INCREMENTAL ENCODER SIGNAL ANALYZER

A Design Project Report

Presented to the School of Electrical and Computer Engineering of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering, Electrical and Computer Engineering

Submitted by
Pratik Panchal
M.Eng Field Advisor: Bruce Land
Degree Date: 26th May, 2012

Abstract

Master of Engineering Program

School of Electrical and Computer Engineering

Cornell University

Design Project Report

Project Title: **Incremental Encoder Signal Analyzer**

Author: **Pratik Panchal**
pp423@cornell.edu

Abstract: 'Incremental Encoder Signal Analyzer' is a PIC18F4685 based device which employs various algorithms to analyze the received signals from a rotary and linear incremental encoder, either online or offline, and can give output in terms of various parameters such as follows:

- Digital read outs - with and without direction compensation, which can also be used to detect presence of any initial and/or sustained jerks in the motion.
- Duty jitter & phase jitter and missed pulse detection – to validate the precision of the encoder output and detect any missing pulses from any of the channels.
- Phase evaluation – to validate the identity of the channels.
- Other derived computations and hardware checks which help in understanding the total healthiness of the device.

It is integrated with a graphical touch screen LCD of 128x64 pixels as a HMI. The device uses the powerful architecture of PIC18 family, and hence several further extensions are anticipated in terms of computational features and communication.

1. Overview

'Incremental Encoder Signal Analyzer' is a PIC based device which has various algorithms to analyze the received signals from a rotary and linear incremental encoder, either online or offline, and can give output in terms of various parameters, which can help in understanding the total healthiness of the device.

A missing pulse in a marker-dereferenced measuring system can lead to a major mis-calculation and even damage to mechanical drive system.

2. Introduction

A motor encoder is a feedback device which can monitor and measure the motions on the motor and is directly connected to the numerical system which controls the motor. Since the device lies in the feedback path, it plays a very important role in keeping the system up and going.

An encoder can face several breakdown causes which are difficult to identify and analyze without a proper methodology. The usual maintenance methodology of using oscilloscope based testing and verification is inherently ambiguous, and leads to incorrect diagnostic decisions and increased breakdown time of the machine. Commercially, there are very few devices available which can perform this function, but they are either too costly or under-featured. This project demonstrates various possible algorithms for such a possible device which is simple to use, quick, full-featured and cost effective.

The project employs a Microchip PIC18F4685 8-bit microcontroller which has a large code space and wide peripheral options. A 128x64 graphical LCD is used to display the data and a resistive touch screen overlay is used as an input device. A touchscreen introduces a versatility and ease of use in the device.

3. Encoder

The incremental encoders can either be rotary or linear with either, optical or magnetic sensing.

The signals of encoder are delivered in 6 channels. Channel A and channel B are phase shifted by 90°, with channel A leading. Channel Z is the 'marker' channel which gives a pulse per revolution of the encoder shaft in case of rotary encoder and a pulse per unit linear distance in case of linear encoder. The encoder ppr (pulse per revolution) is the number of pulses a channel A (and channel B) gives on one encoder shaft revolution. The other three channels, A', B’ and Z’ are complimentary digital states of channel A, B and Z respectively. The states of these channels are depicted in figure 1.

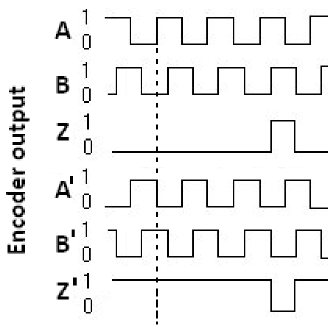


Figure 1: Encoder output

For the purpose of this project, I used the Siemens rotary ‘PGcoder’ of type EN-B01-1000 of 1000 ppr. The voltage rating of this device is 5VDC.

3.1 Possible Failure Modes of the Encoder:

An encoder can encounter several failure / malfunction modes during its lifetime. Some of them which are commonly seen are discussed below. These are some of the known faced issues and there can be more additions to this list.

- Internal Power Supply Failure: A fault of this kind renders the device useless. It can either be an open-circuit or short-circuit and can affect the preceding system stage accordingly. A malfunctioning of the power supply section can also lead to presence of voltage spikes in channel output.
- Channel failure: One of the most common problems seen is a failure of a specific channel. Since most of the NC systems use only a set of channels, the complementary set can be used in case the main channel fails.
- Phase and duty jitters: The channel output gets ‘imbalanced’ due to mechanical mis-alignment of the internal discs. It is necessary to identify such kind of an abnormality to embark upon the decision to continue or replace the device. The level of permissible jitter is dependent on the measuring system used.
- Loss of channel identification: A device which has suffered a loss of its output channel identification and specifications renders useless. It is necessary to establish these parameters to regain the possible use of the device.

4. Hardware Development

4.1 Hardware Development System: The project is built on a EasyPic v7 – a PIC development board from MikroElectronika (www.mikroe.com). It is an extremely versatile board which exploits most of the features of the microcontroller. It supports all of the DIP packages which enable parallel development of peripheral microcontroller (extension) system. It has an onboard programmer and debugger. However, I have used the PicKit3 programmer from Microchip for the development due to its complete compatibility with MPLAB. The on-board programmer is compatible with the software development system provided by MikroElektronika. It also has USART and USB communication hardware which has proved extremely useful for easy debugging. Since I had to integrate the GLCD, the onboard GLCD driver had been a quick start for the development. All I/Os have selectable pull-ups and pull-downs with LEDs. The board can be powered up by an external power supply or through the USB port. The snapshot of the development system is shown in appendix III. More information on this hardware can be found on product page:

http://www.mikroe.com/eng/products/view/757/easypic-v7-development-system/

4.2 Microcontroller

The main criteria for choosing PIC18F4685 was the available program code space of 96K bytes and 3328 bytes of internal RAM making it very comfortable to write code without worrying of running out of space. It also has on-board I2C hardware which is (extended) to use with resistive touchscreen driver AR1020 from Microchip. The 3 timers and the CCP help in recording the events precisely and without peripheral shortage. The pin diagram is shown in appendix 1.

The following table 1 shows the assigned functions to the pins and the hardware connections.

Pin #	Description	Assigned Function
1	MCLR/Vpp/RE3	Device Reset pin
2	RA0/AN0/CVref	Touchscreen X-read
3	RA1/AN1	Touchscreen Y-read
4	RA2/AN2/Vref-	<empty>
5	RA3/AN3/Vref+	<empty>
6	RA4/T0CK1	Encoder internal fault and level read out
7	RA5/AN4/SS/HLVDIN	Ch. Z’
8	RE0/RD/AN5	Ch. A
9	RE1/WR/AN6/C1OUT	Ch. B
10	RE2/CS/AN7/C2OUT	Ch. Z
11	VDD	+5V
12	VSS	Gnd
13	OSC1/CLK1/RA7	20Mhz Oscillator

14	OSC2/CLK0/RA6	20Mhz Oscillator
15	RC0/T1OSO/T13CKI	Touchscreen Drive A
16	RC1/T1OSI	Touchscreen Drive B
17	RC2/CCP1	GLCD backlight PWM
18	RC3/SCK/SCL	Reserved – I2C for AR1020 interface
19	RD0/PSP0/C1IN+	GLCD data line D0
20	RD1/PSP1/C1IN-	GLCD data line D1
21	RD2/PSP2/C2IN+	GLCD data line D2
22	RD3/PSP3/C2IN-	GLCD data line D3
23	RC4/SDI/SDA	Reserved – I2C for AR1020 interface
24	RC5/SDO	Reserved – I2C for AR1020 interface
25	RC6/TX/CK	Serial communication
26	RC7/RX/DT	Serial communication
27	RD4/PSP4/ECCP1/P1A	GLCD data line D4
26	RD5/PSP5/P1B	GLCD data line D5
29	RD6/PSP6/P1C	GLCD data line D6
30	RD7/PSP7/P1D	GLCD data line D7
31	VSS	Gnd
32	VDD	+5V
33	RB0/INT0/FLT0/AN10	GLCD Chip select 1
34	RB1/INT1/AN8	GLCD Chip select 2
35	RB2/INT2/CANTX	GLCD RS
36	RB3/CANRX	GLCD RW
37	RB4/KBI0/AN9	GLCD E
38	RB5/KBI1/PGM	GLCD RST
39	RB6/KBI2/PGC	Ch. A’
40	RB7/KBI3/PGD	Ch. B’

Table 1: Pin assignments

4.3 Signal Drive and Acquisition:

To isolate any adverse voltage peaks of encoder, optical isolators are used between the encoder and microcontroller pins. Although the configuration complements the encoder signals reaching the microcontroller pins, it doesn’t affect the analysis due to the inherent symmetry between the channels. The maximum bandwidth gets limited by the type of optical isolator used. For this, I’ve used 4N35 general purpose optical isolators for development purpose, which I intend to replace with faster ones (extension). 4N35 has a total turn-on and turn–off delay time of about 20µs, thus forcing the signal to be at least more than 30µs on and off times. Figure 2 shows the schematic.

As previously stated, one of the common failure modes that encoders encounter is internal fault in power section which leads to an open circuit or short circuit internally. An analog NPN-

PNP lockout circuit has been employed to hold on the output voltage to the encoder under normal conditions. A fault on encoder power lines will result into cascade trip of NPN-PNP transistor and cut-off of supply voltage to the encoder. The same line is also fed to RA4 pin to monitor the voltage level on the encoder supply line apart from the visual LED indicators.

The biasing resistors R_{14} and R_{16} have been calculated to 6k as per following equation.

$$R_B = \frac{V_S \bullet HFE}{1.25 \bullet I_L}$$

The schematic is given in figure 3.

