Upon receipt of the product and prior to initial operation, read these instructions thoroughly, and retain for future reference.

YASKAWA INSTRUCTIONS

YASKAWA-□□□ INSTRUCTIONS
CONTROLLER INSTRUCTIONS
OPERATOR’S MANUAL
MAINTENANCE MANUAL

Part Number: 169286-1CD
Revision: 6

Terms of Use and Copyright Notice
All rights reserved. This manual is freely available as a service to YASKAWA customers to assist in the operation of Motoman robots, related equipment and software. This manual is copyrighted property of YASKAWA and may not be sold or redistributed in any way. You are welcome to copy this document to your computer or mobile device for easy access but you may not copy the PDF files to another website, blog, cloud storage site or any other means of storing or distributing online content.

Printed in the United States of America

First Printing, 2014

YASKAWA America, Inc.
Motoman Robotics Division
100 Automation Way
Miamisburg, OH 45342
Phone: 937-847-6200

www.motoman.com
DANGER

- This manual explains the MotoPlus SDK for Visual Studio. Read this manual carefully and be sure to understand its contents before handling the DX100, DX200, FS100 or YRC1000.

- General items related to safety are listed in Chapter 1: Safety of the DX100, DX200, FS100 or YRC1000 Instructions. To ensure correct and safe operation, carefully read the DX100, DX200, FS100 or YRC1000 Instructions before reading this manual.

CAUTION

- Some drawings in this manual are shown with the protective covers or shields removed for clarity. Make sure all covers and shields are replaced before operating this product.

- The drawings and photos in this manual are representative examples and differences may exist between them and the delivered product.

- YASKAWA may modify this model without notice when necessary due to product improvements, modifications, or changes in specifications. If such modification is made, the manual number will also be revised.

- If your copy of the manual is damaged or lost, contact a YASKAWA representative to order a new copy. The representatives are listed on the back cover. Make sure to tell the representative the manual number listed on the front cover.

- YASKAWA is not responsible for incidents arising from unauthorized modification of its products. Unauthorized modification voids your product’s warranty.
We suggest that you obtain and review a copy of the ANSI/RIA National Safety Standard for Industrial Robots and Robot Systems (ANSI/RIA R15.06-2012). You can obtain this document from the Robotic Industries Association (RIA) at the following address:

Robotic Industries Association  
900 Victors Way  
P.O. Box 3724  
Ann Arbor, Michigan 48106  
TEL: (734) 994-6088  
FAX: (734) 994-3338  
www.roboticsonline.com

Ultimately, well-trained personnel are the best safeguard against accidents and damage that can result from improper operation of the equipment. The customer is responsible for providing adequately trained personnel to operate, program, and maintain the equipment. NEVER ALLOW UNTRAINED PERSONNEL TO OPERATE, PROGRAM, OR REPAIR THE EQUIPMENT!

We recommend approved YASKAWA training courses for all personnel involved with the operation, programming, or repair of the equipment.

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications.
Notes for Safe Operation

Read this manual carefully before installation, operation, maintenance, or inspection of the DX100, DX200, FS100, or YRC1000.

In this manual, the Notes for Safe Operation are classified as “DANGER”, “WARNING”, “CAUTION”, or “NOTICE”.

**DANGER**
Indicates an imminently hazardous situation which, if not avoided, will result in death or serious injury. Safety Signs identified by the signal word DANGER should be used sparingly and only for those situations presenting the most serious hazards.

**WARNING**
Indicates a potentially hazardous situation which, if not avoided, will result in death or serious injury. Hazards identified by the signal word WARNING present a lesser degree of risk of injury or death than those identified by the signal word DANGER.

**CAUTION**
Indicates a hazardous situation, which if not avoided, could result in minor or moderate injury. It may also be used without the safety alert symbol as an alternative to “NOTICE”.

**NOTICE**
NOTICE is the preferred signal word to address practices not related to personal injury. The safety alert symbol should not be used with this signal word. As an alternative to “NOTICE”, the word “CAUTION” without the safety alert symbol may be used to indicate a message not related to personal injury.

Even items described as “CAUTION” may result in a serious accident in some situations.

At any rate, be sure to follow these important items.

**NOTE**
To ensure safe and efficient operation at all times, be sure to follow all instructions, even if not designated as “DANGER”, “WARNING” and “CAUTION”.

---

169286-1CD
MotoPlus SDK for Visual Studio
Notes for Safe Operation

---

5 of 77
DANGER

- Maintenance and inspection must be performed by specified personnel.
  Failure to observe this caution may result in electric shock or injury.
- For disassembly or repair, contact your YASKAWA representative.
- Do not remove the motor, and do not release the brake.
  Failure to observe these safety precautions may result in death or serious injury from unexpected turning of the manipulator's arm.
DX100, DX200 or YRC1000 Controllers

WARNING

• Before operating the manipulator, check that servo power is turned OFF pressing the emergency stop buttons on the front door of the DX100, DX200 or YRC1000 controller and the programming pendant.

When the servo power is turned OFF, the SERVO ON LED on the programming pendant is turned OFF.

Injury or damage to machinery may result if the emergency stop circuit cannot stop the manipulator during an emergency. The manipulator should not be used if the emergency stop buttons do not function.

Figure 1: Emergency Stop Button

• Once the emergency stop button is released, clear the cell of all items which could interfere with the operation of the manipulator. Then turn the servo power ON.

Injury may result from unintentional or unexpected manipulator motion.

Figure 2: Release of Emergency Stop

• Observe the following precautions when performing teaching operations within the P-point maximum envelope of the manipulator:
  – View the manipulator from the front whenever possible.
  – Always follow the predetermined operating procedure.
  – Keep in mind the emergency response measures against the manipulator’s unexpected motion toward you.
  – Ensure that you have a safe place to retreat in case of emergency.

Improper or unintended manipulator operation may result in injury.

• Confirm that no person is present in the P-point maximum envelope of the manipulator and that you are in a safe location before:
  – Turning ON the power for the DX100, DX200, or YRC1000 controller.
  – Moving the manipulator with the programming pendant.
  – Running the system in the check mode.
  – Performing automatic operations.

Injury may result if anyone enters the P-point maximum envelope of the manipulator during operation. Always press an emergency stop button immediately if there is a problem.

The emergency stop button is located on the right of front door of the DX100, DX200 or YRC1000 controller and programming pendant.
WARNING

- Before operating the manipulator, check that servo power is turned OFF when the emergency stop button on the programming pendant is pressed. When the servo power is turned OFF, the SERVO ON LED on the programming pendant is turned OFF.

Injury or damage to machinery may result if the emergency stop circuit cannot stop the manipulator during an emergency. The manipulator should not be used if the emergency stop button does not function.

Figure 3: Emergency Stop Button

- In the case of not using the programming pendant, be sure to supply the emergency stop button on the equipment. Then before operating the manipulator, check to be sure that the servo power is turned OFF by pressing the emergency stop button. Connect the external emergency stop button to the 5-6 pin and 16-17 pin of the robot system signal connector (CN2).

- Upon shipment of the FS100, this signal is connected by a jumper cable in the dummy connector. To use the signal, make sure to supply a new connector, and then input it.

If the signal is input with the jumper cable connected, it does not function, which may result in personal injury or equipment damage.

- Once the emergency stop button is released, clear the cell of all items which could interfere with the operation of the manipulator. Then turn the servo power ON.

Injury may result from unintentional or unexpected manipulator motion.

Figure 4: Release of Emergency Stop Button

- Observe the following precautions when performing teaching operations within the manipulator’s operating range:
  - View the manipulator from the front whenever possible.
  - Always follow the predetermined operating procedure.
  - Ensure that you have a safe place to retreat in case of emergency.

Improper or unintended manipulator operation may result in injury.
Definition of Terms Used In this Manual

The MOTOMAN is the YASKAWA industrial robot product.

The MOTOMAN usually consists of the manipulator, the controller, the programming pendant, and supply cables.

