MAD-Cyphal

CAN Communication Protocol

Version upgraded record

Version	Date	Technician	Modify Content
V1.0.0	2023/04/19		first edition
V1.0.0	2023/05/15		Added the description of register 05H to the content
V1.0.0	2023/08/17		Revised the BIT13, encoder setting, under the term of 4.3.4

Table of contents

1. Overview 2
2. Definition of technical terms 2
3. Introduction of protocol 2
3.1. Baud rate and sampling point 2
3.2. CAN-ID divided as per Cyphal standard 2
3.2.1. CAN-ID split 2
This protocol is based on the Cyphal standard, and only uses the extended format data frame (29bit CAN-ID)for communication. The Cyphal protocol divides the CAN-ID as follows. 2
3.3. Cyphal protocol data transmission specification 3
3.3.1. Single-frame transmission 4
3.3.2. Multi-frame transmission 4
4. Cyphal protocol application 5
4.1. ESC function introduction 5
4.2. Supplier ID field usage Plan 5
4.3. Broadcast data type description 5
4.3.1. Command control.6144 (3 byte) 5
4.3.2. Device ID address setting 6
4.3.3. Throttle transmission.[6152~6159] (7 byte) 6
4.3.4. Information upload 6160 (6 byte) 7
4.3.5. Information upload 6161（7 byte） 7
4.3.6. Heartbeat packet.7509standard（6 byte） 7
4.4. Service data type description 8
4.4.1. Register read and write.256 8
4.4.2. Get node information.430standard 9
4.4.3. Node command control 435standard 9
4.5. Part of the function description of the register table 10
4.5.1. CAN baud rate setting 10
5. References 10
6. Appendix 1 - Reference Procedures 11
7. Appendix 2 - Register List 12

1. Overview

This protocol is based on the CAN 2.0B standard. It uses extended format data frames for data interaction, and the Cyphal protocol is used to manage CAN transmission to achieve single-point/overall data transmission. For example, register read and write, message upload, data fast write or command control, etc.

Remark: extended format data frames belongs to traditional CAN, which can transmit up to 8 data.

2. Definition of technical terms

Table 1 Terminology

CAN	Controller Area Network，to control LAN communication protocol
CAN-ID	Controller Area Network Identifier
Cyphal	CAN application protocol. Please refer to the official website https://opencyphal.org/
Node-ID	Device number on the CAN bus

Remark：

1). Except for CRC which adopts big-endian format, all other data types use little-endian format.

2). The ID of the ESC starts from 0x10. The broadcast receiving range of the ESC is nodes 0~0x0f, 126 and 127. The broadcast messages of other nodes are all ignored.

3). On the CAN bus, all connected devices can communicate with each other

3. Introduction of protocol

3.1. Baud rate and sampling point

Table 2 Baud rate and reference sampling point

Baud rate	Bit time	Reference sampling point	Recommended number of ESCs
1000K	1.00us	75.0%	ESC's Qty.≦16
800K	1.25us	80.0%	ESC's Qty.≦16
500K(default)	**2.00us**	**87.5%**	ESC's Qty.≦8
250K	4.00us	87.5%	ESC's Qty.≦4

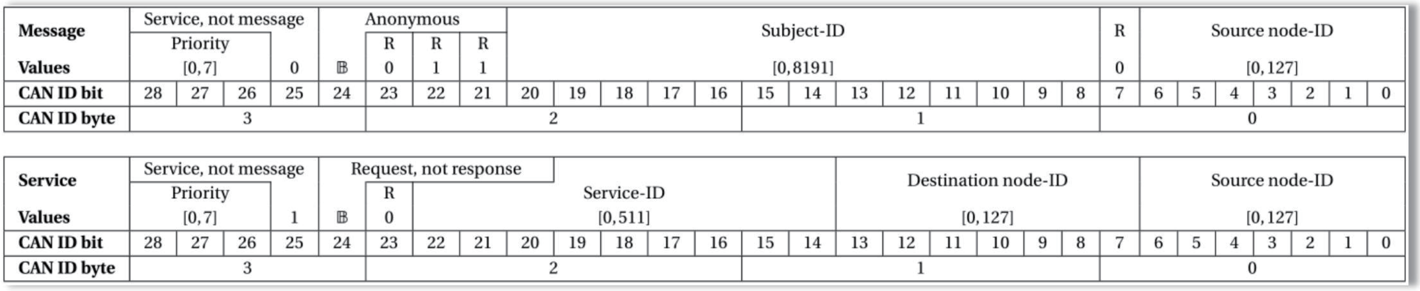
Remark:

1). It is recommended that the bus load rate not exceed 70%. The number of ESCs corresponding to different baud rates in the above table is only for reference. The actual situation also needs to be determined according to the refresh rate of the throttle configured by the user and the upload rate of the operating parameters.

3.2. CAN-ID divided as per Cyphal standard

3.2.1. CAN-ID split

This protocol is based on the Cyphal standard, and only uses the extended format data frame (29bit CAN-ID)for communication. The Cyphal protocol divides the CAN-ID as follows.



<table>
<tr><td><b>Message</b></td><td colspan="4">Service, not message</td><td></td><td colspan="3">Anonymous</td><td colspan="13">Subject-ID</td><td>R</td><td colspan="7">Source node-ID</td></tr>
<tr><td></td><td colspan="3">Priority</td><td></td><td></td><td>R</td><td>R</td><td>R</td><td colspan="13"></td><td></td><td colspan="7"></td></tr>
<tr><td><b>Values</b></td><td colspan="3">[0,7]</td><td>0</td><td>𝔹</td><td>0</td><td>1</td><td>1</td><td colspan="13">[0,8191]</td><td>0</td><td colspan="7">[0,127]</td></tr>
<tr><td><b>CAN ID bit</b></td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td><b>CAN ID byte</b></td><td colspan="5">3</td><td colspan="8">2</td><td colspan="8">1</td><td colspan="8">0</td></tr>
</table>

<table>
<tr><td><b>Service</b></td><td colspan="4">Service, not message</td><td></td><td>Request, not response</td><td colspan="9">Service-ID</td><td colspan="7">Destination node-ID</td><td colspan="7">Source node-ID</td></tr>
<tr><td></td><td colspan="3">Priority</td><td></td><td></td><td>R</td><td colspan="9"></td><td colspan="7"></td><td colspan="7"></td></tr>
<tr><td><b>Values</b></td><td colspan="3">[0,7]</td><td>1</td><td>𝔹</td><td>0</td><td colspan="9">[0,511]</td><td colspan="7">[0,127]</td><td colspan="7">[0,127]</td></tr>
<tr><td><b>CAN ID bit</b></td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td><b>CAN ID byte</b></td><td colspan="5">3</td><td colspan="8">2</td><td colspan="8">1</td><td colspan="8">0</td></tr>
</table>

Table 3 CAN ID bit fifields

Bit field definition	Width	Description	
Priority	3	According to CAN2.0B, the smaller the CAN-ID, the higher the priority when the bus competes for transmission, and the 3-bit width forms 8 priorities. highest priority: 0 Lowest priority: 7 Broadcast: Multi-frame transmission Priority is the same Service: The priority of multi-frame transmission is the same, and both the requester and the responder have the same requirements.	0 = Exceptional 1 = Immediate 2 = Fast 3 = High 4 = Nominal 5 = Low 6 = Slow 7 = Optional
Service,not message	1	Service frame ID	0: non-service frame 1: service frame
Anonymous	1	Anonymous transmission frame identification, anonymous transmission is not used temporarily, and will not be described later	0: broadcast message frame 1: Anonymous message frame, mainly used for plug-and-play identification management
Subject-ID	13	Broadcast frame message type identification, range [0~8191]	[0000, 6143]: spare/other [6144, 7167]: user defined area [7168, 8191]: standard area
Source node-ID	6	Currently, the node-ID number of the sender of the CAN frame (0~127)	
Request,not response	1	Data request and response identification of the service frame	0: Response back to service data request 1: service data request
Service-ID	9	service frame message type identification，range [0~511]	[000, 255]: spare/other [256, 383]: user defined area [384, 511]: standard area
Destination node-ID	7	Currently, the receiver node-ID number of the CAN frame (0~127)	
R		R = reserve, currently not used, reserved as a specific value (as shown in the CAN-ID split diagram)	

3.3. Cyphal protocol data transmission specification

The Cyphal protocol stipulates that a CAN data segment of a frame is divided into a Transfer payload segment and a Tail byte segment. Since the CAN2.0B protocol stipulates that a maximum of 8 bytes can be transmitted at a time, in the Cyphal protocol, the last control byte Tail byte is removed, and then a maximum of 7 effective bytes can be transmitted.

