1. The basic CAN commands (by Frank Voorburg, Feaser)

CAN Commands

```
CANOPEN speed
```

Configures the CAN controller for the specified communication speed and synchronizes to the CAN bus. The message reception acceptance filter is configured to receive all valid CAN message identifiers, both 11-bit STD (0..7FFh) and 29-bit EXT (0..1FFFFFFF).

The communication speed should be in the range 10 kbps to 1 Mbps.

Example:

```
CANOPEN 500000
```

```
CANCLOSE
```

Closes the connection with the CAN bus by placing the CAN controller back into its configuration mode.

Example:

```
CANCLOSE
```

```
CANSEND id, type, len, data(), ok
```

Places a CAN messages in the transmit queue and the moment a CAN transmit message slot becomes available, the CAN message is send onto the CAN bus.

Argument 'id' sets the message identifier and 'type' the identifier type. Set 'type' to 0 for a 11-bit STD message identifier (most commonly used) or to 1 for a 29-bit EXT message identifier. The number of data bytes in the messages is set by argument 'len'. This can be anywhere in the range 1..8. The actual message data is passed as a 1-dimensional array called 'data'.

Pass in a variable for argument 'ok' to determine if the message transmission was successful. In this case 'ok' holds the value 1. If it holds the value 0 then either the transmit queue is full so you can try a little later again or the connection with the CAN bus has not yet been opened (see command `CANOPEN`).

Example:

```
100 DIM txData(2) : DIM txOk
110 CANOPEN 500000
120 txData(0) = &H55
130 txData(1) = &HAA
140 CANSEND &H123,0,2,txData(0),txOk
150 IF txOk = 0 THEN PRINT "Could not send CAN message"
160 CANCLOSE
```

`CANRCV id, type, len, data(), ok`

Checks if a CAN message was received and is stored in the internal receptio queue.

The message identifier is stored in the variable that is passed as argument 'id'. The identifier type is stored in the variable that is passed as argument 'type'. A value of 0 means that it is a 11-bit STD identifier and a value of 1 means that it is a 29-bit EXT identifier. The number of data bytes in the messages is stored in the variable that is passed as argument 'len'. The actual received data bytes is stored in the 1-dimensional array that is passed as argument 'data'.

Pass in a variable for argument 'ok' to determine if a messages was received. In this case 'ok' holds the value 1. If it holds the value 0 then either there was no message present in the reception queue or the connection with the CAN bus has not yet been opened (see command `CANOPEN`).

Example:
```
100 DIM rxId : DIM rxType : DIM rxLen : DIM rxData(8) : DIM rxOk
110 CANOPEN 500000
120 DO
130 CANRCV rxId,rxType,rxLen,rxData(0),rxOk
140 IF rxOk = 1 THEN
150 PRINT "CAN message received"
160 EXIT
170 ENDIF
180 LOOP
190 CANCLOSE
```

---

2. The extended commands (by Kees Zagers, SI-Kwadraat B.V.)

The extended commands are made upward compatible with the standard commands.

The extended version is a specific firmware. It uses the open DMBasic version 2.7, which is open for every user. To make it fit, the Gameduino commands are removed. To work with the CAN firmware a license has to be installed. If no license is installed only the basic commands will work. The license is determined per unit. If the license is correct all functions available, otherwise only available for demonstration.

*At initialisation license, to be used FIFO space and optional protocol are determined. The license cannot be changed by the user; the to be used FIFO can be changed. The original number of FIFO's is 64. If this is the maximum which is needed, no extra memory is reserved for the FIFO's. However if the value is bigger (65 - 1024) a Basic array is created, CANMESSAGEFIFOAREA with a dimension of 4 * No. of FIFO's, eg at 1024: DIM CANMESSAGEFIFOAREA(4096).*

This array is created at the first CANOPEN or CANFIFO statement in a program or in the command shell. The variable CANMESSAGEFIFOAREA is forbidden in Basic in this case.

If an optional protocol is specified in the license file, this protocol is used as custom protocol in the CANLOG command. Optional protocols are available from SI-Kwadraat, but can also be created or extended by the user using an online program on the SI-Kwadraat site.

Timestamps are used in commands like CANLOG, CANVIEW and CANOBJECT. They are derived from in the internal timer in the CANcontroller. This timer is activated by the CANOPEN command. Also the resolution can be controlled by CANOPEN. Default resolution is 100uS. The disadvantage of this timer is the fact that it only has a 16 bit counter (0 - 65535), which means, that it is reset every 6.5 sec. In CANLOG and CANVIEW the timestamp is corrected in the firmware. This means as long as at least one message occurs within 6.5 seconds the timestamp is corrected. This is not the case in the CANOBJECT command, because the timing of these objects can be programmed by the user. Therefore the user has to correct the timestamps in Basic if necessary.

Overview of the BASIC commands:

1. CANOPEN <speed>[,<special>][,<ts resolution>]
   (special and ts resolution added; 1.2 extra functionality in speed)
2. CANCLOSE
   (not changed)
3. CANSTATUS
   (new; 1.2 canport added; 1.4 canobjects added)
4. CANFIFO <fifono (0-31)>,<rx/tx (0/1)>,<length (1-32)>
   (new)
5. CANMASK <maskno(0 - 3)>[,<std mask(0-2047)>][,<ext mask(0-262143)>]
   (new)
6. CANFILTER <filterno (0-31) >[,<std id(0-2047)>][,<ext id(0-262143)>]
   (new)
7. CANLINK <filter no (0-31)>,[<enable (0/1)>,<mask no>,<fifo no>]
   (new)
8. CANSEND <id>,<type>,[<id ext>,]<len>,<data()>,<ok>
   (type extended; id ext added; 1.1 bugfix)
9. CANRCV <id>,<type>[,<id_ext>],<len>,<data()>,<ok>
   (type extended; id ext added)
10. CANLOG [#<fileno>][,<format>][,<fifono>][,<period>][,<lineno>]
    (new; 1.1 functionality extended)
11. CANREG [<regno (0 – 180)||(500-680)||(1000-2023)||(3000-4023)>]
    (new; 1.1 upgrade with write; 1.2 upgrade with FIFO registers from CANFREG)
12. CANFREG [<fregno (0 – 1023)>] or [<fregno (0 – 2047)>] with CANTEST
    (new; 1.1 upgrade with write; deleted in 1.2)
13. CANRESET [canport]
    (new; extended in 1.2)
14. CANVIEW [<format>][,<fifono>][,<period>][,<lineno>][,<interval>]
    (new in 1.2)
15. CANBRIDGE [[#]<fileno>]
    (new in 1.4)
16. CANREPLAY [#<fileno>][,<fifono>][,<format>]
    (new in 1.4)
17. CANOBJECT [#<fileno>][,<fifono>][,<timer>]
    (new in 1.4)

