Trio Motion Technology Ltd.
Shannon Way, Tewkesbury,
Gloucestershire. GL20 8ND
United Kingdom
Tel: +44 (0)1684 292333
Fax: +44 (0)1684 297929

1000 Gamma Drive
Suite 206
Pittsburgh, PA 15238
United States of America
Tel: +1 412.968.9744
Fax: +1 412.968.9746

Tomson Centre
118 Zhang Yang Rd., B1701
Pudong New Area, Shanghai,
Postal code: 200122
P. R. CHINA
Tel/Fax: +86-21-58797659

Doc No.: **AN-233**
Version: **1.1**
Date: **20 April 2009**
Subject: **CANOpen DS402 in the Motion Coordinator**

APPLICATION NOTE

www.triomotion.com

1. Introduction

The Trio Motion Coordinator series 2xx have the ability to control Servo and Stepper Drives via CANOpen. DS402 is the CANopen device profile for drives and motion control. Within this "profile" there are many options, so when a drive manufacturer announces that a product has DS402 compatibility it is not a guarantee that it will connect to a Motion Coordinator. This document gives details which will allow a system designer to decide whether a particular CANopen based drive/amplifier will function with the Trio CANOpen software implementation.

2. Communication Overview

The Motion Coordinator is a multi-axis motion controller which generates motion profiles, gearing and cam based motion using a defined cyclic time base. On each "tick" of the servo cycle, the Motion Coordinator reads the feedback values of all axes, calculates the required position or speed values and sends them to the drives. When using CANbus, the CAN network cycle time is set to the same as the Motion Coordinator's SERVO_PERIOD.

Figure 1 shows a diagram of the CANbus cycle.

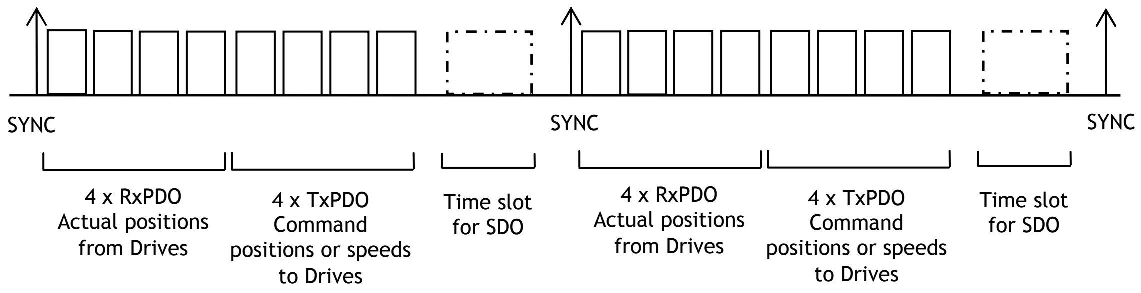
SYNC
SYNC
SYNC
4 x RxPDO Actual positions from Drives
4 x TxPDO Command positions or speeds to Drives
Time slot for SDO
4 x RxPDO Actual positions from Drives
4 x TxPDO Command positions or speeds to Drives
Time slot for SDO

Fig. 1 CANbus cycle – 1Mbit/sec, SYNC telegram sent once per millisecond

3. CAN Drive Characteristics

The CAN and servo cycle described in 2, follows the “slim drive” model. By this we mean that the drive must simply follow the commands given to it by the Motion Coordinator – millisecond by millisecond. This model is the same one that is used in drives which are commanded by +/-10V and give position by encoder feedback. The “slim drive” model does **not** expect the drive to do any intelligent functions like profiling or homing. It **must** be able to follow synchronisation and speed/position telegrams very efficiently.

When selecting a DS402 CANopen servo or stepper drive for use with a Trio Motion Coordinator, these are the minimum requirements for the Drive.