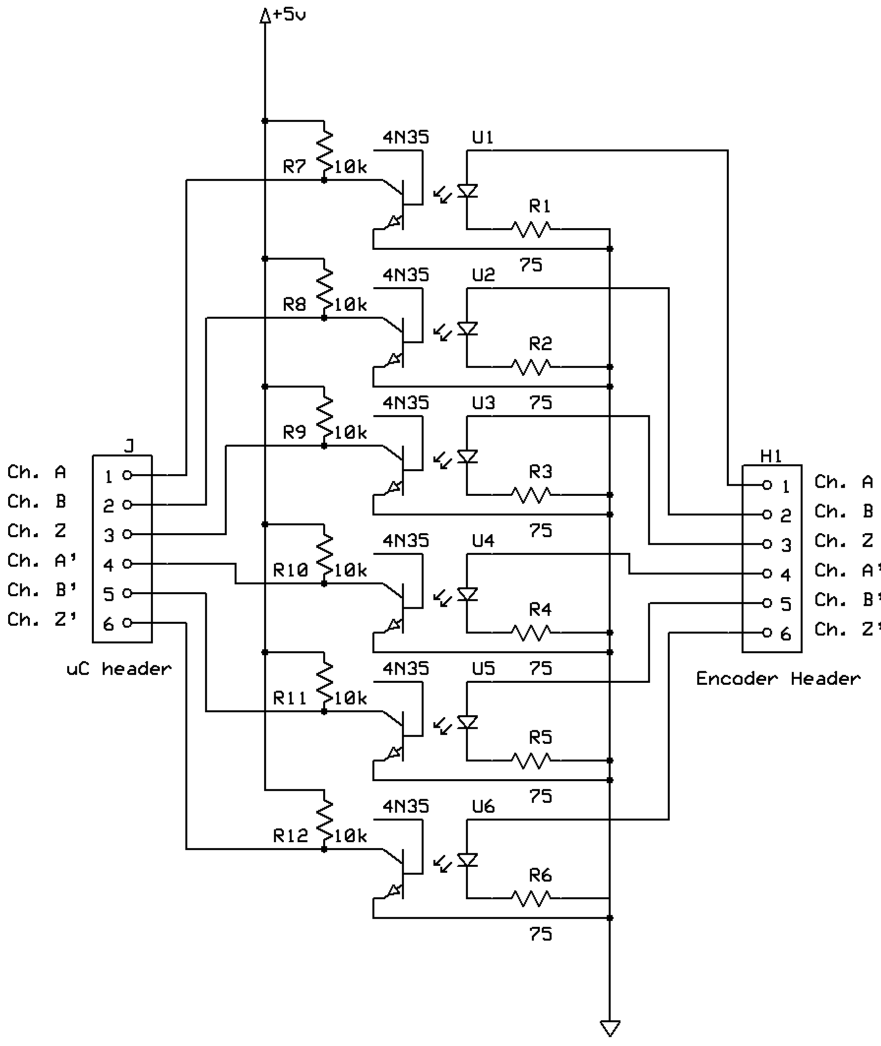


Figure 2: Signal Acquisition from Encoder

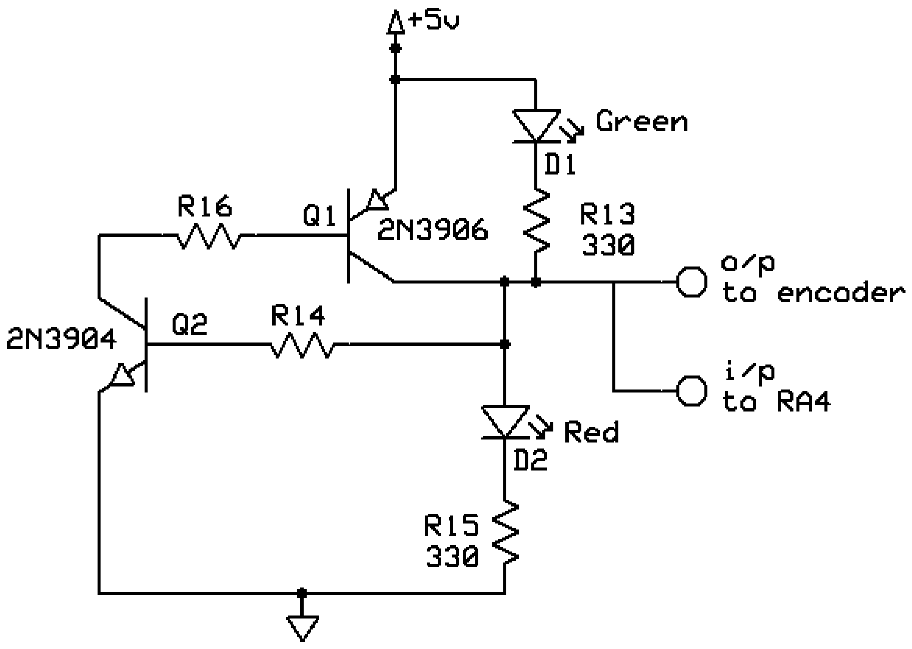
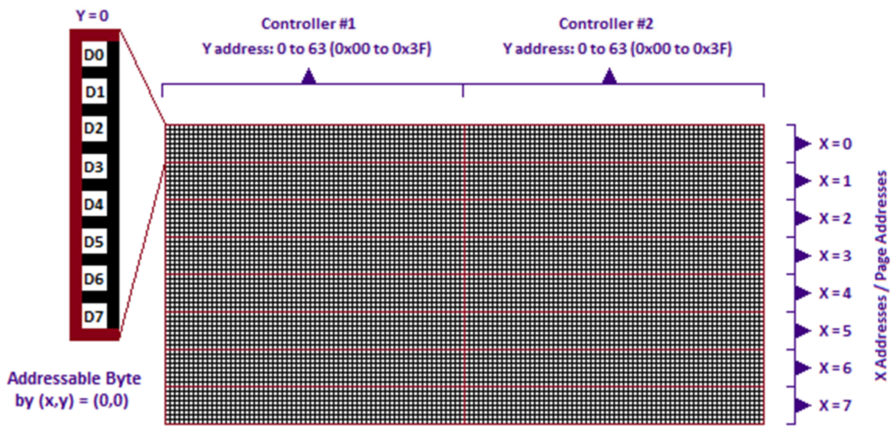


Figure 3: Fault Isolation Circuit Schematic

4.4 HMI

I used a 128x64 pixel GLCD as the display device and a 4-wire resistive overlay to make it a touch screen. The only other hard button is the reset push button of the microcontroller.

GLCD doesn't have inbuilt ASCII converter/generator; rather, it is possible to control state of a single pixel and hence it is complex to send data to it. I have used NT7108C controller based GLCD (which is quite similar to KS0108B module). A 128X64 GLCD has 128 columns and 64 rows as shown in the figure 4.



Y = 0
D0
D1
D2
D3
D4
D5
D6
D7
Addressable Byte by (x,y) = (0,0)
Controller #1
Y address: 0 to 63 (0x00 to 0x3F)
Controller #2
Y address: 0 to 63 (0x00 to 0x3F)
X = 0
X = 1
X = 2
X = 3
X = 4
X = 5
X = 6
X = 7
X Addresses / Page Addresses

Figure 4: 128x64 GLCD pixel layout and addressing nomenclature

1. The 128 columns are divided into sets of two 64 columns; each controlled by independent NT7108C controllers. The columns are addressed by Y address. Thus, for both the sets (both the controllers), Y varies from 0 to 63 (0x00 to 0x3F).

2. The 64 rows are divided into 8 pages of 8 rows (bits) each. Each page can be addressed by X address (or a page address) which varies from 0 to 7 (0x00 to 0x07).

3. Each page comprises of 8 bits x 128 columns (Y address) = 1024 pixels. A byte is a row of a page with its LSB (i.e. D0) on top and MSB (i.e. D7) at bottom. Writing digital ‘1’ to a pixel, turns it on and darkens it. Thus, if a 8 bit data is sent, say 0xFF to (x,y) = (0,0), all the 8 pixels of 1st column of 1st page turns on.

To control the other half – the right side of the display, it is necessary to activate the NT7108C controller of that section and repeat the procedure. There are hardware pins to select a specific controller. The pin layout is comparable to the 16x2 LCD module.

Pin 1 & 2: These are chip selection lines. A low on CS1 selects the NT7108C of right half of the screen and low on CS2 selects the other half. If you want to write simultaneously on both the halves, you can select both.

Pin 3 & 4: These are power supply lines for the module. Works on +5VDC.

Pin 5: This is GLCD contrast control and is connected to the viper of a 10kohm pot. One end of the pot goes to gnd and the other goes to pin 18 (Vee) of the GLCD.

Pin 6: RS (D/I) – This is data / instruction selection pin. A high on this pin indicates to the NT7108C that a data byte is being written and a low indicates that an instruction is being fed.

Pin 7: RW – Read/Write pin – A high on this pin enables reading from NT7108C and low enables writing to it.

Pin 8: E – Enable pin – A high on this pin enables the GLCD and a high to low transition latches the data (read or write).

Pin 9 to 16: Data lines from LSB, D0 to MSB, D7.

Pin 17: RST – Reset line – A low on this resets the module. While working with GLCD, this must be held high.

Pin 18: Vee – Negative voltage output pin – *refer description of pin 5, above.*

Pin 19 & 20: These are back light LED lines. +5V with proper polarity on these lines lights up the back light. Although GLCD internally consists of a limiting resistor and it is possible to directly feed in +5V to these lines, I added a small resistor of 20 ohms in between. To control the back light, a PWM signal via a switching transistor BC846 is given.

The schematic is shown in figure 5.

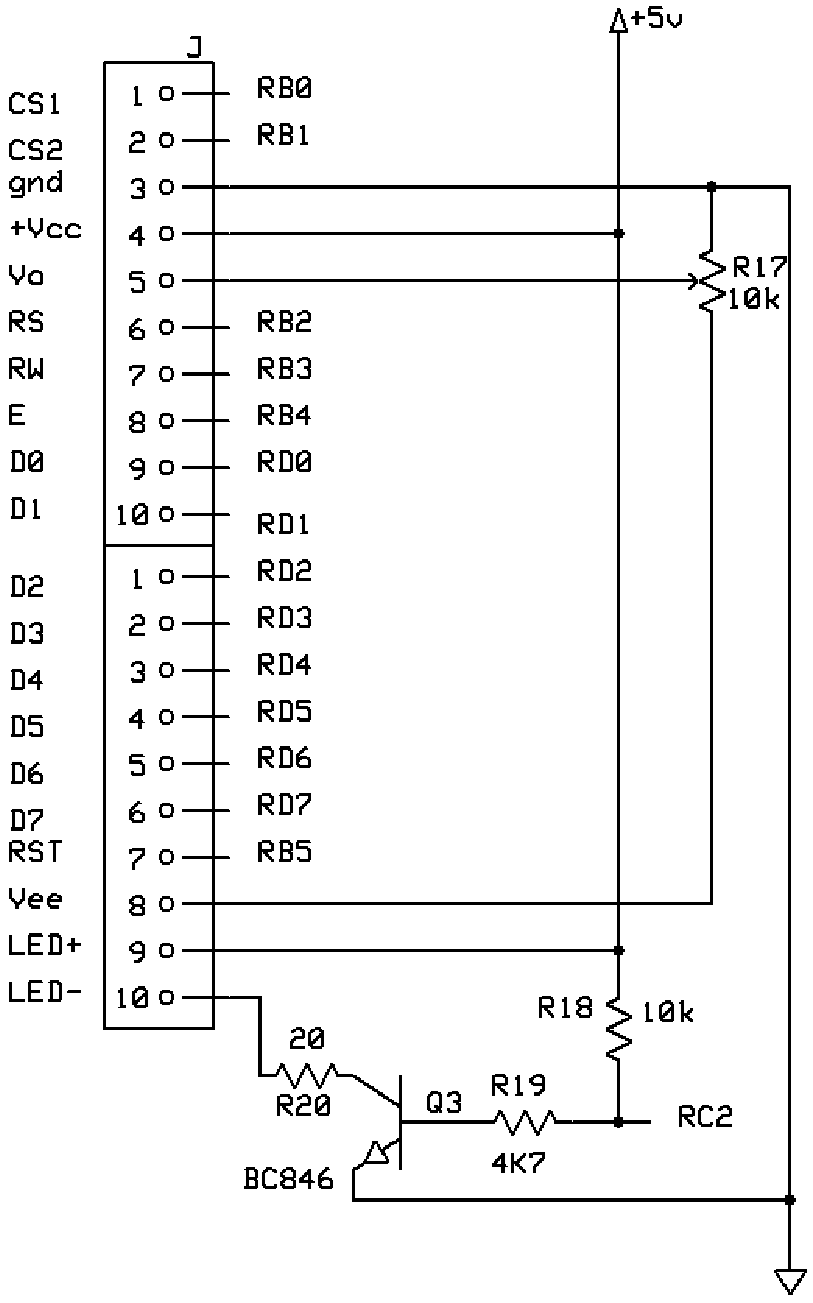


Figure 5: GLCD hardware connections with microcontroller

4.5 Touch screen overlay:

A 4 wire touch screen overlay has two resistive layers assembled perpendicular to each other and separated electrically by spacer dots. A light-pressurized touch will force the two layers to come in contact which can be read as an analog voltage by applying +5V and gnd to one layer and reading from another and vice-versa to obtain (x.y) touch points which are unique to the touch point on the screen. When Drive A (connected to RC0) is made high and Drive B (connected to RC1) is made low, a horizontal analog voltage representing x coordinate is available on AN0 pin to read. Similarly, when Drive A is made low and Drive B is made high, a vertical analog voltage representing y coordinate is available on AN1 pin to read. The necessary hardware has already been provided on the development board by MikroElektronika, the schematic of which is shown in figure 6.

