TicStep Firmware Documentation - PRELIMINARY (Jeffrey Kerr 11/5/2018)

TicStep firmware is alternate firmware for the Pololu Tic T500 stepper motor controller board. It uses a multi-drop serial interface that allows precisely coordinated motion of any number of motors, as is required for controlling robot arms, CNC machines, or other devices with multiple axes.

Theory of Operation

Control of the stepping is governed by two timers (a Step timer and a Path timer), and by a Path Point buffer. The Step timer is a high resolution timer (running off of 12Mhz clock) that produces one step pulse per Step period. The period of the Step timer is programmable to achieve the desired stepping speed. The Path timer is a lower speed timer with a fixed period of 20millisec.. Every time the Path timer triggers, a new Step period value is taken from the Path Point buffer and programmed into the Step timer. For a given Step period value, an exact number of steps will be executed during the 20 millisec. Path interval. Given a desired trajectory (*i.e.*: position as a function of time), the Path Point buffer is filled with the incremental numbers of steps to be taken at each 20millisec. interval.

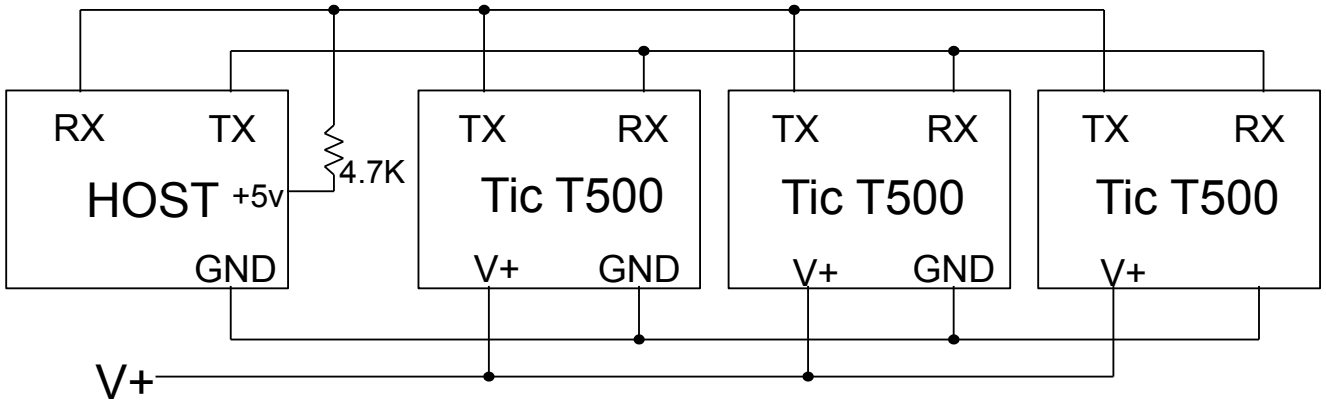
The Path Point buffer is a circular buffer with 128 entries. Thus, a full buffer can hold about 2.5 seconds worth of path data. To start a motion, the buffer is loaded with some number of path points and then a separate command is issued to start executing path points. As long as new path points are added before the buffer empties, the motor will continue to move smoothly for any length of motion. It is up to the host controller loading the path points to make sure that the incremental number of steps in each path point represent a smooth motion to keep the stepper from stalling or losing steps.

The multi-drop communications interface requires each controller board to be programmed with an Individual Address and also a Group Address. Multiple controllers can share the same Group Address. Command data is sent out to all the controllers, but is only executed by controllers with a matching address (group or individual). In typical operation, path point data is sent to controllers' Individual Addresses, and then a single Start Motion command is sent to the Group Address for all controllers involved in the multi-axis motion. This ensures that all motors start moving at exactly the same time. The paths then remain synchronized within the accuracy of the processors' timing crystal. As long as none of the moving motors has depleted its path buffer, new path points can be added to each controller asynchronously.