- DS402 Mode. At least one of these modes must be supported. (Object 6060H)

2	Velocity Mode
3	Profile Velocity Mode [(1)]
7	Interpolated Position Mode

- Tx PDO must be either:
  1. Single object, 32 bit, actual position value. (Typical object: 6064H)
  2. Two objects; 16 bit status word, 32 bit actual position value. (Typical objects: 6041H, 6064H)
- Rx PDO must be either:
  1. Single object, 32 bit, position command value. (Typical object: 60C1H, Sub-Index 01H)
  2. Single object, 32 bit, speed command value. (Typical object: 60FFH)
  3. Two objects; 16 bit control word, 32 bit position command value. (Typical objects: 6040H, 60C1H, Sub-Index 01H)
  4. Two objects; 16 bit control word, 32 bit speed command value. (Typical objects: 6040H, 60FFH)
- Tx PDO transmission type must be set to cyclic and synchronous with 1 transmission per SYNC telegram. (Object 180xH = 1)
  - CAN Drives that send their Actual Position only on Change-of-State will **not** be able to work with the Motion Coordinator.
- The drive must have software to synchronise its internal control loop cycle to the SYNC telegram. Usually this is done with a phase-locked-loop in the software of the drive.
  - CAN Drives that do not synchronise their internal servo loop to the CAN SYNC telegram will have “bumps” in the motion occurring cyclically as the clock frequencies beat together.

Also the following are recommendations.

- For Interpolated Position Mode, the interpolation time should be set to the same as the can cycle time. (Object 60C2H)
- One freely configurable PDO is recommended because fixed function PDOs may not fit the required configuration.

Any PDO can be used. The Motion Coordinator is quite flexible and the COB-ID allocated for the PDO of each axis can be modified as required. The size and contents of the PDO are fixed in the Motion Coordinator firmware and PDO data size is determined by which axis type (ATYPE) is set when initialising the axis.

(1) For Profile Velocity Mode to be used, it must be possible to set the acceleration and deceleration ramps to go to the set speed in 1 servo cycle. Speed change must happen without needing a trigger.

4. Axis Type (ATYPE)

In the Motion Coordinator, there is an axis parameter called ATYPE. This parameter tells the system software how to handle the command and feedback data of the axis. For CANOpen the following ATYPEs apply.

- `ATYPE=18` : CANopen position axis
- `ATYPE=19` : CANopen speed axis
- `ATYPE=26` : CANopen position axis, DS402 predefined PDO3*
- `ATYPE=27` : CANopen speed axis, DS402 predefined PDO3*

`* Predefined PDO structure is PDO3 type, but can use other PDO numbers.`

4.1. CANopen Position Axis

ATYPE = 18

Transmit PDO : Single object 32 bit position command value

Receive PDO : Single object 32 bit actual position feedback

Motion Coordinator axis parameters;

DAC_OUT : Shows the value being sent on TxPDO. Follows movement of DPOS.

ENCODER : Shows the value coming in on RxPDO. MPOS follows changes in ENCODER.

Drive CAN settings;

Object Index	Sub Index	Value	Size	Description
6060H	0	7	8 bit	Drive Mode
1600H	1	60C1 0120H	32 bit	Position command object
1600H	0	1	8 bit	Number of objects in RxPDO
1A00H	1	6064 0020H	32 bit	Actual position object
1A00H	0	1	8 bit	Number of objects in TxPDO
1800H	2	1	8 bit	TxPDO synchronous and cyclic
1006H	0	1000	32 bit	Communication cycle period (μsecs)
60C2H	1	1	8 bit	Interpolation time mantissa
60C2H	2	-3	8 bit	Interpolation time exponent

Note: The above are typical settings only. They are given here as a guide.

4.2. CANopen Speed axis

ATYPE = 19

Transmit PDO : Single object 32 bit velocity command value

Receive PDO : Single object 32 bit actual position feedback

Motion Coordinator axis parameters;

DAC_OUT : Shows the value being sent on TxPDO. Output value from PID control loop.

ENCODER : Shows the value coming in on RxPDO. MPOS follows changes in ENCODER.

Drive CAN settings;

Object Index	Sub Index	Value	Size	Description
6060H	0	2	8 bit	Drive Mode
1600H	1	60FF 0020H	32 bit	Speed command object
1600H	0	1	8 bit	Number of objects in RxPDO
1A00H	1	6064 0020H	32 bit	Actual position object
1A00H	0	1	8 bit	Number of objects in TxPDO
1800H	2	1	8 bit	TxPDO synchronous and cyclic
1006H	0	1000	32 bit	Communication cycle period (μsecs)

Note: The above are typical settings only. They are given here as a guide.