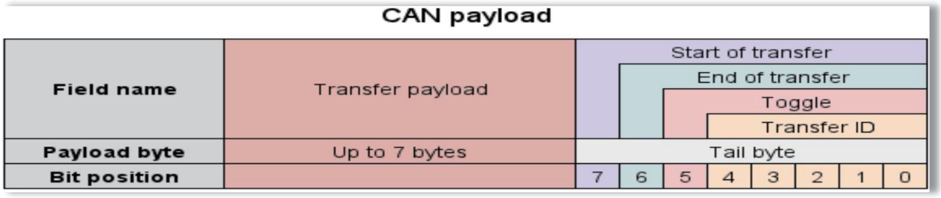
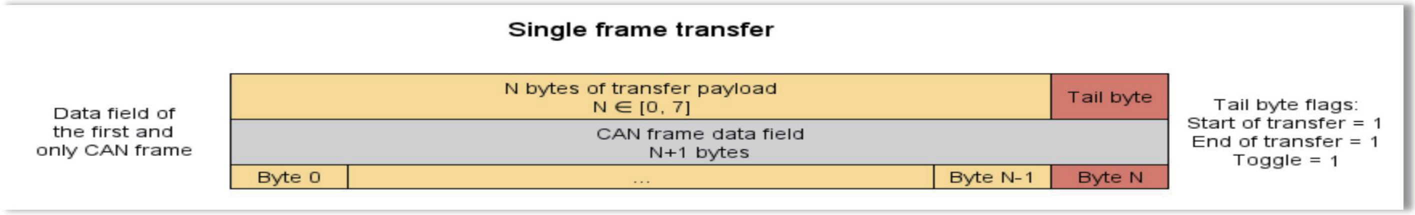


Table 4 tail byte split description

Tail byte domain name	Single-frame transfer	Multi-frame transmission
Start of transfer	The value is always 1	The first frame is 1, the other frames are 0
End of transfer	The value is always 1	Other frames are 0, the last frame is 1
Toggle	The value is always 1	The 1st frame is 1, the 2nd frame is 0, the 3rd frame is restored to 1, and the cycle is repeated between 1 and 0 (1, 0, 1, 0...) so-call flip bit, until the last frame
Transfer ID	For the same Type ID, every time a complete data packet is transmitted, the Transfer ID will increase by 1, and it will change cyclically from 0 to 31 Note: Multi-frame transmission only counts as one data packet from the beginning to the end! Each Type ID has its own Transfer ID!	

3.3.1. Single-frame transmission

If the information transmitted at one time does not exceed 7 bytes, one frame of CAN data packet can be used to complete the transmission, which is called single frame transmission.

