PropCAN User's Guide (early draft)

Contents

Getting Started

Overview

Setup

Supported Software

CAN-Do! Network Controller (CDNC)

User Housekeeping Unit (UHU)

Command Set

propCAN supports a small number of use models. The core command set is for programmatic control of CAN traffic. Additional commands are added to enhance interactive use. A set of extended commands is offered for users who are interested in more direct control of the MicroChip MCP2515 CAN Controller and the LEDs. Lastly a number of diagnostic commands are added in support of debugging the firmware. These generally will not be used but are recorded here in case they can be of use. Most of the diagnostic commands are not active unless the device is run in "Debug Mode" (a more verbose, interactive mode of operation).

These four sets of commands are presented in the following sections.

Program Control

The program control commands are the heart of the command set. With these commands you set the CAN bit-rate, open the channel, transmit and receive packets, and close the channel. Additional commands exist for determining if errors are occurring and for configuring advanced behaviors such as time-stamping packets as they are received and automatically sending received packets to the PC without waiting for the PC to ask for them.

S – CAN Bus Speed (Bit-rate)

Sn<CR> where n=[0-8]	Set speed	
S<CR>	Show current speed setting	(Debug mode)

The 'S' Series of commands are used to set the desired CAN speed in bits-per-second as well as show the current setting.

Normal use

Set the speed once before you first open the CAN bus. It should not need to be changed thereafter.

Returns <CR> if succeeded, or <BEL> if failed.

Restrictions

CAN Speed can only be changed while the CAN channel is closed.

Examples

```
S7<CR>   set speed to 800k-bps
<CR>     device says OK, (this would be <BEL> if channel was open)
O<CR>    open CAN channel
<CR>     device says OK
```

O – Open CAN Channel

O<CR>	Open CAN Channel	
OW<CR>	Open CAN Channel for write	
OR<CR>	Open CAN Channel for Read only	(Advanced User)
OL<CR>	Open CAN Channel in Loop-back mode	(Advanced User)
O?<CR>	Show Open State	(Interactive Use)

The 'O'pen CAN channel command series is used to establish connection to the CAN bus after the CAN bit-rate has been setup [See: 'S'peed command]. In general CAN nodes acknowledge each message it sees on the CAN bus. The Open and Open for Write commands instruct propCAN to behave in this manner. In addition to this normal operating mode propCAN supports a lurking (not acknowledging traffic) mode and a loop-back mode wherein no CAN traffic is sent to the CAN bus from the propCAN device. This loop-back mode is used during testing of the propCAN device and is used during testing of the new software interacting with propCAN devices.

Normal Use

After verifying the device is responding and after setting the CAN bit-rate then open the CAN channel for use.

Returns <CR> is the command succeeds (channel was closed) and returns <BEL> if the command fails (e.g., if the channel was already open.)

Restrictions

This Open command can only be send while the channel is closed. You must close the channel before attempting to open it in another mode (or even the same mode).

Example

```
S7<CR>   set 800k bps
<CR>     device says OK
O<CR>    open channel for reads/writes
<CR>     device says OK
… send/receive packets …
C<CR>    close the Channel
<CR>     device says OK
```

t – Transmit a CAN frame (Standard mode – 11-bit ID)

tiiiL1122334455667788<CR>	Transmit Standard-mode Packet

Normal use

After the CAN channel is open then use this command to send Standard-mode packets. Alternatively you can use the 'T' command to send Extended-mode packets.

Returns <CR> if the channel is open and the 't' construction is correct (number of payload characters matches the L character which says how many there should be, etc.)

Returns <BEL> if the channel is not open or the 't' packet was formatted badly.

NOTE: see the D1<CR> command to enable interactive messaging if you wish to understand why the 't' packet was incorrect. Use D0<CR> to turn the interactive messaging back off again.

Restrictions

This command cannot be sent unless the CAN Channel is open.

Example

```
t1430<CR>   send a bus-census message to a widget
<CR>     device says channel is open and packet is formed correctly.
         It will send it.
```

T – Transmit a CAN frame (Extended mode – 29-bit ID)

TiiiiiiiiL1122334455667788<CR>	Transmit Extended-mode Packet

Normal use

After the CAN channel is open then use this command to send Extended -mode packets. Alternatively you can use the 't' command to send Standard-mode packets.