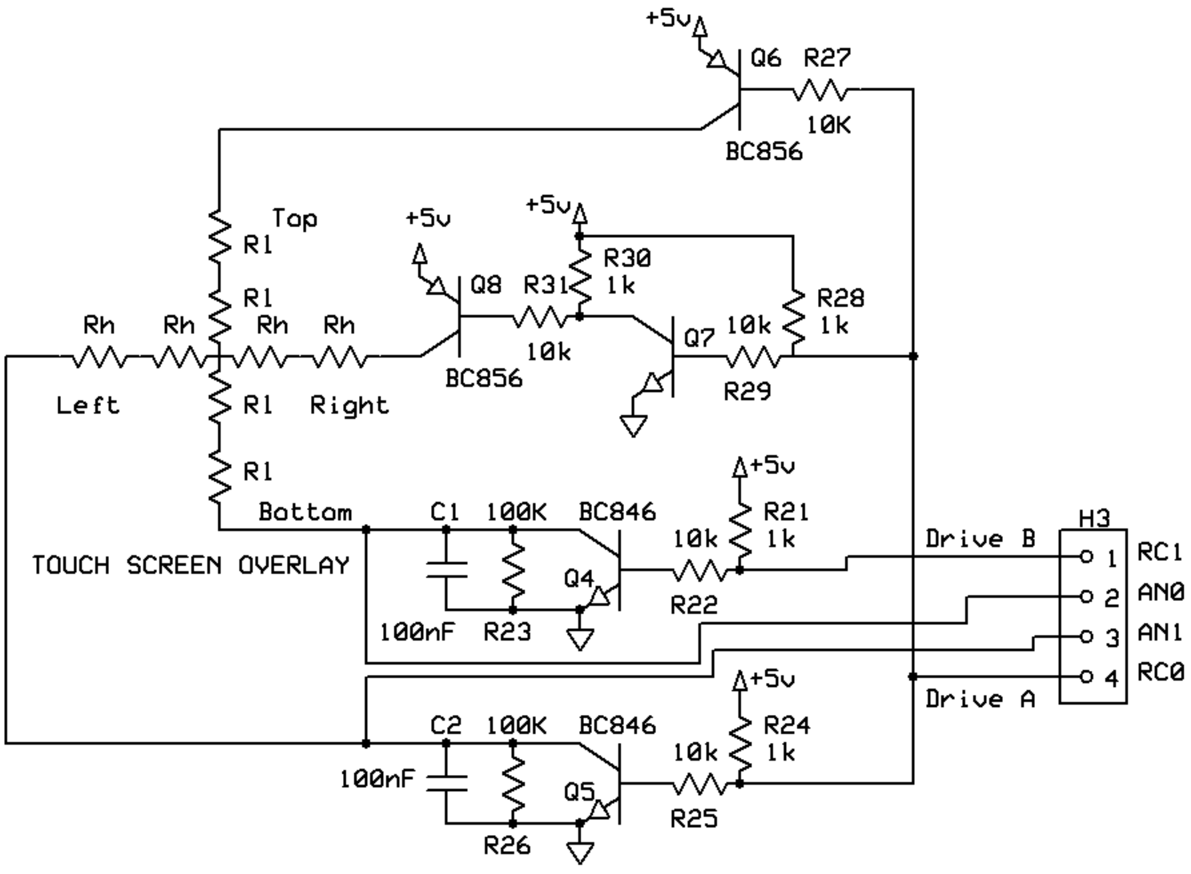


Figure 6: Resistive touch screen overlay driver

As an independent system, it is necessary to include this hardware to interface the touchscreen with the microcontroller.

5.0 Software Development

5.1 Software Development System: The software has been developed in C on the C18 compiler. MPLAB, an IDE supporting all PIC controllers has been used as the development environment and to manage all the code files. Both the tools are from Microchip and are available on Microchip website: www.microchip.com

5.2 Main Flowcharts & Algorithms: The algorithms for following features are presented:

5.2.1 Digital Read-outs (DROs) – uni-directional (s4_1_2_1a)
5.2.2 DROs – bi-directional (s4_1_2_1b)
5.2.3 Missed pulses detection (s4_1_2_2a)
5.2.4 Direction sense (s4_1_2_2b)
5.2.5 Duty Jitter (s4_1_2_3b)
5.2.6 Phase Jitter (s4_1_2_3a)
5.2.7 Phase evaluation (s4_1_2_5a)
5.2.8 Hardware checks (s4_1_2_4)

All these tests can be clubbed into two categories – Category A: which require a uniform motion (constant rpm) and category B, which do not. All A-categories tests (extended) require an external drive.

5.2.1 DROs – uni-directional (s4_1_2_1b)

All the six channels digital level tests are paralleled. The counts are displayed on the screen in real time. This test is prone to errors introduced by a motion jerk due to its uni-directionality counting mechanism in which the counts are added in either directional motion. The difference between the number of counted pulses and rated number of pulses gives the intensity and frequency of the jerking / reverse directionality events. Refer figure 7 for the flowchart.

5.2.2 DROs – bi-directional (s4_1_2_1b)

The counter rotations are compensated in this case by sensing the directions. Since we have 4 channels representing quadrature motions precisely, each period (of a specific channel) is divided into 4 quadrature counts recorded by a 2-byte unsigned counter cZ which either adds or subtracts depending upon the direction that is sensed. The final count of the ppr is given by

$$ppr = \frac{cZ}{4}$$

The direction is sensed by following a specific pattern appearing on channel A and B. The simultaneity of these two channels form a 2-bit word which follows a sequence 11-01-00-10 in case of a clockwise rotation and 11-10-00-01 in case of counter clockwise rotational direction, as shown in figure 8. Note that a complete sequence of 4 2-bit words is not needed to sense the direction in case of a uniform motion (non-jerking / non-vibrating conditions). At any given state of the 2-bit word, the processor expects two any of the other three words to represent either of the two directions. If the fourth word appears, the process is terminated abruptly to indicate an error and restarted. This might be caused due to missing pulses.

For example, If the current state is 01, the next word can either be 01 (no change), 00 (cw direction) or 11 (ccw direction). In case of 00, cZ is incremented by one whereas in case of 11, it is decremented. Appearance of 10 in this case is erroneous. The process is initiated and terminated on channel Z H-L transition ISR. The complete process is shown in flowchart figure 9.

