Stepper Motor Controller

RevA

Center for Cold Matter

Noah Fitch
February 11, 2015

Contents

Known Issues

1. Somehow I forgot to put a pullup resistor on the $\overline{\text{MCLR}}$ line of the PIC. In order to operate, put a 10kΩ pullup resistor to 5V. Just under the programming connector is a convenient place (0603/0805 between that and the adjacent pin).

Minimum Requirements

1. If using the provided GUI, you will need windows XP or newer (system has only tested on Windows 7).
2. If PC control/interaction with the controller is desired, you must at the very least have the FTDI COM port drivers installed. These are included in the gui installation executable.

1 System Overview

This circuit and associated firmware/software act as a controller for a standard unipolar (6 wire) stepper motor. The rotary position of the motor can be controlled from a usb connection using a simple serial protocol, or with the provided gui. In addition, a number of potentially useful predefined operation modes are available for controlling the motor. The hardware also includes a rotation sensor, making possible both open and closed-loop operation.

All files and folders discussed in this documentation are located in the `CCM_0001_Unipolar_Stepper_Motor_Controller\RevA` directory. File and folder paths referenced herein are described *relative* to this location.

1.1 Hardware

The heart of the controller is a PIC18F46K80, with a number of peripherals to facilitate motor control. These include a DS3231M real time clock, 24AA1025 external eeprom, HCTL-2021 quadrature encoder, AMT102 rotation sensor, L297 motor driver, and power FETs for the output drive stages. Two external bnc inputs (trigger and direction control) and two pushbuttons facilitate control of the motor by the user in various operation modes. An LCD screen provides visual feedback of motor behaviour to the user.

The motor used for the purpose of this documentation, and that which was used in the initial builds of the prototype circuits, is part number 191-8299 from RS components. This is a 12V, 0.4A/phase 1.8°/step motor. The motor runs quite hot when supplied with the full specified voltage, for this reason the prototypes were built using a 7.5V supply instead. The lower supply voltage sacrifices a bit of switching speed and torque but otherwise has no adverse consequences. With these operating parameters, the fasted stepping rate is on the order of 500 Hz. Stepping speed performance and reliability is load dependent and should be tested for the particular applications.

1.2 Open vs. Closed Loop Operation

Although probably not needed for most applications, the controller can be operated in closed-loop mode. When operated this way, the current position of the motor is determined by the AMT-102 rotation sensor, rather than by how many steps the controller has instructed it to take.[1] When operated closed-loop, the controller will attempt to make this position equal to the desired position (determined by the number of triggers received in external mode, or where it thinks the motor should be in other modes). Closed-loop operation will probably be most useful in situations where the load torque is significant, which can cause the stepping motor to "slip", i.e. not actually take a step even though the controlling currents toggled correctly. If it is crucial that the motor position be accurate, it is recommended to use closed-loop mode.

[1] The rotation sensor is set to output 800 quadrature encoded pulses per rotation. The quadrature encoder used multiplies this by 4 so we get 16 counts for every step of the motor.

2 GUI

In addition to the developed firmware and hardware, there is also a provided GUI that facilitates control of the motor position and configuration of the predefined operation modes. The GUI itself was written in Labview, which is horrible of me I know. However, this was based on an old project and I didn't see the value in re-inventing the wheel there.

The controller system can be installed by running `gui\Installer_Builds\Volume\setup.exe`. This will carry out installation of the gui, FTDI device drivers, and Labview runtime engine. A shortcut to the gui executable will be created on your desktop. Updates of the gui can be carried out by simply replacing the executable itself, there is no need to rerun the entire installer.

When the gui executable is run, the first thing the back end code will do is search for a controller on the specified COM port. The result of this test is indicated with the "Controller Detected?" LED. Each time a button is depressed, and an action is taken, the "Task Completed?" LED will indicate the success (or lack thereof) of carrying out the desired operation. For tasks that take a long time to execute, such as a goto, this LED will blink yellow while the task is carried out. Explicit views of the data sent to and returned from the controller are shown in the two boxes to the left of the tab control. Responses of AOK typically indicates the command was received and executed successfully. ERR will indicate there was an error.

2.1 Basic Operation

The basic operation tab of the controller gui allows for immediate control of the motor position (goto/move) as well as setting a number of operation variables. These include uploading the current time, setting position limits, step rates, etc.

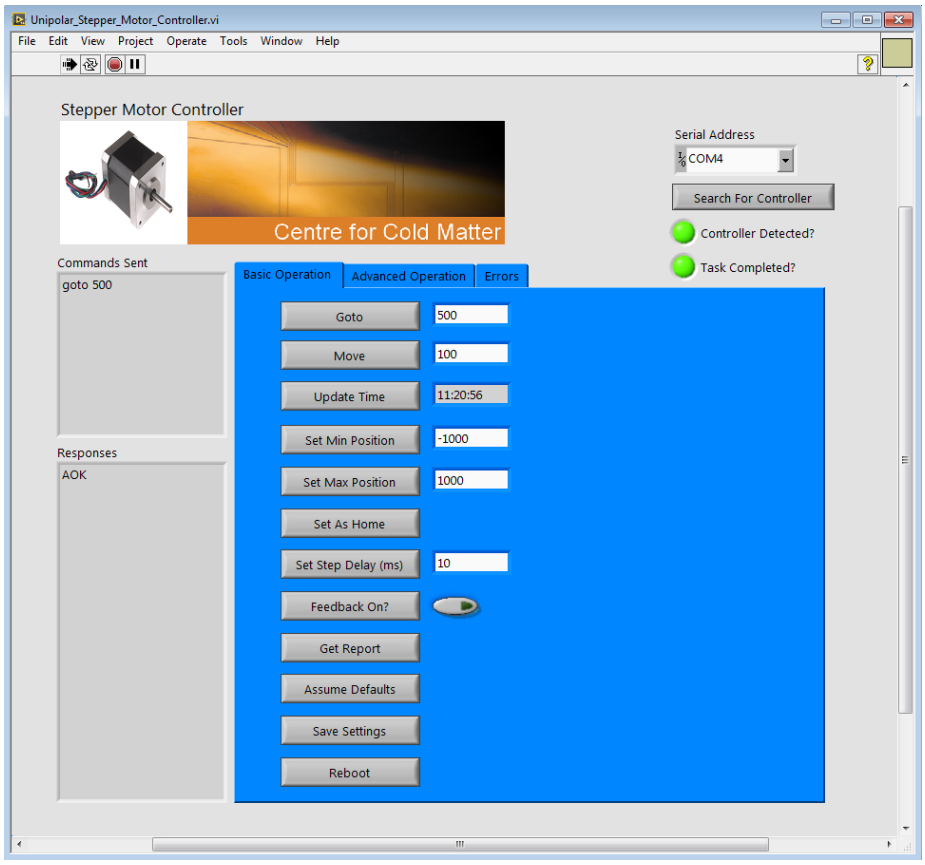


Figure 1: Basic operation tab of the motor controller gui.