Overview of the BASIC variables

- *CANMESSAGEFIFOAREA:* Array dimensioned for CAN buffer; don't use in BASIC
- *CANFLOAT1, -2, -3:* 3 Basic variables which can be used in the custom protocol (see 3.0)
- *CANTEST:* Array variable used by the commands CANSTATUS, CANFIFO, CANMASK, CANFILTER, CANLINK, CANREG, CANVIEW and CANOBJECT. If this array is defined the output of the command is written in this array and not on the screen (see 3.1).
- *CANPRETRIGGER, CANPOSTTRIGGER, CANTRIGGERERRCNT, CANTRIGGERID and CANTRIGGEREXTID:* Variables used in triggered logging (see 3.2).
- *CANOBJECTPERIOD:* Variable with period time (in ms) of displaying/logging of the objectview in CANLOG

In version 1.1. the following variables have been added:

- *CANBLINK:* Variable when >0 used for blinking of application LED during CAN communication
- *CANCONTINUE:* If used and !=0 CANLOG continues where it ended the last time
- *CANERRCNT:* If used contains the maximum value of the Rx Error Counter after CANLOG
- *CANOVFCNT:* If used contains the number of buffer overflows during a CANLOG
- *CANPERIOD:* Can be used both for setting or reading the period time of CANLOG
- *CANLINENO:* Can be used both for setting or reading the number of lines of CANLOG
- *CANLOAD:* If used contains the overall busload during CANLOG
- *CANLOADMAX:* If used contains the maximum busload during CANLOG
- *CANLOADMAXTIME:* If used contains the time when maximum busload occured.

In version 1.3. the following variable has been added:

- *CANENDOFLINE:* Variable when >0 used for line end during CANLOG

In version 1.4. the following variables have been added:

- *CANEVENTINT:* Variable when >0 used for the script timer interrupt in custom protocol.
- CANOBJnnID: Variable used for storage of ID of object nn (00 - 31)
- CANOBJnnCTRL: Variable used for storage of CTRL of object nn
- CANOBJnnDATA(8): Variable used for storage of DATA of object nn (array of 8)
- CANOBJnnOK: Variable used for result of Tx or Rx

2.1 CANOPEN

CANOPEN still has the speed parameter. The routine to set the speed has been improved. Now we get a better match for all bitrates. Please notice that the lowest bitrate which can be set with this hardware is 25 kbit/sec and not 10 kbit/sec. This is due to the 80 MHz X-tal which has been used. When CANOPEN runs for the first time and CANFIFO has not been executed yet, it will setup two FIFO's, both 32 messages deep. The first one for sending messages the second one for receiving. Just as it does in the standard version. Standard the receiving filter is set to receiving all messages. Standard also the timestamp is enabled and set to a resolution of 100 uS. This is used in CANLOG. Two extra optional parameters are added. Since version 1.2 all parameters are optional and the speed parameter has been extended with more functionality.
In version 1.3 some delay is added after the command and the status is checked.

```
CANOPEN [<speed>[,<special>[,<ts resolution>]]]
```

<speed> is by default the parameter which specifies the CAN bitrate in bits/sec. The range is 25000 up to 1000000. Since version 1.2 it is also possible to specify the bitrate in kbits/sec (25 - 1000). The default value of <speed> is 0. This means the value is not changed. In this way one can change the <special> or <ts resolution> parameter without having to know the actual <speed>.
The values 1-7 are used for autobauding and the values 8-24 to change configuration and control registers of the CAN controller in a very specific way. See table 3.2 (experienced users only)

<special> is used to open the CAN-port in a non-standard way. Speed must be specified in this case. The value of special can be 0 to 255 (8 bits) where all bits have their specific meaning: <oowtmsjj>.
The 2 LSB's (jj) are the two bits to configure the SJW (synchronisation jump width):

00 : 1 clock cycle (default)	01 : 2 clock cycles
10 : 3 clock cycles	11 : 4 clock cycles

The next one (s) determines the number of samples:
- 0 : 1 sample (default)
- 1 : 3 samples

The next one (m) determines the position in the period for sampling:
- 0 : 75% (default)
- 1 : 80%

The next one (t) determines the availability of timestamps:
- 0 : timestamps available (default)
- 1 : timestamps off

The next one (w) enables or disables the wakeup facility
- 0 : disabled (default)
- 1 : enabled

The two most significant bits (oo) determine the operation mode:
- 00 : normal operation (default)
- 01 : loopback mode (also own tx messages can be received)
- 10 : listen-only mode (no errorframes; no ACK)
- 11 : all message mode (all messages, also fault ones, are received)

The third optional parameter (<special> must also be specified in this case) sets the timestamp resolution. This value ranges of 1 uSec minimal to 800 uSec maximal. Default value is 100 uSec.

2.2 CANCLOSE

CANLOSE has no additional features.

2.3 CANSTATUS

CANSTATUS displays the status of the CAN controller. No further parameters (see also 3.1).

```
CANSTATUS
```

The control register: Serial no., Port no., Timestamps (on/off), Mode. Updated in version 1.2/1.4.
The config register: Bitrate, sample point, SJW and no of samples.
The FIFO's are marked T (Tx) or when Rx: _ (not active), + (>1), 1-9 (no of links), 0 (> 9 links).
FIFO status shows if data is available in the FIFO (for Tx always 1; for Rx 1 if data is not read)
FIFO rxlost shows the FIFO's which have had an overflow (1).
Next line shows the actual values of Rx and Tx error counters.
In version 1.4 the status line of the CANOBJECT is incuded here. 1 HEX byte status for every FIFO, meaning: 4 LSB's: fileno; Bit 5: file on/off; Bit 6: Basic parameter on/off; Bit 7: Tx/Rx; Bit 8: ON/OFF. After all objects the CANport of the objects is listed.
If an optional protocol is specified the name of this protocol is on the last line.

