

# Oakland County Competitive Robotics Association - 2013

Marc Center, GM Electrification Engineer,  
OCCRA VEX Mentor 2008-2013

VEX WORKSHOP 2 – Tuesday Sept 17, 2013

Closed-Loop PI Motor Speed Control

## Closed-Loop PI Motor Speed Control

1. Youtube slow motion demo
2. Why Closed-Loop Speed Control?
3. Test Setup – 6 inch wheel with Hall Effect Sensor for Speed Measurement (6 pulses/revolution)
4. VEX Cortex wiring for speed sensor/motor
5. VEX Cortex EasyC controller configuration
6. VEX Cortex EasyC PWM information
7. Using Cortex program to establish RPM setpoints vs PWM open-loop command
8. PWM vs RPM curve (~75 rpm/PWM count)
9. EasyC Motor speed control with Joystick
10. Cherry Hall Effect Sensor details – Jameco

## Closed-Loop PI Motor Speed Control

- 11. Speed Sensor uses Optical Shaft Encoder input
- 12. Determination of RPM – Engineering Unit Cancellation
- 13. Cortex V4 Easy-C software implementation of RPM Determination (1 second Loop)
- 14. Closed-Loop Correction Mathematics
- 15. Closed-Loop Correction = PropCorr + IntCorr
- 16. Bench Test Case Results – Terminal & Graphic output



Slow motion frisbee shooting



Upload



GUIDE



MORE RESULTS  
Slow motion frisbee ...



0:09 / 0:24



## FRC 3081 Frisbee Shooter Test Slow Motion Video



stevep001 · 2 videos



102 views



Like



About

Share

Add to



Published on Jan 14, 2013  
Shot at 480fps.

Remix this video!

# Why Closed-Loop Speed Control?

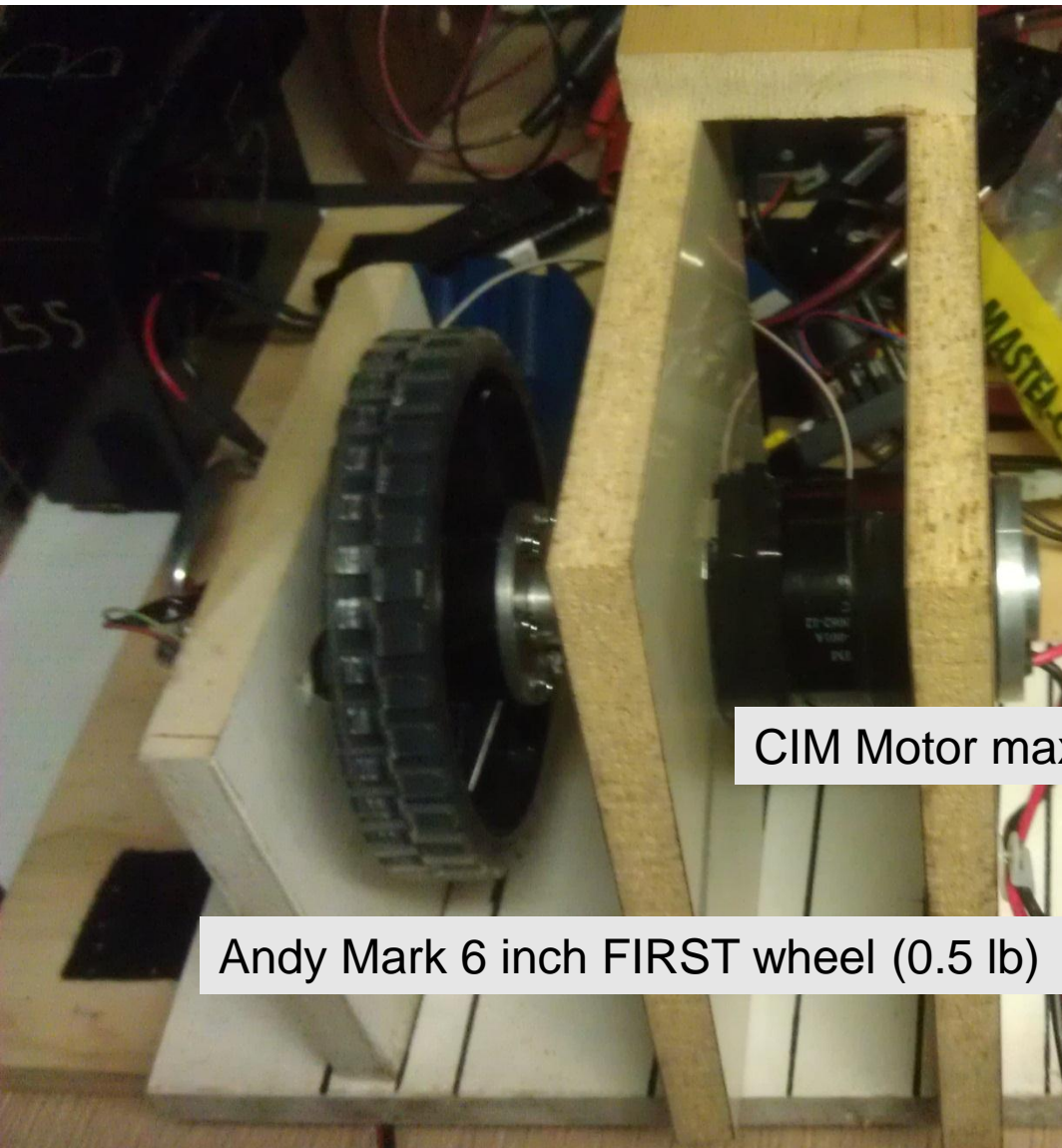
- If you want to shoot 3 balls in succession, the shooting device will slow down after shooting each ball because it is transferring energy from it's rotating member to the stationary ball.
- Closed-Loop Speed Control can very quickly detect the change in speed (0.1 seconds) and compensate by commanding the motor to increase speed to maintain the speed setpoint.

## Without Closed-Loop Speed Control (Open-Loop)

Wait 5 seconds to get to steady 2500 rpm, Shoot ball,  
Repeat two more times – total time (15 seconds)

## With Closed-Loop Speed Control

Wait 3 seconds to get to steady 2500 rpm, Shoot ball,  
Repeat two more times – total time (9 seconds)

**Product Overview:**

2.5 inch CIM, Brushed DC Motor

Motor has this label, stamped in white:

AM802-001A  
PM25R-45F-1003  
12VDC PM25R  
RoHS  
101512

**Specifications:**Physical Specs:

- Size: 2.5 inch diameter, 4.34 inch long body
- Output Shaft size: 0.313 +/- 0.0004, with 2mm keyway
- Weight: 2.82 pounds
- Mounting Holes: #10-32 tapped holes (2), on a 2" bolt circle

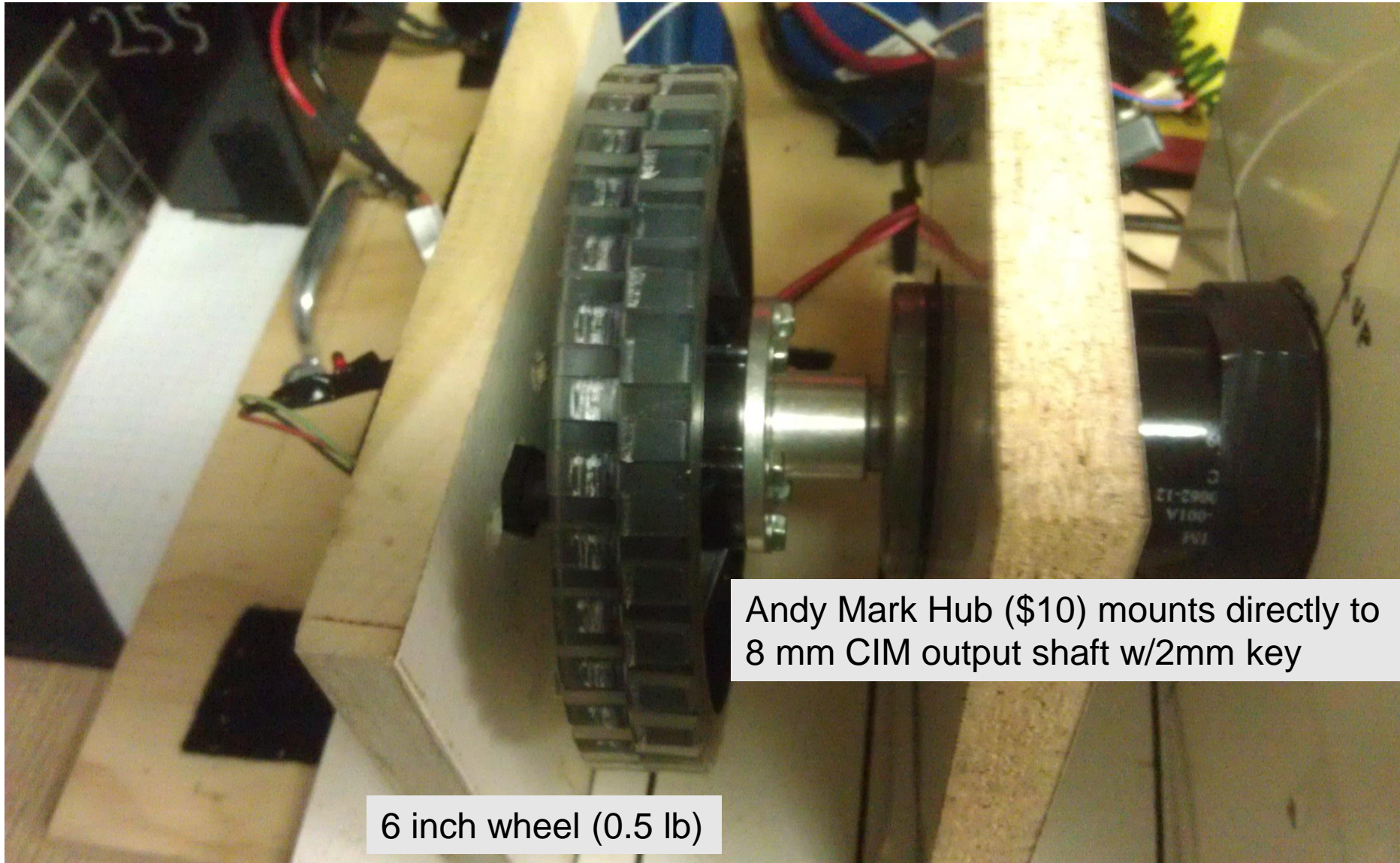
Performance Specs:

- Voltage: 12 volt DC
- No load RPM: 5,310 (+/- 10%)
- Free Current: 2.7 amps
- Maximum Power: 337 Watts (at 2655 rpm, 172 oz-in, and 68 amps)
- Stall Torque: 2.42 N-m, or 343.4 oz-in
- Stall Current: 133 amps

CIM Motor max rpm 5310 +/- 10%

Andy Mark 6 inch FIRST wheel (0.5 lb)

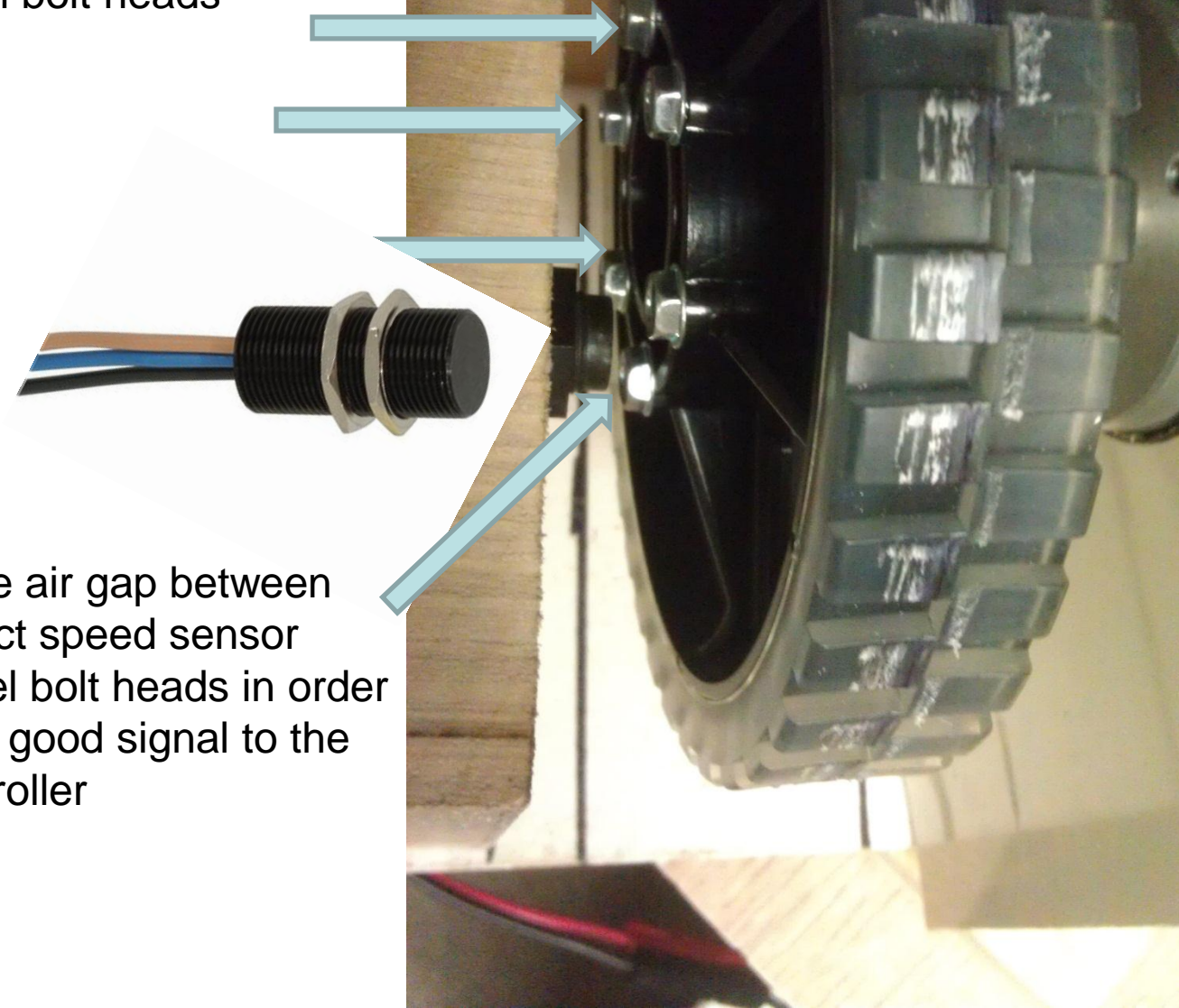




Andy Mark Hub (\$10) mounts directly to 8 mm CIM output shaft w/2mm key

6 inch wheel (0.5 lb)

The Cherry Hall Effect Sensor is positioned in proximity to the 6 – steel bolt heads



Minimize the air gap between the hall effect speed sensor and the steel bolt heads in order to provide a good signal to the Cortex controller



6 inch wheel (0.5 lb)

## **CHERRY** Gear Tooth Speed Sensor

### **Commercial Grade Hall Effect**

- Operating voltage range: 4.5-24VDC
- Output type: sink
- Output current: 25mA max.
- Supply current: 6mA max.
- Barrel length: 1.0"
- Output voltage: 400mV max.
- Thread: 15/32" – 32
- Wire: 20 AWG x 1m
- Reverse battery protection from 24VDC
- Operating temperature range: -40°C to 125°C

Part No.	Mfr. Part No.	1	5	25
<u>512401</u>	GS100701 .....	\$29.95	\$26.95	\$23.95

Cherry Gear Tooth Speed Sensor



### CHERRY Gear Tooth Speed Sensor

#### Commercial Grade Hall Effect

- Operating voltage range: 4.5-24VDC
- Output type: sink
- Output current: 25mA max.
- Supply current: 6mA max.
- Barrel length: 1.0" • Output voltage: 400mV max.
- Thread: 15/32" - 32 • Wire: 20 AWG x 1m
- Reverse battery protection from 24VDC
- Operating temperature range: -40°C to 125°C

Part No.	Mfr. Part No.	1	5	25
512401	GS100701 .....	\$29.95	\$26.95	\$23.95



Motor 2 output (missing) and Encoder 2 input are used

# Controller Configuration – where digital inputs/outputs can be selected

easyC V4 for Cortex - Joystick Project (WiFi) - Workshop 2 Calculate RPM program 19AUG13

File Edit View Project Build and Download Tools Window Help

Function Blocks Start Page Main

### Controller Configuration

#### ANALOG & DIGITAL

Description
DIO2 Encoder/Hall Effect Speed Sensor

1 2 3 4 5 6 7 8 9 10 11 12 SP

ANALOG

DIGITAL

MOTORS

UART1 UART2 I2C

ROBOT VEXNet GAME

**VEX**

#### INTEGRATED MOTOR ENCODERS

I2C #	Motor Port #	Description
1	1	
2	2	
3	3	
4	4	
5	5	
6	6	
7	7	
8	8	
9	9	
10	10	

#### MOTORS

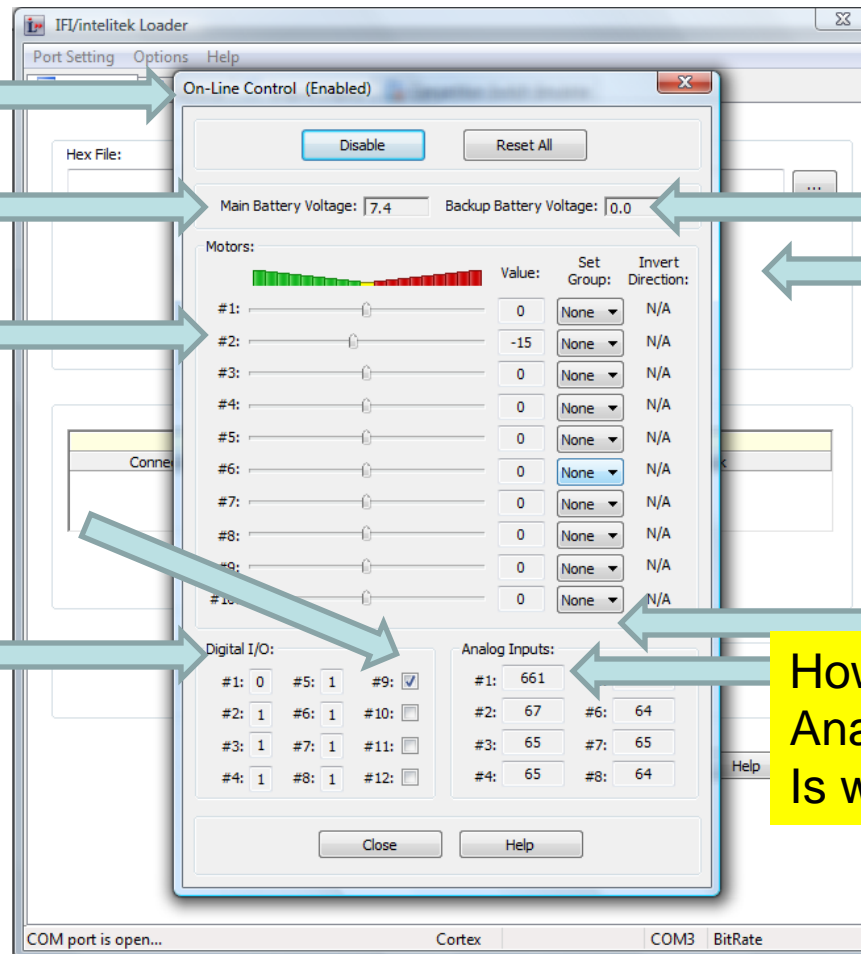
Motor Type	Description
1 n/a	
2 n/a	Motor2
3 n/a	
4 n/a	
5 n/a	
6 n/a	
7 n/a	
8 n/a	
9 n/a	
10 n/a	

#### Motor Type Information

n/a - Motor Type is not provided  
Standard - Motor Module without Integrated Encoder  
Small IME - 269 with Integrated Encoder  
Big IME - 393 with Integrated Encoder  
Big IME HS - 393 High Speed Gearing with Integrated Encoder

Left-Click to set Digital I/O Restore Defaults OK Cancel Help

# EasyC On-Line Control



How can we see if Motor's are wired up and determine Direction if positive or Negative?

