Shannon Way
Tewkesbury
Gloucester GL20 8ND
Ph: 011.44.1684.292333
Fx: 011.44.1684.297929

1000 Gamma Drive
Suite 206
Pittsburgh, PA 15238
Ph: 412.968.9744
Fx: 412.968.9746
www.triomotion.com

A P P L I C A T I O N   N O T E

Doc No.:	**TN30-18**
Date:	**February 2, 2004**
Version:	**1.0**
Subject:	**CANopen communication to Elmo Harmonica Digital Servo Drive**

OVERVIEW:

A simple implementation of CANopen communication to an Elmo Harmonica digital servo drive is provided by the "ELMO_CAN.BAS" *Trio*BASIC program. A number of the Harmonica software commands are implemented that allow setting of drive motion parameters, reading drive inputs, and initiating motion. This document describes setting up and using the example program, and provides some details of the CiA DS 301 CANopen protocol used. It is provided for example purposes and no guarantee is made as to its suitability for any particular application.

Additional information can be found in the "Elmo Motion Control CANopen DS 301 Implementation Guide", "Harmonica Command Reference Manual", and "Harmonica Installation Guide", which can be downloaded from www.elmomc.com .

SETUP:

1. The Harmonica drive must be configured as normal using the Elmo Composer software; this would include any servo loop tuning. The drive should be set to operate in "Single Loop Position Control" mode (UM=5). For CAN communication the CAN-ID should be set to 1 (PP[13]=1), and the CAN baud rate should be set to 500KHz (PP[14]=1).
2. Make the CAN bus connection between the Trio Motion Coordinator and the Harmonica drive. Reference the "Harmonica Installation Guide" page 3-15 and the "Trio Motion Technology Technical Reference Manual" for wiring details.
3. Load the "ELMO_CAN.BAS" program file into the Trio Motion Coordinator.

OPERATION:

Once the above setup is complete the "ELMO_CAN.BAS" program can be executed.

The MotionPerfect VR Viewer Window can be used to set and view the VR() variables used by the program. The VR() variables used are setup by the "vrbase" program variable. The program uses VR(vrbase) through VR(vrbase + 18).

The MotionPerfect Terminal Window Channel 5 can also be opened to see program debug messages.

The Harmonica software commands implemented are shown in the table below:

Harmonica Command	Description	VR() Variable	Setting VR() Variable Operations
MO	Motor enable/disable	VR(mo_cmd)	=1 enables motor, =0 disables motor
BG	Begin motion	VR(bg_cmd)	=1 begins motion
ST	Stop motion	VR(st_cmd)	=1 stops motion
IP	Read Input Port bits	VR(ip_cmd)	=1 reads input port bits
JV	Set Jog Velocity	VR(jv_cmd)	=*value* sets jog velocity to *value*
PR	Set Position Relative	VR(pr_cmd)	=*value* sets relative move distance to *value*
SP	Set Speed	VR(sp_cmd)	=*value* sets move speed to *value*
AC	Set Acceleration	VR(ac_cmd)	=*value* sets move acceleration to *value*
DC	Set Deceleration	VR(dc_cmd)	=*value* sets move deceleration to *value*

The example program can be expanded to implement additional Harmonica commands and communicate to multiple drives on a CAN network.

CANopen Protocol Details:

In this example the PDO2 Transmit buffer (TPDO2) and PDO2 Receive buffer (RPDO2) are used to manipulate the Harmonica drive. This allows the use of any of the Harmonica software commands via the CAN bus. The Harmonica software commands are two letter characters. The TPDO2 message format to the Harmonica drive simply uses the ASCII code numbers for these two characters, along with some data bytes. The RPDO2 reply message from the Harmonica drive follows a similar format.

Much of the following information is taken from the "Elmo Motion Control CANopen DS 301 Implementation Guide".

In the Harmonica drive, TPDO2 is mapped by default to the transmit binary interpreter object (0x2012) and RPDO2 is mapped by default to the receive binary interpreter object (0x2013). TPDO2 is transmitted as an unsynchronized "Binary Interpreter complete" event. These can be used for setting and retrieving all numerical data from the Harmonica drive.