2.4 CANFIFO

CANFIFO is the command to change the FIFO configuration. It has three optional parameters.

```
CANFIFO [<fifono (0-31)>][,<rx/tx (0/1)>][,<depth (1-32)>]
```

CANFIFO without further parameters will give the status of all 32 FIFO's (see also 3.1).
Default the FIFO 0 is configured for Tx (length 2) by CANOPEN and FIFO 1 for Rx (32). All other FIFO's are default for Rx with depth 1. However as long as they are not enabled by any link to a filter (CANLINK), they will be disabled.
CANFIFO gives the opportunity to change any FIFO to any mode (Tx or Rx) with any length (1-32). If CANFIFO is entered with only the fifono, the FIFO is reset to rx and depth 1. If it is entered without the depth parameter, depth will be reset to 1.
Please keep in mind that the available space can be limited by the buffers in the license file.

2.5 CANMASK

CANMASK configures the masks which are used to filter the received messages.

```
CANMASK [<maskno(0 - 3)>][,<std mask(0-2047)>][,<ext mask(0-262143)>]
```

CANMASK without parameters will show the configuration of the masks (see also 3.1).
CANMASK with only the number will sets the mask to all 0, meaning all bits are not relavant for filtering.
CANMASK with number and std mask will mask the bits in the standard part which are set to 1. It will not mask the IDE bit itself.
CANMASK with number, std mask and ext mask masks the 1 bits in both standard and extended part. Also the IDE bit is masked.

2.6 CANFILTER

CANFILTER configures the filters, which are used to filter the received messages.

```
CANFILTER [<filterno (0-31) >][,<std id(0-2047)>][,<ext id(0-262143)>]
```

CANFILTER with no parameters will show the status of all filters (see also 3.1).
CANFILTER with only the number sets both std and ext id to 0.
CANFILTER with number and std id sets the standard part to the id.
CANFILTER with number, std id and ext id sets the complete id.

2.7 CANLINK

CANLINK links a filter to a mask and a FIFO. When all values are filled in, the link is also activated or deactivated.

```
CANLINK [<filter no (0-31)>][,<enable (0/1)>][,<mask no>][,<fifo no>]
```

CANLINK with no parameters will show all links (see also 3.1).
CANLINK with only the filter no. disables the filter, no change in mask and FIFO
CANLINK with filter no and enable will enable or disable the link.
CANLINK with filter no, enable and mask will also set the mask
CANLINK with filter, enable/disable, maskno and fifono sets the complete link, including FIFO.
A filter can only be linked to one FIFO; a FIFO however can be linked to more than one filter.
Before a new link can be made the former link has to be disabled.

2.8 CANSEND

CANSEND still uses the same parameters. Only one optional parameter (id ext) has been added. The type parameter has been extended.

CANSEND <id>,<type>,[<id ext>,]<len>,<data()>,<ok>

The new type consists of 16 bits (<f><pp><lllll><nnnnn><s><r><e>), where

- f = Fill bit: 1 means not really a sending action, but fill only
- pp = Send priority bits in FIFO
- lllll = The location address in the FIFO; only used at specific write
- nnnnn = The Fifo no.
- s = The special bit when set; special actions can be undertaken
- r = RTR bit; when s=0 used as RTR bit; s=1 the RTR enable bit is set
- e = Extension bit; when e=1 the <id ext> contains the 18 LSB of ID

The optional <id ext>. The problem in Basic is that it only has single precision floating point variables. So the maximum number which can be calculated without exponential description is 999999. The extended ID however can go up to 536870911. That is why we advise to split the ID in 11 (std) and 18 (ext) parts, when the ID is calculated. If ID is used as a constant it is no problem to go up to 536870911 in ID (was bug in 1.0; repaired in 1.1).

The kind of SEND action is determined by f, s en r bit

- s r f
- 0 0 0 : The standard Tx action: message is added to FIFO and sent
- 0 0 1 : Message is added to FIFO, but not sent yet
- 0 1 0 : As 0 0 0 but with RTR bit set
- 0 1 1 : As 0 0 1 but with RTR bit set
- 1 0 0 : Send only action; all messages in FIFO are sent
- 1 0 1 : Fills the specific location lllll with the message
- 1 1 0 : Sets the RTR enable bit for this FIFO
- 1 1 1 : Reset RTR enable bit; Update priority with pp; Reset FIFO

2.9 CANRCV

CANRCV still uses the same parameters. Only one optional parameter (id ext) has been added. The type parameter has been extended.

CANRCV <id>,<type>[,<id_ext>],<len>,<data()>,<ok>

The new type consists of 16 bits (<g><ss><ttttt><nnnnn><o><r><e>), where

- g = global type; g=1 (set before CANRCV) to receive the message in the new format.
- ss = specific global type:
  - 00 The filter no. which was hit for this message included in ttttt
  - 01 The 5 LSB's of timestamp incuded in ttttt
  - 10 The 5 MSB's of timestamp included in ttttt
  - 11 The complete 16 bit timestamp in type (all other info lost)
- ttttt = The result of ss
- nnnnn = The Fifo no.
- o = Overflow flag o=1 when the overflow flag was set in the FIFO
- r = RTR bit; when r=0 a data frame is received; r=1 an RTR frame
- e = Extension bit; when e=1 the <id ext> contains the 18 LSB of ID, if id ext available.

2.10 CANLOG

CANLOG is the command to be used for analysis of a bus system. It can show the data on a screen/USB terminal as well as write it to a file.

```
CANLOG [#<fileno>][,<format>][,<fifono>][,<period>][,<lineno>]
```

All parameters are optional. If a parameter is to be used, all previous parameters should be filled in also.

- #<fileno>: Fileno has to be used as in PRINT #. Fileno=0 (default) means write to screen/USB.
- <format> Data format, with also selection of trace/object view and optional protocols
- <fifono> 0 -31; default value: FIFO 1
- <period> The measument time in ms.
- <lineno> The number of CAN messages to be logged.

CANLOG can be stopped in four ways:

- CTRL-C: When used on the command line this will come back to the basic prompt.
- ESC does the same as CTRL-C, however this can be used within a BASIC program.
- <period>: When period is given (>0), it will stop after this period time in ms.
- <lineno> When lineno is given, it will stop after the no of messages.

A combination of <period> en <lineno> is possible. It will stop on the event which comes first. If only <lineno> is used, period should be set to 0. See also the variables CANPERIOD and CANLINENO. If <period> or <lineno> are used <ESC> is disabled (since v1.3).

