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

Please have the following information available when contacting Yaskawa Customer Support:

- System
- Primary Application
- Software Version (Located on Programming Pendant by selecting: {Main Menu} - {System Info} - {Version})
- Robot Serial Number (Located on robot data plate)
- Robot Sales Order Number (Located on controller data plate)

Part Number: 166241-1CD
Revision: 3
MANDATORY

• This manual explains MotoPlus of the DX200 system. Read this manual carefully and be sure to understand its contents before handling the DX200.

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

CAUTION

• Some drawings in this manual are shown with the protective covers or shields removed for clarity. Be 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. Be 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.
NOTES FOR SAFE OPERATION

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

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

**DANGER**
Indicates an imminent hazardous situation which, if not avoided, could result in death or serious injury to personnel.

**WARNING**
Indicates a potentially hazardous situation which, if not avoided, could result in death or serious injury to personnel.

**CAUTION**
Indicates a potentially hazardous situation which, if not avoided, could result in minor or moderate injury to personnel and damage to equipment. It may also be used to alert against unsafe practices.

**MANDATORY**
Always be sure to follow explicitly the items listed under this heading.

**PROHIBITED**
Must never be performed.

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”.

WARNING

• Before operating the manipulator, check that servo power is turned OFF when the emergency stop buttons on the front door of the DX200 and the programming pendant are 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 buttons do not function.

*Fig. : 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.

*Fig. : Release of Emergency Stop*

• Observe the following precautions when performing teaching operations within the P-point maximum envelope of the manipulator:
  – Be sure to use a lockout device to the safeguarding when going inside. Also, display the sign that the operation is being performed inside the safeguarding and make sure no one closes the safeguarding.
  – 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 DX200 power
  – 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 buttons are located on the right of the front door of the DX200 and the programming pendant.
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 DX200 cabinet after use.

The programming pendant can be damaged if it is left in the manipulator’s work area, on the floor, or near fixtures.

- Read and understand the Explanation of the Warning Labels in the DX200 Instructions before operating the manipulator.

Definition of Terms Used Often in This Manual

The MOTOMAN is the YASKAWA industrial robot product.

The MOTOMAN usually consists of the manipulator, the controller, the programming pendant, and manipulator 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>DX200 Controller</td>
<td>DX200</td>
</tr>
<tr>
<td>DX200 Programming Pendant</td>
<td>Programming Pendant</td>
</tr>
<tr>
<td>Cable between the manipulator and the controller</td>
<td>Manipulator cable</td>
</tr>
</tbody>
</table>
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 /Symbol Keys</td>
<td>The keys which have characters or symbols printed on them are denoted with [ ]. e.g. [ENTER]</td>
</tr>
<tr>
<td>Axis Keys /Numeric Keys</td>
<td>[Axis Key] and [Numeric Key] 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 “+” sign between them, e.g. [SHIFT]+[COORD].</td>
</tr>
<tr>
<td>Mode Switch</td>
<td>Mode Switch can select three kinds of modes that are denoted as follows: REMOTE, PLAY or TEACH.</td>
</tr>
<tr>
<td>Button</td>
<td>The three buttons on the upper side of the programming pendant are denoted as follows: START, HOLD, or EMERGENCY STOP.</td>
</tr>
<tr>
<td>Displays</td>
<td>The menu displayed in the programming pendant is denoted with { }. e.g. {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.

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.
# Outline

1. Outline .................................................................................................................... 1-1  
   1.1 Introduction ........................................................................................................ 1-1  
   1.2 Features ............................................................................................................ 1-1  
   1.3 What MotoPlus Can Do .................................................................................... 1-2  
   1.4 Supported Services ......................................................................................... 1-3  
   1.5 Application Program Development Environment .......................................... 1-5  
      1.5.1 Development Environment Configuration ............................................ 1-5  
      1.5.2 Development Environment Component ............................................. 1-6  
      1.5.3 Application Program Execution Environment ...................................... 1-7  
      1.5.4 Development Flow ................................................................................ 1-8  
   1.6 Notes ............................................................................................................. 1-9  

2. Development Environment Setup ............................................................................... 2-1  
   2.1 DVD Contents .................................................................................................. 2-1  
   2.2 Installation of Development Environment Software ....................................... 2-2  
   2.3 Installation of TELNET Connection Utility ....................................................... 2-5  

3. Creating Application Software .................................................................................... 3-1  
   3.1 Rules for Creating Program Source Code ....................................................... 3-1  
   3.2 Creating MotoPlus Project ............................................................................ 3-2  
   3.3 Outline of MotoPlus IDE ............................................................................. 3-3  
   3.4 Text Search .................................................................................................... 3-4  
   3.5 MotoPlus Library Files ............................................................................... 3-4  
   3.6 Program Build (Compile/Link) ..................................................................... 3-4  
   3.7 Changing Build Settings ............................................................................. 3-5  
   3.8 How to Use the DX100 Projects on DX200 ..................................................... 3-6  

