ADAMPOWER

# User Manual

## UC57

CANopen

## Integrated Stepper Motor Controller

# Contents

# UC57 CANopen communication

# Integrated stepper motor controller

## 1. Overview

### 1.1 Product introduction

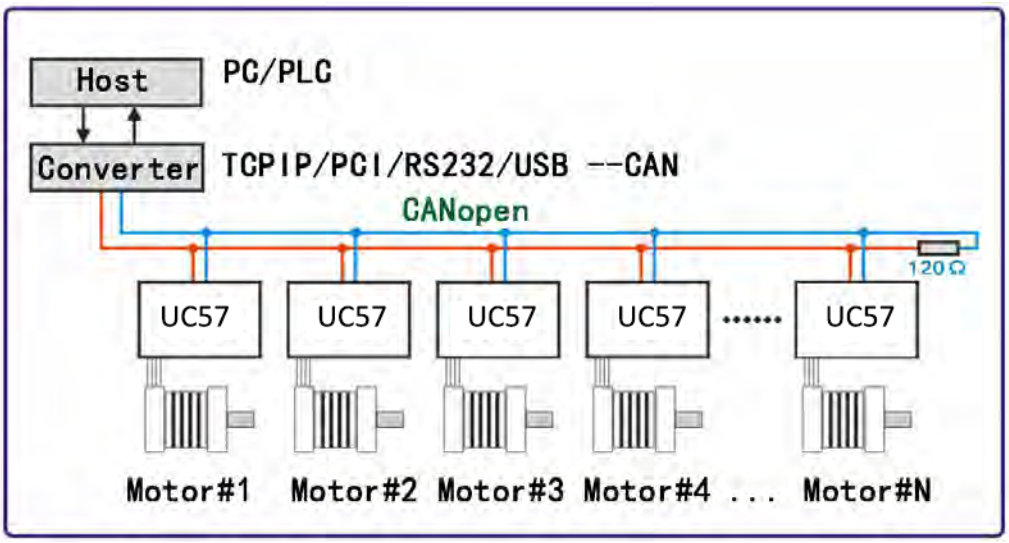
UC57 is a highly integrated stepper motor driver launched by Adampower. It supports the CiA301 and CiA402 sub-protocols of the CANopen protocol. It can be connected to PLC, industrial computer, controller and other host computers with only two communication lines. Through the built-in motion Control instructions can realize a network of up to 100-axis stepper motors.

UC57 uses the latest 32-bit DSP digital chip and has advanced drive control algorithms and noise suppression technology to ensure smooth motor operation. Stable, low noise, and temperature controllable. Users can set any ID address within 1-255 and any current value within 0-8A through the host computer.

Maximum output peak current of the UC57 bus driver is 5.6A. It can be assembled into NEMA17 size integrated stepper motor.

UC57 can be set to 1-256 subdivisions and adopts built-in micro-subdivision technology , which can achieve high subdivision effects even under low subdivision conditions, ensuring that the motor operates with uniform step intervals and no large or small step problems.

3 input signal ports and 2 output signal ports, supporting position, speed, and return-to-origin control modes. with highest communication rate in 1Mbps. especially suitable for long-distance multi-axis applications, which can reduce wiring and enhance the reliability of drive operation.




### 1.2 Product Characteristics

√ New generation 32-bit DSP technology, high cost performance, good stability, low noise and low vibration.

√ Supports CiA301 and CiA402 sub-protocols of CANopen protocol, and can mount up to 100 devices.

√ The current,subdivision and shaft locking current can be set through instructions; control the start and stop of the motor and query the real-time running status.

√ Built-in single-axis controller function: Users can set the starting speed, acceleration time, deceleration time, maximum speed and Parameters, such as the total number of pulses realize the trapezoidal acceleration and deceleration position control function.

√ Support position control, speed control and return-to-origin working modes.

√3-channel photoelectrically isolated programmable input interface, receiving external control signals to realize functions such as enable, start and stop, emergency stop and limit.

√ 2-channel photoelectrically isolated programmable output interface to output driver status and control signals.

√ Current control is smooth and precise, and the motor generates little heat.

√ When the step pulse stops for more than 200ms, the motor current is halved.

√ Excellent stability in low-frequency and small subdivisions, output peak current range 0 - 5.6A.

√ With overvoltage, undervoltage, overcurrent and other protection functions.

### 1.3 Application area

Mainly used in electronic equipment, semiconductors, medical instruments, environmental protection equipment, automatic detection equipment, small automatic processing equipment and other automation equipment with multiple motor shaft applications and compact requirements for equipment space.

## 2. Electrical, Mechanical and environmental

### 2.1 Electrical

| Parameters | UC57 | | | |
|---|---|---|---|---|
| | Min. Value | Typical Value | Max. Value | Unit |
| Continuous output | 0 | - | 5.6 | A |
| Input Power voltage | 20 | 24/36 | 50 | VDC |
| Logic input current | 6 | 10 | 16 | mA |
| Logic input voltage | 36 | 38 | 40 | VDC |
| Pulse frequency | 0 | - | 100 | kHz |
| Resistance | 100 | - | - | MΩ |

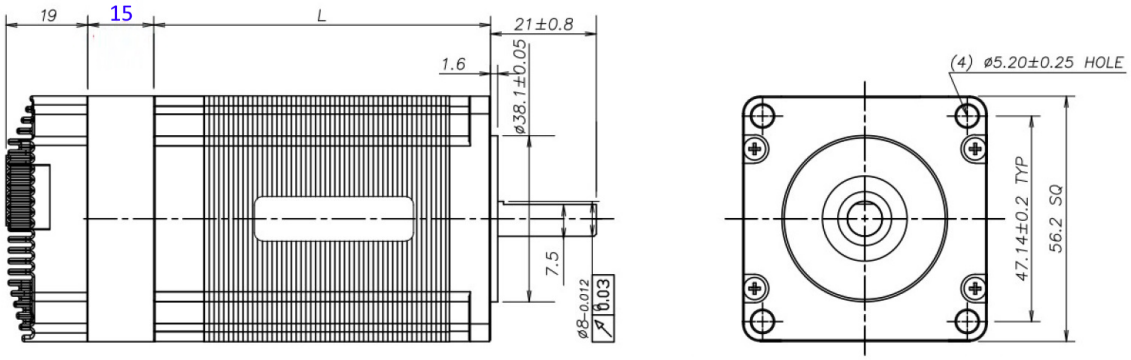
### 2.2 Environment

| Cooling method | Heat sink cooling | |
|---|---|---|
| usage environment | usage occasions | Try to stay away from other heating devices and avoid dust, oil mist, and corrosive gases and Strong vibration areas, prohibition of combustible gases and conductive dust |
| | Temperature | 0 ~ 50℃ |
| | humidity | 40 ~ 90%RH(No condensation) |
| | vibration | 10 ~ 55 Hz/0.15mm |
| Storage temperature | -20 ~ 80 ℃ | |

### 2.3 Dimensions and Motors

UC57 is designed for NEMA23 stepper motor, we supply NEMA23 integrated stepper motor with torque 1.0, 2.0 and 3.0Nm, if just choose UC57 stepper motor controller with requirement for low vibration, please inform us for setting parameters before sale.

| Model No. | Holding Torqure(Nm) | Motor Length(mm) | overall length(mm) | Encoder |
|---|---|---|---|---|
| UC5710IEC | 1.0 | 56 | 75 | 1000 line Magnetic Encoder |
| UC5720IEC | 2.0 | 76 | 95 | |
| UC5730IEC | 3.0 | 90 | 109 | |
| UC5710IEP | 1.0 | 56 | 90 | 1000 line ABZ Optoelectronic encoder |
| UC5720IEP | 2.0 | 76 | 105 | |
| UC5730IEP | 3.0 | 90 | 124 | |



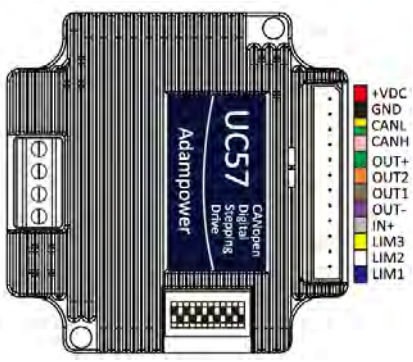

Note: NEMA23 integrated stepper motor with Optoelectronic encoder, the length add 15mm

### 2.4 Installation

When installing the driver, please use an upright side installation to create strong air convection on the surface of the heat sink; If necessary, install a fan near the driver to forcibly dissipate heat and ensure that the driver operates within a reliable operating temperature range (the operating temperature of the driver is usually within 60 °C， and motor is within 80°C)

## 3. Controller port and wiring

### 3.1 Port definition




Control Signal port:

| Port | pin | symbols | Name | Function |
|---|---|---|---|---|
| CN1 | 1 | IN1 | input, default positive limit | Support 5/24VDC control signal, only NPN,OC or C signal input, port function support modification by software |
| | 2 | IN2 | input, default origin | |
| | 3 | IN3 | input, default negative limit | |
| | 4 | IN- | input, signal power negative | signal common negative terminal |
| | 5 | OUT- | output, signal power positive | signal common negative terminal |
| | 6 | OUT1 | output signal, default alarm | high current output, built in pull-up resistor,and freewheeling diode |
| | 7 | OUT2 | output signal, controller status | |
| | 8 | OUT+ | output, signal power negative | signal common positive terminal |
| | 9 | CANH | Communication port | CAN communication port |
| | 10 | CANL | | |
| | 11 | GND | Power supply Ground | Power supply Ground |
| | 12 | VDC | Power Supply 12~ 36VDC | Recommended DC24V |