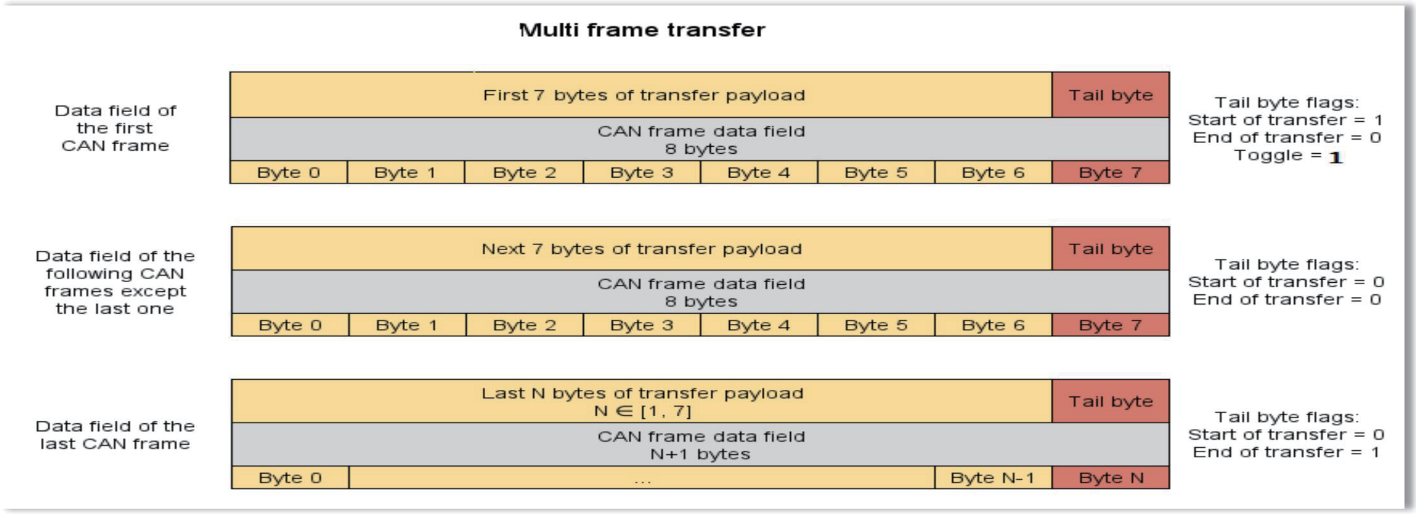


3.3.2. Multi-frame transmission

When the transmitted data exceeds 7 bytes, it should be converted to multi-frame CAN data packet transmission. The need for multi-frame transmission introduces two new concepts.

- CRC and flip bit (the flip bit is involved in the above table 4)

If there are M **(M>7)** bytes for multi-frame transmission, perform 16-bit CRC calculation on the M data, put the two-byte CRC after M bytes, and form N = M+2 bytes data packets, and then pack/unpack the multi-frame transmission protocol as shown in the figure below.



Remark：

1). The Cyphal protocol suggests that single-frame transmission has higher data throughput and lower latency than multi-frame transmission. Therefore, single-frame transmission is given priority in design.

2). CRC calculation method (C program) is placed in Appendix 1

4. Cyphal protocol application

4.1. ESC function introduction

- ✓ Send a heartbeat packet (including node status and other messages) about every 1s after power-on
- ✓ Save parameters/emergency stop/factory settings/upgrade control/node shutdown/node restart
- ✓ Register parameter reading and writing
- ✓ Data message upload
- ✓ . . .

4.2. Supplier ID field usage Plan

The Cyphal protocol stipulates that the broadcast frame provider can customize the ID range [6144~7167], a total of 1023 user data channels

The Cyphal protocol stipulates that the service frame provider can customize the ID range [256~383], a total of 127 user data channels

Table 5 Supplier ID field planning table

Frame type	Function	ID field	ID segment length	Function Description
Broadcast	overall command control	[6144~6151]	8	All devices connected to the bus respond uniformly
	throttle refresh	[6152~6159]	8	Throttle transmission, throttle transmission adopts single frame mode
	information upload	[6160~6167]	8	The components connected to the node, its voltage/current/rotational speed and other data upload
	heartbeat packet	[7509]$_{standard}$	1	After connecting to the node, upload the heartbeat status information regularly
Serve	Register read and write	[256]	1	Register reading and writing, all parameters of the ESC can be read and written on this channel
	node information	[430]$_{Standard}$	1	Used to view the version number of the ESC and the unique code information of the ESC
	Service Order Control	[435]$_{Standard}$	1	ESC command control, save parameters, reset...

