

# Motion Coordinator Application Note

Number:MC-1045, Revision 1, 2/28/2007

Subject: Peer to Peer using CAN

---

The simple CAN network is implemented in 2 MC BASIC programs;

CANSLAVE.BAS is the slave node message processor.

CANMSTR.BAS contains a set of subroutines that can be included in the user's program for sending and fetching data from remote Motion Coordinators.

## **CAN Network Services:-**

When these programs are implemented on more than one Motion Coordinator connected together on a CAN network, it enables the controllers to access the TABLE memory of a remote Motion Coordinator, either as an UNSIGNED LONG (32 bit) or as a FLOAT within the range  $\pm 2^{32} \cdot 2^{16}$ .

The Master and Slave Nodes support these messages:

1. Write LONG to remote TABLE – (Sent as 1 CAN Telegram)
2. Write FLOAT to remote TABLE – (Sent as 2 CAN Telegrams)
3. Read LONG from remote TABLE – (1 Telegram request + 1 Telegram reply)
4. Read FLOAT from remote TABLE – (1 Telegram request + 2 Telegram reply)

Additionally, for every telegram that is sent by the master, a confirmation telegram is transmitted by the slave, using the CANOpen Upload/Download Domain protocol.

# Motion Coordinator Application Note

Number:MC-1045, Revision 1, 2/28/2007

Subject: Peer to Peer using CAN

---

## **CANMSTR.BAS:-**

The subroutines in CANMSTR can be inserted into the user's own program. The only restriction is that the CANSLAVE program must be running before any CANMSTR routines can be called, because the CAN chip buffers are set up by the slave program.

Certain variables must be set up before calling a subroutine as in the following description of the subroutines.

### **vars:**

```
'=====
' Pre-set variables and named constants
'=====
```

This must be called once during program initialization. It contains pre-sets for the CAN channel and variables names used in the main subroutines.

The following may need changing:

**chan=-1** Set to the Slot Number if a CAN Daughter Board is being used.

**vrbase=230** The program uses 10 VR() variables, set this to avoid a clash with other programs.

**cmd\_timeout=500** Time in milliseconds that a routine will wait for a reply from a slave before signaling a failure. Set this to a lower number for quick error detection. Increase it if spurious errors occur.

**VR(vrbase+9)** This must be set either manually at the MP2 terminal or from a MC BASIC program. The number entered is the module\_id of the Motion Coordinator on the CAN network and must be a number between 1 and 127.

### **read\_remote:**

```
'=====
'Read float data from Slave
' call with:
' index = slave target TABLE location
' slave_id = target slave Node ID
'Returns:
' dataok flag TRUE if good data
' vr(vrbase) contains the data value
'=====
```

# Motion Coordinator Application Note

Number:MC-1045, Revision 1, 2/28/2007

Subject: Peer to Peer using CAN

---

```
read_remote_int:
'=====
'Read int data from Slave
' call with:
'   index = slave target TABLE location
'   slave_id = target slave Node ID
'Returns:
'   dataok flag TRUE if good data
'   vr(vrbase) contains the data value
'=====
```

```
write_remote:
'=====
'Send float data to Slave
' call with:
'   index = slave target TABLE location
'   slave_id = target slave Node ID
'   vr(vrbase) = data value
'Returns:
'   dataok flag TRUE if data sent
'=====
```

```
write_remote_int:
'=====
'Send Int data to Slave
' call with:
'   index = slave target TABLE location
'   slave_id = target slave Node ID
'   vr(vrbase) = data value
'Returns:
'   dataok flag TRUE if data sent
'=====
```

# Motion Coordinator Application Note

Number:MC-1045, Revision 1, 2/28/2007

Subject: Peer to Peer using CAN

---

## **CANSLAVE.BAS:**

The CAN Slave program initializes the Motion Coordinator CAN port and then runs a continuous loop that polls the CAN chip to check for incoming messages. When a message is received it is parsed and the appropriate response sent to the reply channel. For details of the CAN telegram structure, see the section below "CANOpen telegram format".

Like the CAN Master program, there are some initialization parameters and variables that may need to be changed to suit a particular application.

**chan=-1** Set to the Slot Number if a CAN Daughter Board is being used.  
**vrbase=240** The program uses 10 VR() variables, set this to avoid a clash with other programs. (This set of 10 must be different to those used by CANMSTR)  
**cmd\_timeout=500** Time in milliseconds that a routine will wait for the second of a pair of telegrams from the master before resetting the "domain is busy" flag. Set this to a lower number speed up multi-master response. Increase it if spurious data errors occur.  
**VR(vrbase+9)** This must be set either manually at the MP2 terminal or from a MC BASIC program. The number entered is the module\_id of the Motion Coordinator on the CAN network and must be a number between 1 and 127. This number must be the same as the one set in CANMSTR.  
**CAN(chan,baudrt,1)** Set baudrate : 1=500k, 2=250k, 3=125k.

# Motion Coordinator Application Note

Number:MC-1045, Revision 1, 2/28/2007

Subject: Peer to Peer using CAN

---

## **CANOpen Telegram Format:**

These messages are used to access the TABLE data of a node. The CANOpen “Service Data Object (SDO)” channel is used to transfer data using the “CAN based Message Specification” protocols for Upload Domain and Download Domain. The CMS Multiplexed Domain Protocols are used. For further information see the CiA Document *DS202-2 CMS Protocol Specification*.

