GCAN-305

Embedded CANopen (slave) to UART

User Manual

Document version 3.23 (2017/03/22)

Revision History:

Version	Date	Reason
V1.00	2013/06/16	Create document
V2.01	2013/12/20	Fixed working parameter
V3.01	2015/04/22	Add some parameters
V3.02	2015/08/08	Add some parameter
V3.23	2017/03/22	Add Modbus common function code description

Contents

1. Introduction

1.1 Overview

GCAN-305 is an embedded CANopen(slave) conversion module. The CANopen slave protocol stack code has been integrated in this module and does not require to do secondary development. Protocol stack conform to CANopen protocol description file DS301, DS302, DS303, and DS305. By default, CANopen slave station enable predefined connections and support parameters storage.

GCAN-305 has 1 channel CAN interface, which can be connected to CANopen Bus. In addition, GCAN-305 has two UART interfaces - a communication UART interface (communication baud rate of 1200 ~ 115200bps), a debug UART interface (fixed baud rate 115200bps).

GCAN-305 is suitable for any serial port system. With this module, the existing serial communication device can have CANopen communication capability at the fastest speed. Users only need to use a simple serial communication protocol to achieve communication with GCAN-305, complete the module configuration and data exchange with the CANopen Bus.

Kernel program can be upgrated remotely, customized EDS file and program firmware is also available, and this help to maintain the CANopen function of the your device.

1.2 Performance

- Network management service object (NMT: Boot up, Node Guarding /Life guarding, Heartbeat Producer)
- Process data object (12 TPDO and 12 RPDO)
- Support service object (SDO server)
- Support emergency message object (Emergency)
- Support synchronization message object (Sync)
- Support network configuration object (LSS slaves)
- Support network time consumption
- Serial baud rate: 1200 ~ 115200bps, can be customized
- Serial communication capability (UART, TTL level)
- 96-byte input and output data buffer (I/O)
- Support DIP switch to set slave station number (1~127) and CAN baud rate (20kbps,

50kbps, 100kbps, 125kbps, 250kbps, 500kbps, 800kbps, 1000kbps), this is function of GCAN-305 Development board

- Max CAN->UART conversion time: 1ms
- Max UART->CAN conversion time: 2ms
- Size: 32mm×20.4mm×11mm (DIP24 package)
- Working voltage/current: +5V/80mA
- IO voltage: 3.3V
- Working and storage temperature: -40℃~85℃

1.3 Static parameters

Shown in Table 1.1. Each parameter is measured at room temperature.

Symbol	Definition	Test Conditions	Max	Min	Unit
Power					
V_{DD}	Supply voltage		4.5	6	V
I_{DD}	Supply current	All I/O no connect, input voltage 5V	65	85	mA
I/O interface					
V_{IL}	Low-level input voltage	V_{DD} = 5V	-	0.8	V
V_{IH}	High-level input voltage	V_{DD} = 5V	2.0	5.5	V
V_{OL}	Low-level output voltage	I_{OL} = -4mA	-	0.4	V
V_{OH}	High-level output voltage	I_{OH} = -4mA	2.6	3.3	V
I_{OL}	Low-level output current	V_{OL} = 0.4V	-	4	mA
I_{OH}	High-level output current	$2.6V \le V_{OH} \le V_{DD}$	-	-4	mA
I_{OL}	Low-level short circuit current	$2.6V \le V_{OL} \le 3.3V$	-	50	mA
I_{OH}	High-level short circuit current	V_{OH} = 0V	-	-45	mA

Table 1.1 Static Parameters of GCAN-305

2. Installation

This chapter describes the connection methods and precautions of GCAN-305.

2.1 System connection structure

System structure of GCAN-305 as shown in Figure 2.1, connect CPU serial port (TTL level) with the serial port of GCAN-305, and connect the CAN-bus interface of the module to the CANopen network through the CAN transceiver to establish the CPU and Bridging of CANopen networks. The data sent by the serial port of the CPU will be sent to the CAN-bus in the form of PDO messages and read the RPDO data from the CAN-bus. Of course, the user can also configure the parameters of module through the serial port.

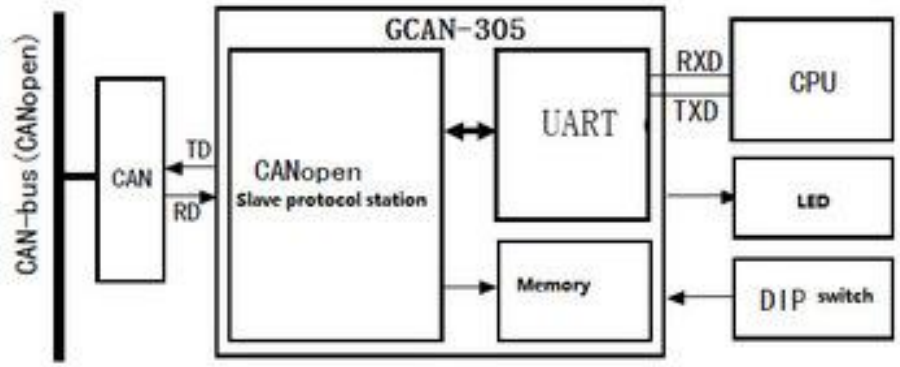


Figure 2.1 GCAN-305 system structure

The GCAN-305 module can use the DIP switch to set the device's Node ID and CAN communication baud rate. In special cases, it can also be set without using the DIP switch. The module's node number and CAN communication baud rate can be set via the UART interface or CANopen layer (LSS).

2.2 Module interface definition

The dimensions of the GCAN-305 are shown in Figure 2.2, 32*20.4*11 (length*width*height, unit: mm). Its definition is shown in Table 2.1.

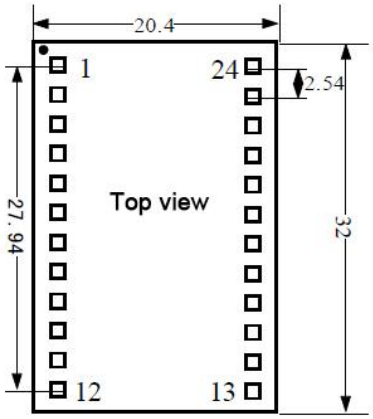


Figure 2.2 RS485 interface definitions

Number	Name	Function	Number	Name	Function
1	GND	ground	24	V_{CC}	Power input（5V）
2	/Rst	Reset	23	CAN-R	CAN receive
3	UART1-T	Serial port transmit	22	CAN-T	CAN transmit
4	UART1-R	Serial port receive	21	ID0	Input node number 0
5	UART0-T	Debug/Update serial port transmit	20	ID1	Input node number 1
6	UART0-R	Debug/Update serial port receive	19	ID2	Input node number 2
7	/INT	Interrupt pin	18	ID3	Input node number 3
8	Brt0	Baud rate set 0	17	ID4	Input node number 4
9	Brt0	Baud rate set 1	16	ID5	Input node number 5
10	Brt0	Baud rate set 2	15	ID6	Input node number 6
11	Brt0	Baud rate set 3	14	E-Led	Error indicator（red）
12	/ISP-EN	Enable update	13	R-Led	Run indicator（green）

Table 2.1 The pin definition of module

Since the CAN transceiver is not integrated in the GCAN-305 module, an external CAN transceiver is required.

UART0 serial port is the debugging output and program upgrade interface of GCAN-305 module. It is recommended that the serial port be exported during the product debugging stage to facilitate users to debug their own programs. It is not necessary to export the serial port when the product stability is guaranteed.

Typical schematic and PCB design, please see the GCAN-305 technical documentation for details.

3. Usage

3.1 GCAN-305 evaluation baseboard

Provide users with GCAN-305 supporting evaluation, development, debugging, test baseboard, as shown in Figure 3.1. The baseboard covers all GCAN-305 functions, making it easy for users to develop and debug modules.

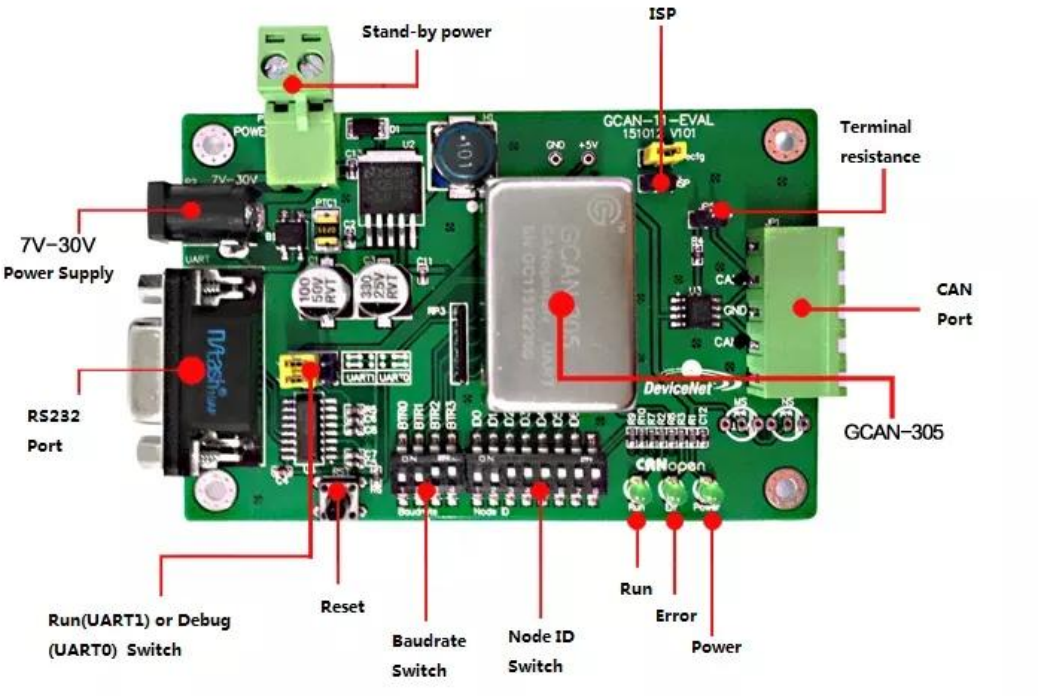


Figure 3.1 GCAN-305 evaluation board

3.2 Module state transition

The GCAN-305 state transition diagram shown in Figure 3.2, the various letters in the figure can be performed under various conditions,

a. NMT

b. Node Guard

c. SDO

d. Emergency

e. PDO

f. Boot-up

The arrows in the figure represent the conversion relations between the various states,

and the numbers indicate the operations that need to be performed in this conversion:

1：Start_ Remote_ Node (0x01)

2：Stop_ Remote_ Node (0x02)

3：Enter_ Pre-Operational_ State (0x80)

4：Reset_ Node (0x81)

5：Reset_ Communication (0x82)

6：Finish the initialization, enter Pre_ Operational state automatically, send Boot-up message

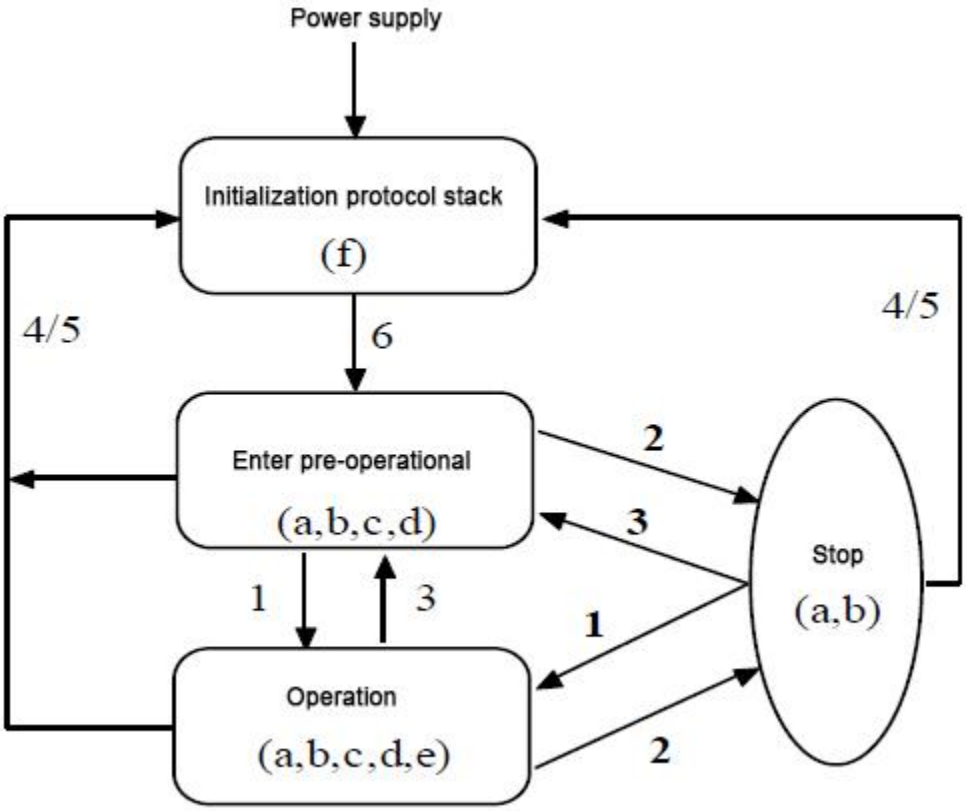


Figure 3.2 GCAN-305 state transition diagram

3.3 System state indicator

According to the definition of the CANopen protocol specification document DS303-3, two LEDs are used in the GCAN-305 module to indicate the current state of the module, as shown in Table 3.1.

Indicator	Color	Pin

RUN	Green	13
ERR	Red	14

Table 3.1 System state indicator

The meanings of various statuses indicated by the status indicators are shown in Table 3.2 and Table 3.3.

NO.	ERR LED	State	Description
1	Dark	No error	Device is in working condition
2	Flash it	Arrival alert value	At least one error counter of the CAN controller reached or exceeded the alert value (too many error frames)
3	Bright	Bus closed	CAN controller bus close

Table 3.2 Error Status Indicator (ERR) State Meaning

NO.	RUN LED	State	Description
1	dark	breakdown	Please check whether the module reset pin and power supply are connected correctly
2	Flash	Pre-operational	pre-operational state
3	bright	Operation	working state

Table 3.3 Run Status Indicator (ERR) State Meaning

3.4 GCAN-305 node ID and CAN-bus baud rate

The GCAN-305 module provides three ways to set the node ID and node baud rate. The setting order is shown in Figure 3.3 and Figure 3.4. If the DIP switch value is valid, enable the value at power-on. And even if a valid ID value is stored in the memory, it will not be used. If the master station performs LSS setting on the module during operation, the value set by the LSS is used. However, after the module is powered on or restarted, the value of the DIP switch is still used.

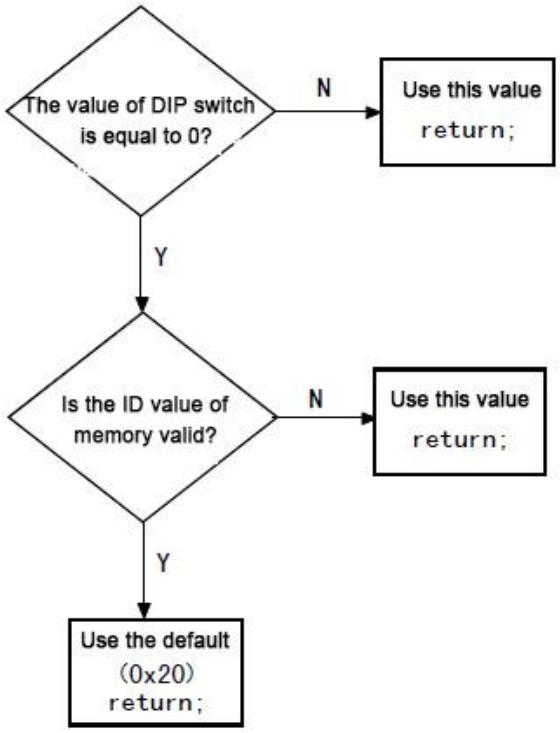


Figure 3.3 Node ID setting order

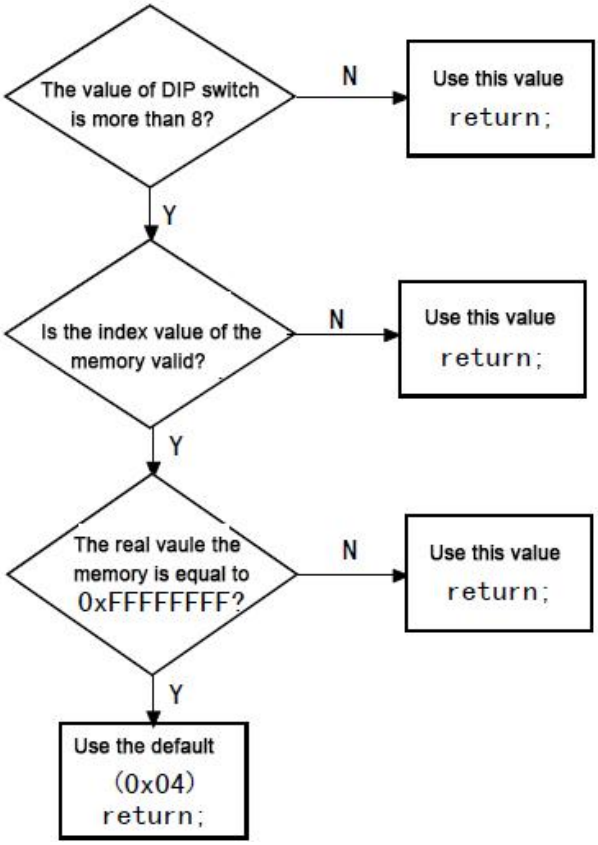
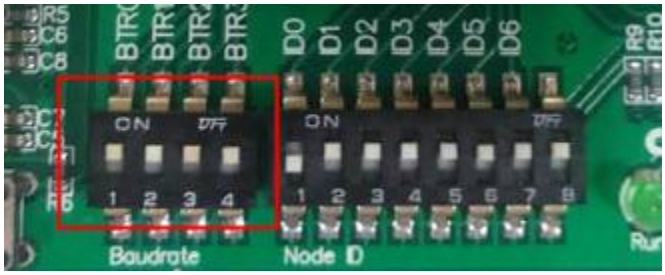


Figure 3.4 Baud rate setting order

The DIP switch for setting the node ID uses 7 bits. When the value is 0, the DIP switch is invalid. The value range is 1 to 127. The index value of baud rate uses 4 bits, and dial up to 0 and down to 1. **When the baud rate DIP switch 4 is 1 (the DIP switch 4 is down), the baud rate is determined by software. When the baud rate dip switch 4 is 0 (that is, the dip switch 4 is upward), the baud rate is determined by the dip switch. See Table 3.4 for details.**



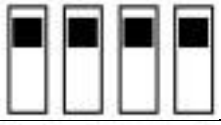
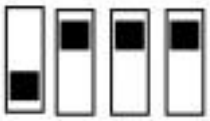
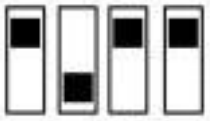
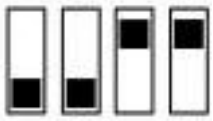
	Baud rate
	1000k
	800k
	500k
	250k
	125k
	100k
	50k
	20k

Figure 3.5 The relationship between baud rate index and actual comparison

4. CANopen protocol used in GCAN-305 (DS301)

4.1 GCAN-305 predefined connection

The GCAN-305 module uses the 0x1000~0x1FFF of the object dictionary and the 0x2000~0x5FFF of the manufacturer's custom area. These object dictionaries are responsible for communication and data exchange between CANopen and other application data on the CAN network. The object dictionary is defined as indexes and sub-indexes. Each object dictionary item has its own data length (UINT8, UINT16, UINT32, etc) and attributes (RO, WO, RW, CONST, MAPPALE). The data of these object dictionaries can be modified by the SDO service. Of course, only the attributes of these items must be modified by WO or RW.

