ADC Resolution vs Sensor Less Motor Control Performance

Cristian Ornelas
Tyler Hawkins
Tamara Hussam A Basfar
John Adam King

Concept of Operations

REVISION – 3
05 December 2024

CONCEPT OF OPERATIONS
FOR
ADC Resolution vs Sensor Less Motor Control Performance

TEAM <1>

APPROVED BY:

_____
Project Leader Date

_____
Prof. Kalafatis Date

_____
T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
1	[9/15/2024]	[ADC Resolution vs Sensor Less Motor Control Performance]		Draft Release
2	[9/26/2024]	[ADC Resolution vs Sensor Less Motor Control Performance]		Second Revision
3	[12/04/2024]	[ADC Resolution vs Sensor Less Motor Control Performance]		Third Revision

Table of Contents

List of Figures

1.0 Executive Summary

The focus of our project is to develop software for the c2000 TI chip to implement sensorless field-oriented motor control for motor systems. A microcontroller will be used to estimate the position of the rotor instead of traditional sensors. To improve and optimize motor position estimation the ADC resolution will be increased. Testing will be conducted under various conditions such as different motor speeds to evaluate how improved ADC resolutions impact the accuracy of position estimation. To test accuracy results from the sensorless motor controller will be compared to an incremental encoder.

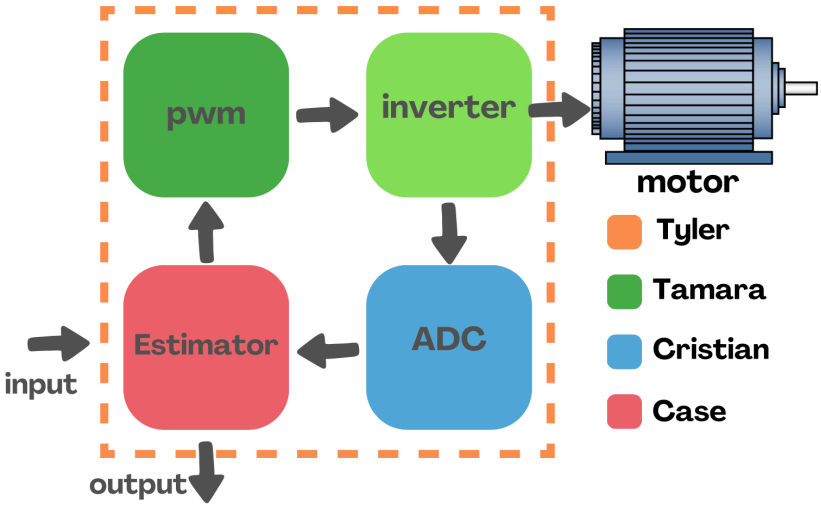


Figure 2. Simplified subsystem flow diagram

2.0 Introduction

2.1 Background

The most common motor design today is the Brushed DC Motor and the Brushless(BL) DC Motor. The brushed DC motor works by using mechanical logic and brushes to deliver electricity to the stator so that the motor spins, however, these brushes bend over time and reduce contact with the rotor. The BLDC motors often use Hall Effect sensors to detect where the rotor is, and this information is fed either into a microcontroller or analog circuit that will then control which magnets receive electricity in the stator. These motors are widely employed in all areas, but as more fields require precision motor controls, these designs have become insufficient.

The sensorless motor design aims to remove the need for sensors by using the feedback emf, electromotive force, of the motor and using this information to estimate the position of the rotor. This will allow the motor to operate with enhanced accuracy as an angle can be given for the orientation of the rotor as opposed to a region, and reduce the overall number of components needed for the motor to function.

2.2 Overview

The full system code will implement sensorless field-oriented control (FOC) of a motor. Our team will be provided hardware for the project, a motor/ motor controller, and a software development kit provided by Texas Instruments.

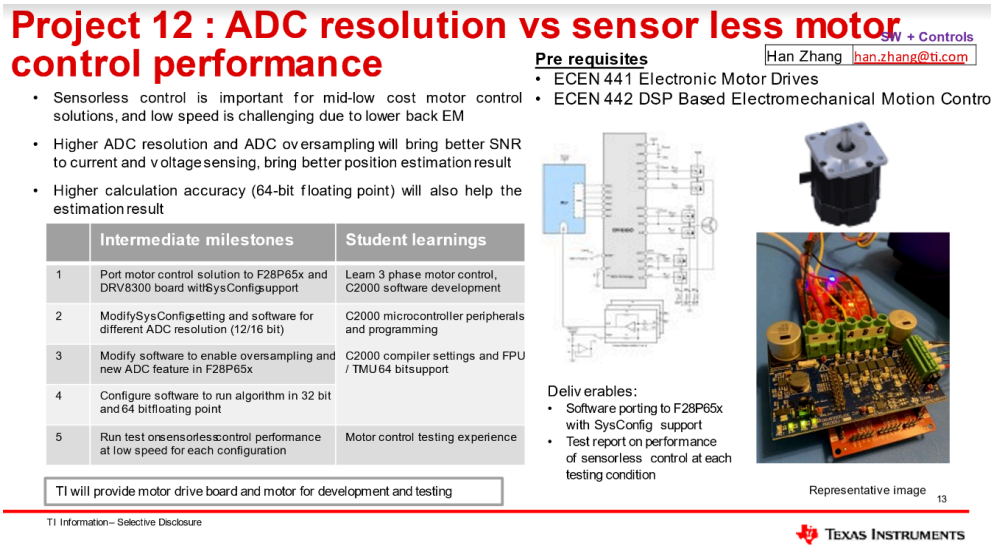


Figure 2. Project synopsis

To begin, we will port software to the F28P65x board with SysConfig support. Next, we will be implementing code to estimate the angle of the motor and load it onto the motor controller. To output results, our group will perform the first round of tests on the software that measures the motor operating with different ADC resolutions at low motor speeds. To be more precise, we will be performing additional tests on the software by enabling oversampling and 32/64-bit floating-point operators. The results will also be recorded at low motor speeds. Once this is completed, we will repeat our process at various motor speeds to capture a wide range of operating conditions that our sensorless motor could be in in the real world. Progress on developing the code and test results will be documented; this information is presented on the project as our final results.

The findings of this project should be the comparison between ADC resolutions and their effect on estimating the position accuracy. This will provide valuable insight into optimizing motor control for lower costs and how we can use this innovation to positively impact our environment and the intended users. Below is a breakdown of our team's subsystems:

Subsystem 1: PWM - Tamara
Subsystem 2: Estimator - Case
Subsystem 3: ADC Driver - Cristian
Subsystem 3: Porting - Tyler

2.3 Referenced Documents and Standards

1. *Motor Control Compendium*, https://www.ti.com/download/trng/docs/c2000/TI_MotorControlCompendium_2010.pdf Accessed 13 Sept. 2024.
2. "Analog-to-Digital Converter (ADC)#."*Analog-to-Digital Converter (ADC) - C28x Academy*, dev.ti.com/tirex/explore/content/c28x_academy_1_00_00_00/_build_c28x_academy_1_00_00_00/source/c2000_analog_subsystem/c2000_analog_to_digital_converter.html. Accessed 13 Sept. 2024.
3. "Sensor vs Sensorless Motor Controllers: A Head-to-Head Comparison." *Solo Motor Controllers*, 27 Jan. 2024, www.solomotorcontrollers.com/blog/sensor-vs-sensorless/?srsltid=AfmBOooTJsLnUA1zI53QF1GIPMYkBLSi7XJWOyLefSThTds0uwlZ2OQw.
4. *User's Guide Motor Control SDK Universal Project and Lab,* https://www.ti.com/lit/ug/spruj26a/spruj26a.pdf?ts=1727366836338&ref_url=https%253A%252F%252Fwww.google.com%252F
5. *DRV8300: 100-V Three-Phase BLDC Gate Driver datasheet (Rev. D),* https://www.ti.com/lit/ds/symlink/drv8300.pdf?ts=1729946514499
6. *TMS320F28002x Real-Time Microcontrollers datasheet (Rev. C),* https://www.ti.com/lit/ds/sprsp45c/sprsp45c.pdf?ts=1729987040570&ref_url=https%253A%252F%252Fwww.ti.com%252Ftool%252FLAUNCHXL-F280025C
7. *TMS320F28P65x Real-Time Microcontrollers datasheet (Rev. B),* https://www.ti.com/lit/ds/symlink/tms320f28p650dk.pdf?ts=1729638180938&ref_url=https%3A%2F%2Fwww.ti.com%2Fproduct%2FTMS320F28P650DK
8. *TMS320F28P65x Real-Time Microcontrollers Technical Reference Manual (Rev. B),* https://www.ti.com/lit/ug/spruiz1b/spruiz1b.pdf?ts=1731304485120&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FTMS320F28P650DK
9. *TMS320F28002x Real-Time Microcontrollers Technical Reference Manual (Rev. C),* https://www.ti.com/lit/ug/spruin7c/spruin7c.pdf?ts=1731455337533&ref_url=https%253A%252F%252Fwww.google.com%252F
10. *LAUNCHXL-F28P65X LanchPad Box Sticker and Insert,* https://www.ti.com/lit/ml/spaz022/spaz022.pdf?ts=1730999920772&ref_url=https%253A%252F%252Fwww.ti.com%252Ftool%252FLAUNCHXL-F28P65X
11. *LAUNCHXL-F28P65X Schematic,* https://www.ti.com/lit/df/sprr480/sprr480.pdf?ts=1729638767085
12. *LAUNCHXL-F280025C Schematic (Rev. A),* https://www.ti.com/lit/df/sprr425a/sprr425a.pdf?ts=1730854635311&ref_url=https%253A%252F%252Fwww.google.com%252F
13. *LAUNCHXL-F280025C Quick start guide,* https://www.ti.com/lit/ug/spruiw5/spruiw5.pdf

3.0 Operating Concept

3.1 Scope

The ADC sensorless motor control performance will be designed to fit the proper parameters for professionals to carry out the finished product. The milestones for the scope are listed below:

- Port motor control solution to F28p65x and DRV8300 board with SysConfig support
- Modify SysConfig setting and software for different ADC resolutions (12/16 bit)
- Modify the software to enable oversampling and new ADC feature in F28p65x
- Configure software to run an algorithm, in 32-bit and 64-bit floating point
- Run tests on sensorless control performance at low speed for each configuration

The focus of the project will be on the development and testing of the control system under different testing conditions.

3.2 Operational Description and Constraints

Sensorless Motor control allows a user to control a motor's operations without the use of any onboard sensors. The software created will be able to determine the motor's position, speed, and torque using in/out current and voltage. The following diagram shows the flow of software operations used to control the motor.