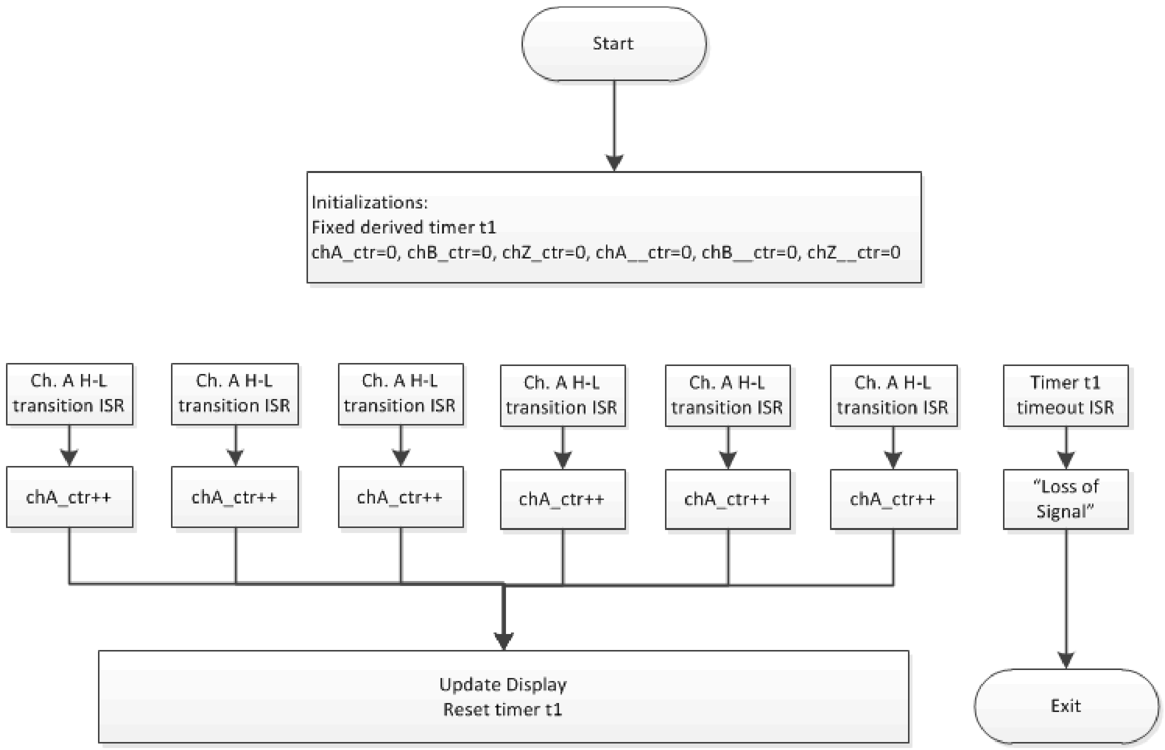


Figure 7: DROs – uni-directional

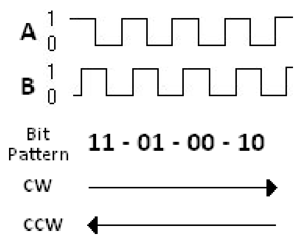


Figure 8: Directional sense and compensation

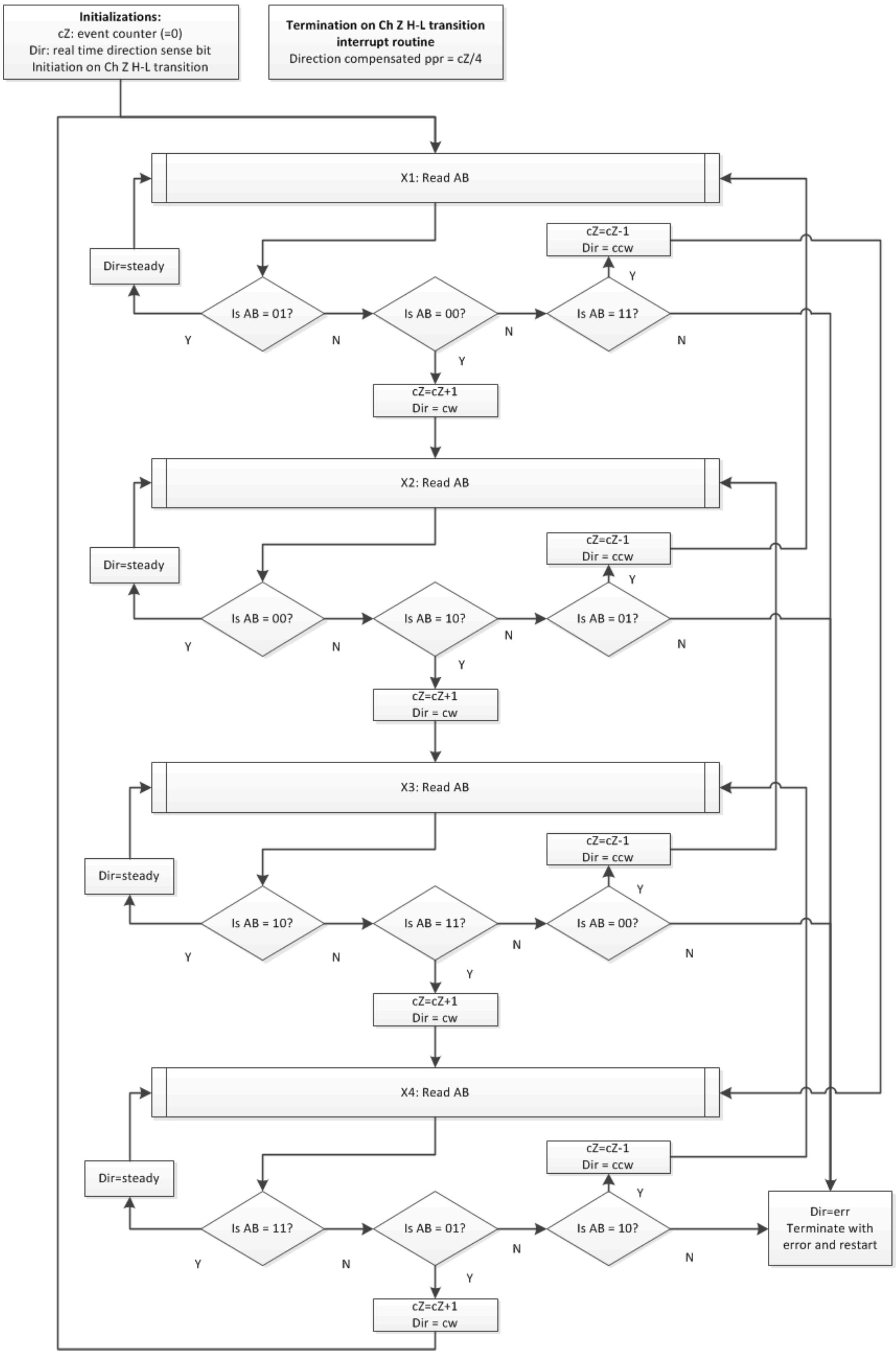


Figure 9: Flowchart for DROs – with bi-directionality compensation

5.2.3 Missed pulses detection (s4_1_2_2a)

One of the most common sources of error in DRO is the missed pulses on a particular channel. Such an error is profound and increases with time where there is no cross-referencing based on other channel is done. The reason for such damage is usually slit-blockage in case of optical encoders and un-resolvable. If an exact position at which the pulse is missed is known, the servo system controller can be compensated.

The channel digital output is continuously matched with its complementary to detect a missing pulse. Whenever the digital vector addition of a state of a channel with its complementary is 0, i.e. the exclusive OR-ing is 0, the position is recorded. Whether a channel or its complementary is missing the pulse, is decided by comparing the current state of the channel with its previous one. For the channel, if both are different, it implies that the complementary of the channel has missed the pulse at recorded position 'pos'. A timeout timer t1 is used to detect absence of marker pulse and t2 is used to detect the signal loss. The process is carried out similarly for the other complementary channel pair, referencing the complementary marker channel. At each successful exclusive OR-ing operation the 'pos' is incremented whereas at each failure the location is assigned to location[x] variable and x is incremented to create a new space for the next possible error. Again, the process is initiated and terminated on channel H-L transition ISR. The complete process for channel A and A' is shown in flowchart figure 10.

5.2.4 Direction Sense (s4_1_2_2b)

This is the derived functionality from DROs – bi-directional (s4_1_2_1b) block. At each quadrature direction sensing, a byte status is changed to 0 if clockwise direction is detected (dir=cw) and to 1 if counter clockwise direction is detected (dir=ccw). The status of this bit can be monitored to give real time direction. All the values of dir and their derived meaning are populated in table 2.

Dir	Value	Implies
cw	0	Clock wise direction detected
ccw	1	Counter Clockwise direction detected
steady	2	No motion detected
err	3	Erroneous motion detected

Table 2: Dir values and its meaning

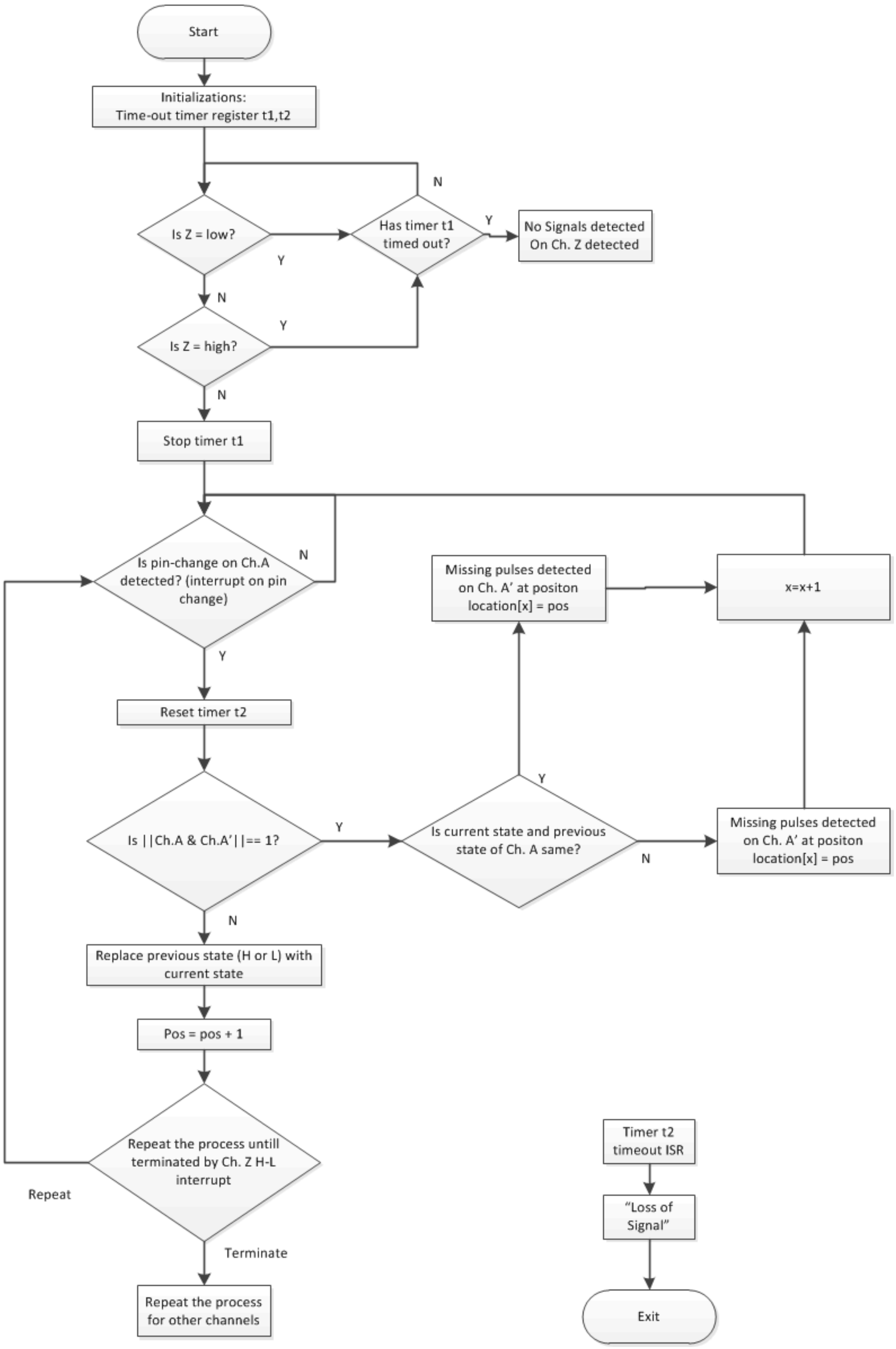

Start
Initializations: Time-out timer register t1,t2
Is Z = low?
Has timer t1 timed out?
N
Y
No Signals detected On Ch. Z detected
Y
N
Is Z = high?
Y
N
Stop timer t1
Is pin-change on Ch.A detected? (interrupt on pin change)
N
Y
Missing pulses detected on Ch. A' at positon location[x] = pos
x=x+1
Reset timer t2
Is ||Ch.A & Ch.A'||== 1?
Y
Is current state and previous state of Ch. A same?
Y
N
Missing pulses detected on Ch. A' at positon location[x] = pos
N
Replace previous state (H or L) with current state
Pos = pos + 1
Repeat the process untill terminated by Ch. Z H-L interrupt
Repeat
Terminate
Repeat the process for other channels
Timer t2 timeout ISR
"Loss of Signal"
Exit

Figure 10: Flowchart for Missed Pulse Detection

5.2.5 Duty Jitter (s4_1_2_3b)

Duty jitter can be caused by external vibrations or by internal miss-alignment of the disc. It can lead to a systematic error which increases with time. The idea of detecting duty jitter is to compare the on level and off level durations of a pulse by a derived timer t1 and t2. A fine uniform motion is required to nullify the effect of false triggers at the edges. Timers t1 and t2 capture the on level and off level durations respectively as shown in figure 11.

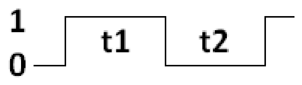


Figure 11: Duty jitter timing t1 and t2 capture

A jitter in duty cycle will cause breakage in symmetry of t1 and t2 timings. The duty jitter is expressed as percentage of period by:

$$\% \, Duty \, Jitter = \frac{t_1 - t_2}{t_1 + t_2} \, x \, 100$$

5.2.6 Phase Jitter (s4_1_2_3a)

Phase jitter is a derived abnormality of the problems mentioned for duty jitter. The idea of capturing phase jitter is to find the factor of dissymmetry existing between two channel pairs A & B, and A' & B' in terms of time durations. Again, two derived timers t1 and t2 are used; t1 captures time between L-H transition on channel A and L-H transition on channel B; t2 captures time between L-H transition on channel B and H-L transition on channel A, as shown in figure 12.

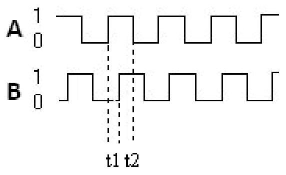
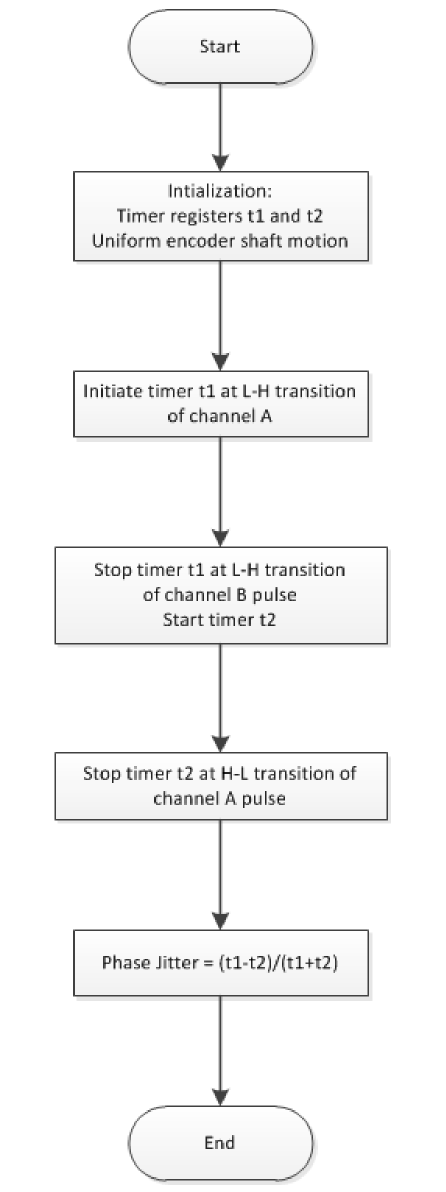


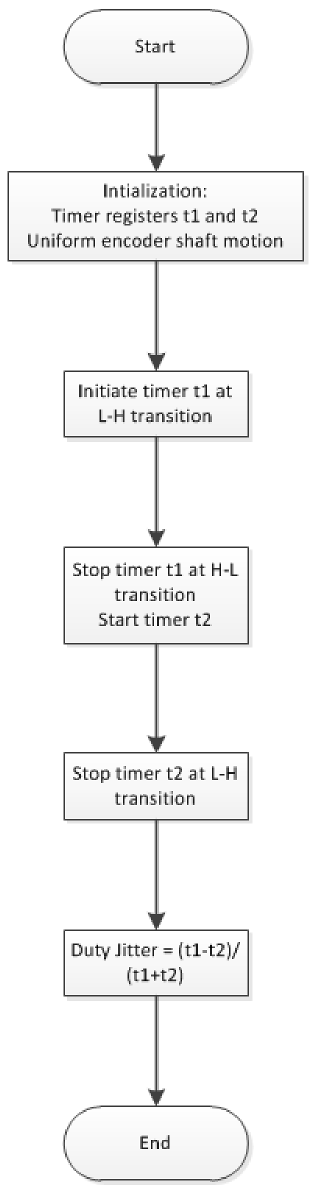
Figure 12: Phase jitter timing t1 and t2 capture

A jitter in phase angles between channels A and B, other than 90° will cause breakage in symmetry between t1 and t2 timings. The phase jitter is expressed in percentage of period by same expression

$$\% \, Phase \, Jitter = \frac{t_1 - t_2}{t_1 + t_2} \, x \, 100$$

The complete process of finding phase jitter and duty jitter is shown in flowcharts figures 13 and 14.