In this manual, the equipment is designated as follows:

<table>
<thead>
<tr>
<th>Equipment</th>
<th>Manual Designation</th>
</tr>
</thead>
<tbody>
<tr>
<td>DX100, DX200, FS100 or YRC1000 Controller</td>
<td>Controller</td>
</tr>
<tr>
<td>DX100, DX200, FS100 or YRC1000 Programming Pendant</td>
<td>Programming Pendant</td>
</tr>
<tr>
<td>Cable between the manipulator and Controller</td>
<td>Manipulator Cable</td>
</tr>
<tr>
<td>FS100 Programming Pendant Dummy Connector</td>
<td>Programming pendant dummy connector</td>
</tr>
</tbody>
</table>

CAUTION

- Perform the following inspection procedures prior to conducting manipulator teaching. If problems are found, repair them immediately, and be sure that all other necessary processing has been performed.
  - Check for problems in manipulator movement.
  - Check for damage to insulation and sheathing of external wires.
- Always return the programming pendant to the hook on the DX100, DX200, FS100 or YRC1000 cabinet after use.
  The programming pendant can be damaged if it is left in the P-point maximum envelope of the manipulator, on the floor, or near fixtures.
- Read and understand the Explanation of Warning Labels in the DX100, DX200, FS100 or YRC1000 Instructions before operating the manipulator.
Description of the Operation Procedure

Descriptions of the programming pendant keys, buttons, and displays are shown as follows:

<table>
<thead>
<tr>
<th>Equipment</th>
<th>Manual Designation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programming Pendant</td>
<td></td>
</tr>
<tr>
<td>Character Keys</td>
<td>The keys which have characters printed on them are denoted with [ ]. ex. [ENTER]</td>
</tr>
<tr>
<td>Symbol Keys</td>
<td>The keys which have a symbol printed on them are not denoted with [ ] but depicted with a small picture. ex. page key The cursor key is an exception, and a picture is not shown.</td>
</tr>
<tr>
<td>Axis Keys Number Keys</td>
<td>&quot;Axis Keys&quot; and &quot;Number Keys&quot; are generic names for the keys for axis operation and number input.</td>
</tr>
<tr>
<td>Keys pressed simultaneously</td>
<td>When two keys are to be pressed simultaneously, the keys are shown with a &quot;+&quot; sign between them, ex. [SHIFT]+[COORD]</td>
</tr>
<tr>
<td>Displays</td>
<td>The menu displayed in the programming pendant is denoted with { }. ex. {JOB}</td>
</tr>
</tbody>
</table>

Description of the Operation Procedure

In the explanation of the operation procedure, the expression “Select • • •” means that the cursor is moved to the object item and the SELECT key is pressed, or that the item is directly selected by touching the screen.

Registered Trademark

In this manual, names of companies, corporations, or products are trademarks, registered trademarks, or brand names for each company or corporation. The indications of (R) and TM are omitted.
Safeguarding Tips

All operators, programmers, maintenance personnel, supervisors, and anyone working near the system must become familiar with the operation of this equipment. All personnel involved with the operation of the equipment must understand potential dangers of operation. General safeguarding tips are as follows:

- Improper operation can result in personal injury and/or damage to the equipment. Only trained personnel familiar with the operation of this equipment, the operator's manuals, the system equipment, and options and accessories should be permitted to operate this equipment.
- Improper connections can damage the equipment. All connections must be made within the standard voltage and current ratings of the equipment.
- The system must be placed in Emergency Stop (E-Stop) mode whenever it is not in use.
- In accordance with ANSI/RIA R15.06-2012, section 4.2.5, Sources of Energy, use lockout/tagout procedures during equipment maintenance. Refer also to Section 1910.147 (29CFR, Part 1910), Occupational Safety and Health Standards for General Industry (OSHA).

Mechanical Safety Devices

The safe operation of this equipment is ultimately the users responsibility. The conditions under which the equipment will be operated safely should be reviewed by the user. The user must be aware of the various national codes, ANSI/RIA R15.06-2012 safety standards, and other local codes that may pertain to the installation and use of this equipment.

Additional safety measures for personnel and equipment may be required depending on system installation, operation, and/or location. The following safety equipment is provided as standard:

- Safety barriers
- Door interlocks
- Emergency stop palm buttons located on operator station

Check all safety equipment frequently for proper operation. Repair or replace any non-functioning safety equipment immediately.
Programming, Operation, and Maintenance Safety

All operators, programmers, maintenance personnel, supervisors, and anyone working near the system must become familiar with the operation of this equipment. Improper operation can result in personal injury and/or damage to the equipment. Only trained personnel familiar with the operation, manuals, electrical design, and equipment interconnections of this equipment should be permitted to program, or maintain the system. All personnel involved with the operation of the equipment must understand potential dangers of operation.

• Inspect the equipment to be sure no potentially hazardous conditions exist. Be sure the area is clean and free of water, oil, debris, etc.

• Be sure that all safeguards are in place. Check all safety equipment for proper operation. Repair or replace any non-functioning safety equipment immediately.

• Check the E-Stop button on the operator station for proper operation before programming. The equipment must be placed in Emergency Stop (E-Stop) mode whenever it is not in use.

• Back up all programs and jobs onto suitable media before program changes are made. To avoid loss of information, programs, or jobs, a backup must always be made before any service procedures are done and before any changes are made to options, accessories, or equipment.

• Any modifications to the controller unit can cause severe personal injury or death, as well as damage to the robot! Do not make any modifications to the controller unit. Making any changes without the written permission from YASKAWA will void the warranty.

• Some operations require standard passwords and some require special passwords.

• The equipment allows modifications of the software for maximum performance. Care must be taken when making these modifications. All modifications made to the software will change the way the equipment operates and can cause severe personal injury or death, as well as damage parts of the system. Double check all modifications under every mode of operation to ensure that the changes have not created hazards or dangerous situations.

• This equipment has multiple sources of electrical supply. Electrical interconnections are made between the controller and other equipment. Disconnect and lockout/tagout all electrical circuits before making any modifications or connections.

• Do not perform any maintenance procedures before reading and understanding the proper procedures in the appropriate manual.

• Use proper replacement parts.

• Improper connections can damage the equipment. All connections must be made within the standard voltage and current ratings of the equipment.
Maintenance Safety

Turn the power OFF and disconnect and lockout/tagout all electrical circuits before making any modifications or connections.

Perform only the maintenance described in this manual. Maintenance other than specified in this manual should be performed only by Yaskawa-trained, qualified personnel.

Summary of Warning Information

This manual is provided to help users establish safe conditions for operating the equipment. Specific considerations and precautions are also described in the manual, but appear in the form of Dangers, Warnings, Cautions and Notice.

It is important that users operate the equipment in accordance with this instruction manual and any additional information which may be provided by Yaskawa. Address any questions regarding the safe and proper operation of the equipment to Yaskawa Customer Support.
**Customer Support Information**

If you need assistance with any aspect of your MotoPlus SDK for Visual Studio system, please contact YASKAWA Customer Support at the following 24-hour telephone number:

(937) 847-3200

For **routine** technical inquiries, you can also contact YASKAWA Customer Support at the following e-mail address:

techsupport@motoman.com

When using e-mail to contact YASKAWA Customer Support, please provide a detailed description of your issue, along with complete contact information. Please allow approximately 24 to 36 hours for a response to your inquiry.

**NOTE**

Please use e-mail for **routine** inquiries only. If you have an urgent or emergency need for service, replacement parts, or information, you must contact YASKAWA Customer Support at the telephone number shown above.

Please have the following information ready before you call Customer Support:

- **System**: MotoPlus SDK for Visual Studio
- **Robots**: ___________________________
- **Primary Application**: ___________________________
- **Controller**: DX100, DX200, FS100 or YRC1000
- **Software Version**: Access this information on the Programming Pendant's LCD display screen by selecting {MAIN MENU} - {SYSTEM INFO} - {VERSION}
- **Robot Serial Number**: Located on the robot data plate
- **Robot Sales Order Number**: Located on the controller data plate
# Table of Contents

1 Outline ............................................................................................................................................ 1-1  
  1.1 Introduction................................................................................................................................. 1-1  
  1.2 What MotoPlus Can Do ............................................................................................................. 1-1  
    1.2.1 Supported Services ............................................................................................................. 1-1  
  1.3 Development Process.............................................................................................................. 1-3  
  1.4 Notes ........................................................................................................................................ 1-4  

2 SDK Installation and Setup................................................................................................. ............ 2-1  
  2.1 System Requirements ............................................................................................................. 2-1  
  2.2 Installation of SDK .................................................................................................................. 2-2  
  2.3 Setup of Robot Controller .................................................................................................... 2-3  
    2.3.1 Install VGA Adapter Cable for Debugging ........................................................................ 2-3  
    2.3.2 Enabling MotoPlus ............................................................................................................ 2-5  
  2.4 Enable Telnet for Debugging ............................................................................................... 2-7  

3 Creating MotoPlus Application.............................................................................................. .......... 3-1  
  3.1 Rules for Application Source Code....................................................................................... 3-1  
  3.2 Creating a MotoPlus Project in Visual Studio ..................................................................... 3-2  
  3.3 Adding Files to the Project..................................................................................................... 3-3  
  3.4 Creating a MotoPlus Library .................................................................................................. 3-4  
  3.5 Using a MotoPlus Library In Your Project.......................................................................... 3-6  
  3.6 Building Your Application...................................................................................................... 3-7  
  3.7 Building Your Project for Multiple Controller Platforms.................................................. 3-10  

4 Installation and Startup of MotoPlus Application..................................................................... 4-1  
  4.1 Installation of Application Program in Maintenance Mode................................................ 4-1  
  4.2 Start-up of Application Program............................................................................................ 4-6  
  4.3 To Temporarily Prevent MotoPlus Application from Starting ......................................... 4-6  

5 Debugging ...................................................................................................................................... 5-1  
  5.1 Debugging Environment ........................................................................................................ 5-1  
  5.2 Commands for Printing to Standard-output (stdout)......................................................... 5-1  
  5.3 Viewing Output on Computer Monitor................................................................................... 5-1
# Table of Contents

