IB-11B023

DELTAMAX MOTION CONTROL SYSTEM JANUARY 1997

IIS
CAN-BUS
PROTOCOL

APPLICATION NOTE

INDUSTRIAL INDEXING SYSTEMS, Inc.

Revision - 0
Approved By:

TABLE OF CONTENTS

PURPOSE

This document shall describe the DeltaMax Multidrop Can-Bus Software and Hardware protocol.

Effectively:

Hardware: To include CAN-chip hardware.

Software: Firmware to drive the CAN-Bus with a protocol similar to Industrial Indexings Packet Protocol utilized over the DeltaMax RS232 port.

New Part Numbers:

DeltaMax Firmware # is SF-3902.

INTRODUCTION

DeltaMax communicates data over the CAN hardware interface as a slave device. The master device typically a PC can communicate with up to 63 DeltaMax controllers, data bandwidth limited, at 125K, 250k, and 500K baud. The DeltaMax controller shall acknowledge all requests by a master as described here in.

COMMUNICATION FEATURES

1) Multidrop communication, one bus master and up to 63 slave devices.

2) 63 slave DeltaMax devices on line using CAN hardware protocol.

3) Read/Write DeltaMax Integer Variable Memory space.

4) Read/Write DeltaMax Flag bit map.

5) Request DeltaMax Status.

CAN-Bus

The CAN interface protocol is a serial asynchronous transmission. A complete transmission consists of the following.

Start of Frame
Arbitration Field (holds message targets identifier)
Control Field
Data Field (from 0 to 8 bytes)
CRC Field
Acknowledge Field
End-Of_Frame

The CAN controller chips within the DeltaMax and PC hardware handle much of the individual field information in CAN controller hardware.

CAN Controller (82C200) Registers

The programmer has access to the following registers in the CAN controller chip to handle master/slave device addressing/arbitration and data exchanges. The following register descriptions pertain to the Philips 82C200 CAN-Controller, all registers are 8 bits:

Transmit registers (Holds CAN message):

DSCR1; Description byte 1, holds identifier bit 10 thru 3
DSCR2; Description byte 2, identifier bits 2 thru 0, RTR bit, and data length
BYTE_0; 1st transmission data byte
BYTE_1; 2nd transmission data byte
BYTE_2; 3rd transmission data byte
BYTE_3; 4th transmission data byte
BYTE_4; 5th transmission data byte
BYTE_5; 6th transmission data byte
BYTE_6; 7th transmission data byte
BYTE_7; 8th transmission data byte

Receive registers:

Same as the transmit registers, in a different address space, however there are two sets. While one set of the receive buffer is being read by a external device the other may be in the process of receiving the next incoming message.

Arbitration setup registers:

ACR; Acceptance Code Register, 8-bit pattern to define a unique CAN-Bus address. Which associate directly to identifier bits 10 thru 3.

AMR; Acceptance Mask Register, 8 bits to define which identifier bits 10 thru 3 in the incoming message are to be used for acceptance arbitration.

The 125k-baud rate is setup in the CAN controller registers as follows in DeltaMax:

BTR0 = 03 hex
BTR1 = 1C hex
OCR = FA hex

CAN-Bus LEDs

There are two LEDs on the DeltaMax to give a visual definition of the CAN-Bus present status, defined below.

LEDs Off:
CAN-Controller is disabled. All CAN-Bus activity is ignored.

Green Solid:
CAN-Controller is enabled, bus status is normal. The Macroprogram must issue a **can_set** to enable the CAN-Controller. Resetting the DeltaMax shall disable the CAN-Controller, putting it in a reset condition.

Green Flashing:
CAN-Controller is involved in CAN-Bus traffic.

Red Flashing:
Can-Controller was enabled, however external CAN-Bus power is missing.

Red Solid:
Executed a **can_set** with a bad argument or Can-Controller is unable to communicate successfully over CAN_Bus. Could reflect the absence of correct bus termination.

PACKET PROTOCOL

DSCR1

- Register shall contain the target address from 0 to 63, by writing identifier bits 10 thru 3 within DSCR1.

