From the introduction to smart motors, we will introduce how to use them and typical programs. For the introductory edition, we have prepared the following programs and supplementary materials. (Excerpted part = *)

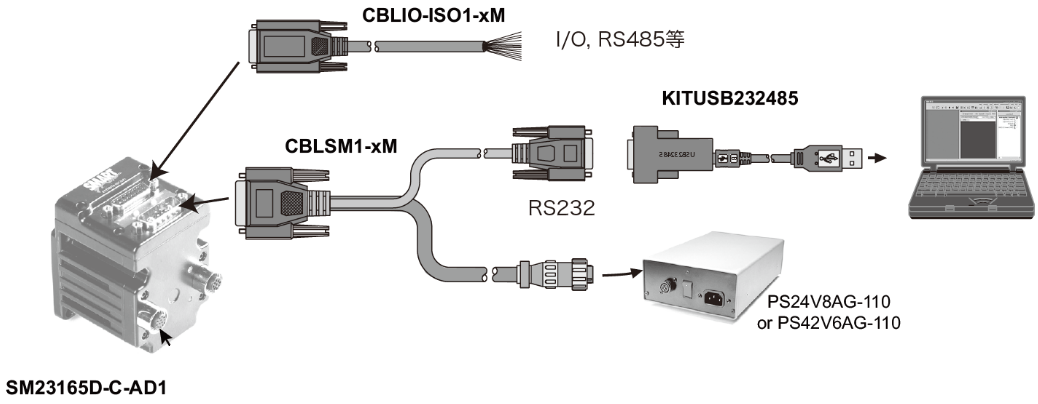
If you need an introductory version, please contact us or our authorized distributor.

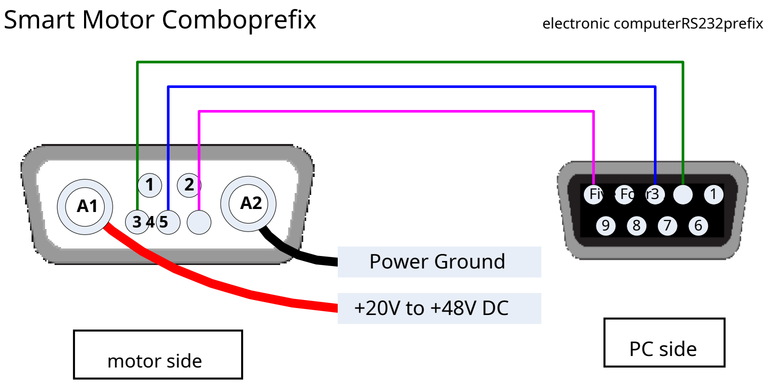
sample program

appendix

1Chapter Basic settings and servo motor control

Connection example (basic)





7pin comboDsub connector

A1 +20～+48DVC

A2power supplyGND

1 I/OG

2 +5Voutput

3 RS232send(Tx)

FourRS232Receive (Rx)

FivesignalGND

9pinRS232

2 RS232Receive (Rx)

3 RS232send (Tx)

FiveRS232 GND

1-2 Establishing a communication link

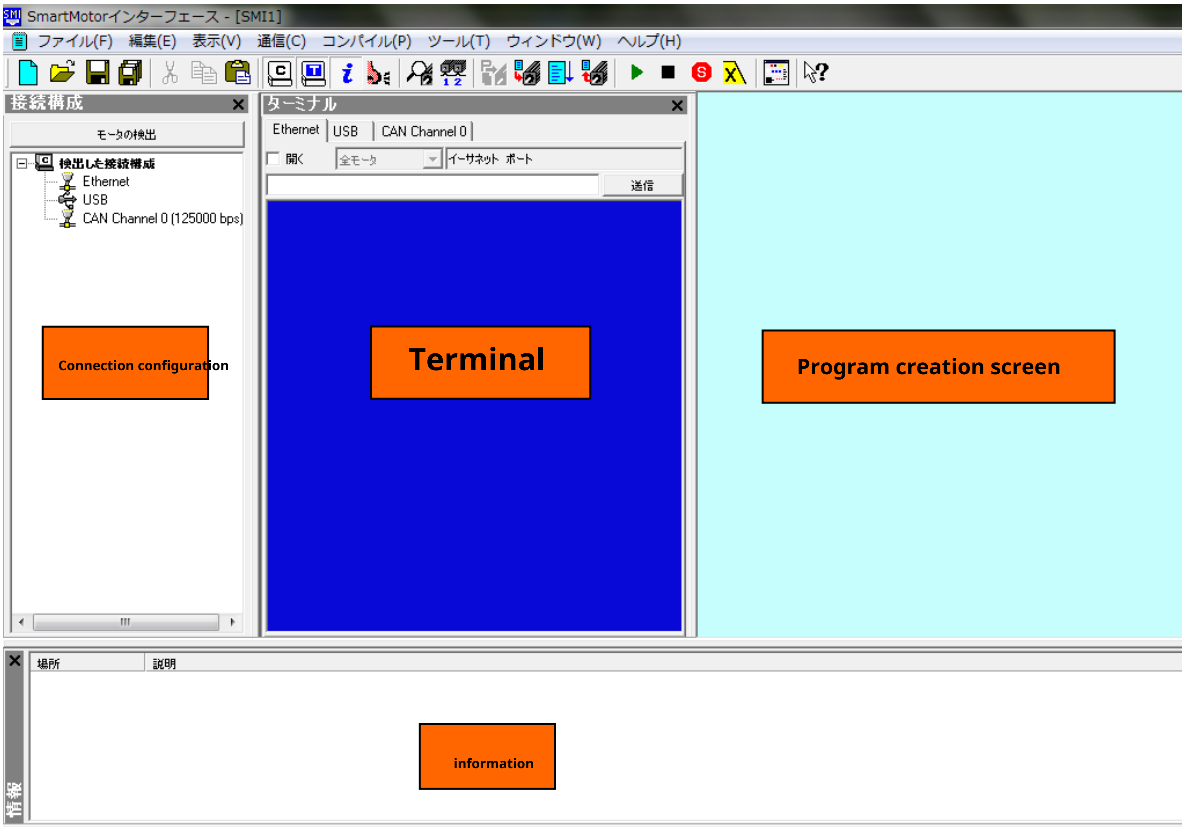
Launch SMI using a PC

SMIofPCWhen installing, follow the steps below.URLInstrumented from

Please roll. (free of charge)

https://www.moog.co.jp/products/motors-servomotors/smartmotor.html

SMI interface basic screen

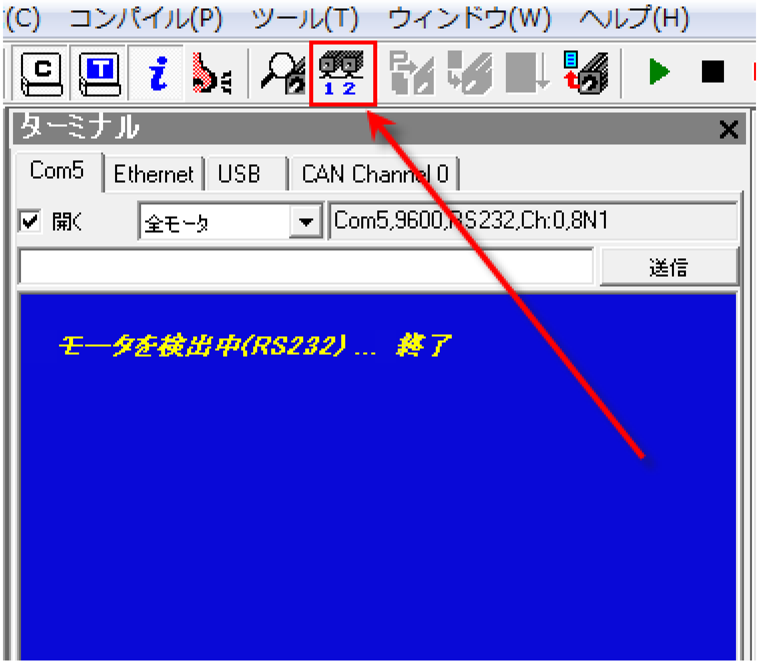


---

*Serial data analyzer

SMIThis is a convenient function for checking data information between the smart motor and the smart motor. Please refer to the appendix.