4.3. CANopen Position Axis (DS402 Predefined PDO)

ATYPE = 26

Transmit PDO : Two objects :

1. 16 bit control word
2. 32 bit position command value

Receive PDO : Two objects :

1. 16 bit status word
2. 32 bit actual position feedback

Motion Coordinator axis parameters;

DAC_OUT : Shows the value being sent on TxPDO. Follows movement of DPOS.

ENCODER : Shows the value coming in on RxPDO. MPOS follows changes in ENCODER.

DRIVE_CONTOL : Word value is sent cyclically in TxPDO.

DRIVE_STATUS : Word value received cyclically in RxPDO.

Drive CAN settings;

Object Index	Sub Index	Value	Size	Description
6060H	0	7	8 bit	Drive Mode
1600H	1	6040 0010H	32 bit	Controlword object
1600H	2	60C1 0120H	32 bit	Position command object
1600H	0	2	8 bit	Number of objects in RxPDO
1A00H	1	6041 0010H	32 bit	Statusword object
1A00H	2	6064 0020H	32 bit	Actual position object
1A00H	0	2	8 bit	Number of objects in TxPDO
1800H	2	1	8 bit	TxPDO synchronous and cyclic
1006H	0	1000	32 bit	Communication cycle period (μsecs)

Note: The above are typical settings only. They are given here as a guide.

When using this ATYPE, the number of axes per CAN connection at 1 msec servo period is limited to 3. This is due to the longer telegram size for TxPDO and RxPDO. 4 axes can be run at 2 msec servo period.

4.4. *CANopen Speed Axis (DS402 Predefined PDO)*

ATYPE = 27

Transmit PDO : Two objects :

1. 16 bit control word
2. 32 bit velocity command value

Receive PDO : Two objects :

1. 16 bit status word
2. 32 bit actual position feedback

Motion Coordinator axis parameters;

DAC_OUT : Shows the value being sent on TxPDO. Output value from PID control loop.

ENCODER : Shows the value coming in on RxPDO. MPOS follows changes in ENCODER.

DRIVE_CONTOL : Word value is sent cyclically in TxPDO.

DRIVE_STATUS : Word value received cyclically in RxPDO.

Drive CAN settings;

Object Index	Sub Index	Value	Size	Description
6060H	0	2	8 bit	Drive Mode
1600H	1	6040 0010H	32 bit	Controlword object
1600H	2	60C1 0120H	32 bit	Speed command object
1600H	0	2	8 bit	Number of objects in RxPDO
1A00H	1	6041 0010H	32 bit	Statusword object
1A00H	2	6064 0020H	32 bit	Actual position object
1A00H	0	2	8 bit	Number of objects in TxPDO
1800H	2	1	8 bit	TxPDO synchronous and cyclic
1006H	0	1000	32 bit	Communication cycle period (μsecs)

Note: The above are typical settings only. They are given here as a guide.

When using this ATYPE, the number of axes per CAN connection at 1 msec servo period is limited to 3. This is due to the longer telegram size for TxPDO and RxPDO. 4 axes can be run at 2 msec servo period.

5. Example STARTUP program

The following program is a general example. Different drives usually require slightly different settings. For some drives there are objects listed below that cannot be set. For other drives there may be additional objects that require setting.

5.1. Position mode

This example is for a P290 or P293 located in Slot 0 of the Motion Coordinator. CAN PDO messages are to be transferred using PDO2.

