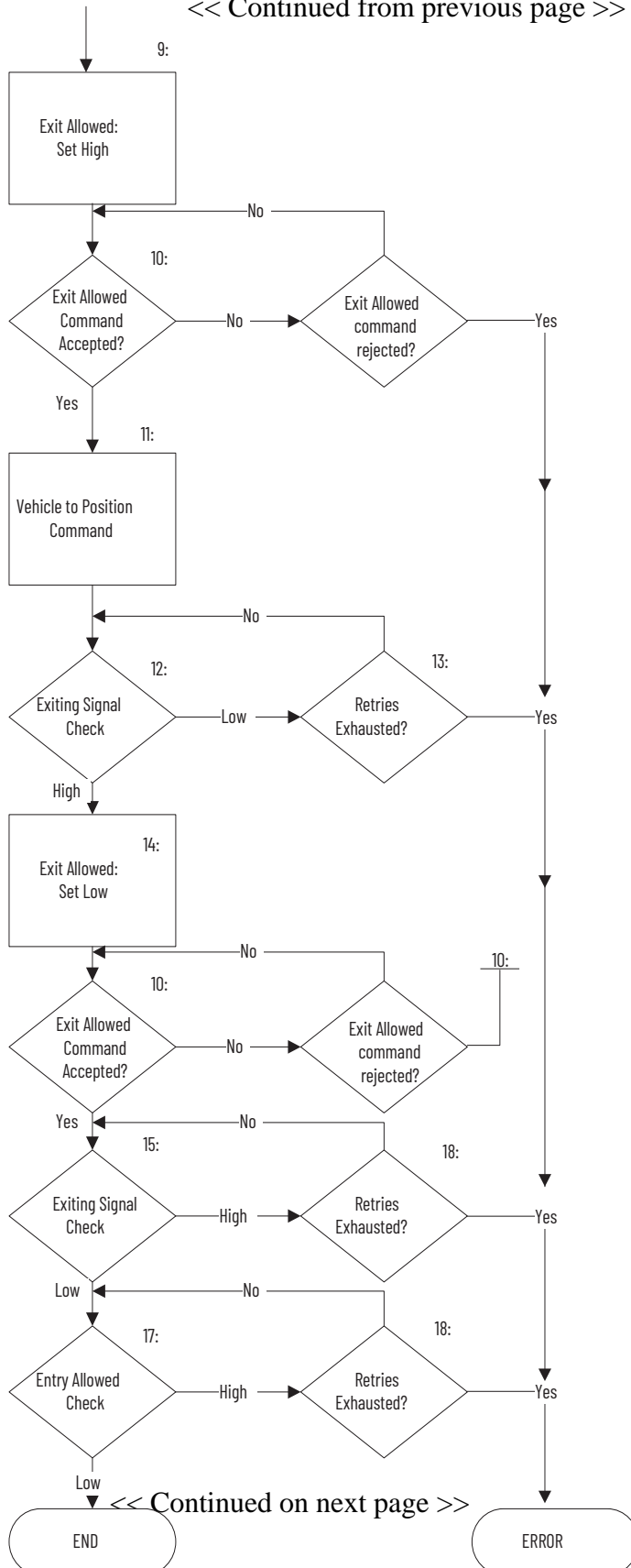
6: For Retries Exhausted you need to determine how long to wait for the node to be clear. Entry allowed will be high when a vehicle has been given permission to enter and while the vehicle is in the process of entering. Entry Allowed will be low when a vehicle has completed the entry into the node.

7: This is to confirm that Entry_Allowed was set high by the system setting Entry_Requested to low.

This completed the set-up for a vehicle to enter a terminus node.

8: This command is to order the vehicle which is entering the terminus node to it's target position. This can be done so when the vehicle starts to enter the terminus node it has a destination. Otherwise the vehicle will stop once entered until a move command is issued.

<< Continued from previous page >>



<< Continued on next page >>

Explanation of steps in the flow chart (cont.):

9: This is the host commanding the terminus node that there is space ready for the vehicle to move beyond the terminus end of the motor. This is the host giving permission that it is OK for the vehicle to leave.

10: This is the standard response for command accepted. This should always be received. If there is an error in the format of the command then it will be rejected.

11: This command is sending the vehicle off the end of the terminus EXIT node.

12: This will be high while the vehicle has permission to move, is moving or still owns the node.

13: For Retries Exhausted you need to determine how long to wait for the node to be clear. Exiting will be high when a vehicle has been given permission to exit and while the vehicle is in the process of exiting. Exiting will be low when a vehicle has completed the exit from the node.

14: This is the host acknowledging that Exiting has been set high and the vehicle has permission to leave.

15: This is to check if the vehicle has completed the exit and no longer owns the node.

16: This is a timer to see if there is a reason that the vehicle is not exiting. You should also check the vehicle status to be sure it is moving.

17: This confirms that the vehicle has finished entering the Entry Allowed node.

18: For Retries Exhausted you need to determine how long to wait for the node to be clear. Entry allowed will be high when a vehicle has been given permission to enter and while the vehicle is in the process of entering. Entry Allowed will be low when a vehicle has completed the entry into the node.

Gateway Node

Gateway Nodes allow a vehicle to move from one Control Group to another Control Group within a transport system. Control Groups are typically sections of a larger transport system. Handshaking through the local subnet is used by the node controllers in each Control Group responsible for the Gateway Node to pass vehicles from Control Group to Control Group.

A Gateway Node is used to connect a path in one Control Group to a path in another Control Group within a larger transport system. Each Control Group has separate high-level controllers and can have separate host controllers. The end of the path where the node is located requires one connection to the node controller. See [Figure 3-31](#), where the shaded circles represent the Gateway Nodes.

NOTE: The Gateway Nodes definitions in the Node Controller Configuration File for each Control Group must reference the matching Gateway Node in the other Control Group to achieve proper vehicle transfer.

Only two Gateway Nodes can be configured per node controller. One at an upstream path end and one at a downstream path end. Both node controllers that are connected to the Gateway Node must be connected to the same network subnet.

The HLC can run on the same hardware as the node controller (for example, an NC LITE), but may need to be on another NC depending on the communications load.

If the vehicle ID of the vehicle entering a Control Group through a Gateway Node exists in that Control Group, entry is refused.



PINCH/CRUSH HAZARD: Moving mechanisms (vehicles) have no obstruction sensors.

Do not operate the transport system without barriers in place or personal injury could result in the squeezing or compression of fingers, hands, or other body parts between moving mechanisms.

Support for this node type is provided in:

- MagneMover LITE transport systems.
- QuickStick transport systems.
- QuickStick HT transport systems.

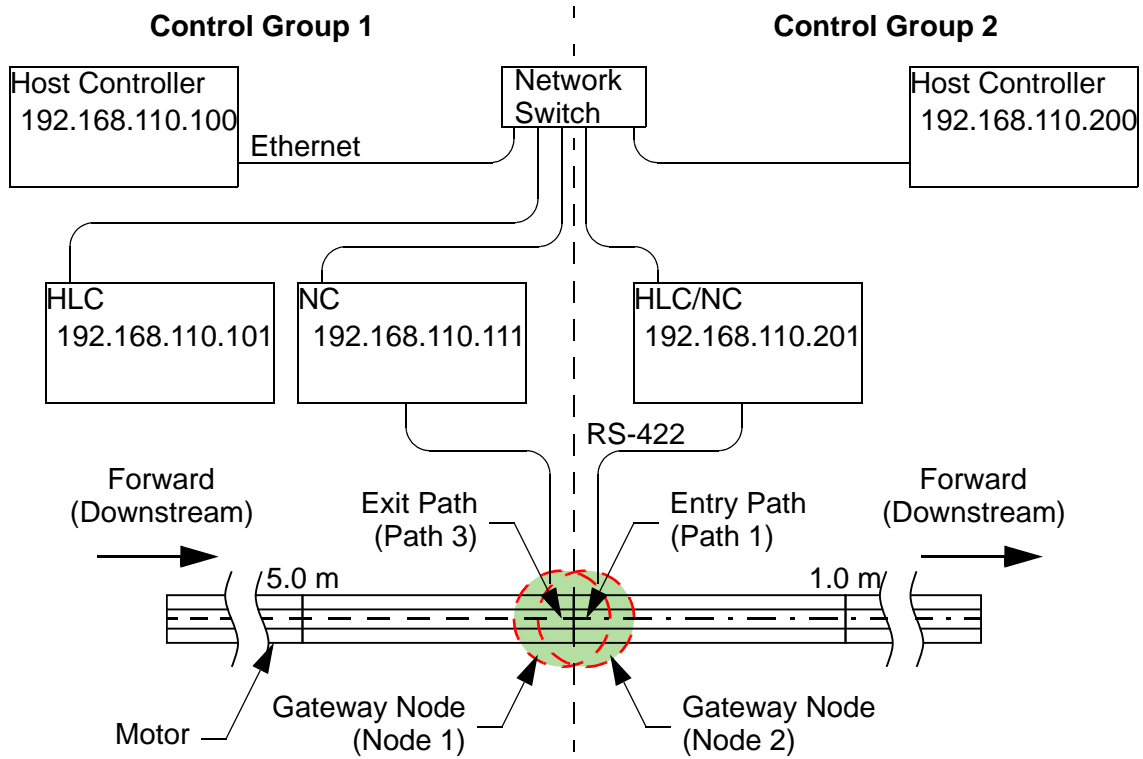


Figure 3-31: Gateway Node

Node Operation

- Normal vehicle clearances apply across the node.
 - As a vehicle approaches the Gateway Node, the node controller requests permission from the target HLC to enter the other Control Group. As long as the path in the target Control Group is operational and no duplicate vehicle ID exists, entry is granted.
 - If entry is granted, the vehicle continues to move through the node.
 - If entry is not granted, the vehicle decelerates and stops before crossing the node. The leading edge of the vehicle stops at the node boundary.
- Vehicles moving in the same direction can queue up to and through the node.
 - If multiple vehicles are sent to the node, vehicles can queue before, across, and through the node. The queuing is similar to a Relay Node.
- The move profile for the vehicle is maintained across the node so the vehicle crosses the node at a consistent velocity and acceleration.
 - The vehicle ID is maintained across the Gateway Node if the sending and receiving node controllers are configured in the same HLC control group.
 - If the destination is just beyond the node, the vehicle decelerates to follow the move profile in the same manner as on any path.

- The default destination for the Gateway Node is defined in the Node Controller Configuration File for entering vehicles.
 - The Node Controller Configuration Files for the Control Groups that share a Gateway Node must have default destination paths and positions defined. These positions become the default locations for any vehicle that enters the node in either direction.
 - The host controller can supersede the default position with a vehicle motion command as soon as the vehicle record is detected.
- Node controllers coordinate exit and entry.
 - The host controller does not control the access through the Gateway Node, the node controllers in the two Control Groups handle all access handshaking. Using the node controllers to handle all handshaking eliminates the need to have entry/exit requests coordinated by the host controllers when a vehicle moves through the pair of Gateway Nodes.
- The host controller detects the Gateway Node ownership when a vehicle enters it via the *MMI_node_status* tag, which provide notification that a vehicle is entering.
 - Once target requests are granted, a vehicle record with the current vehicle ID is generated and is accessible to the host controller.
 - Once the vehicle record exists, the host controller can command the vehicle to a destination. This motion command is the same as any other vehicle command (move to position or move to station).

Moving Downstream Through a Gateway Node

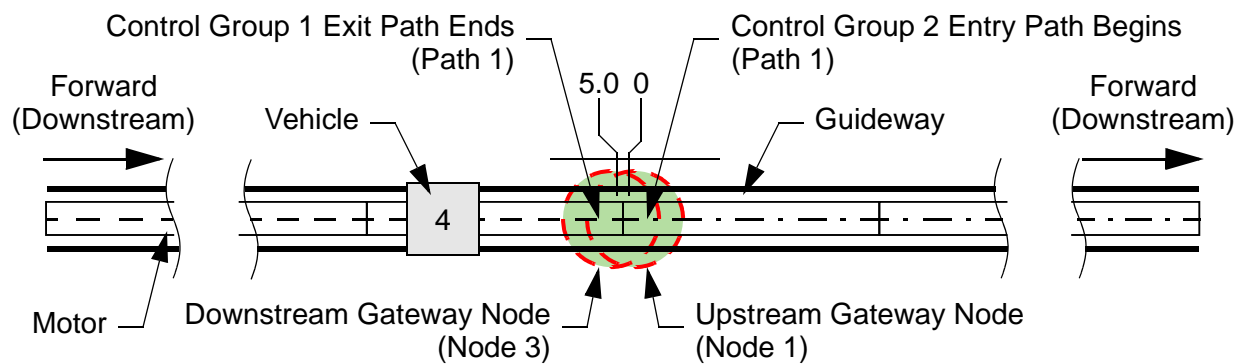


Figure 3-32: Gateway Node Motion

1. The host controller issues a vehicle move command that moves the vehicle so the leading edge of the vehicle crosses the border of the Gateway Node. The table represents the *MMI_vehicle_position_order* message.

vehicle_id	4
path_id	1

position	5.0
velocity_limit	1.0
acceleration_limit	1.0
order_number	5
flags_and_direction	0x11
active_flag	0x01

Once the vehicle has permission to enter the node from the target Control Group the move continues through the node. When the target position for the vehicle is past the Gateway Node, the following occurs:

- The node controller in the first Control Group responsible for the Gateway Node deletes the vehicle record for the vehicle and changes the vehicle ID for the node to idle.
- The node controller in the second Control Group responsible for the Gateway Node creates a vehicle record for the vehicle and changes the vehicle ID for the node to the entering vehicle.

NOTE: If the vehicle ID of the vehicle entering a Control Group through a Gateway Node exists in that Control Group, entry is refused and the vehicle stops before the node border.

2. To prepare for vehicle entry through the Gateway Node, the host controller for the second Control Group monitors the status of the Gateway Node for a new vehicle ID. The table represents the *MMI_node_status* message.

node_id	1
vehicle_id	4
node_type	9
terminus_signals	0100 0000
requested_position	0
reported_position	0
device_status	0x00

3. Once a new vehicle ID is detected, the host controller issues a vehicle move command (move to position or move to station) for the new vehicle. The table represents the *MMI_vehicle_position_order* message.

vehicle_id	4
path_id	1
position	1.0
velocity_limit	1.0
acceleration_limit	1.0
order_number	27
flags_and_direction	11
active_flag	0x01

NOTE: If no command is placed before the vehicle completes its transfer, the vehicle travels to the default destination defined in the Node Controller Configuration File for the first Control Group and stops.

The vehicle command completion message is transmitted to the new transport system (where the vehicle is located) not the system that issued the command.

Merge and Diverge Nodes

Merge or Diverge nodes allow vehicles to move from one path to another. The downstream orientation of the switch mechanism determines if the switch is configured as either a Merge Node or a Diverge Node. The paths in a Merge Node are called Straight Entry, Curve Entry, and Merged Exit as shown in [Figure 3-33](#). The paths in a Diverge Node are called Single Entry, Straight Exit, and Curve Exit as shown in [Figure 3-34](#).

NOTE: Merge and Diverge nodes are only available on MagneMover LITE systems.



PINCH/CRUSH HAZARD: Moving mechanisms (vehicles and switch mechanisms) have no obstruction sensors.

Do not operate the transport system without barriers in place or personal injury could result in the squeezing or compression of fingers, hands, or other body parts between moving mechanisms.

When the motion from one path to another path is in the same direction (for example, motion on all paths is forward, or downstream), no special considerations are required. This type of motion (forward or backward through a switch) is the normal use for a switch.

When the motion from one path to another path is not in the same direction, the motion must be configured as a pair of moves. Each of these moves is a separate command where the motion is in only one direction. This type of motion (reversing the direction of a vehicle through a switch) must be used carefully. Motion permissions are granted upon request and another vehicle could get permission through the switch before the original vehicle requests permission for its second move. Permissions are granted up to the point where the same block is being requested for both vehicles. Once both vehicles are requesting the same block, motion stops and both vehicles report they are obstructed.

NOTE: Even if a motion command is configured as bidirectional, two moves are required as a vehicle cannot change direction in the middle of a move. Use caution when attempting to create motion profiles of this type as there is nothing in the system to prevent a deadlock. Deadlocks can occur when, due to permissions already granted, no new permissions can be given and the vehicles that are involved are not able to move.

Merge Node

A Merge Node is where the downstream ends of two paths merge into the upstream end of a third path. This type of node is used to merge paths or create multiple loops. See [Figure 3-33](#), where the shaded circle represents the Merge Node.

Support for this node type is provided in MagneMover LITE transport systems.

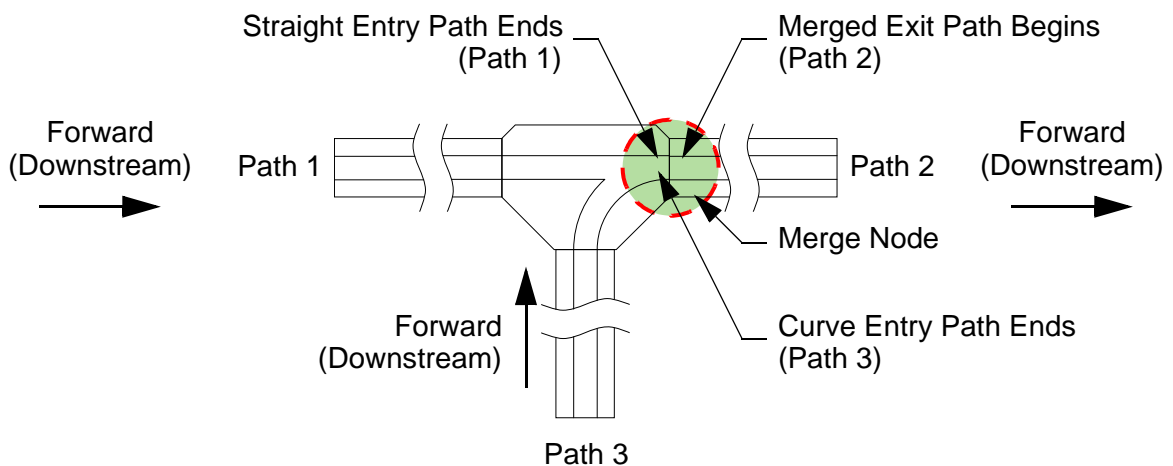


Figure 3-33: Merge Node

Node Operation

- Normal vehicle clearances apply across the node.
- Vehicles moving in the same direction can queue across the node.
- The move profile for the vehicle is maintained across the node so the vehicle crosses the node at a consistent velocity and acceleration.

Reversing Through a Merge Node

Reversing direction through an MM LITE Merge node (for example, path 1 to path 3 as shown in [Figure 3-33](#)) requires two moves. The first move is forward from path 1 to path 2. Before the first move can be performed, the switch mechanism must be positioned for the appropriate entry path. The second move is backward from path 2 to path 3 where motion continues backwards. Before the second move can be performed, the switch mechanism must be positioned for the appropriate entry path. The tables represent the *MMI_vehicle_position_order* messages.

NOTE: The first motion command must move the vehicle far enough past the switch to clear the switch mechanism and any configured clearance distance.

vehicle_id	4
path_id	2
position	0.25
velocity_limit	1.0
acceleration_limit	1.0
order_number	2
flags_and_direction	01
active_flag	0x01

Once the HLC updates the *MMI_vehicle_order_status* tag for the vehicle to show that it has completed its motion, the second move can be commanded. The table represents the *MMI_vehicle_position_order* message.

vehicle_id	4
path_id	3
position	5.75
velocity_limit	1.0
acceleration_limit	1.0
order_number	3

flags_and_direction	02
active_flag	0x01

Diverge Node

A Diverge Node is where the downstream end of a path splits into the upstream ends of two other paths. This type of node is used to split a path or create multiple loops. See [Figure 3-34](#), where the shaded circle represents the Diverge Node.

Support for this node type is provided in MagneMover LITE transport systems.

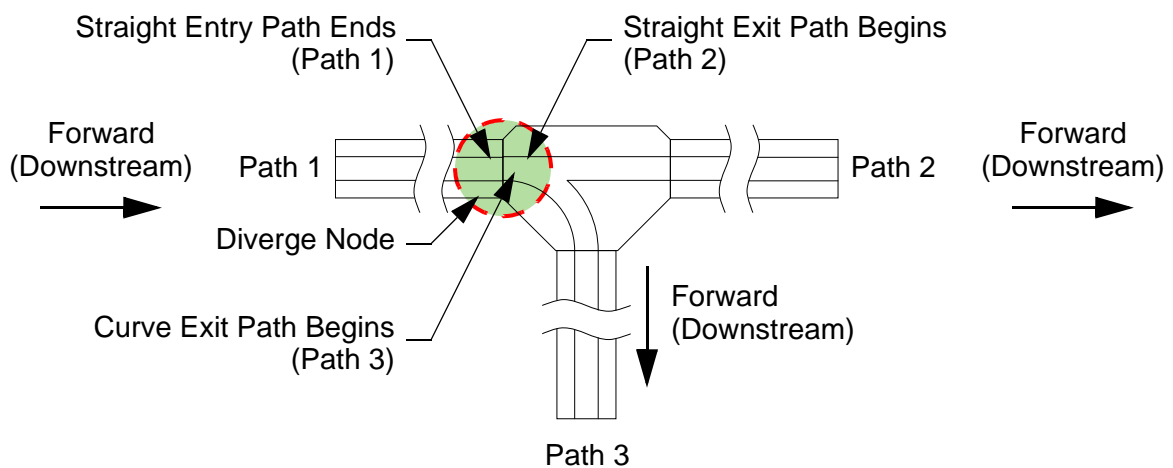


Figure 3-34: Diverge Node

Node Operation

- Normal vehicle clearances apply across the node.
- Vehicles moving in the same direction can queue across the node.
- The move profile for the vehicle is maintained across the node so the vehicle crosses the node at a consistent velocity and acceleration.

Reversing Through a Diverge Node

Reversing the direction of a vehicle through an MM LITE Diverge node (for example, path 2 to path 3 as shown in [Figure 3-34](#)) requires two moves. The first move is backward from path 2 to path 1. In MagneMover LITE systems, the switch mechanism is automatically positioned for the appropriate exit path. The second move is forward from path 1 to path 3 where motion continues forwards. As part of the second move in MagneMover LITE systems, the switch mechanism is automatically positioned for the appropriate exit path. The tables represent the *MMI_vehicle_position_order* messages.

NOTE: The first motion command must move the vehicle far enough to clear the switch mechanism and any configured clearance distance.

vehicle_id	4
path_id	2
position	5.75
velocity_limit	1.0
acceleration_limit	1.0
order_number	2
flags_and_direction	02
active_flag	0x01

Once the HLC updates the *MMI_vehicle_order_status* tag for the vehicle to show that it has completed its motion, the second move can be commanded. The table represents the *MMI_vehicle_position_order* message.

vehicle_id	4
path_id	3
position	1.5
velocity_limit	1.0
acceleration_limit	1.0
order_number	3
flags_and_direction	01
active_flag	0x01

Merge-Diverge Node

A Merge-Diverge Node is used where the downstream ends of two paths connect to the upstream ends of two other paths. A merge-diverge is created using two switches, where the common path of each switch is connected together. See [Figure 3-35](#), where the shaded circle represents the Merge-Diverge Node.

Support for this node type is provided in MagneMover LITE transport systems using high-payload switches only.

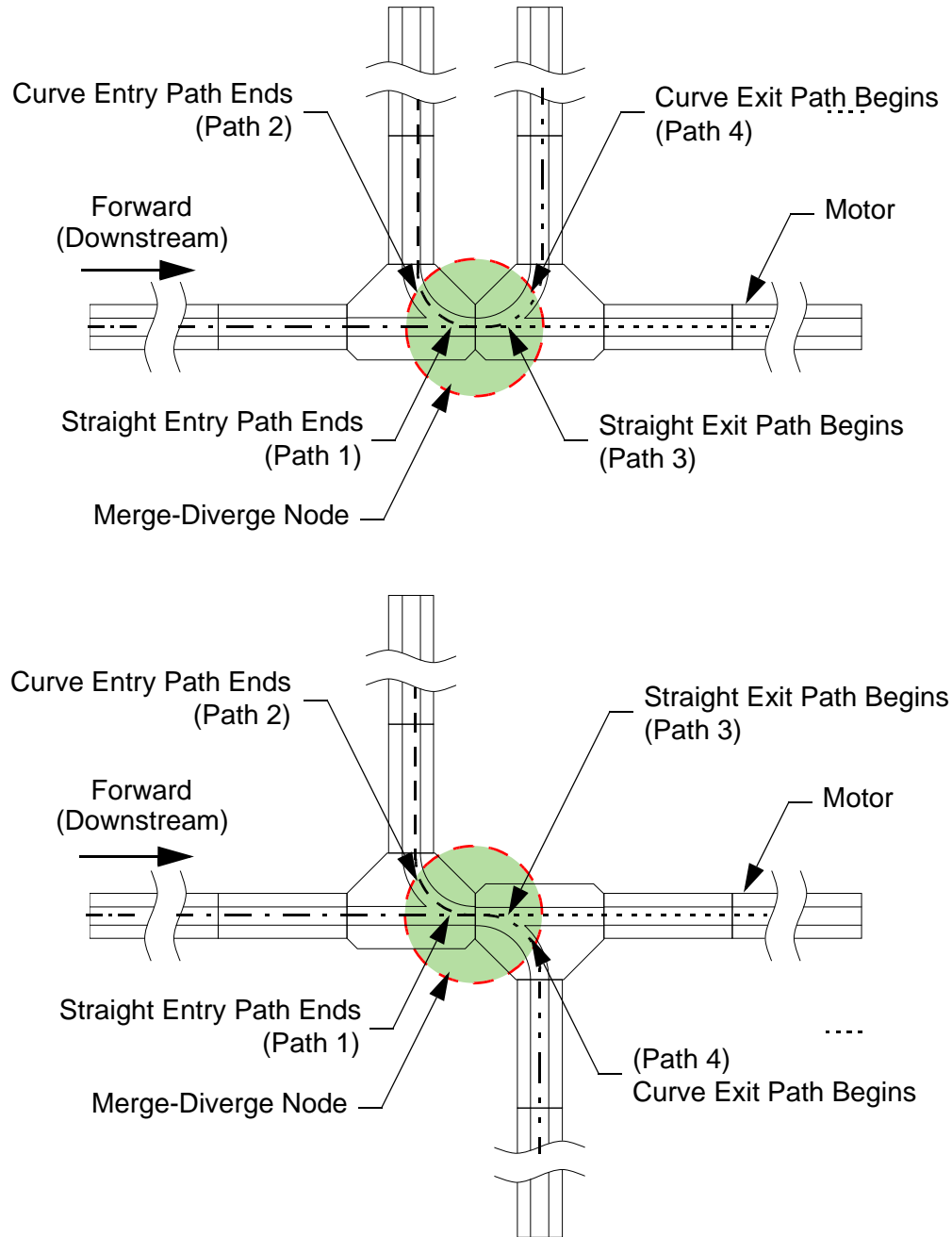


Figure 3-35: Merge-Diverge Node

Node Operation

- Normal vehicle clearances apply across the node.
- Vehicles moving in the same direction can queue across the node.
- The move profile for the vehicle is maintained across the node so the vehicle crosses the node at a consistent velocity and acceleration.

Overtravel Node

An Overtravel Node is used to permit part of a vehicle to move past the end of the motor at the end of a path. Additional support structure (guideway) for the vehicle must be provided, or the supports (wheels) for the vehicle must not be allowed to move past the end of the path. See [Figure 3-36](#) and [Figure 3-37](#), where the shaded area represents Overtravel Nodes.

The overtravel node is used for the following applications:

- Providing space for vehicle motion during startup ([Figure 3-36](#)).
- Allowing motion of a vehicle that is longer than the path ([Figure 3-37](#)).

Support for this node type is provided in:

- QuickStick transport systems.
- QuickStick HT transport systems.

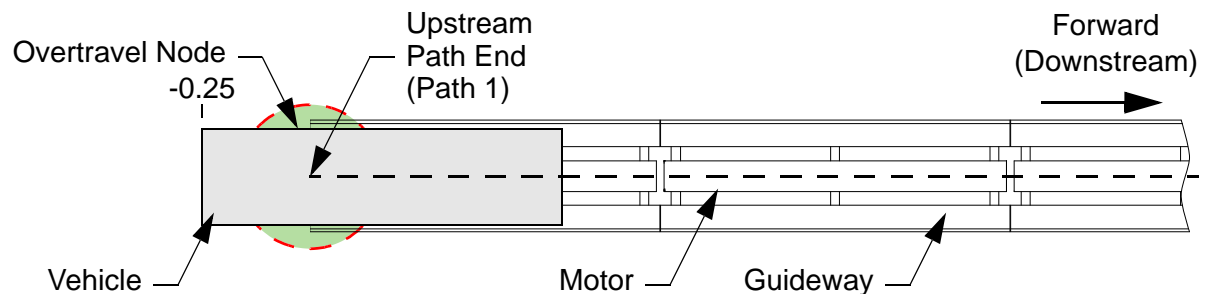


Figure 3-36: Overtravel Node, Startup Top View

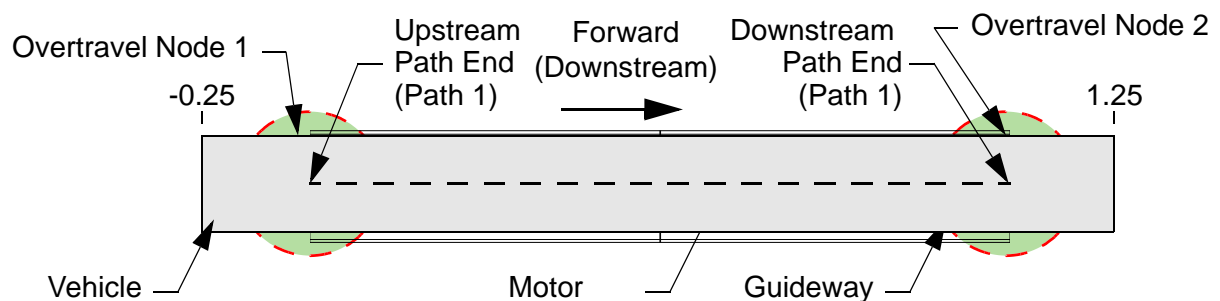


Figure 3-37: Overtravel Node, Extended Vehicle Top View

Node Operation

- Normal vehicle clearances apply.

Moving Path Node

A Moving Path Node is used to connect the ends of multiple paths using a path that a user-supplied drive mechanism moves. This mechanism can be any user-supplied mechanism, including a QuickStick path. The Moving Path Node enables vehicles to move between multiple guideways. The paths in the Moving Path Node are called entry and exit paths. Either type of path can be moved but the connecting paths must all be of the other type (that is, entry paths move and all exit paths are fixed). Up to 12 paths can be configured as either entry paths or exit paths. See [Figure 3-38](#), where the shaded circle represents the Moving Path Node.

Use of the Moving Path Node requires the host controller to command the drive mechanism to position one of the moving paths so that it aligns with one of the fixed paths. The host controller must then use the [MMI_mp_link_command](#) tag to connect the two paths to allow vehicle motion. Once the vehicle has moved beyond the node, the Moving Path must be unlinked before it can be moved to a new position.

This mechanism can either move a section of guideway and motors with vehicles on them as shown in [Figure 3-39](#) or a mechanical switch that diverts vehicle motion. A user-supplied mechanism supports and moves the moving path and aligns the end of the moving path with a fixed guideway, which allows vehicles to move onto and off the moving path. The ends of all paths in the node must be connected to the same node controller.

Support for this node type is provided in:

- QuickStick transport systems.
- QuickStick HT transport systems.



PINCH/CRUSH HAZARD: Moving mechanisms (vehicles and moving paths) have no obstruction sensors.

Do not operate the transport system or any switching mechanisms without barriers in place or personal injury could result in the squeezing or compression of fingers, hands, or other body parts between moving mechanisms.

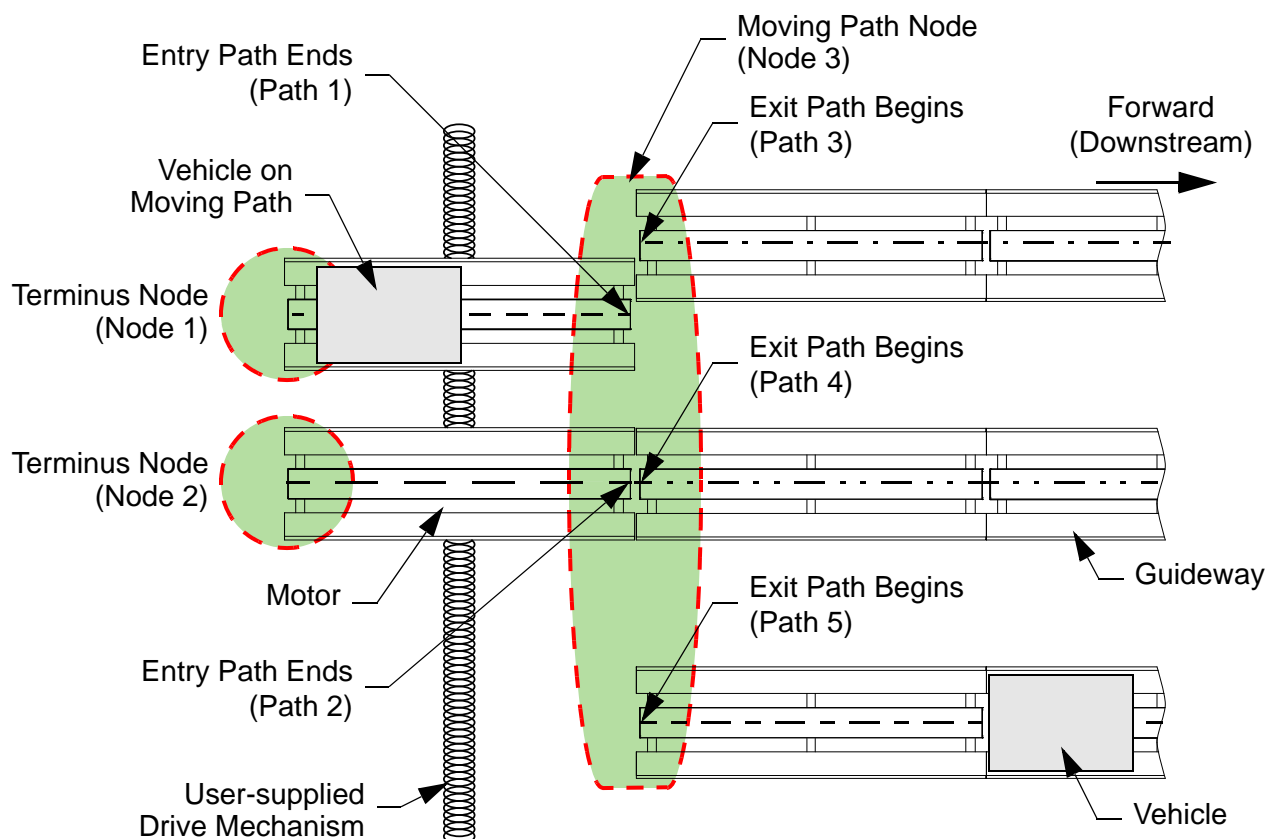


Figure 3-38: Moving Path Node

Node Operation

- Entry gates keep vehicles from entering the protected area around the path junctions unless the linked path provides the shortest route to its destination. To be granted permission to pass an entry gate, a vehicle must be under an order to a destination beyond the end of the Entry Path.
- A linked junction cannot be unlinked unless all vehicles are beyond the configured clearance distances of the junction.
- When positioning the moving path is not required, normal vehicle-to-vehicle clearances apply across the linked junction allowing vehicles to move across the junction.
- When positioning the moving path is not required, the move profile for the vehicle is maintained across the node so the vehicle crosses the node at a consistent velocity and acceleration.
- When positioning the moving path is not required, vehicles can queue across the node.
- Vehicles moving in the same direction can queue on the entry paths.
- Vehicles can queue on the moving path.
- Normal vehicle-to-vehicle clearances apply across the linked junction allowing vehicles on the same route to platoon across the junction.
- Vehicles can queue across a junction.
- The move profile for the vehicle is maintained across the node so the vehicle crosses the junction at a consistent velocity and acceleration only when the move is such that the moving path does not need to move.

Path Startup Considerations

During path startup in QuickStick and QSHT transport systems, unlocated vehicles move to determine their precise location. If the edge of a vehicle is over a motor, the vehicle does not always move to locate for QSHT and QS 150, for QS 100 they will always move. This move is termed a “locate move”. The node controller determines if there is room for the locate move. If there are already-located vehicles that are blocking the move, they must be moved to make room to locate the unlocated vehicle. This move is termed a “push move”. If there is no room to locate a vehicle in the downstream direction, the node controller searches for room in the upstream direction.

If a moving path junction is linked, the locating vehicle is allowed to locate into the linked junction. If a moving path junction is unlinked, the area past the entry gate is not available for a locate move or a push move. However, vehicles that extend past an entry gate can locate in the direction of the junction if there is room. Vehicles near an entry gate at an unlinked junction can only locate away from the junction.

It is possible that after an unlocated vehicle that was within the gate area is located it is still within the gate area at an unlinked path end. If a vehicle is positioned this way, the path end status transitions to the *unlinked_vehicle_present* state and the device_status field (bits 0...3 in the *MMI_node_status* tag) is set to “Junction Fault”. The host controller can either move the vehicle clear of the gate to cause the path end state to transition to *unlinked*, or link the junction. Once none of the member path ends of the Moving Path node are in the *unlinked_vehicle_present* state, the node device status transitions to “Operational”.

NOTE: A vehicle in the gate area has the potential to interfere with a path that is being moved to align it. Moving Path nodes are only available on QuickStick and QuickStick HT transport systems.

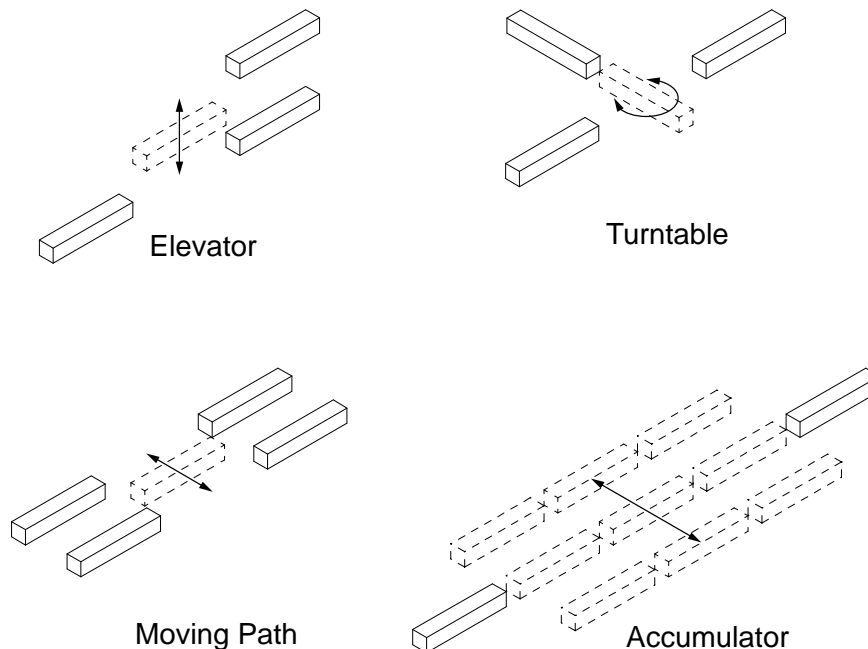


Figure 3-39: Examples of Moving Path Node Configurations Using Moving Guideways

QuickStick Startup using Moving Paths

Figure 3-40 describes the steps involved in the startup sequence for a QuickStick system. For QS 100 system vehicles are ALWAYS required to move to locate. For QS 150 and QSHT vehicles only have to move if a magnet array edge is not over a motor. The distance required to move will depend on the vehicle location, vehicle length and magnet array length. If there is not enough room to locate, previously located vehicles are “pushed” one motor block to create space for a new vehicle to locate. Startup will first attempt to locate vehicles by moving them in the downstream direction. If there is not enough room, the process will attempt to move vehicles in the upstream direction. This will continue until all vehicles are located or fail when there is not enough room to locate a vehicle.

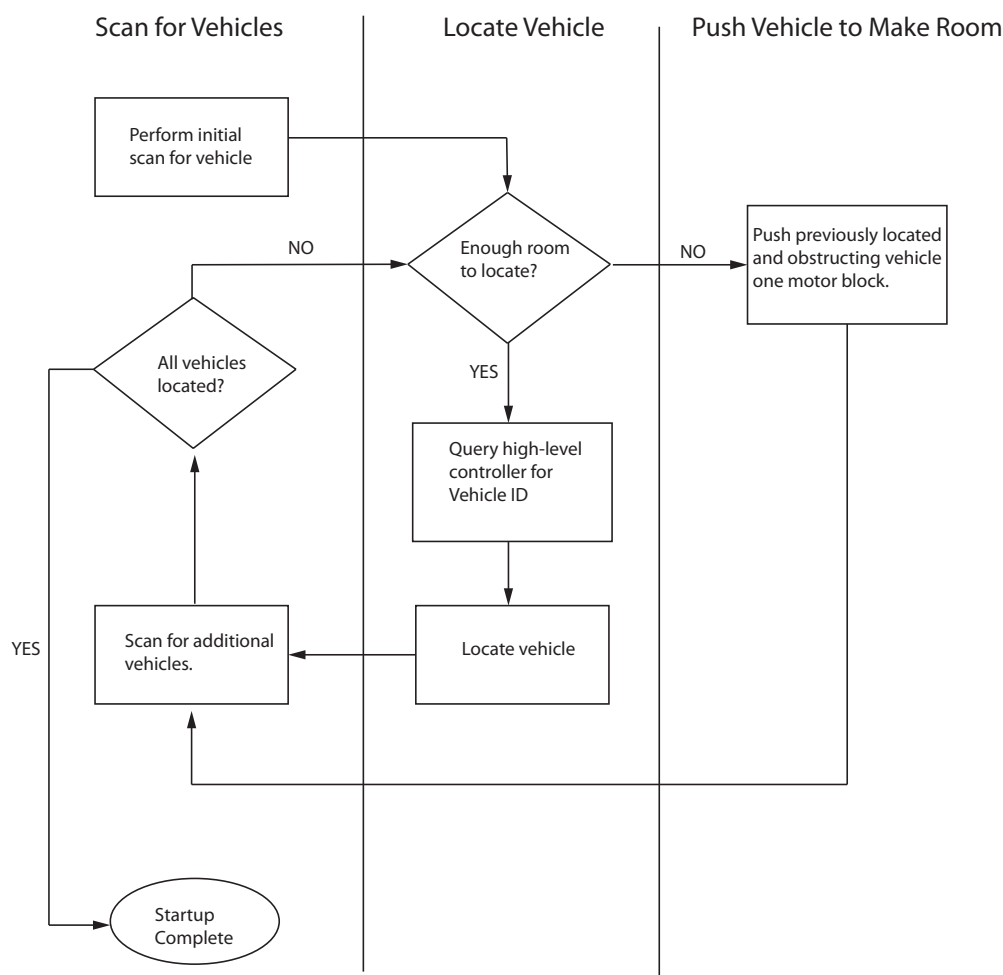


Figure 3-40: Startup Sequence - Worst Case

Moving Path Startup

For QuickStick systems with the Moving Path Node that fail to startup, the moving Paths can be used to assist neighboring fixed Paths for startup. This will increase the available number of blocks used for startup. [Figure 3-41](#) explains the startup routine and how it should be implemented with Moving Paths.

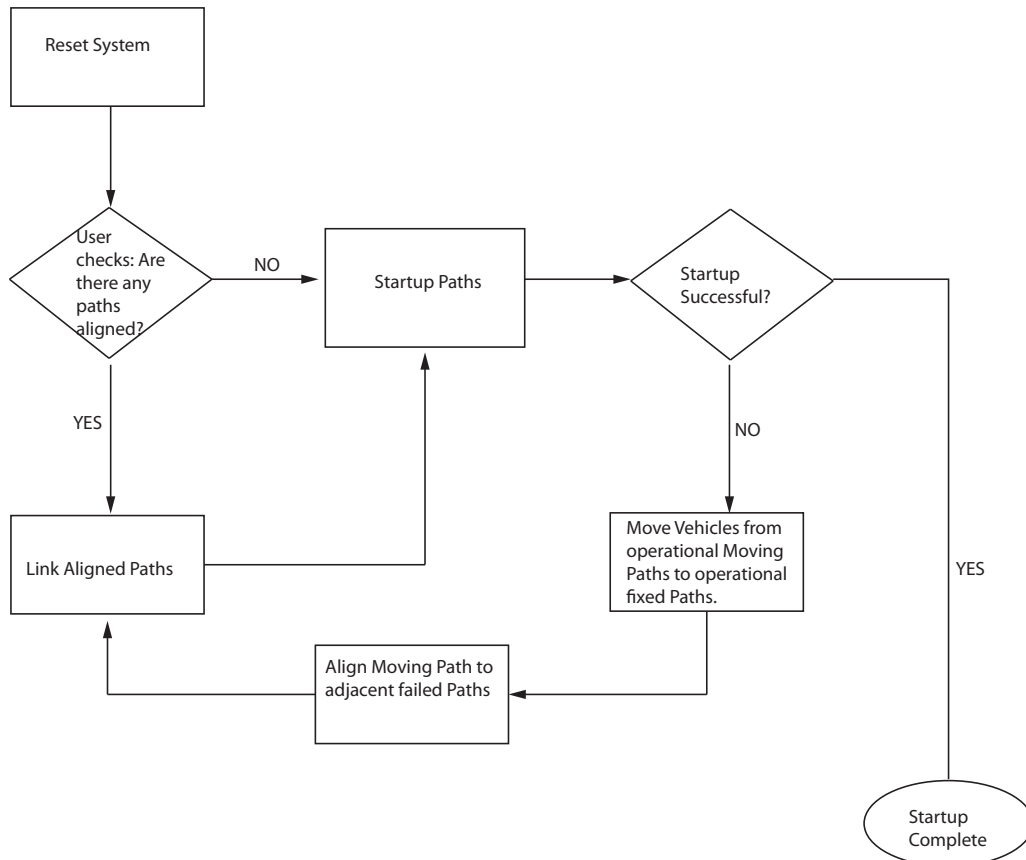


Figure 3-41: Startup Routine with Assistance from a Moving Path

During a Reset 0 Command, all Nodes go to the unlinked state. The host controller should check the location of all the moving Paths. If there are aligned Paths, a vehicle could be located between Paths. For this reason, it is important to "link" aligned Paths before sending a startup command. If a Path fails startup, moving Paths can be aligned and linked to those Paths and retry the startup routine. [Figure 3-42](#) show how linking two Paths allow for vehicles to startup across the junction by having more space to locate.

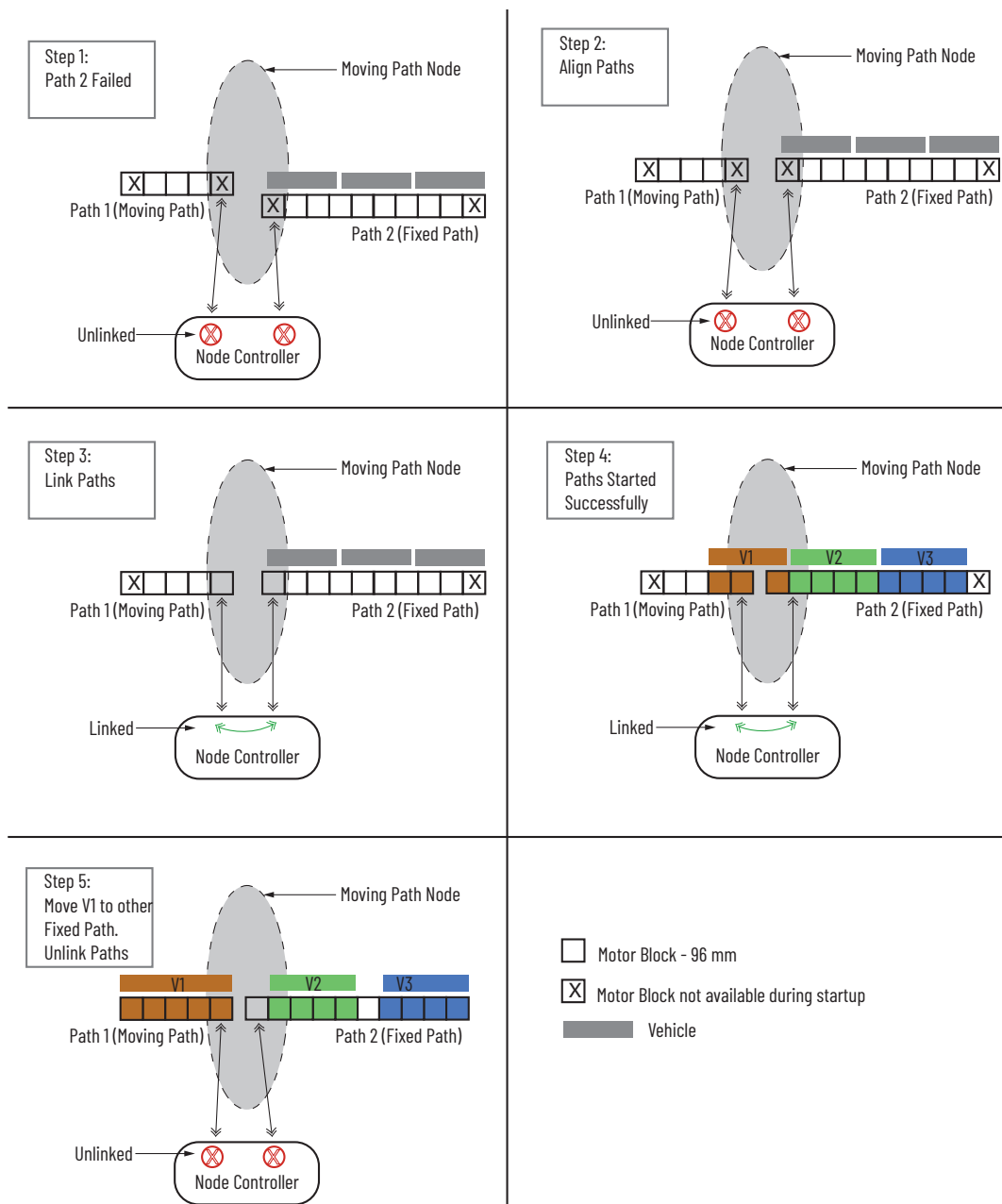


Figure 3-42: Startup Sequence for a Packed Path using a Moving Path for Assistance

NOTE: The X in the motor blocks show unavailable blocks due to the entry gate/clearance distance or vehicle overhang.

Path Linking/Unlinking

When two path ends are linked to form a junction, the node controller forwards motor-to-motor messages to allow vehicles to navigate the junction.

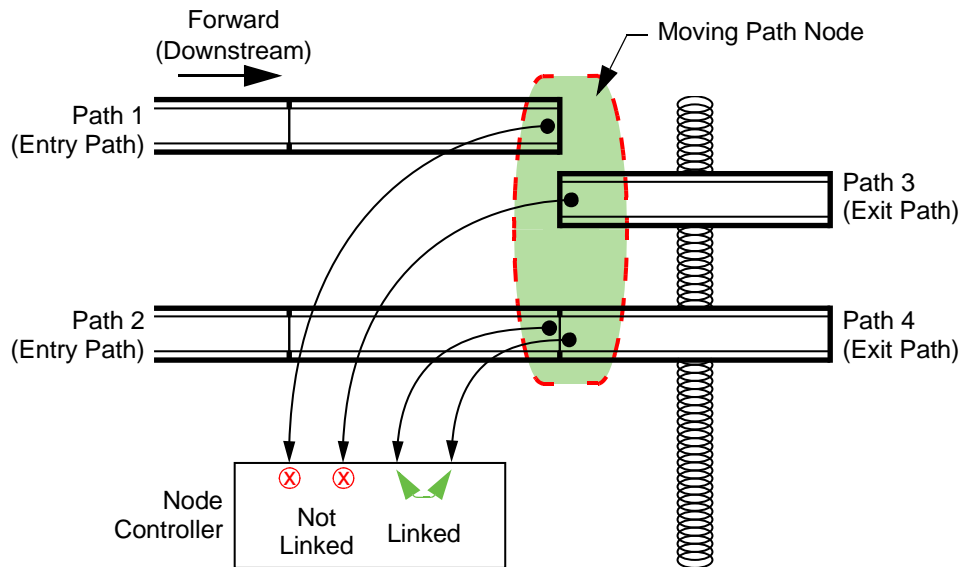


Figure 3-43: Path Linking/Unlinking in Moving Path Node

Linking the Paths in a Moving Path Node

When the paths in a Moving Path node junction are linked, the node controller forwards motor-to-motor messages between the path ends that are linked. This linking allows vehicle headway to advance through the junction and permits vehicle motion through the junction. When unlinked, motor-to-motor messages are not forwarded, which causes vehicles that are ordered through a junction to become obstructed, preventing vehicle motion past the Entry Gate until the junction is linked.

To link a moving path and a fixed path (for example, path 2 to path 4 as shown in Figure 3-44) to permit vehicles to move from one path to the other an *MMI_mp_link_command* command must be used. The table represents the *MMI_mp_link_command* message.

node_id	3
control_path_id	2
peer_path_id	4
last_allowed_vehicle_id	0
alignment_request_count	0x01
command_count	0x53
active_flag	0x01

The HLC updates the *MMI_mp_command_status* tag to show the *MMI_mp_link_command* was accepted (0x00) and that it completed (0x80).

Host Control of Link (last allowed vehicle id = 0)

If the host controller links the path with an *MMI_mp_link_command* tag that has *last_allowed_vehicle_id* set to “0”, the path end remains in the linked state until an *MMI_mp_unlink_command* tag is received. While in the linked state, vehicles are allowed to navigate the node if the linked path offers a route to the destination for the vehicle.

The host controller must send an *MMI_mp_unlink_command* tag to unlink the path. If the host controller sends the unlink command while a vehicle is navigating the junction, the junction state changes to *linked_unlink_pending* and no additional vehicles are granted permission to navigate the junction. As soon as the vehicle clears the junction, the path end state transitions to the *unlinked* state.

Host Control of Link (last allowed vehicle id = Specific Vehicle ID)

If the host controller links the path with an *MMI_mp_link_command* tag that has *last_allowed_vehicle_id* set to a specific vehicle ID, the path end remains in the linked state until the specified vehicle is granted permission to enter the junction. The path end state transitions to *linked_unlink_pending* as soon as the last allowed vehicle is granted permission to enter. No other vehicles are allowed permission to navigate the junction after the last allowed vehicle enters.

Once in the *linked_unlink_pending* state, the path transitions to the *unlinked* state as soon as all vehicles are clear of the junction. The host controller does not need to send an unlink command.

NOTE: Additional link commands can be issued to modify the *last_allowed_vehicle_id* for the linked paths but must specify the same *control_path_id* and *peer_path_id* used initially to establish the junction.

Unlinking the Paths in a Moving Path Node

Once use of the node is complete, the linked paths in a Moving Path node must be unlinked to permit linking to different paths in the node. To unlink the paths when the node is not configured to automatically unlink after a specific vehicle, an *MMI_mp_unlink_command* command must be used. The table represents the *MMI_mp_unlink_command* message.

node_id	3
control_path_id	2
command_count	54
active_flag	0x01

The HLC updates the *MMI_mp_command_status* tag to show the *MMI_mp_unlink_command* was accepted (0x00) and that it completed (0x80).

Moving Path Node Configuration

When using a moving path switch, the host controller is responsible for positioning the guideway for the moving path to provide the required routing of the vehicles. The Moving Path node provides support to route the vehicles from the entry paths to their exit paths by linking the paths, it does not provide any mechanism control or monitoring.

A user-supplied mechanism supports and moves the guideway for the moving path and aligns the end of the moving path with a fixed guideway, which allows vehicles to move on and off the moving path. The paths in a moving path switch can be configured in different layouts including moving paths and turntables. When using a moving path style configuration (see [Figure 3-44](#)), any user-supplied mechanism can be used to position the guideway, including a QuickStick path. When using a turntable configuration, a user-supplied rotary indexer is used to position the guideway.

Either entry or exit paths can move but the connecting paths must all be of the other type (that is, entry paths move and all exit paths are fixed). The type of node controller and the communication method that is used determines the number of paths that can be configured as either entry paths or exit paths. The ends of all paths in the node must be connected to the same node controller. All paths in the node must maintain the downstream to upstream relationship of the motors. See [Figure 3-44](#) where the large shaded area represents the Moving Path node and the entry paths can move. There can also be a Moving Path node at each end of a moving path where vehicles move onto or off the moving path as shown in [Figure 3-45](#).

Use of a moving guideway requires the host controller to command the drive mechanism to position one of the moving paths so that it aligns with one of the fixed paths. The host controller must then issue an *MMI_mp_link_command* to connect the two paths to allow vehicle motion. Once the vehicle has moved beyond the node, the host controller must issue an *MMI_mp_unlink_command* before the Moving Path can be moved to a new position.

[Figure 3-44](#) shows a configuration with one Moving Path node. Vehicles can move between the three fixed exit paths using either of the two moving entry paths. Vehicles can enter the moving paths either through the Terminus Nodes or from the fixed paths.