Note: The "Standard" in the subscript refers to the standard fixed instruction adopted. Those without a subscript are manufacturer-defined

4.3. Broadcast data type description

4.3.1. Command control.6144 (3 byte)

Frame type	SUB-ID	command	NodeId	reserve, fill0
broadcast frame	6144	Payload[0]	[1]	[2]

Remark: Priority = Fast

➢ command： ✧ 0: Disable all information uploads ✧ 1: Disable all information uploads, excluding heartbeat packets ✧ 10: Manually trigger a heartbeat packet (prohibit all information upload invalid) ✧ 100:Enable ESC automatic upload ✧ FEH: All ESCs restart	➢ Node Id： >127: All ESCs respond Others: Corresponding ESC response

4.3.2. Device ID address setting

frame type	SUB-ID	cmd	NodeId
broadcast frame	6145	Payload[0]	Payload[1]

Remark:Priority = Nominal

➢ command： ✧ 0: Set the new address of the node (it can only be set when the ESC is not working. After the setting is successful, it will restart and use the set address). ✧ 1: Cancel the address setting, if it has an address encoder, use the address encoder value, the parameter NodeId is invalid	➢ NodeId： Set a new ID address, value range [0x10~0x10+31]

4.3.3. Throttle transmission.[6152~6159] (7 byte)

Throttle data transmission frame (the first group of throttle, including 4 throttle data, transmitted to 1~4 axis ESC)

Frame type	SUB-ID	Throttle data
broadcast frame	6152	Payload[0,6]

Throttle data transmission frame (the second group of throttle, including 4 throttle data, transmitted to 5~8 axis ESC)

Frame type	SUB-ID	Throttle data
broadcast frame	6153	Payload[0,6]

Throttle data transmission frame (the third group of throttle, including 4 throttle data, transmitted to the 9~12 axis ESC)

Frame type	SUB-ID	Throttle data
broadcast frame	6154	Payload[0,6]

....By analogy, 8 groups of throttles can be sent

➢ Remark: Priority = High

1) The nodeId of the throttle sender must be less than 0x10, or equal to 126 or 127, and the others will not be recognized

2) The number of throttles sent depends on the user's usage, but each set of throttles is defined as 7 throttle data.

3) Each set of throttle occupies 14 bits, that is, int14, and the highest bit is the sign bit. At present, the throttle value can only be greater than or equal to 0, and if it is less than 0, it will be discarded. If the transmitted throttle value has been out of range, it will report a throttle loss fault. Since only 7 bytes (56bit) can be transmitted at most each time, we give the method of parsing 56bit data into 4 pieces of 14bit throttle data as follows

Throttle to be sent HEX [0x123, 0x234, 0x345, 0x456]

Delete the upper two bits of each 16-bit throttle

Binary throttle（BIN） [~~00~~000001_00100011, ~~00~~000010_00110100,~~00~~000011_01000101,~~00~~000100_01010110]

Split the upper 6 digits of the last throttle into 00,01,00 and insert them into the upper two digits of the first 3 throttles.

Binary throttle（BIN） [00000001_00100011, 01000010_00110100,00000011_01000101,~~00000100~~_01010110]

Split the throttle by 8 bits to get 7 bytes of CAN format data,

Binary throttle（BIN） [00100011,00000001,00110100,01000010,01000101,00000011,01010110]

CAN data（HEX） [0x23, 0x01, 0x34, 0x42, 0x45, 0x03, 0x56]

See Appendix 1 for splitting the C program

void x_MakeThrot(uint16_t *throt,uint8_t *throtOut)

4) The minimum refresh cycle of the CAN throttle is 1ms, and the adjustment frequency of 1Khz is supported, and the throttle value ranges from 0 to 2048

5) CAN throttle

For throttle package, please refer to the appendix

4.3.4. Information upload 6160 (6 byte)

Frame type	SUB-ID	Electrical speed	bus current	Operating status
broadcast	6160	Payload[0,1]	[2,3]	[4,5]

➢ Remark: Priority = Low

✧ Electrical speed: data type/uint16_t, value range/0 ~ 65535, unit/0.1Hz

✧ Bus current: data type/int16_t, value range/-32768 ~ +32767, unit/0.1A,

✧ Running Status: The current running status of the system

➢ The information upload cycle defaults to 25ms, and the automatic upload is turned on when the machine is turned on, and the upload can be turned off by calling the overall command control .6144.