## 5.4 Viewing Output on PC Console Window
- 5.4.1 Setup of Robot Controller ........................................................................5-2
- 5.4.2 Establishing Telnet Connection .................................................................5-3
  - 5.4.2.1 Automatic Login to Telnet Shell .........................................................5-3
  - 5.4.2.2 Manual Login to Telnet Shell ...............................................................5-4

## 5.5 Development Support Tools for YRC1000
- 5.5.1 Introduction ...............................................................................................5-5
- 5.5.2 Overview of Functions ................................................................................5-5
- 5.5.3 Using the Support Tools .............................................................................5-6
- 5.5.4 Basic Settings .............................................................................................5-7
  - 5.5.4.1 Connecting a PC to a YRC1000 ...........................................................5-7
  - 5.5.4.2 Communication Settings .......................................................................5-7
- 5.5.5 Online Download .......................................................................................5-8
- 5.5.6 System Settings ..........................................................................................5-8
- 5.5.7 Inserting the USB Memory in the Controller ..............................................5-9
  - 5.5.7.1 Operating Procedure ...........................................................................5-10
- 5.5.8 Application Property List ..........................................................................5-11
  - 5.5.8.1 Tool Operating Procedure: ................................................................5-11
  - 5.5.8.2 How to Set Application Information ......................................................5-12
  - 5.5.8.3 Setting Procedures ................................................................................5-12
- 5.5.9 Port Debug ..................................................................................................5-14
  - 5.5.9.1 Measuring Procedures ..........................................................................5-15
  - 5.5.9.2 Optional for Measuring ..........................................................................5-15
  - 5.5.9.3 Registering Data for the Port Debug Function .......................................5-17
  - 5.5.9.4 Debug Port APIs ..................................................................................5-18
- 5.5.10 Copy Protection .......................................................................................5-22
  - 5.5.10.1 Protection Activation ............................................................................5-22
  - 5.5.10.2 Acquiring the Serial File (MotoPlus Monitor) .......................................5-23
  - 5.5.10.3 Generating the Protection Key ..............................................................5-24
  - 5.5.10.4 Loading the Protection Key (MotoPlus Monitor) ..................................5-24

## 6 Alarm List ...........................................................................................................6-1

## 7 Advanced mpBuilder Usage .............................................................................7-1
- 7.1 Visual Studio MakeFile Project ......................................................................7-1
- 7.2 Compiler Arguments ......................................................................................7-3
  - 7.2.1 CompilerArguments.mps .........................................................................7-3
  - 7.2.2 LinkerArguments.mps ..............................................................................7-3
# Table of Contents

**MotoPlus SDK for Visual Studio**

Appendix A .................................................................................................................. A-1
  A.1 Compatibility with Windows 8/10 ....................................................................... A-1
    A.1.1 Symptom of Problem: .................................................................................... A-1

Appendix B ..................................................................................................................... B-1
  B.1 Compatibility with Visual Studio 2015 ................................................................. B-1
    B.1.1 Symptom of Problem: .................................................................................... B-1
1 Outline

1.1 Introduction

MotoPlus Software Development Kit (SDK) is used to develop customer specific applications that operate on the main CPU board of the YASKAWA robot controller. The application is run concurrently to the robot firmware, in real time. Programming of the application is done using C language with Microsoft Visual Studio. Users can develop a variety of application programs on their own without modifying the source code of the robot controller.

This advanced toolset is intended only for experienced software developers who are familiar with the C programming language.

1.2 What MotoPlus Can Do

Users can use the C language, a general programming language, and MotoPlus libraries to develop a customer specific application program in which various types of resources are used, e.g., the Ethernet communication port, the RS232C serial communication port, and the programming pendant of the robot controller.

1.2.1 Supported Services

MotoPlus SDK provides users with a variety of services as follows:

- **Application task control**
  Application task (thread) start/stop, data transmission between application programs by the mailbox and semaphore, exclusive control and execution synchronization

- **Robot control**
  Robot control from the application program

- **JOB control**
  Start-stop control of the JOB by the application, data transmission between Jobs by variables (byte, integer, double-precision, floating point, character string, and position variable), and execution synchronization

- **CIO control**
  I/O between the application program and the robot CIO including read/write access to registers

- **Ethernet communication control**
  Industry standard Ethernet (TCP/IP) communication available through standard sockets

- **Programming pendant communication**
  Data transmission between the application and the programming pendant application

- **Robot event notification**
  Event notification to the application at every I/O control cycle and interpolation control cycle of the system

- **RS232C serial communication control**
  RS232C serial communication from the application
1.2 What MotoPlus Can Do

- **Sensor control**
  Robot path correction, adjustment, and control can be achieved while reacting to external sensors

- **Memory management**
  Dynamic and static memory allocation

- **General-purpose file control (DX200, FS100 or YRC1000 only)**
  Interface with custom data files to save/load application settings

- **Existing file control (DX200, FS100 or YRC1000 only)**
  Interface with standard robot files (DAT, CND, JBI, etc…)

- **Servo control**
  Control servo power for automated robot operation

- **User watchdog**
  Watchdog to monitor whether the application operates normally

- **Coordinate conversion (DX200, FS100 or YRC1000 only)**
  Forward and inverse kinematic calculations available for position and frame calculations
1.3 Development Process

**Develop**
- Write application source code in C language
- Compile using MotoPlus plugin for Microsoft Visual Studio on Windows PC

**Install**
- Copy compiler output file to CF/USB memory
- Boot robot controller into MAINTENANCE mode
- Load application executable using pendant menu

**Run**
- Application is automatically loaded and started during robot bootup

**Debug**
- Monitor application progress:
  - Robot I/O and variables
  - VGA monitor connection to view "print" statements (DX controller only)
  - Telnet connection to view "print" statements in console window

**Deploy**
- Remove debug output for maximum performance
- Load MotoPlus application onto robots during integration
  - OR- Provide MotoPlus library to other developers without exposing source code
1.4 Notes

- **C language allows for dynamic memory operations.** When programming in C language, pointers with memory addresses may be used. If the pointer variable is set incorrectly, the system memory area may be rewritten. If the system memory area is rewritten, critical problems occur, e.g., the software hangs up (the hang-up status is detected by a CPU exception or the watchdog check function, then the servo power turns OFF and the system shuts down), the robot stops its operation due to an alarm, or the programming pendant becomes inoperable. If the system memory operation is incorrect, the application itself may not operate as it is designed.

- **The task “mpUsrRoot”, which starts when the power is turned ON, has a high priority to start other application tasks and initialize the entire application quickly.** Due to its high task priority, if a time-consuming process (0.1 milliseconds or longer) is done in this task, the processing time for the robot control becomes insufficient. Then system alarms may occur or the programming pendant or the HOLD button may freeze. Thus, as shown in the sample program (refer to Task Control Sample Program” in “Programmer’s Manual for New Language Environment MotoPlus”, make sure to complete the task “mpUsrRoot” after starting up another application or creating a semaphore.

- **Debugging via Telnet**
  1. After completing debugging, set the Telnet function to INVALID (S2C1119 = 0). This is because the Telnet function consumes system processing time, and an alarm may occur due to insufficient processing time for robot operation.
  2. With the Telnet function, debug only tasks with normal priority. For debugging the tasks with high priority (the tasks started with the task priority: MP_PRI_IO_CLK_TAKE, MP_PRI_IP_CLK_TAKE, or MP_TIME_CRITICAL, or task “mpUsrRoot”), use the robot variables visible on the programming pendant. This is because, if printf() or puts() is done with a high priority task, processing time for robot operation control is consumed, and a system alarm may occur due to insufficient processing time.
  3. After completing debugging, make sure to disable the debugging code using printf() or puts(). This is because the processing time of printf() and puts() influences the system, and an error may occur such as insufficient processing time during robot operation.
  4. When using VxWorks SHELL commands, which become available by connecting with the controller via Telnet, a system alarm may occur due to insufficient processing time for robot operation. Do NOT use the VxWorks SHELL commands.
2 SDK Installation and Setup

2.1 System Requirements

The following requirements must be met for the host computer used to develop the MotoPlus application:

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC with Microsoft Windows</td>
<td>Windows XP SP2, Vista, 7, 8, or 10</td>
</tr>
<tr>
<td>.Net Framework</td>
<td>3.5</td>
</tr>
<tr>
<td>USB Hardware Key Dongle</td>
<td>Included with SDK package</td>
</tr>
<tr>
<td>Microsoft Visual Studio</td>
<td>2008 Standard, Professional</td>
</tr>
<tr>
<td></td>
<td>2010 Professional, Premium, Ultimate</td>
</tr>
<tr>
<td></td>
<td>Visual C++ 2010 Express (free download from Microsoft)</td>
</tr>
<tr>
<td></td>
<td>2012 Professional, Premium, Ultimate</td>
</tr>
<tr>
<td></td>
<td>2012 Express for Windows Desktop (free download from Microsoft)</td>
</tr>
<tr>
<td></td>
<td>2013 Professional, Premium, Ultimate</td>
</tr>
<tr>
<td></td>
<td>2013 Express for Windows Desktop (free download from Microsoft)</td>
</tr>
<tr>
<td></td>
<td>2015 Professional, Enterprise</td>
</tr>
<tr>
<td></td>
<td>2015 Community (free download from Microsoft)</td>
</tr>
<tr>
<td></td>
<td>2017 Professional, Enterprise</td>
</tr>
<tr>
<td></td>
<td>2017 Community (free download from Microsoft)</td>
</tr>
<tr>
<td></td>
<td><em>Not compatible with “Visual Studio Code”</em>.</td>
</tr>
<tr>
<td>External memory card to load on robot pendant</td>
<td>Compact Flash (CF) or USB Flash (USB)</td>
</tr>
<tr>
<td>User Rights on PC</td>
<td>Administrator rights required</td>
</tr>
</tbody>
</table>
2.2 Installation of SDK

1. Insert the MotoPlus SDK for Visual Studio disc into your Windows PC. If AutoRun is enabled, the installation will automatically start. Otherwise, run "setup.exe" from the root of the DVD.