[Figure 3-45](#) shows a configuration with dual Moving Path nodes. Vehicles can move from either of the two fixed entry paths in Moving Path node 3 to any of the fixed exit paths in Moving Path node 4 using either of the two moving paths. In this type of configuration, the moving paths are associated with both Moving Path nodes. For Moving Path node 3, these paths are moving exit paths. For Moving Path node 4, these paths are moving entry paths.

In the example shown in [Figure 3-45](#), if the vehicle is moving from Moving Path node 3 entry path 2 to Moving Path node 4 exit path 6 the moving path does not need to move. Once path 4 is linked to both path 2 and path 6, vehicles can move directly from path 2 to path 6.



PINCH/CRUSH HAZARD: Moving mechanisms (motors and moving paths) have no obstruction sensors.

Do not operate the transport system without barriers in place or personal injury could result in the squeezing or compression of fingers, hands, or other body parts between moving mechanisms.

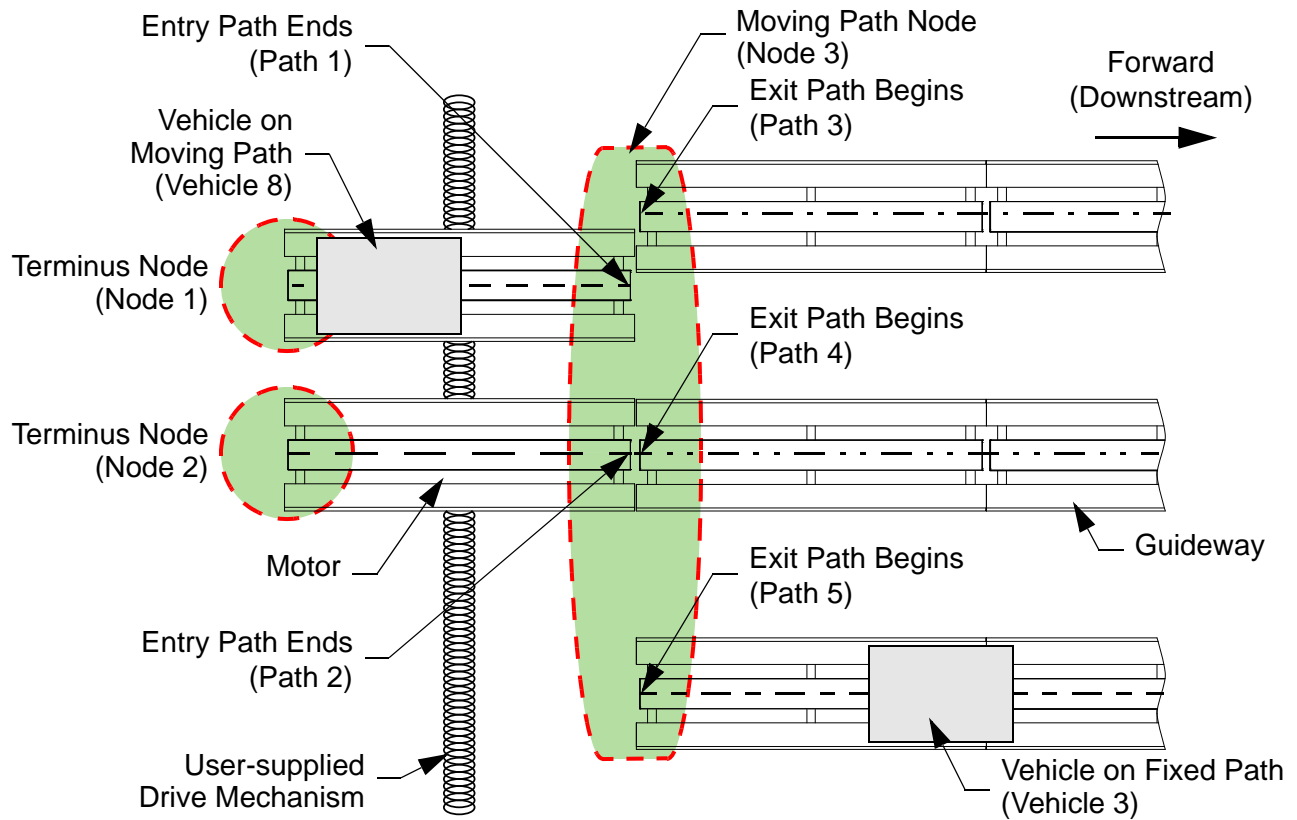


Figure 3-44: Moving Path Node, Top View

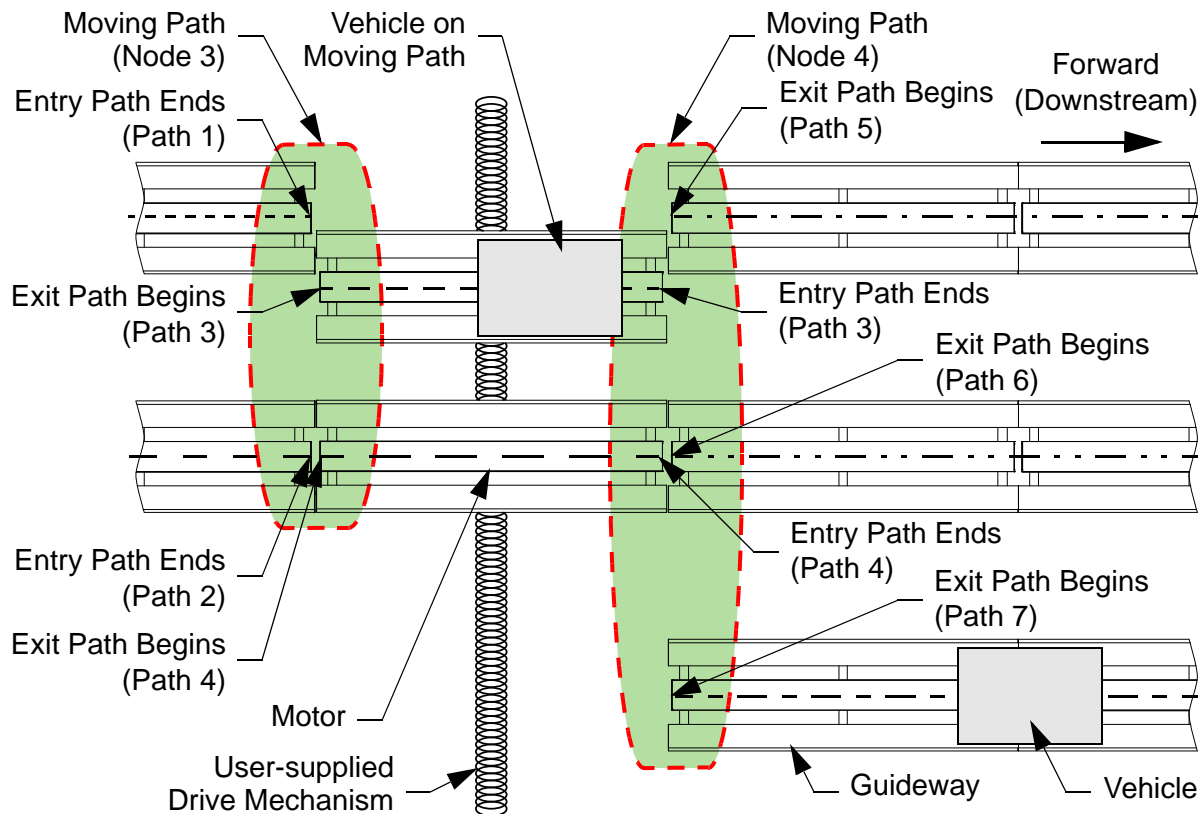


Figure 3-45: Moving Path Node Pair, Top View

Forward Through a Moving Path Node

Forward motion through a Moving Path node (for example, path 1 to path 3 as shown in [Figure 3-44](#)) requires positioning the moving path so it aligns with the fixed path. Once the paths are aligned, the paths are linked as described in [Linking the Paths in a Moving Path Node](#). Once the paths are linked, this move is configured as a basic move from the entry path to the exit path where motion continues forwards. Once the move is complete, the paths should be unlinked as described in [Unlinking the Paths in a Moving Path Node](#). The table represents the *MMI_vehicle_position_order* message.

vehicle_id	8
path_id	3
position	1.5
velocity_limit	1.0
acceleration_limit	1.0
order_number	72
flags_and_direction	01
active_flag	0x01

The HLC updates the MMI_vehicle_order_status tag to show the MMI_vehicle_position_order was accepted (0x00) and that it completed (0x80).

Backward Through a Moving Path Node

Backward motion through a Moving Path node (for example, path 5 to path 2 as shown in [Figure 3-44](#)) requires positioning the moving path so it aligns with the fixed path. Once the paths are aligned, the paths are linked as described in [Linking the Paths in a Moving Path Node](#). Once the paths are linked, this move is configured as a basic move from the exit path to the entry path where motion continues backwards. Once the move is complete, the paths should be unlinked as described in [Unlinking the Paths in a Moving Path Node](#). The table represents the *MMI_vehicle_position_order* message.

vehicle_id	3
path_id	2
position	5.75
velocity_limit	1.0
acceleration_limit	1.0
order_number	4
flags_and_direction	02
active_flag	0x01

The HLC updates the MMI_vehicle_order_status tag to show the MMI_vehicle_position_order was received (0x00) and that it completed (0x80).

Reversing Through a Moving Path Node

Reversing direction through a Moving Path node (for example, path 3 to path 4 as shown in [Figure 3-44](#)) requires two moves. The first move is backward from path 3 (the fixed path) to path 1 (the moving path) and requires positioning the moving path so it aligns with the fixed path. Once the paths are aligned, the paths are linked as described in [Linking the Paths in a Moving Path Node](#). After the paths are linked, this move is configured as a basic move backwards from path 3 to path 1 to center the vehicle on the moving path. Once the move is complete, the paths are unlinked as described in [Unlinking the Paths in a Moving Path Node](#).

NOTE: The first move command must move the vehicle fully onto the moving path to clear the guideway it had been on, which allows the path to move. This is typically achieved by centering the vehicle on the moving path. Or, the vehicle must be positioned far enough on the moving path that the appropriate node clearance (see [Table 3-1](#)) is achieved.

The second move is forward from path 1 to path 4 and requires positioning path 1 (the moving path) so it aligns with path 4 (the fixed path). Once the paths are aligned, the paths are linked as described in [Linking the Paths in a Moving Path Node](#). After the paths are linked, this move is configured as a basic move forwards from path 1 to path 4. Once the vehicle exits path 1 the paths are unlinked as described in [Unlinking the Paths in a Moving Path Node](#).

NOTE: The first move command must move the vehicle far enough onto the moving path so that it is clear of the moving path mechanism and any configured clearance distance.

vehicle_id	3
path_id	1
position	0.5
velocity_limit	1.0
acceleration_limit	1.0
order_number	13
flags_and_direction	02
active_flag	0x01

The HLC updates the MMI_vehicle_order_status tag to show the MMI_vehicle_position_order was accepted (0x00) and that it completed (0x80).

Once the first move completes, path 1 and path 3 must be unlinked (see [Unlinking the Paths in a Moving Path Node](#)). Once the unlink completes and the moving path (with the vehicle on it) is aligned with the new fixed path, path 1 and path 4 must be linked (see [Linking the Paths in a Moving Path Node](#)). Once the link completes, the vehicle can be moved.

NOTE: The second move command must move the vehicle far enough onto the fixed path so that it is clear of the moving path mechanism and any configured clearance distance.

vehicle_id	3
path_id	1
position	1.5
velocity_limit	1.0
acceleration_limit	1.0
order_number	14
flags_and_direction	01
active_flag	0x01

The HLC updates the *MMI_vehicle_order_status* tag to show the *MMI_vehicle_position_order* was accepted (0x00) and that it completed (0x80).

Once the second move completes, path 1 and path 4 must be unlinked (see *Path Linking/Unlinking* on page 133).

Moving Path Example 1: Moving Onto a Moving Path

In the example in Figure 3-46, with one Moving Path node, the vehicles move from a fixed path (defined in the Moving Path node configuration as the Entry Path) and enter the moving path (defined in the Moving Path node configuration as the Exit Path).

The example in Figure 3-46 has two fixed paths, path 1 and path 2, which connect to two moving paths, path 3 and path 4.

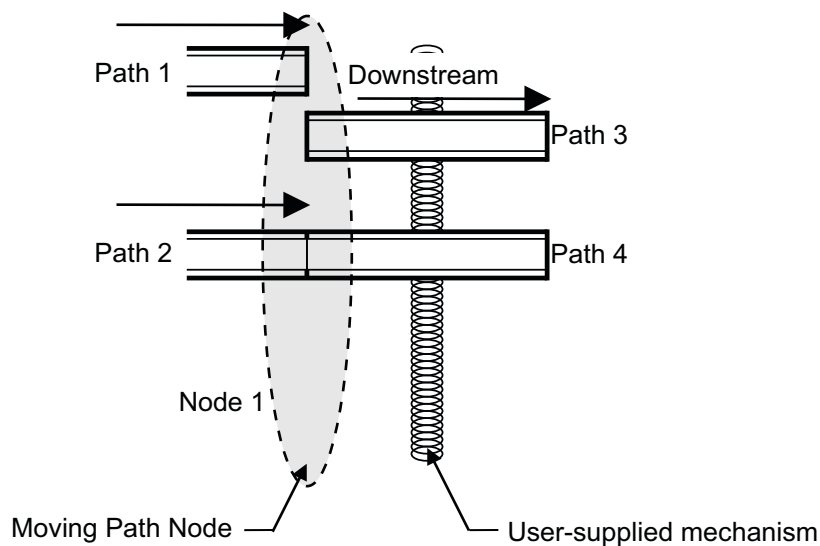


Figure 3-46: Moving Onto a Moving Path

1. To move onto either path 3 or path 4, the host controller must order the vehicle to path 3 or path 4.
2. Once the vehicle headway reaches the entry gate, the HLC then requests alignment of the destination path that is specified in the vehicle order, setting *MMI_mp_path_end_status* for the path the vehicle is on to the *unlinked_alignment_requested* state, see [Table 4-57 on page 256](#).
3. The host controller then commands the mechanism to position the moving path as requested, such that two paths are physically aligned.
4. Once the moving path are aligned, the host controller issues an *MMI_mp_link_command* to link the two paths and the vehicle can move across the junction between the paths.
5. Depending upon the usage of the linked paths, once the vehicle clears the junction, the paths can be unlinked, or the paths can remain linked (see [Path Linking/Unlinking on page 133](#)).

Moving Path Example 2: Moving Across a Moving Path

This Moving Path node application allows one or more moving paths to span a gap between one or more fixed paths. [Figure 3-47](#) shows an example of a Moving Path node pair (node 1 and node 2). Node 1 is a Moving Path node linking the downstream end of fixed paths, path 1 and path 2, to the upstream end of moving paths, path 3 or path 4. Node 2, is a Moving Path node linking the downstream end of moving paths, path 3 and path 4, to the upstream end of fixed paths, path 5, path 6, or path 7.

Considering movement in the downstream direction (indicated by the arrows) the upstream end of Exit Paths 3 and 4 (the moving paths) offer equivalent routes for vehicles navigating node 1 to destinations on paths 5, 6, or 7. Either can satisfy a vehicle move from path 1 or path 2 to path 5, path 6, or path 7. Paths 3 and 4 are termed equivalent-route paths. At node 2, still moving downstream, the host controller must align to either path 5, path 6, or path 7 depending upon the vehicles destination. Paths 5, 6, and 7 are termed specific-route paths.

Similarly, considering movement in the upstream direction, the downstream end of path 3 or 4 offer equivalent routes to path 1 and path 2 when navigating node 2. Paths 1 and 2 require specific alignment at node 1.

Two node configuration items must be set, one for the set of Entry Paths and one for the set of Exit Paths, designating the set as specific-route or equivalent-route. The HLC assumes that any moving path in an equivalent-route set can satisfy the route. Because the host controller application chooses the path to align to satisfy an equivalent route alignment request from the HLC, the host controller can provide application-specific management strategy for utilization of moving paths.

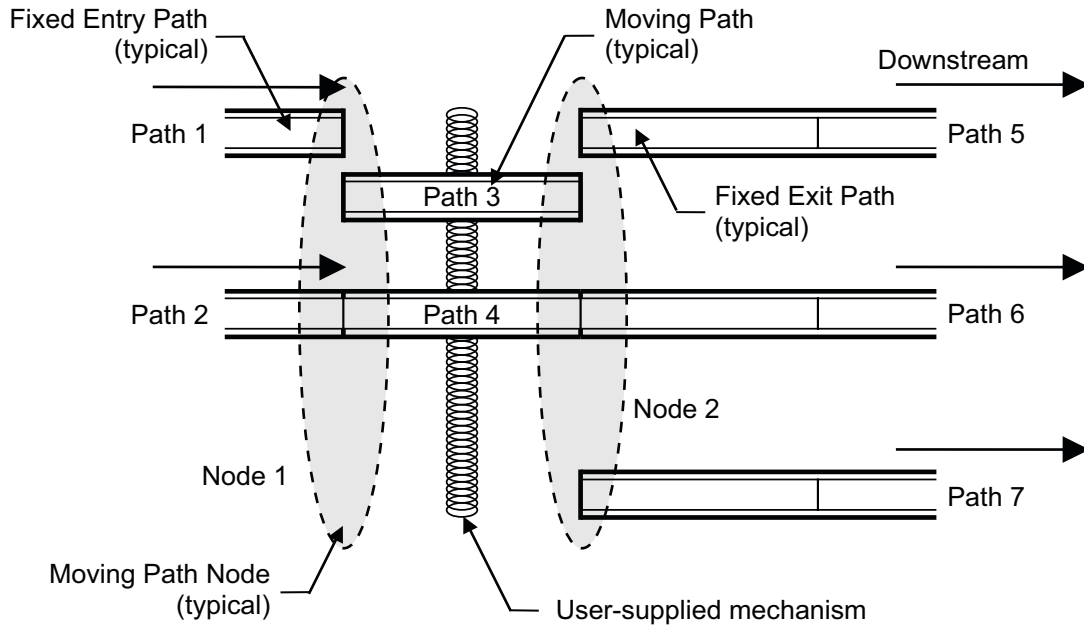


Figure 3-47: Moving Across a Moving Path

1. To move onto one of the Exit Paths from either of the Entry Paths, path 1 or path 2, the host controller must order the vehicle to either path 5, path 6, or path 7.
2. Once the vehicle headway reaches the entry gate, the HLC requests alignment of a moving path to complete the vehicle order by setting *MMI_mp_path_end_status* for the path the vehicle is on to the *unlinked_alignment_requested* state, see [Table 4-57 on page 256](#) for Moving Path Node End States.
3. The host controller determines which path to move and align and then commands the mechanism to position the moving path as requested.
4. Once the moving path is positioned, the host controller notifies the HLC that the paths are aligned by issuing an *MMI_mp_link_command* to link the two paths so the vehicle can move across the junction between the paths onto the moving path.
5. Once the last allowed vehicle is granted permission to navigate the junction, the HLC sets *MMI_mp_path_end_status* for the path to *linked_unlinked_pending* state to notify the host controller that permission to navigate the junction is not granted to any additional vehicles.
6. Once all vehicles that are involved in this junction clear the junction, the paths are unlinked (see *Path Linking/Unlinking on page 133*).
7. The vehicle on the moving path continues to move toward the other end of the moving path and the path end state of that path end transitions to *unlinked_alignment_requested* this time specifying the specific path end where the host controller must align the moving path.

8. The host controller commands the mechanism to position the moving path and aligns it with the Exit Path. The moving path can be moved while the vehicle is in motion.
9. Once the moving path is positioned, the host controller issues an [*MMI_mp_link_command*](#) to link the two paths and the vehicle can move across the junction between the paths onto the Exit Path.
10. Depending upon the usage of the linked paths, once the vehicle clears the junction, the paths can be unlinked, or the paths can remain linked (see [*Path Linking/Unlinking on page 133*](#)).

Moving Path Switch Configuration

When using a mechanism to move a section of guideway with the vehicle on it, the host controller is responsible for positioning the guideway to provide the required routing of the vehicles. The Moving Path node provides support to route the vehicles from the entry paths to the moving path and then to their exit paths by linking the paths, it does not provide any position control or monitoring.

When using a switching mechanism to divert vehicle motion, the host controller is responsible for positioning the switch mechanism to provide the required routing of the vehicles. The Moving Path node provides support to route the vehicles from the entry paths to their exit paths by linking the paths, it does not provide any mechanism control or monitoring.

Moving Path can be used to control a switch that can merge two path together (see [Figure 3-48](#)) or diverge one path into two paths (see [Figure 3-49](#)).

NOTE: Moving Path node types are not the same node types as those used in Merge, Diverge, or Merge-Diverge nodes.

Merge/Diverge Switches are only available as switch modules on MagneMover LITE transport systems. QuickStick and QuickStick HT use Moving Path nodes.

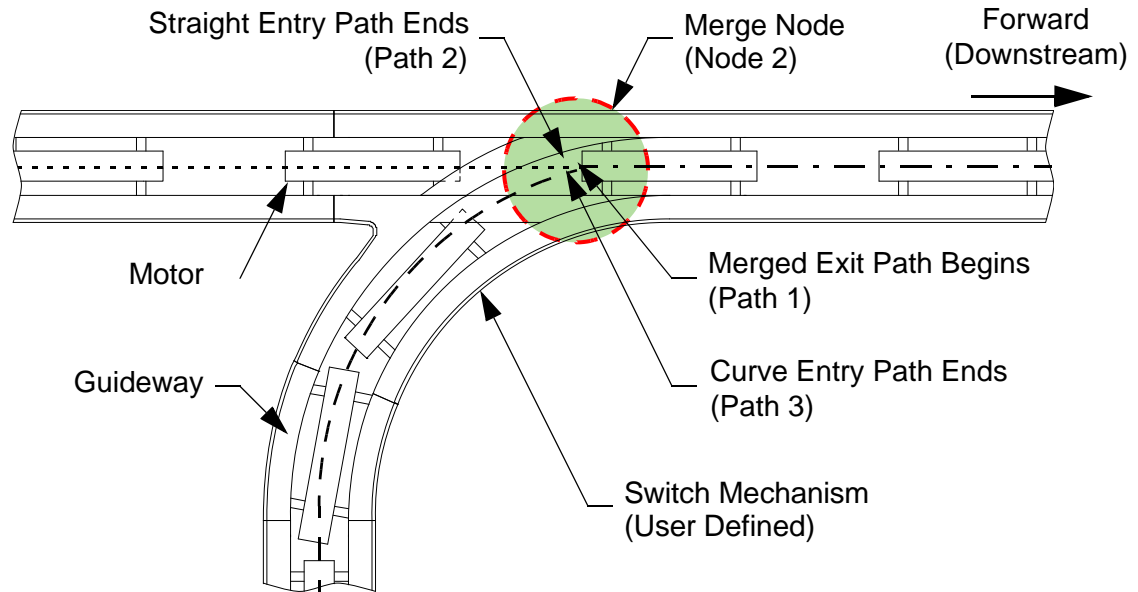


Figure 3-48: Moving Path Node Merge, Top View

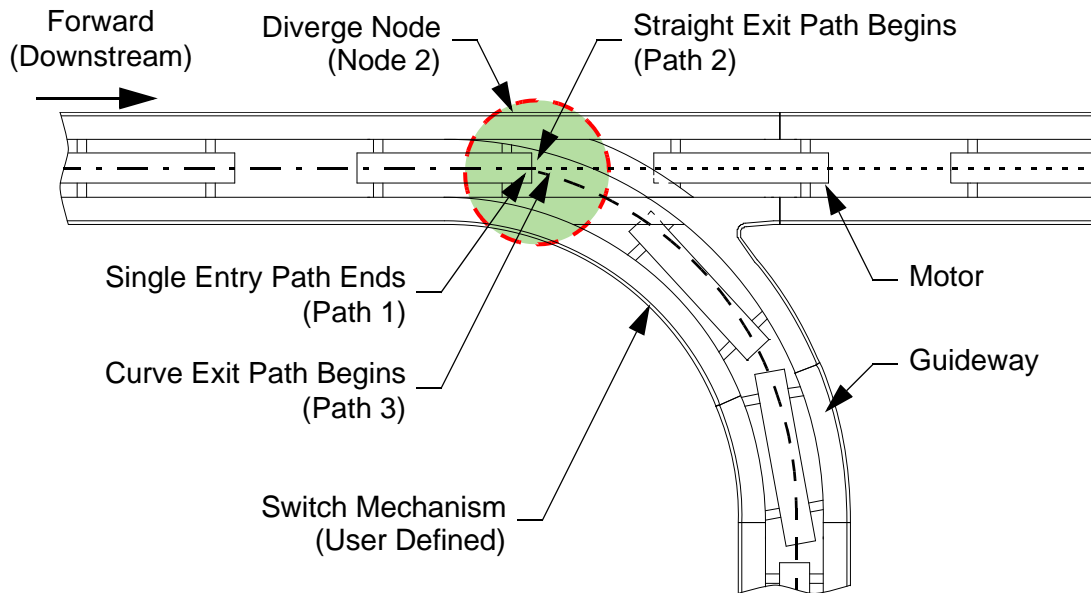


Figure 3-49: Moving Path Node Diverge, Top View

Forward Through a Moving Path Based Switch

Forward motion through a Moving Path based switch (for example, path 2 to path 1 as shown in [Figure 3-48](#)) requires linking the paths as described in *Path Linking/Unlinking* [on page 133](#). Once the paths are linked, this move is configured as a basic move from the entry path to the exit path where motion continues forwards. Before the move can be performed, the switch mechanism must be positioned appropriately. Once the move is complete, the paths should be unlinked as described in *Path Linking/Unlinking* [on page 133](#).

Backward Through a Moving Path Based Switch

Backward motion through a Moving Path based switch (for example, path 1 to path 2 as shown in [Figure 3-48](#)) requires linking path 1 to path 2 as described in *Path Linking/Unlinking* [on page 133](#). Once the paths are linked, this move is configured as a basic move from the exit path to the entry path where motion continues backwards. Before the move can be performed, the switch mechanism must be positioned appropriately. Once the move is complete, the paths should be unlinked.

Reversing Through a Moving Path Based Switch

Reversing direction through a Moving Path based switch requires two moves. This example shows reversing through a Moving Path merge switch (path 2 to path 3 as shown in [Figure 3-48](#)). The first move is forward from path 1 to path 2 and requires linking the paths as described in *Path Linking/Unlinking* [on page 133](#). Before the first move can be performed, the switch mechanism must be positioned appropriately.

The second move is backward from path 1 to path 3 and requires unlinking path 1 and path 2 as described in *Path Linking/Unlinking* [on page 133](#). Once the paths are unlinked, path 1 can be linked to path 3. Before the second move can be performed, the switch mechanism must be positioned appropriately. Before the second move can be performed, the switch mechanism must be positioned appropriately. Once the move is complete, unlink the paths. Once the HLC updates the *MMI_vehicle_order_status* tag for the vehicle to show that it has completed its motion, the second move can be commanded.

NOTE: The first move command must move the vehicle far enough past the switch to clear the switch mechanism and any configured clearance distance.

Overview

This chapter provides an overview of the command and response protocol for the Host Controller EtherNet/IP Communication Protocol and details of each User-defined Data Tag (UDT) and the host controller interface. The high-level controller provides a broad range of command options for MagneMover[®] LITE and QuickStick[®] transport systems. The high-level controller also provides the host controller the ability to monitor, and be notified of, status changes within the transport system.

NOTE: Specific builds of the Node Controller Software Image File may not implement all Host Controller EtherNet/IP Communication Protocol features described in this manual. See the Release Notes that are supplied with the Software for more information.

All QuickStick, QuickStick HT, and MagneMover LITE product lines support all EtherNet/IP tags except as stated within these references.

Introduction

This chapter provides an overview of the communications between the host controller and the high-level controller and detailed information on the protocol and tags. The standard method of communication uses EtherNet/IP communication protocols (see *Communications Protocol on page 393*). Any undocumented tags that are included in the protocol are used by Rockwell Automation to support internal functions and are not intended for customer use. Use only the instructions and tags that are documented in this Protocol and User-defined Data Tag Reference.

This communication protocol is provided for communications between the HLC and the host controller, which is established using an EtherNet/IP connection. Only one device can be connected through the control port at this interface to the HLC at a time.

For information about Logix and Data Types, see Logix 5000 Controllers Design Considerations, publication [1756-RM094](#).

Example of Explicit Message Setup

The following example shows how to create an explicit message that targets the high-level controller when the message is triggered. All screen captures are from the Allen-Bradley® Studio 5000 Logix Designer® programming utility that is supplied by Rockwell Automation®. The message is populated with data from a tag in the host controller memory that is a single data type (typically a UDT) or an array of data types.

Choose **MESSAGE** for the **Data Type** for the new tag type as shown in [Figure 4-1](#). Then select **Configure** to bring up the message configuration dialogue as shown in [Figure 4-2](#).

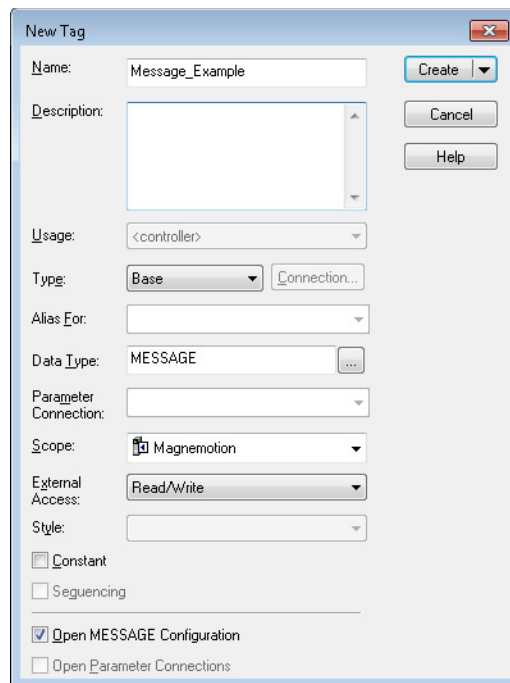


Figure 4-1: New Tag dialogue creating a message

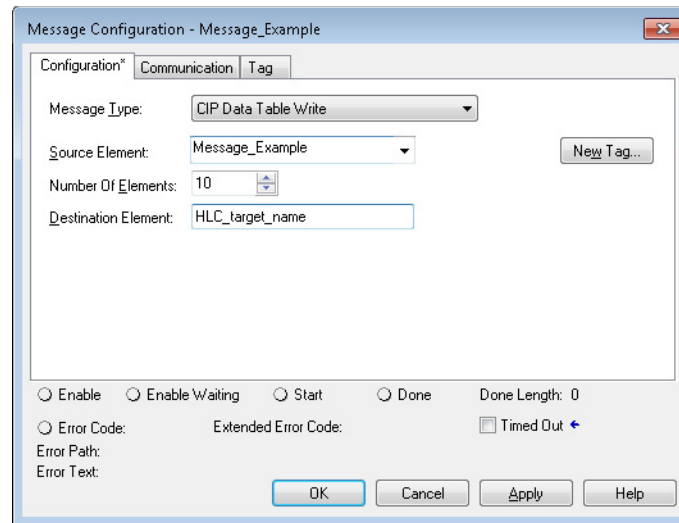


Figure 4-2: Message Configuration

In the Configuration tab of the message configuration, set the **Message Type** to CIP™ Data Table Write. The **Source Element** is any standard tag or an array of tags (typically UDT) with the **Number Of Elements** as required by the message definitions in this section. It is assumed the source element tag has already been defined before creating the message. The **Destination Element** is the name that the high-level controller expects to be used to identify the message.

The **Source Element**, **Number Of Elements**, and **Destination Elements** items in this dialogue are the items that vary depending on the action that is desired of the high-level controller. The message definitions in this section describe valid settings for these values.

After defining the values in the Configuration tab, the values in the Communication tab must be set as shown in Figure 4-3.

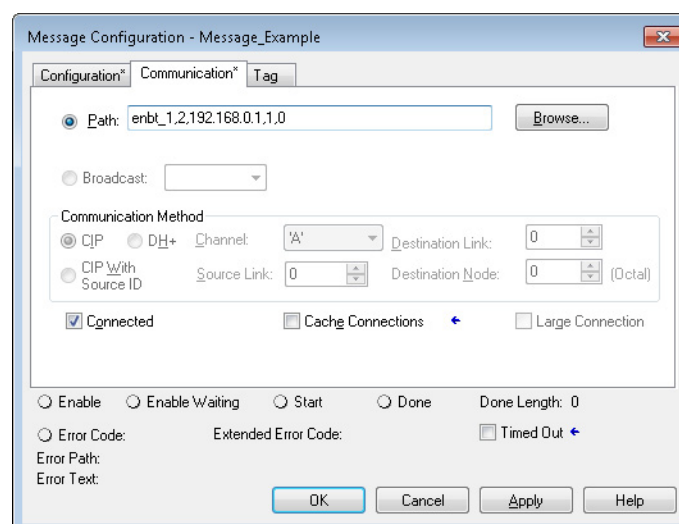


Figure 4-3: Message Communication Setup

In the communication tab, the important items to set are the following:

- The **Path** is formatted Local Module, Port, Address, Port, Address and may vary depending on where the ENBT module is installed in the host controller. The IP address that is listed in the path (the third item) must be the IP address of the high-level controller. See the host controller documentation for a complete description of the path syntax.

NOTE: If the HLC IP Address changes, every message configuration Path will need to change as well.

- The “Cache Connections” option must be turned off (unchecked).

Explicit Message Cyclic Send Glitch Workarounds

It is important to note that an odd behavior has been seen with explicit message sending, where the message continues to be sent cyclically once triggered in the logic. Two things must be done to prevent this behavior:

1. Make sure that Cache Connections is unchecked as shown in [Figure 4-3](#).
2. Use a one-shot (ONS) instruction in the Ladder Logic so the message only executes once when the rung with the message is active.

Host Controller to Node Controller Compatibility

If the Node Controller Software Image File for a transport system is upgraded to version 7.2.13 or newer, care must be taken to maintain compatibility with the host controller code. The **Use Extended Vehicle Status** must be configured properly for the newer version to support the *MMI_extended_vehicle_status* tag.

- Use the Node Controller Configuration File, which doesn't define **Use Extended Vehicle Status**, or use the Configurator Utility Tool to make sure that it is unselected (cleared). Either condition causes the node controller to support standard vehicle status (*MMI_vehicle_status*), which is passed from the node controller to the host controller.
- Rewrite the host controller code to support the extended vehicle status, and select **Use Extended Vehicle Status** in the Node Controller Configuration File. In this case extended vehicle status (*MMI_extended_vehicle_status*) is passed from the node controller to the host controller.

The host controller interface on the node controller doesn't support both standard and extended vehicle status simultaneously. Use the *MagneMotion System Configurator User Manual*, publication [MMI-UM046](#) to set or clear **Use Extended Vehicle Status**, which causes **PLC_use_extended_vehicle_status** to be true or false in the XML of the Node Controller Configuration File, which determines the vehicle status tag to use.

Host Controller to HLC Explicit Message Request Definitions

The command messages that are shown in [Table 4-1](#) are sent from the host controller to the HLC as asynchronous requests for the transport system to perform an action. To use these command messages the **Use an EtherNet/IP Controller for Host Control** option must be selected (checked) on the EtherNet/IP Host Controller Settings page in the Configurator Utility when creating the Node Controller Configuration File (see *MagneMotion System Configurator User Manual*, publication [MMI-UM046](#)).

For the HLC to know which host controller to communicate with, the IP address must be defined in the **Controller IP Address** option and the location of the controller must be defined in the **Controller CPU Slot** option on the EtherNet/IP Host Controller Settings page in the Configurator Utility when creating the Node Controller Configuration File.

Once the host controller is configured, the HLC is able to access the specific memory tags that are listed in this section. To interact properly with the HLC, all message tags that are listed in [Table 4-1](#) must be defined in the HLC logic as described.

- Required column indicates those tags that must always be defined.
- Use column indicates the motor types that can use the tag and any conditions that are related to the tag use.

These messages are implemented with the host controller message (MSG) command. Host controller message commands are implemented as CIP data table write explicit messages. These requests are responded to by the HLC by routing the command to the appropriate node controller for completion and updating the corresponding response tags in the host controller memory, which is shown in [Table 4-47](#). This implementation allows the flexibility of sending an array (a table) of data types (typically UDT) to the high-level controller to request an action that is described by the content of the items in the data table. Each specific message definition describes the meaning of each element in the message.

Table 4-1: Host Controller to HLC Explicit Message Requests

Message Name	Page
MMI_create_traffic_light_cmd	151
MMI_delete_traffic_light_cmd	154
MMI_generic_node_command	157
MMI_mgmt_nc_restart_cmd	160
MMI_mgmt_nc_set_config_cmd	162
MMI_motor_inverter_command	165
MMI_mp_link_command	169
MMI_mp_unlink_command	173

Table 4-1: Host Controller to HLC Explicit Message Requests

Message Name	Page
<i>MMI_node_controller_command</i>	176
<i>MMI_path_command</i>	180
<i>MMI_set_prop_power_state_cmd</i>	184
<i>MMI_set_traffic_light_cmd</i>	188
<i>MMI_sm_poll_command</i>	191
<i>MMI_sm_subscription_command</i>	197
<i>MMI_terminus_node_command</i>	202
<i>MMI_vehicle_command</i>	205
<i>MMI_vehicle_delete_order</i>	208
<i>MMI_vehicle_follow_order</i>	211
<i>MMI_vehicle_position_order</i>	216
<i>MMI_vehicle_station_order</i>	221

MMI_create_traffic_light_cmd

Source/Destination

Host Controller ➔ HLC

Purpose

Used with the Traffic Lights option to contain one or more entries of type `udt_MMI_create_traffic_light_cmd` used to create a traffic light at the specified position on the designated path. When a new traffic light is created, it is assigned the next available Traffic Light ID and set to green to allow vehicles to move beyond the traffic light position.

There is a limit of one traffic light per motor block and 32 traffic lights per path. The HLC rejects commands that attempt to create a traffic light on a motor block where a traffic light is already located or create more than 32 traffic lights on a path. The motor block lengths for each motor type are provided in [Table 3-1 on page 92](#) for reference.

Each command can be monitored via its corresponding `udt_MMI_traffic_light_cmd_status` entry in the [MMI_traffic_light_cmd_status](#) tag.

NOTE: The high-level controller, node controllers, and the path the traffic light resides on must be in the operational state.

These changes are not persistent. Any traffic lights created using this command must be recreated in the following situations; whenever the node controller is restarted or rebooted, or whenever any paths where traffic lights are located are reset.

The **Enable Traffic Lights** option must be selected (checked) on the EtherNet/IP Host Controller Settings page in the Configurator Utility when creating the Node Controller Configuration File (see *MagneMotion System Configurator User Manual*, publication [MMI-UM046](#)).

Support

This message is supported in the latest software release for the following product lines when **Enable Traffic Lights** is specified in the Node Controller Configuration File:

- MagneMover LITE transport systems.
- QuickStick transport systems.
- QuickStick HT transport systems.

UDT Field Descriptions

The UDT described in [Table 4-2](#) shows the type of each element that comprises the `MMI_create_traffic_light_cmd` tag. Each UDT field is described in more detail following the table.

Table 4-2: UDT Fields for `udt_MMI_create_traffic_light_cmd`

Field Name	Data Type	Style	Range
<code>path_id</code>	INT	Decimal	1...65535
<code>position</code>	REAL	Float	0.0...+41.0 (m, floating-point)
<code>command_index</code>	SINT	Decimal	0...31
<code>active_flag</code>	SINT	Hex	0x00, 0x01
<code>command_count</code>	DINT	Hex	0x0...0xFFFFFFFF
Data Type Size:		16 bytes	

UDT Field Details

`path_id` – The ID of the path where the traffic light is to be placed. The ID must be a nonzero positive integer that references a path that exists in the configuration.

`position` – The position (in meters) where a traffic light is to be placed, relative to the start of the specified path (expressed as a 32-bit single-precision floating-point number). Zero position is defined as the beginning of the path.

`command_index` – The index of the entry in the [MMI_traffic_light_cmd_status](#) tag to update when this `MMI_create_traffic_light_cmd` completes.

`active_flag` – This flag is used in configurations where a static table of `udt_MMI_create_traffic_light_cmd` entries is used and the same table is sent every time.

Value	Description
0x00	The HLC skips this entry.
0x01	The HLC processes this entry.

`command_count` – Host controller derived unique counter, which is used to confirm that the HLC has received the command and to determine command status. This counter provides a convenient handshake mechanism to determine command status by matching the counter with the fields in [MMI_traffic_light_cmd_status](#) tag.

Creating a DINT tag for use as a counter is suggested. The DINT tag can then be incremented and copied to `command_count` each time a new traffic light command is issued so the new

command can be tracked against the various count fields that are described in [MMI_traffic_light_cmd_status](#) tag.

Message Configuration

The source element create_traffic_light_cmd described in [Table 4-3](#) is an array of type udt_MMI_create_traffic_light_cmd. It can be a single tag with the Number of Elements set to 1 or it can be an array with the Number of Elements set to the maximum of 32.

Table 4-3: Message Configuration for create_traffic_light_cmd

Source Element Type	udt_MMI_create_traffic_light_cmd
Number of Elements	Minimum array size: 1 Maximum array size: 32
Destination Element	MMI_create_traffic_light_cmd

Response

After receiving the command, the HLC verifies the command parameters and updates [MMI_traffic_light_cmd_status](#) with either a “Command Accepted” response (0x00) or a “Command Rejected” response as appropriate. If the command is accepted, the HLC handles creating the traffic light on the targeted motor block and setting it to green. If the command is rejected, the response includes a status code identifying the reason for rejection (see *HLC Status Codes on page 333*).

On completion of the command, the HLC updates [MMI_traffic_light_cmd_status](#) with a command status response (see *HLC Status Codes on page 333*). On successful completion of the command, the HLC updates the corresponding traffic light status entry in [MMI_traffic_light_status](#) to reflect the status of the new traffic light. The ID of the new traffic light is returned in the **last_command_accepted_traffic_light_id** field of the [MMI_traffic_light_cmd_status](#) tag.

See Also

[MMI_delete_traffic_light_cmd on page 154](#)
[MMI_set_traffic_light_cmd on page 188](#)
[MMI_traffic_light_cmd_status on page 315](#)
[MMI_traffic_light_status on page 318](#)

MMI_delete_traffic_light_cmd

Source/Destination

Host Controller ➔ HLC

Purpose

Used with the Traffic Lights option to contain one or more entries of type `udt_MMI_delete_traffic_light_cmd` used to delete the specified traffic light. When a traffic light is deleted, the associated motor block is set to green to allow vehicles to move beyond the traffic light position.

Each command can be monitored via its corresponding `udt_MMI_traffic_light_cmd_status` entry in the [MMI_traffic_light_cmd_status](#) tag.

NOTE: The high-level controller, node controllers, and the path the traffic light resides on must be in the operational state.

The **Enable Traffic Lights** option must be selected (checked) on the EtherNet/IP Host Controller Settings page in the Configurator Utility when creating the Node Controller Configuration File (see *MagneMotion System Configurator User Manual*, publication [MMI-UM046](#)).

Support

This message is supported in the latest software release for the following product lines when **Enable Traffic Lights** is specified in the Node Controller Configuration File:

- MagneMover LITE transport systems.
- QuickStick transport systems.
- QuickStick HT transport systems.

UDT Field Descriptions

The UDT described in [Table 4-4](#) shows the type of each element that comprises the `MMI_delete_traffic_light_cmd` tag. Each UDT field is described in more detail following the table.

Table 4-4: UDT Fields for `udt_MMI_delete_traffic_light_cmd`

Name	Data Type	Style	Range
<code>traffic_light_id</code>	INT	Decimal	1...4096
<code>command_index</code>	SINT	Decimal	0...31
<code>active_flag</code>	SINT	Hex	0x00, 0x01
<code>command_count</code>	DINT	Hex	0x0...0xFFFFFFFF
Data Type Size:		8 bytes	

UDT Field Details

traffic_light_id – The ID of the specific traffic light to delete. The ID must be a nonzero positive integer that references a traffic light that exists in the configuration.

command_index – The index of the entry in the [MMI_traffic_light_cmd_status](#) tag to update when this `MMI_delete_traffic_light_cmd` completes.

active_flag – This flag is used in configurations where a static table of `udt_MMI_delete_traffic_light_cmd` entries is used and the same table is sent every time.

Value	Description
0x00	The HLC skips this entry.
0x01	The HLC processes this entry.

command_count – Host controller derived unique counter, which is used to confirm that the HLC has received the command and to determine command status. This counter provides a convenient handshake mechanism to determine command status by matching the counter with the fields in [MMI_traffic_light_cmd_status](#) tag.

Creating a DINT tag for use as a counter is suggested. The DINT tag can then be incremented and copied to `command_count` each time a new delete traffic light command is issued so the new command can be tracked against the various count fields that are described in [MMI_traffic_light_cmd_status](#) tag.

Message Configuration

The source element `delete_traffic_light_cmd` described in [Table 4-5](#) is an array of type `udt_MMI_delete_traffic_light_cmd`. It can be a single tag with the Number of Elements set to 1 or it can be an array with the Number of Elements set to the maximum of 32.

Table 4-5: Message Configuration for `delete_traffic_light_cmd`

Source Element Type	<code>udt_MMI_delete_traffic_light_cmd</code>
Number of Elements	Minimum array size: 1 Maximum array size: 32
Destination Element	<code>MMI_delete_traffic_light_cmd</code>

Response

After receiving the command, the HLC verifies the command parameters and updates [MMI_traffic_light_cmd_status](#) with either a “Command Accepted” response (0x00) or a “Command Rejected” response as appropriate. If the command is accepted, the HLC handles deleting the specified traffic light and setting the motor block to green on the targeted motor (see [HLC Status Codes on page 333](#)).

On completion of the command, the HLC updates [MMI_traffic_light_cmd_status](#) with a command status response (see [HLC Status Codes on page 333](#)). On successful completion of the command, the HLC updates the corresponding traffic light status entry in [MMI_traffic_light_status](#) to reflect the deletion of the traffic light.

See Also

[MMI_create_traffic_light_cmd on page 151](#)
[MMI_set_traffic_light_cmd on page 188](#)
[MMI_traffic_light_cmd_status on page 315](#)
[MMI_traffic_light_status on page 318](#)

MMI_generic_node_command

Source/Destination

Host Controller ➔ HLC

Purpose

Contains one or more entries of type `udt_MMI_generic_node_command` that specifies a node command. In this version of the node controller software, the only supported command subtype is to enable or disable a node, which is only used for Gateway Nodes.

Each node command can be monitored via its corresponding `udt_MMI_node_command_status` entry in the [MMI_node_command_status](#) tag. The status of the nodes can be monitored via the entries in the [MMI_node_status](#) tag.

NOTE: The high-level controller, node controllers, and paths must be in the operational state.

These changes are not persistent. If the node controller is restarted or rebooted, the Gateway Nodes default to disabled.

Support

This message is supported in the latest software release for the following product lines:

- MagneMover LITE transport systems.
- QuickStick transport systems.
- QuickStick HT transport systems.

UDT Field Descriptions

The UDT described in [Table 4-6](#) shows the type of each element that comprises the `MMI_generic_node_command` tag. Each UDT field is described in more detail following the table.

Table 4-6: UDT Fields for `udt_MMI_generic_node_command`

Name	Data Type	Style	Range
node_id	INT	Decimal	0...65535
node_command_subtype	SINT	Hex	0x00
command_data	SINT	Hex	0x00, 0x01
command_count	DINT	Hex	0x0...0xFFFFFFFF
active_flag	SINT	Hex	0x00, 0x01
Data Type Size:		12 bytes	

UDT Field Details

node_id – The ID of the node for this command to operate on. The ID must be a nonzero positive integer that references a node that exists in the configuration.

node_command_subtype – Supported node command subtypes.

Value	Description
0x00	Node enable or disable command

command_data – Specifies the node enable/disable action.

Value	Description
0x00	Disable the node, prevents vehicles from crossing the specified Gateway Node.
0x01	Enable the node, allows vehicles to cross the specified Gateway Node.

command_count – Host controller derived unique counter, which is used to confirm that the HLC has received the command and to determine command status. This counter provides a convenient handshake mechanism to determine command status by matching the counter with the fields in [MMI_node_command_status](#) tag.

Creating a DINT tag for use as a counter is suggested. The DINT tag can then be incremented and copied to command_count each time a new node command is issued so the new command can be tracked against the various count fields that are described in [MMI_node_command_status](#) tag.

active_flag – This flag is used in configurations where a static table of udt_MMI_generic_node_command entries is used and the same table is sent every time.

Value	Description
0x00	The HLC skips this entry.
0x01	The HLC processes this entry.

Message Configuration

The source element generic_node_command described in [Table 4-7](#) is an array of type udt_MMI_generic_node_command. It can be a single tag with the Number of Elements set to 1 or it can be an array with the Number of Elements set to the number of nodes in the transport system.

Table 4-7: Message Configuration for generic_node_command

Source Element Type	udt_MMI_generic_node_command
Number of Elements	Minimum array size: 1 Maximum array size: max Nodes
Destination Element	MMI_generic_node_command

Response

After receiving the command, the HLC verifies the command parameters and updates [MMI_node_command_status](#) with either a “Command Accepted” response (0x00) or a “Command Rejected” response as appropriate. If the command is accepted, it is forwarded to the appropriate node controller for execution. If the command is rejected, the response includes a rejection code identifying the reason for rejection (see *HLC Status Codes* [on page 333](#)).

After the command is executed, the status of the node may change. Any change is reported in the [MMI_node_status](#) tag in bit 4 of the [device_status](#) field.

See Also

[MMI_node_command_status](#) [on page 261](#)

[MMI_node_status](#) [on page 269](#)

MMI_mgmt_nc_restart_cmd

Source/Destination

Host Controller ➔ HLC

Purpose

Used with the NC Remote Management option, contains one tag of type *udt_MMI_mgmt_nc_restart_cmd*, which is used to restart all Node Controllers in the transport system.

Each restart node controllers command can be monitored via its corresponding *udt_MMI_mgmt_nc_cmd_status* entry in the [MMI_mgmt_nc_cmd_status](#) tag.

NOTE: The high-level controller and node controllers must be in the operational state.

The **Enable NC Remote Management** option must be selected (checked) on the EtherNet/IP Host Controller Settings page in the Configurator Utility when creating the Node Controller Configuration File (see *MagneMotion System Configurator User Manual*, publication [MMI-UM046](#)).

Support

This message is supported in the latest software release for the following product lines when **Enable NC Remote Management** is specified in the Node Controller Configuration File:

- MagneMover LITE transport systems.
- QuickStick transport systems.
- QuickStick HT transport systems.

UDT Field Descriptions

The UDT described in [Table 4-8](#) shows the type of each element that comprises the *MMI_mgmt_nc_restart_cmd* tag. Each UDT field is described in more detail following the table.

Table 4-8: UDT Fields for udt_MMI_mgmt_nc_restart_cmd

Name	Data Type	Style	Range
command_count	DINT	Hex	0x0...0xFFFFFFFF
Data Type Size:		4 bytes	

UDT Field Details

command_count – Host controller derived unique counter, which is used to confirm that the HLC has received the command and to determine command status. This counter provides a convenient handshake mechanism to determine command status by matching the counter with the fields in [MMI_mgmt_nc_cmd_status](#) tag.

Creating a DINT tag for use as a counter is suggested. The DINT tag can then be incremented and copied to command_count each time a new Restart Node Controller command is issued so the new command can be tracked against the various count fields that are described in [MMI_mgmt_nc_cmd_status](#) tag.

Message Configuration

The source element mgmt_nc_restart_cmd described in [Table 4-9](#) is single element of type udt_MMI_mgmt_nc_restart_cmd.

Table 4-9: Message Configuration for mgmt_nc_restart_cmd

Source Element Type	udt_MMI_mgmt_nc_restart_cmd
Number of Elements	Minimum array size: 1 Maximum array size: 1
Destination Element	MMI_mgmt_nc_restart_cmd

Response

After receiving the command, the HLC verifies the command parameters and updates [MMI_mgmt_nc_cmd_status](#) with either a “Command Accepted” response (0x00) or a “Command Rejected” response as appropriate. If the command is accepted, the HLC handles restarting all Node Controllers in the transport system. If the command is rejected, the response includes a rejection code identifying the reason for rejection (see *HLC Status Codes* [on page 333](#)).

On completion of the command, the HLC updates the [MMI_mgmt_nc_cmd_status](#) tag with a Command Complete or a Command Failed response.

See Also

[MMI_mgmt_nc_set_config_cmd on page 162](#)
[MMI_extended_hlc_status on page 228](#)
[MMI_extended_nc_status on page 231](#)
[MMI_mgmt_nc_cmd_status on page 246](#)

MMI_mgmt_nc_set_config_cmd

Source/Destination

Host Controller → HLC

Purpose

Used with the NC Remote Management option, contains one tag of type *udt_MMI_mgmt_nc_set_config_cmd*, which is used to set the specified managed Node Controller Configuration File as active on all Node Controllers in the transport system.

Managed Node Controller Configuration Files allows the creation of up to 20 custom files with different system behavior (for example different PID settings). Each file is assigned a unique Configuration ID when it is uploaded to the HLC.

Each set node controller configuration command can be monitored via its corresponding *udt_MMI_mgmt_nc_cmd_status* entry in the *MMI_mgmt_nc_cmd_status* tag.

NOTE: The high-level controller and node controllers must be in the operational state.

The **Enable NC Remote Management** option must be selected (checked) on the EtherNet/IP Host Controller Settings page in the Configurator Utility when creating the Node Controller Configuration File (see *MagneMotion System Configurator User Manual*, publication [MMI-UM046](#)).

Support

This message is supported in the latest software release for the following product lines when **Enable NC Remote Management** is specified in the Node Controller Configuration File:

- MagneMover LITE transport systems.
- QuickStick transport systems.
- QuickStick HT transport systems.

UDT Field Descriptions

The UDT described in [Table 4-10](#) shows the type of each element that comprises the MMI_mgmt_nc_set_config_cmd tag. Each UDT field is described in more detail following the table.

Table 4-10: UDT Fields for udt_MMI_mgmt_nc_set_config_cmd

Name	Data Type	Style	Range
configuration_id	SINT	Decimal	1...20
command_count	DINT	Hex	0x0...0xFFFFFFFF
Data Type Size:		8 bytes	

UDT Field Details

configuration_id – The ID of the managed Node Controller Configuration File to set as active on all Node Controllers in the transport system.

command_count – Host controller derived unique counter, which is used to confirm that the HLC has received the command and to determine command status. This counter provides a convenient handshake mechanism to determine command status by matching the counter with the fields in [MMI_mgmt_nc_cmd_status](#) tag.

Creating a DINT tag for use as a counter is suggested. The DINT tag can then be incremented and copied to command_count each time a new Set Node Controller Configuration command is issued so the new command can be tracked against the various count fields that are described in [MMI_mgmt_nc_cmd_status](#) tag.

Message Configuration

The source element mgmt_nc_set_config_cmd described in [Table 4-11](#) is single element of type udt_MMI_mgmt_nc_set_config_cmd.

Table 4-11: Message Configuration for mgmt_nc_set_config_cmd

Source Element Type	udt_MMI_mgmt_nc_set_config_cmd
Number of Elements	Minimum array size: 1 Maximum array size: 1
Destination Element	MMI_mgmt_nc_set_config_cmd

Response

After receiving the command, the HLC verifies the command parameters and updates *MMI_mgmt_nc_cmd_status* with either a “Command Accepted” response (0x00) or a “Command Rejected” response as appropriate. If the command is accepted, the HLC handles distributing and setting the specified Node Controller Configuration File as active on all Node Controllers in the transport system. If the command is rejected, the response includes a rejection code identifying the reason for rejection (see *HLC Status Codes on page 333*).

On completion of the command, the HLC updates the *MMI_mgmt_nc_cmd_status* tag with a Command Complete or a Command Failed response.

See Also

MMI_mgmt_nc_restart_cmd on page 160

MMI_extended_hlc_status on page 228

MMI_extended_nc_status on page 231

MMI_mgmt_nc_cmd_status on page 246

MMI_motor_inverter_command

Source/Destination

Host Controller ➔ HLC

Purpose

Used with the Motor Inverter Command option, contains one or more entries of type `udt_MMI_motor_inverter_command` to enable/disable one or more blocks driven by a motor controller (inverter) associated with a QuickStick HT motor. Each motor inverter command can be monitored via its corresponding `udt_MMI_motor_inverter_cmd_status` entry in the *MMI_motor_inverter_cmd_status* tag. The status of the individual inverters can be determined using *MMI_path_qs_ht_faults_status* tag.

Motor blocks for QuickStick HT motors are configured as follows:

- In a 1 m QSHT motor, the upstream block is Block 1 and the downstream block is Block 2.
- In a 1/2 m QSHT motor, there is only one block, which is Block 1.
- When two 1/2 m QSHT motors are configured as a QSHT Dual Half motor, the upstream stator is configured as Block 1 and the downstream stator is configured as Block 2.
- In a 1/2 m double-wide QSHT motor, there is only one block, which is Block 1.

NOTE: The high-level controller, node controllers, path, and the motors must be in the operational state. If any component required to execute the command is not in an operational state, the command is rejected.

These changes are not persistent. If the motor is restarted or rebooted, the inverters default to enabled.

The **Enable Motor Inverter Command** option must be selected (checked) on the EtherNet/IP Host Controller Settings page in the Configurator Utility when creating the Node Controller Configuration File (see the *MagneMotion System Configurator User Manual*, publication [MMI-UM046](#)).

