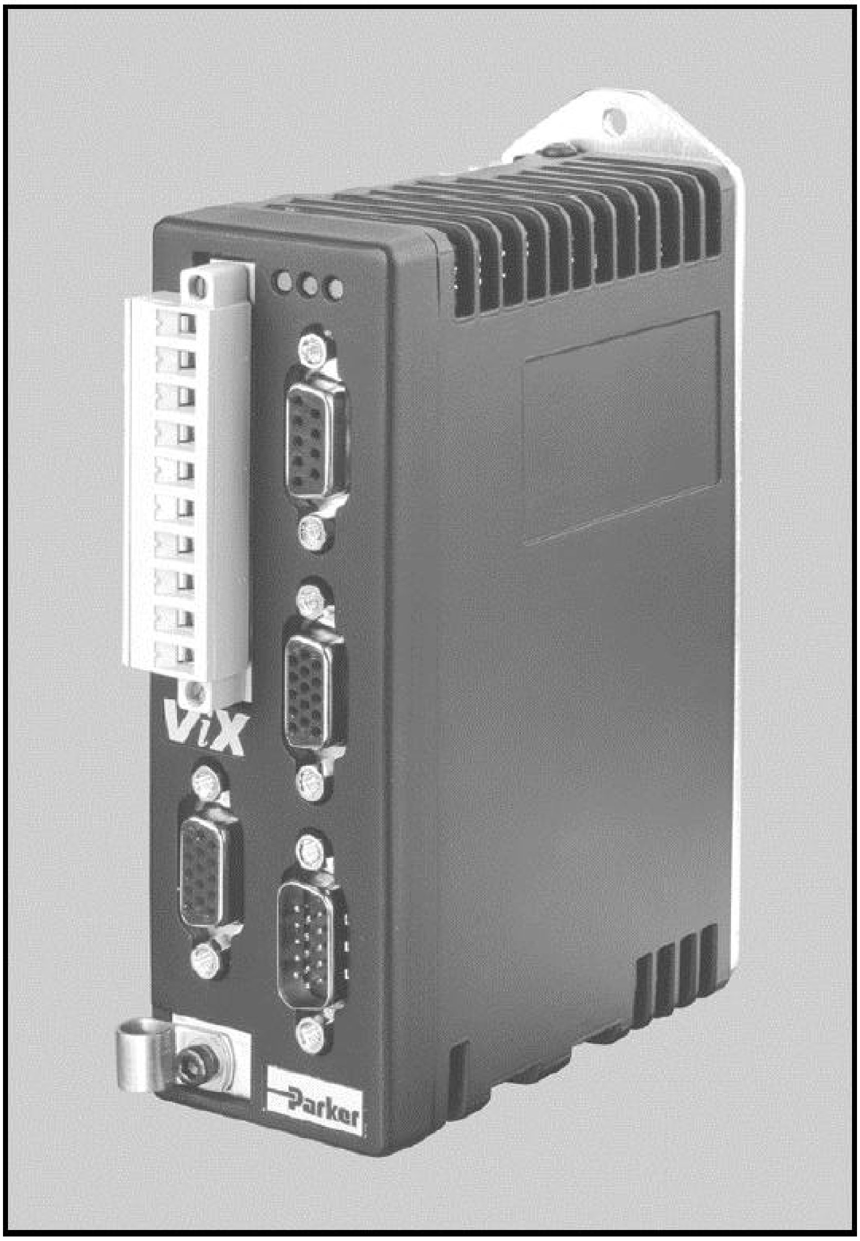
Parker Automation

ViX CANOpen Series

Digital Drives

User Guide



Parker

Automation

CE

ViX250CE, ViX500CE, ViX250CM, ViX500CM, ViX250CH & ViX500CH CANopen User Guide

Part No: 1600.330.02, June 2005

IMPORTANT INFORMATION FOR USERS

Installation and Operation of Motion Control Equipment

It is important that motion control equipment is installed and operated in such a way that all applicable safety requirements are met. It is your responsibility as an installer to ensure that you identify the relevant safety standards and comply with them; failure to do so may result in damage to equipment and personal injury. In particular, you should study the contents of this user guide carefully before installing or operating the equipment.

The installation, set-up, test and maintenance procedures given in this User Guide should only be carried out by competent personnel trained in the installation of electronic equipment. Such personnel should be aware of the potential electrical and mechanical hazards associated with mains-powered motion control equipment - please see the safety warning below. The individual or group having overall responsibility for this equipment must ensure that operators are adequately trained.

Under no circumstances will the suppliers of the equipment be liable for any incidental, consequential or special damages of any kind whatsoever, including but not limited to lost profits arising from or in any way connected with the use of the equipment or this user guide.

⚠SAFETY WARNING

High-performance motion control equipment is capable of producing rapid movement and very high forces. Unexpected motion may occur especially during the development of controller programs. ***KEEP WELL CLEAR*** of any machinery driven by stepper or servo motors. Never touch any part of the equipment while it is in operation.

This product is sold as a motion control component to be installed in a complete system using good engineering practice. Care must be taken to ensure that the product is installed and used in a safe manner according to local safety laws and regulations. In particular, the product must be enclosed such that no part is accessible while power may be applied.

If the equipment is used in any manner that does not conform to the instructions given in this user guide, then the protection provided by the equipment may be impaired.

The information in this user guide, including any apparatus, methods, techniques, and concepts described herein, are the proprietary property of Parker EME or its licensors, and may not be copied, disclosed, or used for any purpose not expressly authorised by the owner thereof.

Since Parker EME constantly strives to improve all of its products, we reserve the right to modify equipment and user guides without prior notice. No part of this user guide may be reproduced in any form without the prior consent of Parker EME.

Contact Addresses

Parker Hannifin plc
Electromechanical Automation
Arena Buisness Centre, Holy Rood Close,
Poole, Dorset. BH17 7BA UK
Tel: +44 (0)1202 606300
Fax: +44 (0)1202 606301
Website : www.parker-eme.com
e-mail : sales.digiplan@parker.com

Parker Hannifin GmbH
Electromechanical Automation
Robert-Bosch-Str. 22
D-77656 Offenburg, Germany
Tel: +49 (0)781 509-0
Fax: +49 (0)781 509-98176
Website : www.parker-eme.com
e-mail : sales.hauser@parker.com

Parker Hannifin S. p. A
Electromechanical Automation
Via Gounod 1
I-20092 Cinisello Balsamo (MI), Italy
Tel: +39 0266012459
Fax: +39 0266012808
Website : www.parker-eme.com
e-mail : sales.sbc@parker.com

Parker Hannifin Corporation
Compumotor Division
5500 Business Park Drive, Suite D
Rohnert Park
CA 94928, USA
Tel: +1 (800) 358-9070
Fax: +1 (707) 584-3793
FaxBack System: (800) 936-6939
e-mail: CMR_help@parker.com
Website: www.compumotor.com

Contents

User Guide Change Summary

This user guide, version 1600.330.02, is the second version of the ViX CANopen User Guide. It should be noted that there has been extensive modifications by the applications team at Parker EME. So much so that the standard vertical line used formerly to indicate changes has been omitted. It is hoped that the reader will not compare this manual to the previous version.

Associated Documentation

CIA Draft Standard 301 Version 3.0
CIA Draft Standard Proposal 402 Version 1.1
Hauser COMPAX-M/S Bus-Option: CANopen

1. Introduction

User Guide Assumptions

This user guide assumes you have a working knowledge of CANopen Fieldbus Protocol and you are familiar with the programming and operation of motion control equipment. The guide is intended as a reference only.

Structure of the User Guide

The guide is presented in six sections, summarised below:

Section 1: Introduction

This section, which introduces you to the structure and scope of CANopen used with Parker ViX drives.

Section 2: Software Requirements

Provides an introduction to CANopen as implemented within the ViX drive.

Section 3: External I/O Modules

Describes the use of I/O modules for use on the CANOpen fieldbus.

Section 4: Object Types

Describes the type of Data Object used in CANopen. This section includes SDO and PDO definitions combined with configuration and mapping parameters.

Section 5: Object Library

Describes the various forms of Object that are used. These are sub-divided into the following types:

Communication objects:	0x1000 to 0x1A01
Manufacturer specific objects:	0x2004 to 0x21A7
Device Profile objects:	0x603F to 0x6504

Section 6: State Machine

The state diagram for CANopen used with Parker ViX drives.

Appendices

Appendix 1 contains an ASCII table. This is for the users reference. There are several objects that return ASCII strings and it has been included to help.
Appendix 2 contains the DS-301 state diagram, again this is used for user reference.
Appendix 3 TxPDO and RxPDO transmission types. This is included to show the user the differences in transmission types and the associated ViX node behaviour.

Index

Customer Feedback

2. Software Requirements

Overview

The CANopen Fieldbus is designed for the motion and control market. The fast data rates and data formatting make it ideal for closing low bandwidth control loops with remote feedback devices. It can support up to 127 nodes at up to 500metres distance. Baud rates range from 20kHz up to 1MHz. Each message can consist of up to 8 bytes of information.

The communications objects for setting the COB-ID, SDOs and PDOs are described in the CIA draft standard 301.

The standard for the device profile (DSP-402) has been followed at the application level for configuring the modes of operation and various other commands where applicable.

In order to satisfy the functionality of ViX stepper and servo products, a set of manufacturer specific objects have been implemented. These objects allow further drive profile configuration and allow the use of the specific modes of operation.

Scope of CAN Bus Control

The objects and their implementation are described later in this guide. Further details are also given of the software operation and flow. It may be worth considering at this point some of the system limitations for those not needing to know any more than the basics to get a system running.

The objects do not provide facility for defining labels or some of the higher level report facilities. The intention is that the drive is pre-programmed with labels and sequences using EASI-V and the RS232 serial port. This configuration can be saved in each unit prior to being incorporated into a FieldBus system. A facility is provided to send commands and data over the CANopen protocol in ASCII format (object 0x2005). This will only accept one command per message. The RS232 port should only be used as a maintenance port. The operation of the software cannot be guaranteed if both the RS232 and CANopen ports are used at the same time.

The CanOpen implementation makes use of an object dictionary that is immediately updated on a data write access. This maintains the high data transfer rates. The interface software tokenises the data and places it in a buffer to await execution. The buffer is accessed using the FIFO principle. As a result of this, an actual write to a parameter such as acceleration may update the object dictionary immediately but the actual value change may not take place in the application for several milliseconds, depending on what is already in the buffer. The minimum time for update would be one millisecond regardless of any impending token in the buffer.

The data read, however, is taken directly from the application variable. The read gives an immediate indication of the **currently programmed** value. This may cause some confusion, as the parameter may be loaded with one value but, because the buffer is still executing previous commands, may read back a different value. To be sure that the buffer is clear it is advisable to read the system status first.

Software Settings

Four parameters relevant to the CanOpen protocol are initiated and saved via the RS232 port. The parameters Node-ID, Baud rate, Protocol and Control are configured using ASCII or CanOpen commands and then saved. The settings are then automatically loaded on the next power cycle.

Parameter	System Variable	Range	Comments	Effective From
Node ID	FN	1 ….127	Default is 99	Power up
Baudrate	FB	0 ….1000	See table (below)	Power up
Protocol	FP	0 ….255	See table (below)	Power up
Control	FC	0 ….255	See table (below)	Power up

The address of the node can be specified as different to the controller address but must be unique on the Can bus.

The node address can be configured via the RS232 ASCII link by entering the command:

nW(FN,x)

Where **n** is the axis (drive address) **x** is the required CanOpen node address

The parameter can be checked by reading back the data, that is: **nR(FN)**.

These variables are also accessible over CanOpen via object 0x2008 and the relevant sub-index. Care must be taken when using these variables as wrong settings may cause the CanOpen link to fail.

FieldBus Node ID (FN)

The default node ID is 99. The Node ID can be set via RS232 this is done by using the following command:

nW(FN,X)

Where **n** is the axis (drive address), **X** is the node address 1 - 127

It can be set so that the CAN node address automatically takes the axis address by setting the fieldbus control (see below). After a change of value of Node ID a **SAVE** (1SV) must be executed and power cycled to make the new data valid.

FieldBus Baud Rate (FB)

The parameters for baud rate are limited to the following settings:

Parameter Value	20	50	100	125	250	500	800	1000
Selected Baud Rate	20000	50000	100000	125000	250000	500000	800000	1000000

The required baud rate for CanOpen can be configured via the RS232 ASCII link by entering the command:

nW(FB,x)

Where **n** is the axis (drive address), **x** is the parameter value for the required baud rate shown above.

The parameter can be checked by reading back the data i.e., **nR(FB)**.

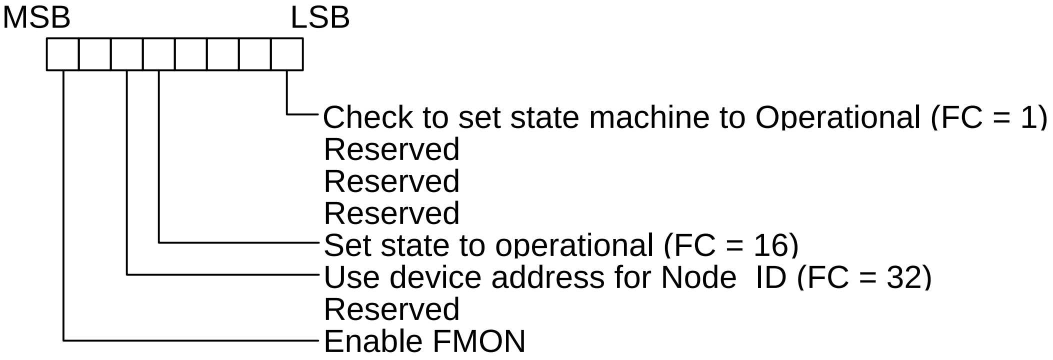
Both of the above set-up parameters can be defined over CanOpen by using the object 0x2008 (Variable Configuration) with the appropriate sub-index. The revised values will not become active until the controller has been reset.

Fieldbus Protocol (FP)

This variable is used for setting fieldbus communication options and has currently only one parameter. Bit 1 is set to indicate that when node guarding is selected, a message is displayed over RS232 to indicate a change in status of the node guarding. This is relevant to the port implementation of node-guarding only.

Fieldbus Control (Variable FC)

The options for functionality selected using Fieldbus Control are as listed below:


MSB
LSB
Check to set state machine to Operational (FC = 1)
Reserved
Reserved
Reserved
Set state to operational (FC = 16)
Use device address for Node ID (FC = 32)
Reserved
Enable FMON

State Machine and State Machine over-ride

Access to motion control parameters, or any associated parameter, must have the device state machine set to operation-enabled before being used. Example parameters are velocity and acceleration. Less obvious parameters are those to do with running labels as these may command motion within their routines.

The CanOpen specification defines a state machine to achieve operational state to allow the motion control parameters. Bit 5 of the FC (i.e. FC = 16) will set the state machine to Operational mode immediately.

Set State to Operational

The CanOpen specification defines a second state machine for the type of message that will be accepted as valid.

State/Service	SDO	PDO	EMCY	TIME	SYNC	NMT	Error Control	Boot Up
Initialisation	-	-	-	-	-	-	-	X
Stopped	-	-	-	-	-	X	X	-
Pre-Operational	X	-	X	X	X	X	X	-
Operational	X	X	X	X	X	X	X	-

For PDOs to be active the state machine must be Operational. With Bit 5 of the Fieldbus Control set the internal state machine will be set to Operational and will make PDOs valid without the need for a master axis to send an NMT message.

Use Device Address as Node ID

When this bit is set the saved Node ID is over-written and the device address is used as the Node ID. The parameter is only valid on power-up or after a reset.