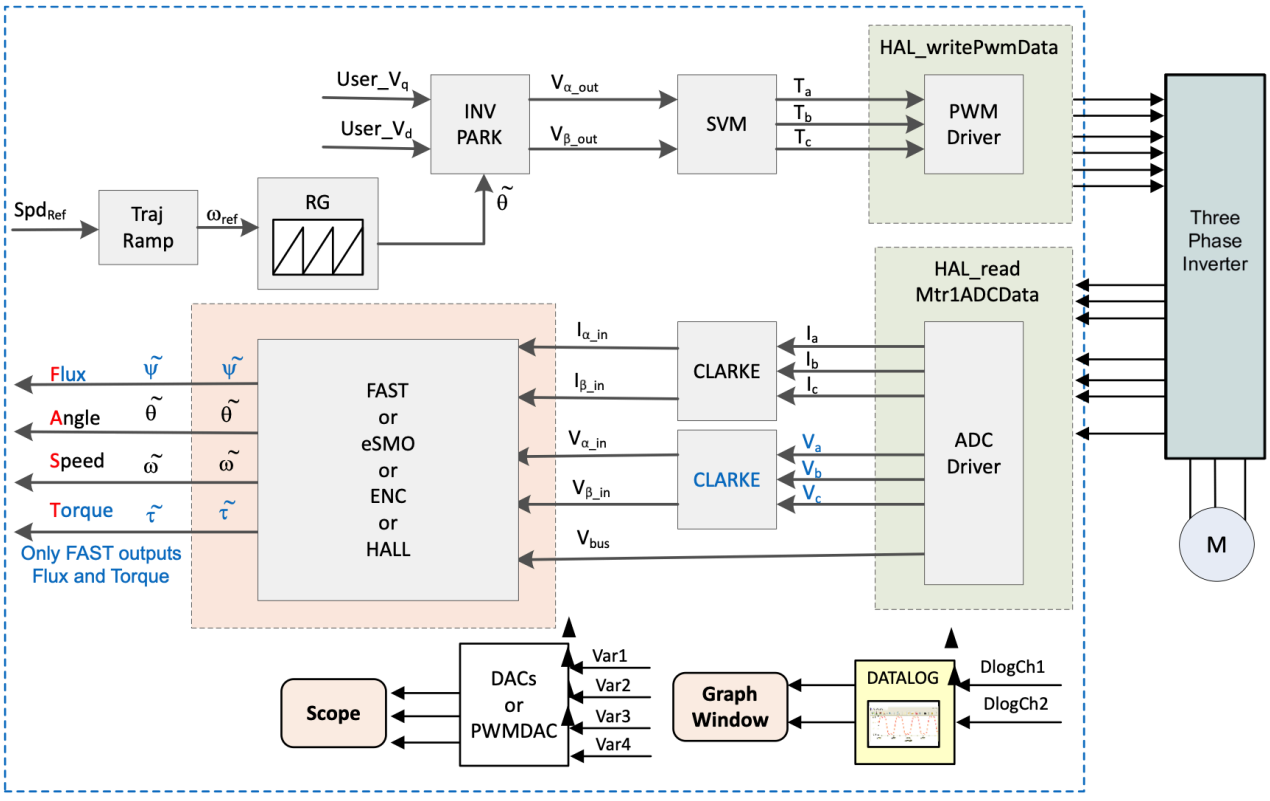


Figure 3. Block Diagram of System

The resulting constraints from this operational description are as follows:

- The Motor should be able to be controlled without any sensors
- The ability to control the motor using both 12 & 16-bit resolution
- The ability to control the motor using both 32 & 64-bit floating point operations
- All code used to control the motor should run on TI's F28P65x Launchpad board and the DRV8300 motor driver board.
- All software should be able to be configured using sysconfig

3.3 System Description

- Microcontroller and Estimater Code: The microcontroller contains the code that will take the input voltage and current to estimate the position of the rotor. The code itself will operate by taking these values, calculating using the back EMF feedback, then force on the motor, and then using this data in conjunction with its internal clock to calculate the angular orientation of the rotor. This data will then be fed back to the connected computer, as well as the motor controller. The resulting rotor angle and speed are used for real time control purposes.

- Motor Controller and control code: The motor controller will take the position data of the rotor and determine which poles should then be powered to provide a continuous force on the rotor. This subsystem includes a module that manages the motor operation at high speeds, a controller regulating phase currents to achieve desired torque and speed, and a module that converts the reference voltage vectors into PWM signals.

- Motor: The motor is being driven by the motor controller, but is also the driving force behind the calculations run in the microcontroller. The motor operates through electrical to mechanical conversion, converting signals from the inverter to mechanical rotation, the torque generated is directly influenced by the supplied currents. Another key part of its operation are the resulting electrical motor characteristics that are imputed back into the estimate creating a feedback loop that is essential for real-time position and speed control without sensors.

3.4 Modes of Operations

The software is designed to cooperate utilizing an input ADC resolution to control the size of each sample taken by the microcontroller when calculating the rotor position. ADC resolution refers to how many bits are used for each data point. While any input is theoretically possible, the system will be tested on 32-bit and 64-bit resolutions. Additionally, the motor will operate in closed or

open loop controls. In an open loop, the microcontroller will only use the BackEMF to determine the rotor position. In closed loop however, the previous estimation will be fed back into the estimation software and used in the next position test.

3.5 Users

The sensorless motor control system is intended for users who work with sensored motors and want to reduce the cost of production. Whether it's electric vehicle manufacturers who want to minimize complexity and increase reliability or HVAC engineers who wish to optimize fan motors, sensorless motors will revolutionize motor control. The intended users must have some intermediate to advanced knowledge of this system. They must understand basic motor control principles such as field-oriented control, programming languages like C++, error analysis, ADC resolution settings, etc.

3.6 Support

Support for the sensorless motor will be supplied in a user manual providing detailed instructions/ information on how to function the sensorless motor control system. There will also be troubleshooting guidance if the system doesn't work as intended.

4.0 Scenario(s)

4.1 Unfavorable/ Extreme Conditions

Without the need for any sensors, a sensorless controlled motor can operate much more reliably in harsher conditions. Motor sensors can often run into issues in environments whose variables frequently change. An example of this is frequent temperature fluctuations within an environment can often mess with the accuracy of a sensor. A sensor is also another point of failure for the motor within a harsh environment. Without the sensor, the motor is less likely to fail within these conditions.

4.2 Cheaper Alternative

Removing sensors from a motor within a device brings the cost down for that component. Examples commonly seen today include Electric transportation, such as E-bikes & scooters, Drones, and other robotic devices. A sensorless motor makes purchasing these devices more accessible to an everyday consumer, due to the cost reduction when compared to a sensored motor.

5.0 Analysis

5.1 Summary of Proposed Improvements

- Elimination of physical sensors for position estimation: This reduces hardware costs, and simplifies maintenance by reducing points of failure.
- Smaller motor designs: By eliminating physical sensors, the system enables more compact motor designs, saving space in applications.
- Increased flexibility and reliability: Sensorless motor control allows for a more flexible and reliable system due to fewer parts susceptible to damage; they have higher lifespans and can be used in harsh environments.
- Increasing ADC resolutions: an increased ADC resolution can potentially improve control speed precision which can improve performance at lower motor speeds.
- Optimal configuration identification: By testing the system at low and high speeds, the system will help identify optimal configurations for accurate position estimation and motor control.
- Environmental benefits: The system has the potential to contribute to energy-efficient designs and lower material costs, making it a more sustainable solution.

5.2 Disadvantages and Limitations

- Performance limitations at very low speeds due to lower back EMF
- sensorless motor controls use advanced algorithms which can complicate the control system needing a more sophisticated microprocessor.
- longer development time due to the need for calibration and testing.

5.3 Alternatives

- Hall effect motors: use hall effect sensors to detect the magnetic field from the rotor to estimate the position.
- Incremental motors: uses three wave pulses for position detection.
- Tachometer motor: uses speed sensors to estimate rotor position.

- Potentiometer motors: use a potentiometer to measure the angular displacement and estimate the position of the rotor.
- LVDTs: use linear differential transformers to sense linear displacement in some motors to estimate the rotor position.

These motor controllers have their advantages but are more costly and space-consuming.

5.4 Impact

The impact of this technology would be a reduction in the cost and material needed in precision motor applications, thus allowing more to be done for less. Overall, the impact on the environment and society would be quite small, though it would act as another point towards the idea of streamlining current technologies to use only the materials that they require.

ADC Resolution vs Sensor Less Motor Control Performance

Cristian Ornelas
Tyler Hawkins
Tamara Basfar
John Adam King

Interface Control Document

REVISION – 1
26 September 2024

Interface Control Document
FOR
ADC Resolution vs Sensor Less Motor Control Performance

Prepared by:

_____
Author Date

Approved by:

_____
Project Leader Date

_____
John Lusher II, P.E. Date

_____
T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
1	[9/26/24]	[ADC Resolution vs Sensor Less Motor Control Performance]		Draft Release
2	[12/04/2024]	[ADC Resolution vs Sensor Less Motor Control Performance]		Second Revision

Table of Contents

List of Tables

List of Figures

1. Overview

This Interface Control Document (ICD) defines the technical specifications and interface requirements for the ADC Resolution vs Sensor Less Motor Control Performance project. It outlines how the subsystems defined in the Concept of Operations report and the Functional System Requirements report will be implemented and integrated. The ICD includes detailed descriptions of reference documents, key definitions, and physical and electrical interfaces. The report outlines the physical interface specifications, such as weight, dimensions, and mounting locations, as well as the electrical interface, covering primary input power, polarity reversal protection, and signal interfaces. Additionally, it details the user control interface and communication protocols necessary for seamless integration between the host device and peripheral components. This ICD shows how all subsystems are going to be implemented and produced in accordance with the defined requirements, facilitating compatibility and seamless operation across the entire system.

2. References and Definitions

2.1 References

SBVS144C
Integrated MCU Power Solution for C2000 Microcontrollers
2012

IEEE-STD-1241
IEEE Standard for Terminology and Test Methods for Analog-to-Digital Converters
2001

v1.3
C2000 software guide
2021

SLVUBV6
DRV8300Dxxx-EVM User's Guide
2020

Motor Control Compendium

2011

Sensor vs Sensorless Motor Controllers: A Head-to-Head Comparison

2024

2.2 Definitions

CCS	Code Compiler Studio
mA	Milliamp
mW	Milliwatt
MHz	Megahertz (1,000,000 Hz)
TBD	To Be Determined
TTL	Transistor-Transistor Logic
VME	VERSA-Module Europe
PWM	pulse width modulation
ADC	Analog to Digital converter
INV	inverter
c2000	real-time microcontroller
SysConfig	development tool for c2000

3. Physical Interface

3.1 Weight

Component	Weight
C2000™ real-time MCU F28P65x LaunchPad™ development kit	1 oz
DRV8300DIPW evaluation module for three-phase BLDC	1 oz
Low Voltage Servo Motor - Low voltage servo (encoder) motor and wiring harness	23.1 oz
Total	25.1 oz

Table 1. Component's Weight

3.2 Dimensions

Component	Length	Width	Height
C2000™ real-time MCU F28P65x LaunchPad™ development kit	6.75 in.	2.3 in.	0.925 in.
DRV8300DIPW evaluation module for three-phase BLDC	3.5 in.	3.125 in.	0.75 in.
Low Voltage Servo Motor - Low voltage servo (encoder) motor and wiring harness	2.73 in.	2.73 in.	3.98 in.

Table 2. Component Dimensions

3.3 Mounting Locations

3.1.1 Mounting of the DRV8300Dxxx-EVM

The DRV8300Dxxx-EVM must plug into the lower LAUNCHXL-F280049C Launchpad headers as shown below.

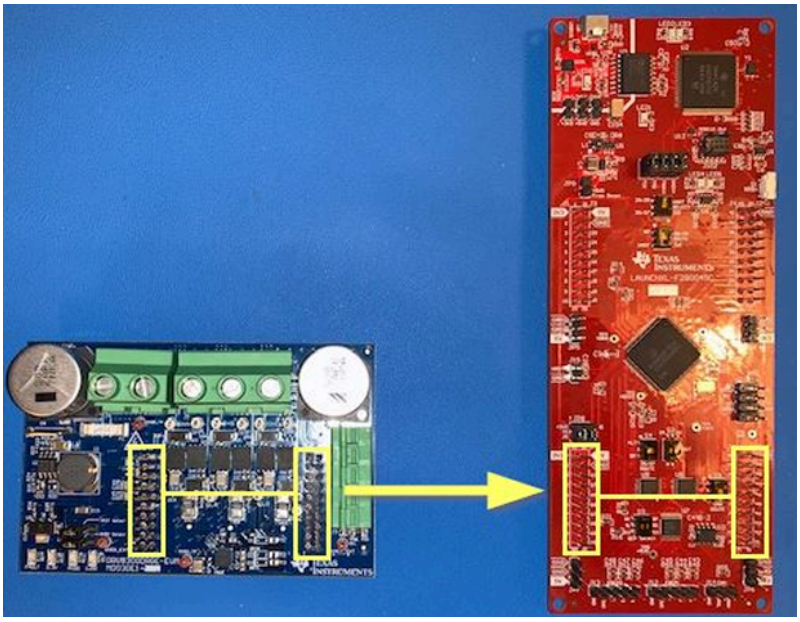


Figure 1. Mounting Location of the DRV8300Cxxx-EVM

4. Electrical Interface

4.1 Primary Input Power

Primary power through the DRV8300 board shall be connected to an external power supply. Power to the F28P65x board shall be provided through the USB connection.

4.2 Voltage Levels

The F28P65x Board shall use 5 VDC
The DRV8300 Board shall use 6 VDC to 100 VDC

4.3 User Control Interface

The user control interface will be the Sysconfig IDE on the host device.