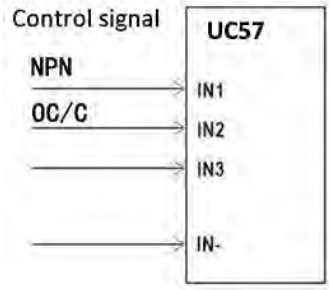
Stepper Motor Port:

| Port | pin | symbols | Name | Function |
|---|---|---|---|---|
| CN2 | 1 | A+ | Stepper Motor phase A & B | Suitable for 2 phase hybrid stepper motor, especially perfect math the NEMA17 size stepper motor. |
| | 2 | A- | | |
| | 3 | B+ | | |
| | 4 | B- | | |

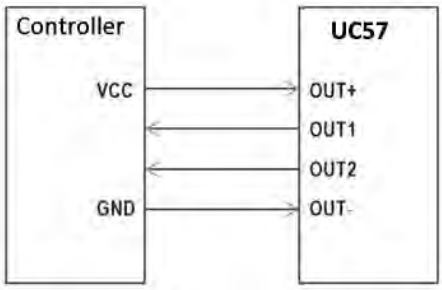
Status indicator light:

| Port | Fault type | number of flashes | Function |
|---|---|---|---|
| CN3 | Normal | 0 | no fault, indicator light is always on |
| | Overcurrent | 1 | Overcurrent, or phase to phase short circuit fault. |
| | Overvoltage | 2 | Overvoltage fault(voltage > 43DVDC) |
| | Hardware | 3 | Hardware malfunction |
| | Phase Loss | 4 | Motor open circuit or poor contact failure |
| | EEPROM error | 5 | EEPROM error |
| | Undervoltage | 6 | Under voltage fault (voltage <9VDC) |
| Note: The indicator light flashes in a cyclic cycle of 3 seconds. When the fault is cleared by the user, the indicator light is always on. | | | |

Wiring diagram:




Input signal wiring diagram




Output signal wiring diagram

# 4. CANopen protocol

## 4.1 CANopen protocol

### 4.1.1 CANbus and CANopen

CAN stands for Controller Area Network, which stands for Controller Area Network. Originally designed by BOSCH, a German company, for monitoring and control of automobiles, the application of CAN is no longer limited to the automotive industry, but has developed into fields such as process industry, mechanical industry, robotics, CNC machine tools, medical devices, and sensors. Compared to other buses, CAN buses have the following characteristics:

1. Multi master control: When the bus is idle, all units can start sending messages. When multiple units start sending messages at the same time, the unit that sends high priority ID messages can obtain the right to send.
2. Communication speed: According to the size of the entire network, a suitable communication speed can be set, and the CAN bus supports communication speeds up to 1Mbit/s.
3. Communication verification: The CAN protocol adopts CRC and can provide corresponding error handling functions to ensure the reliability of data communication.
4. Error detection, notification, and recovery: All units can detect errors, and the unit that detects an error will immediately notify all other units at the same time. At the same time, CAN can identify the type of error, and when a continuous data error occurs on the bus, the unit that caused the fault can be isolated from the bus

The CAN bus communication interface specifies the physical layer and data link layer functions of the CAN protocol, but does not specify the application layer, which is not complete in itself. A high-level protocol is needed to define how to use the 11/29 bit identifier COB-ID and 8-byte data in CAN messages. Therefore,UC57 driver introduces CANopen communication protocol

The CANopen protocol is one of the standards defined by CAN in Automation (CiA) and gained widespread recognition shortly after its release. Especially in Europe, the CANopen protocol is considered a leading standard in CAN based industrial systems.

The CANopen protocol consists of a series of sub protocols, which are further divided into communication sub protocols and device sub protocols.
The communication sub protocol proposes the concept of an object dictionary and defines the objects and parameters of the communication sub protocol area in the object dictionary. Each CANopen device must comply with at least the communication sub protocol, which has been expanded based on different industries or device application fields.
CiA301 is the most basic communication sub protocol, which standardizes the framework of CANopen networks and also defines communication methods and behavioral norms between different CANopen devices.
The UC57 driver supports the CiA 301 communication sub protocol and the CiA 402 device sub protocol for the driver.

### 4.1.1 CANopen Function

**Object Dictionary**

Object Dictionary (OD: Object Dictionary) is the core concept of CANopen, and every CANopen device in the network has an object dictionary.
An object dictionary is an ordered collection of data objects that describe all communication and device parameters of the device, And its position in the object dictionary is determined through a 16 bit index and an 8-bit subindex.

**Message Format**

As the application layer protocol of CAN bus,The CANopen protocol mainly defines the arbitration domain (11 bits) and data domain (up to 8 bytes) in CAN messages.

| 601 | 40 | 60 | 61 | 00 | 00 | 00 | 00 | 00 |
|---|---|---|---|---|---|---|---|---|
| COB-ID | Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 0 | Byte 1 | Byte 2 | Byte3 |

In the CANopen protocol, it is stipulated that the 11 arbitration bits are divided into the upper 4 function codes And the node address of the lower 7 bits (Node ID),Known as COB-ID (Communication Object Identifier)
The structure of the CANopen identifier is shown in the table below, the range of node addresses is from 1 to 127

| CANopen predefined master/slave connection set | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Function Code | | | | Node ID | | | | | | |

The UC57 driver supports the following types of CANopen messages:
1. PDO (Process Data Object, process data object) message
2. SDO (Service Data Object, Service Data Object) message
3. NMT (Network Management Object, Network Management Object) message
4. SYNC (Synchronisation Object, synchronization) message
5.EMCY (Emergency Object, emergency event) message

The following table shows the function codes and corresponding COB-IDs of various messages predefined in the communication sub-protocol CiA301.

| Object | Function Code | COB-ID |
|---|---|---|
| NMT | 0000 | 0h |
| SYNC | 0001 | 80h |
| PDO1（TX） | 0011 | 181h-lFFh |
| PDO1（RX） | 0100 | 201h-27Fh |
| PDO2（TX） | 0101 | 281h-2FFh |
| PDO2（RX） | 0110 | 301h-37Fh |
| PDO3（TX） | 0111 | 381h-3FFh |
| PDO3（RX） | 1000 | 401h-47Fh |
| PDO4（TX） | 1001 | 481h-4FFh |
| PDO4（RX） | 1010 | 501h-57Fh |
| SDO（TX） | 1011 | 581h-5FFh |
| SDO（RX） | 1100 | 601h-67Fh |
| Heart Beat | 1110 | 701h-77Fh |

### Process Data Object (PDO)

The SDO protocol is used to operate object dictionaries, process data with low real-time performance, and data with high real-time requirements, Usually transmitted via PDO.

The PDO communication method is based on the producer/consumer (Producer/Consumer) model. Data is sent from a device (producer),the data is sent to another device (consumer) or many other devices (broadcast mode) and is transmitted in unacknowledged mode.
Data transfers are limited to 1 to 8 bytes. CANopen devices describe two parameters of PDO: communication parameters (Communication Parameter) and Mapping Parameter (Mapping Parameter) to complete receiving or sending.

The UC57 driver supports 4 channels of RPDO and 4 channels of TPDO, and communicates with each PDO communication port in accordance with the CiA 301 sub-protocol, The communication parameters and mapping parameters are described.

### Service Data Object (SDO)

SDO messages are mainly used to access the object dictionary of the device and configure the devices in the CANopen network. SDO pass the communication mode is based on the client/server model, that is, the sent message must be confirmed by the receiver.
The visitor is called the client, and the device where the object dictionary is accessed and responds to read and write requests is called the server.
The protocol stipulates that reading the value of the object dictionary 4ed is called uploading, and modifying the value of the parameter, it is called Download

The UC57 driver supports two transmission methods: the fast SDO protocol and the normal SDO protocol described in CiA301.

### Network Management Object(NMT)

NMT network management is based on the master-slave structure (Master/Slaver). The master station can perform operations on the slave station through NMT messages. After the CANopen device is powered on or reset, the device will first enter the Initialization state. After the program initialization is completed, the device will automatically send a boot-up message and then automatically enter the pre-operation mode

(Pre-Operational) status. After that, the slave device will change between different states according to the NMT message sent by the master station.

### Synchronization Object(SYNC)

The synchronization object SYNC provides a reference clock for the network to synchronize devices in the network.
SYNC belongs to the communication relationship between producers and consumers, SYNC objects are sent by a SYNC producer and can be received by all other devices in the networkSYNC. Assuming that the devices in the network support synchronous PDO function, SYNC can be used to achieve simultaneous actions of multiple devices.The COB-ID of the SYNC message is 0x80, which has a high priority and ensures the normal transmission of SYNC.In addition, SYNC messages may not contain data to reduce the amount of data in SYNC messages.

