

GigaDevice Semiconductor Inc.

**Reference Manual for EtherCAT Protocol
Stack Code Porting**

**Application Note
AN246**

Revision 1.2

(Sep. 2025)

Table of Contents

Table of Contents	2
List of Figures	3
List of Tables	4
1. Preface.....	5
2. Generating Protocol Stack Code Using SSC TOOL.....	6
3. Embedding Protocol Stack Code into the Project	11
4. EEPROM Update Method	11
5. Methods for Adding SDO & PDO in COE	14
5.1 Adding SDO	16
5.2 Adding PDO	20
6. Revision history.....	27

List of Figures

Figure 2-1. SSC Tool Options Page	6
Figure 2-2. ConfigurationsPage	7
Figure 2-3. Project Creation Page	7
Figure 2-4. Select GD Project Page	8
Figure 2-5. Save Path Page	8
Figure 2-6. EXCEL File Select Page	9
Figure 2-7. Import Page	9
Figure 2-8. Protocol Stack Code Generation Page	10
Figure 2-9. Save Path Page	10
Figure 3-1. Source File Directory Page	11
Figure 3-2. Project Compilation Page	11

List of Tables

Table 4-1. Revision history	27
-----------------------------------	----

1. Preface

This document is specifically designed for users of GD32 MCU products who need a brief explanation of the steps and methods for porting EtherCAT protocol stack code.

This application note is divided into two parts: the first part describes the process of generating protocol stack code using the SSC TOOL, and the second part explains the process of embedding and compiling the protocol stack code within a project.

Before using this application, users are required to be members of the ETG organization in order to obtain access to the SSC TOOL for generating protocol stack code.

2. Generating Protocol Stack Code Using SSC TOOL

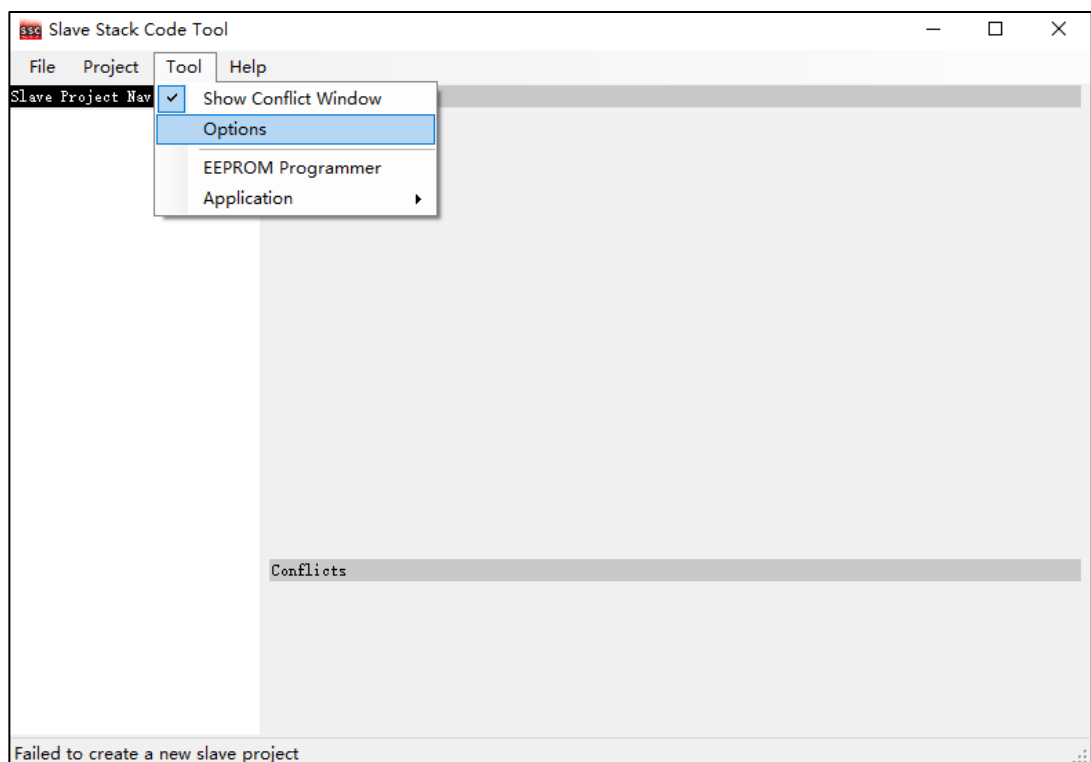
Users must first complete the download of the SSC TOOL. Refer to the ETG official website for download instructions:

https://www.ethercat.org/en/downloads/downloads_01DCC32A10294F2EA866F7E46FB0285F.htm.

Refer to the following steps to generate protocol stack code:

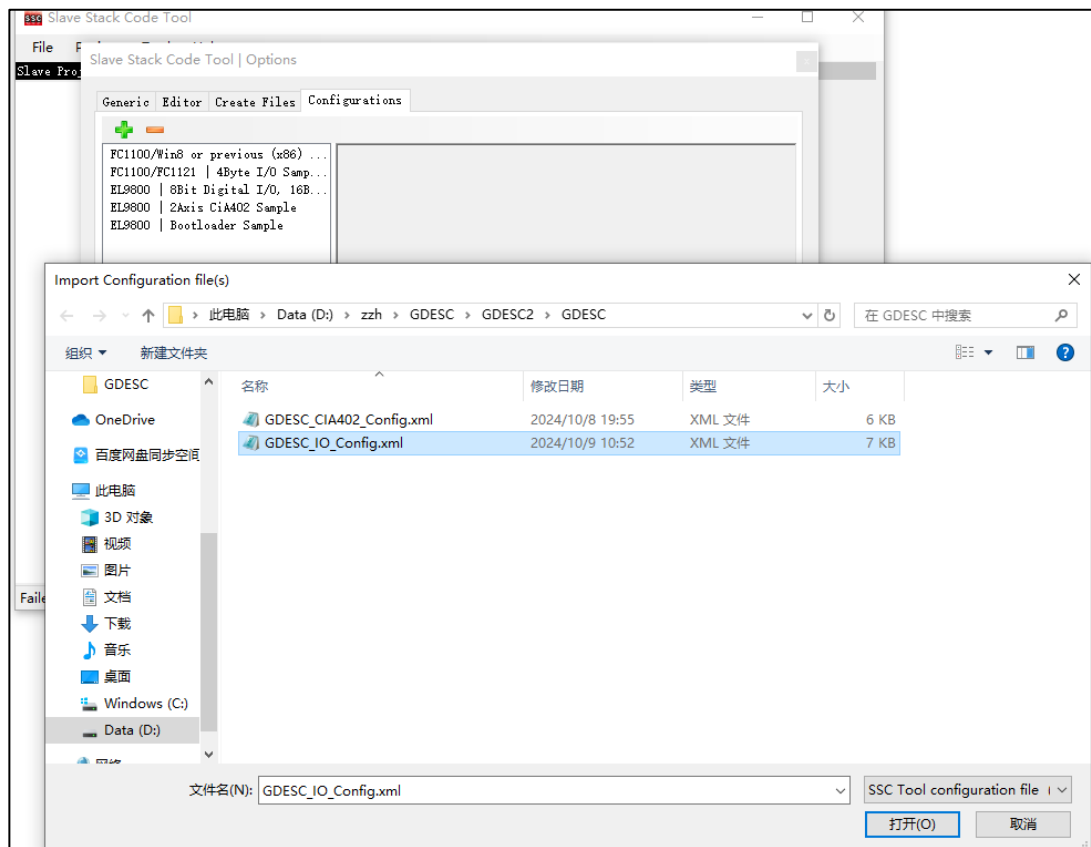
1. Open the SSC Tool and select ****Tool->Options****.

Figure 2-1. SSC Tool Options Page



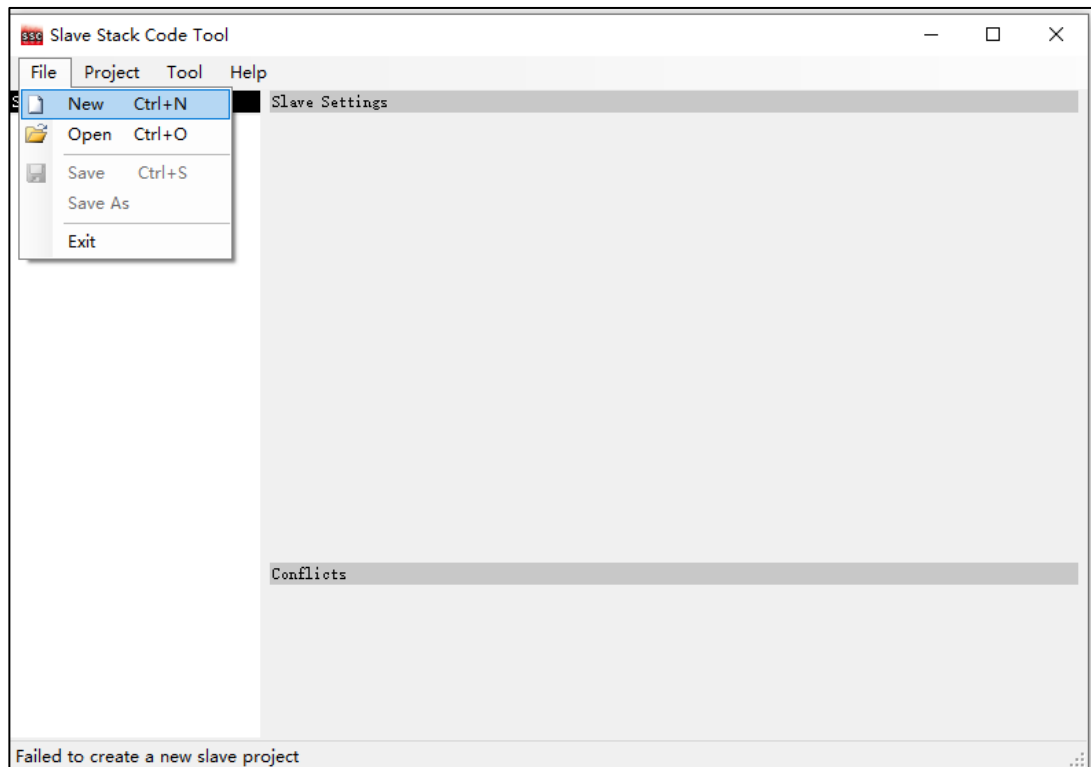
2. Click on Configurations and select to add the configuration file GDESC_XXX_Config.xml.

Figure 2-2. ConfigurationsPage



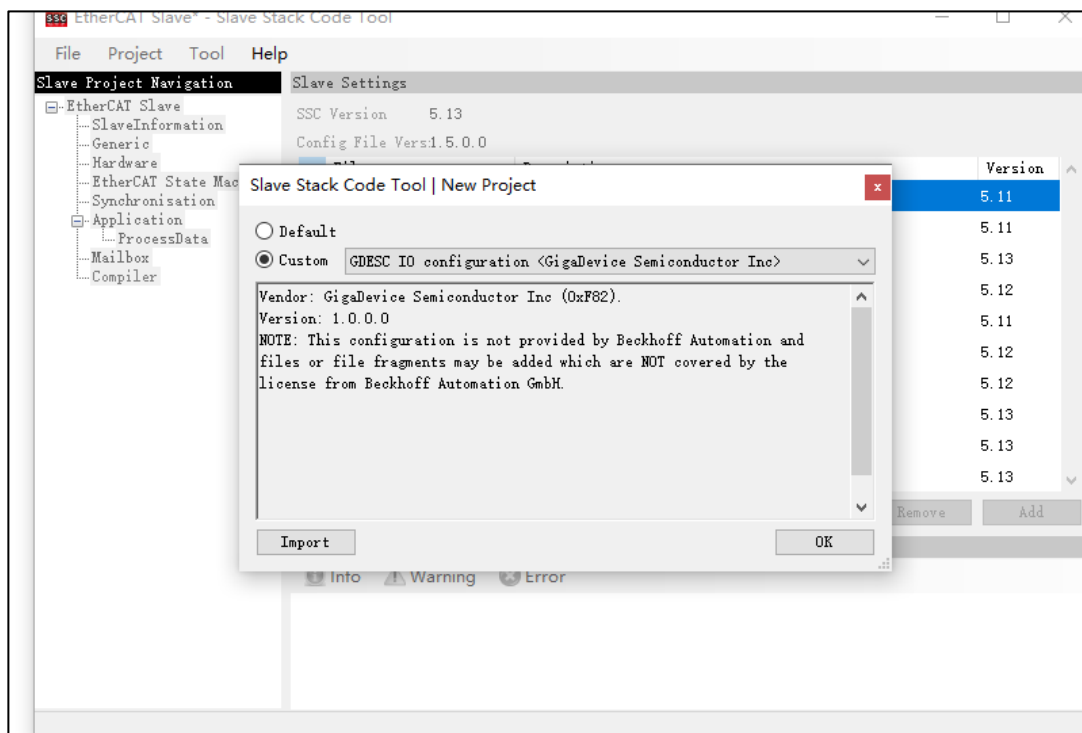
3. Create a new project in the SSC Tool by clicking ****File->New****.