5. Communications / Device Interface Protocols

5.1 Host Device

The host device will be the computer system connected to the microcontroller. Here, there commands will be given to the motor. As the host device requires Texas Instruments Code Composer Studio, it is recommended that the host device meet the same minimum and recommended requirements. The minimum hardware requirements are 4GB memory, 2.5 GB of disk space, and a 2.0 GHz single-core processor. The recommended hardware requirements are 8GB of memory, 5.0 GB of disk space, and a multicore processor.

5.2 Device Peripheral Interface

For connecting the computer system to the microcontroller, the project will be utilizing a wire protocol. The wire required is a USB to UART connector. The launchpad used in this project features a Fast Serial Communications Peripheral, enabling robust high-speed communications. See document Users Guide C2000™ F28P65x Series LaunchPad™ Development Kit, paragraph 3.1.5 FSI for specific details, and 3.1.6 and 3.1.7 for additional attributes.

ADC Resolution vs Sensor Less Motor Control Performance

Cristian Ornelas
Tyler Hawkins
Tamara Hussam A Basfar
John Adam King

FUNCTIONAL SYSTEM REQUIREMENTS

REVISION – 2
5 December 2024

Functional System Requirements
for
ADC Resolution vs Sensor Less Motor Control Performance

Prepared by:

_____
Author Date

Approved by:

_____
Project Leader Date

_____
John Lusher, P.E. Date

_____
T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
1	[9/26/24]	[ADC Resolution vs Sensor Less Motor Control Performance]		Draft Release
2	[12/04/2024]	[ADC Resolution vs Sensor Less Motor Control Performance]		Second Revision

Table of Contents

List of Tables

List of Figures

1. Introduction

1.1 Purpose and Scope

The project focuses on developing software for the TI c2000 chip to implement sensorless field-oriented control (FOC) for motor systems, as well as porting and testing an older sensorless motor code provided by TI. Instead of traditional sensors, a microcontroller will estimate the rotor's position. By implementing the sensorless motor control, there will be cost reductions, improved efficiency and increased reliability. To enhance and optimize this estimation, the ADC resolution will be increased. Our team will test our sensorless motor control at different speeds to evaluate the ADC resolution and its accuracy. The accuracy of the motor controller will be validated by comparing its performance to those from an incremental encoder.

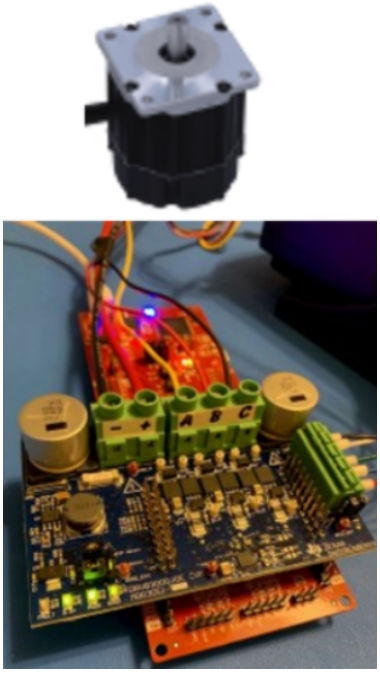


Figure 1. Conceptual Sensorless Motor Controller

The ADC sensorless motor control performance will be designed to fit the proper parameters for professionals to carry out the finished product. The milestones for the scope are listed below:

- Implement new Sensorless Motor Control subsystems in isolated environments.
- Implement Sensorless Motor Control subsystems onto F28p65x and DRV8300 board
- Port older motor control solution to F28p65x and DRV8300 board with SysConfig support

- Modify SysConfig setting and software for different ADC resolutions (12/16 bit)
- Modify the software to enable oversampling and new ADC feature in F28p65x
- Configure software to run an algorithm, in 32-bit and 64-bit floating point
- Run tests on sensorless control performance at low speed for each configuration

1.2 Responsibility and Change Authority

The team leader, Tyler Hawkins, is responsible for ensuring that all system specifications are met. Any changes to the deliverables or project specifications must be approved by both Tyler Hawkins and the sponsor's official representative Kevin Allen.

2. Applicable and Reference Documents

2.1 Applicable Documents

The following documents, of the exact issue and revision shown, form a part of this specification to the extent specified herein:

Document Number	Revision/Release Date	Document Title
SBVS144C	2012	Integrated MCU Power Solution for C2000 Microcontrollers
IEEE-STD-1241	2001	IEEE Standard for Terminology and Test Methods for Analog-to-Digital Converters
MCU117	2023	LAUNCHXL-F28P65X layout
v.0.3.0	2023	Servo Drive with CAN - Lab Projects User's guide

Table 1: Applicable documents

2.2 Reference Documents

The following documents are reference documents utilized in the development of this specification. These documents do not form a part of this specification and are not controlled by their reference herein.

Document Number	Revision/Release Date	Document Title
v1.3	2021	C2000 software guide
SLVUBV6	2020	DRV8300Dxxx-EVM User's guide

Table 2: Reference documents

2.3 Order of Precedence

In the event of a conflict between the text of this specification and an applicable document cited herein, the text of this specification takes precedence without any exceptions.

All specifications, standards, exhibits, drawings or other documents that are invoked as "applicable" in this specification are incorporated as cited. All documents that are referred

to within an applicable report are considered to be for guidance and information only, except ICDs that have their relevant documents considered to be incorporated as cited.

3. Requirements

This section defines the minimum requirements that the development item(s) must meet. The requirements and constraints that apply to performance, design, interoperability, reliability, etc., of the system, are covered.

3.1 System Definition

The project is split up into four main subsystems: Motor Control Porting, PWM driver/three-phase inverter, ADC driver, and estimator. The porter is responsible for taking the provided motor control code and updating it to function on the new launchpad. The PWM driver/three-phase inverter is responsible for converting a DC signal into 3 phases of AC waveforms. The ADC driver focuses on ensuring that the analog signal is properly conditioned to meet the input requirements of the ADC for accurate conversion. This includes signal conditioning, impedance matching, noise filtering, and signal buffering. Finally, the estimator is responsible for taking the back EMF of the motor and using it to determine the rotor orientation. The rest of the blocks in the diagram below are responsible for general control requirements and operation dependencies.

This system/ project will include testing for ADC oversampling and different ADC resolutions (12/16 bit). This will bring better SNR to currency and voltage sensing, ultimately bringing more accurate estimation results.

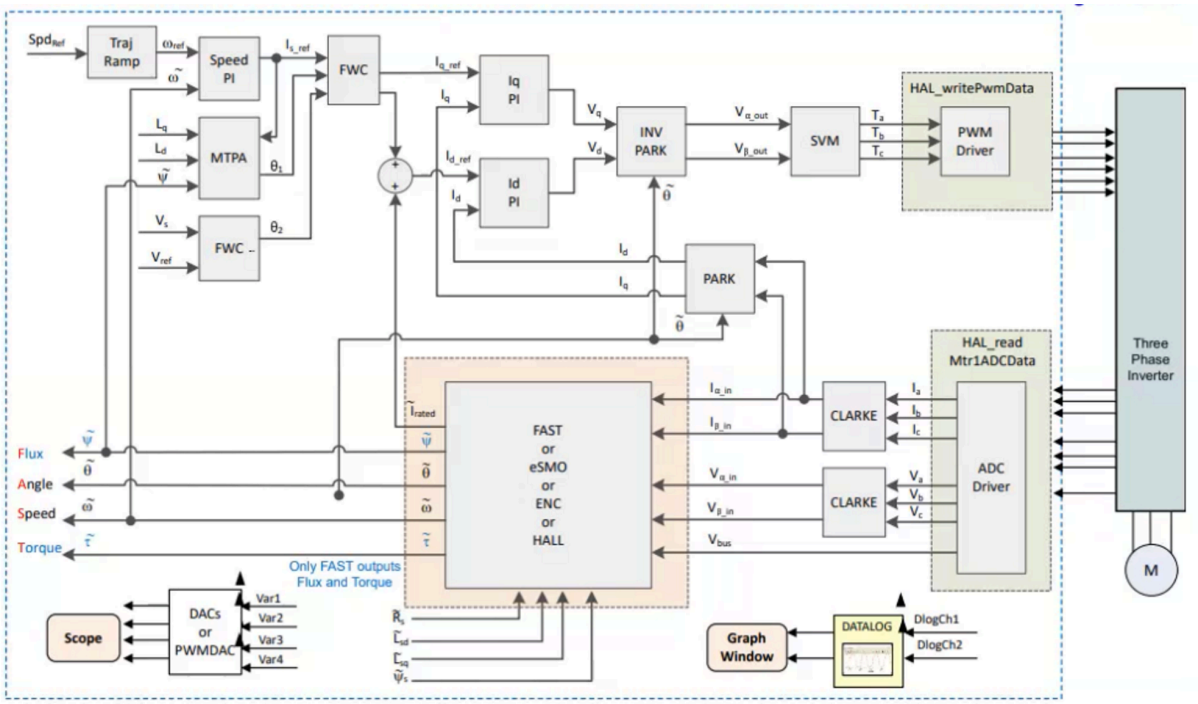


Figure 2. Block Diagram of System

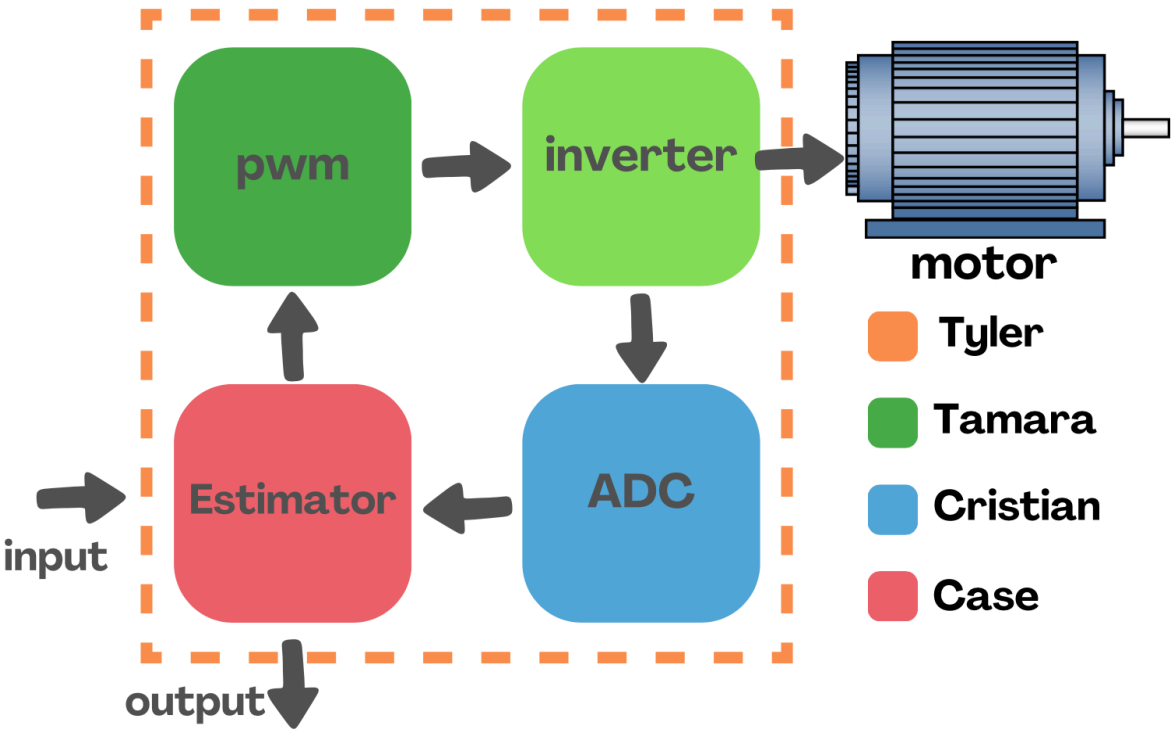


Figure 3. Simplified Block Diagram of System

3.2 Characteristics

3.2.1 Functional / Performance Requirements

3.2.1.1 Basic Motor Control

The sensorless control solution developed should effectively control the Teknic M-2310P-LN-04K Motor. The control solution should effectively determine the rotor's position, speed, torque, and rotation direction.