The predefined connection means that the COB-ID related communication is associated with the node ID. The specific predefined connection set is shown in Table 4.1.

Object	Function code	Node address	COB-ID	Object dictionary index
Broadcast message				
NMT	0000	-	0	-
SYNC	0001	-	0x80	0x1005, 0x1006, 0x1007
TIME STAMP	0010	-	0x100	0x1012, 0x1013
Point-to-point messages				
Emergency	0001	1-127	0x81-0xFF	0x1014, 0x1015
TPDO1	0011	1-127	0x181-0x1FF	0x1800
RPDO1	0100	1-127	0x201-0x27F	0x1400
TPDO2	0101	1-127	0x281-0x2FF	0x1801
RPDO2	0110	1-127	0x301-0x37F	0x1401
TPDO3	0111	1-127	0x381-0x3FF	0x1802
RPDO3	1000	1-127	0x401-0x47F	0x1402
TPDO4	1001	1-127	0x481-0x4FF	0x1803
RPDO4	1010	1-127	0x501-0x57F	0x1403
Default SDO (tx)	1011	1-127	0x581-0x5FF	0x1200
Default SDO (rx)	1100	1-127	0x601-0x67F	0x1200
NMT error control	1110	1-127	0x701-0x77F	0x1016, 0x1017

Table 4.1 CANopen predefined connection set

4.2 GCAN-305 operation

4.2.1 Network management Service (NMT)

1. Network control (NMT Module Control)

The GCAN-305 supports the network management commands defined by the DS301. These network management commands can be sent from the CANopen master station or other slave nodes. The operation commands are shown in Table 4.2. When Node ID=0,

all slave devices are controlled (broadcast mode), CS is the command word corresponding to different control actions as shown in Table 4.3.

COB-ID(CAN-ID)	DLC	BYTE0	BYTE1
0x000	2	CS (command specifier)	Node ID (Node number)

Table 4.2 NMT operation instruction

CS （command specifier）	NMT service （control action）
0x01	Start slave device
0x02	Stop slave node device
0x80	Enable slave into preoperational
0x81	Reset slave node
0x82	Reset node communication

Table 4.3 NMT command specifier and its function service

Example: To start all nodes in the CANopen network, use the commands shown in Table 4.4.

COB-ID(CAN-ID)	DLC	BYTE0	BYTE1
0x000	2	0x01	0x00

Table 4.4 NMT start slave node

If you need control a specific device in the network to enter the pre-operation state, assuming that the node address is 0x03, the commands are shown in Table 4.5.

COB-ID(CAN-ID)	DLC	BYTE0	BYTE1
0x000	2	0x80	0x03

Table 4.5 NMT start specified slave node

2. Node guarding (NMT Node Guarding)

With the node guarding service, the NMT master node can check the current status of each node. This service is particularly meaningful when these nodes have no data transmission. The master node triggers the node guarding of the corresponding slave node by sending a remote frame. The command format is shown in Table 4.6. The slave node responds to the corresponding format as shown in Figure 4.7.

Master node → Slave node (Command)

COB-ID(CAN-ID)	DLC
0x700 + Node ID	1

Table 4.6 NMT master node guarding command frame (remote frame)

Slave node → master node (response):

COB-ID(CAN-ID)	DLC	BYTE0
0x700 + Node ID	1	Bit7:trigger bit, Bit0~Bit6 status

Table 4.7 NMT slave node response frame

The top bit (bit7) in Byte0 is the trigger bit. That is, every time slave station sends a frame, the answer will alternate change(0, 1) to show the difference between frames and frames. Among them, Bit0~Bit6 is the status of the slave node. The status of the slave station expressed by this value is shown in Table 4.8.

Value	Status
0x00	Initializing
0x04	Stopped
0x05	Operational
0x7F	Pre-operational

Table 4.8 The status value of node guarding

Example: Assume that the master node needs node guarding for the slave node number 0x03. The commands are shown in Table 4.9. The slave node response frames are shown in Table 4.10.

Master node → Slave node:

COB-ID(CAN-ID)	DLC
0x703	1

Table 4.9 Node guarding (remote frame)

Slave node → master node:

COB-ID(CAN-ID)	DLC	BYTE0
0x703	1	0x85

Table 4.10 Slave node(0*03) response frame

Bit7=1 of BYTE0 and status=0x05, indicating that the slave with node number 0x03 is in operation.

3. Life guarding (NMT Life Guarding)

Node guarding is mainly aimed at obtaining the status of the slave node from the master node of the NMT, and the lifetime guarding is the monitoring of the node to another node. It includes two parameters, namely guarding time and life factor. The node with lifetime guarding receives remote frames from another node (the remote frame format is the same as the node guarding frame format shown in Table 4.6). The node with lifetime guarding receives it, and the remote frame responds to the state of the node (the response frame format is shown in Table 4.7).

The two parameters of life guarding: guarding time and life factor (0x100C and 0x100D respectively in the object dictionary) constitute the life time of the node (life time = guarding time x life factor), the unit of guarding time is milliseconds. If one of the two parameters is 0, indicating that life protection is not enabled. If the remote frame is not received within the guarding time, the “Message Lost” message will appear. If the

remote frame is not received within the lifetime, the “Connection Lost” message will appear. These messages will be printed out in the debug serial port. The error indicator appears "blinks twice" to indicate the loss of current life guarding.

4. Start message (NMT Boot-up)

When the GCAN-305 is initialized (Boot-up), an identification message will be sent. The format of the message is shown in Table 4.11.

COB-ID(CAN-ID)	DLC	BYTE0
0x700 +NodeID	1	0x00

Table 4.11 Initialization identifier message format

Example: If the node number of GCAN-305 is 0x03, then the start message sent is shown in Table 4.12.

COB-ID(CAN-ID)	DLC	BYTE0
0x703	1	0x00

Table 4.12 0x703 node initialization identifier message format

5. Heartbeat Producer

Heartbeat producers are divided into producers and consumers. In the GCAN-305 module, only heartbeat producer is supported. That is, GCAN-305 can produce heartbeat producer. This parameter is defined in the object dictionary 0x1017 (data length is 16 bits, unit: milliseconds), and its heartbeat producer is shown in Table 4.7. It is the same as node guarding and lifetime guarding response frame.

Example: Assume that the node address is 0x03, it is in the operating state and the parameter in 0x1017 is set to 100. Then the slave node sends a frame as shown in Table 4.13 every 100 milliseconds.

COB-ID(CAN-ID)	DLC	BYTE0
0x703	1	0x05

Table 4.13 Slave node (0x03) heartbeat producer

Note: Life guarding and heartbeat messages cannot be used at the same time in the same GCAN-305 module.

4.2.2 Synchronous message object (SYNC)

Synchronous messages are divided into consumption and production. In GCAN-305, only the consumption of synchronous messages is supported. That is, synchronous messages are received from the master node or other nodes. The frame structure of the synchronous messages is shown in Table 4.14. 0x1005 of the object dictionary defines the COB-ID of the received synchronization message, its value is defined as 0x80 in the CANopen predefined connection set, and 0x1007 of the object dictionary defines the synchronous

time window (Require the maximum time interval for updating the corresponding data after receiving the synchronous message). The synchronous message is mainly used in the process of receiving and transmitting PDO. Its usage is shown in the following PDO data transmitting and receiving processes.

COB-ID(CAN-ID)	DLC
0x80	0

Table 4.14 Synchronous message frame format (remote frame)

4.2.3 Emergency message object (EMCY)

The GCAN-305 supports emergency messages, that is, when the GCAN-305 has an internal error, the message is sent. The format of the message is shown in Table 4.15. The emergency error code specifies the specific type of error that is currently occurring. The error register stores the current error type. Based on this value, the error type represented by the current emergency message can be determined. The definition is shown in Table 4.16.

COB-ID(CAN-ID)	DLC	BYTE0	BYTE1	BYTE2	BYTE3－BYTE7
0x80 + Node ID	8	emergency error code		error register	Manufacturer specified information
		Index 0x1003		Index 0x1001	

Table 4.15 Emergency message frame format

Bit	Error Type
0	Generic
1	Current
2	Voltage
3	Temperature
4	Communication
5	Device profile specific
6	Reserved(=0)
7	Manufacturer specific

Table 4.16 Error register

The meaning of the emergency error code is shown in Table 4.17.