3. External I/O Modules[1]

Overview

Parker has introduced a range of I/O modules for use on the CANOpen FieldBUS. The implementation of CANOpen on the ViX product range has been enhanced to include the monitor and trigger on condition of external I/O modules. The command structure has been made such that the status of the inputs and outputs can be can be accessed through two new variables IE and OE. To date the FMON command supports up to 32 digital inputs and 32 outputs. Analogue I/O is not yet supported.

This command has been designed with the Parker PIO in mind although so long as the I/O device does not send more than 32 bits within its' TxPDO the device will be compatible.

The CANOpen ViX drive has been equipped with a simple NMT capable of setting the chosen node into the 'Operational' state. This can be achieved using the 'FC' variable. It is recommended that upon completion of a correct FMON command that the user sets the 'FC' variable correctly.

With reference to the 'FC' command the MSB and LSB should be set to enable and hence run external CANOpen I/O.

1W(FC,129)

On sending this command, save (SV) and reset (Z) the unit. The user will notice that the drive will then go through the boot sequence, the 'FB' LED will flash for 3 seconds and then remain on. This is confirmation of correct operation of the NMT, the user should also notice that the I/O 'RUN' or 'STATUS' LED should stay on.

Configuration

A single command is used to configure the input source.

aFMON(remote_node_ID,remote_inputs,remote_outputs)

where:

a	is the axis number of the ViX product
remote_node_ID	is the node number of the input module.
remote_inputs	is the number of expected inputs.
remote_outputs	is the number of expected outputs.

The configured parameters must be saved and will become active on the next power cycle. The range of the inputs are as follows: **8, 16** and **32**. It is important this value is either greater than or equal to the actual number used.

To check the configuration a single command will report the state of both the inputs and outputs.

[1] For the interested user, some more information and FAQ's can be found in Appendix 4

aFMON

where:

a is the axis number of the ViX product

An example using this is shown below:

```
1FMON
*FMON(3,16,16)
```

Reporting the received data for each input is prompted with a single command.

aIS1

where:

a is the axis number of the ViX product

The value is reported as a bit pattern, an example response is shown below, where input 1 is the first. It can be seen that the input 8 is set on.

```
1IS1
*0000_0001_0000_0000_0000_0000_0000_0000
```

If the module number is omitted then the **'IS'** command will revert to its RS232 state and report the ViX product input status as a bit pattern. It should be noted that to check each bank of the external inputs then the **'IE'** command should be used, please refer to the section 'Decision Making on External I/O'.

Reporting and setting the current output status is also covered by a single command.

aOE

where:

a is the axis number of the ViX product

Unlike the **'IS'** command the information is reported back in a hexadecimal format. The example below shows how to set the last four bits i.e. the first four outputs of the CAN I/O and report back the current output status.

```
1OE(000F)
1OE
*0x0000000F
```

It should be noted that the CAN outputs function in a similar way to the **'O'** command. This command is immediate and not saveable thus when the drive is reset or the 24V logic lost the outputs are set to ***zero***.

Decision Making on External I/O

It is possible to use the **'IF'** and the **'TR'** commands with the external I/O to affect program flow. This can only be done as a bit mask for banks of 8 bits at once and applies to both the external inputs **'IE'** and external outputs **'OE'**.

Using a Mask (Bit Pattern)

The format of the command is no different to the standard **'IF'** or **'TR'** test. The structure of the command is:

aTR(IEn,cond,val) or **aIF(IEn,cond,val)**
aTR(OEn,cond,val) or **aIF(OEn,cond,val)**

where;

a	is the axis number of the ViX product
n	is the bank number of the 8-bit external I/O to be addressed, this can be IE1 .. 4 or OE1 .. 4.
cond	is the condition to be executed to the following value for the next line to execute, this condition can be = or <>.
val	is entered as an 8 bit mask (binary) for the comparison, bits can be denoted in the mask as 0, 1 or X 'don't care'.

Example: **1TR(IE1,=,XXXXXXX1)** or **1IF(IE2,=,11XX0XX1)**
1TR(OE1,=,1XX0XXX1) or **1IF(OE3,=,100X0XXX)**

4. Object Types

Addressing

Every object type is accessed through a Communication OBject IDentifier (COB-ID). The COB-ID is made up from a function code representing the object type followed by a seven-bit device address.

Communication Object Identifier (COB-ID)

Function Code	Device Address (Node ID): 1 ... 127

The COB-ID also defines the priority of the message with the highest priority going to the lowest COB-ID. The following table gives an overview of the object availability.

Object Type	Function Code (Bin)	COB-ID (Hex)	Defined Index (Hex)	Description
Broadcast Objects				
NMT	0000	0x000	-	Network Management
SYNC	0001	128 (0x080)	0x1005	COB-ID of the SYNC object
TIME	0010	256 (0x100)	0x1012	
Peer to Peer Objects				
EMCY	0001	129 – 255 (0x081 - 0x0FF)	0x1014	Emergency (fault)
TxPDO1	0011	385 - 511 (0x181 - 0x1FF)	0x1800	Allocated Index 1A00h, 1st Transmit PDO
TxPDO2	0101	641 - 767 0x281 - 0x2FF	0x1801	Allocated Index 1A01h, 2nd Transmit PDO
RxPDO1	0100	513 - 639 0x201 - 0x27F	0x1400	Allocated Index 1600h, 1st Receive PDO
RxPDO2	0110	769 - 895 0x301 - 0x37F	0x1401	Allocated Index 1601h, 2nd Receive PDO
TxSDO1	1011	1409 - 1535 0x581 - 0x5FF	0x1200	Transmit Service Data Object 1
RxSDO1	1100	1537 - 1663 0x601 - 0x67F	0x1200	Receive Service Data Object 1
NMT Error Control	1110	1793 - 1919 0x701 - 0x77F	0x100E	Node Guarding checking Bus integrity

Service Data Messages

The Parker EME implementation of CANopen supports a single transmit Service Data Object (TxSDO1) and a single receive SDO (RxSDO1). The configuration and addressing are shown below.

SDO Configuration

0x1200 SDO Configuration Object Details

Index	Sub Index	Name	Object Code	Elements	Attribute	PDO Mapping
0x1200	00	Server SDO Parameter	Array	3	RO	No

0x1200.00 SDO Configuration Number of Entries

Index	Sub Index	Name	Type	Attribute	Object Values		
					Default	Minimum	Maximum
0x1200	00	Number of Entries	Unsigned8	RO	0x02	0x02	0x03

0x1200.01 SDO Configuration RxSDO1 COB ID

Index	Sub Index	Name	Type	Attribute	Object Values		
					Default	Minimum	Maximum
0x1200	01	RxSDO1	Unsigned32	RO	0x600 + Node ID	0x601	0x67F

This object specifies the COB ID of the SDO parameter. This is calculated automatically by the drive. The direction of this SDO is: **Bus Master → ViX Node.**

0x1200.02 SDO Configuration TxSDO1 COB ID

Index	Sub Index	Name	Type	Attribute	Object Values		
					Default	Minimum	Maximum
0x1200	02	TxSDO1	Unsigned32	RO	0x580 + Node ID	0x581	0x5FF

This object specifies the COB ID of the SDO parameter. This is calculated automatically by the drive. The direction of this SDO is: **ViX Node → Bus Master.**

These data messages are used for read and write access to all entries of the object dictionary. Messages of this type are relatively slow and where possible the use of PDOs is suggested. For example the 0x2004 and 0x2007 would be ideal objects to be sent by SDO.

Process Data Messages (PDO)

The Parker EME implementation of CANopen supports up to two transmit Process Data Objects (TxPDO) and two receive PDOs (RxPDO). PDOs are sent with no protocol overhead and are therefore very fast. They are ideal for real-time data to be transferred. They can be programmed to be cyclic or acyclic. They can be configured by using SDOs.

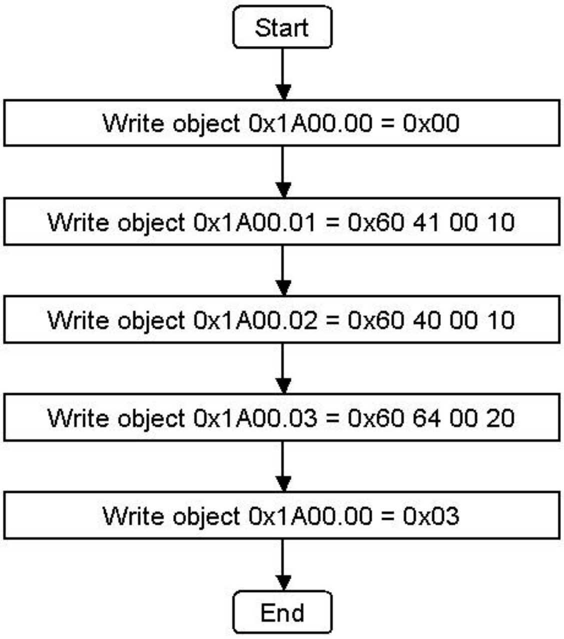
Example PDO Mapping RxPDO1

From the ‘Software Requirements – Overview’ section the user will remember that the ViX can support 8 data bytes of information mapped per PDO. To aid understanding an example has been written below.

We already have a default mapping for the TxPDO (0x1A00) first entry we will use this and two others in this example.

Index	Sub Index	Object	PDO Entry	PDO Length	Byte Count
0x6041	00	Status Word	60 41 00 10	00 00	2 bytes
0x6040	00	Control Word	60 40 00 10	00 00	2 bytes
0x6064	00	Position Actual	60 64 00 20	00 00 00 00	4 bytes

The correct operation to perform this mapping is shown in the flow diagram below.


Start
Write object 0x1A00.00 = 0x00
Write object 0x1A00.01 = 0x60 41 00 10
Write object 0x1A00.02 = 0x60 40 00 10
Write object 0x1A00.03 = 0x60 64 00 20
Write object 0x1A00.00 = 0x03
End

When the NMT is used to start the node, it can be seen that the packet sent is 8 bytes long.

RxPDO1 Configuration (Bus Master → ViX Node)

0x1400 RxPDO1 Configuration Object Details

Index	Sub Index	Name	Object Code	Elements	Attribute	PDO Mapping
0x1400	00	RxPDO1 Parameter	Array	4	-	No

0x1400.00 RxPDO1 Configuration Number of Entries

Index	Sub Index	Name	Type	Attribute	Object Values		
					Default	Minimum	Maximum
0x1400	00	Number of Entries	Unsigned8	RO	0x03	0x02	0x03

0x1400.01 RxPDO1 Configuration RxPDO1 COB ID

Index	Sub Index	Name	Type	Attribute	Object Values		
					Default	Minimum	Maximum
0x1400	01	COB-ID of RxPDO1	Unsigned32	RW	0x200 + Node ID	0x201	0x27F

0x1400.02 RxPDO1 Configuration Transmission Type

Index	Sub Index	Name	Type	Attribute	Object Values		
					Default	Minimum	Maximum
0x1400	02	Transmission Type	Unsigned8	RW	0xFE	0x00	0xFF

For further information on the transmission types, please refer to Appendix 3.

0x1400.03 RxPDO1 Configuration Inhibit Time

Index	Sub Index	Name	Type	Attribute	Object Values		
					Default	Minimum	Maximum
0x1400	03	Inhibit Time	Unsigned16	RW	0x03E8	0x0000	0xFFFF

It is important to remember that this value is specified in units where: 1 unit = 100µs.

RxPDO2 Configuration (Bus Master → ViX Node)

0x1401 RxPDO2 Configuration Object Details

Index	Sub Index	Name	Object Code	Elements	Attribute	PDO Mapping
0x1401	00	RxPDO2 Parameter	Array	4	-	No

0x1401.00 RxPDO2 Configuration Number of Entries

Index	Sub Index	Name	Type	Attribute	Object Values		
					Default	Minimum	Maximum
0x1401	00	Number of Entries	Unsigned8	RO	0x03	0x02	0x03

0x1401.01 RxPDO2 Configuration RxPDO2 COB ID

Index	Sub Index	Name	Type	Attribute	Object Values		
					Default	Minimum	Maximum
0x1401	01	Number of Entries	Unsigned32	RW	0x300 + Node ID	0x301	0x37F

0x1401.02 RxPDO2 Configuration Transmission Type

Index	Sub Index	Name	Type	Attribute	Object Values		
					Default	Minimum	Maximum
0x1401	02	Transmission Type	Unsigned8	RW	0xFE	0x00	0xFF

For further information on the transmission types, please refer to Appendix 3.

0x1401.03 RxPDO2 Configuration Inhibit Time

Index	Sub Index	Name	Type	Attribute	Object Values		
					Default	Minimum	Maximum
0x1401	02	Inhibit Time	Unsigned16	RW	0x03E8	0x0000	0xFFFF

It is important to remember that this value is specified in units where: 1 unit = 100µs.

RxPDO1 Mapping Parameter

0x1600 RxPDO1 Mapping Object Details

Index	Sub Index	Name	Object Code	Elements	Attribute	PDO Mapping
0x1600	00	RxPDO1 Parameter	Array	5	-	No

0x1600.00 RxPDO1 Mapping Number of Entries

Index	Sub Index	Name	Type	Attribute	Object Values		
					Default	Minimum	Maximum
0x1600	00	Number of Entries	Unsigned8	RW	0x02	0x00	0x04

0x1600.01 RxPDO1 Mapping PDO Mapping Entry 1

Index	Sub Index	Name	Type	Attribute	Object Values		
					Default	Minimum	Maximum
0x1600	01	PDO Mapping Entry	Unsigned32	RW	0x60400010	0x00000000	0xFFFFFFFF

This PDO has a default mapping of 0x6040 the ViX Control Word.

0x1600.02 RxPDO1 Mapping PDO Mapping Entry 2

Index	Sub Index	Name	Type	Attribute	Object Values		
					Default	Minimum	Maximum
0x1600	02	PDO Mapping Entry	Unsigned32	RW	0x60640020	0x00000000	0xFFFFFFFF

This PDO has a default mapping of 0x6064 the ViX Position Actual value.