Rationale: This is the core system performance requirement. The sensorless control solution's performance is intended to be comparable to a motor sensor control solution.

3.2.1.2 SysConfig Support

Sensorless control of the motor should be configurable using the SysConfig IDE.

Rationale: This is the core system performance requirement. Using SysConfig allows for easy change of system options such as ADC Resolutions and 32/64 floating point.

3.2.1.3 12-Bit & 16-Bit ADC Resolutions

The sensorless control solution should support ADC Resolutions of 12 bit & 16 bit

Rationale: This is the core system performance requirement.

3.2.1.4 32 Bit & 64 Bit Floating Point

The sensorless control solution should support 32 Bit & 64 Bit Floating Point

Rationale: This is the core system performance requirement.

3.2.2 Physical Characteristics

The project is centered around a software application, however, this software is being designed to run with specific components; a microcontroller, motor controller, and motor. For development, the microcontroller is an F28p65x launchpad from Texas Instruments. While this board has dual processing cores, only Core-1 is being used. The motor controller is a DRV8300DIPW-EVM evaluation module from Texas Instruments. Lastly, the motor itself is a Low Voltage Servo Motor from Texas Instruments.

3.2.2.1 Board Connections

Our F28p65x Launchpad Board should be connected to the DRV8300 motor driver board and the Teknic M-2310P-LN-04K Motor.

Rationale: In order to control the motor listed above, the boards should be connected using the board's contact pins, in addition to jumper wires.

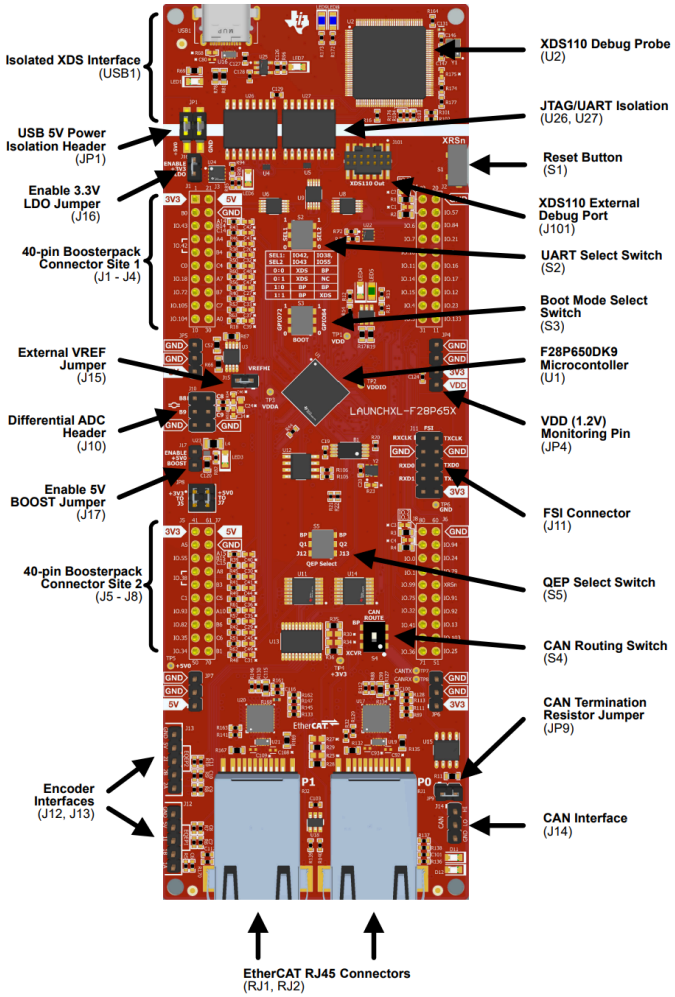


Figure 4. F28P65x Board Connections

3.2.3 Electrical Characteristics

3.2.3.1 Inputs

a. USB-C connection to the host device for the purpose of receiving control commands.

b. The Motor Driver Booster Pack is connected using the 40-pin connection at site 2 shown in the image above.

c. The motor is connected to the driver board's corresponding phase a,b & c headers, as well as positive power & ground.

d. When applicable for testing connections to the motor's sensor will be connected to the respective phase connectors on the driver board

Rationale: By design, should limit the chance of damage or malfunction by user/technician error.

3.2.3.2 Input Voltage Level

The input voltage of the F28P65x Board shall be 5 VDC through the USB connection
The DRV8300Dxxx-EVM is designed for an input external supply from 6 VDC to 100 VDC and up to 25-A continuous drive current (software limited).

Rationale: TI F28P65x & DRV83000 Board specifications

3.2.3.3 External Commands

The motor control solution shall document all external commands in the appropriate ICD.

Rationale: The ICD will capture all interface details from the low-level electrical to the high-level packet format.

3.2.3.6 Outputs

3.2.3.7 Data Output

The motor control solution shall include the Sysconfig data interface that is compatible with the system.

Rationale: Using Sysconfig, data output motoring can be viewed through the host device.

3.2.3.8 Diagnostic Output

The motor control solution shall include the Sysconfig control and data logging interface.

Rationale: Provides the ability to control things for debugging.

3.2.3.9 Wiring

The motor control solution shall follow the guidelines outlined in DRV8300xxx-EVM User's Guide paragraph 2.1 Hardware Connections Overview.

Rationale: Conform to Driver Board standard.

3.2.4 Failure Propagation

3.2.4.1 Diagnostic Errors

The motor control solution will monitor the voltage & current levels allowing visibility of any errors of the hardware and its operation.

Rationale: This will help preserve the integrity of the data being collected

4. Support Requirements

4.1.1. Computer USB Port

A computer is required to compile & run motor control solutions using the CCS environment. This requires a minimum of 4GBs of RAM, 2.5GBs disk space, & 2.0GHz single-core processor.

Appendix A: Acronyms and Abbreviations

EMF	Electro-motive Force
BIT	Built-In Test
CCA	Circuit Card Assembly
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
EO/IR	Electro-optical Infrared
FOR	Field of Regard
FOV	Field of View
GPS	Global Positioning System
GUI	Graphical User Interface
Hz	Hertz
ICD	Interface Control Document
kHz	Kilohertz (1,000 Hz)
LCD	Liquid Crystal Display
LED	Light-emitting Diode
mA	Milliamp
MHz	Megahertz (1,000,000 Hz)
MTBF	Mean Time Between Failure
MTTR	Mean Time To Repair
mW	Milliwatt
PCB	Printed Circuit Board
RMS	Root Mean Square
TBD	To Be Determined
TTL	Transistor-Transistor Logic
USB	Universal Serial Bus
VME	VERSA-Module Europe
PWM	pulse width modulation
ADC	Analog to Digital converter
INV	inverter
c2000	real-time microcontroller
sysConfig	development tool for c2000
CCS	Code Compiler Studio
CMPSS	Comparator Subsystem

Appendix B: Definition of Terms

Sysconfig: a configuration tool designed to simplify hardware and software configuration challenges to accelerate software development. SysConfig provides an intuitive graphical user interface for configuring pins, peripherals, radios, software stacks, RTOS, clock tree and other components. SysConfig will automatically detect, expose and resolve conflicts to speed software development.

Status Indicators	
On schedule	
In progress	
Completed	
Behind schedule	
Important Deadlines	

Execution Plan:

Task	Owner	Deadline	Status
Conops Report	All	9/15/24	
FSR Report	All	9/26/24	
ICD Report	All	9/26/24	
Order required hardware	All	9/15/24	
Become familiar with the CCS & sysconfig environments	All	9/21/24	
Become familiar with the motor control SDK	All	9/28/24	
Midterm Presentation	All	9/30/24	
Subsystem Research	Tyler	10/1/24	
Subsystem Research	Tamara	10/1/24	
Subsystem Research	Case	10/1/24	
Subsystem Research	Cristian	10/1/24	
Subsystem Introduction Project	Tyler	10/8/24	
Subsystem Introduction Project	Tamara	10/8/24	
Subsystem Introduction Project	Case	10/8/24	
Subsystem Introduction Project	Cristian	10/8/24	
Understand needed Functions/Data for the Estimator	Case	10/10/24	

Have Code Outline Complete	Case, Tamara, Cristian	10/10/24	
Begin porting process	Tyler	10/12/24	
Begin writing code for Inverter/PWM for the F28P65x device	Tamara	10/14/24	
Begin writing code for the ADC driver for F28P65x device	Cristian	10/14/24	
Begin writing code for the Estimator for the F28P65x device	Case	10/14/24	
Project Update Presentation	All	10/21/24	
Begin the debugging process for PWM/Inverter	Tamara	11/4/24	
Begin the debugging process for ADC driver	Cristian	11/4/24	
Begin the debugging process for Estimator	Case	11/11/24	
Port existing solution to F28P65x	Tyler	11/18/24	
Finish up Subsystem Demo requirements	All	11/18/24	
Begin working on 16 ADC resolution	Cristian	11/26/24	
Begin Working DRV8300 support	Tyler	11/26/24	
Final Presentation	All	11/18/24	
Subsystem Demo	All	11/26/24	
Final Report	All	12/5/24	

Validation Plan:

Task	Specification	Result	Owner
Test individual function Operation (test cases)	Each block function outputs correct data and the integrated project compiles	PASSED	Tamara PWM
Test individual function Operation (in/out waveforms)	Waveforms behave as expected	PASSED	Tamara PWM
Compile Full PWM Project	Integrated project compiles	PASSED	Tamara PWM
Integrated Project Test	The PWM project shows correct duty cycles and can change in real-time	PASSED	Tamara PWM
Test Estimator Operation	The estimator correctly collects motor information. This is done by comparing with sensored motor outputs	PASSED	Case EST
ADC Driver Operation	ADC Correctly converts Analog signal to digital signal through test cases and plotting	PASSED	Cristian ADC
ADC Overview	The code will compile with 0 errors.	PASSED	Cristian ADC
ADC/ GPIO Behavior	The memory result buffer will show the behavior of ADC A (Toggle)	PASSED	Cristian ADC
ADC/ GPIO Behavior	The memory result buffer will show the behavior of ADC B (Low)	PASSED	Cristian ADC
ADC/ GPIO Behavior	The memory result buffer will show the behavior of ADC C (High)	PASSED	Cristian ADC
Plotting/ Graph	Code Composer Studio will show the PWM waveform updating in real time	PASSED	Cristian ADC
Plotting/ Graph	An oscilloscope will show the GPIO 3 toggling in regards to ADC A	PASSED	Cristian ADC
f28p65x Behavior	LEDs show the behavior of the GPIO pin on the f28p65x board	PASSED	Cristian ADC
System Compiling	Full system compile	PASSED	Tyler PORTING

Motor control With sensorless Operation	The full system of a motor without using a sensor operates correctly on the F28002x board	PASSED	Tyler PORTING
Porting Solution onto F28p65x board	Ported solution compiles without any errors	PASSED	Tyler PORTING
Porting Solution onto F28p65x board	Ported motor control software onto f28p65x board successfully.	BEHIND SCHEDULE	Tyler PORTING
Motor control With sensor Operation	The full system of a motor using a sensor operates correctly. Porting existing sensored motor control software onto 02x launch pad successfully.	PASSED	Tyler PORTING
SysConfig support	The system correctly operates using the SysConfig IDE	UNTESTED	All
16 Bit ADC resolution	The system correctly operates using 16-bit ADC resolution	UNTESTED	Cristian
12 Bit & 16 Bit ADC resolution Support	The system correctly operates using either 12-bit or 16-bit ADC resolutions	UNTESTED	Cristian
64-bit Floating point operation	The system Correctly Operates using 64-bit Floating Point	UNTESTED	Tyler
32-bit & 64-bit Floating point operation	The system Correctly Operates using either 32-bit or 64-bit Floating Point	UNTESTED	Tyler
Test 12 vs 16 Bit performance	Test the accuracy of both 12-bit and 16-bit ADC resolutions when compared to a motor with a sensor	UNTESTED	All
Test 32-bit & 64-bit Floating point operation	Test the accuracy of both 32-bit and 64-bit ADC Floating point operations when compared to a motor with a sensor	UNTESTED	All
Test the effectiveness of oversampling	Test the effectiveness of oversampling by comparison to the old system using a sensor	UNTESTED	All

Performance of Execution & Validation Plan

The execution and validation plan was completed entirely to our expectations for the semester. The only exception is that the full port of the motor control solution needs some final debugging. This will be fixed and tested over the winter break so that we are adequately prepared for next semester.