The formats from 0 to 31 are the trace formats:

0 -7:	The standard trace formats
8 - 15:	As 0 - 7 however only registration during pressing of USER button
16 - 23:	As 0 - 7 however triggered by SPACE key
24 - 31:	As 0 - 7 however, triggered by a CANbus error or a pre-defined ID

If the triggered formats are used, the BASIC variables CANPRETRIGGER and CANPOSTTRIGGER can be used to determine the number of messages before and after the trigger moment, which are included in the log. If CANPRETRIGGER is not defined it is supposed to be 0 and if CANPOSTTRIGGER is not defined the logging will continue until it is stopped by one of the other methods. The maximum value of pre-triggered messages is the depth of the buffer which is used for logging. The messages which are in the buffer are copied to another bufferarea to prevent lost messages. The pre-triggered messages are displayed or logged after the post-triggered messages.
If the formats 24-31 are used also an optional variable CANTRIGGERERRCNT is available, which has the value or the Rx error counter when the trigger occurs (1 by default). CANTRIGGERID and CANTRIGGEREXTID (for extended IDs) can be used to determine the ID to trigger on rx ID.

The existing trace formats are:

- 0. The Basic Microchip format (16 byte per message HEX)
- 1. The Basic Microchip format (bytes separated by spaces)
- 2. The UNICANNER format including error information.
- 3. The UNICANNER error only format
- 4. The HEX string format
- 5. The DEC string format
- 6. The ASCII string format
- 7. The Custom protocol format

If format is greater than 31 the object format is chosen. The format can now be divided as follows:
<dddddddd><r><e><f><cc><pp><s>
In the object view the upper free FIFO space is used for storage of the objects. In the small object size one FIFO location (16 bytes) is used per object. In the large object size two FIFO locations (32 bytes). In the small object view only ID, CTRL byte, databytes and no of messages are displayed. In the large object view we see in addition the last timestamp, the first timestamp, the last interval, the smallest interval and the largest interval..

<f>:	The switch between trace and object format; must be 1 for object.
<s>:	The size of the object; 0 means the default large; 1 the small size
<pp>:	The protocol for data: 00 means HEX; 01 DEC; 10 ASCII; 11 no data.
<cc>:	The presentation of the custom protocol line (if any custom protocol is defined) 00 No custom protocol show 01 Custom protocol as additional line 10 Custom protocol on first line, no data 11 Custom protocol on first line, no ID, CTRL and data
<dddddddd><r><e>:	Used for optional object specification; by id, but also by data, RTR and IDE. The first d is first databyte D1; the last is D8. r and e: RTR/IDE. Default they are all 0 , which means only ID object definition.

CANLOG has a timestamp on every received CAN message. The timestamp from the CAN module of the PIC32 controller is used. The default resolution is 100 uS, which can be changed by CANOPEN. As this is a 16 bit number, after about 6.5 sec it will start over again. In the software this is compensated: if timestamp < prev timestamp then add 65536. This works only if messages are seen in every period of 6.5 sec, otherwise gaps in the time are the result.

Two major additions have been added in CANLOG in v1.1:

To show activity on the bus during CANLOG, the application LED can be used for blinking. If the variable CANBLINK has a value >0 the USER LED will toggle between ON and OFF at the number of messages which is given to CANBLINK.

If at least one of the basic variables CANLOAD or CANLOADMAX is defined, a busload measurement is done during CANLOG. CANLOAD contains the overall busload in % and CANLOADMAX the maximum busload during 1 sec. If also the variable CANLOADMAXTIME is defined, this variable contains the second when the maximum busload occured.

Also some additional basic variables are added:

Normally every time CANLOG is called the message number and the timestamp are reset. If the variable CANCONTINUE is defined and has a value other than zero, the log is continued from what was logged before. The CANLOAD values however are only based on the last run.

CANPERIOD and CANLINENO can be used in stead of the same parameters in the command. However they have a double function. If they are defined they will contain the actual measuring period in ms and the actual number of lines at the end of the CANLOG. Please be aware to reset the values before the next CANLOG. Also when the parameters in the command are used, they will overrule the variables.

CANERRCNT and CANOVFCNT are also two new variables. If defined they contain resp. the maximum value of the CAN RX error counter and the value of the number of buffer overflows which occured.

One variable has been added in version 1.3: CANENDOFLINE. Default when the parameter is not defined or zero, every line in the CANLOG trace is ended by CRLF. If CANENDOFLINE=1 the line is only ended by CR, which gives an overlay for every line on the screen. If CANENDOFLINE=2 the line is ended by 2 spaces. If CANENDOFLINE=3 no characters at all are placed at the end of the line. This parameter can be very usefull if the custom protocol is used.

In version 1.4 the custom protocol is extended and better described in chapter 3.0 of this manual. Also the variable CANEVENTINT has beed added. This sets an event bit, which can be used in the custom protocol.

2.11 CANREG

CANREG is implemented for debugging only (see also 3.1).

CANREG [<regno (0 – 180)||(500-680)>] (changed in version 1.1 and 1.2)

It shows the actual hex value in the CAN register(s) with regno when regno<181.
If CANTEST has been made active the values 500-680 can be used to set the registers 0-255 to the values as given in CANTEST(0) and CANTEST(1).

See Microchip datasheet for details about the registers.

2.12 CANFREG

CANFREG is implemented for debugging only (see also 3.1).
In version 1.2 this command is deleted and the CANREG command is used in stead with higher values for the registers.

CANREG [<regno (1000-2023)||(3000-4023)>]

It shows the actual hex values in the FIFO register(s) 0 to 1023 if 999<regno <2024.
If CANTEST has been made active the values 3000-4023 can be used to set the registers 0-1023 to the values as given in CANTEST(0) up to CANTEST(7).

See Microchip datasheet for details about the FIFO registers.

2.13 CANRESET

CANRESET is a command which should be handled very carefully.

CANRESET [<canport>]

This command will initialise the complete CAN configuration again. So if you changed the buffer size or the protocol in the license file, the new parameters will be activated after this command. The command however will not reinitialise the memory. So if you started with a buffer size of 1024 and go back to 64, you also have to give the Basic command *ERASE CANMESSAGEFIFOAREA* or *CLEAR* to free the memory again.