0x1600.03 RxPDO1 Mapping PDO Mapping Entry 3

Index	Sub Index	Name	Type	Attribute	Object Values		
					Default	Minimum	Maximum
0x1600	03	PDO Mapping Entry	Unsigned32	RW	0x00000000	0x00000000	0xFFFFFFFF

0x1600.04 RxPDO1 Mapping PDO Mapping Entry 4

Index	Sub Index	Name	Type	Attribute	Object Values		
					Default	Minimum	Maximum
0x1600	04	PDO Mapping Entry	Unsigned32	RW	0x00000000	0x00000000	0xFFFFFFFF

RxPDO2 Mapping Parameter

0x1601 RxPDO2 Mapping Object Details

Index	Sub Index	Name	Object Code	Elements	Attribute	PDO Mapping
0x1601	00	RxPDO2 Parameter	Array	5	-	No

0x1601.00 RxPDO2 Mapping Number of Entries

Index	Sub Index	Name	Type	Attribute	Object Values		
					Default	Minimum	Maximum
0x1601	00	Number of Entries	Unsigned8	RW	0x02	0x00	0x04

0x1601.02 RxPDO2 Mapping PDO Mapping Entry 1

Index	Sub Index	Name	Type	Attribute	Object Values		
					Default	Minimum	Maximum
0x1601	01	PDO Mapping Entry	Unsigned32	RW	0x00000000	0x00000000	0xFFFFFFFF

0x1601.02 RxPDO2 Mapping PDO Mapping Entry 2

Index	Sub Index	Name	Type	Attribute	Object Values		
					Default	Minimum	Maximum
0x1601	02	PDO Mapping Entry	Unsigned32	RW	0x00000000	0x00000000	0xFFFFFFFF

0x1601.03 RxPDO2 Mapping PDO Mapping Entry 3

Index	Sub Index	Name	Type	Attribute	Object Values		
					Default	Minimum	Maximum
0x1601	03	PDO Mapping Entry	Unsigned32	RW	0x00000000	0x00000000	0xFFFFFFFF

0x1601.04 RxPDO2 Mapping PDO Mapping Entry 4

Index	Sub Index	Name	Type	Attribute	Object Values		
					Default	Minimum	Maximum
0x1601	04	PDO Mapping Entry	Unsigned32	RW	0x00000000	0x00000000	0xFFFFFFFF

Objects Available for RxPDO Mapping

Index	Sub Index	Object	System Variable	Attributes	Byte Count	PDO Entry
0x2004	00	Control	-	WO	2	20 04 00 10
0x2050	00	Incremental Position	PI	RW	4	20 50 00 20
0x2051	00	Position Error	PE	RW	4	20 51 00 20
0x2052	00	Position Target	PT	RW	4	20 52 00 20
0x6040	00	Control Word	-	RW	2	60 40 00 10
0x6064	00	Position Actual	PA	RW	4	60 64 00 20
0x607A	00	Target Position	D	RW	4	60 7A 00 20
0x6081	00	Profile Velocity	V	RW	4	60 81 00 20
0x6300	00	Output Word	O	RW	2	63 00 00 10

TxPDO1 Configuration (ViX Node → Bus Master)

0x1800 TxPDO1 Configuration Object Details

Index	Sub Index	Name	Object Code	Elements	Attribute	PDO Mapping
0x1800	00	TxPDO1 Parameter	Array	6	-	No

0x1800.00 TxPDO1 Configuration Number of Entries

Index	Sub Index	Name	Type	Attribute	Object Values		
					Default	Minimum	Maximum
0x1800	00	Number of Entries	Unsigned8	RO	0x05	0x02	0x05

0x1800.01 TxPDO1 Configuration TxPDO1 COB ID

Index	Sub Index	Name	Type	Attribute	Object Values		
					Default	Minimum	Maximum
0x1800	01	COB-ID of TxPDO1	Unsigned32	RW	0x180 + Node ID	0x181	0x1FF

0x1800.02 TxPDO1 Configuration Transmission Type

Index	Sub Index	Name	Type	Attribute	Object Values		
					Default	Minimum	Maximum
0x1800	02	Transmission Type	Unsigned8	RW	0xFE	0x00	0xFF

For further information on the transmission types, please refer to Appendix 3.

0x1800.03 TxPDO1 Configuration Inhibit Time

Index	Sub Index	Name	Type	Attribute	Object Values		
					Default	Minimum	Maximum
0x1800	03	Inhibit Time	Unsigned16	RW	0x03E8	0x0000	0xFFFF

It is important to remember that this value is specified in units where: 1 unit = 100µs.

0x1800.04 TxPDO1 Configuration Reserved

Index	Sub Index	Name	Type	Attribute	Object Values		
					Default	Minimum	Maximum
0x1800	04	Reserved	Unsigned8	N/A	0x00	0x00	0xFF

0x1800.05 TxPDO1 Configuration Event Timer

Index	Sub Index	Name	Type	Attribute	Object Values		
					Default	Minimum	Maximum
0x1800	05	Event Timer	Unsigned16	RW	0x000A	0x0000	0xFFFF

Asynchronous TxPDOs can be transmitted cyclically with the event timer. If its value is greater than 0, it becomes a millisecond timer. When this is expired, the PDO is transmitted. Transmission therefore takes place both when an external device input is altered and when the event timer is lapsed.

TxPDO2 Configuration (ViX Node → Bus Master)

0x1801 TxPDO2 Configuration Object Details

Index	Sub Index	Name	Object Code	Elements	Attribute	PDO Mapping
0x1801	00	TxPDO2 Parameter	Array	6	-	No

0x1801.00 TxPDO2 Configuration Number of Entries

Index	Sub Index	Name	Type	Attribute	Object Values		
					Default	Minimum	Maximum
0x1801	00	Number of Entries	Unsigned8	RO	0x05	0x02	0x05

0x1801.01 TxPDO2 Configuration TxPDO2 COB ID

Index	Sub Index	Name	Type	Attribute	Object Values		
					Default	Minimum	Maximum
0x1801	01	COB-ID of TxPDO2	Unsigned32	RW	0x280 + Node ID	0x281	0x2FF

0x1801.02 TxPDO2 Configuration Transmission Type

Index	Sub Index	Name	Type	Attribute	Object Values		
					Default	Minimum	Maximum
0x1801	02	Transmission Type	Unsigned8	RW	0xFE	0x00	0xFF

For further information on the transmission types, please refer to Appendix 3.

0x1801.03 TxPDO2 Configuration Inhibit Time

Index	Sub Index	Name	Type	Attribute	Object Values		
					Default	Minimum	Maximum
0x1801	02	Inhibit Time	Unsigned16	RW	0x03E8	0x0000	0xFFFF

It is important to remember that this value is specified in units where: 1 unit = 100µs.

0x1801.04 TxPDO2 Configuration Reserved

Index	Sub Index	Name	Type	Attribute	Object Values		
					Default	Minimum	Maximum
0x1800	04	Reserved	Unsigned8	N/A	0x00	0x00	0xFF

0x1801.05 TxPDO2 Configuration Event Timer

Index	Sub Index	Name	Type	Attribute	Object Values		
					Default	Minimum	Maximum
0x1800	05	Event Timer	Unsigned16	RW	0x000A	0x0000	0xFFFF

Asynchronous TxPDOs can be transmitted cyclically with the event timer. If its value is greater than 0, it becomes a millisecond timer. When this is expired, the PDO is transmitted. Transmission therefore takes place both when an external device input is altered and when the event timer is lapsed.

TxPDO1 Mapping Parameter

0x1A00 TxPDO1 Mapping Object Details

Index	Sub Index	Name	Object Code	Elements	Attribute	PDO Mapping
0x1A00	00	TxPDO1 Parameter	Array	5	-	No

0x1A00.00 TxPDO1 Mapping Number of Entries

Index	Sub Index	Name	Type	Attribute	Object Values		
					Default	Minimum	Maximum
0x1A00	00	Number of Entries	Unsigned8	RW	0x02	0x00	0x04

0x1A00.01 TxPDO1 Mapping PDO Mapping Entry 1

Index	Sub Index	Name	Type	Attribute	Object Values		
					Default	Minimum	Maximum
0x1A00	01	PDO Mapping Entry	Unsigned32	RW	0x60410010	0x00000000	0xFFFFFFFF

This PDO has a default mapping of 0x6041 the ViX Status Word.

0x1A00.02 TxPDO1 Mapping PDO Mapping Entry 2

Index	Sub Index	Name	Type	Attribute	Object Values		
					Default	Minimum	Maximum
0x1A00	02	PDO Mapping Entry	Unsigned32	RW	0x00000000	0x00000000	0xFFFFFFFF

0x1A00.03 TxPDO1 Mapping PDO Mapping Entry 3

Index	Sub Index	Name	Type	Attribute	Object Values		
					Default	Minimum	Maximum
0x1A00	03	PDO Mapping Entry	Unsigned32	RW	0x00000000	0x00000000	0xFFFFFFFF

0x1A00.04 TxPDO1 Mapping PDO Mapping Entry 4

Index	Sub Index	Name	Type	Attribute	Object Values		
					Default	Minimum	Maximum
0x1A00	04	PDO Mapping Entry	Unsigned32	RW	0x00000000	0x00000000	0xFFFFFFFF

TxPDO2 Mapping Parameter

0x1A01 TxPDO2 Mapping Object Details

Index	Sub Index	Name	Object Code	Elements	Attribute	PDO Mapping
0x1A01	00	TxPDO2 Parameter	Array	5	-	No

0x1A01.00 TxPDO2 Mapping Number of Entries

Index	Sub Index	Name	Type	Attribute	Object Values		
					Default	Minimum	Maximum
0x1A01	00	Number of Entries	Unsigned8	RW	0x00	0x00	0x04

0x1A01.02 TxPDO2 Mapping PDO Mapping Entry 1

Index	Sub Index	Name	Type	Attribute	Object Values		
					Default	Minimum	Maximum
0x1A01	01	PDO Mapping Entry	Unsigned32	RW	0x00000000	0x00000000	0xFFFFFFFF

0x1A01.02 TxPDO2 Mapping PDO Mapping Entry 2

Index	Sub Index	Name	Type	Attribute	Object Values		
					Default	Minimum	Maximum
0x1A01	02	PDO Mapping Entry	Unsigned32	RW	0x00000000	0x00000000	0xFFFFFFFF

0x1A01.03 TxPDO2 Mapping PDO Mapping Entry 3

Index	Sub Index	Name	Type	Attribute	Object Values		
					Default	Minimum	Maximum
0x1A01	03	PDO Mapping Entry	Unsigned32	RW	0x00000000	0x00000000	0xFFFFFFFF

0x1A01.04 TxPDO2 Mapping PDO Mapping Entry 4

Index	Sub Index	Name	Type	Attribute	Object Values		
					Default	Minimum	Maximum
0x1A01	04	PDO Mapping Entry	Unsigned32	RW	0x00000000	0x00000000	0xFFFFFFFF

Objects Available for Transmit PDO Mapping

Index	Sub Index	Object	System Variable	Attributes	Byte Count	PDO Entry
0x2007	01	Analogue Input	AI	RO	2	20 07 01 10
0x2007	03	Indexer Status	ST	RO	4	20 07 03 10
0x2007	05	Ready / Busy Flag	RB	RO	1	20 07 05 10
0x2007	06	Position Registration	PR	RO	4	20 07 06 10
0x2050	00	Incremental Position	PI	RW	4	20 50 00 20
0x2051	00	Position Error	PE	RW	4	20 51 00 20
0x2052	00	Position Target	PT	RW	4	20 52 00 20
0x6040	00	Control Word	-	RW	2	60 40 00 10
0x6041	00	Status Word	ST	RO	2	60 41 00 10
0x6064	00	Position Actual	PA	RO	4	60 64 00 20
0x606C	00	Velocity	V	RO	4	60 6C 00 20
0x607A	00	Target Position	D	RW	4	60 7A 00 20
0x6081	00	Profile Velocity	V	RW	4	60 81 00 20
0x6100	00	Input Word	IS	RO	2	61 00 00 10
0x6300	00	Output Word	O	RW	2	63 00 00 10

5. Object Library

Communication Objects (DS-301 V 3.0)

Object Table

Object	Object Name	Sub Indexes	Object Type	Attributes
0x1000	Device Type	00	Unsigned32	RO
0x1001	Error Register	00	Unsigned8	RO
0x1003	Pre Defined Error Field	01	Unsigned32	RW
0x1008	Manufacturers Device Name	00	Visible String	RO
0x1009	Manufacturers Hardware Version	00	Visible String	RO
0x100A	Manufacturers Software Version	00	Visible String	RO
0x100C	Guard Time	00	Unsigned16	RW
0x100D	Life Time Factor	00	Unsigned8	RW
0x1014	COB-ID Emergency Object	00	Unsigned32	RW
0x1018	Identity Object	01	Unsigned32	RO
0x1200	1st Server SDO	03	Various	RO
0x1400	1st RxPDO Parameter	03	Various	RW
0x1401	2nd RxPDO Parameter	03	Various	RW
0x1600	1st RxPDO Mapping	04	Various	RW
0x1601	2nd RxPDO Mapping	04	Various	RW
0x1800	1st TxPDO Parameter	05	Various	RW
0x1801	2nd TxPDO Parameter	05	Various	RW
0x1A00	1st TxPDO Mapping	04	Various	RW
0x1A01	2nd TxPDO Mapping	04	Various	RW

General Notes

The table (above) shows the Objects that are implemented in ViX products configured for CANopen protocol. The implementation of these objects is based on the Draft Standard of the Communication Profile DS-301 version 3.0.

0x1000 Device Type

Index	Sub-Index	Name	Type	Attribute	Default Value
0x1000	00	Device Type	Unsigned32	RO	00 02 01 92 – ViX Servo Drive
					00 04 01 92 – ViX Stepper Drive

The ’01 92’ refers to the device profile for drives and motion control. This profile number is 402_d. The ’00 02’ and ’00 04’ refer to the additional information that can be specified. In the case of ’00 02’ this specifies a Servo Drive while ’00 04’ specifies a Stepper Drive.

0x1001 Error Register

Index	Sub-Index	Name	Type	Attribute	Default Value
0x1001	00	Error Register	Unsigned8	RO	N/A

This register contains internal errors. This register is also part of the emergency message. Further information as to the layout has been shown below. In the event of an error bit ‘0’ is always set.

Bit	Meaning
0	General Error
1	Current
2	Voltage
3	Temperature
4	Communication
5	Device Profile Specific
6	Reserved
7	Manufacturer Specific

0x1003 Pre-defined Error Field

Index	Sub-Index	Name	Type	Attribute	Default Value
0x1003	00	Number of Errors	Unsigned8	RW	00
	01	Standard Error Field	Unsigned32	RO	00 00 00 00

The sub-index 0 contains the errors currently stored in the field. If a new error occurs it will be entered in sub-index 1 and all existing errors are moved down by one. Since the ViX only has one sub-index only the last error is stored. The error memory is deleted by writing a ‘0’ into sub-index 00.

The error field follows the standard design as shown below:

Bit 31 Bit 15	Bit 0
Additional Information	Error Code

0x1008 Manufacturer Device Name

Index	Sub-Index	Name	Type	Attribute	Default Value
0x1008	00	Manufacturer Device Name	Visible String	RO	See below

The object indicates the name of the ViX node the user is currently addressing. The following values are returned:

Default Value	String Translation
43 45 2D 53 65 72 76 6F 20 77 69 74 68 20 43 41 4E 00	CE-Servo with CAN
43 48 2D 53 65 72 76 6F 20 77 69 74 68 20 43 41 4E 00	CH-Servo with CAN
43 4D 2D 53 74 65 70 70 65 72 20 77 69 74 68 20 43 41 4E	CM-Stepper with CAN

0x1009 Manufacturer Hardware Revision

Index	Sub-Index	Name	Type	Attribute	Default Value
0x1009	00	Manufacturer Hardware Revision	Visible String	RO	See below

The object indicates the hardware type of the ViX node the user is currently addressing. The following values are returned:

Default Value	String Translation
56 69 58 32 35 30 43 45	VIX250CE
56 69 58 32 35 30 43 4D	VIX250CM
56 69 58 32 35 30 43 48	VIX250CH
56 69 58 35 30 30 43 45	VIX500CE
56 69 58 35 30 30 43 4D	VIX500CM
56 69 58 35 30 30 43 48	VIX500CH

0x100A Manufacturer Software Revision

Index	Sub-Index	Name	Type	Attribute	Default Value
0x100A	00	Manufacturer Software Revision	Visible String	RO	See text

The object indicates the software revision code of the ViX node the user is currently addressing. The information is consistent with the 1R(RV) command when used within EASI-V. An example response has been shown below:

Example Response	String Translation
32 2E 34 43 00 43	2.4C C

0x100C Guard Time

Index	Sub-Index	Name	Type	Attribute	Default Value
0x100C	00	Guard Time	Unsigned16	RW	0

The object indicates the time in milli-seconds that a CAN bus master cyclically interrogates the CAN slave for its status. The time between two interrogations is termed as the guard time.