DSCR2

- Identifier bits 2 and 1 not used.
- Identifier bit 0 if set indicates a fragmented message and more data to follow.
- RTR bit always 0 during normal operation.
- The remaining 4 bits contain the data length in bytes which have been received or to be transmitted. The smallest number of data bytes in a packet shall be 2.

BYTE_0

- Transmitting controllers identifier bits 10 thru 3. Such that once a CAN message is received then the receiving device can decipher where it originated. Therefore a reply CAN message may be directed correctly by setting the DSCR1 with the same identifier as in BYTE_0.

BYTE_1

- Bits 3 thru 0 shall contain the Packet-type number.

CAN_RQSTAT	1
CAN_DREAD	2
CAN_DWRTE	3
READ_FLAG	4
WRITE_FLAG	5
CAN_ERROR	15

- Bits 7 thru 4 shall contain the data format number for data reads and data writes.

BOOLEAN	0
INT_CONSTANT	1
FLOAT_CONSTANT	2
INT_VARIABLE	3
FLOAT_VARIABLE	4

PACKET DEFINITIONS

The following table has examples of the ALCOA packet-types and associated data bytes making up the remaining CAN message data bytes for transmissions between a master and a DeltaMax slave.

PROTOCOL EXAMPLES

Packet Type #	Packet Desc.	Master's Request	Slave's Response
1	Request Status	DCR1 = Slave_identifier DCR2 = 02 BYTE_0 = Master_identifier BYTE_1 = 01hex	DCR1 = Master_identifier DCR2 = 06 BYTE_0 = Slave_identifier BYTE_1 = 01hex BYTE_2 = 4th Data MSB BYTE_3 = 3rd Data byte BYTE_4 = 2nd Data byte BYTE_5 = 1st Data LSB
2	Data Read DeltaMax integer variables	DCR1 = Slave_identifier DCR2 = 04 BYTE_0 = Master_identifier BYTE_1 = 32hex BYTE_2 = Adrress offset HB BYTE_3 = Address offset LB	DCR1 = Master_identifier DCR2 = 06 BYTE_0 = Slave_identifier BYTE_1 = 32hex BYTE_2 = 4th Data MSB BYTE_3 = 3rd Data byte BYTE_4 = 2nd Data byte BYTE_5 = 1st Data LSB
2	Data Read DeltaMax float variables	DCR1 = Slave_identifier DCR2 = 04 BYTE_0 = Master_identifier BYTE_1 = 42hex BYTE_2 = Adrress offset HB BYTE_3 = Address offset LB NOTE: On slaves second transmit the fragment bit is cleared.	First transmit: DCR1 = Master_identifier DCR2 = 26 BYTE_0 = Slave_identifier BYTE_1 = 42hex BYTE_2 = 4th Data MSB BYTE_3 = 3rd Data byte BYTE_4 = 2nd Data byte BYTE_5 = 1st Data LSB Second transmit: DCR1 = Master_identifier DCR2 = 06 BYTE_0 = Slave_identifier BYTE_1 = 42hex BYTE_2 = 4th Data MSB BYTE_3 = 3rd Data byte BYTE_4 = 2nd Data byte BYTE_5 = 1st Data LSB

