GCUSB-nStep cserial commands @ 2010-2013 GC
For use in development of any RTS2 driver

Position manipulation

Device supports saving 5 stored positions for later GoTo

GoTo saved position 'index' 0 to 4: :GX# X = 0 to 4

```
           sprintf(buf,"#:G%01d#", index);
      No response returned
```

Save current position :CQX# at index 'X' = 0 to 4:

```
           sprintf(buf,"#:CQ%01d#",index);
      No response returned
```

Maintained in device flash acroos power cycles

Read saved position :RQX# at index X =0 to 4:

```
          sprintf(buf,"#:RQ%01d#",index);
      Response:
         num_bytes = read(portFID,buf1,7);
      Format +123456 or -123456
```

Force set current position, NOT a Move command, SIGN and 6 digits required, e.g. +123456 or -123456

```
      Example for '0'
        sprintf(buf,"#:CP+000000#");
      No response returned
```

Device current position maintained in device flash acroos power cycles

Read current position:

```
        sprintf(buf,"#:RP");
```

Last position maintained in device flash acroos power cycles

```
      Response:
            num_bytes = read(portFID,Pos,7);
            7 bytes with sign, e.g. -123456 or +123456 or +000000
```

Set max step speed, range 1 to 254, lower is faster

```
Step speed roughly in ms is value/1465
      Example value = 1, rate = 1465 steps/second
            value = 2, rate = 1465/2 = 732 steps/second
            value = 3, rate = 1465/3 = 488 steps/second
            value = 10, rate = 1465/10 = 146 steps/second
        sprintf(buf,":CS%03d#", MaxSpeed);
      No response returned
```

Maintained in device flash acroos power cycles

Set requested current step rate

```
      Same range and values as max step rate above
        sprintf(buf,":CO%03d#", Step_Size);
      If step speed is faster than max step speed, command ignored
      No response returned
```

Maintained in device flash acroos power cycles

Set stepping phase:

```
       sprintf(buf,":CW%01d#", Phase);
      Range 0 to 2
      Device supports 3 different step sequences which handle all wiring orders
      going to motor
      See http://www.stepperworld.com/Tutorials/pgUnipolarTutorial.htm
      Search for: "Shortcut for finding the proper wiring sequence"
```

Maintained in device flash acroos power cycles

Read current temperature

```
        sprintf(buf,"#:RT");
      Response:
            num_bytes = read(portFID,Temp,4);
      Four bytes including sign, fixed point
      Example: -101 = -10.1C, +275 = +27.5C
      If response = -888 then no temperature sensor found
```

Move focuser forward, backward using wave/half/full torque stepping

```
            case DIR_FORWARD:
                  sprintf(buf,":F0%d%03d#",mode,count);
                  break;
            case DIR_REVERSE:
                  sprintf(buf,":F1%d%03d#",mode,count);
                  break;
            case DIR_STOP:
                  sprintf(buf,":F1%d%03d#",mode,0);
                  break;
      No Response returned

      Valid range 001 to 999 steps for 'count'
      'Mode' is stepping type:
            0 - wave (1 wire energized per step
            1 - half alternate 1 wire, 2 wire energized/step
            2 - fÃ¼ll torque (2 phases active per step)
      Can be written at any time, even during current stepping
      To force a stop, send 000 in last direction commanded
```

Focuser moving?

```
           sprintf(buf1,"S");
      Response: 1 byte,  '0' not moving, '1' moving
```

Set coil state after move:

```
        sprintf(tempstr,":CC1");  <-- de-energize coil
        sprintf(tempstr,":CC0"); <-- Keep coils energized, WATCH for motor heating!!!
      No response returned
```