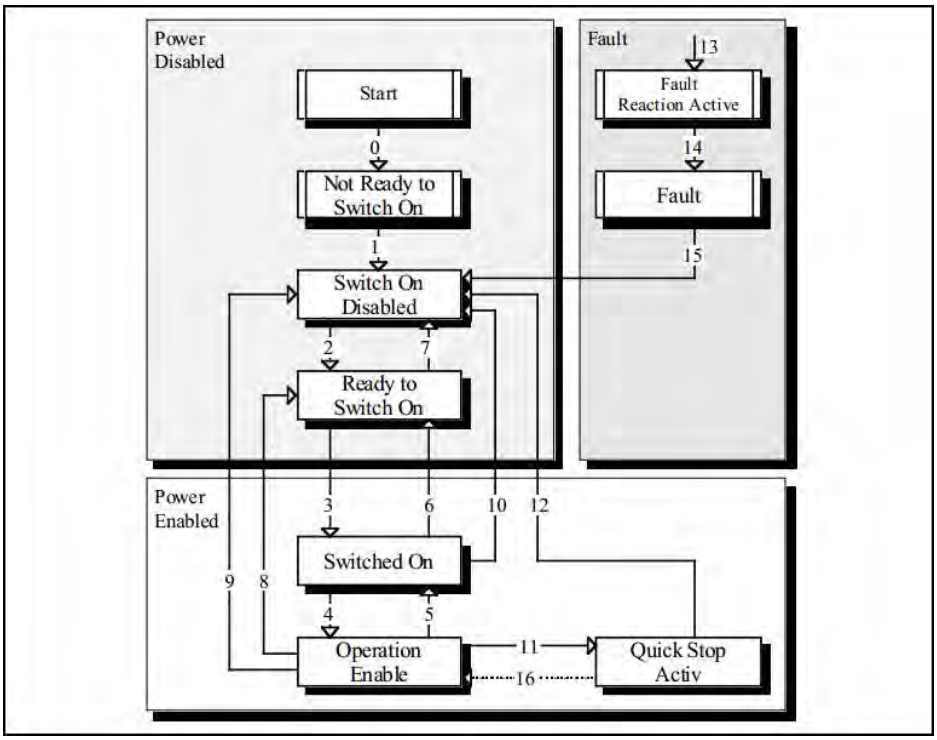
### Emergency Object(EMCY)

The device can report its internal faults to the CANopen network through the emergency object EMCY.EMCY belongs to the producer/consumer communication model, and all devices in the network can consume this message. The EMCY message occupies all 8 bytes of data.
Among them, byte 0 and byte 1 are error codes, which correspond to the device Various types of errors that occur.
Byte 2 is an error register, whose value is stored in unit 1001h of the object dictionary, and corresponds to various types of faults that occur in the device.The contents of bytes 3 to 7 are error fields defined by the manufacturer, which can be specific types of faults.
Through the EMCY object, the master station can easily grasp the specific situation of faults that occur in the slave station.

## 4. 2 CiA402 control protocol

#### 4.2.1 CiA402 Satus

The CiA402 protocol defines the standard state machine for motion control devices, as well as various operating modes and their definitions in the object dictionary.A standard state machine describes the state of a device and the possible control sequence of its drivers. Each step represents a specific internal or external behavior, and the device's state also determines which commands can be received.




| Status | Description |
|---|---|
| Not Ready to Switch on | Power on, the driver initialized and an internal self-test performed, and the brake has also been activated. |
| Switch on Disabled | CANopen communication started, and the SDO communication service used to set the parameters of the driver |
| Ready to Switch on | The driver continues with parameter settings, but the motor is not excited |
| Switch on | The driver motor is ready, and the output stage voltage will be finally connected in this state,<br>But it cannot execute the driving function |
| Operation Enable | The driver motor is enabled, and the driver is operating normally. |
| Quick Stop Active | The quick stop function has been activated, the drive function has been activated, |
| Fault Reaction Active | The driver detected an alarm and stopped according to the set method, but the motor is still enabled |
| Fault | Error occurred, allow change driver's parameters |

**The driver state is controlled by the bit0~bit3, and bit7 bits of the control word (object 6040h),**
**The specific description is shown in the table below:**

| Command | Control Word | | | | | Status switching |
|---|---|---|---|---|---|---|
| | Bit7 | Bit3 | Bit2 | Bit1 | Bit0 | |
| Shutdown | 0 | X | 1 | 1 | 0 | 2,6,8 |
| Switchon | 0 | 0 | 1 | 1 | 1 | 3 |
| Switch on + Enable operation | 0 | 1 | 1 | 1 | 1 | 3+4 |
| Disable voltage | 0 | X | X | 0 | X | 7, 9, 10, 12 |
| Quick stop | 0 | X | 0 | 1 | X | 7, 10, 11 |
| Disable Operation | 0 | 0 | 1 | 1 | 1 | 5 |
| Enable Operation | 0 | 1 | 1 | 1 | 1 | 4,16 |
| Fault reset | 0→1 | X | X | X | X | 15 |

Each state can be displayed through the bit 0~bit 3, bit 5, and bit 6 of the status word (object 6041h), The specific description is shown in the table below:

| Control Word | | | | | | Status |
|---|---|---|---|---|---|---|
| Bit6 | Bit5 | Bit3 | Bit2 | Bit1 | Bit0 | |
| 0 | X | 0 | 0 | 0 | 0 | Not ready to switch on |
| 1 | X | 0 | 0 | 0 | 0 | Switch on disabled |
| 0 | 1 | 0 | 0 | 0 | 1 | Ready to switch on |
| 0 | 1 | 0 | 0 | 1 | 1 | Switched on |
| 0 | 1 | 0 | 1 | 1 | 1 | Operation enabled |
| 0 | 0 | 0 | 1 | 1 | 1 | Quick stop active |
| 0 | X | 1 | 1 | 1 | 1 | Fault reaction active |
| 0 | X | 1 | 0 | 0 | 0 | Fault |

### 4.2.2 Control word and Satus word

The start stop control command and status description of the driver are mainly achieved through the control word 6040h and the status word 6041h, Therefore, it is necessary to proficiently use control words and status words. The following table briefly describes the definitions of each control word and status word.

| Control word | Command | Description |
|---|---|---|
| 6040h | 00 | Initialization 0: the status of the low 4 bits of 6041 is: 0000, motor is released; |
| | 06 | Initialization 1: the status of the low 4 bits of 6041 is: 0001, motor is released; |
| | 07 | Initialization 2: the status of the low 4 bits of 6041 is: 0011, motor is enabled; |
| | 0F | Initialization 3: the status of the low 4 bits of 6041 is: 0111, motor is enabled; |
| | 0F | Start command in speed mode (6061=3); |
| | 0F->1F | Start command in return to origin mode (6061=6); |
| | | Absolute motion start command in position mode (6061=1) |
| | 4F>5F | Relative motion start command in position mode (6061=1) |

| Status word | definition | Function Description |
|---|---|---|
| 6041h | Bit0~Bit 3 | 6040=0：xxxx xxxx xxxx 0000<br>6040=6：xxxx xxxx xxxx 0001<br>6040=7：xxxx xxxx xxxx 0011<br>6040=F：xxxx xxxx xxxx 0111 |
| | Bit7 | 0: Driver is Normal |
| | | 1: Driver alarm |
| | Bit8 | 0: Return to origin is not completed |
| | | 1: Return to origin has been completed |
| | Bit11 | 0: indicate the status of Bit4 of 6040h is 0 |
| | | 1: indicate the status of Bit4 of 6040h is 1 |
| | Bit13 | 0: Motor Release |
| | | 1: Motor Enabled |
| | Bit14 | 0: Motor is stopping |
| | | 1: Motor is running |
| | Bit15 | 0: In Position mode, the movement is not in place. |
| | | 1: In Position mode, the movement is in place |

Example: Initialize the driver after power-on. After initialization, it enters the normal working state.
This operation is generally performed after power-on.

| Main Site | Slave Site | Status word of slave site |
|---|---|---|
| 00: 01 00 | NMT initialization | NMT Initialization |
| 601：2B 40 60 00 00 00 00 00 | 581：60 40 60 00 00 00 00 00 | 6041：xxxx xxxx xxxx 0000 |
| 601：2B 40 60 00 06 00 00 00 | 581：60 40 60 00 00 00 00 00 | 6041：xxxx xxxx xxxx 0001 |
| 601：2B 40 60 00 07 00 00 00 | 581：60 40 60 00 00 00 00 00 | 6041：xxxx xxxx xxxx 0011 |
| 601：2B 40 60 00 0F 00 00 00 | 581：60 40 60 00 00 00 00 00 | 6041：xxxx xxxx xxxx 0111 |

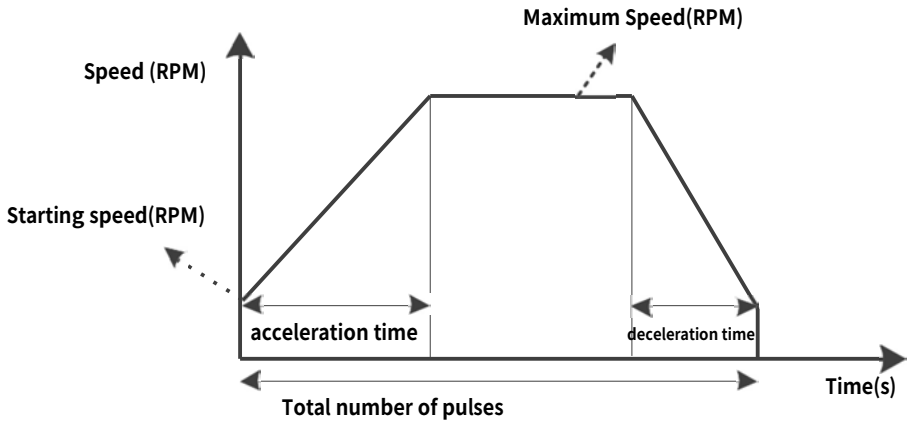
### 4.2.3 Working mode

CANopen sets the operating mode of the driver through the object 6060h (Mode of Operation),
And reflect the current working mode status of the drive through object 6061h (Mode of operation display).
UC57 currently supports three working modes: Profile Position Mode, Profile Velocity Mode, and Homing Mode