In version 1.2 the CANRESET is extended with a parameter. By default this parameter will be 0. This means the standard CAN interface on the hardware is used. This is directly available. However the PIC32 controller has a second CAN interface internally. The pins are available on the Arduino and GPIO extensions of the hardware, however it is not buffered with any CAN transceiver. The pins are shared on the connectors with digital I/O pins 0 and 1 (PIN(11) and PIN(12) in Basic) and also the serial port (COM1 or COM4 in Basic). These are not available if the second CAN port is used. See also notes in 3.5 of the Duinomite manual.

CANRESET 1 disables the standard CAN port and enables the second one. All other commands are now available on the other CAN port.

If one of the two CANports are enabled and also switched on by the CANopen command, the other CANport can be activated in parallel by the CANRESET 2 command. This will automatically detect which port is already running and will now reset to the other port. In this case nothing can be changed on the first port anymore. The 2nd port can be configured now. This is new in version 1.4. See also CANBRIDGE and CANOBJECT. These are the only two commands which benefit from this option.

2.14 CANVIEW

CANVIEW is a new command in version 1.2

CANVIEW [<format>][,<fifono>][,<period>][,<lineno>][,<interval>]

CANVIEW gives an overview of a running CAN network on one line. The parameters which are displayed are: the number of messages, the number of ID's, the busload and the error situation. Values are displayed periodically, both over the specific period and overall. The optional parameter <interval> can be used to set the measurement period in ms (default value 1000).

The optional parameters <fifono>, <period> and <lineno> are used in the same way as in the CANLOG command. The optional parameter <format> is used for the format of the line.

The default value 0 displays all value in one line as: <SMlLPTiIcCEO> with:

S:	The measurement time in seconds
N:	No of messages
A:	Actual busload in the last interval
L:	The busload overall
M:	The maximum busload
T:	Time of maximum busload
i:	The number of different message ID's in the last interval *
I:	The total number of different ID's *
c:	The maximum value of the Rx Error Counter in the last interval
C:	The maximum value of RxC overall
E:	The total number of errors (in fact the number of increments of the RxC) **
O:	The overall number of overflows

If format has the value 1 to 12 one of the above parameters is shown in the order SMlLPTiIcCEO.

Format = 13: Busload graphical: Dynamic scaling with values A, L, and M

Format = 14: Error Counter value graphical: Scale 0 - 136 with values c and C

Format = 15: The values 1 to 12 sequential. Every parameter is displayed during one period.

* To store the used ID's the upper memory area of the FIFO space is used as in CANLOG. Default 30 FIFO's are free for this purpose. Each FIFO can store 4 ID's. So a maximum of 120 different ID's can be stored. If this maximum is exceeded, the value 999 is stored and the ID read is stopped.

** If the increment of the error counter > 8 the error will probably be caused by the test system itself and therefore this value is set to the maximum value 9999. The user should check for the correct physical and datalink settings of the testbox.

2.15 CANBRIDGE

CANBRIDGE is a new command in version 1.4

CANBRIDGE [[#]<fileno>]

To activate the CANBRIDGE command the following sequence of commands should have taken place:

CANRESET, CANRESET 0 or CANRESET 1.(setting the CAN port)
Optional: CANFIFO (FIFO 0 should be configured for Tx, FIFO 1 for Rx)
Optional: CANFILTER/CANMASK/CANLINK (enable filters on the 1st CAN port)
CANOPEN bitrate, <special>, <timestamp> (if special is used: set to Normal operation)
CANRESET 2 (we switch to the other CAN port now; this will now be the one for all other commands)
Optional: CANFIFO (FIFO 0 should be configured for Tx, FIFO 1 for Rx)
Optional: CANFILTER/CANMASK/CANLINK (enable filters on the 1st CAN port)
CANOPEN bitrate, <special>, <timestamp> (if special is used: set to Normal operation)

CANBRIDGE can now be started and all messages on CANport 0 will be copied to CANport 1 and vice versa. Ofcourse when filters are active, these filters decide which messages are actually copied. If the optional <fileno> is included in the command, logging is done conform the UNICANNER format (format 2 in CANLOG). 0 or #0 will do a screen logging 1 (#1) up to 9 (#9) file logging to the file which is opened for input by this number.

The Basic parameters CANPERIOD, CANLINENO, CANCONTINUE, CANENDOFLINE and CANBLINK can be used as in CANLOG. <ESC> or <CTRL-C> can be used to stop the CANBRIDGE command.

2.15 CANREPLAY

CANREPLAY is a new command in version 1.4

CANREPLAY [#<fileno>][,<fifono>][,<format>]

The CANREPLAY command can be used to send a logged file on the CANbus again. The logfile must be in the UNICANNER format.
The file opend for output by <fileno> (1 by default) will be sent on the CANbus, using <fifono> (0 by default).
If <fifono> is specified it is possible to specify also the <format> parameter. If this is done the file is logged to the screen in the format as it has been specified in CANLOG.

The Basic parameters CANPERIOD, CANLINENO, CANCONTINUE, CANENDOFLINE and CANBLINK can be used as in CANLOG. <ESC> or <CTRL-C> can be used to stop the CANREPLAY command.

2.16 CANOBJECT

CANOBJECT is a new command in version 1.4

```
CANOBJECT [#<fileno>][,<fifono>][,<timer>]
```

The CANOBJECT command has been added to perform send and receive on the CANbus in the background. Once an object has started it will continue running until it is stopped again with the same CANOBJECT command.

A total of 32 CANOBJECTs can run in the background. Every CANOBJECT is linked to an unique FIFO. So by default only 2 objects are active (Object 0 for Tx and object 1 for Rx). If more objects are needed they should be configured first by the CANFIFO and CANLINK (to be active for Rx) commands. Please keep in mind that the objects are scanned sequential starting at 0 and ending as soon as an inactive FIFO has been detected. An object is inactive when the timer is set to 0.

As the objects are linked to the FIFO's it is obvious that we have Tx and Rx objects. The values of the object can be linked to Basic variables. The following variables can be declared (DIM) for this purpose: CANOBJnnID, CANOBJnnCTRL, CANOBJnnDATA(8), CANOBJnnTS. nn should be replaced by the FIFOno. The DATA array contains the databytes. The CTRL parameter by default the DLC, added with 64 when the RTR is activ and added with 128 when EXT is activ. The ID contains the identifier and TS the timestamp of the last object action (see comment on timestamps in 2.0). As the floating point numbers in Basic are only accurate up to 1000000 and the extended identifier can go up to 536870911, it has been decided that the millions of the ID are multiplied by 1000 and added to the CTRL parameter. So a message with ID 536870911 and 8 bytes of data will have: ID=870911 and CTRL=536136 (136=128+8; 536 has to be multiplied by 1000000 and added to ID).