Operating status table

Bit	Bit function	Bit description	Bit	Bit function	Bit description
BIT0	overvoltage	0: normal, 1: overvoltage	BIT8	Stall	0: normal, 1: stalled
BIT1	undervoltage	0: normal, 1: undervoltage	BIT9	MOS open circuit	0: normal, 1: open circuit
BIT2	overcurrent	0: normal, 1: overcurrent	BIT10	MOS short circuit	0: normal, 1: short circuit
BIT3	Throttle signal source	0:PWM，1:CAN	BIT11	motor over temperature	0: normal, 1: over temperature
BIT4	throttle lost	0: normal, 1: lost	BIT12	Abnormal current sampling	0: normal, 1: abnormal
BIT5	Throttle not reset to 0	0: reset to zero, 1: not reset to zero	BIT13	Encoder setting	0: soft setting, 1: code disc
BIT6	MOS over temperature	0: normal, 1: over temperature	BIT14	Motor three-phase line status	0: normal, 1: short circuit
BIT7	capacitance over temperature	0: normal, 1: over temperature	BIT15	motor running status	0: stop, 1: running

4.3.5. Information upload 6161（7 byte）

Frame type	SUB-ID	output throttle	bus voltage	Temperature		
				MOS	Capacitance	Motor
broadcast	6161	Payload[0,1]	[2,3]	[4]	[5]	[6]

➢ Remark:Priority = Low

✧Output throttle: The throttle value received by the current ESC

✧Bus voltage: data type/int16_t, value range/-32768 ~ +32767, unit/0.1V,

✧Temperature: temperature value = transmission data -40, for example, transmission temperature data is 23, then the corresponding temperature is 23-40 = -17, unit/℃

➢ The information upload cycle defaults to 50ms, and the automatic upload is turned on when the device is turned on, and the upload can be turned off by calling the global command control .6144.

4.3.6. Heartbeat packet.7509standard（6 byte）

Frame type	SUB-ID	Power-on time	Node health status	Node current mode	User-defined
broadcast	7509	Payload[0,3]	[4]	[5]	[6]

➢ Remark: Priority = Nominal. The heartbeat packet is the standard Cyphal protocol

✧ Power-on time: record the time from power-on to the present, unit/sec

Health status		Current mode		User defined (reserved)
:0, normal mode		:0, operating mode		
:1, System parameter failure		:1, initialization mode		
:2, component major failure		:2, sensor calibration mode		
:3, system serious failure		:3, Firmware update mode		

4.4. Service data type description

4.4.1. Register read and write.256

Request : Type A, read (2 byte)

Frame type	SER-ID	Operation command	Register index
service → request	256	Payload[0]	[1]

Request : Type B, write**(4 byte)**

Frame type	SER-ID	Operation command	Register index	Register value
service → request	256	Payload[0]	[1]	[2,3]

➢ Remark: Priority >Fast

✧Operation command

Value = 0	read register value
Value = 1	Read all parameters of the register
Value = 2	write register value

✧Index, the operation index address of the register.

Response: Type 1, other errors **(2byte)**

Frame type	SER-ID	Operating state	Index
service ← response	256	Payload[0]	[1]

Response: Type 2, return register value **(4byte)**

Frame type	SER-ID	Operating state	Index	Register value
service ← response	256	Payload[0]	[1]	[2,3]

Response: Type 3, returns all parameters of the register **(23byte)**

Frame type	SER-ID	Operating state	Index	Register value	Name	Defaults	Lower limit	Upper limit	Properties
service ← response	256	Payload[0]	[1]	[2,3]	[4,15]	[16,17]	[18,19]	[20,21]	[22]

Remark:

✧Operating state description

Operating return value	Operation status return value description	return frame type
Value = 10H	Successful operation	Type 2 or Type 3
Value = 11H	Register index does not exist	Type 1, the return index is the maximum index supported by the register
Value = 12H	Operation attribute error, such as read-only register write value or unreadable and unwritable	Type 1
Value = 13H	The written value is out of range, for example, the value written to the register exceeds the upper and lower limits	Type 2

✧ Register value: the current value of the register

✧ Name: The corresponding 12-bit register name, the characters used in the register name are Cyphal protocol standard characters

✧ Default value: If it is a power-off memory register, it represents the value of restoring the factory settings, and others represent the value of the initial power-on

✧ Lower limit: the minimum value that can be set by the register value

✧ Upper limit: the maximum value that can be set by the register value

✧ Properties: Properties include

Bit0	Bit1	Bit2	Bit3~7
0: readable, 1: unreadable	0: not writable, 1: writable	0: reset parameters are lost, 1: reset parameters are not lost	reserve