Support

This message is supported in the latest software release for the following product lines when **Enable Motor Inverter Command** is specified in the Node Controller Configuration File:

- QuickStick HT transport systems.

UDT Field Descriptions

The UDT described in [Table 4-12](#) shows the type of each element that comprises the MMI_motor_inverter_command tag. Each UDT field is described in more detail following the table.

Table 4-12: UDT Fields for udt_MMI_motor_inverter_command

Name	Data Type	Style	Range
path_id	INT	Decimal	1...65535
motor_id	INT	Decimal	0...22
inverter_id	INT	Decimal	0...2
inverter_control	SINT	Decimal	0, 1
active_flag	SINT	Hex	0x00, 0x01
command_count	DINT	Hex	0x0...0xFFFFFFFF
Data Type Size:		12 bytes	

UDT Field Details

path_id – The ID of the Path where the specified motors are located. The ID must be a non-zero positive integer referencing a Path that exists in the configuration.

motor_id – The ID of the motors selected by this command.

Value	Description
0	All motors on the specified path are selected
1...22	Only the specified motor is selected.

inverter_id – The ID of the inverters (motor controller) selected by this command.

Value	Description
0	All inverters (motor controllers) on the specified motors are selected.
1, 2	Only the specified inverter is selected.

inverter_control – Controls whether to enable or disable the inverters (motor controller) that are targeted by this command.

Value	Description
0	The specified inverter (motor controller) is enabled.
1	The specified inverter (motor controller) is disabled.

active_flag – This flag is used in configurations where a static table of `udt_MMI_motor_inverter_command` entries is used and the same table is sent every time.

Value	Description
0x00	The HLC skips this entry.
0x01	The HLC processes this entry.

command_count – Host controller derived unique counter, which is used to confirm that the HLC has received the command and to determine command status. This counter provides a convenient handshake mechanism to determine command status by matching the counter with the fields in [MMI_motor_inverter_cmd_status](#) tag.

Creating a DINT tag for use as a counter is suggested. The DINT tag can then be incremented and copied to `command_count` each time a new motor inverter command is issued so the new command can be tracked against the various count fields that are described in [MMI_motor_inverter_cmd_status](#) tag.

Message Configuration

The source element `motor_inverter_command` described in [Table 4-13](#) is an array of type `udt_MMI_motor_inverter_command`. It can be a single tag with the Number of Elements set to 1 or it can be an array with the Number of Elements set to the maximum of 32.

Table 4-13: Message Configuration for motor_inverter_command

Source Element Type	<code>udt_MMI_motor_inverter_command</code>
Number of Elements	Minimum array size: 1 Maximum array size: 32
Destination Element	<code>MMI_motor_inverter_command</code>

Response

After receiving the command, the HLC verifies the command parameters and updates *MMI_motor_inverter_cmd_status* with either a “Command Accepted” response (0x00) or a “Command Rejected” response as appropriate. If the command is accepted, the HLC handles enabling/disabling the motor controller on the targeted motors. If the command is rejected, the response includes a rejection code identifying the reason for rejection (see *HLC Status Codes on page 333*).

On completion of the command, the HLC updates the *MMI_motor_inverter_cmd_status* tag with a Command Complete or a Command Failed response.

See Also

MMI_motor_inverter_cmd_status on page 249

MMI_mp_link_command

Source/Destination

Host Controller ➔ HLC

Purpose

Contains one or more entries of type `udt_MMI_mp_link_command` that specify which control path and peer path ends are now physically aligned and can be “linked” to allow vehicles to navigate a path junction.

The initial link command, sent to form a junction between two unlinked path ends, establishes the control path. Subsequent link commands can be issued to modify the **last_allowed_vehicle_id** but must specify the same control path and peer path IDs used initially to establish the junction.

Each link command can be monitored via its corresponding `udt_MMI_mp_command_status` entry in the [*MMI_mp_command_status*](#) tag.

NOTE: The high-level controller, node controllers, and the paths that are specified must be in the operational state. If any component required to execute the command is not in an operational state, the command is rejected.

Support

This message is supported in the latest software release for the following product lines:

- QuickStick transport systems.
- QuickStick HT transport systems.

UDT Field Descriptions

The UDT described in [Table 4-14](#) shows the type of each element that comprises the MMI_mp_link_command tag. Each UDT field is described in more detail following the table.

Table 4-14: UDT Fields for udt_MMI_mp_link_command

Name	Data Type	Style	Range
node_id	INT	Decimal	1...65535
control_path_id	INT	Decimal	1...65535
peer_path_id	INT	Decimal	1...65535
last_allowed_vehicle_id	INT	Decimal	0...65535
alignment_request_count	INT	Hex	0x01...0xFFFF
command_count	DINT	Hex	0x0...0xFFFFFFFF
active_flag	SINT	Hex	0x00, 0x01
Data Type Size:		20 bytes	

UDT Field Details

node_id – The ID of the Moving Path node that this link command operates on.

control_path_id – The ID of the control path end to link to the specified peer path end to form a path junction. Once linked, this path remains the Control Path for the life of the junction until the junction transitions to the unlinked state.

peer_path_id – The ID of the peer path end to link to the specified control path end to form a path junction. Once linked, this path remains the Peer Path for the life of the junction until the junction transitions to the unlinked state.

NOTE: A linking path can be defined as a control path for one node and a peer path for another node. However, it should not be designated as a control path for two nodes or as a peer path for both nodes.

last_allowed_vehicle_id – Used to set the conditions for unlinking the path junction.

Value	Description
0	The HLC keeps the Control and Peer Paths linked until the host controller sends an MMI_mp_unlink_command . Vehicles approaching the junction are granted permission to navigate the junction if the Control Path offers an equivalent route to the vehicle's destination. If the approaching vehicle requires another path alignment, the junction is unlinked as soon as there are no vehicles navigating the node and a new alignment request is sent to the host controller.
1...65535	Once the last allowed vehicle begins to navigate the path junction, the Control Path end transitions to the linked_unlink_pending state and no other vehicles are granted permission to navigate the junction. The HLC unlinks the path junction as soon as all navigating vehicles are clear of the path junction.

alignment_request_count – A sequence count, unique to the most recent alignment request for a path end, which the HLC increments whenever alignment for a path end is requested.

The host controller logic can use this field to track which alignment request a link command is responding to by including the [alignment_request_count](#) from a [MMI_mp_path_end_status](#) tag in the link command.

command_count – Host controller derived unique counter, which is used to confirm that the HLC has received the command and to determine command status. This counter provides a convenient handshake mechanism to determine command status by matching the counter with the fields in [MMI_mp_command_status](#) tag.

Creating a DINT tag for use as a counter is suggested. The DINT tag can then be incremented and copied to command_count each time a new moving path link command is issued so the new command can be tracked against the various count fields that are described in [MMI_mp_command_status](#) tag.

active_flag – This flag is used in configurations where a static table of udt_MMI_mp_link_command entries is used and the same table is sent every time. This can be useful for configurations where a static table of udt_MMI_mp_link_command is used and the same table is sent every time with the active_flag for each entry set or cleared to direct the HLC whether to process the entry or not.

Value	Description
0x00	The HLC skips this entry.
0x01	The HLC processes this entry.

Message Configuration

The source element `mp_link_command` described in [Table 4-15](#) is an array of type `udt_MMI_mp_link_command`. It can be a single tag with the Number of Elements set to 1 or it can be an array with the Number of Elements set to the maximum number of path connections on the node controller.

Table 4-15: Message Configuration for `mp_link_command`

Source Element Type	<code>udt_MMI_mp_link_command</code>
Number of Elements	Minimum array size: 1 Maximum array size: MP Node Member Path Limit (1)*
Destination Element	<code>MMI_mp_link_command</code>

* The Member Path Limit is limited to the maximum number of path connections on the node controller. All node controllers support Ethernet communication with the host controller and the motors, and depending on the model, provide RS-422 ports for communication with the motors.

Response

After receiving the command, the HLC verifies the command parameters and updates [MMI_mp_command_status](#) with either a “Command Accepted” response (0x00) or a “Command Rejected” response as appropriate. If the command is accepted, the HLC handles linking the specified path ends for the selected Moving Path node. If the command is rejected, the response includes a rejection code identifying the reason for rejection (see *HLC Status Codes on page 333*).

On completion of the command, the HLC updates [MMI_mp_command_status](#) with a Command Complete or a Command Failed response. If the command failed, the response includes a code identifying the reason for failure (see *HLC Status Codes on page 333*).

Additionally, the HLC updates one or more path end status entries in the [MMI_mp_path_end_status](#) array to reflect changes resulting from executing this link command.

See Also

[MMI_mp_unlink_command on page 173](#)

[MMI_mp_command_status on page 251](#)

[MMI_mp_path_end_status on page 254](#)

MMI_mp_unlink_command

Source/Destination

Host Controller ➔ HLC

Purpose

Contains one or more entries of type `udt_MMI_mp_unlink_command` used to unlink one or more path junctions that are associated with the specified Moving Path node.

Each unlink command can be monitored via its corresponding `udt_MMI_mp_command_status` entry in the *MMI_mp_command_status* tag.

NOTE: The high-level controller, node controllers, and the paths that are specified must be in the operational state. If any component required to execute the command is not in an operational state, the command is rejected.

Support

This message is supported in the latest software release for the following product lines:

- QuickStick transport systems.
- QuickStick HT transport systems.

UDT Field Descriptions

The UDT described in [Table 4-16](#) shows the type of each element that comprises the `MMI_mp_unlink_command` tag. Each UDT field is described in more detail following the table.

Table 4-16: UDT Fields for `udt_MMI_mp_unlink_command`

Name	Data Type	Style	Range
node_id	INT	Decimal	1...65535
control_path_id	INT	Decimal	0...65535
command_count	DINT	Hex	0x0...0xFFFFFFFF
active_flag	SINT	Hex	0x00, 0x01
Data Type Size:		12 bytes	

UDT Field Details

node_id – The ID of the Moving Path node that this unlink command operates on.

control_path_id – The ID of one or more path ends to unlink from path junctions that are associated with the specified Moving Path node. If a vehicle is navigating a path junction, the path junction transitions to the *linked_unlink_pending* state; otherwise, the path junction transitions to the *unlinked* state. See [Table 4-57, Moving Path Node Path End States, on page 256](#) for the description of the path end states.

Value	Description
0	All member path ends are unlinked for the specified Moving Path node.
1...65535	Only the specified control path end is unlinked. Additionally, the control path_id specified must be the same control path that was used initially to establish the junction.

command_count – Host controller derived unique counter, which is used to confirm that the HLC has received the command and to determine command status. This counter provides a convenient handshake mechanism to determine command status by matching the counter with the fields in *MMI_mp_command_status* tag.

Creating a DINT tag for use as a counter is suggested. The DINT tag can then be incremented and copied to command_count each time a new moving path link command is issued so the new command can be tracked against the various count fields that are described in *MMI_mp_command_status* tag.

active_flag – This flag is used in configurations where a static table of udt_MMI_mp_unlink_command entries is used and the same table is sent every time. This can be useful for configurations where a static table of udt_MMI_mp_link_command is used and the same table is sent every time with the active_flag for each entry set or cleared to direct the HLC whether to process the entry or not.

Value	Description
0x00	The HLC skips this entry.
0x01	The HLC processes this entry.

Message Configuration

The source element `mp_unlink_command` described in [Table 4-17](#) is an array of type `udt_MMI_mp_unlink_command`. It can be a single tag with the Number of Elements set to 1 or it can be an array with the Number of Elements set to the maximum number of path connections on the node controller.

Table 4-17: Message Configuration for `mp_unlink_command`

Source Element Type	<code>udt_MMI_mp_unlink_command</code>
Number of Elements	Minimum array size: 1 Maximum array size: MP Node Member Path Limit (1)*
Destination Element	<code>MMI_mp_unlink_command</code>

* The Member Path Limit is limited to the maximum number of path connections on the node controller. All node controllers support Ethernet communication with the host controller and the motors, and depending on the model, provide RS-422 ports for communication with the motors.

Response

After receiving the command, the HLC verifies the command parameters and updates [`MMI_mp_command_status`](#) with either a “Command Accepted” response (0x00) or a “Command Rejected” response as appropriate. If the command is accepted, the HLC handles unlinking the specified path junctions for the selected Moving Path node. If the command is rejected, the response includes a rejection code identifying the reason for rejection (see *HLC Status Codes on page 333*).

On completion of the command, the HLC updates the [`MMI_mp_command_status`](#) tag with a Command Complete or a Command Failed response. If the command failed, the response includes a code identifying the reason for failure (see *HLC Status Codes on page 333*).

Additionally, the HLC updates one or more path end status entries in the [`MMI_mp_path_end_status`](#) array to reflect changes resulting from executing this unlink command.

See Also

[MMI_mp_link_command on page 169](#)
[MMI_mp_command_status on page 251](#)
[MMI_mp_path_end_status on page 254](#)

MMI_node_controller_command

Source/Destination

Host Controller ➔ HLC

Purpose

Used with the Digital I/O Commands option, contains one or more entries of type `udt_MMI_node_controller_dio_command` that specifies a digital I/O command to perform on one or more node controllers in the transport system.

Each digital I/O command can be monitored via its corresponding `udt_MMI_node_controller_cmd_status` entry in the [MMI_node_controller_cmd_status](#) tag. The status of the digital I/O can be monitored via the entries in the [MMI_node_controller_dio_status](#) tag.

NOTE: The high-level controller and node controllers must be in the operational state.

The **Enable Digital I/O Commands** option must be selected (checked) on the EtherNet/IP Host Controller Settings page in the Configurator Utility when creating the Node Controller Configuration File (see the *MagneMotion System Configurator User Manual*, publication [MMI-UM046](#)).

Support

This message is supported in the latest software release for the following product lines when **Enable Digital I/O Commands** is specified the Node Controller Configuration File:

- MagneMover LITE transport systems.
- QuickStick transport systems.
- QuickStick HT transport systems.

UDT Field Descriptions

The UDT described in [Table 4-18](#) shows the type of each element that comprises the MMI_node_controller_command tag. Each UDT field is described in more detail following the table.

Table 4-18: UDT Fields for udt_MMI_node_controller_dio_command

Name	Data Type	Style	Range
node_controller_id	INT	Decimal	1...96
output_mask	DINT	Hex	0x0...0xFFFFFFFF
output_data	DINT	Hex	0x0...0xFFFFFFFF
command_count	DINT	Hex	0x0...0xFFFFFFFF
active_flag	SINT	Hex	0x00, 0x01
Data Type Size:		20 bytes	

UDT Field Details

node_controller_id – The ID of the node controller for which the digital I/O outputs are being set. The ID must be a nonzero positive integer that references a node controller that exists in the configuration.

output_mask – Specifies which bits of the Output Data field to write into the digital I/O outputs on the node controller. For each bit set to “1” in the Output Data Mask field, the corresponding bit in the Output Data field is written into the node controller’s digital I/O outputs. For each bit set to “0” in the Output Data Mask field, the corresponding bit in the node controller’s digital I/O outputs is not changed.

The Output Data Mask field must only specify digital I/O outputs implemented by the specified node controller (see [Table 4-19](#)).

output_data – The digital I/O output data to set, consisting of 1 to 32 individual outputs. For each bit set to “1” in the Output Data Mask field, the corresponding bit in the Output Data field is written into the node controller’s digital I/O outputs.

If a node controller does not implement all 32 digital I/O outputs, place output data in the available low-order bits of the Output Data field. [Table 4-19](#) lists the available digital I/O output bits for each node controller type.

Table 4-19: Node Controller Digital Outputs

Node Controller	Available Digital I/O Outputs
NC LITE*	None
NC-12	0x0000FFFF
NC-E Series A	0x0000000F
NC-E Series B	0x000000FF
NC-S*	None

* The NC LITE and NC-S node controllers do not provide any digital I/O.

command_count – Host controller derived unique counter, which is used to confirm that the HLC has received the command and to determine command status. This counter provides a convenient handshake mechanism to determine command status by matching the counter with the fields in [MMI_node_controller_cmd_status](#) tag.

Creating a DINT tag for use as a counter is suggested. The DINT tag can then be incremented and copied to command_count each time a new node controller command is issued so the new command can be tracked against the various count fields that are described in [MMI_node_controller_cmd_status](#) tag.

active_flag – This flag is used in configurations where a static table of udt_MMI_node_controller_dio_command entries is used and the same table is sent every time.

Value	Description
0x00	The HLC skips this entry.
0x01	The HLC processes this entry.

Message Configuration

The source element `node_controller_command` described in [Table 4-20](#) is an array of type `udt_MMI_node_controller_dio_command`. It can be a single tag with the Number of Elements set to 1 or it can be an array with the Number of Elements set to the number of node controllers in a transport system. When the source element is configured to use an array, the message can send multiple commands to the HLC with one message.

Table 4-20: Message Configuration for `node_controller_command`

Source Element Type	<code>udt_MMI_node_controller_dio_command</code>
Number of Elements	Minimum array size: 1 Maximum array size: max Node Controllers
Destination Element	<code>MMI_node_controller_command</code>

Response

After receiving the command, the HLC verifies the command parameters and updates [`MMI_node_controller_cmd_status`](#) with either a “Command Accepted” response (0x00) or a “Command Rejected” response as appropriate. If the command is accepted, it is forwarded to the appropriate node controller for execution. If the command is rejected, the response includes a rejection code identifying the reason for rejection (see [HLC Status Codes on page 333](#)).

After the command is executed, the status of the node controller digital I/O may change. Any change is shown in the [`MMI_node_controller_dio_status`](#) tag.

See Also

[MMI_node_controller_cmd_status on page 263](#)

[MMI_node_controller_dio_status on page 265](#)

MMI_path_command

Source/Destination

Host Controller ➔ HLC

Purpose

Contains one or more entries of type `udt_MMI_path_command` that specifies one or more commands to perform on one or more paths in the transport system.

Each path command can be monitored via its corresponding `udt_MMI_path_command_status` entry in the [MMI_path_command_status](#) tag. The status of the paths can be monitored via the entries in the [MMI_path_status](#) tag.

NOTE: The high-level controller, node controllers, and paths must be in the operational state.

Support

This message is supported in the latest software release for the following product lines:

- MagneMover LITE transport systems.
- QuickStick transport systems.
- QuickStick HT transport systems.

UDT Field Descriptions

The UDT described in [Table 4-21](#) shows the type of each element that comprises the `MMI_path_command` tag. Each UDT field is described in more detail following the table.

Table 4-21: UDT Fields for `udt_MMI_path_command`

Name	Data Type	Style	Range
path_id	INT	Decimal	0...65535
command	SINT	Hex	0xB2...0xB4, 0xB8, 0xBC, 0xBD
active_flag	SINT	Hex	0x00, 0x01
command_count	DINT	Hex	0x0...0xFFFFFFFF
Data Type Size:		8 bytes	

UDT Field Details

path_id – The ID of the path to execute the command on. The ID must be either a nonzero positive integer that references a path that exists in the configuration, or zero to execute the command on all paths in the transport system. See the note under [Message Configuration](#) about using zero.

command – The command to execute on the specified paths.

Value	Description
0xB2	Startup – Initiates the start-up sequence for locating all vehicles on the specified path. If the path_id specified is zero, the start up sequence for locating all vehicles is executed on all paths in the transport system.
0xB3	Resume Motion – Enables motion after it has been disabled with a suspend motion command on the specified path. If the path_id specified is zero, motion is resumed on all paths in the transport system. When the Resume Motion command is issued, all motion that is based on all currently active motion commands continues.
0xB4	Suspend Motion – Suspends all motion on the specified path. If the path_id specified is zero, motion is suspended on all paths in the transport system. Vehicles immediately decelerate to a controlled stop at the location that they were last given permission to move to. Vehicles in motion within a keep-out area do not stop until they exit the keep-out area. Previously commanded motion does not resume until a Resume Motion command is received. The control loop is still enabled while motion is suspended holding all vehicles in place. Additional vehicles are not allowed to enter the path while it is suspended.
0xB8	Reset – Performs a cold reset of all motors on the specified path. The runtime software in each motor is restarted and all records for vehicles on the specified paths are deleted, and the network chip inside each motor is reset. If the path_id specified is zero, all paths in the transport system are reset and all vehicles records deleted. After any path is reset, a Startup command must be executed on it to bring it to the operational state.
0xBC	FastStop – Suspends all motion on the specified path. If the path ID specified is zero, motion is suspended on all paths in the transport system. Vehicles immediately decelerate with maximum thrust opposing motion. Vehicles in motion within a keep-out area stop within the keep-out area. Previously commanded motion does not resume until a Resume Motion command is received. The control loop is still enabled while motion is suspended holding all vehicles in place. Additional vehicles are not allowed to enter the path while it is suspended. NOTE: This command is not supported on MagneMover LITE transport systems.
0xBD	Warm Reset – Performs a warm reset of all motors on the specified path. The runtime software in each motor is restarted and all records for vehicles on the specified paths are deleted. However, the network chip inside each motor is not reset, which allows the network link to be operational through the reset. If the path_id specified is zero, all paths in the transport system are reset and all vehicles records deleted. After any path is reset, a Startup command must be executed on it to bring it to the operational state.

active_flag – This flag is used in configurations where a static table of udt_MMI_path_ command entries is used and the same table is sent every time.

Value	Description
0x00	The HLC skips this entry.
0x01	The HLC processes this entry.

command_count – Host controller derived unique counter, which is used to confirm that the HLC has received the command and to determine command status. This counter provides a convenient handshake mechanism to determine command status by matching the counter with the fields in [MMI_path_command_status](#) tag.

Creating a DINT tag for use as a counter is suggested. The DINT tag can then be incremented and copied to command_count each time a new path command is issued so the new command can be tracked against the various count fields that are described in [MMI_path_command_status](#) tag.

Message Configuration

The source element path_command described in [Table 4-22](#) is an array of type udt_MMI_path_command. It can be a single tag with the Number of Elements set to 1 or it can be an array with the Number of Elements set to the number of paths in a transport system. When the source element is configured to use an array, the message can send multiple commands to the HLC with one message.

When it is desired to apply a command to all paths in the transport system, it is recommended a single UDT element is sent with its path_id set to zero and the command, command_count, and active_flag set appropriately to apply the command to all paths in the transport system.

Table 4-22: Message Configuration for path_command

Source Element Type	udt_MMI_path_command
Number of Elements	Minimum array size: 1 Maximum array size: 256
Destination Element	MMI_path_command

NOTE: When the HLC receives a path command message with multiple elements of type udt_MMI_path_command in it, only the first element is checked to see if its path_id field is zero to determine if a command is to be applied to all paths in the transport system. If any other elements have the path_id field set to zero, they are ignored even if the active bit is set. When multiple elements are sent and the first element path_id is zero, the first element command is applied to all paths and the HLC drops all remaining elements in the message.

Response

After receiving the command, the HLC verifies the command parameters and updates *MMI_path_command_status* with either a “Command Accepted” response (0x00) or a “Command Rejected” response as appropriate. If the command is accepted, it is forwarded to the appropriate node controller for execution. If the command is rejected, the response includes a rejection code identifying the reason for rejection (see *HLC Status Codes on page 333*).

For Startup (0xB2), Reset (0xB8), and Warm Reset (0xBD) commands, once the command completes the HLC updates the *MMI_path_command_status* tag to indicate completion.

See Also

MMI_path_command_status on page 273

MMI_path_status on page 295

MMI_set_prop_power_state_cmd

Source/Destination

Host Controller ➔ HLC

Purpose

Used with the Propulsion Power option, contains one or more entries of type `udt_MMI_set_prop_power_state_cmd` to process state changes in a propulsion power supply. Each propulsion power supply can be affiliated with one or more QuickStick HT 5700 Motor Controllers in a transport system. Whenever a state change occurs in a propulsion power supply, the new state is propagated to its affiliated motor controllers.

Each command can be monitored via its corresponding `udt_MMI_propulsion_power_cmd_status` entry in the [MMI_propulsion_power_cmd_status](#) tag.

NOTE: This feature is only available on QSHT systems using QuickStick HT 5700 Motor Controller with Kinetix 2198-Pxxx DC-bus power supplies.

The high-level controller and the propulsion power supplies specified must be in the operational state. If any component required to execute the command is not in an operational state, the command is rejected.

The **Enable Propulsion Power** option must be selected (checked) on the EtherNet/IP Host Controller Settings page in the Configurator Utility when creating the Node Controller Configuration File (see the *MagneMotion System Configurator User Manual*, publication [MMI-UM046](#)).

Support

This message is supported in the latest software release for the following product lines when **Enable Propulsion Power** is specified the Node Controller Configuration File:

- QuickStick HT 5700 transport systems.

UDT Field Descriptions

The UDT described in [Table 4-23](#) shows the type of each element that comprises the MMI_set_prop_power_state_cmd tag. Each UDT field is described in more detail following the table.

Table 4-23: UDT Fields for udt_MMI_set_prop_power_state_cmd

Name	Data Type	Style	Range
supply_id	INT	Decimal	1...32
type	SINT	Decimal	1
state	SINT	Hex	0x00...0xFF
active_flag	SINT	Hex	0x00, 0x01
command_count	DINT	Hex	0x0...0xFFFFFFFF
Data Type Size:		12 bytes	

UDT Field Details

supply_id – The ID of the specified propulsion power supply undergoing a state change.

type – The type of propulsion power supply undergoing a state change.

Value	Description
1	Diode Front End (DFE) propulsion power supply

state – The state for the specified propulsion power supply (2198-Pxxx DC-bus Power Supply) used with QuickStick HT 5700 Motor Controller.

Table 4-24: QuickStick HT 5700 Propulsion Power Supply States

Value	State	Description
0x00	Unconnected	The drive is trying to establish communication with the EtherNet/IP controller.
0x01	Precharge	The drive is ready for mains input power.
0x02	Stopped	The drive has DC bus ready, but the control loops are not enabled.
0x03	Starting	The drive is enabled and checking various conditions before entering the RUNNING or TESTING state. For example, the drive checks the Brake Release delay time during the STARTING state.

Table 4-24: QuickStick HT 5700 Propulsion Power Supply States (Continued)

Value	State	Description
0x04	Running	<ul style="list-style-type: none"> The drive is enabled, configured with an active control mode, and actively tracking a command. The drive is configured for No Control and is fully operational.
0x05	Testing	The drive is actively executing a test procedure, for example, a hookup test.
0x06	Stopping	The drive is decelerating to a stop as the result of a disable.
0x07	Aborting	The drive is decelerating to a stop as the result of a fault or an abort request.
0x08	Faulted	The drive is faulted due to an existing or past fault condition.
0x09	Start Inhibited	The drive has an active condition that inhibits it from being enabled.
0x0A	Shutdown	The drive has been shut down.
0x0B	Axis Inhibited	The axis is inhibited. If this is the only instance that is supported by the CIP Motion connection, the connection is closed
0x0C	Not Grouped	A CIP Motion axis instance exists but is not associated with a Motion Group.
0x0D	No Device	A CIP Motion axis instance exists but is not associated with a CIP Motion device.
0x0E	Configuring	The drive is receiving configuration information from the controller.
0x0F	Synchronizing	The drive is waiting for a successful Group Sync service.
0x10	Waiting for Group	There are other axes in the Motion Group that are still being configured or synchronized.
0x11...0xFF	Reserved	Reserved

active_flag – This flag is used in configurations where a static table of udt_MMI_set_prop_power_state_cmd entries is used and the same table is sent every time.

Value	Description
0x00	The HLC skips this entry.
0x01	The HLC processes this entry.

command_count – Host controller derived unique counter, which is used to confirm that the HLC has received the command and to determine command status. This counter provides a convenient handshake mechanism to determine command status by matching the counter with the fields in [MMI_propulsion_power_cmd_status](#) tag.

Creating a DINT tag for use as a counter is suggested. The DINT tag can then be incremented and copied to command_count each time a new set propulsion power state command is issued so the new command can be tracked against the various count fields that are described in [MMI_propulsion_power_cmd_status](#) tag.

Message Configuration

The source element set_prop_power_state_cmd described in [Table 4-25](#) is an array of type udt_MMI_set_prop_power_state_cmd. It can be a single tag with the Number of Elements set to 1 or it can be an array with the Number of Elements set to the maximum of 32.

Table 4-25: Message Configuration for set_prop_power_state_cmd

Source Element Type	udt_MMI_set_prop_power_state_cmd
Number of Elements	Minimum array size: 1 Maximum array size: 32
Destination Element	MMI_set_prop_power_state_cmd

Response

After receiving the command, the HLC verifies the command parameters and updates [MMI_propulsion_power_cmd_status](#) with either a “Command Accepted” response (0x00) or a “Command Rejected” response as appropriate. If the command is accepted, the HLC handles propagating the new state to motor controllers affiliated with the specified propulsion power supply. If the command is rejected, the response includes a status code identifying the reason for rejection (see *HLC Status Codes on page 333*).

On completion of the command, the HLC updates [MMI_propulsion_power_cmd_status](#) with a command status response (see *HLC Status Codes on page 333*). Additionally, the HLC updates the corresponding propulsion power supply status entry in [MMI_propulsion_power_status](#) to reflect changes resulting from executing this command.

See Also

[MMI_propulsion_power_cmd_status on page 298](#)
[MMI_propulsion_power_status on page 301](#)

MMI_set_traffic_light_cmd

Source/Destination

Host Controller ➔ HLC

Purpose

Used with the Traffic Lights option, contains one or more entries of type `udt_MMI_set_traffic_light_cmd`, used to set the color of a traffic light to either green or red. Depending on color, a traffic light controls vehicle motion as follows:

- When set to green, vehicles are granted permission to move beyond the position of the traffic light.
- When set to red, vehicles are not granted permission to move into or beyond the motor block at the position of the traffic light. The obstructed status bit for the vehicle is set if the vehicle is stopped at a traffic light. The obstructed status bit is also set if the vehicle is waiting in a queue of vehicles or waiting at a switch.
- Vehicles that have permission to move beyond the motor block at the position of the traffic light do not stop if the traffic light is commanded to turn red.

Each command can be monitored via its corresponding `udt_MMI_traffic_light_cmd_status` entry in the [MMI_traffic_light_cmd_status](#) tag.

NOTE: The high-level controller, node controllers, and the path the traffic light resides on must be in the operational state.

These changes are not persistent. Any traffic lights set using this command must be reset in the following situations; whenever the node controller is restarted or rebooted, or whenever any paths where traffic lights are located are reset.

The **Enable Traffic Lights** option must be selected (checked) on the EtherNet/IP Host Controller Settings page in the Configurator Utility when creating the Node Controller Configuration File (see *MagneMotion System Configurator User Manual*, publication [MMI-UM046](#)).

Support

This message is supported in the latest software release for the following product lines when **Enable Traffic Lights** is specified the Node Controller Configuration File:

- MagneMover LITE transport systems.
- QuickStick transport systems.
- QuickStick HT transport systems.

UDT Field Descriptions

The UDT described in [Table 4-26](#) shows the type of each element that comprises the MMI_set_traffic_light_cmd tag. Each UDT field is described in more detail following the table.

Table 4-26: UDT Fields for udt_MMI_set_traffic_light_cmd

Name	Data Type	Style	Range
traffic_light_id	INT	Decimal	1...4096
color	SINT	Decimal	0, 1
command_index	SINT	Decimal	0...31
active_flag	SINT	Hex	0x00, 0x01
command_count	DINT	Hex	0x0...0xFFFFFFFF
Data Type Size:		12 bytes	

UDT Field Details

traffic_light_id – The ID of the specified traffic light to change. The ID must be a nonzero positive integer that references a traffic light that exists in the configuration.

color – The color to set for the specified traffic light.

Value	Description
0	Green – Allows traffic to pass.
1	Red – Stops traffic.

command_index – The index of the entry in the [MMI_traffic_light_cmd_status](#) tag to update when this MMI_set_traffic_light_cmd completes.

active_flag – This flag is used in configurations where a static table of udt_MMI_set_traffic_light_cmd entries is used and the same table is sent every time.

Value	Description
0x00	The HLC skips this entry.
0x01	The HLC processes this entry.

command_count – Host controller derived unique counter, which is used to confirm that the HLC has received the command and to determine command status. This counter provides a convenient handshake mechanism to determine command status by matching the counter with the fields in [MMI_traffic_light_cmd_status](#) tag.

Creating a DINT tag for use as a counter is suggested. The DINT tag can then be incremented and copied to `command_count` each time a new traffic light command is issued so the new command can be tracked against the various count fields that are described in [MMI_traffic_light_cmd_status](#) tag.

Message Configuration

The source element `set_traffic_light_cmd` described in [Table 4-27](#) is an array of type `udt_MMI_set_traffic_light_cmd`. It can be a single tag with the Number of Elements set to 1 or it can be an array with the Number of Elements set to the maximum of 32.

Table 4-27: Message Configuration for `set_traffic_light_cmd`

Source Element Type	<code>udt_MMI_set_traffic_light_cmd</code>
Number of Elements	Minimum array size: 1 Maximum array size: 32
Destination Element	<code>MMI_set_traffic_light_cmd</code>

Response

After receiving the command, the HLC verifies the command parameters and updates [MMI_traffic_light_cmd_status](#) with either a “Command Accepted” response (0x00) or a “Command Rejected” response as appropriate. If the command is accepted, the HLC handles setting the specified traffic light to the specified color. If the command is rejected, the response includes a status code identifying the reason for rejection (see [HLC Status Codes on page 333](#)).

On completion of the command, the HLC updates [MMI_traffic_light_cmd_status](#) with a command status response (see [HLC Status Codes on page 333](#)). On successful completion of the command, the HLC updates the corresponding traffic light status entry in [MMI_traffic_light_status](#) to reflect the new status of the traffic light.

See Also

[MMI_create_traffic_light_cmd on page 151](#)
[MMI_delete_traffic_light_cmd on page 154](#)
[MMI_traffic_light_cmd_status on page 315](#)
[MMI_traffic_light_status on page 318](#)

MMI_sm_poll_command

Source/Destination

Host Controller ➔ HLC

Purpose

Used with the System Monitoring option, contains one or more entries of type `udt_MMI_sm_poll_command` used to poll for metric data on the specified component.

Each poll command can be monitored via its corresponding `udt_MMI_sm_command_status` entry in the [MMI_sm_command_status](#) tag.

NOTE: The high-level controller, node controllers, and the components that are specified must be in the operational state.

The **Enable System Monitoring** option must be selected (checked) on the EtherNet/IP Host Controller Settings page in the Configurator Utility when creating the Node Controller Configuration File (see *MagneMotion System Configurator User Manual*, publication [MMI-UM046](#)).

Support

This message is supported in the latest software release for the following product lines when **Enable System Monitoring** is specified the Node Controller Configuration File:

- QuickStick transport systems.

UDT Field Descriptions

The UDT described in [Table 4-28](#) shows the type of each element that comprises the MMI_sm_poll_command tag. Each UDT field is described in more detail following the table.

Table 4-28: UDT Fields for udt_MMI_sm_poll_command

Name	Data Type	Style	Range
component_type	UDT	—	See Embedded udt_MMI_component_type for available component types.
metric_data_index	INT	Hex	0...65535
command_options	INT	Hex	0x0001
metric_id	INT	Hex	0x00...0xFFFF
metric_instance	INT	Decimal	0...65535
command_index	INT	Decimal	0...255
command_count	DINT	Hex	0x0...0xFFFFFFFF
active_flag	SINT	Hex	0x00, 0x01
Data Type Size:		28 bytes	

UDT Field Details

component_type – ID of the component targeted for system monitoring. See [Embedded udt_MMI_component_type](#) for descriptions of available component types.

metric_data_index – The index of the entry in [MMI_sm_metric_data](#) tag to update when new metric data responses are received from the targeted component.

command_options – Permits the selection of one or more processing options.

Option Bitmask	Command Option Description
0x0000	None
0x0001	Clear Metric Data on Sampling Start – Instructs the component to clear metric data at the beginning of each sampling interval. Ignored by “unsubscribe” subscription commands.

metric_id – The unique non-zero ID of a metric to report to the host controller, see [Table 4-29](#).

Table 4-29: System Monitoring, Hardware Metrics

Name	ID	Instance	Metric Syntax*	Units	Conversion	Min	Max	Description
Propulsion Soft Start Input Voltage [†]	0x1015	0	GaugeU16	Volts DC	$\frac{\text{Value} * 128}{65535}$	0V	+128V	The propulsion voltage present at the input connector after the input fuse.
Propulsion Soft Start Output Voltage [‡]	0x1016	0	GaugeU16	Volts DC	$\frac{\text{Value} * 128}{65535}$	0V	+128V	Propulsion Voltage at input of motor controller.
Raw Board Temperature [§]	0x1020	0	GaugeU16	Celsius	$\frac{\text{Value} * 120}{65535}$	0° C	120 °C	Raw hardware board temperature in degrees C. The value is represented as an unsigned 16-bit number as temperature ranging from 0...120 °C [32...248 °F]
External Board Temperature [†]	0x1021	0	GaugeU16	Celsius	$\frac{\text{Value} * 120}{65535}$	0° C	120 °C	The external hardware board temperature in degrees C. The value is represented as an unsigned 16-bit number as temperature ranging from 0...120 °C [32...248 °F]

* See [Table 4-30](#)

[†] Supported on QuickStick 100 motors.

[‡] Supported on QuickStick 100 and QuickStick 150 (QuickStick) motors.

[§] Supported on QuickStick 150 motors.

Table 4-30: System Monitoring Syntax Types

Syntax	Description
Integer16	Single signed 16-bit integer. Values range between -32768 and 32767. Returns only the last updated value when read.
Integer32	Single signed 32-bit integer with values range between -2147483648 and 2147483647. Returns only the last updated value when read.
Unsigned16	Single unsigned 16-bit integer with values ranging between 0 and 65535. Returns only the last updated value when read.
Unsigned32	Single unsigned 32-bit integer with values ranging between 0 and 4294967295. Returns only the last updated value when read.
Counter16	Single unsigned 16-bit integer that monotonically increases until it reaches its maximum value, when it wraps around and starts increasing again from zero. Counters only return the last updated value when read.
Counter32	Single unsigned 32-bit integer that monotonically increases until it reaches its maximum value, when it wraps around and starts increasing again from zero. Counters only return the last updated value when read.
Counter64	Single unsigned 64-bit integer that monotonically increases until it reaches its maximum value, when it wraps around and starts increasing again from zero. Counters only return the last updated value when read.
Gauge16	Four signed 16-bit values that can increase or decrease, which returns the following properties (in order): last value, minimum value, maximum value, and average value. When the internal software updates a Gauge metric, it updates all four properties to maintain statistical consistency. When this pattern is read, all four values are returned.
GaugeU16	Four unsigned 16-bit values that can increase or decrease, which returns the following properties (in order): last value, minimum value, maximum value, and average value. When the internal software updates a Gauge metric, it updates all four properties to maintain statistical consistency. When this pattern is read, all four values are returned.

metric_instance – Identifies a unique instances of a metric.

Value	Description
0	For metrics that have only one instance, this field must always be zero. To unsubscribe from all instances of a metric, this field must be zero.
1...65535	For metrics that have multiple unique instances, this field selects the specific metric instance to report to the host controller (see Table 4-29).

command_index – The index of the entry in the [MMI_sm_command_status](#) array to update when this poll command completes.

command_count – Host controller derived unique counter, which is used to confirm that the HLC has received the command and to determine command status. This counter provides a convenient handshake mechanism to determine command status by matching the counter with the fields in [MMI_sm_command_status](#) tag.

Creating a DINT tag for use as a counter is suggested. The DINT tag can then be incremented and copied to command_count each time a new poll command is issued so the new command can be tracked against the various count fields that are described in [MMI_sm_command_status](#) tag.

active_flag – This flag is used in configurations where a static table of udt_sm_poll_command entries is used and the same table is sent every time.

Value	Description
0x00	The HLC skips this entry.
0x01	The HLC processes this entry.

Message Configuration

The source element sm_poll_command described in [Table 4-31](#) is an array of type udt_MMI_sm_poll_command. It can be a single tag with the Number of Elements set to 1 or it can be an array with the Number of Elements set to the maximum of 256. When the source element is configured to use an array, the message can send multiple commands to the HLC with one message.

Table 4-31: Message Configuration for sm_poll_command

Source Element Type	udt_MMI_sm_poll_command
Number of Elements	Minimum array size: 1 Maximum array size: 256
Destination Element	MMI_sm_poll_command

Response

After receiving the command, the HLC verifies the command parameters and updates [MMI_sm_command_status](#) with a “Command Accepted” or a “Command Rejected” response as appropriate. If the command is accepted, the HLC updates entries in the [MMI_sm_metric_data](#) tag with new metric data. If the command is rejected, the response includes a rejection code identifying the reason for rejection (see *HLC Status Codes* [on page 333](#)).

On completion of the specified command, the HLC updates the [MMI_sm_command_status](#) tag with a Command Complete or a Command Failed response.

See Also

Embedded udt_MMI_component_type [on page 336](#)

MMI_sm_subscription_command [on page 197](#)

MMI_sm_command_status [on page 306](#)

MMI_sm_metric_data [on page 309](#)

MMI_sm_subscription_command

Source/Destination

Host Controller ➔ HLC

Purpose

Used with the System Monitoring option, contains one or more entries of type `udt_MMI_sm_subscription_command` used to register metric updates on the specified component type. Once registered, metric updates are reported continuously to the host controller at the specified interval until stopped with an “unsubscribe” subscription command.

Each subscription command can be monitored via its corresponding `udt_MMI_sm_command_status` entry in the [MMI_sm_command_status](#) tag.

NOTE: The high-level controller, node controllers, and the components that are specified must be in the operational state.

The **Enable System Monitoring** option must be selected (checked) on the EtherNet/IP Host Controller Settings page in the Configurator Utility when creating the Node Controller Configuration File (see *MagneMotion System Configurator User Manual*, publication [MMI-UM046](#)).

Support

This message is supported in the latest software release for the following product lines when **Enable System Monitoring** is specified the Node Controller Configuration File:

- QuickStick transport systems.

UDT Field Descriptions

The UDT described in [Table 4-32](#) shows the type of each element that comprises the `MMI_sm_subscription_command` tag. Each UDT field is described in more detail following the table.

Table 4-32: UDT Fields for `udt_MMI_sm_subscription_command`

Name	Data Type	Style	Range
<code>component_type</code>	UDT	—	See Embedded <code>udt_MMI_component_type</code> for available component types.
<code>metric_data_index</code>	INT	Decimal	0...65535
<code>command_options</code>	INT	Hex	0x0001
<code>subscription_interval</code>	INT	Decimal	0...65535
<code>metric_id</code>	INT	Hex	0x0...0xFFFF
<code>metric_instance</code>	INT	Decimal	0...65535
<code>command_index</code>	INT	Decimal	0...255
<code>command_count</code>	DINT	Hex	0x0...0xFFFFFFFF
<code>active_flag</code>	SINT	Hex	0x00, 0x01
Data Type Size:		28 bytes	

UDT Field Details

`component_type` – ID of the component targeted for system monitoring. See [Embedded `udt_MMI_component_type`](#) for descriptions of available component types.

`metric_data_index` – The index of the entry in [MMI_sm_metric_data](#) tag to update when new metric data responses are received from the targeted component.

`command_options` – Permits the selection of one or more processing options.

Option Bitmask	Command Option Description
0x0000	None
0x0001	Clear Metric Data on Sampling Start – Instructs the component to clear metric data at the beginning of each sampling interval. Ignored by “unsubscribe” subscription commands.

subscription_interval – Specifies the sampling interval (in seconds) that must elapse before a report of metric data for the specified metric is returned to the host controller. Specify zero to unsubscribe (that is, stop) the current subscription for the specified metric. When nonzero, it specifies a sampling interval for the specified metric from 1 second to over 18 hours (see [Table 4-33](#) for common system monitoring intervals).

Table 4-33: Common System Monitoring Intervals

Time Interval (seconds)	Time Duration
0	None – Unsubscribe from the specified metric.
1	1 second (minimum interval)
30	30 seconds
60	1 minute
300	5 minutes
900	15 minutes
1800	30 minutes
3600	1 hour
14400	4 hours
28800	8 hours
43200	12 hours
64800	18 hours
65535	18 hours, 12 minutes, 15 seconds (maximum interval)

metric_id – For “subscribe” commands, the unique non-zero ID of a metric to report to the host controller, see [Table 4-29](#).

For “unsubscribe” commands (**Subscription Interval** = 0), the ID of one or more metrics to stop reporting to the host controller. If the metric_id field is zero, all subscriptions for the targeted component are unsubscribed. If the metric_id field is non-zero, only the subscription for the specified metric is unsubscribed.

metric_instance – Identifies a unique instances of a metric.

Value	Description
0	For metrics that have only a single instance, this field must always be zero. To unsubscribe from all instances of a metric, this field must be zero.
1...65535	For metrics that have multiple unique instances, this field selects the specific metric instance to report to the host controller (see Table 4-29)

command_index – The index of the entry in the *MMI_sm_command_status* array to update when this subscription command completes.

command_count – Host controller derived unique counter, which is used to confirm that the HLC has received the command and to determine command status. This counter provides a convenient handshake mechanism to determine command status by matching the counter with the fields in *MMI_sm_command_status* tag.

Creating a DINT tag for use as a counter is suggested. The DINT tag can then be incremented and copied to **command_count** each time a new subscription command is issued so the new command can be tracked against the various count fields that are described in *MMI_sm_command_status* tag.

active_flag – This flag is used in configurations where a static table of *udt_MMI_sm_subscription_command* entries is used and the same table is sent every time.

Value	Description
0x00	The HLC skips this entry.
0x01	The HLC processes this entry.

Message Configuration

The source element *sm_subscription_command* described in [Table 4-34](#) is an array of type *udt_MMI_sm_subscription_command*. It can be a single tag with the Number of Elements set to 1 or it can be an array with the Number of Elements set to the maximum of 256. When the source element is configured to use an array, the message can send multiple commands to the HLC with one message.

Table 4-34: Message Configuration for sm_subscription_command

Source Element Type	<i>udt_MMI_sm_subscription_command</i>
Number of Elements	Minimum array size: 1 Maximum array size: 256
Destination Element	<i>MMI_sm_subscription_command</i>

Response

After receiving the command, the HLC verifies the command parameters and updates *MMI_sm_command_status* with a “Command Accepted” or a “Command Rejected” response as appropriate. If the command is rejected, the response includes a rejection code identifying the reason for rejection (see *HLC Status Codes on page 333*).

- If a “subscribe” command is accepted, the HLC handles registering metric data updates from the selected component. Additionally, the HLC updates entries in the *MMI_sm_metric_data* tag with new metric data responses as they are received from the selected component.
- If an “unsubscribe” command is accepted, the HLC terminates one or more metric updates from the selected component.

On completion of the specified command, the HLC updates the *MMI_sm_command_status* tag with a Command Complete or a Command Failed response.

See Also

Embedded udt_MMI_component_type on page 336
MMI_sm_poll_command on page 191
MMI_sm_command_status on page 306
MMI_sm_metric_data on page 309

MMI_terminus_node_command

Source/Destination

Host Controller ➔ HLC

Purpose

Sets and clears signals that are affiliated with a Terminus Node, which is used to move a vehicle on or off a path to or from customer-supplied equipment. See [MMI_node_command_status](#) for the current state of the signals. See *Terminus Node on page 105* for a description of handshaking using these signals.

Each Terminus Node command can be monitored via its corresponding udt_MMI_node_command_status entry in the [MMI_node_command_status](#) tag. The status of the Terminus Nodes can be monitored via the entries in the [MMI_node_status](#) tag.

NOTE: The high-level controller, node controllers, and the path where the Terminus Node is located must be in the operational state. If not, the command is rejected.

Support

This message is supported in the latest software release for the following product lines:

- MagneMover LITE transport systems.
- QuickStick transport systems.
- QuickStick HT transport systems.

UDT Field Descriptions

The UDT described in [Table 4-35](#) shows the type of each element that comprises the MMI_terminus_node_command tag. Each UDT field is described in more detail following the table.

Table 4-35: UDT Fields for udt_MMI_terminus_node_command

Name	Data Type	Style	Range
node_id	INT	Decimal	1...65535
signal_number	SINT	Decimal	0, 1
signal_level	SINT	Decimal	0, 1
command_count	DINT	Hex	0x0...0xFFFFFFFF
vehicle_id	INT	Decimal	0...65535
active_flag	SINT	Hex	0x00, 0x01
Data Type Size:		12 bytes	

UDT Field Details

node_id – The ID of the Terminus Node for the signal. The ID must be a nonzero positive integer that references a node that exists in the configuration.

signal_number – Specifies the signal to set.

Value	Signal
0	ENTRY_REQUESTED
1	EXIT_ALLOWED

signal_level – Specifies the level of the signal.

Value	Signal Level
0	Low (inactive)
1	High (active)

command_count – Host controller derived unique counter, which is used to confirm that the HLC has received the command and to determine command status. This counter provides a convenient handshake mechanism to determine command status by matching the counter with the fields in [MMI_node_command_status](#) tag.

Creating a DINT tag for use as a counter is suggested. The DINT tag can then be incremented and copied to command_count each time a new terminus node command is issued so the new command can be tracked against the various count fields that are described in [MMI_node_command_status](#) tag.

vehicle_id – Only valid for a request for a vehicle to enter, this field is ignored for a request to exit. This value is the ID to use for the vehicle that is entering (when setting ENTRY_REQUESTED High).

- To specify a vehicle ID, the ID must be a nonzero positive integer that is not currently used by another vehicle. This ID must be within the limits that are defined by the **Max Vehicle ID setting** on the EtherNet/IP Host Controller Settings page in the Configurator Utility when creating the Node Controller Configuration File (see *MagneMotion System Configurator User Manual*, publication [MMI-UM046](#)).
- To have the HLC assign the vehicle ID, enter zero. The [MMI_node_status](#) [node_id] entry contains the vehicle ID assigned by the transport system when the host controller specifies a vehicle_id of “0” in an entry request.

NOTE: Rockwell Automation recommends not specifying a vehicle ID and allowing the system to assign it (vehicle_id = 0).

active_flag – This flag is used in configurations where a static table of udt_MMI_terminus_node_command entries is used and the same table is sent every time.

Value	Description
0x00	The HLC skips this entry.
0x01	The HLC processes this entry.

Message Configuration

The source element terminus_node_command described in [Table 4-36](#) is an array of type udt_MMI_terminus_node_command. It can be a single tag with the Number of Elements set to 1 or it can be an array with the Number of Elements set to the number of nodes in the transport system. When the source element is configured to use an array, the message can send multiple commands to the HLC with one message.

Table 4-36: Message Configuration for terminus_node_command

Source Element Type	udt_MMI_terminus_node_command
Number of Elements	Minimum array size: 1 Maximum array size: 256
Destination Element	MMI_terminus_node_command

Response

After receiving the command, the HLC verifies the command parameters and updates [MMI_node_command_status](#) with either a “Command Accepted” response (0x00) or a “Command Rejected” response as appropriate. If the command is accepted, it is forwarded to the appropriate node controller for execution. If the command is rejected, the response includes a rejection code identifying the reason for rejection (see *HLC Status Codes* [on page 333](#)).

After the command is executed, the status of the node may change. Any change is shown in the [MMI_node_status](#) tag.

See Also

[MMI_node_status on page 269](#)

[MMI_node_command_status on page 261](#)

MMI_vehicle_command

Source/Destination

Host Controller ➔ HLC

Purpose

Used with the Vehicle Commands option, contains one or more entries of type udt_MMI_vehicle_command used to specify vehicle commands. Each command is monitored via its corresponding udt_MMI_vehicle_command_status entry in the [MMI_vehicle_command_status](#) tag.

NOTE: The high-level controller, node controllers, and the path the vehicle resides on must be in the operational state.

The **Enable Vehicle Commands** option must be selected (checked) on the EtherNet/IP Host Controller Settings page in the Configurator Utility when creating the Node Controller Configuration File (see *MagneMotion System Configurator User Manual*, publication [MMI-UM046](#)).

Support

This message is supported in the latest software release for the following product lines when **Enable Vehicle Commands** is specified the Node Controller Configuration File:

- MagneMover LITE transport systems.
- QuickStick transport systems.
- QuickStick HT transport systems.

UDT Field Descriptions

The UDT described in [Table 4-37](#) shows the type of each element that comprises the MMI_vehicle_command tag. Each UDT field is described in more detail following the table.

Table 4-37: UDT Fields for udt_MMI_vehicle_command

Name	Data Type	Style	Range
vehicle_id	INT	Decimal	1...65535
subcommand	SINT	Hex	0x00
active_flag	SINT	Hex	0x00, 0x01
command_count	DINT	Hex	0x0...0xFFFFFFFF
Data Type Size:		8 bytes	

UDT Field Details

vehicle_id – The ID of the vehicle targeted by this command. The ID must be a nonzero positive integer that references a vehicle that exists in the transport system.

Subcommand – Fixed value that identifies the vehicle subcommand to execute.

Value	Description
0x00	Clear vehicle suspect bit – Commands the motor controller responsible for the specified vehicle to clear the suspect bit in the vehicle record for the vehicle.

active_flag – This flag is used in configurations where a static table of udt_MMI_vehicle_command entries is used and the same table is sent every time.

Value	Description
0x00	The HLC skips this entry.
0x01	The HLC processes this entry.

command_count – Host controller derived unique counter, which is used to confirm that the HLC has received the command and to determine command status. This counter provides a convenient handshake mechanism to determine command status by matching the counter with the fields in [MMI_vehicle_command_status](#) tag.

Creating a DINT tag for use as a counter is suggested. The DINT tag can then be incremented and copied to command_count each time a new vehicle command is issued so the new command can be tracked against the various count fields that are described in [MMI_vehicle_command_status](#) tag.

Message Configuration

The source element vehicle_command described in [Table 4-38](#) is an array of type udt_MMI_vehicle_command. It can be a single tag with the Number of Elements set to 1 or it can be an array with the Number of Elements set to the maximum of 256.

Table 4-38: Message Configuration for vehicle_command

Source Element Type	udt_MMI_vehicle_command
Number of Elements	Minimum array size: 1 Maximum array size: 256
Destination Element	MMI_vehicle_command

Response

After receiving the command, the HLC verifies the command parameters and updates *MMI_vehicle_command_status* with either a “Command Accepted” response (0x00) or a “Command Rejected” response as appropriate. If the command is accepted, it is forwarded to the motor responsible for the vehicle for execution. If the command is rejected, the response includes a rejection code identifying the reason for rejection (see *HLC Status Codes on page 333*).

On completion of the command, the HLC updates *MMI_vehicle_command_status* to indicate completion. After the command is executed, the status of the vehicle may change. Any change is shown in the *MMI_extended_vehicle_status* or the *MMI_vehicle_status* tag.

See Also

MMI_vehicle_command_status on page 321

MMI_vehicle_delete_order

Source/Destination

Host Controller ➔ HLC

Purpose

This message is used to command a node controller and all related motors to delete any vehicle records with a matching ID. The node controller deletes the record so the vehicle can't own any nodes and the motors delete their records if the vehicle has permissions for blocks on that motor.

This message must be used carefully. It is typically used only when a vehicle is physically removed from the guideway because it is inoperable, or when a vehicle entry is canceled. If a vehicle entry or exit is active for the vehicle, the entry/exit state machine is also cleared.

Each Vehicle Delete command can be monitored via its corresponding `udt_MMI_vehicle_order_status` entry in the *MMI_vehicle_order_status* tag. The status of the vehicle can be monitored via its corresponding entry in the *MMI_extended_vehicle_status* or *MMI_vehicle_status* tags.

NOTE: The high-level controller, node controllers, and the path the vehicle resides on in the transport system must be in the operational state. If any component required to fill the command is not in an operational state, the command is rejected.

Once the command is issued, the area on the motor where the vehicle is located is defined as empty. Caution must be taken, as a collision can occur if another vehicle attempts to move into that area before the vehicle is physically removed. Therefore, when using the Delete Vehicle command, the vehicle must be removed before issuing the command.

Rockwell Automation recommends using a Terminus Node to add or remove vehicles on the transport system.

Support

This message is supported in the latest software release for the following product lines:

- MagneMover LITE transport systems.
- QuickStick transport systems.
- QuickStick HT transport systems.

UDT Field Descriptions

The UDT described in [Table 4-39](#) shows the type of each element that comprises the MMI_vehicle_delete_order tag. Each UDT field is described in more detail following the table.

Table 4-39: UDT Fields for udt_MMI_vehicle_delete_order

Name	Data Type	Style	Range
vehicle_id	INT	Decimal	1...65535
order_number	DINT	Decimal	0...65535
active_flag	SINT	Hex	0x00, 0x01
Data Type Size:		12 bytes	

UDT Field Details

vehicle_id – The ID of the vehicle to delete. The ID must be a nonzero positive integer that references a vehicle that exists in the transport system.

order_number – Host controller derived unique order number, which is used to confirm that the HLC has received the command and to determine command status. This counter provides a convenient handshake mechanism to determine order status by matching the counter with the fields in the *MMI_vehicle_order_status* tag.

Rockwell Automation suggests creating a DINT tag for use as a counter. The DINT tag can then be incremented and copied to order_number each time a new vehicle command is issued so the new command can be tracked against the various count fields that are described in *MMI_vehicle_order_status*.

active_flag – This flag is used in configurations where a static table of udt_MMI_vehicle_delete_order entries is used and the same table is sent every time.

Value	Description
0x00	The HLC skips this entry.
0x01	The HLC processes this entry.