![Installation Screen]

2. Click [Next] and continue through the installation.

3. When installation is complete, you must reboot the computer to register system environment variables required by the compiler.

4. After restarting the PC, please install the Sentinel driver located on the root of your installation DVD. This is required for the USB hardware key dongle.

![Sentinel Driver Screen]

5. Insert the USB hardware key dongle into your PC.

Installation of the SDK is now complete.

**NOTE**
Do not insert the USB hardware key dongle into the computer until after installing the Sentinel driver.

**NOTE**
The license agreement is located on the root of your installation DVD. Please read this prior to installing the software. You will be required to agree to the terms of the licensing.

**NOTE**
If you are installing the Sentinel driver on a notebook/laptop computer, it is recommended to first remove the computer from any docking station(s) prior to installation.
2.3 Setup of Robot Controller

2.3.1 Install VGA Adapter Cable for Debugging (DX100, DX200 and YRC1000 controllers only)

For DX100, DX200 and YRC1000 controllers, the SDK package comes with a VGA adapter cable. This cable allows you to connect a standard VGA computer monitor to view program output. It should only be used for application debugging purposes and not shipped with a production robot controller.

**CAUTION**

Static electricity can severely damage sensitive electronics. Follow standard ESD safe procedures when handling the DX CPU board!

1. Power off the robot controller.
2. Locate and remove the CPU board in the controller chassis. See Table for board number. The board is secured with two screws on top and bottom. You may also have to temporarily disconnect all cables going to connectors CN103, CN104, CN105 and CN106.

<table>
<thead>
<tr>
<th>Controller</th>
<th>CPU Board Part Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>DX100</td>
<td>JANCD-YCP01</td>
</tr>
<tr>
<td>DX200</td>
<td>JANCD-YCP21</td>
</tr>
<tr>
<td>YRC1000</td>
<td>ACP01</td>
</tr>
</tbody>
</table>
3. Connect the VGA adapter cable to the pin header on the CPU board and route the cable out of the chassis. If equipped carefully open the plastic tab which is securing the opening and install the cable.

4. Reinsert the CPU board into the controller chassis, tighten the screws securing the board, and reconnect all cables removed in the previous steps.
2.3.2 Enabling MotoPlus (DX200, FS100 or YRC1000 only)

On DX200, FS100 or YRC1000 controllers, the MotoPlus function must be enabled in MAINTENANCE mode.

1. Boot the robot controller in MAINTENANCE mode.

2. Upgrade to MANAGEMENT level security. Touch [System] > [Setup].

3. Select "OPTION FUNCTION".
2.3 Setup of Robot Controller

4. Set “MotoPlus FUNC.” to “USED”. Press {ENTER} to save your change. If you are asked to “Initialize related file SRAM.DAT”, touch [YES].

5. Set “MOTOMAN DRIVER” to “USED”.
   - Make sure to set this AFTER “MotoPlus FUNC.”. If these are set in the wrong order, please set both functions to “NOT USED” and then toggle them back to “USED” in the correct order.

   **NOTE**
   The MOTOMAN DRIVER option is only available if the controller has the following software version or greater:

   - FS100: FS3.30.00-00
   - DX200: DN2.21.00-00
   - YRC1000: YAS1.11.00-00
2.4 Enable Telnet for Debugging

One of the options for debugging is to connect to the robot using a telnet connection. (See Chapter 5 “Debugging”).

On Windows 7, the default telnet console is not enabled by default. This must be enabled prior to debugging.

1. Click the Windows START orb, and type “programs and features”, then press [Enter].

2. From the menu on the left, select “Turn Windows features on or off”.

3. In the Windows Features list, check the box next to “Telnet Client” and click [OK].
Creating MotoPlus Application

3.1 Rules for Application Source Code

Be sure to observe the following rules when creating source code.

Refer to the MotoPlus Programmer’s manual for detailed information on programming configuration and sample programs.

Note: Sample source code files are located in your installation directory and on the MotoPlus for Visual Studio installation disc.

Please remember that the source code should follow ANSI-C syntax. The only exception is that users should not define a “main (...)” function. This is replaced by “mpUsrRoot(...)”.

1. Up to 32 one-byte characters can be used for a file name. Up to 16 two-byte characters can also be used. File names are case-sensitive. Set ‘.c’ as the file extension. (C++ library is not fully supported on the MotoPlus platform)

2. Put “#include <MotoPlus.h>” at the top of each source code file. When “MotoPlus.h” is included, all services provided by MotoPlus are available.

3. MotoPlus applications do not have a “main” function. This is replaced by “mpUsrRoot”. This function must have the following signature:

   void mpUsrRoot (int arg1, int arg2, int arg3, int arg4, int arg5, int arg6, int arg7, int arg8, int arg9, int arg10)

4. mpExitUsrRoot must be called at the end of mpUsrRoot.

5. The task “mpUsrRoot” is launched at a high task priority. Due to its high task priority, if a time-consuming process (0.1 milliseconds or longer) is done in this task, the processing time for the robot control becomes insufficient. This can interrupt robot startup and freeze the pendant. Therefore, you should immediately launch a new normal priority task from mpUsrRoot and then call mpExitUsrRoot.

Note: When creating a new MotoPlus project in Visual Studio, a “main” template is automatically generated.
3.2 Creating a MotoPlus Project in Visual Studio

Open Visual Studio on your development PC. Select [File] > [New Project]. The MotoPlus project is found under {Templates} > {Other Languages} > {Wizards}.

Click [OK] and a new project is created in the specified location. This project is automatically configured to build using the MotoPlus compiler. A template source file (mpMain.c) is added to the project. Also included are files which specify the project-specific arguments to the compiler and linker.

**NOTE** Do not include any spaces when naming your project.

Click [OK] and a new project is created in the specified location. This project is automatically configured to build using the MotoPlus compiler. A template source file (mpMain.c) is added to the project. Also included are files which specify the project-specific arguments to the compiler and linker.

**NOTE** The .mps files must not be removed from your MotoPlus project.
3.3 Adding Files to the Project

Files can be added to your MotoPlus project through the standard Visual Studio interface. Right-click on the project tree and select {Add} > {New Item…} or {Add} > {Existing Item…}. Alternatively, you can perform the same action from the {Project} dropdown in the Main Menu.

When adding new items to the project, the file must end in a supported extension. Header files should have a ".h" extension. Source files should have a ".c" extension.

**NOTE**

C++ (.cpp) files are not supported by the MotoPlus environment.

Files for the project do not have to be in a flat directory. They can be in any folder relative to the Project Directory.
3.4 Creating a MotoPlus Library

As a MotoPlus developer, you can share routines by distributing a compiled MotoPlus library. This library can be used by other developers without exposing proprietary source code. Instead of building an executable module (.out), this will build a library without the main entry point for execution.

A MotoPlus library is created in the same manner as a MotoPlus project. From the New Project window, select the MotoPlusLibrary wizard.

**NOTE**

MotoPlus library files end in extension .mpLib (DX100),.fsLib (FS100), .dnLib (DX200) and .yrcLib (YRC1000). Libraries must be compiled for each controller separately.
A project is generated with a template source file (mpLibrary.c) and header file (mpLibrary.h) to get you started.

The project is automatically configured to build a library instead of executable file.

When distributing a MotoPlus library, be sure to include a header file which provides functional declarations for each public function in your library. Function declarations should begin with “extern”. Example:

```c
extern void mpLibraryFunction1();
```
3.5 Using a MotoPlus Library In Your Project

MotoPlus library files can be added to your project like any other file. Right-click the project tree and select {Add} > {Existing Item…}.

The library developer should have also provided a header file with the declarations for all public functions in the library. Each declaration should begin with “extern”. Example:

```c
extern void mpLibraryFunction1();
```

If your application is targeted toward multiple controller platforms, you must add a library for each controller type. When building the project, the compiler will ignore libraries which are targeted for a different controller platform.

**NOTE** MotoPlus library files end in extension .mpLib (DX100), .fsLib (FS100), .dnLib (DX200) and .yrcLib (YRC1000).
3.6 Building Your Application

The MotoPlus application (or library) can be built directly from within Visual Studio. The project is automatically configured to pass all files in the Solution Explorer to the MotoPlus compiler.