Communications Protocol

The TicStep firmware uses 230,400 baud asynchronous communications, using the RX and TX pins on the Tic T500 board (5v TTL levels). The protocol is a strict master/slave system where a Tic module will only send data in response to a host command addressed to it. To implement a multi-drop network, firstly, the host's TX signal is connected to all of the Tic modules' RX pins. Secondly, the Tic's TX pin output driver is automatically disabled whenever a module receives ***any command*** data from the host (independent of who the command is addressed to). This way, all module TX pins can safely be connected together and then to the host's RX input. A module's TX pin is only re-enabled when the

module is ready to start sending back status data. Note that the host's RX input will need a pull-up resistor to +5v to keep it from receiving spurious data when none of the modules are transmitting.



A command packet from the host consists of the following bytes:

Header byte: always 0xAA
Address byte: an Individual Address, Group Address or Universal Address of 0xFF
Command byte: lower nibble = command value, upper nibble = # additional data bytes
0-15 additional data bytes (depending on the command)
Checksum byte: 8-bit sum of the Address byte thru the last additional data byte.

Once a command is received, if the address matches the module's Individual Address, its Group Address or if it equals the Universal Address of 0xFF, the command will be processed. Processing typically takes a few microseconds.

One additional addressing attribute is that each module can optionally be declared a "Group Leader" for its Group Address. (Only one module should be declared "leader" for any single Group Address.)

After a command has been processed, if the command was sent to the Individual Address, or if the command was sent to the Group Address *and a module is also the Group Leader*, the leader module will send back a status data packet. For commands sent to the Group Address where the module is *not* the Group Leader, or any command sent to the Universal Address, the module will not send any status data packet in return.

A status data packet consists of the following 7 bytes:

Status byte: individual bits indicating operating status
Path Buffer size: number of un-executed path points still in the path point buffer
Position Counter (4 bytes): 32 bit step counter, LSByte first
Checksum byte: 8-bit sum of the 6 bytes above

The bits of the Status byte are defined as follows:

Bit 0: MOVING (1 = motor is moving, 0 = motor is stopped)
Bit 1: FAULT (stepper driver fault pin state)
Bit 2: LIM1 (limit switch 1/FWD input state)
Bit 3: LIM2 (limit switch 2/REV input state)
Bit 7: Cksum Error (1 = error in received data, 0 = data OK)

Performance Limitations (*see Tic T500 documentation for hardware specifications*)

Step rate: Max = 32000 steps/sec, Min = 50 steps/sec un-dithered, unlimited if dithered
Communications time: 0.0011 sec. to send a command adding 7 path points at a time
e.g.: sending motion commands to a 6-axis robot will take about 50 millisec. per second of motion

TicStep Command Set

Command: Reset Position
Command Byte Value: 0x00
Data Bytes: none
Description:

Resets the internal position counter to zero.

Command: Set Address
Command Byte Value: 0x31
Data Bytes: 3

Individual Address (0 – 254) – *factory default of 1*
Group Address (0 – 254) – *power-up default of 128*
Group Leader (1 = group leader, 0 = not a group leader) – *power-up default of not a group leader*

Description:

Sets the module's address attributes. If the Individual Address is changed from the original power-up value, it will also be stored in EEPROM so that it is available on power-up. Initially, each module should be connected to a host individually (with no other modules connected) to give it a unique Individual Address. Once unique Individual Addresses have been assigned (and marked on each board), multiple modules with unique addresses can be interconnected. The Group Address and Group Leader attributes can be dynamically changed at anytime during normal operation. Setting the address can be done with the Windows TicStep utility program. If a module's address is unknown, it can be set to a known value by sending a Set Address command to the Universal Address of 0xFF. Note: writing to EEPROM takes about 3 millisec, so there will be a delay in receiving any status data back.

Command: Start Motion
Command Byte Value: 0x05
Data Bytes: none
Description:

Starts the execution of the path stored in the path point buffer. This command will also automatically enable the on-board driver chip. Note: when the path point buffer is depleted, the motor will stop, with the driver still enabled.