How can we see if Encoder is working?

How can we see if Analog potentiometer Is working/



PWM

127

112

96

80

64

48

32

16

0

-16

-32

-48

-64

-80

-96

-112

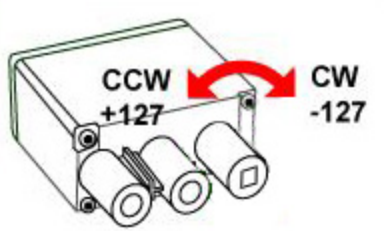
-127

Motor Module

Motor Number: 2 (Value Range: 1..10)  
// Motor2

Motor Direction (Value Range: -127..127):

Counter-clockwise ☐ 127  
Stop ☐ 0  
Clockwise ☐ -127  
User Value ☒ Motor2\_Output\_PWM



Code:  
SetMotor ( 2 , Motor2\_Output\_PWM ) ;

Comment:

F6 – Globals and Constants    Ctrl + F6 – Local Variables

OK Cancel Help

-127 -112 -96 -80 -64 -48 -32 -16 0 16 32 48 64 80 96 112 127

## Motor2 Output PWM vs Measured Motor2 rpm

Motor 2 Output Speed (rpm)

2500

Motor 2 PWM of 35 produces about 2500 rpm

$$y = 73.468x - 66.198$$

$$y = 73.468x - 66.198$$
$$R^2 = 0.9912$$

$$y = mx + b$$

$$m = 73.468$$

$$b = -66.198$$

37 ~ 2650 rpm  
35 ~ 2500 rpm  
33 ~ 2350 rpm

2000

1500

1000

500

0

Motor rpm

Linear (Motor rpm)