Emergency error code	Code function description
00xx	Error Reset or No Error
10xx	Generic Error
20xx	Current
21xx	Current，device input side
22xx	Current，inside the device
23xx	Current，device output side
30xx	Voltage
31xx	Mains voltage
32xx	Voltage inside the device
33xx	Output voltage
40xx	Temperature

41xx	Ambient temperature
42xx	Device 16emperature
50xx	Device hardware
60xx	Device software
61xx	Internal software
62xx	User software
63xx	Data set
70xx	Additional modules
80xx	Monitoring
81xx	communication
8110	CAN overrun
8120	Error Passive
8130	Life Guard Error or Heartbeat Error
8140	Recovered from Bus-Off
82xx	Protocol Error
8210	PDO no processed Due to length error
8220	Length exceed
90xx	External error
F0xx	Additional functions
FFxx	Device specific

Table 4.17 Emergency error code

Example: If the node address is 0x03 and the CAN-bus error exceeds the alert value, an "Error Passive" (8120) warning will appear. As shown in Table 4.18.

COB-ID(CAN-ID)	DLC	BYTE0	BYTE1	BYTE2	BYTE3－BYTE7
0x83	8	0x20	0x81	0x11	0x00000000

Table 4.18 Emergency message (bus error)

Note: These errors are issued automatically when an emergency occurs in the GCAN-305 module.

4.2.4 Service data object (SDO)

Object dictionary acts as the main data exchange medium between application layer and communication layer. All data items of a CANopen device can be managed in the object dictionary. Each object dictionary item can be positioned using indexes and sub-indexes. The service data objects (SDO) of the CANopen definition to access these items. GCAN-305 supports 1 SDO server, which can provide SDO service, and SDO uses the predefined connection to transmit and receive COB-ID, 0x580 + Node ID (transmit) and 0x600 + Node ID (receive). SDO is divided into accelerated transmission, segment transmission, and block transmission. Because the acceleration transmission of SDO is often used in GCAN-305, this document focuses on the acceleration transmission, and the other transmission types, please refer to CANopen DS301 and related protocol documents.

1. SDO data transmission

Accelerated transmission of a frame can only transmit up to 4 bytes of data. The basic structure of the message is shown in Table 4.19 and Table 4.20. The SDO command word distinguish the type of the frame data.

COB-ID(CAN-ID)	DLC	Byte 0	Byte 1-2	Byte 3	Byte 4-7
0x600 + Node ID	8	CMD (SDO command word)	Object index	Object sub-index	**

Table 4.19 SDO message format (Client→Server)

COB-ID(CAN-ID)	DLC	Byte 0	Byte 1-2	Byte 3	Byte 4-7
0x580 + Node ID	8	CMD (SDO command word)	Object index	Object sub-index	**

Table 4.20 SDO response format (Server→Client)

2. SDO abort service

When an error occurs during SDO transmission, both the SDO client and the server can send this message to inform each other to abort the current operation. The format of the abort message is shown in Table 4.21 and Table 4.22. The specific meaning of the abort error code can refer to Table 4.23.

COB-ID(CAN-ID)	DLC	Byte 0	Byte 1-2	Byte 3	Byte 4-7
0x600 + Node ID /0x580+Node ID	8	CMD(SDO command word)	index	sub-index	**

Table 4.21 Abort message format

CMD（SDO command byte）								
Bit	7	6	5	4	3	2	1	0
Client→Server/ Client→Server	1	0	0	-	-	-	-	-

Table 4.22 The definition of abort message command word

Abort code	Code function description
0503 0000	Trigger bits do not alternate
0504 0000	SDO timeout
0504 0001	Illegal or unknown Client/Server command word
0504 0002	Invalid block size (Block Transfer mode only)
0504 0003	Invalid serial number (Block Transfer mode only)
0503 0004	CRC error (Block Transfer mode only)
0503 0005	Memory overflow
0601 0000	Object does not support access
0601 0001	Attempt to read a write-only object
0601 0002	Attempt to write a read-only object
0602 0000	The object does not exist in the object dictionary
0604 0041	Objects cannot be mapped to PDO
0604 0042	The number and length of mapped objects exceed the PDO length
0604 0043	General parameters are not compatible
0604 0047	General device internal is incompatibility
0606 0000	Hardware error causes object access to fail
0606 0010	Data type mismatch, service parameter length does not match

0606 0012	Data type mismatch, service parameter length is too large
0606 0013	Data type mismatch, service parameter length is too short
0609 0011	Sub-index does not exist
0609 0030	Out of parameter value range (when write access)
0609 0031	parameter value written is too large
0609 0032	parameter value written is too small
0609 0036	The maximum value is less than the minimum
0800 0000	General error
0800 0020	Data cannot be transferred or saved to the application
0800 0021	Data cannot be transferred or saved to the application due to local control
0800 0022	Data cannot be transferred or saved to the application due to current device status
0800 0023	The object dictionary dynamically generates an error or the object dictionary does not exist (for example, an object dictionary is generated from a file, but an error occurs due to damaged file)

Table 4.23 Abort error code

4.2.5 Process data object (PDO)

Process data objects (PDOs) are used for transmitting real-time data. The recipients of PDOs can be master nodes or other slave nodes and do not need to respond. Four TPDOs (index range 0x1800~0x1803) and four RPDOs (0x1400~0x1403) defined by the connection set are predefined at the factory.

1. Receive process data object (RPDO)

When the GCAN-305 is at the factory, the mapping object is predefined for each PDO. As shown in Figure 4.1, all RPDO data mapping items are connected to the 8 bit output area of the GCAN-305 module by default. That is, when the RPDO receives data from the network, it updates the data to the corresponding output data area. When the data update is completed, GCAN-305 will give an interrupt signal (high level→low level). The interrupt signal pin will remain low level when data is not read, and the interrupt pin will remain high level after data is read.

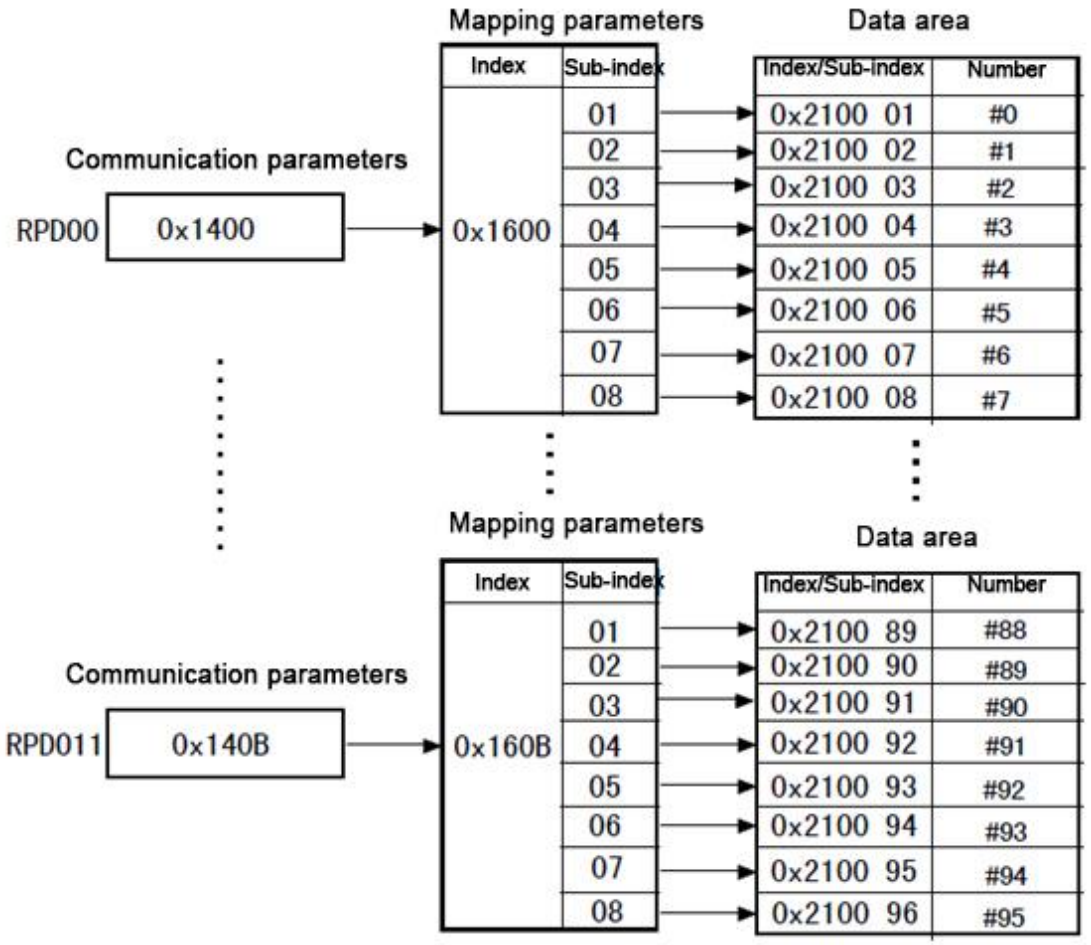


Figure 4.23 RPDO mapping relationship

The communication parameters of RPDO are less than TPDO, and RPDO only has one transmission type corresponding communication. Its value is defined in Table 4.24.

Transmission Type	Receive PDO	Data update
0	PDO will always be received and analyzed. If necessary, the data is updated when the next valid SYNC message is received.	Data is analyzed when a SYNC message is received. If the previous RPDO has changed, the data will be updated on the output. The transmission of SYNC messages is acyclic.
1-240		The data is analyzed when the nth numbered SYNC message is received. If the previous RPDO has changed, the data will be updated on the output. The transmission type corresponds to the value n. The transmission of SYNC messages is cyclic.
241-251	Retain	

252	Retain	
253	Retain	
254	PDO will always be received	The application defines and updates the events of the output data.
255	PDO will always be received	The device sub-protocol defines and updates the events of the output data.