Start
Intialization:
Timer registers t1 and t2
Uniform encoder shaft motion
Initiate timer t1 at L-H transition of channel A
Stop timer t1 at L-H transition of channel B pulse
Start timer t2
Stop timer t2 at H-L transition of channel A pulse
Phase Jitter = (t1-t2)/(t1+t2)
End
Figure 13: Phase jitter calculations


Start
Intialization:
Timer registers t1 and t2
Uniform encoder shaft motion
Initiate timer t1 at L-H transition
Stop timer t1 at H-L transition
Start timer t2
Stop timer t2 at L-H transition
Duty Jitter = (t1-t2)/(t1+t2)
End
Figure 14: Duty jitter calculations

5.2.7 Phase evaluation (s4_1_2_5a)

One of the common problems while using an old encoder is the identification of its lead, either due to time or due to wear and tear, the labeling and indication of the channels disappears. Such an ‘unknown’ device can be subjected to channel identification by this algorithm, provided that the availability of pulses on all the channels is confirmed by the general test. Once, confirmed, the channels are connected to the device at random and a uniform motion is initiated on the encoder shaft. A complex algorithm checks for various timings and states to establish the identity of channels.

The check is started with the six unknown channels connected at Ch1, Ch2, Ch3, Ch4, Ch5 and Ch6. The period of pulses on each of the channel is captured in t1, t2, t3, t4, t5 and t6 channels. Since marker channel pulse (and its complimentary) has the highest period, they can be differentiated at this point into ChZx and ChZy.

Suppose the remaining channels are Chx1, Chx2, Chx3, Chx4. A digital vector addition (exclusive OR-ing) is performed on the Chx1 & Chx2, Chx1 & Chx3 or Chx1 & Chx4 to identify the complementary pairs of channels into Ch1x & Ch1y (complementary) and Ch2x & Ch2y (complementary). For further evaluation, timings are compared between the channels.

A derived timer t1 and t2 are used to capture off period and on period on ChZx channel.

$$If, t_1 > \ t_2, ChZx\ is\ channel\ Z\ and\ ChZy\ is\ channel\ Z'$$

$$If, t_1 < \ t_2, ChZx\ is\ channel\ Z'\ and\ ChZy\ is\ channel\ Z$$

Now, the identity of other channels is established on level tests by referencing with channel Z. At the L-H transition of channel Z, Ch1x level is checked. If it is high, it is confirmed that it is either channel A or channel B. An H-L transition on Ch1x is waited for and then, channel Z is checked. A high confirms Ch1x as channel A and Ch1y as channel A'; whereas a low on channel Z confirms Ch1x as channel B and Ch1y as channel B'.

If as first place, Ch1x level is low, it is confirmed that Ch1y is either channel A or channel B. Again, an H-L transition on Ch1y is waited for and either channel A and A' or channel B and B' are confirmed. Refer figure 15.

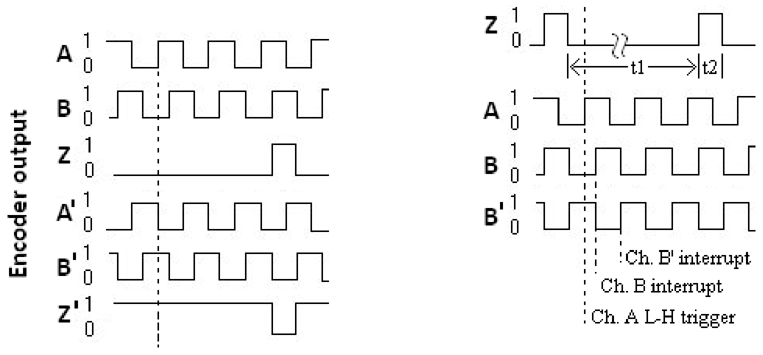


Figure 15: Phase Evaluation

In the last step, to confirm the identity of other channel and its complimentary, the timings of L-H event are observed for the two unknown channels, referencing on the known channel A or channel B. If in previous step, identity of channel A had been confirmed, the B and B' are found out by the 'first' ISR call – the first being channel B and the other being channel B'. Similar check is done to establish identity between A and A', in the other case.

The complete process is mapped into flowchart figure 17 and 18.

5.2.8 Hardware checks (s4_1_2_4)

The output from the fault isolation circuit is also fed back to the microcontroller pin RA4 which monitors the level on it. The voltage at this pin is proportional to the current drawn by the device and detects the current drawn by the encoder. An internal fault will cause the voltage to be cut-off by the PNP-NPN transistor lock-out and will not affect the functioning of the microcontroller. This fall in voltage will be detected and a message is flashed on the screen to alert the user.

5.3 Peripheral Flowcharts and Algorithms.

5.3.1 GLCD interface

The basic GLCD timing diagram to display a byte of data is shown in flowchart figure 16. It takes around 18µS to display a byte of data. Although, this is not the time mentioned in datasheet, I found it best to on which the GLCD works well without producing junk on screen. The font library consist each font of 6 bytes, thus making the display time delay as 108µS per character.