4. Valid/Invalid Setting of MotoPlus Function .................................................................. 4-1  

5. Installation and Start-up of Application Program....................................................... 5-1  
   5.1 Installation of Application Program in Maintenance Mode ......................... 5-1  
   5.2 Start-up of Application Program .................................................................... 5-6  
   5.3 To Temporarily Prevent MotoPlus Application from Starting ....................... 5-6  

6. User Defined File ...................................................................................................... 6-1  
   6.1 Saving in and Reading from External Memory ............................................... 6-1
6.1.1 Procedure to Save or Read User Defined File ..................................................... 6-1
6.2 Initialization of User Defined File ........................................................................... 6-3
6.3 Setting of File Control Task Priority Limit ............................................................... 6-5
   6.3.1 Changing File Control Task Priority Limit ......................................................... 6-5

7 Debugging ...................................................................................................................... 7-1
   7.1 Debugging Environment ............................................................................................. 7-1
   7.2 Connecting DX200 with PC ....................................................................................... 7-2
   7.3 Debugging via Telnet Communication ...................................................................... 7-4
      7.3.1 Operation Procedure .......................................................................................... 7-4
      7.3.2 Example Usage of Printf() .............................................................................. 7-7
   7.4 Debugging by User Variables of DX200 ................................................................. 7-8

8 Alarm List ..................................................................................................................... 8-1
1 Outline

1.1 Introduction

This function is used to develop customer specific applications that operate on the main CPU board (YCP21 circuit board) of the DX200. The function name MotoPlus is an abbreviation of Moto man Professional Programming Language for Superior Use. MotoPlus is a programming IDE (integrated development environment) for professionals, with which the application program developed on the PC using C language runs as a task of the robot controller. Users can develop a variety of application programs on their own without modifying the source codes of the robot controller.

NOTE
To use the MotoPlus function, set the MotoPlus FUNC. to “USED” in the OPTION FUNCTION of the maintenance mode.

1.2 Features

1. With C language as the system development language and a wealth of standard libraries, users can create customer specific application programs.

2. With the text editor of MotoPlus IDE, users can program off-line.

3. The created application program runs as a task on the main CPU of the robot controller. Thus, no additional hardware is required.

4. The created application program runs as the native code of the CPU. Thus, the execution speed becomes faster.

5. With a variety of included APIs (application program interface) to transmit data from/to the job, the Ethernet communication port, and the RS232C serial communication port, etc., users can easily correct the manipulator position and connect the robot to the external PC or sensors.
1.3 What MotoPlus Can Do

Users can use the C language, a general programming language, and the C 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. Refer to *fig. 1-1 “Available Resources for MotoPlus”*

*Fig. 1-1: Available Resources for MotoPlus*

Application program examples:

1. Operation to correct the manipulator position by a communication program with a vision sensor or other sensors, and by data from the sensor
2. Data transmission with an external PC via the Ethernet (TCP/IP) communication
3. JOB execution sequence control
1.4 Supported Services

MotoPlus provides users with a variety of services as follows:

1. Application task control
   Application task start/stop, data transmission between application programs by the mailbox and semaphore, exclusive control and execution synchronization

2. Robot control
   Robot control from the application program

3. 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

4. CIO control
   I/O between the application program and the CIO, and read/write of the register

5. Ethernet communication control
   The Ethernet (TCP/IP) communication from the application, especially using the socket function library included as an API

6. Programming pendant communication
   Data transmission between the application and the programming pendant application

7. EVENT
   Event notification to the application at every I/O control cycle and interpolation control cycle of the system

8. RS232C serial communication control
   RS232C serial communication from the application

9. Sensor control
   The APIs which transfer data to and from a job and change operating conditions such as path correction and speed change, and the instructions which transfer data between a job and MotoPlus application

10. Memory management
    The protected memory management
    Specific instructions, malloc and mfree, support the data area which the system manages for MotoPlus.

11. General-purpose file control
    The function to access multiple general-purpose files by using the fixed area on CMOS as a drive

12. Existing file control
    The function to access existing files (jobs and condition files, etc. which can be loaded and saved by external memory)

13. Servo control
    Servo control from the application program

14. User watchdog
    Watchdog to monitor whether the application operates normally

15. Coordinate conversion
    Calculation of the manipulator order or inverse kinematics, coordinate conversion, pulse conversion from the feedback pulse to the arithmetic pulse, or linear algebra calculation.
16. Development environment

(1) Integrated development environment (MotoPlus IDE) for source file project management, program editor, and compiler and linker

(2) Installation of the MotoPlus application from the CF/USB memory in the programming pendant

(3) Debugging by printf() via Telnet
1.5 Application Program Development Environment

A MotoPlus application program is written in C language on a PC and debugged on the DX200 controller. Use the text editor of MotoPlus IDE to write the application program, then compile and link it to create the execution object. This is installed and executed on the DX200 to debug.

1.5.1 Development Environment Configuration

Use a PC to develop an application program. The system is configured as shown in fig. 1-2 “Development Environment Configuration” to debug using Telnet. Install the developed application by loading it to the CF of the main CPU (YCP21) using the CF/USB memory in the programming pendant.