Table 4.24 The transmission type of RPDO

Example: Assume that the GCAN-305 node is 0x03 and adopts a predefined connection, and the COB-ID of RPDO0 is 0x203. The received TPDO COB-ID should also be 0x203 as shown in Table 4.25. The TPDO is exactly the same as the COB-ID of RPDO1 with Node ID 0x03, the RPDO receives this frame of PDO data, and the mapping diagram as shown in Figure 4.1. updates the data to the data output area. The corresponding data of the output buffer is shown in Table 4.26.

COB-ID	DLC	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x203	8	0x11	0x22	0x33	0x44	0x55	0x66	0x77	0x88

Table 4.25 RPDO0 receives TPDO of the other nodes

The number of data area	#0	#1	#2	#3	#4	#5	#6	#7
Data	0x11	0x22	0x33	0x44	0x55	0x66	0x77	0x88

Table 4.26 The data in data area

2. Transmit process data object (TPDO)

Up to 12 TPDOs are supported in the GCAN-305. The four pre-defined PDOs are available at the factory, namely TPDO0~TPDO3, and TPDO4~TPDO11 are not available. Pre-defined TPDOs have been pre-defined mapping parameters at the factory and mapped to data input areas 0x2000 01 ~ 0x2000 96, as shown in Figure 4.2.

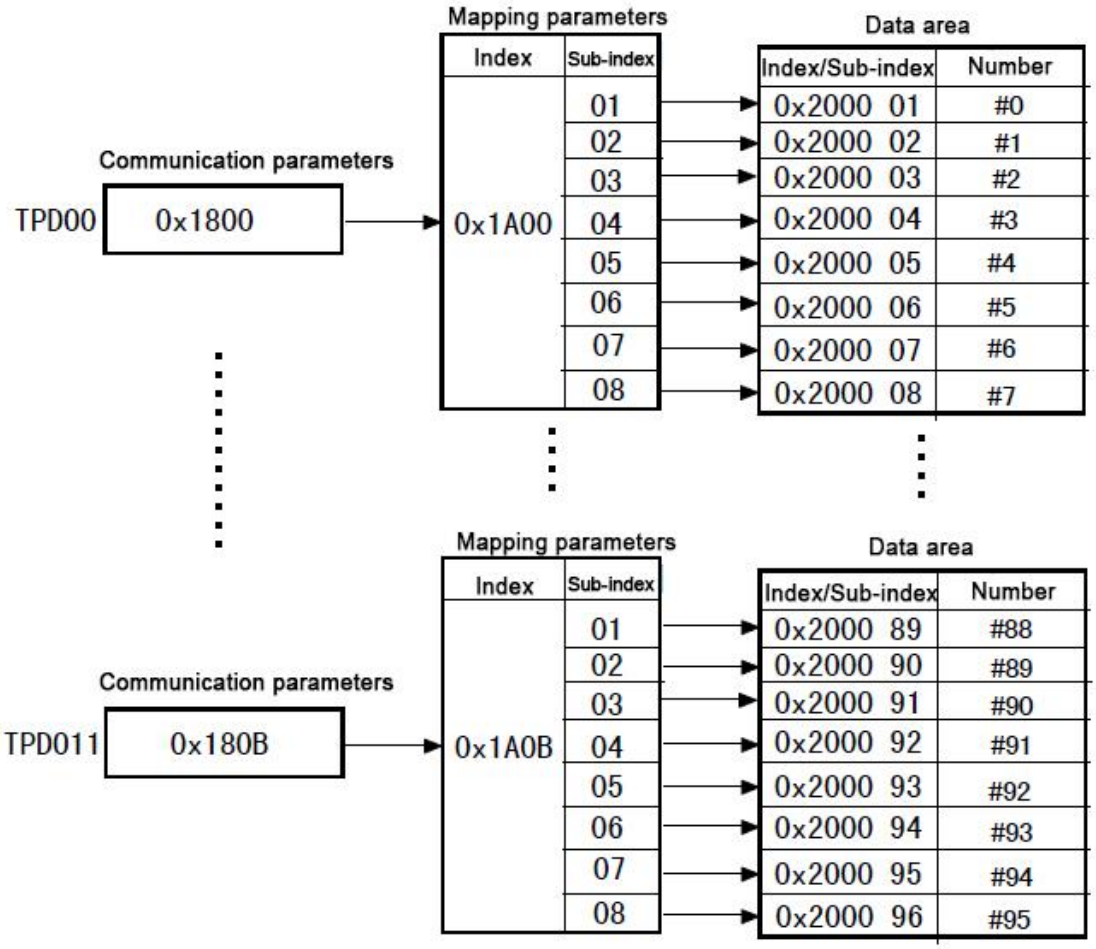


Figure 4.2 TPDO mapping relationship

Each TPDO contains corresponding communication parameters. These communication parameters determine the type of TPDO transmission and the trigger conditions for sending. Among these parameters, there are mainly three types, which are the transmission type, the inhibit time, and the event timing.

- Transmission type

The transmission type defines the TPDO transmission mode. The sub-index 2 of the communication parameters defines the object. The definition of the specific values is shown in Table 4.27.

Transmission type	Data request	Transmit PDO
0	The data (input value) is read when a SYNC message is received.	If the PDO data has been changed compared to the previous PDO, the PDO will be sent.
1—240	The data is collected and updated as the nth numbered SYNC message is received and then transmitted on the bus. The transmission type corresponds to the value n.	

241－251	Retain	
252	The data (input value) is read when a SYNC message is received.	PDO is sent via a remote frame when requested.
253	The application continuously collects and updates input data.	
254	The application defines the events that cause data requests and PDO transfers. The event caused the PDO transmission may be that the time of the event timer has expired. The event timer period is configured with sub-index 5. PDO transmissions (regardless of whether the event and event timers are configured) always start a new event timer period.	
255	The device profile defines the events that cause data requests and PDO transfers. The event caused the PDO transmission may be that the time of the event timer has expired. The event timer period is configured with sub-index 5. PDO transmissions (regardless of whether the event and event timers are configured) always start a new event timer period.	

Table 4.27 TPDO transmission type

- Inhibit Time

The definition of inhibit time prevent TPDO from sending too much and occupying a large amount of bus bandwidth to affect the bus communication. Thus defines the minimum time interval (milliseconds) that the same TPDO send PDO. When this parameter is 0, it is invalid and defined in sub-index 3 of the communication parameter.

- Event Time

The timing parameter defines the transmitting cycle time (milliseconds) of the PDO. The PDO transmission type needs to be set to 254 or 255. When this parameter is 0, it is invalid and defined in sub-index 5 of the communication parameter.

For example, suppose that the current Node ID is 0x20, the TPDO0 event time parameter is 1000, the transmission type is 254, and the data in the data input area #0~#7 is 0x18, then TPDO0 transmits data as shown in Figure 4.3.

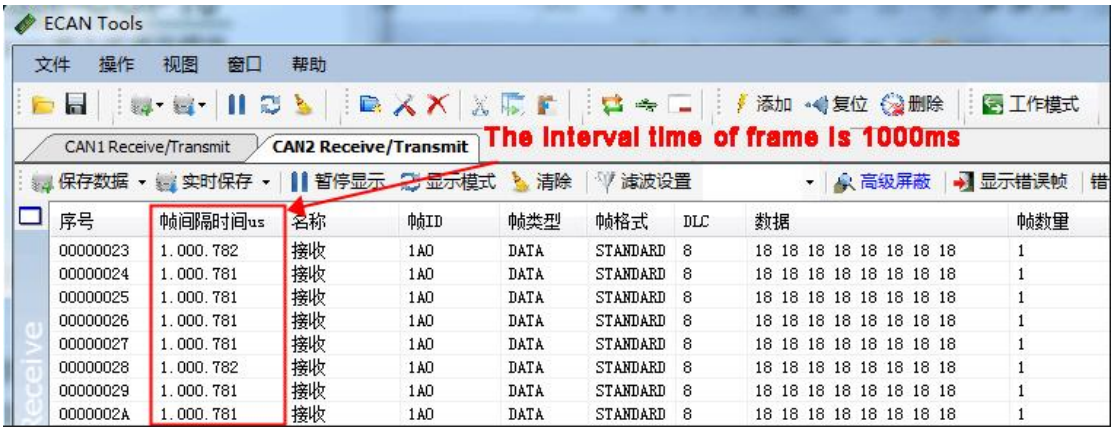


Figure 4.3 TPDO transmission data

5. Serial port operation

5.1 Serial port communication protocol

GCAN-305 communicates with users using asynchronous serial communication. The communication mode is half duplex, the communication signal is TTL level, and the communication protocol adopts a custom serial communication protocol.

Asynchronous serial data frame format

Each 1 byte is transmitted with 10 bits, 1 start bit, 8 data bits, no parity bit, 1 stop bit, baud rate 1200 ~ 115200 bps. Response mode: user device asks (main), GCAN-305 passively answers (slave). The master/slave response frame structure realizes common data communication. The data communication is initiated by the host and is called the command frame. The frame format is shown in Table 5.1. After the slave receives the command frame and responds it, it is called the response frame, as shown in Table 5.2.

1Byte	1Byte	1Byte	1Byte	nByte	1Byte
Start word 0x7E	Command code CMD	Command information (length) CMDinfo	Specific parameters SpeByte	Command data DATA	Check code CRC

Table 5.1 Command frame format

1Byte	1Byte	1Byte	1Byte	nByte	1Byte
Start word 0x7E	Command code ACK	Command information (length) ACKinfo	Specific parameters SpeByte	Command data DATA	Check code CRC

Table 5.2 Response frame format

The length of the command, response frames is: Command/response information CMDinfo/ ACKinfo (data length) + 5Byte,

- Start character of frame SOF, fixed 0x7E, one byte in length
- Command code CMD/response code ACK, generally CMD=ACK, one byte in length
- The command information CMDinfo/response information ACKinfo indicates the information length (in bytes) not including itself. CMDinfo/ACKinfo = 0 indicates no data, and CMDinfo/ACKinfo = 1 indicates that this frame contains 1 byte of data.
- The special parameter SpeByte includes the Error, AllDataSegSize, and DataSegNum information. The specific meanings are shown in Table 5.3.

