Knowledgebase

CANSERVO

The original CANSERVO has been enhanced with additional functions to give the CANSERVO8C. The CANSERVO is described first followed by the CANSERVO8C.

CANSERVO

The CANSERVO is an alternative code that will run on a CANACC8 module or on the CANSERVO module. When used with the CANACC8 module, the output IC (U4), which is a Darlington driver, should be replaced with a resistor array (resnet) of 8 x 1K resistors. The 12V version of the CANACC8 (ACC8_2) has output options for either the Darlington array or the resnet. On this board, the jumpers for the outputs should be set to the resistor network. The same code can be used in the CANACC5 board but this is a waste of the bipolar output driver ICs. Also, the voltage for the output driver stage must be set to 5V for the servo inputs.

While the CANSERVO code uses the common pushbutton and LED sequence for allocation of a node number, it can only be configured in FLiM mode, using the facilities of the FCU or a similar software tool.

Node variables 36 (NV# of 1 to 36)

The first NV (NV1) is used for determining whether a servo output signal cuts off when the end position is reached or not. NV2 is used for testing the settings of a servo during the teaching process. NV3 and NV4 are currently unused.

The remaining NVs are in blocks of four, one block per servo output. Within each block, the first two NVs set the end positions for an ON and OFF event respectively and the second two NVs set the speed of travel corresponding to the ON and OFF events.

Number of stored events 32 or 128 depending on the firmware version (EN# of 1 to 32 or 128)

Number of EVs per event 2. (EV# of 1 or 2)

EV1 determines which servo output the event applies to. Allowed values of 00000000 to 11111111 where each bit corresponds to one of the 8 outputs. A bit set makes that output active. Multiple servos can move with a single event.

EV2 determines the polarity of each active servo output. Allowed values of 00000000 to 11111111. A bit set reverses the output relative to the ON or OFF OpCode.

The module ID number is 11. The manufacturer number is 165.

Teaching the CANSERVO module.

Give the module its FLiM Node Number by the usual process. The CANSERVO cannot be used in SLiM mode.

The CANSERVO will drive 8 conventional R / C servos, one from each output and numbered 1 to 8. Each servo can be set for its end positions and movement speeds. With a servo connected to the appropriate output, put the module into learn mode.

<0x53><NN hi><NN lo>

The positions and speeds are set using Node Variables (NVs) These start at NV5 and are in groups of 4. The sequence is:

NV(n) Position for an ON event NV(n + 1) Position for an OFF event NV(n + 2) Speed for an ON event NV(n + 3) Speed for an OFF event.
The configuration software device should send the teaching command to each NV using its index. (NVSET)

<0x96><NN hi><NN lo><NV#><NVval>

When setting the servo positions, the above command can be sent repeatedly, varying the NVval each time and the servo will follow the value. The centre value is 127 with the extremes of 0 and 255. The same process is followed for both the ON and OFF positions. There is no reason which way should be ON or OFF, i.e. the ON position can be clockwise and the OFF be anticlockwise or vice versa.

The speed values are set by the same process. However, the value range is 0 to 7 with 0 being the fastest (set only by the servo mechanism) and 7 is the slowest. If a speed is set, then subsequent tracking when setting the positions will be at the set speed. It is recommended that the speeds be set to 0 when setting the positions and then changed for the required speed. This makes setting the positions easier.

The effect of a setting process can be tested while in learn mode by sending a value to NV2.

The format is 'D000NNNN' where D is direction (1 is ON and 0 is OFF). NNNN is the servo number, 0001 to 1000 for servos 1 to 8. The servo will now act as if an event had been sent so you can check if the positions and speeds are satisfactory.

The above process can be repeated for each servo.

NV1 is used to set whether a servo output cuts off after about 2 seconds when the end position is reached or not. Some servos have been found to buzz or jitter when stationary and removing the drive signal prevents this. However, there is now no reference for the servo position so if the output shaft is moved by external forces, it will not try to drive back to its correct position. Hence the option in NV1. A one bit for each servo in NV1 sets the cutoff. The default is all set to all cutoff. (NV1 = 0xFF)

When the process is completed, take the module out of learn mode.

<0x54><NN hi><NN lo>

All NV settings can be read back using NVRD.

<0x71><NN hi><NN lo><NV#>

The response is NVANS.

Teaching events. This follows the same process as for the other consumer modules. There are 32 or 128 allowed events depending on the firmware version which can be either long or short. Each event has two EVs, EV1 sets which servo(s) the event applies to and EV2 the direction of that servo movement relative to an ON or OFF event.

Stored events and EVs can be read back using NERD or NENRD.

Supported OpCodes (HEX value and mnemonic) in FLiM mode

HEX	Mnemonic		HEX	Mnemonic		HEX	Mnemonic
			59	WRACK			
10	RQNP		5C	BOOTM			
						98	ASON
42	SNN		6F	CMDERR		99	ASOF
50	RQNN		70	EVNLF		9B	PARAN
51	NNREL		71	NVRD		9C	REVAL
52	NNACK		72	NENRD		B2	REQEV
53	NNLRN		73	RQNPN			
54	NNULN		74	NUMEV			
55	NNCLR					D2	EVLRN
56	NNEVN		90	ACON		D3	EVANS
57	NERD		91	ACOF			

58	RQEVN		95	EVULN		EF	PARAMS
						F2	ENRSP

Hardware considerations.