0x100D Life Time Factor

Index	Sub-Index	Name	Type	Attribute	Default Value
0x100D	00	Life Time Factor	Unsigned8	RW	0

This factor is part of the **Node Guarding Protocol**. The CAN slave checks if it was interrogated within the **Node Life Time** (guard time multiplied with the life time factor). If not, the slave works on the basis that the NMT master is no longer in its normal operation. It then triggers a **Life Guarding Event**. If the node lifetime is zero, no monitoring will take place.

To reduce the load on the processor, it is recommended that the Node Guarding Protocol is not used.

0x1014 COB-ID Emergency Object

Index	Sub-Index	Name	Type	Attribute	Default Value
0x1014	00	COB-ID EMCY	Unsigned32	RW	0x80 + Node ID

This object defines the COB-ID for the ‘EMCY’ message.

0x1018 Identity Object

Index	Sub-Index	Name	Type	Attribute	Default Value
0x1018	00	Max. Supported Entries	Unsigned8	RO	01
	01	Manufacturer ID	Unsigned32	RO	00 00 00 89
	02	Device Description	Unsigned32	RO	00 00 00 00
	03	Revision Number	Unsigned32	RO	00 00 00 00
	04	Serial Number	Unsigned32	RO	00 00 00 00

This object defines the device used. Any Parker product has been given the number 0x89. The other fields are left blank. Please refer to 0x1008, 0x1009 and 0x100A for further product information.

Manufacturer Specific Objects

Object Table

The following object table defines the objects using the CANopen protocol that are specific to stepper products. A more detailed description of each object follows.

Index	Sub Indexes	Object	System Variable	Object Type	Attributes	PDO Mapping	Notes
0x2004	00	CONTROL	N/A	Unsigned8	WO	Rx	
0x2005	00	ASCII Command	N/A	Visible String	WO	N/A	
0x2007	06	Variable Read	N/A	Various	RO	Various	
0x2008	13	Variable Configure	N/A	Various	RW	No	
0x2050	00	Incremental Position	PI	Integer32	RW	Rx / Tx	
0x2051	00	Position Error	PE	Integer32	RW	Rx / Tx	
0x2052	00	Target Position	PT	Integer32	RW	Rx / Tx	
0x2060	00	Filter Time	FT	Unsigned8	RW	No	Servo
0x2061	00	Feed forward Gain	GF	Unsigned32	RW	No	Servo
0x2062	00	Integral Gain	GI	Unsigned32	RW	No	Servo
0x2063	00	Proportional Gain	GP	Unsigned32	RW	No	Servo
0x2064	00	Velocity Gain	GV	Unsigned32	RW	No	Servo
0x2100	00	ARM Command	ARM	Unsigned8	RW	No	
0x2153	00	GOTO Command	GOTO	Visible String	WO	No	
0x2155	02	LOOP Command	LOOP	Various	WO	No	
0x21A0	03	POSMAIN Command	POSMAIN	Various	RW	No	Stepper
0x21A1	06	PROFILE Command	PROFILE	Various	RW	No	
0x21A2	04	REG Command	REG	Various	RW	No	
0x21A3	02	STALL Command	STALL	Various	RW	No	Stepper
0x21A4	00	USE	USE	Unsigned8	WO	No	
0x21A5	02	FOLLOW Command	FOLLOW	Various	RW	No	
0x21A6	03	BRAKE Command	BRAKE	Various	RW	No	Servo
0x21A7	00	FRATE Command	FRATE	Unsigned8	RW	No	

0x2004 CONTROL Execute Commands

Index	Sub Index	Name	Type	Attribute	Object Values			PDO Mapping
					Default	Minimum	Maximum	
0x2004	00	Control	Unsigned8	WO	00	0x81	0xA4	RxPDO

The CONTROL word executes commanded actions as defined by the table overleaf. The data range is set in the high band of the 8-bit value to be compliant with other Parker-EME products.

The actions on receiving the values have been listed in the table below along with the equivalent ASCII command.

Data Value		Commanded Action	System Variable
Dec	Hex		
128	80	Reserved	
129	81	Go Command	GO
130	82	Go Home	GH
131	83	Kill	K
132	84	De-Energise	OFF
133	85	Energise	ON
134	86	Stop	S
135-139	87 – 8B	Reserved	
140	8C	Toggle Direction	H
141	8D	Positive Direction	H+
142	8E	Negative Direction	H-
143	8F	Reserved	
144	90	Exit Program Loop	EXIT
145-160	91 – A0	Reserved	
161	A1	Clear **ALL** of program memory	CLEAR(ALL)
162	A2	Return to Factory Settings	RFS
163	A3	Save	SV
164	A4	Reset	Z
165-255	A5 – FF	Reserved	

The Pause and Continue commands are intended for pausing a move and then continuing when ready. This function is not currently supported on stepper products. The Pause command implemented in standard product code is for ‘Pausing’ execution from the command buffer. As the implementation of CANOpen is for faster code execution, the commands would seem inappropriate for inclusion here.

The Clear All command will clear ***all*** labels from memory but a save must be executed to make this permanent.

These commands will respond as if entered via an ASCII terminal. The CANopen protocol will still be valid over the CAN interface but the responses, if any, to the commands will be transmitted over RS232 (e.g. the command to Save will respond with the checksum sent over RS232 after successful completion).

When executing the RFS command (value 162) the CAN interface will continue communication even though the fieldbus node identity and the fieldbus baud rate default values may be reported back as different to required. The new settings do not come into effect until the power is re-cycled. The default parameters must be saved before cycling the power.

0x2005 COMMAND Execute ASCII formatted command

Index	Sub-Index	Name	Type	Attribute	Default Value
0x2005	00	Command	Visible String	WO	00

Data details

Coding Format	ASCII	Accepted Data	0x20 ... 0x7F

Example

Command	System Command	ASCII String
Energise Axis 1	1ON	31 4F 4F
Goto START Program Axis 1	1GOTO(START)	31 47 4F 54 4F 28 53 54 41 52 54 29

Commands in ASCII format can be sent over the CAN bus with this object. Enough characters are allowed to define the longest command (PROFILE). This is an inefficient use of the CAN protocol as several messages have to be sent in order for the full command to be read (in the case of defining a profile). The maximum number of data bytes per message is 8. In order to send 56 characters, 9 separate messages require transmitting and receiving. This is transparent to the user, but the overhead in using the Fieldbus may affect other devices connected to the bus.

When using this object the ASCII data must be in upper case and preceded by the axis address (note this may be different to the node-ID). An invalid command (example a command sent in lower case) will result in an error being displayed over RS232.

The RS232 port is still an option for configuring the drive and setting up labels before placing the drive onto the fieldbus. Remember to use SAVE after configuration and before recycling power.

0x2007 VR Access read variable data

Index	Sub-Index	Name	Type	Attribute	PDO Mapping
0x2007	00	VR	Various	RO	See Below

Data details

Index	Sub Index	Object	System Variable	Object Type	Attributes	Default Value
0x2007	00	Number of entries	N/A	Unsigned8	RO	0x06
0x2007	01	Analogue Input	AI	Integer16	RO	N/A
0x2007	02	Drive Fault Status	DF	Unsigned32	RO	N/A
0x2007	03	Indexer Status	ST	Unsigned32	RO	N/A
0x2007	04	User Fault	UF	Unsigned32	RO	N/A
0x2007	05	Ready / Busy Flag	RB	Unsigned8	RO	N/A
0x2007	06	Position Registration	PR	Integer32	RO	N/A

The parameters in the table above can be addressed by using the sub-index. The sub-index is used to call the relevant parameter to read the object. Limited data checking is carried out during the data transfer.

The equivalent ASCII command for the above is nR(*system variable*) or nW(*system variable*, x) where n is the axis number and the system variable is that shown in the table above

0x2007.00 VR Number of Entries

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x2007	00	Number of Entries	Unsigned8	RO	0x00	0x06	No

This sub index reports back the number of entries used in the object 0x2007.

0x2007.01 VR Analogue Input (AI)

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x2007	01	Analogue Input	Integer16	RO	0xF800	0x07FF	TxPDO

The actual value read on the ADC is reported back by this sub-index (1). The value may be altered by change on the hardware analogue input or fine-tuned by altering the value of the offset.

It is important to remember the negative value is generated using a ‘twos complement’.

0x2007.02 VR Drive Fault Status (DF)

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x2007	02	Drive Fault Status	Unsigned32	RO	0x00000000	0xFFFFFFFF	No

Sub-index 2 defines the drive fault status. The status is latched and is not effected by reading the status. The normal method for clearing a drive fault is to remove the original fault condition and then execute an energise command to clear the fault flags.

Note

To date the information reported back from this register is in a different state to that when read using EASI-V. The following example should explain.

EASI-V Report	1R(DF)	1000_0000	1010_0000	0000_0000	0000_0000
CANOpen Report	0x2007.02	01	05	00	00

Thus by observation, it can be seen that the CANOpen value reports the data in an inverted form.

0x2007.03 VR Indexer Status (ST)

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x2007	03	Indexer Status	Unsigned32	RO	0x00000000	0xFFFFFFFF	TxPDO

Sub-index 3 reports back the indexer status. The individual bits of the status will indicate true whilst the condition exists. See table for further information

0x2007.04 VR User Fault (UF)

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x2007	04	User Fault	Unsigned32	RO	0x00000000	0xFFFFFFFF	No

The user fault status is reported back by sub-index 4. Reading the fault status will cause the fault flags to be reset and the status cleared. Faults will stay valid until read and so there may be a flag from a previous fault still set. See table for further information.

0x2007.05 VR Ready / Busy Flag (RB)

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x2007	05	Ready / Busy Flag	Unsigned8	RO	0x00	0x01	TxPDO

This flag shows when the drive is either Ready or Busy. With the motor stationary, this bit is set to ‘0’ when the motor is moving this is set to ‘1’. It should be noted that this flag can be used for a checking system but is different to the Moving / Not Moving or MV system variable.

0x2007.06 VR Position Registration (PR)

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x2007	06	Reg Capture	Integer32	RO	0x00000080	0x7FFFFFFF	TxPDO

Sub-index 6 reports the position recorded at the point of a registration interrupt being received.

0x2008 VC Configure Variable Data

Index	Sub-Index	Name	Type	Attribute	PDO Mapping
0x2008	00	VC	Various	RO	See Below

Data details

Index	Sub Index	Object	System Variable	Object Type	Attributes	Default Value
0x2008	00	Number of entries	N/A	Unsigned8	RO	0x0D
0x2008	01	Analogue Offset	AO	Integer16	RW	0x00
0x2008	02	Analogue Dead band	AB	Integer16	RW	0x00
0x2008	03	Encoder Input	EI	Unsigned8	RW	0x02
0x2008	04	Encoder Output	EO	Unsigned8	RW	0x02
0x2008	05	Motor Standby	MS	Unsigned8	RW	0x0A
0x2008	06	Current Clamp	CL	Unsigned8	RW	0x64
0x2008	07	Comms Response Style	EX	Unsigned8	RW	0x03
0x2008	08	Limit Mask	LIMITS	Unsigned32	RW	0x00000000
0x2008	09	FieldBUS Baud rate	FB	Unsigned16	RW	0x00
0x2008	10	FieldBUS Protocol	FP	Unsigned8	RW	0xE8
0x2008	11	FieldBUS Node Address	FN	Unsigned8	RW	0x63
0x2008	12	FieldBUS Control	FC	Unsigned8	RW	
0x2008	13	S Curve Enable	SC	Unsigned8	RW	

The parameters in the table above can be addressed by using the sub-index. The sub-index is used to call the relevant parameter to modify the object dictionary. Limited data checking is carried out during the data transfer. Care must be taken when downloading new parameters to avoid ambiguous values, as the fault detection software is not user-friendly.

The equivalent ASCII command for the above is nR(*system variable*) or nW(system variable, x) where n is the axis number and the mnemonic is that shown in the table above

0x2008.00 VC Number of Entries

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x2008	00	Number of Entries	Unsigned8	RO	0x00	0x0D	No

This sub index reports back the number of entries used in the object 0x2008.

0x2008.01 VC Analogue Offset (AO)

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x2008	01	Analogue Offset	Integer16	RW	0xF801	07FF	No

This object allows the user to offset the differential analogue speed / torque control input.

0x2008.02 VC Analogue Dead Band (AB)

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x2008	02	Analogue Dead band	Integer16	RW	0x00	0xFF	No

This object allows the user to widen or narrow the region in which the analogue control system ignores the analogue input. This is useful if the analogue source is noisy when at zero position.

0x2008.03 VC Encoder Input (EI)

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x2008	03	Encoder Input	Unsigned8	RW	0x00	0x02	No

This object controls encoder when connected on the X4 connector. The table below shows the operation of the EI variable. The default value is 0x02.

X4	EI = 0	EI = 1	EI = 2
12	STEP+	CW+	A+
7	STEP-	CW-	A-
13	DIR+	CCW+	B+
8	DIR-	CCW-	B-

0x2008.04 VC Encoder Output (EO)

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x2008	04	Encoder Output	Unsigned8	RW	0x00	0x02	No

This object controls encoder output when connected on the X4 connector. The table below shows the operation of the EO variable. The default value is 0x02.

X4	EO = 0	EO = 1	EO = 2
14	STEP+	CW+	A+
9	STEP-	CW-	A-
15	DIR+	CCW+	B+
10	DIR-	CCW-	B-

0x2008.05 VC Motor Standby (MS) [Stepper Only]

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x2008	05	Motor Standby	Unsigned8	RW	0x0A	0x64	No

When the motor is stationary, reduce its current to minimise heating or to conserve power. MS sets the reduction in current as a percentage of the programmed current (the value set in the MOTOR command). When selected, the drive will switch to standby 25mS after the last motor step.

Motor standby current reduction is capped at a value of 70% of the drive's maximum output current. Consequently, if you attempt to set an MS value greater than 70 the current reduction value will always be equal to 70% of the drive's maximum output current. For example, using a ViX500 (max. output current of 5.6A) and setting MS to 90 will give a current reduction value of 4A (70% of 5.6A).

0x2008.06 VC Current Clamp (CL) [Servo Only]

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x2008	06	Current Clamp	Unsigned8	RW	0x01	0x64	No

This object limits the current output of the drive to protect low current motors or to set a particular torque level, and Peak Current (PC) can allow a controlled boost of motor current when required.

CL can be set as a percentage (1 to 100%) of the peak drive current and once set drive output current cannot be exceeded using any other command or system variable. The default value is 0x64.

0x2008.07 VC Comms Response Style (EX)

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x2008	07	Comms Response Style	Unsigned8	RW	0x00	0x07	No

System variable EX controls the style and protocol of the drive’s serial communications link. This does not affect the CAN parameters only the RS232. There are 4 options listed in the table below.

Value	Parameter	Action
0	Echo Off	Speak when spoken to, default for RS485 comms
1	Echo Off	Speak whenever
2	Echo On	Speak when spoken to
3	Echo On	Speak whenever, default for RS232 comms
4	Echo Off	See description for ‘6’
5	Echo Off	See description for ‘7’
6	Echo Off	Speak when spoken to, only echo responses
7	Echo Off	Speak whenever, only echo responses

It is useful to note that with the ***Echo On*** the drive transmits characters received so that commands may be passed to other axes in a RS232 chain while with ***Echo Off*** the drive doesn’t transmit any characters received.