Returns <CR> if the channel is open and the 'T' construction is correct (number of payload characters matches the L character which says how many there should be, etc.)

Returns <BEL> if the channel is not open or the 'T' packet was formatted badly.

NOTE: see the D1<CR> command to enable interactive messaging if you wish to understand why the 'T' packet was incorrect. Use D0<CR> to turn the interactive messaging back off again.

Restrictions

This command cannot be sent unless the CAN Channel is open.

Example

P – Retreive a (P)ending Frame

P<CR>	Receive a single Pending Packet

Normal use

propCAN is in polled mode. Use this command to poll to see if any packets are available. This command returns one if one or more is available. Alternatively you can use the 'A'-series command to retrieve all waiting packets instead of just one.

Returns <CR> if the channel is open but there are no packets.
Returns t... (or T...) followed by a <CR> if a packet was present in the receive queue.
Returns <BEL> if the channel is NOT open.

NOTE: if time-stamping is enabled the received packet will have the 4-character encoded time-stamp appended to the packet. The <CR> follows the time-stamp in this case.

NOTE^2: Use the 'X'-series command to switch between the auto-forward and polled receive modes.

Restrictions

Cannot be sent if channel is NOT open of if in auto-forward mode.

Example

```
P<CR>    give me one available packet
T1430<CR>   device says "here it is"
P<CR>    give me another packet
<CR>     device says "there are no more…"
```

A – Retreve (A)ll Frames (those received but not yet delivered)

A<CR>	Receive all Pending Packets

Normal use

propCAN is in polled mode. Use this command to poll to see if any packets are available. This command returns all the packets that are available. Alternatively you can use the 'P'-series command to retrieve a single waiting packet instead of all of them.

Returns A<CR> if the channel is open but there are no packets.
Returns <BEL> if the channel is NOT open.
Returns a t... (or T...) followed by a <CR> for each packet that is present in the receive queue. After the final t... or T... packet is returned "A<CR>" is returned indicating that there are no more packets.

NOTE: if time-stamping is enabled each received packet will have a 4-character encoded time-stamp appended to it. The <CR> follows the time-stamp in this case.

NOTE[2]: Use the 'X'-series command to switch between the auto-forward and polled receive modes.

Restrictions

Cannot be sent if channel is NOT open or if in auto-forward mode.

Example

```
A<CR>    give me all available packets
T1430<CR>   device says "here's first one"
T15524500<CR>     device says "here's next"
A<CR>    device says "no more, we're done"
A<CR>    any more packets, yet?
A<CR>    device says "no more packets, yet"
```

C – Close CAN Channel

C<CR>	Close the CAN Channel

Normal use

The channel has been open and packets have been sent and received. This is now complete so let's close the CAN channel as we no longer will use it.

Returns <CR> if the channel was open and is now closed.
Returns <BEL> if the channel was NOT open.

Restrictions

Only send if CAN channel is open.

Example

```
C<CR>    Close the channel, we're done
<CR>     device says, OK its closed
```

E – Packet Error Counters

E<CR>	Show the Accumulated Error Counts
EC<CR>	Clear the Error Counts

Normal use

Retrieve the current transmit/receive and error counters. Alternatively, clear/reset the counters to zero (meaning start counting from now).

NOTE: counters are zeroed at power-on and then only manually via this command.

Restrictions

Only send clear if CAN channel is closed.

Example

```
E<CR>    What are our current counters at?
E:Rx0,RE0,RO0,Tx0,TxE0,mE0,uE0,wE0<CR>    device says "these values"
```

F –Read Status

F<CR>	Show the Current Error Status

Normal use

Periodically check the status of the propCAN device by sending this command and interpreting the results.

Returns <BEL> if the channel was NOT open.
Returns F99<CR> where '99' is a single byte BCD encoded (two chars hex) value which is interpreted as follows:

Bit 7	Bus Error
Bit 6	Arbitration Lost
Bit 5	Error Passive
Bit 4	{not used}
Bit 3	Data Overrun
Bit 2	Error-Warning
Bit 1	CAN Transmit FIFO queue full
Bit 0	CAN Receive FIFO queue full

NOTE: this response is formatted to be compatible with the Lawicel CAN232 - V1220 device

NOTE^2: this is sent any time the bits change when in auto-forward mode.

