HP series Servo CAN Bus Communication Protocol V1.00

CANOpen Drone CAN ARINC 825

HP series Servo CANopen Communication Protocol
HP series Servo Drone CAN Communication Protocol
Based on the ARINC 825 communication protocol

General Description

Scope

This manual brings together the CANOpen, Drone CAN and ARINC 825 communication specifications used for the servo and actuator interfaces described in the three protocol parts. It provides the original protocol descriptions, command definitions, data formats, parameter information and related diagrams in a single reference manual.

Manual Structure

Part I contains the CANOpen Communication Protocol of HP series Servo, including default settings, command descriptions, error codes, the CAN bus wiring diagram and communication port upgrade information.

Part II contains the HP series Servo Drone CAN Interface Description, including supported interfaces, status reports, parameter access, ESC and ACT control, dynamic node allocation and firmware update information.

Part III contains based on ARINC 825 communication protocol, revision 1.3, including the CAN identifier format, actuator functionality, command and reply definitions, and CAN identifier code examples.

Use of the Protocol Parts

Select the part that corresponds to the communication interface fitted to the product. Use the identifiers, data lengths, byte order, value definitions and operating requirements stated in that part. The three protocol parts use their own command and parameter definitions.

CONTENTS

General Description.....2
Part I CANOpen Communication Protocol.....7
1 Default Value.....8
1.1 Standard Frame.....8
1.2 Extended Frame.....8
2 Command Description.....8
2.1 Reporting Start.....8
2.2 Reporting Stop.....8
2.3 Automatic Reporting Format.....8
2.4 Preparation.....8
2.5 Reset.....9
2.6 Communication Reset.....9
2.7 Low Voltage Alarm Reporting Format.....9
2.8 Overcurrent Alarm Reporting Format.....9
2.9 Overtemperature Alarm Reporting Format.....10
2.10 Reporting Interval Setting Command.....10
2.11 Command of Reading the Reporting Interval.....10
2.12 Baud Rate Setting Command.....10
2.13 Command of Reading Baud Rate.....11
2.14 Node ID Setting Command.....11
2.15 Reading Node ID.....11
2.16 Position Setting Command (no response required from slave).....12
2.17 Reading Position.....12
2.18 Midpoint Setting Command.....12
2.19 Midpoint Reading Command.....13
2.20 Saving Command.....13
2.21 Read Status.....13
2.22 Setting of enabling the Function.....13
2.22.1 Position Reverse Function.....14
2.22.2 Contrarotation.....14
2.22.3 Current Limiting Function.....15
2.22.4 Temperature Limiting Function.....15

2.22.5 Current Alarm Function..... 15
2.22.6 Temperature Alarm Function..... 15
2.23 Setting Servo Enable Bit..... 15
2.23.1 Enabling Command Reading..... 15
2.23.2 Enable Command Writing..... 16
2.23.3 Priority Wait Time Setting..... 16
2.23.4 Priority Wait Time Reading..... 16
2.24 Rotation Range Setting..... 17
2.24.1 Negative Rotation Range Command Setting..... 17
2.24.2 Positive Rotation Range Command Setting..... 17
2.25 Rotation Range Reading..... 17
2.25.1 Negative Rotation Range Reading Command..... 17
2.25.2 Positive Rotation Range Reading Command..... 18
2.26 Self-Test Low Range Setting Command..... 18
2.27 Self-Test Low Range Reading Command..... 18
2.28 Self-Test High Range Setting Command..... 19
2.29 Self-Test High Range Reading Command..... 19
2.30 Midpoint Voltage Setting Command..... 19
2.31 Midpoint Voltage Reading Command..... 19
2.32 Angle Proportion Setting Command..... 20
2.33 Angle Proportion Reading Command..... 20
2.34 Operating Time Reset Command..... 20
2.35 Operating Time Reading Command..... 20
2.36 Midpoint Count Reset Command..... 21
2.37 Midpoint Count Reading Command..... 21
2.38 Power-On Count Reset Command..... 21
2.39 Power-On Count Reading Command..... 21
2.40 Temperature Proportion Setting Command..... 22
2.41 Temperature Proportion Reading Command..... 22
2.42 Current Proportion Setting Command..... 22
2.43 Current Proportion Reading Command..... 22
2.44 Torque Release Command..... 23
2.45 Guard Function..... 23
2.45.1 Read/Set Guard Enable..... 23
2.45.2 Guard Interval Reading..... 24

2.45.3 Guard Interval Setting..... 24
2.45.4 Guard Position Reading.....24
2.45.5 Guard Position Setting.....25
2.45.6 Guard Command Sending Format of Host.....25
2.46 Angle ADC Voltage Reading Command..... 25
2.47 Status Data Reading Command.....26
2.48 Factory Configuration Restoring.....26
2.49 Software Version Reading.....26
2.50 Hardware Version Reading..... 26
2.51 Read-write Failure Response.....27
Appendix I: Error Code Description.....27
Appendix II: CAN Bus Wiring Diagram.....28
Appendix III: Communication Port Upgrade..... 28
Part II Drone CAN Communication Protocol..... 29
1、DroneCAN interface support：.....30
1.1 Node status report..... 30
1.2 ACT status report.....30
1.3ESC status report..... 30
1.4 Read Servo Information.....30
1.5 Read and Write parameters.....30
1.6 ESC control.....30
1.7 ACT control..... 30
1.8 Specific operation..... 31
1.9 DNA support.....31
1.10 Firmware updata support.....错误！未定义书签。
Part III ARINC 825 Communication Protocol.....32
1 Introduction.....33
2 CAN Protocol.....33
2.1 Bitrate.....33
2.2 Endianness..... 34
2.3 General ARINC 825 29 -bit CAN Identifier Structure..... 34
2.4 ARINC825 CAN ID format..... 35
3 Actuator functionalities.....36
3.1 [V1.3] Blocked actuator detection and protection.....36

4 CAN commands.....37
4.1 Set / write functions.....37
4.1.1 Set position command.....37
4.1.2 Set torque command.....38
4.1.3 Set speed command.....38
4.1.4 Set parameter.....40
4.1.5 Set user defined data.....41
4.2 Read functions.....41
4.2.1 Get position command.....41
4.2.2 Get torque command.....41
4.2.3 Get speed command.....42
4.2.4 Get parameter.....42
4.2.5 [V1.3] Get firmware version.....43
4.2.6 Get user defined data.....43
5 Code examples.....43
5.1 CAN ID creation via C macro.....43

Part I CANOpen Communication Protocol

CANOpen Communication Protocol of HP series Servo

The Previous Version			
Version Number	**Date**	**Author**	**Revised Content**
V1.00	2026	Alan	Initial version

1 Default Value

1.1.Standard Frame:

Node ID: 0x25 (range 1-127, 0 is broadcast)

Standard frame: Baud Rate: 500kbps

Communication：CANOpen standard frame format-CAN2.0B

1.2Extended Frame:

Version CAN2.0B

CAN_SJW CAN_SJW_1tq

CAN_BS1 CAN_BS1_4tq

CAN_BS2 CAN_BS2_1tq

ID 29bit (extended frame)

Baud 500Kbps (default)

2 Command Description

2.1 Reporting Start

ID	Length	Bit 1	Bit 2
0x 00 00 00 00 +Node ID	2	01	00

Reporting start indicates successful startup.

2.2 Reporting Stop

ID	Length	Bit 1	Bit 2
0x 00 00 00 00 +Node ID	2	02	00

Reporting stop indicates successful stop.