Establish communication between PC and integrated servo motor



Com5,9600,RS232,Ch:0,8N1

モータを検出中(RS232) … 終了

When using a single axis, specify the address.

Although it is not necessary to specify

For network use,SMI

Automatically assign addresses in order from

If you don't have a terminal window (SMI

(blue screen) provides accurate control instructions.

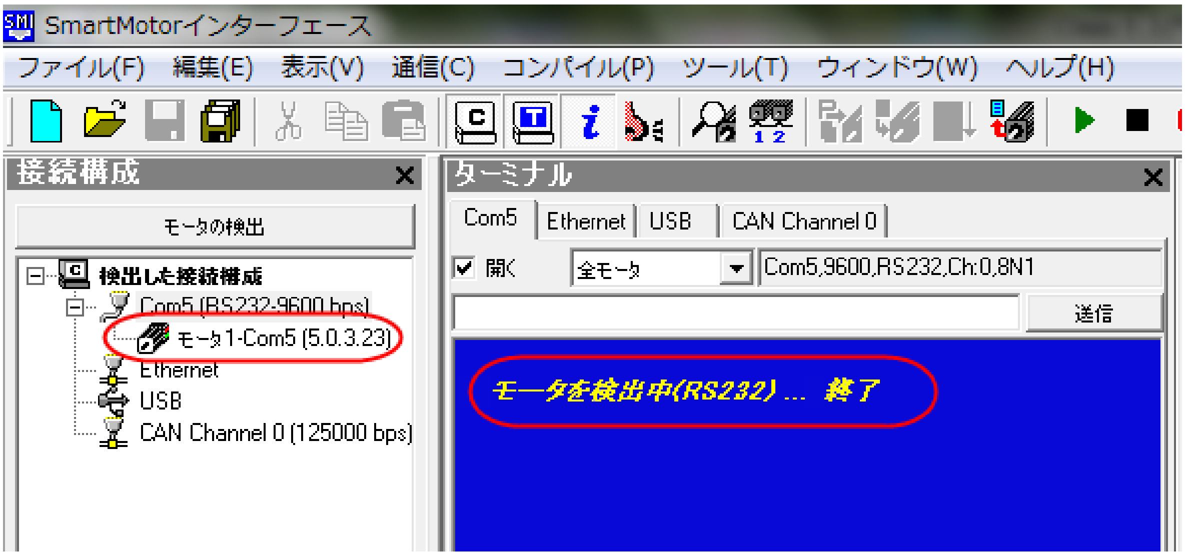
I can't. (Note: Address each axis in advance.)

and enter the address

When detecting, Click

please.)

Check the establishment of communication


SmartMotorインターフェース
ファイル(F) 編集(E) 表示(V) 通信(C) コンパイル(P) ツール(T) ウィンドウ(W) ヘルプ(H)
1 2
接続構成
モータの検出
ターミナル
Com5
Ethernet
USB
CAN Channel 0
☑ 開く
全モータ
送信

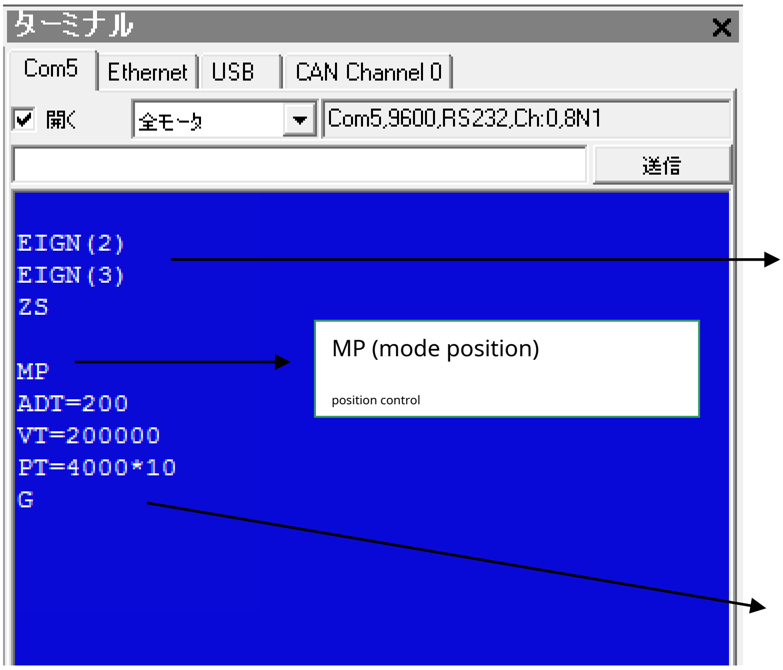
検出した接続構成
- Com5 (RS232-9600 bps)
  - モータ1-Com5 (5.0.3.23)
- Ethernet
- USB
- CAN Channel 0 (125000 bps)

Com5,9600,RS232,Ch:0,8N1

モータを検出中(RS232) … 終了

1-3 in the positive directionTenRotate (position control)

Write the command in the terminal (inside the blue screen) and press G(GO) + Enter to start rotating.



```
EIGN(2)
EIGN(3)
ZS

MP
ADT=200
VT=200000
PT=4000*10
G
```

MP (mode position)

position control

Ports 2&3 have overtravel limits
defaults to as input
is. Disable the limit or
Limit sensor (normally closed)

It works only when connected and set to Low state.

I don't.

EIGN(2): Sets port 2 as input and resets
Disable mitt input

EIGN(3): Set port 3 as input and restart.
disable mitt

ZS: Clear status bit

SM23XX: 4000 counts/rotation (enco
resolution)

(SM34XX): 8000 counts/rotation (engine
coder resolution)

Absolute target position

PT=absolute target position

Example: PT=4000 *10(rotation speed)=40000

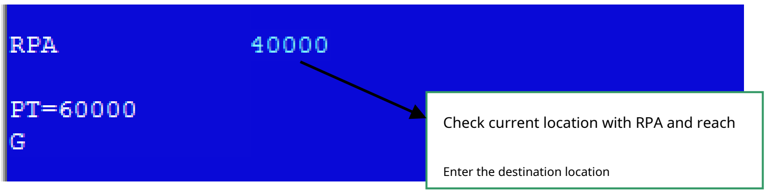
(Regardless of the current position, the absolute value
Move to 40000. 0=>40000,
2500=>40000)

Relative target position

PRT=relative position movement amount

Example: PRT=4000 (current absolute position is
For 40000, 40000+4000=44000

Operate by specifying absolute position



```
RPA          40000

PT=60000
G
```

Check current location with RPA and reach

Enter the destination location

[Achieved speed value]VT

Encoder resolution4000with the motor ofPIDThe settings arePID2in the case of:

VT=rev/sec*32768

example:200,000/32,768=6.10rps

6.10*60=366rpm

Encoder resolution8000with the motor ofPIDThe settings arePID2in the case of:

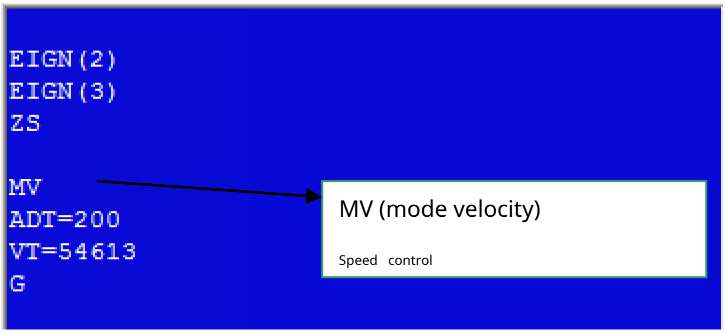
VT=rev/sec*65536

[Acceleration/deceleration value]ADT **(Acceleration value:AT,Deceleration value:DT)**

ADT=4.096 xrevolutions/second^2(encoder resolution4000with the motor ofPIDThe settings arePID2in the case of)

ADT=8.192 xrevolutions/second^2 (Encoder resolution8000with the motor ofPIDThe settings arePID2in the case of)

1-4Setting speed100RPM(speed control)



```
EIGN(2)
EIGN(3)
ZS

MV
ADT=200
VT=54613
G
```

MV (mode velocity)

Speed control

If the encoder resolution is 4000:

VT=rev/sec*32768

Ex: VT=1.66rps*32768

=54613

Hint: If you change the previously set parameters to

If you want to continue using it, please re-pair it.

