Introduction

This manual explains the procedure for performing motion control by EtherCAT® communication using Renesas Electronics CN032 AC Servo Solution Kit.

The sample program for RZ/T2M is dual core control, and by performing motion control with Cortex®-R52 and performing EtherCAT communication with Cortex®-R52, communication processing is possible without degrading the processing performance of motion control. Data is exchanged between the two cores via the shared memory, and by access right management using the semaphore register, access conflict to the shared memory is prevented.

The sample program for RZ/T2L or RZ/N2L is single core control, and by performing motion control and EtherCAT communication with Cortex®-R52.

EtherCAT communication processing implements the CiA402 drive profile and it is possible to evaluate more practical control.

For details of each function, you can download documents separately from the Renesas Electronics website.

<<Caution when handling the solution board>>
Don’t touch the board while power is supplied because CN032 AC servo solution board contains high voltage circuits.

Target Device

RZ/T2M Group
RZ/T2L Group
RZ/N2L Group

Related Document

- CN032 AC Servo Solution Hardware Manual (for RZ/T2M, RZ/N2L)
- CN032 AC Servo Solution Hardware Manual (for RZ/T2L)
- CN032 AC Servo Solution Firmware Manual
- CN032 AC Servo Solution Startup Guide (for Motion Utility Control)
- CN032 AC Servo Solution Startup Guide (for EtherCAT) (this manual)

- RZ/T2M Group User’s Manual: Hardware
- RZ/T2L Group User’s Manual: Hardware
- RZ/N2L Group User’s Manual: Hardware
Contents

1. Operating Environment.................................................................4
2. Preparation of CN032 AC Servo Solution Kit.............................5
   2.1 Connecting the Logic and Motor Power.................................5
   2.2 Connecting the Motor Windings ..............................................6
   2.3 Connecting the Encoder .......................................................6
   2.4 Connecting the UART communication cable .......................7
   2.4.1 RS232 to USB .................................................................7
   2.4.2 RS485 to USB .................................................................7
   2.4.3 Selection for RS232 and RS485 .........................................8
   2.5 Connecting the JTAG interface ..............................................8
   2.6 Connecting the Ethernet Interface .......................................8
   2.7 Overall connection configuration .......................................9
   2.8 Power supply to the board .................................................9
3. Preparing EtherCAT Communication ...........................................10
   3.1 TwinCAT®3 Installation ......................................................10
   3.2 Copying the ESI File .........................................................14
4. Confirm Communication with TwinCAT®3 ..................................15
   4.1 Starting TwinCAT®3 ...........................................................15
   4.2 Checking the Communication Status ....................................19
   4.3 CiA402 Drive Profile Operation ........................................21
      4.3.1 Operation mode setting ...............................................21
      4.3.2 Profile Position Mode (pp) ............................................22
      4.3.3 Cyclic Synchronous Position Mode (csp) .................25
      4.3.4 Cyclic Synchronous Velocity Mode (csv) ...............27
5. CiA402 Drive Profile .................................................................30
   5.1 Operation Modes ..............................................................30
   5.2 State Transition ...............................................................31
   5.3 Object Dictionary .............................................................31
   5.4 Implementing the Motor Control Program .........................33
6. Appendix ..................................................................................40
   6.1 Preparation in advance ......................................................40
      6.1.1 Power Supply ............................................................40
      6.1.2 Generating the Slave Stack Code .................................41
   6.2 Development Environments Install .....................................44
   6.3 Program Writing Procedure .............................................45
      6.3.1 Program writing by IAR EWARM .............................45

RZ/T2M Group, RZ/T2L Group, RZ/N2L Group Startup Guide (for EtherCAT)
How to Use This Manual

1. Purpose and Target Readers

This manual is designed to provide the user with an understanding of the set-up of the CN032 AC Servo Solution Kit. It is intended for users evaluating the RZ/T2M or RZ/N2L. A basic knowledge of electric circuits, logical circuits, and MCUs is necessary in order to use this manual. The manual comprises a step-by-step description of the installation and initial usage of an application software package that includes the CN032 AC Servo Solution Kit package.

Particular attention should be paid to the precautionary notes when using the manual. These notes occur within the body of the text, at the end of each section, and in the Usage Notes section.

The revision history summarizes the locations of revisions and additions. It does not list all revisions. Refer to the text of the manual for details.

2. List of Abbreviations and Acronyms

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>PC</td>
<td>Personal Computer</td>
</tr>
<tr>
<td>UART</td>
<td>Universal Asynchronous Receiver / Transmitter</td>
</tr>
<tr>
<td>FSP</td>
<td>Flexible Support Package</td>
</tr>
<tr>
<td>FSA</td>
<td>Finite State Automaton</td>
</tr>
<tr>
<td>SSC</td>
<td>Slave Stack Code</td>
</tr>
</tbody>
</table>

All trademarks and registered trademarks are the property of their respective owners.
1. Operating Environment

The sample program in this manual assumes the following environment.

<table>
<thead>
<tr>
<th>Item</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MPU</strong></td>
<td><strong>Series</strong></td>
</tr>
<tr>
<td></td>
<td>RZ/T2M</td>
</tr>
<tr>
<td></td>
<td>Dual Arm Cortex®-R52</td>
</tr>
<tr>
<td></td>
<td>RZ/T2L</td>
</tr>
<tr>
<td></td>
<td>Single Arm Cortex®-R52</td>
</tr>
<tr>
<td></td>
<td>RZ/N2L</td>
</tr>
<tr>
<td></td>
<td>Single Arm Cortex®-R52</td>
</tr>
<tr>
<td><strong>Package</strong></td>
<td>R9A07G075M24: 225-pin FBGA</td>
</tr>
<tr>
<td></td>
<td>R9A07G074M08: 196-pin FBGA</td>
</tr>
<tr>
<td></td>
<td>R9A07G084M04: 225-pin FBGA</td>
</tr>
<tr>
<td><strong>Operating frequency</strong></td>
<td>800MHz</td>
</tr>
<tr>
<td><strong>Operation mode</strong></td>
<td>xSPI0 boot mode (x1 boot Serial flash)</td>
</tr>
<tr>
<td></td>
<td>800MHz</td>
</tr>
<tr>
<td></td>
<td>400MHz</td>
</tr>
<tr>
<td><strong>Operating voltage</strong></td>
<td>3.3V/1.8V/1.1V</td>
</tr>
<tr>
<td><strong>Communication protocol</strong></td>
<td>EtherCAT</td>
</tr>
<tr>
<td></td>
<td>IAR Systems Embedded Workbench for ARM 9.32.2</td>
</tr>
<tr>
<td></td>
<td>Renesas Electronics e²studio 2023-07</td>
</tr>
<tr>
<td></td>
<td>Renesas Electronics FSPSC 2023-07 (*)</td>
</tr>
<tr>
<td></td>
<td>Flexible Support Package (FSP)</td>
</tr>
<tr>
<td></td>
<td>RZT FSP v1.3.0</td>
</tr>
<tr>
<td></td>
<td>RZN FSP v1.3.0</td>
</tr>
<tr>
<td><strong>Emulator</strong></td>
<td>IAR Systems I-jet</td>
</tr>
<tr>
<td></td>
<td>SEGGER J-Link EDU Version 11.0</td>
</tr>
<tr>
<td><strong>SSC Tool</strong></td>
<td>Provided by EtherCAT Technology Group (ETG)</td>
</tr>
<tr>
<td><strong>Software PLC</strong></td>
<td>Beckhoff Automation</td>
</tr>
</tbody>
</table>

(*) FSP SC (Smart Configurator) is a code generation tool for IAR Embedded Workbench.

Table 1-1 Operation Environment

The installation of the integrated development environment, SSC Tool, software PLC has been completed.
2. Preparation of CN032 AC Servo Solution Kit

The first step of getting started with the CN032 AC Servo Solution Kit is connecting the power supply, the motor, the encoder and the communication cable. Follow the steps using the cables and the motor included with the kit.

2.1 Precaution before operation

When using this kit, please kindly comply with the following 1 to 3:

1. Use stable power supply with current 1[A] limit setting to inverter board.
2. Do not use DC12-24[V] power supply jack to controller board.
   ※The power of controller is supplied from the inverter board.
3. Even after turning off, please do not touch the inverter board when high voltage LED(D27) is on which could be about 3 minutes or more.

If motor malfunctions or makes noise during use, please turn off the power to the inverter board immediately.
If this malfunction symptom continues, please contact our sales office or agency.

2.2 Connecting the Logic and Motor Power

The power can be inputted from 100V to 250V AC. From this chapter onwards, the procedure for operating the 220V AC servo motor control system with 220V AC power supply is described.