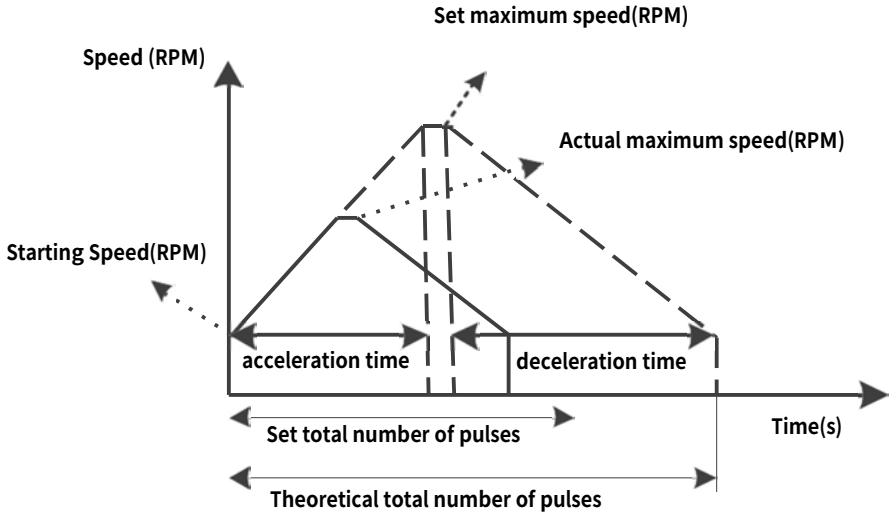
| Index | Subindex | Name | Type | Attribute | PDO Mapping | Parameter Range | Default |
|---|---|---|---|---|---|---|---|
| 606h | 00 | working mode | i8 | RW | YES | 0: undefined<br>1: Profile Position mode<br>2. Profile Velocity mode<br>3. Homing mode | 0 |

### 4.2.4 Profile position mode

The position mode is achieved using a trapezoidal acceleration and deceleration curve,
Users can achieve precise position control by setting the starting speed (address 200E0010h), maximum speed (address 60810020h), acceleration time (address 60830020h), deceleration time (address 60840020h), and total pulse count (address 607A0020h) through the bus.
The trapezoidal acceleration and deceleration curve is shown in the following figure.
Position mode acceleration and deceleration curve:




When the total number of pulses set by the user is small, the motor may need to decelerate before accelerating to the maximum speed (i.e. the motor does not accelerate to the maximum speed set by the user during actual operation), and the speed curve is shown in the following figure.The solid line in the figure shows the actual operating curve of the motor, while the dashed line represents the curve that needs to be run to accelerate to the set maximum speed.
The theoretical total pulse count is the minimum total pulse count calculated based on user set parameters (starting speed, maximum speed, acceleration time, deceleration time).
When the total number of pulses set by the user is less than the theoretical total number of pulses, the motor will run according to the solid line shown in the figure below.




**Position mode acceleration and deceleration curve (not accelerated to the set maximum speed)**

Contents of related object dictionary

| Index | Subindex | Name | Type | Attribute | Range | Default |
|---|---|---|---|---|---|---|
| 6060h | 00 | Working mode | i8 | RW | 0, 1, 3, 6 | 1 |
| 607Ah | 00 | Total number of pulses | i32 | RW | -1000000 ~ 10000000 | 5000 |
| 6081h | 00 | Max. Speed | U32 | RW | 5 ~ 3000 RPM | 120 RPM |
| 6083h | 00 | acceleration time | U32 | RW | 1 ~ 1200 | rps/s |
| 6084h | 00 | deceleration time | U32 | RW | 1 ~ 1200 | rps/s |

Control word and status word

The control words in position mode are controlled through bit4~bit6, and bit8:

| Byte | Name | Value | Description |
|---|---|---|---|
| Bit4 | New set-point | 0 | no target position |
| | | 1 | set target position |
| Bit5 | Change set immediately | 0 | Complete current position before starting next position |
| | | 1 | Interrupt current position and start next position |
| Bit6 | absolute/relative | 0 | target position is absolute value |
| | | 1 | target position is relative value |
| Bit8 | Halt | 0 | End the current position |
| | | 1 | Decelerate and stop through the set deceleration |

Notice:
According to above table.
The absolute position motion command control word is sent as 0x0F ->0x1F,
Send relative position motion command control word as 0x4F ->0x5F.

The bit 10 and bit 15 of the status word display the driver status

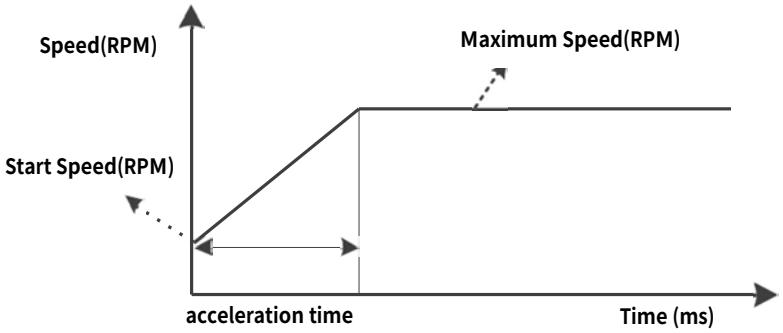
| Byte | Name | Value | Description |
|---|---|---|---|
| Bit10 | Target Reached | 0 | Halt=0, target position has not been reached<br>Halt=1, deceleration |
| | | 1 | Halt=0, target position reached<br>Halt=1, speed is 0. |
| Bit12 | Set-point acknowledge | 0 | Target position to be take into effect |
| | | 1 | Target position takes effect |
| Bit15 | Pend | 0 | Not in place |
| | | 1 | In place |

For example, make the motor move relative to the parameters
(acceleration time 100ms, deceleration time 100ms, maximum speed 60r/min, total pulse count 5000).
Assuming the drive slave station number is 1, the CANopen instruction controls the following table:

| Main Site | Slave Site | Function | Description |
|---|---|---|---|
| 00: 01 00 | Function,depending on PDO mapping | Initialize NMT Status | Initial NMT Status |
| 601: 2B 40 60 00 00 00 00 00 | 581: 60 40 60 00 00 00 00 00 | Initialize servo state | Initialize servo state |
| 601: 23 83 60 00 64 00 00 00 | 581: 60 83 60 00 00 00 00 00 | set acceleration time: | 100 ms |
| 601: 23 84 60 00 64 00 00 00 | 581: 60 84 60 00 00 00 00 00 | set deceleration time: | 100 ms |
| 601: 23 81 60 00 3C 00 00 00 | 581: 60 81 60 00 00 00 00 00 | set maximum speed: | 60RPM |
| 601: 23 7A 60 00 88 13 00 00 | 581: 60 7A 60 00 00 00 00 00 | set number of pulses: | 5000 pulses |
| 601: 2F 60 60 00 01 00 00 00 | 581: 60 60 60 00 00 00 00 00 | switch working mode to : | Position mode |
| 601: 2B 40 60 00 06 00 00 00 | 581: 60 40 60 00 00 00 00 00 | Switch Driver's status Refer to 402 protocol | switch driver status |
| 601: 2B 40 60 00 07 00 00 00 | 581: 60 40 60 00 00 00 00 00 | | |
| 601: 2B 40 60 00 0F 00 00 00 | 581: 60 40 60 00 00 00 00 00 | | |
| 601: 2B 40 60 00 4F 00 00 00 | 581: 60 40 60 00 00 00 00 00 | Send relative motion | command 1 |
| 601: 2B 40 60 00 5F 00 00 00 | 581:60 40 60 00 00 00 00 00 | Send relative motion | command 2 |

### 4.2.5 Profile Velocity mode

Unlike profile position mode, The acceleration curve of the profile velocity mode is shown in the following figure.
The speed mode only requires three parameters to be set: starting speed (address 200E00h), maximum speed (address 608100h), and acceleration time (address 608300h).
After accelerating to the maximum speed according to these three parameter settings,
the motor runs at a constant speed at the maximum speed.
Profile Velocity mode acceleration curve:




Contents of related object dictionary:

| Index | Subindex | Name | Type | Attribute | Range | Default |
|---|---|---|---|---|---|---|
| 6060h | 00 | Working mode | i8 | RW | 0, 1, 3, 6 | 3 |
| 60FFh | 00 | Max. Speed | i32 | RW | -1000~ 1000RPM | 0 |
| 6083h | 00 | acceleration time | U32 | RW | 1 ~ 2000 | rps/s |
| 6084h | 00 | deceleration time | U32 | RW | 1 ~ 2000 | rps/s |

Control word and Status word
In Profile Velocity mode, the Control word is controlled through bit8

| Byte | Name | Value | Description |
|---|---|---|---|
| Bit8 | Halt | 0 | Perform movement |
| | | 1 | Stop motion |

Status word is controlled through bit10, bit12

| Byte | Name | Value | Description |
|---|---|---|---|
| Bit10 | Target Reached | 0 | Halt=0, target position has not been reached<br>Halt=1, deceleration |
| | | 1 | Halt=0, target position reached<br>Halt=1, speed is 0. |
| Bit12 | Speed | 0 | Speed is not 0. |
| | | 1 | Speed is 0. |