There is no need to write parameters.

Rotate at 100RPM in negative direction

```
VT=-54613
G
```

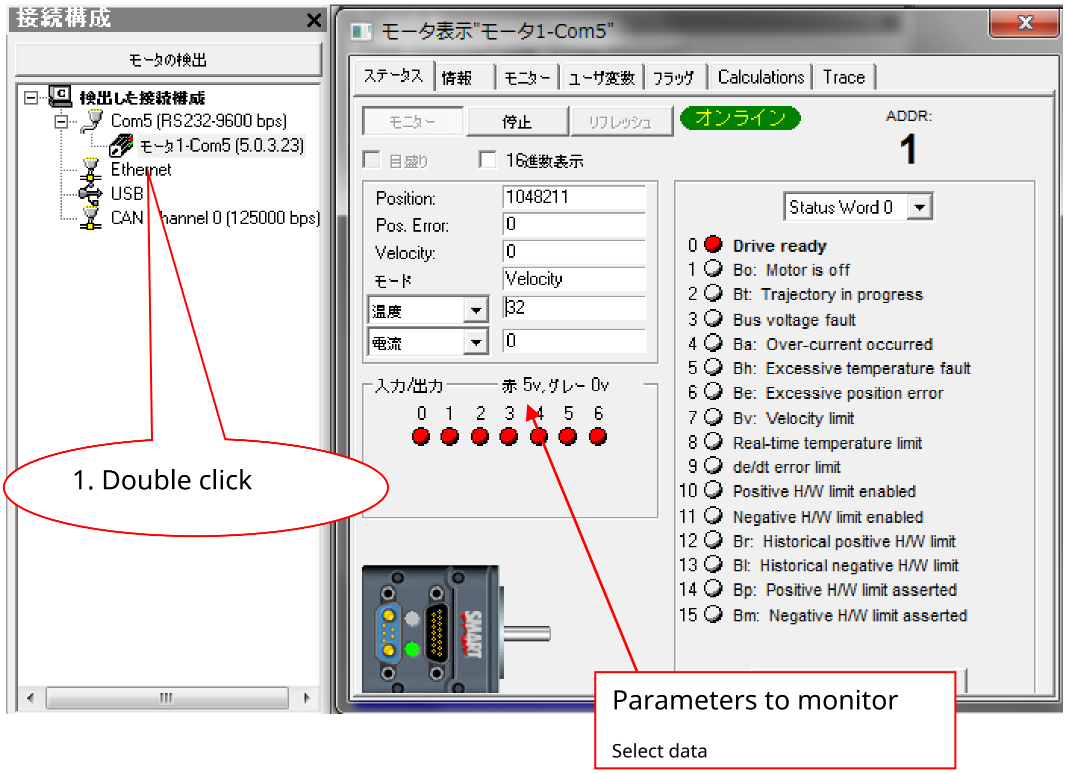
Rotate forward at 200RPM

```
VT=109227
G
```

Rotation stop (deceleration stop)

```
X
```

1-5 Parameter monitoring



接続構成

モータの検出

- 検出した接続構成
  - Com5 (RS232-9600 bps)
    - モータ1-Com5 (5.0.3.23)
  - Ethernet
  - USB
  - CAN Channel 0 (125000 bps)

モータ表示"モータ1-Com5"

ステータス | 情報 | モニター | ユーザ変数 | フラッグ | Calculations | Trace

モニター | 停止 | リフレッシュ | オンライン

ADDR: 1

□ 目盛り □ 16進数表示

Position:	1048211
Pos. Error:	0
Velocity:	0
モード	Velocity
温度	32
電流	0

入力/出力 赤 5v, グレー 0v

0 1 2 3 4 5 6

● ● ● ● ● ● ●

Status Word 0

- 0 ● Drive ready
- 1 ○ Bo: Motor is off
- 2 ○ Bt: Trajectory in progress
- 3 ○ Bus voltage fault
- 4 ○ Ba: Over-current occurred
- 5 ○ Bh: Excessive temperature fault
- 6 ○ Be: Excessive position error
- 7 ○ Bv: Velocity limit
- 8 ○ Real-time temperature limit
- 9 ○ de/dt error limit
- 10 ○ Positive H/W limit enabled
- 11 ○ Negative H/W limit enabled
- 12 ○ Br: Historical positive H/W limit
- 13 ○ Bl: Historical negative H/W limit
- 14 ○ Bp: Positive H/W limit asserted
- 15 ○ Bm: Negative H/W limit asserted

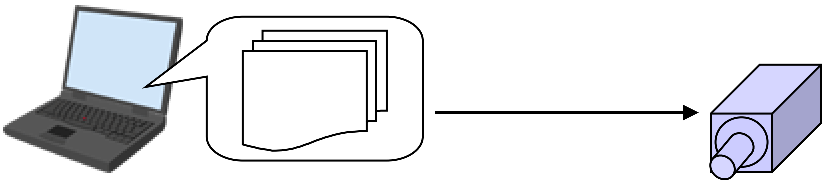
1. Double click

Parameters to monitor

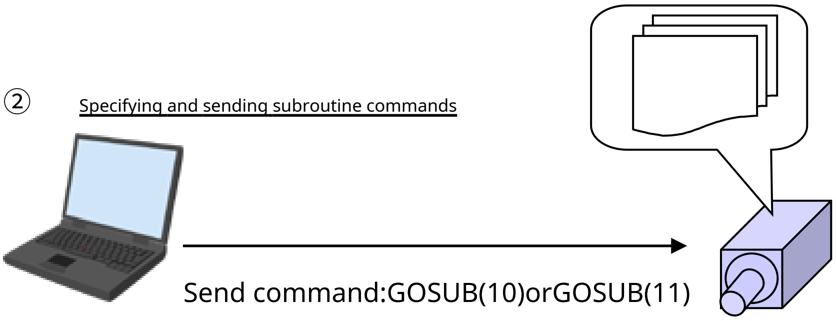
Select data

2Chapter Uploading the control program to the motor

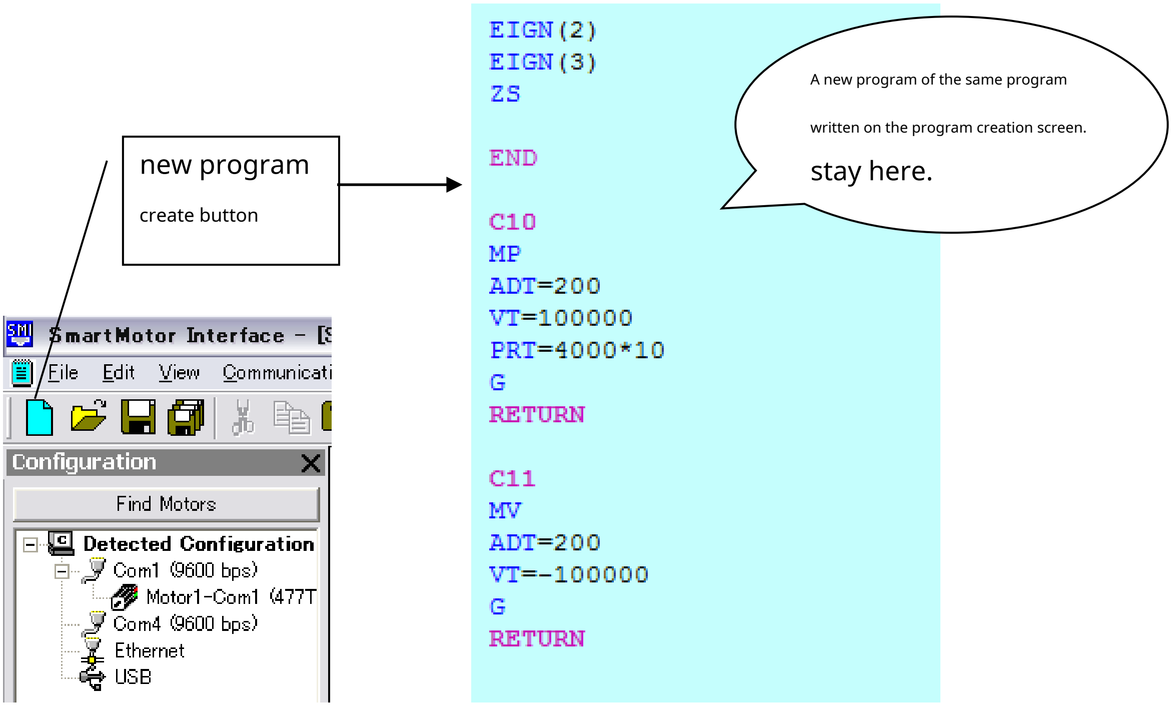
① Create a new program and upload it to the motor