First select your target controller platform from the Configuration Manager dropdown.

NOTE
You must have a license key for each target controller platform. Upgrade licenses are available for each controller type.

Next, select the appropriate Build option from the (Build) menu.
If there are any errors/warnings in your source files, they will be displayed in the Visual Studio Error List. You can double-click the error entry to go directly to the problematic line of code.

The Output pane contains the full output text of the compiler.
If there are no errors in the code, the Output pane will indicate “Build Successful!” By default, the compiled module will be located in the folder:

<Solution Directory> \ <Controller Platform> \ <Project name>

NOTE
The output directory and file name can be customized by modifying the NMake Configuration Properties. See Chapter 7 “Advanced mpBuilder Usage” for more details.
3.7 Building Your Project for Multiple Controller Platforms

The MotoPlus for Visual Studio SDK allows you to build your source code for each of the MotoPlus-enabled YASKAWA robot controllers. This allows you to use the same source code project for each controller. However, there are minor differences between each controller which may affect your application. For example, the Network Inputs I/O addresses start at #25010 for DX100 and FS100. However, the starting address is #27010 on the DX200 and YRC1000.

For this reason, the build environment has been configured to automatically provide a pre-processor definition for each controller type. Using preprocessor directives, you can put a decision statement around your code at compile-time, instead of run-time.

Example:

```c
//evaluated at compile-time
#if (DX100 || FS100)
    #define STARTING_NETIN_ADDRESS 25010
#elif (DX200)
    #define STARTING_NETIN_ADDRESS 27010
#endif

void myFunction()
{
    //set at run-time
    int startingIoAddress = STARTING_NETIN_ADDRESS;
}
```

NOTE
You must have a license key for each target controller platform. Upgrade licenses are available for each controller type.
4 Installation and Startup of MotoPlus Application

4.1 Installation of Application Program in Maintenance Mode

1. Load Function

Load the application program and install by following the instructions below.

(1) Boot the robot controller into MAINTENANCE mode by holding [Main Menu] on the programming pendant.

(2) Set the security mode to MANAGEMENT mode.
Select the Main Menu {MotoPlus APL.}. Then {LOAD (USER APPLICATION)}, {FILE LIST}, {DELETE}, {DEVICE}, {FOLDER}, and {MotoPlus FUNC. SETTING} appear as submenus.

(3) Specify the file location.
Select an appropriate file location with the submenus {DEVICE (to select CF/USB)} and {FOLDER (to select the folder which has the application program)}. 
(4) Select and load a file.

- Select the submenu {LOAD (USER APPLICATION)}. Then, ".out" files in the specified folder appear as shown below.

- Move the cursor and press [SELECT]. Then, the selection indicator "★" appears on the left of the application file name. Press [SELECT] again to clear "★".

- Press [ENTER], then the following confirmation dialog box appears.
4.1 Installation of Application Program in Maintenance Mode

- When {YES} is selected and the selected application file contains the file which already exists in the controller, the following confirmation dialog box appears. Select “YES” to load.

2. List
   Select the submenu {FILE LIST}. Then, the list of the application files which already exist in the controller appears.

3. Deletion of application program
   By the submenu {DELETE}, the application file which already exists in the controller can be deleted.
   
   (1) Move the cursor and press [SELECT]. Then, the selection indicator “★” appears on the left of the application file name.
   Press [SELECT] again to clear “★”.

- When {YES} is selected and the selected application file contains the file which already exists in the controller, the following confirmation dialog box appears. Select “YES” to load.
4. Installation and Startup of MotoPlus Application

4.1 Installation of Application Program in Maintenance Mode

1. MotoPlus Function Setting
When selecting “MotoPlus FUNC. SETTING”, the following windows appear to enable or disable the autostart of the MotoPlus application program. For the detailed usage methods, refer to section 4.2 “Start-up of Application Program” on page 4-6 and section 4.3 “To Temporarily Prevent MotoPlus Application from Starting” on page 4-6.

(1) Move the cursor and press [SELECT]. The setting is changed from “ENABLE” to “DISABLE”.

NOTE
This screen may have different options based on which robot controller is being used. Please refer to the Programmer’s Manual for more details.
4. Installation and Startup of MotoPlus Application

4.1 Installation of Application Program in Maintenance Mode

(2) Press [ENTER], and then the confirmation dialog box appears.

![Confirmation Dialog Box]

(3) Select {YES}.

–When {YES} is selected on the confirmation dialog box, the parameter is set automatically, and then it returns to the MotoPlus setting window.

![Setting Window]
4.2 Start-up of Application Program

When the controller is turned ON, the application program is loaded automatically and started.

<Limitations>

- Only one "*.out" file can be loaded. Thus, install only one application. If two or more applications are installed, the alarm 1020: "MOTOPLUS APPLICATION LOAD ERROR [1]" occurs on loading. The error status can be checked by reading $B051 from the job.
- The loadable memory size is up to 2 Mbyte as the total of the code area and the static memory area. If it is more than 2 Mbyte, the start-up fails and the alarm 1020: "MOTOPLUS APPLICATION LOAD ERROR [4]" occurs on loading. The error status can be checked by reading $B051 from the job.

$B051 0: The application is loaded successfully on start-up.
1: The number of files exceeds the limit.
2: The memory is insufficient. (Available memory area is less than 2 Mbyte.)
3: APPLICATION folder cannot be found in the CF.
4: The size of the MotoPlus application exceeds the limit.

4.3 To Temporarily Prevent MotoPlus Application from Starting

Due to a problem of the application program, the system may hang up during robot operation, and then the controller may not start normally. For recovery, it is necessary to start the controller normally while preventing the application program from operating. In this case, it is necessary to start the controller in the MAINTENANCE mode, change the settings so that the application program does not operate, start the controller normally, and operate the robot. Follow one of the following two procedures:

1. Start the controller in the MAINTENANCE mode, set the security mode to the MANAGEMENT MODE. Select "MotoPlus APL.", then "DELETE" to delete the application.
2. Start the controller in the MAINTENANCE mode, set the security mode to the MANAGEMENT MODE. Select "MotoPlus APL.", then "MotoPlus FUNC. SETTING" to show the setting window. Then set "APPLI. AUTOSTART AT POWER ON" to "DISABLE".

If the above problem occurs during robot operation, choose procedure 2. The above problem during robot operation may be caused by a specific condition. If the MotoPlus application is deleted with procedure 1, after the specific condition is cleared, the MotoPlus application must be loaded again in the CF for recovery. In this case, if the application to be loaded does not exist there, the application cannot operate and the system cannot recover.
5  Debugging

5.1 Debugging Environment

Due to the nature of the real-time robotic system, MotoPlus applications cannot be debugged using a traditional source-level debugger. The most effective way to debug a MotoPlus application is to monitor application output. This is done by setting variables and I/O to view on the pendant, or by printing to standard output (stdout) within your application.

Print-statements sent to stdout can be viewed by using one of two methods. On DX and YRC1000 robot controllers, your SDK comes with a VGA adapter cable to connect a standard computer monitor to the robot controller. Alternatively, you can connect to the robot via telnet to view output within a console window on your PC.

NOTE
Print-statements are extremely time consuming during execution. It is highly recommended to put a debug-switch around all print statements so that they can be disabled for final release.

5.2 Commands for Printing to Standard-output (stdout)

The commands "puts" and "printf" are available to print to stdout. The command "puts" will print a string and automatically append a new line. The command "printf" is used to print a formatted string with variable arguments. The line-feed is not automatically appended to printf strings.

5.3 Viewing Output on Computer Monitor (DX100, DX200 and YRC1000)

Please see section 2.3.1 “Install VGA Adapter Cable for Debugging” on page 2-3 for instructions on installing the VGA adapter cable included with the MotoPlus SDK.

To view output using the monitor, the monitor must be connected prior to turning on the robot controller. If the monitor is blank, please cycle power on the robot controller.
5.4 Viewing Output on PC Console Window

5.4.1 Setup of Robot Controller

Please see section 2.4 “Enable Telnet for Debugging” on page 2-7 for instructions on enabling the Windows Telnet Client in Windows 7.

Set robot parameter S2C1119 = 2 to enable telnet output. The robot must be rebooted after setting this parameter.

When debugging is completed, be sure to set S2C1119 = 0. This will prevent the telnet task from interfering with robot operation.

Connect your development PC to the robot controller LAN port via Ethernet connection.

- **DX and YRC1000 Controller**
  - **Without a Hub**
  - **With a Hub**

- **FS Controller**
  - **Without a Hub**
  - **With a Hub**
5.4.2 Establishing Telnet Connection

- Your computer IP address must be on the same subnet as the robot controller.
  - **Example:**
    - Robot IP - 192.168.255.1
    - Computer IP - 192.168.255.xxx

- Alarms such as "0500 SEGMENT PROC NOT READY" may occur when using the telnet connection. This is because the telnet task runs at a high priority in the system and can interfere with robot operation. If the alarm occurs, reboot the robot controller.
  - Set S2C1119 = 0 when you are done debugging to prevent these alarms during normal operation.