**Fig. 1-2: Development Environment Configuration**

![Development Environment Configuration Diagram](image-url)
## 1.5.2 Development Environment Component

<table>
<thead>
<tr>
<th>No.</th>
<th>Device</th>
<th>Details</th>
<th>Remarks</th>
</tr>
</thead>
</table>
| 1   | Windows PC     | CPU: 1GHz or more  
Memory: 1 Gbyte or more  
Hard disk space: 40 Gbyte or more  
OS: Windows XP Service Pack 2, Windows Vista, Windows 7 (32 bit) | Prepared by the user      |
| 2   | Text editor    | Creates the application source program in C language. MotoPlus IDE editor is recommended. (A commercially-available text editor and Windows Notepad can also be used.) | Prepared by the user      |
| 3   | Compiler       | Converts the source program into the machine language of the target CPU. Integrated in MotoPlus IDE, and can be executed from the menu. | Provided by Yaskawa       |
| 4   | Linker/library | Links the C language library provided by GNU with the developed application program to create an executable load module. Integrated in MotoPlus IDE, and can be executed from the menu. | Provided by Yaskawa       |
| 5   | MotoPlus library | API library which provides application program services of the robot controller such as data communication with JOB. Integrated in MotoPlus IDE. | Provided by Yaskawa       |
| 6   | CF/USB memory  | Compact Flash memory or USB memory to load the developed load module into the controller. | Prepared by the user      |
| 7   | Ethernet cable | Connects the PC with the controller via Ethernet to debug the developed program. CAT5 or greater (cross/straight) cable | Prepared by the user      |
| 8   | Ethernet hub   | Connects the PC with the controller. | Prepared by the user      |
| 9   | Installation DVD | Installs the above-mentioned software on the Windows PC. | Provided by Yaskawa       |
1.5.3 Application Program Execution Environment

The application program developed in the above-mentioned program development environment is stored in the Compact Flash on the main CPU (YCP21) of the robot controller, as shown in fig. 1-3 “Program Execution Environment”.

Fig. 1-3: Program Execution Environment

When the controller power is turned ON, the application program MotoPlusApl stored in the CF (Compact Flash memory) on the main CPU (YCP21), is loaded into the memory of the YCP21 circuit board. Then, the application program is started as a task on the real-time operating system controller OS, and executed synchronously with the system program. The application task can start multiple tasks, and can perform data transmission with JOB or I/O, synchronous execution control, and exclusive control, by using the mailbox or semaphore.
1 Outline
1.5 Application Program Development Environment

1.5.4 Development Flow

Fig. 1-4: Program Development Flow

- Write the source program in C language by using the editor of MotoPlus IDE.
- Modify the program
- By using MotoPlus IDE, compile the source program (e.g. abc.c) and create an application program (e.g. abc.o).
- No compile error?
- Link the above-mentioned object modules with the C language library and the robot library to create the execution module (e.g. MotoPlusApl.out).
- No compile error?
- Load the application program into the robot controller.
- Start the application program.
- Debug.
- OK?
- Development completed.

<3.1 Rules for Creating Program Source Code>
Use the editor of MotoPlus IDE. Notepad, WordPad, or another text editor can also be used.

<3.2 Creating MotoPlus Project>
Use GNU C compiler provided by Yaskawa (included in MotoPlus IDE).

<3.2 Creating MotoPlus Project>
Use GNU linker and library. Only static links are supported. ELF (Executable and Linkable Format) is used as the module. Compiling and linking are done simultaneously by MotoPlus IDE.

<4.1 Installation of Application Program in Maintenance Mode>
Relocatable module with ELF object format.

<4.2 Start-up of Application Program>
1. Starts with turning ON the power.

<5. Debugging>
Debug by connecting to SHELL, system software, via TELNET. Use printf() and puts().
1.6 Notes

• C language memory operation
  When programming in C language, the pointer variable which stores the memory address can 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. Thus, by testing adequately, the error location can be found. Check the functions of the application software well enough, and make sure that all functions operate as designed.

• Task “mpUsrRoot” which starts when the power is turned ON
  The task “mpUsrRoot” 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 (100 microseconds or longer) is done in this task, the processing time for the robot control becomes insufficient. Then the system alarm may occur or the programming pendant or the HOLD button may freeze. Thus, as the sample program (refer to “16.3 Task Control Sample Program” in “Programmer’s Manual for New Language Environment MotoPlus (HW1482095)”, 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 the system processing time, and an alarm may occur due to insufficient processing time for robot operation.
  2. With the Telnet function, debug only the 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 the task “mpUsrRoot”), use the variables of the DX200. This is because, if printf() or puts() is done with a high priority task, the processing time for the robot operation control is consumed, and a system alarm may occur due to insufficient processing time.
  3. After completing debugging, make sure to clear the debugging code using printf() or puts() which is described in the source code. 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 the SHELL debugging function of the controller OS which becomes available by connecting with the DX200 via Telnet, a system alarm may occur due to insufficient processing time for robot operation. Do NOT use the SHELL debugging function.
2 Development Environment Setup

2.1 DVD Contents

The DVD contains the following items.