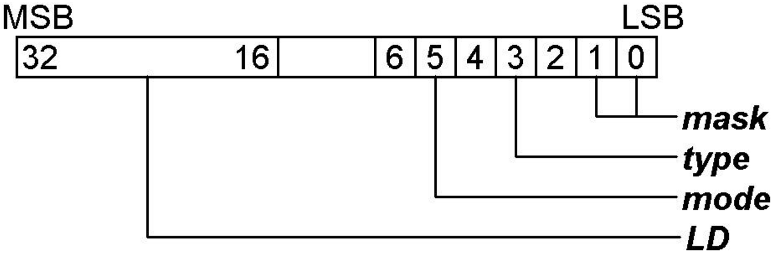
If the parameter is set to ***Speak Whenever*** then the drive will transmit a message if required, for example, **E* when a limit is hit, without being specifically requested. This mode is dangerous to use in a daisy chain RS232 application as it could corrupt a valid message.

0x2008.08 VC Limit Mask (LIMITS)

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x2008	08	Limit Mask	Unsigned32	RW	0x10000	0xFFFF003F	No

This object is identical to the EASI-V LIMITS command. This entry is made of several values, these are the limit deceleration (***LD***), the mode of operation (***mode***), the type of limits (***type***) and the bit mask to enable or disable them (***mask***).

aLIMITS(mask,type,mode,*LD*)



mode	***type***	***mask***		**Action**
Bit 5	**Bit 3**	**Bit 1**	**Bit 0**	
		0	0	Positive and Negative limits enabled
		0	1	Positive limit enabled Negative limit disabled
		1	0	Negative limit enabled Positive limit disabled
		1	1	Positive and Negative limits disabled
	0			Limit normally open
	1			Limit normally closed
0				Stop motion when a limit is hit but continue the program
1				Stop motion when a limit is hit and abort the program. Then run the FAULT program

Example

EASI-V Command - 1LIMITS(0,1,0,100)

- Enable both limits
- Normally open switches
- Stop motion when a limit is hit
- Deceleration rate of 100 revs^{-2}

CANOpen Response 0x27 10 00 28

We can calculate that 0x2710 is our LD value. This is equal to 10,000 thus the value used within EASI-V has been multiplied by 100.

We must then convert 0x28 into a binary pattern and thus we see 0b00101000. Using the table above, we can confirm the following:

Bit 5	True	Stop motion when a limit is hit and abort the program
Bit 3	True	Normally closed switches
Bit 2 & Bit 1	False	Enable both limit switches

0x2008.09 VC FieldBUS Baud Rate (FB)

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x2008	09	FieldBUS Baud Rate	Unsigned16	RW	0x014	0x3E8	No

This object allows the user to defile the baud rate. The values are in kHz (i.e, 1000 = 1000kHz or 1MHz). Although this parameter can be written to and changed, it will not become active until a power cycle or software reset is carried out.

0x2008.0A VC FieldBUS Protocol (FP)

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x2008	0A	FieldBUS Protocol	Unsigned8	RW	0x00000000	0xFF	No

This object allows the user to control the state machine within the ViX drive. Please refer to the earlier section for more information.

0x2008.0B VC FieldBUS Node (FN)

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x2008	0B	FieldBUS Node	Unsigned8	RW	0x00	0x7F	No

This object defines the node identity. Although this parameter can be written to and changed, it will not become active until a power cycle or software reset is carried out.

0x2008.0C VC FieldBUS Control (FC)

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x2008	0C	FieldBUS Control	Unsigned8	RW	0x00	0xFF	No

This object defines the fieldbus control variable. Please refer to the earlier section for more information.

0x2008.0D VC S Curve Enable (SC)

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x2008	0D	S Curve Enable	Unsigned8	RW	0x00	0x01	No

This object writes to the SC variable and hence enables s-curve operation during acceleration and deceleration.

0x2050 PI Incremental Position

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x2050	00	Incremental Position	Integer32	RW	0x80000000	0x7FFFFFFF	RxPDO / TxPDO

This object reports the distance moved by the last move (G) command.

0x2051 PE Position Error

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x2051	00	Position Error	Integer32	RW	0x80000000	0x7FFFFFFF	RxPDO / TxPDO

This object reports the position error, that is, the difference between PT and PA.

0x2052 PT Position Target

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x2051	00	Position Target	Integer32	RW	0x80000000	0x7FFFFFFF	RxPDO / TxPDO

This object reports the target position of the motor, that is, where you have commanded the motor to move to.

0x2060 FT Filter Time (Servo Only)

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x2051	00	Filter Time	Unsigned8	RW	0x00	0xFF	No

Fast positioning systems need high proportional and velocity gains. By limiting the bandwidth, the digital filter prevents a high gain system from becoming too lively. The filter also serves to average the effects of the digital control loop, reducing the jitter at standstill and the audible noise. The value of FT should be kept as low as possible. The arbitrary units used to set the value of FT cannot be directly related to any time value.

0x2061 GF Feed Forward Gain (Servo Only)

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x2061	00	Feed Forward Gain	Unsigned32	RW	0x000	0x3FF	No

The opposing action of proportional and velocity gains result in a position error that depends on speed. This is called ‘following error’. Feed forward gain can be used to offset the following error and improve tracking accuracy. This is important in contouring applications.

0x2062 GI Integral Gain (Servo Only)

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x2062	00	Integral Gain	Unsigned32	RW	0x000	0x3FF	No

Proportional action may be insufficient to overcome static position errors caused by gravitational load effects. Integral action accumulates a steady state error until sufficient torque is produced to move the load. It improves overall positioning accuracy but may produce low frequency oscillation around the commanded position.

0x2063 GP Proportional Gain (Servo Only)

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x2063	00	Proportional Gain	Unsigned32	RW	0x000	0x3FF	No

Proportional gain determines the amount of torque produced in response to a given position error. It sets the stiffness of the system and affects the following error. A high proportional gain gives a stiff, responsive system but results in overshoot and oscillations that require damping.

0x2064 GV Velocity Gain (Servo Only)

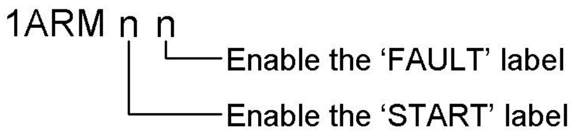
Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x2064	00	Velocity Gain	Unsigned32	RW	0x000	0x3FF	No

Velocity feedback is a signal that increases with shaft speed. It acts in a negative sense opposing the proportional action and helping to stabilise the motion. The damping action of velocity feedback allows a higher proportional gain to be used.

0x2100 ARM ARM Command

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x2100	00	ARM Command	Unsigned8	RW	0x00	0x80	No

The **ARM** command allows you to enable (arm) or disable (disarm) the START label. It also enables/disables the FAULT label. This has been shown in the diagram below.



Naturally this forms a binary pattern of 0, 1, 2 and 3, this has been explained in the table below.

Index	Sub Index	Value	System Command	Action
0x2100	00	0x00	1ARM00	START label does not run on power up FAULT is not used if the drive goes into error
	00	0x01	1ARM01	FAULT is used if the drive goes into error
	00	0x02	1ARM10	START label does run on power up.
	00	0x03	1ARM11	Both used

0x2153 GOTO GOTO ASCII Program Label

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x2153	00	GOTO Command	Visible String	WO	N/A	N/A	No

This object allows the user to directly move to a pre-defined program label. The limitations to this are as follows:

1. The label must have been defined within the EASI-V program using the DECLARE function.
2. The label must be not longer than 5 ASCII characters.
3. The ASCII commands must be in upper-case.

Example

In order to move to the START label it is necessary to use the following format. The user only need send the NAME of the program. Thus in the example below only the word ‘START’ is sent.

Command	System Command	ASCII String
Goto START Program Axis 1	1GOTO(START)	53 54 41 52 54

0x2155 LOOP Loop Program Label

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x2155	00	LOOP Command	Various	RW	0x02	0x02	No

This object allows the user to loop a pre-defined program label and has the same functionality as the LOOP command within EASI-V

0x2155.01 LOOP Number of Loops

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x2155	01	Number of Loops	Unsigned16	RW	0x0000	0xFFFF	No

This sub index defines the number of loops to be executed. The label must already be defined on the target for the execution to be successful. The process of writing to sub-index 1 will initiate the loop using the label name in sub-index 2

0x2155.02 LOOP Label Name to be Looped

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x2155	02	Label Name to be Looped	Visible String	RW	0x0000000000	0xFFFF	No

This object defines the label to be looped. The data format contains a maximum of 5 ASCII characters. The label must already be defined on the target for the execution to be successful. The ASCII characters sent must be in upper case and do not require an address.

0x21A0 POSMAIN (Stepper Only)

Index	Name	Sub Indices	Elements	Attribute	PDO Mapping
0x21A0	POSMAIN Command	0x03	0x04	RW	No

This object allows access to the POSMAIN functionality.

0x21A0.01 POSMAIN ARM Status (Stepper Only)

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x21A0	01	POSMAIN ARM Status	Unsigned16	RW	0x00	0x31	No

This object defines the status of the ARM in bit 0 and the output is defined in the upper nibble of the lower byte.

0x21A0.02 POSMAIN Dead Band (Stepper Only)

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x21A0	02	POSMAIN Dead Band	Unsigned16	RW	0x0000	0x7FFF	No

This contains the dead-band information. This is the area in which the drive is considered to be in position. If the motor moves out of this location, the POSMAIN command becomes active.

0x21A0.03 POSMAIN Settle Time (Stepper Only)

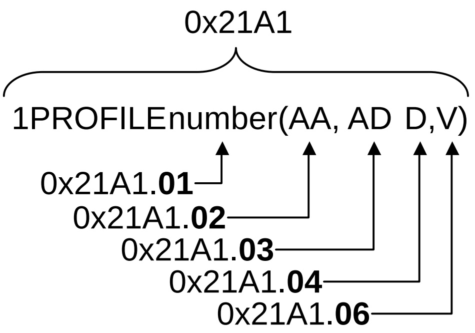
Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x21A0	03	POSMAIN Settle Time	Unsigned16	RW	0x0000	0xFFFF	No

This object specifies how long in milliseconds that the indexer will wait after motion has ceased, before checking the feedback encoder. This value is specified in milliseconds (mS).

0x21A1 PROFILE

Index	Name	Sub Indices	Elements	Attribute	PDO Mapping
0x21A0	PROFILE Command	0x06	0x05	RW	No

This object allows the user to define the PROFILE command and is broken down into the same parts as the EASI-V command. For the relation between the objects and the PROFILE command please see the diagram overleaf.



It should be noted that 21A1.05 has been omitted. This is reserved for the ‘VS’ command. Described as the Start/Stop velocity it used a property that a stepper motor can literally jump to a speed from stationary. Since this has not been implemented on the ViX sub-index 0x05 has been removed.

Example

We wish to specify the following profile over the CAN bus:

1PROFILE3(5,10,8000,5)

It is important to remember that the values have a scaling factor applied to them the details are listed in the table below.

System Variable	Description	Scaling Factor
AA	Acceleration Rate	x 100
AD	Deceleration Rate	x 100
D	Distance	x 1
V	Velocity	x 1000

The reason this is done is because internally it is not possible for the controller to handle floating point (i.e. fractional) numbers. Thus all values are scaled to make them whole. It is important to remember this when specifying a PROFILE over can.

Continuing our example the values we enter over CAN are as follows:

Index	Sub Index	Value Hex	Value Dec	Received Value
0x21A1	0x02	0x01F4	500	5
0x21A1	0x03	0x03E8	1000	10
0x21A1	0x04	0x1F 40 00 00	8000	8000
0x21A1	0x06	0x1388	5000	5

When we have specified all the values for the PROFILE we then write to sub index 0x01 the value of the profile we wish to change. Thus 0x21A1.01 = 03 this will then write all the values to the PROFILE register REMEMBER to store this permanently the SV function should be used. The operation can be confirmed using EASI-V.

0x21A1.01 PROFILE Number

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x21A1	01	PROFILE Number	Unsigned16	RW	0x01	0x08	No

This object defines the profile number. The user can select from profile 1 to 8. The process of writing to this object will initiate execution of the profile function and make use of whatever values are in the sub-indices at that time.

0x21A1.02 PROFILE Acceleration

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x21A1	02	PROFILE Acceleration	Unsigned16	RW	0x0000	0x98967F	No

This object defines the ‘AA’ parameter within the PROFILE command. It should be noted that this value is scaled buy a factor of 100. The units are revolutions per second^2 (rps^2).

0x21A1.03 PROFILE Deceleration

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x21A1	03	PROFILE Deceleration	Unsigned16	RW	0x0000	0x98967F	No

This object defines the ‘AD’ parameter within the PROFILE command. Again, like the ‘AA’ command, this value is scaled by 100 when read back. The units are revolutions per second^2 (rps^2).

0x21A1.04 PROFILE Distance

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x21A1	04	PROFILE Distance	Integer32	RW	0x80000000	0x7FFFFFFF	No

This object defines the distance to be traveled. Similar to the ‘D’ command the units are in motor increments. This value is not scaled.

0x21A1.06 PROFILE Velocity (Stepper)

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x21A1	06	PROFILE Velocity	Unsigned16	RW	0x0001	0xC350	No

This object defines velocity to be used for the profile. This value is scaled by a factor of 1000 and is in units or revolutions per second (rps).

The servo information is overleaf.

0x21A1.06 PROFILE Velocity (Servo)

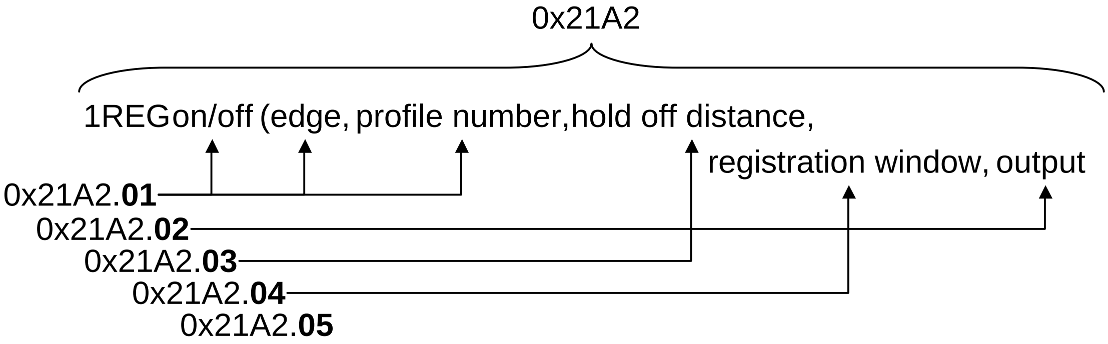
Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x21A1	06	PROFILE Velocity	Unsigned16	RW	0x0001	0x4C4B40	No

This object defines velocity to be used for the profile. This value is scaled by a factor of 1000 and is in units or revolutions per second (rps).

0x21A2 REG

Index	Name	Sub Indices	Elements	Attribute	PDO Mapping
0x21A2	REG Command	0x04	0x03	RW	No

This object allows the user to define the REG command and is broken down into the same parts as the EASI-V command. For the relation between the objects and the REG command please see the diagram below.

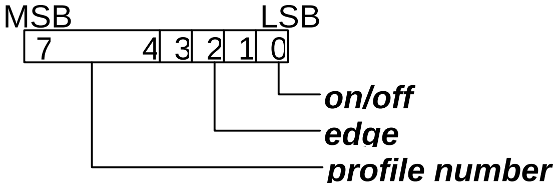

0x21A2
1REGon/off (edge, profile number,hold off distance,
registration window, output
0x21A2.01
0x21A2.02
0x21A2.03
0x21A2.04
0x21A2.05

It can be seen that within 0x21A2.01 there is the on/off, edge and profile number information. To aid understanding this has been broken down into its' component parts within the sub index information.

0x21A2.01 REG On/Off, Edge and Profile Number

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x21A2	01	REG On/Off, Edge and Profile Number	Unsigned8	RW	0x10	0x8F	No

This object defines three parameters of the REG function. The breakdown of the bit allocation has been shown in the diagram below.