The power consumption when the controller is idle is around 0.288A but can go up to 0.312A depending on the connected encoders, sensors and other loads.

The motor that is delivered with the kit is prepared for 220V operation. Power is supplied via 12-pin connector P4 on the inverter board as shown in Figure 2-1. The 4-pin cable for P4 is included in the kit.

Note: It is recommended to use separate power supplies for logic and motor.

![Figure 2-1 power connection via P4](image-url)
2.3 Connecting the Motor Windings

The motor windings are connected using P4 on the inverter board. When connecting, you need to consider the signal names printed on the motor cable and the inverter board. Figure 2-2 shows the position of the connections for the motor windings.

Note: silk for U/W on the inverter board is wrong printed.

![Figure 2-2 Connecting the motor windings to P4](image)

2.4 Connecting the Encoder

The encoder is connected to J13 on the controller board via a DB 15 connector. A second encoder (not included in the kit) can be connected to J14 if desired. The position of J13 on the controller board is illustrated in Figure 2-3. The encoder of the motor that is supplied with the kit is prepared for immediate connection. AC Servo Solution Kit is supported Tamagawa encoder.

![Figure 2-3 Connecting the encoder](image)
2.5 Connecting the UART communication cable

2.5.1 RS232 to USB
Communication with the host PC is done via a serial interface available on J5 of the controller board. The RS232 to USB converter is used when connecting. A suitable serial cable is included in the kit. The connection is shown in Figure 2-4.

![Figure 2-4 Serial connection to host PC](image)

Table 2-1 Connection RS232 to USB Converter with controller board

<table>
<thead>
<tr>
<th>RS232 to USB converter</th>
<th>J5</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>TXD(Green)</td>
<td>RXD(J5-1)</td>
<td></td>
</tr>
<tr>
<td>RXD(White)</td>
<td>TXD(J5-2)</td>
<td></td>
</tr>
<tr>
<td>NC</td>
<td>3V3(J5-3)</td>
<td>Connection is not needed</td>
</tr>
<tr>
<td>GND(Black)</td>
<td>GND(J5-4)</td>
<td></td>
</tr>
</tbody>
</table>

2.5.2 RS485 to USB
The RS485 to USB converter is connected the J6 or J8 of the controller board. Table 2-5 is shown the connection with controller board.

![Figure 2-5 Serial connect to host PC](image)
2.5.3 Selection for RS232 and RS485
RS232 or RS485 can be selected by SW2-1 according to Table 2-2.

<table>
<thead>
<tr>
<th>SW2-1</th>
<th>RS232</th>
<th>RS485(default)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
</tr>
</tbody>
</table>

SW2 on the board

2.6 Connecting the JTAG interface
This connection is required for development purposes or in case new firmware has to be loaded. The JTAG connector is using a MIPI10 connector (as specified by IAR). The pins are spaced 1.27mm apart and the connector is polarized.

The connection is shown in Figure 2-6.

2.7 Connecting the Ethernet Interface

An Ethernet connection to the PC or a PLC can be prepared using one of the two RJ-45 connectors J11 or P12. For running the EtherCAT sample program you can use any of the two connectors.
2.8 Overall connection configuration

After connecting chapter 2.1 to 2.6 above, the boards will look like the figure below.

![Figure 2-8 Overall connection configuration](image)

2.9 Power supply to the board

① xSPI0 boot mode setting
   Set SW1 of the controller board to the following.

![Power lamp of the inverter board](image)

② Power supply to the inverter board, and then the red lamp lights up.
   Additionally, the controller board is supplied 5V DC power from inverter board and then LED1 lights up.

![Power LED of the controller board](image)

If LED7 does not light up, program for motor control is not written to the flash memory. So, the program need to be written to the flash memory according to the chapter 6.1 and 6.3.
3. Preparing EtherCAT Communication

This chapter describes a couple of steps required before actually running the EtherCAT sample project on the RZ/T2M, RZ/T2L or RZ/N2L.

3.1 TwinCAT®3 Installation

The TwinCAT®3 installation is straightforward and simple. You can download TwinCAT®3 from

http://www.beckhoff.com/english/download/tc3-downloads.htm?id=1905053019883865

Please keep in mind that screenshot and operation method may change without notice for newer versions of the software. Above URL will lead you to a screen as shown in Figure 3-1.

Select the TE1xxx|Engineering software package for download. If you are no registered user, you can register and download as guest; the registration is free of charge. You will then get an e-mail with a download link. After download, TwinCAT®3 can be installed on your PC. The installation procedure guides you through a number of steps that we will not detail here; simply follow the default installation.
• The installation procedure asks for a serial number. This field can be left empty.
• In the ‘Select Installation Level’ dialog select TwinCAT PLC – IEC 61131-3 PLC system
• In the ‘Select Installation Type’ dialog select 30 days demo version (the functions that are required to run our example last longer than 30 days)
• Install all features
• Target directory for the installation should be C:\TwinCAT

After installation one important point needs to be checked in order to control whether the installation has completed properly. It is the selection of the network adapter for TwinCAT®3. To check that, start TwinCAT®3 and in the start-up screen select “Show Realtime Ethernet Compatible Devices” from the “TwinCAT” menu (see Figure 3-2).

![Figure 3-2 Checking available network adapters for TwinCAT®3]

TwinCAT®3 will then show you a list of the network adapters in your PC, grouped into several categories. There should be at least one adapter listed under “Compatible devices” as shown in Figure 3-2. Select one of the compatible devices and click “Install”. The network adaptor is then moved to the “Installed and ready to use devices (realtime capable)” category.
Check the properties of the network adapter that you use for the EtherCAT connection. Open the Network and Sharing Center in MS Windows, right-click on the network adapter used for EtherCAT and select “Properties”. You will then see a list of supported protocols and/or services for this adapter similar to Figure 3-4.

Please uncheck all items that are not EtherCAT related as shown in the screenshot. Then retry to scan for devices.
Figure 3-4 Setting the network adapter used for EtherCAT
3.2 Copying the ESI File

Obtain the ESI (EtherCAT Slave Information) file "Renesas_CN032_AC_Servo_Solution_CiA402.xml" file from the following location shown in Figure 3-5.

Case of the AC Servo Solution Kit (RZ/T2M)
"\12an0123xxXXXX-cn032-ac-servo-solution\Software\Firmware\rzt2m\Common\ethercat\src\r_ecat\utilities\esi"

Case of the AC Servo Solution Kit (RZ/T2L)
"\12an0123xxXXXX-cn032-ac-servo-solution\Software\Firmware\rzt2l\Common\ethercat\src\r_ecat\utilities\esi"

Case of the AC Servo Solution Kit (RZ/N2L)
"\12an0123xxXXXX-cn032-ac-servo-solution\Software\Firmware\rzn2l\Common\ethercat\src\r_ecat\utilities\esi"

Then, copy the obtained file to the following folder, in which TwinCAT®3 has been installed:
"\TwinCAT\3.x\Config\Io\EtherCAT"
4. Confirm Communication with TwinCAT®3

4.1 Starting TwinCAT®3

(1) Start the "TwinCAT XAE" program by one of the following methods.

1. From the task tray, select [TwinCAT Config Mode] > [TwinCAT XAE (VS2013)]
2. From the start menu, select [All Programs] > [Beckhoff] > [TwinCAT 3] > [TwinCAT XAE (VS2013)]

(2) After starting the program, select [New TwinCAT Project] as shown in Figure 4-1 and create a new project of type TwinCAT XAE Project. Note that the creation of the new project may take several 10 seconds and watch the progress bar in the lower right corner of the TwinCAT®3 window.

![TwinCAT®3 start-up screen](image)

Figure 4-1 TwinCAT®3 start-up screen
When the project is created, open the “TWINCAT” menu, chose “EtherCAT Devices” and execute “Reload Device Descriptions”. This ensures, that the device description (aka .ESI file), that you have copied into the TwinCAT®3 installation file structure in chapter 3.2 can really be used in our project.

Make sure that the CN032 AC Servo Solution firmware project on the RZ/T2M, RZ/T2L or RZ/N2L is up and running before the next steps in TwinCAT®3. When the project is not up and running, see chapter 6 to execute the project.
(3) Right-click [Devices] and select [Scan] (Figure 4-3)

After you have clicked [Scan], you will be prompted that not all devices in the network can be found automatically. Click ok. TwinCAT®3 will then show you a list of available devices in the network. Select the [EtherCAT] device connected to the network controller that you use for the CN032 AC Servo Solution board (see Figure 4-4) and click ok.

![Figure 4-3 Scanning the network for new devices](image)

(4) Select only [EtherCAT] and click [OK].