② Specifying and sending subroutine commands



2-1 Creating a new program



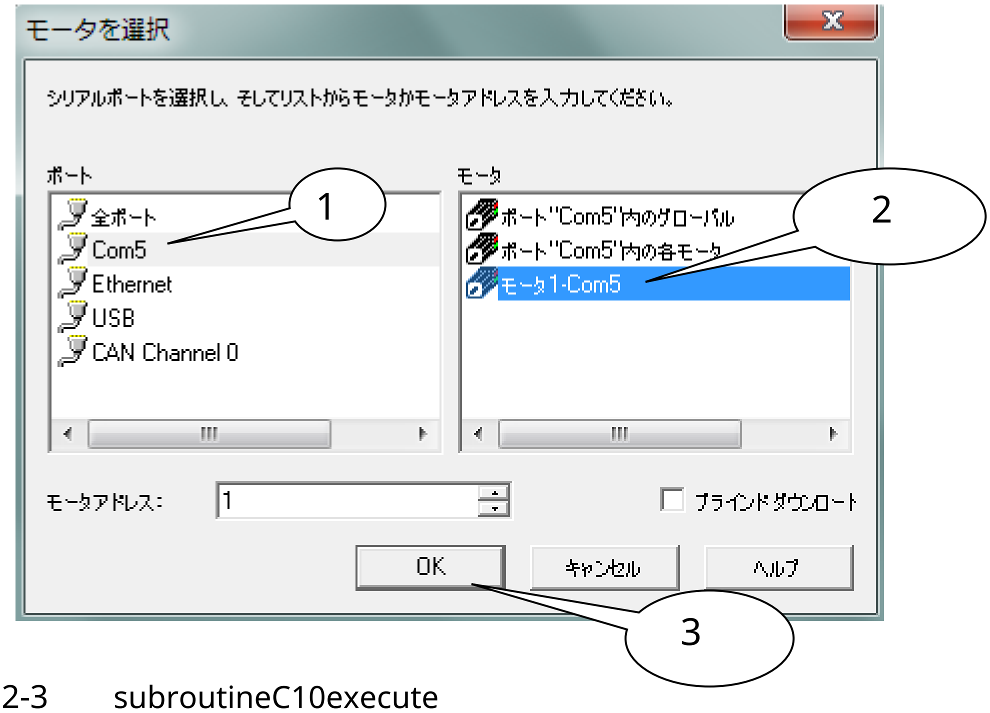
```
EIGN(2)
EIGN(3)
ZS

END

C10
MP
ADT=200
VT=100000
PRT=4000*10
G
RETURN

C11
MV
ADT=200
VT=-100000
G
RETURN
```

A new program of the same program

written on the program creation screen.

stay here.

2-2　Download program to motor



モータを選択

シリアルポートを選択し、そしてリストからモータかモータアドレスを入力してください。

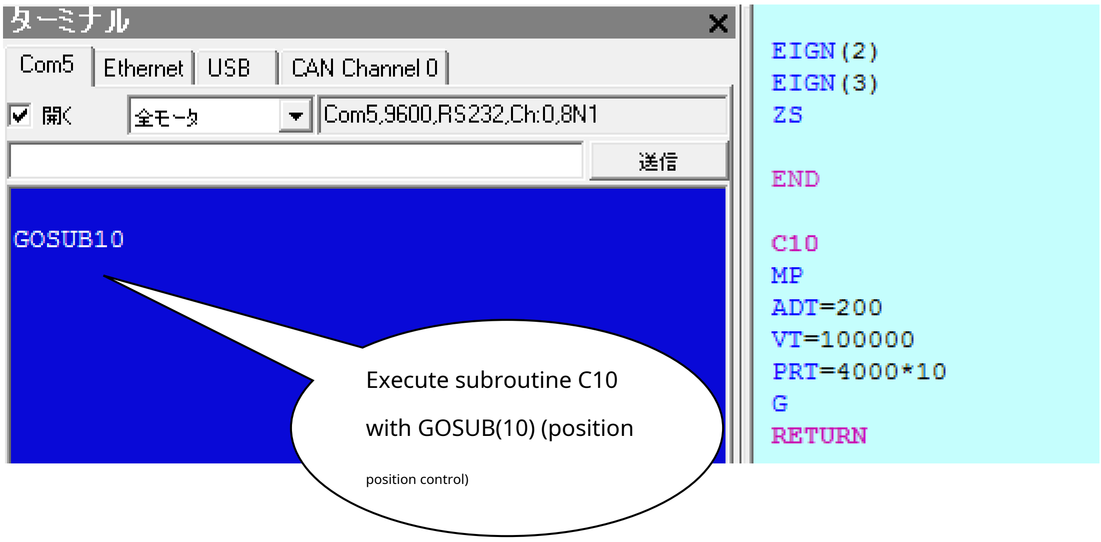
ポート
- 全ポート
- Com5
- Ethernet
- USB
- CAN Channel 0

モータ
- ポート"Com5"内のグローバル
- ポート"Com5"内の各モータ
- モータ1-Com5

モータアドレス: 1

□ ブラインドダウンロード

2-3　subroutineC10execute



ターミナル

Com5 | Ethernet | USB | CAN Channel 0

☑ 開く　全モータ　Com5,9600,RS232,Ch:0,8N1

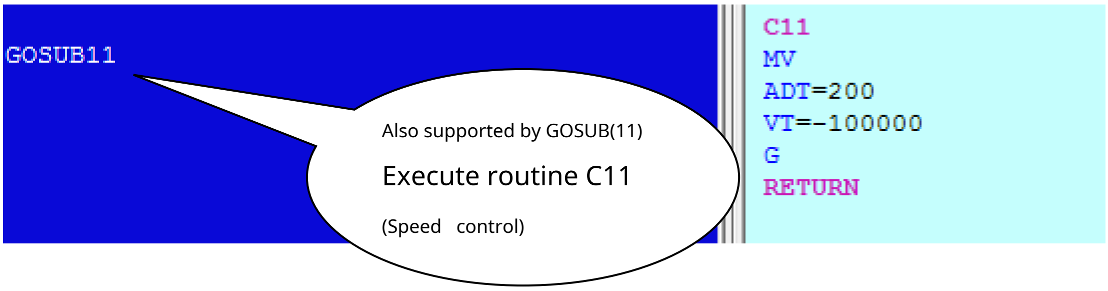
```
GOSUB10
```

```
EIGN(2)
EIGN(3)
ZS

END

C10
MP
ADT=200
VT=100000
PRT=4000*10
G
RETURN
```

Execute subroutine C10

with GOSUB(10) (position

position control)



```
GOSUB11
```

```
C11
MV
ADT=200
VT=-100000
G
RETURN
```

Also supported by GOSUB(11)

Execute routine C11

(Speed　control)

3-8Speed change & control via digital input

Start with position control, digital inputAbutLowThe speed will become faster.

```
EIGN(W,0)                     'Hardware limitI/ODisable
KD=10010                      'KDchange the value of
F                             'KDValid value of
O=0                           'Origin setting
ADT=100                       'Acceleration settings
VT=10000                      'Speed   setting
PT=40000                      'Achievement point setting
MP                            'Position mode setting
G                             'start
WHILEBt                       'Loop command during operation
  IFIN(0)==0                  '0port islowWhen
    IFVT==10000               'VTThe value of10000when it becomes
        VT=12000              'newVTadd value of
      G                       'start
    ENDIF
  ENDIF
LOOP'Loop command in action (WHILE Bt) END
```

3-7Speed change and control using analog input

This program includes a dead band function so that it does not react sensitively to analog signals. Operator is Ana