Error is a specific meaning bit. In the command information, Error is a reserved bit,

generally Error = 0. In the response frame, Error is an identity bit, Error = 1, indicating that the command execution error, DATA area follows the error code. Error = 0, indicating that the request is successful, DATA area follows the response data.

AllDataSegSize and DataSegNum indicate the segment information. When the data volume exceeds 255 bytes, it needs to use multiple frames to transmit. In this case, the segment transmission mode is used, which can be divided into 7 segments, namely 7 frames.

BIT.7	BIT.6	BIT.5	BIT.4	BIT.3	BIT.2	BIT.1	BIT.0
Error	AllDataSegSize			reserved bit	DataSegNum		

Table 5.3 The definition of specific parameters（SpeByte）

AllDataSegSize indicates data size of information data, DataSegNum indicates data number of information data. When AllDataSegSize = DataSegNum, indicates that the information data is transmitted. AllDataSegSiz and DataSegNum have a minimum of 1 and cannot be 0.

This protocol will not use segment. AllDataSegSize and DataSegNum are fixed to "1".

- Command/response DATA, this part is combined with command/response code to describe the specific meaning of the data. The length is specified in CMDinfo/ ACKinfo, up to 255 bytes/frame (bytes of data area only)
- Check and CRC: checksum of command/response data, the length is one byte. Checksum is the exclusive-or value of all previous data. The following is the CRC calculation formula: CRC ＝ 0x7E^ CMD ^ CMDinfo ^ SpeByte ^DATA[0]^DATA[1]^…^DATA[n-1] or CRC ＝ 0x7E^ ACK ^ ACKinfo ^ SpeByte ^DATA[0]^DATA[1]^…^DATA[n-1]

5.2 Operation command

The user operates the device through the communication serial port of GCAN-305. The operation command is as shown in section 5.2.1. All operation commands in this section are considered to have been executed successfully and correctly. If an error occurs, its error response frame and error code are described in Section 5.3.

5.2.1 Read device information (Command code: 0x01)

Read the device information set by the user through the serial port, and is located in 0x2404 of the object dictionary. The device type is located in the object dictionary 0x1000 00. Because the device is a general device and does not use the standard device description, this parameter should be 0x00000000 according to CiA definition. It is recommended that users do not change this parameter. Operation commands and response frames are shown in Table 5.4 and Table 5.5, n indicates the data length of the

command frame. The first byte of the command data field is the operation mode, and the type of information read by the command is judged according to the byte. The following describes the operation command in details.

Note: the device information read through the serial port is not the GCAN-305 device information, is the user device-related device information.

Start byte	Command code	Command information	Specific parameters	Command data (n Byte)	CRC
0x7E	0x01	n	0x11	DAT	Check code

Table 5.4 Read device information command

Start byte	Command code	Command information	Specific parameters	Command data (n Byte)	CRC
0x7E	0x01	n	0x11/0x91	DAT	Check code

Table 5.5 Read device information response

1. Read device type (operation mode: 0x01)

The device type is located in the object dictionary 0x1000 00, the data length is 4 bytes, the upper 2 bytes are the GCAN-305 device type, and the lower 2 bytes are the user device type code (user settings).

Example: The following command reads the device type 0x0A000011, that is, the current GCAN-305 device type is 0x00 A0, and the user's device type is 0x0000.

Command：0x7E 0x01 0x01 0x11 **0x01** 0x6E
Response：0x7E 0x01 0x05 0x11 **0x01 0x11 0x00 0x0A 0x00** 0x71

Note: The use of this module is recommended not to change the device type. Because GCAN-305 does not use the standard device description, the device type is generally 0x00000000.

2. Read hardware version (operation mode: 0x02)

The hardware version information is located in the object dictionary 0x2404 01, and the data length is 4 bytes. This value is user setting information.

Example: The following command reads the hardware version information as 0x00000100.

Command：0x7E 0x01 0x01 0x11 **0x02** 0x6D
Response：0x7E 0x01 0x05 0x11 **0x02 0x00 0x01 0x00 0x00** 0x68

3. Read software version (operation mode: 0x03)

The software version information is located in the object dictionary 0x2404 02, and the

data length is 4 bytes. The value is the user setting information.

Example: The following command reads the software version information as 0x00000100.

Command：0x7E 0x01 0x01 0x11 **0x03** 0x6C
Response：0x7E 0x01 0x05 0x11 **0x03 0x00 0x01 0x00 0x00** 0x69

4. Read product code (operation mode: 0x04)

The product code is located in the object dictionary 0x2404 03, the data length is 4 bytes, the value is the user setting information.

Example: The following command reads the product code as 0x00000001

Command：0x7E 0x01 0x01 0x11 **0x04** 0x6B
Response：0x7E 0x01 0x05 0x11 **0x04 0x01 0x00 0x00 0x00** 0x6E

5. Read product revised code (operation mode: 0x05)

The product code is located in the object dictionary 0x2404 04, and the data length is 4 bytes. This value is the user setting information.

Example: The following command reads the product revision code 0x00000001

Command：0x7E 0x01 0x01 0x11 0x05 0x6A
Response：0x7E 0x01 0x05 0x11 0x05 0x01 0x00 0x00 0x00 0x6F

6. Read product serial number (operation mode: 0x06)

The product code is located in the object dictionary 0x2404 05, the data length is 4 bytes, the value is the user setting information

Example: The following command reads the product serial number 0x00000001

Command：0x7E 0x01 0x01 0x11 **0x06** 0x69
Response：0x7E 0x01 0x05 0x11 **0x06 0x01 0x00 0x00 0x00** 0x6C

7. Read product name (operation mode: 0x07)

The product code is located in the object dictionary 0x2404 06, and the data length is 12 bytes (string, the first 11 bytes are valid). This value is the user setting information.

Example: The following command reads the product name "DeviceName"

Command：0x7E 0x01 0x01 0x11 0x07 0x68
Response：0x7E 0x01 0x0D 0x11 0x07 0x44 0x65 0x76 0x69 0x63 0x65 0x4E 0x61 0x6D 0x65 0x00 0x84 0xFF

5.2.2 Write device information (Command code: 0x02)

The write device information corresponds to the read device information, and the same

information corresponds to the same object dictionary. The device type is located in the object dictionary 0x1000 00. Because the device is a general device and does not use the standard device description, this parameter should be 0x00000000 according to CiA definition. It is recommended that users do not change this parameter. Its operation command and response frame are shown in Table 5.6 and Table 5.7, n indicates the data length of the command. The first byte of the command data is the operation mode, and the type of information read by the command is judged according to the byte. The operation command will be described in details below.

Note: the device information written through the serial port does not change the GCAN-305 device information, but changes the user device-related device information.

Start byte	Command code	Command information	Specific parameters	Command data (n Byte)	CRC
0x7E	0x02	n	0x11	DAT	Check code

Table 5.6 Write device information command

Start byte	Command code	Command information	Specific parameters	Command data (n Byte)	CRC
0x7E	0x02	n	0x11	DAT	Check code

Table 5.7 Write device information response

1. Write device type (operation mode: 0x01)

The device type is located in the object dictionary 0x1000 00, the data length is 4 bytes, the upper 2 bytes are the GCAN-305 device type, and the lower 2 bytes are the user device type code (user settings).

Example: Write device type value is 0x0011

Command：0x7E 0x02 0x05 0x11 **0x01 0x11 0x00 0x00 0x00** 0x69
Response：0x7E 0x02 0x01 0x11 **0x01** 0x6D

2. Write hardware version (operation mode: 0x02)

The hardware version information is located in the object dictionary 0x2404 01, and the data length is 4 bytes. This value is user setting information.

Example: The following command writes the hardware version to 0x22222222

Command：0x7E 0x02 0x05 0x11 **0x02 0x22 0x22 0x22 0x22** 0x6A
Response：0x7E 0x02 0x01 0x11 **0x02** 0x6E

3. Write software version (operation mode: 0x03)

The software version information is located in the object dictionary 0x2404 02, and the data length is 4 bytes. The value is the user setting information.

Example: The following command writes the software version information as 0x33333333.

Command：0x7E 0x02 0x05 0x11 **0x03 0x33 0x33 0x33 0x33** 0x6B
Response：0x7E 0x02 0x01 0x11 **0x03** 0x6F

4. Write product code (operation mode: 0x04)

The product code is located in the object dictionary 0x2404 03, the data length is 4 bytes, the value is the user setting information.

Example: The following command writes the product code as 0x44444444

Command：0x7E 0x02 0x05 0x11 **0x04 0x44 0x44 0x44 0x44** 0x6C
Response：0x7E 0x02 0x01 0x11 **0x04** 0x68

5. Write product revised code (operation mode: 0x05)

The product code is located in the object dictionary 0x2404 04, and the data length is 4 bytes. This value is the user setting information.