2.2 Advanced Operation

The advanced operation tab (see Figure 2) of the gui is used to configure the predefined operation modes native to the controller firmware. Various configuration settings will be enabled/disabled, depending on the operation mode selected in the top dropdown menu. The sequence of serial commands sent to the controller to configure the mode (in the Figure the triggered-goto mode has just been configured) is explained in detail in Section 3.2.

2.2.1 Motor Disabled (0)

In this mode the motor is disabled (stepping currents are turned off), allowing for free rotation.

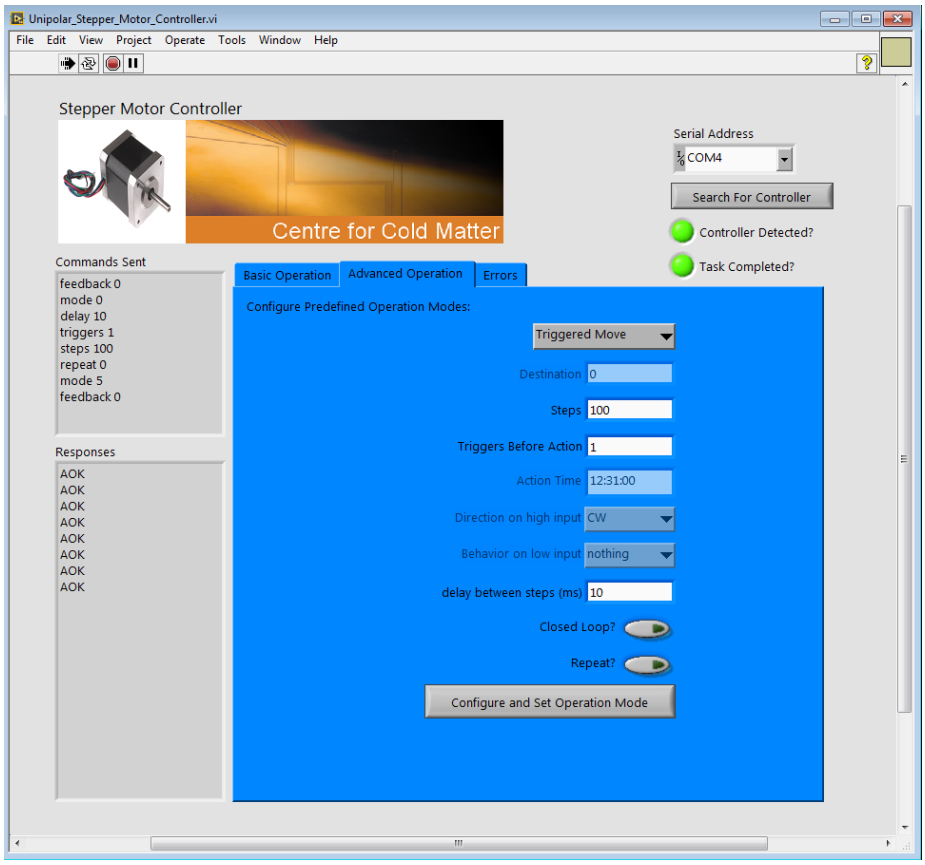


Figure 2: Advanced operation tab of the motor controller gui, used for configuring the predefined modes of operation.

2.2.2 Manual (1)

In manual mode, the position of the motor is controlled using the two front-panel pushbuttons. Push once to step the motor once, push and hold to continually rotate the motor.

2.2.3 External (2)

In external mode, the stepping rate and direction of the motor are controlled by the two bnc inputs, steps being taken on rising edges of the trigger input. At the time of the rising edge of the trigger, the direction logic level determines if a positive (1) or negative (0) step is to be taken. The direction logic input should be stable for at least $\pm 10\mu$s around the rising trigger edge.

2.2.4 Triggered Goto (3)

In triggered-goto mode, the controller is loaded with a goto destination and required number of triggers. It then counts incoming triggers on the external bnc input (rising edges), the goto being completed when the desired number of triggers has been reached. After completion, the controller automatically assumes manual mode.

2.2.5 Timed Goto (4)

In timed-goto mode, the controller is loaded with a goto destination and action time. The current time is monitored until a match is detected for hours/minutes/seconds, at which point the goto is completed. After completion, the controller automatically assumes manual mode.

2.2.6 Triggered Move (5)

In triggered-move mode, the controller is loaded with a step value and required number of triggers. It then counts incoming triggers on the external bnc input (rising edges), the move being completed when the desired number of triggers has been reached. Behaviour after the move is determined by the "repeat" setting. If true, the controller resets the trigger count and prepares to repeat the move procedure once the required number of triggers has again been reached. If false, the controller assumes the manual operation mode.

2.2.7 Timed Move (6)

In timed-move mode, the controller is loaded with a step value and action time. The current time is monitored until a match is detected for hours/minutes/seconds, at which point the move is completed. Behaviour after the move is determined by the "repeat" setting. If true, the controller resets the trigger count and prepares to repeat the move procedure once the required number of triggers has again been reached. If false, the controller assumes the manual operation mode.

2.2.8 State-Based Rotation (7)

In state-rotation mode, the controller continually rotates the motor. The rotation direction is determined by the instantaneous state of the external "trigger" bnc input and the "rotate" and "behave" settings, as summarized in Table 1. When the external trigger input is high, the rotation direction is determined by the rotate setting. When low, the motor can either do nothing (behave=0), or anti-rotate (behave=1) against the normal "external trigger is high" direction. No feedback is possible during this mode.

2.2.9 Triggered Shutter (8)

In the triggered-shutter mode, the controller waits for a number of triggers (rising edges on the external trigger input) before moving the motor a specified

Ext Trigger	rotate	behave	motion
1	0	-	negative rotation
1	1	-	positive rotation
0	-	0	no motion
0	-	1	anti-rotate ($\overline{\text{rotate}}$)

Table 1: Motor control in the "state rotate" operation mode is controlled by a single external input (Ext Trigger) and two internally set bits (rotate and behave).

number of steps, in effect "opening" a shutter. The controller then resets the trigger count and waits for the same number of triggers before returning to the home position. This mode repeats indefinitely.

2.2.10 State Shutter (9)

In the state-shutter mode, the controller monitors the external trigger input and positions the motor in one of two places depending on the current logic state. The direction and rotation amplitude is set by the "steps" variable. This mode repeats indefinitely.

3 Serial Communication

In addition to the GUI, general operation and configuration of the individual predefined operation modes can be configured directly over a serial port connection. This project uses a baud rate of 4800, 8 data bits, 1 stop bit, no port control, and no parity bit.

Each exchange between the host (PC) and the controller consists of the host giving the controller a task. Task-dependent behaviour, including exchange of data or changing settings, can then take place. The assignment and carrying out of each task follows a fixed protocol, as summarized in the following list. For clarity of discussion regarding communication between the host and controller, red characters/data are sent from the host to the controller and blue characters/data are sent from the controller to the host.