<table>
<thead>
<tr>
<th>Folder</th>
<th>Contents</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root</td>
<td>MotoPlus IDE installer</td>
<td></td>
</tr>
<tr>
<td>Manual (Japanese/</td>
<td>MotoPlus user’s manual</td>
<td>MotoPlus outline, installation procedure of the development environment, and development</td>
</tr>
<tr>
<td></td>
<td></td>
<td>procedure</td>
</tr>
<tr>
<td></td>
<td>MotoPlus programmer’s manual</td>
<td>Programming procedure</td>
</tr>
<tr>
<td></td>
<td>MotoPlus reference manual</td>
<td>API specifications</td>
</tr>
<tr>
<td></td>
<td>GCC manual</td>
<td></td>
</tr>
<tr>
<td>Sample</td>
<td>Sample source code</td>
<td></td>
</tr>
<tr>
<td>.NET Framework</td>
<td>Installer of .NET Framework version3.5 Service Pack 1</td>
<td></td>
</tr>
<tr>
<td>Sentinel Driver</td>
<td>Installer of the system driver for the</td>
<td></td>
</tr>
<tr>
<td></td>
<td>hardware key</td>
<td></td>
</tr>
</tbody>
</table>
2.2 Installation of Development Environment Software

Install MotoPlus IDE as indicated in the following procedure.

1. Insert the MotoPlusIDE installation DVD in the DVD slot on the PC.
   If AutoRun is enabled, the installation starts.
   If not, run “setup.exe” in the DVD.

2. Confirm .NET Framework 3.5 Service Pack1 (SP1)
   If the .NET Framework 3.5 SP1 is not installed on the PC, the following window appears and the installation starts.

   ![Image of installation process]

   **NOTE**
   To install .NET Framework 3.5 SP1, PC must be connected to the Internet.
2 Development Environment Setup
2.2 Installation of Development Environment Software

3. Install MotoPlusIDE
After the installation of .NET Framework 3.5 SP1 is completed (or if it is already installed), the following MotoPlusIDE installation window appears.

4. Click 「Next>」
The following window appears to specify the installation destination folder.
The default installation folder is “C:\Program Files\Yaskawa\MotoPlusIDE_DN”
5. Click [Next>]
The following confirmation window for installation appears.

6. Click [Next>]
The progress of the installation appears.
When the installation is completed properly, the following window appears.
Upon completing the installation, shortcuts of MotoPlusIDE are added on the desktop and in the Programs folder of the Start menu.

7. After completing the installation, restart the PC
Shut down the PC and restart it.

8. Install the system driver for the hardware key
After restarting the PC, start “SentinelSystemDriverInstaller7.5.7.exe” in the “SentinelDriver” folder in the installation DVD.

9. Install the hardware key
Insert the provided hardware key into any available USB port. The message “Your new hardware is installed and ready to use.” will appear in the taskbar.
2.3 Installation of TELNET Connection Utility

To debug the MotoPlus application, connect the DX200 and the PC via TELNET and describe using printf() in the application program. Thus, the TELNET connection utility is prepared. Install it by following the procedure below.

1. Copy the entire “DX200_TELNET” folder in the MotoPlus installation DVD into an appropriate hard disk of the PC. (“DX200_TELNET.exe” and “DX200_TELNET.ini” must be in the same folder.) Run “DX200_TELNET.EXE” to connect to the DX200 via TELNET.

2. Also, configure the settings of TELNET connection by “DX200_TELNET.ini”. Set the IP address of the DX200 here.
   - ipaddress: IP address of the DX200 (Set the same IP address as the one set in the HOST SETUP window.)
   - sendtime: Waiting time from the startup of “DX200_TELNET.EXE” until starting the TELNET connection (default value: 8, unit: sec)
   - inputTime: Waiting time from starting the TELNET connection until starting to login (default value: 4, unit: sec)

3. Insert the provided hardware key into any available USB port. The message “Your new hardware is installed and ready to use.” will appear in the taskbar.

NOTE
When the setup utility completes, you must also install the system driver for the hardware key. Start “Sentinel System Driver Installer 7.5.7.exe” to install the driver.
3 Creating Application Software

3.1 Rules for Creating Program Source Code

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

Refer to “Programmer’s Manual for New Language Environment MotoPlus (HW1482095)” for detailed information on programming configuration, files to be included, and sample programs.

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 extension.

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

3. The application entry point is the function which is called from the system software when the DX200 is started. This function must be described as:

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

   Be sure to describe mpExitUserRoot at the end of the mpUserRoot function.

   
   ```c
   void mpUsrRoot (int arg1, int arg2, int arg3, int arg4, int arg5, int arg6, int arg7, int arg8, int arg9, int arg10)
   {
   /* 10 parameters from arg1 to arg10*/
   .
   mpExitUsrRoot;; End of mpUsrRoot program (End of the task)
   }
   
   e.g. filename: abc.c
   ```

   * The original argument of arg1 to arg10 is mpUsrRoot. arg1 to arg10 are prepared to receive arguments in the future.
   At present, arg1 is the function name of the user entry, and arg2 to arg10 are reserved for the manufacturer (if referenced, all of them are "0").