Message Configuration

The source element `vehicle_delete_order` described in [Table 4-40](#) is an array of type `udt_MMI_vehicle_delete_order`. It can be a single tag with the Number of Elements set to 1 or it can be an array with the Number of Elements set to the maximum of 256.

Table 4-40: Message Configuration for `vehicle_delete_order`

Source Element Type	<code>udt_MMI_vehicle_delete_order</code>
Number of Elements	Minimum array size: 1 Maximum array size: 256
Destination Element	<code>MMI_vehicle_delete_order</code>

Response

After receiving the command, the HLC verifies the command parameters and updates `MMI_vehicle_order_status` with either a “Command Accepted” response (0x00) or a “Command Rejected” response as appropriate. If the command is accepted, it is forwarded to the appropriate node controller for execution. If the command is rejected, the response includes a rejection code identifying the reason for rejection (see *HLC Status Codes* [on page 333](#)).

On completion of the command, the HLC updates the `MMI_vehicle_order_status` tag with a “Command Status – Command completed successfully” response (0x80).

See Also

`MMI_vehicle_order_status` [on page 324](#)

MMI_vehicle_follow_order

Source/Destination

Host Controller ➔ HLC

Purpose

Orders a vehicle to catch up to and follow another vehicle using a specified profile and then maintain a specified following distance. Once the vehicle is positioned at the commanded following distance, the system maintains the following distance indefinitely, matching the movement profile of the followed vehicle. The vehicle continues to follow the specified vehicle at the specified distance until the follow order is overridden with an `MMI_vehicle_position_order`, an `MMI_vehicle_station_order`, or a new `MMI_vehicle_follow_order`.

There is an auto decouple feature, where a decouple path ID and position are defined. When the platoon with the following vehicle approaches that position, the vehicle with that position specified as part of its follow order decouples from the platoon, slows down, and stops at that position.

Each Vehicle Follow order can be monitored via the corresponding `udt_MMI_vehicle_order_status` entry in the `MMI_vehicle_order_status` tag. The status of the vehicle can be monitored via its corresponding entry in the [*MMI_extended_vehicle_status*](#) or [*MMI_vehicle_status*](#) tags.

NOTE: The high-level controller, node controllers, and the path the vehicle resides on must be in the operational state. If the path where the vehicle is located is not in an operational state, the command is rejected. If any component required to execute the command is not in operational state, the command does not complete.

Vehicles must be stopped to couple them into a platoon.

Platoons must be stopped to decouple vehicles unless the command coupling a vehicle to the platoon specified a decouple destination for that vehicle.

If the acceleration or velocity values in a motion command are higher than the limit that is defined in the Node Controller Configuration File, the command is rejected.

If the commanded velocity value is higher than the value for a specific motor, the motor overrides that value while the vehicle is on that motor.

If a motion command is issued to a vehicle already in motion and the command has a lower acceleration than the previous command to that vehicle, the command is rejected.

Support

This message is supported in the latest software release for the following product lines:

- MagneMover LITE transport systems.

UDT Field Descriptions

The UDT described in [Table 4-41](#) shows the type of each element that comprises the MMI_vehicle_follow_order tag. Each UDT field is described in more detail following the table.

Table 4-41: UDT Fields for udt_MMI_vehicle_follow_order

Field Name	Data Type	Style	Range
vehicle_id	INT	Decimal	1...65535
follow_distance	REAL	Float	0.0...+41.0 (m, floating point)
followed_vehicle_id	INT	Decimal	1...65535
pid_set_index	SINT	Decimal	0...15
catchup_or_decouple_acceleration	REAL	Float	0.0...100.0 (m/s ² , floating point)
catchup_or_decouple_velocity	REAL	Float	0.0...100.0 (m/s, floating point)
decouple_destination_path_id (MM LITE only):	INT	Decimal	0...65535
decouple_destination_position (MM LITE only):	REAL	Float	0.0...+41.0 (m, floating point)
order_number	DINT	Hex	0x0...0xFFFFFFFF
direction	SINT	Decimal	1, 2
active_flag	SINT	Hex	0x00, 0x01
Data Type Size:		36 bytes	

UDT Field Details

vehicle_id – The ID number of the vehicle executing this vehicle follow order. This vehicle follows the vehicle that **followed_vehicle_id** specifies. The ID must be a nonzero positive integer that references a vehicle that exists in the transport system.

follow_distance – The distance (in meters) center-to-center that this vehicle maintains behind the vehicle being followed (specified by **followed_vehicle_id**) in the **direction** that is specified, once the pair of vehicles are coupled into a platoon.

NOTE: When coupling to a platoon, the vehicle must be within 30 mm of this distance, and at a complete stop before sending the vehicle follow order.

followed_vehicle_id – The ID number of the vehicle being followed by this vehicle (executing this vehicle follow order). The followed_vehicle_id must be a nonzero positive integer.

pid_set_index – The PID Set to use when executing the command.

catchup_or_decouple_acceleration – The maximum acceleration/deceleration for the vehicle (in m/s^2) to use when catching up to the vehicle to follow or to use when auto-decoupling from the vehicle that it is following. This value is checked against the system limit that is defined in the Node Controller Configuration File, and if higher, the command is updated to the value from the Node Controller Configuration File. Only available in MagneMover LITE systems.

This value is system-dependent, with the system maximums shown in MagneMotion Node Controller Interface User Manual, publication [MMI-UM001](#).

NOTE: Currently only used for decoupling while the platoon is moving as vehicles must not be moving when creating a platoon, but it must be defined.

catchup_or_decouple_velocity – The maximum velocity for the vehicle (in m/s) to use when catching up to the vehicle to follow or to use when decoupling from the vehicle that it is following. This value is checked against the system limit that is defined in the Node Controller Configuration File, and if higher, the command is updated to the value from the Node Controller Configuration File. Only available in MagneMover LITE systems.

This value is system-dependent, with the system maximums shown in MagneMotion Node Controller Interface User Manual, publication [MMI-UM001](#).

NOTE: Currently only used for decoupling while the platoon is moving as vehicles must not be moving when creating a platoon, but it must be defined.

decouple_destination_path_id – The ID number of the path where the decoupling position is located (where the vehicle stops after decoupling). Destination position is defined as the location on a path, from the beginning of the path to the midpoint of the vehicle. The ID must be a nonzero positive integer that references a path that exists in the configuration. If the value is zero, the decoupling feature is disabled. Only available in MagneMover LITE systems.

decouple_destination_position – The location of the decouple location in meters relative to the start of the specified path (where the vehicle stops after decoupling). Zero position is defined as the midpoint of the vehicle at the beginning of the path. This value is only used when **decouple_destination_path_id** is not zero. Only available in MagneMover LITE systems.

order_number – Host controller derived unique order number, which is used to confirm that the HLC has received the command and to determine command status. This counter provides a convenient handshake mechanism to determine order status by matching the counter with the fields in the *MMI_vehicle_order_status* tag.

Creating a DINT tag for use as a counter is suggested. The DINT tag can then be incremented and copied to *command_count* each time a new vehicle command is issued so the new command can be tracked against the various count fields that are described in the *MMI_vehicle_order_status* tag.

direction – Direction on the track relative to this vehicle where the **followed_vehicle_id** is located:

Value	Description
1	Vehicle to follow is downstream of this vehicle (motion is downstream).
2	Vehicle to follow is upstream of this vehicle (motion is upstream).

active_flag – This flag is used in configurations where a static table of `udt_MMI_vehicle_follow_order` entries is used, and the same table is sent every time.

Value	Description
0x00	The HLC skips this entry.
0x01	The HLC processes this entry.

Message Configuration

The source element `vehicle_follow_order` described in [Table 4-42](#) is an array of type `udt_MMI_vehicle_follow_order`. It can be a single tag with the Number of Elements set to 1 or it can be an array with the Number of Elements set to the maximum of 256.

Table 4-42: Message Configuration for vehicle_follow_order

Source Element Type	<code>udt_MMI_vehicle_follow_order</code>
Number of Elements	Minimum array size: 1 Maximum array size: 256
Destination Element	<code>MMI_vehicle_follow_order</code>

Response

After receiving the command, the HLC verifies the command parameters and updates the `MMI_vehicle_order_status` tag with either a “Command Accepted” response (0x00) or a “Command Rejected” response as appropriate (see *HLC Status Codes* [on page 333](#)). If the command is accepted, it is forwarded to the appropriate node controller for execution. If the command is rejected, the `MMI_vehicle_order_status` tag includes a rejection code that shows the reason for rejection.

- **Vehicle Following** – If the auto-decouple feature is not configured in the vehicle follow order (Destination Path ID = 0), when the command completes the following events occur:
 - A. When the vehicle is within the following distance from the specified vehicle, the **Caught Up** bit is set in the `MMI_extended_vehicle_status` Vehicle Flags.

- B. The HLC updates the *MMI_vehicle_order_status* tag with a “Command Status -Vehicle has caught up” response (0x81).

NOTE: If the vehicle never catches up to the specified vehicle, this command never completes.

- **Vehicle Auto-decouple** – If the auto-decouple feature is configured in the vehicle follow order (Destination Path ID > 0), when the vehicle begins to brake for arrival at the destination path and position that is supplied in the follow order the following events occur:
 - A. The HLC updates the *MMI_vehicle_order_status* tag with a “Vehicle Decoupling” response (0x82).
 - B. The **Vehicle Decoupled** bit is set in the *MMI_extended_vehicle_status* Vehicle Flags.
 - C. The order type for the command the vehicle is under changes from a Vehicle Follow Order (0xB7) to a Vehicle Position Order (0xB1).
 - D. The HLC updates the *MMI_vehicle_order_status* tag with a “Command Complete” response (0x80) for the Vehicle Position Order when the vehicle arrives at the defined decouple position. If the vehicle has followers during auto decoupling, it becomes the head of a new platoon for these followers.
 - E. The **Vehicle Decoupled** bit is cleared in the *MMI_extended_vehicle_status* Vehicle Flags.

See Also

MMI_vehicle_position_order [on page 216](#)
MMI_extended_vehicle_status [on page 234](#)
MMI_vehicle_order_status [on page 324](#)
MMI_vehicle_status [on page 328](#)

MMI_vehicle_position_order

Source/Destination

Host Controller ➔ HLC

Purpose

Moves the specified vehicle to a specified position, relative to the start of a path.

Each vehicle position order can be monitored via its corresponding `udt_MMI_vehicle_order_status` entry in the *MMI_vehicle_order_status* tag. The status of the vehicle can be monitored via its corresponding entry in the *MMI_vehicle_status* or *MMI_extended_vehicle_status* tags.

NOTE: The high-level controller, node controllers, and the path the vehicle resides on must be in the operational state. If the path where the vehicle is located is not in the operational state, the command is rejected. If any component required to execute the command is not in an operational state, the command does not complete.

If the commanded acceleration or velocity values are higher than the limit that is defined in the Node Controller Configuration File, the command is rejected.

If the commanded velocity value is higher than the value for a specific motor, the motor velocity value overrides the commanded value while the vehicle is on that motor.

If a motion command is issued to a vehicle already in motion and the command has a lower acceleration than the previous command to that vehicle, the command is rejected.

Support

This message is supported in the latest software release for the following product lines:

- MagneMover LITE transport systems.
- QuickStick transport systems.
- QuickStick HT transport systems.

UDT Field Descriptions

The UDT described in [Table 4-43](#) shows the type of each element that comprises the MMI_vehicle_position_order tag. Each UDT field is described in more detail following the table.

Table 4-43: UDT Fields for udt_MMI_vehicle_position_order

Name	Data Type	Style	Range
vehicle_id	INT	Decimal	1...65535
path_id	INT	Decimal	1...65535
position	REAL	Float	-41.0...+41.0 (m, floating point)
velocity_limit	REAL	Float	0.0...100.0 (m/s, floating point)
acceleration_limit	REAL	Float	0.0...100.0 (m/s ² , floating point)
order_number	DINT	Decimal	0...65535
flags_and_direction	SINT	Binary	Bits 0...3 are direction values Bits 4...7 are the PID set index
active_flag	SINT	Hex	0x00, 0x01
Data Type Size:		24 bytes	

UDT Field Details

vehicle_id – The ID of the vehicle to move. The ID must be a nonzero positive integer that references a vehicle that exists in the transport system.

path_id – The ID of the path where the desired position is located. The ID must be a nonzero positive integer that references a path that exists in the configuration.

position – The destination (in meters) of the vehicle, relative to the start of the specified path (expressed as a 32-bit single-precision floating point number). Zero position is defined as the midpoint of the vehicle at the beginning of the path. When decoupling a vehicle from a stationary platoon, the position that is specified must be the current target for the vehicle as returned through the Extended Vehicle Status. Negative positions can only be used with node types that support movement to a position before the beginning of a path.

velocity_limit – A positive number (in m/s) that defines the maximum velocity for vehicle motion during the move command (expressed as a 32-bit single-precision floating point number). This value is checked against the limit that is defined in the Node Controller Configuration File, and if higher the command is rejected. If this value is within the limit that is defined in the Node Controller Configuration File but higher than the value for any specific motor in the vehicle's route, the value from the motor overrides the command value while the vehicle is on that motor.

This value is system-dependent, with the system maximums shown in MagneMotion Node Controller Interface User Manual, publication [MMI-UM001](#).

acceleration_limit – A positive number (in m/s^2) that defines the maximum acceleration and deceleration rate for the move command (expressed as a 32-bit single-precision floating point number). This value is checked against the limit that is defined in the Node Controller Configuration File, and if higher the command is rejected. If this value is within the limit that is defined in the Node Controller Configuration File but higher than the value for any specific motor in the vehicle's route, the value from the motor is used and the command is updated to that lower value.

This value is system-dependent, with the system maximums shown in MagneMotion Node Controller Interface User Manual, publication [MMI-UM001](#).

order_number – Host controller derived unique order number, which is used to confirm that the HLC has received the command and to determine command status. This counter provides a convenient handshake mechanism to determine order status by matching the counter with the fields in the *MMI_vehicle_order_status* tag.

Rockwell Automation suggests creating a DINT tag for use as a counter. The DINT tag can then be incremented and copied to order_number each time a new vehicle command is issued so the new command can be tracked against the various count fields that are described in the *MMI_vehicle_order_status* tag.

flags_and_direction – Bits 0...3 indicate the direction to move the vehicle. Bits 4...7 indicate the PID Set to use when executing the move command.

Bit	Description
0...3	<p>Direction:</p> <p>0–Bidirectional – If the destination is on the same path the vehicle resides on, the vehicle can move either direction as required to get to the destination in the shortest distance. If the destination is on a path other than the path where the vehicle is located, the forward direction takes precedence for a transport system that is a closed-loop.</p> <p>1–Forward – Force the vehicle to move forward (downstream) only, useful to implement a unidirectional loop. If the destination is not reachable, the command fails and the vehicle is not moved.</p> <p>2–Backward – Force the vehicle to move backward (upstream) only, useful to implement a unidirectional loop in the backwards direction. If the destination is not reachable, the command fails and the vehicle is not moved.</p> <p>3...15 – Reserved</p>

Bit	Description
4...7	<p>PID Loop Set:</p> <p>0—Use user-defined PID Set 0 – Unloaded PID values.</p> <p>1—Use user-defined PID Set 1 – Loaded PID values.</p> <p>2—Use user-defined PID Set 2.</p> <p>3—Use user-defined PID Set 3.</p> <p>4—Use user-defined PID Set 4.</p> <p>5—Use user-defined PID Set 5.</p> <p>6—Use user-defined PID Set 6.</p> <p>7—Use user-defined PID Set 7.</p> <p>8—Use user-defined PID Set 8.</p> <p>9—Use user-defined PID Set 9.</p> <p>10—Use user-defined PID Set 10.</p> <p>11—Use user-defined PID Set 11.</p> <p>12—Use user-defined PID Set 12.</p> <p>13—Use user-defined PID Set 13.</p> <p>14—Use user-defined PID Set 14.</p> <p>15—Use user-defined PID Set 15 – Startup PID values. This PID set is automatically used during startup. If it is not defined, PID set 0 is scaled by 25% and used for startup.</p>

active_flag – This flag is used in configurations where a static table of udt_MMI_vehicle_position_order entries is used and the same table is sent every time.

Value	Description
0x00	The HLC skips this entry.
0x01	The HLC processes this entry.

Message Configuration

The source element `vehicle_position_order` described in [Table 4-44](#) is an array of type `udt_MMI_vehicle_position_order`. It can be a single tag with the Number of Elements set to 1 or it can be an array with the Number of Elements set to the number of vehicles in a transport system. When the source element is configured to use an array, the message can send multiple commands to the HLC with one message.

Table 4-44: Message Configuration for `vehicle_position_order`

Source Element Type	<code>udt_MMI_vehicle_position_order</code>
Number of Elements	Minimum array size: 1 Maximum array size: 256
Destination Element	<code>MMI_vehicle_position_order</code>

Response

After receiving the command, the HLC verifies the command parameters and updates `MMI_vehicle_order_status` with either a “Command Accepted” response (0x00) or a “Command Rejected” response as appropriate. If the command is accepted, it is forwarded to the appropriate node controller for execution. If the command is rejected, the response includes a rejection code identifying the reason for rejection (see *HLC Status Codes* [on page 333](#)).

On completion of the command, the HLC updates the `MMI_vehicle_order_status` tag with a “Command Status – Command completed successfully” response (0x80).

NOTE: If the vehicle never reaches the specified destination, this command never completes.

See Also

`MMI_extended_vehicle_status` [on page 234](#)

`MMI_vehicle_order_status` [on page 324](#)

`MMI_vehicle_status` [on page 328](#)

MMI_vehicle_station_order

Source/Destination

Host Controller ➔ HLC

Purpose

The specified path must be defined in the Node Controller Configuration File. The station is either defined statically in the Node Controller Configuration File or defined dynamically.

Each Vehicle Station Order can be monitored via its corresponding `udt_MMI_vehicle_order_status` entry in the `MMI_vehicle_order_status` tag or via its corresponding index entry in the [MMI_station_arrivals](#) tag. The status of the vehicle can be monitored via its corresponding entry in the [MMI_extended_vehicle_status](#) or [MMI_vehicle_status](#) tags.

If the transport system has fewer stations than vehicles or vehicles are tracked strictly by station arrivals and departures, it is possible to handshake Vehicle Station Orders with the `MMI_station_arrivals` tag that is updated by the HLC when vehicles arrive and depart stations, see [MMI_station_arrivals on page 313](#) for more details.

NOTE: The high-level controller, node controllers, and the path the vehicle resides on must be in the operational state. If the path where the vehicle is located is not in the operational state, the command is rejected. If any component required to execute the command is not in an operational state, the command does not complete.

If the commanded acceleration or velocity values are higher than the limit that is defined in the Node Controller Configuration File, the command is rejected.

If the commanded velocity value is higher than the value for a specific motor, the motor velocity value overrides the commanded value while the vehicle is on that motor.

If a motion command is issued to a vehicle already in motion and the command has a lower acceleration than the previous command to that vehicle, the command is rejected.

Support

This message is supported in the latest software release for the following product lines:

- MagneMover LITE transport systems.
- QuickStick transport systems.
- QuickStick HT transport systems.

UDT Field Descriptions

The UDT described in [Table 4-45](#) shows the type of each element that comprises the MMI_vehicle_station_order tag. Each UDT field is described in more detail following the table

Table 4-45: UDT Fields for udt_MMI_vehicle_station_order

Name	Data Type	Style	Range
vehicle_id	INT	Decimal	1...65535
station_id	INT	Decimal	1...2048
velocity_limit	REAL	Float	0.0...100.0 (m/s, floating point)
acceleration_limit	REAL	Float	0.0...100.0 (m/s ² , floating point)
order_number	DINT	Decimal	0...65535
flags_and_direction	SINT	Binary	Bits 0...3 are direction values Bits 4...7 are the PID set index
active_flag	SINT	Hex	0x00, 0x01
Data Type Size:		20 bytes	

UDT Field Details

vehicle_id – The ID of the vehicle to move. The ID must be a nonzero positive integer that references a vehicle that exists in the transport system.

This value is system-dependent, see MagneMotion Node Controller Interface User Manual, publication [MMI-UM001](#).

station_id – The ID of the station for the vehicle destination. The ID must be a nonzero positive integer from 1 to 2048 that references a station that is defined in the Node Controller Configuration File or defined dynamically.

This value is system-dependent, see MagneMotion Node Controller Interface User Manual, publication [MMI-UM001](#).

velocity_limit – A positive number (in m/s) that defines the maximum velocity for vehicle motion during the move command (expressed as a 32-bit single-precision floating point number). This value is checked against the limit that is defined in the Node Controller Configuration File, and if higher the command is rejected. If this value is within the limit that is defined in the Node Controller Configuration File but higher than the value for any specific motor in the vehicle's route, the value from the motor overrides the command value while the vehicle is on that motor.

This value is system-dependent, with the system maximums shown in MagneMotion Node Controller Interface User Manual, publication [MMI-UM001](#).

acceleration_limit – A positive number (in m/s^2) that defines the maximum acceleration and deceleration rate for the move command (expressed as a 32-bit single-precision floating point number). This value is checked against the limit that is defined in the Node Controller Configuration File, and if higher the command is rejected. If this value is within the limit that is defined in the Node Controller Configuration File but higher than the value for any specific motor in the vehicle's route, the value from the motor is used and the command is updated to that lower value.

This value is system-dependent, with the system maximums shown in MagneMotion Node Controller Interface User Manual, publication [MMI-UM001](#).

order_number – Host controller derived unique order number, which is used to confirm that the HLC has received the command and to determine command status. This counter provides a convenient handshake mechanism to determine order status by matching the counter with the fields in the *MMI_vehicle_order_status* tag.

Rockwell Automation suggests creating a DINT tag for use as a counter. The DINT tag can then be incremented and copied to order_number each time a new vehicle command is issued so the new command can be tracked against the various count fields that are described in the *MMI_vehicle_order_status* tag.

flags_and_direction – Bits 0...3 indicate the direction to move the vehicle. Bits 4...7 indicate the PID Set to use when executing the move command.

Bit	Description
0...3	<p>Direction:</p> <p>0–Bidirectional – If the destination is on the same path the vehicle resides on, the vehicle can move either direction as required to get to the destination in the shortest distance. If the destination is on a path other than the path where the vehicle is located, the forward direction takes precedence for a transport system that is a closed-loop.</p> <p>1–Forward – Force the vehicle to move forward (downstream) only, useful to implement a unidirectional loop. If the destination is not reachable, the command fails and the vehicle is not moved.</p> <p>2–Backward – Force the vehicle to move backward (upstream) only, useful to implement a unidirectional loop in the backwards direction. If the destination is not reachable, the command fails and the vehicle is not moved.</p> <p>3...15 – Reserved</p>

Bit	Description
4...7	<p>PID Loop Set:</p> <p>0—Use user-defined PID Set 0 – Unloaded PID values.</p> <p>1—Use user-defined PID Set 1 – Loaded PID values.</p> <p>2—Use user-defined PID Set 2.</p> <p>3—Use user-defined PID Set 3.</p> <p>4—Use user-defined PID Set 4.</p> <p>5—Use user-defined PID Set 5.</p> <p>6—Use user-defined PID Set 6.</p> <p>7—Use user-defined PID Set 7.</p> <p>8—Use user-defined PID Set 8.</p> <p>9—Use user-defined PID Set 9.</p> <p>10—Use user-defined PID Set 10.</p> <p>11—Use user-defined PID Set 11.</p> <p>12—Use user-defined PID Set 12.</p> <p>13—Use user-defined PID Set 13.</p> <p>14—Use user-defined PID Set 14.</p> <p>15—Use user-defined PID Set 15 – Startup PID values. This PID set is automatically used during startup. If it is not defined, PID set 0 is scaled by 25% and used for startup.</p>

active_flag – This flag is used in configurations where a static table of udt_MMI_vehicle_station_order entries is used and the same table is sent every time.

Value	Description
0x00	The HLC skips this entry.
0x01	The HLC processes this entry.

Message Configuration

The source element `vehicle_station_order` described in [Table 4-46](#) is an array of type `udt_MMI_vehicle_station_order`. It can be a single tag with the Number of Elements set to 1 or it can be an array with the Number of Elements set to the number of vehicles in a transport system. When the source element is configured to use an array, the message can send multiple commands to the HLC with one message.

Table 4-46: Message Configuration for `vehicle_station_order`

Source Element Type	<code>udt_MMI_vehicle_station_order</code>
Number of Elements	Minimum array size: 1 Maximum array size: 256
Destination Element	<code>MMI_vehicle_station_order</code>

Response

After receiving the command, the HLC verifies the command parameters and updates `MMI_vehicle_order_status` with either a “Command Accepted” response (0x00) or a “Command Rejected” response as appropriate. If the command is accepted, it is forwarded to the appropriate node controller for execution. If the command is rejected, the response includes a rejection code identifying the reason for rejection (see *HLC Status Codes* [on page 333](#)).

On completion of the command, the HLC updates the `MMI_vehicle_order_status` tag with a “Command Status – Command completed successfully” response (0x80).

NOTE: If the vehicle never reaches the specified station, this command never completes.

See Also

MMI_vehicle_order_status [on page 324](#)

MMI_station_arrivals [on page 313](#)

HLC to Host Controller Status/Response Memory Tags

The high-level controller writes to tags defined in the host controller memory. In order for the HLC to know which host controller to communicate with, the IP address must be defined in the **Controller IP Address** option and the location of the controller must be defined in the **Controller CPU Slot** option on the EtherNet/IP Host Controller Settings page in the Configurator Utility when creating the Node Controller Configuration File (see *MagneMotion System Configurator User Manual*, publication [MMI-UM046](#)).

Once the host controller is configured, the HLC is able to access the specific memory tags that are listed in this section. To interact properly with the HLC, all the status tags that are listed in [Table 4-47](#) must be defined in the HLC logic as described.

- Required column indicates those tags that must always be defined.
- Use column indicates the motor types that can use the tag and any conditions that are related to the tag use.

Each tag description provides the conditions under which the tags are written to by the HLC. These conditions dictate handshaking the host controller logic can key off when creating logic to control the transport system with the explicit messages that are described in the previous section.

The HLC updates the status memory tags shown in [Table 4-47](#) periodically, on state change, or when the EtherNet/IP link between the HLC and host controller has been re-established after going down so the information in these tags is always current. These tags are informational for the host controller for disseminating status about the transport system as the programmer sees fit.

The HLC only updates the response memory tags shown in [Table 4-47](#) in response to specific commands from the host controller issued via explicit messages.

Table 4-47: HLC to Host Controller Status and Response Memory Tags

Tag Name	Type	Required	Use	Page
MMI_extended_hlc_status	Status		All, must be enabled	228
MMI_extended_nc_status	Status		All, must be enabled	231
MMI_extended_vehicle_status	Status		All, must be enabled*	234
MMI_heartbeat	Status	Yes	Always	242
MMI_HLC_status	Status	Yes	Always	244
MMI_mgmt_nc_cmd_status	Response		All, must be enabled	246
MMI_motor_inverter_cmd_status	Response		QSHT, must be enabled	249
MMI_mp_command_status	Response		QS, QSHT	251

Table 4-47: HLC to Host Controller Status and Response Memory Tags

Tag Name	Type	Required	Use	Page
MMI_mp_path_end_status	Status		QS, QSHT	254
MMI_node_command_status	Response	Yes	All	261
MMI_node_controller_cmd_status	Response		All, must be enabled	263
MMI_node_controller_dio_status	Status		All, must be enabled	265
MMI_node_controller_status	Status	Yes	All	267
MMI_node_status	Status	Yes	All	269
MMI_path_command_status	Response	Yes	All	273
MMI_path_ml_faults_status	Status	Yes	MM LITE Only [†]	276
MMI_path_qs_faults_status	Status	Yes	QuickStick Only [†]	280
MMI_path_qs_ht_faults_status	Status	Yes	QSHT Only [†]	287
MMI_path_status	Status	Yes	All	295
MMI_propulsion_power_cmd_status	Response		QSHT 5700, must be enabled	298
MMI_propulsion_power_status	Status		QSHT 5700, must be enabled	301
MMI_qs_ht_sensor_map	Status		QSHT, must be enabled	303
MMI_sm_command_status	Response		All, must be enabled	306
MMI_sm_metric_data	Status		All, must be enabled	309
MMI_station_arrivals	Response	Yes	All	313
MMI_traffic_light_cmd_status	Response		All, must be enabled	315
MMI_traffic_light_status	Status		All, must be enabled	318
MMI_vehicle_command_status	Response		All, must be enabled	321
MMI_vehicle_order_status	Response	Yes	All	324
MMI_vehicle_status	Status	Yes	All, only if enabled [*]	328

* Only one vehicle status array can be used. If MMI_extended_vehicle_status is enabled, MMI_vehicle_status is disabled by the node controller.

† Only one motor fault status array can be used.

MMI_extended_hlc_status

Source/Destination

HLC ➔ Host Controller

Purpose

Used with the Extended HLC Status option. Reports the extended status of the High Level Controller in a transport system. This tag consists of a single element of type udt_MMI_extended_hlc_status.

The HLC updates the MMI_extended_hlc_status tag whenever the link between the HLC and the host controller is re-established and on content change, as soon as the change occurs the tag is updated asynchronously.

NOTE: The high-level controller must be in the operational state.

The **Enable Extended HLC Status** option must be selected (checked) on the EtherNet/IP Host Controller Settings page in the Configurator Utility when creating the Node Controller Configuration File (see *MagneMotion System Configurator User Manual*, publication [MMI-UM046](#)).

Support

This status tag is supported in the latest software release for the following product lines when **Enable Extended HLC Status** is specified in the Node Controller Configuration File:

- MagneMover LITE transport systems.
- QuickStick transport systems.
- QuickStick HT transport systems.

Tag Format

Tag Name	MMI_extended_hlc_status
Type	udt_MMI_extended_hlc_status
Array	No
Array Dimension Limits	—

UDT Field Descriptions

The UDT described in [Table 4-48](#) shows the type of the element that comprises the MMI_extended_hlc_status tag. Each UDT field is described in more detail following the table.

Table 4-48: UDT Fields for udt_MMI_extended_hlc_status

Name	Data Type	Style	Range
state	SINT	Decimal	1...3
software_major_version	SINT	Decimal	0...127
software_minor_version	SINT	Decimal	0...127
software_patch_version	SINT	Decimal	0...127
config_id	SINT	Decimal	0...20
config_valid_flag	SINT	Decimal	0...1
Data Type Size:		8 bytes	

UDT Field Details

state – Indicates the operational state of the high-level controller.

Value	Description
1	Initialization – Loading the Node Controller Configuration File or an error was detected in the configuration preventing the node controller from exiting this state. Consult the high-level controller log for additional details when the HLC does not exit this state.
2	Degraded – The high-level controller is unable to communicate with one or more node controllers in the transport system. See the high-level controller log for additional details.
3	Operational – The high-level controller configuration is valid and successfully communicating with all node controllers that are configured in the transport system.

software_major_version – The HLC software major version number.

software_minor_version – The HLC software minor version number.

software_patch_version – The HLC software patch version number.

config_id – The ID of the active Node Controller Configuration File for the high-level controller.

Value	Description
0	Using a static Node Controller Configuration File.
1...20	Using the specified managed Node Controller Configuration File.

config_valid_flag – Indicates if the high-level controller's configuration is valid.

Value	Description
0	Configuration not valid – There are errors in the Node Controller Configuration File.
1	Configuration valid – The Node Controller Configuration File is valid.

See Also

MMI_mgmt_nc_restart_cmd [on page 160](#)
MMI_mgmt_nc_set_config_cmd [on page 162](#)
MMI_extended_nc_status [on page 231](#)
MMI_HLC_status [on page 244](#)
MMI_mgmt_nc_cmd_status [on page 246](#)
MMI_node_controller_status [on page 267](#)

MMI_extended_nc_status

Source/Destination

HLC ➔ Host Controller

Purpose

Used with the Extended NC Status option. Reports the extended status of all Node Controllers in a transport system. This tag is a one-dimensional array of type `udt_MMI_extended_nc_status` indexed by Node Controller ID. Index 0 is not used since a Node Controller ID of 0 in the transport system is invalid.

This array must be sized to the maximum Node Controller ID configured in the transport system plus 1.

The HLC updates the `MMI_extended_nc_status` array whenever the link between the HLC and the host controller is re-established and on content change, as soon as the change occurs the tag is updated asynchronously.

NOTE: The high-level controller must be in the operational state.

The **Enable Extended NC Status** option must be selected (checked) on the EtherNet/IP Host Controller Settings page in the Configurator Utility when creating the Node Controller Configuration File (see *MagneMotion System Configurator User Manual*, publication [MMI-UM046](#)).

Support

This status tag is supported in the latest software release for the following product lines when **Enable Extended NC Status** is specified in the Node Controller Configuration File:

- MagneMover LITE transport systems.
- QuickStick transport systems.
- QuickStick HT transport systems.

Tag Format

Tag Name	MMI_extended_nc_status
Type	udt_MMI_extended_nc_status [Node Controllers+1]
Array	Yes
Array Dimension Limits	Minimum array size: 2 Maximum array size: max Node Controllers + 1 Array index corresponds to Node Controller ID. Since Node Controller ID 0 is invalid, array element [0] is not used.

UDT Field Descriptions

The UDT described in [Table 4-49](#) shows the type of the element that comprises the MMI_extended_nc_status tag. Each UDT field is described in more detail following the table.

Table 4-49: UDT Fields for udt_MMI_extended_nc_status

Name	Data Type	Style	Range
state	SINT	Decimal	1...3
software_major_version	SINT	Decimal	0...127
software_minor_version	SINT	Decimal	0...127
software_patch_version	SINT	Decimal	0...127
config_id	SINT	Decimal	0...20
config_valid_flag	SINT	Decimal	0, 1
Data Type Size:		8 bytes	

UDT Field Details

state – Indicates the operational state of the node controller.

Value	Description
1	Initialization – Loading the Node Controller Configuration File or an error was detected in the configuration preventing the node controller from exiting this state. Consult the node controller log for additional details when a node controller does not exit this state.
2	Disconnected – The TCP/IP connection from the high-level controller to this node controller is down.
3	Operational – The high-level controller connection to this node controller is established and the node controller is operational.

software_major_version – The node controller software major version number.

software_minor_version – The node controller software minor version number.

software_patch_version – The node controller software patch version number.

config_id – The ID of the active Node Controller Configuration File for the node controller.

Value	Description
0	Using a static Node Controller Configuration File.
1...20	Using the specified managed Node Controller Configuration File.

config_valid_flag – Indicates if this node controller configuration is valid.

Value	Description
0	Configuration not valid – There are errors in the Node Controller Configuration File.
1	Configuration valid – The Node Controller Configuration File is valid.

See Also

[*MMI_mgmt_nc_restart_cmd on page 160*](#)
[*MMI_mgmt_nc_set_config_cmd on page 162*](#)
[*MMI_extended_hlc_status on page 228*](#)
[*MMI_HLC_status on page 244*](#)
[*MMI_mgmt_nc_cmd_status on page 246*](#)
[*MMI_node_controller_status on page 267*](#)

MMI_extended_vehicle_status

Source/Destination

HLC ➔ Host Controller

Purpose

Reports the version of extended status (supported by the running node controller image) for all vehicles in the transport system. This tag is a one-dimensional array of type `udt_MMI_extended_vehicle_status` indexed by vehicle ID. Index 0 is not used since a vehicle ID of 0 in the transport system is invalid and not used. A vehicle entry is in use when the **path_id** field in the entry is nonzero. A vehicle entry is not in use when the **path_id** field in the entry is zero (that is, no vehicle exists at that index).

This array must be sized to the maximum number of vehicles in the transport system plus one (vehicle 0 is invalid). The **Max Vehicle ID** setting establishes the maximum vehicle ID that the HLC reports vehicle status for to the `MMI_extended_vehicle_status` tag in the host controller memory. **Max Vehicle ID** is defined on the **EtherNet/IP Host Controller Settings** page in the Configurator Utility when creating the Node Controller Configuration File. The maximum number of vehicles allowed in a system varies, see *MagneMotion System Configurator User Manual*, publication [MMI-UM046](#) and *MagneMotion Node Controller Interface User Manual*, publication [MMI-UM001](#) for additional information.

The HLC updates the `MMI_extended_vehicle_status` tag in the host controller memory based on the **Vehicle Records/Status Period** and **Send Vehicle Status Period** settings. Vehicle Records per Status Period and Send Vehicle Status Period are defined on the **EtherNet/IP Host Controller Settings** page in the Configurator Utility when creating the Node Controller Configuration File.

The maximum update rate is 144 vehicle records per status period. When sizing this tag, make sure that the number of vehicle records per status period does not exceed the number of entries in the `MMI_extended_vehicle_status` tag. The HLC round-robins, if needed, the writing of vehicle records in batches that range from 1 (minimum) to 144 (maximum) records as specified by **Vehicle Records/Status Period**.

NOTE: The high-level controller, node controllers, and paths in the transport system must be in the operational state.

When updating more than 144 vehicles, the HLC still updates at **Send Vehicle Status Period** intervals, but round-robins 144 vehicle records at a time. For instance, for 288 vehicles, vehicles 1...144 get updated, then next status period vehicles 145...288 get updated, next status period vehicles 1...144 get updated, and so on.

The **Use Extended Vehicle Status** must be selected (checked) on the EtherNet/IP Host Controller Settings page in the Configurator Utility when creating the Node Controller Configuration File (see *MagneMotion System Configurator User Manual*, publication [MMI-UM046](#)).

Only one vehicle status array can be used. If `MMI_extended_vehicle_status` is enabled, [MMI_vehicle_status](#) is disabled by the node controller.

Support

This status tag is supported in the latest software release for the following product lines when **Use Extended Vehicle Status** is specified in the Node Controller Configuration File:

- MagneMover LITE transport systems.
- QuickStick transport systems.
- QuickStick HT transport systems.

Tag Format

Tag Name	MMI_extended_vehicle_status
Type	udt_MMI_extended_vehicle_status [Vehicles+1]
Array	Yes
Array Dimension Limits	Minimum array size: 2 Maximum array size: max Vehicles + 1 Array index corresponds to HLC vehicle ID. Since vehicle ID 0 is invalid, array element [0] is not used.

UDT Field Descriptions

The UDT that is described in [Table 4-50](#) shows each element that is used to create the MMI_extended_vehicle_status array and its data type, style, and range. Each field is described in more detail following the table.

Table 4-50: UDT Fields for udt_MMI_extended_vehicle_status

Name	Data Type	Style	Range
path_id	INT	Decimal	0...65535
dest_path_id	INT	Decimal	0...65535
position	REAL	Float	-41.0...+41.0 (m, floating point)*
commanded_position	REAL	Float	-41.0...+41.0 (m, floating point)†
velocity	REAL	Float	0.0...100.0 (m/s, floating point)*
dest_station_id	INT	Decimal	0...2048
command	SINT	Hex	0x00, 0xB0, 0xB1, 0xB7
flags	INT	Binary	Bits 0...15 are flags
followed_vehicle_id	INT	Decimal	Command = 0x00, 0xB0, 0xB1 – 0 Command = 0xB7 – 1...65535
current_target	REAL	Float	-41.0...+41.0 (m, floating point)

Table 4-50: UDT Fields for *udt_MMI_extended_vehicle_status* (Continued)

Name	Data Type	Style	Range
reported_pid_set_index	SINT	Decimal	0...15
ordered_pid_set_index	SINT	Decimal	0...15
time_since_last_update	REAL	Float	0...2 ¹²⁷ (Seconds, floating point)
ordered_acceleration_limit	REAL	Float	0...60.0 (m/s ² , floating point)*
ordered_velocity_limit	REAL	Float	0...41.0 (m/s, floating point)*
active_station_or_follow_offset	REAL	Float	-41.0...+41.0 (m, floating point)
Data Type Size:		48 bytes	

- * This is the value from the motor, which is converted from the fixed-point value that is used by the motor.
† This is the value that is sent by the host controller.

UDT Field Details

path_id – The ID of the path where the vehicle is located. If this field is a zero, the vehicle at this index is not in use (for example, no such vehicle exists). In a multi-path transport system, this field changes as the vehicle progresses from path to path.

dest_path_id – The ID of the path where the vehicle is heading when the vehicle is under a motion command. Equal to the ID of the path where the vehicle is located if motion has completed. Equal to “0” if the vehicle has never moved (has not been given an order since startup) or if the vehicle is under an *MMI_vehicle_follow_order*.

- Vehicle Command is equal to 0x00: When the Node Controller starts up for the first time, there is no destination path, so this value is set to zero. After the vehicle has been moved, this value is the ID of the last destination path.
- Vehicle Command is equal to 0xB0 or 0xB1: The Path ID of the path for which the vehicle is heading to.
- Vehicle Command is equal to 0xB7: For a platooned follower configured for auto-decouple, this will be the destination path the vehicle will end up on after decoupling. If auto-decouple is not configured this will be 0 for this platoon follower

position – The last reported position (in meters) of the vehicle, relative to the start of the specified path (expressed as a 32-bit single-precision floating point number). Zero position is defined as the midpoint of the vehicle at the beginning of the path.

commanded_position – The commanded destination of the vehicle.

- For 0xB0 and 0xB1 is the position on the destination path.

- For 0xB7, if auto-decouple is configured in this command, it is the expected destination position after decoupling. If not auto-decoupling, it is 0.

velocity – The last reported velocity (in m/s) of the vehicle on the specified path (expressed as a 32-bit single-precision floating point number).

dest_station_id – The ID of the station the vehicle is heading to when under a station command 0xB0, otherwise it is set to zero. This field is only valid when the **command** field is 0xB0.

command – Shows the type of command currently active for the vehicle.

Value	Description
0x00	No command is in progress.
0xB0	Move vehicle to station order.
0xB1	Move vehicle to position order.
0xB7	Vehicle follow order.

flags – Vehicle status indicators.

Bit	Description
0	Vehicle Signal: 0–The motor does not detect the vehicle. 1–The motor currently in charge of the vehicle detects the vehicle.
1	Obstructed Status: 0–The vehicle is not obstructed and able to acquire permission to move further. 1–The vehicle is obstructed and unable to acquire permission to move further. Obstructions are due to either a vehicle in the way, a hardware fault, or motion is suspended. Vehicles in the way occur during normal operation when vehicles are in a queue or when a vehicle is in a switch, which prevents another vehicle from entering the switch. If after three consecutive queries of the vehicle status the obstructed bit is still set, check for an actual obstruction.

Bit	Description
2	<p> Hindered Status:</p> <p>0—The vehicle is making progress on its current move profile.</p> <p>1—The vehicle is not making progress towards the position that it has most recently been granted permission to move to. The motor continues to apply force on the vehicle to try to move it indefinitely.</p> <p>Lack of progress can happen in the following situations:</p> <ul style="list-style-type: none"> • A vehicle is held back by some external force including a foreign object on the guideway that prevents vehicle motion. • A vehicle is not in motion while under sync control. • A vehicle command uses a velocity of zero. • A vehicle command uses a PID set equal to zero. • The Control Off Position Tolerance or the Integrator Distance Threshold positions are set outside of the arrival tolerance. • The motor does not have propulsion power.
3	<p> Stall Status:</p> <p>0—The vehicle is not stalled.</p> <p>1—The vehicle has stalled. A stalled vehicle is defined as a vehicle that has jammed and the motor in control of the vehicle has had to reduce power to prevent overheating.</p>
4	<p> Lock Status:</p> <p>0—The vehicle is not locked.</p> <p>1—The vehicle is locked. Until the vehicle is unlocked, the system rejects execution of move commands for the vehicle.</p>
5	<p> Locate Status:</p> <p>0—Vehicle locate is not completed (startup is in progress).</p> <p>1—Vehicle locate has completed.</p>
6	Reserved
7	<p> Suspect Status (MM LITE, QuickStick only):</p> <p>0—The vehicle is not suspect and is free to move.</p> <p>1—The vehicle is suspect. The motor has detected that the vehicle has been manually moved out of the control region.</p> <p>Typically, control of the vehicle cannot be regained. Even if control is regained, the vehicle continues to be suspect until an MMI_vehicle_command is issued to clear the Suspect Bit or an MMI_vehicle_delete_order is issued to delete the vehicle record.</p>
8	<p> Following Upstream:</p> <p>0—This vehicle is not following another vehicle.</p> <p>1—This vehicle is following the vehicle that is specified in the followed_vehicle_id field and that vehicle is upstream from this vehicle.</p>

Bit	Description
9	<p>Following Downstream:</p> <p>0—This vehicle is not following another vehicle, or not caught up to its follow distance.</p> <p>1—This vehicle is following the vehicle that is specified in the followed_vehicle_id field and that vehicle is downstream from this vehicle.</p>
10	<p>Caught Up:</p> <p>0—This vehicle is not following another vehicle.</p> <p>1—The motor controller is reporting that this following vehicle has caught up to the vehicle that it is following and is at the commanded following distance.</p>
11	<p>Profile Stale Error:</p> <p>0—There is no profile error with the following vehicle, or this vehicle is not following another vehicle.</p> <p>1—The motor controller is reporting that this following vehicle has not received a profile from the vehicle that it is following in the expected time frame.</p>
12	<p>Excessive Following Error:</p> <p>0—There is no excess following error with the following vehicle, or this vehicle is not following another vehicle.</p> <p>1—The motor controller is reporting that this following vehicle is too far from where it is expected to be (possibly stopped receiving following profile data).</p>
13	<p>Vehicle Decoupled:</p> <p>0—This vehicle is a follower in a platoon and is not in the decoupling state, or this vehicle is not in a platoon.</p> <p>1—This vehicle is a platoon follower vehicle that is decoupling from the vehicle that it is following. This bit stays set from when the motor controller reports decoupling, until the decoupled vehicle reaches its new destination under the 0xB1 position order that was converted from the 0xB7 follow order.</p>
14, 15	Reserved

followed_vehicle_id – The ID number of the vehicle that this vehicle is following when this vehicle is under a vehicle follow order and is part of a platoon (vehicle command type is 0xB7). Equal to 0 if this vehicle is not under a vehicle follow order (vehicle command type is 0x00, 0xB0, or 0xB1).

current_target – The position that the vehicle has permission to move to (in meters) as measured from the upstream end of the current path (expressed as a 32-bit single-precision floating point number). If the vehicle has completed its current order and is stopped, this value is equal to the position of the vehicle.

reported_pid_set_index – The PID control loop set currently in use for the vehicle.

Table 4-51: PID Control Loop Sets

Value	Description
0	Use user-defined PID Set 0 – Unloaded PID values.
1	Use user-defined PID Set 1 – Loaded PID values.
2	Use user-defined PID Set 2.
3	Use user-defined PID Set 3.
4	Use user-defined PID Set 4.
5	Use user-defined PID Set 5.
6	Use user-defined PID Set 6.
7	Use user-defined PID Set 7.
8	Use user-defined PID Set 8.
9	Use user-defined PID Set 9.
10	Use user-defined PID Set 10.
11	Use user-defined PID Set 11.
12	Use user-defined PID Set 12.
13	Use user-defined PID Set 13.
14	Use user-defined PID Set 14.
15	Use user-defined PID Set 15 – Startup PID values. This PID Set is automatically used during start up. If it is not defined, PID Set 0 is scaled to 25% and used for start up.

ordered_pid_set_index – The commanded PID control loop set for vehicle movement if a movement order is in progress. This field is zero if the vehicle has not been given an order since startup. See [Table 4-51, PID Control Loop Sets, on page 240](#). See the *MagneMotion System Configurator User Manual*, publication [MMI-UM046](#) for additional information.

time_since_last_update – The time in seconds since the last vehicle status for this vehicle was received from the motor (expressed as a 32-bit single-precision floating point number).

ordered_acceleration_limit – The last acceleration order from the host (in m/s²) for the vehicle on the specified path, expressed as a 32-bit single-precision floating point number when a movement order is in progress. This field is zero if the vehicle has not been given an order since startup.

ordered_velocity_limit – The last velocity order from the host (in m/s) for the vehicle on the specified path, expressed as a 32-bit single-precision floating point number when a movement order is in progress. This field is zero if the vehicle has not been given an order since startup.

active_station_or_follow_offset –

- For a station move command: Specifies the offset from a station as the vehicle's destination (expressed as a 32-bit single precision floating-point number). Offsets are relative to station position. Offsets downstream from a station are represented by positive values whereas offsets upstream from a station are represented by negative values.
- For a platooned follow vehicle: this is the distance the follower vehicle stays behind the vehicle it is following.
 - Vehicle Command is equal to 0x00 or B1: 4 byte value set to zero
 - Vehicle Command is equal to 0xB0: 4 byte value is the offset of the station the vehicle is commanded to.
 - Vehicle Command is equal to 0xB7: 4 byte value is the offset that this vehicle is commanded to stay behind the vehicle it is following.

See Also

MMI_vehicle_position_order [on page 216](#)

MMI_vehicle_station_order [on page 221](#)

MMI_vehicle_status [on page 328](#)

MMI_heartbeat

Source/Destination

HLC ➔ Host Controller

Purpose

Provides a connection status indicator. This tag is a single DINT that the HLC increments at a rate of 3X the retry time-out for a tag request (the **Tag Request Retry Timeout** setting is defined in the Node Controller Configuration File) when connectivity from the HLC to the host controller is good. The default retry time-out is 250 ms, which sets the heartbeat rate to 750 ms. This incrementing counter acts as a CIP application layer keepalive for the EtherNet/IP implementation in the HLC. When any tag is unable to be written to by the HLC, the HLC considers the link down and indefinitely continues to try to write to this tag. The HLC considers the link up on a successful write to this tag. After the link is declared up, all status-related tags and tag arrays are written with their latest values to the host controller memory.

The host controller can monitor MMI_heartbeat tag with a timer that runs every second or whatever period is fast enough for the specific application. If the MMI_heartbeat tag doesn't change with each timer expiry, the host controller logic can declare connectivity to the HLC down.

NOTE: Explicit messages that are used to send commands to the HLC use their own independent TCP connections from the host controller to the HLC managed internally by the host controller. If the MMI_heartbeat tag is not incrementing, it may still be possible to send explicit messages to the HLC. This can happen if the message configuration for the explicit messages has the correct IP address for the HLC and Ethernet connectivity is good, but the HLC has the incorrect host controller IP address that is defined in the Node Controller Configuration File. Configuration on both sides of the connectivity setup between host controller and HLC must be correct.

Support

This status tag is supported in the latest software release for the following product lines:

- MagneMover LITE transport systems.
- QuickStick transport systems.
- QuickStick HT transport systems.

Tag Format

Tag Name	MMI_heartbeat
Type	DINT
Array	No
Array Dimension Limits	—

MMI_HLC_status

Source/Destination

HLC ➔ Host Controller

Purpose

Reports the status of the high-level controller. See *Chapter 5, Troubleshooting* for details on troubleshooting the status messages. This tag is a single tag of type `udt_MMI_HLC_status` that reports the overall state and other details of the HLC.

The high-level controller updates the `MMI_HLC_status` tag whenever the link between HLC and the host controller is re-established, or when the HLC changes state.

NOTE: The high-level controller must be in the operational state.

The **Enable Extended HLC Status** option must be selected (checked) on the EtherNet/IP Host Controller Settings page in the Configurator Utility when creating the Node Controller Configuration File (see *MagneMotion System Configurator User Manual*, publication [MMI-UM046](#)).

Support

This status tag is supported in the latest software release for the following product lines:

- MagneMover LITE transport systems.
- QuickStick transport systems.
- QuickStick HT transport systems.

Tag Format

Tag Name	MMI_HLC_status
Type	udt_MMI_HLC_status
Array	No
Array Dimension Limits	—

UDT Field Descriptions

The UDT described in [Table 4-52](#) shows the type of the element that comprises the MMI_HLC_status tag. Each UDT field is described in more detail following the table.

Table 4-52: UDT Fields for udt_MMI_HLC_status

Name	Data Type	Style	Range
state	SINT	Decimal	1...3
software_major_version	SINT	Decimal	0...127
software_minor_version	SINT	Decimal	0...127
software_patch_version	SINT	Decimal	0...127
Data Type Size:		4 bytes	

UDT Field Details

state – Shows the state of the high-level controller. The following are valid high-level controller states.

Value	Description
1	Initialization – Loading the Node Controller Configuration File or an error was detected in the configuration preventing the node controller from exiting this state. Consult the high-level controller log for additional details when the HLC does not exit this state.
2	Degraded – The high-level controller is unable to communicate with one or more node controllers in the transport system. See the high-level controller log for additional details.
3	Operational – The high-level controller configuration is valid and successfully communicating with all configured node controllers in the transport system.

software_major_version – The HLC software major version number.

software_minor_version – The HLC software minor version number.

software_patch_version – The HLC software patch version number.

See Also

[MMI_mgmt_nc_restart_cmd on page 160](#)

[MMI_mgmt_nc_set_config_cmd on page 162](#)

[MMI_extended_hlc_status on page 228](#)

[MMI_extended_nc_status on page 231](#)

[MMI_HLC_status on page 244](#)

[MMI_mgmt_nc_cmd_status on page 246](#)

[MMI_node_controller_status on page 267](#)

MMI_mgmt_nc_cmd_status

Source/Destination

HLC ➔ Host Controller

Purpose

Used with the NC Remote Management option. Reports the status of node controller remote management commands in the transport system. This tag consists of a single element of type udt_MMI_mgmt_nc_cmd_status.

The MMI_mgmt_nc_cmd_status tag is updated only in response to a [MMI_mgmt_nc_set_config_cmd](#) or a [MMI_mgmt_nc_restart_cmd](#) explicit message sent from the host controller to the HLC.

This tag can be used to handshake node controller remote management commands so the host controller logic can know that the HLC received a command, and either rejected it with an appropriate error code, or accepted it for processing. The tag can be consulted to determine if a command completed and whether it completed successfully, or if an error occurred.

NOTE: The high-level controller and node controllers in the transport system must be in the operational state.

The **Enable NC Remote Management** option must be selected (checked) on the EtherNet/IP Host Controller Settings page in the Configurator Utility when creating the Node Controller Configuration File (see *MagneMotion System Configurator User Manual*, publication [MMI-UM046](#)).

Support

This status tag is supported in the latest software release for the following product lines when **Enable NC Remote Management** is specified in the Node Controller Configuration File:

- MagneMover LITE transport systems.
- QuickStick transport systems.
- QuickStick HT transport systems.

Tag Format

Tag Name	MMI_mgmt_nc_cmd_status
Type	udt_MMI_mgmt_nc_cmd_status
Array	No
Array Dimension Limits	—

UDT Field Descriptions

The UDT described in [Table 4-53](#) shows the type of the element that comprises the MMI_mgmt_nc_cmd_status tag. Each UDT field is described in more detail following the table.

Table 4-53: UDT Fields for udt_MMI_mgmt_nc_cmd_status

Name	Data Type	Style	Range
last_command_received_count	DINT	Hex	0x0...0xFFFFFFFF
last_command_accepted_count	DINT	Hex	0x0...0xFFFFFFFF
last_command_received	SINT	Hex	0x01, 0x02
last_command_received_status	SINT	Hex	0x00, 0x0B, 0x0C, 0x0E, 0x41
last_command_accepted	SINT	Hex	0x01, 0x02
last_command_accepted_completion_status	SINT	Hex	0x00, 0x41, 0x80
Data Type Size:		12 bytes	

UDT Field Details

last_command_received_count – The HLC writes the **command_count** field of the most recently received node controller remote management command to this field. The host controller logic can use this field to determine if the HLC received a node controller remote management command.

last_command_accepted_count – The HLC writes the **command_count** field of the most recently accepted node controller remote management command to this field when the HLC accepts a command. The host controller logic can use this field to determine if the HLC accepted a node controller remote management command.

last_command_received – The HLC writes the command type of the most recently received node controller remote management command to this field.

Value	Description
0x01	Set NC Configuration command
0x02	Restart NCs command

last_command_received_status – The HLC writes an acceptance or rejection status to this field upon receiving a new node controller remote management command from the host controller. See *HLC Status Codes* [on page 333](#) for the meaning of the status codes.

last_command_accepted – The HLC writes the command type of the most recently accepted node controller remote management command to this field. This field is not updated when a received command is rejected.

Value	Description
0x01	Set NC Configuration command
0x02	Restart NCs command

last_command_accepted_completion_status – The HLC writes the completion status of the **last_command_accepted** to this field when a node controller remote management command completes or fails. See *HLC Status Codes* [on page 333](#) for the meaning of the status codes.

See Also

MMI_mgmt_nc_restart_cmd [on page 160](#)

MMI_mgmt_nc_set_config_cmd [on page 162](#)

MMI_extended_hlc_status [on page 228](#)

MMI_extended_nc_status [on page 231](#)

MMI_motor_inverter_cmd_status

Source/Destination

HLC ➔ Host Controller

Purpose

Used with the Motor Inverter Command option. Reports the status of the MMI_motor_inverter_command. This tag is a one-dimensional array of type udt_MMI_motor_inverter_cmd_status indexed by its associated motor inverter command index. The MMI_motor_inverter_cmd_status is updated only in response to a *MMI_motor_inverter_command* explicit message sent from the host controller to the HLC.

This tag can be used to handshake inverter commands so the host controller logic can know that the HLC received a command, and either accepted it for processing or rejected it, and provides an appropriate status code.

NOTE: The high-level controller, node controllers, and motors in the transport system must be in the operational state.

The **Enable Motor Inverter Command** option must be selected (checked) on the EtherNet/IP Host Controller Settings page in the Configurator Utility when creating the Node Controller Configuration File (see the *MagneMotion System Configurator User Manual*, publication [MMI-UM046](#)).