Figure 2-3. Project Creation Page



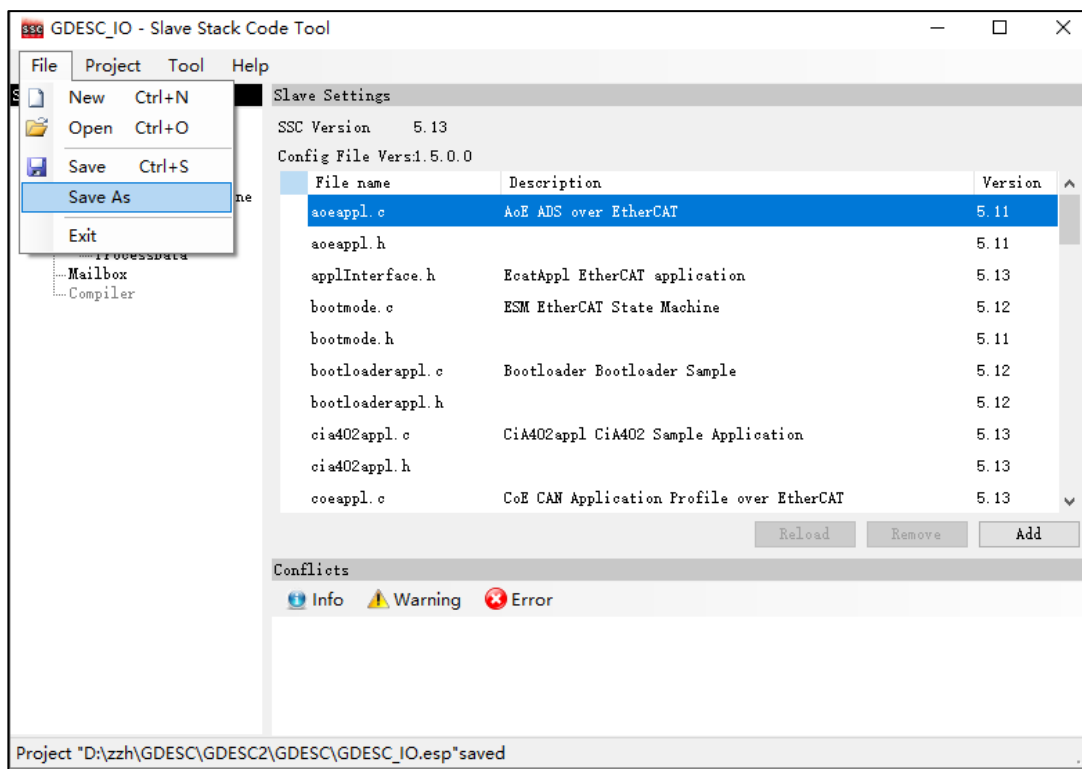
- Click on the **Custom** option, and select **GDESC IO configuration <GigaDevice Semiconductor Inc >** from the dropdown list.

Figure 2-4. Select GD Project Page



- Save the project file and specify the save path.

Figure 2-5. Save Path Page



6. Import the application EXCEL file, If the imported file is GDESC_CIA402_Config_Demo.xml, this step can be skipped.

Figure 2-6. EXCEL File Select Page

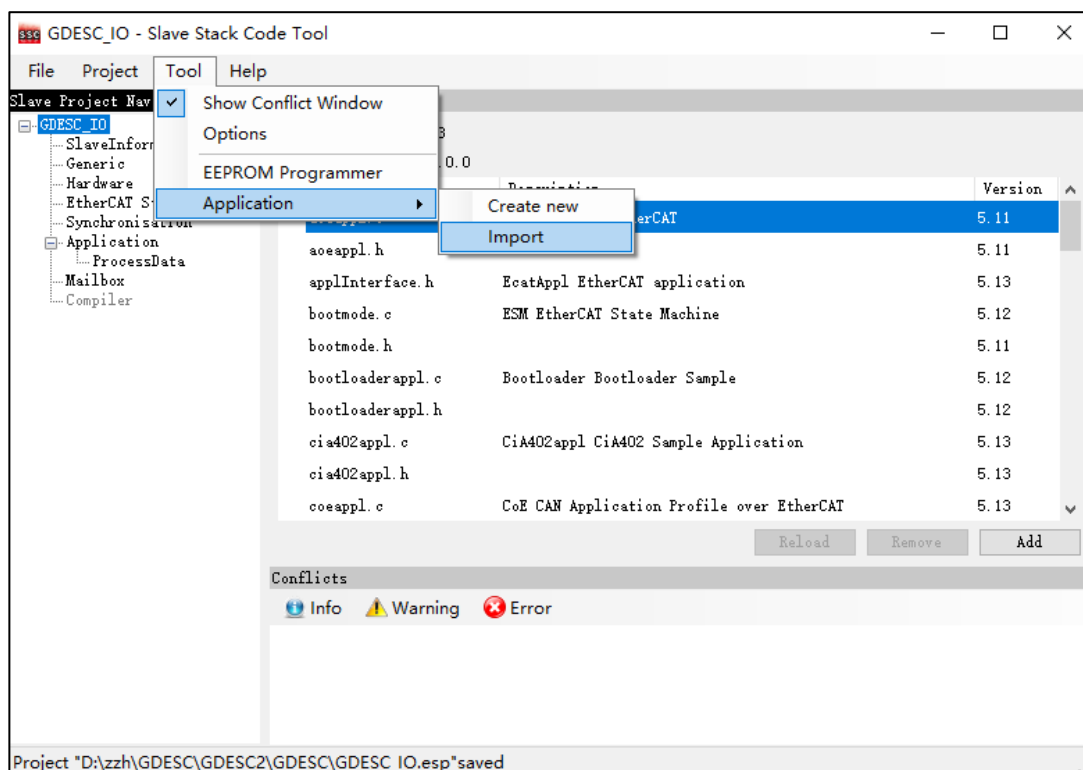
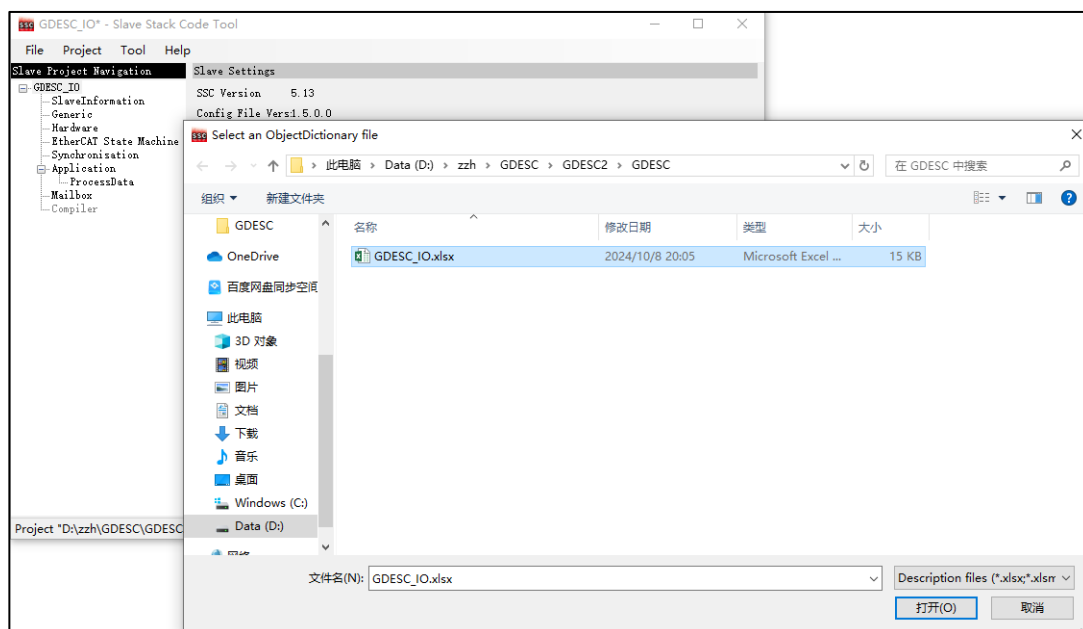
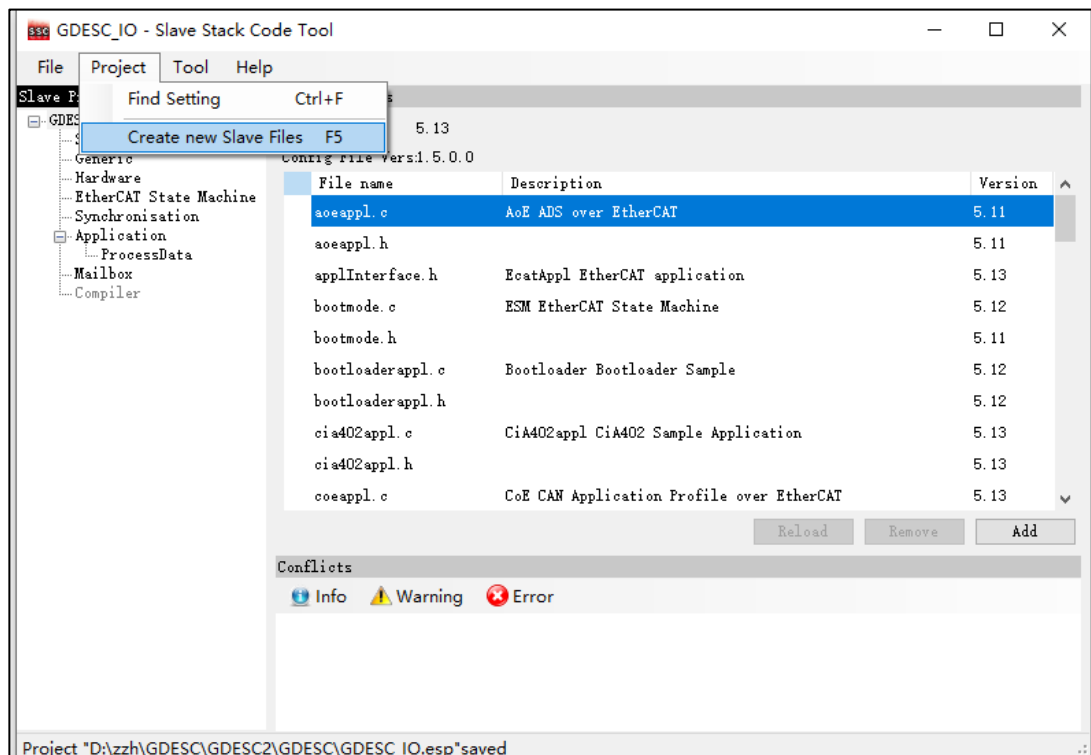