1. The wakeup ascii character ? is sent to the controller to instigate communication.
2. When ready to receive the command, the controller returns !. If the wakeup characters was incorrect, the controller returns ? instead and aborts the operation.
3. The command, with required arguments, is sent to the controller. If a command has arguments, the command and it's arguments are separated by a space. The command/argument sequence terminates with a carriage return ascii character (CR, 0x13) or line feed (LF, 0x10).[2]
4. The controller carries out the specified task.
5. The controller ends the current task by sending "AOK" if the task was carried out without errors, or "ERR" if an error occurred, potentially with an error message, terminated by a line feed (LF, 0x10).

An example of adjusting settings on the controller from a serial port terminal (TeraTerm in this case) is shown in Figure 3. The terminal is setup to echo host keyboard entries to the screen. As shown, each command begins with the exchange of "?" (sent by the host), with "!" returned from the controller.

[2] Technically, *any* non-alphabetic or numeric ascii character will work, other than a space.

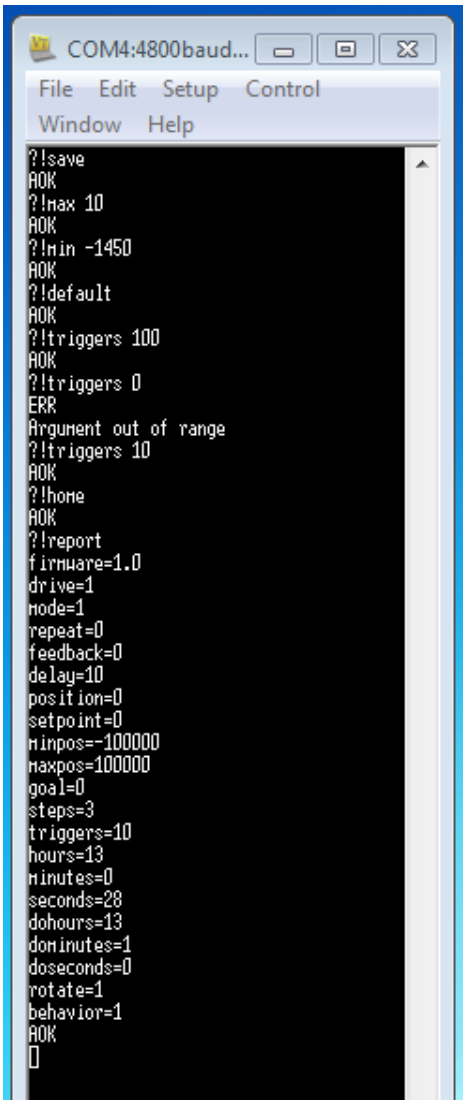


Figure 3: Example interaction with the controller over USB using individual commands sent from teraterm.

3.1 Available Commands

Available commands sent to the controller and brief description of the associated task are summarized in Table 2. The "goto" and "move" commands are used for immediate motor operation. Some of the other commands are only used in certain operation modes.

Examples of the usage of each command in Table 2 appears below. Some

commands require arguments, while others do not. In each example, commas (,) indicate carriage returns and periods (.) indicate line feeds.

3.1.1 report

Instructs the controller to return a list of the current settings, each setting is of the format "setting=value", with a carriage return between each list entry.

?! report , $data1, data2, ... AOK.$

3.1.2 default

Instructs the controller to restore default values.

?! default , AOK.

Command	Description	used in modes
report	report current settings	-
default	assume default values	-
save	save settings	-
home	set position as home	-
reboot	reboot the controller	-
move	move the motor N steps	-
goto	goto position	-
max	set maximum position	all
min	set minimum position	all
delay	set delay between steps	all
feedback	set feedback on/off	all
mode	set operation mode	-
steps	set number of steps	5,6,8,9
triggers	set trigger events before action	3,5,8
goal	set destination	3,4
hours	set current time (hours)	4,6
minutes	set current time (minutes)	4,6
seconds	set current time (seconds)	4,6
dohours	set action time (hours)	4,6
dominutes	set action time (minutes)	4,6
doseconds	set action time (seconds)	4,6
repeat	set loop behaviour condition	5,6
rotate	set rotation direction for logic high input	7
behave	set state-rotate behaviour for logic low input	7

Table 2: Available commands to send to the controller from the host.

3.1.3 save

Instructs the controller to save the current settings. These will be restored if the controller is reset.

?! save , AOK.

3.1.4 home

Instructs the controller to set the current position as home (0).

?! home , AOK.

3.1.5 reboot

Carries out a restart of the controller equivalent to that during a hard power supply cycle. The communication flow looks like:

?! reboot , AOK.

3.1.6 move

Moves the motor a specified number of steps. The number of steps must be between -2000000 and 2000000.

?! move -100, AOK.

3.1.7 goto

Moves the motor to a specified position. The destination must be between -1000000 and 1000000.

?! goto 1000, AOK.

3.1.8 max

Set the maximum allowed motor position, -1000000 to 1000000.

?! max 10000, AOK.

3.1.9 min

Set the minimum allowed motor position, -1000000 to 1000000.

?! min -10000, AOK.

3.1.10 delay

Set the approximate delay time between steps (for moves, gotos, etc.), 10-65535 (ms). This is not a guaranteed timescale, the user should test for themselves if precise timing is required.

?! delay 10, AOK.

3.1.11 feedback

Instructs the controller to turn the feedback off (0) or on (1).

?! feedback 0, AOK.

3.1.12 mode

Set the operation mode, as summarized in Table 3. The destination mode should be fully configured before transferring.

?! mode 1, AOK.

Mode	Behaviour
0	motor disabled
1	manual
2	external
3	triggered goto
4	timed goto
5	triggered move
6	timed move
7	state-based rotation
8	triggered shutter
9	state-based shutter

Table 3: Pre-defined operation modes.

3.1.13 steps

Set the number of steps, -2000000 to 2000000. Used to configure various predefined operation modes (shutter and move modes).

?! steps 87, AOK.

3.1.14 triggers

Set the number of triggers (rising edges on external trigger input) before action is to be taken. The allowed range is 1-65535. This parameter is used for various predefined operation modes.

?! triggers 10, AOK.

3.1.15 goal

Set the destination for goto-style predefined operation modes.

?! goal -145, AOK.

3.1.16 hours

Set the current time (hours, 0-23). Used for time-dependent operation modes.

?! hours 12, AOK.

3.1.17 minutes

Set the current time (minutes, 0-59). Used for time-dependent operation modes.

?! minutes 30, AOK.

3.1.18 seconds

Set the current time (seconds, 0-59). Used for time-dependent operation modes.

?! seconds 30, AOK.

3.1.19 dohours

Set the action time (hours, 0-23). Used for time-dependent operation modes.

?! dohours 12, AOK.

3.1.20 dominutes

Set the action time (minutes, 0-59). Used for time-dependent operation modes.

?! dominutes 30, AOK.

3.1.21 doseconds

Set the action time (seconds, 0-59). Used for time-dependent operation modes.

?! doseconds 30, AOK.

3.1.22 repeat

Set the repetition behaviour for various operation modes, such as a triggered move. If turned off (0), after the move is completed, the operation mode will be set to manual. If turned on (1), the mode will remain unchanged and the desired operation will repeat again once the necessary conditions are met (triggers received or time-of-day match etc.).