Example

We wish to specify the following REG command over the CAN bus:

1REG1(1,8,2000,10,3)

From the information on 0x21A1.01 we can look at the first section of the REG command i.e. REG1 (1,8…. The construction has been broken down into a series of steps for the user.

Profile Number					Edge		On / Off
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	0	0	0	0	1	0	1

This binary number converted into hex will give the value 0x85. This will be the value used in 0x21A2.01.

0x21A2.02 REG Output

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x21A2	02	REG Output	Unsigned8	RW	0x00	0x03	No

This object allows the user to program an output to indicate that a move that has been armed and is ready for registration. The accepted values have been listed in the table below.

Output Number	Value Binary	Value Hex
0	0b0000	0x00
1	0b0110	0x01
2	0b0111	0x02
3	0b1000	0x03

0x21A2.03 REG Hold Off Distance

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x21A2	03	REG Hold Off Distance	Unsigned32	RW	0x00000000	0x7FFFFFFF	No

This object specifies the hold off distance, which is a number of steps after which the controller will begin to search for a valid registration signal.

0x21A2.04 REG Registration Window

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x21A2	04	REG Registration Window	Unsigned32	RW	0x00000000	0x7FFFFFFF	No

This object specifies the number of motor steps (after the hold off distance) that the registration mark will occur in.

0x21A3 STALL (Stepper)

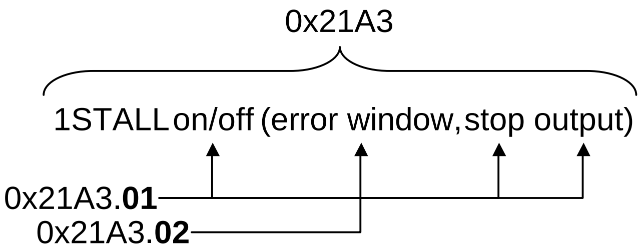
Index	Name	Sub Indices	Elements	Attribute	PDO Mapping
0x21A3	STALL Command	0x02	0x02	RW	No

This object allows the user to specify the STALL parameters

0x21A3.01 STALL On/Off, Stop and Output

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x21A3	01	STALL On/Off, Stop and Output	Unsigned16	RW	0x0000	0x0085	No

This object defines three parameters of the REG function. The breakdown of the bit allocation has been shown in the diagram below.



Example

We wish to specify the following STALL command over the CAN bus:

1STALL1(100,1,3)

From the information on 0x21A3.01 we can look at the first section of the STALL command i.e. STALL1(…,1,3) The construction has been broken down into a series of steps for the user.

Setting the 'STOP' parameter to '1' will run a fault routine (if one is defined) once the motor has stopped. However, no further action is taken if the 'STOP' parameter is set to '0'.

Output					Stop		On / Off
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	0	0	0	0	1	0	1

It should be noted that the 'Output' field takes the same format as REG command and this has been shown in the table below:

Output Number	Value Binary	Value Hex
0	0b0000	0x00
1	0b0110	0x01
2	0b0111	0x02
3	0b1000	0x03

This binary number converted into hex will give the value 0x85. This will be the value used in 0x21A3.01.

0x21A3.02 STALL Error Window

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x21A3	02	STALL Error Window	Unsigned32	RW	0x0000	0xFFFF	No

This object specifies the number of steps that the motor can lose before the shaft is considered to have stalled. This value is specified in motor steps.

0x21A4 USE

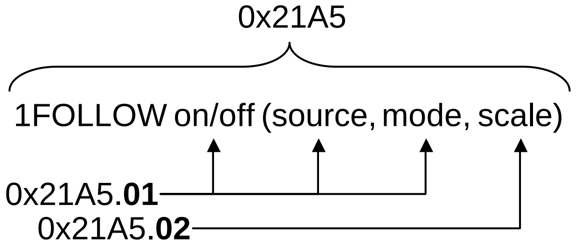
Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x21A4	00	USE Command	Unsigned8	WO	0x01	0x08	No

This object allows the user to specify a pre-defined PROFILE. From object 0x21A1 we have shown how to specify one over the CAN bus. The current PROFILE is always ‘0’ and by selecting a new profile this value is copied into the PROFILE0 location.

0x21A5 FOLLOW

Index	Name	Sub Indices	Elements	Attribute	PDO Mapping
0x21A5	FOLLOW Command	0x02	0x03	RW	No

This object allows the user to specify the FOLLOW parameters



0x21A5.01 FOLLOW On/Off, Source and Mode

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x21A5	01	FOLLOW On/Off, Source and Mode	Unsigned32	RW	0x0000	0xFFFF	No

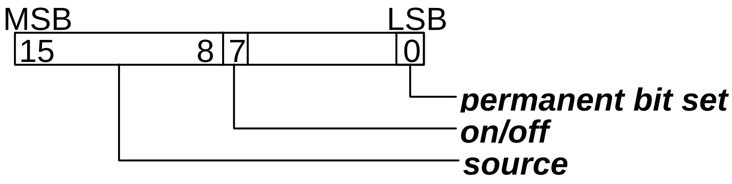
This object defines three parameters of the FOLLOW function. The breakdown of the bit allocation has been shown in the diagram below.

Example

We wish to specify the following FOLLOW command over the CAN bus:

1FOLLOW1(A,1,100)

From the information on 0x21A5.01 we can look at the first section of the FOLLOW command i.e. FOLLOW1(A,1,… The construction has been broken down into a series of steps for the user.



Bits 15 to 8 specify the signal source the ViX is to follow. This takes the form of an ASCII character. The ViX supports the sources described below.

Source	FOLLOW Variable	ASCII Value
Analogue Input X4/1 and X4/2	A	0x41
Encoder Input X4/7, X8/8, X4/12, X4/13	E	0x45

Source							On/Off							Permanent Bit Set
15	**14**	**13**	**12**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
1	0	0	0	1	0	1	1	0	0	0	0	0	0	1

This binary number converted into hex will give the value 0x4181. This will be the value used in 0x21A5.01.

0x21A5.02 FOLLOW Scale

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x21A5	02	FOLLOW Scale	Integer16	RW	0xEC78	0x1388	No

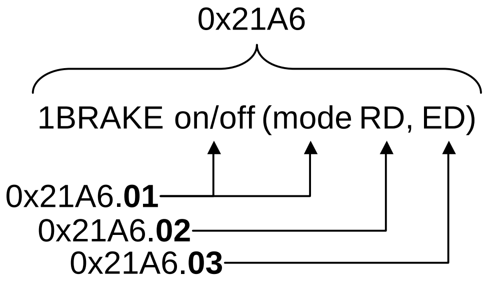
This object allows the user to specify the SCALE term used in the FOLLOW command. It should be noted that the current range is –5000% to +5000%. The default value is 100%.

The SCALE is scaled too. That is the value read back over CAN is increased by a factor of 10. Thus if the value of 100 is user as per the example the value of 0X21A5.02 is 0x03E8.

0x21A6 BRAKE (Servo)

Index	Name	Sub Indices	Elements	Attribute	PDO Mapping
0x21A6	BRAKE Command	0x03	0x04	RW	No

This object allows the user to specify the BRAKE parameters. This object defines two parameters of the BRAKE function. The breakdown of the bit allocation has been shown in the diagram overleaf.



0x21A6.01 BRAKE On/Off and Mode

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x21A6	01	BRAKE On/Off and Mode	Integer16	RW	0x0000	0x4DFF	No

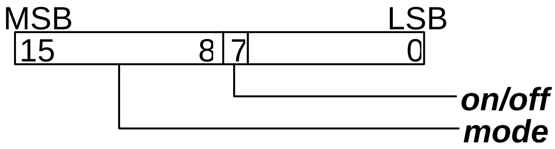
This object specifies the On/Off and Mode action. To better explain this an example has been written.

Example

We wish to specify the following BRAKE command over the CAN bus:

1BRAKE1(A,50,200)

From the information on 0x21A6.01 we can look at the first section of the BRAKE command i.e. BRAKE1(A,… The construction has been broken down into a series of steps for the user.



Bits 15 to 8 specify the brake operation mode. Again, this takes the form of an ASCII character. The ViX supports the following modes:

Mode	BRAKE Variable	ASCII Value
Automatic Holding Brake	A	0x41
Automatic Dynamic Brake	D	0x44
Manual Brake	M	0x4D

Thus for our application we have chosen ‘A’ and so our bit pattern breaks down as follows:

Mode							On/Off							
15	**14**	**13**	**12**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
1	0	0	0	0	0	1	1	0	0	0	0	0	0	0

This binary number converted into hex will give the value 0x4180. This will be the value used in 0x21A6.01.

0x21A6.02 BRAKE RD

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x21A6	02	BRAKE RD	Unsigned16	RW	0x0000	0x1388	No

This object specifies the time in milliseconds for the brake to be released **after** the drive has been energised. The default value is 50mS.

0x21A6.02 BRAKE ED

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x21A6	02	BRAKE ED	Unsigned16	RW	0x0000	0x1388	No

This object specifies the time in milliseconds for the brake to be engaged **before** the drive de-energises. The default value is 50mS.

0x21A7 FRATE

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x21A7	00	FRATE Command	Unsigned8	RW	0x00	0x01	No

This object specifies the **F**eed **RATE** override, the **FRATE** command, is used together with the analogue input to scale the peak velocity of the drive (V). With this object set to 0x01, the FRATE is enabled.

Defined Device Profile Objects (DSP-402 V1.1)

Object table

Index	Sub Indexes	Object	System Variable	Object Type	Attributes	PDO Mapping
0x603F	00	Error Code		Unsigned16	RO	No
0x6040	00	Control Word		Unsigned16	RW	RxPDO
0x6041	00	Status Word		Unsigned16	RO	TxPDO
0x6060	00	Operation Mode		Integer8	WO	No
0x6061	00	Report Current Operation Mode	M	Integer8	RO	No
0x6064	00	Position Actual	PA	Integer32	RO	TxPDO
0x6065	00	Following Error Window		Integer32	RW	No
0x6067	00	In Position Window		Unsigned32	RW	No
0x6068	00	In Position Time	IT	Unsigned16	RW	No
0x606C	00	Velocity	V	Integer32	RO	TxPDO
0x607A	00	Target Position	D	Integer32	RW	RxPDO / TxPDO
0x6081	00	Profile Velocity	V	Unsigned32	RW	RxPDO / TxPDO
0x6083	00	Profile Acceleration	AA	Unsigned32	RW	No
0x6084	00	Profile Deceleration	AD	Unsigned32	RW	No
0x6086	00	Motion Profile Type	SC	Integer16	RW	No
0x6098	00	Homing Method		Integer8	RW	No
0x6099	02	Homing Velocity	HF	Unsigned32	RW	No
0x609A	00	Homing Acceleration / Deceleration		Unsigned32	RW	No
0x60F4	00	Position Error	PE	Unsigned16	RW	No
0x6100	00	Input Word	IS	Unsigned16	RO	TxPDO
0x6300	00	Output Word	O	Unsigned16	RW	RxPDO / TxPDO
0x6504	00	Drive Manufacturer		Visible String	RW	No

0x603F Error Code

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x603F	00	Error Code	Unsigned8	RW	0x0000	0x1000	No

This object captures the last known drive error. It corresponds to the value of the lower 16 bits of object 0x1003.

Owing to a lack of space within the ViX memory, it was decided to report all errors as 0x1000, which is a 'Generic Error' as defined by the communications profile (DS-301), and leave the detail in the User Fault (UF), Drive Fault (DF) and Status (ST) registers.

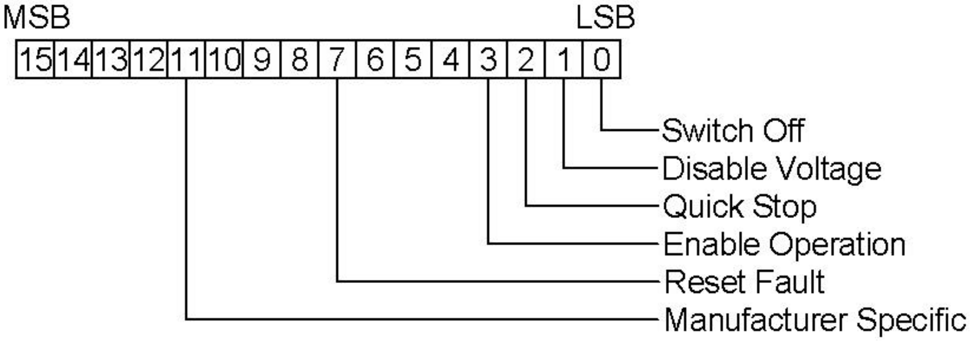
0x6040 Control Word

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x6040	00	Control Word	Unsigned16	RW	0x0000	0xFFFF	No

This object allows the user to control the ViX drive using a bit-pattern. A limited implementation has been written and it is hoped the user will take advantage of the 0x2004 object for more precise control for this only the mandatory states have been implemented.

Data Details

The control word breaks down as follows



Bit Number	Function	Description
12 - 15	Reserved	
11	Manufacturer Specific	Unknown Operation
8 - 10	Reserved	
7	Reset Fault	This command does not work
4 - 6	Reserved	
3	Enable Operation	This command does not work
2	Quick Stop	De-Energise the drive
1	Disable Voltage	This command does not work
0	Switch Off	De-energise the drive

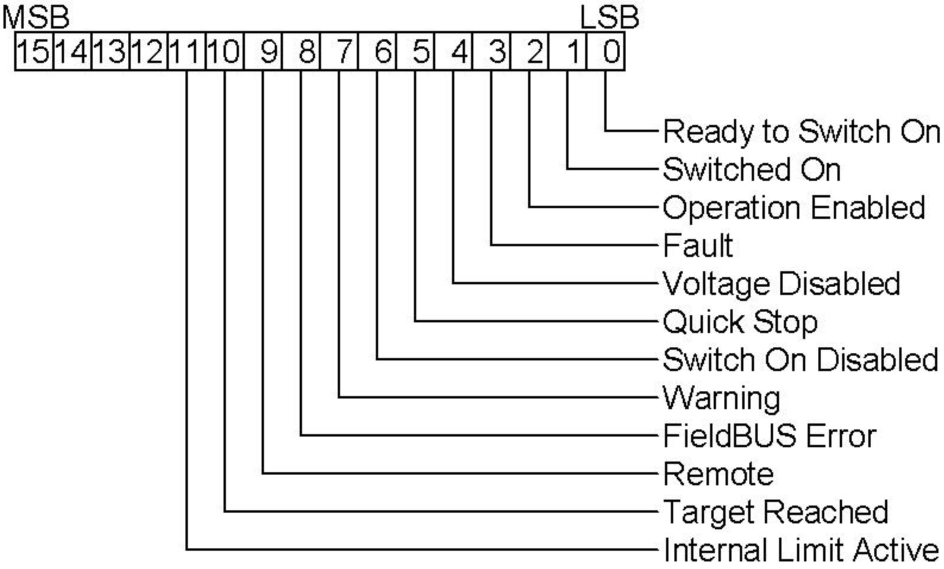
0x6041 Status Word

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x6041	00	Status Word	Unsigned16	RO	0x0000	0xFFFF	RxPDO

This object allows the user to check the current status of the ViX drive. It is recommended that this value is mapped to one of the PDOs.

Data Details

The status word breaks down as follows.



Bit	Definition	Description
15	Manufacturer Specific	
14	Manufacturer Specific	
13	Operation Mode Specific	
12	Operation Mode Specific	
11	Internal Limit Active	A drive limit has been activated check LIMITS command
10	Target Reached	The motor is in position and stationary (IP = 1)
9	Remote	
8	FieldBUS Error	
7	Warning	
6	Switch On Disabled	
5	Quick Stop	
4	Voltage Disabled	
3	Fault	
2	Operation Enabled	
1	Switched On	
0	Ready to Switch On	

It should be noted that a healthy drive should return the value 0x0627.