Packet Type #	Packet Desc.	Master's Request	Slave's Response
3	Data Write DeltaMax integer variables	DCR1 = Slave_identifier DCR2 = 08 BYTE_0 = Master_identifier BYTE_1 = 33hex BYTE_2 = Adrress offset HB BYTE_3 = Address offset LB BYTE_4 = 4th Data MSB BYTE_5 = 3rd Data byte BYTE_6 = 2nd Data byte BYTE_7 = 1st Data LSB	DCR1 = Master_identifier DCR2 = 02 BYTE_0 = Slave_identifier BYTE_1 = 33hex
3	Data Write DeltaMax float variables	First transmit: DCR1 = Slave_identifier DCR2 = 08 BYTE_0 = Master_identifier BYTE_1 = 43hex BYTE_2 = Adrress offset HB BYTE_3 = Address offset LB BYTE_4 = 4th Data MSB BYTE_5 = 3rd Data byte BYTE_6 = 2nd Data byte BYTE_7 = 1st Data LSB Second transmit: DCR1 = Slave_identifier DCR2 = 08 BYTE_0 = Master_identifier BYTE_1 = 43hex BYTE_2 = Adrress offset HB BYTE_3 = Address offset LB BYTE_4 = 4th Data MSB BYTE_5 = 3rd Data byte BYTE_6 = 2nd Data byte BYTE_7 = 1st Data LSB	First acknowledget: DCR1 = Master_identifier DCR2 = 02 BYTE_0 = Slave_identifier BYTE_1 = 43hex Second acknowledget: DCR1 = Master_identifier DCR2 = 02 BYTE_0 = Slave_identifier BYTE_1 = 43hex NOTE: Master must increment float data address in second float transmit by 32 bytes.
3	Data Write DeltaMax integer variables	DCR1 = Slave_identifier DCR2 = 08 BYTE_0 = Master_identifier BYTE_1 = 33hex BYTE_2 = Adrress offset HB BYTE_3 = Address offset LB BYTE_4 = 4th Data MSB BYTE_5 = 3rd Data byte BYTE_6 = 2nd Data byte BYTE_7 = 1st Data LSB	DCR1 = Master_identifier DCR2 = 02 BYTE_0 = Slave_identifier BYTE_1 = 33hex

Packet Type #	Packet Desc.	Master's Request	Slave's Response
3	Data Write DeltaMax float variables	First transmit: DCR1 = Slave_identifier DCR2 = 08 BYTE_0 = Master_identifier BYTE_1 = 43hex BYTE_2 = Adrress offset HB BYTE_3 = Address offset LB BYTE_4 = 4th Data MSB BYTE_5 = 3rd Data byte BYTE_6 = 2nd Data byte BYTE_7 = 1st Data LSB Second transmit: DCR1 = Slave_identifier DCR2 = 08 BYTE_0 = Master_identifier BYTE_1 = 43hex BYTE_2 = Adrress offset HB BYTE_3 = Address offset LB BYTE_4 = 4th Data MSB BYTE_5 = 3rd Data byte BYTE_6 = 2nd Data byte BYTE_7 = 1st Data LSB	First acknowledget: DCR1 = Master_identifier DCR2 = 02 BYTE_0 = Slave_identifier BYTE_1 = 43hex Second acknowledget: DCR1 = Master_identifier DCR2 = 02 BYTE_0 = Slave_identifier BYTE_1 = 43hex NOTE: Master must increment float data address in second float transmit by 32 bytes.
4	Read Flag When flag 16 is set.	DCR1 = Slave_identifier DCR2 = 03 BYTE_0 = Master_identifier BYTE_1 = 04hex BYTE_2 = 00hex BYTE_3 = 10hex , (Flag num. LSB)	DCR1 = Master_identifier DCR2 = 03 BYTE_0 = Slave_identifier BYTE_1 = 04hex BYTE_2 = 01hex
4	Write Flag When flag 32 is to be set.	DCR1 = Slave_identifier DCR2 = 05 BYTE_0 = Master_identifier BYTE_1 = 05hex BYTE_2 = 00hex BYTE_3 = 20hex BYTE_4 = 01hex	DCR1 = Master_identifier DCR2 = 02 BYTE_0 = Slave_identifier BYTE_1 = 05hex

SLAVE ERROR REPLY PACKETS

The error code (in HEX) is in BYTE_1 in a DeltaMax reply to a master.

1F CAN Controller error, CRC data-overrun.
2F Bad paket number.
3F Address offset out of range.
4F Wrong byte count received.
5F Master requested write to a constant.

Example:

Master's Request	Slave's Response
DCR1 = Slave_identifier DCR2 = 02 BYTE_0 = Master_identifier BYTE_1 = 00	DCR1 = Master_identifier DCR2 = 02 BYTE_0 = Slave_identifier BYTE_1 = 2F, error