5.4.2.1 Automatic Login to Telnet Shell

(DX200, FS100, or YRC1000)

On the controller, an automatic login utility is provided with the MotoPlus for Visual Studio installation. This is accessed from the START menu under Motoman\MotoPlus Debugging.

Prior to opening the automatic login utility, you must input the IP address of the robot controller. An .ini configuration file is located in your MotoPlus for Visual Studio installation directory. (Default: C:\Program Files (x86)\Motoman\MotoPlus for Visual Studio\Telnet Debugging)

Open the .ini file for your target controller. Modify the value of "ipaddress" to match your robot controller. After saving this file, you can now open the automatic login utility.

After opening the automatic login utility, do not click any other windows until the login process is complete. The process is completed when you see the telnet shell prompt “>.”
5.4.2.2 Manual Login to Telnet Shell

(all controllers)

As an alternative to the automatic login utility, you can manually login to the telnet shell.

Open a Windows console window. The easiest method is through the “Run…” command in the START menu.

START > Run… > Type “cmd”

Type “telnet” to open the Windows Telnet Client utility. Then type “open <ipaddress>” to open a connection to the robot. It will prompt you for the VxWorks login information. You have 30 seconds to type in the login username and password before it times out.

<table>
<thead>
<tr>
<th>Controller</th>
<th>Username</th>
<th>Password</th>
</tr>
</thead>
<tbody>
<tr>
<td>DX100</td>
<td>Net_Maint_Mng</td>
<td>999999999</td>
</tr>
<tr>
<td>DX200</td>
<td>MOTOMANrobot</td>
<td>MOTOMANrobot</td>
</tr>
<tr>
<td>FS100</td>
<td>MOTOMANrobot</td>
<td>MOTOMANrobot</td>
</tr>
<tr>
<td>YRC1000</td>
<td>MOTOMANrobot</td>
<td>MOTOMANrobot</td>
</tr>
</tbody>
</table>

If the login is successful, the console window will display the shell prompt “->”.
5.5 Development Support Tools for YRC1000

These tools function with the YRC1000 controller or newer. They will not work with the DX or FS controllers.

5.5.1 Introduction

This chapter explains the functions of the tools to support development of the MotoPlus application (hereafter referred to as “the application”).

5.5.2 Overview of Functions

The MotoPlus application development support tool contains the following functions.

<table>
<thead>
<tr>
<th>No.</th>
<th>Function</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Online Download</td>
<td>A function which downloads the application from Visual Studio to the robot controller (hereafter referred to as “the controller”).</td>
</tr>
<tr>
<td>2</td>
<td>Application Property List</td>
<td>A function which displays the detailed information, such as the operation status of the application.</td>
</tr>
<tr>
<td>3</td>
<td>Port Debug</td>
<td>A function which measures the processing time of the application.</td>
</tr>
<tr>
<td>4</td>
<td>Copy Protection</td>
<td>A function which prevents applications that are copied illegally from being used.</td>
</tr>
</tbody>
</table>
5.5.3 Using the Support Tools

(Visual Studio)

Each function is available from the “Tools” menu in Visual Studio. These tools are also accessible from the Windows Start menu.

Fig. 5-1: External Tools in Visual Studio

(MotoPlus Monitor)

On the programming pendant from Main Menu select {MotoPlus APL.} and then select {MotoPlus MONITOR} to use the functions.

Fig. 5-2: MotoPlus Monitor
5.5.4 Basic Settings

5.5.4.1 Connecting a PC to a YRC1000

Connect the development PC to the YRC1000 controller with a standard Ethernet cable. Depending on which network interface(s) is enabled on the controller, the PC should be connected to CN106 (LAN2) or CN107 (LAN3). If other devices are already connected to the network interface, then an Ethernet switch may be used.

5.5.4.2 Communication Settings

When using any of the development support tools, you must specify the robot to connect to via IP address. The addresses for each of the robot controllers must first be registered using the Communication Settings tool.

From within the Communication Settings tool, simply type a name and IP address for each robot that you will be connecting to. This list of robots will be selectable within each of the development support tools.

Fig. 5-3: Communication Settings

![Communication Settings](image)
5.5.5 Online Download

This function downloads the application from the development PC to the robot controller.

Loading the application is usually performed in the maintenance mode, however, this function allows the application to load while the controller is online. Since this function bypasses the time and effort required for restarting in maintenance mode, the operating time can be reduced when debugging the application. A USB storage drive is required when using this function.

Fig. 5-4: Online Download

5.5.6 System Settings

The following parameter is required to enable the Online Download function.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Setting value</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>S2C1113</td>
<td>1</td>
<td>Enable Online Download function 0: Invalid, 1: Valid</td>
</tr>
</tbody>
</table>

Use this function only when debugging the application. When the application is released, return the parameter to the original value, and then load the application from within maintenance mode as usual.
5.5.7 Inserting the USB Memory in the Controller

This function allows the application file to be saved to the controller. Attach the flash drive to the USB port on the CPU board.

*Fig. 5-5: Location of the USB Port on CPU Board*
5.5.7.1 Operating Procedure

From the “Tools” menu in Visual Studio, select “M+ Load App into Controller”.

NOTE This tool automatically launches when selecting “Start Debugging” [F5].

1. Select the controller to receive the download.
2. Select the application file (.out file) to download. This field automatically populates with the path of the application file which was compiled last.
3. Click [Download].
4. Turn OFF the controller and then turn it back ON. If the “Reset Robot Controller” option is selected, the controller automatically restarts when the download completes.

Fig. 5-6: Online Download Window
5.5.8 Application Property List

Acquires and displays the application properties. The following properties can be displayed.

- Application name
- Application version
- Application file size
- Task name
- Task executing priority

5.5.8.1 Tool Operating Procedure:

- **Using Development PC:** select "M+ App Property List" from the "Tools" menu in Visual Studio.
- **Using Programming Pendant:** select {MotoPlus APL} > {MotoPlus Monitor}. Then choose {Application Property List}.

*Fig. 5-7: Application Property List Window*
5.5.8.2 How to Set Application Information

The following code example demonstrates how to set the Application Information data.

Fig. 5-8: Setting the Application Information

```c
void mpUserRoot(int arg1, int arg2, int arg3, int arg4, int arg5, int arg6, int arg7) {
    int taskId;
    MP_APPINFO_SEND_DATA appInfo;
    MP_SID_RSP_DATA tData;
    sprintf(appInfo.AppName, "testApp1");
    sprintf(appInfo.Version, "1.0.0.0");
    sprintf(appInfo.Comment, "Test");
    mpApplicationInfoNotify(appInfo, tData);
    taskId = mpCreateTask(MP_PRI_TIME_NORMAL, MP_STACK_SIZE, (FUNCPT)mpTask3,
                          arg1, arg2, arg3, arg4, arg5, arg6, arg7, arg8, arg9, arg10);
    mpTaskProphameSet(taskId, "task1_in_inAppl1");
    taskId = mpCreateTask(MP_PRI_TIME_NORMAL, MP_STACK_SIZE, (FUNCPT)mpTask4,
                          arg1, arg2, arg3, arg4, arg5, arg6, arg7, arg8, arg9, arg10);
    mpTaskProphameSet(taskId, "task2_in_inAppl1");
    mpExitUserRoot;
}
```

5.5.8.3 Setting Procedures

1. Set the application name and the version by using `mpApplicationInfoNotify()`. Use the MotoPlus reference manual for details on `mpApplicationInfoNotify()`.

2. Set the task name by using `mpTaskPropNameSet()`.
mpTaskPropNameSet

Set the task name to be displayed in the Application Property List.

- **Syntax**
  ```c
  int mpTaskPropNameSet(
      int taskId,
      char* name
  );
  ```

- **Argument**
  - `[taskId]`
    Specify the task ID that is returned by using `mpCreateTask()`.
  - `[name]`
    Specify the task name. For the task name, use a maximum of 32 characters.

- **Return value**
  - `0`: Normal end
  - `-1`: Error

**NOTE**

When the information is not set, the default setting value is displayed.
5.5.9 Port Debug

The Port Debug tool measures code execution, similar to an oscilloscope function. The data for this function is set using special MotoPlus API functions for debugging (mpDebugPortXXXX).