Figure 2-7. Import Page



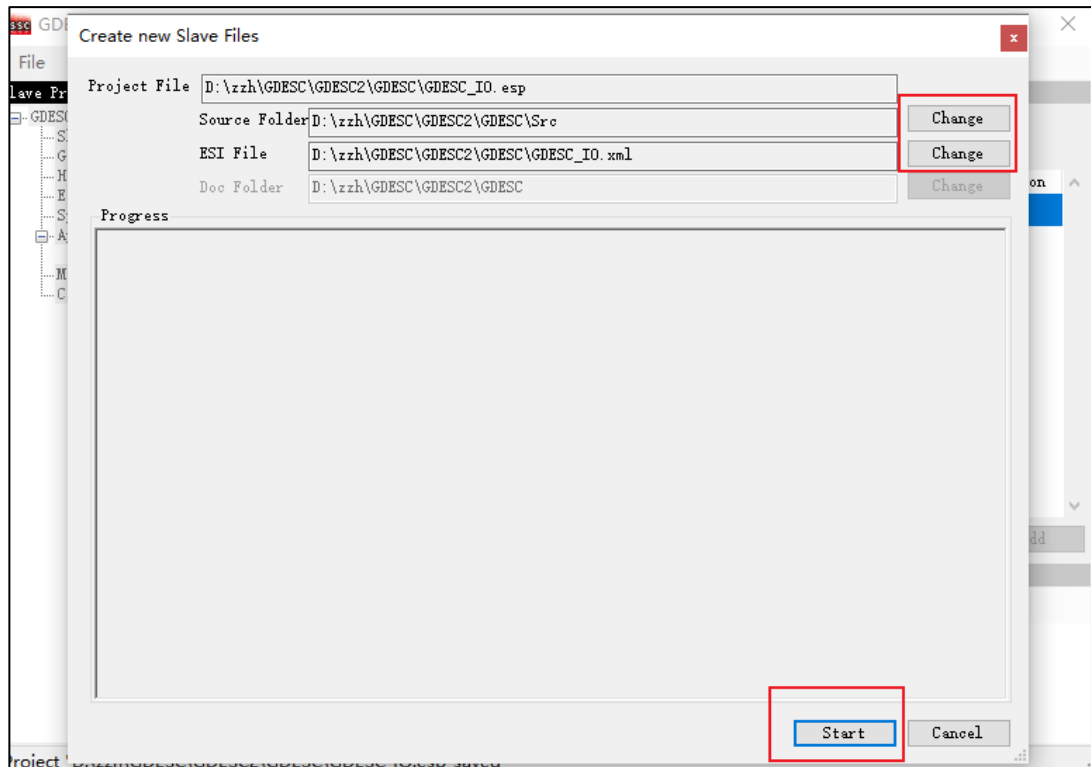
7. Protocol Stack Code Generation Page

Figure 2-8. Protocol Stack Code Generation Page



8. Select the path to save the protocol stack code and configuration XML file.

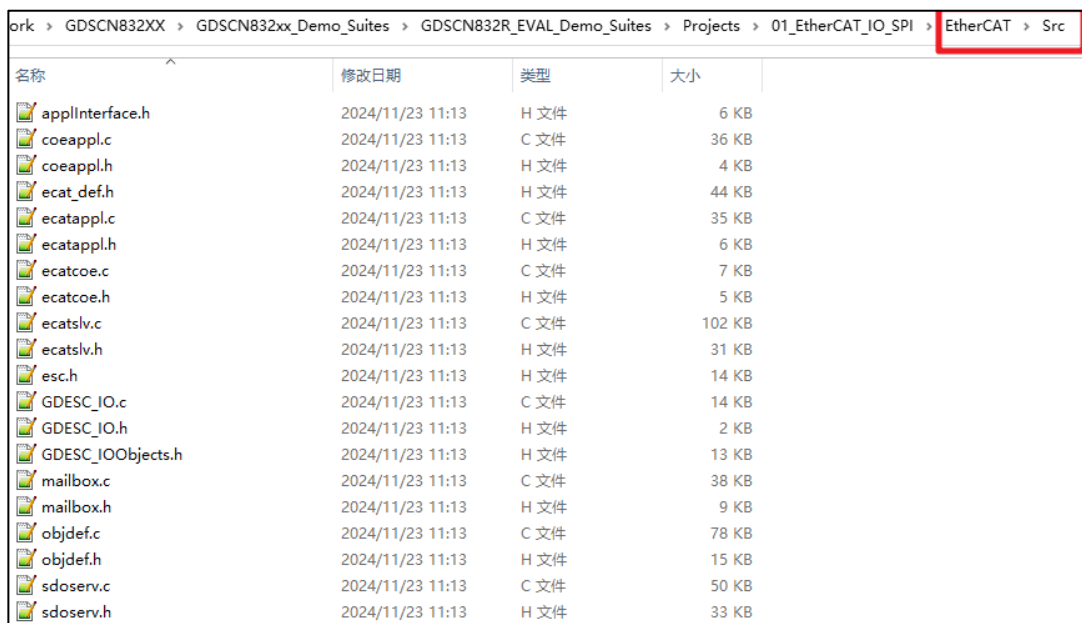
Figure 2-9. Save Path Page



3. Embedding Protocol Stack Code into the Project

1. Copy the generated protocol stack code to the EtherCAT/Src directory under the project folder.

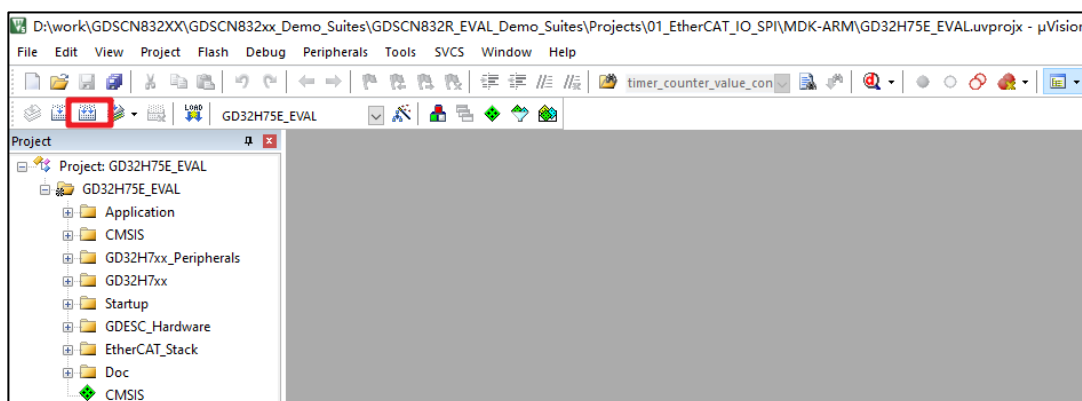
Figure 3-1. Source File Directory Page



名称	修改日期	类型	大小
applInterface.h	2024/11/23 11:13	H 文件	6 KB
coeappl.c	2024/11/23 11:13	C 文件	36 KB
coeappl.h	2024/11/23 11:13	H 文件	4 KB
ecat_def.h	2024/11/23 11:13	H 文件	44 KB
ecatappl.c	2024/11/23 11:13	C 文件	35 KB
ecatappl.h	2024/11/23 11:13	H 文件	6 KB
ecatcoe.c	2024/11/23 11:13	C 文件	7 KB
ecatcoe.h	2024/11/23 11:13	H 文件	5 KB
ecatslv.c	2024/11/23 11:13	C 文件	102 KB
ecatslv.h	2024/11/23 11:13	H 文件	31 KB
esc.h	2024/11/23 11:13	H 文件	14 KB
GDESC_IO.c	2024/11/23 11:13	C 文件	14 KB
GDESC_IO.h	2024/11/23 11:13	H 文件	2 KB
GDESC_IOObjects.h	2024/11/23 11:13	H 文件	13 KB
mailbox.c	2024/11/23 11:13	C 文件	38 KB
mailbox.h	2024/11/23 11:13	H 文件	9 KB
objdef.c	2024/11/23 11:13	C 文件	78 KB
objdef.h	2024/11/23 11:13	H 文件	15 KB
sdoserv.c	2024/11/23 11:13	C 文件	50 KB
sdoserv.h	2024/11/23 11:13	H 文件	33 KB

2. Directly compile the project file.

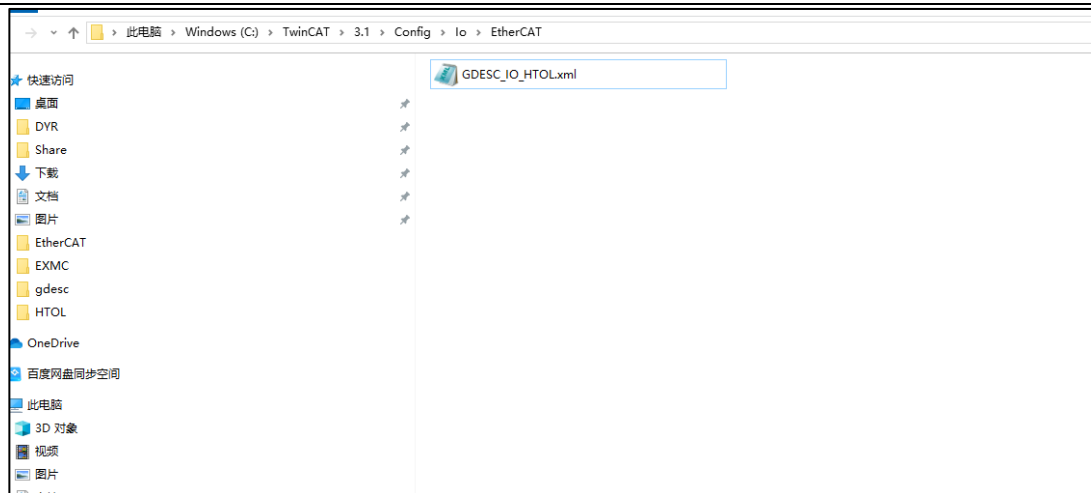
Figure 3-2. Project Compilation Page



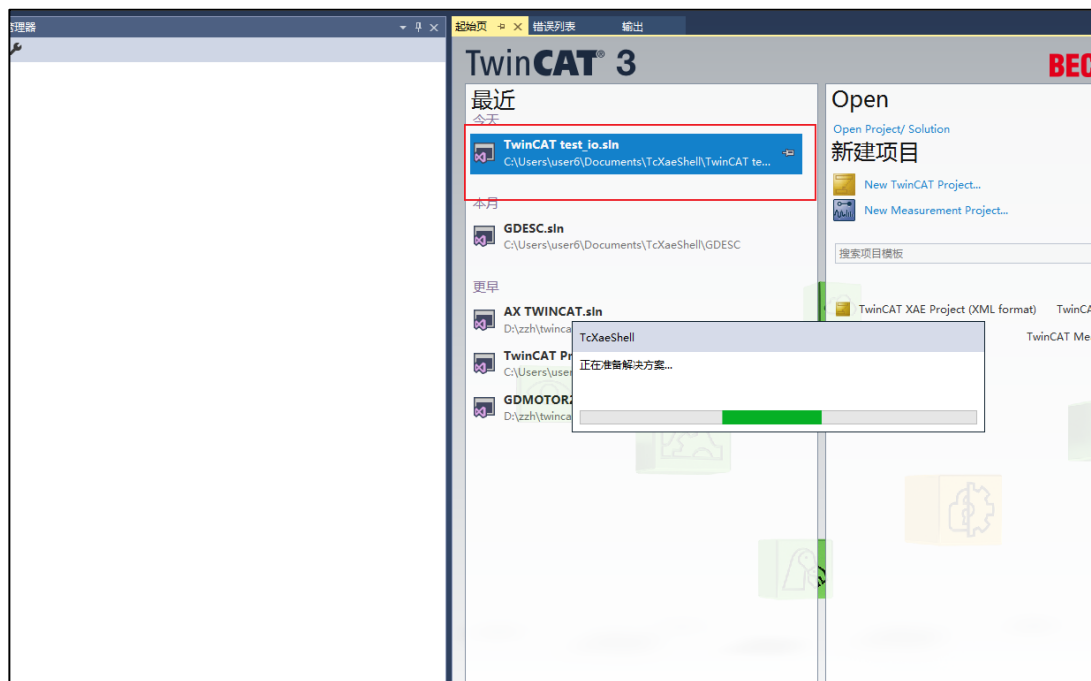
4. EEPROM Update Method

The following steps illustrate how to update the EEPROM using the Twincat master software as an example:

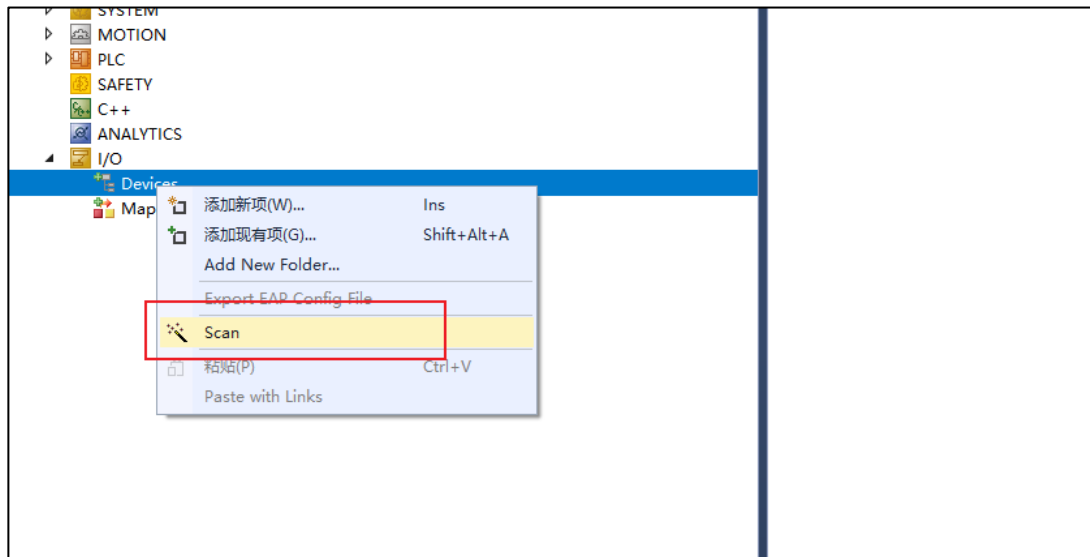
1. Copy the XML file to be updated into the designated directory of the Twincat software, such as XXX\TwinCAT\3.1\Config\Io\EtherCAT directory.