The motor speed will not change unless you change the log input.

```
EIGN(W,0)          'Disable hardware limits
KP=3020            'PIDvalue(Proportional) from default
KD=10010           'PIDvalue(Differential) changed from default
F                  'PIDSetting buffer value valid
ADT=100            'Acceleration settings
MV                 'Speed   mode
d=10               'analog dead band,5000 =fault scale
o=2500             'Offset for negative movement
m=40               'Speed   (multiple)
w=10               'time(10mseconds)
b=0                'speed
C10
a=INA(V1,3)-o      '5VAnalog value reading
x=ab               'Set x to change input value
IFx>d              'Check if the dead band is exceeded
    VT=b*m         'Set speed value
    G              'Start at a new speed
ELSEIFx<-d         'Check if the dead band is exceeded
    VT=b*m         'Set speed value
    G              'Start at the set speed
ENDIF              '
b=a                'Anti-hunting update
WAIT=w             'wWait for seconds
GOTO10             'labelTenthe call of
END
```

Dsub15pin I/Os (5V I/Os)

- INA(A, exp)Raw analog read:10Bit resolution,0-32736=0-5VDC
- INA(V1, exp)Voltage scale (millibo) (loaded with root),3456=3.456VDC

Expansion 10 points24DC I/O

- INA(A, exp)Raw analog read:10Bit resolution,0-32736=0-41.25VDC
    - The resolution is10bitin32Every increase/decrease,
    - voltage value0-41.25VDCis inside the board Reference value.I/OThe maximum input value is 24VDC
- INA(V, exp)Voltage scale reading (24000-0), 15500=15.5V
- INA(V1, exp) Read voltage scale (5000-0), 550=0.55V

3-1 Determining the origin from sensor input

```
EILP                    'Forward limit switch enabled (default setting)
EILN                    'Negative limit switch enabled (default setting)
ZS
END
C0                      'Origin search subroutine
EIGN(3)                 'port3Disable the limit switch for the negative direction of
ZS
MV                           'Speed   mode setting
ADT=100                      'Homing acceleration setting
VT=-70000                    'Homing speed setting
G                            'Start rotating towards the origin (limit sensor)
WHILEIN(3)==0LOOP            'Port 3LowLoop when ->Highexits the loop when
MTB                          'Stop using torque brake mode
WAIT=50                      '50mWait for seconds
O=-8000                      'Current position minus8000and setting
MPPT=0G TWAIT                'Start the motor to position zero
EILN                         'port3Set for limit switch (negative direction limit switch enabled)
RETURN
```

3-3Pushing origin search

What you will understand from this program:

1.How to return to origin by pushing without using sensor input

Caution: When pressing, be careful of the set speed and use the lowest possible deviation limit value. high speed

If the motor stops due to collision or if the position deviation value is set to a large value and the motor is pressed, the motor may be damaged.

Please use the pushing origin search at your own discretion.

```
'================================================= ==============

'Return to origin routine (return to origin by pushing)


EIGN(2) EIGN(3)ZS               'Remove travel limit,I/Oset as normal input/output
'parameter settings
rr=-1                           'Origin return rotation direction
vv=70000                        'Homing speed
aa=1000                         'Home return acceleration
ee=300                          'Home return deviation limit
tt=3000                         'Homing torque limit
hh=4000                         'Homing offset
ZS                              'Clear error bit

MV                              'Speed   mode setting
ADT=aa                          'Homing acceleration setting
VT=vv*rr                        'Homing speed setting
G                               'Start rotating in the pushing direction
WHILEABS(EA)<eeLOOP'Values   within the deviation limit are looped
MTB                             'Stop using torque brake mode
MT                              'When pressing, it bounces, so press again in torque mode.
T=tt*rr                         'Preset torque value
G
WAIT=50                         '50mWait for seconds
O=hh*rr                         'Set the origin offset position as the origin
MPPT=0G TWAIT                   'Set motor to position zero
END
```

3-4error handling

```
EIGN(W,0,12)'Remove travel limit,I/Oset as normal input/output
              'Bit converts all inputs into binary values12and input2and3set as general input
              Ru.
ZS            'Clear fault bit (Note: If the travel limit is not grounded at boot-up)
              (if applicable)

              'Fault interrupt settings
              'ITR(interrupt number, status word, status bit, bit status,C
              label call)
              'Note: interrupt is8configurable,0～7, 0should be given the highest priority
ITR(0,0,0,0,0)'set interrupts to zero
              'status word zero
              'bit zero
              'Set bit status to zero (drive ready bit)
              'Subroutine zero (C0)call

EITR(0)       'Enable zero interrupts
ITRE          'Enable global interrupt scanner


PAUSE         'Pause command (ENDcommand prevents interrupts from being disabled)
END

C0
'Place error code
'This routine is interrupted on travel limit, overtemperature, position deviation error, and overcurrent. IF
Be
   PRINT("Position      Error",#13)
ENDIF
IFBh
   PRINT("Over    Temp Error",#13)
ENDIF
IFBa
   PRINT("Over    Current Error",#13)
ENDIF
RETURNI
```

3-13Program example for linear actuator

inputFourby point16Point positioning, soft limit, push origin search

Our company sellsSmartBox BCDIt is convenient to use.

Four16 subroutines are called based on the point input, and input 6 (G)It works when received.

```
'ECHO                       'Echo on
EIGN(W,0)                   'All ofI/Oset to input
ZS                          'Reset all error bits
MDS                         'Sine wave drive mode
'================================================= ================= OUT
(4)=1                       'ESet port to busy output
OUT(5)=0                    'FSet port to error output
'================================================= =================
'Set parameters
        rr=-1               'Return to origin
        vv=100000           'speed
        aa=100              'acceleration
        ee=300              'Current error value
        tt=4000             'Torque limit
        hh=4000             'offset
        nn=-300             'Software limit (minus direction)
        pp=110000           'Software limit (plus direction)
'================================================= ================= SLE
                            'Enable software limits
SLN=nn                      'Minus direction rotation limit
SLP=pp                      'Plus direction rotation limit
SLM(1)                      'Set to soft limit mode
'================================================= ================= ITR
(0,0,0,0,100)               'Interrupt on error(4-6reference)
EITR(0)                     'Enable zero interrupts
'================================================= ================= ITR
(1,16,6,0,101)              'Interrupt settings(4-6reference)
ITRE                        'Enable global interrupt scanner
EITR(1)                     'interrupt1enable
WHILE1LOOP
END
'================================================= =================
C100
OUT(5)=1                        'Eporthigh
IFBe                            'Position deviation error bit
   PRINT("Position Error",#13) ENDIF

IFBh                            'Overtemperature error bit
   PRINT("Over Temp Error",#13) ENDIF

IFBa                            'Overcurrent error bit
   PRINT("Over Current Error",#13) ENDIF

RETURNI
'================================================== ================= C101
                'portFWhen there is an input to101call
PRINT("GO PRESSED",#13)
x=15-(IN(W,0)&15) GOSUB(x)      'Input port0,1,2,3 (A, B, C, D)confirm
                                'subroutine0-15call
ITR(1,16,6,0,101)               '(4-6reference)
RETURNI
'================================================== =================

'Press return to origin
C0
ZS                                     'Reset all error bits
S.L.D.                                 'Soft limit disabled
OUT(4)=0                               'Dport outputLow
VT=vv*rr                               'speed
ADT=aa                                 'acceleration
MV                                     'Speed   mode
ZS                                     'Clear error bit
G
WHILEABS(EA)<eeLOOP MTB                'Loop within deviation limit
                                       'Torque brake
MT                                     'Torque mode
T=tt*rr                                'Torque value setting
G
WAIT=500                               '500mWait for seconds
O=hh*rr                                'Position settings
MPPT=0G TWAIT                          'Position mode/Return to origin
```

```
OUT(4)=1                                    'Dport outputHigh
SLE                                         'Soft limit enabled
RETURN
'================================================== ================= C1

OUT(4)=0
MP
ADT=300
VT=900000
P.T.=90000
G TWAIT
P.T.=0
G TWAIT
P.T.=90000
G TWAIT
P.T.=0
G TWAIT
OUT(4)=1
RETURN
'================================================== ================= C2

C3
C4
C5
C6
C7
C8
C9
C10
C11
C12
C13
C14
C15
PRINT("Called SUBROUTINE ",x,#13) PRINT
("EXITING SUBROUTINE ",x,#13) RETURN
```

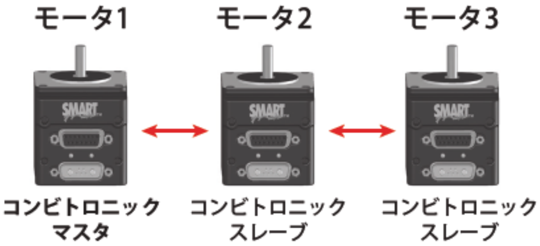
CAN&CombitronicConnection example (multi-axis control) **1.**

Upper controller (PC/PLC)If you don't use:

all smart motorsCANnetwork connection

Uses Combitronic communication

Multi-axis motor control and each axis using any smart motor as the masterI/Ocontrol.