ADC Resolution vs Sensor Less Motor Control Performance

Cristian Ornelas
Tyler Hawkins
Tamara Hussam A Basfar
John Adam King

SUBSYSTEM REPORTS

REVISION – Original
2 December 2024

SUBSYSTEM REPORTS
FOR
ADC Resolution vs Sensor Less Motor Control Performance

TEAM <1>

APPROVED BY:

_____
Project Leader Date

_____
Prof. Kalafatis Date

_____
T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
1	[12/02/2024]	[ADC Resolution vs Sensor Less Motor Control Performance]		Original Release

Table of Contents

List of Figures

1.0 Introduction

The ADC Resolution vs Sensorless Motor Control Performance project involves the development and integration of key subsystems required to achieve precise and efficient motor control without relying on physical position sensors. This approach enhances system reliability, reduces costs, and enables compact design by replacing hardware sensors with advanced computational algorithms. The project is structured around four critical subsystems: the PWM subsystem, the ADC subsystem, the estimator subsystem, and the overall porting subsystem. Each subsystem plays a pivotal role in achieving the project's goals and ensuring seamless operation of the motor.

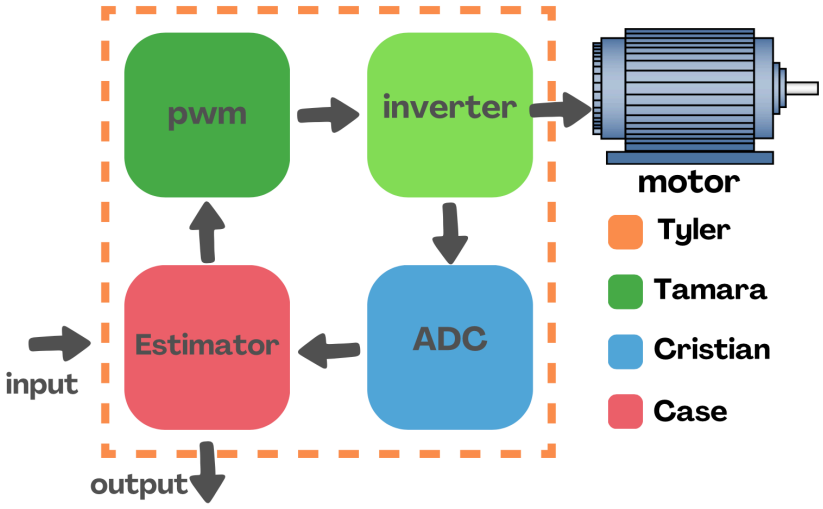


Figure 1: Subsystem Diagram

2.0 PWM Subsystem Report - Tamara Basfar

2.1 Subsystem Introduction

Pulse Width Modulation (PWM) is a powerful technique used in various applications to control the amount of power delivered to an electrical load without dissipating energy in the form of heat. This subsystem explores the principles and practical implementations of PWM, focusing on its applications in sensorless open-loop motor control visualized by LED dimming.

The primary objective of this subsystem was to design and implement a PWM controller that can efficiently manage power delivery in different scenarios. The design is based on the block diagram in figure 2 that includes an angle generator, an inverse Park transformation, Space Vector Modulation (SVM), and a PWM driver. By varying the duty cycle of the PWM signal, the speed, torque, and flux of the three phase motor can be controlled precisely, thereby achieving desired performance characteristics.

This report will cover the design process of the PWM controller, the submodule descriptions, and the validation of each submodule as well as the whole subsystem.

2.2 Subsystem Design

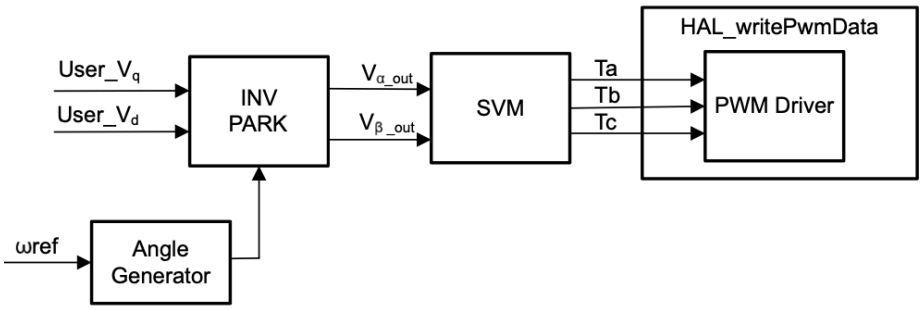


Figure 2: PWM subsystem software flow

2.2.1 Software Flow Overview

The software flow for the PWM subsystem of the sensorless motor control project ensures seamless generation of control signals to drive the motor. It begins with the Angle Generator, which calculates the rotor angle (θ) based on a reference speed (ω_ref) and updates this angle dynamically over time. This angle is passed to the Inverse Park Transformation (INV PARK) block, which converts control signals (Vq, Vd) from the rotating reference frame (d-q) into stationary reference frame signals (Vα, Vβ). These signals are then processed by the Space Vector Modulation (SVM) block, which calculates the precise duty cycles (Ta, Tb, Tc) required for the three motor phases. Finally, the PWM Driver uses these duty cycles to generate actual PWM signals that control the motor's three-phase inverter.

2.2.2 Submodule Description

The Angle Generator is responsible for computing the rotor angle (θ) based on a user-defined reference frequency. It first calculates the angle delta (angle_delta_rad) by multiplying the

reference frequency with a time step factor. This delta is then added to the current angle (angle_rad) to simulate the rotor's rotation. To ensure smooth operation, the angle is wrapped around to stay within the range of [-π, π] using a masking condition. This continuous and dynamic angle generation allows the control system to maintain synchronization with the rotor's position.

The Inverse Park Transformation translates the d-q frame control voltages (Vq, Vd) into the stationary reference frame (Vα, Vβ) using trigonometric relationships. The formulas $Vα=Vdcos(θ)−Vqsin(θ)Vα=Vdcos(θ)−Vqsin(θ)$, $Vβ=Vdsin(θ)+Vqcos(θ)Vβ=Vdsin(θ)+Vqcos(θ)$ are employed, where the angle θ is supplied by the Angle Generator. This transformation ensures that control signals designed in the d-q reference frame are compatible with the motor's stationary frame operation.

The Space Vector Modulation (SVM) block takes the stationary frame voltages (Vα, Vβ) and calculates the appropriate duty cycles (Ta, Tb, Tc) for the PWM signals. It identifies the sector of the hexagonal voltage space where the input voltages lie and determines the durations for the active and zero vectors. These durations are then used to compute the duty cycles for each phase, ensuring optimal inverter switching to produce sinusoidal currents with minimal harmonic distortion.

The PWM Driver takes the duty cycles (Ta, Tb, Tc) generated by the SVM block and applies them to the motor's three-phase inverter. This subsystem directly controls the switching of the inverter's transistors, translating the calculated duty cycles into physical PWM signals. These signals dictate the current flow in the motor phases, effectively controlling the motor's speed and torque.

2.2.2 Submodule Interactions

The four submodules work together in a modular and synchronized manner to achieve precise PWM signal control. The Angle Generator produces a continuously updated rotor angle (θ), which is a critical input to the Inverse Park Transformation (INV PARK) block. This transformation adapts the control voltages (Vq, Vd) into stationary frame signals (Vα, Vβ) to match the motor's electrical requirements. The SVM block then processes these voltages to compute the exact duty cycles (Ta, Tb, Tc) required to generate sinusoidal phase currents. Finally, the PWM Driver converts these duty cycles into PWM signals that operate the inverter, providing efficient and accurate control of the motor.

This modular architecture offers flexibility and scalability. Each submodule is focused on a specific function, enabling ease of development and debugging. Additionally, any submodule can be modified or replaced without impacting the others, making the design robust and adaptable to future enhancements.

2.3 Subsystem Validation

The first step of validation was to test all the functions individually to ensure that the logic was correct and the outputs behaved as expected. This approach ensured that if any errors occurred during integration, the debugging process would be more straightforward and efficient. Each

function was validated against its intended behavior with controlled inputs and visualized outputs to confirm its correctness.

2.3.1 Angle Generator

The first function to be developed and validated was the Angle Generator, as explained in the design section. After finalizing the necessary logic, the function was tested with various input frequencies to observe its response. The output angle was graphed to ensure it followed a sawtooth waveform, wrapping around correctly within the range of [−π,π][−π,π].
Figures 2, 3, and 4 show the outputs of the angle generator at frequencies of 10 Hz, 50 Hz, and 100 Hz, respectively. As seen in the graphs, the angle generator produces a tighter sawtooth waveform as the frequency increases, confirming that the angle delta calculation and wrap-around logic are functioning as expected.

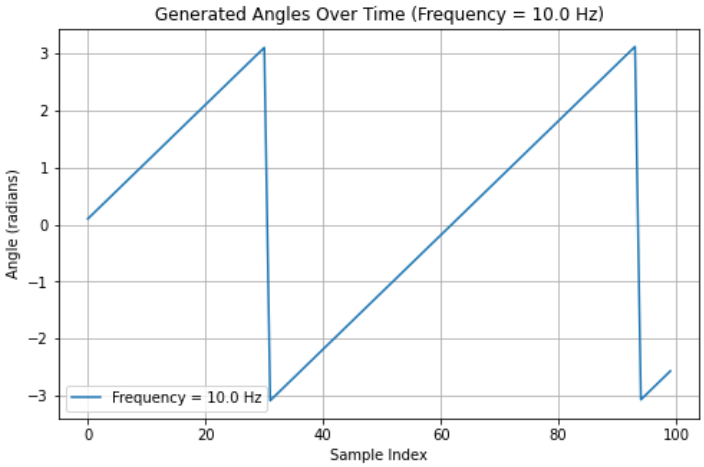


Figure 3: output for angle generator at 10Hz

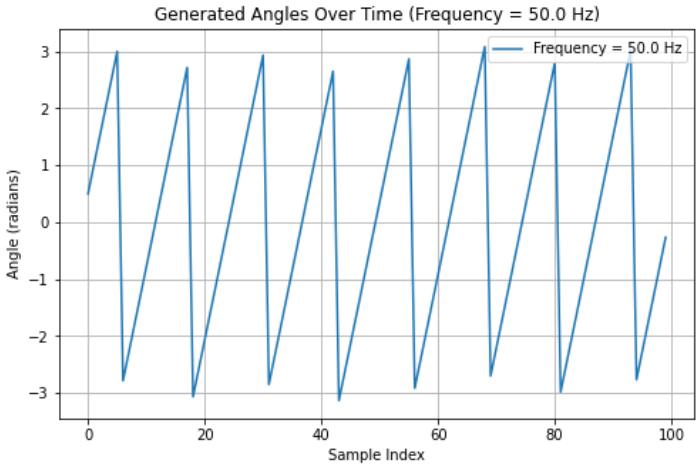


Figure 4: output for angle generator at 50Hz

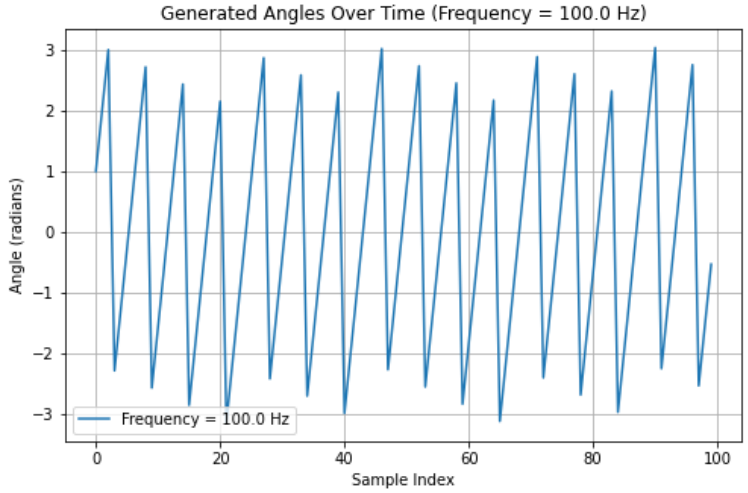


Figure 5: output for angle generator at 100Hz

2.3.2 Inverse Park

The Inverse Park Transformation was validated next. This function was tested by providing a set of known d-q inputs and a controlled rotor angle (θ) from the Angle Generator. The corresponding α-β outputs were calculated and compared with theoretical values derived from the transformation equations.