Each Motor 2 PWM unit produces about 75 rpm

5

10

15

20

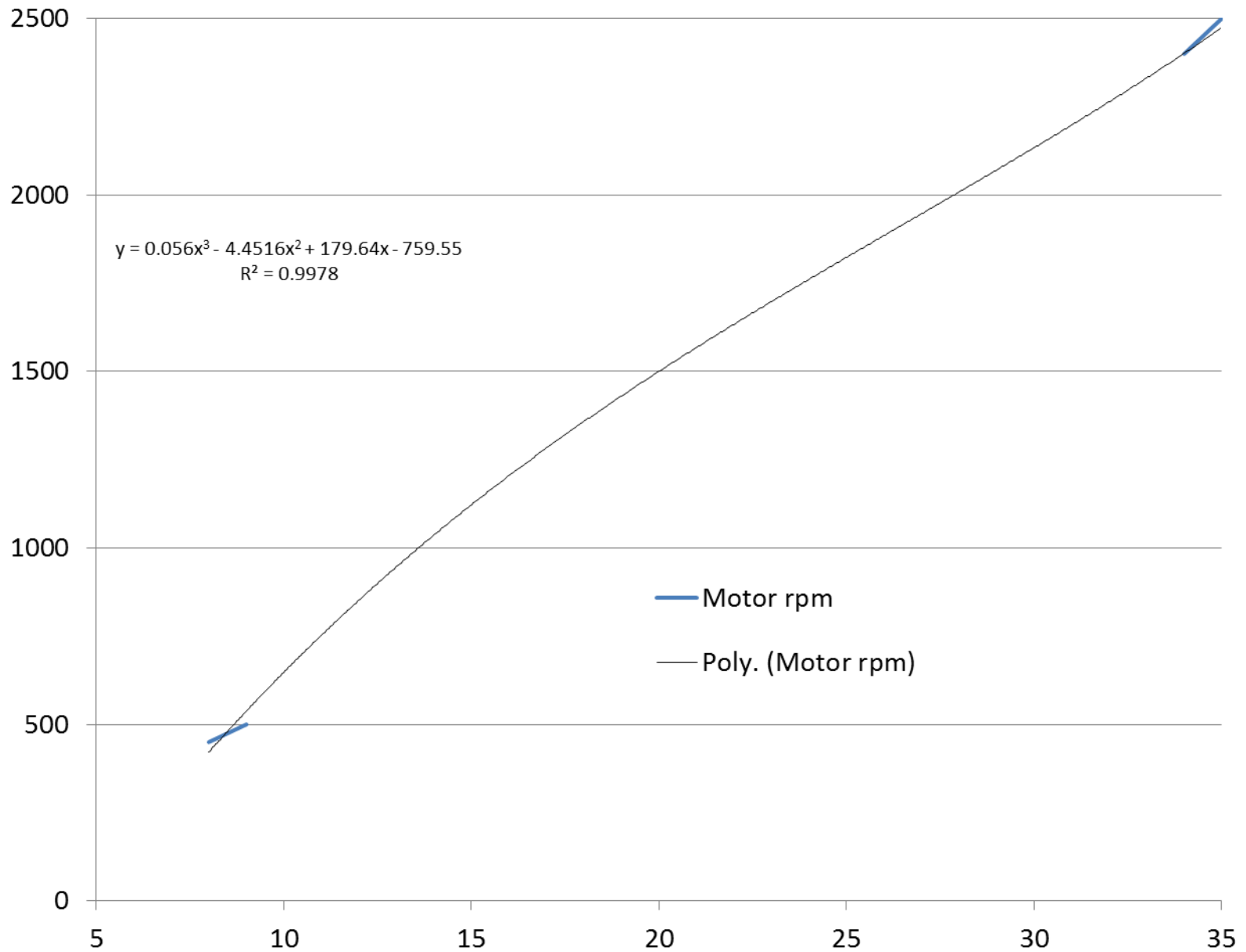
25

30

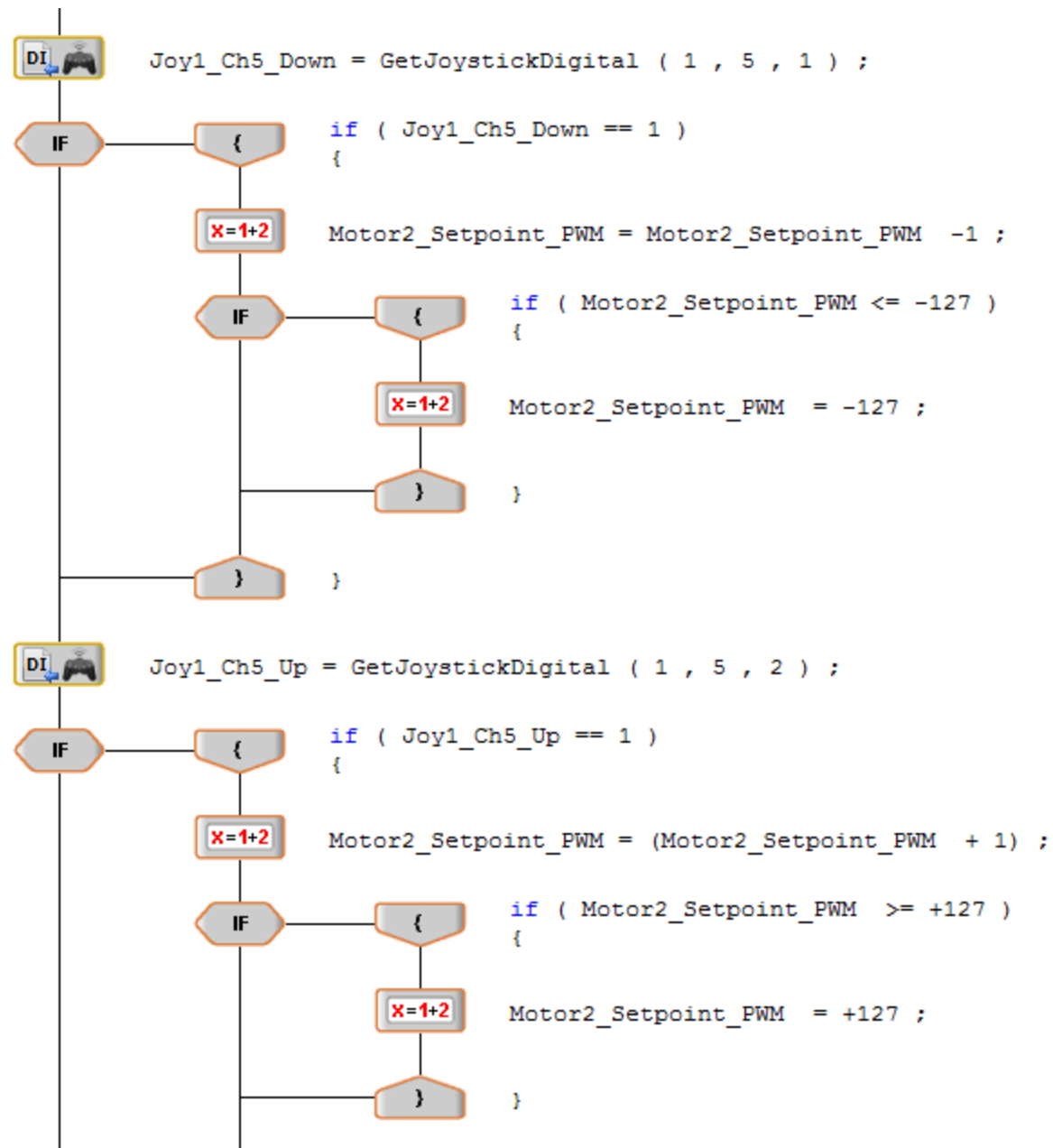
35

Motor 2 Output PWM

## Motor2 Output PWM vs Measured Motor2 rpm



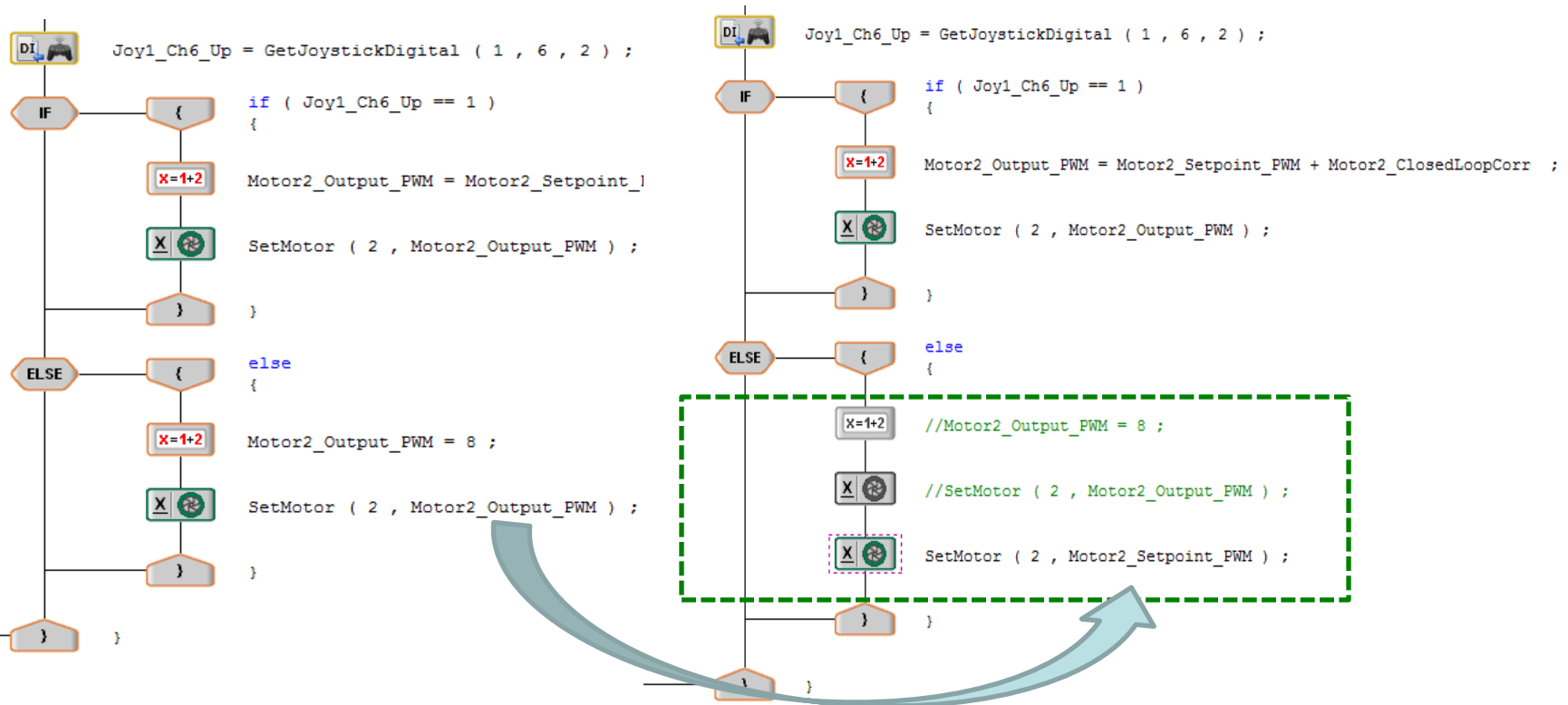
EasyC V4  
for Cortex  
- Program  
allows  
manual  
speed  
control of  
Motor 2  
using  
Joystick



This logic modifies the Motor2\_Setpoint\_PWM Cortex variable



# EasyC V4 for Cortex - Program allows manual speed control of Motor 2 using Joystick



This program change outputs the Motor2\_Setpoint\_PWM variable

You are here: [Home](#) >> [Electromechanical](#) >> [Sensors](#) >> [More Products](#)

## Geartooth Speed Sensor 4-Pin

Jameco Part no. 512401

Manufacturer [CHERRY CORPORATION](#)

Manufacturer no. GS100701

[Catalog 132 , page 76](#)

[Data Sheet \(current\) \[136 KB\]](#)



[View Larger Image](#)  
Image is representative only



### Pricing & Availability

**\$29.95**

# of Unit	Price
1+	\$29.95
5+	\$26.95
25+	\$23.95

Availability: Ship today

+ [Add to my favorites](#)

Quantity

ADD TO CART

### Overview

### Specifications

### Related Products

Specification	Value
Family	GS100701
Product Length	25mm
Product Type	Sensor Misc
Category	Geartooth Speed Sensor
Minimum Operating Temperature	-40°C

Specification	Value
Maximum Operating Temperature	125°C
Mounting	Panel
Pin_Count	4

[Report a problem](#)  
[Suggest a product](#)

### RELATED PRODUCTS

PICSTART PLUS AND MPLAB  
COMPATIBLE PROGRAMMER



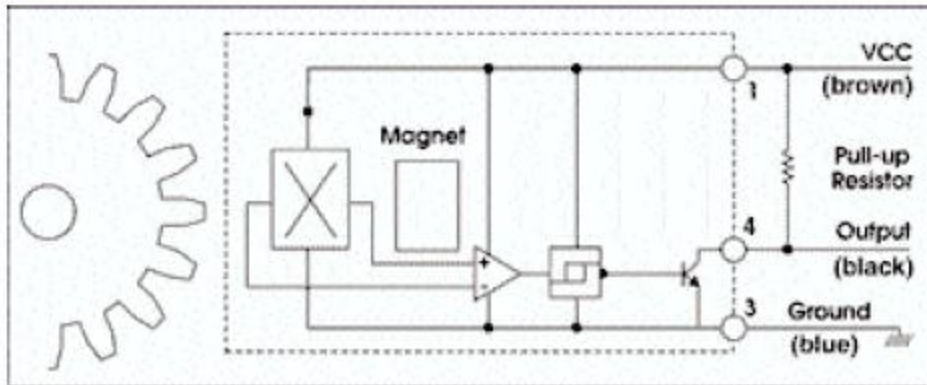
[Buy Now \\$169.95](#)

12V SPST Illuminated LED Pushbutton



[Buy Now \\$4.95](#)

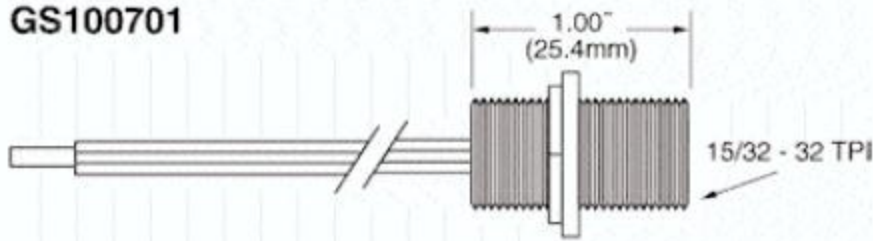
## OPEN COLLECTOR SINKING BLOCK DIAGRAM



Use 1 kilohm resistor for 5V systems like the Cortex V4

## DIMENSIONS

GS100701



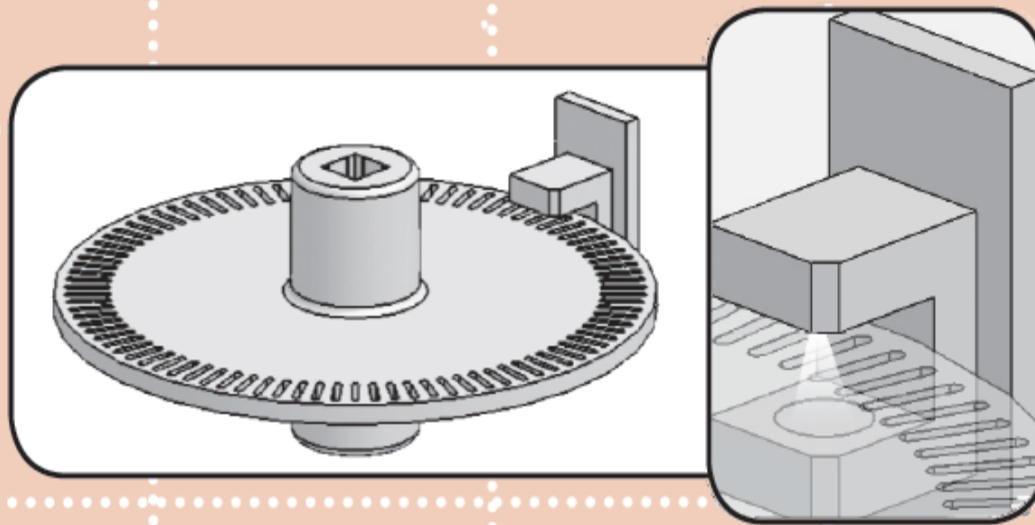
A resistor must be added between the brown and black wire in order For the Cortex system to read the hall effect sensor signals

# Sensor Accessories

## Optical Shaft Encoder (2-pack), continued

### 1 Technical Overview

The Optical Shaft Encoder uses an infrared light sensor to detect illumination from an infrared LED passing through slots cut in the circumference of a rotating wheel.