2.3 Automatic Reporting Format

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4
0x 00 00 05 80 +Node ID	4	Low	High	00	00

(The definition of High and Low is the same as in the position setting command thereinafter.)

2.4 Preparation

ID	Length	Bit 1	Bit 2
0x 00 00 00 00 +Node ID	2	80	00

Prepare to enter operational state.

2.5 Reset

ID	Length	Bit 1	Bit 2
0x 00 00 00 00 +Node ID	2	81	00

Perform system reset.

2.6 Communication Reset

ID	Length	Bit 1	Bit 2
0x 00 00 00 00 +Node ID	2	82	00

Communication reset.

2.7 Low Voltage Alarm Reporting Format

When the host receives the following command, it indicates that the system supply voltage is too low (typically 70% of the servo's rated operating voltage). At this point, the servo will lose its torque. When the host no longer receives the following command, there are two possibilities: 1. The system voltage has returned to normal; 2. The system voltage is too low and the servo cannot operate. These two situations can be distinguished by checking whether communication between the host and the servo is normal. Low voltage alarm has no relationship with automatic reporting. When the low voltage alarm is triggered, reporting happens immediately, and then it happens every 10 seconds. Reporting ceases, once the alarm stops. (This reporting command is disabled by default. User can enable the function by setting.)

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4
0x 00 00 03 80 +Node ID	4	0x0A	0x02	00	00

2.8 Overcurrent Alarm Reporting Format

When the host receives the following command, it indicates that the system current has reached the current limitation (the limitation value can be set by the user; but factory default current limitation is recommended.). At this point, the servo current will be limited to a value that is set. When the host no longer receives the following command, it indicates that the system current is below the current limitation. This situation can be determined by checking whether communication between the host and the servo is normal. The overcurrent alarm has no relationship with automatic reporting. When the overcurrent alarm is triggered, reporting happens immediately, and then it happens every 10 seconds. Once the alarm stops, reporting would cease. (This reporting command is enabled by default. User can disable the function by setting.)

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4
0x 00 00 03 80 +Node ID	4	0x0A	0x03	00	00

2.9 Overtemperature Alarm Reporting Format

When the host receives the following command, it indicates that the system temperature has reached the temperature limitation (this limitation shall be set by users; a setting of 100°C is recommended). At this point, the servo temperature will be limited to the set value. When the host no longer receives the following command, it indicates that the system temperature has fallen below the temperature limitation. This situation can be determined by checking whether communication between the host and the servo is normal. The overtemperature alarm has no relationship with reporting. When overtemperature alarm is triggered, reporting happens immediately, and then it happens every 10 seconds. Reporting ceases, once the alarm stops. (This reporting command is enabled by default. User can disable the function by setting.)

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4
0x 00 00 03 80 +Node ID	4	0x0A	0x04	00	00

2.10 Reporting Interval Setting Command

The default reporting interval is 50ms. You can set the interval to a value between 10 and 255ms through relevant command.

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	22	00	22	00	X	00	00	00

X is the Bit used to set the reporting interval. For example, 0x32 corresponds to 50ms, 0x64 corresponds to 100ms, and 0x0A corresponds to 10ms.

Receiving the following response indicates that the setting is successful:

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +Node ID	8	60	00	22	00	00	00	00	00

Once it is successful, the interval setting takes effect immediately. If the setting is not saved through save command, it will be lost after a restart.

2.11 Command of Reading the Reporting Interval

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	40	00	22	00	00	00	00	00

Response:

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +Node ID	8	4B	00	22	00	X	00	00	00

2.12 Baud Rate Setting Command

The default baud rate is 500K. Using the setting command, the baud rate can be set to an optional value from the lookup table.

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	22	01	30	00	X	00	00	00

X is the baud rate setting Bit. The baud rate setting lookup table is as follows:

X	00	01	02	03	04	05	06	07
Baud Rate	20K	50K	100K	125K	250K	500K	800K	1000K

Receiving the following response indicates that the setting is successful:

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +Node ID	8	60	01	30	00	00	00	00	00

Changes to the baud rate must be saved using the save command and will take effect only after a restart.

2.13 Command of Reading Baud Rate

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	40	01	30	00	00	00	00	00

Response:

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +Node ID	8	4F	01	30	00	X	00	00	00

X represents the baud rate.

2.14 Node ID Setting Command

The default Node ID is 0x25.

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	22	00	30	00	X	00	00	00

X is the new Node ID.

Receiving the following response indicates that the setting is successful:

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +Node ID	8	60	00	30	00	00	00	00	00

Changes to the Node ID must be saved using the save command and will take effect only after a restart.

2.15 Reading Node ID

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	40	00	30	00	00	00	00	00

Response:

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +Node ID	8	4F	00	30	00	X	00	00	00

X represents the Node ID.

2.16 Position Setting Command (no response required from slave)

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	22	03	60	00	Low	High	00	00

The position data accuracy can be set to 0.1°, with a range of -100° to +100°. Negative values are represented in two's complement format, as follows:

Position	-100°	-50°	-10.9°	-0.8°	0°	0.8°	10.9°	50°	100°
High	0xFC	0xFE	0xFF	0xFF	0x00	0x00	0x00	0x01	0x03
Low	0x18	0x0C	0x93	0xF8	0x00	0x08	0x6D	0xF4	0xE8

2.17 Reading Position

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	40	02	60	00	00	00	00	00

Response:

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +Node ID	8	4B	02	60	00	Low	High	00	00

Reading position is defined in the same way as the High and Low Bits in the position setting.

2.18 Midpoint Setting Command

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	22	09	30	00	00	00	00	00

The midpoint setting command is used to set the current position as the midpoint. This command is intended for fine-tuning the midpoint and is limited to a range of ±20°; for example, to set the 5° position as the midpoint, first send the command to set the position to 5°, then send the midpoint setting command, which will set the 5° position as the midpoint.

Receiving the following response indicates that the setting is successful:

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +NodeID	8	60	09	30	00	00	00	00	00

Once the midpoint is set, it takes effect immediately. If you do not save it using the save command, the midpoint will revert to its original one after a restart.

2.19 Midpoint Reading Command

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	40	09	30	00	00	00	00	00

Response:

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +Node ID	8	4B	09	30	00	Low	High	00	00

The definition of the High and Low Bits for reading the midpoint position is the same as that for setting the position.

2.20 Saving Command

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	22	10	10	01	73	61	76	65

This command is used to save the Node ID, baud rate, etc. (The slave does not need to respond to this command.)

2.21 Read Status

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	40	05	60	00	00	00	00	00

Response:

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +Node ID	8	43	05	60	00	X1	X2	X3	00

X1X2 represents the operating current (accuracy ±5%, or 0.01 A), in units of 10mA. For example:

X2	0x00	0x00	0x00	0x01	0x01
X1	0x32	0x4B	0x64	0x00	0xF4
Current	0.50A	0.75A	1.00A	2.56A	5.00A

X3 represents the operating temperature (temperature range: -40 to +120°C, accuracy: ±2°C), expressed in two's complement format. For example:

X3	0xD8	0xF6	0x00	0x19	0x78
Temperature	-40°C	-10°C	0°C	25°C	120°C

2.22 Setting of enabling the Function

This command is used to enable or disable specific functions in the system. First, you must read the data from the enabling Bit, then modify only the relevant bits to enable the corresponding function (leaving the other bits unchanged). “1” is for “enable” and “0” is for “disable” by default;

The command below is for reading the enabling:

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	40	0F	30	00	00	00	00	00

Response:

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +Node ID	8	4B	0F	30	00	X1	X2	00	00

X1 and X2 represent the read back enable Bits; the functions of each bit are as follows:

Bit	7	6	5	4	3	2	1	0
Function (X1)	Power-On Count	Position Reverse	Low Voltage Alarm	Current Sampling	Power-On Self-Test	Position Response	Midpoint Count	Working Time
Bit	15	14	13	12	11	10	9	8
Function (X2)	X	Temperature Protection (reserved)	Current Protection (reserved)	Temperature Alarm	Current Alarm	Temperature Limitation	Current Limitation	Contrarotation

X indicates a reserved bit; modifying X will make servo malfunction.