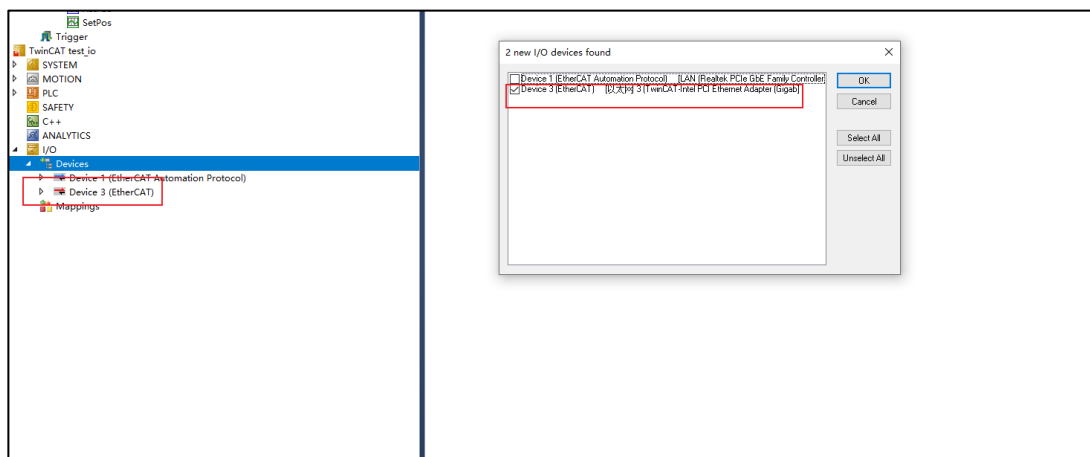
2. Open the TwinCAT software and click to open any project file.



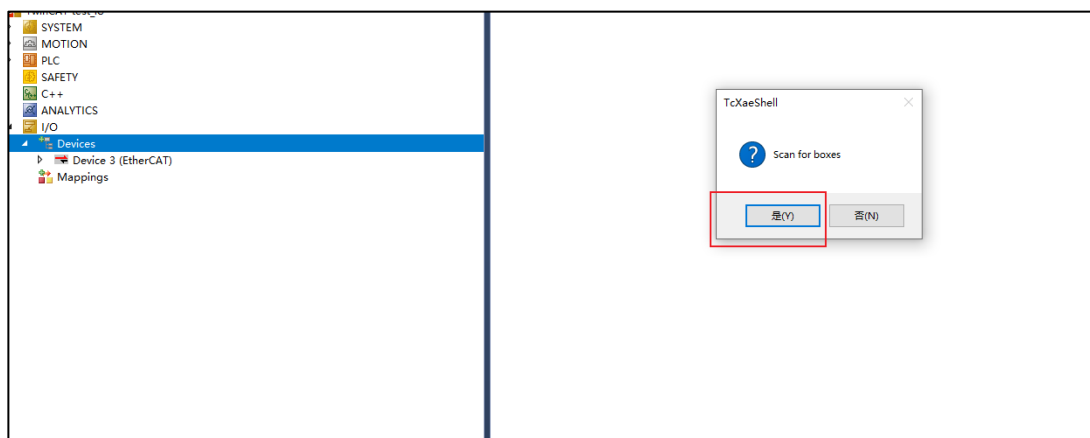
3. After opening the project, click on Device, right-click and select Scan.

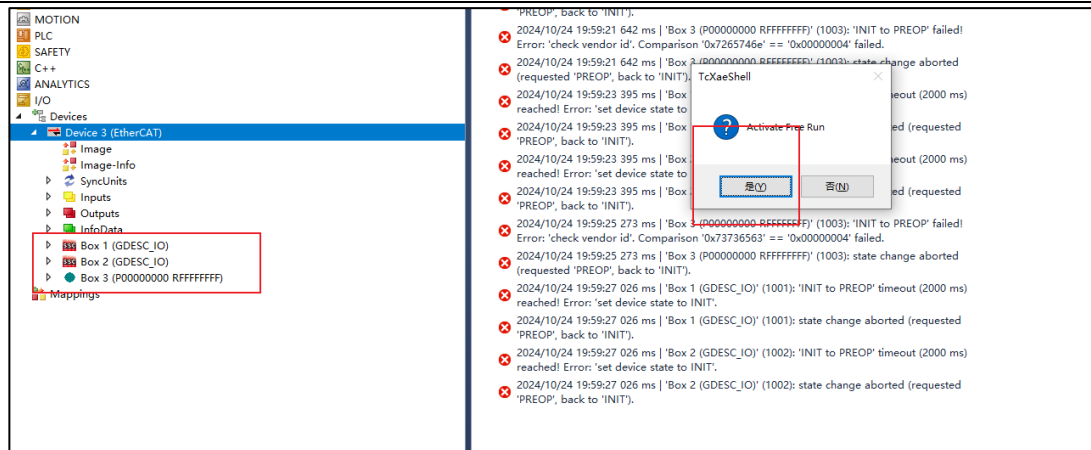


4. Select the corresponding network card connection. After successfully identifying the slave device, check the corresponding network card and click OK.

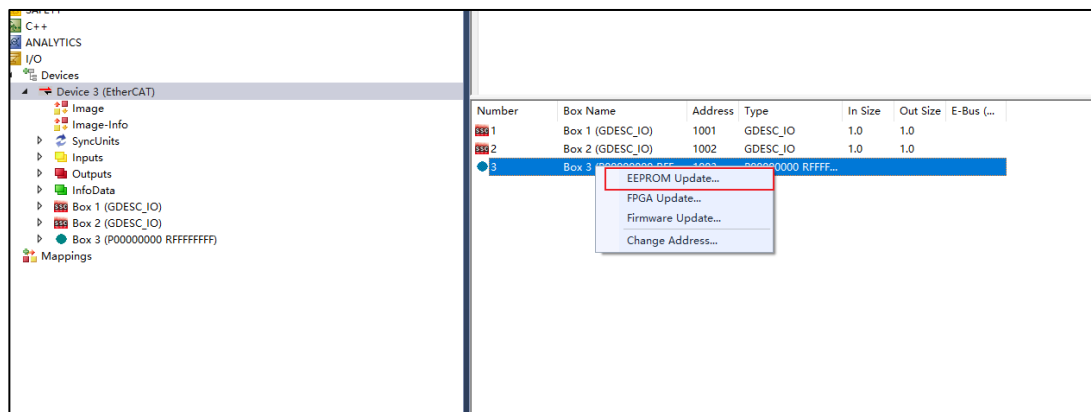


5. Click "Yes" consecutively until the scanning is complete.





- After the scanning is complete, double-click on Device. The corresponding Box will appear on the right side. Right-click on the Box and select EEPROM for updating.



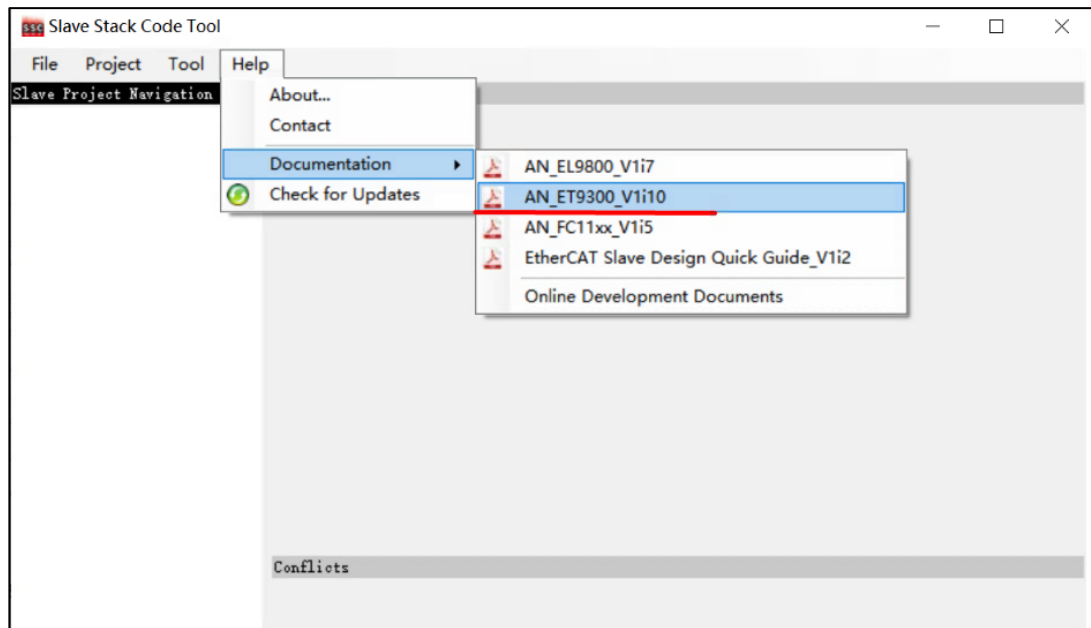
5. Methods for Adding SDO & PDO in COE

Typically, we can choose two methods to add CoE object dictionary content:

One is to directly create a new object dictionary using the OD tool in the SSC Tool.

The other is to add new object dictionary content based on existing routines by modifying the slave device XML file and the slave device program.

《AN_ET9300》 is the help documentation provided after installing the SSC Tool. This document introduces the usage of the SSC Tool, which not only aids in learning how to use the SSC Tool but also helps in understanding the processing flow of the slave protocol stack code. It is very helpful in guiding how to add CoE dictionary content.



The first method is suitable for applications where the object dictionary is not extensively used and the mapping is relatively simple. To add an object dictionary using the SSC TOOL, you need to modify the Excel file (GDESC_IO.xlsx) provided in the GD demo routine according to the actual application. Then, re-import the file into the SSC TOOL to generate new protocol stack code (refer to the steps in section 2.6 of this document). This will allow you to add the corresponding object dictionary parameters. When adding the corresponding object dictionary, pay attention to the arrangement order of the high byte and low byte in parameter naming to ensure the naming complies with the standards.

Refer to sections 6.4 "Create an own Application" and 13 "OD Tool" in 《AN_ET9300》 to edit the Excel file and add the required object dictionary content.

Device Profile:

5001

Modular Device Profile

Model Profile:

0

IndexIncrement:

0

PdoIndexIncrement:

0

Usage Notes:

- The PDO mapping object and SyncManager assignment object doesn't need to be defined. In that case they are created automatically.

- The following objects are fixed included in the SSC and shall not be defined in the file : 0x1000, 0x1001, 0x1008, 0x1009, 0x100a, 0x1010, 0x1011, 0x1018, 0x1019, 0x101f, 0x101f3, 0x1c00, 0x1c32, 0x1c33

- Entries less or equal one SBIT shall not overlap byte borders.

