Application Development Recommendations

Purpose: This application note describes the recommended program structure for host applications developed to control a MagneMotion System.

Introduction

In the MagneMotion system, a PLC or PC (referred to as the "host controller" or "host") sends instructions to and receives responses from a high level controller (HLC) on our system. The details of this messaging are described in our communications manuals, provided on the documentation CD with our system and available upon request.

This document will describe the recommended method to use when architecting the program running on your host controller. This will describe the best ways to make use of the feedback provided by the system in order to minimize communication and network congestion, maintain full status reporting of your system, optimize system performance, and simplify your controls architecture. We highly recommend using the architecture defined below both for reliability and troubleshooting reasons. Our customer support representatives will be much better able to assist with your host program if it follows the pattern described below.

All of the procedures outlined in this document are demonstrated in our sample PLC logic program, "MMI_sample_logic_ladder_only.ACD". This file can be found on the software CD provided with your system or made available on request.

Responses and Status updates

The MMI system gives a large amount of feedback. This feedback mainly falls into two categories; response and status.

A response is sent without the prompting of a user when events occur in the system. The most common response you will see will be the command status response. This response is sent when a command is received and processed by the system and again when that command completes. Responses are the cornerstone of the architecture described below and, because they are sent without the requirement for polling, are the fastest way to get information out of the system.

A status message provides information on the current state of elements within your system. These include the vehicles, paths, node controllers, and individual motors. If you are using a PC host, these are sent only when requested by the host. On a PLC, they are sent asynchronously as the status changes with the exception of vehicle status, which is sent on a fixed interval specified in the configuration file.

Startup Program Sequence

The startup sequence host program must follow a certain sequence to ensure that the system starts up correctly. The steps are laid out in order below.

1. Connect to the system. This is done automatically on a PLC when power is turned on to both units. On a PC, the host program needs to open a socket to the HLC.
2. Get the initial status. On a PLC, this can be accomplished by waiting approximately 3 seconds for the memory tags to fully populate. On a PC, this involves requesting the status of the paths, nodes, node controllers, and HLC.
3. Ensure that all node controllers in the system are in the operational state. This can take over a minute in very large systems or on long paths. During this time, PC based hosts will need to poll for node controller status to get updates. PLC systems will be updated automatically when the node controller state changes.
4. Send a reset command to all paths in the system. This can be accomplished in a single command by sending a reset command with a Path ID of 0 specified. Wait for the command status response to come back. There will be one response per path that was reset. Make sure all paths report reset accepted in this command status response.
    a. If a path reports reset rejected, parse the response to find out why. Response codes are listed in both the PLC and PC communication manuals.
5. Wait for all paths to report reset has completed. This is done through another command status response which includes a status of "Command completed successfully". Times to complete this process may vary based on layout. Please select a timeout appropriate for your configuration for this waiting period.
    a. The system can also respond with "Command failed – Timed out" during this period. In this case, prompt the user to check the HLC and node controller logs to determine the problem.
6. Repeat steps 4 and 5 with a Startup command. On a MagneMover LITE system this should happen very quickly. On a QuickStick System, this process can take up to several minutes depending where vehicles are stopped and track length.

At this point your system is operational and ready to receive orders. It is desirable at this point to order all vehicles to a startup station. This ensures that all vehicles are at a known location when you begin executing the program. An easy way to do this is to check the status of all vehicles and for every vehicle with a record on a PC host and for every vehicle with a non-zero path ID on a PLC host, send it an order to a specific location. More complex startup vehicle positioning logic can also be implemented at this point, based on vehicles' locations on startup or current payload. After this point, all vehicle movement will be based on command responses, so all vehicles should be given an order to an initial location as part of the startup procedure. Make sure each of these orders is accepted. This can be done in a separate routine or as part of your standard movement functionality.

Ordering Vehicles

Upon completion of the startup routine, including ordering vehicles to their initial stations, further vehicle orders can be issued. This is done as a reaction to vehicles arriving at their destinations.

Vehicles can be moved using move to position or move to station commands. We generally recommend using move to position commands with station locations defined in the PLC. This allows you to move station locations without adjusting your configuration file.