Enabling command rewriting is as follows: X1 and X2 are the data written by user.

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	22	0F	30	00	X1	X2	00	00

Response of following command indicates that rewriting is successful.

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +Node ID	8	60	0F	30	00	00	00	00	00

Once the rewriting is enabled, it takes effect immediately. If you do not save the changes using the save command, the system will revert to its previous state after a restart.

2.22.1 Position Reverse Function

This function is implemented via the “Position Reverse” bit in the function enable setting. When the “Position Reverse” bit is set to be “0”, the position returned by the read position command is positive; when the “Position Reverse” bit is set to be “1”, the position returned by the read position command is negative. For example, if the current position is 50°, and the “Position Reverse” bit in the function enable settings is set to be “0”, the read position command returns a position of “50°”; if the “Position Reverse” bit is set to be “1”, the read position command returns a position of “-50°”.

2.22.2 Contrarotation

This function is enabled via the “contrarotation” bit in the function enable setting. When the “contrarotation” bit is set to be “0”, the servo rotates in the default direction; when the

"contrarotation" bit is set to be "1", the servo rotates in the opposite direction of the default, and the position command received synchronously is also reversed.

2.22.3 Current Limiting Function

This function is enabled via the "Current Limit" bit in the function enable setting. When the "Current Limit" bit is set to be "0", the firmware does not limit the servo's operating current. When the "Current Limit" bit is set to be "1", the firmware limits the servo's maximum operating current (which also limits the servo's output torque). The current upper limit could be adjusted by using the Current Limit setting function.

2.22.4 Temperature Limiting Function

This function is enabled via the "Temperature Limit" bit in the function enable setting. When the "Temperature Limit" bit is set to be "0", the firmware does not limit the servo's operating temperature; when the "Temperature Limit" bit is set to be "1", the firmware limits the servo's maximum operating temperature (which also limits the servo's output torque). The temperature upper limit could be adjusted by using the Temperature Limit setting function.

2.22.5 Current Alarm Function

This function is enabled via the "Current Alarm" bit in the function enable setting. When the "Current Alarm" bit is set to be "0", the current alarm function is disabled; when the "Current Alarm" bit is set to be "1", the current alarm function is enabled. A current alarm is triggered when the current reaches the current limitation.

2.22.6 Temperature Alarm Function

This function is enabled via the "Temperature Alarm" bit in the function enable setting. When the "Temperature Alarm" bit is set to be "0", the temperature alarm function is disabled; when the "Temperature Alarm" bit is set to be "1", the temperature alarm function is enabled. A temperature alarm is triggered when the temperature reaches the temperature limitation.

2.23 Setting Servo Enable Bit

This command is used to determine whether certain bits in the system are enabled. First, read the data in the enable bits, and then only change the corresponding bits to enable the functions (while leaving the other bits unchanged). The default state of "1" indicates enabled, and "0" indicates disabled;

2.23.1 Enabling Command Reading

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	40	11	30	00	00	00	00	00

Response:

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +Node ID	8	4F	11	30	00	X	00	00	00

The enable bit read back has the following meaning for each bit:

Bit	7	6	5	4	3	2	1	0
Function	Reverse Servo	X	485 Priority		CAN Priority		PWM Priority	

Reverse Servo: The effect of this function is equivalent to that of contrarotation.

X: Indicates reserved bit. Changing this bit may cause servo malfunction;

Priority: 0 (high) > 1 (medium) > 2 (low) > 3 (reserve)

When the loss time of higher-priority communication reaches the preset waiting time and lower-priority communication is functioning normally, control switches to the lower-priority communication.

If the higher-priority communication is re-established, control will switch directly to the higher-priority communication regardless of the lower-priority status.

Priority settings are effective only for control commands and do not affect non-control commands such as configuration or upgrades.

2.23.2 Enable Command Writing

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	22	11	30	00	X	00	00	00

Response of the following command indicates that the rewrite is successful:

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +Node ID	8	60	11	30	00	00	00	00	00

Once the rewrite function is enabled successfully, it takes effect immediately. If you do not save the changes by using the save command, the system will revert to its previous state after a restart.

2.23.3 Priority Wait Time Setting

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	22	12	30	00	X	00	00	00

X represents the wait time for switching between different priorities, ranging from 1 to 250 in unit of 4ms, i.e. 4ms to 1000ms.

Response of the following indicates that the setting is successful:

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +Node ID	8	60	12	30	00	00	00	00	00

Once the rewrite is enabled successfully, it takes effect immediately. If you do not save the changes using the save command, the system will revert to its previous state after a restart.

2.23.4 Priority Wait Time Reading

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	40	12	30	00	00	00	00	00

Response:

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +Node ID	8	4F	12	30	00	X	00	00	00

2.24 Rotation Range Setting

This function allows users to set the range of servo's travel angle. The rotation range limits must be within ±100°(positive range: 0 to +100°, negative range: 0 to -100°; the factory default is ±100°). After successfully sending the range setting command, user must send a save command; the rotation range will take effect after a restart. Once the range is successfully set, if a position command exceeds the range, the servo will not move and will only return an error code in response. The range includes both positive and negative directions; the corresponding setting commands are described as follows:

2.24.1 Negative Rotation Range Command Setting

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	22	0A	30	00	Low	High	00	00

The definitions of the High and Low bits are the same as that of position setting command.

The following response indicates that the setting is successful:

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +Node ID	8	60	0A	30	00	00	00	00	00

2.24.2 Positive Rotation Range Command Setting

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	22	0B	30	00	Low	High	00	00

The definitions of the High and Low bits are the same as that of position setting command.

The following response indicates that the setting is successful:

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +Node ID	8	60	0B	30	00	00	00	00	00

2.25 Rotation Range Reading

This function reads servo's current range of rotation. The command is described as follows:

2.25.1 Negative Rotation Range Reading Command

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	40	0A	30	00	00	00	00	00

The following response indicates that reading is successful:

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +Node ID	8	4B	0A	30	00	Low	High	00	00

The definitions of the High and Low bits are the same as that of position setting command.

2.25.2 Positive Rotation Range Reading Command

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	40	0B	30	00	00	00	00	00

The following response indicates that reading is successful:

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +Node ID	8	4B	0B	30	00	Low	High	00	00

The definitions of the High and Low bits are the same as that of position setting command.

2.26 Self-Test Low Range Setting Command

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	22	0C	30	00	Low	High	00	00

The definitions of the High and Low bits are the same as that of position setting command. Unit: 0.1 degrees.

The following response indicates that the setting is successful:

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +Node ID	8	60	0C	30	00	00	00	00	00

Once the setting command has been successfully executed, user should send the save command; the self-test range setting will take effect after a restart.

2.27 Self-Test Low Range Reading Command

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	40	0C	30	00	00	00	00	00

Response:

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +Node ID	8	4B	0C	30	00	Low	High	00	00

The definitions of the High and Low bits are the same as that of position setting command. Unit: 0.1 degrees.

2.28 Self-Test High Range Setting Command

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	22	0D	30	00	Low	High	00	00

The definitions of the High and Low bits are the same as that of position setting command. Unit: 0.1 degrees.