Support

This status tag is supported in the latest software release for the following product lines when **Enable Motor Inverter Command** is specified in the Node Controller Configuration File:

- QuickStick HT transport systems.

Tag Format

Tag Name	MMI_motor_inverter_cmd_status
Type	udt_MMI_motor_inverter_cmd_status [<i>cmd-index</i>]
Array	Yes
Array Dimension Limits	Minimum array size: 1 Maximum array size: 32 Array indexes correspond to the command index.

UDT Field Descriptions

The UDT described in [Table 4-54](#) shows the type of each element that comprises the `MMI_motor_inverter_cmd_status` tag. Each UDT field is described in more detail following the table.

Table 4-54: UDT Fields for `udt_MMI_motor_inverter_cmd_status`

Name	Data Type	Style	Range
<code>last_command_received_count</code>	DINT	Hex	0x0...0xFFFFFFFF
<code>last_command_accepted_count</code>	DINT	Hex	0x0...0xFFFFFFFF
<code>last_command_received_status</code>	SINT	Hex	0x00, 0x03, 0x09, 0x0B, 0x0C, 0x0D, 0x0E, 0x10, 0x16, 0x1B, 0x23
<code>last_command_accepted_completion_status</code>	SINT	Hex	0x41, 0x42, 0x80
Data Type Size:		12 bytes	

UDT Field Details

`last_command_received_count` – The HLC writes the **`command_count`** field of the most recently received motor inverter command to this field. The host controller logic can use this field to determine if the HLC received an inverter command.

`last_command_accepted_count` – The HLC writes the **`command_count`** field of the most recently accepted motor inverter command to this field when the HLC accepts a command. The host controller logic can use this field to determine if the HLC accepted a motor inverter command.

`last_command_received_status` – The HLC writes an acceptance or rejection status to this field upon receiving a new motor inverter command from the host controller. See *HLC Status Codes* [on page 333](#) for the meaning of the status codes.

`last_command_accepted_completion_status` – The HLC writes the completion status of the last command accepted to this field when a motor inverter command completes or fails. See *HLC Status Codes* [on page 333](#) for the meaning of the status codes.

See Also

MMI_motor_inverter_command [on page 165](#)

MMI_mp_command_status

Source/Destination

HLC ➔ Host Controller

Purpose

Reports the status of link/unlink commands for member paths of a Moving Path node. This tag is a two-dimensional array of type `udt_MMI_mp_command_status` indexed by Node ID and by Path ID of a member path of a Moving Path Node. Index 0 of both array dimensions is not used since a Node ID of 0 and a Path ID of 0 is invalid and not used.

The `MMI_mp_command_status` array is updated only in response to an [MMI_mp_link_command](#) or an [MMI_mp_unlink_command](#) explicit message sent from the host controller to the HLC.

This tag array can be used to handshake link/unlink commands so the host controller logic can know that the HLC received a command, and either accepted it for processing, or rejected it with an appropriate error code. The tag array can be consulted to determine if a command completed and whether it completed successfully, or if an error occurred.

Support

This command is supported in the latest software release for the following product lines:

- QuickStick transport systems.
- QuickStick HT transport systems.

Tag Format

Tag Name	MMI_mp_command_status
Type	<code>udt_MMI_mp_command_status</code> [Nodes+1, Paths+1]
Array	Yes
Array Dimension Limits *	<p>Minimum array size: [2, 2] Maximum array size: max Nodes + 1, max Paths + 1 Array indexes correspond to HLC Node ID and HLC Path ID Since Node ID 0 is invalid and Path ID 0 is invalid, array elements [0, 0], [0, 1], [0, 2], and so on, and [1, 0], [2, 0], [3, 0], and so on, are not used.</p>

* Array indexes that correspond to a Path and Node which don't coincide with each other, will be empty.

UDT Field Descriptions

The UDT described in [Table 4-55](#) shows the type of each element that comprises the MMI_mp_command_status tag. Each UDT field is described in more detail following the table.

Table 4-55: UDT Fields for udt_MMI_mp_command_status

Field Name	Data Type	Style	Range
last_command_received_count	DINT	Hex	0x0...0xFFFFFFFF
last_command_accepted_count	DINT	Hex	0x0...0xFFFFFFFF
last_command_received	SINT	Hex	0x02, 0x03
last_command_received_status	SINT	Hex	0x00, 0x03, 0x09, 0x0C, 0x0D, 0x10, 0x19, 0x20
last_command_accepted	SINT	Hex	0x02, 0x03
last_command_accepted_completion_status	SINT	Hex	0x00, 0x41, 0x42, 0x80
Data Type Size:		12 bytes	

UDT Field Details

last_command_received_count – The HLC writes the **command_count** field of the most recently received link/unlink command for the member path of the corresponding Moving Path node to this field. The host controller logic can use this field to determine if the HLC received a link/unlink command.

last_command_accepted_count – The HLC writes the **command_count** field of the most recently accepted link/unlink command for the member path of the corresponding Moving Path node to this field when the HLC accepts a command. The host controller logic can use this field to determine if the HLC accepted a link/unlink command.

last_command_received – The HLC writes the command type of the most recently received link/unlink command for the member path of the corresponding Moving Path node to this field.

last_command_received_status – The HLC writes an acceptance or rejection status to this field upon receiving a new link/unlink command from the host controller. See *HLC Status Codes* [on page 333](#) for the meaning of the status codes.

last_command_accepted – The HLC writes the command type of the most recently accepted link/unlink command for the member path of the corresponding Moving Path node to this field. This field is not updated when a received command is rejected.

last_command_accepted_completion_status – The HLC writes the completion status of the **last_command_accepted** to this field when the link/unlink command completes or fails. See *HLC Status Codes* [on page 333](#) for the meaning of the status codes.

See Also

MMI_mp_link_command [on page 169](#)

MMI_mp_unlink_command [on page 173](#)

MMI_mp_command_status [on page 251](#)

MMI_mp_path_end_status

Source/Destination

HLC ➔ Host Controller

Purpose

Reports the status of member path ends of all Moving Path nodes in the transport system. This tag is a two-dimensional array of type `udt_MMI_mp_path_end_status` indexed by node ID and by the path ID of the Moving Path node's member paths. Index 0 of both array dimensions is not used since a Node ID of 0 and a Path ID of 0 is invalid and not used.

The HLC updates the `MMI_mp_path_end_status` array whenever the link between the HLC and the host controller is re-established and on content change, as soon as the change occurs the tag is updated asynchronously.

Support

This command is supported in the latest software release for the following product lines:

- QuickStick transport systems.
- QuickStick HT transport systems.

Tag Format

Tag Name	MMI_mp_path_end_status
Type	udt_MMI_mp_path_end_status [Nodes+1, Paths+1]
Array	Yes
Array Dimension Limits *	Minimum array size: [2, 2] Maximum array size: max Nodes + 1, max Paths + 1 Array indexes correspond to HLC Node ID and HLC Path ID Since Node ID 0 is invalid and Path ID 0 is invalid, array elements [0, 0], [0, 1], [0, 2], and so on, and [1, 0], [2, 0], [3, 0], and so on, are not used.

* Array indexes that correspond to a Path and Node which don't coincide with each other, will be empty.

UDT Field Descriptions

The UDT described in [Table 4-56](#) shows the type of each element that comprises the MMI_mp_path_end_status tag. Each UDT field is described in more detail following the table.

Table 4-56: UDT Fields for udt_MMI_mp_path_end_status

Name	Data Type	Style	Range
path_end_state	SINT	Decimal	1...7
path_end_role	SINT	Decimal	1...3
path_end_type	SINT	Decimal	1, 2
peer_node_id	INT	Decimal	0...65535
requested_path_id	INT	Decimal	0...65535
linked_path_id	INT	Decimal	0...65535
last_allowed_vehicle_id	INT	Decimal	0...65535
requesting_vehicle_id	INT	Decimal	0...65535
last_entered_vehicle_id	INT	Decimal	0...65535
owner_vehicle_id	INT	Decimal	0...65535
last_exited_vehicle_id	INT	Decimal	0...65535
alignment_request_count	INT	Hex	0x0...0xFFFF
status_change_count	DINT	Hex	0x0...0xFFFFFFFF
Data Type Size:		28 bytes	

UDT Field Details

path_end_state – The state for this path end.

Table 4-57: Moving Path Node Path End States

Value	Name	Description
1	unlinked	<p>The path end is not linked.</p> <ul style="list-style-type: none"> This is the path end state for all paths in the node following the reset of any path that is connected to the node or a node controller restart. Motor-to-motor message forwarding is disabled in this state. Vehicles cannot navigate the node when a path end is in this state. When the headway for a vehicle approaches a path end, the motor sends an entry request to the NC responsible for the node requesting permission to move the vehicle beyond the entry gate and navigate the node. When the node controller receives the entry request from a path end in the <i>unlinked</i> state, it checks the routing that is needed for the vehicle to reach its destination, updates the <i>requested_path_id</i> and <i>requesting_vehicle_id</i> to notify the host controller of the alignment that is required, and changes the path end state to the <i>unlinked_alignment_requested</i> state. The HLC sends the updated path end status to the host controller. The host controller responds with an <i>MMI_mp_link_command</i> when a path that provides a route to the vehicle's destination is aligned.
2	linked_unlink_pending	<p>The path end is linked with an unlink pending.</p> <ul style="list-style-type: none"> A junction remains linked allowing vehicles already navigating the junction to proceed. Vehicles that request entry to the junction are not granted permission to breach the entry gates. When all navigating vehicles are clear of the junction, the path end transitions to the <i>unlinked</i> state.
3	unlinked_alignment_requested	<p>The path end is not linked and a request to align it with another path is pending.</p> <ul style="list-style-type: none"> Motor-to-motor message forwarding is disabled in this state. Vehicles cannot navigate the node when a path end is in this state. A transition to this state indicates that a vehicle is requesting permission to navigate the node. Entry requests are processed in this state to check that the requested alignment is still a valid route for the entering vehicle. When an <i>MMI_mp_link_command</i> is received from the host controller the path end transitions to the <i>linked</i> state.

Table 4-57: Moving Path Node Path End States (Continued)

Value	Name	Description
4	linked	<p>The Control Path is linked to a Peer Path to form a path junction.</p> <ul style="list-style-type: none"> Motor-to-motor message forwarding is enabled in this state. Vehicles can navigate the node from this path end. In the <i>linked</i> state, the node controller grants entry requests if the path junction offers an equivalent route to the requesting vehicle's destination. If <i>last_allowed_vehicle_id</i> is zero, entry requests are granted as long as the path junction offers a route to the vehicle's destination. If <i>last_allowed_vehicle_id</i> is set, the path junction continues to let vehicles in until the Last Allowed Vehicle ID enters. When the last vehicle enters, the path junction transitions to the <i>linked_unlink_pending</i> state. If the host controller sends an unlink command when the junction is in the linked state and there are no vehicles navigating the junction, the junction is unlinked and both the Control Path and Peer Path ends transition to the <i>unlinked</i> state. If there is a vehicle navigating the junction, the path junction transitions to the <i>linked_unlink_pending</i> state.
5	linked_comm_loss	<p>Communication to a linked path end is lost.</p> <ul style="list-style-type: none"> Path ends in the <i>linked</i> or <i>linked_unlink_pending</i> states enter this state when communication between the host controller and the HLC, or communication between the HLC and an NC is lost. Motor-to-motor message forwarding is enabled in this state. No additional vehicles are granted permission to navigate. Navigating vehicles are allowed to continue navigation. A link command or an unlink command is required once communication is restored to transition from this state.
6	linked_peer	<p>The Peer Path is linked to a Control Path to form a path junction.</p> <ul style="list-style-type: none"> The Peer Path remains in this state until the path junction is unlinked.

Table 4-57: Moving Path Node Path End States (Continued)

Value	Name	Description
7	unlinked_vehicle_present	<p>There is a vehicle that is in an unlinked junction.</p> <ul style="list-style-type: none"> A vehicle was located during startup and extends past the entry gate on a path end in the Moving Path node. The host controller must either clear the vehicle by moving it away from the path end or link a path to move it through the junction. <p>NOTE: When any Moving Path node path end is in this state, the <i>device_status</i> field (bits 0-3) in the node's <i>MMI_node_status</i> tag is set to "Startup Fault".</p>

path_end_role – The role for this path end.

Value	Description
1	Unlinked Path – The path is not linked to another path.
2	Control Path – The path that is specified in an MP Link command to link to a Peer Path. Once linked, this path remains the Control Path for the life of the junction until the junction transitions to the unlinked state.
3	Peer Path – The path that is specified in an MP Link command to link from a Control Path. Once linked, this path remains the Peer Path for the life of the junction until the junction transitions to the unlinked state.

NOTE: A linking path can be defined as a peer path for one node and a control path for another node. However, it should not be designated as a peer path for two nodes or as a control path for both nodes.

path_end_type – The type of path end.

Value	Description
1	Fixed Path End – A path end that is configured as a specific-route path.
2	Moving Path End – A path end that is configured as an equivalent-route path.

peer_node_id – The ID of the node at the far end of this Moving Path node member path.

Value	Description
0	The member path is a moving path and there is no Moving Path node at the far end of the path. The member path is a fixed path (that is, configured as a specific-route path).
1...65535	The member path is a moving path (that is, configured as an equivalent-route path). This is the ID of the node at the far end of the path.

requested_path_id – The Requested Path ID field is written when a path end transitions to the [unlinked_alignment_requested](#) state. This state is a signal to the host controller that a vehicle is requesting permission to navigate a Moving Path node.

The requested_path_id persists on the Control Path from the time the Control Path enters the [unlinked_alignment_requested](#) state until the Control Path is unlinked. The Requested Path ID field is cleared on the Peer Path when linked with a Control Path to form a junction. See [Table 4-57](#) for detailed descriptions of the path end states.

Value	Description
0	The path end is not linked and the host controller can align any equivalent moving path to provide a route to the vehicle's destination.
1...65535	The requested path must be aligned to provide a route to the vehicle's destination.

linked_path_id – The ID of the Peer Path that is linked to this Control Path.

Value	Description
0	No Peer Path is linked.
1...65535	The Control Path and specified Peer Path are linked to form a path junction.

last_allowed_vehicle_id – The ID of the last vehicle that is allowed through the node. The HLC updates the Last Allowed Vehicle ID field only on the Control Path of a linked junction.

Value	Description
0	The HLC keeps the Control and Peer Paths linked until the host controller sends an MMI_mp_unlink_command . Vehicles approaching the junction are granted permission to navigate the junction if the Control Path offers an equivalent route to the vehicle's destination.
1...65535	Once the last allowed vehicle begins to navigate the path junction, the Control Path end status transitions to the linked_unlink_pending state and no other vehicles are granted permission to navigate the junction. The HLC unlinks the path junction as soon as all navigating vehicles are clear of the path junction.

requesting_vehicle_id – The ID of the vehicle requesting permission to navigate the specified Moving Path node.

When in the [unlinked_alignment_requested](#) state, the requesting_vehicle_id field identifies the vehicle requesting permission to navigate the specified Moving Path node. The requesting_vehicle_id field is cleared once the path end is linked and the vehicle is granted permission to navigate the junction.

last_entered_vehicle_id – The ID of the last vehicle that was granted permission to navigate the junction. Updated by the HLC only on the Control Path of a linked junction.

Value	Description
0	No active vehicle. Set to zero before any vehicle is granted permission and is cleared when the last entered vehicle clears the junction.
1...65535	Set to the ID of the entering vehicle when that vehicle is granted permission to enter.

owner_vehicle_id – The ID of the vehicle that currently owns the path junction. Updated by the HLC only on the Control Path of a linked junction.

Value	Description
0	There are no vehicles navigating the path junction (no vehicle owns the junction).
1...65535	The ID of the vehicle that currently owns the path junction.

last_exited_vehicle_id – The ID of the most recent vehicle to clear the path junction. Updated by the HLC only on the Control Path of a linked junction.

Value	Description
0	No vehicle has exited the junction since it was linked. The last_exited_vehicle_id field is zeroed when the path junction is unlinked.
1...65535	ID of the most recent vehicle to clear the junction.

alignment_request_count – A sequence count, unique to the most recent alignment request, which the HLC increments whenever alignment for a path end is requested.

The HLC maintains the Alignment Request Count for each path end. It is initialized to 0 when the HLC restarts and no alignment requests have been issued, it is incremented for each new alignment request, and continues from 1 when it rolls over.

status_change_count – A sequence count, unique to the specified path end, incremented by the HLC whenever status for a path end is updated.

The HLC maintains the status_change_count for each member path end. It is initialized to 0 when the HLC restarts.

See Also

[MMI_mp_link_command](#) [on page 169](#)
[MMI_mp_unlink_command](#) [on page 173](#)
[MMI_mp_command_status](#) [on page 251](#)

MMI_node_command_status

Source/Destination

HLC ➔ Host Controller

Purpose

Reports the status of all node commands in the transport system. This tag is a one-dimensional array of type `udt_MMI_node_command_status` indexed by Node ID. Index 0 is not used since a Node ID of 0 in the transport system is invalid and not used.

This array must be sized to the maximum Node ID configured in the transport system plus 1. For example if the maximum Node ID is 5, this array must be dimensioned to 6. The maximum number of nodes allowed in a system varies, see the *MagneMotion System Configurator User Manual*, publication [MMI-UM046](#) and *MagneMotion Node Controller Interface User Manual*, publication [MMI-UM001](#) for additional information.

The high-level controller updates the `MMI_node_command_status` tag only in response to an [MMI_generic_node_command](#) or [MMI_terminus_node_command](#) explicit message that is sent from the host controller to the HLC.

This tag array can be used to handshake a node command so the host controller logic can know that the HLC received a node command, and either rejected it with an appropriate error code, or accepted it for processing.

Support

This status tag is supported in the latest software release for the following product lines:

- MagneMover LITE transport systems.
- QuickStick transport systems.
- QuickStick HT transport systems.

Tag Format

Tag Name	MMI_node_command_status
Type	udt_MMI_node_command_status [Nodes+1]
Array	Yes
Array Dimension Limits	Minimum array size: 2 Maximum array size: max Nodes + 1 Array index corresponds to HLC Node ID. Since Node ID 0 is invalid, array element [0] is not used.

UDT Field Descriptions

The UDT described in [Table 4-58](#) shows the type of each element that comprises the `MMI_node_command_status` tag. Each UDT field is described in more detail following the table.

Table 4-58: UDT Fields for `udt_MMI_node_command_status`

Name	Data Type	Style	Range
<code>last_command_received_count</code>	DINT	Hex	0x0...0xFFFFFFFF
<code>last_command_accepted_count</code>	DINT	Hex	0x0...0xFFFFFFFF
<code>last_command_received_status</code>	SINT	Hex	MMI_generic_node_command 0x00, 0x0B, 0x41 MMI_terminus_node_command 0x00, 0x01, 0x05, 0x09...0x0F, 0x10, 0x41
Data Type Size:		12 bytes	

UDT Field Details

`last_command_received_count` – The HLC writes the **`command_count`** field of the most recently received generic node or terminus node command for the corresponding Node ID to this field. The host controller logic can use this field to determine if the HLC received a node command.

`last_command_accepted_count` – The HLC writes the **`command_count`** field of the most recently accepted generic node or terminus node command for the corresponding Node ID to this field when the HLC accepts a command. The host controller logic can use this field to determine if the HLC accepted a node command.

`last_command_received_status` – The HLC writes an acceptance or rejection status to this field upon receiving a new generic node or terminus node command from the host controller. See *HLC Status Codes* [on page 333](#) for the meaning of the status codes.

See Also

[MMI_generic_node_command on page 157](#)
[MMI_terminus_node_command on page 202](#)
[MMI_node_status on page 269](#)

MMI_node_controller_cmd_status

Source/Destination

HLC ➔ Host Controller

Purpose

Used with the Digital I/O Commands option. Reports the status of all node controller-related commands in the transport system. This tag is a one-dimensional array of type `udt_MMI_node_controller_cmd_status` indexed by Node Controller ID. Index 0 is not used since a Node Controller ID of 0 in the transport system is invalid and not used.

This array must be sized to the maximum Node Controller ID configured in the transport system plus 1. For example if the maximum Node Controller ID is 3, this array must be dimensioned to 4. The maximum number of node controllers allowed in a system varies, see the *MagneMotion System Configurator User Manual*, publication [MMI-UM046](#) and *MagneMotion Node Controller Interface User Manual*, publication [MMI-UM001](#) for additional information.

The high-level controller updates the `MMI_node_controller_cmd_status` tag in response to an *MMI_node_controller_command* explicit message that is sent from the host controller to the HLC.

This tag array can be used to handshake the digital I/O commands so the host controller logic can know that the HLC received a digital I/O command, and either rejected it with an appropriate error code, or accepted it for processing.

NOTE: The high-level controller and node controllers in the transport system must be in the operational state.

The **Enable Digital I/O Commands** option must be selected (checked) on the EtherNet/IP Host Controller Settings page in the Configurator Utility when creating the Node Controller Configuration File (see *MagneMotion System Configurator User Manual*, publication [MMI-UM046](#)).

Support

This status tag is supported in the latest software release for the following product lines when **Enable Digital I/O Commands** is specified in the Node Controller Configuration File:

- MagneMover LITE transport systems.
- QuickStick transport systems.
- QuickStick HT transport systems.

Tag Format

Tag Name	MMI_node_controller_cmd_status
Type	udt_MMI_node_controller_cmd_status [Node Controllers+1]
Array	Yes
Array Dimension Limits	Minimum array size: 2 Maximum array size: max Node Controllers + 1 Array index corresponds to the Node Controller ID. Since Node Controller ID 0 is invalid, array element [0] is not used.

UDT Field Descriptions

The UDT described in [Table 4-59](#) shows the type of each element that comprises the MMI_node_controller_cmd_status tag. Each UDT field is described in more detail following the table.

Table 4-59: UDT Fields for udt_MMI_node_controller_cmd_status

Name	Data Type	Style	Range
last_command_received_count	DINT	Hex	0x0...0xFFFFFFFF
last_command_accepted_count	DINT	Hex	0x0...0xFFFFFFFF
last_command_received_status	SINT	Hex	0x00, 0x0B, 0x0C, 0x25, 0x41*
Data Type Size:	12 bytes		

* Indicates a communication error while sending a message to the specific node controller containing the Digital Outputs.

UDT Field Details

last_command_received_count – The HLC writes the **command_count** field of the most recently received node controller digital I/O command to this field. The host controller logic can use this field to determine if the HLC received a node controller digital I/O command.

last_command_accepted_count – The HLC writes the **command_count** field of the most recently accepted node controller digital I/O command to this field when the HLC accepts a command. The host controller logic can use this field to determine if the HLC accepted a node controller digital I/O command.

last_command_received_status – The HLC writes an acceptance or rejection status to this field upon receiving a new node controller digital I/O command from the host controller. See *HLC Status Codes* [on page 333](#) for the meaning of the status codes.

See Also

MMI_node_controller_command [on page 176](#)

MMI_node_controller_dio_status

Source/Destination

HLC ➔ Host Controller

Purpose

Used with the Digital I/O Commands option. Reports the status of the digital I/O for all node controllers in the transport system. This tag is a one-dimensional array of type `udt_MMI_node_controller_dio_status` indexed by Node Controller ID. Index 0 is not used since a Node Controller ID of 0 in the transport system is invalid and not used.

This array must be sized to the maximum Node Controller ID configured in the transport system plus 1. For example if the maximum Node Controller ID is 10, this array must be dimensioned to 11. The maximum number of node controllers allowed in a system varies, see the *MagneMotion System Configurator User Manual*, publication [MMI-UM046](#) and *MagneMotion Node Controller Interface User Manual*, publication [MMI-UM001](#) for additional information.

The high-level controller updates the `MMI_node_controller_dio_status` tag whenever the link between HLC and host controller is re-established, when the link from the HLC to a particular node controller is re-established, and whenever the digital I/O values change.

NOTE: The high-level controller and node controllers in the transport system must be in the operational state.

The **Enable Digital I/O Commands** option must be selected (checked) on the EtherNet/IP Host Controller Settings page in the Configurator Utility when creating the Node Controller Configuration File (see *MagneMotion System Configurator User Manual*, publication [MMI-UM046](#)).

Support

This status tag is supported in the latest software release for the following product lines when **Enable Digital I/O Commands** is specified in the Node Controller Configuration File:

- MagneMover LITE transport systems.
- QuickStick transport systems.
- QuickStick HT transport systems.

Tag Format

Tag Name	MMI_node_controller_dio_status
Type	udt_MMI_node_controller_dio_status [Node Controllers+1]
Array	Yes
Array Dimension Limits	Minimum array size: 2 Maximum array size: max Node Controllers + 1 Array index corresponds to the Node Controller ID. Since Node Controller ID 0 is invalid, array element [0] is not used.

UDT Field Descriptions

The UDT described in [Table 4-60](#) shows the type of each element that comprises the MMI_node_controller_dio_status tag. Each UDT field is described in more detail following the table.

Table 4-60: UDT Fields for udt_MMI_node_controller_dio_status

Name	Data Type	Style	Range
num_dio_input_bits	SINT	Decimal	0...32
num_dio_output_bits	SINT	Decimal	0...32
dio_inputs	DINT	Hex	0x0...0xFFFFFFFF
dio_outputs	DINT	Hex	0x0...0xFFFFFFFF
Data Type Size:		12 bytes	

UDT Field Details

num_dio_input_bits – The number of digital I/O Inputs available on the specified node controller. Typical values are 0, 8, 16, or 32.

num_dio_output_bits – The number of digital I/O Outputs available on the specified node controller. Typical values are 0, 8, 16, or 32.

dio_inputs – The digital I/O input data for the specified node controller.

dio_outputs – The digital I/O output data for the specified node controller.

See Also

MMI_node_controller_command [on page 176](#)

MMI_node_controller_status

Source/Destination

HLC ➔ Host Controller

Purpose

Reports the status of all node controllers in the transport system. See [Chapter 5, Troubleshooting](#) for details on troubleshooting the status messages. This tag is a one-dimensional array of type `udt_MMI_node_controller_status` indexed by Node Controller ID. Index 0 is not used since a Node Controller ID of 0 in the transport system is invalid and not used.

This array must be sized to the maximum Node Controller ID configured in the transport system plus 1. For example if the maximum Node Controller ID is 10, this array must be dimensioned to 11. The maximum number of node controllers allowed in a system varies, see the *MagneMotion System Configurator User Manual*, publication [MMI-UM046](#) and *MagneMotion Node Controller Interface User Manual*, publication [MMI-UM001](#) for additional information.

The high-level controller updates the `MMI_node_controller_status` tag whenever the link between HLC and host controller is re-established, when the link from the HLC to a particular node controller changes status, and when a node controller changes state.

NOTE: The high-level controller must be in the operational state.

Support

This status tag is supported in the latest software release for the following product lines:

- MagneMover LITE transport systems.
- QuickStick transport systems.
- QuickStick HT transport systems.

Tag Format

Tag Name	MMI_node_controller_status
Type	udt_MMI_node_controller_status [Node Controllers+1]
Array	Yes
Array Dimension Limits	Minimum array size: 2 Maximum array size: max Node Controllers + 1 Array index corresponds to the Node Controller ID. Since Node Controller ID 0 is invalid, array element [0] is not used.

UDT Field Descriptions

The UDT described in [Table 4-61](#) shows the type of each element that comprises the MMI_node_controller_status tag. Each UDT field is described in more detail following the table.

Table 4-61: UDT Fields for udt_MMI_node_controller_status

Name	Data Type	Style	Range
state	SINT	Decimal	1...3
software_major_version	SINT	Decimal	0...127
software_minor_version	SINT	Decimal	0...127
software_patch_version	SINT	Decimal	0...127
Data Type Size:		4 bytes	

UDT Field Details

state – Shows the state of the node controller.

Value	Description
1	Initialization – Loading the Node Controller Configuration File or an error was detected in the configuration preventing the node controller from exiting this state. Consult the node controller log for additional details when the node controller does not exit this state.
2	Disconnected – The TCP/IP connection from the high-level controller to this node controller is down.
3	Operational – The high-level controller connection to this node controller is established and the node controller is operational.

software_major_version – The node controller software major version number.

software_minor_version – The node controller software minor version number.

software_patch_version – The node controller software patch version number.

See Also

[MMI_mgmt_nc_restart_cmd on page 160](#)
[MMI_mgmt_nc_set_config_cmd on page 162](#)
[MMI_extended_hlc_status on page 228](#)
[MMI_extended_nc_status on page 231](#)
[MMI_HLC_status on page 244](#)
[MMI_mgmt_nc_cmd_status on page 246](#)

MMI_node_status

Source/Destination

HLC ➔ Host Controller

Purpose

Reports the status of all nodes in the transport system. See *Chapter 5, Troubleshooting* for details on troubleshooting the status messages. This tag is a one-dimensional array of type `udt_MMI_node_status` indexed by Node ID. Index 0 is not used since a Node ID of 0 in the transport system is invalid and not used.

This array must be sized to the maximum Node ID configured in the transport system plus 1. For example if the maximum Node ID is 4, this array must be dimensioned to 5. The maximum number of nodes allowed in a system varies, see the *MagneMotion System Configurator User Manual*, publication [MMI-UM046](#) and *MagneMotion Node Controller Interface User Manual*, publication [MMI-UM001](#) for additional information.

The high-level controller updates the `MMI_node_status` tag whenever the link between HLC and host controller is re-established and on content change (status changes), as soon as the change occurs the tag is updated asynchronously.

Support

This status tag is supported in the latest software release for the following product lines:

- MagneMover LITE transport systems.
- QuickStick transport systems.
- QuickStick HT transport systems.

Tag Format

Tag Name	MMI_node_status
Type	udt_MMI_node_status [Nodes+1]
Array	Yes
Array Dimension Limits	Minimum array size: 2 Maximum array size: max Nodes + 1 Array index corresponds to Node ID. Since Node ID 0 is invalid, array element [0] is not used.

UDT Field Descriptions

The UDT described in [Table 4-62](#) shows the type of each element that comprises the MMI_node_status tag. Each UDT field is described in more detail following the table.

Table 4-62: UDT Fields for udt_MMI_node_status

Name	Data Type	Style	Range
vehicle_id	INT	Decimal	1...65535
node_type	SINT	Decimal	0...12
terminus_signals	SINT	Binary	Bits 0...7 are flags
requested_position	INT	Decimal	0...8
reported_position	INT	Decimal	0...8
device_status	SINT	Hex	Bits 0...7 are flags
Data Type Size:		12 bytes	

UDT Field Details

vehicle_id – The ID of the vehicle currently navigating this node or zero when the node is idle and no vehicle is navigating the node. A vehicle is said to be navigating the node (or owning the node) when the node is within the motor permissions that are required for brick-wall headway for the vehicle. The vehicle is considered to have left the node once the center of the vehicle is 1/2 vehicle length plus 1/4 motor cycle past the node in the vehicle’s direction of travel.

node_type – The type of the node for which status is provided (see *Node Type Descriptions and Usage* [on page 102](#)).

Value	Node Type	Transport Systems
0	Relay Node	MagneMover LITE, QuickStick, QuickStick HT
1	Merge Node	MagneMover LITE
2	Diverge Node	MagneMover LITE
3	Reserved	—
4	Terminus Node	MagneMover LITE, QuickStick, QuickStick HT
5	Simple Node	MagneMover LITE, QuickStick, QuickStick HT
6	Reserved	—
7	Reserved	—
8	Merge-Diverge Node	MagneMover LITE
9	Gateway Node	MagneMover LITE, QuickStick, QuickStick HT
10	Overtravel Node	QuickStick, QuickStick HT
11	Moving Path Node	QuickStick, QuickStick HT
12	Reserved	—

terminus_signals – The terminus signals for the node where the bits have the following meanings.

- Valid for **node_type** 4 (Terminus) only.
- Not used for all other node types (the terminus signal bits are 0).

Bit	Description
0	ENTRY_REQUESTED (last input state)
1	EXIT_ALLOWED (last input state)
2, 3	Reserved
4	ENTRY_CLEAR (output) NOTE: This bit is masked to be always Off for MagneMover LITE systems.
5	ENTRY_ALLOWED (output)
6	EXITING (output)
7	Reserved

requested_position – The position the node apparatus was last commanded to move to or zero if no position previously commanded.

- Valid for a **node_type** of 1, 2, or 8 (Merge, Diverge, or Merge-Diverge).
- Not used for a **node_type** of 0, 4, 5, or 9 (Relay, Terminus, Simple, or Gateway), this byte is zero.

Value	Description
0	No switch present or position has not yet been requested
1	Switch position 1 (straight if merge or diverge)
2	Switch position 2 (curve if merge or diverge)
3	Switch position 3
4	Switch position 4
5	Switch position 5
6	Switch position 6
7	Switch position 7
8	Switch position 8

reported_position – The position the node apparatus was last reported to be in or zero if no position previously reported.

- Valid for a **node_type** of 1, 2, or 8 (Merge, Diverge, or Merge-Diverge).
- Not used for a **node_type** of 0, 4, 5, or 9 (Relay, Terminus, Simple, or Gateway), this byte is zero.

Value	Description
0	No switch present
1	Switch position 1 (straight if merge or diverge)
2	Switch position 2 (curve if merge or diverge)
3	Switch position 3
4	Switch position 4
5	Switch position 5
6	Switch position 6
7	Switch position 7
8	Switch position 8

device_status – Bits 0...4 provide the status of the node.

- Valid for a **node_type** of 1, 2, or 8 (Merge, Diverge, or Merge-Diverge). Bits 0...3 provide the status of the device that is affiliated with the node. Bit 4 is not used.
- Not used for a **node_type** of 0, 4, or 5 (Relay, Terminus, or Simple), bits 0...4 are zero.
- Valid for a **node_type** of 9 (Gateway), Bits 0...3 are not used, Bit 4 is the status of the node.

Bit	Description
0...3	Node Status: 0–No Device Present 1–Initializing 2–Faulted 3–Operational 4–Junction/Startup Fault 5–Reserved 6–Reserved
4	Node Flag: 0–Enabled 1–Disabled
5...7	Reserved

See Also

MMI_generic_node_command on page 157
MMI_terminus_node_command on page 202
MMI_node_command_status on page 261

MMI_path_command_status

Source/Destination

HLC ➔ Host Controller

Purpose

Reports the status of all path commands in the transport system. This tag is a one-dimensional array of type `udt_MMI_path_command_status` indexed by path ID. Index 0 is not used since a path ID of 0 in the transport system is invalid and not used.

This array must be sized to the maximum path ID configured in the transport system plus 1. For example if the maximum path ID is 10, this array must be dimensioned to 11. The maximum number of paths allowed in a system varies, see the *MagneMotion System Configurator User Manual*, publication [MMI-UM046](#) and *MagneMotion Node Controller Interface User Manual*, publication [MMI-UM001](#) for additional information.

The high-level controller updates the `MMI_path_command_status` tag only in response to an `MMI_path_command` explicit message that is sent from the host controller to the HLC.

This tag array can be used to handshake the path commands so the host controller logic can know that the HLC received a path command, and either rejected it with an appropriate error code, or accepted it for processing. For path commands that take significant time to complete execution (start-up and reset), this tag array can be consulted to determine if the command completed and whether it completed successfully, or if an error occurred.

Support

This status tag is supported in the latest software release for the following product lines:

- MagneMover LITE transport systems.
- QuickStick transport systems.
- QuickStick HT transport systems.

Tag Format

Tag Name	MMI_path_command_status
Type	udt_MMI_path_command_status [Paths+1]
Array	Yes
Array Dimension Limits	Minimum array size: 2 Maximum array size: max Paths + 1 Array index corresponds to path ID. Since path ID 0 is invalid, array element [0] is not used.

UDT Field Descriptions

The UDT described in [Table 4-63](#) shows the type of each element that comprises the MMI_path_command_status tag. Each UDT field is described in more detail following the table.

Table 4-63: UDT Fields for udt_MMI_path_command_status

Name	Data Type	Style	Range
last_command_received_count	DINT	Hex	0x0...0xFFFFFFFF
last_command_accepted_count	DINT	Hex	0x0...0xFFFFFFFF
last_command_received	SINT	Hex	0xB2...0xB4, 0xB8, 0xBC, 0xBD
last_command_received_status	SINT	Hex	Startup (0xB2) 0x00, 0x03, 0x05, 0x06, 0x0B...0x0D, 0x10, 0x24 Resume (0xB3) 0x00, 0x03, 0x05, 0x06, 0x0A...0x0D, 0x10 Suspend (0xB4) 0x00, 0x03, 0x0B...0x0D, 0x10 Reset (0xB8) 0x00, 0x03, 0x0C, 0x0D, 0x10, 0x41 FastStop (0xBC) 0x00, 0x03, 0x0B...0x0D, 0x10 Warm Reset (0xBD) 0x00, 0x03, 0x0C, 0x0D, 0x10, 0x41
last_command_accepted	SINT	Hex	0xB2...0xB4, 0xB8, 0xBC, 0xBD
last_command_accepted_completion_status	SINT	Hex	Startup 0x41, 0x42, 0x44, 0x80 Resume 0x41 Suspend 0x41 Reset 0x41, 0x80 FastStop 0x41 Warm Reset 0x41, 0x80
Data Type Size:		12 bytes	

UDT Field Details

last_command_received_count – The HLC writes the **command_count** field of the most recently received path command for the corresponding path ID to this field. The host controller logic can use this field to determine if the HLC received a path command.

last_command_accepted_count – The HLC writes the **command_count** field of the most recently accepted path command for the corresponding path ID to this field when the node controller accepts the command. The host controller logic can use this field to determine if the HLC accepted a path command.

last_command_received – The HLC writes the command field of the most recently received path command for the corresponding path ID to this field.

Value	Description
0x02	Startup
0xB3	Resume
0xB4	Suspend
0xB8	Reset
0xBC	FastStop
0xBD	Warm Reset

last_command_received_status – The HLC writes an acceptance or rejection status to this field upon receiving a new path command for the corresponding path ID from the host controller. See *HLC Status Codes* [on page 333](#) for the meaning of the status codes.

last_command_accepted – The HLC writes the command field of the most recently accepted path command for the corresponding path ID to this field. This field is not updated when a received command is rejected.

last_command_accepted_completion_status – The HLC writes the completion status of the **last_command_accepted** to this field when a path command completes or fails. For path commands that complete or fail instantly upon acceptance (for example, Suspend and Resume commands), this field should be ignored. See *HLC Status Codes* [on page 333](#) for the meaning of the status codes.

See Also

MMI_path_command [on page 180](#)

MMI_path_status [on page 295](#)

MMI_path_ml_faults_status

Source/Destination

HLC ➔ Host Controller

Purpose

Reports the status of all MagneMover LITE motors in the transport system. See [Table 5-3, *MagneMover LITE Motor Fault Troubleshooting*, on page 345](#) for detailed fault definitions, set and clear conditions, and user actions for these faults. This tag is a two-dimensional array of data type `udt_MMI_ml_faults` indexed by path ID and motor position within the path. Index 0 of both array dimensions is not used since both a path ID of 0 and a motor ID of 0 in the transport system are invalid and not used.

This array must be sized to the maximum path ID configured in the transport system plus 1 for the first dimension, and the maximum number of motors along the longest path plus 1 for the second dimension. For example if the maximum path ID is 10, and the path with the greatest number of motors along it has 15 motors, this array must be dimensioned to [11, 16]. The maximum number of paths and motors per path allowed in a system varies, see the *MagneMotion System Configurator User Manual*, publication [MMI-UM046](#) and *MagneMotion Node Controller Interface User Manual*, publication [MMI-UM001](#) for additional information.

The high-level controller updates the `MMI_path_ml_faults_status` tag whenever the link between HLC and host controller is re-established and on content change (faults getting set or cleared), as soon as the change occurs the tag is updated asynchronously.

NOTE: Motors from different product lines cannot be mixed in the same transport system. If this tag is being used, `MMI_path_qs_faults_status` or `MMI_path_qs_ht_faults_status` do not need to be defined.

Intermittent, temporary propulsion under-voltage and over-voltage warnings (not faults) can occur as a result of power distribution anomalies, dissipating regenerative braking energy when vehicles decelerate, or power consumption when accelerating vehicles. These conditions are not reported to the host controller. To check for their occurrence, set the HLC or NC Fault module logging level in the Node Controller Web Interface to Warning (see the *MagneMotion Node Controller Interface User Manual*, publication [MMI-UM001](#)). Log messages are displayed for the motors when the motor reports the condition. The warnings clear when the condition is removed.

Support

This status tag is supported in the latest software release for the following product lines:

- MagneMover LITE transport systems.

Tag Format

Tag Name	MMI_path_ml_faults_status
Type	udt_MMI_ml_faults [Paths+1, Motors+1]
Array	Yes
Array Dimension Limits	Minimum array size: [2, 2] Maximum array size: max Paths + 1, max Motors + 1 Array indexes correspond to path ID and motor number. Since path ID 0 is invalid, and motor 0 is not used, array elements [0, 0], [1, 0], [2, 0], and so on, are not used.

UDT Field Descriptions

The UDT described in [Table 4-64](#) shows the type of each element that comprises the MMI_path_ml_faults_status tag. Each UDT field is described in more detail following the table.

Table 4-64: UDT Fields for udt_MMI_path_ml_faults_status

Name	Data Type	Style	Range
os_scheduler	SINT	Binary	Bits 0...7 are flags
upstream_comm	SINT	Binary	Bits 0...7 are flags
downstream_comm	SINT	Binary	Bits 0...7 are flags
motor_overall	SINT	Binary	Bits 0...7 are flags
master_board_faults_a	SINT	Binary	Bits 0...7 are flags
master_board_faults_b	SINT	Binary	Bits 0...7 are flags
driver_board_1_faults	SINT	Binary	Bits 0...7 are flags
Driver board faults repeat for a total of 8 driver boards	.	.	.
	.	.	.
	.	.	.
driver_board_8_faults	SINT	Binary	Bits 0...7 are flags
Data Type Size:	16 bytes		

UDT Field Details

See [Table 5-3, MagneMover LITE Motor Fault Troubleshooting, on page 345](#) for detailed fault definitions, set and clear conditions, and user actions for these faults. For each fault status bit, when the bit is low, the fault is inactive. When the bit is high (1), the fault is active. A normally operational transport system has no fault bits active for any motor along any path

os_scheduler – Fault data for the motor's task scheduler.

Bit	Description
0	Scheduler not initialized
1	Scheduler event queue full
2...7	Reserved

upstream_comm – Fault data for the motor's upstream communications.

Bit	Description
0	Connection inoperative
1	Misc comm warning
2	Transmit buffer full
3	UART settings corrected
4...7	Reserved

downstream_comm – Fault data for the motor's downstream communications.

Bit	Description
0	Connection inoperative
1	Misc comm warning
2	Transmit buffer full
3	UART settings corrected
4...7	Reserved

motor_overall – Fault data for the motor's overall status.

Bit	Description
0	Motor not in operational mode
1	Motor in configuration mode
2	Motor in diagnostic mode
3	Movement suspended by node controller
4...6	Reserved
7	Motor not responding (reported by the node controller)

master_board_faults_a – Fault data for the motor's master board A faults.

Bit	Description
0	Unlocated vehicle fault
1	Switch position fault
2	Propulsion power not ready
3	Under-voltage fault
4	Over-voltage fault
5	Over-temperature fault
6	Over-current fault
7	Configuration fault

master_board_faults_b – Fault data for the motor's master board B faults.

Bit	Description
0...7	Reserved

driver_board_n_faults – Eight bytes of fault data (1 byte per driver board), where each bit represents a fault. Some MagneMover LITE motors do not populate every driver board fault slot in the fault data. For such motors, the unpopulated driver board fault slots in the fault data contain “0”.

Bit	Description
0	Estimated coil temperature too high
1	Under-voltage fault
2	Over-voltage fault
3	Over-temperature fault
4	Not responding
5	Reserved
6	Switch unable to complete move.
7	Switch mechanism out of position.

MMI_path_qs_faults_status

Source/Destination

HLC ➔ Host Controller

Purpose

Reports the status of all QuickStick motors in the transport system. See [Table 5-4, QuickStick 100 Motor Fault Troubleshooting, on page 353](#) and [Table 5-5, QuickStick 150 Motor Fault Troubleshooting, on page 362](#) for detailed fault definitions, set and clear conditions, and user actions for these faults. This tag is a two-dimensional array of data type `udt_MMI_qs_faults` indexed by path ID and motor position within the path. Index 0 of both array dimensions is not used since both a path ID of 0 and a motor ID of 0 in the transport system are invalid and not used.

This array must be sized to the maximum path ID configured in the transport system plus 1 for the first dimension, and the maximum number of motors along the longest path plus 1 for the second dimension. For example if the maximum path ID is 10, and the path with the greatest number of motors along it has 15 motors, this array must be dimensioned to [11, 16]. The maximum number of paths and motors per path allowed in a system varies, see the *MagneMotion System Configurator User Manual*, publication [MMI-UM046](#) and *MagneMotion Node Controller Interface User Manual*, publication [MMI-UM001](#) for additional information.

The HLC updates the `MMI_path_qs_faults_status` tag whenever the link between HLC and host controller is re-established and on content change (faults getting set or cleared), as soon as the change occurs the tag is updated asynchronously.

NOTE: Motors from different product lines cannot be mixed in the same transport system. If this tag is being used, [MMI_path_ml_faults_status](#) or [MMI_path_qs_ht_faults_status](#) do not need to be defined.

Intermittent, temporary propulsion under-voltage and over-voltage warnings (not faults) can occur as a result of power distribution anomalies, dissipating regenerative braking energy when vehicles decelerate, or power consumption when accelerating vehicles. These conditions are not reported to the host controller. To check for their occurrence, set the HLC or NC Fault module logging level in the Node Controller Web Interface to Warning (see the *MagneMotion Node Controller Interface User Manual*, publication [MMI-UM001](#)). Log messages are displayed for the motors when the motor reports the condition. The warnings clear when the condition is removed.

Support

This status tag is supported in the latest software release for the following product lines:

- QuickStick transport systems.

Tag Format

Tag Name	MMI_path_qs_faults_status
Type	udt_MMI_qs_faults [Paths+1, Motors+1]
Array	Yes
Array Dimension Limits	Minimum array size: [2, 2] Maximum array size: max Paths + 1, max Motors + 1 Array indexes correspond to HLC path ID and motor number. Since path ID 0 is invalid, and motor 0 is not used, array elements [0, 0], [1, 0], [2, 0], and so on, are not used.

UDT Field Descriptions

The UDT that is described in [Table 4-65](#) shows each element that is used to create the MMI_path_qs_faults_status array and its data type, style, and range. Each field is described in more detail following the table.

Table 4-65: UDT Fields for udt_MMI_qs_faults_status

Name	Data Type	Style	Description
motor_type	SINT	Binary	Bits 0...7 are flags
os_scheduler	SINT	Binary	Bits 0...7 are flags
upstream_comm	SINT	Binary	Bits 0...7 are flags
downstream_comm	SINT	Binary	Bits 0...7 are flags
motor_overall	SINT	Binary	Bits 0...7 are flags
block_n_faults *	SINT	Binary	Bits 0...15 are flags
master_board_faults	SINT	Binary	Bits 0...7 are flags
secondary_core_faults	SINT	Binary	Bits 0...7 are flags
drive_core_1_faults	SINT	Binary	Bits 0...7 are flags
drive_core_2_faults	SINT	Binary	Bits 0...7 are flags
ethernet_comm_faults	SINT	Binary	Bits 0...31 are flags
Data Type Size:		32 bytes	

* Block faults can repeat for a total of 10 blocks

UDT Field Details

See [Table 5-4, QuickStick 100 Motor Fault Troubleshooting, on page 353](#) and [Table 5-5, QuickStick 150 Motor Fault Troubleshooting, on page 362](#) for detailed fault definitions, set and clear conditions, and user actions for these faults. For each bit in each fault status, when the bit is low, the fault is inactive. When the bit is high (1), the fault is active. A normally operational transport system has no fault bits active for any motor along any path.

motor_type – Two bytes that identify the motor type.

Bit	Description
0	Reserved
1	QuickStick 100
2...4	Reserved
5	QuickStick 150
6...7	Reserved

os_scheduler – One byte of fault data for the task scheduler.

Bit	Description
0	Scheduler not initialized
1	Scheduler event queue full
2...7	Reserved

upstream_comm – One byte of fault data for upstream communications.

Bit	Description	
	QuickStick 100	QuickStick 150
0	Connection inoperative	Connection inoperative
1	Misc comm warning	Reserved
2	Transmit buffer full	Transmit buffer full
3	UART settings corrected	Reserved
4	Reserved	Reserved
5	Reserved	Link down
6...7	Reserved	Reserved

downstream_comm – One byte of fault data for downstream communications.

Bit	Description	
	QuickStick 100	QuickStick 150
0	Connection inoperative	Connection inoperative
1	Misc comm warning	Reserved
2	Transmit buffer full	Transmit buffer full
3	UART settings corrected	Reserved
4	Reserved	Reserved
5	Reserved	Link down
6...7	Reserved	Reserved

motor_overall – One byte of fault data for status.

Bit	Description
0	Motor not in operational mode
1	Motor in configuration mode
2	Motor in diagnostic mode
3	Movement suspended by node controller
4	Movement stopped – FastStop active
5, 6	Reserved
7	Motor not responding (reported by the node controller)

block_n_faults – Where each bit represents a fault. Some QuickStick motors do not populate every *block_n_faults* slot in the fault data. For such motors, the unpopulated *block_n_faults* slots in the fault data contains “0”. The available fault blocks are numbered 1...10.

Bit	Description	
	QuickStick 100	QuickStick 150
0	Over-current fault	Gate driver over-current fault
1	Under-voltage fault	Gate driver under-voltage fault
2	Over-voltage fault	Gate driver over-temperature fault
3	Motor stall detected	Motor stall detected
4	Slave module not configured	Reserved

Bit	Description	
	QuickStick 100	QuickStick 150
5	Over-temperature fault	Stator over-temperature fault
6	Hall Effect sensor fault	Hall Effect sensor fault
7	Slave communication fault	Hall Effect sensor not responding
8	Inverter disabled	Reserved
9	Motor under-voltage warning (only block 1) * Vehicle not located (block 2...10)	Reserved
10	Motor over-voltage warning (only block 1) * Unexpected slave module reset (block 2...10)	Gate driver not responding
11	Reserved	Reserved
12	In bootloader mode	Reserved
13	Slave module not responding	Reserved
14	Soft start not complete (only block 1) Reserved (block 2...10)	Reserved
15	Slave processor reset initiated	Reserved

* Motor under-voltage and over-voltage warnings faults are: “Displayed only in NC/HLC logs when the ‘fault’ log level is set to ‘warning’.” and “Not pushed to the Host Controller”.

master_board_faults – Represents faults on the master board.

Bit	Description	
	QuickStick 100	QuickStick 150
0	Reserved	Aux Core not responding
1	Reserved	Unexpected Aux Core reset detected
2	Reserved	Reserved
3	Reserved	Reserved
4	Reserved	Manufacturing Data CRC error
5	Reserved	Calibration Data CRC error
6...7	Reserved	Reserved

secondary_core_faults – Represents faults on the secondary core.

Bit	Description	
	QuickStick 100	QuickStick 150
0	Reserved	Aux Core not responding
1	Reserved	Unexpected Aux Core reset detected
2	Reserved	Reserved
3	Reserved	Reserved
4	Reserved	Manufacturing Data CRC error
5	Reserved	Calibration Data CRC error
6...7	Reserved	Reserved

drive_core_1_faults – Represents faults on the first drive core

Bit	Description	
	QuickStick 100	QuickStick 150
0	Reserved	Bus under-voltage fault
1	Reserved	Bus over-voltage fault
2	Reserved	Fuse open
3	Reserved	Soft start not complete
4	Reserved	Bus under-voltage warning *
5	Reserved	Bus over-voltage warning *
6...7	Reserved	Reserved

* Motor under-voltage and over-voltage warnings faults are: “Displayed only in NC/HLC logs when the 'fault' log level is set to 'warning'.” and “Not pushed to the Host Controller”.

drive_core_2_faults – Represents faults on the second drive core.

Bit	Description	
	QuickStick 100	QuickStick 150
0	Reserved	Bus under-voltage fault
1	Reserved	Bus over-voltage fault
2	Reserved	Fuse open
3	Reserved	Soft start not complete
4	Reserved	Bus under-voltage warning *
5	Reserved	Bus over-voltage warning *
6...7	Reserved	Reserved

* Motor under-voltage and over-voltage warnings faults are: “Displayed only in NC/HLC logs when the ‘fault’ log level is set to ‘warning’.” and “Not pushed to the Host Controller”.

ethernet_comm_faults – One byte of fault data for downstream communications.

Bit	Description	
	QuickStick 100	QuickStick 150
0	Reserved	Duplicate IP address detected
1	Reserved	IP address not configured
2...31	Reserved	Reserved

MMI_path_qs_ht_faults_status

Source/Destination

HLC ➔ Host Controller

Purpose

Reports the status of all QuickStick High Thrust motors in the transport system. See [Table 5-6, QSHT Gen 2 and QSHT 5700 Motor Fault Troubleshooting, on page 372](#) for detailed fault definitions, set and clear conditions, and user actions for these faults. This tag is a two-dimensional array of data type `udt_MMI_qs_ht_faults` indexed by path ID and motor position within the path. Index 0 of both array dimensions is not used since both a path ID of 0 and a motor ID of 0 in the transport system are invalid and not used.

This array must be sized to the maximum path ID configured in the transport system plus 1 for the first dimension, and the maximum number of motors along the longest path plus 1 for the second dimension. For example if the maximum path ID is 10, and the path with the greatest number of motors along it has 15 motors, this array must be dimensioned to [11, 16]. The maximum number of paths and motors per path allowed in a system varies, see the *MagneMotion System Configurator User Manual*, publication [MMI-UM046](#) and *MagneMotion Node Controller Interface User Manual*, publication [MMI-UM001](#) for additional information.

The high-level controller updates the `MMI_path_qs_ht_faults_status` tag whenever the link between HLC and host controller is re-established and on content change (that is, change in one or more fault conditions), as soon as the change occurs the tag is updated asynchronously.

NOTE: Motors from different product lines cannot be mixed in the same transport system. If this tag is being used, [MMI_path_ml_faults_status](#) or [MMI_path_qs_faults_status](#) do not need to be defined.

Intermittent, temporary propulsion under-voltage and over-voltage warnings (not faults) can occur as a result of power distribution anomalies, dissipating regenerative braking energy when vehicles decelerate, or power consumption when accelerating vehicles. These conditions are not reported to the host controller. To check for their occurrence, set the HLC or NC Fault module logging level in the Node Controller Web Interface to Warning (see the *MagneMotion Node Controller Interface User Manual*, publication [MMI-UM001](#)). Log messages are displayed for the motors when the motor reports the condition. The warnings clear when the condition is removed.

The faults status response is updated to support motors using the QSHT 5700 Motor Controller. Motors with QSHT controllers and motors with QSHT 5700 motor controller report different data for some blocks and for some fields within blocks, which is indicated. All host control software must reference the updated response.

Support

This status tag is supported in the latest software release for the following product lines:

- QuickStick HT transport systems.

Tag Format

Tag Name	MMI_path_qs_ht_faults_status
Type	udt_MMI_qs_ht_faults [Paths+1, Motors+1]
Array	Yes
Array Dimension Limits	<p>Minimum array size: [2, 2] Maximum array size: max Paths + 1, max Motors + 1 Array indexes correspond to path ID and motor number. Since path ID 0 is invalid, and motor 0 is not used, array elements [0, 0], [1, 0], [2, 0], and so on, are not used.</p>

UDT Field Descriptions

The UDT described in [Table 4-66](#) shows the type of each element that comprises the MMI_path_qs_ht_faults_status tag. Each UDT field is described in more detail following the table.

Table 4-66: UDT Fields for udt_MMI_path_qs_ht_faults_status

Name	Data Type	Style	Description
os_scheduler	SINT	Binary	Bits 0...7 are flags
upstream_comm	SINT	Binary	Bits 0...7 are flags
downstream_comm	SINT	Binary	Bits 0...7 are flags
motor_overall	SINT	Binary	Bits 0...7 are flags
master_board_faults	INT	Binary	Bits 0...15 are flags
block_1_hes_faults	INT	Binary	Bits 0...15 are flags
block_2_hes_faults	INT	Binary	Bits 0...15 are flags
block_1_inverter_faults	INT	Binary	Bits 0...15 are flags
block_2_inverter_faults	INT	Binary	Bits 0...15 are flags
ethernet_comm_faults	DINT	Binary	Bits 0...31 are flags
block_1_safety_faults	SINT	Binary	Bits 0...7 are flags
block_2_safety_faults	SINT	Binary	Bits 0...7 are flags
digital_input_status	SINT	Binary	Bits 0...7 are flags
Data Type Size:		24 bytes	

UDT Field Details

See [Table 5-6, QSHT Gen 2 and QSHT 5700 Motor Fault Troubleshooting, on page 372](#) for detailed fault definitions, set and clear conditions, and user actions for these faults. For each bit in each fault status, when the bit is low, the fault is inactive. The descriptions of the faults are provided for motors using both the QSHT controllers (QSMC) and the QSHT 5700 motor controller (QSHT 5700). When the bit is high (1), the fault is active. A normally operational transport system has no fault bits active for any motor along any path.

NOTE: Some QuickStick HT motors do not populate every block fault slot in the fault data. For those motors, the unpopulated blocks in the fault data contain 0.