R/C servos consume current in pulses while driving to position. These pulses can be as high as 1 Amp per servo with a typical average of 200mA. While servos use a DC supply of 5 to 6V, it may not be wise to run them off the same 5V DC supply as is used by the CANSERVO module for its processor. The pulsatile nature of the current may interfere with correct operation of the processor. The MERG CANSERVO PCB and the CANACC8 servo adapter board allow for a separate DC supply for the servos.

When at the end positions, with the pulse train stopped, the current in each servo is steady and about 10mA per servo depending on the make and type.

CANSERVO8C (provisional)

The CANSERVO8C is a development of the CANSERVO that will run on a CANACC8 module or on the CANSERVO module. When used with the CANACC8 module, the output IC (U4), which is a Darlington driver, should be replaced with a resistor array (resnet) of 8 x 1K resistors. The 12V version of the CANACC8 (ACC8_2) has output options for either the Darlington array or the resnet. On this board, the jumpers for the outputs should be set to the resistor network. The same code can be used in the CANACC5 board but this is a waste of the bipolar output driver ICs. Also, the voltage for the output driver stage must be set to 5V for the servo inputs.

While the CANSERVO8C code uses the common pushbutton and LED sequence for allocation of a node number, it can only be configured in FLiM mode, using the facilities of the FCU or a similar software tool.

Node variables 37 (NV# of 1 to 37)

The first NV (NV1) is used for determining whether a servo output signal cuts off when the end position is reached or not. NV2 sets the position at startup. Each bit corresponds to one servo. A bit set will cause the servo to run to the OFF end and a bit clear will cause it to run to the ON end as set in the servo position NVs. NV3 is used in conjunction with NV2 to determine whether a particular servo moves at all on startup. A bit set will allow it to run and a bit cleared will give no movement on startup. NV4 determines whether a particular servo will move on a command event while other servos are running or wait till others have finished. A bit set will cause that servo to wait. A bit clear will allow servos to move together. This NV will sequence servo movement so power requirements are not excessive if several servos try to move at the same time. The remaining NVs are in blocks of four, one block per servo output. Within each block, the first two NVs set the end positions for an ON and OFF event respectively and the second two NVs set the speed of travel corresponding to the ON and OFF events. The NV with index of 37 is for testing during setup with the FCU.

Number of stored events 128 (EN# of 128)

Number of EVs per event 4. (EV# of 1 to 4)

EV1 determines which servo output the event applies to. Allowed values of 00000000 to 11111111 where each bit corresponds to one of the 8 outputs. A bit set makes that output active. Multiple servos can move with a single event.

EV2 determines the polarity of each active servo output. Allowed values of 00000000 to 11111111. A bit set reverses the output relative to the ON or OFF OpCode.

EV3 is used to allocate events for position reporting by the CANSERVO8C code. There are 4 such 'feedback' events allowed per servo.

EV4 is available but not currently used.

The module ID number is 19. The manufacturer number is 165. **Teaching the CANSERVO8C module.**

Give the module its FLiM Node Number by the usual process. The CANSERVO8C cannot be used in SLiM mode.

The CANSERVO8C will drive 8 conventional R/C servos, one from each output and numbered 1 to 8. Each servo can be set for its end positions and movement speeds. With a servo connected to the appropriate output, put the module into learn mode.

<0x53><NN hi><NN lo>

The positions and speeds are set using Node Variables (NVs) These start at NV5 and are in groups of 4. The sequence is:

NV(n) Position for an ON event NV(n + 1) Position for an OFF event NV(n + 2) Speed for an ON event NV(n + 3) Speed for an OFF event. The configuration software device should send the teaching command to each NV using its index. (NVSET)

<0x96><NN hi><NN lo><NV#><NVval>

When setting the servo positions, the above command can be sent repeatedly, varying the NVval each time and the servo will follow the value. The centre value is 127 with the extremes of 0 and 255. The same process is followed for both the ON and OFF positions. There is no reason which way should be ON or OFF, i.e. the ON position can be clockwise and the OFF be anticlockwise or vice versa.

The speed values are set by the same process. However, the value range is 0 to 7 with 0 being the fastest and 7 is the slowest. If a speed is set, then subsequent tracking when setting the positions will be at the set speed. It is recommended that the speeds be set to 0 when setting the positions and then changed for the required speed. This makes setting the positions easier.

The effect of a setting process can be tested while in learn mode by sending a value to NV37.

The format is 'D000NNNN' where D is direction (1 is ON and 0 is OFF). NNNN is the servo number, 0001 to 1000 for servos 1 to 8. The servo will now act as if an event had been sent so you can check if the positions and speeds are satisfactory.

The above process can be repeated for each servo.