For example, making the motor follow the parameters Rotate with an acceleration time of 100ms, deceleration: 100ms, and a maximum speed of 60RPM.
Assuming the drive slave station number is 1, the CANopen instruction controls the following table:

| Main Site | Slave Site | Function | Description |
|---|---|---|---|
| 00: 01 00 | Function,depending on PDO mapping | Initialize NMT Status | Initial NMT Status |
| 601: 2B 40 60 00 00 00 00 00 | 581: 60 40 60 00 00 00 00 00 | Initialize servo state | Initialize servo state |
| 601: 23 83 60 00 64 00 00 00 | 581: 60 83 60 00 00 00 00 00 | set acceleration time: | 100 ms |
| 601: 23 84 60 00 64 00 00 00 | 581: 60 84 60 00 00 00 00 00 | set deceleration time: | 100 ms |
| 601: 23 FF 60 00 3C 00 00 00 | 581: 60 81 60 00 00 00 00 00 | set maximum speed: | 60RPM |
| 601: 2F 60 60 00 03 00 00 00 | 581: 60 60 60 00 00 00 00 00 | switch working mode to: | Velocity mode |
| 601: 2B 40 60 00 06 00 00 00 | 581: 60 40 60 00 00 00 00 00 | Switch Driver's status<br>Refer to 402 protocol | Switch driver status |
| 601: 2B 40 60 00 07 00 00 00 | 581: 60 40 60 00 00 00 00 00 | | |
| 601: 2B 40 60 00 0F 00 00 00 | 581: 60 40 60 00 00 00 00 00 | | |

### 4.2.6 Homing mode

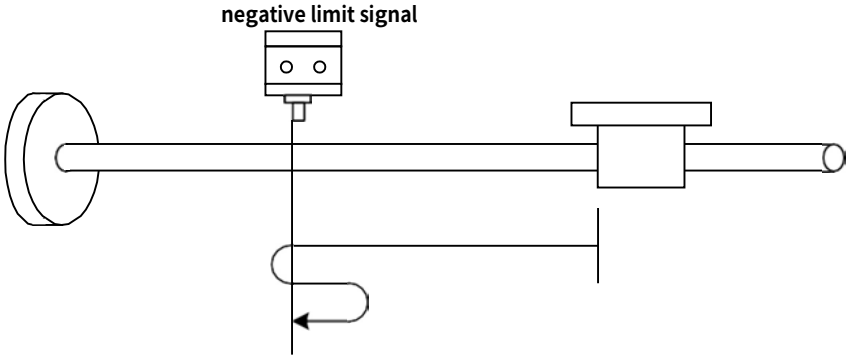
In the Homing mode, the origin signal needs to be connected to the input end of the driver, and the driver completes the search and positioning of the mechanical origin internally.
At the same time, the working mode object 6060h value needs to be set to 6,
When the working mode status object 6061h register is read as 6, operations related to HM working mode can be performed.
The objects involved in this mode are as follows

Related object dictionary content:

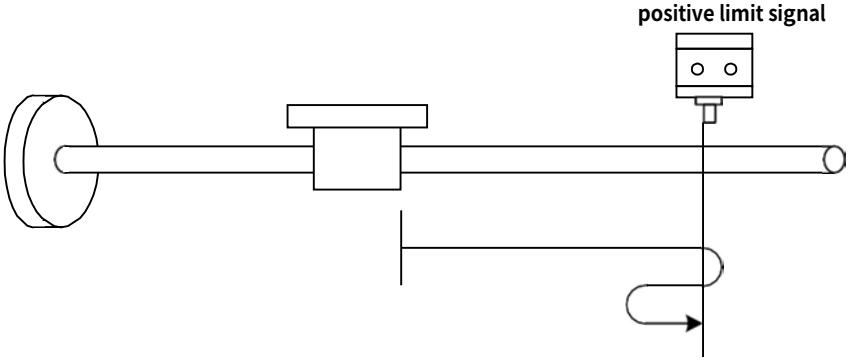
| Index | Subindex | Name | Type | Attribute | Range | Default |
|---|---|---|---|---|---|---|
| 6060h | 00 | Working mode | i8 | RW | 0, 1, 3, 6 | 6 |
| 6098h | 00 | Homing Mode | U8 | RW | 17: negative limit mode<br>18: positive limit mode<br>24: forward homing<br>29: reverse homing<br>41: reverse homing, stop<br>42: forward homing, stop | 0 |
| 6099h | 01 | Homing Speed | U32 | RW | 1~1000RPM | 120 RPM |
| 6099h | 02 | query homing speed | U32 | RW | 1 ~ 1000RPM | 60RPM |
| 609Ah | 00 | acceleration time | U32 | RW | 1 ~ 2000 | rps/s |
| 607Ch | 00 | Homing offset compensation | i32 | RW | -0x7FFFFFFF ~0x7FFFFFFF | 0 |

The drive supports 6 homing modes:

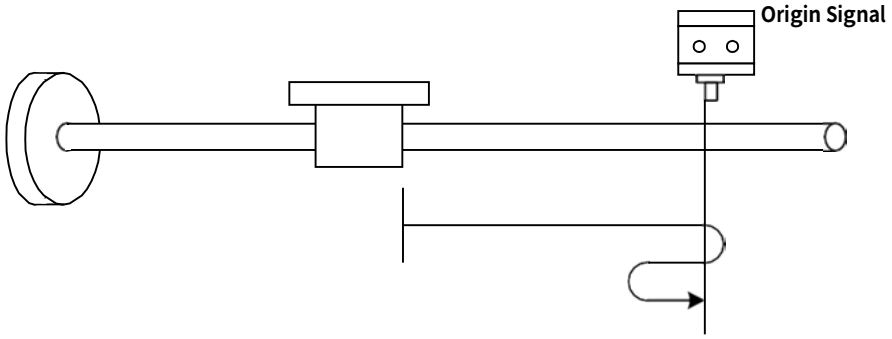
1. 6098h=17: Look for the negative limit signal in the opposite direction.
When encountering the negative limit signal, slow down and stop, then move back a certain distance,
After that, slowly reverse the search for the negative limit signal, find it, stop, and complete the return to the origin action



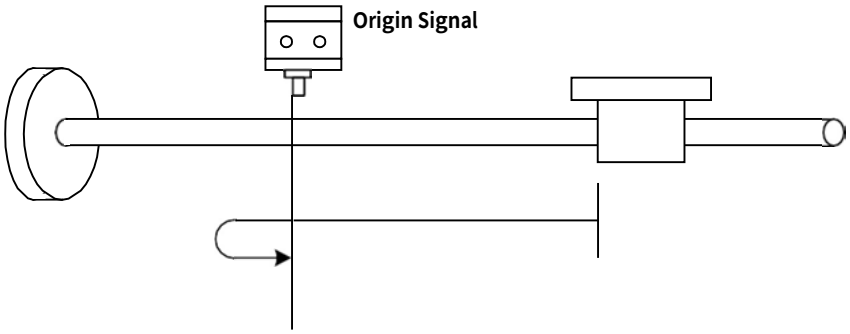

2. 6098h=18: Look for the positive limit signal in the positive direction.
When encountering the positive limit signal, slow down and stop, then move back a certain distance,
Afterwards, search for the positive limit signal in a slow forward direction, stop after finding it, and complete the return to the origin action.



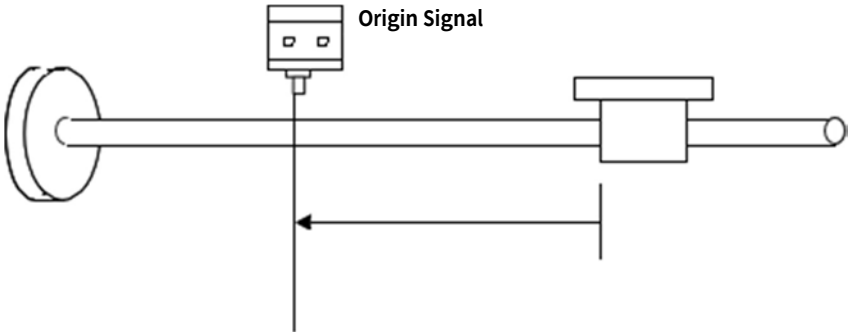
3. 6098h=24: Find the origin in the positive direction.
When encountering the origin, slow down and stop, then retreat a certain distance,
Afterwards, search for the origin in a slow forward direction, stop when found, and complete the return to origin action.
When encountering the positive limit during the process of returning to the origin, the reverse motion continues to search for the origin.



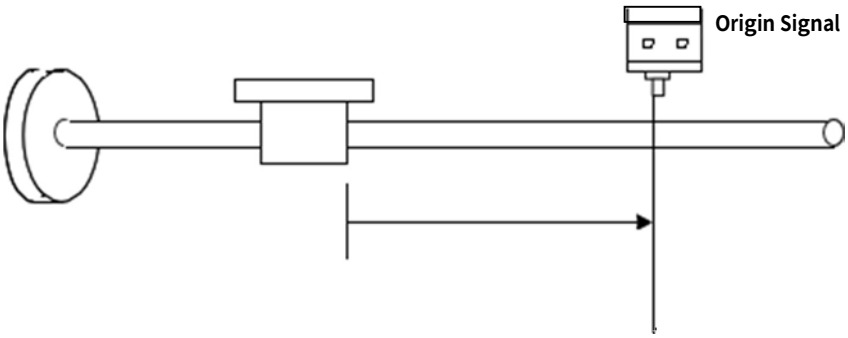

4. 6098h=29: Find the origin in the opposite direction. When encountering the origin, slow down and stop,
Afterwards, search for the origin in a slow forward direction, stop when found, and complete the return to origin action.
When encountering a negative limit during the process of returning to the origin, reverse motion continues to search for the origin




5. 6098h=41: Find the origin in the opposite direction.
When encountering the origin, stop immediately and complete the homing action. If negative limit is encountered, stop.




6. 6098h=42: Find the origin in the positive direction.
When encountering the origin, stop immediately and complete the return to the origin action.
If negative limit is encountered, stop.



609901h object as origin search speed, The object at 609902h has a slow search speed for the origin, and the lower the value at 6009902h, the higher the search accuracy, In addition, if it is necessary to set the origin position to a position that deviates from the mechanical origin, it can be achieved by setting the origin offset of 607Ch.