Figure 5 shows the input and output waveforms for the Inverse Park Transformation. The results confirmed that the function correctly maps d-q voltages into the stationary α-β frame. The waveforms of Vα and Vβ followed the expected sinusoidal patterns, demonstrating that the trigonometric relationships were implemented correctly.

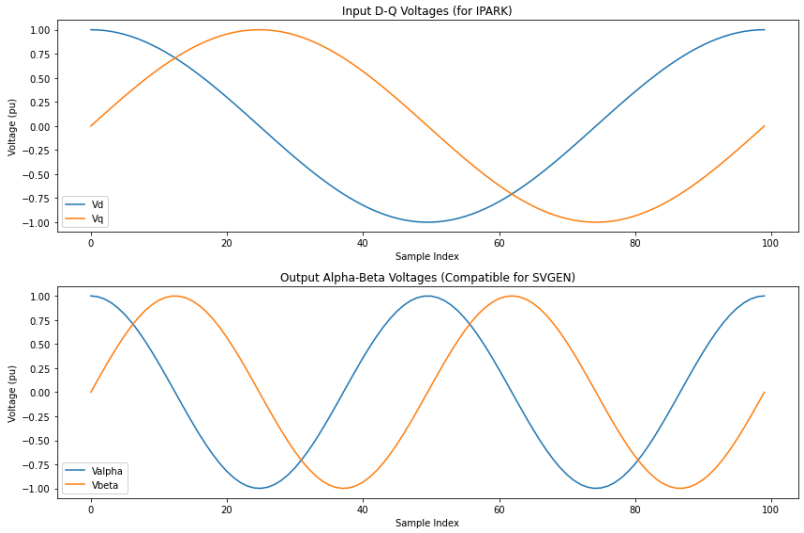


Figure 6: Inverse park input and output waveforms

2.3.3 Space Vector Modulation

The Space Vector Modulation (SVM) block was validated by supplying it with the outputs from the Inverse Park Transformation and analyzing the resulting duty cycles (Ta, Tb, Tc). The validation involved ensuring that the calculated duty cycles were consistent with the input voltage space vectors and adhered to the expected PWM switching patterns.
Figure 6 illustrates the input waveforms to the SVM block, while Figure 7 shows the corresponding duty cycle outputs. The results indicate that the SVM block correctly identified the active sectors, calculated the appropriate vector durations, and generated the expected duty cycles for the three phases.

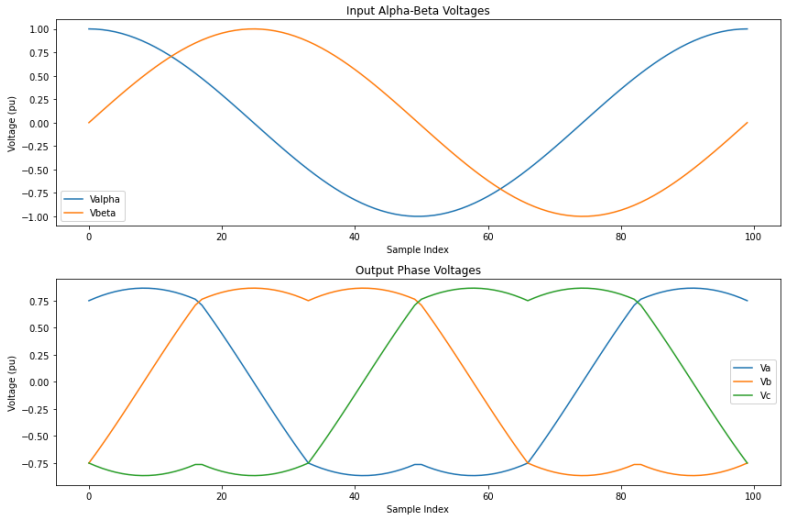


Figure 7: SVM input and output waveforms

2.3.3 PWM Driver

Finally, the PWM Driver was validated by analyzing the PWM signals it generated based on the SVM duty cycles. The validation process involved capturing the PWM signals for each phase and ensuring they matched the calculated duty cycles. Oscilloscope waveforms were used to verify that the switching signals aligned correctly with the expected timing, providing smooth and balanced phase currents.
The PWM signals shown in Figure 7 demonstrate clean transitions and proper switching sequences. This confirms that the integration of the PWM Driver with the SVM block is working correctly and that the PWM signals are suitable for controlling the three-phase inverter.

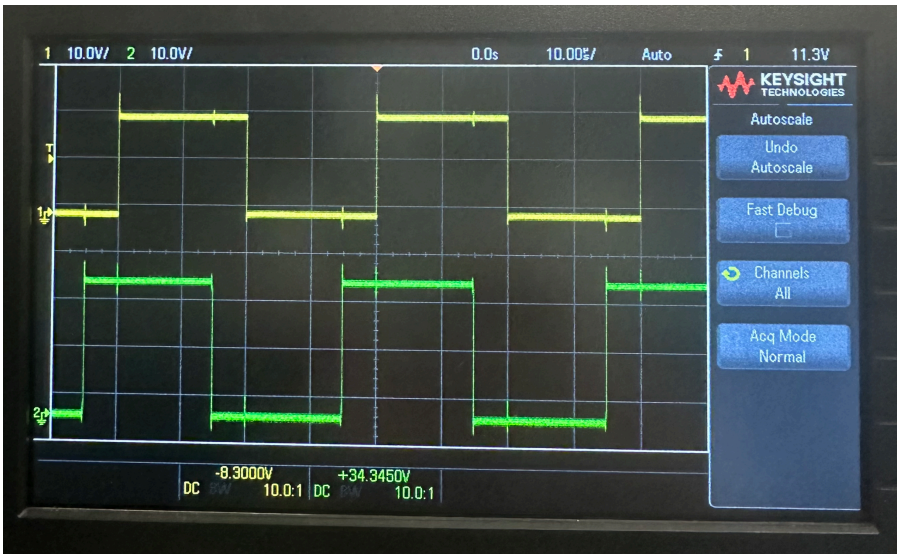


Figure 8: PWM signals

2.4 Subsystem Conclusion

By testing each function individually and then integrating them step-by-step, it was ensured that each subsystem functioned as intended before moving to the next. The results from the validation process confirmed that the PWM subsystem is capable of generating accurate and efficient control signals for sensorless motor operation. The outputs of each subsystem closely matched the theoretical expectations, validating the design and implementation of the overall system.

3.0 ADC Subsystem Report - Cristian Ornelas

3.1 – Subsystem Introduction

Analog-to-digital conversion is an important technique used in the modern industry to convert analog signals into digital data that digital systems can process. This introduction explores the principles and practical implementations of the ADC subsystem.

The primary objective of an analog-to-digital converter in sensorless motor control is to take analog voltage and current values of the PWM/ inverter and convert them into digital values for the estimator to estimate the position of the rotor accurately. This subsystem includes the start of conversion signals/ control, post-processing blocks, input channel mux, and a reference voltage generator as shown in the figure below. When the ADC module is sampled at a certain rate, we can see and verify the conversion to a digital format that can be seen and validated in Texas Instrument's Code Composer Studio.

3.2 – Subsystem Design

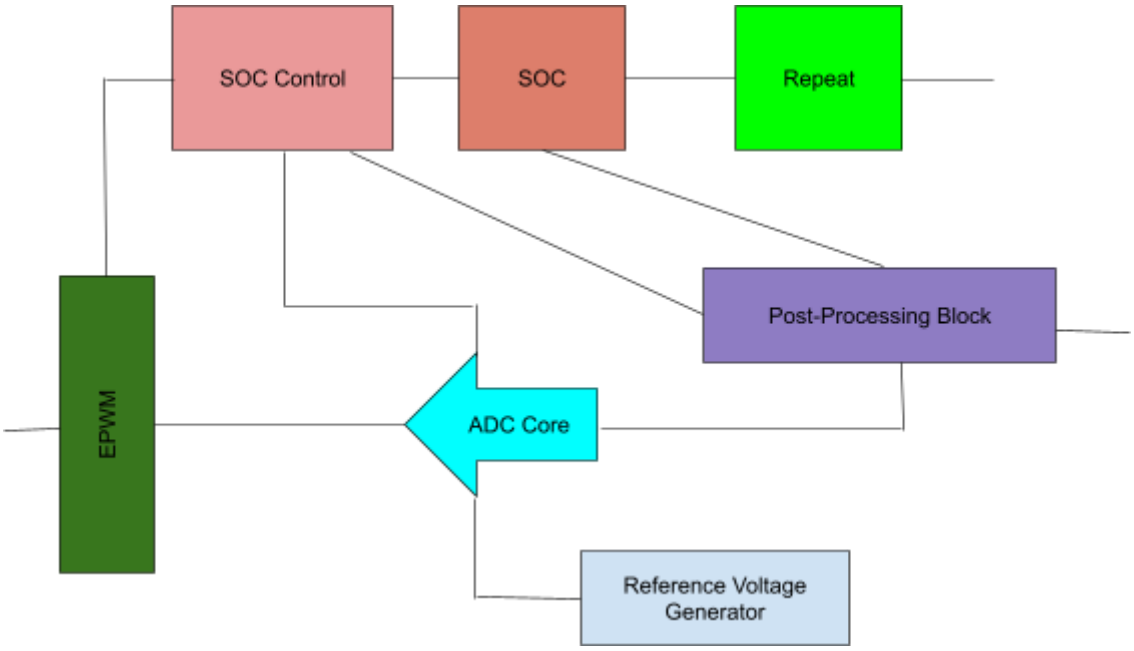


Figure 9: Analog-to-Digital Converter Block Diagram

3.2.1 Software Flow Overview

To start the process of the ADC conversion, the input channel MUX selects the input signal to convert. Next, I configured an EPWM module in TI's Sysconfig. that interacts with the start-of-conversion interrupt and triggers the conversion process. The SOC arbitration makes sure that there is an orderly fashion of scheduling for the several SOC requests. The ADC conversion priority determines the order in which the conversion happens. The submodule contains 3 ADCs that operate in a round-robin scheme where no SOC has more priority than another. In other words, it happens circularly. Next, the ADC core performs the actual conversion. The post-processing block refines the raw output and reduces the error. A baseline of accurate voltage measurements is created by the reference voltage generator. Finally, the repeat block is utilized for a continuous conversion in real time.

3.2.2 Submodule Description

GPIO pins on the board are utilized to show the different behaviors of the 3 ADC modules. Each ADC has a different behavior to show an interesting output and to add more validation for the ADC conversion. For the high/low toggle behavior of converted values, GPIO 3 is configured with ADCA to show the values toggle from 4095 to 0. The reason for the ceiling of 4095 is because of the 12-bit format. There are ((2^12) - 1) values (4095) in the 12-bit range. GPIO 16 is configured with ADCB to show the behavior of only high values being updated continuously (4095). Finally, GPIO 32 on the board is configured with ADCC to show only low values being updated continuously (0).

The input channel multiplexer is used to select one input signal from several analog signals and route it to the ADC core for conversion. In this case, this block allows the program to handle the multiple input channels that were configured in the code. The round-robin method previously mentioned makes the conversion process efficient by setting up the conversion in a circular scheme.

The start-of-conversion arbitration block is utilized as it manages and prioritizes the many SOC requests when input channels simultaneously call for conversion. The EPWM module triggers the SOC interrupt and sets up the different configurations such as sampling time.

The post-processing block handles the raw ADC conversion results and corrects the errors to help with scaling. The PPB makes sure the output values are efficient and usable for further processing in the system.

The ADC core is the heartbeat of the whole subsystem. The core is responsible for the resolution, speed, and accuracy of the conversions. As mentioned before, the 12-bit ADCs in this subsystem produce digital values ranging from 0 to 4095 for the input voltage range. The reference voltage generator interacts with the core as it provides a stable reference voltage that acts as a basis for determining input signal levels.