Example: The following command writes the product revised code 0x55555555

Command：0x7E 0x02 0x05 0x11 **0x05 0x55 0x55 0x55 0x55** 0x6D
Response：0x7E 0x02 0x01 0x11 **0x05** 0x69

6. Write product serial number (operation mode: 0x06)

The product code is located in the object dictionary 0x2404 05, the data length is 4 bytes, the value is the user setting information

Example: The following command writes the product serial number 0x66666666

Command：0x7E 0x02 0x05 0x11 **0x06 0x66 0x66 0x66 0x66** 0x6E
Response：0x7E 0x02 0x01 0x11 **0x06** 0x6A

7. Write product name (operation mode: 0x07)

The product code is located in the object dictionary 0x2404 06, and the data length is 12 bytes (string, the first 10 bytes are valid). This value is the user setting information.

5.2.3 Write GCAN-305 input buffer data (Command code: 0x10)

The GCAN-305's input buffer is 8 bits wide from #0 to #95. Corresponding to the input buffer 16 bits #0 to #47 and 32 bits #0 to #23, they occupy the same memory area. If you need to operate on the #0 number of the 32-bit area, you can operate #0~#3 in the 8bit area and #0~#1 in the 16bit area to obtain the same result. And so on, only the serial port provides operations on the 8bit area.

The GCAN-305 input buffer is a write-only area that can write 96 bytes of data at most. The operation and response command are shown in Table 5.8 and Table 5.9. The first

byte of the command data indicates the current data offset in the input data area. The length of the data written is n-1.

Note: Offset + data length should not be greater than 96.

Start byte	Command code	Command information	Specific parameters	Command data (n Byte)		CRC
0x7E	0x10	n	0x11	Address offset	DAT	Check code

Table 5.8 Write buffer command

Start byte	Command code	Command information	Specific parameters	Command data (n Byte)	CRC
0x7E	0x10	n	0x11	DAT	Check code

Table 5.9 Write buffer response

Example: Eight bytes of data are written to the address (that is, the offset is 0) starting from the number #0 of 8-bit area, and the data is 0x12 0x12 0x12 0x12 0x12 0x12 0x12 0x12.

Command：0x7E 0x10 0x09 0x11 0x00 0x12 0x12 0x12 0x12 0x12 0x12 0x12 0x12 0x76

Response：0x7E 0x10 0x01 0x11 0x00 0x7E

5.2.4 Read GCAN-305 output buffer data (Command code: 0x11)

The GCAN-305's output buffer is 8 bits wide from #0 to #95. Corresponding to the output buffer 16 bits #0 to #47 and 32 bits #0 to #23, they occupy the same memory area. If you need to operate on the #0 number of the 32-bit area, you can operate #0~#3 in the 8bit area and #0~#1 in the 16bit area to obtain the same result. And so on, only the serial port provides operations on the 8bit area.

The GCAN-305 output buffer is a read-only area that can read 96 bytes of data at most. The operation and response command are shown in Table 5.10 and Table 5.11. The first byte of the command data indicates the current data offset in the output data area, and the second byte indicates the length of the data that needs to be read. The first byte in the response command data indicates the offset in the output buffer, and the returned data length is n-1.

Note: Offset + data length should not be greater than 96.

Start byte	Command code	Command information	Specific parameters	Command data (n Byte)		CRC
0x7E	0x11	n	0x11	Address offset	DAT	Check code

Table 5.10 Read buffer command

Start byte	Command code	Command information	Specific parameters	Command data (n Byte)		CRC
0x7E	0x11	n	0x11	Address offset	DAT	Check code

Table 5.11 Read buffer response

Example: Read the offset of eight bytes data starting from the number #0 in output buffer area, and the data is 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88

Command：0x7E 0x11 0x02 0x11 **0x00 0x08** 0x74
Response：0x7E 0x11 0x09 0x11 **0x00 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88** 0xFF

5.2.5 Read NodeID of GCAN-305 (Command code: 0x12)

When GCAN-305 doesn't use DIP switch or other ways to set NodeID, user can set NodeID of GCAN-305 through serial port. The set NodeID is stored in memory, and the module takes effect after reset (the setting value must be in the range of 1~127).

Note: When the setting NodeID doesn't use the DIP switch or the setting is invalid (0x00), the setting value via the serial port will take effect.

1. Write NodeID of GCAN-305 (operation mode: 0x00)

The NodeID is located in the object dictionary 0x2403 01, which represents the NodeID value set by the current module. After the setting to reset or restart, the module will take effect. The NodeID command format and response commands for the write module are shown in Table 5.12 and Table 5.13. The first byte of the command data indicates the current operation mode, and the second byte indicates the value of the NodeID to be written. The NodeID written by GCAN-305 returns a correct answer if successful (n = 0x01, the first byte of the command byte is the operating mode).

Start byte	Command code	Command information	Specific parameters	Command data (2 Byte)		CRC
0x7E	0x12	2	0x11	operation mode (0x00)	DAT (NodeID)	Check code

Table 5.12 Write NodeID command

Start byte	Command code	Command information	Specific parameters	Command data (n Byte)	CRC
0x7E	0x12	n	0x11	DAT	Check code

Table 5.13 Write NodeID response

Example: Set the NodeID of the GCAN-305 module to 0x03, and its commands and responses are as follows,

Command：0x7E 0x12 0x02 0x11 **0x00 0x03** 0x5F
Response：0x7E 0x12 0x01 0x11 **0x00** 0x7C

2. Read NodeID of GCAN-305 (operation mode: 0x01)

The NodeID read by the read node NodeID is the NodeID that the current node is using, and is located in a different object dictionary from the write NodeID. The read NodeID is located in the object dictionary 0x2400 00, so the written NodeID and the read back NodeID may be different. Its command mode is shown in Table 5.14 and Table 5.15.

Start byte	Command code	Command information	Specific parameters	Command data (1 Byte)	CRC
0x7E	0x12	0x01	0x11	operation mode (0x01)	Check code

Table 5.14 Read NodeID command

Start byte	Command code	Command information	Specific parameters	Command data (2 Byte)		CRC
0x7E	0x12	2	0x11	operation mode (0x00)	DAT (NodeID)	Check code

Table 5.15 Read NodeID response

Note: The read NodeID value is the NodeID that the current module is using, and it is not necessarily the same as the written NodeID.

Example: Read the current NodeID of GCAN-305 and its value is 0x03

Command：0x7E 0x12 0x01 0x11 **0x01** 0x7D
Response：0x7E 0x12 0x02 0x11 **0x01 0x03** 0x5E

5.2.6 Read baud rate index value of GCAN-305 (Command code: 0x13)

Similar to setting the NodeID of the GCAN-305, when the CAN communication baud rate is not set by the DIP switch or other methods, the user can set the GCAN-305 baud rate index value through the serial port. The value is stored in memory and will take effect when the GCAN-305 is powered on or reset. The range of this value is 0~8, and the other values are invalid.

1. Write baud rate index value of GCAN-305 (Operation mode: 0x00)

The baud rate index value is in the object dictionary 0x2403 02, and the setting value is valid from 0 to 8, otherwise, an error code is returned. The command format is shown in Table 5.16 and Table 5.17. When the set command is executed correctly (n = 0x01, the first byte of the command byte is the operation mode). The correspondence between the baud rate index value and the actual value is shown in Table 3.4.

Start byte	Command code	Command information	Specific parameters	Command data (2 Byte)		CRC
0x7E	0x13	2	0x11	operation mode (0x00)	DAT (Baud rate index)	Check code

Table 5.16 Write node baud rate index command

Start byte	Command code	Command information	Specific parameters	Command data (n Byte)	CRC
0x7E	0x13	n	0x11	DAT	Check code

Table 5.17 Write node baud rate index command response

Example: Set the baud rate index of GCAN-305 module to 0x04. According to Table 3.4, set the CAN baud rate index to 0x04, and the CAN communication baud rate is 125Kbps. Command and response frames are as follows.

Command：0x7E 0x13 0x02 0x11 **0x00 0x04** 0x7A

Response：0x7E 0x13 0x01 0x11 **0x00** 0x7D

2. Read baud rate index value of GCAN-305 (Operation mode: 0x01)

The read baud rate index is in the object dictionary 0x2401 00. Similar to the read NodeID, the read baud rate index value is not necessarily the same as the written value. Because the read baud rate index value is the value that the current module is using, it will be the same after the written value is enabled (module reset or restart). Its command frame format is shown in Table 5.18 and Table 5.19.

Start byte	Command code	Command information	Specific parameters	Command data (n Byte)	CRC
0x7E	0x13	1	0x11	operation mode (0x00)	Check code

Table 5.18 Read node baud rate index command

Start byte	Command code	Command information	Specific parameters	Command data (2 Byte)		CRC
0x7E	0x13	2	0x11	operation mode (0x01)	DAT (Baud rate index)	Check code

Table 5.19 Read node baud rate index command response

Example: Assuming that the baud rate of the current module is 125 Kbps, the read index value is 0x04, and its commands and responses are as follows:

Command：0x7E 0x13 0x01 0x11 **0x01** 0x7C
Response：0x7E 0x13 0x02 0x11 **0x01 0x04** 0x7B

5.2.7 Read and write CAN controller timing parameters of GCAN-305 (Command code: 0x14)

The real baud rate is a 32-bit value including CAN controller timing and frequency division parameter. Unlike the baud rate index value, a standard baud rate table has been fixed in the GCAN-305 module. Therefore, the CAN communication baud rate can be set through the index. If the user needs to use baud rate of the table that hasn't been provided, the user can set any baud rate through command code 0x14. Please contact us for the specific baud rate value.

Note: If the DIP switch setting and baud rate index value are invalid, the baud rate timing parameter set by the user is valid.

1. Write baud rate value (Operation mode: 0x00)

The real baud rate is located in the object dictionary 0x2403 03. It is enabled when the DIP switch setting and the baud rate index value are invalid (0x00000000 and 0xFFFFFFFF are invalid). The command format is shown in Table 5.20 and Table 5.21. If the command is executed successfully, the correct answer is returned (n=0x01, the first byte of the command byte is the operation mode).