The following response indicates that the setting is successful:

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +Node ID	8	60	0D	30	00	00	00	00	00

Once the setting command has been successfully executed, user should send the save command; the self-test range setting will take effect after a restart.

2.29 Self-Test High Range Reading Command

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	40	0D	30	00	00	00	00	00

Response:

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +Node ID	8	4B	0D	30	00	Low	High	00	00

The definitions of the High and Low bits are the same as that of position setting command. Unit: 0.1 degrees.

2.30 Midpoint Voltage Setting Command

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	22	02	30	00	Low	High	00	00

The written value is the midpoint voltage ADC value. Low indicates the low bit, High indicates the high bit.

The following response indicates that the setting is successful:

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +Node ID	8	60	02	30	00	00	00	00	00

Once the setting command has been successfully executed, user should send the save command; the set midpoint voltage will take effect after a restart.

2.31 Midpoint Voltage Reading Command

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	40	02	30	00	00	00	00	00

Response:

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +Node ID	8	4B	02	30	00	Low	High	00	00

The read value is the midpoint voltage ADC value. Low indicates the low bit, High indicates the high bit.

2.32 Angle Proportion Setting Command

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	22	03	30	00	Low	High	00	00

Low indicates the Low bit, High indicates the High bit.

The following response indicates that the setting is successful:

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +Node ID	8	60	03	30	00	00	00	00	00

Once the setting command has been successfully executed, user should send the save command; the angle proportion will take effect after a restart.

2.33 Angle Proportion Reading Command

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	40	03	30	00	00	00	00	00

Response:

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +Node ID	8	4B	03	30	00	Low	High	00	00

Low indicates the Low bit, High indicates the High bit.

2.34 Operating Time Reset Command

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	22	04	30	00	00	00	00	00

The following response indicates that the setting is successful:

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +Node ID	8	60	04	30	00	00	00	00	00

2.35 Operating Time Reading Command

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	40	04	30	00	00	00	00	00

Response:

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +Node ID	8	43	04	30	00	X1	X2	X3	X4

X1 represents the Low bit; the Low bit comes first, followed by the High Bit. Unit: 10 minutes.

2.36 Midpoint Count Reset Command

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	22	05	30	00	00	00	00	00

The following response indicates that the setting is successful:

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +Node ID	8	60	05	30	00	00	00	00	00

2.37 Midpoint Count Reading Command

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	40	05	30	00	00	00	00	00

Response:

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +Node ID	8	4B	05	30	00	Low	High	00	00

Low indicates the Low bit, High indicates the High bit.

2.38 Power-On Count Reset Command

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	22	06	30	00	00	00	00	00

The following response indicates that the setting is successful:

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +Node ID	8	60	06	30	00	00	00	00	00

2.39 Power-On Count Reading Command

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	40	06	30	00	00	00	00	00

Response:

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +Node ID	8	43	06	30	00	X1	X2	X3	X4

X1 represents the Low bit; the Low bit comes first, followed by the High bit.

2.40 Temperature Proportion Setting Command

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	22	07	30	00	X	00	00	00

X represents the temperature proportion; the formula is: |(operating temperature - X)| < 20. The operating temperature can be read at 2.19.

The following response indicates that the setting is successful:

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +Node ID	8	60	07	30	00	00	00	00	00

Once the setting command has been successfully executed, user should send the save command; the temperature proportion setting will take effect after a restart.

2.41 Temperature Proportion Reading Command

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	40	07	30	00	00	00	00	00

Response:

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +Node ID	8	4F	07	30	00	X	00	00	00

X represents the temperature proportion.

2.42 Current Proportion Setting Command

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	22	08	30	00	Low	High	00	00

Low indicates the Low bit, High indicates the High bit.

The following response indicates that the setting is successful:

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +Node ID	8	60	08	30	00	00	00	00	00

Once the setting command has been successfully executed, user should send the save command; the set current proportion will take effect after a restart.

2.43 Current Proportion Reading Command

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	40	08	30	00	00	00	00	00

Response:

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +Node ID	8	4B	08	30	00	Low	High	00	00

Low indicates the Low bit, High indicates the High bit.

2.44 Torque Release Command

When this command is sent, the servo's output torque is 0. If a position command is sent again at this point, the servo will immediately move to the corresponding position;

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	22	10	30	00	00	00	00	00

The following response indicates that the operation is successful:

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +Node ID	8	60	10	30	00	00	00	00	00

2.45 Guard Function

The guard function is designed to determine whether communication between the host and the servo is normal. It works by having the host send a request command at fixed intervals; if the servo detects the command within the set time, communication is considered normal. If no valid command is detected within the set time, communication is deemed to be abnormal, and the servo travels to a pre-specified position. When the guard enable is set to "0", the guard function is disabled. (The default guard enable upon power-on is "0", indicating that the guard function is disabled.)

2.45.1 Read/Set Guard Enable

This command is used to enable or disable specific functions in the system. First, read the data in the enable bit, then modify only the corresponding bits to enable the desired functions (while leaving the other bits unchanged). The default state "1" indicates enabled, and "0" indicates disabled.

The Read Enable command is as follows:

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	40	0F	30	00	00	00	00	00

The following response indicates that the operation was successful:

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +Node ID	8	4B	0C	10	00	X1	X2	X3	00

X1, X2, and X3 are the enable Bits of read back. X1 and X2 indicate the functions of viewing the setting configuration, while X3 indicates the functions as follows:

Bit	7	6	5	4	3	2	1	0
Function (X1)	X	X	X	X	X	X	X	Guard Function

X: Indicates reserved bit. Changing this bit may cause servo malfunction;

Enable command rewriting is as follows: X1, X2, and X3 are the data written by users.

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	22	0F	30	00	X1	X2	X3	00

The following response indicates a successful write.

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +Node ID	8	60	0F	30	00	00	00	00	00

Once the rewriting is successful, it takes effect immediately. If not saved using the save command, it will revert to the previous state after a restart.

2.45.2 Guard Interval Reading

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	40	0C	10	00	00	00	00	00

The following response indicates that the operation is successful:

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +Node ID	8	4B	0C	10	00	Low	High	00	00

The guard time is indicated by two bits: High and Low. Units: ms and a range of 100ms to 5,000ms.

2.45.3 Guard Interval Setting

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	22	0C	10	00	Low	High	00	00

Low and High have the same definition as the guard interval reading.

The following response indicates that the operation was successful:

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +Node ID	8	60	0C	10	00	00	00	00	00

2.45.4 Guard Position Reading

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	40	21	30	00	00	00	00	00

The following response indicates that the operation is successful:

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +Node ID	8	4B	21	30	00	Low	High	00	00

The definitions of the Low and High bits in the guard position are the same as those for the position setting in Section 2.8.

2.45.5 Guard Position Setting

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	22	21	30	00	Low	High	00	00

The following response indicates that the operation is successful:

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +Node ID	8	60	21	30	00	00	00	00	00

2.45.6 Guard Command Sending Format of Host

ID	Length
0x 00 00 07 00 +Node ID	This command does not carry any data and must be sent in remote frame format.

The servo responds following commands, indicating that communication is working properly:

ID	Length	Bit 1
0x 00 00 07 00 +Node ID	1	X

“X” is the data replied by the servo; the most significant bit of “X” alternates between ‘0’ and “1”;

2.46 Angle ADC Voltage Reading Command

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	40	04	60	00	00	00	00	00

Response:

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +Node ID	8	4B	04	60	00	Low	High	00	00

The data read is the raw ADC sample value. “Low” refers to the Low bit, and “High” refers to the High bit.