We will be using the EasyC commands for the optical shaft encoder to read the Digital Hall Effect Speed Sensor (Trick!)



## easyC V4 for Cortex ♦ Function Blocks ♦ Optical Shaft Encoder



The Optical Encoder function block allows the user to define behavior based on signals from a simple (non quadrature) Optical Encoder, such as the one found in Vex.

An Encoder is a [digital sensor](#) whose signals can be used to measure and define rotational movement. The encoder uses a disk with equally-spaced slots around its outer edge. An infrared light sensor aimed toward the disk will generate a digital signal based on whether its light hits the disk or passes through one of the slots in the disk. When the encoder senses the disk, it generates a digital signal of 1; if it senses a slot, it generates a 0 (zero) signal.

As the disk rotates, the encoder generates a string of signals (i.e. 0101010). The encoder can use these signals to count and record the number of slots read, thereby reading the amount of rotation. A simple optical encoder can count in only one direction, and signals generated while running in reverse will be indistinguishable from the forward direction.

When you drag an Optical Encoder block into the [Programming Window](#), the Optical Encoder dialog box appears, allowing you to choose one of four commands for the encoder function to process:

**Optical Encoder**

Select command:

☐ Start  
☐ Preset  
☒ Get  
☐ Stop

Interrupt Port #:  (Value Range: 1..12)

Retrieve to:  (Returns 'unsigned long')

Code:

```
local_variable = GetEncoder ( 1 );
```

Comment:

F6 – Globals and Constants    Ctrl + F6 – Local Variables

The [Start option](#) tells the encoders to start counting encoder pulses.

The [Preset option](#) allows the user to set the encoder to a fixed value or to the value of a variable.

The [Get option](#) allows the user to store the feedback from the encoder into a variable (the default type to return is long).

The [Stop option](#) tells the encoders to stop counting encoder pulses.

- Select the Interrupt Port number that corresponds with the port the encoder is plugged into on the controller.
- Select a [predefined variable](#) in the "Retrieve to" field. Alternatively, you can enter your own variable name now, in which case you must still define the variable as either local or global prior to compiling. You may hit either F6, or Ctrl + F6 at any time to open the [variable definition](#) windows.
- A description field lies below each combo box. The description field will display [comments and descriptions](#) from the selection above, should any exist.
- In the Code area, view a preview of the C code that will be generated with the defined properties.
- In the Comment field, enter comments that will help you read the program and understand the function of the block without knowing all the properties defined by the block.
- To see a sample program demonstrating how the Optical Shaft Encoder is used, open the "Encoder Test.ECPX" Project located in the Samples folder.

# EasyC V4 for Cortex - Use the encoder functions to count digital Hall Effect Sensor pulses for calculating RPM

The screenshot shows the EasyC V4 for Cortex software interface. The main window displays a program flow diagram with the following steps:

- Config
- Globals
- BEGIN
- Variables
- StartEncoder ( 2 ) ;
- StartTimer ( 1 ) ;
- StartTimer ( 2 ) ;
- StartTimer ( 3 ) ;
- StartTimer ( 4 ) ;
- WHILE { while ( 1 == 1 ) {

A "Local Variables" dialog box is open, showing a table of variables:

#	Type	Name	Value	Comment
1	unsigned long	Timer1	0	Initial Value
2	unsigned long	Timer2	0	
3	unsigned long	Timer3_Cnt	0	
4	unsigned long	Encoder	0	Initial Value
5	int	Joy1_Ch5_Down	0	Initial Value
6	int	Joy1_Ch5_Up	0	Initial Value
7	int	Joy1_Ch6_Up	0	
8	int	Motor2_Setpoint_PWM	35	2500 rpm open-loop command
9	int	Motor2_Total_Cnt	0	Initial Value
10	int	Motor2_LoopCtr	0	

## Digital Hall Speed Sensor conversion to RPM

1200	rev	X	<del>6 pulses</del>	X	<del>1 min</del>	=	<del>120 pulses</del>
1	min		1 rev		60 sec		second

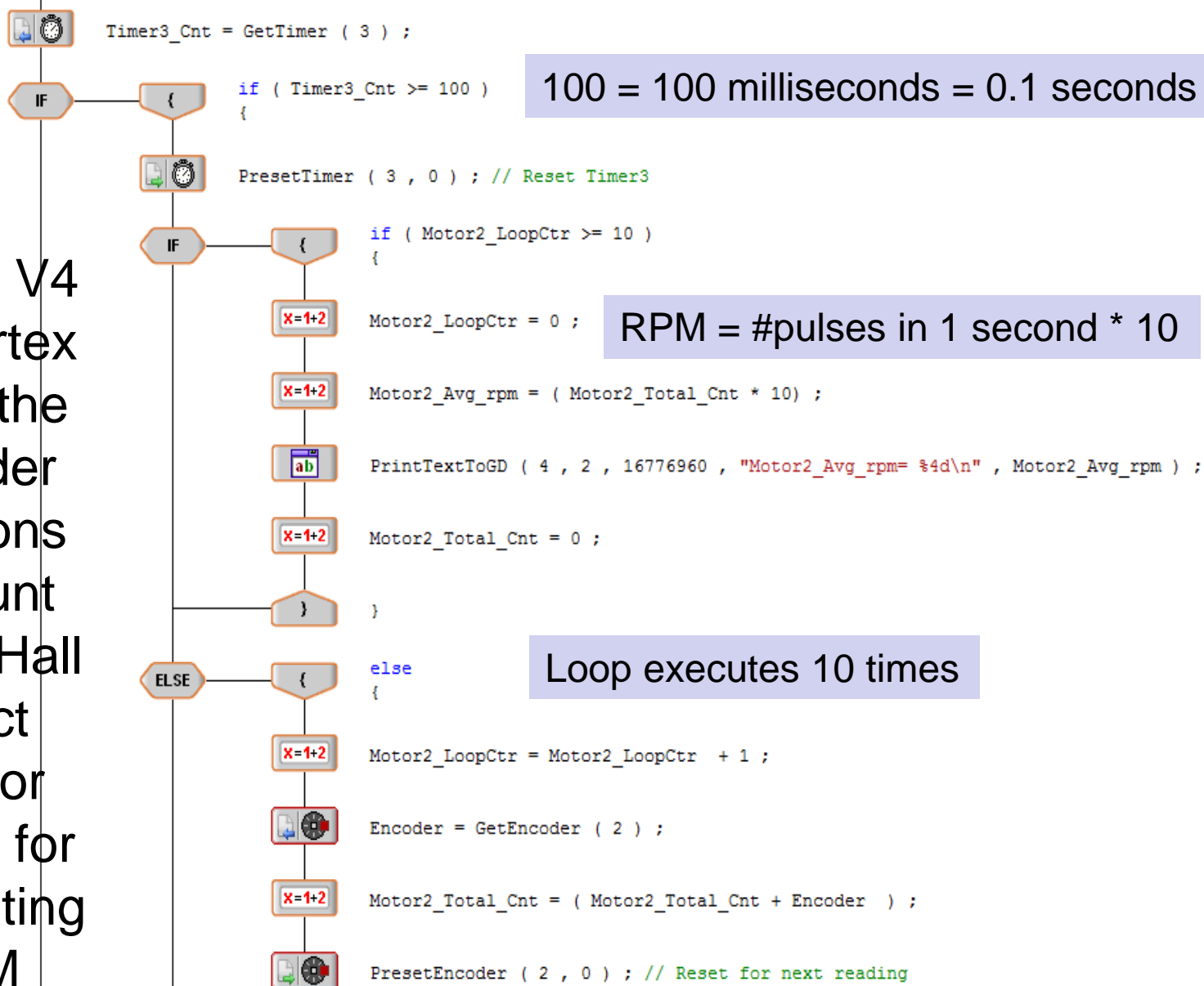
The number of pulses counted in one second multiplied by 10 equals the RPM

The EasyC program counts pulses ten times per second (100 milliseconds) to calculate RPM

The EasyC program counts pulses four times per second (250 milliseconds) to calculate RPM

The number of pulses counted in 250 msec multiplied by (10 \* 4) equals the RPM

EasyC V4  
for Cortex  
- Use the  
encoder  
functions  
to count  
digital Hall  
Effect  
Sensor  
pulses for  
calculating  
RPM





Motor2\_Setpoint\_PWM = 35(40)

Measured rpm (2555) - Setpoint rpm (2500) = Error rpm (55)

Closed-Loop (Prop)ortional (Corr)ection = -[Integer (55/32)] = -[1]

Integer

Closed-Loop (Int)egral (Corr)ection = -[1]

Closed-Loop (Corr)ection (-2) = Closed-Loop PropCorr + Closed-Loop IntCorr

Motor2\_Output\_PWM (33) = Motor2\_Setpoint\_PWM (35) + Closed\_Loop Corr (-2)

The PWM output command has been reduced from 35 to 33 due to reduce speed

Closed-Loop Correction Calculation					2500 +/- 10%	
				32		
Low	High	Low	High	divide by 32	Setpoint RPM = 2500	
Measured	Measured	Error	Error	Closed-Loop	Closed-Loop	Closed-Loop
RPM	RPM	RPM	RPM	Prop Corr	Integral Corr	Correction
				Integer		
2500	2531	0	31	0	-1	-1
2532	2563	32	63	-1	-1	-2
2564	2595	64	95	-2	-1	-3
2596	2627	96	127	-3	-1	-4
2628	2659	128	159	-4	-1	-5
2660	2691	160	191	-5	-1	-6
2692	2723	192	223	-6	-1	-7
2724	2755	224	255	-7	-1	-8
2756	2787	256	287	-8	-1	-9
2788	2819	288	319	-8	-1	-9
2820	2851	320	351	-8	-1	-9
2852	2883	352	383	-8	-1	-9
2884	2915	384	415	-8	-1	-9
Since the Measured RPM is higher than the Setpoint RPM, the PWM command sent to Motor 2 must be reduced					2500	