The control word in homing mode is controlled through bit4 and bit8

| Byte | Name | Value | Description |
|---|---|---|---|
| Bit4 | Homing operation start | 0 | Homing mode is not active |
| | | 0 → 1 | start Homing mode |
| | | 1 | Homing mode is actived |
| | | 1 → 0 | Interrupt Homing mode |
| Bit8 | Halt | 0 | Execute the command of bit4 |
| | | 1 | Stop the motor through homing acceleration |

Note: According to the table above ,
the control word for returning to the origin command is sent as 0x0F ->0x1F.

The bit 8 and bit 10 of the status word display the drive status

| Byte | Name | Value | Description |
|---|---|---|---|
| Bit8 | Homing attained | 0 | Homing mode is not completed |
| | | 1 | Homing mode is completed |
| Bit10 | Target Reached | 0 | Halt=0, target position has not been reached<br>Halt=1, deceleration |
| | | 1 | Halt=0, target position reached<br>Halt=1, speed is 0. |

For example, to complete the work of returning to the origin, select the forward limit+origin mode
to return to the origin mode, Return to origin speed of 120RPM,
The query speed for returning to the origin is 60RPM, with an acceleration and deceleration time of 100ms.
The origin is not compensated.
Assuming the drive slave station number is 1, the CANopen instruction controls the following table:

| Main Site | Slave Site | Function | Description |
|---|---|---|---|
| 00: 01 00 | Function,depending on PDO mapping | Initialize NMT Status | Initial NMT Status |
| 601: 2B 40 60 00 00 00 00 00 | 581: 60 40 60 00 00 00 00 00 | Initialize servo state | Initialize servo state |
| 601: 2F 98 60 00 64 00 00 00 | 581: 60 98 60 00 00 00 00 00 | set to Homing mode, | set as Homing mode |
| 601: 23 99 60 00 78 00 00 00 | 581: 60 99 60 00 00 00 00 00 | set homing speed is: | 120RPM |
| 601: 23 99 60 01 3C 00 00 00 | 581: 60 99 60 01 00 00 00 00 | set homing query speed: | 60RPM |
| 601: 23 9A 60 00 64 00 00 00 | 581: 60 9A 60 00 00 00 00 00 | set acceleration/deceleration | time: 100ms |
| 601: 2F 60 60 00 06 00 00 00 | 581: 60 60 60 00 00 00 00 00 | Switch working mode: | Homing mode |
| 601: 2B 40 60 00 06 00 00 00 | 581: 60 40 60 00 00 00 00 00 | Refer to 402 protocol | Switch driver status |
| 601: 2B 40 60 00 07 00 00 00 | 581: 60 40 60 00 00 00 00 00 | | |
| 601: 2B 40 60 00 0F 00 00 00 | 581: 60 40 60 00 00 00 00 00 | | |
| 601: 2B 40 60 00 1F 00 00 00 | 581: 60 40 60 00 00 00 00 00 | send Homing command | Send Homing Command |

# 5. Object Dictionary

**The UC57 CANopen type stepper motor driver parameter register consists of three parts, namely the 1000h~1FFFh register defined by CIA301, the 2000h~2FFFh register customized by the manufacturer, and the 6000h~6FFFh register defined by CIA402. The 1000h~1FFFh register is the basic communication parameter related to CANopen defined by CIA301, including SDO, PDO, and mapping registers;**

**The 2000h~2FFFh register is a register content customized by the manufacturer, including subdivision and current modification, which can be achieved within this set of parameters;**

**The 6000h~6FFFh registers are motion parameters related to motion control defined by CIA402, including position mode and speed Mode, Homing mode and other operating mode registers, as well as related motion parameter registers.**

| Index | Subindex | Name | Description | Type | Attribute | PDO Mapping | Default Value |
|---|---|---|---|---|---|---|---|
| CiA 301 basic communicatioon parameters | | | | | | | |
| 1000h | 00 | Device Type | Support CIA301、CIA402 protocol | U32 | RO | NO | 0x00040192 |
| 1001h | 00 | Error register | Driver current error status<br>Bit0：generic error, such as phase loss<br>Bit1：current<br>Bit2：voltage<br>Bit3：temperature<br>Bit4：communication error<br>Bit5：device profile specific<br>Bit6：Reserved<br>Bit7：manufacturer specific | U8 | RO | NO | 0 |
| 1003h | 00 | number of subindex | | U8 | RO | NO | 8 |
| | 01~08 | Error memory | Recent errors that have caused emergency messages, supporting 5 error storage units; | U32 | RO | NO | 0 |
| 1009h | 00 | Hardware version | | U16 | RO | NO | 1.0 |
| 100Ah | 00 | Software version | | U16 | RO | NO | 1.0 |
| 1014h | 00 | Emergency message COB | EMNC Emergency message COB | U32 | RW | NO | 0x80 |
| 1017h | 00 | Producer heartbeat interval: ms | | U16 | RW/S | NO | 0 |
| 1018h | 00 | Manufacturer | Subindex | U8 | RO | NO | 4 |
| | 01 | Vendor ID | | U32 | RO | NO | 0x00000322 |
| | 02 | Product Code | | U32 | RO | NO | 0x00000000 |
| | 03 | REVISION_NR | | U32 | RO | NO | 0x00000000 |
| | 04 | SERIAL_NR | | U32 | RO | NO | 0x00000000 |
| 1200h | 00 | number of subindex | | U8 | RO | NO | 2 |
| | 01 | COB-ID | COB-ID（received from slave） | U32 | RO | NO | 600h+Node- ID |
| | 02 | COB-ID | COB-ID（Setn from slave） | U32 | RO | NO | 580h+Node- ID |
| 1400h | 00 | number of subindex | | U8 | RO | NO | 5 |
| | 01 | RPDO0-COB-ID | Identifier COB-ID | U32 | RO | NO | 200+Node- ID |
| | 02 | Transmission | | U8 | RW/S | NO | FFh |
| | 03 | Prohibited time | | U16 | RW/S | NO | 0 |
| | 05 | Event Timer | | U16 | RW/S | NO | 0 |
| 1401h | 00 | number of subindex | | U8 | RO | NO | 5 |
| | 01 | RPDO1-COB-ID | Identifier COB-ID | U32 | RO | NO | 300+Node- ID |
| | 02 | Transmission | | U8 | RW/S | NO | FFh |
| | 03 | Prohibited time | | U16 | RW/S | NO | 0 |
| | 05 | Event Timer | | U16 | RW/S | NO | 0 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1402h | 00 | number of subindex | | U8 | RO | NO | 5 |
| | 01 | RPDO1-COB-ID | Identifier COB-ID | U32 | RO | NO | 400+Node- ID |
| | 02 | Transmission | | U8 | RW/S | NO | FFh |
| | 03 | Prohibited time | | U16 | RW/S | NO | 0 |
| | 05 | Event Timer | | U16 | RW/S | NO | 0 |
| 1403h | 00 | number of subindex | | U8 | RO | NO | 5 |
| | 01 | RPDO1-COB-ID | Identifier COB-ID | U32 | RO | NO | 500+Node- ID |
| | 02 | Transmission | | U8 | RW/S | NO | FFh |
| | 03 | Prohibited time | | U16 | RW/S | NO | 0 |
| | 05 | Event Timer | | U16 | RW/S | NO | 0 |
| 1600h | 00 | number of subindex | | U8 | RO | NO | 3 |
| | 01 | RPDO0-Map 1 | mapped to 6040h register | U32 | RW/S | NO | 60400010h |
| | 02 | RPDO0-Map 2 | mapped to 60FFh register | U32 | RW/S | NO | 60FF0020h |
| | 03 | RPDO0-Map 3 | mapped to 6060h register | U32 | RW/S | NO | 60600008h |
| | 04 | RPDO0-Map 4 | Unmapped | U32 | RW/S | NO | - |
| 1601h | 00 | number of subindex | | U8 | RO | NO | 2 |
| | 01 | RPDO0-Map 1 | mapped to 607Ah register | U32 | RW/S | NO | 607A0020h |
| | 02 | RPDO0-Map 2 | mapped to 6081h register | U32 | RW/S | NO | 60810020h |
| | 03 | RPDO0-Map 3 | Unmapped | U32 | RW/S | NO | - |
| | 04 | RPDO0-Map 4 | Unmapped | U32 | RW/S | NO | - |
| 1602h | 00 | number of subindex | | U8 | RO | NO | 0 |
| | 01 | RPDO0-Map 1 | Unmapped | U32 | RW/S | NO | - |
| | 02 | RPDO0-Map 2 | Unmapped | U32 | RW/S | NO | - |
| | 03 | RPDO0-Map 3 | Unmapped | U32 | RW/S | NO | - |
| | 04 | RPDO0-Map 4 | Unmapped | U32 | RW/S | NO | - |
| 1603h | 00 | number of subindex | | U8 | RO | NO | 0 |
| | 01 | RPDO0-Map 1 | Unmapped | U32 | RW/S | NO | - |
| | 02 | RPDO0-Map 2 | Unmapped | U32 | RW/S | NO | - |
| | 03 | RPDO0-Map 3 | Unmapped | U32 | RW/S | NO | - |
| | 04 | RPDO0-Map 4 | Unmapped | U32 | RW/S | NO | - |
| 1800h | 00 | number of subindex | | U8 | RO | NO | 5 |
| | 01 | RPDO1-COB-ID | Identifier COB-ID | U32 | RO | NO | 180+Node- ID |
| | 02 | Transmission | | U8 | RW/S | NO | FFh |
| | 03 | Prohibited time | | U16 | RW/S | NO | 0 |
| | 05 | Event Timer | | U16 | RW/S | NO | 0 |
| 1801h | 00 | number of subindex | | U8 | RO | NO | 5 |
| | 01 | RPDO1-COB-ID | Identifier COB-ID | U32 | RO | NO | 280+Node- ID |
| | 02 | Transmission | | U8 | RW/S | NO | FFh |
| | 03 | Prohibited time | | U16 | RW/S | NO | 0 |
| | 05 | Event Timer | | U16 | RW/S | NO | 0 |
| 1802h | 00 | number of subindex | | U8 | RO | NO | 5 |
| | 01 | RPDO2-COB-ID | Identifier COB-ID | U32 | RO | NO | 380+Node-ID |
| | 02 | Transmission | | U8 | RW/S | NO | FFh |
| | 03 | Prohibited time | | U16 | RW/S | NO | 0 |
| | 05 | Event Timer | | U16 | RW/S | NO | 0 |
| 1803h | 00 | number of subindex | | U8 | RO | NO | 5 |
| | 01 | RPDO3-COB-ID | Identifier COB-ID | U32 | RO | NO | 480+Node-ID |
| | 02 | Transmission | | U8 | RW/S | NO | FFh |
| | 03 | Prohibited time | | U16 | RW/S | NO | 0 |
| | 05 | Event Timer | | U16 | RW/S | NO | 0 |
| 1A00h | 00 | number of subindex | | U8 | RO | NO | 3 |
| | 01 | RPDO0-Map 1 | mapped to 6041h register | U32 | RW/S | NO | 60410010h |
| | 02 | RPDO0-Map 2 | mapped to 60FDh register | U32 | RW/S | NO | 60fd0020h |
| | 03 | RPDO0-Map 3 | mapped to 606h register | U32 | RW/S | NO | 60610008h |
| | 04 | RPDO0-Map 4 | Unmapped | U32 | RW/S | NO | - |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1A01h | 0 | number of subindex | | U8 | RO | NO | 0 |
| | 1 | RPDO0-Map 1 | mapped to 6064h register | U32 | RW/S | NO | 60640020h |
| | 2 | RPDO0-Map 2 | mapped to 606Ch register | U32 | RW/S | NO | 606c0020h |
| | 3 | RPDO0-Map 3 | Unmapped | U32 | RW/S | NO | - |
| | 4 | RPDO0-Map 4 | Unmapped | U32 | RW/S | NO | - |
| 1A02h | 0 | number of subindex | | U8 | RO | NO | 0 |
| | 1 | RPDO0-Map 1 | Unmapped | U32 | RW/S | NO | - |
| | 2 | RPDO0-Map 2 | Unmapped | U32 | RW/S | NO | - |
| | 3 | RPDO0-Map 3 | Unmapped | U32 | RW/S | NO | - |
| | 4 | RPDO0-Map 4 | Unmapped | U32 | RW/S | NO | - |
| 1A03h | 0 | number of subindex | | U8 | RO | NO | 0 |
| | 1 | RPDO0-Map 1 | Unmapped | U32 | RW/S | NO | - |
| | 2 | RPDO0-Map 2 | Unmapped | U32 | RW/S | NO | - |
| | 3 | RPDO0-Map 3 | Unmapped | U32 | RW/S | NO | - |
| | 4 | RPDO0-Map 4 | Unmapped | U32 | RW/S | NO | - |