2.47 Status Data Reading Command

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	40	01	60	00	00	00	00	00

Response:

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +Node ID	8	4B	01	60	00	S0	S1	00	00

S0 indicates the Low bit, and S1 indicates the High bit, for debugging purposes only.

2.48 Factory Configuration Restoring

This function restores the servo configuration data to factory defaults by sending a command from the host.

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	22	11	10	01	6C	6F	61	64

The following response indicates that the operation is successful:

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +Node ID	8	60	11	10	01	00	00	00	00

2.49 Software Version Reading

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	40	0B	10	00	00	00	00	00

Response:

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +Node ID	8	43	0B	10	00	X1	X2	X3	X4

X1X2X3X4 is the software version string.

2.50 Hardware Version Reading

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 06 00 +Node ID	8	40	0A	10	00	00	00	00	00

Response:

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +Node ID	8	43	0A	10	00	X1	X2	X3	X4

X1X2X3X4 is the hardware version string.

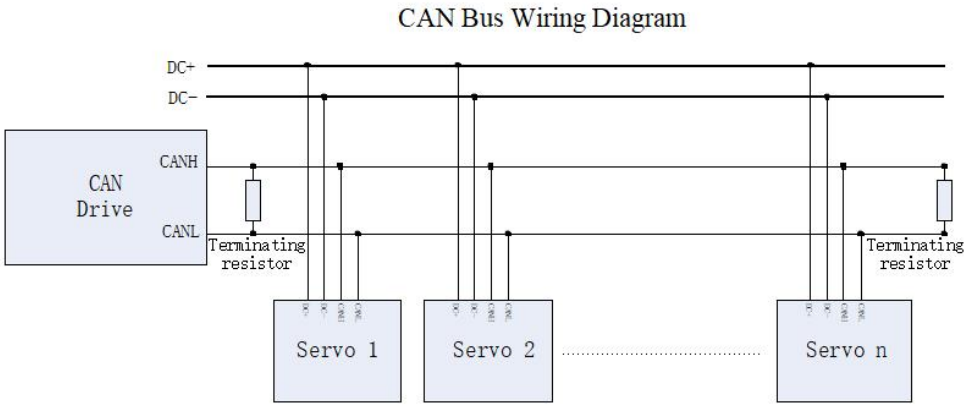
2.51 Read-write Failure Response

ID	Length	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
0x 00 00 05 80 +Node ID	8	80	Same as Read/Write Index			Error Code			

Appendix I: Error Code Description

Error Code	Error Description
0503 0000	Non-alternating position
0504 0000	SDO protocol timeout
0504 0001	Client/server command specifier invalid or unknown
0504 0002	Invalid block size (block mode only)
0504 0003	Invalid sequence number (block mode only)
0504 0004	CRC error (block mode only)
0504 0005	Out of memory
0601 0000	Unsupported access to an object
0601 0001	Attempt to read a write-only object
0601 0002	Attempt to write a read-only object
0602 0000	Object does not exist in the object dictionary
0604 0041	Object cannot be mapped to the PDO
0604 0042	Number and length of objects to be mapped exceeds PDO length
0604 0043	General parameter incompatibility reason
0604 0047	General internal incompatibility in the device
0606 0000	Access failed due to a hardware error
0607 0010	Data type mismatch, service length parameter mismatch
0607 0012	Data type mismatch, service length parameter too high
0607 0013	Data type mismatch, service length parameter too low
0609 0011	Sub-index does not exist
0609 0030	Sub-index does not exist, value range of parameter exceeded (write access only)
0609 0031	Parameter value written too high
0609 0032	Parameter value written too low
0609 0036	Maximum value is less than minimum value
0800 0000	General error
0800 0020	Data cannot be transferred or stored to the application
0800 0021	Data cannot be transferred or stored to the application due to local control
0800 0022	Data cannot be transferred or stored to the application due to the current device state
0800 0023	Object dictionary dynamic generation failed or object dictionary does not exist

Appendix II: CAN Bus Wiring Diagram



1. Connect one 120 Ω terminating resistor at each end of the Bus.
2. Keep the stub length from the Bus to each servo as short as possible.
3. The Bus baud rate depends on the number of servos on the Bus and the stub length from the Bus to each servo.

Appendix III: Communication Port Upgrade

For details, please refer to the “Communication Port Upgrade Software Instructions”.

Part II Drone CAN Communication Protocol

HP series Servo Drone CAN Interface Description

Document modifications			
RN	**Date**	**Author**	**Modifications**
V1.00	2026-08-25	Alan	Creation;

The HP series Servo Drone CAN interface V1.00

1、DroneCAN interface support：

1.1 Node status report

Interface ID :UAVCAN_PROTOCOL_NODESTATUS_ID

Interface Object : uavcan_protocol_NodeStatus

1.2 ACT status report

Interface ID:UAVCAN_EQUIPMENT_ACTUATOR_STATUS_ID

Interface Object :uavcan_equipment_actuator_Status

Field description:

Position: Current servo position, ranging from -1 to 1, mapped to -100° to 100°

1.3ESC status report

Interface ID:UAVCAN_EQUIPMENT_ESC_STATUS_ID

Interface Object :uavcan_equipment_esc_Status

Field description:

Rpm: Current servo position, ranging from 0 to 8191, mapped to -100° to 100°

1.4 Read Servo Information

Interface ID:UAVCAN_PROTOCOL_GETNODEINFO_ID

Interface Object :uavcan_protocol_GetNodeInfoResponse

1.5 Read and Write parameters

Interface ID:UAVCAN_PROTOCOL_PARAM_GETSET_ID

Interface Object : parameter list

Index	Parameter	Range	Default	
0	can_baud	0~8	8	0~8:10k,20k,50k,100k, 125k,250k,500k,800k,1M
1	pwm_channel	0~19/0~255	0	The index of ESC or the id of ACT
2	control_type	0、1	0	0 : ESC data, 1 : ACT array
3	set_to_zero	1		Set the current position as the midpoint
4	SetLimt_Low	-1000~1000	-1000	0.1° unit
5	SetLimit_High	-1000~1000	1000	0.1° unit
6	Set_isCCW	0、1	1	0:CW; 1:CCW
7	node_ID	1~127	-	Default dynamic allocation

1.6 ESC control

Interface ID: UAVCAN_EQUIPMENT_ESC_RAWCOMMAND_ID

Interface Object : uavcan_equipment_esc_RawCommand

Operation performed:

Input values ranging from 0 to 8191 are mapped to a range of -100° to 100°. Commands exceeding the mapped limits will be ignored.

1.7 ACT control

Interface ID: UAVCAN_EQUIPMENT_ACTUATOR_ARRAYCOMMAND_ID

Interface Object : uavcan_equipment_actuator_ArrayCommand

Operation performed:

When command_type is UNITLESS:

Input values ranging from -1.0 to 1.0 are mapped to a range of -100° to 100°. Commands exceeding the mapped limits will be ignored.

When command_type is PWM:

Input values ranging from 500 to 2500 are mapped to a range of -100° to 100°. Commands exceeding the mapped limits will be ignored.

1.8 Specific operation

Interface ID:UAVCAN_PROTOCOL_PARAM_EXECUTEOPCODE_ID

Interface Object :uavcan_protocol_param_ExecuteOpcodeRequest

1.9 DNA support

Provide a 12 byte unique ID and a 64 bit US timer to support automatic address allocation.

1.10 Firmware Update Support

The built-in Bootloader supports firmware updates. Firmware updates use standard CAN frames, while UAVCAN uses CAN extended frames, so the two won't conflict.