The faults status response is updated to support motors using the QSHT 5700 Motor Controller. Motors with QSHT controllers and motors with QSHT 5700 motor controller report different data for some blocks and for some fields within blocks, which is indicated. All host control software must reference the updated response.

os_scheduler – Fault data for the motor's task scheduler.

Bit	Description
0	Scheduler not initialized
1	Scheduler event queue full
2...7	Reserved

upstream_comm – Fault data for the motor's upstream communications.

Bit	Description
0	Connection inoperative
1	Reserved
2	Transmit buffer full
3	UART settings corrected
4	Reserved
5	Link down
6, 7	Reserved

downstream_comm – Fault data for the motor's downstream communications.

Bit	Description
0	Connection inoperative
1	Reserved
2	Transmit buffer full
3	UART settings corrected
4	Reserved
5	Link down
6, 7	Reserved

motor_overall – Fault data for the motor's overall status.

Bit	Description
0	Motor not in operational mode
1	Motor in configuration mode
2	Motor in diagnostic mode
3	Movement suspended by node controller
4	Movement stopped – FastStop active
5, 6	Reserved
7	Motor not responding (reported by the node controller)

master_board_faults – Fault data for the motor's master board faults.

Bit	Description (QSHT)	Description (QSHT 5700)
0	Logic under-voltage fault	Logic under-voltage fault
1	Logic over-voltage fault	Logic over-voltage fault
2	Logic over-temperature fault	Logic over-temperature fault
3	Reserved	Safety Core not responding
4	Reserved	Aux Core not responding
5	Reserved	Safety Core invalid software
6...15	Reserved	Reserved

block_1_hes_faults – Fault data for Block 1 HES faults.

Bit	Description
0...4	Reserved
5	Stator over-temperature fault
6	HES board not responding
7, 8	Reserved
9	Stator mismatch
10	Reserved
11	In diagnostic mode
12	Reserved
13	In bootload mode
14, 15	Reserved

block_2_hes_faults – Fault data for Block 2 HES faults. Some QuickStick HT motors do not populate this block in the fault data. For such motors, the unpopulated blocks in the fault data contain “0”.

Bit	Description
0...4	Reserved
5	Stator over-temperature fault
6	HES board not responding
7, 8	Reserved
9	Stator mismatch
10	Reserved
11	In diagnostic mode
12	Reserved
13	In bootload mode
14, 15	Reserved

block_1_inverter_faults – Fault data for Block 1 Inverter faults.

Bit	Description (QSHT)	Description (QSHT 5700)
0	Hardware over-current fault	Hardware over-current fault
1	Under-voltage fault	Under-voltage fault
2	Over-voltage fault	Over-voltage fault
3	Motor stall detected	Motor stall detected
4	Reserved	Guard stop request status
5	Inverter over-temperature fault	Inverter over-temperature fault
6	Inverter not responding	Inverter not responding
7	Reserved	Gate driver under-voltage lockout
8	Inverter disabled by command	Inverter disabled by command
9	Soft Start switch Off	Inverter disabled by power supply not ready
10	Reserved	Fuse open
11	Hardware over-current warning	Hardware over-current warning
12	Software over-current fault	Reserved
13	In bootload mode	In bootload mode
14	Fan fault	Reserved
15	Inverter fault	Inverter fault

block_2_inverter_faults – Fault data for Block 2 Inverter faults. Some QuickStick HT motors do not populate this block in the fault data. For such motors, the unpopulated blocks in the fault data contain “0”.

Bit	Description (QSHT)	Description (QSHT 5700)
0	Hardware over-current fault	Hardware over-current fault
1	Under-voltage fault	Under-voltage fault
2	Over-voltage fault	Over-voltage fault
3	Motor stall detected	Motor stall detected
4	Reserved	Guard stop request status
5	Inverter over-temperature fault	Inverter over-temperature fault
6	Inverter not responding	Inverter not responding
7	Reserved	Gate driver under-voltage lockout

Bit	Description (QSHT)	Description (QSHT 5700)
8	Inverter disabled by command	Inverter disabled by command
9	Soft Start switch Off	Inverter disabled by power supply not ready
10	Reserved	Fuse open
11	Hardware over-current warning	Hardware over-current warning
12	Software over-current fault	Reserved
13	In bootloader mode	In bootloader mode
14	Fan fault	Reserved
15	Inverter fault	Inverter fault

ethernet_comm_faults – Fault data for Ethernet communications.

Bit	Description
0...31	Reserved

block_1_safety_faults – Fault data for Block 1 safety faults. Some QuickStick HT motors do not populate this block in the fault data. For such motors, the unpopulated blocks in the fault data contain “0”.

Bit	Description (QSHT)	Description (QSHT 5700)
0	Reserved	Safety Core Fault
1	Reserved	Safe Torque Off Fault
2	Reserved	Guard Stop Input Fault
3...7	Reserved	Reserved

block_2_safety_faults – Fault data for Block 2 safety faults. Some QuickStick HT motors do not populate this block in the fault data. For such motors, the unpopulated blocks in the fault data contain “0”.

Bit	Description (QSHT)	Description (QSHT 5700)
0	Reserved	Safety Core Fault
1	Reserved	Safe Torque Off Fault
2	Reserved	Guard Stop Input Fault
3...7	Reserved	Reserved

digital_input_status – Fault data for motor's digital input status. Some QuickStick HT motors do not populate this block in the fault data. For such motors, the unpopulated blocks in the fault data contain “0”.

Bit	Description (QSHT)	Description (QSHT 5700)
0	Reserved	Digital Input 1
1	Reserved	Digital Input 2
2	Reserved	Digital Input 3
3	Reserved	Digital Input 4
4...7	Reserved	Reserved

MMI_path_status

Source/Destination

HLC ➔ Host Controller

Purpose

Reports the status of all paths in the transport system. See *Chapter 5, Troubleshooting* for details on troubleshooting the status messages. This tag is a one-dimensional array of type udt_MMI_path_status indexed by path ID. Index 0 is not used since a path ID of 0 in the transport system is invalid and not used.

This array must be sized to the maximum path ID configured in the transport system plus 1. For example if the maximum path ID is 10, this array must be dimensioned to 11. The maximum number of paths allowed in a system varies, see the *MagneMotion System Configurator User Manual*, publication [MMI-UM046](#) and *MagneMotion Node Controller Interface User Manual*, publication [MMI-UM001](#) for additional information.

The high-level controller updates the MMI_path_status tag whenever the link between HLC and host controller is re-established and on content change (status changes), as soon as the change occurs the tag is updated asynchronously.

Support

This status tag is supported in the latest software release for the following product lines:

- MagneMover LITE transport systems.
- QuickStick transport systems.
- QuickStick HT transport systems.

Tag Format

Tag Name	MMI_path_status
Type	udt_MMI_path_status [Paths+1]
Array	Yes
Array Dimension Limits	Minimum array size: 2 Maximum array size: max Paths + 1 Array index corresponds to path ID. Since path ID 0 is invalid, array element [0] is not used.

UDT Field Descriptions

The UDT described in [Table 4-67](#) shows the type of each element that comprises the MMI_path_status tag. Each UDT field is described in more detail following the table.

Table 4-67: UDT Fields for udt_MMI_path_status

Name	Data Type	Style	Range
path_state	SINT	Decimal	0...4
path_motion_status	SINT	Binary	Bits 0...7 are flags
upstream_link_status	SINT	Decimal	0...2
downstream_link_status	SINT	Decimal	0...2
Data Type Size:		4 bytes	

UDT Field Details

path_state – The current state of the path.

Value	Description
0	Initialization
1	Startup (locating vehicles)
2	Operational
3	Reset in progress
4	Programming in progress

path_motion_status – Bit field, when all bits are Low (0), motion on the path is enabled. When any bit is High (1), motion on the path is suspended and the bits that are High describe the reason for motion suspension.

Bit	Description
0x01	Path suspended by suspend command
0x02	Path suspended by E-Stop active
0x04	Path suspended by Interlock active
0x08	Path suspended by FastStop active (QuickStick and QuickStick HT only)

upstream_link_status – The state of the communication link at the upstream end of this path.

Value	Description
0	Link OK
1	Link FAILED
2	Link Not Configured (no physical connection configured or no connection defined in the Node Controller Configuration File)

downstream_link_status – The state of the communication link at the downstream end of this path.

Value	Description
0	Link OK
1	Link FAILED
2	Link Not Configured (no physical connection configured or no connection defined in the Node Controller Configuration File)

See Also

MMI_path_command [on page 180](#)

MMI_propulsion_power_cmd_status

Source/Destination

HLC ➔ Host Controller

Purpose

Used with the QSHT 5700 Propulsion Power option to report the status of [MMI_set_prop_power_state_cmd](#) commands in the transport system. This tag is a one-dimensional array of type udt_MMI_propulsion_power_cmd_status indexed by power supply ID. Index 0 is not used since a power supply ID of 0 in the transport system is invalid.

This array must be sized to the maximum power supply ID configured in the transport system plus 1. The MMI_propulsion_power_cmd_status array is updated only in response to a [MMI_set_prop_power_state_cmd](#) explicit message sent from the host controller to the HLC.

This tag array can be used to handshake propulsion power state commands so the host controller logic can know that the HLC received a command, and either accepted it for processing, or rejected it with an appropriate error code. The tag array can be consulted to determine if a command completed and whether it completed successfully, or if an error occurred.

NOTE: The high-level controller, node controllers, and motors in the transport system must be in the operational state.

The **Enable Propulsion Power** option must be selected (checked) on the EtherNet/IP Host Controller Settings page in the Configurator Utility when creating the Node Controller Configuration File (see the *MagneMotion System Configurator User Manual*, publication [MMI-UM046](#)).

Support

This status tag is supported in the latest software release for the following product lines when **Enable Propulsion Power** is specified in the Node Controller Configuration File:

- QuickStick HT 5700 transport systems

Tag Format

Tag Name	MMI_propulsion_power_cmd_status
Type	udt_MMI_propulsion_power_cmd_status [<i>Power Supply</i> + 1]
Array	Yes
Array Dimension Limits	Minimum array size: 2 Maximum array size: max <i>Power Supply</i> + 1 Array index corresponds to HLC Propulsion Power Supply ID. Since Propulsion Power Supply ID 0 is invalid, array element [0] is not used.

UDT Field Descriptions

The UDT described in [Table 4-68](#) shows the type of each element that comprises the MMI_propulsion_power_cmd_status tag. Each UDT field is described in more detail following the table.

Table 4-68: UDT Fields for udt_MMI_propulsion_power_cmd_status

Name	Data Type	Style	Range
last_command_received_count	DINT	Hex	0x0...0xFFFFFFFF
last_command_accepted_count	DINT	Hex	0x0...0xFFFFFFFF
last_command_received	SINT	Hex	0x02
last_command_received_status	SINT	Hex	0x00, 0x0B, 0x0C, 0x0E, 0x1C, 0x41
last_command_accepted	SINT	Hex	0x02
last_command_accepted_completion_status	SINT	Hex	0x00, 0x41, 0x42, 0x80
Data Type Size:	12 bytes		

UDT Field Details

last_command_received_count – The HLC writes the **command_count** field of the most recently received propulsion power command to this field. The host controller logic can use this field to determine if the HLC received a propulsion power command.

last_command_accepted_count – The HLC writes the **command_count** field of the most recently accepted propulsion power command to this field when the HLC accepts a command. The host controller logic can use this field to determine if the HLC accepted a propulsion power command.

last_command_received – The HLC writes the command type of the most recently received propulsion power command to this field.

Value	Description
0x02	Set Prop Power State command

last_command_received_status – The HLC writes an acceptance or rejection status to this field upon receiving a new set propulsion power state command from the host controller. See *HLC Status Codes* [on page 333](#) for the meaning of the status codes.

last_command_accepted – The HLC writes the command type of the most recently accepted propulsion power command to this field. This field is not updated when a received command is rejected.

Value	Description
0x02	Set Prop Power State command

last_command_accepted_completion_status – The HLC writes the completion status of the **last_command_accepted** to this field when a propulsion power command completes or fails. See *HLC Status Codes* [on page 333](#) for the meaning of the status codes.

See Also

MMI_set_prop_power_state_cmd [on page 184](#)
MMI_propulsion_power_status [on page 301](#)

MMI_propulsion_power_status

Source/Destination

HLC ➔ Host Controller

Purpose

Used with the QSHT 5700 Propulsion Power option to report the status of all propulsion power supplies in the transport system. This tag is a one-dimensional array of type `udt_MMI_propulsion_power_status` indexed by power supply ID. Index 0 is not used since a power supply ID of 0 in the transport system is invalid. An entry is in use when the `supply_id` field in the entry is non-zero. If the `supply_id` field is zero, the entry is not in use, that is, no such power supply at the index corresponding to the supply ID exists.

This array must be sized to the maximum power supply ID configured in the transport system plus 1. When sizing this tag, make sure that the number of power supplies does not exceed the number of entries in the `MMI_propulsion_power_status` tag.

The HLC updates the `MMI_propulsion_power_status` array whenever the link between the HLC and the host controller is re-established and on content change, as soon as the change occurs the tag is updated asynchronously.

NOTE: The high-level controller, node controllers, and motors in the transport system must be in the operational state.

The **Enable Propulsion Power** option must be selected (checked) on the EtherNet/IP Host Controller Settings page in the Configurator Utility when creating the Node Controller Configuration File (see the *MagneMotion System Configurator User Manual*, publication [MMI-UM046](#)).

Support

This status tag is supported in the latest software release for the following product lines when **Enable Propulsion Power** is specified in the Node Controller Configuration File:

- QuickStick HT 5700 transport systems.

Tag Format

Tag Name	MMI_propulsion_power_status
Type	udt_MMI_propulsion_power_status [<i>Power Supply</i> +1]
Array	Yes
Array Dimension Limits	Minimum array size: 2 Maximum array size: max <i>Power Supply</i> + 1 Array index corresponds to Propulsion Power Supply ID. Since Propulsion Power Supply ID 0 is invalid, array element [0] is not used.

UDT Field Descriptions

The UDT described in [Table 4-69](#) shows the type of each element that comprises the MMI_propulsion_power_status tag. Each UDT field is described in more detail following the table.

Table 4-69: UDT Fields for udt_MMI_propulsion_power_status

Name	Data Type	Style	Range
supply_id	INT	Decimal	1...32
type	SINT	Hex	1
state	SINT	Hex	0x0...0xFF
status_change_count	DINT	Hex	0x0...0xFFFFFFFF
Data Type Size:		8 bytes	

UDT Field Details

supply_id – The ID of the propulsion power supply associated with this propulsion power status entry. This field is zero if the power supply at this index in the MMI_propulsion_power_status tag is not in use.

type – The type of the propulsion power supply.

Value	Description
1	Diode Front End (DFE) propulsion power supply

state – The state of the propulsion power supply. See [Table 4-24 on page 185](#) for propulsion power supply states.

status_change_count – A sequence count, unique to the targeted propulsion power supply, which the HLC increments whenever status for a propulsion power supply is updated.

The HLC maintains the status_change_count for each propulsion power supply. It is initialized to 0 when the HLC restarts.

See Also

MMI_set_prop_power_state_cmd on page 184

MMI_propulsion_power_cmd_status on page 298

MMI_qs_ht_sensor_map

Source/Destination

HLC ➔ Host Controller

Purpose

Used with the Sensor Mapping option, to report the sensor map data of all QuickStick HT motors in a transport system. This tag is a two-dimensional array of type `udt_MMI_qs_ht_sensor_map` indexed by Path ID and motor position in the Path. Index 0 of both array dimensions is not used since Path ID 0 in the transport system is invalid and motors on Paths always start with 1.

This array must be sized to the maximum Path ID configured in the transport system plus 1 for the first dimension and the maximum number of motors along any given Path plus 1 for the second dimension. For example, if your maximum Path ID is 10, and the Path with the greatest number of motors has 15 motors, this array must be dimensioned as [11, 16].

The HLC updates the `MMI_qs_ht_sensor_map` tag in host controller memory based on the selected sensor map push period and paths per push period XML element values as described in the *MagneMotion System Configurator User Manual*, publication [MMI-UM046](#).

NOTE: The high-level controller, node controllers, and motors in the transport system must be in the operational state.

The **Enable Sensor Mapping** option must be selected (checked) on the EtherNet/IP Host Controller Settings page in the Configurator Utility when creating the Node Controller Configuration File (see the *MagneMotion System Configurator User Manual*, publication [MMI-UM046](#)).

Support

This status tag is supported in the latest software release for the following product lines when **Enable Sensor Mapping** is specified in the Node Controller Configuration File:

- QuickStick HT transport systems.

Tag Format

Tag Name	MMI_qs_ht_sensor_map
Type	<code>udt_MMI_qs_ht_sensor_map</code> [Paths+1, Motors+1]
Array	Yes
Array Dimension Limits	<p>Minimum array size: [2, 2] Maximum array size: max Paths + 1, max Motors + 1 Array indexes correspond to HLC Path ID and Motor number Since Path ID 0 is invalid and motor 0 is not used, array elements [0, 0], [0, 1], [0, 2], and so on, and [1, 0], [2, 0], [3, 0], and so on, are not used.</p>

UDT Field Descriptions

The UDT described in [Table 4-70](#) shows the type of each element that comprises the MMI_qs_ht_sensor_map tag. Each UDT field is described in more detail following the table.

Table 4-70: UDT Fields for udt_MMI_qs_ht_sensor_map

Name	Data Type	Style	Range
number_of_blocks	SINT	Decimal	1, 2
block_1_hes_sensors	INT	Hex	0x0...0x7FFF
block_2_hes_sensors	INT	Hex	0x0...0x7FFF
block_1_vehicle_id	INT	Decimal	0...65535
block_2_vehicle_id	INT	Decimal	0...65535
Data Type Size:		12 bytes	

UDT Field Details

number_of_blocks – he number of blocks comprising the specified motor.

block_1_hes_sensors – The state of Hall Effect Sensors (HES) in the first block of the specified motor.

A bit set to “1” in the block_1_hes_sensors field indicates that the specified HES sensor has signal (detects the magnet array). A bit set to “0” in the block_1_hes_sensors field indicates that the specified HES sensor has no signal. Bits in the block_1_hes_sensors field not associated with a HES sensor are always “0”.

The least significant bit of the block_1_hes_sensors field corresponds to the most upstream HES sensor in the motor block. The most significant bit of the block_1_hes_sensors field corresponds to the furthest downstream HES sensor in the motor block.

block_2_hes_sensors – The state of Hall Effect Sensors (HES) in the second block (if present) of the specified motor.

A bit set to “1” in the block_2_hes_sensors field indicates that the specified HES sensor has signal (detects the magnet array). A bit set to “0” in the block_2_hes_sensors field indicates that the specified HES sensor has no signal. Bits in the block_2_hes_sensors field not associated with a HES sensor are always “0”.

The least significant bit of the block_2_hes_sensors field corresponds to the most upstream HES sensor in the motor block. The most significant bit of the block_2_hes_sensors field corresponds to the furthest downstream HES sensor in the motor block.

block_1_vehicle_id – Indicates block ownership. A block can be owned by a vehicle whether or not any HES sensors are reporting signal. If any part of a vehicle, whether the magnet array or the overhang occupies the block, this field is set to the ID of the vehicle that owns the block.

Value	Description
0	The block is not owned by a vehicle.
1...65535	The vehicle ID of the vehicle that owns the block.

block_2_vehicle_id – Indicates block ownership. A block can be owned by a vehicle whether or not any HES sensors are reporting signal. If any part of a vehicle, whether the magnet array or the overhang occupies the block, this field is set to the ID of the vehicle that owns the block.

Value	Description
0	The block is not owned by a vehicle.
1...65535	The vehicle ID of the vehicle that owns the block.

See Also

MMI_vehicle_order_status [on page 324](#)

Example

The transport system shown in [Figure 4-4](#) has 1 path with three 1 meter QSHT motors and two vehicles. The sensor maps for the motors reported by MMI_qs_ht_sensor_map is shown after the figure.

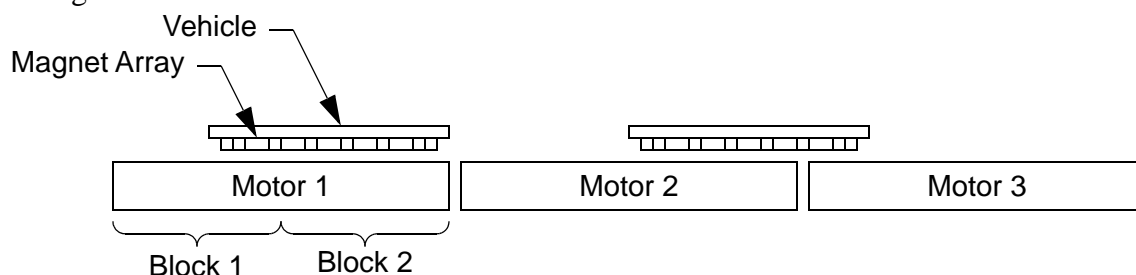


Figure 4-4: Sensor Map Example

Array Index 1,n	Motor ID			
	n=0	n=1	n=2	n=3
number_of_blocks	-	2	2	2
block_1_hes_sensors	-	0x7F00	0x0000	0x007F
block_2_hes_sensors	-	0x7FFF	0x7FFF	0x0000

MMI_sm_command_status

Source/Destination

HLC ➔ Host Controller

Purpose

Used with the System Monitoring option to report the status of subscribe/poll system monitor commands in the transport system. This tag is an array of type `udt_MMI_sm_command_status` indexed by its associated command index.

The `MMI_sm_command_status` array is updated only in response to an [MMI_sm_subscription_command](#) or an [MMI_sm_poll_command](#) explicit message sent from the host controller to the HLC.

This tag array can be used to handshake subscribe/poll commands so the host controller logic can know that the HLC received a command, and either rejected it with an appropriate error code, or accepted it for processing. The tag array can be consulted to determine if a command completed and whether it completed successfully or an error occurred.

NOTE: The high-level controller, node controllers, and motors in the transport system must be in the operational state.

The **Enable System Monitoring** option must be selected (checked) on the EtherNet/IP Host Controller Settings page in the Configurator Utility when creating the Node Controller Configuration File (see *MagneMotion System Configurator User Manual*, publication [MMI-UM046](#)).

Support

This status tag is supported in the latest software release for the following product lines when **Enable System Monitoring** is specified in the Node Controller Configuration File:

- MagneMover LITE transport systems.
- QuickStick transport systems.
- QuickStick HT transport systems.

Tag Format

Tag Name	MMI_sm_command_status
Type	udt_MMI_sm_command_status [256]
Array	Yes
Array Dimension Limits	Minimum array size: 1 Maximum array size: 256 Array index corresponds to command index.

UDT Field Descriptions

The UDT described in [Table 4-71](#) shows the type of each element that comprises the MMI_sm_command_status tag. Each UDT field is described in more detail following the table.

Table 4-71: UDT Fields for udt_MMI_sm_command_status

Name	Data Type	Style	Range
last_command_received_count	DINT	Hex	0x0...0xFFFFFFFF
last_command_accepted_count	DINT	Hex	0x0...0xFFFFFFFF
last_command_received	SINT	Hex	0x01, 0x03
last_command_received_status	SINT	Hex	0x00, 0x03, 0x09, 0x0B...0x0E, 0x10, 0x16, 0x26...0x29, 0x41, 0x42
last_command_accepted	SINT	Hex	0x01, 0x03
last_command_accepted_completion_status	SINT	Hex	0x00, 0x41, 0x42, 0x80
Data Type Size:		12 bytes	

UDT Field Details

last_command_received_count – The HLC writes the **command_count** field of the most recently received subscription/poll system monitor command to this field. The host controller logic can use this field to determine if the HLC received a subscription/poll system monitor command.

last_command_accepted_count – The HLC writes the **command_count** field of the most recently accepted subscription/poll system monitor command to this field when the HLC accepts a command. The host controller logic can use this field to determine if the HLC accepted a subscription/poll system monitor command.

last_command_received – The HLC writes the command type of the most recently received subscription/poll system monitor command to this field.

Value	Description
0x01	SM Subscription Command
0x03	SM Poll Command

last_command_received_status – The HLC writes an acceptance or rejection status to this field upon receiving a new subscription/poll system monitor command from the host controller. See *HLC Status Codes* [on page 333](#) for the meaning of the status codes.

last_command_accepted – The HLC writes the command type of the most recently accepted subscription/poll system monitor command to this field. This field is not updated when a received command is rejected.

Value	Description
0x01	SM Subscription Command
0x03	SM Poll Command

last_command_accepted_completion_status – The HLC writes the completion status of the **last_command_accepted** to this field when a subscription/poll system monitor command completes or fails. See *HLC Status Codes* [on page 333](#) for the meaning of the status codes.

See Also

MMI_sm_poll_command [on page 191](#)
MMI_sm_subscription_command [on page 197](#)
MMI_sm_metric_data [on page 309](#)

MMI_sm_metric_data

Source/Destination

HLC ➔ Host Controller

Purpose

Used with the System Monitoring option to report metric instance data of all components in a transport system. This tag is a one dimensional array of type udt_MMI_sm_metric_data indexed by the metric data response index supplied in a subscription/poll command.

The HLC updates the MMI_SM_metric_data array whenever the link between the HLC and the host controller is re-established and on content change, as soon as the change occurs the tag is updated asynchronously.

NOTE: The high-level controller, node controllers, and motors in the transport system must be in the operational state.

The **Enable System Monitoring** option must be selected (checked) on the EtherNet/IP Host Controller Settings page in the Configurator Utility when creating the Node Controller Configuration File (see *MagneMotion System Configurator User Manual*, publication [MMI-UM046](#)).

Support

This status tag is supported in the latest software release for the following product lines when **Enable System Monitoring** is specified in the Node Controller Configuration File:

- QuickStick transport systems.

Tag Format

Tag Name	MMI_sm_metric_data
Type	udt_MMI_sm_metric_data [65535]
Array	Yes
Array Dimension Limits	Minimum array size: 1 Maximum array size: 65535 Array index corresponds to subscription/poll command's data response index.

UDT Field Descriptions

The UDT described in [Table 4-72](#) shows the type of each element that comprises the MMI_sm_metric_data tag. Each UDT field is described in more detail following the table.

Table 4-72: UDT Fields for udt_MMI_sm_metric_data

Name	Data Type	Style	Range
component_type	UDT		See Embedded udt_MMI_component_type for available component types.
sequence_number	INT	Decimal	0...65535
metric_id	INT	Hex	0x0...0xFFFF
metric_instance	INT	Decimal	0...65535
metric_data_element_size	SINT	Decimal	0, 1, 2, 4, 8
metric_data_element_count	SINT	Decimal	0, 1, 4
metric_data_element_1	DINT	Hex	0x0...0xFFFFFFFF
metric_data_element_2	DINT	Hex	0x0...0xFFFFFFFF
metric_data_element_3	DINT	Hex	0x0...0xFFFFFFFF
metric_data_element_4	DINT	Hex	0x0...0xFFFFFFFF
metric_data_change_count	DINT	Hex	0x0...0xFFFFFFFF
Data Type Size:		36 bytes	

UDT Field Details

component_type – The ID of the component targeted for system monitoring (see [Table 4-81 on page 337](#)). See [Embedded udt_MMI_component_type](#) for descriptions of available component types.

sequence_number – A sequence number, unique to each subscription, incremented by the targeted component whenever new metric data is reported to the HLC. It is initialized to zero when a subscription for a metric is established, is incremented for each new metric update, and continues from zero when it rolls over. The sequence_number field is always zero for metric data updates resulting from poll commands.

metric_id – The unique ID of a metric reported to the host controller.

metric_instance – The specific metric instance reported to the host controller.

Value	Description
0	This field is always zero.
1...65535	For metrics that have multiple unique instances, this field identifies the specific metric instance reported to the host controller. (see Table 4-29).

metric_data_element_size – Size of the data elements in the metric instance data reported by the targeted component.

metric_data_element_count – Count of the data elements in the metric instance data reported by the targeted component.

metric_data_element_1 - metric_data_element_4 – The metric instance data reported by the targeted component.

The layout of metric instance data varies with the metric data element size and count. See [Table 4-73](#) for possible metric instance data layouts.

Table 4-73: Metric Instance Data Layouts

Data Element Size	Data Element Count	Metric Instance Data Layout																				
1	1	<div>31 ➡ 24 23 ➡ 16 15 ➡ 08 07 ➡ 00</div> <table><tr><td></td><td></td><td></td><td></td><td>: 1</td></tr></table>					: 1															
				: 1																		
1	4	<div>31 ➡ 24 23 ➡ 16 15 ➡ 08 07 ➡ 00</div> <table><tr><td></td><td></td><td></td><td></td><td>: 1</td></tr><tr><td></td><td></td><td></td><td></td><td>: 2</td></tr><tr><td></td><td></td><td></td><td></td><td>: 3</td></tr><tr><td></td><td></td><td></td><td></td><td>: 4</td></tr></table>					: 1					: 2					: 3					: 4
				: 1																		
				: 2																		
				: 3																		
				: 4																		
2	1	<div>31 ➡ 24 23 ➡ 16 15 ➡ 08 07 ➡ 00</div> <table><tr><td></td><td></td><td></td><td></td><td>: 1</td></tr></table>					: 1															
				: 1																		

Table 4-73: Metric Instance Data Layouts (Continued)

Data Element Size	Data Element Count	Metric Instance Data Layout																				
2	4	<div>31 ➡ 24 23 ➡ 16 15 ➡ 08 07 ➡ 00</div> <table><tr><td></td><td></td><td></td><td></td><td>: 1</td></tr><tr><td></td><td></td><td></td><td></td><td>: 2</td></tr><tr><td></td><td></td><td></td><td></td><td>: 3</td></tr><tr><td></td><td></td><td></td><td></td><td>: 4</td></tr></table>					: 1					: 2					: 3					: 4
				: 1																		
				: 2																		
				: 3																		
				: 4																		
4	1	<div>31 ➡ 24 23 ➡ 16 15 ➡ 08 07 ➡ 00</div> <table><tr><td></td><td></td><td></td><td></td><td>: 1</td></tr></table>					: 1															
				: 1																		
4	4	<div>31 ➡ 24 23 ➡ 16 15 ➡ 08 07 ➡ 00</div> <table><tr><td></td><td></td><td></td><td></td><td>: 1</td></tr><tr><td></td><td></td><td></td><td></td><td>: 2</td></tr><tr><td></td><td></td><td></td><td></td><td>: 3</td></tr><tr><td></td><td></td><td></td><td></td><td>: 4</td></tr></table>					: 1					: 2					: 3					: 4
				: 1																		
				: 2																		
				: 3																		
				: 4																		
8	1	<div><div>31 ➡ 24 23 ➡ 16 15 ➡ 08 07 ➡ 00</div><table><tr><td></td><td></td><td></td><td></td><td>: 1</td></tr></table><div><div>63 ➡ 56 55 ➡ 48 47 ➡ 40 39 ➡ 32</div><table><tr><td></td><td></td><td></td><td></td><td>: 2</td></tr></table></div></div>					: 1					: 2										
				: 1																		
				: 2																		

metric_data_change_count – A sequence count, unique to the targeted component, incremented by the HLC whenever metric data for a component is updated. The HLC maintains the metric_data_change_count for each metric data instance. It is initialized to 0 when the HLC restarts.

See Also

[MMI_sm_poll_command on page 191](#)
[MMI_sm_subscription_command on page 197](#)
[MMI_sm_command_status on page 306](#)

MMI_station_arrivals

Source/Destination

HLC ➔ Host Controller

Purpose

Reports the status of all stations in the transport system. The MMI_station_arrivals tag is an array of built-in type of INT that is indexed by station ID to show when a vehicle has arrived at a station or when a vehicle has departed from a station. When a vehicle arrives at station ID “X” for instance, MMI_station_arrivals [X] is written with the ID of the vehicle that arrived at the station. When a vehicle departs station ID “X” due to being ordered elsewhere or if the path where the station is located is reset, MMI_station_arrivals [X] is set to “0”.

NOTE: Configuration defined stations are for use only when host defined stations are not an option. It is highly advised that stations are assigned and managed in the host controller rather than defined here.

Stations must be defined in the Node Controller Configuration File. This array must be sized to the maximum station ID configured in the transport system plus 1 since a station ID of 0 is invalid and not used. The maximum number of stations allowed in a system varies, see the *MagneMotion System Configurator User Manual*, publication [MMI-UM046](#) and *MagneMotion Node Controller Interface User Manual*, publication [MMI-UM001](#) for additional information.

When a vehicle is deleted from the transport system, any station where the vehicle may have been is zeroed in MMI_station_arrivals. If all paths are issued an [MMI_path_command](#) of Reset, the entire MMI_station_arrivals tag is zeroed.

NOTE: This array can be useful for the host controller logic to key off when ordering vehicles from one station to the next when there are many more vehicles in the transport system than stations and it is too cumbersome to scan each *MMI_vehicle_order_status* entry after issuing a vehicle station command. If there is a large number of stations in the transport system with a small number of vehicles, consider using the order_numbers in the vehicle commands and the data in the *MMI_vehicle_order_status* tag to handshake commanding vehicles from station to station, or from position to position.

Support

This status tag is supported in the latest software release for the following product lines:

- MagneMover LITE transport systems.
- QuickStick transport systems.
- QuickStick HT transport systems.

Tag Format

Tag Name	MMI_station_arrivals
Type	INT [Stations+1]*
Array	Yes
Array Dimension Limits	Minimum array size: 2 Maximum array size: max Stations + 1 Array index corresponds to station ID. Since station ID 0 is invalid, array element [0] is not used.

* MMI_station_arrivals is an array of built-in type INT.

See Also

MMI_vehicle_station_order [on page 221](#)
MMI_vehicle_order_status [on page 324](#)

MMI_traffic_light_cmd_status

Source/Destination

HLC ➔ Host Controller

Purpose

Used with the Traffic Lights option to report the status of traffic light commands in the transport system. This tag is an array of type `udt_MMI_traffic_light_cmd_status` indexed by its associated command index. The `MMI_traffic_light_cmd_status` array is updated only in response to a Create Traffic Light, Set Traffic Light, or Delete Traffic Light command sent from the host controller to the HLC.

This tag array is used to handshake traffic light commands so the host controller logic can know that the HLC received a command, and either accepted it for processing, or rejected it with an appropriate error code. The tag array can be consulted to determine if a command completed and whether it completed successfully, or if an error occurred.

NOTE: The high-level controller, node controllers, and motors in the transport system must be in the operational state.

The **Enable Traffic Lights** option must be selected (checked) on the EtherNet/IP Host Controller Settings page in the Configurator Utility when creating the Node Controller Configuration File (see *MagneMotion System Configurator User Manual*, publication [MMI-UM046](#)).

Support

This status tag is supported in the latest software release for the following product lines when **Enable Traffic Lights** is specified in the Node Controller Configuration File:

- MagneMover LITE transport systems.
- QuickStick transport systems.
- QuickStick HT transport systems.

Tag Format

Tag Name	MMI_traffic_light_cmd_status
Type	udt_MMI_traffic_light_cmd_status [<i>command_index</i>]
Array	Yes
Array Dimension Limits	Minimum array size: 1 Maximum array size: 32 Array index corresponds to command index.

UDT Field Descriptions

The UDT described in Table 77 shows the type of each element that comprises the MMI_traffic_light_cmd_status tag. Each UDT field is described in more detail following the table. The MMI_traffic_light_cmd_status array is updated only in response to a Create Traffic Light, Set Traffic Light, or Delete Traffic Light command sent from the host controller to the HLC.

Table 4-74: UDT Fields for udt_MMI_traffic_light_cmd_status

Name	Data Type	Style	Range
last_command_received_count	DINT	Hex	0x0...0xFFFFFFFF
last_command_accepted_count	DINT	Hex	0x0...0xFFFFFFFF
last_command_received	SINT	Hex	0x01, 0x02, 0x04
last_command_received_status	SINT	Hex	0x00, 0x03, 0x04, 0x09, 0x0A, 0x0B, 0x0C, 0x0D, 0x0E, 0x10, 0x11, 0x14, 0x41
last_command_accepted	SINT	Hex	0x01, 0x02, 0x04
last_command_accepted_completion_status	SINT	Hex	0x00, 0x41, 0x42, 0x80
last_command_accepted_traffic_light_id	INT	Decimal	0...4096
Data Type Size:		16 bytes	

UDT Field Details

last_command_received_count – The HLC writes the **command_count** field of the most recently received traffic light command to this field. The host controller logic can use this field to determine if the HLC received a traffic light command.

last_command_accepted_count – The HLC writes the **command_count** field of the most recently accepted traffic light command to this field when the HLC accepts a command. The host controller logic can use this field to determine if the HLC accepted a traffic light command.

last_command_received – The HLC writes the command type of the most recently received traffic light command to this field.

Value	Description
0x01	Create Traffic Light command
0x02	Set Traffic Light command
0x04	Delete Traffic Light command

last_command_received_status – The HLC writes an acceptance or rejection status to this field upon receiving a new traffic light command from the host controller. See *HLC Status Codes* [on page 333](#) for the meaning of the status codes.

last_command_accepted – The HLC writes the command type of the most recently accepted traffic light command to this field. This field is not updated when a received command is rejected.

Value	Description
0x01	Create Traffic Light command
0x02	Set Traffic Light command
0x04	Delete Traffic Light command

last_command_accepted_completion_status – The HLC writes the completion status of the **last_command_accepted** to this field when a traffic light command completes or fails. See *HLC Status Codes* [on page 333](#) for the meaning of the status codes.

last_command_accepted_traffic_light_id – The HLC writes the traffic light id of the **last_command_accepted** to this field when a traffic light command completes or fails.

See Also

MMI_create_traffic_light_cmd [on page 151](#)
MMI_delete_traffic_light_cmd [on page 154](#)
MMI_set_traffic_light_cmd [on page 188](#)
MMI_traffic_light_status [on page 318](#)

MMI_traffic_light_status

Source/Destination

HLC ➔ Host Controller

Purpose

Used with the Traffic Lights option to report the status of all traffic lights in the transport system. This tag is a one-dimensional array of type `udt_MMI_traffic_light_status` indexed by Traffic Light ID. Index 0 is not used since a Traffic Light ID of 0 in the transport system is invalid. An entry is in use when the `traffic_light_id` field in the entry is nonzero. If the `traffic_light_id` field is zero, the entry is not in use, that is, no such traffic light at the index corresponding to the Traffic Light ID exists.

This array must be sized to the maximum Traffic Light ID configured in the transport system plus 1. Use the **Maximum Traffic Light ID** parameter on the EtherNet/IP Host Controller Settings page in the Configurator Utility when creating the Node Controller Configuration File to establish the size of the `MMI_traffic_light_status` tag in the host controller memory. When sizing this tag, make sure that the number of traffic lights does not exceed the number of entries in the `MMI_traffic_light_status` tag.

The HLC updates the `MMI_traffic_light_status` array whenever the link between the HLC and the host controller is re-established and on content change, as soon as the change occurs the tag is updated asynchronously.

NOTE: The high-level controller, node controllers, and the path the traffic light resides on must be in the operational state.

The **Enable Traffic Lights** option must be selected (checked) on the EtherNet/IP Host Controller Settings page in the Configurator Utility when creating the Node Controller Configuration File (see *MagneMotion System Configurator User Manual*, publication [MMI-UM046](#)).

Support

This status tag is supported in the latest software release for the following product lines when **Enable Traffic Lights** is specified in the Node Controller Configuration File:

- MagneMover LITE transport systems.
- QuickStick transport systems.
- QuickStick HT transport systems.

Tag Format

Tag Name	MMI_traffic_light_status
Type	udt_MMI_traffic_light_status [<i>Traffic Lights</i> +1]
Array	Yes
Array Dimension Limits	Minimum array size: 2 Maximum array size: max Traffic Lights + 1 Array index corresponds to Traffic Light ID. Since Traffic Light ID 0 is invalid, array element [0] is not used.

UDT Field Descriptions

The UDT described in [Table 4-75](#) shows the type of each element that comprises the MMI_traffic_light_status tag. Each UDT field is described in more detail following the table.

Table 4-75: UDT Fields for udt_MMI_traffic_light_status

Name	Data Type	Style	Range
traffic_light_id	INT	Decimal	0...4096
path_id	INT	Decimal	1...65535
position	REAL	Float	0.0...41.0 (m, floating-point)
color	SINT	Decimal	0, 1
status_change_count	DINT	Hex	0x0...0xFFFFFFFF
Data Type Size:		16 bytes	

UDT Field Details

traffic_light_id – The ID of the traffic light that is associated with this traffic light status entry. This field is zero if the traffic light at this index in the MMI_traffic_light_status tag is not in use.

path_id – The ID of the path where the traffic light is located.

position – The position (in meters), relative to the start of the specified path, where this traffic light is located (expressed as a 32-bit single-precision floating-point number).

color – The color of the specified traffic light.

Value	Description
0	Green – Allows traffic to pass.
1	Red – Stops traffic.

status_change_count – A sequence count, unique to the targeted traffic light, which the HLC increments whenever status for the traffic light is updated.

The HLC maintains the status_change_count for each traffic light. It is initialized to 0 when the HLC restarts.

See Also

MMI_create_traffic_light_cmd on page 151

MMI_delete_traffic_light_cmd on page 154

MMI_set_traffic_light_cmd on page 188

MMI_traffic_light_cmd_status on page 315

MMI_vehicle_command_status

Source/Destination

HLC ➔ Host Controller

Purpose

Used with the Vehicle Commands option, reports the status of all vehicle commands in the transport system. This tag is a one-dimensional array of type `udt_MMI_vehicle_command_status` indexed by vehicle ID. Index 0 is not used since a vehicle ID of 0 in the transport system is invalid and not used.

This array must be sized the same as the `MMI_vehicle_status` tag. The current minimum size is 257 entries. The `MMI_vehicle_command_status` tag is updated only in response to an [MMI_vehicle_command](#) message sent from the host controller to the HLC.

This tag array is used to handshake vehicle commands so the host controller logic can know that the HLC received a command. The tag can then be either accepted for processing, or rejected with an appropriate error code. The tag array can be consulted to determine if a command completed and whether it completed successfully, or if an error occurred.

NOTE: The high-level controller, node controllers, and the path the vehicle resides on must be in the operational state.

The **Enable Vehicle Commands** option must be selected (checked) on the EtherNet/IP Host Controller Settings page in the Configurator Utility when creating the Node Controller Configuration File (see *MagneMotion System Configurator User Manual*, publication [MMI-UM046](#)).

Support

This message is supported in the latest software release for the following product lines when **Enable Vehicle Commands** is specified the Node Controller Configuration File:

- MagneMover LITE transport systems.
- QuickStick transport systems.
- QuickStick HT transport systems.

Tag Format

Tag Name	MMI_vehicle_command_status
Type	udt_MMI_vehicle_command_status
Array	Yes
Array Dimension Limits	Minimum array size: 257 Maximum array size: 1281 Array index corresponds to HLC vehicle ID. Since vehicle ID 0 is invalid, array element [0] is not used.

UDT Field Descriptions

The UDT described in [Table 4-76](#) shows the type of each element that comprises the MMI_vehicle_command_status tag. Each UDT field is described in more detail following the table.

Table 4-76: UDT Fields for udt_MMI_vehicle_command_status

Name	Data Type	Style	Range
last_command_received_count	DINT	Hex	0x0...0xFFFFFFFF
last_command_accepted_count	DINT	Hex	0x0...0xFFFFFFFF
last_command_received_type	SINT	Hex	0x00
last_command_received_status	SINT	Hex	0x00, 0x01, 0x0C, 0x0D, 0x12, 0x41
last_command_accepted_type	SINT	Hex	0x00
last_command_accepted_completion_status	SINT	Hex	0x41, 0x42, 0x80
Data Type Size:		12 bytes	

UDT Field Details

last_command_received_count – The HLC writes the **command_count** field of the most recently received vehicle command for the corresponding vehicle ID to this field. The host controller logic can use this field to determine if the HLC received a vehicle command.

last_command_accepted_count – The HLC writes the **command_count** field of the most recently accepted vehicle command for the corresponding vehicle ID to this field when the HLC accepts a command. The host controller logic can use this field to determine if the HLC accepted a vehicle command.

last_command_type_received – The HLC writes the command type of the most recently received vehicle command for the corresponding vehicle ID to this field.

Value	Description
0x00	Clear vehicle suspect bit – Commands the motor controller responsible for the specified vehicle to clear the suspect bit in the vehicle record for the vehicle.

last_command_received_status – The HLC writes an acceptance or rejection status to this field upon receiving a new vehicle command for the corresponding vehicle ID from the host controller. See *HLC Status Codes* [on page 333](#) for the meaning of the status codes.

last_command_accepted_type – The HLC writes the command type of the most recently accepted vehicle command for the corresponding vehicle ID to this field. This field is not updated when a received command is rejected.

Value	Description
0x00	Clear vehicle suspect bit – Commands the motor controller responsible for the specified vehicle to clear the suspect bit in the vehicle record for the vehicle.

last_command_accepted_completion_status – The HLC writes the completion status of the **last_command_type_accepted** for the corresponding vehicle ID to this field when a vehicle command completes or fails. See *HLC Status Codes* [on page 333](#) for the meaning of the status codes.

See Also

MMI_vehicle_command [on page 205](#)

MMI_vehicle_order_status

Source/Destination

HLC ➔ Host Controller

Purpose

Reports the status of all vehicle motion commands in the transport system. This tag is a one-dimensional array of type `udt_MMI_vehicle_order_status` indexed by vehicle ID. Index 0 is not used since a vehicle ID of 0 in the transport system is invalid and not used.

This array must be sized the same as the vehicle status tag being used. When using `MMI_vehicle_status` the minimum array size is 225, see *MMI_vehicle_status* on page 328 for more details. When using `MMI_extended_vehicle_status` the minimum array size is 145, see *MMI_extended_vehicle_status* on page 234 for more details. This value is system-dependent. The maximum number of vehicles allowed in a system varies, see *MagneMotion System Configurator User Manual*, publication [MMI-UM046](#) and *MagneMotion Node Controller Interface User Manual*, publication [MMI-UM001](#) for additional information.

The high-level controller updates the `MMI_vehicle_order_status` tag only in response to an *MMI_vehicle_position_order*, an *MMI_vehicle_station_order*, an *MMI_vehicle_follow_order*, or an *MMI_vehicle_delete_order* sent from the host controller to the HLC.

This tag array can be used to handshake vehicle commands so the host controller logic can know that the HLC received a vehicle command, accepted or rejected it, and that the command completed. For transport systems with a few vehicles and many stations, or no stations at all, this tag array is the suggested method for handshaking vehicle commands from host controller to HLC. For transport systems with many vehicles and a smaller number of stations, the *MMI_station_arrivals* tag can be used as an alternative form of vehicle command handshaking.

Support

This status tag is supported in the latest software release for the following product lines:

- MagneMover LITE transport systems.
- QuickStick transport systems.
- QuickStick HT transport systems.

Tag Format

Tag Name	MMI_vehicle_order_status
Type	udt_MMI_vehicle_order_status [Vehicles + 1]
Array	Yes
Array Dimension Limits	<p>Minimum array size: (<i>MMI_vehicle_status</i> or <i>MMI_extended_vehicle_status</i>)</p> <p>Maximum array size: max Vehicles + 1</p> <p>Array index corresponds to vehicle ID.</p> <p>Since vehicle ID 0 is invalid, array element [0] is not used.</p>

UDT Field Descriptions

The UDT described in [Table 4-77](#) shows the type of each element that comprises the MMI_vehicle_order_status tag. Each UDT field is described in more detail following the table.

Table 4-77: UDT Fields for udt_MMI_vehicle_order_status

Name	Data Type	Style	Range
last_order_number_received	DINT	Hex	0x0...0xFFFFFFFF
last_order_number_accepted	DINT	Hex	0x0...0xFFFFFFFF
last_order_type_received	SINT	Hex	0xB0, 0xB1, 0xB7, 0xB9
last_order_received_status	SINT	Hex	<p>Vehicle Station Order (0xB0) 0x00, 0x01, 0x02, 0x04, 0x0A, 0x0B, 0x0C, 0x0D, 0x10, 0x13, 0x1E, 0x30, 0x41, 0x80</p> <p>Vehicle Position Order (0xB1) 0x00, 0x01, 0x03, 0x04, 0x0A, 0x0B, 0x0C, 0x0D, 0x10, 0x13, 0x1E, 0x30, 0x41, 0x80</p> <p>Vehicle Follow Order (0xB7) 0x00, 0x01, 0x03, 0x04, 0x0B, 0x0C, 0x13, 0x1D, 0x41, 0x80, 0x81, 0x82, 0x83</p> <p>Vehicle Delete Order (0xB9) 0x00, 0x01, 0x0C, 0x41</p>
last_order_type_accepted	SINT	Hex	0xB0, 0xB1, 0xB7, 0xB9
last_order_accepted_is_complete	SINT	Hex	0x00...0x02
Data Type Size:		12 bytes	

UDT Field Details

last_order_number_received – The HLC writes the **order_number** field of the most recently received vehicle command for the corresponding vehicle ID to this field. The host controller logic can use this field for a specific vehicle to determine if the HLC received a command. The host controller can then optionally retry the command after a timeout (nominally 5 seconds) if the **order_number** in the original command doesn't match this field, which indicates that the HLC never received the command.

last_order_number_accepted – The HLC writes the **order_number** field of the most recently accepted vehicle command for the corresponding vehicle ID to this field when the node controller accepts a command. The host controller logic can use this field to determine if a command received by the HLC (**last_order_number_received** = **order_number** for the most recently issued command) is accepted.

When this field matches **last_order_number_received**, the host controller knows that the HLC has accepted the command. This acceptance of the command also implies the HLC has written a 0x00 to the **last_order_received_status** field.

last_order_type_received – The HLC writes the command type of the most recently received vehicle command for the corresponding vehicle ID to this field.

Value	Description
0xB0	Vehicle station order
0xB1	Vehicle position order
0xB7	Vehicle follow order
0xB9	Vehicle delete order

last_order_received_status – The HLC writes an acceptance or rejection status to this field upon receiving a new vehicle command from the host controller. See *HLC Status Codes* on [page 333](#) for the meaning of the status codes.

last_order_type_accepted – The HLC writes the command type of the most recently accepted vehicle command for the corresponding vehicle ID to this field.

Value	Description
0xB0	Vehicle station order
0xB1	Vehicle position order
0xB7	Vehicle follow order
0xB9	Vehicle delete order

last_order_accepted_is_complete – The HLC writes the order completion status of the most recently accepted vehicle command for the corresponding vehicle ID to the lower four bits of this field as a value. The upper four bits of this field are used as flags.

Value (lower 4 bits)	Description
0x00	The last_order_number_accepted is still in progress.
0x01	The last_order_number_accepted is complete. The vehicle has completed its commanded motion or has been deleted.
0x02	The last_order_number_accepted failed.

Bit (upper 4 bits)	Description
4	Auto-decouple not complete and vehicle arrived.
5...7	Reserved

See Also

[MMI_vehicle_delete_order on page 208](#)
[MMI_vehicle_follow_order on page 211](#)
[MMI_vehicle_position_order on page 216](#)
[MMI_vehicle_station_order on page 221](#)
[MMI_extended_vehicle_status on page 234](#)
[MMI_station_arrivals on page 313](#)
[MMI_vehicle_status on page 328](#)

MMI_vehicle_status

Source/Destination

HLC ➔ Host Controller

Purpose

Reports the status of all vehicles in the transport system. This tag is a one-dimensional array of type `udt_MMI_vehicle_status` indexed by vehicle ID. Index 0 is not used since a vehicle ID of 0 in the transport system is invalid and not used. A vehicle entry is in use when the **path_id** field in the entry is nonzero. A vehicle entry is not in use when the **path_id** field is zero (that is, no vehicle exists at that index).

This array must be sized to the maximum number of vehicles in the transport system plus one (vehicle 0 is invalid). The **Max Vehicle ID** setting establishes the maximum vehicle ID that the HLC reports vehicle status for to the `MMI_vehicle_status` tag in the host controller memory. See the *MagneMotion Node Controller Interface User Manual*, publication [MMI-UM001](#) for the maximum number of vehicles per system. **Max Vehicle ID** is defined on the **EtherNet/IP Host Controller Settings** page in the Configurator Utility when creating the Node Controller Configuration File. See *MagneMotion System Configurator User Manual*, publication [MMI-UM046](#).

The HLC updates the `MMI_vehicle_status` tag in the host controller memory based on the **Vehicle Records per Status Period** and **Send Vehicle Status Period** settings. Vehicle Records per Status Period and Send Vehicle Status Period are defined on the **EtherNet/IP Host Controller Settings** page in the Configurator Utility when creating the Node Controller Configuration File.

The maximum update rate is 224 vehicle records per status period. When sizing this tag, make sure that the number of vehicle records per status period does not exceed the number of entries in the `MMI_vehicle_status` tag. The HLC round-robins, if needed, the writing of vehicle records in batches that range from 1 (minimum) to 224 (maximum) records as specified by **Vehicle Records per Status Period**.

NOTE: The high-level controller, node controllers, and paths in the transport system must be in the operational state.

When updating more than 224 vehicles, the HLC still updates at **Send Vehicle Status Period** intervals, but round-robins 224 vehicle records at a time. For instance, for 448 vehicles, vehicles 1...224 get updated, then next status period vehicles 225...448 get updated, next status period vehicles 1...224 get updated, and so on.

The **Use Extended Vehicle Status** must be cleared (not checked) on the EtherNet/IP Host Controller Settings page in the Configurator Utility when creating the Node Controller Configuration File (see the *MagneMotion System Configurator User Manual*, publication [MMI-UM046](#)).

Only one vehicle status array can be used. If [MMI_extended_vehicle_status](#) is enabled, `MMI_vehicle_status` is disabled by the node controller.

Support

This status tag is supported in the latest software release for the following product lines when **Use Extended Vehicle Status** is not specified in the Node Controller Configuration File:

- MagneMover LITE transport systems.
- QuickStick transport systems.
- QuickStick HT transport systems.

Tag Format

Tag Name	MMI_vehicle_status
Type	udt_MMI_vehicle_status [max Vehicles+1]
Array	Yes
Array Dimension Limits	Minimum array size: 2 Maximum array size: max Vehicles + 1 Array index corresponds to vehicle ID. Since vehicle ID 0 is invalid, array element [0] is not used.

UDT Field Descriptions

The UDT described in [Table 4-78](#) shows the type of each element that comprises the MMI_vehicle_status tag. Each UDT field is described in more detail following the table.

Table 4-78: UDT Fields for udt_MMI_vehicle_status

Name	Data Type	Style	Range
path_id	INT	Decimal	1...65535
dest_path_id	INT	Decimal	0...65535
position	REAL	Float	-41.0...+41.0 (m, floating point)*
commanded_position	REAL	Float	-41.0...+41.0 (m, floating point)†
velocity	REAL	Float	0.0...100.0 (m/s, floating point)*
dest_station_id	INT	Decimal	0...2048
command	SINT	Hex	0x00, 0xB0, 0xB1, 0xB7
flags	SINT	Binary	Bits 0...7 are flags
Data Type Size:		20 bytes	

* This is the value from the motor, which is converted from the fixed-point value that is used by the motor.

† This is the value that is sent by the host controller.

UDT Field Details

path_id – The ID of the path where the vehicle is located. If this field is a zero, the vehicle at this index in the MMI_vehicle_status tag is not in use (for example, no such vehicle exists). In a multi-path transport system, this field changes as the vehicle progresses from path to path.

dest_path_id – The ID of the path where the vehicle is headed. Equal to the ID of the path where the vehicle is located if motion has completed under an MMI_vehicle_follow_order with no auto-decouple configured. Equal to “0” if the vehicle has never moved or if the vehicle is under an *MMI_vehicle_follow_order*.

position – The last reported position (in meters) of the vehicle, relative to the start of the specified path (expressed as a 32-bit single-precision floating point number). Zero position is defined as the midpoint of the vehicle at the beginning of the path.

commanded_position – The destination of the vehicle.

Cmd	Command Type	Description
0x00	No command or command complete	4-byte value that is set to zero.
0xB0	Vehicle station order	
0xB1	Vehicle position order	4-byte value that is the commanded destination on the specified path (-41.0...+41.0 m) expressed as a 32-bit single-precision floating point number using little-endian format).
0xB7	Vehicle follow order	4-byte value that is the distance to the vehicle being followed (0...41.0 m) expressed as a 32-bit single-precision floating point number using little-endian format).

velocity – The last reported velocity (in m/s) of the vehicle on the specified path (expressed as a 32-bit single-precision floating point number).

dest_station_id – The ID of the station the vehicle is heading to when under a station command. This field is only valid when the **command** field is 0xB0.

command – Shows the type of command currently active for the vehicle.

Value	Description
0x00	No command is in progress.
0xB0	Move vehicle to station order.
0xB1	Move vehicle to position order.
0xB7	Vehicle follow order.

flags – Vehicle status indicators.

Bit	Description
0	<p>Vehicle Signal:</p> <p>0–The motor does not detect the vehicle.</p> <p>1–The motor currently in charge of the vehicle detects the vehicle.</p>
1	<p>Obstructed Status:</p> <p>0–The vehicle is not obstructed and able to acquire permission to move further.</p> <p>1–The vehicle is obstructed and unable to acquire permission to move further. Obstructions are due to either a vehicle in the way, a hardware fault, or motion is suspended. Vehicles in the way occur during normal operation when vehicles are in a queue or when a vehicle is in a switch, which prevents another vehicle from entering the switch.</p> <p>If after three consecutive queries of the vehicle status the obstructed bit is still set, check for an actual obstruction.</p>
2	<p>Hindered Status:</p> <p>0–The vehicle is making progress on its current move profile.</p> <p>1–The vehicle is not making progress towards the position that it has most recently been granted permission to move to. The motor continues to apply force on the vehicle to try to move it indefinitely.</p> <p>Lack of progress can happen in the following situations:</p> <ul style="list-style-type: none"> • A vehicle is held back by some external force including a foreign object on the guideway that prevents vehicle motion. • A vehicle is not in motion while under sync control. • A vehicle command uses a velocity of zero. • A vehicle command uses a PID set equal to zero. • The Control Off Position Tolerance or the Integrator Distance Threshold positions are set outside of the arrival tolerance. • The motor does not have propulsion power.