Some storage local to the host will need to be allocated to allow the program flow described below to function. Memory will need to be allocated to keep track of the current destination of each vehicle and the last order number sent to that vehicle. You can also use this storage to store data about the payload on each vehicle or any other relevant information. If using a state machine to transition between steps, the each vehicle's state can be stored in this memory as well.

This is implemented as a three step process.

1. Monitor. On a PLC, monitor the "MMI_vehicle_order_status" tag (for loops make this easy) and check for vehicles with a "last order accepted is complete" flag set or a current order status of "command completed successfully". In the PC, you will want to watch incoming messages for a vehicle command status with a status of "Command completed successfully". The vehicle record corresponding to the completed command should be noted as having arrived and can be transitioned to the next step.
2. Process and Launch. Once the arrival of a vehicle at its destination has been registered, this can be used to trigger whatever process needs to occur at this station. This can be a simple delay, a command to third party hardware, or just moving data in memory. Once the process for that station is complete, determine where the vehicle is going next and order it to that location. Store the destination location in the local storage described above, so that you know where the vehicle arrived when it reports its order has been completed.
3. Check for confirmation. After you send the order, watch for the command status response that comes back. If the order was accepted, you can put that vehicle back to step 1, waiting for arrival. If it is rejected, the reason should be noted and reacted to accordingly. That same order or a revised version will then need to be sent again.

While in certain circumstances it is necessary to deviate from this process, for example to override an in process order, it will support the vast majority of applications.

Polling for Status

On a PC, it is necessary to poll (request at a fixed interval) for status updates. How you do this can dramatically affect the responsiveness and functionality of your system. If you are polling at too high a rate, it can saturate your Ethernet communications network and cause significant delays in returning statuses and delivering commands.

The appropriate polling rate for your system depends on what you are trying to accomplish. If you are using the vehicle movement method described above, the only reason for polling is for fault detections. All other functions are handled through asynchronous, unrequested messages. In that case, polling cycles that would be reasonable for fault detection during normal operation would be:
HLC status: 30 seconds
Node controller status: 30 seconds
Path status: 10 seconds
Motor status: 30 seconds

Vehicle status will be handled differently depending on the number of vehicles in the system and what you are using the status for. If your system has a large number of vehicles, you may only want to get status for a few vehicles at a time or poll at a very low rate. For lower numbers of vehicles, higher rates are possible. The level of traffic you can support will vary based on your hardware, but we recommend not collecting more than 32 vehicle records in a 100ms period. This limit will work for most levels of host controller hardware and network hardware. You may be able to pull more than this with higher end hardware. If you are only using the vehicle status for jam detection, then you can poll once a second or less and not miss a jam.

If you are triggering something off of a vehicle passing a certain point or off of a node being in a certain state, faster polling rates may be required. You may need to experiment to see what your communication system can support. If you notice erratic behavior or order timing, you are probably polling too often and are congesting your network.

On a PLC, vehicle status is updated on a regular rate specified in the configuration file. The same limitations specified above still apply, and can still be overcome with more powerful networking hardware and Ethernet cards.

Conclusion

MagneMotion highly recommends the use of the above method of application design for a number of reasons:

- Ease of programming – Many customers run into problems when they attempt to constantly order and manage every vehicle. By using a fire and forget methodology, you remove this potential.
- Optimized network and resource loading – The method specified here allows you to control the system with the minimum possible load on your network and control hardware. This limits the effect of traffic to the MMI system on your network devices and frees up controller resources for other functions.
- Supportability – This is the format that our support personnel are most familiar with. While using a different methodology does not preclude support, support personnel will be better able to quickly grasp the design of your program and offer help when this common methodology is used.

If you run into any trouble when designing your host program, please give our customer support team a call at 978-757-9102 or email us at customersupport@magnemotion.com.

Related Documents:
990000436 – MANUAL, HOST CONTROLLER TCP COMMUNICATION PROTOCOL
990000437 – MANUAL, HOST CONTROLLER ETHERNET/IP COMMUNICATION PROTOCOL

More Information
MagneMotion website: www.magnemotion.com
Questions & Comments: http://www.magnemotion.com/about-magnemotion/contact.cfm

---