0x6060 Change Operation Mode

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x6060	00	Change Operation Mode	Integer8	WO	0xE0	0xFF	No

This object allows the user to change the mode of operation of the drive identical to the ‘M’ command within EASI-V.

Mode	BRAKE Variable	ASCII Value
Mode Absolute	MA	0xF0
Mode Incremental	MI	0xFE
Mode Continuous	MC	0xFF
Mode Position	MP	0xE0
Mode Bi-Directional	MB	0xF8

0x6061 Report Operation Mode

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x6061	00	Report Operation Mode	Integer8	RO	0xE0	0xFF	No

This object allows the user to see the current mode of operation of the ViX drive. See object 0x6060 for information on the returned values.

0x6064 PA Position Actual

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x6064	00	Position Actual	Unsigned32	RO	0x00000000	0x7FFFFFFF	No

This object reports the absolute position of the motor. Similar to the 1R(PA) command within EASI-V.

0x6065 Following Error Window (Stepper)

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x6065	00	Following Error Window	Unsigned32	RW	0x0000	0xFFFF	No

This object reports the following error window. This writes to the '***error window'*** used in the stepper STALL (0x21A3.02) command.

0x6065 Following Error Window (Servo)

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x6065	00	Following Error Window	Unsigned32	RW	0x0000	0x61A80	No

See previous for operation.

0x6067 In Position Window (Stepper)

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x6067	00	In Position Window	Unsigned32	RW	0x80000000	0x7FFFFFFF	No

This object reports the in position window. This object writes to the '***deadband***' of the POSMAIN command (0x21A0.02)

0x6068 IT In Position Time

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x6068	00	In Position Time	Unsigned16	RO	0x0001	0x01F4	No

This object writes to the system variable IT similar to the EASI-V command 1W(IT). The IP flag can only go high once movement has stopped and the IT timer value has timed-out.

0x606C V Velocity

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x606C	00	Velocity	Integer32	RW	0x0001	0x30D4	No

This object reads and writes to the current velocity of the drive. This value is scaled by a factor of 100 when read back over CAN.

0x607A D Target Position

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x607A	00	Velocity	Integer32	RW	0x80000000	0x7FFFFFFF	No

This object sets the target position of the drive. It should be noted that this value is the same as the EASI-V command D. There is no scaling for this value.

0x6081 V Profile Velocity (Stepper)

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x6081	00	Profile Velocity	Integer32	RW	0x0001	0xC350	No

This writes to the velocity of the current profile i.e. Profile 0. Similar to the 606C command it is left to the user to select either to change. The minimum velocity specified over CAN is 0x0001 which is a speed of 0.01rps and a maximum of 0x30D4 or 125rps.

This object uses a scaling factor of 100.

0x6081 V Profile Velocity (Servo)

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x6081	00	Profile Velocity	Integer32	RW	0x0001	0x4C4B40	No

See previous for operation.

0x6083 AA Profile Acceleration

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x6083	00	Profile Acceleration	Unsigned32	RW	0x000001	0x98967F	No

This object allows the user to change the acceleration rate of the currently selected Profile i.e Profile 0. The range of this command is from 0.01rps^2 to 99999.99rps^2.

This object uses a scaling factor of 100.

0x6084 AD Profile Deceleration

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x6084	00	Profile Deceleration	Unsigned32	RW	0x000001	0x98967F	No

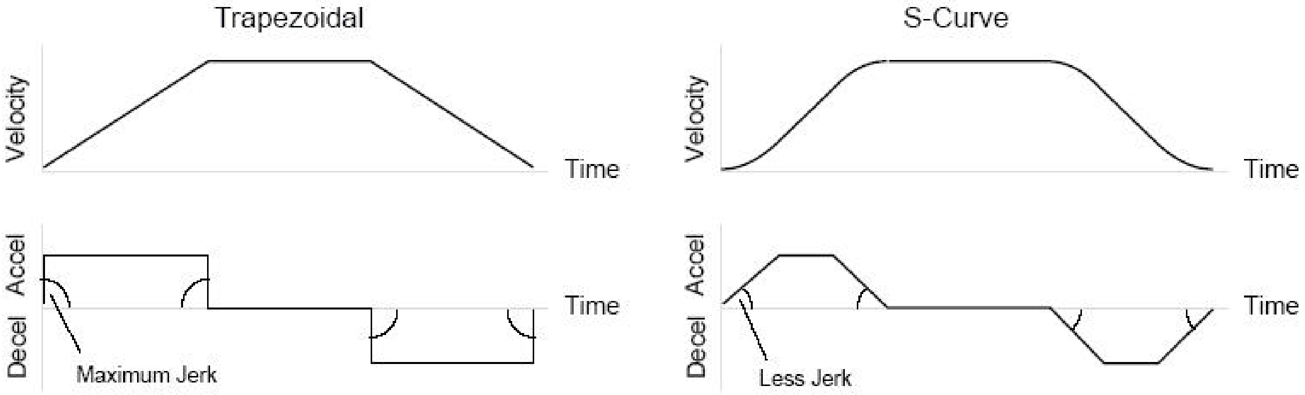
This object allows the user to change the deceleration rate of the currently selected Profile i.e Profile 0. The range of this command is from 0.01rps^2 to 99999.99rps^2.

This object uses a scaling factor of 100.

0x6086 SC Motion Profile Type

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x6086	00	Motion Profile Type	Integer16	RW	0x0000	0x0001	No

This object allows the user to select between the standard trapezoid or S curve motion profile. To reduce the rate of change of acceleration or deceleration within a move, select SC. When enabled, this variable smoothes-out rapid changes of acceleration. See the diagram below for more information.



To achieve this type of S curve correction an average acceleration value is used which is set at half the value of the maximum acceleration. In all cases, the value of AA will be used for acceleration and deceleration. Asymmetric move profiles are not possible when using S-curve correction

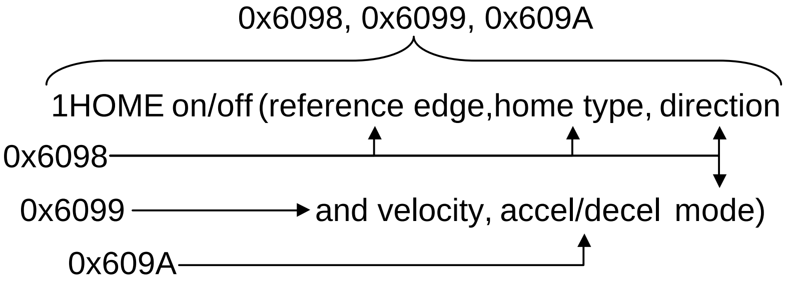
Data Details

Mode	System Variable	Object Value
Trapezoidal Move	1W(SC,0)	0x0000
S-Curve Profile	1W(SC,1)	0x0001

0x6098 Homing Method Edge, Type and Direction

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x6098	00	Homing Method	Integer8	RW	0x000001	0x98967F	No

This object defines the type of homing method to be used. Similar to the HOME command, this specifies the home edge i.e. towards the positive or negative travel limit, whether the switch is normally open or closed and finally the direction the search should start in.


0x6098, 0x6099, 0x609A
1HOME on/off (reference edge,home type, direction
0x6098
0x6099
and velocity, accel/decel mode)
0x609A

Data Details

Object Value	Home Edge	Switch Type	Direction	Homing Mode
FF	+	N/O	+	WINDOW
FE	-	N/O	+	WINDOW
FD	+	N/O	-	WINDOW
FC	-	N/O	-	WINDOW
FB	+	N/O	+	EDGE
FA	-	N/O	+	EDGE
F9	+	N/O	-	EDGE
F8	-	N/O	-	EDGE
F7	+	N/O	+	ZERO
F6	-	N/O	+	ZERO
F5	+	N/O	-	ZERO
F4	-	N/O	-	ZERO
F3	+	N/O	+	Z
F2	-	N/O	+	Z
F1	+	N/O	-	Z
F0	-	N/O	-	Z
EF	+	N/C	+	WINDOW
EE	-	N/C	+	WINDOW
ED	+	N/C	-	WINDOW
EC	-	N/C	-	WINDOW
EB	+	N/C	+	EDGE
EA	-	N/C	+	EDGE
E9	+	N/C	-	EDGE
E8	-	N/C	-	EDGE
E7	+	N/C	+	ZERO
E6	-	N/C	+	ZERO
E5	+	N/C	-	ZERO
E4	-	N/C	-	ZERO
E3	+	N/C	+	Z
E2	-	N/C	+	Z
E1	+	N/C	-	Z
EO	-	N/C	-	Z

The user should select from the above table the combination that suits the machine and set the corresponding object value.

Example

Using the example in the ViX manual then the EASI-V command we have is as follows:

HOME1(-,1,-15,100,1)	Stop on the negative edge Switch is normally closed Search direction is negative (15rps) Position to the edge of the switch	0x6098 = 0xE8

0x6099 Homing Velocity

Index	Name	Sub Indices	Elements	Attribute	PDO Mapping
0x6099	Home Velocity	0x02	0x03	RW	No

This object defines the speed at which the drive looks for the home switch. The units are revolutions per sec (rps).

0x6099.01 Homing Velocity Velocity

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x6099	01	Homing Velocity	Unsigned32	RW	0xEC78	0x1388	No

This object defines the velocity. This value should be +/-5000.

This value has a scaling factor of 1000.

0x6099.02 Home Final Velocity

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x6099	02	Home Final Velocity	Unsigned32	RW	0x00000000	0xFFFFFFFF	No

This object defines the final ‘creep’ or slow speed that the drive uses to complete the last part of the move.

This value has a scaling factor of 1000.

0x609A Homing Acceleration/Deceleration

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x609A	00	Homing Acceleration / Decleration	Unsigned32	RW	0x0001	0x1388	No

This object configures the acceleration rate used during a Homing operation. The acceleration is reported in units of revolutions per second^2.

This value has a scaling factor of 100.

0x60F4 Position Error (PE)

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x60F4	00	Position Error	Unsigned32	RW	0x00000000	0x	No

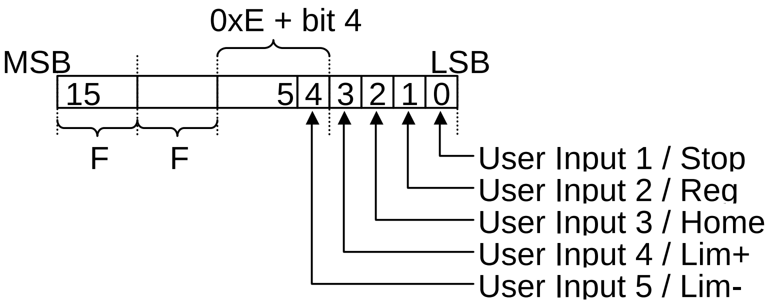
This object allows the user to configure the position error window. PE reports the position error, that is, the difference between PT and PA.

0x6100 Input Word (IS)

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x6100	00	Input Word	Unsigned16	RO	0xFFE0	0xFFFF	TxPDO

This object defines the current input status of the drive. That is the inputs that are active.

Data Details



It should be noted that the input 5 is part of the second nibble. An example has been shown below.

Input Bit Pattern	Object Value
00000	0xE0
00001	0xF0
00011	0xF8
00111	0xFC
01111	0xFE
11111	0xFF

0x6300 Output Word (O)

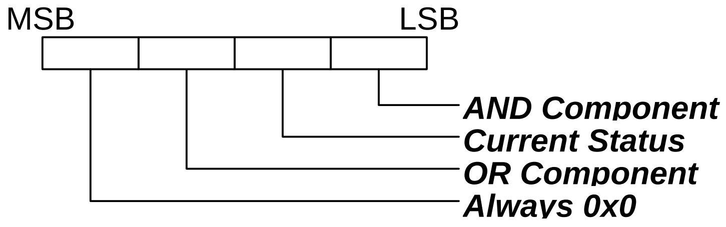
Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x6300	00	Output Word	Unsigned16	RW	0x0000	0x0000	TxPDO / RxPDO

This object allows the user to read and write to the outputs. The breakdown of this object is complex and thus has been split into two parts.

When the status of the outputs are read, if all are set off then the value returned is 0x0010. It should be remembered that this bit is always set. The bit pattern has been shown below.

Object Value	Easi-V Response
0x0010	*000
0x0030	*100
0x0050	*010
0x0090	*001

Secondly if the user wishes to set an output then a more complex procedure must be followed. The lower byte of the word is an OR component and is OR-ed with the existing output status. The upper byte of the word is the mask for the output and is AND-ed with the result of the previous OR.



Example

If the current state of the output is 0x0070 i.e. the first two outputs are on and we wish to set the last output we must set the third bit so the result of the OR is ‘1’ and the result of the AND is ‘1’. Thus we choose the value 0x0707. We can ignore the ‘Current Status’ as this is updated as soon as the new output status command is received.

Checking with the ‘Current Status’ after the object write we see the value 0x00F0.

0x6504 Drive Manufacturer

Index	Sub Index	Name	Type	Attribute	Object Values		PDO Mapping
					Minimum	Maximum	
0x6504	00	Drive Manufacturer	Visible String	RW	0x0000	0x0000	No

This object is read and write accessible. The default value reports back Parker EME. The maximum number of characters allowable is 27. When writing to this location the existing buffer is overwritten only for the number of characters entered. The location is currently non-savable.