![Figure 4-4 Selecting the proper EtherCAT device](image)
When scanning the device, the CiA402's ESI is used, so the axis configuration settings are displayed. Select [NC-Configuration].

After clicking ok you will be prompted, whether you want to "scan for boxes" and to "activate Free Run". In both cases click "yes". When Free Run was activated, the RUN and ERR LEDs begin flashing and the L/A IN LED is flashing rapidly.

Note that if you repeat the “Scan” process in the same project in the same environment, you will be informed that no new devices were found. This is no bad thing as there has been no change in the network.

(5) The scan result is displayed as "Drive x (CN032 AC Servo Solution CiA402)".

When the scan result is displayed as "Box 1 (Pxxxxxxxx Rxxxxxxxx)" as like Figure 4-6. EEPROM data should be written according to chapter 6.5. After EEPROM write, Box 1 will be displayed as "Drive x (CN032 AC Servo Solution CiA402)". EEPROM rewrite is described in chapter 6.5.

Your TwinCAT®3 project is now ready for operation.
4.2 Checking the Communication Status

To confirm that TwinCAT®3 and the software on the RZ/T2M are running properly, it is recommended to check the communication status in TwinCAT®3. For the check, double click the "Drive 2 (CN032 AC Servo Solution CiA402)" device (1), select the "Online" tab (2) and check, whether the current state is “OP” (3).

In case that the current state is not “OP” (i.e. “INIT”) please restart establishing the communication with the [Restart TwinCAT (Config Mode)] button shown in Figure 4-8. You will then be prompted for

- Restarting TwinCAT System in Config Mode
- Loading I/O Devices
- and Activating Free Run

In all three dialogues simply click “yes”. When communication is re-established, the device state in the “Online” tab must change to “OP” as shown in Figure 4-7.
Figure 4-8 Restarting TwinCAT®3 system in config mode
4.3 CiA402 Drive Profile Operation

The operation mode of CiA402 drive profile supported by CN032 AC Servo Solution Kit is shown as 5.1 Operation Modes.

4.3.1 Operation mode setting

When change the operation mode, following the steps below.

1. Select “Drive x (CN032 AC Servo Solution CiA402)”.
2. Select “Slots” tab.
3. Select “Motor 1”.
4. Push “X” button to remove current module setting.
5. Select any module from multiple modules supported CN032 AC Servo Solution.
6. Push “<” button to apply selected module setting.

Check the module of Motor 1 was changed to module you selected and then push “ ” button to reload the device.
4.3.2 Profile Position Mode (pp)

Make sure Module 1 is applied to the operation of Profile position mode (1). Expand the “Drive x (CN032_AC_Servo_Solution_CiA402)” tree and expand “Module 1” as far as possible as shown in Figure 4-9. For the sake of comfort add the input values to an extra watch window by right-clicking at a value (2) and selecting “Add to watch” (3).

If you repeat this exercise for all five input values for “Module 1”, you will get a permanently updated watch window as illustrated in Figure 4-10.

Note: “Input” and “output” are always seen from the perspective of the EtherCAT PLC respectively the TwinCAT®3 GUI. Therefore you send commands to the drive (i.e. the CN032 AC Servo Solution) using output variables, while the feedback from the drive is reflected by the input variables.

The values listed under “Output” in the tree structure in TwinCAT®3 can be altered online. By clicking on “Controlword” more detailed information about the variable can be shown in the TwinCAT®3 window (1). A value for “Controlword” can then be sent to the drive via EtherCAT by clicking on the “Online” tab (2) and on the “Write” button (3). This opens a dialog for entering “Controlword” in different formats. Please enter 128 in decimal format. Closing the dialog with “Ok” (4) sends the value via EtherCAT.
Controlword 128 (or 0x80) is a kind of reset command that sets the drive to an initial state. If the error code in the watch window has been different from 0 before sending 128 (as in Figure 4-11), it should be 0 afterwards. Statusword should show the value 545.

You can start drive operation by sending the Controlword 15 (or 0x0F).

Next, send values for “Target position”, “Profile velocity”, “Profile acceleration” and “Profile deceleration” with the same method. Good values for a first experiment are

- 800000 for target position
- 50 for profile velocity
- 100 for profile acceleration and deceleration
- 0 for motion profile type, profile jerk 1 and 2

Figure 4-12 Sending a Controlword for profile velocity illustrates this for “Profile velocity”; use the same method for the other values. Having sent these values you can finally start motor spinning by sending the Controlword 31 (0x1F). The motor will now start spinning with the preset acceleration, velocity and deceleration until the target position is reached. In the watch window you can observe current speed and position. after motor spinning, Statusword is changed to 545 by sending the Controlword 6 (0x6). Figure 4-13 shows you three screen shots of the watch window before, during and after operation.

In case that the motor does not start spinning, you may try to send 128 again and/or to reset the CN032 AC Servo Solution board, which will automatically re-connect to TwinCAT®3.
Figure 4-12 Sending a Controlword for profile velocity

1) before spinning

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
<th>Type</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error code</td>
<td>0</td>
<td>UNINT</td>
<td>I/O.Devices.Device 2 (EtherCAT).Drive 2 (CNSN22 AC Servo Solution_GA4002)</td>
</tr>
<tr>
<td>Statusword</td>
<td>545</td>
<td>UNINT</td>
<td>I/O.Devices.Device 2 (EtherCAT).Drive 2 (CNSN22 AC Servo Solution_GA4002)</td>
</tr>
<tr>
<td>Modes of operation display</td>
<td>1</td>
<td>USINT</td>
<td>I/O.Devices.Device 2 (EtherCAT).Drive 2 (CNSN22 AC Servo Solution_GA4002)</td>
</tr>
<tr>
<td>Position actual value</td>
<td>0</td>
<td>DINT</td>
<td>I/O.Devices.Device 2 (EtherCAT).Drive 2 (CNSN22 AC Servo Solution_GA4002)</td>
</tr>
<tr>
<td>Velocity actual value</td>
<td>0</td>
<td>DINT</td>
<td>I/O.Devices.Device 2 (EtherCAT).Drive 2 (CNSN22 AC Servo Solution_GA4002)</td>
</tr>
</tbody>
</table>

2) while spinning

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
<th>Type</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error code</td>
<td>4663</td>
<td>UNINT</td>
<td>I/O.Devices.Device 2 (EtherCAT).Drive 2 (CNSN22 AC Servo Solution_GA4002)</td>
</tr>
<tr>
<td>Statusword</td>
<td>545</td>
<td>UNINT</td>
<td>I/O.Devices.Device 2 (EtherCAT).Drive 2 (CNSN22 AC Servo Solution_GA4002)</td>
</tr>
<tr>
<td>Modes of operation display</td>
<td>1</td>
<td>USINT</td>
<td>I/O.Devices.Device 2 (EtherCAT).Drive 2 (CNSN22 AC Servo Solution_GA4002)</td>
</tr>
<tr>
<td>Position actual value</td>
<td>13595</td>
<td>DINT</td>
<td>I/O.Devices.Device 2 (EtherCAT).Drive 2 (CNSN22 AC Servo Solution_GA4002)</td>
</tr>
<tr>
<td>Velocity actual value</td>
<td>44</td>
<td>DINT</td>
<td>I/O.Devices.Device 2 (EtherCAT).Drive 2 (CNSN22 AC Servo Solution_GA4002)</td>
</tr>
</tbody>
</table>

3) after spinning

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
<th>Type</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error code</td>
<td>1591</td>
<td>UNINT</td>
<td>I/O.Devices.Device 2 (EtherCAT).Drive 2 (CNSN22 AC Servo Solution_GA4002)</td>
</tr>
<tr>
<td>Statusword</td>
<td>545</td>
<td>UNINT</td>
<td>I/O.Devices.Device 2 (EtherCAT).Drive 2 (CNSN22 AC Servo Solution_GA4002)</td>
</tr>
<tr>
<td>Modes of operation display</td>
<td>1</td>
<td>USINT</td>
<td>I/O.Devices.Device 2 (EtherCAT).Drive 2 (CNSN22 AC Servo Solution_GA4002)</td>
</tr>
<tr>
<td>Position actual value</td>
<td>150000</td>
<td>DINT</td>
<td>I/O.Devices.Device 2 (EtherCAT).Drive 2 (CNSN22 AC Servo Solution_GA4002)</td>
</tr>
<tr>
<td>Velocity actual value</td>
<td>44</td>
<td>DINT</td>
<td>I/O.Devices.Device 2 (EtherCAT).Drive 2 (CNSN22 AC Servo Solution_GA4002)</td>
</tr>
</tbody>
</table>

Figure 4-13 Watch value changes while motor spins
4.3.3 Cyclic Synchronous Position Mode (csp)

Confirm that the operation mode is “csp”. Otherwise, change the operation mode to “csp” by following 4.3.1 Operation mode setting.