   
   The task “mpUsrRoot” 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 (100 microseconds or longer) is done in this task, the processing time for the robot control becomes insufficient. Then the system alarm may occur or the programming pendant or the HOLD button may freeze. Thus, as the sample program (refer to “16.3 Task Control Sample Program” in “Programmer’s Manual for New Language Environment MotoPlus (HW1482095)”), make sure to complete the task “mpUsrRoot” after starting up another application or creating a semaphore.
3.2 Creating MotoPlus Project

Start MotoPlus IDE from the Windows Start menu. The window of MotoPlus IDE, similar to that of Visual Studio, appears.

To create a MotoPlus application, create a project first. The project is the unit to manage the source files for each application program.

All of the source files required for the application are included in the project. Double-click an existing project in Windows Explorer, then MotoPlus IDE automatically starts and the project is opened.

To create a new project, go to the top menu bar, then click File > New > Project. Then, the following window appears.

Type a project name (up to 32 single-byte characters which can be used as a file name), then a folder to store the source file will be automatically created (the folder can be located anywhere in the PC). Select the type of the MotoPlus file to be created from “Project Settings”.

There are following options for “Project Settings”:

1. Executable program (default): Creates an *.out file. This is a normal MotoPlus application which can be executed as a task of the DX200.

2. MotoPlus library file: Creates a static library file (*.dnLib). This type of library cannot be executed on the DX200 but can be called from MotoPlus application. Thus, import it from another project file to use it.

When a new project is created, the source code template “mpMain.c” opens. Use it if necessary. Delete it if not necessary.
3.3 Outline of MotoPlus IDE

The left pane of MotoPlus IDE contains the project tree. This lists all source files, header files, and library files that will be compiled when the project is built. To edit a file that is in the project tree, double-click the filename, then the contents of the file appears as a new tab in the source editor window. See fig. 3-1 “Main interface for MotoPlus IDE”.

Fig. 3-1: Main interface for MotoPlus IDE

Right-click on the project tree to show the following options:
1. Remove a file from the project
2. Create sub-folders to organize files (click and drag files to move them to a new folder)
3. Add files to the project (new or existing)
4. Add MotoPlus libraries to the project
5. Compile the code and build your program

The source editor has the following features:

- Automatic syntax highlighting of keywords
- Automatic indenting
- Undo/redo up to 50 actions
- Tab for each open document for easy navigation between files
- Right-click menu to instantly open the header file that has been included
- Search for text in the entire project
3.4 Text Search

To search for text, go to the top menu bar and click Edit > Find and Replace (shortcut is Ctrl + f). Then, the “Find” dialog appears. When “Find All” is selected, the cursor will not immediately go to the next searched text. Instead, the bottom pane will change to the “Find Results” tab to show the search results. Double-click a result in this pane to go to the corresponding source code tab and move the cursor to the searched text.

3.5 MotoPlus Library Files

A MotoPlus library file (*.dnLib) is a file containing the source code which is already compiled. After the source code is compiled to .mpLib format, it cannot be edited or viewed. (The library can be distributed and used without distributing the source code.)

To create a library file, select “Create MotoPlus Library file” when creating a new project. This library can be used by other projects, but in this case, prepare an appropriate header file (*.h) in advance. Also, DO NOT use mpUsrRoot() function for the library.

To use the library file in an executable project, go to the top menu bar and click Project > Add MotoPlus library file. Then you can locate an .mpLib file on your computer. When you select the file, it will be copied to the current project folder along with the corresponding header file. If the header is not present (or is named differently), it will not be copied. In this case, you must refer to the library documentation provided by the creator of the library. When you build your project, the compiled code from the library will be added to the final executable object (.out).

3.6 Program Build (Compile/Link)

To compile the created source code and build either an executable object (*.out) or a library file (*.dnLib), go to the top menu bar and click Build > Build Project. This will save any changes made to the project.

The bottom pane will automatically switch to the “Output” tab during a build attempt. The text box in this window will log the build process. The “Build Errors” tab will display any errors or warnings given by the compiler. If there is a fault in the source code, it will indicate the line number where it encountered the error. To quickly go to the appropriate file and line number, just double-click the error message. Please note that due to the structure of C language, an error on any line could be caused by a fault in any line above the error. If there is a fault in an include file, the error message may not even indicate the correct file.

If the Output window indicates the build was successful, you will be able to locate your *.out file in a folder labeled “output”. This folder will be in the directory of your project file. If your project is set to build a library file, the output folder will contain the *.dnLib file and the *.h file.
3.7 Changing Build Settings

To change build settings, select “Build”, then “Build settings...” under the menu.

The sample program in the DVD is created in the default installation folder (C:\Program Files\Yaskawa\MotoPlusIDE_DN). Thus, when it is installed in another folder, the build settings must be initialized.