Closed-Loop Correction Calculation					2500 +/- 10%	
				32		
Low	High	Low	High	divide by 32	Setpoint RPM = 2500	
Measured	Measured	Error	Error	Closed-Loop	Closed-Loop	Closed-Loop
RPM	RPM	RPM	RPM	Prop Corr	Integral Corr	Correction
				Integer		
2500	2469	0	-31	0	0	0
2468	2437	-32	-63	1	1	2
2436	2405	-64	-95	2	1	3
2404	2373	-96	-127	3	1	4
2372	2341	-128	-159	4	1	5
2340	2309	-160	-191	5	1	6
2308	2277	-192	-223	6	1	7
2276	2245	-224	-255	7	1	8
2244	2213	-256	-287	8	1	9
2212	2181	-288	-319	8	1	9
2180	2149	-320	-351	8	1	9
2148	2117	-352	-383	8	1	9
2116	2085	-384	-415	8	1	9
Since the Measured RPM is lower than the Setpoint RPM,					2500	
the PWM command sent to Motor 2 must be increased						

easyC V4 for Cortex - Joystick Project (WiFi) - Workshop 2 Calculate RPM program 19AUG13

File Edit View Project Build and Download Tools Window Help

0.9

Function Blocks

Program Flow

Inputs

Outputs

Motor Module

Servo Module

Digital Output

Integrated Motor Encoders

Joystick

LCD

Battery

Mathematics

Smart Tasks

User Functions

Start Page

Main

Project Explorer

Controller Configuration

Macros and Constants

Global Variables

Block Diagram

Main

Source Files

Header Files

API.h

UserInclude.h

Library Files

ReadMe.txt

Timer1 = GetTimer ( 1 ) ;

IF { if ( Timer1 >= 100 ) // 100 msec motor control loop {

PresetTimer ( 1 , 0 ) ;

//StartTimer ( 1 ) ;

IF { if ( Motor2\_Avg\_rpm >= Motor2\_Setpoint\_rpm ) {

X-1+2 Motor2\_Error\_rpm = Motor2\_Avg\_rpm - Motor2\_Setpoint\_rpm ;

X-1+2 Motor2\_PropCorr = -( Motor2\_Error\_rpm / 32 ) ;

IF { if ( Motor2\_PropCorr <= - 8 ) {

X-1+2 Motor2\_PropCorr = -8 ; // RPM over setpoint, reduce PWM (speed)

}

X-1+2 Motor2\_IntCorr = -1 ;

}

ELSE { else {

X-1+2 Motor2\_Error\_rpm = Motor2\_Setpoint\_rpm - Motor2\_Avg\_rpm ;

X-1+2 Motor2\_PropCorr = ( Motor2\_Error\_rpm / 32 ) ;

IF { if ( Motor2\_PropCorr >= 8 ) {

X-1+2 Motor2\_PropCorr = 8 ; // RPM over setpoint, reduce PWM (speed)

}

X-1+2 Motor2\_IntCorr = +1 ;

}

X-1+2 Motor2\_ClosedLoopCorr = (Motor2\_PropCorr + Motor2\_IntCorr) ;

}

Output & Tasks

Ready

CAP NUM STM32F103VD Program size: 100%

Meeting Reminder

Electric Pump Motor Cable P...

02:00 PM - 03:00 PM

Merritt J Morris (Host)

Starts in 12 mins.

Join

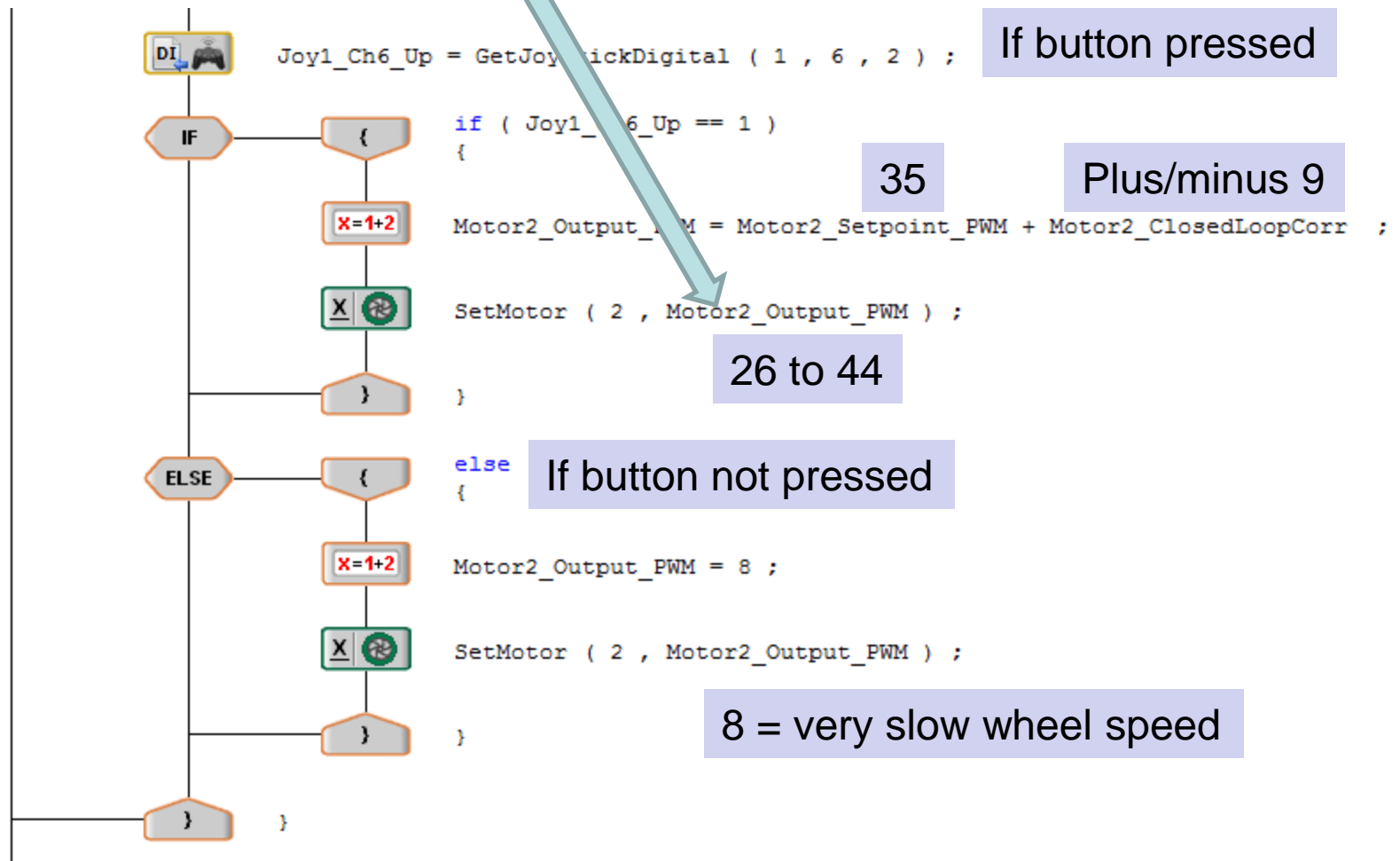
16.66 x 12.50 in

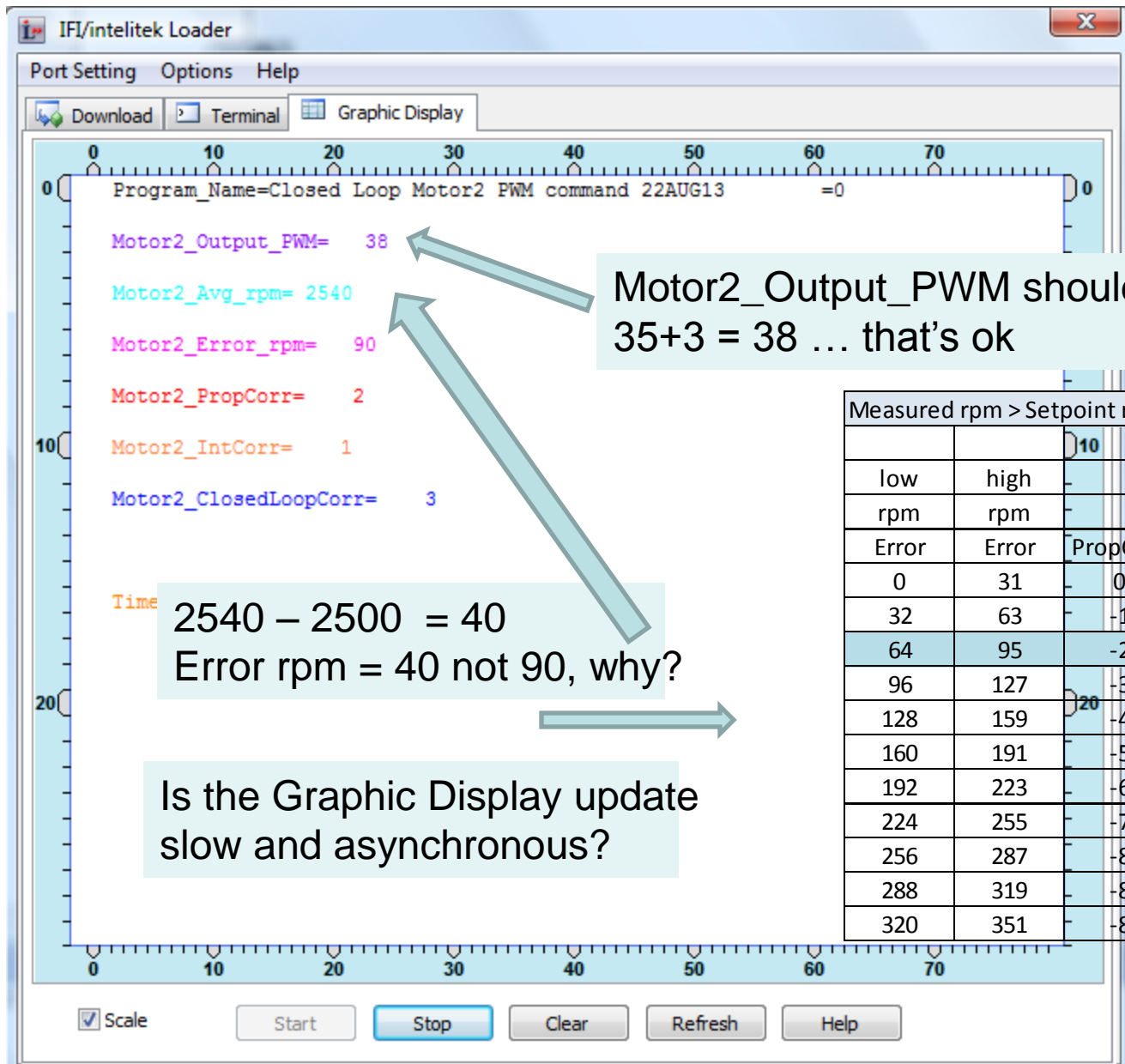
easyC V4 for ... intelitek easy... Untitled - Pai... Cortex encod...