Just like the steps described for pp mode, add the input values to an extra watch window for the sake of comfort.

Follow steps (1) to (3) to open a dialog to send the value of Controlword.
Follow steps A) to B) to send the Controlword value and make the motor controllable.

A) Confirm that Statusword is “4641”. Otherwise, set Controlword to 128 to reset Statusword.

B) Set Controlword to “15” and then Statusword will be “4659” (Power on) and then immediately change to “4663” (Servo on).

Follow steps (1) to (3) to open a dialog to send the value of Target Position. Enter the Target Position value and closing the dialog with “Ok” (4) sends the value via EtherCAT.
Then, the motor rotates until the Position actual value counts up to the Target Position value.

<table>
<thead>
<tr>
<th>ADS Symbol Watch</th>
<th>Value</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error code</td>
<td>0</td>
<td>UINT</td>
</tr>
<tr>
<td>Statusword</td>
<td>4641</td>
<td>UINT</td>
</tr>
<tr>
<td>Modes of operation display</td>
<td>8</td>
<td>USINT</td>
</tr>
<tr>
<td>Position actual value</td>
<td>15383</td>
<td>DINT</td>
</tr>
<tr>
<td>Velocity actual value</td>
<td>3</td>
<td>DINT</td>
</tr>
</tbody>
</table>

When stop the motor control, set Controlword to “6” and then Statusword will be “4641”.

4.3.4 Cyclic Synchronous Velocity Mode (csv)

Confirm that the operation mode is “csp”. Otherwise, change the operation mode to “csp” by following 4.3.1 Operation mode setting.

Just like the steps described for pp mode, add the input values to an extra watch window for the sake of comfort.
Follow steps (1) to (3) to open a dialog to send the value of Controlword.

Follow steps A) to B) to send the Controlword value and make the motor controllable.

A) Confirm that Statusword is “4641”. Otherwise, set Controlword to 128 to reset Statusword.

B) Set Controlword to “15” and then Statusword will be “4659” (Power on) and then immediately change to “4663” (Servo on).
Follow steps (1) to (3) to open a dialog to send the value of Target Velocity. Enter the Target Velocity value and closing the dialog with “Ok” (4) sends the value via EtherCAT.

Then, the motor rotates while controlling the Velocity actual value to the Target Velocity value.

When stop the motor control, set Controlword to “6” and then Statusword will be “4641”.

You can now modify the parameters as you like for further experiments. The full “command set” that is covered by the Controlword and the specific meaning of error codes or Statusword can be studied in the “CANopen Device Profile Drives and Motion Control” control document, that is available from the CAN in Automation organization at https://www.can-cia.org/can-knowledge/canopen/cia402/

Please note that the document can only be downloaded by organization members. Copies of old versions are vagabonding on the www.
5. CiA402 Drive Profile

The CiA402 drive profile is a device profile for driving motors and motion control and mainly defines functional operations for servo drives, sine-wave inverters and stepping motor controllers. In this profile, the multiple operation modes and corresponding parameters are defined as an object dictionary. Also, Finite State Automaton (FSA) to define the internal and external behavior in every state is included. When changing the state, the result after transition is reflected in the status word object that shows the current state by specifying the state through the control word object. The control word and various command values (such as speed) are assigned to RxPDO, and the status word and various real values (such as position) are assigned to TxPDO. Please see the contents of the CiA402 standard for more details.

![CiA402 Communication Flow](image)

**Figure 5-1 CiA402 Communication Flow**

5.1 Operation Modes

In the application note, the following modes are supported from among the operation modes defined in the CiA402 standard.

<table>
<thead>
<tr>
<th>Operation Mode</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profile position mode</td>
<td>Yes</td>
</tr>
<tr>
<td>Velocity mode (frequency converter)</td>
<td>No</td>
</tr>
<tr>
<td>Profile velocity mode</td>
<td>No</td>
</tr>
<tr>
<td>Profile torque mode</td>
<td>No</td>
</tr>
<tr>
<td>Homing mode</td>
<td>No</td>
</tr>
<tr>
<td>Interpolated position mode</td>
<td>No</td>
</tr>
<tr>
<td>Cyclic synchronous position mode</td>
<td>Yes</td>
</tr>
<tr>
<td>Cyclic synchronous velocity mode</td>
<td>Yes</td>
</tr>
<tr>
<td>Cyclic synchronous torque mode</td>
<td>No</td>
</tr>
<tr>
<td>Cyclic synchronous torque mode with commutation angle</td>
<td>No</td>
</tr>
<tr>
<td>Manufacturer specific mode</td>
<td>No</td>
</tr>
</tbody>
</table>

**Table 5-1 List of Supported Operation Modes**
5.2 State Transition
In this application note, the following is supported as FSA defined in the CiA402 standard.

![State Transition Diagram](image)