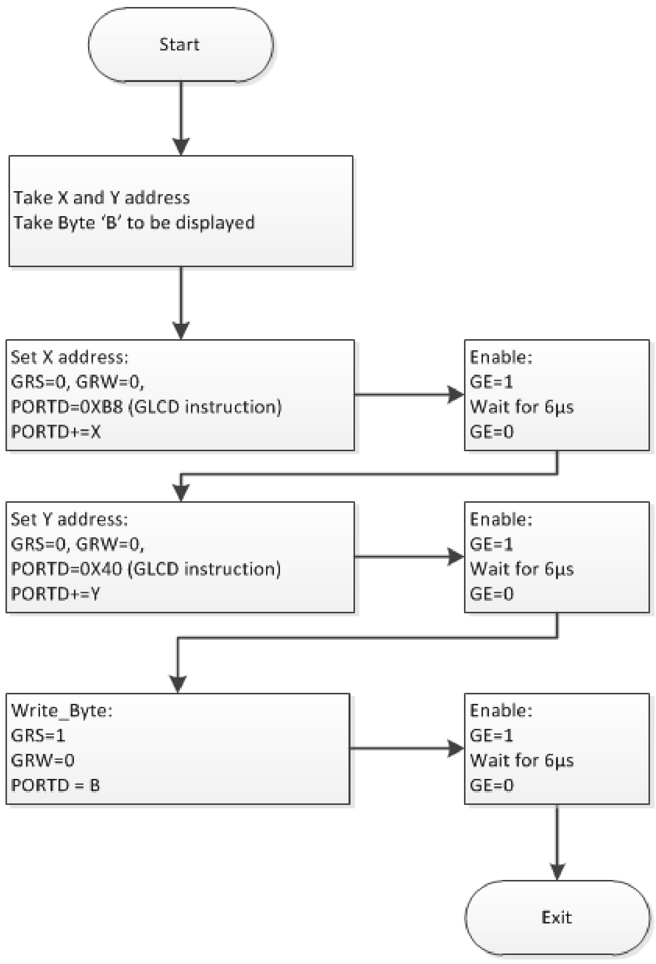


Figure 16: Byte printing timing flowchart for GLCD

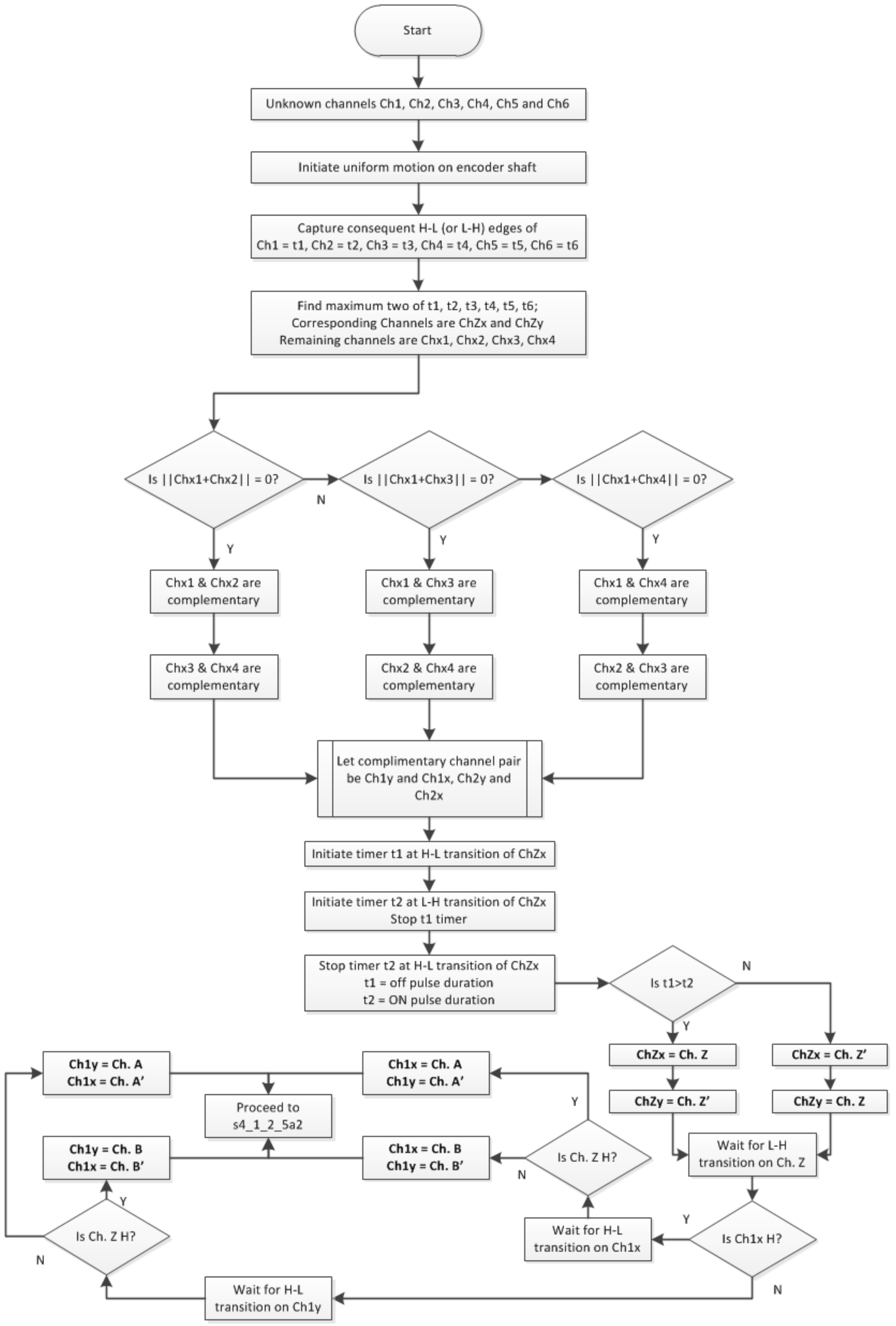

Start
Unknown channels Ch1, Ch2, Ch3, Ch4, Ch5 and Ch6
Initiate uniform motion on encoder shaft
Capture consequent H-L (or L-H) edges of Ch1 = t1, Ch2 = t2, Ch3 = t3, Ch4 = t4, Ch5 = t5, Ch6 = t6
Find maximum two of t1, t2, t3, t4, t5, t6; Corresponding Channels are ChZx and ChZy Remaining channels are Chx1, Chx2, Chx3, Chx4
Is ||Chx1+Chx2|| = 0?
N
Is ||Chx1+Chx3|| = 0?
Is ||Chx1+Chx4|| = 0?
Y
Chx1 & Chx2 are complementary
Chx3 & Chx4 are complementary
Y
Chx1 & Chx3 are complementary
Chx2 & Chx4 are complementary
Y
Chx1 & Chx4 are complementary
Chx2 & Chx3 are complementary
Let complimentary channel pair be Ch1y and Ch1x, Ch2y and Ch2x
Initiate timer t1 at H-L transition of ChZx
Initiate timer t2 at L-H transition of ChZx Stop t1 timer
Stop timer t2 at H-L transition of ChZx t1 = off pulse duration t2 = ON pulse duration
Is t1>t2
N
Y
ChZx = Ch. Z
ChZy = Ch. Z'
ChZx = Ch. Z'
ChZy = Ch. Z
Wait for L-H transition on Ch. Z
Is Ch1x H?
Y
N
Wait for H-L transition on Ch1x
Is Ch. Z H?
Y
N
Ch1x = Ch. A Ch1y = Ch. A'
Ch1x = Ch. B Ch1y = Ch. B'
Proceed to s4_1_2_5a2
Ch1y = Ch. A Ch1x = Ch. A'
Ch1y = Ch. B Ch1x = Ch. B'
Is Ch. Z H?
Y
N
Wait for H-L transition on Ch1y

Figure 17: Flowchart for Phase Evaluation – part1

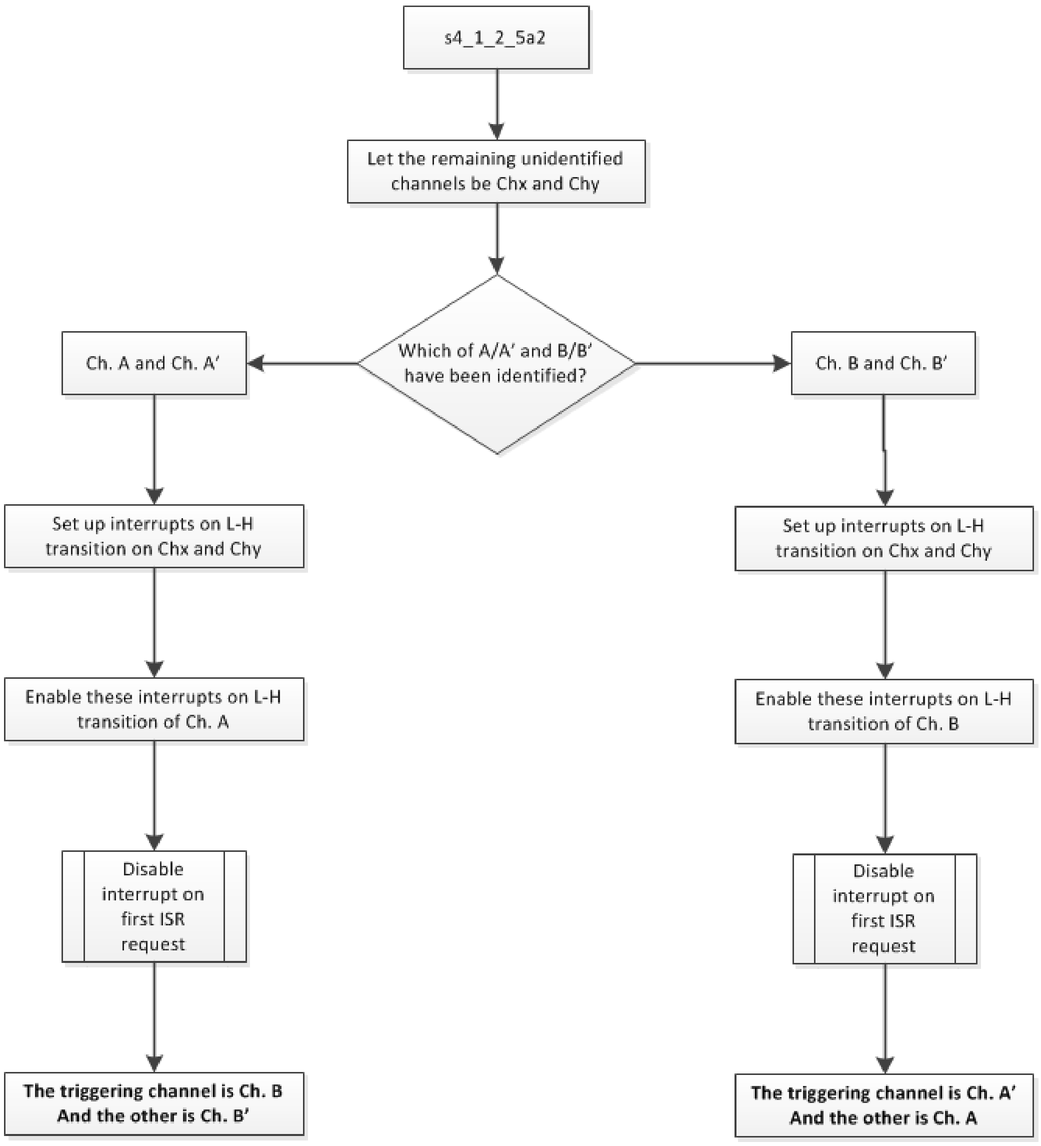

s4_1_2_5a2
Let the remaining unidentified channels be Chx and Chy
Which of A/A' and B/B' have been identified?
Ch. A and Ch. A'
Ch. B and Ch. B'
Set up interrupts on L-H transition on Chx and Chy
Enable these interrupts on L-H transition of Ch. A
Disable interrupt on first ISR request
The triggering channel is Ch. B And the other is Ch. B'
Set up interrupts on L-H transition on Chx and Chy
Enable these interrupts on L-H transition of Ch. B
Disable interrupt on first ISR request
The triggering channel is Ch. A' And the other is Ch. A

Figure 18: Flowchart for Phase Evaluation – part2

5.3.2 Resistive Touchscreen Interface

The touchscreen() function has been coded which retrieves X and Y as shown in the flowchart figure 19.

When the touchscreen is not pressed, the read-outs are under (2, 11) coordinates. This enables to detect the press and enable touch de-bounce. This is done by function Release() which does not exit unless the touch is released.

```
Void touchscreen(void)
{

        unsigned int t2;
        //SETTING UP ADC MEASUREMENT

        PORTCbits.RC0=1;                    //Drive A=1, Drive B=0
        PORTCbits.RC1=0;

        ADCON0 = 0b000000;
        ADCON2 = 0b10011101;
        ADCON0bits.ADON = 1;                //turn on ADC
        Delay10TCYx(1);
        ADCON0bits.GO_DONE = 1;             //start conversion

        while(ADCON0bits.GO_DONE==1);

        t2=(ADRESH*256)+ADRESL;
        t2=t2/5;

        adc_x=t2-0;                         //x-coordinate

        PORTCbits.RC0=0;                    //DriveA=0, DriveB=1
        PORTCbits.RC1=1;
        ADCON0 = 0b000100;
        ADCON2 = 0b10011101;
        ADCON0bits.ADON = 1;                //turn on ADC
        Delay10TCYx(1);
        ADCON0bits.GO_DONE = 1;             //start conversion

        while(ADCON0bits.GO_DONE==1);

        t2=(ADRESH*256)+ADRESL;
        //t2=(t2-215)/7;
        t2=t2/5;
        adc_y=t2-0;                         //y-coordinate

        ADCON0bits.ADON = 0;                //turn off ADC
}
```

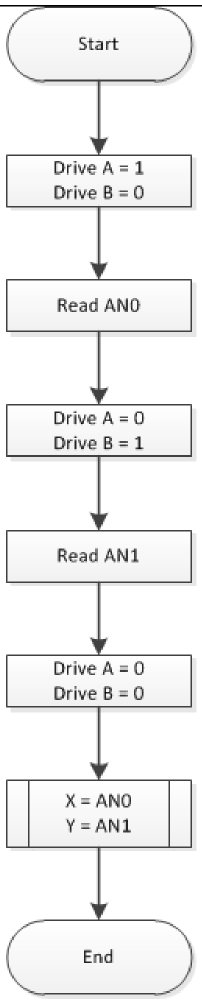


Figure 19: Flowchart for X,Y coordinate capture

```
void Release(void)
{
        while(adc_x>2+1 && adc_y>11+1)
        {
                touchscreen();
        }
}
```

5.3.3 Touch Screen Calibration

The resistance of the touchscreen layers changes by temperature which changes the coordinates. A compensation of this is done by calibrating the touchscreen at the start of use. In the code, the calibration subroutine has been kept optional and is initiated if the user touches the screen when the appropriate message is displayed at power-up. The calibration is skipped after pre-set 4 seconds.

The calibration routine captures the four corner points of the touchscreen by asking user to press the displayed dot. To avoid the false capture, a message to 'release' is flashed on screen until the user releases the press. The captured coordinates are

$$(X_{tl}, Y_{tl}), (X_{tr}, Y_{tr}), (X_{br}, Y_{br}), and\ (X_{bl}, Y_{bl})$$

A resolution of sense area in X and Y direction are established by

$$X_{res} = 8,\ Y_{res} = 4$$

This means that there are 8 sense areas along X axis and 4 along Y axis.

The calibration compensation is achieved by

$$X_{psa} = \frac{(X_{tr} + X_{br}) - (X_{tl} + X_{bl})}{(\ 2\ X\ X_{res})}, Y_{psa} = \frac{(Y_{tl} + Y_{tr}) - (Y_{bl} + Y_{br})}{(2\ X\ Y_{res})}$$

This is a simple averaging linear compensation and works well for low values of $X_{res}\ and\ Y_{res}$ and is sufficient for current needs. A higher sense resolution requiring stylus use to write or sign on the screen requires high $X_{res}\ and\ Y_{res}$ values, typically full resolution of screen of 128 and 64 pixels. In such a case a quadratic compensation needs to be employed.

5.3.4 Backlight control

The PWM is increase / decreased depending on the area touched on the screen in 'backlight settings' screen. The PWM goes from low 10 to high 103 counts for full intensity.

```
PR2 = 103;
CCPR1L = 0b00110011 ;
CCP1CON = 0b00111100 ;
T2CON = 0b00000111;

testcoordinates(6,12,15,31);                // test for decrement button press
if(pressconfirm==1)
        {
        if(CCPR1L>10)
        {
                Delay10KTCYx(10);
                CCPR1L = CCPR1L-1;          //decrement PWM count
                printbar();                 //print status bar
        }
}

testcoordinates(16,25,14,25);               //test for increment button press
if(pressconfirm==1)
{
        if(CCPR1L<103)
        {
                Delay10KTCYx(10);
                CCPR1L = CCPR1L+1;          //increment PWM count
                printbar();                 //print status bar
        }
}
```

5.4 GUI

A simple hierarchal top-down navigation is used to browse through all the options available in the device. The pre-sections and post-sections can be accessed by CANCEL, NEXT, CONTINUE buttons wherever applicable. The selectable options are enclosed by a rectangular on-screen box to make their presence more pro-found. Since a linear averaging calibration is used to get the touch points, fewer options can be accommodated on the screen; but that has led to comfortable operate the touch screen without the use of any pointing device like stylus. The operation is very satisfactory with fingers. The complete navigation through the available menus is shown in figure 20.