Bit	Description
3...6	<p>PID Loop Set Index:</p> <p>0—Use user-defined PID Set 0 – Unloaded PID values.</p> <p>1—Use user-defined PID Set 1 – Loaded PID values.</p> <p>2—Use user-defined PID Set 2.</p> <p>3—Use user-defined PID Set 3.</p> <p>4—Use user-defined PID Set 4.</p> <p>5—Use user-defined PID Set 5.</p> <p>6—Use user-defined PID Set 6.</p> <p>7—Use user-defined PID Set 7.</p> <p>8—Use user-defined PID Set 8.</p> <p>9—Use user-defined PID Set 9.</p> <p>10—Use user-defined PID Set 10.</p> <p>11—Use user-defined PID Set 11.</p> <p>12—Use user-defined PID Set 12.</p> <p>13—Use user-defined PID Set 13.</p> <p>14—Use user-defined PID Set 14.</p> <p>15—Use user-defined PID Set 15 – Startup PID values. This PID set is automatically used during startup. If it is not defined, PID set 0 is scaled by 25% and used for startup.</p>
7	<p>Suspect Status (MM LITE, QuickStick only):</p> <p>0—The vehicle is not suspect and is free to move.</p> <p>1—The vehicle is suspect. The motor has detected that the vehicle has been manually moved out of the control region.</p> <p>Typically, control of the vehicle cannot be regained. Even if control is regained, the vehicle continues to be suspect until an MMI_vehicle_command is issued to clear the Suspect Bit or an MMI_vehicle_delete_order is issued to delete the vehicle record.</p>

See Also

[MMI_vehicle_follow_order](#) *on page 211*
[MMI_vehicle_position_order](#) *on page 216*
[MMI_vehicle_station_order](#) *on page 221*
[MMI_extended_vehicle_status](#) *on page 234*

HLC Status Codes

When the HLC accepts, rejects, or determines completion for a command, a status code is written to an appropriate tag in the host controller memory depending on the command being accepted/rejected/completed (see the individual command descriptions). [Table 4-79](#) lists the meaning of each status code.

NOTE: Some of these status codes are only returned by specific builds of the Node Controller Software Image File.

Table 4-79: HLC Command Status Codes

Value	Status Description	Possible Cause
0x00	Command Accepted	—
0x01	Command Rejected – Invalid vehicle ID	Specified vehicle does not exist.
0x02	Command Rejected – Invalid station ID	Specified station does not exist.
0x03	Command Rejected – Invalid path ID	Specified path does not exist.
0x04	Command Rejected – Invalid position	Commanded position is off the path that is specified.
0x05	Command Rejected – E-stop signal active	E-stop is on. E-stop circuit is not powered.
0x06	Command Rejected – Interlock signal active	Interlock is on. Interlock circuit is not powered.
0x07	Command Rejected – Motion suspended, Stop active	Path is in Suspended state.
0x08	Command Rejected – Startup sequence already completed	Path has already been started.
0x09	Command Rejected – Startup sequence already started	Path is in process of starting up.
0x0A	Command Rejected – Startup sequence not initiated/not completed	Attempted to move a vehicle before startup completed.
0x0B	Command Rejected – Invalid parameter	Incorrectly formatted command (acceleration, velocity, direction, node, set signal, signal level, and so on) not within the correct range.
0x0C	Command Rejected – Initialization has not completed	Sent a command while the system is initializing.
0x0D	Command Rejected – Reset active	Sent a path command while path is resetting.

Table 4-79: HLC Command Status Codes (Continued)

Value	Status Description	Possible Cause
0x0E	Command Rejected – No record available	Attempted to access an item (for example, station or vehicle) that does not exist.
0x0F	Command Rejected – Terminus node busy	A vehicle is already entering or exiting the Terminus Node.
0x10	Command Rejected – Programming active	Sent command while programming motors.
0x11	Command Rejected – Invalid traffic light index	—
0x12	Command Rejected – Unrecognized command	Command was not formatted correctly.
0x13	Command Rejected – Vehicle lock active	Attempted to move a locked vehicle.
0x14	Command Rejected – Duplicate record	—
0x15	Command Rejected – Station in use	—
0x16	Command Rejected – Invalid motor index	Sent a command to a motor that does not exist.
0x17	Command Rejected – Motor busy	—
0x18	Command Rejected – Invalid coil board index	—
0x19	Command Rejected – Invalid node ID	Sent a command to a node that does not exist.
0x1A	Command Rejected – Invalid controller type	—
0x1B	Command Rejected – Invalid slave index	—
0x1C	Command Rejected – Resource busy	—
0x1D	Command Rejected – Vehicle in motion	Sent a command to a vehicle that requires the vehicle to be stopped.
0x1E	Command Rejected – Decouple not complete	A platoon follower has started to decouple, but auto-decouple hasn't completed.
0x20	Command Rejected – Invalid node type	Sent a command to a node, but specified the wrong node type.
0x21	Command Rejected – Invalid item type	—
0x22	Command Rejected – Invalid item index	—
0x23	Command Rejected – Invalid motor type	—
0x24	Command Rejected – FastStop active	—

Table 4-79: HLC Command Status Codes (Continued)

Value	Status Description	Possible Cause
0x25	Command Rejected – Invalid node controller ID	Sent a command to a node controller that does not exist.
0x26	Command Rejected – Invalid System Monitoring metric index	—
0x27	Command Rejected – Item in use	—
0x28	Command Rejected – No such item	—
0x29	Command Rejected – Scrubbing not complete	—
0x30	Command Rejected – Gateway entry incomplete	—
0x40	Command Failed – Unable to acquire status from motor	Communication to motor disconnected. Logic power is off.
0x41	Command Failed – Unable to complete	Vehicle commanded, but no route to destination available (consult NC/HLC logs for specifics). Path is in incorrect state.
0x42	Command Failed – Timed out	Completion response not received in time. <ul style="list-style-type: none"> • If executing a Reset command, a motor failed to reset. • If executing a vehicle subcommand, the command completed but the suspect bit did not clear.
0x43	Command Failed – Soft Start active	—
0x44	Command Failed – FastStop active	—
0x45	Command Failed – Downstream neighbor not in startup	—
0x46	Command Failed – Upstream neighbor not in startup	—
0x80	Command Status – Command completed successfully	—
0x81	Command Status – Vehicle has caught up to the vehicle in a platoon it is following	—

Table 4-79: HLC Command Status Codes (Continued)

Value	Status Description	Possible Cause
0x82	Command Status – Vehicle in a platoon is in the process of decoupling from the vehicle that it is following	Auto-decouple start
0x83	Command Status – Vehicle decoupling from a platoon has arrived at its destination before completing the decouple process	Arrival position tolerance permits the decoupling vehicle to be identified as having arrived at its destination in the following cases: <ul style="list-style-type: none">• Waiting for decoupling to start.• Waiting for decoupling to complete.

Embedded udt_MMI_component_type

Source/Destination

Host Controller ➔ HLC

Purpose

Used with the System Monitoring option to identify the component type targeted for monitoring.

NOTE: The high-level controller, node controllers, and the components that are specified must be in the operational state.

The **Enable System Monitoring** option must be selected (checked) on the EtherNet/IP Host Controller Settings page in the Configurator Utility when creating the Node Controller Configuration File (see *MagneMotion System Configurator User Manual*, publication [MMI-UM046](#)).

Support

This message is supported in the latest software release for the following product lines when **Enable System Monitoring** is specified in the Node Controller Configuration File:

- MagneMover LITE transport systems.
- QuickStick transport systems.
- QuickStick HT transport systems.

UDT Field Descriptions

The UDT that is described in [Table 4-80](#) shows each element that is used to create the MMI_component_type message and its data type, style, and range. Each field is described in more detail following the table.

Table 4-80: UDT Fields for udt_MMI_component_type

Field Name	Data Type	Style	Range
component_type	SINT	Decimal	1...7
selector_1	INT	Decimal	0...65535
selector_2	INT	Decimal	0...65535
selector_3	INT	Decimal	0...65535
Data Type Size:		8 bytes	

UDT Field Details

component_type – ID of the component targeted for system monitoring.

Table 4-81: System Monitoring Component Types

Value	Description
1	HLC Component Type*
2	Node Controller Component Type*
3	Node Component Type*
4	Path Component Type*
5	Vehicle Component Type*
6	Station Component Type*
7	Motor Component Type

* System monitoring for this component is currently not supported.

selector_1 – First selector value used to identify an instance of a component type for system monitoring.

Component Type	Selector 1 Usage	Range
1	Unused	Must be zero
2	Node Controller ID	1...128
3	Node ID	1...256
4	Path ID	1...65535
5	Vehicle ID	1...65535
6	Station ID	1...255
7	Path ID	1...65535

selector_2 – Second selector value used to identify an instance of a component type for system monitoring.

Component Type	Selector 1 Usage	Range
1...6	Unused	Must be zero
7	Motor ID	1...65535

selector_3 – Third selector value used to identify an instance of a component type for system monitoring.

Component Type	Selector 1 Usage	Range
1...6	Unused	Must be zero
7	Slave ID	Reserved (must be 0xFFFF)

See Also

MMI_sm_poll_command on page 191
MMI_sm_subscription_command on page 197
MMI_sm_command_status on page 306
MMI_sm_metric_data on page 309

Overview

This chapter describes the common difficulties that may be encountered when using the MagneMotion® Host Controller EtherNet/IP Communication Protocol. Included in this chapter are:

For assistance, see *This section describes the common vehicle faults, and general solutions.* [on page 390](#).

EtherNet/IP Communications Troubleshooting

This section describes the common difficulties that may be encountered with the Host Controller EtherNet/IP Communication Protocol, and general solutions.

Table 5-1: Initial Host Controller Communications Troubleshooting

Symptom	Problem	Solution
The host controller does not connect to the high-level controller even though it did previously.	The IP address of the HLC has been changed.	Check the IP setting through the node controller web interface or the console interface to verify the address.
	The network is not connected or there are network problems.	Verify all network connections.
	The HLC Log file contains an entry similar to: ENET_IP, CRITICAL: handle_cip_msg_router_ response: Tag: MMI_path_ ml_faults_status Type: 0x02a0 Array Index: 0,0 Operation: 1 FAILED, general status: 0x04.	MMI_path_ml_faults_status is not properly defined on the host controller. See MMI_path_ml_faults_status on page 276 .

Table 5-1: Initial Host Controller Communications Troubleshooting (Continued)

Symptom	Problem	Solution
Vehicles do not move to the expected positions.	The vehicle command message is configured incorrectly.	Verify that the distance to the positions is measured from the start of the path.
		Verify that the correct path is specified.
Vehicles do not cross a Gateway Node.	Path in the target system is suspended.	Issue a Resume command to the path.
		Wait for Interlock or E-stop on the path to be cleared.
	Duplicate vehicle ID exists in the target system.	Verify Min vehicle ID = 1 and Max vehicle ID = 65535 in the Global Settings for all Control Groups.
	The Gateway Node is disabled.	Enable the Gateway Node (see MMI_generic_node_command).
Vehicles do not cross a Terminus Node.	Handshaking is not properly implemented.	Verify entry/exit handshaking is structured correctly
Vehicles do not move.	Propulsion power is disabled.	Turn on the propulsion power.
	A physical obstruction to vehicle motion.	Verify that vehicle motion is not obstructed.
	The vehicle is binding on the guideway.	Verify that the vehicle can move freely.
	A merge or diverge node is not responding or functioning properly.	Verify that the merge or diverge node is configured correctly and the switch hardware is operational.
The host controller continues to send messages cyclically once they are triggered in the logic.	Cache Connections is selected on the Communication tab for the message.	Make sure that Cache Connections is unchecked in the message configuration (see Explicit Message Cyclic Send Glitch Workarounds on page 148). Use a one-shot (ONS) instruction in the Ladder Logic so the message only executes once when the rung with the message is active.

Motor Fault Troubleshooting

Table 5-2 describes the common motor faults and general solutions. All motor faults are identified through the path faults status tag.

- Table 5-3 provides detailed descriptions and solutions for all MagneMover® LITE motor faults as identified through the [MMI_path_ml_faults_status](#) tag.
- Table 5-4 provides detailed descriptions and solutions for all QuickStick® 100 motor faults as identified through the [MMI_path_qs_faults_status](#) tag.
- Table 5-5 provides detailed descriptions and solutions for all QuickStick 150 motor faults as identified through the [MMI_path_qs_faults_status](#) tag.
- Table 5-6 provides detailed descriptions and solutions for all QuickStick HT and QuickStick HT 5700 motor faults as identified through the [MMI_path_qs_ht_faults_status](#) tag.

Table 5-2: Initial Motor Fault Troubleshooting

Fault	Problem	Solution
The upstream connection for the motor is reported as inoperative.	NC cannot connect to the upstream end of the motor.	Check the communication connection between the motor and the previous motor or NC.
The downstream connection for the motor is reported as inoperative.	NC cannot connect to the downstream end of the motor.	Check the communication connection between the motor and the next motor or NC.
		Check the power being supplied to the motor downstream.
	No motor downstream, but the configuration indicates that there is a motor.	Correct the Node Controller Configuration File.
Soft Start not complete.	Propulsion power insufficient to activate hot-plug protection circuits.	Check the power being supplied to the motor.
Motor reports the node controller has suspended motion.	NC has issued a Suspend command to the motor.	Issue a Reset command.
		Clear the E-stop or Interlock.
Motor does not respond.	NC cannot communicate with the motor.	Check the communication connection.
		Check the power being supplied to the motor.

Table 5-2: Initial Motor Fault Troubleshooting (Continued)

Fault	Problem	Solution
Motor reports “Not in operational mode”.	All motors currently enter this state for 100 ms, and then automatically exit. This wait lets sampled A/D inputs and observers settle before using this data. This fault is informational only, there is no lockout of motor operation.	Wait 100 ms after reset or power on before sending any commands to the motor.
Unexpected Slave Module reset.	A slave board in the motor reset without being commanded to reset.	Check the power being supplied to the motor.
		Verify the motor and the vehicle are properly grounded.
Motor reports the Slave Module is not responding.	A slave board in the motor is not responding to commands or queries from the motor controller.	Verify that the motor is programmed before commanding it.
		Verify the motor and the vehicle are properly grounded.
		Return the motor for service if this fault continues.
Over-current fault.	Too much power being pulled through the motor or block.	Check all motor power wiring. Reset the motor to clear the fault. If the problem persists, contact Technical Support.
		See the following motor faults: MML™: <i>Over-current fault Gen 2</i> on page 349 . QS 100: <i>Over-current fault</i> on page 356 . QS 150: <i>Gate driver over-current fault</i> on page 365 . QSHT: <i>Hardware over-current fault</i> on page 380 and <i>Software over-current fault (QSHT + Gen 2 Motor Controller)</i> on page 384 .

Table 5-2: Initial Motor Fault Troubleshooting (Continued)

Fault	Problem	Solution
Under-voltage fault.	Not enough power being supplied to the motor.	Check the power being supplied to the motor.
		<p>See the following motor faults:</p> <p>MML: Under-voltage fault on page 349 and Under-voltage fault on page 350.</p> <p>QS 100: Under-voltage fault on page 356.</p> <p>QS 150: Gate driver under-voltage fault on page 365 and Bus under-voltage fault on page 368.</p> <p>QSHT: Under-voltage fault (QSHT + Gen 2 Motor Controller) on page 380 and Under-voltage fault (QSHT + 5700 Motor Controller) on page 380.</p>
Over-voltage fault.	Too much power (wrong voltage) being supplied to motor.	Check the power being supplied to the motor.
		<p>See the following motor faults:</p> <p>MML: Over-voltage fault on page 349 and Over-voltage fault on page 351.</p> <p>QS 100: Over-voltage fault on page 357.</p> <p>QS 150: Bus over-voltage fault on page 369</p> <p>QSHT: Over-voltage fault (QSHT + Gen 2 Motor Controller) on page 380 and Over-voltage fault (QSHT + 5700 Motor Controller) on page 381.</p>

Table 5-2: Initial Motor Fault Troubleshooting (Continued)

Fault	Problem	Solution
Over-temperature fault.	Motor controller temperature has exceeded limits.	Verify that there is nothing preventing vehicle motion on the motor.
		Verify that the temperature of the operating environment is within specifications.
		Verify that the demanded rms current on the motor does not exceed motor limitations.
		See the following motor faults: MML: <i>Over-temperature fault on page 349</i> and <i>Over-temperature fault on page 351</i> . QS 100: <i>Over-temperature fault on page 358</i> . QS 150: <i>Gate driver over-temperature fault on page 365</i> . QSHT: <i>Inverter over-temperature fault on page 382</i> .
Estimated coil temperature too high.	Coil temperature estimate that is based on current draw has exceeded limits.	Verify that there is nothing preventing vehicle motion on the motor.
		Verify that the temperature of the operating environment is within specifications.
		See the following motor faults: MML: <i>Estimated coil temperature too high on page 350</i> .

[Table 5-3](#) provides detailed fault definitions, set and clear conditions, and user actions for the MagneMover LITE motor faults reported in [MMI_path_ml_faults_status](#). The Motor Suspend Fault field shows when motion on the motor is suspended. Motion cannot resume until all faults causing a Motor Suspend are cleared. Vehicles are not allowed to enter the section of the path where the motor is located while it is suspended.

Table 5-3: MagneMover LITE Motor Fault Troubleshooting

OS Scheduler Faults

Scheduler not initialized

Definition	Scheduler didn't finish initialization.
Set condition	Scheduler was requested to schedule a nonexistent function.
Clear condition	Motor reset.
User action	<ul style="list-style-type: none"> Reset motor. Send NC/HLC log files and motor runtime version to Technical Support for evaluation.
Motor Suspend Fault	Yes

Scheduler event queue full

Definition	Software fault condition that is used for reporting when the scheduler event queue in the motor controller is full.
Set condition	The motor controller detects that the scheduler event queue is full.
Clear condition	Motor reset.
User action	<ul style="list-style-type: none"> Reset motor. Send NC/HLC log files and motor runtime version to Technical Support for evaluation.
Motor Suspend Fault	Yes

Upstream and Downstream Comm Faults

Connection inoperative

Definition	The motor controller sends periodic ping requests and expects ping responses to determine the health of the communication link.
Set condition	The motor controller did not receive a ping response after sending four consecutive ping requests.
Clear condition	The motor controller received a ping response after sending a ping request.
User action	Check communication link cabling and/or power cabling to adjacent motors.
Motor Suspend Fault	Yes

Table 5-3: MagneMover LITE Motor Fault Troubleshooting (Continued)

Misc comm warning

Definition

Set condition

Clear condition

User action

Motor Suspend Fault

This fault has been deprecated in the firmware and is no longer applicable.

Transmit buffer full

Definition

The communication transmit buffer in the motor controller is full and cannot send the current message. This message is diagnostic only, no action takes place.

Set condition

The motor controller cannot send a message because the transmit buffer is full.

Clear condition

The motor controller is able to send a message.

User action

- Check to see if a Virtual Scope data stream has been started but not stopped.
- Verify that the path length does not exceed the limit that is shown in *MagneMotion Node Controller Interface User Manual*, publication [MMI-UM001](#).

Motor Suspend Fault No

UART settings corrected

Definition

The UART settings are not correct.

NOTE: Only used for serial type motors.

Set condition

External UART settings are not correct.

Clear condition

Cleared on reset.

User action

None

Motor Suspend Fault No

Motor Overall Faults

Motor not in operational mode

Definition

Initial state of motor when it comes out of reset.

Set condition

Reset complete.

Clear condition

Cleared after 100 ms.

User action

None. Fault clears automatically.

Motor Suspend Fault Yes

Table 5-3: MagneMover LITE Motor Fault Troubleshooting (Continued)

Motor in configuration mode	
Definition	Master board is being configured.
Set condition	Node Controller sets the motor in configuration mode.
Clear condition	Node Controller clears motor from configuration mode.
User action	None
Motor Suspend Fault	Yes
Motor in diagnostic mode	
Definition	Master board is in a diagnostic mode.
Set condition	Node Controller sets the motor in diagnostic mode.
Clear condition	Node Controller clears the motor from diagnostic mode.
User action	None - Could be user driven
Motor Suspend Fault	Yes
Motion suspended by node controller	
Definition	A node controller issued a suspend command to the motor. This message can be in response to a suspend command or a digital I/O based interlock or emergency stop.
Set condition	Suspend command is sent from a node controller.
Clear condition	Resume command is sent from a node controller.
User action	Not Applicable
Motor Suspend Fault	Yes
Motor not responding	
Definition	A node controller is not able to communicate with a motor.
Set condition	No status report is received from a motor for 300 ms.
Clear condition	Status report is received from a non-responsive motor.
User action	<ul style="list-style-type: none"> Check communication connections between the node controller and the first motor in the path that is not responding. Check that power is being supplied to the first motor in the path that is not responding.
Motor Suspend Fault	Yes

Table 5-3: MagneMover LITE Motor Fault Troubleshooting (Continued)

Master Board Faults A

Unlocated vehicle fault

Definition	The motor detects that there are blocks with a magnet array present (detects field array) but no Vehicle ID.
Set condition	A block detects a magnet array with no Vehicle ID assigned.
Clear condition	All blocks that detect a magnet array have an associated Vehicle ID.
User action	Start up the path to locate the vehicle.
Motor Suspend Fault	No

Switch position fault (Gen 3)

Definition	Switch flipper is not in its expected position. NOTE: Used for MagneMover LITE Gen 3 switches only.
Set condition	No sensor active detected for 30 ms after expected arrival.
Clear condition	Once the switch automatically rehomes.
User action	None
Motor Suspend Fault	No

Propulsion power not ready

Definition	The Soft Start circuit that limits inrush current upon application of the DC bus has not turned on. If the Soft Start circuit does not turn on, the motor is in suspend mode.
Set condition	At initial power-up, until the DC Input bus is higher than +32V DC and the difference between the DC Input bus and the downstream load (through the Soft Start resistor) is less than 1V DC. Once cleared, it is not set again until the DC Input bus is lower than +27V DC. Repeatedly toggling the DC bus on/off causes the Soft Start PTC resistor to heat up and open.
Clear condition	Input DC bus is greater than +32V DC and the difference between the DC input bus and the downstream load is less than 1V DC (note hysteresis).
User action	<ul style="list-style-type: none"> Verify that propulsion power is being provided to the motor at the correct voltage. Remove power and wait 60 seconds before reapplying power (limit cycling of the propulsion power to three times or less per minute to allow the PTC to cool-down).
Motor Suspend Fault	Yes

Table 5-3: MagneMover LITE Motor Fault Troubleshooting (Continued)

Under-voltage fault	
Definition	Propulsion power to a block is lower than +27V DC.
Set condition	Propulsion power falls lower than +27V DC.
Clear condition	Propulsion power rises higher than +28V DC (note hysteresis).
User action	Verify that propulsion power is being provided to the motor at the correct voltage.
Motor Suspend Fault	Yes
Over-voltage fault	
Definition	Propulsion power to a block is higher than +42V DC.
Set condition	Propulsion power rises higher than +42V DC.
Clear condition	Propulsion power falls lower than +38V DC (note hysteresis).
User action	Verify that propulsion power is being provided to the motor at the correct voltage.
Motor Suspend Fault	Yes
Over-temperature fault	
Definition	The temperature of the motor controller for the motor has exceeded safe limits for the hardware and the motor has shut down to prevent thermal damage.
Set condition	Ambient internal temperature exceeds 80 °C.
Clear condition	Ambient internal temperature is lower than 75 °C (note hysteresis).
User action	<ul style="list-style-type: none"> • Verify that there is nothing preventing vehicle motion on the motor. • Verify that the temperature of the operating environment is within specifications. • Verify that the demanded rms current on the motor does not exceed motor limitations.
Motor Suspend Fault	Yes
Over-current fault Gen 2	
Definition	The motor controller has detected an over-current condition. NOTE: Used for MagneMover LITE Gen 2 motors only.
Set condition	Filtered input current is greater than 10 A.
Clear condition	Filtered input current is less than 9 A (note hysteresis).
User action	None
Motor Suspend Fault	Yes

Table 5-3: MagneMover LITE Motor Fault Troubleshooting (Continued)

Configuration fault

Definition	The number of blocks that are configured is less than 1 or greater than 60.
Set condition	Number of blocks is configured to be an invalid number.
Clear condition	Number of blocks is configured to be within valid limits.
User action	<ul style="list-style-type: none"> Verify that the correct MagneMover LITE Motor Type Files are loaded onto the NC. Restart services on the NC and reset the motors to load the new Node Controller Configuration Files.
Motor Suspend Fault	Yes

Master Board Faults B

Reserved

Driver Board 1...8 Faults

Estimated coil temperature too high

Definition	The motor controller has a thermal model of each coil in the motor. If the model exceeds its temperature threshold, the coil is shut off.
Set condition	The estimated coil temperature exceeds 100 °C.
Clear condition	The estimated coil temperature falls lower than 100 °C.
User action	<ul style="list-style-type: none"> This fault may be due to a stalled vehicle or mechanical binding on the track. Check to see if these conditions exist. This fault may arise from areas of the track with high duty cycles, such as start/stop areas with little dwell and high acceleration. These areas are application-dependent.
Motor Suspend Fault	No

Under-voltage fault

Definition	Propulsion power to the driver board is lower than +27V DC.
Set condition	Propulsion power falls lower than +27V DC.
Clear condition	Propulsion power rises higher than +28V DC (note hysteresis).
User action	<ul style="list-style-type: none"> Verify that propulsion power is being provided to the motor at the correct voltage. If this fault only occurs on one driver board, contact Technical Support.
Motor Suspend Fault	No

Table 5-3: MagneMover LITE Motor Fault Troubleshooting (Continued)

Over-voltage fault	
Definition	Propulsion power to the driver board is higher than +42V DC.
Set condition	Propulsion power rises higher than +42V DC.
Clear condition	Propulsion power falls lower than +38V DC (note hysteresis).
User action	<ul style="list-style-type: none"> Verify that propulsion power is being provided to the motor at the correct voltage. If this fault only occurs on one driver board, contact Technical Support.
Motor Suspend Fault	Yes
Over-temperature fault	
Definition	Coil board temperature exceeds a threshold. The coil board temperature limits are higher than the motor controller temperature limits due to proximity of the coils.
Set condition	Coil board temperature exceeds 90 °C.
Clear condition	Coil board temperature is lower than 85 °C (note hysteresis).
User action	<ul style="list-style-type: none"> Verify that there is nothing preventing vehicle motion on the motor. Verify that the temperature of the operating environment is within specifications. Verify that the demanded rms current on the motor does not exceed motor limitations.
Motor Suspend Fault	Yes
Not responding (Gen 4)	
Definition	The motor controller in the motor is unable to communicate with this driver board.
Set condition	Main board in the motor does not get a response from the driver board within 10 ms.
Clear condition	Main board in the motor receives a new update from the driver board.
User action	<ul style="list-style-type: none"> Make sure the driver board is programmed. Reset the motor. If the fault clears on reset, make sure that the motor is not receiving a static shock from the vehicle or an external source. If the fault does not clear on reset, contact Technical Support and replace the motor.
Motor Suspend Fault	Yes

Table 5-3: MagneMover LITE Motor Fault Troubleshooting (Continued)

Switch unable to complete move

Definition	A standard or high payload switch flipper is commanded to move and after some retries, it does not arrive at its position (the stepper moved but the encoder is not updating).
Set condition	Flipper stepper is moving but encoder is not responding.
Clear condition	Encoder responds.
User action	<ul style="list-style-type: none"> • Make sure propulsion power is applied so the flipper can move. • If the fault does not clear on reset, contact Technical Support and replace the motor.
Motor Suspend Fault	Yes

Switch mechanism out of position

Definition	The switch flipper on a standard or high payload switch is at its commanded position, but is outside its tolerance window.
Set condition	Flipper is moved out of its tolerance window.
Clear condition	Flipper is within tolerance window.
User action	<ul style="list-style-type: none"> • Make sure flipper is not being moved out of position externally. • Make sure propulsion power is applied so the flipper can move. • If the fault does not clear on reset, contact Technical Support and replace the motor.
Motor Suspend Fault	Yes

Table 5-4 provides detailed fault definitions, set and clear conditions, and user actions for the QuickStick 100 motor faults reported in *MMI_path_qs_faults_status*. The Motor Suspend Fault field shows when motion on the motor is suspended. Motion cannot resume until all faults causing a Motor Suspend are cleared. Vehicles are not allowed to enter the section of the path where the motor is located while it is suspended.

Table 5-4: QuickStick 100 Motor Fault Troubleshooting

OS Scheduler Faults

Scheduler not initialized

Definition	Scheduler didn't finish initialization.
Set condition	Scheduler was requested to schedule a nonexistent function.
Clear condition	Motor reset.
User action	<ul style="list-style-type: none"> Reset motor. Send NC/HLC log files and motor runtime version to Technical Support for evaluation.
Motor Suspend Fault	

Scheduler event queue full

Definition	Software fault condition that is used for reporting when the scheduler event queue in the motor controller is full.
Set condition	The motor controller detects that the scheduler event queue is full.
Clear condition	Motor reset.
User action	<ul style="list-style-type: none"> Reset motor. Send NC/HLC log files and motor runtime version to Technical Support for evaluation.
Motor Suspend Fault	Yes

Upstream and Downstream Comm Faults

Connection inoperative

Definition	The motor controller sends periodic ping requests and expects ping responses to determine the health of the communication link.
Set condition	The motor controller did not receive a ping response after sending four consecutive ping requests.
Clear condition	The motor controller received a ping response after sending a ping request.
User action	Check communication link cabling and/or power cabling to adjacent motors.
Motor Suspend Fault	Yes

Table 5-4: QuickStick 100 Motor Fault Troubleshooting (Continued)

Misc comm warning

Definition

Set condition

Clear condition

User action

Motor Suspend Fault

This fault has been deprecated in the firmware and is no longer applicable.

Transmit buffer full

Definition

The communication transmit buffer in the motor controller is full and cannot send the current message. This message is diagnostic only, no action takes place.

Set condition

The motor controller cannot send a message because the transmit buffer is full.

Clear condition

The motor controller is able to send a message.

User action

- Check to see if a Virtual Scope data stream has been started but not stopped.
- Verify that the path length does not exceed the limit that is shown in *MagneMotion Node Controller Interface User Manual*, publication [MMI-UM001](#).

Motor Suspend Fault No

UART settings corrected

Definition

The UART settings are not correct.

NOTE: Only used for serial type motors.

Set condition

External UART settings are not correct.

Clear condition

Cleared on reset.

User action

None

Motor Suspend Fault No

Motor Overall Faults

Motor not in operational mode

Definition

Initial state of motor when it comes out of reset.

Set condition

Reset complete.

Clear condition

Cleared after 100 ms.

User action

None. Fault clears automatically.

Motor Suspend Fault Yes

Table 5-4: QuickStick 100 Motor Fault Troubleshooting (Continued)

Motor in configuration mode	
Definition	Master board is being configured.
Set condition	Node Controller sets the motor in configuration mode.
Clear condition	Node Controller clears motor from configuration mode.
User action	None
Motor Suspend Fault	Yes
Motor in diagnostic mode	
Definition	Master board is in a diagnostic mode.
Set condition	Node Controller sets the motor in diagnostic mode.
Clear condition	Node Controller clears the motor from diagnostic mode.
User action	None - Could be user driven.
Motor Suspend Fault	Yes
Motion suspended by node controller	
Definition	A node controller issued a suspend command to the motor. This message can be in response to a suspend command or a digital I/O based interlock or emergency stop.
Set condition	Suspend command is sent from a node controller.
Clear condition	Resume command is sent from a node controller.
User action	Not Applicable
Motor Suspend Fault	Yes
Motion suspended – FastStop active	
Definition	Motors were sent a FastStop command, which decelerates the vehicle at the maximum thrust until velocity reaches zero. At that point, the motors hold the vehicle in place and suspend. A FastStop suspend applies to the full path. NOTE: Valid for NC Software Image version 7.2.21 and higher.
Set condition	Node controller receives a FastStop command.
Clear condition	Node controller receives a Resume command.
User action	Not Applicable
Motor Suspend Fault	Yes

Table 5-4: QuickStick 100 Motor Fault Troubleshooting (Continued)

Motor not responding

Definition	A node controller is not able to communicate with a motor.
Set condition	No status report is received from a motor for 300 ms.
Clear condition	Status report is received from a non-responsive motor.
User action	<ul style="list-style-type: none"> • Check communication connections between the node controller and the first motor in the path that is not responding. • Check that power is being supplied to the first motor in the path that is not responding.
Motor Suspend Fault	Yes

Block n Faults

Over-current fault

Definition	The motor has detected an over-current condition on the inverter for a motor block. The motor controller attempts to reset the fault three times before raising an over-current fault.
Set condition	The motor has detected an over-current fault condition on the inverter for a motor block.
Clear condition	Clear fault command.
User action	<ul style="list-style-type: none"> • If this fault occurs on all motor blocks, check the motor power wiring for intermittent connections and reset the motor to clear the fault. • If this fault only occurs on one block or the problem persists after a reset, contact Technical Support.
Motor Suspend Fault	Yes

Under-voltage fault

Definition	Propulsion power to a block is lower than +41V DC.
Set condition	Propulsion power falls lower than +41V DC.
Clear condition	Propulsion power rises higher than +43V DC (note hysteresis).
User action	<ul style="list-style-type: none"> • Verify that propulsion power is being provided to the motor at the correct voltage. • If this fault only occurs on one block, contact Technical Support.
Motor Suspend Fault	Yes

Table 5-4: QuickStick 100 Motor Fault Troubleshooting (Continued)

Over-voltage fault	
Definition	Propulsion power to a block is higher than +59V DC.
Set condition	Propulsion power rises higher than +59V DC.
Clear condition	Propulsion power falls lower than +57V DC (note hysteresis).
User action	<ul style="list-style-type: none"> Verify that propulsion power is being provided to the motor at the correct voltage. If this fault only occurs on one block, contact Technical Support.
Motor Suspend Fault	Yes
Motor stall detected	
Definition	The block has a thermal model of its inverter. If the model exceeds its thermal threshold, this fault is set. Also, fault modes that damage the current sensing hardware cause the motor to go into this mode.
Set condition	Thermal model rises higher than the thermal limit.
Clear condition	Thermal model falls lower than the lower limit (note hysteresis).
User action	<ul style="list-style-type: none"> Check for a stalled vehicle or mechanical binding in the track. Reset the motor. If the fault persists, return the motor.
Motor Suspend Fault	Yes
Slave module not configured	
Definition	The slave board has not been configured with all of its motor-specific configuration items. This fault is indicative of a slave board reset since all configuration is last downloaded when the slave board is reset.
Set condition	Set at reset and remains set until all slave board configuration items are received from the motor controller.
Clear condition	Required slave board configuration items have been received from the motor controller.
User action	Check for proper grounding of the track and vehicle.
Motor Suspend Fault	Yes

Table 5-4: QuickStick 100 Motor Fault Troubleshooting (Continued)

Over-temperature fault	
Definition	Motor controller temperature has exceeded safe limits for hardware and the motor has shut down to prevent thermal damage.
Set condition	Ambient internal temperature exceeds 95 °C.
Clear condition	Ambient internal temperature is lower than 90 °C (note hysteresis).
User action	<ul style="list-style-type: none"> • Verify that there is nothing preventing vehicle motion on the motor. • Verify that the temperature of the operating environment is within specifications. • Verify that the demanded rms current on the motor does not exceed motor limitations.
Motor Suspend Fault	Yes
Hall Effect board sensor fault	
Definition	Occurs if there is a mismatch of the signal from the corresponding HES pairs.
Set condition	Either because a Hall Effect sensor fails or because the sensor signal is not strong enough to be recognized. Signal strength is typically low when there is mechanical misalignment between the magnet array and the stator's sense board.
Clear condition	None
User action	<ul style="list-style-type: none"> • Reset the motor. • If the problem persists, replace the motor.
Motor Suspend Fault	No
Slave communication fault	
Definition	Controller board unable to communicate with driver board.
Set condition	None
Clear condition	None
User action	Reset if automatically cleared
Motor Suspend Fault	Yes

Table 5-4: QuickStick 100 Motor Fault Troubleshooting (Continued)

Inverter disabled	
Definition	Inverter is disabled due to one of the other block faults
Set condition	See the block fault section that caused the inverter to be disabled for Set condition, Clear condition, User action, and Motor Suspend fault information.
Clear condition	
User action	
Motor Suspend Fault	Yes
Motor under-voltage warning (Block 1 only)	
Definition	Motor detects the bus voltage to be close to the under-voltage threshold.
Set condition	Bus voltage at the motor drops below +42.5 VDC.
Clear condition	Bus voltage at the motor rises above +43.0 VDC.
User action	Check the propulsion power. Make sure that the propulsion power supply and the cables are sized properly to avoid excessive voltage drop.
Motor Suspend Fault	No
Vehicle not located (Block 2...10)	
Definition	Coil board has detected a vehicle but vehicle ID is unknown.
Set condition	Anomaly that occurred.
Clear condition	Issue a Startup on the path to relocate the vehicle.
User action	Perform a Startup on that path to locate the vehicle to delete any vehicle ID set to none.
Motor Suspend Fault	No
Motor over-voltage warning (Block 1 only)	
Definition	Motor detects the bus voltage to be close to the over-voltage threshold.
Set condition	Bus voltage at the motor rises above +57.0 VDC.
Clear condition	Bus voltage at the motor drops below +56.5 VDC.
User action	Check the propulsion power. Make sure that the propulsion power supply and the cables are sized properly to avoid excessive voltage drop.
Motor Suspend Fault	No

Table 5-4: QuickStick 100 Motor Fault Troubleshooting (Continued)

Unexpected slave module reset (Blocks 2...10)	
Definition	Slave module was reset.
Set condition	Slave module went through a reset state.
Clear condition	None
User action	None
Motor Suspend Fault	No
In bootload mode	
Definition	Motor in programming state.
Set condition	None
Clear condition	None
User action	Finalize programming via node controller.
Motor Suspend Fault	No
Slave module not responding	
Definition	Slave board is not responding to inquiries and commands from the motor controller.
Set condition	The motor controller detects no slave communication for 10 ms.
Clear condition	Motor controller-slave communication is restored.
User action	<ul style="list-style-type: none"> • Verify that the motor has been programmed and that motors and vehicles are properly grounded. • If this fault reoccurs, return the motor.
Motor Suspend Fault	Yes

Table 5-4: QuickStick 100 Motor Fault Troubleshooting (Continued)

Soft Start not complete (Block 1 only)

Definition	This fault appears only on block 1, which indicates the Soft Start circuit that limits inrush current upon application of the DC bus has not turned on. If the Soft Start circuit does not turn on, the motor is in suspend mode.
Set condition	<p>The DC Input bus is higher than +43V DC and the difference between the DC Input bus and the downstream load (through the Soft Start resistor) is less than 2V DC. Typically, this fault only occurs at initial power-up.</p> <p>Once cleared, it is not set again until the DC Input bus falls lower than +41V DC.</p> <p>Repeatedly toggling the DC bus on/off causes the Soft Start PTC resistor to heat up and open.</p>
Clear condition	DC Input bus is greater than +43V DC and the difference between the DC input bus and the downstream load is less than 2V DC (note hysteresis).
User action	<ul style="list-style-type: none"> • Verify that propulsion power is being provided to the motor at the correct voltage. • Remove power and wait 60 seconds before reapplying power (limit cycling of the propulsion power to three times or less per minute to allow the PTC to cool-down).
Motor Suspend Fault	Yes

Slave processor reset initiated

Definition	A slave processor is being reset due to not responding.
Set condition	Slave processor not responding for over 2 s.
Clear condition	Slave processors responds.
User action	Check for static on the track – slaves processor could be reset.
Motor Suspend Fault	Yes

[Table 5-5](#) provides detailed fault definitions, set and clear conditions, and user actions for the QuickStick 150 motor faults reported in [MMI_path_qs_faults_status](#). The Motor Suspend Fault field shows when motion on the motor is suspended. Motion cannot resume until all faults causing a Motor Suspend are cleared. Vehicles are not allowed to enter the section of the path where the motor is located while it is suspended.

Table 5-5: QuickStick 150 Motor Fault Troubleshooting

OS Scheduler Faults

Scheduler not initialized

Definition	Scheduler didn't finish initialization.
Set condition	Scheduler was requested to schedule a nonexistent function.
Clear condition	Motor reset.
User action	<ul style="list-style-type: none"> Reset motor. Send NC/HLC log files and motor runtime version to Technical Support for evaluation.
Motor Suspend Fault	Yes

Scheduler event queue full

Definition	Software fault condition that is used for reporting when the scheduler event queue in the motor controller is full.
Set condition	The motor controller detects that the scheduler event queue is full.
Clear condition	Motor reset.
User action	<ul style="list-style-type: none"> Reset motor. Send NC/HLC log files and motor runtime version to Technical Support for evaluation.
Motor Suspend Fault	Yes

Upstream and Downstream Comm Faults

Connection inoperative

Definition	The motor controller sends periodic ping requests and expects ping responses to determine the health of the communication link.
Set condition	The motor controller did not receive a ping response after sending four consecutive ping requests.
Clear condition	The motor controller received a ping response after sending a ping request.
User action	Check communication link cabling and/or power cabling to adjacent motors.
Motor Suspend Fault	Yes

Table 5-5: QuickStick 150 Motor Fault Troubleshooting (Continued)

Transmit buffer full

Definition	The communication transmit buffer in the motor controller is full and cannot send the current message. This message is diagnostic only, no action takes place.
Set condition	The motor controller cannot send a message because the transmit buffer is full.
Clear condition	The motor controller is able to send a message.
User action	<ul style="list-style-type: none"> Check to see if a Virtual Scope data stream has been started but not stopped. Verify that the path length does not exceed the limit that is shown in <i>MagneMotion Node Controller Interface User Manual</i>, publication MMI-UM001.
Motor Suspend Fault	No

Link Down

Definition	Link between the motor and the node controller is down.
Set condition	Link is not detected for 100 ms.
Clear condition	Link is operational.
User action	Verify the connection between the NC and motor.
Motor Suspend Fault	Yes (Configurable)

Motor Overall Faults

Motor not in operational mode

Definition	Initial state of motor when it comes out of reset.
Set condition	Reset complete.
Clear condition	Cleared after 100 ms.
User action	None. Fault clears automatically.
Motor Suspend Fault	Yes

Motor in configuration mode

Definition	Master board is being configured.
Set condition	Node Controller sets the motor in configuration mode.
Clear condition	Node Controller clears motor from configuration mode.
User action	None
Motor Suspend Fault	Yes

Table 5-5: QuickStick 150 Motor Fault Troubleshooting (Continued)

Motor in diagnostic mode	
Definition	Master board is in a diagnostic mode.
Set condition	Node Controller sets the motor in diagnostic mode.
Clear condition	Node Controller clears the motor from diagnostic mode.
User action	None - Could be user driven.
Motor Suspend Fault	Yes
Motion suspended by node controller	
Definition	A node controller issued a suspend command to the motor. This message can be in response to a suspend command or a digital I/O based interlock or emergency stop.
Set condition	Suspend command is sent from a node controller.
Clear condition	Resume command is sent from a node controller.
User action	Not Applicable
Motor Suspend Fault	Yes
Motion suspended – FastStop active	
Definition	<p>Motors were sent a FastStop command, which decelerates the vehicle at the maximum thrust until velocity reaches zero. At that point, the motors hold the vehicle in place and suspend. A FastStop suspend applies to the full path.</p> <p>NOTE: Valid for NC Software Image version 7.2.21 and higher.</p>
Set condition	Node controller receives a FastStop command.
Clear condition	Node controller receives a Resume command.
User action	Not Applicable
Motor Suspend Fault	Yes
Motor not responding	
Definition	A node controller is not able to communicate with a motor.
Set condition	No status report is received from a motor for 300 ms.
Clear condition	Status report is received from a non-responsive motor.
User action	<ul style="list-style-type: none"> Check communication connections between the node controller and the first motor in the path that is not responding. Check that power is being supplied to the first motor in the path that is not responding.
Motor Suspend Fault	Yes

Table 5-5: QuickStick 150 Motor Fault Troubleshooting (Continued)

Block *n* Faults

Gate driver over-current fault

Definition	The motor has detected an over-current condition on the inverter for a motor block. The motor controller attempts to reset the fault three times before raising an over-current fault.
Set condition	The motor has detected an abnormal current draw within the motor block that cannot be cleared automatically.
Clear condition	Reset of the motor commanded by the node controller or power cycle of the motor.
User action	<ul style="list-style-type: none"> • If this fault occurs on all motor blocks, check the motor power wiring for intermittent connections and reset the motor to clear the fault. • If this fault only occurs on one motor block or the problem persists after a reset, contact Technical Support.
Motor Suspend Fault	Yes

Gate driver under-voltage fault

Definition	Propulsion power to a block is lower than +41V DC.
Set condition	Propulsion power falls lower than +41V DC.
Clear condition	Propulsion power rises higher than +43V DC (note hysteresis).
User action	<ul style="list-style-type: none"> • Verify that propulsion power is being provided to the motor at the correct voltage. • If this fault only occurs on one block, contact Technical Support.
Motor Suspend Fault	Yes

Gate driver over-temperature fault

Definition	Motor controller temperature has exceeded safe limits for hardware and the motor has shut down to prevent thermal damage.
Set condition	Ambient internal temperature exceeds 95 °C.
Clear condition	Ambient internal temperature is lower than 90 °C (note hysteresis).
User action	<ul style="list-style-type: none"> • Verify that there is nothing preventing vehicle motion on the motor. • Verify that the temperature of the operating environment is within specifications.
Motor Suspend Fault	Yes

Table 5-5: QuickStick 150 Motor Fault Troubleshooting (Continued)

Motor stall detected	
Definition	The block has a thermal model of its inverter. If the model exceeds its thermal threshold, this fault is set. Also, fault modes that damage the current sensing hardware cause the motor to go into this mode.
Set condition	Thermal model rises higher than the thermal limit.
Clear condition	Thermal model falls lower than the lower limit (note hysteresis).
User action	<ul style="list-style-type: none"> • Check for a stalled vehicle or mechanical binding in the track. • Reset the motor. If the fault persists, return the motor.
Motor Suspend Fault	Yes
Stator over-temperature fault	
Definition	Motor controller temperature has exceeded safe limits for hardware and the motor has shut down to prevent thermal damage.
Set condition	Ambient internal temperature exceeds 95 °C.
Clear condition	Ambient internal temperature is lower than 90 °C (note hysteresis).
User action	<ul style="list-style-type: none"> • Verify that there is nothing preventing vehicle motion on the motor. • Verify that the temperature of the operating environment is within specifications.
Motor Suspend Fault	Yes
Hall Effect board sensor fault	
Definition	Occurs if there is a mismatch of the signal from the corresponding HES pairs.
Set condition	Either because a Hall Effect sensor fails or because the sensor signal is not strong enough to be recognized. Signal strength is typically low when there is mechanical misalignment between the magnet array and the stator's sense board.
Clear condition	None
User action	<ul style="list-style-type: none"> • Reset the motor. • If the problem persists, replace the motor.
Motor Suspend Fault	No

Table 5-5: QuickStick 150 Motor Fault Troubleshooting (Continued)

Hall Effect board sensor not responding

Definition	Occurs if there is a mismatch of the signal from the corresponding HES pairs.
Set condition	Either because a Hall Effect sensor fails or because the sensor signal is not strong enough to be recognized. Signal strength is typically low when there is mechanical misalignment between the magnet array and the stator's sense board.
Clear condition	None
User action	<ul style="list-style-type: none"> Reset the motor. If the problem persists, replace the motor.
Motor Suspend Fault	No

Gate driver not responding

Definition	Gate driver does not respond to a command from the motor.
Set condition	Loss of communication to the gate driver.
Clear condition	Re-establishment of communication to the gate driver.
User action	Reset motor.
Motor Suspend Fault	Yes

Master board and secondary core faults

Aux Core not responding

Definition	Communication to the auxiliary core has communication issues and has not received a response in 10 ms.
Set condition	The motor controller has detected a communication fault and has timed-out.
Clear condition	Aux Core communication is re-established to the motor controller.
User action	<ul style="list-style-type: none"> Power cycle the motor. If the problem persists, return the motor.
Motor Suspend Fault	Yes

Table 5-5: QuickStick 150 Motor Fault Troubleshooting (Continued)

Unexpected Aux Core reset detected

Definition	Secondary processor issued a reset without host command.
Set condition	Secondary processor reset.
Clear condition	Upon reading the secondary processor status this will get cleared.
User action	<ul style="list-style-type: none"> • Power cycle the motor. • If the problem persists, return the motor.
Motor Suspend Fault	Yes

Manufacturing Data CRC error

Definition	Manufacturing data consistency failed.
Set condition	Manufacturing data experienced a change.
Clear condition	Reset and reconfigure motor.
User action	Reset and reconfigure motor.
Motor Suspend Fault	Yes

Calibration Data CRC error

Definition	Calibration data consistency failed.
Set condition	Calibration data experienced a change.
Clear condition	None
User action	None
Motor Suspend Fault	Yes

Drive core 1 and 2 faults

Bus under-voltage fault

Definition	Propulsion power to a block is lower than +41V DC.
Set condition	Propulsion power input voltage falls lower than +41V DC.
Clear condition	Propulsion power input voltage rises higher than +43V DC (note hysteresis).
User action	<ul style="list-style-type: none"> • Verify that the propulsion power is being provided to the motor at the correct voltage. • If this fault only occurs on one block, contact Technical Support.
Motor Suspend Fault	Yes

Table 5-5: QuickStick 150 Motor Fault Troubleshooting (Continued)

Bus over-voltage fault	
Definition	For propulsion power input voltage of 48V the level is higher than +59V DC and for propulsion power input voltage of 72V the level is higher than +83V DC.
Set condition	For propulsion power input voltage of 48V the level is higher than +59V DC and for propulsion power input voltage of 72 V the level is higher than +83V DC.
Clear condition	Bus voltage at the motor drops below +59 VDC when using an input propulsion power voltage of +48 VDC and motor drops below +82 VDC when using an input propulsion power voltage of +72 VDC (note hysteresis).
User action	<ul style="list-style-type: none"> • Verify that the propulsion power is being provided to the motor at the correct voltage. • If the fault only occurs on one block, contact Technical Support.
Motor Suspend Fault	Yes
Fuse open	
Definition	The inverter board fuse is detected open.
Set condition	Fuse is blown open.
Clear condition	None.
User action	Replace the motor.
Motor Suspend Fault	Yes

Table 5-5: QuickStick 150 Motor Fault Troubleshooting (Continued)

Soft start not complete	
Definition	<ul style="list-style-type: none"> The Soft Start circuit that limits inrush current upon application of the DC bus has not turned on. If the Soft Start circuit does not turn on, the motor is in suspend mode.
Set condition	<ul style="list-style-type: none"> The DC Input bus is higher than +43V DC and the difference between the DC Input bus and the downstream load (through the Soft Start resistor) is less than 2V DC. Typically, this fault only occurs at initial power-up. Once cleared, it is not set again until the DC Input bus fall lower than +41V DC. Repeatedly toggling the DC bus on/off causes the Soft Start PTC resistor to heat up and open.
Clear condition	DC Input bus is greater than +43V DC and the difference between the DC input bus and the downstream load is less than 2V DC (note hysteresis).
User action	<ul style="list-style-type: none"> Verify that propulsion power is being provided to the motor at the correct voltage. Remove power and wait 60 seconds before reapplying power (limit cycling of the propulsion power to three times or less per minute to all the PTC to cool-down).
Motor Suspend Fault	Yes
Bus under-voltage warning[*]	
Definition	Motor detects the bus voltage to be close to the under-voltage threshold.
Set condition	Bus voltage at the motor drops below +42.5V DC.
Clear condition	Bus voltage at the motor rises above +43V DC.
User action	Check the propulsion power. Make sure that the propulsion power supply and the cables are sized properly to avoid excessive voltage drop.
Motor Suspend Fault	No

Table 5-5: QuickStick 150 Motor Fault Troubleshooting (Continued)

Bus over-voltage warning *

Definition	Motor detects the bus voltage to be close to the over-voltage threshold.
Set condition	Bus voltage at the motor rises above +57.0V DC when using an input propulsion power voltage of +48V DC and motor rises above +82.0V DC when using an input propulsion power voltage of +72V DC.
Clear condition	Bus voltage at the motor drops below +56.5V DC when using an input propulsion power voltage of +48V DC and motor drops below +81.5V DC when using an input propulsion power voltage of +72V DC (note hysteresis).
User action	Check the propulsion power. Make sure that the propulsion power supply and cables are sized properly to avoid excessive voltage drop.
Motor Suspend Fault	No

Ethernet communication faults

Duplicate IP address detected

Definition	A duplicate IP address has been detected in the system.
Set condition	A device with the same IP address has been detected.
Clear condition	System no longer detects a duplicate IP address.
User action	Correct the duplicate IP address environment to another IP address.
Motor Suspend Fault	Yes

IP address not configured

Definition	Motor missing an IP address or IP address has not been set.
Set condition	System detects an IP address is missing or not configured correctly.
Clear condition	IP address has been corrected and system can communicate with IP address.
User action	<ul style="list-style-type: none"> • Configure the IP address in question. • Verify the MICS file has the proper MAC address to each motor.
Motor Suspend Fault	Yes

* Motor under-voltage and over-voltage warnings faults are: “Displayed only in NC/HLC logs when the 'fault' log level is set to 'warning'.” and “Not pushed to the Host Controller”.

[Table 5-6](#) provides detailed fault definitions, set and clear conditions, and user actions for the QuickStick HT Gen 2 and QSHT 5700 motor faults reported in [MMI_path_qs_ht_faults_status](#). The Motor Suspend Fault field shows when motion on the motor is suspended. Motion cannot resume until all faults causing a Motor Suspend are cleared. Vehicles are not allowed to enter the section of the path where the motor is located while it is suspended.

For definitions of motors, motor controllers, or inverters, see MagneMotion Glossary of Terms, publication [MMI-RM003](#).