**Figure 5-2 CiA402 State Transition Diagram**

5.3 Object Dictionary
The following is the list of the object dictionaries supported in this application note.

<table>
<thead>
<tr>
<th>Operation Mode</th>
<th>OBJECT Name</th>
<th>INDEX</th>
<th>Category</th>
<th>Access</th>
<th>Data Type</th>
<th>PDO Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cyclic synchronous position mode</td>
<td>Position actual value</td>
<td>0x6064</td>
<td>Mandatory</td>
<td>ro</td>
<td>INT32</td>
<td>Yes</td>
</tr>
<tr>
<td>Cyclic synchronous position mode</td>
<td>Following error window</td>
<td>0x6065</td>
<td>Optional</td>
<td>rw</td>
<td>UINT32</td>
<td>No</td>
</tr>
<tr>
<td>Cyclic synchronous velocity mode</td>
<td>Velocity actual value</td>
<td>0x606C</td>
<td>Conditional</td>
<td>ro</td>
<td>INT32</td>
<td>Yes</td>
</tr>
<tr>
<td>Cyclic synchronous velocity mode</td>
<td>Max torque</td>
<td>0x6072</td>
<td>Optional</td>
<td>rw</td>
<td>UINT16</td>
<td>Yes</td>
</tr>
<tr>
<td>Cyclic synchronous velocity mode</td>
<td>Torque actual value</td>
<td>0x6077</td>
<td>Conditional</td>
<td>ro</td>
<td>INT16</td>
<td>Yes</td>
</tr>
<tr>
<td>Cyclic synchronous velocity mode</td>
<td>Target position</td>
<td>0x607A</td>
<td>Optional</td>
<td>rw</td>
<td>INT32</td>
<td>Yes</td>
</tr>
<tr>
<td>Cyclic synchronous velocity mode</td>
<td>Software position limit</td>
<td>0x607D</td>
<td>Optional</td>
<td>c, rw</td>
<td>INT32</td>
<td>No</td>
</tr>
<tr>
<td>Cyclic synchronous velocity mode</td>
<td>Position offset</td>
<td>0x60B0</td>
<td>Optional</td>
<td>rw</td>
<td>INT32</td>
<td>Yes</td>
</tr>
<tr>
<td>Cyclic synchronous velocity mode</td>
<td>Velocity offset</td>
<td>0x60B1</td>
<td>Optional</td>
<td>rw</td>
<td>INT32</td>
<td>Yes</td>
</tr>
<tr>
<td>Cyclic synchronous velocity mode</td>
<td>Torque offset</td>
<td>0x60B2</td>
<td>Optional</td>
<td>rw</td>
<td>INT16</td>
<td>Yes</td>
</tr>
<tr>
<td>Cyclic synchronous velocity mode</td>
<td>Following error actual value</td>
<td>0x60F4</td>
<td>Optional</td>
<td>ro</td>
<td>INT32</td>
<td>Yes</td>
</tr>
<tr>
<td>Cyclic synchronous velocity mode</td>
<td>Target velocity</td>
<td>0x60FF</td>
<td>Conditional</td>
<td>rw</td>
<td>INT32</td>
<td>Yes</td>
</tr>
<tr>
<td>Function Group</td>
<td>OBJECT Name</td>
<td>INDEX</td>
<td>Category</td>
<td>Access</td>
<td>Data Type</td>
<td>PDO Mapping</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------------</td>
<td>---------</td>
<td>----------</td>
<td>--------</td>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>Torque Limiting</strong></td>
<td>Positive torque limit value</td>
<td>0x60E0</td>
<td>Conditional</td>
<td>rw</td>
<td>UINT16</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Negative torque limit value</td>
<td>0x60E1</td>
<td>Conditional</td>
<td>rw</td>
<td>UINT16</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Homing</strong></td>
<td>Home Offset</td>
<td>0x607C</td>
<td>Optional</td>
<td>rw</td>
<td>INT32</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Homing speeds</td>
<td>0x6099</td>
<td>Conditional</td>
<td>c,rw</td>
<td>UINT32</td>
<td>No</td>
</tr>
<tr>
<td><strong>Touch Probe</strong></td>
<td>Touch probe function</td>
<td>0x60B8</td>
<td>Optional</td>
<td>rw</td>
<td>UINT16</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Touch probe status</td>
<td>0x60B9</td>
<td>Optional</td>
<td>ro</td>
<td>UINT16</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Touch probe pos 1 pos value</td>
<td>0x60BA</td>
<td>Optional</td>
<td>ro</td>
<td>INT32</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Touch probe pos 2 pos value</td>
<td>0x60BC</td>
<td>Optional</td>
<td>ro</td>
<td>INT32</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Gear ratio</strong></td>
<td>Gear ratio</td>
<td>0x6091</td>
<td>Optional</td>
<td>c,rw</td>
<td>UINT32</td>
<td>No</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Other object</strong></th>
<th>OBJECT Name</th>
<th>INDEX</th>
<th>Category</th>
<th>Access</th>
<th>Data Type</th>
<th>PDO Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Controlling the power drive system</strong></td>
<td>Error code</td>
<td>0x603F</td>
<td>Optional</td>
<td>ro</td>
<td>UINT16</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Controlword</td>
<td>0x6040</td>
<td>Mandatory</td>
<td>rw</td>
<td>UINT16</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Statusword</td>
<td>0x6041</td>
<td>Mandatory</td>
<td>ro</td>
<td>UINT16</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Quick stop option code</td>
<td>0x605A</td>
<td>Optional</td>
<td>rw</td>
<td>INT16</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Shutdown option code</td>
<td>0x605B</td>
<td>Optional</td>
<td>rw</td>
<td>INT16</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Disable operation option code</td>
<td>0x605C</td>
<td>Optional</td>
<td>rw</td>
<td>INT16</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Halt option code</td>
<td>0x605D</td>
<td>Optional</td>
<td>rw</td>
<td>INT16</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Fault reaction option code</td>
<td>0x605E</td>
<td>Optional</td>
<td>rw</td>
<td>INT16</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Modes of operation</td>
<td>0x6060</td>
<td>Optional</td>
<td>rw</td>
<td>INT8</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Modes of operation disp</td>
<td>0x6061</td>
<td>Optional</td>
<td>ro</td>
<td>INT8</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Supported drive modes</td>
<td>0x6502</td>
<td>Mandatory</td>
<td>ro</td>
<td>INT32</td>
<td>No</td>
</tr>
<tr>
<td><strong>General object</strong></td>
<td>Motor type</td>
<td>0x6402</td>
<td>Optional</td>
<td>rw</td>
<td>INT16</td>
<td>No</td>
</tr>
<tr>
<td><strong>Position control function</strong></td>
<td>Position demand value</td>
<td>0x6062</td>
<td>Optional</td>
<td>ro</td>
<td>INT32</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Position actual internal value</td>
<td>0x6063</td>
<td>Optional</td>
<td>ro</td>
<td>INT32</td>
<td>No</td>
</tr>
<tr>
<td><strong>Optional application FE</strong></td>
<td>Position window</td>
<td>0x6067</td>
<td>Optional</td>
<td>rw</td>
<td>UINT32</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Digital inputs</td>
<td>0x60FD</td>
<td>Optional</td>
<td>ro</td>
<td>UINT32</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Digital outputs</td>
<td>0x60FE</td>
<td>Optional</td>
<td>c,rw</td>
<td>UINT32</td>
<td>No,Yes</td>
</tr>
</tbody>
</table>

Table 5-2 List of Supported Object Dictionaries
5.4 Implementing the Motor Control Program

According to the CiA402 standard from the list of CiA402 protocol stack I/F functions in Table 5-3, implement the motor control application. Each function links the number of each state transition of CiA402 FSA shown in Figure 5-2 and the corresponding function is called in case of state transition. In each function, describe the processing that calls the motor control program or the relevant processing of the main CPU.

<table>
<thead>
<tr>
<th>CiA402_StateTransition1</th>
<th>Description</th>
<th>This function is used when state transition 1 has occurred. Describe the operation in the case of the state transition. This function issues a POWEROFF command.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Usage</td>
<td>#include &quot;cia402appl.h&quot;</td>
</tr>
<tr>
<td></td>
<td>Parameters</td>
<td>TCI402Axis *pCiA402Axis</td>
</tr>
</tbody>
</table>
|                          | Return Value| 0 Normal end  
1 Error  |
|                          | Remark      | In the case of error occurrence during processing, exit the function by setting the appropriate values for each object in accordance with the CiA402 standard. If 1 is set to return value, state transition does not occur. |

<table>
<thead>
<tr>
<th>CiA402_StateTransition2</th>
<th>Description</th>
<th>This function is used when state transition 2 has occurred. Describe the operation in the case of the state transition. This function is not used.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Usage</td>
<td>#include &quot;cia402appl.h&quot;</td>
</tr>
<tr>
<td></td>
<td>Parameters</td>
<td>TCI402Axis *pCiA402Axis</td>
</tr>
</tbody>
</table>
|                          | Return Value| 0 Normal end  
1 Error  |
|                          | Remark      | In the case of error occurrence during processing, exit the function by setting the appropriate values for each object in accordance with the CiA402 standard. If 1 is set to return value, state transition does not occur. |
CiA402_StateTransition3

Description
This function is used when state transition 3 has occurred.
Describe the operation in the case of the state transition.
This function issues a POWERON command and energizes the motor.

Usage
#include "cia402appl.h"

Parameters
TCiA402Axis *pCiA402Axis

Return Value
0 Normal end
1 Error

Remark
In the case of error occurrence during processing, exit the function by setting
the appropriate values for each object in accordance with the CiA402 standard.
If 1 is set to return value, state transition does not occur.

CiA402_StateTransition4

Description
This function is used when state transition 4 has occurred
Describe the operation in the case of the state transition.
This function issues a SERVOON command and put the motor in torque on state.

Usage
#include "cia402appl.h"

Parameters
TCiA402Axis *pCiA402Axis

Return Value
0 Normal end
1 Error

Remark
In the case of error occurrence during processing, exit the function by setting
the appropriate values for each object in accordance with the CiA402 standard.
If 1 is set to return value, state transition does not occur.

CiA402_StateTransition5

Description
This function is used when state transition 5 has occurred.
Describe the operation in the case of the state transition.
This function issues a SERVOOFF command and release the motor from torque on state.

Usage
#include "cia402appl.h"

Parameters
TCiA402Axis *pCiA402Axis

Return Value
0 Normal end
1 Error

Remark
In the case of error occurrence during processing, exit the function by setting
the appropriate values for each object in accordance with the CiA402 standard.
If 1 is set to return value, state transition does not occur.
<table>
<thead>
<tr>
<th>CiA402_StateTransition6</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>This function is used when state transition 6 has occurred.</td>
<td></td>
</tr>
<tr>
<td>Describe the operation in the case of the state transition.</td>
<td></td>
</tr>
<tr>
<td>This function issues a POWEROFF command.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CiA402_StateTransition7</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>This function is used when state transition 7 has occurred.</td>
<td></td>
</tr>
<tr>
<td>Describe the operation in the case of the state transition.</td>
<td></td>
</tr>
<tr>
<td>This function is not used.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CiA402_StateTransition8</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>This function is used when state transition 8 has occurred.</td>
<td></td>
</tr>
<tr>
<td>Describe the operation in the case of the state transition.</td>
<td></td>
</tr>
<tr>
<td>This function issues a POWEROFF command.</td>
<td></td>
</tr>
</tbody>
</table>

**CiA402_StateTransition6**

**Usage**

```c
#include "cia402app1.h"
```

**Parameters**

```c
TCiA402Axis *pCiA402Axis
```

**Return Value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal end</td>
</tr>
<tr>
<td>1</td>
<td>Error</td>
</tr>
</tbody>
</table>

**Remark**

In the case of error occurrence during processing, exit the function by setting the appropriate values for each object in accordance with the CiA402 standard.
If 1 is set to return value, state transition does not occur.

**CiA402_StateTransition7**

**Usage**

```c
#include "cia402app1.h"
```

**Parameters**

```c
TCiA402Axis *pCiA402Axis
```

**Return Value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal end</td>
</tr>
<tr>
<td>1</td>
<td>Error</td>
</tr>
</tbody>
</table>

**Remark**

In the case of error occurrence during processing, exit the function by setting the appropriate values for each object in accordance with the CiA402 standard.
If 1 is set to return value, state transition does not occur.