5.5 Code

The code will be available upon request.

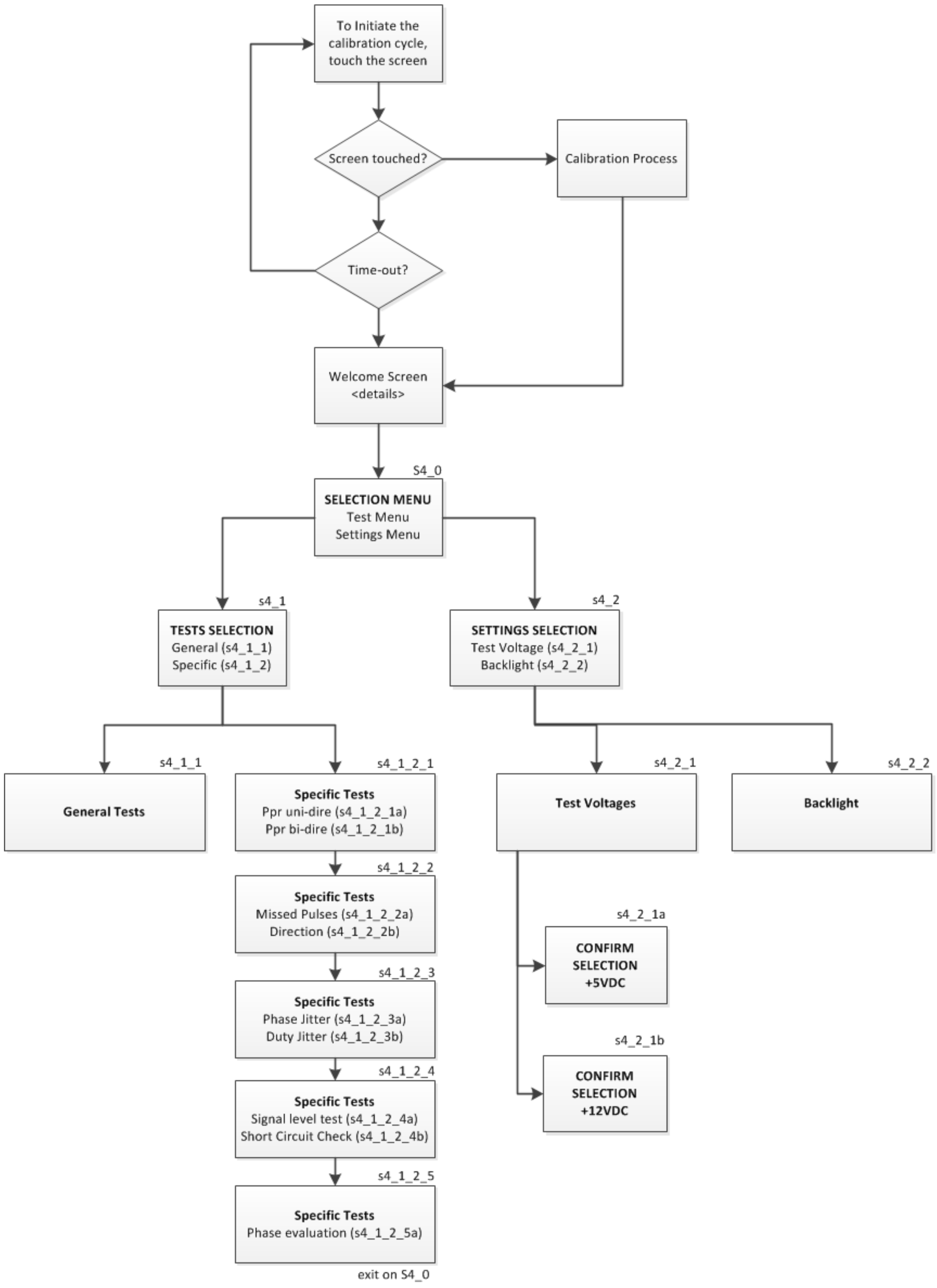
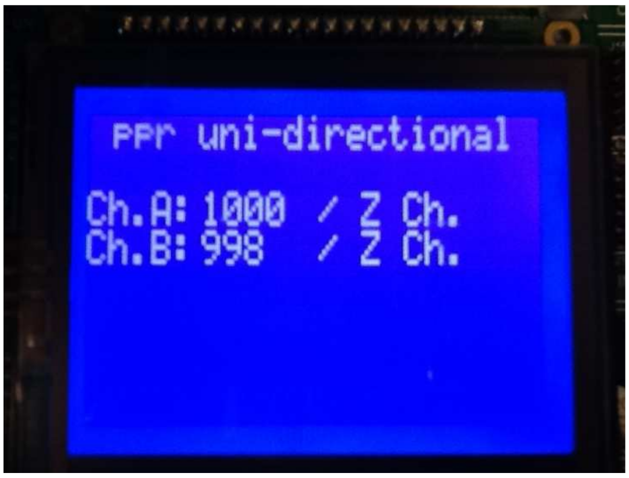


Figure 20: Navigational menus on the GUI

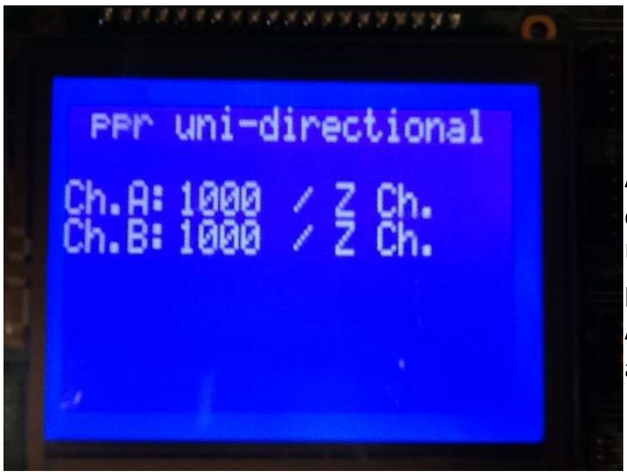
6.0 Results

The signals from the Siemens Pgcoder were captured on ppr uni-directional mode. The screenshot figure 21 shows the ppr measured on channel A and B. The channel B suffered a loss of two pulses due to possible error in signal acquisition due to low rise-time of optical isolators at L-H transition, due to high speed. A jerk would cause an over-count due to hunting oscillation in this mode.



```
ppr uni-directional
Ch.A: 1000  / Z Ch.
Ch.B: 998   / Z Ch.
```

Screenshot figure 21

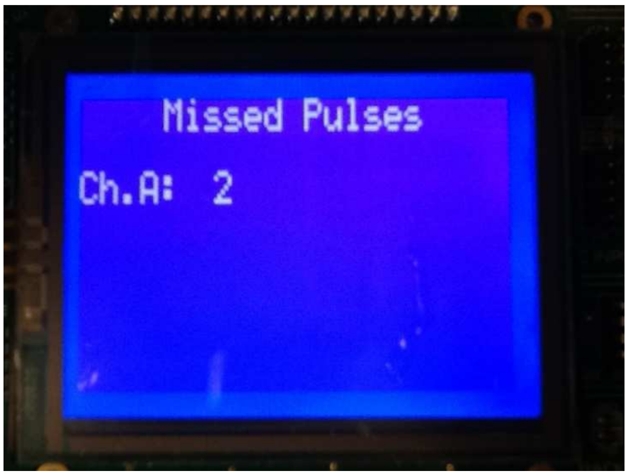


```
ppr uni-directional
Ch.A: 1000  / Z Ch.
Ch.B: 1000  / Z Ch.
```

Screenshot figure 22

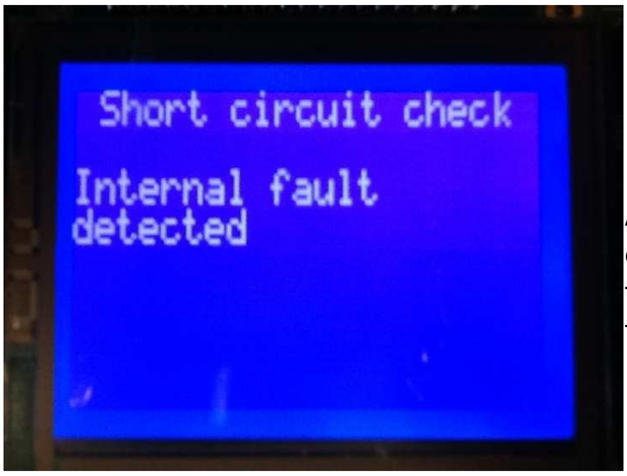
A careful slow motion led to successful capture of correct ppr.
Use of faster opto-isolators will solve this problem.
A video clip showing the pulse capture is available at http://youtu.be/ZWgCtSjk52M

The missed pulse test on channel A captured a count of 2 pulses when the shaft was driven with non-uniform forceful rotation given by hand. This again implies the possible miss-outs of the pulses by 4N35 slow optical isolators.



```
Missed Pulses
Ch.A:  2
```

Screenshot figure 23



Short circuit check

Internal fault detected

An external short-circuit simulated at the output terminals of the 'fault isolation circuit' triggered 'internal fault detected' message on the screen

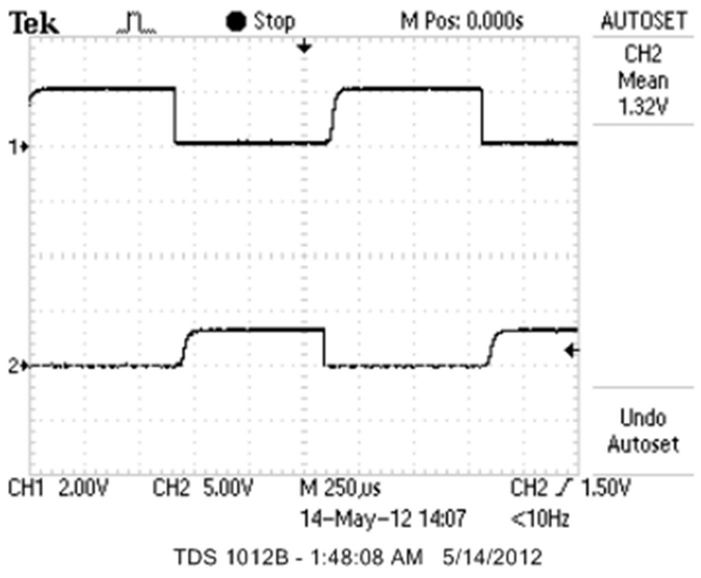
Screenshot figure 24

The test on phase jitter and duty jitter showed the least capture of 1.3% and 3.7% respectively. This error has been introduced because of the fact that the shaft was rotated manually by hand. A more intesive test on this feature can be explored by rotating the shaft with uniform motion by an external drive (extenstion).

The navigation through all the menus was done on GUI. This successfully tests the calibration and touchscreen debounce algorithm. The video clip showing this is available at http://youtu.be/vz_l7hcWIj8

Other screen shots are shown in Appendix II.