Table 5-6: QSHT Gen 2 and QSHT 5700 Motor Fault Troubleshooting

OS Scheduler Faults

Scheduler not initialized

Definition	Scheduler didn't finish initialization.
Set condition	Scheduler was requested to schedule a nonexistent function.
Clear condition	Motor reset.
User action	<ul style="list-style-type: none">Reset motor.Send NC/HLC log files and motor runtime version to Technical Support for evaluation.
Motor Suspend Fault	Yes

Scheduler event queue full

Definition	Software fault condition that is used for reporting when the scheduler event queue in the motor controller is full.
Set condition	The motor controller detects that the scheduler event queue is full.
Clear condition	Motor reset.
User action	<ul style="list-style-type: none">Reset motor.Send NC/HLC log files and motor runtime version to Technical Support for evaluation.
Motor Suspend Fault	Yes

Table 5-6: QSHT Gen 2 and QSHT 5700 Motor Fault Troubleshooting (Continued)

Upstream and Downstream Comm Faults

Connection inoperative

Definition	The motor controller sends periodic ping requests and expects ping responses to determine the health of the communication link.
Set condition	The motor controller did not receive a ping response after sending four consecutive ping requests.
Clear condition	The motor controller received a ping response after sending a ping request.
User action	Check communication link cabling and/or power cabling to adjacent motors.
Motor Suspend Fault	Yes

Transmit buffer full

Definition	The communication transmit buffer in the motor controller is full and cannot send the current message. This message is diagnostic only, no action takes place.
Set condition	The motor controller cannot send a message because the transmit buffer is full.
Clear condition	The motor controller is able to send a message.
User action	<ul style="list-style-type: none"> Check to see if a Virtual Scope data stream has been started but not stopped. Verify that the path length does not exceed the limit that is shown in <i>MagneMotion Node Controller Interface User Manual</i>, publication MMI-UM001.
Motor Suspend Fault	No

UART settings corrected

Definition	The UART settings are not correct. NOTE: Only used for serial type motors.
Set condition	External UART settings are not correct.
Clear condition	Cleared on reset.
User action	None
Motor Suspend Fault	No

Table 5-6: QSHT Gen 2 and QSHT 5700 Motor Fault Troubleshooting (Continued)

Link Down

Definition	Link between the motor and the upstream node controller is down.
Set condition	Link is not detected for 100 ms.
Clear condition	Link is operational.
User action	Verify the connection between NC and motor.
Motor Suspend Fault	Yes (Configurable)

Motor Overall Faults

Motor not in operational mode

Definition	Initial state of motor when it comes out of reset.
Set condition	Reset complete.
Clear condition	Cleared after 100 ms.
User action	None. Fault clears automatically.
Motor Suspend Fault	Yes

Motor not in configuration mode

Definition	Master board is being configured.
Set condition	Node controller sets the motor in configuration mode.
Clear condition	Node controller clears motor from configuration mode.
User action	None.
Motor Suspend Fault	Yes

Motor in diagnostics mode

Definition	Master board is in diagnostic mode.
Set condition	Node controller sets the motor in diagnostic mode.
Clear condition	Node controller clears the motor from diagnostic mode.
User action	None. Could be user driven.
Motor Suspend Fault	Yes

Table 5-6: QSHT Gen 2 and QSHT 5700 Motor Fault Troubleshooting (Continued)

Motion suspended by node controller	
Definition	A node controller issued a suspend command to the motor. This message can be in response to a suspend command or a digital I/O based interlock or emergency stop.
Set condition	Suspend command is sent from a node controller.
Clear condition	Resume command is sent from a node controller.
User action	Not Applicable
Motor Suspend Fault	Yes
Motion suspended – FastStop active	
Definition	<p>Motors were sent a FastStop command, which decelerates the vehicle at the maximum thrust until velocity reaches zero. At that point, the motors hold the vehicle in place and suspend. A FastStop suspend applies to the full path.</p> <p>NOTE: Valid for NC Software Image version 7.2.12 and higher.</p>
Set condition	Node controller receives a FastStop command.
Clear condition	Node controller receives a Resume command.
User action	Not Applicable
Motor Suspend Fault	Yes
Motor not responding	
Definition	A node controller is not able to communicate with a motor.
Set condition	No status report is received from a motor for 300 ms.
Clear condition	Status report is received from a non-responsive motor.
User action	<ul style="list-style-type: none"> Check communication connections between the node controller and the first motor in the path that is not responding. Check that power is being supplied to the first motor in the path that is not responding.
Motor Suspend Fault	Yes

Table 5-6: QSHT Gen 2 and QSHT 5700 Motor Fault Troubleshooting (Continued)

Master Board Faults

Logic under-voltage fault (QSHT + Gen 2 Motor Controller)

Definition	Logic power to a motor is lower than +19V DC.
Set condition	Logic power falls lower than +19V DC.
Clear condition	Logic power rises higher than +20V DC (note hysteresis).
User action	Verify that logic power is provided to the motor at the correct voltage.
Motor Suspend Fault	No

Logic under-voltage fault (QSHT + 5700 Motor Controller)

Definition	Logic power to a motor is lower than +22V DC.
Set condition	Logic power falls lower than +22V DC.
Clear condition	Logic power rises higher than +22V DC (note hysteresis).
User action	Verify that logic power is provided to the motor at the correct voltage.
Motor Suspend Fault	No

Logic over-voltage fault (QSHT + Gen 2 Motor Controller)

Definition	Logic power to a motor is higher than +42V DC.
Set condition	Logic power rises higher than +42V DC.
Clear condition	Logic power falls lower than +38V DC (note hysteresis).
User action	Verify that logic power is provided to the motor at the correct voltage.
Motor Suspend Fault	No

Logic over-voltage fault (QSHT + 5700 Motor Controller)

Definition	Logic power to a motor is higher than +26V DC.
Set condition	Logic power rises higher than +26V DC.
Clear condition	Logic power falls lower than +25V DC (note hysteresis).
User action	Verify that logic power is provided to the motor at the correct voltage.
Motor Suspend Fault	No

Table 5-6: QSHT Gen 2 and QSHT 5700 Motor Fault Troubleshooting (Continued)

Logic over-temperature fault	
Definition	The temperature of the motor controller has exceeded safe limits for the hardware and the motor has shut down to prevent thermal damage.
Set condition	Ambient internal temperature exceeds 80 °C.
Clear condition	Ambient internal temperature is lower than 75 °C (note hysteresis).
User action	<ul style="list-style-type: none"> • Verify that the fan on the motor controller enclosure is not blocked. • Verify that the temperature of the operating environment is within specifications. • Verify that the demanded rms current on the motor does not exceed motor limitations.
Motor Suspend Fault	Yes
Safety Core not responding (QSHT + 5700 Motor Controller)	
Definition	Communication to the safety core has timed-out after 100 ms. This is a major recoverable fault.
Set condition	The motor controller has detected a communication fault and has timed-out.
Clear condition	Safety Core communication is re-established to the motor controller.
User action	<ul style="list-style-type: none"> • Power cycle the inverter. • If the problem persists, return the inverter.
Motor Suspend Fault	Yes
Aux Core not responding (QSHT + 5700 Motor Controller)	
Definition	Communication to the auxiliary core has communication issues and has not received a response in 10 ms.
Set condition	The motor controller has detected a communication fault and has timed-out.
Clear condition	Aux Core Communication is re-established to the motor controller.
User action	<ul style="list-style-type: none"> • Power cycle the inverter. • If the problem persists, return the inverter.
Motor Suspend Fault	No

Table 5-6: QSHT Gen 2 and QSHT 5700 Motor Fault Troubleshooting (Continued)

Safety Core invalid software (QSHT + 5700 Motor Controller)

Definition	Safety core does not have valid software.
Set condition	Detected an invalid safety core software.
Clear condition	None
User action	<ul style="list-style-type: none"> Reset motor. If the problem persists, return the inverter.
Motor Suspend Fault	Yes

HES 1, 2 Faults

Stator over-temperature fault

Definition	Stator temperature as measured by the HES board has exceeded 90 °C.
Set condition	Stator temperature exceeds 90 °C.
Clear condition	Stator temperature falls lower than 85 °C (note hysteresis).
User action	<ul style="list-style-type: none"> Verify that there is nothing preventing vehicle motion on the motor. Check environmental conditions of stator.
Motor Suspend Fault	Yes

HES board not responding

Definition	HES board is not responding to inquiries and commands from the motor controller.
Set condition	The motor controller detects no HES communication for 10 ms.
Clear condition	Communication between the motor controller and the HES board is restored.
User action	<ul style="list-style-type: none"> Check the sense cable between the inverter and the stator. Check the stator module. If the problem persists, return the inverter or stator module.
Motor Suspend Fault	Yes

Table 5-6: QSHT Gen 2 and QSHT 5700 Motor Fault Troubleshooting (Continued)

Stator Mismatch	
Definition	Inverter detects a mismatch between detected and configured stators.
Set condition	The inverter detects an invalid stator connected to this unit.
Clear condition	None
User action	<ul style="list-style-type: none"> • Check HES stator connections to inverter. • Check the motor configuration in the configuration file. • Reset inverter.
Motor Suspend Fault	Yes
In diagnostic mode	
Definition	The HES board is in diagnostic mode.
Set condition	The motor controller detects the HES board is in diagnostic mode.
Clear condition	The motor controller detects the HES board is in operational mode.
User action	Reset the unit.
Motor Suspend Fault	Yes
In bootload mode	
Definition	The HES board is kept in bootloader mode.
Set condition	The motor controller detects the HES board is in bootloader mode.
Clear condition	The motor controller detects the HES board is in operational mode.
User action	<ul style="list-style-type: none"> • Program the HES board. • Check inverter and stator cables.
Motor Suspend Fault	Yes

Table 5-6: QSHT Gen 2 and QSHT 5700 Motor Fault Troubleshooting (Continued)

Inverter 1, 2 Faults

Hardware over-current fault

Definition	The motor has detected an over-current condition.
Set condition	Any phase current or the propulsion power input current exceeds ± 25 A.
Clear condition	Reset of motor controller.
User action	<ul style="list-style-type: none"> Reset the motor. Check propulsion power input cable. Check the drive cable between the motor controller and the stator. If the problem persists, replace the motor controller.
Motor Suspend Fault	Yes

Under-voltage fault (QSHT + Gen 2 Motor Controller)

Definition	Propulsion power is lower than +250V DC.
Set condition	Propulsion power falls lower than +250V DC.
Clear condition	Propulsion power rises higher than +270V DC (note hysteresis).
User action	Verify that propulsion power is being provided to the motor at the correct voltage.
Motor Suspend Fault	Yes

Under-voltage fault (QSHT + 5700 Motor Controller)

Definition	Propulsion power is lower than +300V DC.
Set condition	Propulsion power falls lower than +300V DC.
Clear condition	Propulsion power rises higher than +330V DC (note hysteresis).
User action	Verify that propulsion power is being provided to the motor at the correct voltage.
Motor Suspend Fault	Yes

Over-voltage fault (QSHT + Gen 2 Motor Controller)

Definition	Propulsion power is higher than +430V DC.
Set condition	Propulsion power rises higher than +430V DC.
Clear condition	Propulsion power falls lower than +420V DC (note hysteresis).
User action	Verify that propulsion power is being provided to the motor at the correct voltage.
Motor Suspend Fault	Yes

Table 5-6: QSHT Gen 2 and QSHT 5700 Motor Fault Troubleshooting (Continued)

Over-voltage fault (QSHT + 5700 Motor Controller)	
Definition	Propulsion power is higher than +830V DC.
Set condition	Propulsion power rises higher than +830V DC.
Clear condition	Propulsion power falls lower than +820V DC (note hysteresis).
User action	Verify that propulsion power is being provided to the motor at the correct voltage.
Motor Suspend Fault	Yes
Motor stall detected	
Definition	Device thermal overload – The inverter board has a thermal model. If the model exceeds its thermal threshold, this fault is set. The model allows the maximum commanded current to be present in the phases for 6 seconds (lower currents are allowed for longer times). Once tripped, the model takes approximately 6 seconds to fall lower than the lower threshold to turn the inverter back on.
Set condition	Thermal model exceeds the thermal threshold.
Clear condition	Thermal model lowers to within the acceptable range.
User action	<ul style="list-style-type: none"> • Check for a stalled vehicle or mechanical binding of the vehicle in the track. • Lower Acceleration. • Reset the motor. • If the fault persists, return the inverter.
Motor Suspend Fault	Yes
Guard stop request status (QSHT + 5700 Motor Controller)	
Definition	The inverter's gate drivers are disabled due to a STO function from the safety core.
Set condition	Hardwired STO is enabled.
Clear condition	Hardwired STO is disabled.
User action	<ul style="list-style-type: none"> • Check the Safety Core Faults. • If the problem persists after a power cycle, replace the motor controller.
Motor Suspend Fault	Yes

Table 5-6: QSHT Gen 2 and QSHT 5700 Motor Fault Troubleshooting (Continued)

Inverter over-temperature fault	
Definition	The inverter monitors the inverter heatsink/IGBT temperature (different than the Device Thermal Overload).
Set condition	The inverter heatsink/IGBT temperature is higher than 60 °C/100 °C.
Clear condition	The inverter heatsink/IGBT temperature falls lower than 55 °C/90 °C (note hysteresis).
User action	<ul style="list-style-type: none"> • Check environmental conditions of motor controller. • Check that the fan on the motor controller is not blocked or clogged.
Motor Suspend Fault	Yes
Inverter not responding	
Definition	The inverter board has not responded to the motor controller.
Set condition	Inverter board does not respond to motor controller queries within 20 ms.
Clear condition	Inverter board responds to motor controller queries.
User action	Reset the motor. If the problem persists after a reset, replace the motor controller.
Motor Suspend Fault	Yes
Gate driver under-voltage lockout (QSHT + 5700 Motor Controller)	
Definition	The inverter board detects an under-voltage level on the gate drivers.
Set condition	Gate driver voltage drops below 13 V.
Clear condition	Gate driver voltage rises above 13 V.
User action	Power cycle the motor. If the problem persists after a power cycle, replace the motor controller.
Motor Suspend Fault	Yes
Inverter disabled by command	
Definition	Operational state of this inverter.
Set condition	The inverter is disabled by a command from the host.
Clear condition	Normal operational state of this inverter as commanded by the host.
User action	None
Motor Suspend Fault	No

Table 5-6: QSHT Gen 2 and QSHT 5700 Motor Fault Troubleshooting (Continued)

Soft Start switch Off (QSHT + Gen 2 Motor Controller)	
Definition	Propulsion power not ready – The Soft Start circuit that limits inrush current upon application of the DC bus has not turned on. If the Soft Start circuit does not turn on, the motor is in suspend mode.
Set condition	The DC Input bus is higher than +270V DC and the difference between the DC Input bus and the downstream load (through the Soft Start resistor) is less than 5V DC. Typically, this fault only occurs at initial power-up. If this fault is cleared, it is not set again until the DC Input bus is lower than +250V DC. Repeatedly toggling the DC bus on/off causes the Soft Start PTC resistor to heat up and open.
Clear condition	DC Input bus is higher than +270V DC and the difference between the DC input bus and the downstream load is less than 5V DC (note hysteresis).
User action	<ul style="list-style-type: none"> Verify that propulsion power is being provided to the motor at the correct voltage. Remove power and wait 60 seconds before reapplying power (limit cycling of the propulsion power to three times or less per minute to allow the PTC to cool-down).
Motor Suspend Fault	Yes
Inverter disabled by power supply not ready (QSHT + 5700 Motor Controller)	
Definition	The inverter board has not received a state update that the power supply is in the running state.
Set condition	CIP Axis State of the Power Supply is not in the Running state.
Clear condition	CIP Axis State of the Power Supply is in the Running state.
User action	Wait for the Power Supply to close its internal relay before driving current out the inverter.
Motor Suspend Fault	Yes
Fuse Open (QSHT + 5700 Motor Controller)	
Definition	The inverter board fuse is detected open.
Set condition	Fuse is blown open.
Clear condition	Replace the motor controller.
User action	Replace the motor controller.
Motor Suspend Fault	Yes

Table 5-6: QSHT Gen 2 and QSHT 5700 Motor Fault Troubleshooting (Continued)

Hardware over-current warning	
Definition	The inverter receives an over-current shutdown.
Set condition	The motor has detected an over-current condition on the inverter for a motor block.
Clear condition	<ul style="list-style-type: none"> The inverter clears the fault and retries 10 ms after the fault has occurred. If 5 seconds pass and another fault HAS happened, the inverter latches off (issue a reset to clear the latched condition). Once the inverter latches off, a Hardware over-current fault is sent. If 5 seconds pass and another fault has NOT happened, the inverter is allowed to retry if a subsequent over-current fault occurs.
User action	<ul style="list-style-type: none"> Check that propulsion voltage is within range. Check stator cable between motor controller and stators. Reset the motor.
Motor Suspend Fault	Yes
Software over-current fault (QSHT + Gen 2 Motor Controller)	
Definition	The motor software has detected an over-current condition.
Set condition	Set if any of the three phase currents measurements are greater than ± 25 A.
Clear condition	Clear if all three-phase current measurements fall within ± 25 A. Normal operation current limit is set for 15 A.
User action	<ul style="list-style-type: none"> Check that propulsion voltage is within range. Check the drive cable between the motor controller and the stator.
Motor Suspend Fault	Yes
In bootload mode	
Definition	The inverter is kept in bootloader mode.
Set condition	The motor controller detects the inverter board is in bootloader mode.
Clear condition	The motor controller detects the inverter board is in operational mode.
User action	<ul style="list-style-type: none"> Program the inverter board. Might occur when the Node Controller resets the motor.
Motor Suspend Fault	Yes

Table 5-6: QSHT Gen 2 and QSHT 5700 Motor Fault Troubleshooting (Continued)

Fan fault (QSHT + Gen 2 Motor Controller)	
Definition	Status of the fan on the front of the motor controller.
Set condition	Set if fan status is OFF.
Clear condition	Clear if fan status is ON.
User action	Check that the fan is not blocked or clogged.
Motor Suspend Fault	Yes
Inverter fault	
Definition	Indicates that any of the inverter faults that would shut down the inverter are active.
Set condition	<p>QSHT + Gen 2 Motor Controller: Either hardware over-current, propulsion over-voltage, device thermal overload, inverter (heatsink) over-temperature, inverter disabled, or software over-current is detected.</p> <p>QSHT + 5700 Motor Controller: Either hardware over-current, propulsion over-voltage, device thermal overload, inverter disabled by Safety Core, inverter disabled, gate driver under-voltage lockout, or software over-current is detected.</p>
Clear condition	Cleared if all inverter faults are clear.
User action	See the individual fault descriptions for troubleshooting a particular fault.
Motor Suspend Fault	Yes
Ethernet Comm Faults	
Reserved	

Table 5-6: QSHT Gen 2 and QSHT 5700 Motor Fault Troubleshooting (Continued)

Block 1, 2 Safety Faults

Safety Core Fault (QSHT + 5700 Motor Controller)

Definition	The safety core has detected a non-recoverable internal fault.
Set condition	Internal non-recoverable fault.
Clear condition	Power Cycle Module.
User action	<ul style="list-style-type: none"> • Verify safety wiring and connections: <ul style="list-style-type: none"> • Wire terminations at the Safe Torque Off (STO) connector. • Cable/header not seated correctly. • +24V power. • Check state of safety inputs. • Reset fault and run proof test. • If fault persists, return the inverter.

Motor Suspend Fault Yes

Safe Torque Off Fault (QSHT + 5700 Motor Controller)

Definition	The Safe Torque Off (STO) function detected a fault.
Set condition	Internal fault detected when the STO function is requested.
Clear condition	Turn both inputs to the OFF-state for more than 1 second, then return the inputs to ON.
User action	<ul style="list-style-type: none"> • Verify safety wiring and connections: <ul style="list-style-type: none"> • Wire terminations at the Safe Torque Off (STO) connector. • Cable/header not seated correctly. • +24V power. • Check state of safety inputs. • Reset fault and run proof test. • If fault persists, return the inverter.

Motor Suspend Fault Yes

Table 5-6: QSHT Gen 2 and QSHT 5700 Motor Fault Troubleshooting (Continued)

Guard Stop Input Fault (QSHT + 5700 Motor Controller)	
Definition	Monitors the Safe Torque Off function inputs.
Set condition	Safe Torque Off mismatch is detected when safety inputs are in different state for more than 1.0 second.
Clear condition	Turn both inputs to the OFF-state for more than 1 second, then return the inputs to ON.
User action	<ul style="list-style-type: none"> • Verify safety wiring and connections: <ul style="list-style-type: none"> • Wire terminations at the Safe Torque Off (STO) connector. • Cable/header not seated correctly. • +24V power. • Check state of safety inputs. • Reset fault and run proof test. • If fault persists, return the inverter.
Motor Suspend Fault	Yes
Digital Input Status (QSHT + 5700 Motor Controller)	
IN 1...IN 4	
Definition	Digital Input Status of the input pin. Action is user-configurable.
Set condition	Input is held HIGH (24V).
Clear condition	Input is held low (0V) or not connected.
User action	None
Motor Suspend Fault	No

Node Fault Troubleshooting

This section describes the common node faults, and general solutions.

Table 5-7: Initial Node Troubleshooting

Fault	Problem	Solution
Node reports a Device Status of Faulted.	Detected abnormal operation of the device that is associated with the node.	Troubleshoot the node hardware.

Node Controller Fault Troubleshooting

This section describes the common node controller and high-level controller faults, and general solutions.

Table 5-8: Initial Node Controller Troubleshooting

Fault	Problem	Solution
HLC or NC is “stuck” in the Initialization state.	The Node Controller Configuration File is invalid.	See the node controller log file for identification of the specific fault.
	Unable to collect motor information.	
HLC reports a Degraded status.	Network communication issues.	See the node controller log file for identification of the specific fault.
NC is reported as Disconnected .	Network communication issues.	Verify all network connections.
	Power issues.	Verify all power connections.
HLC reports PLC communication status as Not Configured .	The host controller is not correctly defined in the Node Controller Configuration File.	Verify that the host controller IP address is correctly defined.
HLC reports EtherNet/IP communication status as Link Down .	Network communication issues.	Verify all network connections.
		Verify all network configuration.
		Verify that multiple host controllers are not using the same IP address.

Path Fault Troubleshooting

This section describes the common path faults, and general solutions.

Table 5-9: Initial Path Troubleshooting

Fault	Problem	Solution
Upstream link/connection failed.	NC cannot connect to the upstream end of the motor at the node.	Check the communication connection between the motor and the NC.
		Check the power being supplied to the motor.
Downstream link/connection failed.	NC cannot connect to the downstream end of the motor at the node.	Check the communication connection between the motor and the NC.
		Check the power being supplied to the motor.

Vehicle Fault Troubleshooting

This section describes the common vehicle faults, and general solutions.

Table 5-10: Initial Vehicle Troubleshooting

Fault	Problem	Solution
Vehicle Signal for the vehicle is reported as low “0”.	Vehicle is not being detected due to being removed from system.	Issue a Delete Vehicle command and Restart the path.
	Vehicle is not being detected due to being moved from its expected position.	
Vehicle Hindered Status is being reported as high “1”.	Mechanical blocking of the vehicle.	Clear any obstruction from the motor.
	A vehicle is not in motion while under sync control.	Status clears once the vehicle moves.
	A vehicle command uses a velocity of zero.	Reissue the vehicle motion command with a positive velocity.
	A vehicle command uses a PID set equal to zero.	Reissue the vehicle motion command using a different PID set.
	The Control Off Position Tolerance or the Integrator Distance Threshold positions are set outside of the arrival tolerance. This causes the vehicle to appear to have arrived even though it is not in the expected location.	Redefine the Control Off Position Tolerance or the Integrator Distance Threshold positions in the Motors section of the Node Controller Configuration File.
	Motor does not have propulsion power.	Check the power being supplied to the motor.

Table 5-10: Initial Vehicle Troubleshooting (Continued)

Fault	Problem	Solution
Vehicle Obstructed Status is being reported as high “1”.	There is a vehicle in the way.	This occurs during normal operation when vehicles are in a queue or when a vehicle is in a switch, which keeps another vehicle from entering the switch.
	There is a hardware fault.	The vehicle is attempting to move onto a motor that has not completed startup.
	Attempting to move on, or to, a path that is suspended.	The vehicle is attempting to move onto a path that is suspended.
	Attempting to position part of the vehicle past the end of the path.	Send a new vehicle command to reposition the vehicle.
	The vehicle is stopped at a red traffic light.	Change the traffic light to green.

This page intentionally left blank.

Communications Format

The high-level controller supports communication with the host controller via EtherNet/IP as defined in the Open DeviceNet Vendor Association (ODVA) specification “EtherNet/IP Adaptation of CIP™ Specification” Volume 1 and Volume 2. Each layer of network communication within the standard OSI (Open Systems Interconnect, ISO/IEC 7498-1) model is described here. EtherNet/IP is an application that runs directly over the TCP and UDP layer 4 transport layers in the OSI model. The implementation of EtherNet/IP only uses the TCP transport layer.

EtherNet/IP

The standard OSI model for layers 1...5 are outlined here. The high-level controller application runs on top of the TCP transport layer. Standard 10/100/1000 Base-TX, half-duplex or full-duplex twisted-pair Ethernet is used for all network communications.

Physical Layer

The electrical interface that is supported for communication between the high-level controller software running on the node controller hardware and the host controller is twisted-pair Ethernet. This interface is based on the IEEE 802.3 and IEEE 802.3u Ethernet and Fast Ethernet communication standards.

The interface uses IEEE 802.3 and IEEE 802.3u standard signal levels for 10/100 Base-TX. A standard Category 5 or better cable is used to connect the HLC directly to the host controller when a hub, switch, or router is not used for the connection.

Data Link Layer

The data link layer is standard 802.3/802.3u Ethernet packet framing. This framing uses 6 byte destination and source MAC IDs, 2 byte protocol type, 1500 byte MTU (maximum transmission unit) of payload data, and 2 byte FCS (frame check sequence). 10 megabit or 100 megabit signaling rates in full-duplex and half-duplex modes are supported. It is expected that any standard Ethernet device using a twisted-pair physical layer will have no problems inter-operating with the high-level controller computer at the physical and data link layers.

Network Layer

The network layer communications for the high-level controller is Internet Protocol version 4 (IPv4). For diagnostic purposes during initial setup and optional network layer health status checks, the HLC supports accepting Internet Control Message Protocol (ICMP) echo requests and responding to echo requests with ICMP echo responses. The HLC supports the additional ICMP messages that are required to support IPv4 fragmentation and redirection of network layer packets to the next hop IP router if customer premise networking equipment requires the HLC to honor such IP fragmentation or packet routing redirection. It is expected that any standard modern IPv4 implementation will have no problems inter-operating with the HLC computer.

The high-level controller does not support dynamic routing protocols and does not act as a router in the customer premise network. The HLC drops any packets that it receives via some other peer on the network using the HLC computer as its next hop IPv4 router.

The high-level controller supports configuration with a static IPv4 address, network mask, and gateway address of the next hop IP router.

The high-level controller supports IPv4 Address Resolution Protocol (ARP) to support the discovery of peer Ethernet data link layer MAC IDs.

Transport Layer

The high-level controller uses a Transmission Control Protocol (TCP) stream over the IPv4 network layer to support multiple end-to-end connections.

The high-level controller is the server and the host controller is the client for TCP endpoint communication roles. The HLC listens for incoming TCP connections from the host controller on the following ports:

- 800 for the Host Status connections.
- 8000 for the MagneMotion Virtual Scope connection.
- 44818 for standard EtherNet/IP.

The high-level controller supports at least eight simultaneous TCP connections on port 44818. The HLC supports one single TCP connection where the HLC is the client and the host controller is the server for implicit (unconnected) messages to the host controller. This TCP connection is directed to the standard EtherNet/IP TCP port 44818 and is used to support implicit (unconnected) tag memory read and write operations that are performed by the HLC. If the TCP is dropped for any reason, the HLC waits at least 1 second and then attempt to re-establish the connection. This pause is to prevent continually dropped connection attempts from consuming excessive CPU or other resources on the HLC computer and PLC EtherNet/IP interface.

NOTE: The high-level controller can support one Host Control TCP connection to TCP port 799. This connection must not be used while the transport system is under EtherNet/IP control to prevent command conflicts.

The high-level controller supports up to four Host Status TCP connections from the host controller at a time. If a new TCP connection is made to port 800 when the HLC already has four established connections, the oldest connection is dropped and the new connection is used.

NOTE: The Host Status connections can be used to monitor the status of the transport system using the various status request commands. Any other commands are dropped.

The high-level controller supports one Virtual Scope TCP connection from the host controller at a time. If a new TCP connection is made to port 8000 when the HLC already has an established connection, the established connection is dropped and the new connection is used.

When the high-level controller software receives an invalid message on an established TCP connection as determined by the framing that is described in the *MagneMotion Host Controller TCP/IP Communication Protocol User Manual*, [MMI-UM003](#), the HLC software drops the connection.

The high-level controller throttles incoming TCP connections to port 799 and 800 if a problem in the customer premise host controller software causes an excessive number of connections to be established by the host controller and dropped by the HLC due to the requirements described. If the connection is dropped, the HLC waits 1 second before allowing a new connection.

Application Layer

The high-level controller uses EtherNet/IP as defined in the ODVA EtherNet/IP specification over the TCP transport layer. The HLC uses Implicit (unconnected) messaging over TCP as a client of the host controller to perform tag read and write operations as defined in the tag memory formats in *Protocol Reference* [on page 145](#). The HLC uses Explicit (connected) messaging over TCP as a server device that the host controller can connect to for explicit message commands that can be sent by the host controller as defined in the explicit message command formats in *Protocol Reference* [on page 145](#).

CIP Standard Objects

The high-level controller supports a subset of Control and Information Protocol (CIP) standard objects. This CIP subset allows tools that can browse via EtherNet/IP (for example, RSLinx[®] from Rockwell Automation[®]) to probe the HLC for the state of those objects.

CIP Identity Object

Currently, the high-level controller only supports requests for the CIP identity object so the overall health of the HLC can be assessed with a standard EtherNet/IP browsing tool. See the EtherNet/IP specification for a description of each of the fields in the identity object.

Identity Object Status

When the high-level controller is in the operational state with valid configuration, the HLC reports “configured” in the identity object status field. When the HLC is not operational due to an internal problem, the HLC reports an unrecoverable major fault in the identity object status field.

When the high-level controller has no internal problems with an invalid configuration, the HLC reports a recoverable major fault in the identity object status field.

Other CIP Objects

There are several other objects the EtherNet/IP specification calls out as required and optional for an ODVA-compliant EtherNet/IP implementation. Since the implementation of these objects is not important for the operation of the transport system, they are currently not implemented.

Index

A

Acceleration

- commanded, [218](#), [223](#)
- ordered, [240](#)

B

Backup, [14](#)

Backward Vehicle Motion, [58](#)

Bidirectional Vehicle Motion, [58](#)

Block Faults

- MM LITE motor, [279](#)
- QSHT motor, [291](#)
- QuickStick motor, [283](#)

C

Caught Up, [239](#)

CIP, *see* Control and Information Protocol

Clear Suspect Bit, [206](#)

Command Counter, [152](#)

Communication Cables, identification, [17](#)

component_type, [336](#), [337](#)

component_type, [336](#)

Computer Requirements, [21](#)

Configuration File, *see* Node Controller Configuration File

Configurator Utility *see* MagneMotion System Configurator

Connect to High-Level Controller, [25](#)

Console Interface, description, [19](#)

Control and Information Protocol, [395](#)

controller_image, *see* Node Controller Software Image File

ControlLogix

- Logix Designer, [16](#)
- prerequisite, [16](#)
- UDTs, [146](#)

create_traffic_light_cmd, [151](#)

D

Deceleration, [218](#), [223](#)

Delete Vehicle, [208](#)

delete_traffic_light_cmd, [154](#)

Demo Script

- description, [20](#)
- use, [22](#)

Digital I/O

- read inputs, [56](#), [265](#)
- set outputs, [47](#), [176](#)
- status, [56](#)

Direction, Vehicle Motion, [218](#), [223](#)

Diverge Node

- description, [123](#)

Driver Board, MM LITE Motor, [277](#)

E

EMO, [78](#)

Error Codes, [333](#)

E-stop, [37](#), [75](#)

Ethernet

- layers, [393](#)
- node controller address, [21](#)

Ethernet Motor Commissioning Tool, description, [19](#)

EtherNet/IP

- address, [148](#)
- application layer, [395](#)

Excessive Following Error, [239](#)

Exit Path, [105](#)

Explicit Message Requests, [149](#)

extended_hlc_status, [228](#)

extended_nc_status, [231](#)

extended_vehicle_status, [234](#)

F

FastStop, [35](#), [181](#)

Faults

- MagneMover LITE, [276](#)
- motor not responding, [76](#)
- QuickStick, [280](#)
- QuickStick HT, [287](#)
- under-voltage, [76](#)

Flags

- downstream communication link status, [297](#)
- node status, [272](#)
- path motion status, [296](#)
- path status, [296](#)
- terminus signals, [271](#)
- upstream communication link status, [297](#)
- vehicle motion, [218](#), [223](#)
- vehicle status, [237](#), [331](#)

Following Downstream, [239](#)

Following Upstream, [238](#)

Forward Vehicle Motion, [58](#)

G

Gateway Node

- description, [116](#)
- enable/disable, [157](#)
- handshake, [116](#)

generic_node_command, [157](#)

Getting Started, [21](#)

H

Handshake

- Terminus Node entry, [107](#)
- Terminus Node exit, [111](#)

heartbeat, [242](#)

High-Level Controller

- commands, [149](#)
- connection, [25](#)
- extended status, [48](#), [228](#)
- identification, [17](#)
- responses, [226](#)
- status, [48](#)

Hindered

- status, [238](#), [331](#)
- troubleshooting, [390](#)

HLC, *see* High-Level Controller

HLC_status, [244](#)

Host Communication Protocol, description, [16](#)

Host Controller

- firmware version, [16](#)
- identification, [17](#)
- type supported, [16](#)

I

Image Files

- motor, [19](#)
- node controller, [19](#)

Inserting Vehicles, [74](#)

Interlock, [77](#)

Inverters

- disable, [167](#)
- enable, [167](#)

J

Jam, *see* Hindered

L

Load Status, [240](#), [332](#)

Locate Status, [238](#)

Lock Status, [238](#)

M

MagneMotion System Configurator, description, [19](#)

Magnet Array Type File, description, [19](#)

Magnet Array, identification, [17](#)

magnet_array_type.xml, *see* Magnet Array Type File

Manual

- prerequisites, [9](#)

Merge Node

- description, [121](#)

Merge or Diverge Node Operation, [120](#)

Merge or Diverge Nodes, [120](#)

Merge-Diverge Node, description, [124](#)

mgmt_nc_cmd_status, [246](#)

mgmt_nc_restart_cmd, [160](#)

mgmt_nc_set_config_cmd, [162](#)

MICS File, description, [20](#)

MICS_motor_data.xml, *see* MICS File

MMConfigTool.exe, *see* MagneMotion System Configurator

MMI_component_type, *see* udt_MMI_compo-

nent_type

MMI_create_traffic_light_cmd, [151](#)
 MMI_delete_traffic_light_cmd, [154](#)
 MMI_extended_hlc_status, [228](#)
 MMI_extended_nc_status, [231](#)
 MMI_extended_vehicle_status, [234](#)
 MMI_generic_node_command, [157](#)
 MMI_heartbeat, [242](#)
 MMI_HLC_status, [244](#)
 MMI_mgmt_nc_cmd_status, [246](#)
 MMI_mgmt_nc_restart_cmd, [160](#)
 MMI_mgmt_nc_set_config_cmd, [162](#)
 MMI_motor_inverter_cmd_status, [249](#)
 MMI_motor_inverter_command, [165](#)
 MMI_mp_command_status, [251](#)
 MMI_mp_link_command, [169](#)
 MMI_mp_path_end_status, [254](#)
 MMI_mp_unlink_command, [173](#)
 MMI_node_command_status, [261](#)
 MMI_node_controller_cmd_status, [263](#)
 MMI_node_controller_dio_command, [176](#)
 MMI_node_controller_dio_status, [265](#)
 MMI_node_controller_status, [267](#)
 MMI_node_status, [269](#)
 MMI_path_command, [180](#)
 MMI_path_command_status, [273](#)
 MMI_path_ml_faults_status, [276](#)
 MMI_path_qs_faults_status, [280](#)
 MMI_path_qs_ht_faults_status, [287](#)
 MMI_path_status, [295](#)
 MMI_propulsion_power_cmd_status, [298](#)
 MMI_propulsion_power_status, [301](#)
 MMI_qs_ht_sensor_map, [303](#)
 MMI_set_prop_power_state_cmd, [184](#)
 MMI_set_traffic_light_cmd, [188](#)
 MMI_sm_command_status, [306](#)
 MMI_sm_metric_data, [309](#)
 MMI_sm_poll_command, [191](#)
 MMI_sm_subscription_command, [197](#)
 MMI_station_arrivals, [313](#)
 MMI_terminus_node_command, [202](#)
 MMI_traffic_light_cmd_status, [315](#)
 MMI_traffic_light_status, [318](#)

MMI_vehicle_command, [205](#)

MMI_vehicle_command_status, [321](#)
 MMI_vehicle_delete_order, [208](#)
 MMI_vehicle_follow_order, [211](#)
 MMI_vehicle_order_status, [324](#)
 MMI_vehicle_position_order, [216](#)
 MMI_vehicle_station_order, [221](#)
 MMI_vehicle_status, [328](#)

Monitor

digital I/O status, [56](#)
 extended vehicle status, [51](#)
 high-level controller extended status, [48](#)
 high-level controller status, [48](#)
 motor controller status, [55](#)
 motor inverter status, [55](#)
 motor status, [52](#)
 Moving Path node status, [55](#)
 node controller extended status, [49](#)
 node controller status, [49](#)
 node status, [50](#)
 path status, [50](#)
 propulsion power supply status, [56](#)
 station status, [54](#)
 traffic light status, [54](#)
 transport system, [48](#)
 vehicle status, [51](#)

Motion

FastStop, [35](#), [181](#)
 permission, [106](#)
 platooning, [61](#)
 resume, [181](#)
 suspend, [181](#)
 Switch Node, [144](#)
 Terminus Node, [105](#)
 to position, [59](#)
 to station, [60](#)

Motor

identification, [17](#)
 MagneMover LITE, [97](#)
 monitor, [48](#)
 QuickStick 100, [98](#)
 QuickStick 150, [98](#)
 QuickStick HT, [99](#)
 reset, [28](#)
 start-up, [27](#)
 status, [52](#)
 warm reset, [39](#)

Motor Controller, status, [55](#)
 Motor ERF Image File, description, [19](#)
 Motor Inverters, status, [55](#)
 Motor Type File, description, [19](#)
motor_image.erf, *see* Motor ERF Image File
motor_inverter_cmd_status, [249](#)
motor_inverter_command, [165](#)
motor_type.xml, *see* Motor Type File
 Move Vehicles, [57](#)
 Moving Path

- command status, [251](#)
- link, [169](#)
- path end status, [254](#)
- unlink, [173](#)

 Moving Path Node

- description, [127](#)
- status, [55](#)

mp_command_status, [251](#)
mp_link_command, [169](#)
mp_path_end_status, [254](#)
mp_unlink_command, [173](#)

N

NC File Retrieval Tool, description, [19](#)
 NCHost TCP Interface Utility, description, [19](#)
 NCHost.exe, *see* NCHost TCP Interface Utility
 Network, identification, [17](#)
 Node Controller

- extended status, [49](#), [231](#)
- identification, [17](#)
- IP address, [21](#)
- status, [49](#)

 Node Controller Configuration File

- description, [20](#)
- use, [21](#)

 Node Controller Console Interface, *see* Console Interface
 Node Controller Software Image File, description, [19](#)
 Node Controller Web Interface, *see* Web Interface
node_command_status, [261](#)
node_configuration.xml, *see* Node Controller Configuration File
node_controller_cmd_status, [263](#)
node_controller_dio_command, [176](#)
node_controller_dio_status, [265](#)

node_controller_status, [267](#)

Nodes

- Diverge, [123](#)
- Gateway, [116](#)
- Merge, [121](#)
- Merge-Diverge, [124](#)
- Moving Path, [127](#)
- Overtravel, [126](#)
- ownership, [102](#), [270](#)
- Relay, [104](#)
- Simple, [103](#)
- status, [50](#)
- straddling, [30](#)
- Terminus, [105](#)
- types, [103](#)

node_status, [269](#)

Notes, [10](#)

O

Obstructed

- status, [237](#), [331](#)
- troubleshooting, [391](#)

Overtravel Node, description, [126](#)

P

path_command, [180](#)
path_command_status, [273](#)
path_ml_faults_status, [276](#)
path_qs_faults_status, [280](#)
path_qs_ht_faults_status, [287](#)

Paths

- exit, [105](#)
- FastStop, [35](#)
- resume, [37](#)
- start-up, [30](#)
- status, [50](#)
- suspend, [33](#)
- Switch Node, [144](#)
- Terminus Node, [105](#)

path_status, [295](#)

PID Set Status, [240](#)

Platoon, [88](#)

- create, [62](#)
- decouple, [65](#)
- move, [64](#)

Platooning

- change direction, [89](#)
- extend platoon, [85](#), [86](#)
- follow downstream, [83](#)
- follow upstream, [84](#)
- split platoon, [88](#)
- uncouple vehicle while moving, [88](#)
- uncouple vehicles, [87](#)
- Positions
 - move to, [57](#), [59](#)
 - move to command, [216](#)
- Power Cables, identification, [17](#)
- Power Supply, identification, [17](#)
- Profile Stale Error, [239](#)
- Propulsion Power Supply
 - set state, [46](#)
 - status, [56](#)
- `propulsion_power_cmd_status`, [298](#)
- `propulsion_power_status`, [301](#)

R

- Relay Node, description, [104](#)
- Removing Vehicles, [74](#)
- Reset
 - MMI_path_command, [181](#)
 - paths, [29](#), [40](#)
 - transport system, [28](#)
- Response Tags, [226](#)
- Restricted Parameters File, description, [20](#)
- `restricted_parameters.xml`, *see* Restricted Parameters File
- Resume
 - MMI_path_command, [181](#)
 - paths, [37](#)
 - vehicle motion, [37](#)

S

- Safe Stopping Distance, [72](#)
- Safety Alert Types, [10](#)
- Scope, *see* Virtual Scope
- `set_prop_power_state_cmd`, [184](#)
- `set_traffic_light_cmd`, [188](#)
- Simple Node, description, [103](#)
- `sm_metric_data`, [309](#)
- `sm_poll_command`, [191](#)
- `sm_subscription_command`, [197](#)
- Software Types, [18](#)

- Stall Status, [238](#)
- Startup
 - MMI_path_command, [181](#)
 - paths, [30](#)
 - transport system, [27](#)
- `station_arrivals`, [313](#)
- Stations
 - move to, [57](#), [60](#)
 - move to command, [221](#)
 - status, [54](#)
- Status Codes, [333](#)
- Status Tags, [226](#)
- Straddling Node, [30](#)
- Suspect Status, [238](#), [332](#)
- Suspend
 - MMI_path_command, [181](#)
 - paths, [33](#)
 - vehicle motion, [33](#)

- Switch Node, [120](#)
- Switch Node Operation, [120](#), [144](#)
- System Monitoring
 - command status, [306](#)
 - component type, [336](#)
 - metric data, [309](#)
 - poll command, [191](#)
 - subscription command, [197](#)

T

- Tags
 - response, [226](#)
 - status, [226](#)
- Terminus Node
 - command, [202](#)
 - description, [105](#)
 - enter, [106](#)
 - exit, [110](#)
 - handshake, [105](#)
 - motion, [105](#)
 - signals, [203](#)
 - status, [270](#), [271](#)
- `terminus_node_command`, [202](#)
- Text Files
 - Demo Script, [20](#)
 - Track File, [20](#)
- Track File, description, [20](#)
- Track Layout File, description, [20](#)
- `track_file.mmtrk`, *see* Track File

U

track_layout.ndx, *see* Track Layout File

Traffic Lights

- create, [41](#), [151](#)
- delete, [43](#), [154](#)
- set, [42](#), [188](#)
- status, [54](#)

traffic_light_cmd_status, [315](#)

traffic_light_status, [318](#)

Transport System

- components, [17](#)
- MagneMover LITE, [15](#), [97](#)
- monitor, [48](#)
- QuickStick, [15](#)
- QuickStick 100, [98](#)
- QuickStick 150, [98](#)
- QuickStick HT, [99](#)
- reset, [28](#)
- software, [18](#)
- start-up, [27](#)
- vehicle positions, [57](#)
- warm reset, [39](#)

Type Files

- magnet array, [19](#)
- motor, [19](#)

U

UDT

- communication setup, [147](#)
- explicit message requests, [149](#)
- explicit message setup, [146](#)
- response tags, [226](#)
- status tags, [226](#)

udt_MMI_component_type, [336](#)

V

Vehicle Command Order, [205](#)

Vehicle Decoupled, [239](#)

Vehicle Motion

- backward, [58](#)
- bidirectional, [58](#)
- direction, [218](#), [223](#)
- FastStop, [35](#)
- flags, [218](#), [223](#)
- forward, [58](#)
- resume, [37](#)
- start-up, [30](#)

suspend, [33](#)

Vehicle Positions, [57](#)

Vehicle Signal, [237](#), [331](#)

vehicle_command_status, [321](#)

vehicle_delete_order, [208](#)

vehicle_follow_order, [211](#)

vehicle_order_status, [324](#)

vehicle_position_order, [216](#)

Vehicles

- brick-wall headway, [71](#)
- command, [205](#)
- delete, [208](#)
- extended status, [51](#)
- insert, [74](#)
- move, [57](#)
- remove, [74](#)
- status, [51](#)
- straddling node, [30](#)

vehicle_station_order, [221](#)

vehicle_status, [328](#)

Velocity

- commanded, [217](#), [222](#)
- ordered, [240](#)
- reported, [237](#), [330](#)

Virtual Scope, description, [19](#)

W

Warm Reset

- MMI_path_command, [181](#)
- transport system, [39](#)

Web Interface, description, [19](#)

X

XML Files

- Magnet Array Type File, [19](#)
- MICS File, [20](#)
- Motor Type File, [19](#)
- Node Controller Configuration File, [20](#)
- restricted parameters file, [20](#)
- Track Layout File, [20](#)

Changes

Overview

This manual is changed as required to keep it accurate and up-to-date to provide the most complete documentation possible for the MagneMotion® Host Controller EtherNet/IP Communication Protocol. This section provides a brief description of each significant change.

NOTE: Distribution of this manual and all addenda and attachments is not controlled. To identify the current revision, see the [Literature Library](#) on the Rockwell Automation website.

Rev. A

Initial release to support the following NC Software Image Versions:

- MM LITE™ – 1.1.19
- QS 100 – 0.9.147

Rev. B

Revised to support the following NC Software Image Versions:

- MM LITE – 4.1.25
- QS 100 – 7.1.20

Added the following:

- New message tags:
 - MMI_generic_node_command*
 - MMI_node_controller_command*
- New status tags:
 - MMI_node_controller_cmd_status*
 - MMI_node_controller_dio_status*
 - MMI_path_qs_ht_faults_status*
- References to *QuickStick High Thrust*.
- Support for QSHT motors.

Updated the following:

- Trademark and copyright information.
- The Overview Note in this section.
- In the *About This Manual* chapter, the *Notes, Safety Notices, and Symbols* section to describe number and measurement conventions better. The *Notes, Safety Notices, and Symbols* section to include Symbol Identification. And, the list of *Additional Resources*.
- In *Chapter 1, Introduction*, the *Transport System Components Overview* section to include the NC-12 node controller. The *Transport System Software Overview* to update the descriptions of existing Type files. And, expanded the *Getting Started* procedure.
- In *Chapter 2, Transport System Control*, all figures that are used in the examples and all examples to reflect a typical transport system layout better.
- In *Chapter 3, Application Notes*, the *E-stops* cautions and description and the figure for the *Gateway Node*.
- In *Chapter 4, Protocol Reference*, the *MMI_path_command* section to include Fast-Stop. All *PID Set* references to provide 16 PID Sets. and, the *HLC Status Codes* section.
- In *Chapter 5, Troubleshooting*, the *EtherNet/IP Communications Troubleshooting* section.
- The *Glossary* and *Index*.

Removed the following:

- Obsolete status tags:
 - MMI_host_switch_command (unsupported node type).
 - MMI_host_switch_status (unsupported node type).
 - MMI_path_manufacturing_info (replaced by *MMI_path_ml_faults_status*).
 - MMI_path_qs_manufacturing_info (replaced by (*MMI_path_qs_faults_status* and *MMI_path_qs_ht_faults_status*)).
- All references to the standard node controller (replaced by the NC-12 node controller). Support for the standard node controller, including software, spare parts, technical support, and service continues to be available.
- All references to Zero Headway as this feature is no longer supported.
- All references to Automatic Path Recovery as this feature is deprecated.
- All references to unsupported node types (Turntable and Host Switch).

Rev. C

Revised to support the following NC Software Image Versions:

- MM LITE – 4.1.37
- QS 100 – 7.2.15
- QSHT – 7.2.12

Added the following:

- New status tag:
MMI_extended_vehicle_status
- In *Chapter 4, Protocol Reference*, the *Host Controller to Node Controller Compatibility* section to describe the use of the *MMI_extended_vehicle_status* tag. Software version support identification for all commands and responses.
- In *Chapter 5, Troubleshooting*, bit-level troubleshooting for *MagneMover LITE Motor Fault Troubleshooting*, *QuickStick 100 Motor Fault Troubleshooting*, and *QSHT Gen 2 and QSHT 5700 Motor Fault Troubleshooting*.

Updated the following:

- In *Chapter 2, Transport System Control*, the *Running the Transport System* procedure to include verifying the node controller status.
- In *Chapter 3, Application Notes*, the description of *Safe Stopping Distance*. The *Node Type Descriptions and Usage* section to include the requirement for the vehicle to move past the configured clearance distances. The *Gateway Node* section to define the limits on Gateway Node usage. The description of queueing at *Merge Nodes* and *Diverge Nodes*. The description of *Terminus Node* operation to include the clearance requirement.
- In *Chapter 4, Protocol Reference*, the range of the *station_id* field (255 max). The description of the *MMI_path_command* message to provide additional details for the *Suspend Movement* and *FastStop* commands. The description of the *acceleration_limit* for vehicle motion orders. The description of the motor fault bits for *MMI_path_ml_faults_status*, *MMI_path_qs_faults_status*, and *MMI_path_qs_ht_faults_status*. The description of the *0x0E* command status code.
- In *Chapter 5, Troubleshooting*, the *Motor Fault Troubleshooting* section.

Rev. D

Revised to support the following NC Software Image Versions:

- MM LITE – 4.1.37
- QS 100 – 7.2.21
- QSHT – 7.2.12

Added the following:

- In [Chapter 1, Introduction](#), information about the Virtual Scope utility. Description of the Node Controller Console Interface.
- In [Chapter 2, Transport System Control](#), safety warnings that are related to vehicle motion. Examples for [FastStop](#), [Node Controller Digital I/O Control](#), and [Extended Vehicle Status](#) operation.
- In [Chapter 3, Application Notes](#), a process for [Inserting and Removing Vehicles](#).
- In [Chapter 4, Protocol Reference](#), motor status to [MMI_path_qs_faults_status](#). Motor status and hardware over-current warning to [MMI_path_qs_ht_faults_status](#). And, additional [HLC Status Codes](#) (0x1A, 0x1B, 0x26, 0x27, 0x28, 0x43, 0x45).
- In [Chapter 5, Troubleshooting](#), added motor status and hardware over-current warning to [QSHT Gen 2 and QSHT 5700 Motor Fault Troubleshooting](#) troubleshooting.

Updated the following:

- Changed the revision from alpha (Rev. D) to numeric (Ver. 04).
- Changed the logo to “A Rockwell Automation Company” version.
- Trademark and copyright information.
- In the [About This Manual](#) chapter, the Symbol Identification section to show the change in the Pinch/Crush hazard symbol and updated the description of the hazard. Updated the list of [Additional Resources](#) to add the [Host Controller TCP/IP Communication Protocol User Manual](#) and the [Virtual Scope Utility User Manual](#). And, updated the Contact Information section.
- In [Chapter 1, Introduction](#), the [Transport System Software Overview](#) and the [Simplified View of Transport System Software Organization](#).
- In [Chapter 2, Transport System Control](#), the examples of [Suspend](#) and [Resume](#) operation. The descriptions of keep-out area operation for [Suspend](#) and [FastStop](#).
- In [Chapter 3, Application Notes](#), the [Node Type Descriptions and Usage](#), [Terminus Node](#), and [Gateway Node](#) examples. The [Node Type Descriptions and Usage](#) reference section to use the same order as other manuals.
- In [Chapter 4, Protocol Reference](#), all screen shots now show Allen-Bradley® Studio 5000 Logix Designer®. The description of the [dest_path_id](#) argument for [MMI_extended_vehicle_status](#). The “Jammed” vehicle status flag is changed to “Hindered”. The descriptions of [MMI_path_qs_faults_status](#) and [MMI_path_qs_ht_faults_status](#) fault data descriptions. The names of tag fields for the [MMI_node_controller_dio_status](#), [MMI_path_ml_faults_status](#), [MMI_path_qs_faults_status](#), and [MMI_path_qs_ht_faults_status](#) tags. The description of the [MMI_vehicle_status](#) tag.

- In *Chapter 5, Troubleshooting*, the temperature range for the QS 100 *Over-temperature fault*. The current limit for the QSHT *Hardware over-current fault*.

Rev. E

Revised to support the following NC Software Image Versions:

- All Motor Families – 15.11.x.

Added the following:

- New host controller to HLC explicit message:
 - MMI_create_traffic_light_cmd*
 - MMI_delete_traffic_light_cmd*
 - MMI_mgmt_nc_restart_cmd*
 - MMI_mgmt_nc_set_config_cmd*
 - MMI_motor_inverter_command*
 - MMI_mp_link_command*
 - MMI_mp_unlink_command*
 - MMI_set_prop_power_state_cmd*
 - MMI_set_traffic_light_cmd*
 - MMI_sm_poll_command*
 - MMI_sm_subscription_command*
 - MMI_vehicle_command*
 - MMI_vehicle_follow_order*
- Added a section for *Embedded udt_MMI_component_type*
- New HLC to host controller status and response memory tags:
 - MMI_extended_hlc_status*
 - MMI_extended_nc_status*
 - MMI_mgmt_nc_cmd_status*
 - MMI_motor_inverter_cmd_status*
 - MMI_mp_command_status*
 - MMI_mp_path_end_status*
 - MMI_propulsion_power_cmd_status*
 - MMI_propulsion_power_status*
 - MMI_qs_ht_sensor_map*
 - MMI_sm_command_status*
 - MMI_sm_metric_data*
 - MMI_traffic_light_cmd_status*
 - MMI_traffic_light_status*
 - MMI_vehicle_command_status*
 - MMI_vehicle_order_status*
- In *Chapter 1, Introduction*, added descriptions of new utilities (NC File Retrieval Tool and Ethernet Motor Commissioning Tool) and new file types.
- In *Chapter 2, Transport System Control*, added *Warm Reset*, *Traffic Light Control*, *Motor Controller Control*, and *Propulsion Power Supply Control* to the *Transport System Reset, Startup, and Operation* section. Added *Station Status*, *Traffic Light Status*, *Motor Controller (Inverter) Status*, *Moving Path Node Status*, and *Propulsion Power*

Supply Status to the *Monitoring Transport System Status* section. Added *Platooning* to the *Moving Vehicles* section.

- In *Chapter 3, Application Notes*, added *Thrust Limitations* information. Added *Moving Path Node*, *Platooning*, *Traffic Lights*, *System Monitoring*, and *Node Type Descriptions and Usage*.

Updated the following:

- Changed the revision to alpha (Rev. E) only.
- Updated all trademark and copyright information and moved to the back cover.
- Updated the titles and part numbers for all referenced manuals.
- Updated the appearance of the safety notices to match Rockwell Automation standards.
- Updated all support references to reference the information on the *Back Cover*.
- In the *About This Manual* chapter, updated the descriptions of the *Safety Notices* and the list of *Additional Resources*.
- In *Chapter 1, Introduction*, updated the *Transport System Components Overview*, the *Transport System Software Overview*, and the *Getting Started with the EtherNet/IP Communication Protocol*.
- In *Chapter 2, Transport System Control*, updated examples to use the updated *MMI_extended_vehicle_status*.
- In *Chapter 3, Application Notes*, updated *Node Type Descriptions and Usage* to support current nodes.
- In *Chapter 4, Protocol Reference*, updated all tag descriptions to reference a configuration table instead of a screen shot. Updated the *MMI_extended_vehicle_status*, *MMI_path_ml_faults_status*, *MMI_path_qs_faults_status*, and *MMI_path_qs_ht_faults_status* tags.
- In *Chapter 5, Troubleshooting*, updated the *Motor Fault Troubleshooting*, and the *This section describes the common vehicle faults, and general solutions* section.
- Updated the *Glossary* and *Index*.

Removed the following:

- Removed all references to the *Mitsubishi PLC TCP/IP Library User Manual*.
- In *Chapter 3, Application Notes*, replaced Shuttle node which is deprecated, with the Moving Path node.
- List of Table, List of Figures, Glossary, Manual Conventions, Transport System Limits, Audience, Manual Structure, and, Contact Rockwell Automation Support.

Consolidated the following publications into this document:

- MMI-AT035, QuickStick 100 Startup using Moving Paths
- MMI-UM017, Moving Path
- MMI-UM019, System Monitoring
- MMI-UM029, Suspect Vehicle Bit
- MMI-UM036, Vehicle Platooning
- MMI-UM040, Managed Configurations

Rockwell Automation Support

Use these resources to access support information.

Technical Support Center	Find help with how-to videos, FAQs, chat, user forums, Knowledgebase, and product notification updates.	rok.auto/support
Local Technical Support Phone Numbers	Locate the telephone number for your country.	rok.auto/phonesupport
Technical Documentation Center	Quickly access and download technical specifications, installation instructions, and user manuals.	rok.auto/techdocs
Literature Library	Find installation instructions, manuals, brochures, and technical data publications.	rok.auto/literature
Product Compatibility and Download Center (PCDC)	Download firmware, associated files (such as AOP, EDS, and DTM), and access product release notes.	rok.auto/pcdc

Documentation Feedback

Your comments help us serve your documentation needs better. If you have any suggestions on how to improve our content, complete the form at rok.auto/docfeedback.

Waste Electrical and Electronic Equipment (WEEE)



At the end of life, this equipment should be collected separately from any unsorted municipal waste.





Rockwell Automation maintains current product environmental compliance information on its website at rok.auto/pec.

Allen-Bradley, expanding human possibility, MagneMotion, MagneMover, MML, MM LITE, QuickStick, QuickStick HT, Rockwell Automation, and SYNC IT are trademarks of Rockwell Automation, Inc. Microsoft and Windows are registered trademarks of Microsoft Corporation.

EtherNet/IP is a trademark of ODVA, Inc.

Trademarks not belonging to Rockwell Automation are property of their respective companies.

Rockwell Otomasyon Ticaret A.Ş. Kar Plaza İş Merkezi E Blok Kat:6 34752, İçerenköy, İstanbul, Tel: +90 (216) 5698400 EEE Yönetmeliğine Uygundur

Connect with us.    

rockwellautomation.com — expanding **human possibility**

AMERICAS: Rockwell Automation, 1201 South Second Street, Milwaukee, WI 53204-2496 USA, Tel: (1) 414.382.2000, Fax: (1) 414.382.4444

EUROPE/MIDDLE EAST/AFRICA: Rockwell Automation NV, Pegasus Park, De Kleetlaan 12a, 1831 Diegem, Belgium, Tel: (32) 2663 0600, Fax: (32) 2 663 0640

ASIA PACIFIC: Rockwell Automation SEA Pte Ltd, 2 Corporation Road, #04-05, Main Lobby, Corporation Place, Singapore 618494, Tel: (65) 6510 6608, FAX: (65) 6510 6699

UNITED KINGDOM: Rockwell Automation Ltd., Pitfield, Kiln Farm, Milton Keynes, MK11 3DR, United Kingdom, Tel: (44)(1908) 838-800, Fax: (44)(1908) 261-917