## 5.1 Defined Object Dictionary

**The UC57 related object dictionary is shown in the table below:**

| Index | Subindex | Name | Description | Type | Attribute | PDO Mapping | Default Value |
|---|---|---|---|---|---|---|---|
| Defined Object Dictionary | | | | | | | |
| 2000H | 00 | Node ID | Set through 0x2008 register | U16 | RO | YES | - |
| 2001H | 00 | Motor Status | 0：Motor is stationary；<br>1：Motor is running； | U16 | RO | YES | 0 |
| 2003H | 00 | Input signal status | 3 input signal status Bit0~Bit2：<br>X0~X2 input level status； | U16 | RO | YES | 0 |
| 2004H | 00 | output signal status | 2 output signal status Bit0~Bit1：<br>Y0~Y1 output status | U16 | RO | YES | 0 |
| 2005H | 00 | Set Current | 1~25（*0.1A RMS） | U16 | RW/S | YES | 15 |
| 2006H | 00 | subdivision setting | 0—200（Pu/rev）；<br>1—400 （Pu/rev）；<br>2—800 （Pu/rev）；<br>3—1600（Pu/rev）；<br>4—3200（Pu/rev）；<br>5—6400（Pu/rev）；<br>6—12800（Pu/rev）；<br>7—25600（Pu/rev）；<br>8—1000 （Pu/rev）；<br>9—2000（Pu/rev）；<br>10—4000（Pu/rev）；<br>11—5000（Pu/rev）；<br>12—8000（Pu/rev）；<br>13—10000（Pu/rev）；<br>14—20000（Pu/rev）；<br>15—40000（Pu/rev）； | U16 | RW/S | YES | 13 |
| 2007H | 00 | Lock Current | 0：automatic half； 1：full； | U16 | RW/S | YES | 0 |
| 2008H | 00 | Define Node Id | 1~127； | U16 | RW/S | YES | 1 |
| 2009H | 00 | Baud rate setting | 0: 50 kbps<br>1:100 kbps<br>2:125 kbps<br>3:250 kbps<br>4:500 kbps<br>5: 1000kbps | U16 | RW/S | YES | 2 |

| Index | Subindex | Name | Description | Type | Attribute | PDO Mapping | Default Value |
|---|---|---|---|---|---|---|---|
| Defined Object Dictionary | | | | | | | |
| 200AH | 00 | Stop at Limit | 0: Stop, 1: Emergency stop, 2: invalid | U16 | RW/S | YES | 0 |
| 200BH | 00 | Control method | 0：Bus control | U16 | RW/S | YES | 0 |
| 200CH | 00 | Function code value write to EEPROM | 0 ： RW/S attribute write to EEPROM；<br>1：No | U16 | RW | YES | 1 |
| 200EH | 00 | Start Speed | Range 0-1000RPM | U16 | RW | YES | 10 |
| 200FH | 00 | Motor Status | 0: release, 1: enable, 2:invalid | U16 | RW | YES | 2 |
| 2010H | 00 | Register parameter setting | 0: invalid<br>1: Factory default<br>2: Save all RW attributer toEEPROM | U16 | RW | YES | 0 |
| 2011h | 00 | Fault reset | 0: Invalid | U16 | RW | YES | 0 |
| 2012h | 00 | Zero current position | Clear the current position in absolute position mode.<br>0: Invalid<br>1: Clear the current position to zero | U16 | RW | YES | 0 |
| 2030h | 00 | number of subindex | | U16 | RO | NO | 16 |
| | 01 | effective level of input port | Bit0: Input X0 control position;<br>Bit1: Input X1 control position;<br>Bit2: Input X2 control position;<br>Bit3~Bit15: reserved;<br><br>0: Default;<br>1: Level reversal;<br><br>The default input level: rising edge or high level is valid for this driver | U16 | RW/S | YES | 0 |
| | 02 | X0 Function selection | | U16 | RW/S | YES | |
| | 03 | X1 Function selection | | U16 | RW/S | YES | 0 |
| | 04 | X2 Function selection | | U16 | RW/S | YES | 0 |
| | 0C | effective level of output port | Bit0: Output Y0 control position<br>Bit1: Output Y1 control position<br>0: Default<br>1: Level reversal | U16 | RW/S | YES | 0 |
| | 0D | Y0 Function selection | 0: Undefined<br>1: Alarm signal;<br>2: Driver status signal;<br>3: Return to origin completion signal;<br>4: In place signal;<br>5: Output high level<br>6: Output low level | U16 | RW/S | YES | 0 |
| | 0E | Y1 Function selection | | U16 | RW/S | YES | 0 |
| 2040H | 00 | Current loop proportional coefficient | Unit: 0.1%<br>Factory default, normally no need adjustment<br>Range: 10-500 | U16 | RW/S | YES | 220 |
| 2041H | 00 | Current loop integral gain | Unit: 0.1%<br>Factory default, normally no need adjustment<br>Range: 1-500 | U16 | RW/S | YES | 3 |
| 2FF7h | 00 | Running time | Device power on timing | U32 | RO | YES | |