When all Basic parameters are defined for a Tx object, the object will be sent with these parameters. Otherwise the Tx FIFO can be confgured by the CANSEND command (fill only). In this way also multiple messages can be sent by one object. When the Basic parameters are defined for an Rx object, they will be refreshed with every new message for this object. Please keep in mind that all parameters have to be available, otherwise the object will be handled without the Basic parameters.

The timer is linked to the millisecond timer as used in Basic (TIMER). So the minimum repeat time for an object is 1 ms. By default the timer for Tx objects is set to 1000 (Tx every sec) and for Rx objects to 1 (check for messages every 1 ms). Any other value can be given in the optional parameter for every object. The command can be used as toggle between default and off. As the default object (no FIFO specified) is object 0, one can define all objects and after that just toggle them with the command CANOBJECT, because timer0=0 means inactive and also the other objects in this case. If we want only object 0 inactive and all the others still running, we should give the object a high timer value, e.g. 1000000000 or simply define it without BASIC parameters and empty FIFO (no CANSEND).

By specifying a fileno, the object is logged in the UNICANNER format. #0 means logged on the screen, a number #1 up to #9 in a file when it is opened for OUTPUT before. Don't forget to close the file.

The Basic parameter CANBLINK can be used to toggle the green LED.

A counter is used to register all objects which occur, both a successfull Tx as Rx. This counter can be read by the reserved command CANOBJECT 32 (fileno and timer are irrelevant). The value of the counter is returned. If the basic variable CANOBJCOUNTER is defined, the value will be stored in this variable. This variable will only be refreshed after a new CANOBJECT 32 command.

As the CANOBJECTs work in the background it is possible to combine them with other CAN commands. However this can lead to strange situations, e.g. when working with rx-objects and the CANLOG command in parallel. Some messages can be achieved by the CANLOG command and others by the CANOBJECT.

If two CAN ports are available it would be nice to send objects on one port and log them on the other port. Therefore it is possible to start the objects on one port and switch to the other port with CANRESET 2. Now we can use all the other CAN commands on the other port. At the first CANOBJECT command the to be used CAN port for all CANOBJECTs is defined. This can only be reset by going back to this port by CANRESET 0 or CANRESET 1. All objects are reset in this way.

Warning 1: As the CANOBJECT is linked to the millisecond timer it can influence the behaviour of the real-time clock. If the handling of the objects exceeds 1 ms, the next ms interrrupt will be lost. It is known that if logging is activ that the handling exceeds 1 ms, so use logging only for testing and not for an application running in real-time.

Warning 2:Don't use any other CAN Tx or Rx command if the object on that specific FIFO is activ. The behaviour becomes unpredictable otherwise.

CANOBJECT can also be used as a function to get the status:

```
status = CANOBJECT(objectno)
```

For the objects 0 to 31 it will give the value 0 if it is not active and the value of the lower 16 bits of the first address of the FIFO for this object. The higher 16 bits can be read by the CANSTATUS command. Objectno 32 will give the value of the objectcounter.

Examples in basic:

```
> counter = CANOBJECT(32)
> IF CANSTATUS(0) > 0 THEN POKE &HA000, CANSTATUS(0)+8, PIN(0)
```

The last Basic instruction will replace the databyte 0 in the FIFO by the actual value of PIN(0). In this example we use the default configuration with 64 FIFO's and FIFO 0 set as TX FIFO. If the FIFO's are in the Basic memory area, the first parameter will probably be &HA001. Using the POKE command has quite some risk, but it is much faster than searching for the Basic parameter everytime.

3.0 DESCRIPTION OF THE CUSTOM PROTOCOL

The license file can be extended with a custom protocol. Only one protocol can be active, and this protocol will be loaded automatically at bootup. Protocols are described in a readable scriptfile, but have to be converted in a low level machinecode to be read from the runtime CANLOG command.

An on-line utility can be used to convert the protocol script to the embedded code in the licence file.

The machinecode consists of 6 relevant bytes per line: The first one is called the command byte and is specified below; the second one the variable, 3 and 4 are reserved for a constant and 5 and 6 for a jump to another line; in fact the offset of the line. Allthough only 6 relevant bytes are on a line, every line in principle has 10 bytes. This had to be done, because the code is written on the A-drive and if a byte has the value 255, the next byte is skipped. This is compensated in the 4 non-relevant bytes.

If a protocol is added to the license file, two basic commands are extended in fact:
CANSTATUS has an additional line where the name of the protocol is listed.
CANLOG has the custom protocol options enabled and the data both in trace as in object format is converted to the chosen format.

The script has four types of commands:
1. The SPECIAL commands. Used for begin and end of the script. The start command has the name of the protocol included. This is displayed in CANSTATUS. Also special commands are made for floating point variables and since version 1.4 for reading and writing I/O pins, reading the keyboard, configure and send a CAN message and reading and configuring a timer (special values 64 up to 212).
2. The calculation command. One of the 16 user variables gets the value of a calculation. Calculations can be done with only 2 parameters. The first parameter is always a variable. It can be one of the user variables or one of the CAN variables. The second variable can be either a constant value (decimal, hexadecimal or binary) or also one of the user variables.
3. The write command. This command is used to fill the actual protocol line. The parameter of the write command can be some text or a user or CAN parameter.
4. The if command. This command is used to make a jump in the script on a certain condition. The condition has two parameters. The first one is either a user or a CAN variable. The second one can also be such a variable or a constant value (decimal, hexadecimal or binary).

All commands are prefixed by a linenumber and are in principle ended by a ":" and a CRLF. After the ":" can be an optional linenumber to jump to. This is always used in the if command where it is used to jump to if the condition is true. However it can also be used in every other command. Based on this syntax every user specific protocol can be realised.

16 user variables can be used, numbered 0 to 15.

Variable 15 has a specific function.
In the SPECIAL commands the variable gets an error value:
0. No error
1. No key pressed
2. Pin configuration error
3. Pin out of range error
4. Not a Tx Fifo
5. Tx Fifo limit exceeded
6. Tx parameter out of range
7. No messages in Tx Fifo

If used in Calculation it is possible to enter a low level command. See at CALCULATION commands how this is handled.