Update software: (Xmain) can run on Windows 7 and later versions.

Actuator ICD (Interface Control Document) based on ARINC 825

Revision	Date	Creators / Revisors	Comment
1.0	21.10.2025		Initial version
1.1	26.10.2025		Added get functions, corrected typos
1.2	26.10.2025		Added detailed value information for the set position reply
1.3	08.09.2026		Added description for blockage handling. New text data are marked with [V1.3]

Changelog:

[V1.3]:

- Added section for actuator functionalities – Blocked actuator detection and protection
- Specification in which timeframe the reply message shall be sent by an actuator
- Added set parameter to reset the servo's center position to the internal zero position
- Added <get firmware version> to the actuator read functions

1 Introduction

The ARINC825 Standard – ARINC Specification 825: General Standardization of CAN (Controller Area Network) Bus Protocol for Airbome Use, was prepared by the Airlines Electronic Engineering Committee (AEEC) and published by ARINC in 2007. ARINC has since published subsequent revisions. Users are encouraged to check with ARINC as to the status of this standard.

ARINC Specification 825 defines a communication standard for airborne systems using CAN, which is viewed as an important data transfer medium for current and future aircraft. Airbus and Boeing were instrumental in the development of this standard. The supplier community also participated in industry meetings that were organized by ARINC and open to the public.

The use of ARINC Specification 825 is voluntary in nature. Current production aircraft have used CAN as an ancillary subsystem bus to ARINC 664, Part 7, (Deterministic Ethernet). ln some aircraft, CAN buses have been used to link sensors, actuators and other types of avionics devices that typically require low to medium data transmission volumes during operation. ln this role, CAN complements higher capacity data networks to support systems that control other systems; for example, flight deck information flow and display.

ARINC825 shall be used as a guideline for the definition of the CAN actuator identifiers and the message definitions. Furthermore, ARINC825 is giving examples for values for logic communication channels and so called FIDs to which this ICD will adhere.

2 CAN Protocol

CAN 2.0B is a specific version of the Controller Area Network (CAN) protocol that supports both standard (11-bit) and extended (29-bit) message identifiers. It is a serial communication protocol originally developed by Bosch for reducing the wiring harnesses in vehicles. CAN 2.0B is widely used in automotive, industrial, and other embedded systems for robust communication between electronic control units (ECUs).

This ICD fully follows the CAN 2.0B standard and only 29bit addresses will be used.

On the physical layer side this ICD adheres to ISO 11898-2.

2.1 Bitrate

Default actuator CAN bitrate: 500kbps

Optional CAN bitrates (via configuration): 250 kbps, 1 Mbps

2.2 Endianness

All data structures, ordinal datatypes and parameters have little endian notation.

Here, this ICD is not adhering to the ARINC825 standard and this is deliberate to ease implementation on common embedded system which mostly have a little endian data processing.

2.3 General ARINC 825 29 -bit CAN Identifier Structure

For command and data exchange between the flight control computer (FCC) and the actuators, the ARINC825 – one to many – identifier scheme schall be used:

Peer-To-Peer (PTP) Identifier Structure

31 30 29	28 27 26	25 24 23 22 21 20 19	18	17	16	15 14 13 12 11 10 9	8 7 6 5 4 3 2	1 0
	LCC	Client FID	SMT	LCL	PVT	Server FID	SID	RCI
						NID		

Anyone-To-Many (ATM) Identifier Structure

31 30 29	28 27 26	25 24 23 22 21 20 19	18	17	16	15 14 13 12 11 10 9 8 7 6 5 4 3 2	1 0
	LCC	Source FID	RSD	LCL	PVT	DOC	RCI

Field	Description	Field	Description
LCC	Logical Communication Channel	PVT	Private Data
FID	Aircraft Function ID	SID	Server ID
RSD	Reserved	RCI	Redundancy Channel ID
SMT	Service Message Type	NID	Node ID
LCL	Local Bus Only	DOC	Data Object Code

Figure 1: ARINC 825 CAN Identifier Structures

The main ARINC825 fields in the identifier are:

- LCC – Logic communication channel. LLC = 2 = normal operation channel. LLC = 5 = test and maintenance channel.
- FID – system or subsystem of the message - this value will be fixed. FID = 10d for actuators
- RSD, LCL, PVT – bits for future use
- DOC – Data object code – this is the message ID which will contain the information on the command and the actuator ID
- RCI – Redundancy channel index – allows to have multiple FCCs onboard and the actuators could be programmed to listen to a specific one – currently unused

2.4 ARINC825 CAN ID format

The actuator ARINC825 scheme is depicted below.

28 27 26	25 24 23 22 21 20 19	18	17	16	15	14	13 12 11 10 9 8	7 6 5 4 3 2	1 0
LCC	FID = 10	SMT	LCL	PVT	Reply	RepReq	Command = [0 .. 63]	Actuator ID = [0 .. 63]	RCI

Field descriptions:

- The LCC is the same as in the ARINC825 definition (2 = NOC – normal operation channel, 5 = TMC – test and maintenance channel).
- FID is the same as in the ARINC825 definition – FID = 10 for actuators
- SMT = 0, LCL = 1, PVT = 0.
- RepReq – Reply request bit. This bit is set by the FCC when the actuator shall reply to a command. [V1.3] If a reply is requested, the reply message shall be sent no later than 2ms. If this bit is cleared (= 0), the actuator shall not send a reply.
- Reply – bit which is set when the actuator is responding to a command. This prevents the actuator sending the same CAN ID as the FCC which might lead to bus collisions
- Command: Specific command to the actuator – value range 0 .. 63
- Actuator ID: Specific actuator to be addressed – value range 0 .. 63
    - o Actuator ID 0 = broadcast; addresses all servos on the bus.
    - o Actuator ID 1 – 63 = individual actuator
- The RCI is the same as in the ARINC825 definition

3 Actuator functionalities

3.1 [V1.3] Blocked actuator detection and protection

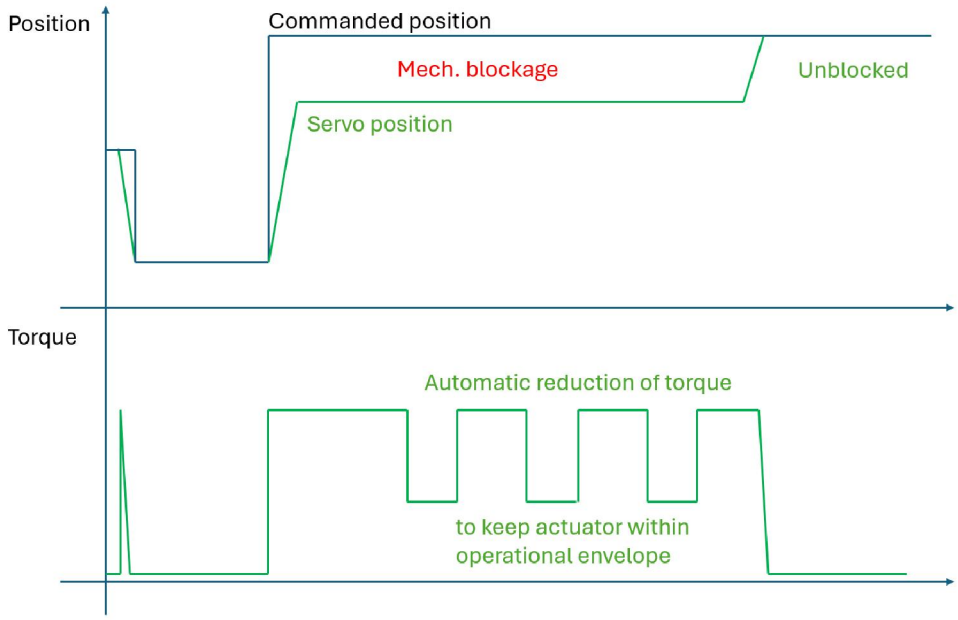
The actuator shall detect internal or external blockages when the commanded position cannot be achieved despite the application of maximum drive/servo motor torque.