Problem faced with Signal Acquisition

One major problem that I faced with using optical isolators is the slow rising time. Figure 25 shows the input waveform to microcontroller of a complementary channel pair. The low to high (L-H) transition is not captured properly and results into an ambiguous read after the microcontroller loading effects on the signal. I suffered erroneous results due to this several times.	 *Figure 25: Optical isolators' output and L-H edge damage*

7.0 Conclusion

The signal acquisition can be enhanced by using faster optical isolators like 6N137 to avoid signal loss and increase the reliability of the device. The ambiguous L-H edge of the optical isolator also needs to be addressed to prevent a miss-capture of the pulse.

All the category B tests can be effectively carried out by this device. To enable the effectiveness of category A tests – which require a uniform smooth motion, the project can be extended to include a stepper motor drive to do the function. A separate drive controller can be interfaced with this main controller for driving this external drive. A flexible mechanical coupling need to be designed which provides possibility of hooking up any encoder. Such an interface also enables to compute several additional parameters like mechanical lag (eccentricity error) and directional repetitive error (while going in opposite directions consequently) of the encoder and auto-test incorporating a complete test of the encoder.

The tests discussed so far, for this device, provide a reliable health checkup of an encoder. The device demonstrated here captures information from these tests as intended and helps in diagnosing the device performance, thus enabling faster and effective maintenance activities, and reducing the machine breakdown time and increasing its productivity.

8.0 Acknowledgements

I am extremely thankful to my advisor, Bruce Land, ECE Cornell University, for useful tips and providing the essential development tools required in the project.

I am also thankful to Pradeep Singh Chokhdayat, Head – Maintenance, Larsen & Toubro Ltd., HED, for helping me understand the device basics and the underlying problems therein.

I am also thankful to the Microchip support team for quick resolution of queries and clarifications related to PIC controllers

9.0 References

1. NT7108C datasheet and application note:
   http://www.newhavendisplay.com/app_notes/NT7108.pdf
2. TI 4-wire and 8-wire touchscreen basics and interfacing
   http://www.ti.com/lit/an/slaa384a/slaa384a.pdf
3. PIC18F1685 datasheet
   http://www.ti.com/lit/an/slaa384a/slaa384a.pdf
4. Rotary encoders – Siemens product catalog and information page
   http://www.cncdesign.com/product/encoders_a1.html

Appendix I – PIC18F4685 pin diagram

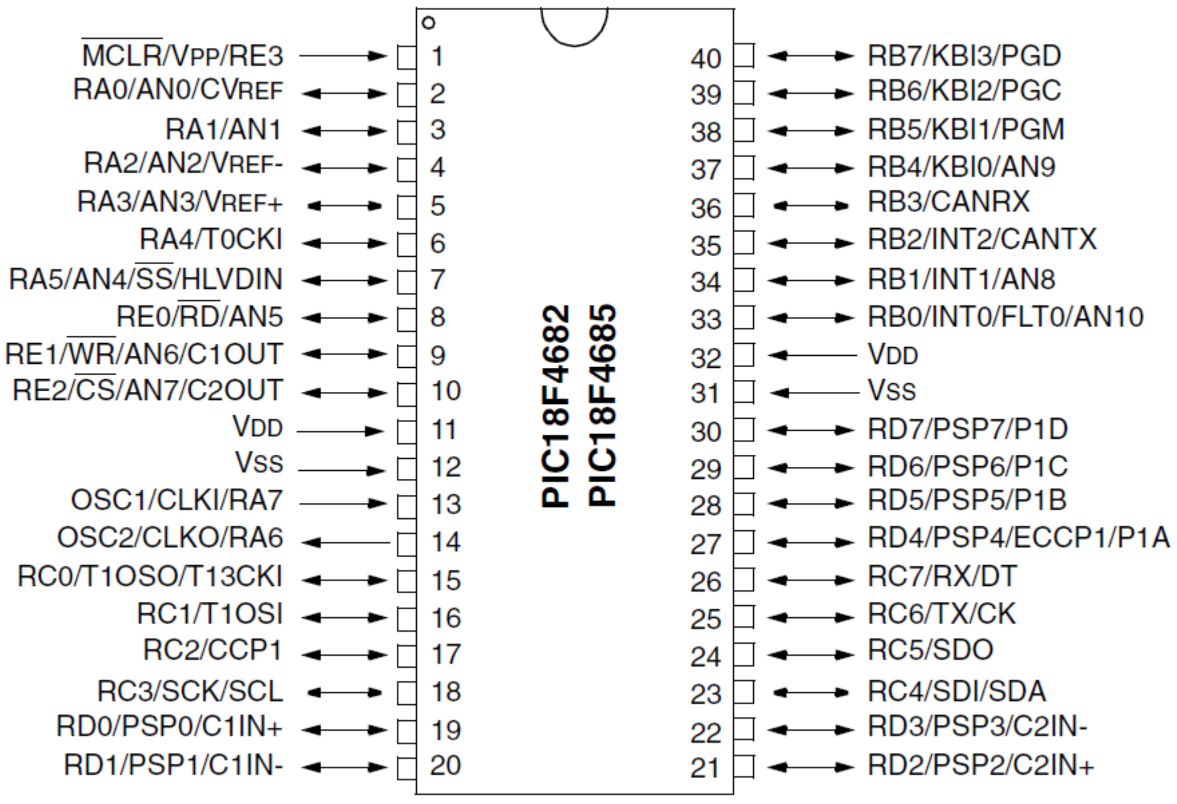
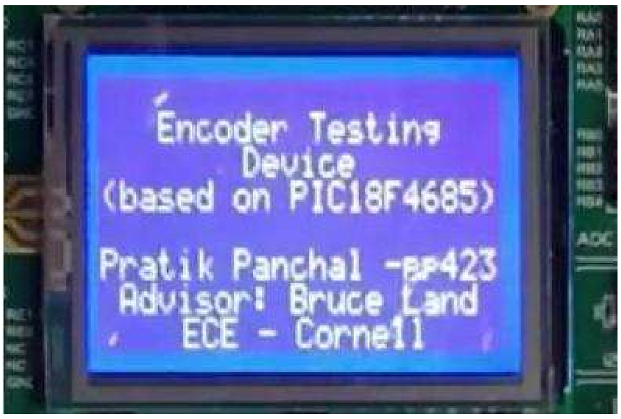


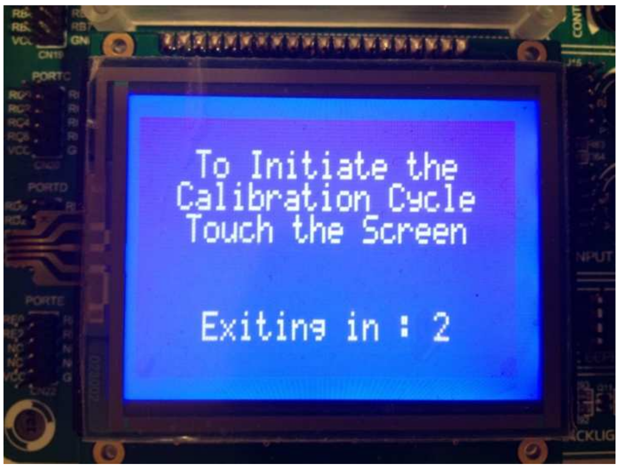
Figure 26

Appendix II – GLCD screenshots / snapshots


Encoder Testing Device (based on PIC18F4685)

Pratik Panchal -pp423
Advisor: Bruce Land
ECE - Cornell

Figure 27: Main screen


To Initiate the Calibration Cycle Touch the Screen

Exiting in : 2

Figure 28: Calibration cycle screen

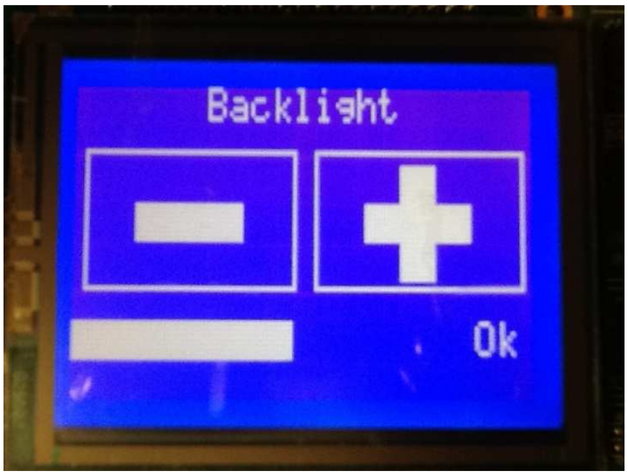


Figure 29: Backlight control

Appendix III – Development System

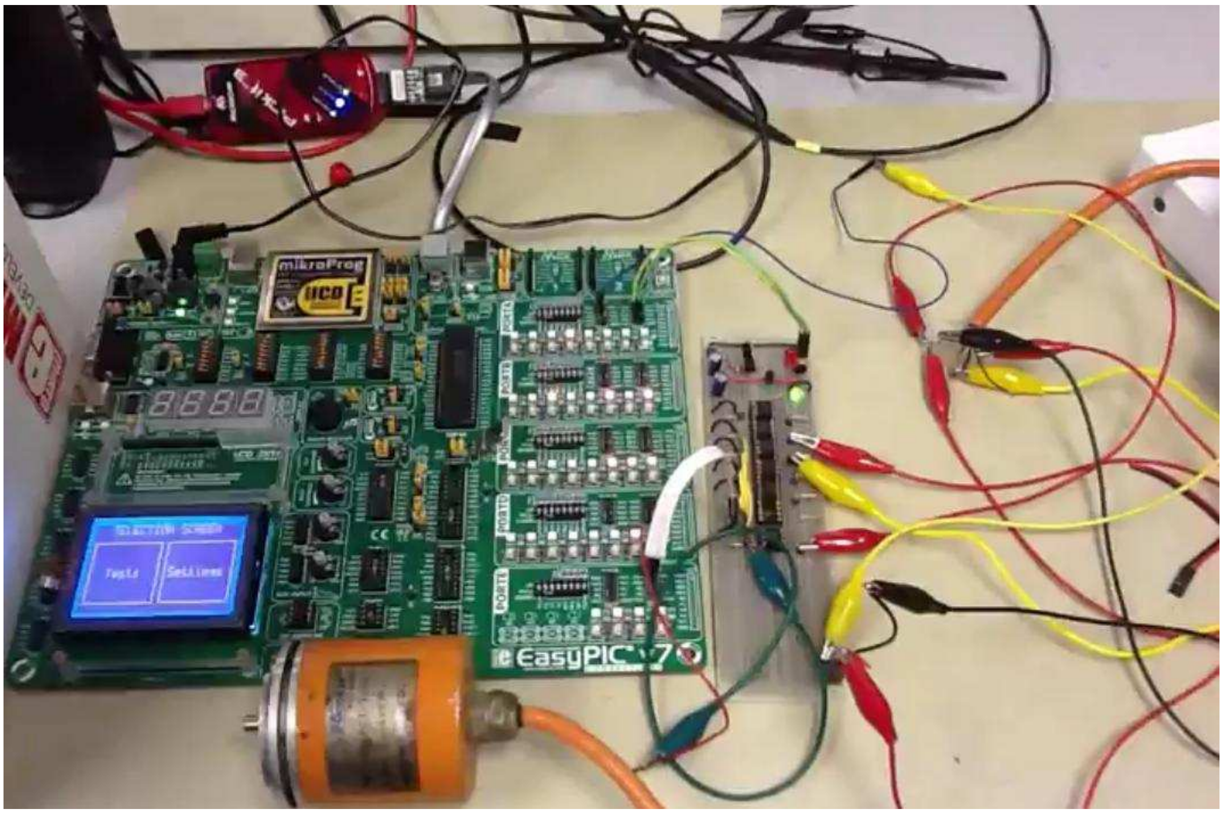


Figure 30

Link to the Poster (presented on ECE day) and other documents:
http://iesa.pratikpanchal.com