### 5.2 CiA402 Object Dictionary

**The UC57 related CiA402 object dictionary is shown in the table below:**

| Index | Subindex | Name | Description | Type | Attribute | PDO Mapping | Default Value |
|---|---|---|---|---|---|---|---|
| CiA402 Object Dictionary | | | | | | | |
| 603Fh | 0 | Driver fault codes | Bit0: overcurrent;<br>Bit1: overvoltage;<br>Bit2: Operational amplifier failure;<br>Bit3: phase loss;<br>Bit4: EEPROM read and write error;<br>Bit5: Undervoltage | U16 | RO | YES | 0 |
| 6040h | 0 | control word | | U16 | RW | YES | 0 |
| 6041h | 0 | status word | | U16 | RO | YES | 0 |

| Index | Subindex | Name | Description | Type | Attribute | PDO Mapping | Default Value |
|---|---|---|---|---|---|---|---|
| CiA402 Object Dictionary | | | | | | | |
| 605Dh | 00 | Halt control | 0: Stop, maintain enable status;<br>1: Emergency stop, maintain enable status; | I16 | RW | NO | 0 |
| 6060h | 00 | Working mode | 0: Undefined;<br>1: Profile position mode;<br>3: Profile velocity mode;<br>6: Homing mode; | I8 | RW | YES | 0 |
| 6061h | 00 | Working mode status | 0: Undefined;<br>1: Profile position mode;<br>3: Profile velocity mode;<br>6: Homing mode; | I8 | RO | YES | 0 |
| 6064h | 00 | Driver actual position, unit: pulses | | I32 | RO | YES | 0 |
| 606Ch | 00 | The motor's current speed, unit: RPM | | I32 | RO | YES | 0 |
| 607Ah | 00 | Total number of pulses | The total number of pulses in Profile position mode operation (including the total number of steps in acceleration, constant speed, and deceleration stages)<br>Scope: -1000000~1000000； | I32 | RW | YES | 5000 |
| 607Ch | 00 | Origin compensation value, -1000000~1000000 P | | I32 | RW | YES | 0 |
| 6081h | 00 | Max. Speed | profile position mode,<br>speed range: 5 ~ 1000RPM | U32 | RW | YES | 60RPM |
| 6083h | 00 | Acccerlation time: 1~ 2000rps/s | | U32 | RW | YES | 50 |
| 6084h | 00 | Deceleratoin time: 1~ 2000rps/s | | U32 | RW | YES | 50 |
| 6098h | 00 | Homing mode | 17：negative limit mode<br>18：positive limit mode<br>24: Forward origin mode;<br>29: Reverse origin mode;<br>41: Finding the origin in the opposite direction - stop immediately<br>42: Finding the origin in the positive direction - stop immediately | U8 | RW | YES | 0 |
| 6099h | 00 | number of subindex | | U8 | RO | NO | 2 |
| | 01 | Homing high Speed range: 1-1000RPM； | | U32 | RW | YES | 120RMP |
| | 02 | Homing low Speed range: 1-1000RPM； | | U32 | RW | YES | 60RMP |
| 609Ah | 00 | homing acccerlation/deceleratoin time: 1~ 2000rps/s | | U32 | RW | YES | 50 |
| 60FDh | 00 | status of the input port | Bit0: Negative limit state;<br>Bit1: Positive limit state;<br>Bit2: Origin state;<br>Bit3~Bit15: reserved;<br>Bit16~Bit31: user-defined; | U32 | RO | YES | 0 |
| 60FFh | 00 | Max. Speed | profile velocity mode,<br>speed range: -1000 ~ 1000RPM | I32 | RW | YES | 120 |

## 6. Troubleshooting and Alarm

**The UC57 driver has four types of alarm messages,**

**After the driver alarms, the alarm indicator light flashes several times according to different alarm codes,**

**The specific alarm codes and processing methods are shown in the table below**

| Error code | Fault information | Indicator flashes | reset |
|---|---|---|---|
| Err1: 0x01 | overcurrent or phase short circuit | Flashing 1 time in a loop | Lock/power off reset |
| Err2: 0x02 | overvoltage(voltage>DC60V) | Flashing 2 times in a loop | Lock/auto reset |
| Err3: 0x03 | hardware failure | Flashing 3 times in a loop | Lock/power off reset |
| Err4: 0x04 | open circuit or poor connection | Flashing 4 times in a loop | Lock/power off reset |
| Err5: 0x05 | EEPROM read/write errors | Flashing 5 times in a loop | Lock/power off reset |
| Err6: 0x06 | undervoltage | Flashing 6 times in a loop | Lock/auto reset |

# 7. Appendix: quick wiring motion control function guide

**At present, there are control units on the market with CANopen master station communication modules, such as PLCs, control cards, etc.**
**Some of them come with motion control libraries that support the CANopen CIA402 control protocol.**
**Users can directly call the motion control library functions that come with the control unit to complete the control of this driver without needing to have too much knowledge of the CANopen protocol.**
**The other part only has the CANopen communication module, which can parse EDS files, but does not support motion control libraries compatible with CANopen CIA402. In this case, users usually need to write motion control functions or function blocks according to the CANopen CIA402 standard protocol to control the driver to complete functions such as jog, JOG, or zero return. Users are required to have a certain understanding of the CANopen protocol.**

**A quick guide for users who need to write their own motion control library functions or function blocks.**
**The commonly used driver control functions are as follows:**

| Function | INPUT | OUTPUT | Description |
|---|---|---|---|
| MC_POWER | 6040h | 6041h | Enable |
| MC_RESET | 6040h | - | Reset |
| MC_STOP | 6040h | - | Stop |
| MC_HOME | 6040h,6060h,6098h,6099h,607Ch | 6041h | Homing |
| MC_MOVABS | 6040h,6060h,607Ah,6081h,6083h,6084h | 6041h | Absolute positioning |
| MC_MOVREL | 6040h,6060h,607Ah,6081h,6083h,6084h | 6041h | Relative positioning |
| MC_MOVVEL | 6040h,6060h,60FFh,6083h,6084h | 6041h | Speed command |
| MC_JOG | 6040h,6060h,60FFh,6083h,6084h | 6041h | JOG command |

**According to the CANopen CIA402 protocol, the control of the motor driver is mainly achieved through Control word 6040h and status word 6041h completed,**
**The following is a detailed introduction to how each function can control the driver by calling control words and status words**

**MC_POWER: Driver Enable**

| Step | Operation/Condition | Description |
|---|---|---|
| 1 | 6040h = 0x00, 6041h.bit0~bit3 = 0x00 | Motor Release Status |
| 2 | 6040h = 0x06, 6041h.bit0~bit3 = 0x01 | Motor Release Status |
| 3 | 6040h = 0x07, 6041h.bit0~bit3 = 0x03 | Motor Release Status |
| 4 | 6040h = 0x0F, 6041h.bit0~bit3 = 0x07 | Motor Release Status |

**MC_RESET: Driver Reset**

| Step | Operation/Condition | Description |
|---|---|---|
| 1 | 6040h .bit7=0 | Reset ready |
| 2 | 6040h .bit7=1 | Reset completed |

**MC_STOP: Driver STOP**

| Step | Operation/Condition | Description |
|---|---|---|
| 1 | 6040h.bit8=0 | Stop ready |
| 2 | 6040h.bit8=1<br>60FFh=0 | Stop completed |
| 3 | 606Ch=0 | Stop completed |

**MC_HOME: HOMING**

| Step | Operation/Condition | Description |
|---|---|---|
| 1 | 6060h=6 | Working mode -->Homing mode |
| 2 | 6061h=6 | Homing mode status |
| 3 | 607Ch=Origin Offset | Set Origin Offset |
| 4 | 6040h.bit4=0 | Homing ready |
| 5 | 6040h.bit4=1 | Homing |
| 6 | 6041h.bit10=1 and 6041h.bit13=1 | Homing failure |
| | 6041h.bit10=1 and 6041h.bit12=1 | Homing completed |

**MC_MOVABS: ABSOLUTE POSITIONING**

| Step | Operation/Condition | Description |
|---|---|---|
| 1 | 6060h=1 | Working mode -->Position mode |
| 2 | 6061h=1 | Position mode status |
| 3 | 607Ah = position<br>6081h= speed<br>6083h= acceleration<br>6084= deceleratoin | Write the motion parameters |
| 4 | 6040h= 0x0F | positioning ready |
| 5 | 6041h.bit12=0 | positioning ready |
| 6 | 6040h= 0x1F | positioning trigger |
| 7 | 6041h.bit12=1 | waiting for positioning |
| 8 | 6040h.bit4=0 | reset positioning trigger |
| 9 | 6041h.bit10=1 and 6041h.bit12=0<br>or 607Ah=6064h<br>or 6041.bit15=1 | target position rechead |

**MC_MOVREL: RELATIVE POSITIONING**

| Step | Operation/Condition | Description |
|---|---|---|
| 1 | 6060h=1 | Working mode -->Position mode |
| 2 | 6061h=1 | Position mode status |
| 3 | 607Ah = position<br>6081h= speed<br>6083h= acceleration<br>6084= deceleratoin | Write the motion parameters |
| 4 | 6040h= 0x4F | positioning ready |
| 5 | 6041h.bit12=0 | positioning ready |
| 6 | 6040h= 0x5F | positioning trigger |
| 7 | 6041h.bit12=1 | waiting for positioning |
| 8 | 6040h.bit4=0 | reset positioning trigger |
| 9 | 6041h.bit10=1 and 6041h.bit12=0<br>or 6041.bit15=1 | target position rechead |

**MC_MOVVEL: VELOCITY COMMAND**

| Step | Operation/Condition | Description |
|---|---|---|
| 1 | 6040h=0x0F | Reset control word, bit8 Halt |
| 2 | 60FFh = 0<br>6083h= acceleration<br>6084= deceleratoin | Write the motion parameters |
| 3 | 6060h= 3 | Swith to Profile Velocity mode |
| 4 | 6061h= 3 | Profile velocity mode status |
| 5 | 60FFh = target speed | Set target speed |
| 6 | 6041h.bit10=1 | target speed reached |
| 7 | 60FFh=0<br>or 6040h.bit8=1 | Stop |

**MC_JOG: JOG MOTION COMMAND**

| Step | Operation/Condition | Description |
|---|---|---|
| 1 | 6040h=0x0F | Reset control word, bit8 Halt |
| 2 | 60FFh = 0<br>6083h= acceleration<br>6084= deceleratoin | Write the motion parameters |
| 3 | 6060h= 3 | Swith to Profile Velocity mode |
| 4 | 6061h= 3 | Profile velocity mode status |
| 5 | Forward:<br>60FFh = target speed<br>Reverse:<br>60FFh = - target speed<br>Other:<br>60FFh = 1 | Forward and reverse movement |
| 6 | 6041h.bit10=1 | target speed reached |
| 7 | 60FFh=0<br>or 6040h.bit8=1 | Stop |