**CiA402_StateTransition8**

**Usage**

```c
#include "cia402app1.h"
```

**Parameters**

```c
TCiA402Axis *pCiA402Axis
```

**Return Value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal end</td>
</tr>
<tr>
<td>1</td>
<td>Error</td>
</tr>
</tbody>
</table>

**Remark**

In the case of error occurrence during processing, exit the function by setting the appropriate values for each object in accordance with the CiA402 standard.
If 1 is set to return value, state transition does not occur.
### CiA402 StateTransition9

**Description**
This function is used when state transition 9 has occurred. Describe the operation in the case of the state transition. This function issues a POWEROFF command.

**Usage**
```
#include "cia402app1h"
```

**Parameters**
```
TCiA402Axis *pCiA402Axis
```

**Return Value**
- 0 Normal end
- 1 Error

**Remark**
In the case of error occurrence during processing, exit the function by setting the appropriate values for each object in accordance with the CiA402 standard. If 1 is set to return value, state transition does not occur.

### CiA402 StateTransition10

**Description**
This function is used when state transition 10 has occurred. Describe the operation in the case of the state transition. This function issues a POWEROFF command.

**Usage**
```
#include "cia402app1h"
```

**Parameters**
```
TCiA402Axis *pCiA402Axis
```

**Return Value**
- 0 Normal end
- 1 Error

**Remark**
In the case of error occurrence during processing, exit the function by setting the appropriate values for each object in accordance with the CiA402 standard. If 1 is set to return value, state transition does not occur.

### CiA402 StateTransition11

**Description**
This function is used when state transition 11 has occurred. Describe the operation in the case of the state transition. This function issues a QUICKSTOP command.

**Usage**
```
#include "cia402app1h"
```

**Parameters**
```
TCiA402Axis *pCiA402Axis
```

**Return Value**
- 0 Normal end
- 1 Error

**Remark**
In the case of error occurrence during processing, exit the function by setting the appropriate values for each object in accordance with the CiA402 standard. If 1 is set to return value, state transition does not occur.
### CiA402_StateTransition12

**Description**
This function is used when state transition 12 has occurred. Describe the operation in the case of the state transition. This function issues a POWEROFF command.

**Usage**
```
#include "cia402appl.h"
```

**Parameters**
```
TCiA402Axis *pCiA402Axis
```

**Return Value**
```
0       Normal end
1       Error
```

**Remark**
In the case of error occurrence during processing, exit the function by setting the appropriate values for each object in accordance with the CiA402 standard. If 1 is set to return value, state transition does not occur.

### CiA402_LocalError

**Description**
This function is used when state transition 13 has occurred. Describe the operation in the case of the state transition. This function is called if an error was detected.

**Usage**
```
#include "cia402appl.h"
```

**Parameters**
```
UINT16 ErrorCode
```

**Return Value**
```
none
```

**Remark**
If the error corresponding to state transition 13 occurs, call this function after processing required and saving data at error location.

### CiA402_StateTransition14

**Description**
This function is used when state transition 14 has occurred. Describe the operation in the case of the state transition. This function issues a POWEROFF command.

**Usage**
```
#include "cia402appl.h"
```

**Parameters**
```
TCiA402Axis *pCiA402Axis
```

**Return Value**
```
0       Normal end
1       Error
```

**Remark**
In the case of error occurrence during processing, exit the function by setting the appropriate values for each object in accordance with the CiA402 standard. If 1 is set to return value, state transition does not occur.
### CiA402_StateTransition15

**Description**
This function is used when state transition 15 has occurred.
Describe the operation in the case of the state transition.
This function is called when transitioning from Fault state to Switch on disabled state.

**Usage**
#include "cia402appl.h"

**Parameters**
TCiA402Axis *pCiA402Axis

**Return Value**
0  Normal end
1  Error

**Remark**
In the case of error occurrence during processing, exit the function by setting the appropriate values for each object in accordance with the CiA402 standard.
If 1 is set to return value, state transition does not occur.

### CiA402_StateTransition16

**Description**
This function is used when state transition 16 has occurred
Describe the operation in the case of the state transition.
This function issues a SERVOON command and put the motor in torque on state.

**Usage**
#include "cia402appl.h"

**Parameters**
TCiA402Axis *pCiA402Axis

**Return Value**
0  Normal end
1  Error

**Remark**
In the case of error occurrence during processing, exit the function by setting the appropriate values for each object in accordance with the CiA402 standard.
If 1 is set to return value, state transition does not occur.
### APPL_MOTOR_MotionControl_Main

**Description**
Implement the motion control code when the state of CiA402 FSA is "Operation enabled". Describe the process for each mode of operation.

**Usage**
#include "cia402appLh"

**Parameters**
TCiA402Axis *pCiA402Axis

**Return Value**
0 Normal end  
1 Error

**Remark**
At the initial state, this function is described in "main.c" and calls "CiA402_DummyMotionControl" function for reference.

---

**Table 5-3 List of CiA402 Protocol Stack I/F Functions**
6. Appendix

6.1 Preparation in advance
This chapter is shown preparation for writing a program to flash memory and debugging

6.1.1 Power Supply
Configure the board connection according to the chapter 2.

③ xSPI0 boot mode setting
   Set SW1 of the controller board to the following.

④ Power supply to the inverter board, and then the red lamp lights up.
   Additionally, the controller board is supplied 5V DC power from inverter board and then LED1 lights up.
6.1.2 Generating the Slave Stack Code

To run the EtherCAT communication you must generate the EtherCAT slave code using a special code generation tool provided by Beckhoff. The slave code generation is performed based on the formal description of the slave properties in the ESI file that you have copied in the previous chapter.

Make sure the version of the installed SSC tool is V5.12. Do not update even if receiving update notification when starting the tool.

(1) Start SSC Tool

Start SSC Tool by double-clicking the SSC project file "CN032 AC Servo Solution EtherCAT CiA402.esp" in the following location:

Case of the AC Servo Solution Kit (RZ/T2M)

\r12an0123xxXXXX-cn032-ac-servo-solution\Software\Firmware\rz2m\Common\ethercat\src\r_ecat\utilities\ssc_config"

Case of the AC Servo Solution Kit (RZ/T2L)

\r12an0123xxXXXX-cn032-ac-servo-solution\Software\Firmware\rz2l\Common\ethercat\src\r_ecat\utilities\ssc_config"

Case of the AC Servo Solution Kit (RZ/N2L)

\r12an0123xxXXXX-cn032-ac-servo-solution\Software\Firmware\rn2l\Common\ethercat\src\r_ecat\utilities\ssc_config"

(2) Generate a slave stack code

In SSC Tool, select [Project] > [Create new Slave Files], and then click [Start] as shown in Figure 6-1. The slave stack code is then generated in the following location:

Case of the AC Servo Solution Kit (RZ/T2M)

"\r12an0123xxXXXX-cn032-ac-servo-solution\Software\Firmware\rz2m\Common\ethercat\src\r_ecat\utilities\ssc_config\Src"

Case of the AC Servo Solution Kit (RZ/T2L)

"\r12an0123xxXXXX-cn032-ac-servo-solution\Software\Firmware\rz2l\Common\ethercat\src\r_ecat\utilities\ssc_config\Src"

Case of the AC Servo Solution Kit (RZ/N2L)

"\r12an0123xxXXXX-cn032-ac-servo-solution\Software\Firmware\rn2l\Common\ethercat\src\r_ecat\utilities\ssc_config\Src"

Then, click [OK], and then [Close] to terminate the SSC Tool.
(3) Prepare a patch command

Last but not least it is required to install a patch file on your computer and to make it usable as an environment variable. Please download the GNU Patch program (version: 2.5.9 or later) from the following website:

http://gnuwin32.sourceforge.net/packages/patch.htm

Then, store the patch.exe file, that is part of the download somewhere, and add the (preferably short) storage path to an environmental variable: In the Windows Control Panel go to ‘System’ and then to ‘Advanced system settings’. In the ‘System Properties’ dialog select ‘Environment Variables’ as illustrated in Figure 6-2.

Figure 6-1 Creating the EtherCAT slave stack code

Figure 6-2 Adding a new environment variable
Select ‘Path’ in the list of system variables and then ‘Edit’. Add the path, where you have stored the patch file (in our case C:\Program Files (x86)\GnuWin32\bin), to the already existing search path. Click ‘Ok’ twice to accept the changes.