### **1. SDO Read Integer**

a. Master to Slave (initiate upload request)

COB-ID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x580+ Node_ID	0x40 (64)	Index (lo)	Index (hi)	0x00 (0)				

b. Reply: Slave to Master (initiate upload response – expedited transfer)

COB-ID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x600+ Node_ID	0x43 (67)	Index (lo)	Index (hi)	0x00 (0)	Data (lsb)	Data	Data	Data (msb)

### **2. SDO Read Float**

a. Master to Slave (initiate upload request)

COB-ID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x580+ Node_ID	0x40 (64)	Index (lo)	Index (hi)	0x01 (1)				

b. Reply: Slave to Master (initiate upload response – normal transfer)

COB-ID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x600+ Node_ID	0x41 (65)	Index (lo)	Index (hi)	0x01 (1)	0x07 (7)			

# Motion Coordinator Application Note

Number:MC-1045, Revision 1, 2/28/2007

Subject: Peer to Peer using CAN

---

c. Master to Slave (upload segment request)

COB-ID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x580+ Node_ID	0x60 (96)							

d. Reply: Slave to Master (upload segment response – last packet)

COB-ID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x600+ Node_ID	0x01 (1)	Sign	lsb (FRAC)	msb (FRAC)	Lsb (INT)	Bit1 (INT)	Bit2 (INT)	msb (INT)

### 3. SDO Write Integer

a. Master to Slave (initiate download request – expedited transfer)

COB-ID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x580+ Node_ID	0x23 (35)	Index (lo)	Index (hi)	0x00 (0)	Data (lsb)	Data	Data	Data (msb)

b. Reply: Slave to Master (initiate download response)

COB-ID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x600+ Node_ID	0x60 (96)	Index (lo)	Index (hi)	0x00 (0)				

### 4. SDO Write Float

a. Master to Slave (initiate download request – normal transfer)

COB-ID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x580+ Node_ID	0x21 (33)	Index (lo)	Index (hi)	0x00 (0)	0x07 (7)			

# Motion Coordinator Application Note

Number:MC-1045, Revision 1, 2/28/2007

Subject: Peer to Peer using CAN

---

b. Reply: Slave to Master (initiate download response)

COB-ID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x600+ Node_ID	0x60 (96)	Index (lo)	Index (hi)	0x00 (0)				

c. Master to Slave (download segment request – last packet)

COB-ID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x580+ Node_ID	0x01 (1)	Sign	lsb (FRAC)	msb (FRAC)	lsb (INT)	Bit1 (INT)	Bit2 (INT)	msb (INT)

d. Reply: Slave to Master (download segment response)

COB-ID	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x600+ Node_ID	0x20 (32)							

# Motion Coordinator Application Note

Number:MC-1045, Revision 1, 2/28/2007

Subject: Peer to Peer using CAN

---

For the Initiate download/upload commands, Byte 0 is made up as follows:

Bits 7, 6, 5 : ccs or scs = client/server command specifier

Valid <b>ccs</b> values	Valid <b>scs</b> values
0: download segment request	0: upload segment response
1: initiate download request	1: download segment response
2: initiate upload request	2: initiate upload response
3: upload segment request	3: initiate download response
4: abort transfer	4: abort transfer

Bit 4 : Always 0

Bits 3, 2 : Only valid if Bit 1 = 1 and Bit 0 = 1. It indicates the number of bytes that do not contain data.

Bit 1 : Transfer type. 0 = normal transfer. 1 = expedited transfer.

Bit 0 : Size indicator. 0 = size not indicated. 1 = size indicated.

# Motion Coordinator Application Note

Number:MC-1045, Revision 1, 2/28/2007

Subject: Peer to Peer using CAN

---

## **Message Security:**

It is possible that more than one master may try to get data from the same slave at the same time. This section sets out the security features in place to make sure the wrong data is not returned to the master.

### 1. CAN Prioritization

The CAN protocol uses a dominant/recessive bit in hardware that means that if 2 telegrams clash, the one with the lower COB-ID wins. The loser simply re-transmits when the bus comes free.

### 2. Index Number Confirmation

Each reply to an "initiate" command contains the INDEX number (i.e. the TABLE index). The master program checks that this is the correct index before using the data.

### 3. Domain Open Flag

When an "initiate download" or "initiate upload" has been accepted, the dom\_open flag is set TRUE.

While this flag is TRUE, the node will not accept any "initiate" requests from another master. Also the download/upload segment cannot occur unless the flag is TRUE. At the completion of the segment transfer, the flag is set FALSE again to allow another master to make a request.

There is scope for an additional security measure, albeit a random check, in the toggle bit of the command specifier. (byte 0) This bit could be set alternately high/low on each transfer so that the master can check the reply toggle bit against the state set. A future revision may include this enhancement.

CANOpen has protocols specified for controlling the state of the slave nodes and organizing a node-guarding sequence to make sure other nodes in the system know if a module goes off-line. These features present an opportunity for expanding the CAN Network software and bringing it more into line with the true CANOpen specification.