To initialize the build settings, check “Edit Enable/Disable” and click on the button “Restore factory defaults”.

Also, compile options, build options, etc. can be changed by using “Compile Code Command” or “Build Executable Command”, but basically, use the default settings.
3.8 How to Use the DX100 Projects on DX200

DX200 is different from DX100 in the execution environment of MotoPlus application. Therefore the MotoPlus application for DX100 is unavailable on DX200.

The MotoPlus application available on DX200 can be made by using the application for DX100. The procedure is as follows:

1. Copy the folder which includes the MotoPlus application for DX100 by another name.
   (When the folder (C:\My_First_MP_Project\Sample) includes any project, copy the folder (C:\My_First_MP_Project\Sample) and make a new folder (C:\My_First_MP_Project\Sample1)).

2. Open the copied folder and change the file extension of “Sample.mpProj” to “Sample.dnProj” in the .
   Change the extension of a project file “Sample.mpProj” in the copied folder to “Sample.dnProj”.

3. Double-click the file Sample.dnProj and start the MotoPlusIDE_DN.

4. Click the MotoPlusIDE_DN menu “build”, then click “Build Setting...”.

5. Click the button [Restore Factory Defaults] in the “Build Settings” dialogue. That makes the build settings into the default state.

6. Click the MotoPlusIDE_DN menu “build”, then click “Build Project F7”.
   When the output window “Build Successful!” appears, making the MotoPlus application for DX200 is completed.
4 Valid/Invalid Setting of MotoPlus Function

The following is the procedure to set the MotoPlus function to valid.

1. Start the FS100 in MAINTENANCE MODE, and change to the “MANAGEMENT MODE” in the security mode.

2. Select {SYSTEM} under the main menu, then select {SETTING}.
   - The option function window appears.

3. Select “MotoPlus FUNC.”
   - The selection list “NOT USED”/“USED” appears.
4. Select “USED”.

– The confirmation dialog appears.

![Confirmation Dialog]

5. Select “YES” on the confirmation dialog.

– If the memory I/F expansion option is valid, the SRAMDRV.DAT initialization confirmation dialog appears. If the memory I/F expansion option is invalid, the dialog doesn’t appear and the MotoPlus function becomes valid.

![Initialization Dialog]


– The MotoPlus function becomes valid.
The following is the procedure to set the MotoPlus function to invalid.

1. Start the DX200 in MAINTENANCE MODE, and change to the "MANAGEMENT MODE" in the security mode.

2. Select (SYSTEM) under the main menu, then select {SETUP}.
   - The option function window appears.

3. Select “MotoPlus FUNC.”
   - The selection list “NOT USED”/“USED” appears.
4. Select “NOT USED”.
   – The setting change confirmation dialog appears.

5. Select “YES” on the confirmation dialog.
   – The MotoPlus function becomes invalid.
5 Installation and Start-up of Application Program

5.1 Installation of Application Program in Maintenance Mode

1. Load function
   Load the application program and install it by following the instructions below.
   
   (1) Set the security mode to the MANAGEMENT MODE. Select the main menu {MotoPlus APL.}. Then {LOAD (USER APPLICATION)}, {FILE LIST (DX)}, {DELETE (DX)}, {DEVICE}, {FOLDER}, and {MotoPlus FUNC. SETTING} appear as submenus.

   (2) 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)}.

   (3) Select and load a file.
   • Select the submenu {LOAD (USER APPLICATION)}. Then, "*.out" files in the specified folder appear as shown below.
5 Installation and Start-up of Application Program
5.1 Installation of Application Program in Maintenance Mode

- 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.

- When [YES] is selected and the selected application file contains the file which already exists in the DX200, the following confirmation dialog box appears. Select “YES” to load.
5 Installation and Start-up of Application Program
5.1 Installation of Application Program in Maintenance Mode

2. List
Select the submenu {FILE LIST (DX)}. Then, the list of the application files which already exist in the DX200 appears.

3. Deletion of application program
By the submenu {DELETE (DX)}, the application file which already exists in the DX200 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 “★”.

(2) Press [ENTER], then the following confirmation dialog box appears.
4. 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 chapter 5.2 “Start-up of Application Program” and chapter 5.3 “To Temporarily Prevent MotoPlus Application from Starting”.

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

   (2) Press [ENTER], and then the confirmation dialog box appears.
(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.
5.2 Start-up of Application Program

When the DX200 is turned ON, the application program is loaded automatically and started.

<Limitions>

• 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 3 Mbyte.)  
3: APPLICATION folder cannot be found in the CF.  
4: The size of the MotoPlus application exceeds the limit.

5.3 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 DX200 may not start normally. For recovery, it is necessary to start the DX200 normally while preventing the application program from operating. In this case, it is necessary to start the DX200 in the MAINTENANCE mode, make the settings so that the application program does not operate, start the DX200 normally, and operate the robot. Follow one of the following two procedures:

1. Start the DX200 in the MAINTENANCE mode, set the security mode to the MANAGEMENT MODE. Select "MotoPlus APL.", then "DELETE (DX)" to delete the application.
2. Start the DX200 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 the procedure 2. The above problem during robot operation may be caused by a specific condition. If the MotoPlus application is deleted with the 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.
6 User Defined File

With MotoPlus, user defined files can be created, read, written, deleted, etc. from an application by using the file control.