1:48 PM 8/20/2013

The Motor2\_Output\_PWM controls the Motor2 Speed

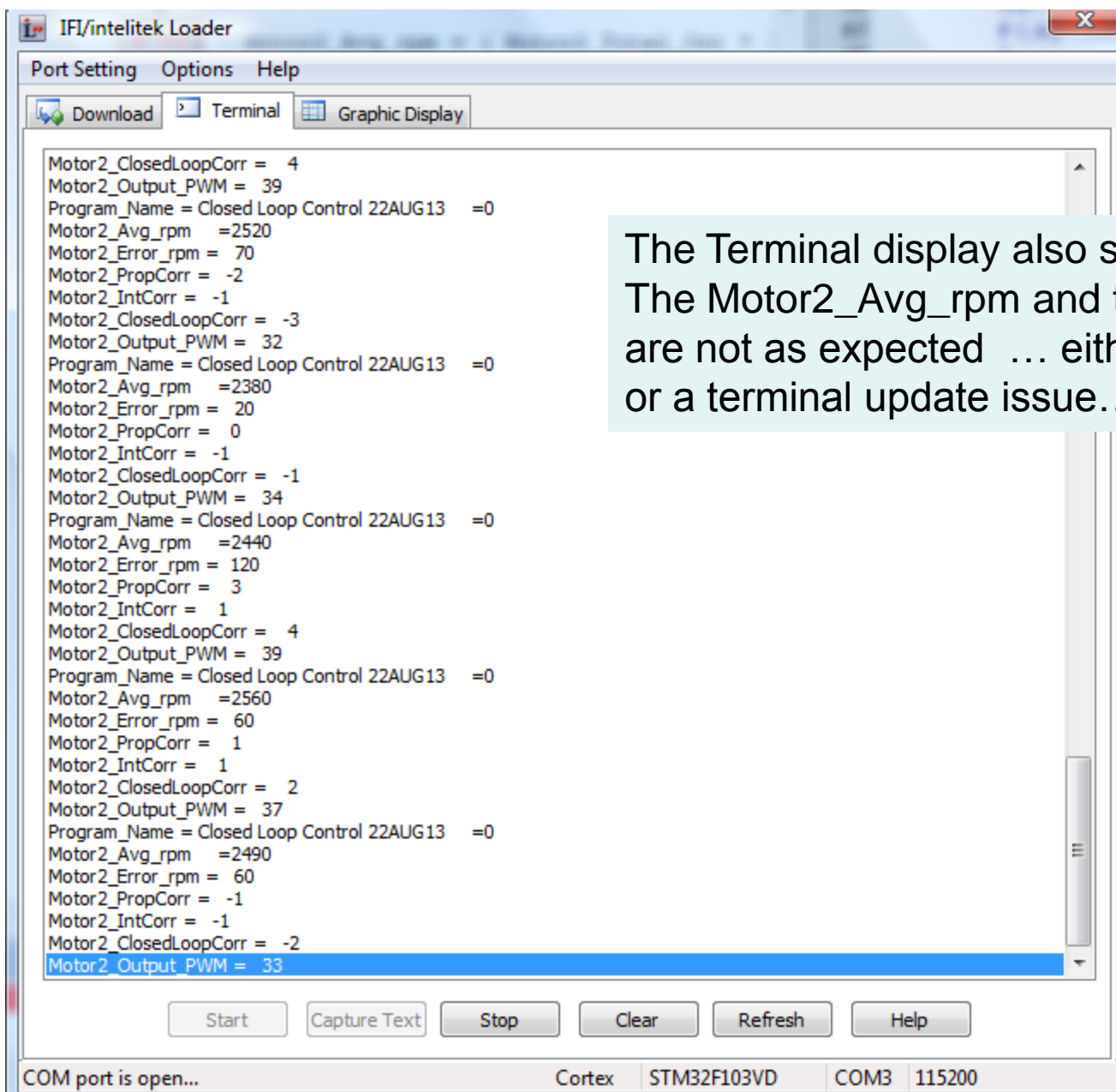
The Motor2\_Output\_PWM is made equal to the Motor2\_Setpoint\_PWM (which is nominally 35 to provide around 2500 rpm) and the Motor2\_CloseLoopCorr which is equal to the PropCorr and IntCorr





Measured rpm > Setpoint rpm			
		10	
low	high		
rpm	rpm		
Error	Error	PropCorr	IntCorr
0	31	0	-1
32	63	-1	-1
64	95	-2	-1
96	127	-3	-1
128	159	-4	-1
160	191	-5	-1
192	223	-6	-1
224	255	-7	-1
256	287	-8	-1
288	319	-8	-1
320	351	-8	-1





The screenshot shows the 'IFI/intelitek Loader' application window. The 'Terminal' tab is active, displaying a series of motor control parameters. The parameters are listed in a repeating pattern, suggesting multiple cycles of data collection. The parameters include Motor2\_ClosedLoopCorr, Motor2\_Output\_PWM, Program\_Name, Motor2\_Avg\_rpm, Motor2\_Error\_rpm, Motor2\_PropCorr, and Motor2\_IntCorr. The values for Motor2\_Avg\_rpm and Motor2\_Error\_rpm vary across the cycles, indicating fluctuations in motor performance. The status bar at the bottom indicates 'COM port is open...', 'Cortex', 'STM32F103VD', 'COM3', and '115200'.

```
Motor2_ClosedLoopCorr = 4
Motor2_Output_PWM = 39
Program_Name = Closed Loop Control 22AUG13  =0
Motor2_Avg_rpm  =2520
Motor2_Error_rpm = 70
Motor2_PropCorr = -2
Motor2_IntCorr = -1
Motor2_ClosedLoopCorr = -3
Motor2_Output_PWM = 32
Program_Name = Closed Loop Control 22AUG13  =0
Motor2_Avg_rpm  =2380
Motor2_Error_rpm = 20
Motor2_PropCorr = 0
Motor2_IntCorr = -1
Motor2_ClosedLoopCorr = -1
Motor2_Output_PWM = 34
Program_Name = Closed Loop Control 22AUG13  =0
Motor2_Avg_rpm  =2440
Motor2_Error_rpm = 120
Motor2_PropCorr = 3
Motor2_IntCorr = 1
Motor2_ClosedLoopCorr = 4
Motor2_Output_PWM = 39
Program_Name = Closed Loop Control 22AUG13  =0
Motor2_Avg_rpm  =2560
Motor2_Error_rpm = 60
Motor2_PropCorr = 1
Motor2_IntCorr = 1
Motor2_ClosedLoopCorr = 2
Motor2_Output_PWM = 37
Program_Name = Closed Loop Control 22AUG13  =0
Motor2_Avg_rpm  =2490
Motor2_Error_rpm = 60
Motor2_PropCorr = -1
Motor2_IntCorr = -1
Motor2_ClosedLoopCorr = -2
Motor2_Output_PWM = 33
```

Start Capture Text Stop Clear Refresh Help

COM port is open... Cortex STM32F103VD COM3 115200

The Terminal display also shows that The Motor2\_Avg\_rpm and the Motor2\_Error\_rpm are not as expected ... either a calculation issue or a terminal update issue...stay tuned

IFI/intelitek Loader

Port Setting

Options

Help

Download

Terminal

Graphic Display

```

Motor2_ClosedLoopCorr = 5
Motor2_Output_PWM = 40
Program_Name = Closed Loop Control 22AUG13  =0
Motor2_Avg_rpm  =2400
Motor2_Error_rpm = 100
Motor2_PropCorr = 3
Motor2_IntCorr = 1
Motor2_ClosedLoopCorr = 4
Motor2_Output_PWM = 39
Program_Name = Closed Loop Control 22AUG13  =0
Motor2_Avg_rpm  =2410
Motor2_Error_rpm = 90
Motor2_PropCorr = 2
Motor2_IntCorr = 1
Motor2_ClosedLoopCorr = 3
Motor2_Output_PWM = 38
Program_Name = Closed Loop Control 22AUG13  =0
Motor2_Avg_rpm  =2370
Motor2_Error_rpm = 130
Motor2_PropCorr = 4
Motor2_IntCorr = 1
Motor2_ClosedLoopCorr = 5
Motor2_Output_PWM = 40
Program_Name = Closed Loop Control 22AUG13  =0
Motor2_Avg_rpm  =2360
Motor2_Error_rpm = 140
Motor2_PropCorr = 4
Motor2_IntCorr = 1
Motor2_ClosedLoopCorr = 5
Motor2_Output_PWM = 40
Program_Name = Closed Loop Control 22AUG13  =0
Motor2_Avg_rpm  =2400
Motor2_Error_rpm = 100
Motor2_PropCorr = 3
Motor2_IntCorr = 1
Motor2_ClosedLoopCorr = 4
Motor2_Output_PWM = 39

```

Start

Capture Text

Stop

Clear

Refresh

Help

RPM issue fixed by moving calculation update after the print statements

## Screen Shots of Closed-Loop System Output

		Measured rpm > Setpoint rpm		2500	
low	high	low	high		
rpm	rpm	rpm	rpm		
Measured	Measured	Error	Error	PropCorr	IntCorr
2500	2531	0	31	0	-1
2532	2563	32	63	-1	-1
2564	2595	64	95	-2	-1
2596	2627	96	127	-3	-1
2628	2659	128	159	-4	-1
2660	2691	160	191	-5	-1
2692	2723	192	223	-6	-1
2724	2755	224	255	-7	-1
2756	2787	256	287	-8	-1
2788	2819	288	319	-8	-1
2820	2851	320	351	-8	-1

Motor2\_Setpoint RPM = 35, soon to be 40

Not going over 2500 rpm – needs more PWM – 12V battery weak?