Command: Set Motor Current
Command Byte Value: 0x26
Data Bytes: 2

Holding Current (0 – 31) - *power-up default of 1 ~174mA*
Running Current (0 - 31) - *power-up default of 1 ~174mA*

Description:

Sets separate motor current levels for holding (not moving), and for running (motor moving). The current level value corresponds to the approximate actual current levels:

0 : 1 mA	8 : 1092 mA	16 : 1762 mA	24 : 2366 mA
1 : 174 mA	9 : 1189 mA	17 : 1835 mA	25 : 2451 mA
2 : 343 mA	10 : 1281 mA	18 : 1909 mA	26 : 2540 mA
3 : 495 mA	11 : 1368 mA	19 : 1982 mA	27 : 2634 mA
4 : 634 mA	12 : 1452 mA	20 : 2056 mA	28 : 2734 mA
5 : 762 mA	13 : 1532 mA	21 : 2131 mA	29 : 2843 mA
6: 880 mA	14 : 1611 mA	22 : 2207 mA	30 : 2962 mA
7 : 990 mA	15 : 1687 mA	23 : 2285 mA	31 : 3093 mA

Yellow Levels are above the passive cooling rating for the board and may cause thermal faults
Red Levels are above the absolute current rating for the driver chip and are not recommended

---

Command: Stop Motion

Command Byte Value: 0x17

Data Bytes: 1

Stop Mode (0 = driver turned off, 1 = driver left enabled)

Description:

Stops the stepping motion of the motor by disabling the Step timer and the Path timer, and the Path Point buffer is cleared. This stops the motor motion abruptly. (To decelerate to a stop, the host would need to add path points to create a deceleration profile.) If the Stop mode byte is set to 1, the driver will remain enabled after stopping. If the Stop mode byte is zero, the driver will be disabled. This command can also be used while not moving to simply enable or disable the driver.

Command: Set Microstepping Rate

Command Byte Value: 0x18

Data Bytes: 1

Microstepping Mode (0 = 1x, 1 = 2x, 2 = 4x, 3 = 8x)

Description:

Sets the microstepping rate for the on-board driver.

Command: Set Limit Switch Mode

Command Byte Value: 0x19

Data Bytes: 1

Stopping Mode (Bit 0: enable autostop, Bit 1: switch inputs active HIGH, Bit 2: disable driver)

Description:

Defines how the limit switch inputs (LIM1 is mapped to the SDA pin, LIM2 is mapped to the SCL

pin) are used. If autostop is enabled (Bit 0 of the mode byte), activating LIM1 will terminate any forward motion, and activating LIM2 will terminate any reverse motion. Bit 1 of the mode byte sets whether the switch inputs are active HI or LO. If Bit 2 is set, the driver will also be disabled when the motion is terminated. Note that the path buffer will also be cleared when autostop occurs.

Command: Single Step
Command Byte Value: 0x1B
Data Bytes: 1
Direction (1 = forward, 0 = reverse)
Description:
Generates a single step pulse to the driver, along with the direction set to forward or reverse. Note that the Single Step command neither enables or disables the driver. If the driver should first be enabled or disabled using the Stop Motion command. Note it is not recommended to use this command while executing a path motion.

Command: Write to Register
Command Byte Value: 0x3C
Data Bytes: 3
Register Address Low Byte
Register Address High Byte
Register Value
Description:
This low-level command can be used to write a value directly to any register within the PIC18F25K50 processor's 16 bit memory space, although it is mostly intended to provide access to the processor's Special Function Registers. This function is primarily intended to provide the ability to patch in or test new features.

Command: Add Path Points
Command Byte Value: 0xnD
Data Bytes: n = 2, 4, 6, 8, 10, 12 or 14
Point 1 Timer Reload Low Byte (LSBit used to determine direction)
Point 1 Timer Reload High Byte
Point 2 Timer Reload Low Byte (LSBit used to determine direction)
Point 2 Timer Reload High Byte
.
.
Point 7 Timer Reload Low Byte (LSBit used to determine direction)
Point 7 Timer Reload High Byte

Description:

This command is used to add between 1 and 7 additional path points to the Path Point buffer. The values added are not the actual number of steps per path period, but a Step timer reload value. The step timer will count up from the reload value up to the counter roll-over at 2^16 counts. To determine the reload value a particular number of steps (*nsteps*), the following formula is used:

reloadval = 0x10000 - (int)((PATH_CYCLES+PADDING)/*nsteps* - STEP_LATENCY + 0.5))