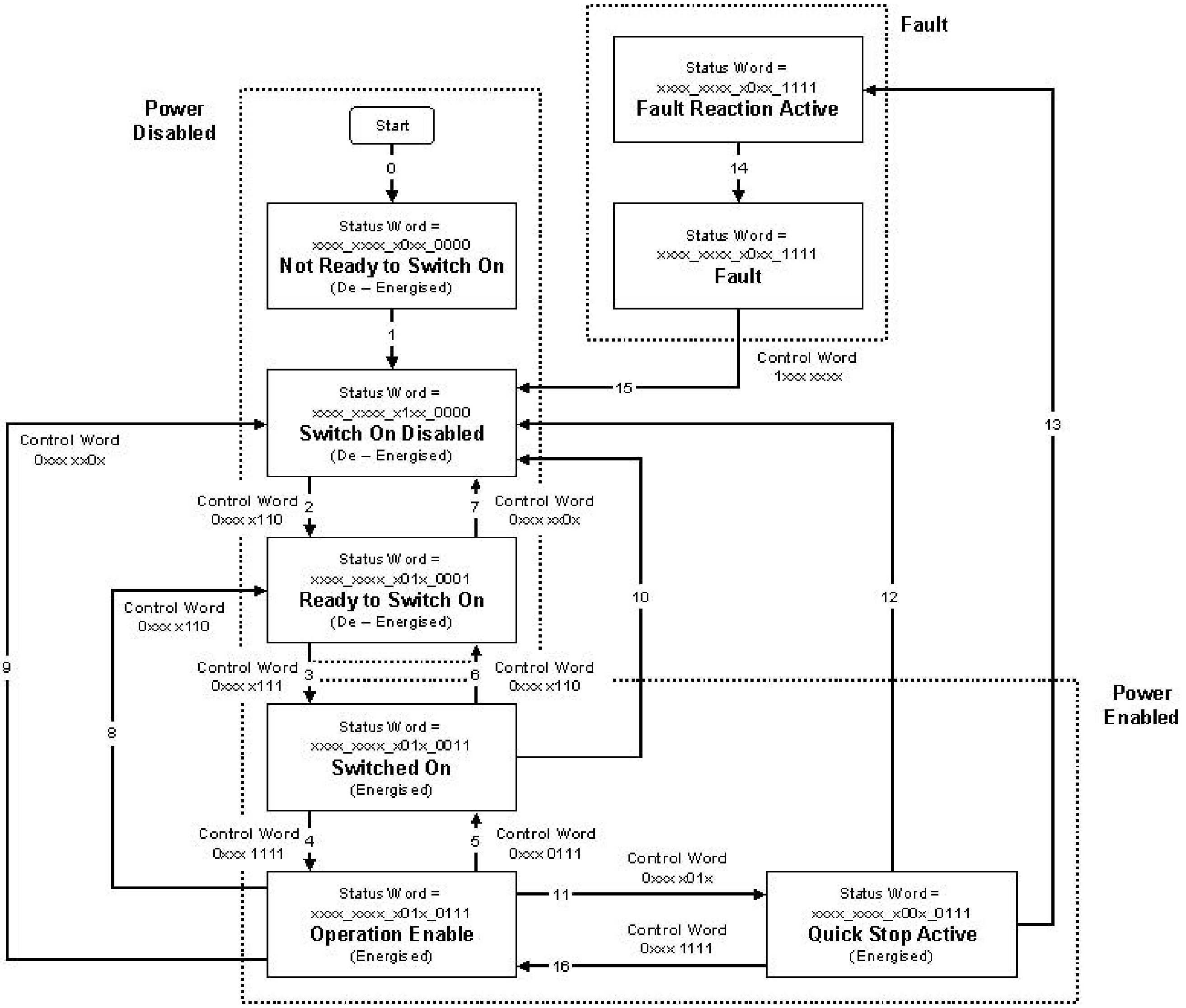
Default Value	String Translation
50 61 72 6B 65 72 20 45 4D 45 00 00 00 00 00	Parker EME

6. STATE MACHINE

Introduction

The state machine implemented within the ViX product has been based on the Draft Standard 402 (issue 1.1) for Device profiles in the drives and motion control applications. The standard defines the sequence of operations and states that enables the controller to power up in a known safe manner. The standard also defines the states for enabling motion and recovery from a fault. The various states of the device can be achieved by bit manipulation of the control object (0x6040) and the status can be read back via the status object (0x6041)

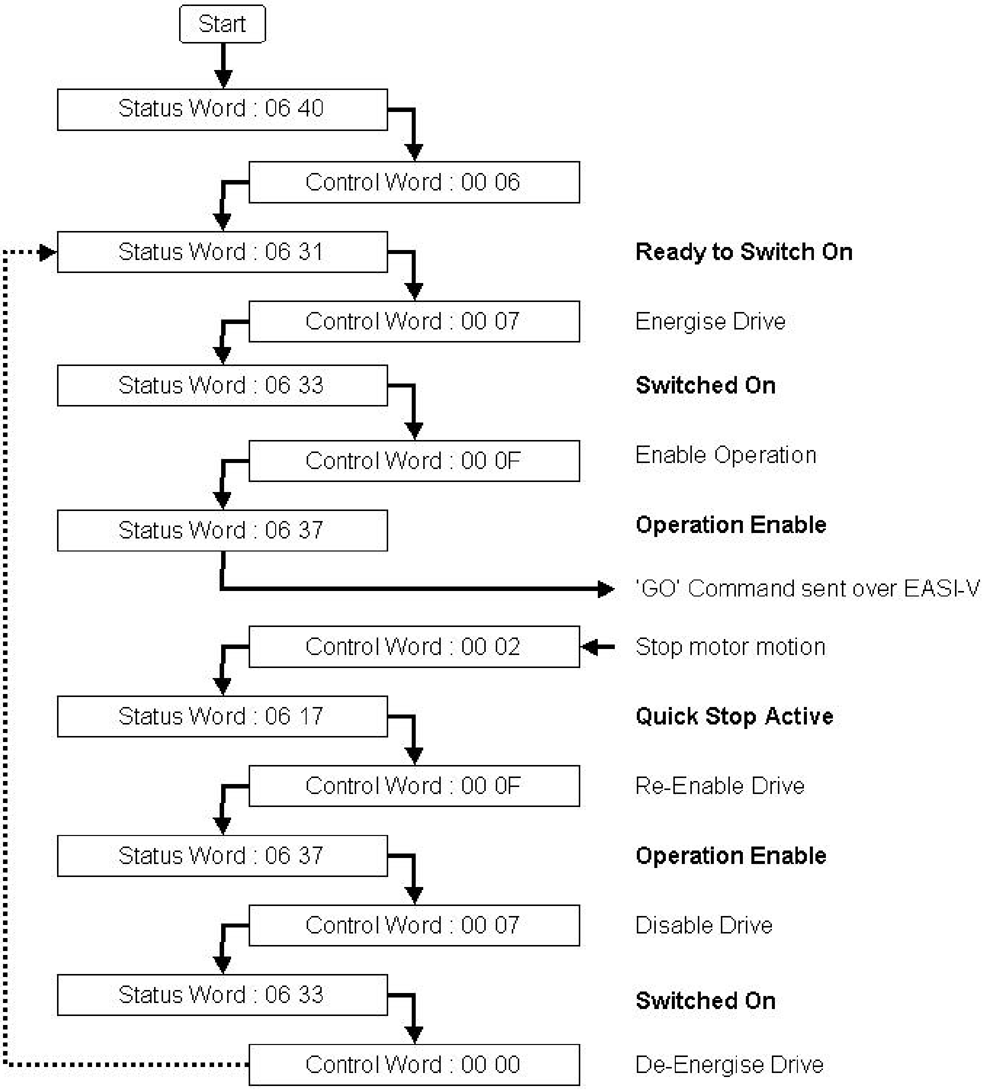
Only the mandatory parts of the state machine have been implemented. Below is a state flow diagram and overleaf an explanation.



Explanation

It should be noted that the previous diagram is only valid if the states are shifted over the CAN bus. If this is not so then it is possible for the status object (0x6041) to return an invalid number. Below is a diagram showing the operation of the state machine over can and the commands sent to shift states.

Example Operation Using CAN State Machine



Appendix 1 – ASCII Table

Introduction

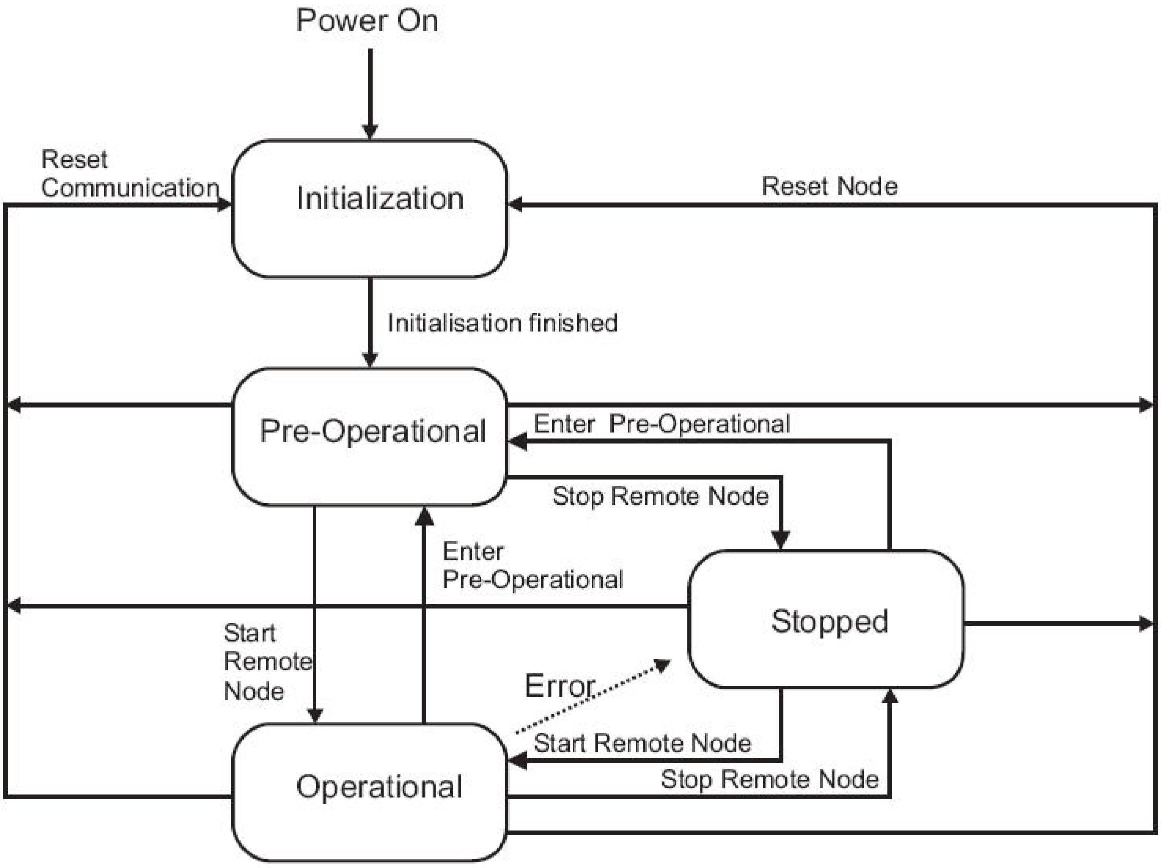
Several of the CANOpen objects are of the type VIS_string when the user looks at this object an ASCII string is returned. For ease of use, a lookup table has been included.

ASCII	HEX	Symbol	ASCII	HEX	Symbol	ASCII	HEX	Symbol	ASCII	HEX	Symbol
0	0	NUL									
1	1	SOH	41	29	)	81	51	Q	121	79	y
2	2	STX	42	2A	*	82	52	R	122	7A	z
3	3	ETX	43	2B	+	83	53	S	123	7B	{
4	4	EOT	44	2C	'	84	54	T	124	7C	l
5	5	ENQ	45	2D	-	85	55	U	125	7D	}
6	6	ACK	46	2E	.	86	56	V	126	7E	~
7	7	BEL	47	2F	/	87	57	W	127	7F	□
8	8	BS	48	30	0	88	58	X			
9	9	TAB	49	31	1	89	59	Y			
10	A	LF	50	32	2	90	5A	Z			
11	B	VT	51	33	3	91	5B	[			
12	C	FF	52	34	4	92	5C	\			
13	D	CR	53	35	5	93	5D	]			
14	E	SOH	54	36	6	94	5E	^			
15	F	SI	55	37	7	95	5F	_			
16	10	DLE	56	38	8	96	60	'			
17	11	DC1	57	39	9	97	61	a			
18	12	DC2	58	3A	:	98	62	b			
19	13	DC3	59	3B	;	99	63	c			
20	14	DC4	60	3C	<	100	64	d			
21	15	NAK	61	3D	=	101	65	e			
22	16	SYN	62	3E	>	102	66	f			
23	17	ETB	63	3F	?	103	67	g			
24	18	CAN	64	40	@	104	68	h			
25	19	EM	65	41	A	105	69	I			
26	1A	SUB	66	42	B	106	6A	j			
27	1B	ESC	67	43	C	107	6B	k			
28	1C	FS	68	44	D	108	6C	l			
29	1D	GS	69	45	E	109	6D	m			
30	1E	RS	70	46	F	110	6E	n			
31	1F	US	71	47	G	111	6F	o			
32	20	(space)	72	48	H	112	70	p			
33	21	!	73	49	I	113	71	q			
34	22	"	74	4A	J	114	72	r			
35	23	#	75	4B	K	115	73	s			
36	24	$	76	4C	L	116	74	t			
37	25	%	77	4D	M	117	75	u			
38	26	&	78	4E	N	118	76	v			
39	27	'	79	4F	O	119	77	w			
40	28	(	80	50	P	120	78	x			

Appendix 2 – CiA DS-301 CAN State Diagram

Introduction

All CANOpen devices utilise the same standard state NMT state machine. It can be seen that from power on the node will immediately pass through to the Pre-Operational state. It is in this state that device configuration using SDOs is possible. When complete the node is then moved into the Operational state by the issue of the 'Start Remote Node' command.



Appendix 3 – TxPDO and RxPDO Transmission Types

Introduction

For both transmit and receive PDOs it is possible to define the transmission type. As default this is pre-chosen as COV or Change Of Value, however it is possible to send a RTR or Remote Transmission Request. For the users information a table has been prepared to show all the options supported.

Transmission Type	**PDO Transmission**				
Dec	**Hex**	**Async**	**RTR Only**	**TxPDO**	**RxPDO**
252	FC		X	Data is read-in with a SYNC, but not sent, request via RTR	Not Supported
253	FD	X	X	Request via RTR	COV
254	FE	X		COV*	COV
255	FF	X		COV*	COV

Objects 0x1400.02 and 0x1401.02 support transmission types 0xFE to 0xFF and is set by default to 0xFF.

Objects 0x1800.02 and 0x1801.02 supports transmission types 0xFC to 0xFF and is set by default to 0xFE.

* - For TxPDO1 or 2, if the transmission type is set to 0xFE or 0xFF, then the data will be transmitted at an interval set by the inhibit time

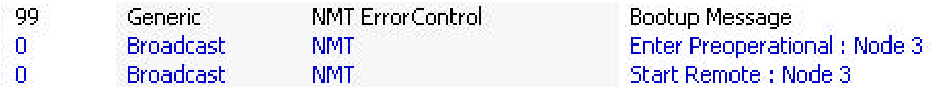
Appendix 4 – Further Information on External I/O

Introduction

It was thought to be useful to include some of the more detailed information about the implementation of the external I/O functionality with the external I/O.

All of the testing and verification was done using the IXAAT USB-to-CAN compact combined with the C3 powerPLmC CANOpen Protocol Analyser.

When the user selects a ‘**1Z**’ the ViX drive re-sets and issues an ‘**Enter Pre-Operational’** and then immediately an ‘**Enter** Operational’ command thus ensuring the remote I/O re-sends the last state of the inputs. Using the CANOpen monitor the commands can be seen as they are sent by the drive. In the example below the ViX drive has node-ID of 99 and the PIO, 3.



99	Generic	NMT ErrorControl	Bootup Message
0	Broadcast	NMT	Enter Preoperational : Node 3
0	Broadcast	NMT	Start Remote : Node 3

The Parker PIO modules used were as follows:

PIO-347	CANOpen Fieldbus Coupler ECO version
PIO-602	PIO Supply Module 24VDC
PIO-402	PIO 4-Channel Digital Input Module
PIO-504	PIO 4-Channel Digital Output Module
PIO-600	PIO Bus End Module

The outputs from the drive are re-set when the user issues a ‘**1Z**’ in a similar way to the ‘**O**’ command.

Analogue inputs and outputs are not supported and there are no plans to implement this.

One TxPDO and one RxPDO can handle all of the inputs and outputs.

The maximum baud rate of 1Mb was used in all tests and has been shown to work reliably.

If the **1IS1** command returns all zeroes but the bus is in the **Operational** state and there are inputs on, then the size of the FMON command is too large. Reduce this to the next size down.

Index

Customer Feedback

If you have spotted any errors, omissions or inconsistent information within this user guide please let us know. Either use this page (or a photocopy) to describe the error and Fax. it to the number given below. Alternatively, you may phone or email the correction.

Name of user guide:

Part number: 1600. _ _ _ . _ _ Found on the title page in the bottom left corner.

Your name:

Contact number or email address:

Description of the error: (Please include page number)

Errors can be reported by Fax:	By phone, via a technical support engineer:	Or by email:
+44 (0)1202 606301	**+44 (0)1202 606300**	**support.digiplan@parker.com**