3.2.2 Submodule Interactions

To help with efficiency, the graphical user interface “Sysconfig.” is implemented in Code Composer Studio to help with PinMux configuration, GPIO setup, ADC setup, etc. in a working environment. Using this tool automatically generates the files needed to help the ADC subsystem run properly.

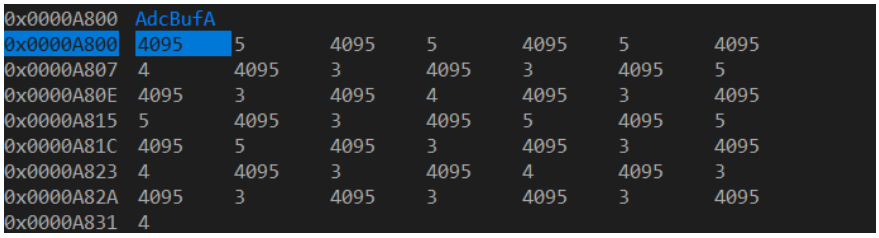
The submodules in the ADC work jointly to ensure efficient analog-to-digital conversion. Before sending the required input signal to the ADC core, the input channel MUX first chooses from the 3 different input channels configured in Sysconfig (A0, B0, C0). The SOC arbitration then ensures a fair scheduling of conversion using a circular buffer scheme. It settles disputes if several start-of-conversion interrupts trigger at the same time. Moreover, the SOC block is triggered by the EPWM and starts the conversion process. For precise signal quantization, the reference voltage generator uses its steady voltage as a baseline for the conversion. The board has a max input voltage of 3.3 V so that the board does not get damaged. The post-processing block then processes the raw converted values and makes any necessary adjustments. After conversion, the voltage values are compiled into the buffer which can be later viewed in the CCS memory buffer. Finally, the end-of-conversion interrupt is utilized to show the completed conversion. Finally, the repeat block shows that the ADC conversion runs continuously in real time if desired.

3.3 – Subsystem Validation

The purpose of subsystem validation is to ensure all design-specified requirements from the submodule are met. This idea helps reduce the risk of issues in the later stages of the integration process. By validating the subsystems, the integration process will lead to a higher-quality product.

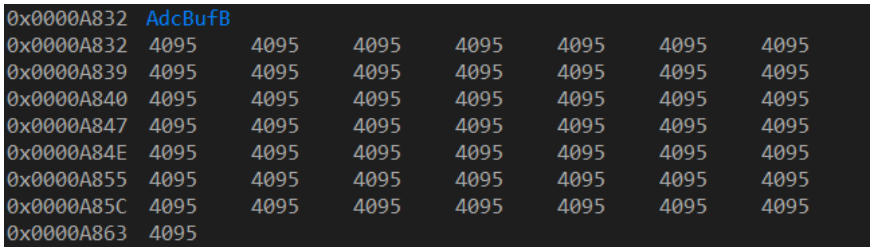
3.3.1 Memory Buffer

The first validation test is to see the ADC result memory buffer. This idea ensures that the ADC is correctly converting analog signals to digital values. If there is any error in the ADC function, this is the first place to examine the problem. Concerning this subsystem module, 3 different ADCs simultaneously convert shadowing different behaviors. First, ADCA converts values that follow a high/ low toggle behavior. Second, ADCB follows a high output conversion behavior. Third, ADCC follows a low output conversion. In the figures below, one should see the values behaving in these configurations.



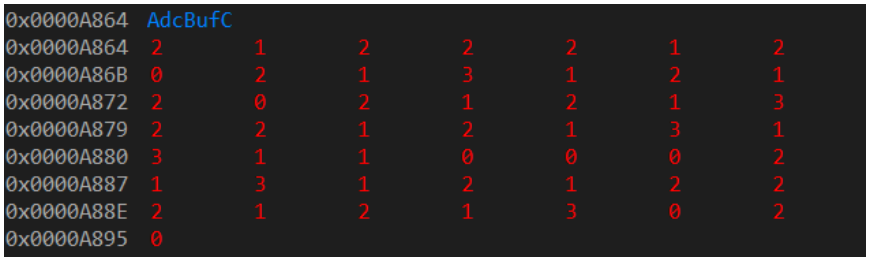
0x0000A800	AdcBufA						
0x0000A800	4095	5	4095	5	4095	5	4095
0x0000A807	4	4095	3	4095	3	4095	5
0x0000A80E	4095	3	4095	4	4095	3	4095
0x0000A815	5	4095	3	4095	5	4095	5
0x0000A81C	4095	5	4095	3	4095	3	4095
0x0000A823	4	4095	3	4095	4	4095	3
0x0000A82A	4095	3	4095	3	4095	3	4095
0x0000A831	4						

Figure 10: ADCA High/Low Toggle Conversion Behavior in Memory Buffer



0x0000A832	AdcBufB						
0x0000A832	4095	4095	4095	4095	4095	4095	4095
0x0000A839	4095	4095	4095	4095	4095	4095	4095
0x0000A840	4095	4095	4095	4095	4095	4095	4095
0x0000A847	4095	4095	4095	4095	4095	4095	4095
0x0000A84E	4095	4095	4095	4095	4095	4095	4095
0x0000A855	4095	4095	4095	4095	4095	4095	4095
0x0000A85C	4095	4095	4095	4095	4095	4095	4095
0x0000A863	4095						

Figure 11: ADCB High Only Value Conversion Behavior in Memory Buffer



0x0000A864	AdcBufC						
0x0000A864	2	1	2	2	2	1	2
0x0000A86B	0	2	1	3	1	2	1
0x0000A872	2	0	2	1	2	1	3
0x0000A879	2	2	1	2	1	3	1
0x0000A880	3	1	1	0	0	0	2
0x0000A887	1	3	1	2	1	2	2
0x0000A88E	2	1	2	1	3	0	2
0x0000A895	0						

Figure 12: ADCC Low Only Value Conversion Behavior in Memory Buffer

3.3.2 ADC Buffer PWM Waveform

Another form of validation for this module is to see the waveforms of the ADC converting in real-time. This waveform is validation to show the ADC is correctly sampling the PWM signal mentioned in the design section above. The waveform shows the data is being processed and displayed correctly along with the timing and synchronization of the subsystem. Below is the ADCA Buffer PWM sawtooth waveform for the high/low GPIO toggle pin behavior.

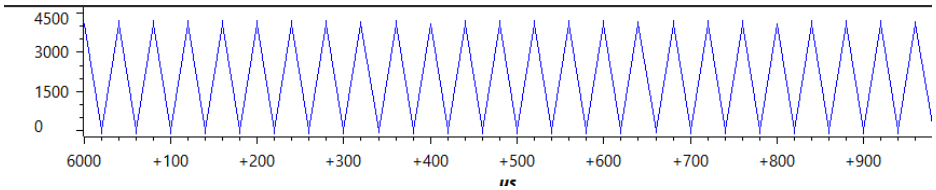


Figure 13: ADCA PWM High/Low Toggle Conversion Behavior Sawtooth Waveform

3.3.3 LED Behavior

LEDs 4 and 5 are utilized on the f28p65x board to show further validation that the ADC module is correctly converting the analog signal to digital values. LED 4 is configured with ADCA which follows the high/low GPIO 3 pin and blinks continuously to show the behavior. LED 5 is configured with ADCB that only converts high values continuously. In this case, the LED is on at all times during the program run to show the proper conversion. Finally, it does not seem necessary to configure an LED with ADCC that only converts low values. This ADC follows the behavior of the GPIO 32 pin and would be off the whole time running.

3.3 – Subsystem Conclusion

By testing each component of the ADC module, it can be seen that the ADC subsystem properly converts the varying voltage into a digital value that the sensorless motor can use for modules such as the estimator. The results of these validation tests show the theoretical behaviors of the ADC which were deemed successful. This allows the subsystem module to be passed into the integration phase.

4.0 Estimator Subsystem Report - Case

4.1 Subsystem Introduction

Rotor Position Estimation is the means by which digital calculations are used in order to estimate the position of a rotor in a motor. Position Estimators arose as a solution for creating less-expensive, more robust motors that can be built much easier by removing the need for position sensors.

The objective of this subsystem was to design and implement a Rotor Position Estimator code that is able to accurately estimate the position of a rotor, as well as its RPM, for use in other subsystems, as well as to be seen by the user. The estimator functions using the BackEMF generated across the stator windings as a result of the interactions between the stator and the rotor. The calculations utilized include the Clarke Transform, BackEMF Calculation, a trigonometric function, and speed calculator, as shown in Figure 4-1.

This report will cover the design process of the Estimator, the purposes of each component, and the validation process used for this system.

4.2 Subsystem Design

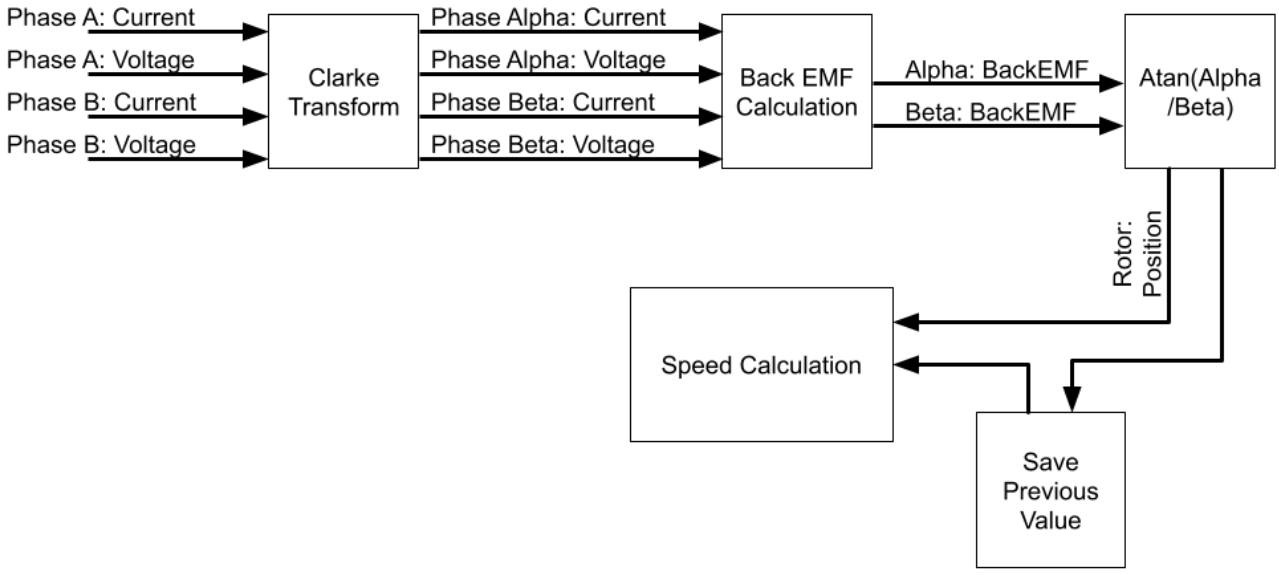


Figure 14: Rotor Estimator Subsystem Flow Diagram

4.2.1 Software Flow Overview

The Estimator software ensures that other portions of the motor know the rough position of the rotor at all times. The motor in use is a three-phase AC motor. The process of determining the rotor position begins by taking the current of voltage of two phases (Ia, Va, Ib, Vb), though this can be done with all three, and performing a Clarke Transform. This takes the two phases, which were originally offset by 120°, and creates Phase Alpha Current and Voltage, and Phase Beta Current and Voltage (Iα, Vα, Iβ, Vβ), which represent the three motor as two phases offset by 90° of each other. The current and voltage of Phases Alpha and Beta, as well as the motor design data, are used to calculate the BackEMF through these specific phases (BackEMFα, BackEMFβ). Using an Atan function, the angular position of the motor can be obtained. When combined with the previous position, and time between estimations, the rotor's angular velocity can be found as well.

4.2.2 Submodule Code

The code that contains the rotor estimation code is written in a way such that they can be easily modified for different three-phase AC motors. Additionally, the functions themselves are linear in nature, and thus the code can be optimized by reducing the number of functions necessary to complete each iteration.