?! repeat 1, AOK.

3.1.23 rotate

Used for rotate mode. Instructs the motor which way to rotate for a logic high on the external trigger input.

?! rotate 1, AOK.

3.1.24 behave

Set the rotation behaviour in the state-rotate mode when the external trigger bnc input is logic low. If 0, the motor does nothing for a logic-low input. If 1, the motor anti-rotates in the direction opposite of that determined by the "rotate" setting.

?! behave 0, AOK.

3.2 Configuring Predefined Operation Modes via USB

Configuring the predefined operation modes over USB, rather than the GUI, consists of completing a sequence of exchanges with the controller. It is recommended that when configuring a mode, the first few commands disable the motor and turn the feedback off. This will ensure that the subsequent commands, used to configure the mode, can be carried out without the motor undergoing uncontrolled motion. If removing the motor drive during configuration is a concern, the manual mode of operation can be assumed instead. Examples of configuring each mode using the serial port follow. Some settings will not need to be configured separately for each mode as shown, depending on the values before configuration.

3.2.1 Configuring "Off" Mode

?! feedback 0, AOK.	disable feedback
?! mode 0, AOK.	disable motor

3.2.2 Configuring "Manual" Mode

?! feedback 0, AOK.	disable feedback
?! mode 0, AOK.	disable motor
?! delay x, AOK.	set delay between steps
?! mode 1, AOK.	set mode to manual
?! feedback 1, AOK.	turn feedback on (if desired)

3.2.3 Configuring "External" Mode

?! feedback 0, AOK.	disable feedback
?! mode 0, AOK.	disable motor
?! mode 2, AOK.	set mode to external
?! feedback 1, AOK.	turn feedback on (if desired)

3.2.4 Configuring "Triggered Goto" Mode

?! feedback 0, AOK.	disable feedback
?! mode 0, AOK.	disable motor
?! delay a, AOK.	set delay between steps, ms (a)
?! triggers b, AOK.	set number of triggers before goto completed (b)
?! goal c, AOK.	set destination position (c)
?! mode 3, AOK.	set mode to triggered goto
?! feedback 1, AOK.	turn feedback on (if desired)

3.2.5 Configuring "Timed Goto" Mode

?! feedback 0, AOK.	disable feedback
?! mode 0, AOK.	disable motor
?! delay a, AOK.	set delay between steps, ms (a)
?! hours b, AOK.	set current time, hours (b)
?! minutes c, AOK.	set current time, minutes (c)
?! seconds d, AOK.	set current time, seconds (d)
?! dohours e, AOK.	set action time, hours (e)
?! dominutes f, AOK.	set action time, minutes (f)
?! doseconds g, AOK.	set action time, seconds (g)
?! goal h, AOK.	set destination position (h)
?! mode 4, AOK.	set mode to timed goto
?! feedback 1, AOK.	turn feedback on (if desired)

3.2.6 Configuring "Triggered Move" Mode

?! feedback 0, AOK.	disable feedback
?! mode 0, AOK.	disable motor
?! delay a, AOK.	set delay between steps, ms (a)
?! triggers b, AOK.	set number of triggers before goto completed (b)
?! steps c, AOK.	set number of steps to take (c)
?! mode 5, AOK.	set mode to triggered move
?! feedback 1, AOK.	turn feedback on (if desired)

3.2.7 Configuring "Timed Move" Mode

?! feedback 0, AOK.	disable feedback
?! mode 0, AOK.	disable motor
?! delay a, AOK.	set delay between steps, ms (a)
?! hours b, AOK.	set current time, hours (b)
?! minutes c, AOK.	set current time, minutes (c)
?! seconds d, AOK.	set current time, seconds (d)
?! dohours e, AOK.	set action time, hours (e)
?! dominutes f, AOK.	set action time, minutes (f)
?! doseconds g, AOK.	set action time, seconds (g)
?! steps h, AOK.	set number of steps to take (h)
?! mode 6, AOK.	set mode to timed move
?! feedback 1, AOK.	turn feedback on (if desired)

3.2.8 Configuring "State Rotate" Mode

?! feedback 0, AOK.	disable feedback
?! mode 0, AOK.	disable motor
?! delay a, AOK.	set delay between steps, ms (a)
?! rotate b, AOK.	set direction to rotate on high input (b)
?! behave c, AOK.	set behaviour on low input (c)
?! mode 7, AOK.	set mode to state rotate

3.2.9 Configuring "Triggered Shutter" Mode

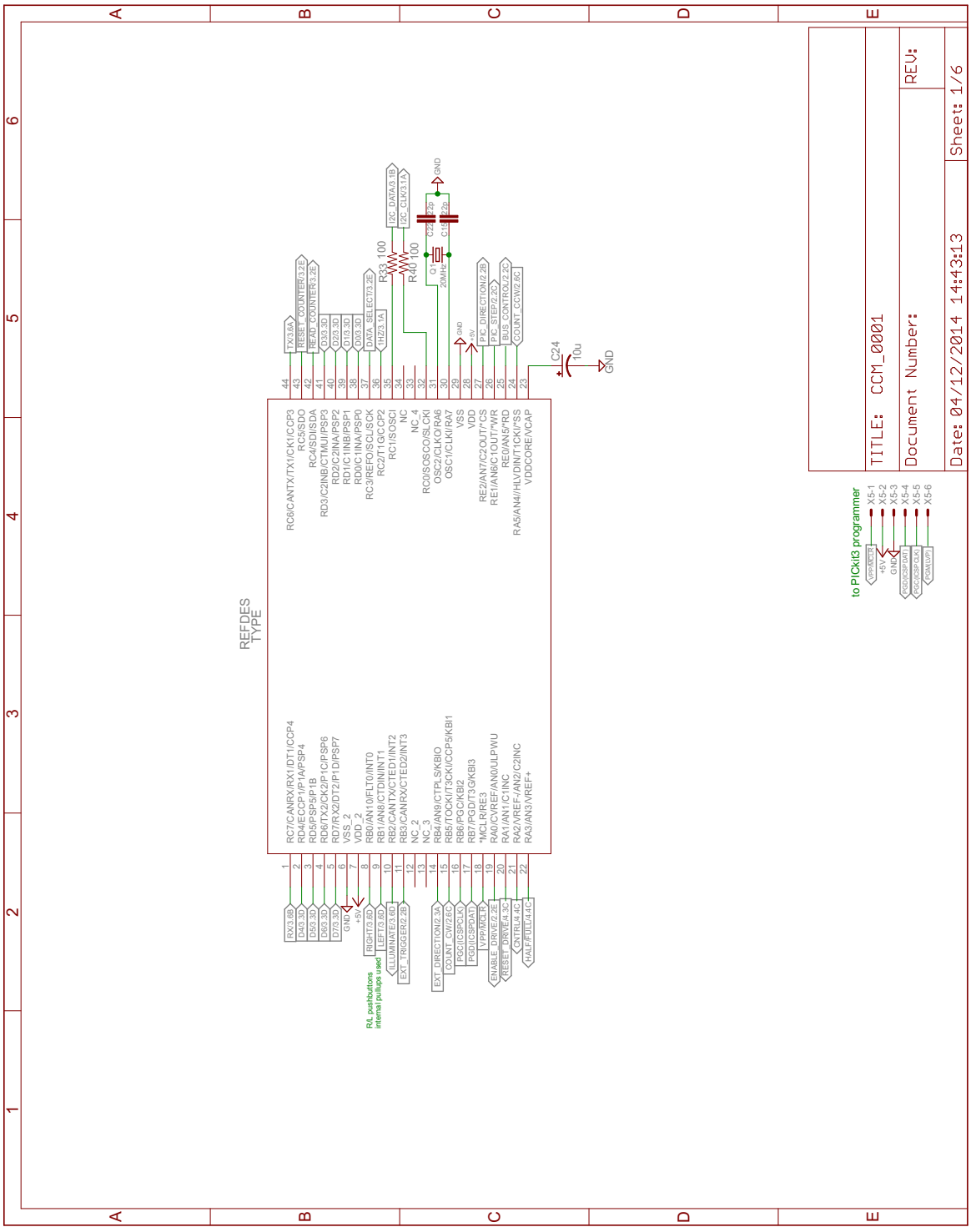
?! feedback 0, AOK.	disable feedback
?! mode 0, AOK.	disable motor
?! delay a, AOK.	set delay between steps, ms (a)
?! triggers b, AOK.	set trigger number for opening/closing (b)
?! steps c, AOK.	set number of steps to take (c)
?! mode 8, AOK.	set mode to triggered shutter
?! feedback 1, AOK.	turn feedback on (if desired)