Maintained in device flash acroos power cycles

Set up serial IO for usb-serial device

Example sequence

Check if device is usb-nStep

```
      Send 0x6 binary,  Response 1 byte 'S' if usb-nStep
      :RP   , read saved current position 7 chars leading '+' or '-'
      :RC   , read coil on/off after stepping, 1 byte response 0, 1 or
2
      :RO   , read step rate, 3 byte response 001 to 254
      :RS   , read max step rate, 3 byte response 001 to 254
      :RW   , Read phase wiring selection, 1 byte response 0, 1 or 2
      :RT   , Read temp to determine if sensor attached, 4 byte
response SXXX fixed point temperature, -888 no sensor connected
```

```
Items stored in device flash:
            Read max speed    (:RS)
                                        buf[0]=':';
buf[1]='R';buf[2]='S';buf[3]=0;
                                        num_bytes =
write(portFID,buf,3);
                                        num_bytes =
read(portFID,buf1,3);
                                        buf1[3] = 0;
                                        MaxSpeed = atoi(buf1);

            Read current step rate (:RO)
                                        buf[2] = 'O';
                                        num_bytes =
write(portFID,buf,3);
                                        num_bytes =
read(portFID,buf1,3);
                                        buf1[3] = 0;
                                        CurrentStepRate = atoi(buf1);

            Read motor wiring phase selection (:RW)
                                        buf[2] = 'W';
                                        num_bytes =
write(portFID,buf,3);
                                        num_bytes =
read(portFID,buf1,3);
                                        buf1[1] = 0;
                                        RigelPhase = atoi(buf1);

            Read stored positions (:RQX X = 0 to 4)
                                        buf[0]=':';
buf[1]='R';buf[2]='Q';
                                        for(i=0;i<5;i++)
                                        {
                                            buf[3]=(char)(i +
0x30);buf[4]=0;
                                            num_bytes =
write(portFID,buf,4);
                                            num_bytes =
read(portFID,buf1,7);
                                             buf1[7] = 0;
                                             Rigel_Positions[i]=
atoi(buf1);
                                         }
```

```
      Send        Action and/or Response
      ctrl-F      response 'n' for gcusb-nFOCUS focuser, 'S' for gcusb-
nStep
      S           response 1 if moving focuser, 0 if not
      QQQQ        Force reboot in flash upgrade mode (HID device)
      :FDSXXX#    Focus in dir D at step type S for XXX steps (S = 0 =
F, 1=H, 2=T)
                  Sending XXX = 000 = stop all motion
      :COXXX#     Configure step time, increment = 0.68ms
      ~~:CFXXX#     Configure focus off time (NOT used gcusb-nstep~~)
      :CSXXX#     Configure Max Speed for main module, 1 = fastest, 250
= slowest
      :CCX        Keep coils X= on(=0)/off(=1) after stepping
      :CWX        Set phase array 0, 1 or 2, three selections cover all
wiring possible
      :CPSXXXXXX  Force set current position to signed value with 6
digits
      :RO         Read focus ON time (current step rate)
      ~~:RF         Read focus off time - NOT USED gcusb-nStep~~
      :RS         Set max speed allowed to step from the speed dial
      :RT         Read temp, format SXXX fixed point, e.g. +123 =
+12.3C, -054 = -5.4C, -888 for no sensor attached
      :RC         Read coil on/off after stepping, 0 = keep on, 1 =
turn off after step
      :RP         Read current position, returns signed number "+/-
XXXXXX" sign + 6 digits returned

      :RQX  Read saved position X, X = 0 to 5
      :CQX  Save position X with the value of the current position
      :GX         Goto saved position X
```

Debug commands, drive phase wiring directly from computer side

```
      :PSX# Output X to port B lower 4 bits
```

Change low level stepping sequences, NOT maintained across reboots

```
      :SXXXXX...# Input lower 4 bits of 72 bytes to stepper array
```

Can be used by gcusb-nStep modular controller w/display or gcusb-nStep with wireless adapter for smart phone wireless link to debug display

```
      :RDXYAAA    Write/Read from display
                  X = Count out to display
                  Y = count in from display
                  AAA.. = out display then read Y back
      :CDXAAA        Write to display
                  X = count, AAA = data out, NO terminating #!!!
```

Advanced commands for using internal temperature compensation

```
      :TTSXXX#    Configure temp change for comp, Sign + XXX , fixed
point decimla -095 = -9.5C
      :TSXXX#           Configure temp comp move, steps per temp
change=XXX
      :TAX        Configure temp comp 0=off, 1=one shot, 2=auto
      :TBXXX#           Configure temp comp backlash, steps=XXX
      :TI               Prime for manual comp
      :TCXX#            Configure temp comp timer, range 1 to 75
seconds

      :RA         Read temp change for comp, format SXXX fixed point
      :RB         Read temp step for comp, format XXX
      :RG         Read temp comp state, format X
      :RE         Read temp comp backlash, format XXX

      :RH         Read temp comp timer
```