Refer to the Motion Coordinator Technical Reference Manual for an explanation of the CAN command functions and parameters.

```
' Initialise axis, CAN baud rate and CAN Identifiers.
WDOG=OFF
SERVO AXIS(0)=0

'set baudrate to 1000kHz
IF CAN(0,2)<>0 OR (CAN(0,0,16) AND 4) THEN CAN(0,2,0)

' Set SYNC message ID in buffer 15:
CAN(0,5,15,$80,0,1)
' Set NMT message ID in buffer 14:
CAN(0,5,14,0,2,1)
' Set drive demand COB-IDs (nodes 1 to 4 in buffers 8-11)
CAN(0,5,8,$301,4,1) ' $201 : PDO1, $301 : PDO2, $401 : PDO3
CAN(0,5,9,$302,4,1)
CAN(0,5,10,$303,4,1)
CAN(0,5,11,$304,4,1)
' Set position receive COB-IDs (nodes 1 to 4 in buffers 4-7)
CAN(0,5,4,$281,8,0) ' $181 : PDO1, $281 : PDO2, $381 : PDO3
CAN(0,5,5,$282,8,0)
CAN(0,5,6,$283,8,0)
CAN(0,5,7,$284,8,0)
ATYPE AXIS(0)=18
AXIS_ADDRESS AXIS(0)=1
OUTLIMIT AXIS(0)=32767

' Set up COB-IDs for SDO messages to node 1
CAN(0,5,13,$601,8,1)
CAN(0,5,1,$581,8,0)

' Set Pre-Operational state on all drives:
CAN(0,7,14,128,0)

' Set the drive into Interpolated Position(7) mode:
IF CAN(0,9 ,13,1,8,$6060,0,7,$0000)=FALSE THEN STOP

' disable PDO2 Rx while we configure it
IF CAN(0,9 ,13,1,8,$1601,0,$0000,$0000)=FALSE THEN STOP
' Set the PDO2 Rx Mapping for Interp. Pos. mode ($60c1, $0120):
IF CAN(0,9 ,13,1,32,$1601,1,$0120,$60c1)=FALSE THEN STOP
' Set PDO2 Rx to cyclic
IF CAN(0,9 ,13,1,8,$1401,2,$0001)=FALSE THEN STOP
' PDO2 Rx enable
IF CAN(0,9 ,13,1,8,$1601,0,$0001)=FALSE THEN STOP

' disable PDO2 Tx while we configure it
IF CAN(0,9 ,13,1,8,$1a01,0,$0000,$0000)=FALSE THEN STOP
' Set the PDO2 Tx Mapping for Actual Position ($6064, $0020)
IF CAN(0,9 ,13,1,32,$1a01,1,$0020,$6064)=FALSE THEN STOP
' Set PDO2 Tx to cyclic
IF CAN(0,9 ,13,1,8,$1801,2,$0001)=FALSE THEN STOP
' PDO2 Tx enable
```

```
IF CAN(0,9 ,13,1,8,$1a01,0,$0001)=FALSE THEN STOP

' Set up interpolation time (1 x 10^-3 seconds)
IF CAN(0,9 ,13,1,8,$60C2,1,1)=FALSE THEN STOP
IF CAN(0,9 ,13,1,8,$60C2,2,-3)=FALSE THEN STOP

' Set CAN update period (1000 microsecs)
IF CAN(0,9,13,1,32,$1006,0,$03e8,$0000) THEN STOP
' Set sync window length
IF CAN(0,9,13,1,32,$1007,0,$0200,$0000) THEN STOP

' Set Operational Mode on all drives:
CAN(0,7,14,1,0)

' Enable the cyclic CAN command transmission from the controller
DRIVE_ENABLE AXIS(0)=1

' Some drives require watchdog ON to enable correctly
WDOG=ON : WA(100)

' Drive Enable sequence
' Disable
IF CAN(0,9 ,13,1,16,$6040,0,$0000,$0000)=FALSE THEN STOP
' Reset Fault
IF CAN(0,9 ,13,1,16,$6040,0,$0080,$0000)=FALSE THEN STOP
' Enable voltage and quick stop
IF CAN(0,9 ,13,1,16,$6040,0,$0006,$0000)=FALSE THEN STOP
' enable operation and switch on
IF CAN(0,9 ,13,1,16,$6040,0,$000f,$0000)=FALSE THEN STOP
' enable interpolated position mode
IF CAN(0,9 ,13,1,16,$6040,0,$001f,$0000)=FALSE THEN STOP
```

5.2. Speed mode - DS402 Predfined PDO

This example is for a P290 or P293 located in Slot 1 of the Motion Coordinator. CAN PDO messages are to be transferred using PDO3. CAN cycle time is set to 2 msecs so SERVO_PERIOD must be set to 2000.