Blockage detection can and should consider the commanded position and/or velocity, measured actuator motion, measure and/or estimated motor current or torque, and the duration of the condition. Detection thresholds and timing parameters shall be selected to prevent false triggering during normal transient operation.

Upon detection of a blockage, the actuator shall:

- Set the corresponding blockage fault status and report it through the actuator/servo status bitfield
- Automatically reduce the commanded motor torque or current to a thermally sustainable level (suggestion is to reduce maximum torque after a pre-defined period, e.g. 5s)
- Limit the motor current and power dissipation such that the permitted temperature limits of the motor and internal electronics are not exceeded
- Apply max torque intermittently trying to reach the commanded position for short periods of time

During a blockage, the actuator shall periodically check whether the blockage has been removed by applying a limited recovery torque or a full torque try for a limited period. Such attempts shall remain within the applicable electrical, mechanical, and thermal limits. See following graph for an implementation option/suggestion:



Normal operation may resume when the blockage is no longer detected and the actuator temperature is within its permissible operating envelope (max current, motor / electronics temperature).

The blockage status, torque-limitation status, and any associated overtemperature or overcurrent conditions shall remain visible (as long as they exist) as diagnostic information through the actuator/servo status bitfield and the general servo replies.

4 CAN commands

The CAN commands can be used to control and configure the actuator as well as to read the current state and settings of the actuator.

4.1 Set / write functions

4.1.1 Set position command

Command: Set position of servo lever arm

LCC = 2, FID = 10, SMT = 0, LCL = 1, PVT = 0, Spare = 0, Reply = 0, Command = 1, RCI = 0

	Reply	D7	D6	D5	D4	D3	D2	D1	D0	DLC
Actuator command	0	-	-	-	-	-	-	i16 servo position		2
Actuator Reply	1	u8 servo current	u8 servo voltage	u16 servo status		u8 servo temp	u8 motor torque	i16 actual position		8

Actuator position command and actuator actual position shall be encoded with a minimum resolution of 0.05°. The data type shall be int16 with 0 being the programmed center position of the actuator. The maximum deflection of the actuator shall be defined by the manufacturer of the actuator.

State information:

- [i16] Actual position: Current lever arm deflection position of the actuator
- [u8] Motor torque: Current motor torque of the actuator. Motor torque can be either the actuator torque or the current drive PWM of the actuator motor. Scale can be chosen to be either in percent [%] or in unitless range [1 – 255]. If no motor torque value is available, then this value shall be set to 0.
- [u8] Actuator temperature in [°C] with an offset of -60°C. 1 LSB = 1°C. A temperature of 25°C of the actuator would be represented by a value of 85.
- [u16] Servo status bitfield: Here the manufacturer shall list the internal (digital) states and flags of the actuator that represent warning and errors. Here are some examples of warning- and error flags: Motor temperature high, undervoltage error,

overvoltage error, Firmware integrity error, Torque overload error, system error, … . The availability of warning and error flags is defined by the manufacturer.

• [u8] Servo voltage: Measured supply voltage of the actuator. The voltage range depends on the allowed voltage range of the actuator. Example: When the supply voltage of the actuator is 24V, the scale could be chosen to 1LSB = 0.12V. That would result in a representable voltage range from 0.12V to 30.6V (= 255 * 0.12V). if no voltage measurement is available the value should be set to 0.

• [u8] Servo current: Measured current of the actuator. The current range depends on the maximum current range of the actuator. Example: When the max. load current of the actuator is 1.6A, the scale could be chosen to 1LSB = 0.01A = 10mA. That would result in a representable current range from 0 to 2,55A (= 255 * 10mA).

4.1.2 Set torque command

Command: Set maximum torque of actuator

LCC = 2, FID = 10, SMT = 0, LCL = 1, PVT = 0, Spare = 0, Reply = 0, Command = 2, RCI = 0

	Reply	D7	D6	D5	D4	D3	D2	D1	D0	DLC
Actuator command	0	-	-	-	-	-	-	u16 servo torque		2
Actuator Reply	1	u8 servo current	u8 servo voltage	u16 servo status		u8 servo temp	u8 motor torque	u16 actual torque		8

The torque command shall allow to control the torque (maximum force) of the servo.

A value of 0 shall disable the actuator motor and the level arm shall be able to be moved externally.

A value of 4095 and larger shall enable the maximum force / torque of the servo. The force shall be roughly linearly scaled between 0 (no force) and 4095 (max. force).

4.1.3 Set speed command

Command: Set maximum speed of actuator

LCC = 2, FID = 10, SMT = 0, LCL = 1, PVT = 0, Spare = 0, Reply = 0, Command = 3, RCI = 0

	Reply	D7	D6	D5	D4	D3	D2	D1	D0	DLC
Actuator command	0	-	-	-	-	-	-	u16 servo speed		2
Actuator Reply	1	u8 servo current	u8 servo voltage	u16 servo status		u8 servo temp	u8 motor torque	u16 actual speed		8

The speed command shall allow to control the rotation speed of the actuator.

A value of 1 shall limit the rotation speed of the actuator to 1°/s = 1 degree per second.

Very high values outside the maximum rotation speed of the actuator shall enable the servo to rotate as fast as possible. The maximum speed shall be defined by the manufacturer of the actuator.

4.1.4 Set parameter

Command: Set actuator parameter

LCC = 2, FID = 10, SMT = 0, LCL = 1, PVT = 0, Spare = 0, Reply = 0, Command = 4, RCI = 0

	Reply	D7	D6	D5	D4	D3	D2	D1	D0	DLC
Actuator command	0	-	-	-	-	u16 Param value		u16 Param ID		4
Actuator Reply	1	-	-	-	-	u16 Param value		u16 Param ID		4

The set parameter command shall allow the configuration of the actuator.

Here is a suggested actuator parameter list:

- 0: Actuator ID
  - Value range: [1 .. 63]
- 1: CAN bitrate
  - Value range: 0: 250kbps, 1: 500kbps (default), 2: 1MBit
- 2: Deadband
  - Value range: [0 – 1000]
- 3: Set center position – Current servo position shall be saved as the center position (= 0). Param value shall be ignored by the actuator.
- 4: Controller P gain
  - Value range to be suggested by actuator manufacturer
- 5: Controller I gain
  - Value range to be suggested by actuator manufacturer
- 6: Controller D gain
  - Value range to be suggested by actuator manufacturer
- 7: Actuator temperature limit
  - Value range [0 .. 85]°C. When reaching the max temperature, the actuator shall automatically decrease the actuator torque to prevent motor melting / electronics burn
- 8: to be extended by available parameters of the actuator specified by the manufacturer of the actuator
- [V1.3] 9: Reset center position – Reset the servo's center position to the internal zero position of the actuator

4.1.5 Set user defined data

Command: Set user defined data – max 8 bytes

LCC = 2, FID = 10, SMT = 0, LCL = 1, PVT = 0, Spare = 0, Reply = 0, Command = 10, RCI = 0

	Reply	D7	D6	D5	D4	D3	D2	D1	D0	DLC
Actuator command		Data7	Data6	Data5	Data4	Data3	Data2	Data1	Data0	1-8
Actuator reply	1	Data7	Data6	Data5	Data4	Data3	Data2	Data1	Data0	1-8

The set user defined data command shall allow the writing of up to 8 user defined bytes into non-volatile memory of the actuator.