These files can be saved in or read from external memory (CF or USB) by using the programming pendant.

6.1 Saving in and Reading from External Memory

Saved files can be checked on the display of the programming pendant as “USER DEFINED FILE”. The files can be saved in or read from external memory such as CF or USB on this display.

6.1.1 Procedure to Save or Read User Defined File

1. Start in the normal mode, and select {EX. MEMORY}, then {SAVE} or {LOAD}.

2. Select {USER DEFINED FILE}.

![Image](image-url)
6 User Defined File
6.1 Saving in and Reading from External Memory

3. Select a file to be saved in external memory or to be loaded from external memory to the controller.

4. ★ appears on the left of the selected file. Multiple files can also be selected.

5. Select a file and press [ENTER], then the following window appears. Select “YES”.

### 6.2 Initialization of User Defined File

1. Start the controller in the maintenance mode. Then, select {Main Menu}, {FILE}, then {INITIALIZE}.

2. Select "USER DEFINED FILE".

3. Select "SRAM RAM DRIVE  SRAMDRV .DAT".
6 User Defined File
6.2 Initialization of User Defined File

4. ★ appears as shown below.

5. Press [ENTER], then the following window appears. Select {YES}.
6.3 Setting of File Control Task Priority Limit

Only the normal priority (MP_PRI_TIME_NORMAL) is used as the task priority for the file control API.

When the file control API is used from the task with higher priority, the operation differs depending on the setting of the parameter S2C1101 as shown in the following table.

This setting can also be changed by using “FileControl TASK PRY. LIMIT” in the maintenance mode.

<table>
<thead>
<tr>
<th>S2C1101</th>
<th>FileControl TASK PRY. LIMIT</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>LIMITED</td>
<td>The API returns an error.</td>
</tr>
<tr>
<td>1 (default value)</td>
<td>AUTO</td>
<td>Temporarily changes the task priority to MP_PRI_TIME_NORMAL and executes the API.</td>
</tr>
</tbody>
</table>

6.3.1 Changing File Control Task Priority Limit

1. Start in the maintenance mode. Select {Main Menu}, {MotoPlus APL.}, then {MotoPlus FUNC. SETTING}.

2. Press [SELECT] at “FileControl TASK PRY. LIMIT” to select “LIMITED” or “AUTO”.
3. Press [ENTER], then select {YES} in the “Modify?” window to complete the setting.
7 Debugging

7.1 Debugging Environment

For the debugging environment, connect the PC for development with the RUN-TIME debugger of the controller OS via Telnet. Then you can insert printf() or puts() in the application source code to display the messages about variable and program execution states when debugging.

NOTE
Delete all the debugging codes described in the source code after completing debugging. Otherwise, the processing time of printf() and puts() may affect the system and may cause insufficient processing time, etc. during robot operation.

NOTE
With the Telnet function, debug only the 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 the task “mpUsrRoot”), use the variables of the DX200. This is because, if printf() or puts() is done with a high priority task, the processing time for the robot operation control is consumed, and a system alarm may occur due to insufficient processing time.

NOTE
When programming in C language, the pointer variable which stores the memory address can 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. Thus, by testing adequately, the error location can be found. Check the functions of the application software well enough, and make sure that all functions operate as designed.

NOTE
When using the SHELL debugging function of the controller OS which becomes available by connecting with the DX200 via Telnet, a system alarm may occur due to insufficient processing time for robot operation. Do NOT use the SHELL debugging function.
7.2 Connecting DX200 with PC

Connect the PC for development with the DX200. Then the PC can be used as the terminal to debug programs via Telnet.

As shown below, connect an Ethernet cable (shielded twisted-pair, CAT 5 or greater) to the LAN connector CN104 (RJ-45) that is mounted on the front of the YCP21 circuit board in the CPU rack of the DX200 controller.

To use the Ethernet service provided by MotoPlus, use a commercially available hub and set up the connection as shown below.
Two RJ-45 connectors are mounted on the front of the YCP21 circuit board. Only CN104, which is located on the lower position, can be used for the Ethernet function that MotoPlus supports.
7 Debugging
7.3 Debugging via Telnet Communication

7.3 Debugging via Telnet Communication

The RUN-TIME debugger for the DX200 system software, can be connected with a debug PC via Telnet to provide the user with various types of debugging function. Thus, printf() and puts() functions can be inserted in the application source code, and the desired memory content or message can be shown on the Telnet terminal.

With this procedure, debug only the 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 the task “mpUsrRoot”), refer to chapter 7.4 “Debugging by User Variables of DX200”.