Start byte	Command code	Command information	Specific parameters	Command data (5 Byte)		CRC
0x7E	0x14	5	0x11	operation mode (0x00)	DAT (Baud rate index)	Check code

Table 5.20 Write node baud rate command

Start byte	Command code	Command information	Specific parameters	Command data (n Byte)	CRC
0x7E	0x14	n	0x11	DAT	Check code

Table 5.21 Write node baud rate command response

Example: Write a baud rate of 125Kbps to the GCAN-305 module. The commands and responses are as follows

Command：0x7E 0x14 0x05 0x11 **0x00 0x0B 0x00 0x2B 0x00** 0x5E
Response：0x7E 0x14 0x01 0x11 **0x00** 0x7A

2. Read baud rate value (Operation mode: 0x01)

The read baud rate value and the write baud rate value are in object dictionary 0x2403 03. The command modes are shown in Table 5.22 and Table 5.23.

Start byte	Command code	Command information	Specific parameters	Command data (1 Byte)	CRC
0x7E	0x14	n(0x01)	0x11	DAT (operation mode)	Check

					code

Table 5.22 Write node baud rate command

Start byte	Command code	Command information	Specific parameters	Command data (5 Byte)		CRC
0x7E	0x14	5	0x11	operation mode (0x01)	DAT (Baud rate)	Check code

Table 5.23 Write node baud rate command response

Example: Assuming that the current baud rate is 0x2B000B, the read command and response command are as follows

Command：0x7E 0x14 0x01 0x11 **0x01** 0x7B
Response：0x7E 0x14 0x05 0x11 **0x01 0x0B 0x00 0x2B 0x00** 0x5F

5.2.8 Transmit emergency code (Command code: 0x15)

When a certain error occurs in the device, it can be sent to the CAN bus through CANopen to inform the CANopen master device that the current equipment has an error. The error code consists of 5 bytes and is defined by the user. Its format is shown in Table 5.24 and Table 5.25. When the device executes correctly, the correct response is answered (n=0x01, the first byte of the command byte is the operation mode).

Start byte	Command code	Command information	Specific parameters	Command data (5 Byte)		CRC
0x7E	0x15	5	0x11	operation mode (0x00)	DAT (5 bytes)	Check code

Table 5.24 Transmit emergency error code command

Start byte	Command code	Command information	Specific parameters	Command data (n Byte)	CRC
0x7E	0x15	n	0x11	DAT	Check code

Table 5.25 Transmit emergency error code command response

Example: Assuming that the current device has an error and the error code is defined as 0x9988776655, the command format is as follows.

Command：0x7E 0x15 0x06 0x11 **0x00 0x55 0x66 0x77 0x88 0x99** 0x29
Response：0x7E 0x15 0x01 0x11 **0x00** 0x7B

5.2.9 Read the status of the current module (Command code: 0x16)

The user can read the status of the current module through this command. The correspondence between the read value and the current status is shown in Table 4.8. Its command format is shown in Table 5.26 and Table 5.27.

Start	Command	Command	Specific	Command data	CRC

byte	code	information	parameters	(1 Byte)	
0x7E	0x15	1	0x11	operation code (0x01)	Check code

Table 5.26 The current GCAN-305 status

Start byte	Command code	Command information	Specific parameters	Command data (2 Byte)		CRC
0x7E	0x15	2	0x11	operation mode (0x01)	DAT (Stays value)	Check code

Table 5.27 The current GCAN-305 status response

Example: Assuming the current GCAN-305 module is in operation, the read status value is 0x05

Command：0x7E 0x16 0x01 0x11 **0x01** 0x79
Response：0x7E 0x16 0x02 0x11 **0x01 0x05** 0x7F

5.2.10 Start the node into operation (Command code: 0x17)

Users can use this command to make all slave devices in the CANopen network enter the operating status, including the module also enter the operating status. Its command format is shown in Table 5.28 and Table 5.29.

Note: Use this command with caution in the case of master management in the network.

Start byte	Command code	Command information	Specific parameters	Command data (1 Byte)	CRC
0x7E	0x17	1	0x11	operation code (0x00)	Check code

Table 5.28 Enable slaves to enter operating status

Start byte	Command code	Command information	Specific parameters	Command data (1 Byte)	CRC
0x7E	0x17	1	0x11	operation code (0x00)	Check code

Table 5.29 Enable slaves to enter operating status response

Example: Enable the current CANopen network to enter the operating status, its commands and responses are as follows

Command：0x7E 0x17 0x01 0x11 **0x00** 0x79
Response：0x7E 0x17 0x01 0x11 **0x00** 0x79

5.2.11 Change the serial port baud rate (Command code: 0x18)

The default baud rate of the communication serial port is 115200bps. The setting range is shown in Table 5.30. Change the baud rate of the serial port according to the demand, at the same time the baud rate index value is saved until it is changed again (Set the baud rate value to use the current baud rate to communicate, after the changed and delayed for

a while, and then use the new setting Baud rate for communication). It is forbidden to frequently change the communication baud rate, which may cause unsuccessful communication or data loss.

Index value	Baud rate(bps)
0	1200
1	2400
2	4800
3	9600
4	19200
5	38400
6	57600
7	115200

Table 5.30 Serial port baud rate and index

Its command format is shown in Table 5.31 and Table 5.32.

Start byte	Command code	Command information	Specific parameters	Command data (2 Byte)		CRC
0x7E	0x18	2	0x11	operation code (0x00)	baud rate index	Check code

Table 5.31 Set the serial port baud rate command

Start byte	Command code	Command information	Specific parameters	Command data (1 Byte)	CRC
0x7E	0x18	1	0x11	operation code (0x00)	Check code

Table 5.32 Serial port baud rate response

Example: If the baud rate of the current module is set to 115200 bps, the index value is 7, and the command frame and response frame are as follows.

Command：0x7E 0x18 0x02 0x11 **0x00 0x07** 0x72
Response：0x7E 0x18 0x01 0x11 **0x00** 0x76

Note: It is recommended to change the baud rate and delay about 20ms, and then use the new baud rate to communicate.

5.2.12 Time_ Stamp (Command code: 0x19)

In order to synchronize the time of all nodes in the CANopen network, the CANopen host sends a synchronization time identification object to the network. As a standard CANopen slave module, GCAN-305 can timely obtain the synchronization time in the network. The user can obtain the network time through the communication serial port. The time format follows the definition of DS301 V4.02.

Note: the correct network time can only be read out after the GCAN-305's network time has been updated. Otherwise, an error code (error code was not updated) can be returned.

This time can only be read once, then the value is invalid and return an error code.

The command format for reading time is shown in Table 5.33 and Table 5.34.

Start byte	Command code	Command information	Specific parameters	Command data (1 Byte)	CRC
0x7E	0x19	1	0x11	operation code (0x01)	Check code

Table 5.33 Read the current network time command

Start byte	Command code	Command information	Specific parameters	Command data (7 Byte)		CRC
0x7E	0x19	7	0x11	operation code (0x01)	Time Of Day	Check code

Table 5.34 Read the current network time response

Example: When the data is read, the network time of the current module has been updated. The time is the relative time starting from January 1, 1984. The returned time is 0x2552 days 0x0237777C milliseconds relative to January 1, 1984, and the current time is calculated as February 27, 2010, and 10:19:49.

Command：0x7E 0x19 0x01 0x11 0x01 0x76
Response：0x7E 0x19 0x07 0x11 0x01 0x7C 0x77 0x37 0x02 0x52 0x25 0x39
When the network time is not received or not updated, the response command code is as follows,
Error response：0x7E 0x19 0x02 0x91 0x01 0x08 0xFD

5.3 GCAN-305 serial port error response

In all serial operation commands, when the parameters of the command are incorrect or other errors occur during communication, GCAN-305 will return an error code (the highest bit of the specified parameter is 1, indicating the current error response frame), error code frame format is shown in Table 5.35. The ACK is the same as the operation command code, and the response operation mode code is the same as the current command operation mode code. The error code indicates the error category of the current operation. The error code table is shown in Table 5.36.

Start byte	Command code	Command information (Length)	Specific parameters	Response data (2 Byte)		CRC
0x7E	ACK=CMD	0x02	0x91	operation mode	error code	Check code

Table 5.35 GCAN-305 command executes error response

Error code (data)	Instructions	Remark
0x01	Command error	Don't support

		this command
0x02	Data length error	Beyond writable area
0x03	Address error	Out of address range
0x04	An error occurred while operating the protocol stack	-
0x05	Error storing data	-
0x06	Data value is out of range	-
0x07	Operation mode is not supported	-
0x08	Network time is not updated and is currently unavailable	-

Table 5.36 Error code table

Example: Assume that the baud rate index of the GCAN-305 is written to 10. Because the baud rate index range is 0~8, the value is invalid and must be error during execution. The error code is 0x06 (the data value is out of range). The command frame and response frame are as follows,

Command：0x7E 0x13 0x02 0x11 **0x00 0x10** 0x6E
Response：0x7E 0x13 0x02 0x91 **0x00 0x06** 0xF8

Sales and service

Shenyang Guangcheng Technology Co., Ltd.

Address: Industrial Design Center, No. 42 Chongshan Middle Road, Huanggu District, Shenyang City, Liaoning Province.

QQ: 2881884588

E-mail: 2881884588@qq.com

Tel: +86-024-31230060

Website: www1.gcanbox.com

Sales and service Tel: +86-18309815706

After - sales service telephone Number: +86-13840170070

WeChat Number: 13840170070