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Time Division</td>
<td>Shows the unit [ms/div] of the time axis (the horizontal axis).</td>
</tr>
<tr>
<td>2</td>
<td>Port 0 - Port 4</td>
<td>Shows the measuring port. Any name can be set by using the debug port API (for details, refer to section 5.5.9.3 “Port Debug”).</td>
</tr>
<tr>
<td>3</td>
<td>Graph Area</td>
<td>Shows the measuring data. Displays the time which is set in the Time Range.</td>
</tr>
<tr>
<td>4</td>
<td>Org Position</td>
<td>Optional for measuring. Sets the home position of the measuring data. (For details refer to section 5.5.9.2 “Port Debug”.)</td>
</tr>
<tr>
<td>5</td>
<td>Time Range</td>
<td>Sets the displaying time [ms] for measuring data.</td>
</tr>
<tr>
<td>6</td>
<td>Port Number</td>
<td>Sets the number of the measuring port.</td>
</tr>
<tr>
<td>7</td>
<td>Trigger</td>
<td>Optional for measuring. Measures by applying the trigger. (For details refer to section 5.5.9.2.)</td>
</tr>
<tr>
<td>8</td>
<td>Robot Controller</td>
<td>The controller to use this function</td>
</tr>
<tr>
<td>9</td>
<td>Measure Time</td>
<td>Optional for measuring. Measures the time. (For details refer to section 5.5.9.2.)</td>
</tr>
<tr>
<td>10</td>
<td>Accumulate</td>
<td>Optional for measuring. Accumulates the measuring data. (For details refer to section 5.5.9.2.)</td>
</tr>
<tr>
<td>11</td>
<td>Start (Stop)</td>
<td>Starts/stops the measuring.</td>
</tr>
<tr>
<td>12</td>
<td>Save Image / Save CSV</td>
<td>Saves the measuring data.</td>
</tr>
<tr>
<td>13</td>
<td>Status Bar</td>
<td>Shows the operating status.</td>
</tr>
<tr>
<td>14</td>
<td>Close</td>
<td>Port Debug exits.</td>
</tr>
</tbody>
</table>
5.5.9.1 Measuring Procedures

From the “Tools” menu in Visual Studio, select “M+ Port Debug”.

This function can also be used on programming pendant. Select {MotoPlus APL} > {MotoPlus Monitor}. Then choose {Port Debug}.

1. Set the measuring conditions.
   - **Time Range**: Sets the displaying time [ms] for measuring data.
   - **Port Number**: Specify the number of the measuring port.

2. Press {Start} to start measuring in Port Debug. After starting, switch {Start} to {Stop}. Press {Stop} to finish measuring.

*Fig. 5-9: Port Debug Window*

5.5.9.2 Optional for Measuring

**Trigger**

If the Trigger is enabled, data is only displayed when the trigger conditions are satisfied.

Set the port number and trigger conditions to apply the trigger. The trigger conditions are shown below.

- **Up**: Detects the rising edge.
- **Down**: Detects the falling edge.
- **Up&Down**: Detects both edges, rising and falling.
- **Level**: Detects exceeding of the threshold level.

*Fig. 5-10: Trigger*
5 Debugging

5.5 Development Support Tools for YRC1000

Org Position Adjusts the display of the home position of the measuring data. Depending on the adjustment of the home position, the displays differ as shown below. (When the trigger is “UP”)

Fig. 5-11: Home Position

Measure Time When this option is enabled, two cursors display for measuring a time lapse in the captured data. When adjusting the position of the cursors, any time can be measured.

Fig. 5-12: Measure Time

Accumulate Accumulates the measuring data. When this option is enabled, all measuring data is drawn by overlapping. The example below shows when the option is enabled. (Trigger is Up.) The deviation of the timing when the port is turned to 0 is caused by the deviation of the position of the rising edge.

Fig. 5-13: Accumulate
5. Debugging

5.5 Development Support Tools for YRC1000

MotoPlus SDK for Visual Studio

Save Image / CSV The measuring results are saved as an image file or a CSV file.

For Visual Studio, specify the destination to save and the file name, and then press the save button.

For MotoPlus Monitor, specify the external memory device to save to, and then press the save button.

The results are saved in the root folder of the selected external memory device.

The name of the image file is “PortDebugImageYYYYMMDD_HHMMSS.bmp”.

The name of the CSV file is “PortDebugImageYYYYMMDD_HHMMSS.csv”.

YYYYMMDD_HHMMSS indicates the time stamp when saving (Year/Month/Date/Hour/Minute/Second).

5.5.9.3 Registering Data for the Port Debug Function

Before using the debug port function, it is necessary to execute mpDebugPortNameSet() once in the application.

The number of ports which are available is 5. It is possible to set values from 0 to 100 for each port.

**Fig. 5-14: How to Implement Port Debug**

```c
void mpTask2(void)
{
    mpDebugPortInit(1, "Task2-1");
    mpDebugPortInit(2, "Task2-2");

    FOREVER
    {
        int i;
        for (i = 0; i <= 10; i++)
        {
            mpDebugPortLevelSet(1, i * 10);
            mpTaskDelay((i + 2) / mpGetRCC() / 2);
        }
        for (i = 0; i <= 10; i++)
        {
            mpDebugPortLevelSet(2, i * 10);
            mpTaskDelay((i + 2) / mpGetRCC() / 2);
        }
    }
}
```
5.5.9.4 Debug Port APIs

mpDebugPortInit

Initializes the debug port for the port debug function. Before using the port, it is necessary to execute it once.

- **Syntax**
  ```c
  int mpDebugPortInit(
    int portNo, 
    char* name
  );
  ```

- **Argument**
  - [portNo]
    Specify the port number from 0 to 4.
  - [name]
    Specifies the name of the debug port. The port name which is specified in this step is displayed in the window of the port debug. For the port name, use a maximum of 32 characters.

- **Return value**
  - 0: Normal end
  - -1: Error
mpDebugPortHigh

Sets the output level of the debug port to High.

- **Syntax**
  ```c
  int mpDebugPortHigh(
    int portNo,
  );
  ```

- **Argument**
  ```
  [portNo]
  ```
  Specify the port number from 0 to 4.

- **Return value**
  ```
  0: Normal end
  -1: Error
  ```
mpDebugPortLow

Sets the output level of the debug port to Low.

- **Syntax**

```c
int mpDebugPortLow(
    int portNo
);
```

- **Argument**

  [portNo]

  Specify the port number from 0 to 4.

- **Return value**

  0: Normal end  
  -1: Error
mpDebugPortLevelSet

Sets the output level of the debug port to a specified value.

■ Syntax
int mpDebugPortLevelSet(
  int portNo,
  int level
);

■ Argument
[portNo]
Specify the port number from 0 to 4.
[level]
Specify the output level. The valid settable range is 0 to 100.

■ Return value
0: Normal end
-1: Error
5.5 Copy Protection

This function allows the developer to provide copy protection to the application. The application will only run on an authorized controller.

Copy protection procedures are shown below.
1. (Developer) Write the protection activation process in the application source.
2. (End User) Acquire the serial file from the controller to use the application.
3. (Developer) Create a protection key from the acquired serial file.
4. (End User) Load the created protection key to the controller.

When protection of the application is enabled, the protection key is checked before starting the application. When the key is unauthorized or the key does not exist, the application does not start.

5.5.10.1 Protection Activation

Add the following function (mpCopyProtectionActive()) in the source file of the application. If this function returns ON, the protection is enabled. If this function returns OFF, the protection is disabled. If this function is not added to the source code, protection is disabled.

```
long mpCopyProtectionActive( void )
{
    return ( ON ); // ON: Activate protection, OFF: Inactivate protection
}
```
5.5.10.2 Acquiring the Serial File (MotoPlus Monitor)

Acquires the serial file from the controller.

1. Select {Protection Key} in MotoPlus Monitor.
2. Select the Memory Device to save the serial file.
3. Press {Save}.

The serial file (MPSERIAL.DAT) is saved in the root folder of the selected memory device.

Fig. 5-15: Serial File Acquisition Window
5.5.10.3 Generating the Protection Key

Generate the protection key from the acquired serial file.

1. Open the project of the application by using Visual Studio, and build once.
2. Select “M+ Protection Key” from the “Tools” menu.
3. Press {Open}, and select the acquired serial file (MPSERIAL.DAT).
4. Press {Generate}.

The key file (application name.KEY) is generated in the same folder as the OUT file of the application.

![Fig. 5-16: Protection Key Generation Window](image)

- The protection key cannot be generated if the project of the application does not exist.
- The protection key cannot be generated if the project of the application is not built.

5.5.10.4 Loading the Protection Key (MotoPlus Monitor)

Load the protection key to the controller.

1. Save the protection key file to the root folder of the memory device, and then insert the memory device into the programming pendant.
2. Select {Protection Key} in MotoPlus Monitor.
3. Select the memory device which the protection key is saved to.
4. Press {Open}.
5. The key file list on the memory device appears. Select the key file to load.
6. Press {Load}.
The protection key loads from the memory device to the controller. The application which is created in section 5.5.10.1 “Copy Protection” can be used. Copy protection procedures are complete.