Also 16 CAN (and other special) related variables can be used. They are specified:

LSB / MSB	00	01	10	11
00	CAN ID	CAN ctrl byte	DB1	DB2
01	DB3	DB4	DB5	DB6
10	DB7	DB8	Mess. no	Mess. spec *
11	Timestamp	Float 1	Float 2	Float 3

* Mess. spec: 32 bit value (mmmmmmmmrrrrrrrroooooooooooooooo) m=max. rxc; r=act. rxc; o=overfl.

The SPECIAL commands: bits 1 and 2 of the command byte are 0:

command	variable	constant	jump	description
0	NA	NA	NA	Not used
4	0	0	0	End of code; start of texts; after this line
8	text	text	text	Text line; can be more than 6 bytes; ended bij CRLF
12	NA	NA	NA	End of protocol file
16	user var	NA	line	Set Float 1 to user variable
20	can var	NA	line	Set Float 1 to CAN variable
24	can dbn	NA	line	Set Float 1 to 4 CAN bytes, n first byte, little endian
28	can dbn	NA	line	Set Float 1 to 4 CAN bytes, n first byte, big endian
32	user var	NA	line	Set Float 2 to user variable
36	can var	NA	line	Set Float 2 to CAN variable
40	can dbn	NA	line	Set Float 2 to 4 CAN bytes, n first byte, little endian
44	can dbn	NA	line	Set Float 2 to 4 CAN bytes, n first byte, big endian
48	user var	NA	line	Set Float 3 to user variable
52	can var	NA	line	Set Float 3 to CAN variable
56	can dbn	NA	line	Set Float 3 to 4 CAN bytes, n first byte, little endian
60	can dbn	NA	line	Set Float 3 to 4 CAN bytes, n first byte, big endian
64	user var	NA	line	Set user variable to ASCII value of key
68	user var	pin no	line	Set user variable to value of I/O pin
72	user var	pin no	line	Set value of I/O pin to user variable
76	user var	NA	line	Set script timer to user variable
80	user var	NA	line	Set user variable to script timer
84	user var	NA	line	Set user variable to event (1 for CANRx, 2 for timer, 3 for both)
88 *	can var	user var	line	Configure CAN FIFO 0 for Tx
92	can var	user var	line	Configure CAN FIFO 1 for Tx
96	can var	user var	line	Configure CAN FIFO 2 for Tx
100	can var	user var	line	Configure CAN FIFO 3 for Tx
104	can var	user var	line	Configure CAN FIFO 4 for Tx
108	can var	user var	line	Configure CAN FIFO 5 for Tx
112	can var	user var	line	Configure CAN FIFO 6 for Tx
116	can var	user var	line	Configure CAN FIFO 7 for Tx

Special commands continued:

command	variable	constant	jump	description
120	can var	user var	line	Configure CAN FIFO 8 for Tx
124	can var	user var	line	Configure CAN FIFO 9 for Tx
128	can var	user var	line	Configure CAN FIFO 10 for Tx
132	can var	user var	line	Configure CAN FIFO 11 for Tx
136	can var	user var	line	Configure CAN FIFO 12 for Tx
140	can var	user var	line	Configure CAN FIFO 13 for Tx
144	can var	user var	line	Configure CAN FIFO 14 for Tx
148	can var	user var	line	Configure CAN FIFO 15 for Tx
152	can var	user var	line	Configure CAN FIFO 16 for Tx
156	can var	user var	line	Configure CAN FIFO 17 for Tx
160	can var	user var	line	Configure CAN FIFO 18 for Tx
164	can var	user var	line	Configure CAN FIFO 19 for Tx
168	can var	user var	line	Configure CAN FIFO 20 for Tx
172	can var	user var	line	Configure CAN FIFO 21 for Tx
176	can var	user var	line	Configure CAN FIFO 22 for Tx
180	can var	user var	line	Configure CAN FIFO 23 for Tx
184	can var	user var	line	Configure CAN FIFO 24 for Tx
188	can var	user var	line	Configure CAN FIFO 25 for Tx
192	can var	user var	line	Configure CAN FIFO 26 for Tx
196	can var	user var	line	Configure CAN FIFO 27 for Tx
200	can var	user var	line	Configure CAN FIFO 28 for Tx
204	can var	user var	line	Configure CAN FIFO 29 for Tx
208	can var	user var	line	Configure CAN FIFO 30 for Tx
212	can var	user var	line	Configure CAN FIFO 31 for Tx
216	fifo no	NA	line	Send CAN FIFO no
220 - 248	NA	NA	NA	220-224-228-232-236-240-244-248 not used
252	0	ptr	line	Start of code; ptr to protocol text

* Only CAN ID, CT and D1 to D8 can be configured. User variables may have offsets of 0 to 31 times 256 to write in a specific buffer of the FIFO. In this way it is possible to fire up to 32 messages with one Send FIFO command. FIFO's and CANSEND must be configured in the right way.

The IF commands: command bit 0 is 0 and command bit 1 is 1

IF x COMP y is true THEN jump to specific line ELSE next line
Comparison:

Command bits 543	comp.	description
000	=	equal
001	<	smaller than
010	>	greater than
011	!=	not equal
100	<=	smaller or equal
101	>=	greater or equal
110	NA	not used
111	NA	not used

Command bit 6 is 0: y = User var; command bit 6 is 1: y = CAN var; not relevant if bit 8 = 0

Command bit 7 is 0: x = User var; command bit 7 is 1: x = CAN var

Command bit 8 is 0: y = Constant (value in constant bytes 3 and 4); Command bit 8 is 1: y is variable.

Variable byte (bits 1-4): y value (all zero if command bit 8 is 0)
Variable byte (bits 5-8): x value

Constant: relevant value if command bit 8 is 0

Jump: the offset of the line no if condition is true.