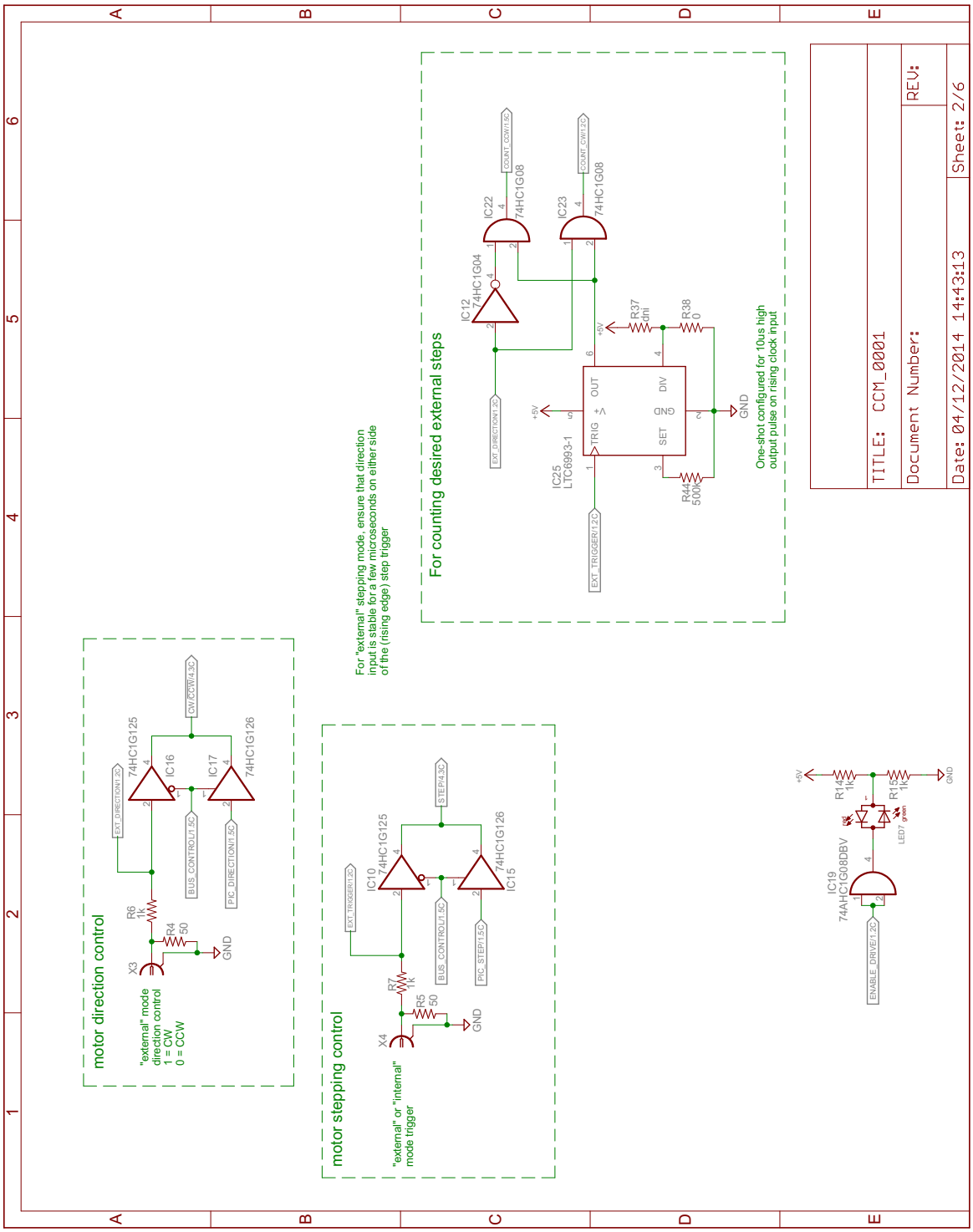
3.2.10 Configuring "State Shutter" Mode

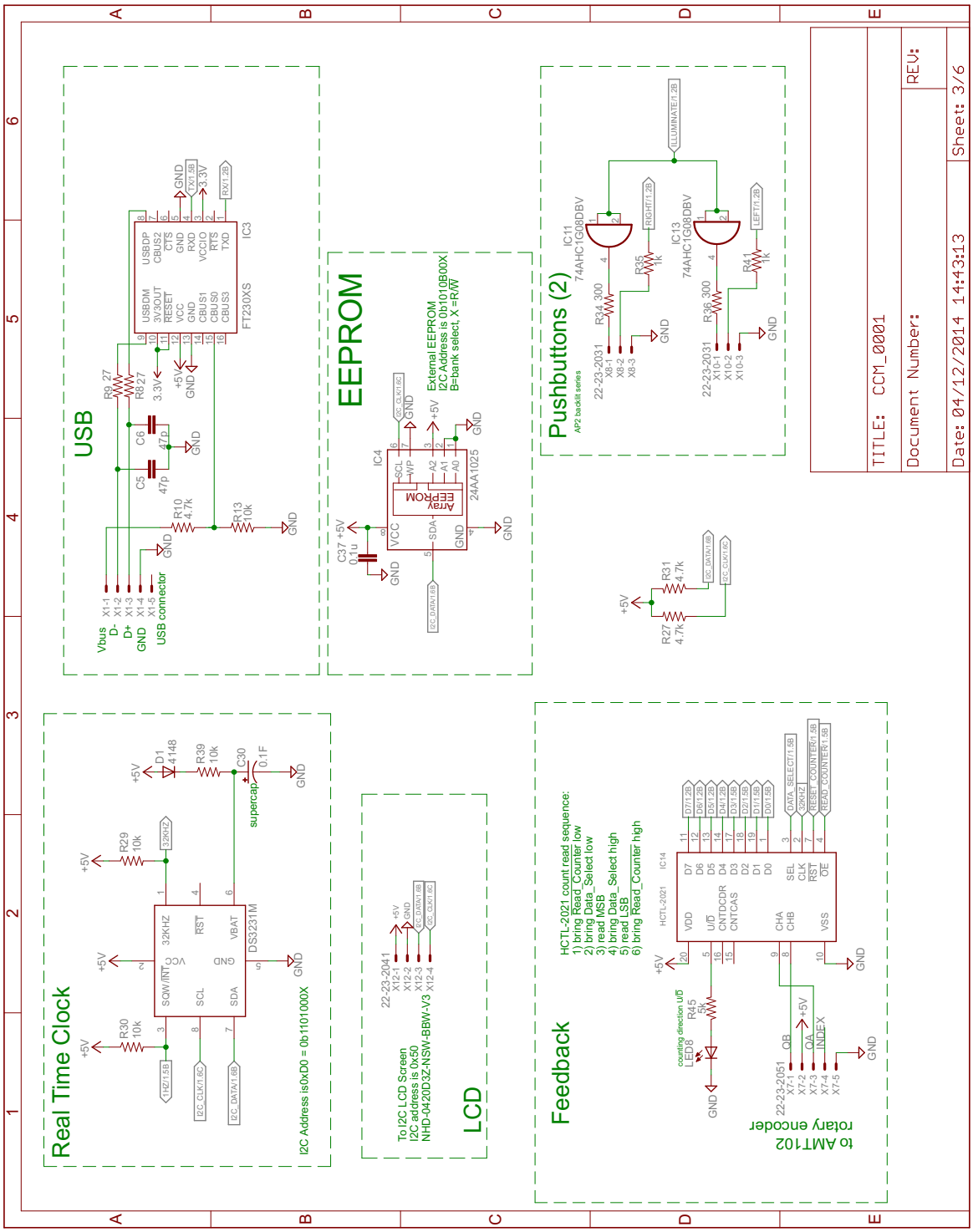
?! feedback 0, AOK.	disable feedback
?! mode 0, AOK.	disable motor
?! delay a, AOK.	set delay between steps, ms (a)
?! steps b, AOK.	set number of steps to take (b)
?! mode 9, AOK.	set mode to state shutter
?! feedback 1, AOK.	turn feedback on (if desired)