PATH_CYCLES, PADDING and STEP_LATENCY are timing values relating to exact timing of the firmware step routines:

PATH_CYCLES = 240095
PADDING = 200
STEP_LATENCY = 47.085

Because of limitations of the timing resolution, for nsteps > 350, it is impossible to get exactly the number of steps desired. The formula above will give you the closest possible reload value, but the actual number of steps for a given reload value can be determined from the formula:

nsteps = (int)(1 + PATH_CYCLES/(0x10000 - *reloadval* + STEP_LATENCY))

The difference between the actual number of steps per path cycle and the desired number of steps must be accounted for in the host software.

The direction of motion is encoded in the LSBit of the reload value, and therefore the reload value calculated above should be decremented by 1 if the value is odd, and the *nsteps* calculation should also use the decremented value. Once the final reload value determined, the Bit 0 should be set (fwd) or cleared (rev) depending on which way you need to move. The firmware will flag this bit internally and then clear the LSBit before using it to reload the Step timer.

The default 12 MHz clock used for the Step timer can actually only provide a minimum of 4 steps per 20 millisec. path cycle. To command only 1, 2 or 3 step per path cycle, use the following special values for the reload value:

1 step per cycle, reverse: 0x02
1 step per cycle, forward: 0x03
2 steps per cycle, reverse: 0x04
2 steps per cycle, forward: 0x05
3 steps per cycle, reverse: 0x06
3 steps per cycle, forward: 0x07

The firmware will detect these special values and use an alternate 3 MHz clock and appropriate reload values to achieve the exact number of steps requested.

Command: No Operation
Command Byte Value: 0x0E
Data Bytes: none
Description:

The No Op command takes no action but causes the current status data packet to be returned.

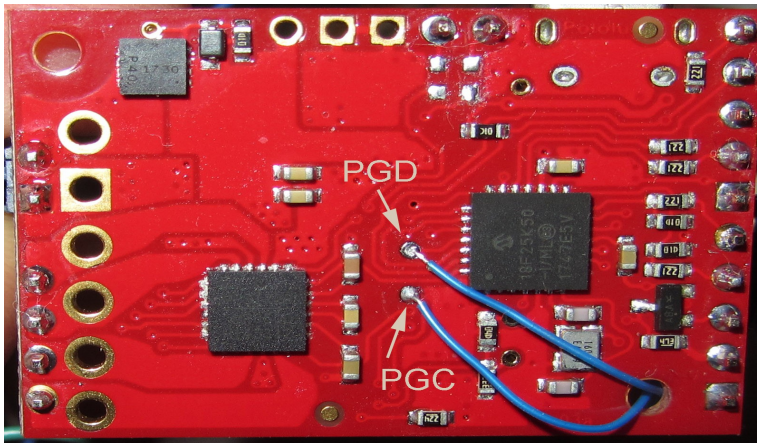
Command: Hard Reset
Command Byte Value: 0x0F
Data Bytes: none
Description:

Resets the on-board processor to its power-up state.

Loading Firmware / Board Modifications

The TicStep source code and firmware, along with a Windows test utility program can be downloaded from: _____ This firmware was developed using the free Microchip MPLAB X IDE (v.5.05) and the XC8 (v2.00) compiler. The MPLAB X IDE connects directly to a Microchip PICKIT 4 or ICD4 programmer, which is needed to program the PIC18F25K50 processor.

To load this firmware, the PICKIT 4 or ICD4 programmer needs to connect directly to the programming pins of the PIC18F25K50 processor. The Ground, Vcc and Reset/Vpp pins are already available on the Tic T500 headers. The PGD and PGC processor pins, however, only appear as exposed pads on the underside of the board as shown in the figure below. You will need to solder two small wires to these pads to connect these pins to the programmer. (Note that these pins are also used to set the microstepping rate for the driver chip. These connections do not interfere with the programming the processor but they do limit using the driver chip while debugging.)



Please use one of the many Microchip MPLAB X IDE tutorials for details on loading and programming the TicStep project.