2. Upper controller (PC/PLC)everything you needCANWhen connecting with:

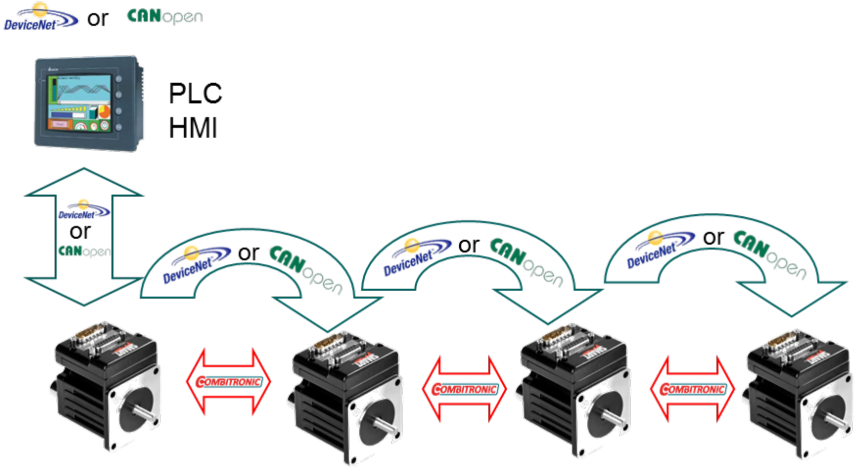
2-1The upper controller is the main master.

CANopenusing the protocol**Smart motor is a slave device**controlled as (CANopen)

2-2The smart motor is the main master.

The upper controller sends and receives minimal data (CANopen)Just do**smart motor star**and control (Combitronic).

Control using Combitronic communication (CANopenandCobitronic)



3-18-4 2axis control

Master Motor(Motor 1)

```
CADDR=1                    'Declare motor address 1
ADDR=CADDR                 'CANaddress andRS232Unify addresses '
EIGN(2) :0                 all motors :Specify the second port as the general input port
EIGN(3) :0                 ' all motors :Specify the third port as the general input port
ZS:                        ' whole mor Taliset
MV:0                       ' all motors :Speed mode
VT:1=64424*3               ' Motor 1 : Speed specification (3rps)
ADT:0=1000                 ' all motors : Acceleration specification
G:1                        ' Motor 1 :execution
WAIT=1000*10               'TenWait for seconds
VT:2=64424*5               ' motor2 : Speed specification (5rps)
G:2                        ' motor2 :execution

WHILEP.A. :1 > P.A:2 LOOP  'loop 【 motor1 position> motor2 Position of】

X:0                        ' all motors :Stop

MP:0                       ' all motors :Position mode
PML:0 =4000                ' all motors : Encoder modulo limit specification
PMT:0 =2000                ' all motors :Encoder modulo target specification
G:0                        ' all motors : Execution (encoder modulo function)

END
```

Slave motor(Motor 2)

```
CADDR=2                    'Declare motor address 2
ADDR=CADDR                 'CANaddress andRS232Unify addresses

END
```

3-18-5 Linear interpolation (synchronous control)

hostPC/PLCLinear interpolation control is possible without using .

```
ADTS=100                     'Set synchronous command acceleration
VTS=100000                   'Set the synchronous command attained speed
PTS(30000;1,40000;2)         'Set the destination position of motors 1 and 2
          'PTS(position1;Axis number1,position2;Axis number2[,position3;Axis number3])
G.S.                         'Start, start synchronization
```

Class5 command table (Ver1_10)

command	explanation
A	
a . . . z	user variable
a=...z=	User variable settings
aa . . . zz	user variable
aa=...zz=	User variable settings
aaa...zzz	user variable
aaa=...zzz=	User variable settings
ab[index]	8 bit variable
ab[index]=...	8-bit variable settings
af[index]	float variable
af[index]=....	Float variable settings
al[index]	32 bit variables
al[index]=....	32 bit variable settings
aw[index]	16 bit variable
aw[index]=....	16-bit variable settings
Ai(0)	Get internal encoder Z phase position at rising edge
Ai(1)	Obtain external encoder Z phase position at rising edge
Aij(0)	Get internal encoder Z phase position at rising edge and falling edge
Aij(1)	Obtain external encoder Z phase position at rising edge and falling edge
Aj(0)	Get internal encoder Z phase position at falling edge
Aj(1)	Obtain external encoder Z phase position at falling edge
Aji(0)	Get internal encoder Z phase position at falling edge and rising edge
Aji(1)	Obtain external encoder Z phase position at falling edge and rising edge
ABS(...)	integer absolute value
A.C.	Command acceleration
ACOS(...)	Arccosine (inverse of cosine) by angle
ADDR	motor serial address
ADDR=...	Motor serial address setting
ADT=...	Acceleration/deceleration setting
ADTS=...	Synchronous operation acceleration/deceleration setting
AMPS	Maximum PWM limit
AMPS=...	Maximum PWM limit setting
ASIN(...)	arc-sine (inverse function of sine) by angle
AT	acceleration
AT=...	Acceleration settings
ATS=...	Synchronous operation acceleration setting
ATAN()	Arctangent by angle (inverse tangent)
ATOF()	Get ASCII and convert to float type
B	
B()	status bit
Ba	Overcurrent status bit
BAUD(0)	channel 0 baud rate
BAUD(1)	channel 1 baud rate
BAUD#	Channel 0 baud rate setting
BAUD(0)=...	Channel 0 baud rate setting
BAUD(1)=...	Channel 1 baud rate setting
Be	Position deviation error status bit

Bh	Overtemperature status bit
Bi(0)	Get internal encoder Z phase status bit on rising edge
Bi(1)	Get external encoder Z phase status bit on rising edge
Bj(0)	Get internal encoder Z phase status bit on falling edge
Bj(1)	Get external encoder Z phase status bit on falling edge
Bk	EEPROM data integrity status bit
Bl	Left/Negative Hardware Overtravel Limit History Status Bit
Bls	Left/Negative Direction Software Overtravel Limit History Status Bit
Bm	Left/Negative Hardware Overtravel Limit Status Bit
Bms	Left/Negative Direction Software Overtravel Limit Status Bit
Bo	Motor OFF status bit
Bp	Right/Forward Hardware Overtravel Limit Status Bit
Bps	Right/Forward Software Overtravel Limit Status Bit
Br	Right/Forward Hardware Overtravel Limit History Status Bit
Brs	Right/Forward Software Overtravel Limit History Status Bit
Bs	syntax error status bit
Bt	Operation in progress status bit
Bv	Speed error status bit
Bw	32-bit position counter Wrap around status bit
Bx(0)	Real-time internal index input status bit
Bx(1)	Real-time external index input status bit
BREAK	Program execution flow control
BRKENG	Brake operation
BRKRLS	brake release
BRKSRV	Brake operation while servo is stopped
BRKTRJ	Brake activation while operation is stopped
C	
C#	Program subroutine label
CADDR	CAN address
CADDR=	CAN address setting
CAN	CAN error
CANCTL(...)	Network function control
CASE#	program flow instructions
CBAUD	CAN baud rate
CBAUD=	CAN baud rate setting
CCHN()	serial channel closed
CHN(0)	RS232 communication error flag
CHN(1)	RS485 communication error flag
CLK	1ms clock variable
CLK=...	1ms clock setting
COS(...)	Cosine (angle)
C.P.	cam pointer
CTA(...)	Add cam table
CTE(...)	Delete cam table
CTR(0)	Primary encoder/pulse train & direction counter
CTR(1)	Second encoder/pulse & direction counter
CTT	Number of cam tables in EEPROM
CTW()	cam table write
D	
DEA	Execution de/dt value