- Entries greater SBIT shall always start at an exact word border

The object dictionary defined here shall be used complementary with ETC.5001 and ETC.1000

Index	ObjectCode	SI	DataType	Name	Default V	M/O/C	B/S	Access	rx/tx	CoefRead	CoefWrite	Description
//0x0xxx Input Data of the Module (0x0000 - 0x0FFF)												
0x0000	RECORD		1 BOOL	SWITCH1	0	M		ro	tx			
			2 BOOL	SWITCH2	0	M		ro	tx			
//0x7xxx Output Data of the Module (0x7000 - 0x7FFF)												
0x7010	RECORD		1 BOOL	LED1	0			rw	rx			
			2 BYTE	LED2	0			rw	rx			
//0xbxxx Configuration Data of the Module (0xb000 - 0xbFFF)												
//0xdxxx Information Data of the Module (0xd000 - 0xdFFF)												
//0xfxxx Diagnosis Data of the Module (0xf000 - 0xfFFF)												
//0xffff Device Objects (0xffff00 - 0xffffFF)												
0xffff00	RECORD			Modular Device Profile		M						
			1 UINT16	Index distance	0x10	M		ro				
			2 UINT16	Maximum number of modules		M		ro				

PDO

SDO

The second method, manually adding SDO & PDO, is suitable for more complex application projects with extensive parameter mapping and mature applications. In such cases, manual addition can be performed. When using the CIA402 routine, the configured xxx config.xml will generate the corresponding code project. At this point, it is not supported to create a new application by importing an Excel file. In other words, if you want to add new CoE objects to

the CIA402 routine, you need to manually modify the XML and slave code based on the existing foundation, as it cannot be done by editing and importing an Excel file.

When manually modifying, pay attention to the following: declare the relevant object dictionary definitions and add them to the object dictionary; ensure consistency between code modifications and XML modifications; if it involves PDO objects, you also need to modify the definitions in the corresponding RxPDO and TxPDO and add objects to the PDO mapping.

5.1 Adding SDO

Taking the 23_EtherCAT example of the GD32H75EY_EVAL board as an example, which uses the CIA402 project, after completing the protocol stack porting and compilation based on the above steps, a new set of parameters, Target Torque Value (0x6071) and Current Actual Value (0x6078), needs to be added for communication purposes. The communication is conducted using the SDO method.

1. Modify the XML file, Add the corresponding object dictionary content in GDESC_CIA402.xml.

```

<Object>
  <Index>#x6071</Index>
  <Name>Target Torque Value</Name>
  <Type>INT</Type>
  <BitSize>16</BitSize>
  <Info>
    <SubItem>
      <Name>Target Torque Value</Name>
      <Info>
        <DefaultData>0</DefaultData>
      </Info>
    </SubItem>
  </Info>
  <Flags>
    <Access>rw</Access>
    <Category>o</Category>
    <PdoMapping>R</PdoMapping>
  </Flags>
</Object>

<Object>
  <Index>#x6078</Index>
  <Name>Current Actual Value</Name>
  <Type>INT</Type>
  <BitSize>16</BitSize>
  <Info>
    <SubItem>
      <Name>Current Actual Value</Name>
      <Info>
        <DefaultData>0</DefaultData>
      </Info>
    </SubItem>
  </Info>
  <Flags>
    <Access>ro</Access>
    <Category>o</Category>
    <PdoMapping>T</PdoMapping>
  </Flags>
</Object>

```

2. Modify the two files, cia402appl.h and cia402appl.c, in the protocol stack code. First, modify cia402appl.c to add the code for initializing dictionary data values.


```

case 0x606C:
    pDiEntry->pVarPtr = &LocalAxes[AxisCnt].Objects.objVelocityActualValue;
    break;
case 0x6071:
    pDiEntry->pVarPtr = &LocalAxes[AxisCnt].Objects.objTargetTorqueValue;
    break;
case 0x6077:
    pDiEntry->pVarPtr = &LocalAxes[AxisCnt].Objects.objTorqueActualValue;
    break;
case 0x6078:
    pDiEntry->pVarPtr = &LocalAxes[AxisCnt].Objects.objCurrentActualValue;
    break;
case 0x607A:
    pDiEntry->pVarPtr = &LocalAxes[AxisCnt].Objects.objTargetPosition;
    break;

```

Cia402_Init函数

Then modify the cia402appl.h file to add specific dictionary declarations and definitions in the structure.

```

typedef struct OBJ_STRUCT_PACKED_START
{
    TOB31600 sRxPDOMap0; /**< \brief csv/csp RxPDO (0x1600)*/
    TOB31601 sRxPDOMap1; /**< \brief csp RxPDO (0x1601)*/
    TOB31602 sRxPDOMap2; /**< \brief csv RxPDO (0x1602)*/
    TOB31A00 sTxPDOMap0; /**< \brief csv/csp TxPDO (0x1A00)*/
    TOB31A01 sTxPDOMap1; /**< \brief csp TxPDO (0x1A01)*/
    TOB31A02 sTxPDOMap2; /**< \brief csv TxPDO (0x1A02)*/

    UINT16 objErrorCode; /**< \brief Error Code (0x603F)*/
    UINT16 objControlWord; /**< \brief Control Word (0x6040)*/
    UINT16 objStatusWord; /**< \brief Status Word (0x6041)*/
    INT16 objQuickStopOptionCode; /**< \brief Quick Stop Option Code (0x605A)*/
    INT16 objShutdownOptionCode; /**< \brief Shutdown Option Code (0x605B)*/
    INT16 objDisableOperationOptionCode; /**< \brief Disable Operation Option Code (0x605C)*/
    INT16 objHaltOptionCode; /**< \brief Halt Option Code (0x605D)*/
    INT16 objFaultReactionCode; /**< \brief Fault Reaction Code (0x605E)*/
    INT16 objModesOfOperation; /**< \brief Modes of Operation (0x6060)*/
    INT16 objModesOfOperationDisplay; /**< \brief Mode of Operation Display (0x6061)*/
    INT32 objPositionDemandValue; /**< \brief Position Demand Value (0x6062)*/
    INT32 objPositionActualInternalValue; /**< \brief Position Actual Internal Value (0x6063)*/
    INT32 objPositionActualValue; /**< \brief Position Actual Value (0x6064)*/
    UINT32 objFollowingErrorWindow; /**< \brief Following Error Window (0x6065)*/
    UINT16 objFollowingErrorTimeOut; /**< \brief Following Error Time Out (0x6066)*/
    UINT32 objPositionWindow; /**< \brief Position Window (0x6067)*/
    UINT16 objPositionWindowTime; /**< \brief Position Window Time (0x6068)*/
    INT16 objVelocityActualValue; /**< \brief Velocity Actual Value (0x606C)*/
    INT16 objTargetTorqueValue; /**< \brief Target Torque Value (0x6071)*/
    INT16 objTorqueActualValue; /**< \brief Torque Actual Value (0x6077)*/
    INT16 objCurrentActualValue; /**< \brief Current Actual Value (0x6078)*/
    INT32 objTargetPosition; /**< \brief Target Position (0x607A)*/
    INT32 objHomeOffset; /**< \brief Home Offset (0x607C)*/
    TOB31607D objSoftwarePositionLimit; /**< \brief Software Position limit (0x607D)*/
    UINT8 objInvertDir; /**< \brief Invert Dir (0x607E)*/
    UINT32 objMaxProfileVelocity; /**< \brief Max Profile Velocity (0x607F)*/
    UINT32 objMaxMotorSpeed; /**< \brief Max Motor Speed (0x6080)*/
    UINT32 objProfileVelocity; /**< \brief Profile Velocity (0x6081)*/
    UINT32 objProfileAcceleration; /**< \brief Profile Acceleration (0x6083)*/
    UINT32 objProfileDeceleration; /**< \brief Profile Deceleration (0x6084)*/
    UINT32 objQuickStopDeclaration; /**< \brief Quick Stop Declaration (0x6085)*/
    INT8 objHomingMethod; /**< \brief Homing Method (0x6098)*/
    TOB36099 objHomingSpeeds; /**< \brief Homing Speeds (0x6099)*/
    UINT32 objHomingAcceleration; /**< \brief Homing Acceleration (0x609A)*/
    TOB360C2 objInterpolationTimePeriod; /**< \brief Interpolation Time Period (0x60C2)*/
    INT32 objPosDemand; /**< \brief Pos Demand (0x60FC)*/
    INT32 objTargetVelocity; /**< \brief Target Velocity (0x60FF)*/
    UINT32 objSupportedDriveModes; /**< \brief Supported Drive Modes (0x6502)*/
}

```

```

/** \brief Object 0x606C (Velocity Actual Value) entry description*/
OBJCONST TS001INFOENTRYDESC OBJMEM sEntryDesc0x606C = {DEFTYPE_INTEGER32, 0x20, (ACCESS_READ | OBJACCESS_TXPDO_MAPPING)};

/** \brief Object 0x606C (Velocity Actual Value) object name*/
OBJCONST UCHAR OBJMEM aName0x606C[] = "Velocity Actual Value";

/** \brief Object 0x6071 (Target Torque Value) entry description*/
OBJCONST TS001INFOENTRYDESC OBJMEM sEntryDesc0x6071 = {DEFTYPE_INTEGER16, 0x10, (ACCESS_READWRITE | OBJACCESS_RXPDO_MAPPING)};

/** \brief Object 0x6071 (Target Torque Value) object name*/
OBJCONST UCHAR OBJMEM aName0x6071[] = "Target Torque Value";

/** \brief Object 0x6077 (Torque Actual Value) entry description*/
OBJCONST TS001INFOENTRYDESC OBJMEM sEntryDesc0x6077 = {DEFTYPE_INTEGER16, 0x10, (ACCESS_READ | OBJACCESS_TXPDO_MAPPING)};

/** \brief Object 0x6077 (Torque Actual Value) object name*/
OBJCONST UCHAR OBJMEM aName0x6077[] = "Torque Actual Value";

/** \brief Object 0x6078 (Current actual value) entry description*/
OBJCONST TS001INFOENTRYDESC OBJMEM sEntryDesc0x6078 = {DEFTYPE_INTEGER16, 0x10, (ACCESS_READ | OBJACCESS_TXPDO_MAPPING)};

/** \brief Object 0x6078 (Current actual value) object name*/
OBJCONST UCHAR OBJMEM aName0x6078[] = "Current Actual Value";

/** \brief Object 0x607A (Target Position) entry description*/
OBJCONST TS001INFOENTRYDESC OBJMEM sEntryDesc0x607A = {DEFTYPE_INTEGER32, 0x20, (ACCESS_READWRITE | OBJACCESS_RXPDO_MAPPING)};

/** \brief Object 0x607A (Target Position) object name*/
OBJCONST UCHAR OBJMEM aName0x607A[] = "Target Position";