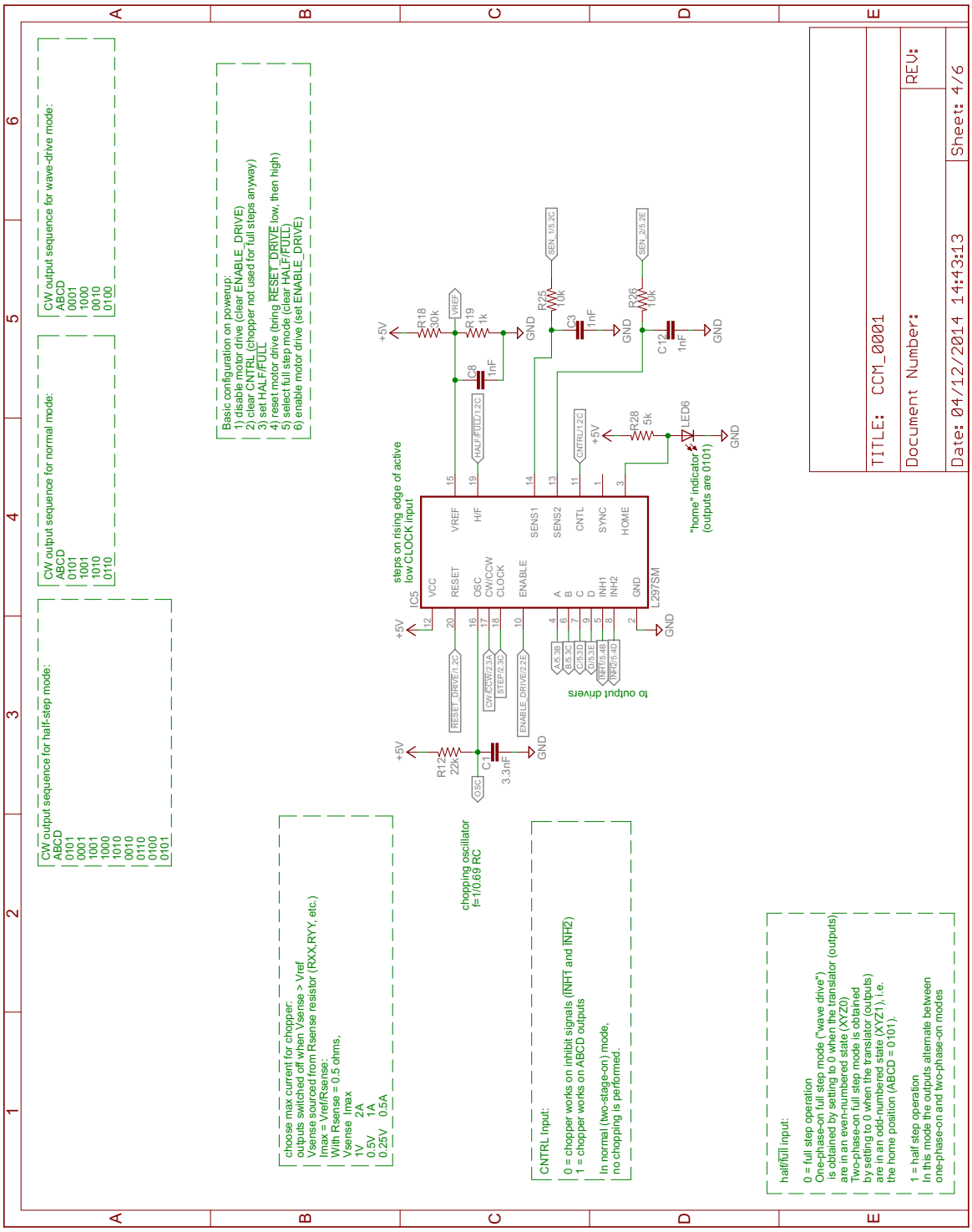
4 Prototype Builds

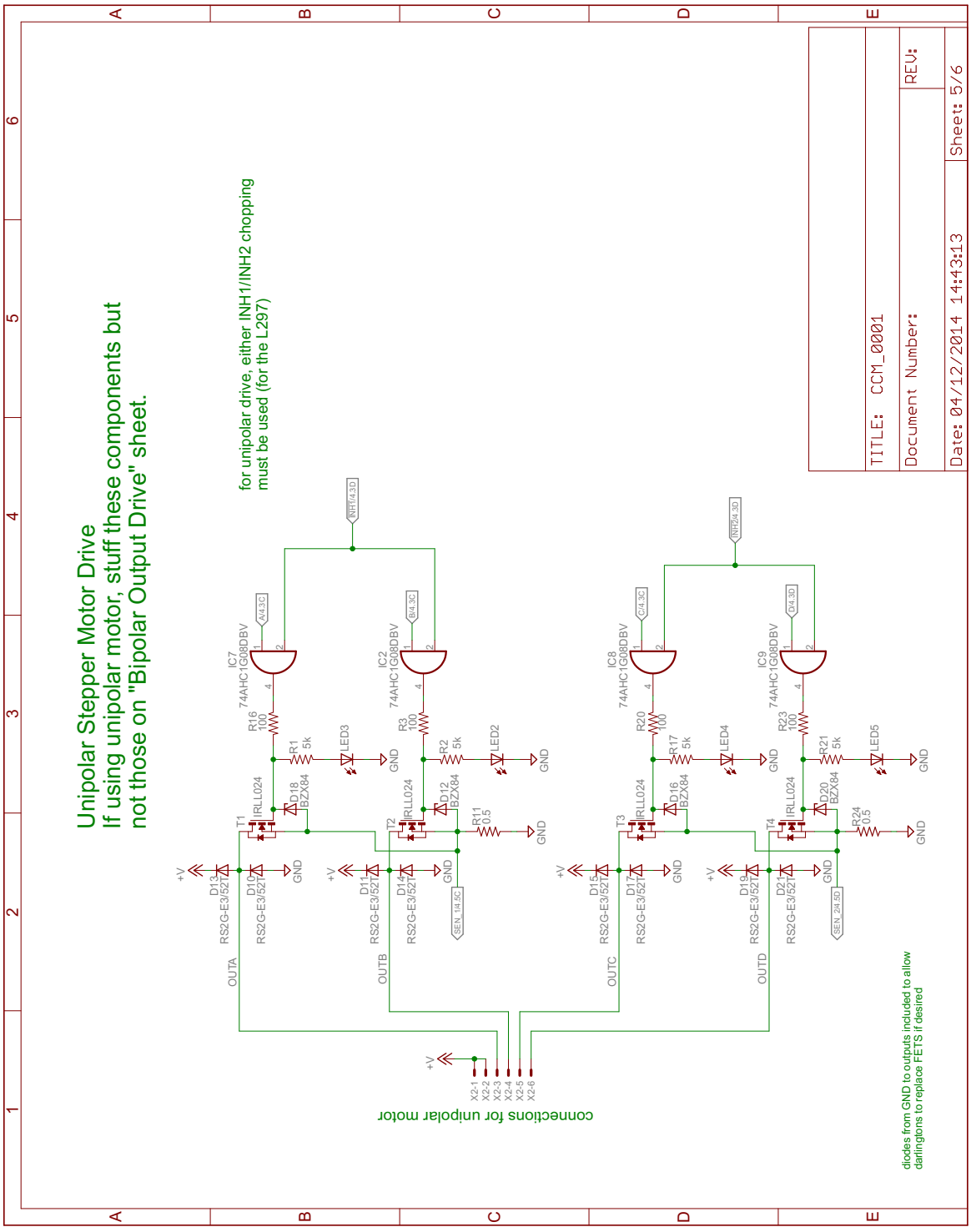
5 Circuit Schematics

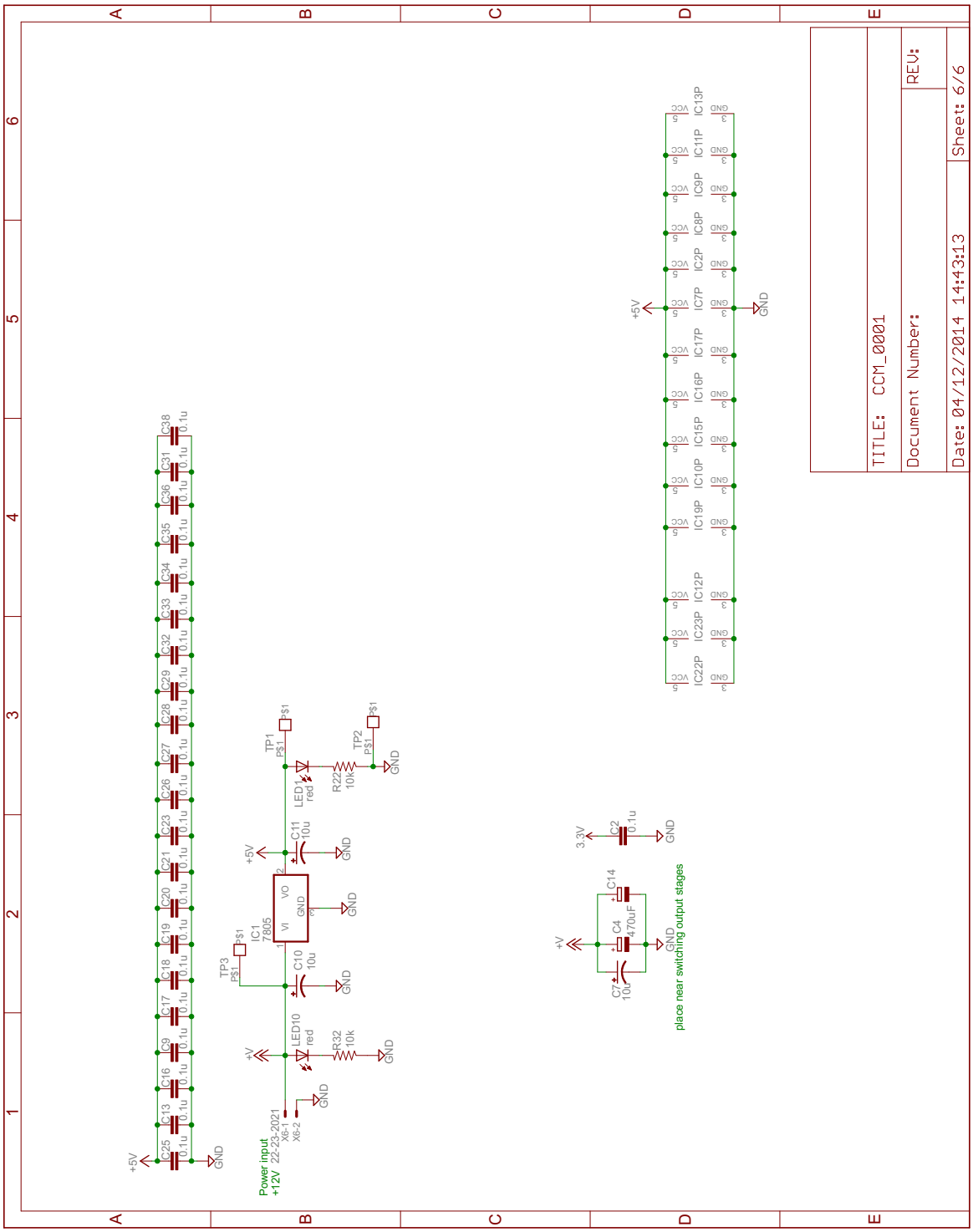

REFDES
TYPE
RC7/CANRX/RX1/DT1/CCP4
RD4/ECCP1/P1A/PSP4
RD5/PSP5/P1B
RD6/TX2/CK2/P1C/PSP6
RD7/RX2/DT2/P1D/PSP7
VSS_2
VDD_2
RB0/AN10/FLT0/INT0
RB1/AN8/CTDIN/INT1
RB2/CANTX/CTED1/INT2
RB3/CANRX/CTED2/INT3
NC_2
NC_3
RB4/AN9/CTPLS/KBI0
RB5/T0CKI/T3CKI/ICCP5/KBI1
RB6/PGC/KBI2
RB7/PGD/T3G/KBI3
MCLR/VPP/RE3
RA0/CVREF/AN0/ULPWU
RA1/AN1/C1INC
RA2/VREF-/AN2/C2INC
RA3/AN3/VREF+
RC6/CANTX/TX1/CK1/CCP3
RC5/SDO
RC4/SDI/SDA
RD3/C2INB/CTMUI/PSP3
RD2/C2INA/PSP2
RD1/C1INB/PSP1
RD0/C1INA/PSP0
RC3/REFO/SCL/SCK
RC2/T1G/CCP2
RC1/SOSCI
NC_4
RC0/SOSCO/SLCKI
OSC2/CLKO/RA6
OSC1/CLKI/RA7
VSS
VDD
RE2/AN7/C2OUT/*CS
RE1/AN6/C1OUT/*WR
RE0/AN5/*RD
RA5/AN4//HLVDIN/T1CKI/*SS
VDDCORE/VCAP
TX/3.6A
RESET_COUNTER/3.2E
READ_COUNTER/3.2E
D3/3.3D
D2/3.3D
D1/3.3D
D0/3.3D
DATA_SELECT/3.2E
HIZ/3.1A
R33 100
R40 100
I2C_DATA/3.1B
I2C_CLK/3.1A
C1
20MHz
C22 22p
C15 22p
GND
+5V
DIR_DIRECTION/2.2B
PIC_STEP/2.2C
BUS_CONTROL/2.2C
COUNT_CCW/2.6C
C24
10u
GND
to PICkit3 programmer
X5-1
X5-2
X5-3
X5-4
X5-5
X5-6
RL: pushbuttons
internal pullups used
RIGHT/3.6D
LEFT/3.6D
ILLUMINATE/3.6D
EXT_TRIGGER/2.2B
EXT_DIRECTION/2.3A
COUNT_CW/2.2C
PGC/ICSPCLK
PGD/ICSPDAT
VPP/MCLR/3.7B
ENABLE_DRIVE/2.2E
RESET_DRIVE/4.4C
CONTROL/4.4C
HALF-FUDE/4.4C
TITLE: CCM_0001
Document Number:
REV:
Date: 04/12/2014 14:43:13
Sheet: 1/6


motor direction control
"external" mode
direction control
1 = CW
0 = CCW
motor stepping control