Restrictions

Only send if CAN channel is open.

Example

```
F<CR>    what is your status?
F01<CR>  device says Rx FIFO is full (guess we aren't getting
         packets fast enough!)
```

X – Auto forward packets to PC

X1<CR>	Enable auto-forward-to-PC mode	
X0<CR>	Disable auto-forward mode	(Power-up State)
X<CR>	Show current setting of auto-forward-mode	

Normal use

The propCAN device can be configured to asynchronously forward all can packets it receives to the PC. This changes the operating interaction from a polled environment to an auto receiving environment which reduces the overall amount of serial traffic. This also serves to reduce the CPU overhead for the program controlling the propCAN as it no longer needs to periodically ask propCAN if it has any packets. (It no longer polls.) It simply waits and they arrive.

Use the X1 command to swich propCAN into auto-forward-to-PC mode. Use the X0 to turn this mode off switching back to polled mode.

NOTE: switching this mode on (sending X1) causes the 'A' and 'P'-series "poll for packets commands" to be disabled (they no longer serve any purpose since packets are automatically forwarded to the PC as they arrive.)

Returns <CR> if the command is accepted, <BEL> if not.

Additionally, if in Debug Mode, another string follows <CR> informing of the mode change.

Restrictions

Only send the X0/X1 commands while the CAN channel is closed.
The show command can be sent at any time.

Example

```
X1<CR>   switch into auto-send mode
<CR>     device says OK
A<CR>    have any packets for me?
<BEL>    device says this is not recognized! (we are in auto-send
         mode)
```

Z – Timestamping of Received Frames

Z1<CR>	Enable time-stamping of receive packets	
Z0<CR>	Disable time-stamping	(Power-up State)
Z<CR>	Show time-stamp enable state and tic value	

Normal use

propCAN can time-stamp CAN packets it recieves. It has an internal counter started upon device power up. This 16-bit counter is set to a count rate appropriate to the CAN bit-rate set so that each CAN packet will have a unique time-stamp value which is accurate relative to other received packets and so that the counter will not wrap for as long a time as possible.

As a packet is retreved from the MCP 2515 device the time-stamp value is placed into the buffer in which the packet is stored. This time-stamp is then sent to the PC when the packet is sent to the PC as the final four characters before the trailing <CR>.

Returns <CR> if the 'Z' command was successful, or <BEL> if the comand was not allowed at this time.

Restrictions

Only send the enable(Z1) or disable(Z0) commands while the CAN channel is closed.
The show command can be sent any time.

Example

```
Z1<CR>   enable receive-packet timestamping
<CR>     device says OK
O<CR>    open our CAN channel
<CR>     device says OK
P<CR>    retrieve a received packet (assumes device on bus sent one)
t1250A452<CR>     here's the packet (A452 is the time-stamp which is
         appended after the final payload byte.)
```

Interactive Commands

The interactive commands are generally not useful while propCAN is under program control. However when one is communicating with the propCAN device via a terminal program [e.g., HyperTerm on Windows, Minicom on Linux, etc.] these additional commands can really make life easier. There's a macro '?' command which asks propCAN to send a summary of its current settings. There is also the D-series of commands which provide control over the Debug mode of propCAN. This Debug mode is useful as it makes most commands behave in a much more verbose manner (e.g., when you enter a t… command to send a packet but the formatting is incorrect then you will see a message telling you what is wrong with the't' command instead of just getting a <BEL> character signifying that "something" is wrong with it.)

? – Current State

?<CR>	**Show description of current propCAN state**

Normal use

Determine the overall state of propCAN quickly with the '?' command.

Restrictions

This can be sent at any time.

Example

```
?<CR>    show your current state
D USB to CAN Controller, Propeller-based - PropCAN by KZ0Q<CR>
D0 F/W Debug: OFF<CR>
VA101<CR>
O CAN Closed<CR>
S7 CAN at 800 Kbps<CR>
X0 AUTO Fwd: OFF<CR>
Z0 Time-stamp: OFF<CR>
<CR>
```

D – Identify Device and control Debug mode

D<CR>	**Show propCAN device-identifying string**	
D0<CR>	**Turn Debug-mode OFF**	**(Power-up State)**
D1<CR>	**Turn Debug-mode ON**	

Normal use

Look up the friendly name of the propCAN device with the 'D' command. Control the "Debug-mode" setting with the 'D0', 'D1' commands.