The CALCULATION commands: command bit 1 is 0 and command bit 2 is 1

x = y CALC z
Calculation

Command bits 543	calc	description
000	+	add
001	-	subtract
010	*	multiply
011	/	divide
100	&	logic and
101	|	logic or
110	<<	shift left (multiply by 2^z)
111	>>	shift right (divide by 2^z)

Command bit 6: reserved for further calculations

Command bit 7 is 0: y = user variable; command bit 7 is 1: y = CAN variable

Command bit 8 is 0: z = constant; command bit 8 is 1: z = user variable (no in constant)

Variable byte bits 1-4: x value

Variable byte bits 5-8: y value

Constant bytes: constant value or user variable

Jump: If value other than 0 the offset for the next line

If the calculation is as follows: var15=varY+varZ, where Y and Z can be any variable between 0 and 14 a new command is calculated with the new command (low byte Y), variable (high byte Y) and constant bytes in varZ. Jump bytes are treated as before.

The WRITE commands: command bit 1 is 1 and command bit 2 is 1

WRITE characters
Kind of characters:

Command bits 543	Description
000	text; constant has the offset value to the text
001	decimal value
010	hex value (no pre string)
011	hex value in format 0xvalue
100	binary value
101	float value
110	ASCII character
111	No of same ASCII characters; variable contains the number; constant the ASCII value

Command bit 6 and bit 8: reserved

Command bit 7 is 0: value is user variable; bit 7 is 1: value is a CAN variable

Variable byte: the user or CAN variable (except if no of ASCII characters is chosen; see above)

Constant bytes: offset to the text (if text is chosen) or ASCII value (no of chr); otherwise not relevant

Jump: If value other than 0 the offset for the next line

In version 1.4. of the firmware a basic variable has been added, CANEVENTINT. When this value >0 the script timer is activated. The value is in ms. The script can now be activated by a received message on the bus, but also based on a time interval. The command 84 is used to check which event occured, 80 is used to write a new value into the timer and 76 to read the current value.

The following protocols are available as script:
CANopen
J1939
FMS

3.1 VALUES OF CANTEST IN THE COMMANDS WHERE IT IS USED

CANTEST should only be used by experienced CAN users. The 61154B.pdf (CAN reference manual) of Microchip should be used for the details about the registers.

Command	00	01	10	11
CANSTATUS	All kind of	variables	(total of 32)	
00xx	MSB sn	Byte 2 +Byte 3 sn	Port no.	mode reg. (0-7)
01xx	Timest. (on/off)	Timest. resolution	Bitrate (b/s)	Sample point (%)
10xx	SJW	Samples	FIFO start LSB	FIFO start MSB
11xx	Used FIFO's	Total FIFO's	Rx Err. cnt.	Tx Err. cnt.
CANFIFO	0 < depth <33	type: rx=0; tx=1	act: rx enable 1/0	act: tx no of mess
	stat status	ovf overflow stat		
0x000-0x1FF	6 LSB's depth +	0x40 * type +	0x100 * act +	
	0x4000 * stat +	0x8000 * ovf		
CANMASK	Two values / mask	1. 11 MSB	2. 18 LSB	999999: mide=1
0xx	Mask 0 MSB	Mask 0 LSB	Mask 1 MSB	Mask 1 LSB
1xx	Mask 2 MSB	Mask 2 LSB	Mask 3 MSB	Mask 3 LSB
CANFILTER	2 values / filter	1. 11 MSB	2. 18 LSB	999999: extid=1
0x000-0x3FF	Filter n MSB	Filter n LSB		
CANLINK	fifono (0-31)	maskno (0-3)	enable (0/1)	
0x000-0x1FF	5 LSB's fifono +	0x20 * maskno +	0x100 * enable	
CANREG	Two values / reg	1. 16 LSB	2. 16 MSB	
0x000-0x3FF	Reg n LSB	Reg n MSB		
CANREG 2 (FREG)	2 bytes of FIFO	in every value		
0xx	Timestamp	sid + filthit	dlc+rtr+extid (lsb)	extid+ide
1xx	D1 + 0x100*D2	D3 + 0x100*D4	D5 + 0x100*D6	D7 + 0x100*D8
CANVIEW	All kind of	variables	(total of 12)	
00xx	All parameters (*)	Measurement time	# Mess total	Busload period **
01xx	Busload total **	Busload max. **	Time busload max.	# IDs period
10xx	# IDs total	Max. RxErrCnt period	Max. RxErrCnt total	Total # errors
11xx	Total # overloads	Busload graphical (*)	Errors graphical (*)	All sequential (*)
CANOBJECT	Status of all	32 FIFO's	(See CANSTATUS)	

(*) Reserved to be compatible with view format; set to 0.
(**) Busload in CANTEST is multiplied by 10 to get a resolution of 0.1% in an integer value.

3.2 Values of the <speed> parameter in CANOPEN

VALUE	DESCRIPTION	REGISTERS
0	Value is not changed	
1-6	Bitrate test 1-2 during 1 sec/bitrate; 3-4 during 0.1 sec; 5-6 during 10 sec	CANCFG for BTR
1, 3, 5	Testing predefined bitrates with display of the process *	CANCON for:
2, 4, 6	As above without display; to be used within a Basic programm *	- Listen Only
7	Testing all possible values **	- SJW = 4
8	Increment PRSEG	CANCFG
9	Decrement PRSEG	CANCFG
10	Increment TSEG1PH	CANCFG
11	Decrement TSEG1PH	CANCFG
12	Increment TSEG2PH	CANCFG
13	Decrement TSEG2PH	CANCFG
14	Increment BRP	CANCFG
15	Decrement BRP	CANCFG
16	Increment measuring point (inc PRSEG or TSEG1PH / dec TSEG2PH)	CANCFG
17	Decrement measuring point (dec PRSEG or TSEG1PH / inc TSEG2PH)	CANCFG
18	Toggle TSEG2PHTS on/off	CANCFG
19	Set ABAT (reset all pending Tx)	CANCON
20	Increment DeviceNet Filter	CANCON
21	Decrement Devicenet Filter	CANCON
22	Toggle SIDLE bit	CANCON
23	Reserved for future use	
24	Toggle CAN module ON/OFF	CANCON
25 -1000	Set bitrate in kbits/sec	CANCFG+CANCON
1001 - 24999	Illegal	
25000 - 1000000	Set bitrate in bits/sec	CANCFG+CANCON
> 1000000	Illegal	

* Bitrates are defined by 4 ranges starting high and taking 50% of it for the next measurement: 1000/500/250/125/62.5, 800/400/200/100/50, 666.7/333.3/166.7/83.3/41.7, 600/300/150/75/37.5

** Bitrate is defined by 1000, decreased with 1% for every next measurement (time 0.1 sec).