The binary interpreter supports three types of commands:

1. Set Value
   These commands are 8 bytes in length. The transmitted message includes either the reflections of the Set command or an error code, if a failure has occurred.
2. Get Value
   These commands can be 4 or 8 bytes in length. An 8 byte response includes the reflection of the command and the resulting numerical value, and an error if a fault has occurred.
3. Execute Command
   This command can be 4 or 8 bytes in length. An 8 byte response includes the reflection of the command and the resulting numerical value, and an error if a fault has occurred.

If an interpreter command cannot be serviced for any reason, bit 6 of byte 3 of TPDO2 is set on, and byte 4 of the response contains the Elmo error code (refer to the EC command section of the "Harmonica Command Reference Manual").

The Trio controller (client) sends commands (RPDO2) for setting variables in 8 bytes (DLC=8). The Harmonica (server) transmits the reply (TPDO2) as an asynchronous event of the received object.

RPDO2 Structure
RPDO2 is used to set values for the drive and query values from it. The structure of the command is as follows:

- Bytes 0 to 3 are the header, which include the command, command index (when needed) and data type (float or integer). Bytes 0 and 1, which represent the command character in ASCII, must be upper case.
- Bytes 4 to 7 are the data, which is always 4 bytes. The format can be integer of float. The bytes are interpreted in little endian format.

Byte	0	1	2	3			4-7
Bits	0…7	0…7	0…7	0…5	6	7	
Description	First command character	Second command character	Index for array parameter. 0 for scalar command		See note	0: integer 1: float	Data in little endian format

Note - Byte 3, bit 6:
When this bit is set to 1, the drive treats the command as a "query" and not a "setting". In this case, the rest of the data bytes are discarded and the drive replies to the command according to 4 bytes DLC. For compatibility reasons, bytes 4 to 7 should be 0.

In array commands in which the index is used (as in IB[16]), the lowest significant bits are in byte 2 and the most significant bits are in byte 3.

Example 1: Setting speed to 1000 (03E8 hex), Harmonica software command SP=1000.

Byte	0	1	2	3	4	5	6	7
Hex value	53	50	00	00	E8	03	00	00

Example 2: Query speed, Harmonica software command SP

Byte	0	1	2	3	4	5	6	7
Hex value	53	50	00	40	00	00	00	00

TPDO2 Structure
The Harmonica (server) replies on TPDO2 to query and set requests in 8 bytes (DLC=8). The structure of the reply is as follows:

- Bytes 0 to 3 are the header, which includes the responding command, command index (when needed) and data type (float or integer). It also indicates whether the response data is true data or an error code.
- Bytes 4 to 7 are the data, which is either a reflection of the host set command or an error code according to the EC command.

Byte	0	1	2	3			4-7
Bits	0…7	0…7	0…7	0…5	6	7	
Description	First character	Second character	Index for array parameter. 0 for scalar command		See note	0: integer 1: float	Data in little endian format

Note - Byte 3, bit 6:
When this bit is set 1 for TPDO2, the data in bytes 4 to 7 should be interpreted as an error code. Refer to the EC command section in the "Harmonica Command Reference Manual" for details.

In array commands in which the index is used (as in IB[16]), the lowest significant bits are in byte 2 and the most significant bits are in byte 3.

Example: If SP=1000 (03E8 hex), the Harmonica reply to the query speed command would be:

Byte	0	1	2	3	4	5	6	7
Hex value	53	50	00	00	E8	03	00	00

Execute Command
These commands are used to instruct the Harmonica to perform a sequence. The reply to these commands is only an acknowledgement or an error code, and is always 8 bytes long. There is no data value for executing the command. Execute commands are a unique case of RPDO2, which can be used with a DLC or either 4 or 8 bytes.

Example: BG command to start motion.

Byte	0	1	2	3	4	5	6	7
Hex value	42	47	00	00	40	00	00	00

Success:

Byte	0	1	2	3	4	5	6	7
Hex value	42	47	00	00	00	00	00	00

Failure: error code 58 (3A hex) for "motor must be on"

Byte	0	1	2	3	4	5	6	7
Hex value	42	47	00	40	3A	00	00	00