Returns <CR> for command OK, <BEL> otherwise.

Restrictions

The D1 command is only allowed when Debug is OFF, while the D0 command is only allowed when Debug is ON. You can't turn it off when it is off nor can you turn it on when it is already on. This all serves to remind us what state we are really in.

The show form of the command can be sent at any time.

Example

```
D1<CR>   turn on debug mode
<CR>D1 F/W Debug: ON<CR>      device says…
D0<CR>   turn debug mode back off
<CR>D0 F/W Debug: OFF<CR>     device says…
```

V – Show hardware / firmware (V)ersions

V<CR>	Show hardware and firmware versions of attached propCAN

Use the 'V'ersion command to retrieve the current version of the propCAN hardware and the propCAN firmware. The hardware version identifies the circuit board revision while the firmware version shows which release of firmware is being run on the board.

Normal use

Interactively you can determine what version of hardware/firmware you have. Programmatically, in the future if we have to do different things for different versions, this command can be used to determine which version of propCAN is currenly attached.

Restrictions

This command can be used at any time.

Example

```
V<CR>       what version is present?
VA211<CR>   device says version A.2 hardware with Ver 1.1 firmware
```

Extended Commands

f – Show MCP2515 Receive Status

f<CR>	Retrieve and show the MCP 2515 receive status

Normal use

Restrictions

Example

h – Show MCP2515 Interrupt line status

h<CR>	Show state of lines arriving from the MCP2515 device

Normal use

Restrictions

Example

I – Show MCP2515 Interrupt Status

I<CR>	Retrieve and show the MCP2515 interrupt status register

Normal use

Restrictions

Example

s – Explicitly configure MCP2515 Bit Timing Registers

s112233<CR>	Set explicit configuration	(Advanced User)

Normal use

I fyou find that one of the built-in speeds is not what you need then you can directly set the MCP2515 CNF1, CNF2 and CNF3 registers with this command. See the MCP 2515 Datasheet for the bit values needed.

Returns <CR> if the CAN channel was closed, <BEL> otherwise.

Restrictions

Only send this command when the CAN channel is closed.

Example

```
sD20305<CR> Set the bit-rate to the 800k-bps AMSAT standard
<CR>     device says OK
```

R – Read the current value of an MCP2515 register

Raa<CR>	Show the current value of register {aa}

Normal Use

Use this command to (R)ead an MCP 2515 register whose address is {aa}.

Return the command with the value appended followed by <CR> if debug mode is enabled and the register address is valid. Returns <BEL> if these conditions are not met.

Restrictions

Only send this command while in debug mode.

Example

```
R0e<CR>  whats the current value of the CANSTAT register?
R0EV80<CR>  device says the register contains the value 0x80.
```

WV – Write new value to MCP2515 Register

WaaVcc<CR>	Write value {cc} to register {aa}

Normal use

Write a given value directly to one of the MCP2515 registers [00-7F].

Returns <CR> if debug mode is enabled and the register address is valid. Returns <BEL> if these conditions are not met.

Restrictions

Only send this command while in debug mode.

Example

```
W35V08<CR>  set the TXB0DLC value to 8 (transmit buffer 0 payload
         length to 8)
<CR>     device says OK
```

WBV – Write new value to specific bits of MCP2515 Register

WaaBbbVcc<CR>	Modify {bb} bits of register {aa} to value {cc}

Normal use

Use this command to modify one or more bits of an MCP 2515 register which supports individual bit modification. The command is read as follows:

{aa} is the register to be (W)ritten
{bb} is the mask value identifying which (B)it(s) to affect
{cc} is the (V)alue of the bits to be set

Returns <CR> if debug mode is enabled, the register address is valid and the register supports bit manipulation. Returns <BEL> if these conditions are not met.

Restrictions

Only send this command while in debug mode.