4.2 Read functions

4.2.1 Get position command

Command: Get position of servo lever arm

LCC = 2, FID = 10, SMT = 0, LCL = 1, PVT = 0, Spare = 0, Reply = 0, Command = 11, RCI = 0

	Reply	D7	D6	D5	D4	D3	D2	D1	D0	DLC
Actuator command	0	-	-	-	-	-	-	-	-	0
Actuator Reply	1	u8 servo current	u8 servo voltage	u16 servo status		u8 servo temp	u8 motor torque	i16 actual position		8

Get actuator position command shall allow to request the position reply from actuator(s) without sending a command first.

If an actuator receives this command with either the actuator ID = 0 (broadcast) or with its own specific ID, the actuator shall send the same reply as to a set position command (see above).

4.2.2 Get torque command

Command: Get maximum torque of actuator

LCC = 2, FID = 10, SMT = 0, LCL = 1, PVT = 0, Spare = 0, Reply = 0, Command = 12, RCI = 0

	Reply	D7	D6	D5	D4	D3	D2	D1	D0	DLC
Actuator command	0	-	-	-	-	-	-	-	-	0
Actuator Reply	1	u8 servo current	u8 servo voltage	u16 servo status		u8 servo temp	u8 motor torque	u16 max torque		8

The get torque command shall allow to get the current torque setting of the actuator(s) without having to send a set torque command.

If an actuator receives this command with either the actuator ID = 0 (broadcast) or with its own specific ID, the actuator shall send the same reply as to a set torque command (see above).

4.2.3 Get speed command

Command: Get maximum speed of actuator

LCC = 2, FID = 10, SMT = 0, LCL = 1, PVT = 0, Spare = 0, Reply = 0, Command = 13, RCI = 0

	Reply	D7	D6	D5	D4	D3	D2	D1	D0	DLC
Actuator command	0	-	-	-	-	-	-	-	-	0
Actuator Reply	1	u8 servo current	u8 servo voltage	u16 servo status		u8 servo temp	u8 motor torque	u16 max speed		8

The get maximum speed command shall allow to get the max speed setting of the actuator(s) without having to send a set maximum speed command.

If an actuator receives this command with either the actuator ID = 0 (broadcast) or with its own specific ID, the actuator shall send the same reply as to a set maximum speed command (see above).

4.2.4 Get parameter

Command: Get actuator parameter

LCC = 2, FID = 10, SMT = 0, LCL = 1, PVT = 0, Spare = 0, Reply = 0, Command = 14, RCI = 0

	Reply	D7	D6	D5	D4	D3	D2	D1	D0	DLC
Actuator command	0	-	-	-	-			u16 Param ID		2
Actuator Reply	1	-	-	-	-	u16 Param value		u16 Param ID		4

The get parameter command shall allow to read the configuration of the actuator.

If an actuator receives this command with either the actuator ID = 0 (broadcast) or with its own specific ID, the actuator shall send the same reply as to a set parameter command (see above).

4.2.5 [V1.3] Get firmware version

Command: Get firmware version of servo firmware– max 8 bytes

LCC = 2, FID = 10, SMT = 0, LCL = 1, PVT = 0, Spare = 0, Reply = 0, Command = 19, RCI = 0

	Reply	D7	D6	D5	D4	D3	D2	D1	D0	DLC
										0
	1	Data7	Data6	Data5	Data4	Data3	Data2	Data1	Data0	1-8

The get user defined data command shall allow the reading of up to 8 firmware version bytes. The firmware version shall fill up the data bytes from the least signification version to the most significant version.

Examples:

FW version 1.2.3.4 → DLC = 4, D0 = 4, D1 = 3, D2 = 2, D3 = 1
FW version 4.2.55 → DLC = 3, D0 = 55, D1 = 2, D2 = 4

4.2.6 Get user defined data

Command: Get user defined data – max 8 bytes

LCC = 2, FID = 10, SMT = 0, LCL = 1, PVT = 0, Spare = 0, Reply = 0, Command = 20, RCI = 0

	Reply	D7	D6	D5	D4	D3	D2	D1	D0	DLC
										0
	1	Data7	Data6	Data5	Data4	Data3	Data2	Data1	Data0	1-8

The get user defined data command shall allow the reading of 8 user defined bytes from the non-volatile memory of the actuator.

5 Code examples

Below are C code examples that help with the definition of CAN ID with C macros.

5.1 CAN ID creation via C macro

```
/* a825_enums.h * *  Created on: 03.03.2017 */
#pragma once

#define FID_ACTUATORS 10

/*** LCC - Logic communication channel ***/
typedef enum LCC
{
        LCC_EEC = 0,                    // Exception event channel
        // RESERVED             = 1,
        LCC_NOC = 2,                    // Normal operations channel
        // RESERVED             = 3,
        LCC_NSC = 4,                    // Node service channel
```

```
        LCC_UDC = 5,            // User defined channel
        LCC_TMC = 6,            // test and maintenance channel
        LCC_FMC = 7,            // CAN base frame migration channel
} LCC;

/*** RCI - Redundance channel ***/
typedef enum RCI
{
        RCI_INVALID = -1,
        RCI0        = 0,
        RCI1        = 1,
        RCI2        = 2,
        RCI3        = 3,
} RCI;

/*** Local flag ***/
typedef enum LCL
{
        LCL_YES = 1,
        LCL_NO = 0
} LCL;

/*** Private flag ***/
typedef enum PVT
{
        PVT_YES = 1,
        PVT_NO = 0,
} PVT;

/*** Masks ***/
typedef enum MASKS
{
        LCC_MASK = 0x0007,
        FID_MASK = 0x007F,
        RSD_MASK = 0x0001,
        LCL_MASK = 0x0001,
        PVT_MASK = 0x0001,
        DOC_MASK = 0x3FFF,
        RCI_MASK = 0x0003,
        // Test and maintenance,
        MID_MASK = 0x07FF,
        MSQ_MASK = 0x003F,
        MSP_MASK = 0x001F,
```

```
} MASKS;

/*** Defines ***/
#define RSD 0x0000

/*** ARINC825 Macros ***/
#define A825(LCC, FID, LCL, PVT, DOC, RCI) (((LCC & LCC_MASK) << 26UL) | ((FID & FID_MASK) << 19UL) | ((RSD & RSD_MASK) << 18UL) | ((LCL & LCL_MASK) << 17UL) | ((PVT & PVT_MASK) << 16UL) | ((DOC & DOC_MASK) << 2UL) | ((RCI & RCI_MASK) << 0UL))

#define FID(ID) ((ID >> 19UL) & FID_MASK)
#define LCC(ID) ((ID >> 26UL) & LCC_MASK)
#define LCL(ID) ((ID >> 17UL) & LCL_MASK)
#define PVT(ID) ((ID >> 16UL) & PVT_MASK)
#define DOC(ID) ((ID >> 2UL) & DOC_MASK)
#define RCI(ID) ((ID >> 0UL) & RCI_MASK)

/*** Example definition of actuator CAN ID ***/
#define CAN_ID_ACT_AIL_LEFT   A825(LCC_NOC, FID_ACTUATORS, LCL_YES, PVT_NO, 1, RCI0)
#define CAN_ID_ACT_AIL_RIGHT  A825(LCC_NOC, FID_ACTUATORS, LCL_YES, PVT_NO, 2, RCI0)
```