Divide by 16 instead of 32 because speed was not going over 2500 rpm

```
Motor2_ClosedLoopCorr = -8
Motor2_Output_PWM = 27
Program_Name = Closed Loop Control 22AUG13  =0
Motor2_Avg_rpm  =2250
Motor2_Error_rpm = 250
Motor2_PropCorr = 15
Motor2_IntCorr = 1
Motor2_ClosedLoopCorr = 16
Motor2_Output_PWM = 51
Program_Name = Closed Loop Control 22AUG13  =0
Motor2_Avg_rpm  =2480
Motor2_Error_rpm = 20
Motor2_PropCorr = 1
Motor2_IntCorr = 1
Motor2_ClosedLoopCorr = 2
Motor2_Output_PWM = 37
Program_Name = Closed Loop Control 22AUG13  =0
Motor2_Avg_rpm  =2560
Motor2_Error_rpm = 60
Motor2_PropCorr = -3
Motor2_IntCorr = -1
Motor2_ClosedLoopCorr = -4
Motor2_Output_PWM = 31
Program_Name = Closed Loop Control 22AUG13  =0
Motor2_Avg_rpm  =2350
Motor2_Error_rpm = 150
Motor2_PropCorr = 9
Motor2_IntCorr = 1
Motor2_ClosedLoopCorr = 10
Motor2_Output_PWM = 45
Program_Name = Closed Loop Control 22AUG13  =0
Motor2_Avg_rpm  =2270
Motor2_Error_rpm = 230
Motor2_PropCorr = 14
Motor2_IntCorr = 1
Motor2_ClosedLoopCorr = 15
Motor2_Output_PWM = 50
```

		Measured rpm > Setpoint rpm		2500	
low	high	low	high		
rpm	rpm	rpm	rpm		
Measured	Measured	Error	Error	PropCorr	IntCorr
2500	2485	0	-15	0	1
2484	2469	-16	-31	1	1
2468	2453	-32	-47	2	1
2452	2437	-48	-63	3	1
2436	2421	-64	-79	4	1
2420	2405	-80	-95	5	1
2404	2389	-96	-111	6	1
2388	2373	-112	-127	7	1
2372	2357	-128	-143	8	1
2356	2341	-144	-159	9	1
2340	2325	-160	-175	10	1
2324	2309	-176	-191	11	1
2308	2293	-192	-207	12	1
2292	2277	-208	-223	13	1
2276	2261	-224	-239	14	1
2260	2245	-240	-255	15	1
2244	2229	-256	-271	16	1

Start

Capture Text

Motor2\_Setpoint RPM = 35

COM port is open...

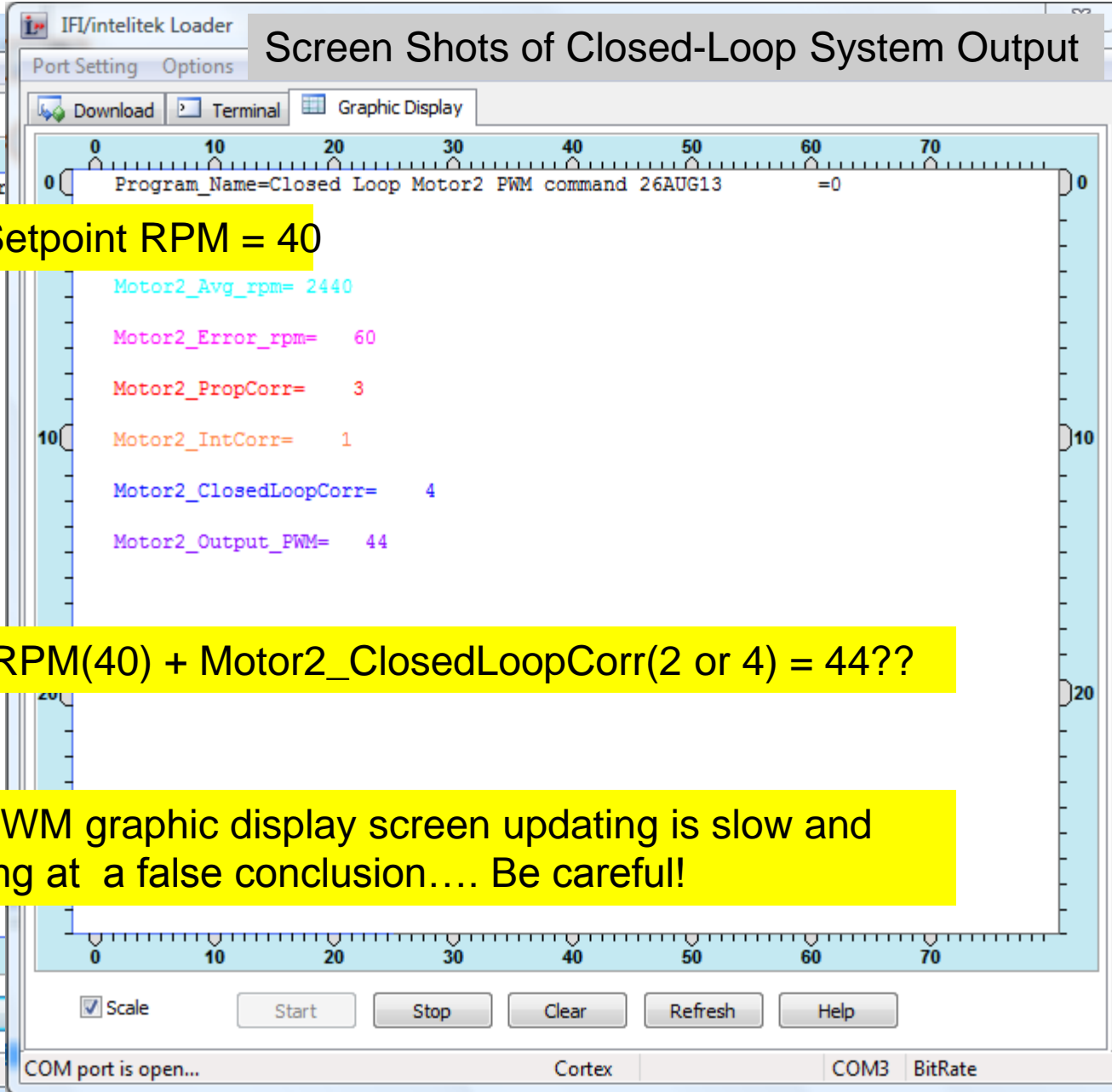
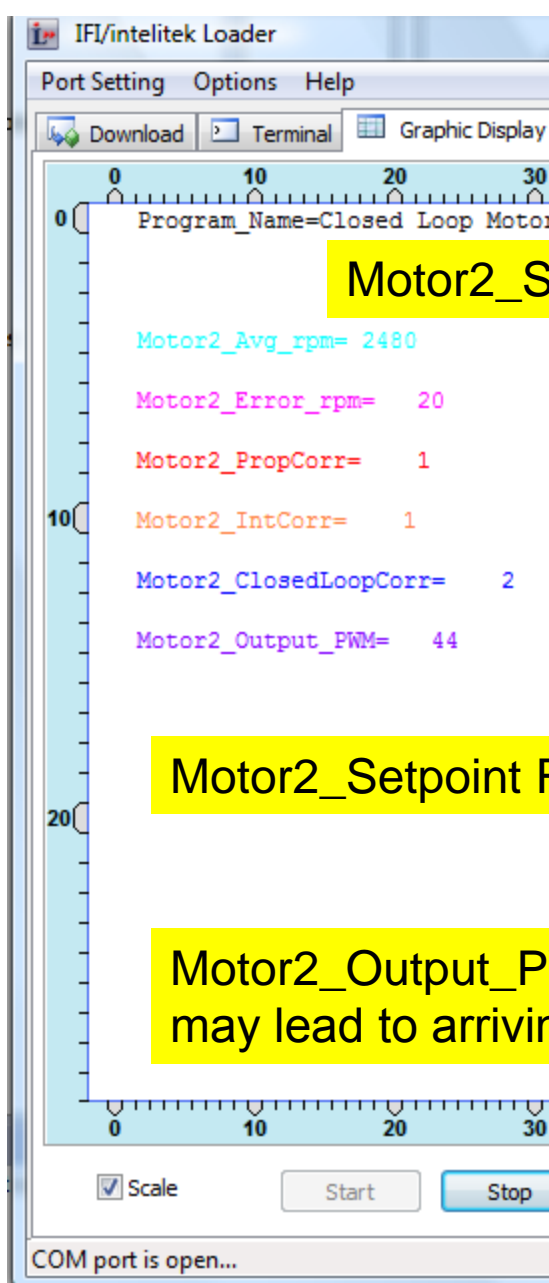
Cortex

STM32F103VD

COM3

115200

## Screen Shots of Closed-Loop System Output



# Screen Shots of Closed-Loop System Output