"external" or "internal"
mode trigger
For "external" stepping mode, ensure that direction
input is stable for a few microseconds on either side
of the (rising edge) step trigger
For counting desired external steps
One-shot configured for 10us high
output pulse on rising clock input
TITLE: CCM_0001
Document Number:
REV:
Date: 04/12/2014 14:43:13
Sheet: 2/6


USB
EEPROM
Pushbuttons (2)
Real Time Clock
LCD
Feedback
TITLE: CCM_0001
Document Number:
REV:
Date: 04/12/2014 14:43:13
Sheet: 3/6
USB connector
External EEPROM
I2C Address is 0b1010B00X
B=bank select, X =R/W
I2C Address is0x50 = 0b1101000X
To I2C LCD Screen
I2C address is 0x50
NHD-0420D3Z-NSW-BBW-V3
HCTL-2021 count read sequence:
1) bring Read_Counter low
2) bring Data_Select low
3) read MSB
4) bring Data_Select high
5) read LSB
6) bring Read_Counter high
to AMT102
rotary encoder


CW output sequence for half-step mode:
ABCD
0101
0001
0011
0010
0110
0100
0101
CW output sequence for normal mode:
ABCD
0101
1001
1010
0110
CW output sequence for wave-drive mode:
ABCD
0001
1000
0010
0100
Basic configuration on powerup:
1) disable motor drive (clear ENABLE_DRIVE)
2) clear CNTRL (chopper not used for full steps anyway)
3) set HALF/FULL
4) reset motor drive (bring RESET_DRIVE low, then high)
5) select full step mode (clear HALF/FULL)
6) enable motor drive (set ENABLE_DRIVE)
choose max current for chopper:
outputs switched off when Vsense > Vref
Vsense sourced from Rsense resistor (RXX,RYY, etc.)
Imax = Vref/Rsense:
With Rsense = 0.5 ohms,
Vsense Imax
1V 2A