4.4.2. Get node information.430standard

Request :(0 byte)

Frame type	SER-ID
service → request	430

Response:(33+N byte)

Frame type	SER-ID	Protocol version (2b)	Hardwre version (2b)	Software version (2b)	Software signature (reserved 8b)
service ← response	430	Payload[0,1]	[2,3]	[4,5]	[6,13]
Device unique code(16b)	Device model（Nb）	Other（reserved 2b）			
[14,29]	[30,30+N]	[31+N,32+N]			

- ➢ Remark:Priority >Fast，This command is a standard command, you can view the protocol document uavcan.node.GetInfo

Protocol version	The number 100 means V1.0.0, and so on for other versions
Hardware version	The number 100 means V1.0.0, and so on for other versions
Software version	The number 100 means V1.0.0, and so on for other versions
software signature	Program hash value or other verification, temporarily reserved, filled with 0
Device unique code	Among them, the first 12 bits are the UID of the chip, and the last 4 bits are reserved and filled with 0
Device model	The first byte of the device name indicates the length, and the latter indicates the model content
Other	It is the length of two groups of characters, it is not used temporarily, and it is filled with 0

4.4.3. Node command control 435standard

Request:(3 . . . 258 byte)

Frame type	SER-ID	Command(2b)	parameter(Nb)(0<N<255)
Service → request	435	[0,1]	[2,2+N]

- ➢ Remark: Priority >=Exceptional, This command is a standard command, you can view the protocol document uavcan.node.ExecuteCommand

✧Command

STORE_PERSISTENT_STATES = 65530	1. Let the node save the data. If the register to be operated is a power-down save type and you want to save the data permanently, you must perform this operation. 2. Saving data must be done when the node stops working.
EMERGENCY_STOP = 65531	Reserved
FACTORY_RESET = 65532	The node restores the factory settings, and the settings are successfully powered off to take effect
BEGIN_SOFTWARE_UPDATE = 65533	1. Let the node enter the firmware upgrade mode, and only when the node exits the normal operation can it respond correctly 2. When the node receives the upgrade instruction, if the node is not working, it will enter the pre-upgrade state. 3. The upgrade mode is irreversible, so the docking password is required and stored in the paramter
POWER_OFF = 65534	Reserved
RESTART = 65535	Restarting a node can be executed in any situation, and no information will be returned. If the requester wants to know whether the node is restarted, it can be judged by querying the continuous running time in the node status.

Response: (1byte)

Frame type	SER-ID	status
Service ← response	435	Payload[0]

Remark: this command is a standard command, you can view the protocol document uavcan.node.ExecuteCommand

- State

SUCCESS = 0	The operation is successful, restarting will not return the command
FAILURE = 1	Unable to start or ineffective operation
BAD_COMMAND = 3	This command does not support
BAD_PARAMETER = 4	Parameter and command do not match
BAD_STATE = 5	The current state of the node does not allow the execution of the command
INTERNAL_ERROR	An accident occurred while operating

4.5. Part of the function description of the register table

All the values of the register are 16-bit, and the name is fixed at 12-bit. If the value is less than 12-bit, it will be filled with spaces, and if the name is too long, it will be abbreviated.

4.5.1. CAN baud rate setting

The option can be set after writing the corresponding baud rate to the register address 04H. It must be noted that after the setting is completed, the command = 65530 of the service command control.435 will be called to save the power-off.