**Figure 6-3 Editing the system variable ‘Path’**

(4) **Apply the patch**

Next, the patch has to be executed once. Double-click the ‘apply_patch.bat’ file in

**Case of the AC Servo Solution Kit (RZ/T2M)**

“r12an0123xxXXXX-cn032-ac-servo-solution\Software\Firmware\rzt2m\Common\ethercat\src\r_ecat\utilities\batch_files”

**Case of the AC Servo Solution Kit (RZ/T2L)**

“r12an0123xxXXXX-cn032-ac-servo-solution\Software\Firmware\rzt2l\Common\ethercat\src\r_ecat\utilities\batch_files”

**Case of the AC Servo Solution Kit (RZ/N2L)**

“r12an0123xxXXXX-cn032-ac-servo-solution\Software\Firmware\rzn2l\Common\ethercat\src\r_ecat\utilities\batch_files”

The script in this file moves the directory that contains the slave stack code, and then applies the patch that makes the corrections for the sample program.

If a “Patching file …” message similar to Figure 6-4 does not appear, the patch is not applied. In this case, right-click “apply_patch.bat”, and then select “Run as administrator”.

**Figure 6-4 Patch file execution messages**

Now the preparations for running the EtherCAT project are finally completed.
6.2 Development Environments Install

AC Servo Solution Kit (RZ/T2M, RZ/T2L)

Download e2studio or FSPSC for RZT FSP v1.3.0 from the following web site.

Release v1.3.0 · renesas/rzt-fsp · GitHub

Download “setup_rztfsp_v1_3_0_e2s_v2023_07.exe” for FSP with e2studio installer.
If using IAR, download “setup_rztfsp_v1_3_0_rzsc_v2023_07.exe” for smart configurator installer.

AC Servo Solution Kit (RZ/N2L)

Download e2studio or FSPSC for RZN FSP v1.3.0 from the following web site.

Release v1.3.0 · renesas/rzn-fsp · GitHub

Download “setup_rznfsp_v1_3_0_e2s_v2023_07.exe” for FSP with e2studio installer.
If using IAR, download “setup_rznfsp_v1_3_0_rzsc_v2023_07.exe” for smart configurator installer.

If using IAR, download IAR Embedded Workbench® for Arm Version 9.32.2 from IAR web site.

Products | IAR Systems
6.3 Program Writing Procedure

This chapter is shown how to write a program to serial Flash ROM.

6.3.1 Program writing by IAR EWARM

(1) Build the project for CPU1

Case of the AC Servo Solution Kit (RZ/T2M)

① Open the sample project on IAR EWARM.

```
"r12an0123xxxxxxxx-cn032-ac-servo-solution\Software\Firmware\rzt2m\Project\iccarm\CPU1\CN032_AC_Servo_Solution_CPU1.eww"
```

② Then, generated the binary file for CPU1 “CN032_AC_Servo_Solution_CPU1.bin” in the following folder

```
"r12an0123xxxxxxxx-cn032-ac-servo-solution\Software\Firmware\rzt2m\Project\iccarm\CPU0_serialboot\CPU1_boot_bin"
```

(2) Build the project for CPU0

① Open the following sample project on IAR EWARM.

Case of the AC Servo Solution Kit (RZ/T2M)

```
"r12an0123xxxxxxxx-cn032-ac-servo-solution\Software\Firmware\rzt2m\Project\iccarm\CPU0_serialboot\CN032_AC_Servo_Solution_CPU0_serialboot.eww"
```

Case of the AC Servo Solution Kit (RZ/T2L)

```
"r12an0123xxxxxxxx-cn032-ac-servo-solution\Software\Firmware\rzt2l\Project\iccarm\serial_boot\CN032_AC_Servo_Solution_serialboot.eww"
```

Case of the AC Servo Solution Kit (RZ/N2L)

```
"r12an0123xxxxxxxx-cn032-ac-servo-solution\Software\Firmware\rzn2l\Project\iccarm\serial_boot\CN032_AC_Servo_Solution_serialboot.eww"
```
② Execute build. Select [Project] > [Rebuild All].

③ Select the [Project] > [Download] > [Download active application] to write the program to serial Flash ROM.
6.3.2 Program writing by Renesas e2studio

① Import the sample project. After the program is started, by selecting [File] → [Import] → [Existing Projects into Workspace]. Check the “select root directory” and select the folder below, and then selecting [Finish].

**Case of the AC Servo Solution Kit (RZ/T2M)**

"r12an0123xxXXXX-cn032-ac-servo-solution\Software\Firmware\rzt2m\Project\gcc"

**Case of the AC Servo Solution Kit (RZ/T2L)**

"r12an0123xxXXXX-cn032-ac-servo-solution\Software\Firmware\rzt2l\Project\gcc"

**Case of the AC Servo Solution Kit (RZ/N2L)**

"r12an0123xxXXXX-cn032-ac-servo-solution\Software\Firmware\rzn2l\Project\gcc"

② Build the “CN032_AC_Servo_Solution_CPU0_serialboot” project

In [Project Explorer] view, right click the node of the project to be debugged and select [Build Project].

③ Press the “RESET” switch of the Controller board
While the board and J-LINK are connected, start writing to the flash memory in the following order.

In [Project Explorer] view, right click the node of the CPU0 project to be debugged and select [Debug As] → [Debug Configurations].

[Renasics DBG Hardware Debugging] → [CN032_AC_Servo_Solution_CPU0_serialboot Debug_Flat] item, then press [Debug].
⑤ Press the terminate button to stop the debugging window

Press the reset button of the Controller board, and then running the program written to the flash memory
6.4 Debugging the Sample Project

6.4.1 Debugging the Sample Project in IAR EWARM

In this chapter we will describe, how the sample project for motion control via EtherCAT communication is debugged using the IAR EWARM environment.

(1) Open the following sample project.

Case of the AC Servo Solution Kit (RZ/T2M)

```
"r\12an0123xxXXXX-cn032-ac-servo-solution\Software\Firmware\rzt2m\Project\iccarm\CPU0\CN032_AC_Servo_Solution_CPU0.eww"
```

Case of the AC Servo Solution Kit (RZ/T2L)

```
"r\12an0123xxXXXX-cn032-ac-servo-solution\Software\Firmware\rzt2l\Project\iccarm\ram_exe\CN032_AC_Servo_Solution.eww"
```

Case of the AC Servo Solution Kit (RZ/N2L)

```
"r\12an0123xxXXXX-cn032-ac-servo-solution\Software\Firmware\rzn2l\Project\iccarm\ram_exe\CN032_AC_Servo_Solution.eww"
```

(2) Select the “Rebuild All” item from the “Project” menu to rebuild the project.

(3) Press the “RESET” switch of the Controller board.
(4) While the board and I-jet are connected, click on the “Download and debug” button in the “Project” toolbar.

Case of the AC Servo Solution Kit (RZ/T2M)

After that, the CPU1 project which name is CN032_AC_Servo_Solution_CPU1 will open and debug connection for CPU1 will also be started automatically. The program will break at the first code in “main” in both projects.
Case of the AC Servo Solution Kit (RZ/T2M)

Follow the step (5) to (7) when using the AC Servo Solution Kit (RZ/T2M).

(5) Set the break point at the first code of `hal_entry()`, and Press the “Go” button in the CPU0 project.

Initial settings of cores and registers in SSLB and the program will break at the first code of “hal_entry()”

(6) Press the “Go” button in the CPU1 project. Initial settings of cores and registers in SSLB and the EtherCAT application program will be run.

(7) While the CPU1 is running, Press the “Go” button in the CPU0 project.

Case of the AC Servo Solution Kit (RZ/T2L, RZ/N2L)

Follow the step (5) when using the AC Servo Solution Kit (RZ/T2L, RZ/N2L).

(5) The program will break at the first code in “main”, and then the program is running by pressing the “Go” button.
6.4.2 Debugging the Sample Project in Renesas e2studio

In this chapter we will describe, how the sample project for motion control via EtherCAT communication is debugged using the Renesas e2studio environment.

Before debugging the sample project in e2studio, make sure the dipswitch setting (SW1).

**Set SW1 to other than xSPI0 boot mode** (x1 boot Serial flash) and then, press the reset switch for board reset.

\[\text{e.g.}\]

\[
\begin{array}{c}
\text{xSPI0 boot mode} \\
\text{(x1 boot Serial flash)}
\end{array}
\]

\[
\rightarrow
\]

\[
\begin{array}{c}
\text{xSPI0 boot mode} \\
\text{(x8 boot Serial flash)}
\end{array}
\]

See the section 3.4 Mode Switch in CN032 AC Servo Solution Hardware Manual about the operating mode.