```

Set default values for parameters

```

/*ifdef _C1A02_
* {
  {5, {0x0400010,0x0700020,0x07F0020,0x0600000,0x0000000}}, /* T0B1600*/
  {3, {0x0400010,0x0700020,0x0000000}}, /*T0B1601*/
  {3, {0x0400010,0x07F0020,0x0000000}}, /*T0B1602*/
  {5, {0x0410010,0x0600020,0x0600020,0x0610000,0x0000000}}, /*T0B1A00*/
  {3, {0x0410010,0x0600020,0x0000000}}, /*T0B1A01*/
  {3, {0x0410010,0x0600020,0x0000000}}, /*T0B1A02*/
  0x0, /*(UINT16) ErrorCode 0x0001*/
  0x0, /*(UINT16) ControlWord 0x0000*/
  0x0, /*(UINT16) StatusWord 0x0001*/
  0x2, /*(INT16) QuickStopOptionCode 0x000A*/
  DISABLE_DRIVE, /*(INT16) ShutdownOptionCode 0x0050*/
  SLOW_DOWN_RAMP, /*(INT16) DisableOperationCode 0x005C*/
  0x1, /*(INT16) HalStopOptionCode 0x005D*/
  QUICKSTOP_RAMP, /*(INT16) FaultReactionCode 0x005E*/
  0x0, /*(INT16) ModeOfOperation 0x0060*/
  0x0, /*(INT16) Mode Of Operation Display 0x0061*/
  0x0, /*(INT32) Position Demand Value 0x0062*/
  0x0, /*(INT32) Position Actual Internal Value 0x0063*/
  0x0, /*(INT32) Position Actual Value 0x0064*/
  0x0, /*(UINT32) Following Error Window 0x0065*/
  0x0, /*(UINT16) Following Error Time Out 0x0066*/
  0x0, /*(UINT32) Position Window 0x0067*/
  0x0, /*(UINT16) Position Window Time 0x0068*/
  0x0, /*(INT32) Velocity Actual Value 0x0069*/
  0x0, /*(INT32) Torque Actual Value 0x0071*/
  0x0, /*(INT16) Current Actual Value 0x0078*/
  0x0, /*(INT16) Target Position 0x007D*/
  0x0, /*(INT32) Home Offset 0x007C*/
  {2,0x0000000,0x7755000}, /*T0B1007D Software Position Limit (minLimit: -2000000000 / maxLimit: 2000000000)*/
  0x0, /*(UINT8) Invert Dir 0x007E*/
  0x0, /*(UINT32) Max Profile Velocity 0x007F*/
  0x0, /*(UINT32) Max Motor Speed 0x0080*/
  0x0, /*(UINT32) Profile Velocity 0x0081*/
  0x0, /*(UINT32) Profile Acceleration 0x0083*/
  0x0, /*(UINT32) Profile Deceleration 0x0084*/
  0x0, /*(UINT32) QuickStopDeceleration 0x0085*/
  0x0, /*(INT8) QuickStopDeceleration 0x0086*/
  {2,0,0}, /*T0B10089 Homing Speed*/
  0x0, /*(UINT32) QuickStopDeceleration 0x009A*/
  {2,1,-3}, /*T0B100C2 Interpolation Time Period*/
  0x0, /*(INT32) Target Velocity 0x009C*/
  0x0, /*(INT32) Target Velocity 0x009F*/
  0x0, /*(UINT32) Supported Drive Nodes 0x0502*/
}
#endif

```

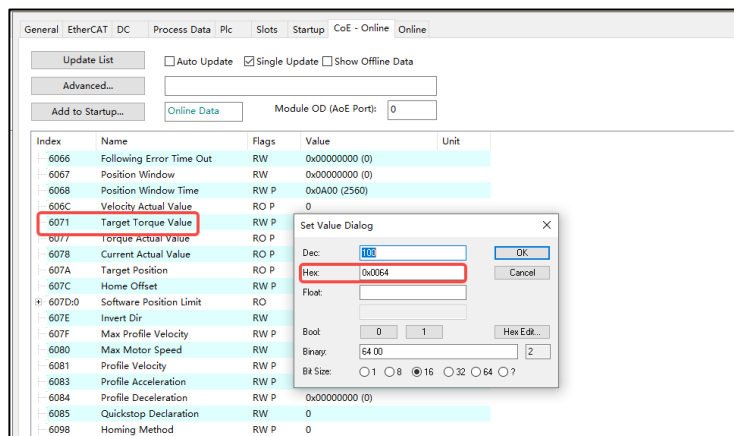
Add new dictionary members to the dictionary

```

/* Object 0x6068 */
{NULL,NULL, 0x6068, {DEFTYPE_UNSIGNED16, 0 | (OBJCODE_VAR << 8)}, &EntryDesc0x6068, aName0x6068, NULL, NULL, NULL, 0x0000 },
/* Object 0x606C */
{NULL,NULL, 0x606C, {DEFTYPE_INTEGER32, 0 | (OBJCODE_VAR << 8)}, &EntryDesc0x606C, aName0x606C, NULL, NULL, NULL, 0x0000 },
/* Object 0x6071 */
{NULL,NULL, 0x6071, {DEFTYPE_INTEGER16, 0 | (OBJCODE_VAR << 8)}, &EntryDesc0x6071, aName0x6071, NULL, NULL, NULL, 0x0000 },
/* Object 0x6077 */
{NULL,NULL, 0x6077, {DEFTYPE_INTEGER16, 0 | (OBJCODE_VAR << 8)}, &EntryDesc0x6077, aName0x6077, NULL, NULL, NULL, 0x0000 },
/* Object 0x6078 */
{NULL,NULL, 0x6078, {DEFTYPE_INTEGER16, 0 | (OBJCODE_VAR << 8)}, &EntryDesc0x6078, aName0x6078, NULL, NULL, NULL, 0x0000 },
/* Object 0x607D */
{NULL,NULL, 0x607D, {DEFTYPE_INTEGER32, 2 | (OBJCODE_ARR << 8)}, &EntryDesc0x607D, aName0x607D, NULL, NULL, NULL, 0x0000 },
/* Object 0x607A */
{NULL,NULL, 0x607A, {DEFTYPE_INTEGER32, 0 | (OBJCODE_VAR << 8)}, &EntryDesc0x607A, aName0x607A, NULL, NULL, NULL, 0x0000 },

```

- The master writes 0x64 to 0x6071, and the corresponding objTargetTorqueValue in the slave will receive the corresponding data. The user layer can read and call the corresponding data structure to obtain the variable value.



Index	Name	Flags	Value	Unit
6066	Following Error Time Out	RW	0x00000000 (0)	
6067	Position Window	RW	0x00000000 (0)	
6068	Position Window Time	RW P	0x0A00 (2560)	
606C	Velocity Actual Value	RO P	0	
6071	Target Torque Value	RW P		
6077	Torque Actual Value	RO P		
6078	Current Actual Value	RO P		
607A	Target Position	RO P		
607C	Home Offset	RW P		
607D:0	Software Position Limit	RO		
607E	Invert Dir	RW		
607F	Max Profile Velocity	RW P		
6080	Max Motor Speed	RW		
6081	Profile Velocity	RW P		
6083	Profile Acceleration	RW P		
6084	Profile Deceleration	RW P		
6085	Quickstop Declaration	RW	0	
6098	Homing Method	RW P	0	

Set Value Dialog

Dec: 108

Hex: 0x0064

Float:

Bool: 0 1

Binary: 64 00 2

Bit Size: 1 8 16 32 64 ?

LocalAxes[0]	0x240008B0 &LocalAx...	struct <untagged>
bAxisIsActive	0x01	uchar
bBrakeApplied	0x01	uchar
bLowLevelPowerApplied	0x01	uchar
bHighLevelPowerApplied	0x00	uchar
bAxisFunctionEnabled	0x00	uchar
bConfigurationAllowed	0x01	uchar
i16State	0x0002	ushort
u16PendingOptionCode	0x0000	ushort
fCurPosition	0	double
u32CycleTime	0x00000000	uint
Objects	0x240008CC	struct <untagged>
sRxPDOMap0	0x240008CC	struct <untagged>
sRxPDOMap1	0x240008E4	struct <untagged>
sRxPDOMap2	0x240008F4	struct <untagged>
sTxPDOMap0	0x24000C04	struct <untagged>
sTxPDOMap1	0x24000C1C	struct <untagged>
sTxPDOMap2	0x24000C2C	struct <untagged>
objErrorCode	0x0000	ushort
objControlWord	0x0000	ushort
objStatusWord	0x1221	ushort
objQuickStopOptionCode	0x0002	short
objShutdownOptionCode	0x0000	short
objDisableOperationOptionC...	0x0001	short
objHaltOptionCode	0x0001	short
objFaultReactionCode	0x0002	short
objModesOfOperation	0x0000	short
objModesOfOperationDisplay	0x0000	short
objPositionDemandValue	0x00000000	int
objPositionActualInternalValue	0x00000000	int
objPositionActualValue	0x00000000	int
objFollowingErrorWindow	0x00000000	uint
objFollowingErrorTimeOut	0x0000	ushort
objPositionWindow	0x00000000	uint
objPositionWindowTime	0x0A00	ushort
objVelocityActualValue	0x00000000	int
objTargetTorqueValue	0x0064	short
objTorqueActualValue	0x0000	short
objCurrentActualValue	0x0000	short
objTargetPosition	0x00000000	int
objHomeOffset	0x00000000	int

For TX data that is read-only by the master, modifying the value of objCurrentActualValue in the slave allows the master to read the data as well.

objModesOfOperationDisplay	0x0000	short
objPositionDemandValue	0x00000000	int
objPositionActualInternalValue	0x00000000	int
objPositionActualValue	0x00000000	int
objFollowingErrorWindow	0x00000000	uint
objFollowingErrorTimeOut	0x0000	ushort
objPositionWindow	0x00000000	uint
objPositionWindowTime	0x0A00	ushort
objVelocityActualValue	0x00000000	int
objTargetTorqueValue	0x0064	short
objTorqueActualValue	0x0000	short
objCurrentActualValue	0x0080	short
objTargetPosition	0x00000000	int
objHomeOffset	0x00000000	int
objSoftwarePositionLimit	0x24000C80	struct <untagged>
objInvertDir	0x00	uchar
objMaxProfileVelocity	0x00000000	uint
objMaxMotorSpeed	0x00000000	uint
objProfileVelocity	0x00000000	uint

6066	Following Error Time Out	RW	0x00000000 (0)
6067	Position Window	RW	0x00000000 (0)
6068	Position Window Time	RW P	0x0A00 (2560)
606C	Velocity Actual Value	RO P	0
6071	Target Torque Value	RW P	100
6077	Torque Actual Value	RO P	0
6078	Current Actual Value	RO P	128
607A	Target Position	RO P	0
607C	Home Offset	RW P	0
607D:0	Software Position Limit	RO	> 2 <
607E	Invert Dir	RW	0x00 (0)
607F	Max Profile Velocity	RW P	0x00000000 (0)
6080	Max Motor Speed	RW	0x00000000 (0)
6081	Profile Velocity	RW P	0x00000000 (0)
6083	Profile Acceleration	RW P	0x00000000 (0)

5.2 Adding PDO

Similarly, using the 23_EtherCAT example of the GD32H75EY_EVAL board as a reference, modify the XML and protocol stack code to add a set of parameters: Target Torque Value (0x6071) and Current Actual Value (0x6078), using PDO communication mode.

First, complete the XML file modification by following the method for adding SDOs, add the variables to the XML file, and then add the variables to the corresponding RXPDO and TXPDO.

1. Add the corresponding indexes to the respective RXPDO (0x1600) and TXPDO (0x1A00).

Add the corresponding indexes in DT1600.

<Name>DT1600</Name>
<BitSize>160</BitSize>
<SubItem>
<SubIdx>4</SubIdx>
<Name>SubIndex 004</Name>
<Type>INT</Type>
<BitSize>16</BitSize>
<BitOffs>128</BitOffs>
<Flags>
<Access>ro</Access>
<Category>o</Category>
</Flags>
</SubItem>
<Index>0x1600</Index>
<Name>csp/csv/pp RxPDO</Name>
<Type>DT1600</Type>
<BitSize>160</BitSize>
<Info>
<SubItem>
<Name>SubIndex 000</Name>
<Info>
<DefaultData>0</DefaultData>
</Info>
</SubItem>
<SubItem>
<Name>SubIndex 001</Name>
<Info>
<DefaultData>10004060</DefaultData>
</Info>
</SubItem>
<SubItem>
<Name>SubIndex 002</Name>
<Info>
<DefaultData>00000060</DefaultData>
</Info>
</SubItem>
<SubItem>
<Name>SubIndex 003</Name>
<Info>
<DefaultData>20007A60</DefaultData>
</Info>
</SubItem>
<SubItem>
<Name>SubIndex 004</Name>
<Info>
<DefaultData>2000FF60</DefaultData>
</Info>
</SubItem>
<SubItem>
<Name>SubIndex 005</Name>
<Info>
<DefaultData>10007160</DefaultData>
</Info>