DEFAULT	Switch-case syntax element
DEL	de/dt fault limit
DEL=...	de/dt fault limit setting
DFS(...)	In 32-bit IEEE format, af[] variable
DITR(...)	Interrupt stop
DT	Deceleration
DT=...	Deceleration setting
DTS=...	Synchronous action deceleration setting
E	
EA	Position deviation error
ECHO	Output input data to main channel
ECHO_OFF	Main channel echo stop
ECHO1	Output input data to secondary channel
ECHO_OFF1	Secondary channel echo stop
EIGN(...)	Set I/O pin to input
EILN	Left/negative side hardware limit switch on
EILP	Right/positive side hardware limit switch on
EIRE	Configuring index signal acquisition pin for external encoder
EIRI	Configure index signal acquisition pin for internal encoder
EISM(6)	Set pin 6 to G command input
EITR(...)	Interrupt settings
EL	Position deviation error fault limit
EL=...	Position deviation error fault limit setting
ELSE	IF syntax element
ELSEIF	ELSE syntax element
ENC0	Internal encoder selection
ENC1	External encoder selection
END	The end of the program
ENDIF	IF statement end
ENDS	switch syntax end
EOBK(...)	Sends brake signal to I/O output
EPTR	EEPROM pointer
EPTR=...	EEPROM pointer setting
ERRC	Most recent command error code
ERRW	Latest command error communication channel
F	
F	Buffered PID settings enabled
FABS(...)	floating point absolute value error
FSA(...)	Fault action settings
FSQRT(...)	floating point square root
FW	Firmware version
G	
G	Motion start (GO)
G(...)	Specified orbit motion start (GO)
G.S.	Synchronous motion motion start (GO)
GETCHR	Main communication channel string
GETCHR1	Secondary communication channel string
GOSUB(...)	Subroutine call by number or variable
GOSUB#	subroutine call
GOTO(...)	Jump to program label by number or variable

GOTO#	Jump to program label
H	
HEX(...)	hex string variable
I	
I(0)(capital i)	Encoder Z phase input position variable (rising edge, internal encoder)
I(1)(capital i)	Encoder Z phase input position variable (rising edge, external encoder)
IF...	IF syntax element
IN(...)	I/O input
INA(...)	analog input
ITR(...)	User interrupt settings
ITRD	Stop all user interrupts
ITRE	All user interrupts enabled
J	
J(0)	Encoder Z phase position variable acquisition (falling edge, internal encoder)
J(1)	Encoder Z phase position variable acquisition (falling edge, external encoder)
K	
K.A.	KA buffer PID value (acceleration)
KA=...	KA buffer PID value setting
K.C.	KC value
KC=...	KC value setting
KCS	KCS value
KCS=...	KCS value setting
KD	KD buffer PID value (differential)
KD=...	KD buffer PID value setting (differentiation)
KG	KG buffer PID value (gravity)
KG=...	KG buffer PID value setting (gravity)
K.I.	KI buffer PID value (integral)
KI=...	KI buffer PID value setting (integral)
KL	KL buffer PID value (integral limit)
KL=...	KL buffer PID value setting (integral limit)
KP	KP buffer PID value (proportional)
KP=...	KP buffer PID value setting (proportional)
K.S.	KS buffer PID value (integral filter control)
KS=...	KS buffer PID value setting (integral filter control)
KV	KV buffer PID value (velocity feedforward)
KV=...	KV buffer OID value setting (velocity feedforward)
L	
LEN	Main communication channel buffer occupancy level (data mode)
LEN1	Secondary communication channel buffer occupancy level (data mode)
LFS(...)	32-bit IEEE format Float value
LOAD	Download program to motor
LOCKP	Prevent program upload until new program is loaded
LOOP	WHILE syntax element
M	
MC	Cam mode enabled
MC(...)	Cam mode enabled (additional trajectory)
MCE(...)	Cam spline enabled
MCW(...)	Cam start point
MDB	TOB commutation enabled
MDC	Sine wave current commutation mode

MDE	Trapezoidal encoder commutation mode
MDS	Sine wave voltage commutation mode
MDT	Trapezoid Hall sensor commutation mode
MF0	Set CTR(1) to 0 and select external encoder to 4x mode
MFA(...)	Follow mode, rising
MFD(...)	Follow mode, falling
MFDIV	Tracking mode divisor
MFDIV=...	Tracking mode divisor setting
MFMUL	Follow mode multiplier
MFMUL=...	Follow mode multiplier setting
MFR	4 multiplication tracking mode selection
MFR(...)	4-fold tracking mode selection (additional trajectory)
MFSDC(...)	Follow mode (stall – pause – continue)
MFSLEW(...)	Follow mode | constant
MINV(...)	reverse commutation
MODE	action mode
MODE(...)	Operation mode (specific orbit)
MP	Position mode enabled
MP(...)	Position mode enabled (additional trajectory)
MS0	Set CTR(1) to 0 and select pulse/direction mode using external encoder
MSR	Select tracking mode in pulse/direction mode
MSR(...)	Select following mode in pulse/direction mode (additional trajectory)
MT	Torque mode enabled
MTB	mode torque brake
MV	Speed mode enabled
MV(...)	Speed mode enabled (additional trajectory)
O	
O=...	Origin setting
O(...)=...	Setting the origin of a specific trajectory
OC(...)	Output status (24V I/O)
OCHN(...)	open communication channel
OF(...)	Output fault (24V I/O)
OFF	servo off
OR(...)	Set output to low
OS(...)	Set output to High
OSH=...	Origin shift
OSH(...)=...	Specific origin shift
OUT(...)=...	Set one or more outputs to a specific state
P	
P.A.	actual position
PAUSE	Pause program execution
PC	Command position
PC(...)	Command position (specific trajectory)
P.I.	Get π value
PID1	16,000Hz PID rate
PID2	8,000 Hz PID rate (default)
PID4	4,000Hz PID rate
PID8	2,000Hz PID rate
PMA	Actual position modulo
PML	position modulo limit

PML=…	Position modulo limit setting
PMT	Position modulo target (position motion)
PMT=…	Position modulo target setting (position operation)
PRA	Acquisition of actual measurement start position
PRC	Operation start command position
PRINT(…)	Data output to main communication channel
PRINT1(…)	Data output to secondary communication channel
PRT	Relative target position
PRT=…	Relative target position setting
PRTS=(…)	Synchronous relative target position setting
PRTSS=(…)	Additional synchronized relative target position setting
P.T.	Target position
PT=	Target position setting
PTS=(…)	Synchronous absolute target position setting
PTSS=(…)	Additional synchronized absolute target position setting
PTSD	Synchronous linear working distance
R	
Ra…Rzzz	Get user variable
Ra…Rzzz	User variable report
Raaa…Rzzz	User variable report
Rab[index]	8-bit array variable report
Raf[index]	Float type variable report
Ral[index]	32-bit variable reporting
Raw[index]	16-bit variable reporting
RABS(…)	Integer type absolute value reporting
R.A.C.	Command acceleration value report
RACOS(…)	Arc-cosine (inverse cosine) value reporting by angle
RADDR	Motor serial address report
RAMPS	Maximum PWM limit reporting
RANDOM	Obtain random number Example: a=RANDOM
RANDOM=…	Random number setting
RASIN(…)	Arc-sin (inverse sine) value reporting by angle
RAT	Target acceleration value report
RATAN(…)	Arc-tan (inverse tangent) value reporting by angle
RATOF(…)	Convert ASCII to Float type and report
RB()	status bit report
RBa	Overcurrent status bit reporting
RBAUD(0)	Channel 0 baud rate report
RBAUD(1)	Channel 1 baud rate report
RBe	Position deviation error status bit report
RBh	Overtemperature status bit reporting
RBi(0)	Encoder Z phase status bit (internal encoder, rising signal) report
RBi(1)	Encoder Z phase status bit (external encoder, rising signal) report
RBj(0)	Encoder Z phase status bit (internal encoder, falling signal) report
RBj(1)	Encoder Z phase status bit (external encoder, falling signal) report
RBk	EEPROM data integrity status bit reporting
RBl	Hardware left/negative overtravel limit history bit reporting
RBls	Software left/negative overtravel limit history bit reporting
RBm	Left/Negative Hardware Overtravel Limit Bit Bit Report
RBms	Left/Forward Software Travel Limit Bit Bit Report