5. References

《Cyphal_Specification.pdf》

6. Appendix 1 - Reference Procedures

```
/*****************************************************************************/
/*-------CRC Calculate the correlation function----------------------------------------------------*/
uint16_t crcAddByte(uint16_t crc_val, uint8_t byte)
{
   crc_val ^= (uint16_t) ((uint16_t) (byte) << 8U);
   for (uint8_t j = 0; j < 8; j++)
   {
      if (crc_val & 0x8000U){crc_val = (uint16_t) ((uint16_t) (crc_val << 1U) ^ 0x1021U);}
      else {crc_val = (uint16_t) (crc_val << 1U);}
   }
   return crc_val;
}
uint16_t crcAdd(uint16_t crc_val, const uint8_t* bytes, uint16_t len)
{
   while (len--){crc_val = crcAddByte(crc_val, *bytes++);}
   return crc_val;
}
uint16_t crc_check(const uint8_t *data1, uint16_t length)
{
   return crcAdd(0xffff,data1,length);
}

/*****************************************************************************/
/*----------Throttle Packing Method Demonstration--------------------------------------------------*/
void x_MakeThrot(uint16_t *throt,uint8_t *throtOut)
{
    /*
        Note: the length of the pointer throt must be more than 4
              the length of the pointer throtOut must be more than 8
    */
    /* Remove the upper two digits */
    throt[0] &= 0x3fffu;
    throt[1] &= 0x3fffu;
    throt[2] &= 0x3fffu;
    throt[3] &= 0x3fffu;
    /* Split the upper 6 bits of the last throttle */
    throt[0] |= ((throt[3]<<2)&0xc000u);
    throt[1] |= ((throt[3]<<4)&0xc000u);
    throt[2] |= ((throt[3]<<6)&0xc000u);
    /* Copy data */
    *(uint16_t *)(&throtOut[0]) = throt[0];
    *(uint16_t *)(&throtOut[2]) = throt[1];
    *(uint16_t *)(&throtOut[4]) = throt[2];
    *(uint16_t *)(&throtOut[6]) = throt[3];
}
```

7 Appendix 2 - Register List

No	Value	Name	Defaults	lower limit	upper limit	R/W/E	Description
00H	-	PR-version	100	0	0xffff	R	Protocol version number, 100 means V1.0.0
01H	-	HW-version	100	0	0xffff	R	Hardware version number, 100 means V1.0.0
02H	-	SF-version	100	0	0xffff	R	Software version number, 100 means V1.0.0
03H	-	UploadCtrBit	0xffff	0	0xffff	R/W	位 bit — Remark: 0 uniformly means off BIT0 — 1: Open the heartbeat package. 7509 BIT1 — 1: enable information upload. 6160 BIT2 — 1: enable information upload. 6161
04H	-	CanBaudRate	5	4	7	R/W/E	CANbaud rate 4:250k 5:500K(default) 6:800k 7:1000K
05H	-	UpdaterProc	0	0	10	R	Query the upgrade progress Others: non-upgrade process 0x0020: waiting for communication area program 0x0030: copying communication area program 0x0040: waiting to download the program in the application area 0x00A0: COM area upgraded successfully 0x00A1: APP area upgraded successfully 0x00E0: upgrade failed, the program size exceeds the range 0x00E1: upgrade failed, unique code verification failed
0cH	-	ExCommand	0	0	0xffff	R	Store the instruction updated by uavcan.node.ExecuteCommand,
0dH	-	HB.health	0	0	3	R	Node status, see <heartbeat packet.7509> chapter for details
0eH	-	HB.mode	1	0	3	R	
0fH	-	HB.VendorSta	0	0	255	R	
10H	-	throtSet	0	0	2048	R/W	Throttle signal, in addition to updating by broadcast, can also be updated by reading and writing registers
11H	-	throtCrtOut	0	0	2048	R	Throttle signal, in addition to updating by broadcast, can also be updated by reading and writing registers
12H	-	throtMode	0	0	1	R/W	0 = CAN throttle, 1 = PWM throttle
13H	-	eleFrequency	0	0	0xffff	R	Electrical frequency of motor rotor, unit/0.1Hz

18H	-	Volt_Bus	0	0	0xffff	R	Bus voltage, unit/0.1V
20H	-	IBus	0	-32768	+32767	R	Bus current, unit/0.1A
30H	-	mos-Tem	75	0	255	R	MOS temperature, subtract 40 to get the real temperature
31H	-	cap-Tem	75	0	255	R	Capacitor temperature, subtract 40 to get the real temperature
32H	-	mot-Tem	75	0	255	R	Motor temperature, subtract 40 to get the real temperature
33H	-	runSta	0	0	0xffff	R	Operating status code, Please refer to <information upload.6160> chapter for bit splitting details

Remark: R/W/E means readable/writable/power-down save attribute