If the Telnet function is enabled, an alarm such as “0500: SEGMENT PROC NOT READY” may occur when the DX200 is started. This is because the Telnet task of the DX200 side is performed with a high priority, so processing of other system tasks are not performed normally. If such an alarm occurs, restart the DX200. Also, after completing debugging, make sure to set the Telnet function to INVALID (S2C1119 = 0).

7.3.1 Operation Procedure

1. Set the IP address, etc.

   The Ethernet communication settings corresponding to your environment (such as the IP address setting at the DX200 side) must be performed in the maintenance mode.
   
   The procedure is as follows:

   (1) Turn OFF the DX200 then back ON while pressing the [MAIN MENU] to start the maintenance mode.

   (2) Select {SYSTEM} under the main menu, then select {SECURITY}. Then, change the security mode to the MANAGEMENT MODE.
(3) Select (SYSTEM) under the main menu, then select (SETUP). In the setup window, select (OPTION) to open (NETWORK). In the NETWORK window, select “DETAIL” for “HOST SETUP”.

(4) Setup the IP address in HOST SETUP window. Normal setting for the IP address is 192.168.255.1. Ensure the PC IP address is on the same subnet as the DX200. In the case of the setting for the following DX200, the setting for PC is as follows. IP address: 192.168.255.xx (xx: 0-255, decimal number, different from DX200 value), Subnet mask: 255.255.255.0

2. Restrictions on the IP address setting:
The Ethernet function on the DX200 does not support “10.0.0.xx” in the local IP address (xx: 0-255, decimal number). Thus, DO NOT use “10.0.0.xx” for the IP address.
3. Restrictions on communication ports:
   For the Ethernet function of the DX200, the system occupies the
   specified ports for UDP and TCP. Thus, the port numbers less than
   10500 cannot be used for MotoPlus applications. Also, the ports to be
   used by the system may be added or changed depending on the
   DX200 software version. If your robot is using any optional functions
   related to the DX200 Ethernet function, be sure not to use the port
   numbers that are described in the instruction manual for each optional
   function.

4. Set the fixed IP address (192.168.255.9) as the network setting of the
   PC side.
   ("192.168.255." is the same as the DX200 setting.)
   (1) Select Control Panel and then double-click “Network connections”.
   (2) Double-click “Local Area Connection”.
   (3) Open the “General” window and select Internet protocol (TCP/IP),
       then click “Properties”.
   (4) Check “Use the following IP address”.
   (5) Enter “192.168.255.9” for the IP address.
   (6) Enter “255.255.255.0” for the subnet mask.

5. Debugging via Telnet Communication

   Execute the automatical log-in tool “DX200_TELNET.EXE” while
   MOTOMAN window is shown.

   When Telnet connection succeeds, prompt becomes to " → ", Run
   debugger of the OS of the controller starts automatically.
7 Debugging

7.3 Debugging via Telnet Communication

7.3.2 Example Usage of Printf()

The following is an example of using the printf() function. This sample displays the counter value of execution while looping.

<Operation>

1. Create the following “sample.c” with MotoPlus IDE to create “sample.out”.

```c
#include "motoPlus.h"

// for GLOBAL DATA DEFINITIONS

void mpUserRoot(int arg1, int arg2, int arg3, int arg4, int arg5,
                 int arg6, int arg7, int arg8, int arg9, int arg10)
{
    unsigned int run_cnt;
    run_cnt = 0;
    while (1)
    {
        mpTaskDelay(2500);
        printf("mpUserRoot task Running[Mel/Mph], run_cnt=", run_cnt);
        run_cnt++;
    }
    mpExitUserRoot; // (or) mpSuspendScf;
}
```

2. Install “sample.out” to the DX200. (Refer to chapter 5 “Installation and Start-up of Application Program”.)

3. Install the created “MotoPlusApl.out” to the DX200. (Refer to chapter 5 “Installation and Start-up of Application Program”.)

4. Turn ON the DX200, and confirm that the DX200 started successfully.

5. Connect the DX200 with the PC via Telnet.

6. The following window appears.
7  Debugging
7.4  Debugging by User Variables of DX200

7.4 Debugging by User Variables of DX200

Debugging with printf() or puts() for 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 the task “mpUsrRoot”) may cause insufficient processing time for the robot operation control task. Thus, use the user variables of the DX200 (B variable, I variable, D variable, and R variable) to debug.

In this case, use the API for writing to the variable “mpPutVarData()” in the MotoPlus application to describe debug information, then debug while checking the data in the VARIABLE window on the programming pendant.
## 8 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>The number of loaded files exceeds the limit.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>The memory is insufficient. (Available memory area is less than 3 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)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6</td>
<td>Load failure (Others: application overloaded)</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>
<tr>
<td>4478</td>
<td>MM TASK NO RESPONSE (MotoPlus)</td>
<td></td>
<td>The process requested from the MotoPlus application to the MM task was not completed within the specified time.</td>
</tr>
<tr>
<td>4479</td>
<td>MOTOPLUS MM TASK WATCHDOG ERROR</td>
<td></td>
<td>The man-machine task does not run for 3 seconds or more.</td>
</tr>
</tbody>
</table>