RBo	Servo off status bit report
RBp	Right/Forward Hardware Overtravel Limit Bit Report
RBps	Right/Forward Soft Limit Overtravel Limit Bit Report
RBr	Hardware right/forward overtravel limit history bit reporting
RBrs	Software right/forward overtravel limit history bit reporting
RBs	Syntax error status bit report
RBt	Operation in progress status bit report
RBv	Speed error bit reporting
RBw	32bit position counter wrap/around status bit report
RBx(...)	Real-time internal index input status bit reporting
RCADDR	CAN address report
RCAN	CAN error reporting
RCBAUD	CAN baud rate report
RCHN(0)	RS-232 communication error flag report
RCHN(1)	RS485 communication error flag report
RCKS	Program checksum report
RCLK	1ms clock variable reporting
RCOS(...)	Report cosine as an angle
RCP	Cam pointer report
RCTR(0)	Primary encoder/pulse train & direction counter reporting
RCTR(1)	Second encoder/pulse & direction counter reporting
RCTT	Report of number of cam tables in EEPROM
RDEA	Execution de/dt value report
RDEL	Setting de/dt fault limit reporting
RDFS	in 32-bit IEEE format, af[] variable reporting
RDT	Setting deceleration value report
REA	Position deviation error report
REL	Position deviation error fault limit report
REPTR	EEPROM pointer data report
RERRC	Recent command error code report
RERRW	Recent command error communication channel report
RES	Encoder resolution report example: a=RES
RESUME	Resume program execution after pausing
RETURN	Return from subroutine
RETURNI	Return from interrupt routine
RFABS(...)	Floating point absolute value error reporting
RFSQRT(...)	floating point square root report
RFW	Firmware version report
RGETCHR	String reporting from main communication channel
RGETCHR1	String report from secondary communication channel
RHEX(...)	Report variables as hex columns
RI(0)	Encoder Z phase input position variable report (rising edge, internal encoder)
RI(1)	Encoder Z phase input position variable report (rising edge, external encoder)
RIN(...)	I/O input report
RINA(...)	Analog input report
RJ(0)	Encoder Z phase input position variable report (falling edge, internal encoder)
RJ(1)	Encoder Z phase input position variable report (falling edge, external encoder)
RKA	KA buffer PID value (acceleration feedforward) report
RKC	KC value report
RKCS	KCS value report

RKD	KD buffer PID value (differential) report
RKG	KG buffer PID value (gravity) report
RKI	KI buffer PID value (integral) report
RKL	KL buffer PID value (integral limit) report
RKP	KP buffer PID value (proportional) report
RKS	KS buffer PID value (integral filter control) report
RKV	KV buffer PID value (velocity feedforward) reporting
RLEN	Main communication channel buffer occupancy level (data mode) report
RLEN1	Secondary communication channel buffer occupancy level (data mode) report
RLFS(...)	32-bit IEEE format Float value reporting
RMFDIV	Tracking mode divisor report
RMFMUL	Follow mode multiplier report
RMODE	Operating mode report
RMODE(...)	Operation mode (specific trajectory) report
ROC(...)	Output status (24V I/O) report
ROF(...)	Output fault (24V I/O) reporting
RPA	Execution position report
RPC	Command motor position report
RPC(...)	Command motor position (specific trajectory) report
RPI	Report pi numbers
RPMA	Execution position modulo report
RPML	Position modulo limit report
RPMT	Position modulo target (position motion) reporting
RPRA	Operation start actual position report
RPRC	Operation start command position report
RPRT	Relative target position report
RPT	Setting target position report
RPTSD	Synchronous linear motion distance report
RPTST	Synchronous operation time report (ms)
RRANDOM	Random number report
RRES	Encoder resolution report
RSAMP	Sample rate (Hz) report
RSIN(..)	Report sine function in degrees
RSLM	Soft limit mode report
RSLN	Soft limit right/forward setting report
R.S.L.P.	Soft limit left/negative direction setting report
RSP	Sample rate, firmware number reporting
RSP1	Firmware compilation date and time report
RSP2	Bootloader revision report
RSQRT(...)	Integer square root value reporting
RT	Request torque value report
RTAN(...)	Report tangent function in degrees
RTEMP	Internal temperature report
RTH	Temperature limit setpoint report
RTHD	Current limit timer setting value report
RTMR(...)	User timer value report
RTRQ	Real-time torque value reporting
RTS	Torque slope setting value report
RUIA	Current value report
RUJA	Voltage value report

RUN	Program execution
RUN?	The program will stop unless a RUN command is sent after the power is turned on.
RVA	Execution speed report (filter value)
R.V.C.	Command speed value report
RVL	Speed limit report
RVT	Specified speed report
RW(...)	Specific status word reporting
S	
S	sudden stop
S(...)	Sudden stop (orbit specification)
SADDR#	Motor addressing
SAMP	Sampling rate (Hz)
SILENT	Ignore commands received on port 1
SILENT1	Ignore commands received on port 2
SIN(...)	Get sine function in degrees
S.L.D.	Disable software limits
SLE	Enable software limits
SLEEP	Start sleep mode on port 1
SLEEP1	Start sleep mode on port 2
SLM	software limit
SLM(...)	Set software limits
SLN	Left direction software limit
SLN=...	Set left direction software limit
SLP	Right direction software limit
SLP=...	Right direction software limit setting
SQRT(...)	integer square root
SRC(...)	Tracking and/or cam encoder source settings
STACK	Nesting reset
STDOUT=...	Set the output destination of the report command
SWITCH...	Program flow command
T	
T	Setting torque value
T=	Torque value setting
TALK	Enable PRINT message output to port 1
TALK1	Enable PRINT message output to port 2
TAN(...)	Get tangent function in degrees
TEMP	temperature
T.H.	temperature limit
TH=...	Temperature limit setting
THD	Obtains a timer until the drive power is turned off when the temperature exceeds (Bh)
THD=	Timer setting until drive power is turned off when temperature exceeds (Bh)
TMR(...)	Specific timer setting time, e.g. a=TMR(0)
TMR(...) (as cmd)	Timer settings, e.g. TMR(0,1000) = set timer 0 to 1 second
T.S.	Torque slope setting value
TS=...	Torque ramp setting
TSWAIT	Synchronous operation in progress Standby
TWAIT	Operating Standby
TWAIT(...)	Operating Standby (orbit specification)
U	
UIA	Motor current value

UJA	bus voltage
UO(...)=...	Set one or more status bits to a specific value
UP	Upload user EEPROM program
UPLOAD	Upload user EEPROM readable program
UR(...)	Set one or more status bits to 0
US(...)	Set one or more status bits to 1
V	
V.A.	execution speed
VAC(...)	Speed filter settings
V.C.	Specified speed
VL	Specified speed limit
VL=...	Speed limit setting
VLD(...)	Reading from non-volatile memory
VST(...)	Writing to non-volatile memory
VT	target speed
VT=...	Set target speed
VTS=...	Set target speed for synchronous operation
W	
W(...)	Report status word
WAIT=...	Wait for specified time (specified in msec)
WAKE	Exit port 1 sleep mode
WAKE1	Exit port 2 sleep mode
WHILE...	infinite loop command
X	
X	deceleration stop
X(...)	Deceleration and stop (specify trajectory)
Z	
Z	Reset all errors
Z(...)	Reset specific errors
Za	Current error reset
Ze	Deviation error reset
Zh	Temperature error reset
Zl	Hardware (sensor input) limit history Left direction status bit release
Zls	Software limit history left direction status bit release
Zr	Hardware (sensor input) limit history Right direction status bit release
Zrs	Software limit history Right direction status bit release
Zs	System tech error bit release
ZS	Clear all errors
Zv	Speed error release
Zw	Encoder value wraparound latch bit reset