*Fig. 5-17: Protection Key Download Window*
# Alarm List

<table>
<thead>
<tr>
<th>Alarm Number</th>
<th>Alarm Name</th>
<th>Subcode</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1020</td>
<td>MOTOPLUS APPLICATION LOAD ERROR</td>
<td>1</td>
<td>Too many MotoPlus applications (.out files) are loaded in the controller.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>The MotoPlus application is too large. (Available memory area is 2 Mbyte.)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>Failed to open the directory.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>Load failure (Failed to open the file.)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>Load failure (Undefined symbol) (You declared an external function or variable that was never defined in the system)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6</td>
<td>Load failure (Others: application overloaded) (Verify that your .out file has been compiled for the correct robot controller)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
<td>Failed to initialize the API library.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8</td>
<td>No user root task.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9</td>
<td>Failed to create the user root task.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>Failed to create RAM-Disk.</td>
</tr>
</tbody>
</table>

| 4478         | MM TASK NO RESPONSE (MotoPlus)  |         | The process requested from the MotoPlus application to the MM task was not completed within the specified time. |

| 4479         | MOTOPLUS MM TASK WATCHDOG ERROR |         | The man-machine task does not run for 3 seconds or more.                |
7 Advanced mpBuilder Usage

mpBuilder is the compile system utilized by MotoPlus for Visual Studio. This section describes the advanced usage for the build system. This should only be modified by experienced developers.

7.1 Visual Studio MakeFile Project

When select a MotoPlus Project from the New Project menu, the wizard is creating a Visual Studio MakeFile project. This project type in Visual Studio issues a command line call to a custom compiler.

The Build Command Line command can be viewed under {Project} > {Properties} > {Configuration properties} > {NMake}.

![Image of NMake configuration]

The default build command is:

```
mpbuilder.exe -c $(Configuration) -p "$(ProjectDir)\" -n "$(ProjectName)" -b "$(OutDir)\" -o build
```

The default clean command is:

```
mpbuilder.exe -c $(Configuration) -p "$(ProjectDir)\" -n "$(ProjectName)" -b "$(OutDir)\" -o clean
```

The fields $(…) are Microsoft macros for MSBuild. These are documented online on the MSDN website.
7.1 Visual Studio MakeFile Project

Using the $(...) syntax, you can also insert any system Environment Variables into the command line. Example: $(MP_VS_Install) evaluates to the MotoPlus installation directory.

The default commands do not use all of the available command-line arguments for mpBuilder. Here is a list of all available arguments:

- `-c`, `--controller` Required. Target YASKAWA robot controller platform. DX100, DX200, FS100 and YRC1000
- `-p`, `--projectFolder` Required. Root folder for the MotoPlus project.
- `-n`, `--projectName` Required. Name of your MotoPlus project.
- `-o`, `--operation` Required. Build operation to perform.
  - 'build' = Build Project. 'clean' = Delete output files. 'rebuild' = Clean then build.
- `-b`, `--binaryLocation` Path to the output binary. If this is not specified, it will default to [ProjectDir]\output.
- `-i`, `--includeDir` This is the path to the standard include-directory for your MotoPlus API header files. If this is not specified, it will default to [InstallDir]\[Controller]\inc.
- `-e`, `--encoding` The character encoding of your source files. Example: "US-ASCII" or "UTF-8". A list of supported encodings can be found in the Microsoft documentation for EncodingInfo.Name
- `-l`, `--library` 'Y' if you are building a MotoPlus Library file. 'N' if you are building a .out Executable module.
- `-d`, `--dontParseErrors` 'T' (true) if you want to see the unmodified error/warning strings returned from the compiler. Otherwise, they will be parsed and formatted to be recognized by Visual Studio.
7.2 Compiler Arguments

The specific compiler will vary based on which robot controller is selected. The proper one is selected based on the arguments specified for mpBuilder. Regardless of which controller is selected, the final compiler will be part of the GCC build system.

The arguments for the GCC build system are specified in the .mps files, which are automatically added to your MotoPlus for Visual Studio project.

### 7.2.1 CompilerArguments.mps

This file is used when compiling source files into binary objects. This can be used to control many variables in the build system.

- Project-wide DEFINE statements
- Enable/disable particular errors and warnings
- Compiler code optimization (not recommended)
- Directories to search for include-files for the project

There are macros which can be placed in the .mps file which will be recognized by mpBuilder and resolved prior to calling the compiler.

- `~ProjectDir~` : This will resolve to the temporary directory where the source files are copied and converted to ASCII encoding at build time.
- `~IncludeDir~` : This is the standard include directory to search for MotoPlus API header files. If the `-i` argument is specified to mpBuilder.exe, then this macro will resolve to the value of the `-i` argument. Otherwise, it will default to `[MP_VS_Install]\[Controller]\inc`
- `~FilePath~` : This will resolve to the full path of the source file being compiled. Please note that it will point to the ASCII copy of the source file, located in the temp directory.
- `~OutputPath~` : This will resolve to the full path of the compiled object file which is the output from the compiler.

### 7.2.2 LinkerArguments.mps

This file is used when linking the compiled object files into a single binary executable/library. There are macros which can be placed in this .mps file which will be recognized by mpBuilder and resolved prior to calling the compiler.

- `~FileList~` : This is the list of compiled object files which will be combined into the final output file.
- `~OutputPath~` : This is the full path to the final output file.

**NOTE**
Please see the GCC documentation for all available arguments to the build system.
Appendix A

A.1 Compatibility with Windows 8/10

The MotoPlus compiler is compatible with Windows 8. However, additional steps must be performed to make this work properly.

A.1.1 Symptom of Problem:

If a console window appears when attempting to build your MotoPlus application, then the compiler will not properly display Errors/Warnings inside of Visual Studio.

Fig. A-1: Compiler Not Displaying Errors/Warning Inside of Visual Studio

NOTE

The following compatibility issues are not applicable to MotoPlus SDK v1.1.0 or higher. The fixes below are only required for older versions.
A.1 Compatibility with Windows 8/10

**Solution:**
You must explicitly run Visual Studio as Administrator each time you launch the application. Please note that simply having an Administrator user account is not sufficient. When launching Visual Studio, you must right-click and select (Run as Administrator).

**NOTE**
You must perform this every time you want to open Visual Studio.

---

**Fig. A-2: From Metro Interface**

![Start Menu](image1)

**Fig. A-3: From Desktop Interface**

![Desktop Interface](image2)
Appendix B

B.1 Compatibility with Visual Studio 2015

MotoPlus SDK for Visual Studio is compatible with Visual Studio 2015.

B.1.1 Symptom of Problem:

If you see the error message "The filename, directory name, or volume label syntax is incorrect," then you must update Visual Studio.

CAUTION

Update Visual Studio 2015 to "update 2" or greater.

• Not updating Visual Studio to "update 2" or greater will prevent compiling MotoPlus apps.
MOTOPLUS SDK
FOR VISUAL STUDIO
USER’S MANUAL

(DX100, DX200, FS100 AND YRC1000)

HEAD OFFICE
2-1 Kurosakishiroishi, Yahatanishi-ku, Kitakyushu 806-0004, Japan
Phone +81-93-645-7703 Fax +81-93-645-7802

YASKAWA America Inc. (Motoman Robotics Division)
100 Automation Way, Miamisburg, OH 45342, U.S.A.
Phone +1-937-847-6200 Fax +1-937-847-6277

YASKAWA Europe GmbH (Robotics Division)
Yaskawasrasse 1, 85391 Allershausen, Germany
Phone +49-8166-90-100 Fax +49-8166-90-103

YASKAWA Nordic AB
Verkstads gatan 2, Box 504, SE-385 25 Torsas, Sweden
Phone +46-480-417-800 Fax +46-486-414-10

YASKAWA Electric (China) Co., Ltd.
22F, One Corporate Avenue, No.222, Hubin Road, Huangpu District, Shanghai 200021, China
Phone +86-21-5385-2200 Fax +86-21-5385-3299

YASKAWA SHOUGANG ROBOT Co. Ltd.
No7 Yongchang North Road, Beijing E&T Development Area, China 100176
Phone +86-10-6788-2858 Fax +86-10-6788-2878

YASKAWA India Private Ltd. (Robotics Division)
#426, Udyog Vihar, Phase- IV, Gurgaon, Haryana, India
Phone +91-124-475-8500 Fax +91-124-475-8542

YASKAWA Electric Korea Corporation
35F, Three IFC, 10 Guikjegeumyung-ro, Yeongdeungpo-gu, Seoul, Korea 07326
Phone +82-2-784-7844 Fax +82-2-784-8495

YASKAWA Electric Taiwan Corporation
12F, No.207, Sec. 3, Beishin Rd., Shindian District, New Taipei City 23143, Taiwan
Phone +886-2-8913-1333 Fax +886-2-8913-1513

YASKAWA Electric (Singapore) PTE Ltd.
151 Lorong Chuan, #04-02A, New Tech Park, Singapore 556741
Phone +65-6282-3003 Fax +65-6289-3003

YASKAWA Electric (Thailand) Co., Ltd.
59,1st-5th Floor, Flourish Building, Soi Ratchadapiske 18,Ratchadapiske Road,
Huaykwang, Bangkok 10310, THAILAND
Phone +66-2-017-0099 Fax +66-2-017-0199

PT. YASKAWA Electric Indonesia
Secure Building-Gedung B Lantai Dasar & Lantai 1 Jl. Raya Protokol Halim Perdanakusuma,
Jakarta 13610, Indonesia
Phone +62-21-2982-6470 Fax +62-21-2982-6741

Specifications are subject to change without notice for ongoing product modifications and improvements.