Add the corresponding indexes in DT1A00.

```
<Name>DT1A00</Name>
<BitSize>160</BitSize>

<SubItem>
  <SubIdx>5</SubIdx>
  <Name>SubIndex 005</Name>
  <Type>INT</Type>
  <BitSize>16</BitSize>
  <BitOffs>128</BitOffs>
  <Flags>
    <Access>ro</Access>
    <Category>o</Category>
  </Flags>
</SubItem>

<Index>#x1A00</Index>
<Name>csp/csv/pp TxP00</Name>
<Type>DT1A00</Type>
<BitSize>160</BitSize>
<Info>
  <SubItem>
    <Name>SubIndex 000</Name>
    <Info>
      <DefaultData>05</DefaultData>
    </Info>
  </SubItem>
  <SubItem>
    <Name>SubIndex 001</Name>
    <Info>
      <DefaultData>10004160</DefaultData>
    </Info>
  </SubItem>
  <SubItem>
    <Name>SubIndex 002</Name>
    <Info>
      <DefaultData>00006160</DefaultData>
    </Info>
  </SubItem>
  <SubItem>
    <Name>SubIndex 003</Name>
    <Info>
      <DefaultData>20006460</DefaultData>
    </Info>
  </SubItem>
  <SubItem>
    <Name>SubIndex 004</Name>
    <Info>
      <DefaultData>20006C60</DefaultData>
    </Info>
  </SubItem>
  <SubItem>
    <Name>SubIndex 005</Name>
    <Info>
      <DefaultData>10007100</DefaultData>
    </Info>
  </SubItem>
</Info>
</SubItem>
```

2. Add the corresponding dictionary members in the required Module. In this example, add them in "#x119800", and add the corresponding dictionary members in RX and TX.

```
<Type ModuleId="x119800">csp,csv,pp - axis</Type>
<Name>dynamic switch between csp/csv/pp</Name>
<RxPdo Fixed="true" Sm="2">
  <Index DependOnSlot="true">#x1600</Index>
  <Name>Outputs</Name>
  <Entry>
    <Index DependOnSlot="true">#x6040</Index>
    <SubIndex>0</SubIndex>
    <BitLen>16</BitLen>
    <Name>Control Word</Name>
    <Comment>object 0x6040:0</Comment>
    <DataType>UINT</DataType>
  </Entry>
  <Entry>
    <Index DependOnSlot="true">#x607A</Index>
    <SubIndex>0</SubIndex>
    <BitLen>32</BitLen>
    <Name>TargetPosition</Name>
    <Comment>object 0x607A:0</Comment>
    <DataType>DINT</DataType>
  </Entry>
  <Entry>
    <Index DependOnSlot="true">#x60FF</Index>
    <SubIndex>0</SubIndex>
    <BitLen>32</BitLen>
    <Name>TargetVelocity</Name>
    <Comment>object 0x60FF:0</Comment>
    <DataType>DINT</DataType>
  </Entry>
  <Entry>
    <Index DependOnSlot="true">#x60FF</Index>
    <SubIndex>0</SubIndex>
    <BitLen>16</BitLen>
    <Name>TargetTorqueValue</Name>
    <Comment>object 0x6071:0</Comment>
    <DataType>INT</DataType>
  </Entry>
</Type>
```

```

<Index DependOnSlot="true">#x1a00</Index>
<Name>Inputs</Name>
<Entry>
  <Index DependOnSlot="true">#x6041</Index>
  <SubIndex>0</SubIndex>
  <BitLen>16</BitLen>
  <Name>Status Word</Name>
  <Comment>object 0x6041:0</Comment>
  <DataType>UINT</DataType>
</Entry>
<Entry>
  <Index DependOnSlot="true">#x6064</Index>
  <SubIndex>0</SubIndex>
  <BitLen>32</BitLen>
  <Name>ActualPosition</Name>
  <Comment>object 0x6064:0</Comment>
  <DataType>DINT</DataType>
</Entry>
<Entry>
  <Index DependOnSlot="true">#x606C</Index>
  <SubIndex>0</SubIndex>
  <BitLen>32</BitLen>
  <Name>ActualVelocity</Name>
  <Comment>object 0x606C:0</Comment>
  <DataType>DINT</DataType>
</Entry>
<Entry>
  <Index DependOnSlot="true">#x6077</Index>
  <SubIndex>0</SubIndex>
  <BitLen>16</BitLen>
  <Name>TorqueActualValue</Name>
  <Comment>object 0x6077:0</Comment>
  <DataType>INT</DataType>
</Entry>

```

Simultaneously add the corresponding dictionary to the Objects under the Dictionary in the Module.

```

<Object>
  <Index DependOnSlot="true">#x6071</Index>
  <Name>Target torque value</Name>
  <Type>INT</Type>
  <BitSize>16</BitSize>
  <Flags>
    <Access>rw</Access>
    <PdoMapping>r</PdoMapping>
  </Flags>
</Object>
<Object>
  <Index DependOnSlot="true">#x6077</Index>
  <Name>Torque actual value</Name>
  <Type>INT</Type>
  <BitSize>16</BitSize>
  <Flags>
    <Access>ro</Access>
    <PdoMapping>t</PdoMapping>
  </Flags>
</Object>
<Object>
  <Index DependOnSlot="true">#x6078</Index>
  <Name>Current actual value</Name>
  <Type>INT</Type>
  <BitSize>16</BitSize>
  <Flags>
    <Access>ro</Access>
    <PdoMapping>t</PdoMapping>
  </Flags>
</Object>

```

- Based on the code added above, continue to add code to support PDO access. First, modify the cia402appl.c file.
Start by adding the data access code for the TXPDO's Current Actual Value (0x6078).

```
void APPL_InputMapping(UINT16* pData)
{
    UINT16 j = 0;
    UINT16 *pTmpData = (UINT16 *)pData;
    UINT8 AxisIndex;

    for (j = 0; j < sTxPDOassign.u16SubIndex0; j++)
    {
        /*The Axis index is based on the PDO mapping offset (0x10)*/
        AxisIndex = ((sTxPDOassign.aEntries[j] & 0xF0) >> 4);

        switch ((sTxPDOassign.aEntries[j] & 0x000F))
        {
            case 0: //copy csp/csv TxPDO entries
            {
                *pTmpData++ = LocalAxes[AxisIndex].Objects.objStatusWord;
                *pTmpData++ = LocalAxes[AxisIndex].Objects.objPositionActualValue & 0xFFFF;
                *pTmpData++ = LocalAxes[AxisIndex].Objects.objPositionActualValue>>16;
                *pTmpData++ = LocalAxes[AxisIndex].Objects.objVelocityActualValue & 0xFFFF;
                *pTmpData++ = LocalAxes[AxisIndex].Objects.objVelocityActualValue>>16;

                *pTmpData++ = LocalAxes[AxisIndex].Objects.objCurrentActualValue & 0xFFFF;
                *pTmpData++ = LocalAxes[AxisIndex].Objects.objModesOfOperationDisplay & 0xFF;

            }
            break;
            case 1://copy csp TxPDO entries
            {
                *pTmpData++ = LocalAxes[AxisIndex].Objects.objStatusWord;
                *pTmpData++ = LocalAxes[AxisIndex].Objects.objPositionActualValue & 0xFFFF;
                *pTmpData++ = LocalAxes[AxisIndex].Objects.objPositionActualValue>>16;

            }
            break;
            case 2://copy csv TxPDO entries
            {
                *pTmpData++ = LocalAxes[AxisIndex].Objects.objStatusWord;
                *pTmpData++ = LocalAxes[AxisIndex].Objects.objVelocityActualValue & 0xFFFF;
                *pTmpData++ = LocalAxes[AxisIndex].Objects.objVelocityActualValue>>16;

            }
            break;
        } //switch TXPDO entry
    }
}
```

Then add the data access code for the RX's Target Torque Value (0x6071).

```
void APPL_OutputMapping(UINT16* pData)
{
    UINT16 j = 0;

    UINT16 *pTmpData = (UINT16 *)pData;
    UINT8 AxisIndex;

    for (j = 0; j < sRxPDOassign.u16SubIndex0; j++)
    {
        /*The Axis index is based on the PDO mapping offset (0x10)*/
        AxisIndex = ((sRxPDOassign.aEntries[j] & 0xF0) >> 4);

        switch ((sRxPDOassign.aEntries[j] & 0x000F))
        {
            case 0: //csp/csv RxPDO entries
            {
                LocalAxes[AxisIndex].Objects.objControlWord = SWAPWORD(*pTmpData++);
                LocalAxes[AxisIndex].Objects.objTargetPosition = SWAPWORD(*pTmpData++);
                LocalAxes[AxisIndex].Objects.objTargetPosition+=(SWAPWORD(*pTmpData++)<<16);
                LocalAxes[AxisIndex].Objects.objTargetVelocity = SWAPWORD(*pTmpData++);
                LocalAxes[AxisIndex].Objects.objTargetVelocity+=(SWAPWORD(*pTmpData++)<<16);
                LocalAxes[AxisIndex].Objects.objTargetTorqueValue = SWAPWORD(*pTmpData++);
                LocalAxes[AxisIndex].Objects.objModesOfOperation = SWAPWORD(*pTmpData++)&0xFF;

            }
            break;
            case 1: //csp RxPDO entries
            {
                LocalAxes[AxisIndex].Objects.objControlWord = SWAPWORD(*pTmpData++);
                LocalAxes[AxisIndex].Objects.objTargetPosition = SWAPWORD(*pTmpData++);
                LocalAxes[AxisIndex].Objects.objTargetPosition+=(SWAPWORD(*pTmpData++)<<16);

            }
            break;
            case 2: //csv RxPDO entries
            {
                LocalAxes[AxisIndex].Objects.objControlWord = SWAPWORD(*pTmpData++);
                LocalAxes[AxisIndex].Objects.objTargetVelocity = SWAPWORD(*pTmpData++);
                LocalAxes[AxisIndex].Objects.objTargetVelocity+=(SWAPWORD(*pTmpData++)<<16);

            }
            break;
        }
    }
}
```

4. Modify the code in cia402appl.h.
Increase the number of entities.

```

/** \brief 0x1600 (csp/csv RxPDO) data structure*/
typedef struct OBJ_STRUCT_PACKED_START {
    UINT16 u16SubIndex0; /**< \brief SubIndex 0*/
    UINT32 aEntries[6]; /**< \brief Entry buffer*/
} OBJ_STRUCT_PACKED_END
TOBJ1600;

/** \brief 0x1601 (csp RxPDO) data structure*/
typedef struct OBJ_STRUCT_PACKED_START {
    UINT16 u16SubIndex0; /**< \brief SubIndex 0*/
    UINT32 aEntries[3]; /**< \brief Entry buffer*/
} OBJ_STRUCT_PACKED_END
TOBJ1601;

/** \brief 0x1602 (csv RxPDO) data structure*/
typedef struct OBJ_STRUCT_PACKED_START {
    UINT16 u16SubIndex0; /**< \brief SubIndex 0*/
    UINT32 aEntries[3]; /**< \brief Entry buffer*/
} OBJ_STRUCT_PACKED_END
TOBJ1602;

/** \brief 0x1A00 (csp/csv TxPDO) data structure*/
typedef struct OBJ_STRUCT_PACKED_START {
    UINT16 u16SubIndex0; /**< \brief SubIndex 0*/
    UINT32 aEntries[4]; /**< \brief Entry buffer*/
} OBJ_STRUCT_PACKED_END
TOBJ1A00;

```

Add specific mapping member definitions.

```

/**
 * \brief Data structure to handle the process data transmitted via 0x1A00 (csp/csv TxPDO)*/
typedef struct STRUCT_PACKED_START
{
    UINT16 ObjStatusWord; /**< \brief Status word (0x6041)*/
    INT32 ObjPositionActualValue; /**< \brief Actual position (0x6064)*/
    INT32 ObjVelocityActualValue; /**< \brief Actual velocity (0x606C)*/
    INT16 ObjCurrentActualValue; /**< \brief Current Actual Value (0x6078)*/
    INT16 ObjModeOfOperation; /**< \brief Current mode of operation (0x6061)*/
}STRUCT_PACKED_END
TCIA402PD01A00;

/** \brief Data structure to handle the process data transmitted via 0x1A01 (csp TxPDO)*/
typedef struct STRUCT_PACKED_START
{
    UINT16 ObjStatusWord; /**< \brief Status word (0x6041)*/
    INT32 ObjPositionActualValue; /**< \brief Actual position (0x6064)*/
    INT16 Padding16Bit; /**< \brief 16bit padding*/
}STRUCT_PACKED_END
TCIA402PD01A01;

/** \brief Data structure to handle the process data transmitted via 0x1A02 (csv TxPDO)*/
typedef struct STRUCT_PACKED_START
{
    UINT16 ObjStatusWord; /**< \brief Status word (0x6041)*/
    INT32 ObjPositionActualValue; /**< \brief Actual position (0x6064)*/
    INT16 Padding16Bit; /**< \brief 16bit padding*/
}STRUCT_PACKED_END
TCIA402PD01A02;

/** \brief Data structure to handle the process data transmitted via 0x1600 (csp/csv RxPDO)*/
typedef struct STRUCT_PACKED_START
{
    UINT16 ObjControlWord; /**< \brief Control word (0x6040)*/
    INT32 ObjTargetPosition; /**< \brief Target position (0x607A)*/
    INT32 ObjTargetVelocity; /**< \brief Target velocity (0x607C)*/
    INT16 ObjTargetTorqueValue; /**< \brief Target Torque Value (0x6071)*/
    INT16 ObjModeOfOperation; /**< \brief mode of operation (0x6060)*/
}STRUCT_PACKED_END
TCIA402PD01600;