(1) Import the sample project. After the program is started, by selecting [File] → [Import] → [Existing Projects into Workspace]. Check the “select root directory” and select the following folder → [Finish].

**Case of the AC Servo Solution Kit (RZ/T2M)**

“r12an0123xxXXXX-cn032-ac-servo-solution\Software\firmware\rzy2m\Project\gcc”

**Case of the AC Servo Solution Kit (RZ/T2L)**

“r12an0123xxXXXX-cn032-ac-servo-solution\Software\firmware\rzy2l\Project\gcc”

**Case of the AC Servo Solution Kit (RZ/N2L)**

“r12an0123xxXXXX-cn032-ac-servo-solution\Software\firmware\ryn2l\Project\gcc”

(2) Press the “RESET” switch of the CN032 AC Servo Solution board.
Case of the AC Servo Solution Kit (RZ/T2M)

Follow the step (3) to (4) when using the AC Servo Solution Kit (RZ/T2M).

(3) While the board and J-LINK are connected, start debugging in the following order

```
<< CPU0 Project >>
```

In [Project Explorer] view, right click the node of the CPU0 project to be built and select [Debug As] → [Build Project].

In [Project Explorer] view, right click the node of the CPU0 project to be debugged and select [Debug As] → [Debug Configurations].
[Renesas DBG Hardware Debugging] → [CN032_AC_Servo_Solution_CPU0 Debug_Flat] item, then press [Debug].

==================================================================================
This step can be skipped in case of using software package Rev4.00 later.

If a bit [5] value of CPSR register is "1b", change it from "1b" to "0b" by using [Register] view.
For example, the register value of CPSR is changed from "0x20000fa" to "0x20000da" in the following.

Press the "Resume" button.

When debugging CPU0 is started, the program is interrupted with "hal_entry();" in main.c.
<< CPU1 Project >>

In [Project Explorer] view, right click the node of the CPU1 project to be built and select [Debug As] → [Build Project].

![Image of Project Explorer]

In [Project Explorer] view, right click the node of the CPU1 project to be debugged and select [Debug As] → [Debug Configurations].

![Image of Debug Configurations]
[Renesas DBG Hardware Debugging] → [CN032_AC_Servo_Solution_CPU1 Debug_Flat] item, then press [Debug].

The message "The Renesas GDB debug session is already active" is displayed. Select "No".
After that, the CPU1 project which name is CN032_AC_Servo_Solution_CPU1 will open and debug connection for CPU1 will also be started automatically.

Click "system_init() at startup.c:430 0x10010000" in CPU1 project Thread, then switches to the debug screen of CPU1. While CPU1 debugging screen, press the "Resume" button.

![Debug Screen CPU1](image)

The program will break at “hal_entry();” in main.c .

Press the “Resume” button again to execute the CPU1 application program.

(4) Restart debugging the CPU0 project. In [Debug] view, Click "main() at main.c:4 0xea6e" in CPU0 project Thread, then switches to the debug screen of CPU0. Press the “Resume” button to execute the CPU0 application program.

![Debug Screen CPU0](image)

Execute the CPU0 application program.
Case of the AC Servo Solution Kit (RZ/T2L, RZ/N2L)

Follow the step (3) when using the Controller board with RZ/T2L, RZ/N2L.

(3) While the board and J-LINK are connected, start debugging in the following order

In [Project Explorer] view, right click the node of the RZ/T2L, RZ/N2L project to be built and select [Debug As] → [Build Project].

In [Project Explorer] view, right click the node of the RZ/T2L, RZ/N2L project to be debugged and select [Debug As] → [Debug Configurations].
[Renesas DBG Hardware Debugging] → [CN032_AC_Servo_Solution Debug_Flat] item, then press [Debug].

This step can be skipped in case of using software package Rev4.00 later.

If a bit [5] value of CPSR register is “1b”, change it from “1b” to “0b” by using [Register] view.
For example, the register value of CPSR is changed from “0x20000fa” to “0x20000da” in the following.

Press the "Resume" button.
When debugging RZ/T2L, RZ/N2L is started, the program is interrupted with “hal_entry ();” in main.c.

And then, press the “Resume" button again to execute the RZ/T2L, RZ/N2L application program.
6.5 EEPROM Data Update on CN032 AC Servo Solution

If the link between TwinCAT®3 and CN032 AC Servo Solution is established, you can update EEPROM data on the kit from TwinCAT®3. The EEPROM contains identification data like VendorID or Product ID.

The EEPROM is blank when purchasing the CN032 AC Servo Solution; in this case Box1 will be displayed as “FFFFFFFF RFFFFFFFF” as shown in Figure 4-6. Depending on the history of your board you may as well find other data in the EEPROM; therefore we recommend to update the EEPROM in any case as described in the next steps.

To update the EEPROM, double-click on Box 1 (1), select the “EtherCAT” tab (2) and click the “Advanced settings” button (3) as illustrated in Figure 6-5.

![Figure 6-5 Updating the EEPROM](image)

Then expand the list of advanced settings accordingly, so that you can select the “Hex Editor” (1). In the “Hex Editor” view click “Download from List” (2) (See Figure 6-6). This list contains numerous devices for which the required EEPROM content is ready to use.

At the bottom of the list you find as well some Renesas devices Select “CN032 AC Servo Solution CiA402” and click ok (see Figure 6-7).
The data is then downloaded to the EEPROM; at the end of the download process the data is automatically verified. With “Upload” (see Figure 6-6) you can check, whether the EEPROM programming was successful.
Now repeat the device scan shown in Figure 6-8. TwinCAT®3 will now scan once more for devices and it will find the CN032 AC Servo Solution board with the updated description data in EEPROM. The subsequent “Check Configuration” dialog shows you the list of found items. Copy the “CN032 AC Servo Solution CiA402” device with “Copy All” into the list of “Configured Items” (1). When the representation of the devices in the list changes to green, leave the dialog with “OK” (2).

Figure 6-8 Copy the found device into the configuration
Website and Support

Renesas Electronics website

http://www.renesas.com/

Inquiries

http://www.renesas.com/inquiry
## Revision History

<table>
<thead>
<tr>
<th>Rev.</th>
<th>Date</th>
<th>Description</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>Jun. 7, 2022</td>
<td>First Edition issued</td>
<td>First Edition issued</td>
</tr>
<tr>
<td>2.00</td>
<td>Aug. 9, 2022</td>
<td>1, 3, 4, 13, 34, 37, 38, 43-53, 40, 42, 23-32</td>
<td>Description for AC Servo Solution (RZ/N2L) added.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Description for program writing by e2studio with RZ/T2M added.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Description for CiA402 Drive Profile added.</td>
</tr>
<tr>
<td>3.00</td>
<td>Sep. 30, 2022</td>
<td>1, 4, 20, 24-28, 29, 13, 40, 42, 43, 45, 48, 51</td>
<td>Caution when handling the solution board added</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RZ/N2L FSP is updated to V1.00.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4.3.1 operation mode setting added.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4.3.3 Cyclic Synchronous Position Mode and 4.3.3 Cyclic Synchronous Velocity Mode added</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Type fixed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>File path is changed for firmware Rev3.00.</td>
</tr>
<tr>
<td>4.00</td>
<td>Feb. 28, 2023</td>
<td>1, 4, 14, 41, 43, 45, 47, 50, 52, 53, 59, 60, 6, 7, 41, 44, 55, 60</td>
<td>Description for AC Servo Solution (RZ/T2L) added.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Description of RS485 communication added.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Description of SSC tool version added.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Description of development environment install added.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Description of CPSR register control changed.</td>
</tr>
<tr>
<td>5.00</td>
<td>Dec. 15, 2023</td>
<td>4, 5, 44</td>
<td>Operating Environment table is updated.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Precaution is added.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Development Environment install is updated.</td>
</tr>
</tbody>
</table>
General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)
   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity.
   Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.

2. Processing at power-on
   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.

3. Input of signal during power-off state
   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.

4. Handling of unused pins
   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.

5. Clock signals
   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.

6. Voltage application waveform at input pin
   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between $V_{IL}$ (Max.) and $V_{IH}$ (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between $V_{IL}$ (Max.) and $V_{IH}$ (Min.).

7. Prohibition of access to reserved addresses
   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.

8. Differences between products
   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.
Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product. Renesas Electronics hereby expressly disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.

2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.

3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.

4. You shall be responsible for determining which licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.

5. You may not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.

6. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.

   - “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   - “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; underwater repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.

7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.

8. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.

9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as software design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.

10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.

11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.

12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.

13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.

14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1  October 2020)

Corporate Headquarters
TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks
Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information
For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/