NV1 is used to set whether a servo output cuts off after about 2 seconds when the end position is reached or not. Some servos have been found to buzz or jitter when stationary and removing the drive signal prevents this. However, there is now no reference for the servo position so if the output shaft is moved by external forces, it will not try to drive back to its correct position. Hence the option in NV1. A one bit for each servo in NV1 sets the cutoff. The default is all set to all cutoff. (NV1 = 0xFF)

NV2 is used to so servos will drive to a known position on power up. Each of the 8 bits corresponds to one servo, the LSbit is servo 1 and the MSbit is servo 8. A bit set (1) will drive the servo to the OFF end of travel and a bit clear (0) to the ON end of travel. Default is all moving to OFF end. (NV2 = 0xFF)

NV3 is used to override NV2 if you don't want the servo to move at all on power up. A bit set allows movement and a bit clear prevents movement. The default is all moving (NV3 = 0xFF)

NV4 is used to enable sequential movement of servos on the module. As servos take current in pulses of up to 0.5 amps peak when moving, the current if all 8 move at once could be excessive. A bit set in NV4 tells that servo to wait till others have stopped. Where several are requested to move, the servo with the lowest number will move first. In NV4, a bit set will cause that servo to wait. A bit clear will allow simultaneous movement. The default is sequential for all. (NV4 = 0xFF)

When the process is completed, take the module out of learn mode.

<0x54><NN hi><NN lo>

All NV settings can be read back using NVRD.

<0x71><NN hi><NN lo><NV#>

The response is NVANS.

Teaching events. This follows the same process as for the other consumer modules for the commands to move. . There are 128 allowed events which can be either long or short. Each event has two EVs for movement. EV1 sets which servo(s) the event applies to and EV2 the direction of that servo movement relative to an ON or OFF event. Note that CANSERVO8C also has EV3. For command events, this should be set to 0.

Stored events and EVs can be read back using NERD or NENRD.

This version has the ability to send back events when end positions are either left or reached and also an event when the mid travel position is crossed. These must be taught in a similar way to the command events. The difference is in the value of EV3.

This must be set as follows.

ABCNNNDD

A indicates a feedback event. It must be set to 1. B indicates the polarity of the response. If set to 0, an ON event will be sent when the servo reaches the end of travel and an OFF event when it leaves that end. Setting it to a 1 will reverse the polarity. C sets the polarity of the mid travel event. If set to 0, it will send an ON event when travelling towards the ON end. If set to 1, it will give an OFF event. This mid travel event is intended to work in conjunction with a CANACC8 and a relay for frog switching. The ability to switch polarity may be useful if the frog changes incorrectly. NNN is the servo in question. 000 is servo 1, 111 is servo 8. DD sets which end or mid position the event applies to. 00 is for the ON end of travel, 01 is for the OFF end of travel, 10 is for the mid position. 11 is not presently used.

Each servo now has three events associated with it for position reporting. These can be either short or long events, depending on what was taught. Normally, they would be taught as short events (NNhi, NNlo = 00) so the end positions become 'device numbers'. These feedback events can be used to set the lights on a control panel, set signals etc and report to a PC for interlocking. They do not guarantee the turnout or signal has actually changed. It is a useful alternative if fitting position microswitches is not wanted.

When teaching the response events, you only need to teach with EV3 set. EV1 and EV2 are not relevant.

Notes:

If a command is sent to move a servo and it is already at that end, it will report back with an ON event for that end even though it hasn't moved.

If the servo end positions (Device numbers) are polled with a short request (ASRQ), they will reply with the corresponding state, as a short response.

This version of the PIC code includes the ability to force a CAN_ID self enumeration using OpCode ENUM (0x5D), to set a specific CAN_ID using CANID (0x75) and will self enumerate if the module pushbutton is pressed briefly.

Supported OpCodes (HEX value and mnemonic) in FLiM mode.

HEX	Mnemonic		HEX	Mnemonic		HEX	Mnemonic
0D	QNN		59	WRACK		96	NVSET
10	RQNP		5C	BOOTM		97	NVANS
11	RQMN		5D	ENUM		98	ASON
42	SNN		6F	CMDERR		99	ASOF
50	RQNN		70	EVNLF		9B	PARAN
51	NNREL		71	NVRD		9C	REVAL
52	NNACK		72	NENRD		B2	REQEV
53	NNLRN		73	RQNPN		B5	NEVAL
54	NNULN		74	NUMEV		B6	PNN
55	NNCLR		75	CANID		D2	EVLRN
56	NNEVN		90	ACON		D3	EVANS
57	NERD		91	ACOF		E2	NAME
58	RQEVN		95	EVULN		EF	PARAMS
						F2	ENRSP

Hardware considerations.

R / C servos consume current in pulses while driving to position. These pulses can be as high as 1 amp per servo with a typical average of 200mA. While servos use a DC supply of 5 to 6V, it may not be wise to run them off the same 5V DC supply as is used by the CANSERVO module for its processor. The pulsatile nature of the current may interfere with correct operation of the processor. The MERG CANSERVO PCB and the CANACC8 servo adapter board allow for a separate DC supply for the servos.

When at the end positions, with the pulse train stopped, the current in each servo is steady and about 10mA per servo depending on the make and type.

cbus, servo

---

cbus/canservo.txt · Last modified: 2021/12/31 22:54 by JohnFletcher