Refer to the Motion Coordinator Technical Reference Manual for an explanation of the CAN command functions and parameters.

```
' Initialise axis, CAN baud rate and CAN Identifiers.
WDOG=OFF
SERVO AXIS(0)=0

'set baudrate to 1000kHz
IF CAN(1,2)<>0 OR (CAN(1,0,16) AND 4) THEN CAN(1,2,0)

' Set SYNC message ID in buffer 15:
CAN(1,5,15,$80,0,1)
' Set NMT message ID in buffer 14:
CAN(1,5,14,0,2,1)

' Set drive demand COB-IDs (nodes 1 to 4 in buffers 8-11)
CAN(1,5,8,$401,4,1) ' $201 : PDO1, $301 : PDO2, $401 : PDO3
CAN(1,5,9,$402,4,1)
CAN(1,5,10,$403,4,1)
CAN(1,5,11,$404,4,1)
' Set position receive COB-IDs (nodes 1 to 4 in buffers 4-7)
CAN(1,5,4,$381,8,0) ' $181 : PDO1, $281 : PDO2, $381 : PDO3
CAN(1,5,5,$382,8,0)
CAN(1,5,6,$383,8,0)
CAN(1,5,7,$384,8,0)
ATYPE AXIS(5)=27
AXIS_ADDRESS AXIS(5)=1
OUTLIMIT AXIS(5)=32767

' Set up COB-IDs for SDO messages to node 1
CAN(1,5,13,$601,8,1)
CAN(1,5,1,$581,8,0)

' Set Pre-Operational state on all drives:
CAN(1,7,14,128,0)

' Set the drive into Velocity (2) mode:
IF CAN(1,9,13,1,8,$6060,0,2,$0000)=FALSE THEN STOP

' disable PDO2 Rx while we configure it
IF CAN(1,9,13,1,8,$1602,0,0)=FALSE THEN STOP
' Set the PDO2 Rx Mapping for Controlword ($6040, $0010):
IF CAN(1,9,13,1,32,$1602,1,$0010,$6040)=FALSE THEN STOP
' Set the PDO2 Rx Mapping for Speed ($60FF, $0020):
IF CAN(1,9,13,1,32,$1602,2,$0120,$60c1)=FALSE THEN STOP
' Set PDO2 Rx to cyclic
IF CAN(1,9,13,1,8,$1402,2,1)=FALSE THEN STOP
' PDO2 Rx enable (with 2 objects)
IF CAN(1,9,13,1,8,$1602,0,2)=FALSE THEN STOP

' disable PDO2 Tx while we configure it
IF CAN(1,9,13,1,8,$1a02,0,0)=FALSE THEN STOP
' Set the PDO2 Tx Mapping for Statusword ($6041, $0010)
IF CAN(1,9,13,1,32,$1a02,1,$0010,$6041)=FALSE THEN STOP
' Set the PDO2 Tx Mapping for Actual Position ($6064, $0020)
IF CAN(1,9,13,1,32,$1a02,1,$0020,$6064)=FALSE THEN STOP
' Set PDO2 Tx to cyclic
IF CAN(1,9,13,1,8,$1802,2,1)=FALSE THEN STOP
' PDO2 Tx enable (2 objects in PDO)
IF CAN(1,9,13,1,8,$1a02,0,1)=FALSE THEN STOP

' Set CAN update period (2000 microsecs)
IF CAN(1,9,13,1,32,$1006,0,$07d0,$0000) THEN STOP
' Set sync window length
IF CAN(1,9,13,1,32,$1007,0,$0400,$0000) THEN STOP

' Set Operational Mode on all drives:
CAN(1,7,14,1,0)

' Enable the cyclic CAN command transmission from the controller
DRIVE_ENABLE AXIS(5)=1

' Some drives require watchdog ON to enable correctly
WDOG=ON
WA(100)
```

```
' Drive Enable sequence uses DRIVE_CONTROL because of ATYPE=27
BASE(0)
' Disable
DRIVE_CONTROL=$00
WA(2)
' Reset Fault
DRIVE_CONTROL=$80
WA(2)
' Enable voltage and quick stop
DRIVE_CONTROL=$06
WA(2)
' enable operation and switch on
DRIVE_CONTROL=$0f
```