```

Add corresponding entity indexes.

```

/**
 * \brief Object 0x1600 (csp/csv RxPDO) entry descriptions
 */
OBJCONST TSDOINFORMATIONDESC OBJMEM asEntryDesc0x1600[] = {
    {DEFTYPE_UNSIGNED8, 0x8, ACCESS_READ }, /* Subindex 000 */
    {DEFTYPE_UNSIGNED32, 0x20, ACCESS_READ}, /* SubIndex 001*/
    {DEFTYPE_UNSIGNED32, 0x20, ACCESS_READ}, /* SubIndex 002*/
    {DEFTYPE_UNSIGNED32, 0x20, ACCESS_READ}, /* SubIndex 003*/
    {DEFTYPE_UNSIGNED32, 0x20, ACCESS_READ}, /* SubIndex 004*/
    {DEFTYPE_UNSIGNED32, 0x20, ACCESS_READ}, /* SubIndex 005*/
    {DEFTYPE_UNSIGNED32, 0x20, ACCESS_READ}; /* SubIndex 006*/
}

```



```

/**
 * \brief Object 0x1A00 (csp/csv TxPDO) entry descriptions
 */
OBJCONST TSDOINFOENTRYDESC OBJMEM asEntryDesc0x1A00[] = {
    {DEFTYPE_UNSIGNED8, 0x8, ACCESS_READ}, /* SubIndex 000 */
    {DEFTYPE_UNSIGNED32, 0x20, ACCESS_READ}, /* SubIndex 001 */
    {DEFTYPE_UNSIGNED32, 0x20, ACCESS_READ}, /* SubIndex 002 */
    {DEFTYPE_UNSIGNED32, 0x20, ACCESS_READ}, /* SubIndex 003 */
    {DEFTYPE_UNSIGNED32, 0x20, ACCESS_READ}, /* SubIndex 004 */
    {DEFTYPE_UNSIGNED32, 0x20, ACCESS_READ}, /* SubIndex 005 */
    {DEFTYPE_UNSIGNED32, 0x20, ACCESS_READ}; /* SubIndex 006 */
}

```

Add dictionary members to the corresponding mapping.

```

PROTO CiA402Objects DefCiA402ObjectValues
#ifdef _CiA402_
= {
    {6, {0x60400010, 0x607A0020, 0x60FF0020, 0x60710010, 0x60600008, 0x00000008}}, /* TOBJ1600 */
    {3, {0x60400010, 0x607A0020, 0x00000010}}, /* TOBJ1601 */
    {3, {0x60400010, 0x60FF0020, 0x00000010}}, /* TOBJ1602 */
    {6, {0x60410010, 0x60640020, 0x606C0020, 0x60780010, 0x60610008, 0x00000008}}, /* TOBJ1A00 */
}

```

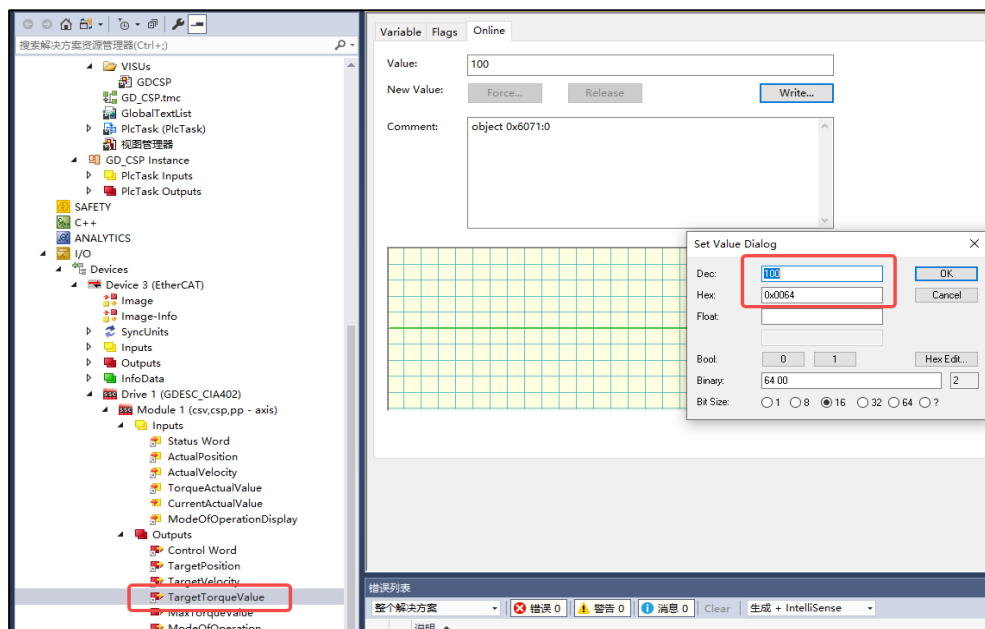
Modify the number of members in the mapping of the CiA402 Axis dictionary.

```

/** \brief Object dictionary related to on CiA402 Axis
 */
PROTO TOBJECT OBJMEM DefCiA402AxisObjDic[]
#ifdef _CiA402_
= {
    /* Object 0x1600 */
    {NULL, NULL, 0x1600, {DEFTYPE_PDOMAPPING, 6 | (OBJCODE_REC << 8)}, asEntryDesc0x1600, aName0x1600, NULL, NULL, NULL, 0x0000},
    /* Object 0x1601 */
    {NULL, NULL, 0x1601, {DEFTYPE_PDOMAPPING, 3 | (OBJCODE_REC << 8)}, asEntryDesc0x1601, aName0x1601, NULL, NULL, NULL, 0x0000},
    /* Object 0x1602 */
    {NULL, NULL, 0x1602, {DEFTYPE_PDOMAPPING, 3 | (OBJCODE_REC << 8)}, asEntryDesc0x1602, aName0x1602, NULL, NULL, NULL, 0x0000},
    /* Object 0x1A00 */
    {NULL, NULL, 0x1A00, {DEFTYPE_PDOMAPPING, 6 | (OBJCODE_REC << 8)}, asEntryDesc0x1A00, aName0x1A00, NULL, NULL, NULL, 0x0000},
}

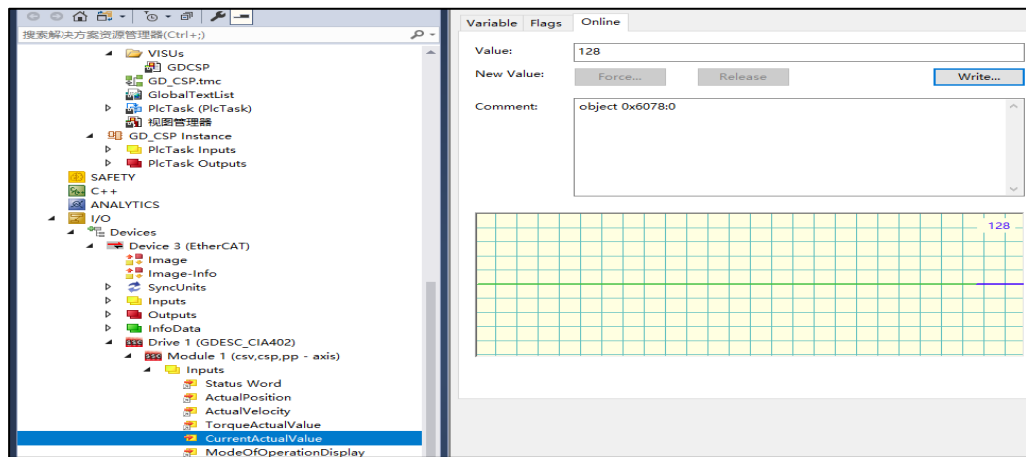
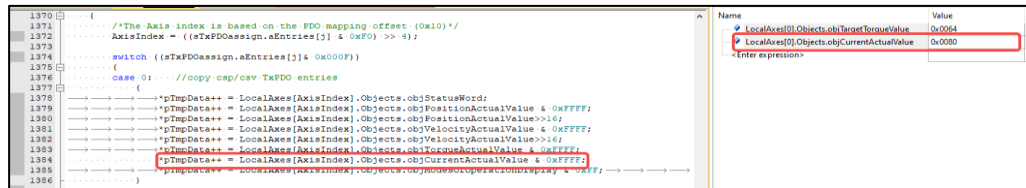
```

- The master writes 0x64 through RXPDO (0x6071), and the corresponding objTargetTorqueValue in the slave will receive the corresponding data. The user layer can read and call the corresponding data structure to obtain the variable value.





- For the TXPDO (0x6078) data that is read-only for the master, modifying the value of `objCurrentActualValue` in the slave allows the master to read the data as well.



6. Revision history

Table 6-1. Revision history

Revision No.	Description	Date
1.0	Initial Release	Apr.08 2025
1.1	1. Add EEPROM update method	Jul.10.2025
1.2	1. Add methods for COE to include SDO & PDO.	Sep.8.2025

Important Notice

This document is the property of GigaDevice Semiconductor Inc. and its subsidiaries (the "Company"). This document, including any product of the Company described in this document (the "Product"), is owned by the Company according to the laws of the People's Republic of China and other applicable laws. The Company reserves all rights under such laws and no Intellectual Property Rights are transferred (either wholly or partially) or licensed by the Company (either expressly or impliedly) herein. The names and brands of third party referred thereto (if any) are the property of their respective owner and referred to for identification purposes only.

To the maximum extent permitted by applicable law, the Company makes no representations or warranties of any kind, express or implied, with regard to the merchantability and the fitness for a particular purpose of the Product, nor does the Company assume any liability arising out of the application or use of any Product. Any information provided in this document is provided only for reference purposes. It is the sole responsibility of the user of this document to determine whether the Product is suitable and fit for its applications and products planned, and properly design, program, and test the functionality and safety of its applications and products planned using the Product. The Product is designed, developed, and/or manufactured for ordinary business, industrial, personal, and/or household applications only, and the Product is not designed or intended for use in (i) safety critical applications such as weapons systems, nuclear facilities, atomic energy controller, combustion controller, aeronautic or aerospace applications, traffic signal instruments, pollution control or hazardous substance management; (ii) life-support systems, other medical equipment or systems (including life support equipment and surgical implants); (iii) automotive applications or environments, including but not limited to applications for active and passive safety of automobiles (regardless of front market or aftermarket), for example, EPS, braking, ADAS (camera/fusion), EMS, TCU, BMS, BSG, TPMS, Airbag, Suspension, DMS, ICMS, Domain, ESC, DCDC, e-clutch, advanced-lighting, etc.. Automobile herein means a vehicle propelled by a self-contained motor, engine or the like, such as, without limitation, cars, trucks, motorcycles, electric cars, and other transportation devices; and/or (iv) other uses where the failure of the device or the Product can reasonably be expected to result in personal injury, death, or severe property or environmental damage (collectively "Unintended Uses"). Customers shall take any and all actions to ensure the Product meets the applicable laws and regulations. The Company is not liable for, in whole or in part, and customers shall hereby release the Company as well as its suppliers and/or distributors from, any claim, damage, or other liability arising from or related to all Unintended Uses of the Product. Customers shall indemnify and hold the Company, and its officers, employees, subsidiaries, affiliates as well as its suppliers and/or distributors harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of the Product.

Information in this document is provided solely in connection with the Product. The Company reserves the right to make changes, corrections, modifications or improvements to this document and the Product described herein at any time without notice. The Company shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. Information in this document supersedes and replaces information previously supplied in any prior versions of this document.