Example

```
W2cM80V00<CR>     clear the MERRF bit (0x80) in the CANINTF (0x2C:
        interrupt flags register)
<CR>    device says OK, command is valid and was sent
```

Rn – Configure Retransmission of CAN packets

R0<CR>	Enable automatic retransmission of CAN packets (Power-up State)
R1<CR>	Disable automatic retransmission (One-shot mode)

Normal use

Use the disable command to temporarily stop automatic retransmission of packets. Use the enable command to restore automatic retransmission of packets. This is generally used for testing with one or more CAN devices under special test conditions where automatic retransmission is not desired.

Returns <CR> if if the CAN channel is closed and the mode has changed. Returns <BEL> if these conditions are not met.

Restrictions

Only send these commands while the CAN channel is closed.

Example

```
C<CR>   close the CAN channel
<CR>    device says channel is closed
R1<CR>  stop automatic retransmission
<CR>    device says OK
O<CR>   open CAN channel
<CR>    device says OK
…           transmit/receive CAN packets (performing desired test)
C<CR>   close the CAN channel
<CR>    device says channel is closed
R0<CR>  restart automatic retransmission
<CR>    device says OK
```

Use of LEDs

Just in case you, the program writer, wants to control the propCAN LEDs yourself, we've added the LED commands for you to do so. At power-up the LEDs are under propCAN firmware control. You can configure propCAN to allow external control of the LEDs. Once that is done then you can explicitly control each LED turning it on / blinking (at a rate you choose) / or off. Should you decide that your program is finished controlling them you can turn control of the LEDs back over to the propCAN firmware.

Lxc – Set use of LEDs

LxI<CR>	Set LED x to internal control	(Power-up State)
LxE<CR>	Set LED x to external (program) control	

Normal use

Restrictions

Example

Lxn/Lxch – Set externally controlled LED blink rate

Lx0<CR>	Set LED x to OFF
Lx9<CR>	Set LED x to ON
Lxch<CR>	Set LED x to blink at fixed rate
Lx?<CR>	Show current blink rate of LED x
L?<CR>	Show the current blink rate of all LEDs

Normal use

Restrictions

Example

Diagnostic Commands

Q – Show Queue Status

Normal use

Restrictions

Example

Programming Guidelines

Achieving an early understanding of a few key concepts will simplify your coding life when trying to interface with the propCAN via serial over USB.

First of all propCAN is a USB 2.0 device which is emulating (behaving like) a serial port. Any code you write to talk to propCAN will be code that interacts with a serial port.

Due to the nature of the device onboard propCANwhich is providing this serial over USB behavior (FTDI FT232R) no special drivers are needed when working with propCAN on Linux or on Windows.

General Sequence of Operations

When you first connect to the propCAN serial port the propCAN device is reset and will need to determine what speed you have selected. So here's a general sequence to follow:

```
Set the serial-port speed to the highest available
Open the serial port
(Ensure DTR is set)
Send 3 <CR>s then continue sending <CR>s until they start arriving
        on the receive side
Send the V<CR> command to see if the device is reponding
You should get <CR>Vhhff<CR> where hh is the hardware version and ff
        is the firmware version.  The first <CR> is the OK response
        to your V<CR> command.
Close the channel using a 'C' command in case it was left open.
        NOTE: if it wasn't left open you'll get a <BEL> character
        response instead of a <CR> character.
Set the desired bit-rate using the 'S' command
Open the channel using an 'O'-series command
… send / receive packets
Close the channel using a 'C' command
```

Communication Style

Communication with propCAN is quite simple. Here are the guidelines:

- Two control characters are used:
  - A carriage-return which we show as <CR> which has the value 13-decimal, 0D-hex.
  - A bell character which we show as <BEL> which has the value 7-decimal, 07-hex.
- All commands sent to propCAN end with a carriage-return character (not a zero terminator and not a line-feed character).
- All commands are in clear-text. No binary values are sent to propCAN.
- All commands are case-sensative. The exact case shown must be used.
- Each command sent to propCAN will be responded to, first, with a single character which will be either <CR> indicating the command was OK or a <BEL> signifying that the command was in ERROR. If the command was OK and has some response then the response will follow this initial <CR> and will itself be followed by a ending <CR> character. Exceptions to this rule are the 'A'-series commands and the 'P'-series commands. The 'A' and 'P' commands can return data without a leading <CR>.

Appendix A – Index of commands