0.5V 1A
0.25V 0.5A
chopping oscillator
f=1/0.69 RC
CNTRL Input:
0 = chopper works on inhibit signals (INH1 and INH2)
1 = chopper works on ABCD outputs
In normal (two-stage-on) mode,
no chopping is performed.
half/full input:
0 = full step operation
One-phase-on full step mode ("wave drive")
is obtained by setting to 0 when the translator (outputs)
are in an even-numbered state (XYZ0).
Two-phase-on full step mode is obtained
by setting to 0 when the translator (outputs)
are in an odd-numbered state (XYZ1), i.e.
the home position (ABCD = 0101).
1 = half step operation
In this mode the outputs alternate between
one-phase-on and two-phase-on modes
steps on rising edge of active
low CLOCK input
IC5
L297SM
VCC
RESET
OSC
CW/CCW
CLOCK
ENABLE
VREF
H/F
SENS1
SENS2
CNTL
SYNC
HOME
A
B
C
D
INH1
INH2
GND
+5V
R18 30k
R19 1k
R25 10k
R26 10k
R28 5k
R12 22k
C8 1nF
C3 1nF
C12 1nF
C 3.3nF
LED6
"home" indicator
(outputs are 0101)
to output drivers
TITLE: CCM_0001
Document Number:
REV:
Date: 04/12/2014 14:43:13
Sheet: 4/6


Unipolar Stepper Motor Drive
If using unipolar motor, stuff these components but not those on "Bipolar Output Drive" sheet.
for unipolar drive, either INH1/INH2 chopping must be used (for the L297)
connections for unipolar motor
diodes from GND to outputs included to allow darlingtons to replace FETS if desired
TITLE: CCM_0001
Document Number:
REV:
Date: 04/12/2014 14:43:13
Sheet: 5/6


TITLE: CCM_0001
Document Number:
REV:
Date: 04/12/2014 14:43:13
Sheet: 6/6
Power input
+12V
place near switching output stages