4.3 Subsystem Validation

The validation plan for this subsystem was designed on the idea that complexity within the system would slowly increase to allow for as much resolution when looking for problems within the code as possible.

4.3.1 Testing Environment Validation

The first validation step was to ensure that a proper testing environment was created for code. This was written in Visual Studio Code in C++, as this allowed for writing and testing of the code to begin much sooner, as well as streamline the testing process. Validation was achieved when the testing environment was able to input and output information to and from a CSV file, using the earliest version of the estimator code.

4.3.2 Ideal Condition Tests

Ideal Test Conditions refer to a constant, steady input of information that follows a sin and cosine wave. There is no or regular variation in the amplitude or frequency, and thus a constant, or regularly changing rotor velocity was expected. The earliest version of the code however would see a constant output of null or nonsensical values, leading to a rewriting of how the code handled floating point integers. Additionally, it showed the need for limits to be placed on what the code would consider a valid output, as during start-up, massive values that interfered with the graphing data. These tests were labeled as:

- Linear Data: No change to the frequency or amplitude of the input wave, so output data corresponding to a constant velocity is expected.
- Rising Data: Frequency or amplitude increase over time, thus a rising velocity for the frequency but no significant change with the amplitudes is expected.

This step was marked complete when these tests produced consistent, believable results. Samples of this data are included below.

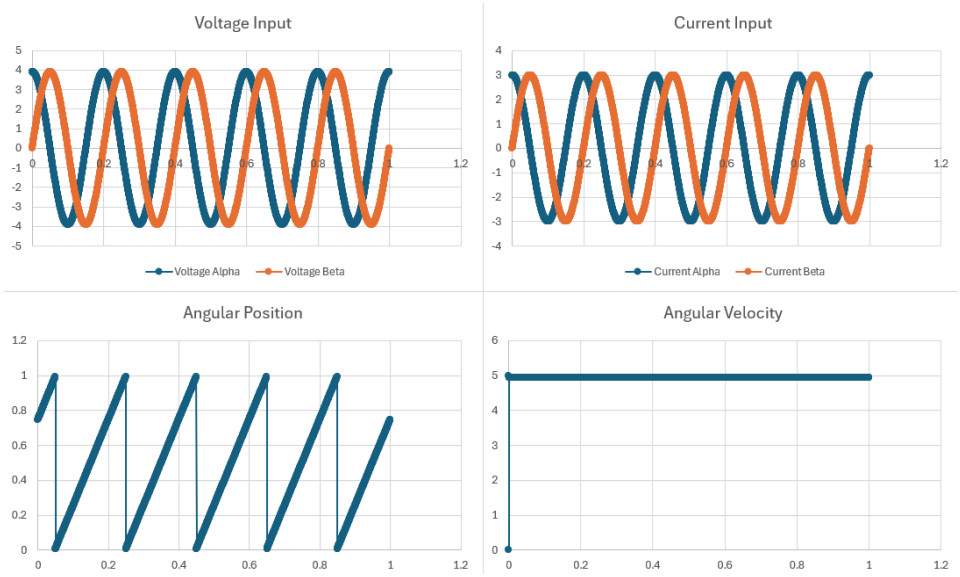


Figure 15: Linear_2 Test Data Input and Output

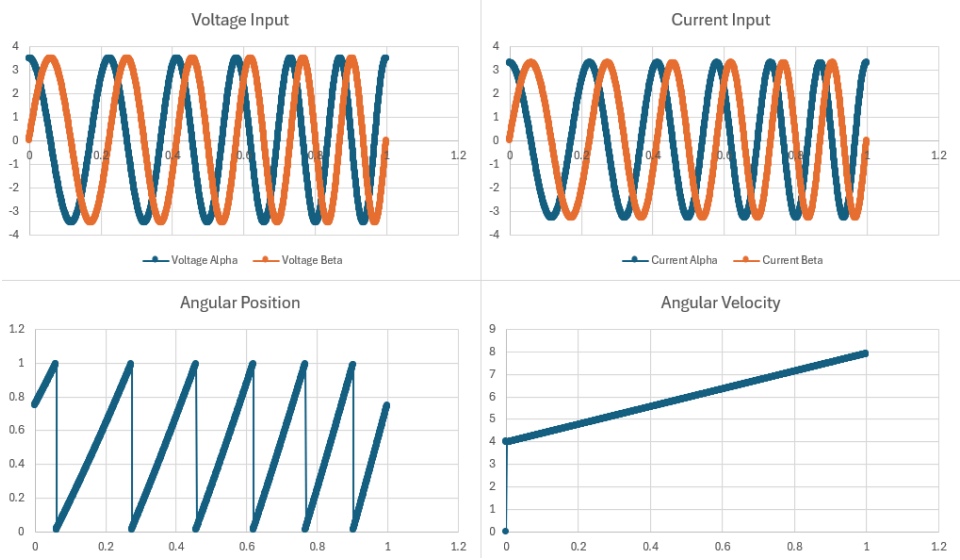


Figure16 : Rising_2 Test Data Input and Output

4.3.3 Non-Ideal Condition Tests

Non-Ideal Test Conditions refer to a mathematical data with noise injected into it. These tests revealed no major faults within the code and that it could handle problematic inputs. These tests were labeled as:

- Low-Noise Data: Low Noise data refers to the added wave having a small amplitude, and thus distortion is minimal.
- High-Noise Data: High Noise data refers to the added wave having a large amplitude, and thus a larger distortion.
- Variance Data: Input waves have mismatched amplitudes, simulating a stator that is unable to get to full current or voltage levels.

This step was marked complete when these tests produced consistent, believable results. Samples of this data are included below.

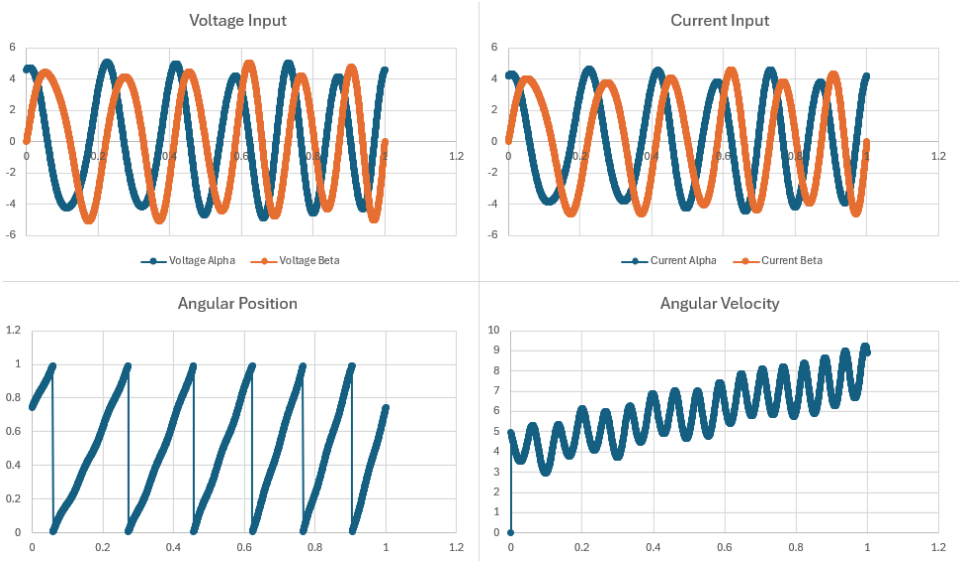


Figure 17: Noise_1 Test Data Input and Output

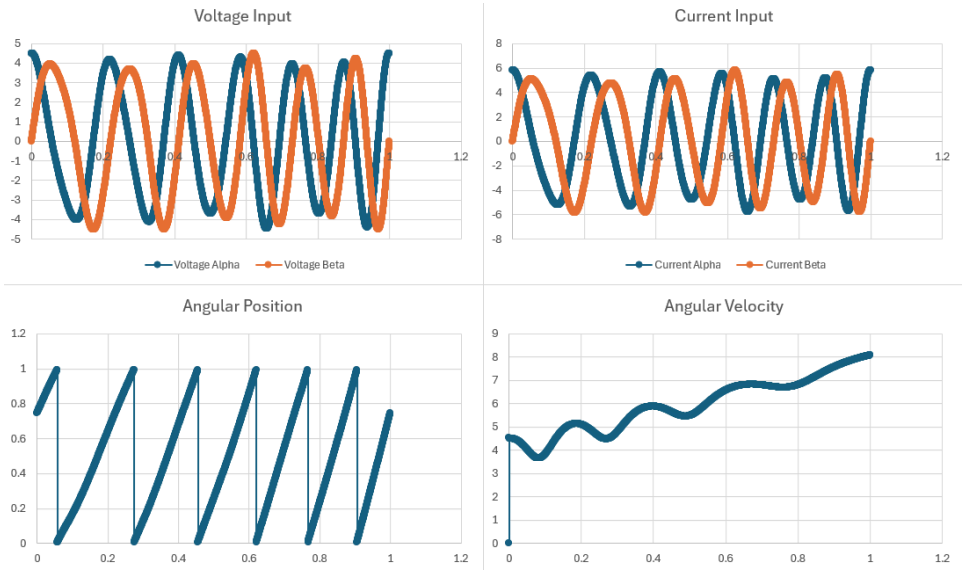


Figure 18: Variance_1 Test Data Input and Output

4.4 Subsystem Conclusion

The testing procedures shown above ensured that the estimator code was functioning as expected by revealing early problems in an environment where it can be most effectively analyzed. Later tests have also demonstrated that the estimator code was even able to make sense from poor or inconsistent input. Possible tests for the code within a purely mathematical environment have been exhausted, and the estimator code is ready for implementation with the other subsystems.

5.0 Porting Subsystem Report - Tyler Hawkins

5.1 Subsystem Introduction

The Ported Solution allows for simple motor control with the use of the software developed, as well as, the F28p65x launchpad and the DRV8323RS driver board. The ported solution is intended to allow the same operations of TI's Universal Motor Control Lab solution for the F28002x board while running on the new F28p65x board.

5.2 Subsystem Details

The porting of the Motor Control Solution can be separated into five sections. The Project Spec Cmd Linker, hal.c, hal.h files, and the hardware setup.

The Project Spec file is used to generate the project folder in the user workspace and includes references to device-specific C2000Ware source files. In this file, the diverboard and all included libraries are defined, as well as, their corresponding file locations within the C200 Motor control SDK.

The cmd Linker files handle the memory reservations and locations of the memory for the board. These reservations include the register locations of the RAM as well as the Flash Memory. In order to correctly define the memory map of the F28p65x board TI's Technical Reference Manual was used to find register locations reserved within the board. The reservations also include the length of the register locations and with the use of CCS the memory allocation can be shown as a percentage of the reserved space used.

The Hal.c and Hal.h files include GPIO, PWM, ADC, and CMPSS modules and defines. These definitions handle the communications between the DRV8323RS driver board and the F28p65x launchpad. By reviewing previously used modules and definitions for the F28002x board as well as studying the differences between TI's Technical Reference Manuals and board schematics of both the new F28p65x and the old F28002x boards, new modules and defines were written In the Hal.c and Hal.h files. One such update was writing all GPIO definitions needed for communication between the F28p65x launchpad and the DRV8323RS board. Using the F28p65x Board schematic PWM, ADC, and CMPSS modules were written in accordance with the connected modules that correspond to the GPIO connections between the launchpad and driverboard.

The Hardware setup of the F28p65x launchpad and DRV8323RS driverboard largely follow the same setup as the F28002x launchpad. After reviewing TI's Technical Reference Manuals on both boards the same pins would need to be bent in order for the solution to run. Additionally, the

Switch configuration was also largely the same with one exception due to a different GPIO pin being used on the F28p65x.

5.3 Subsystem Validation

Due to the project being largely software-based the Subsyestem was greatly validated using CCS and Clearing all errors that occurred during compiling and testing. The project spec file was validated by clearing all errors that occurred when compiling. The cmd Linker files were verified by clearing all related errors as well as checking that memory allocation was less than 90% of the maximum reserved value. Files Hal.c and Hal.h were validated by clearing all errors within the CCS environment. Resolving these errors and validating these files involves the most amount of time. TI changed several general definitions and commands, and therefore definitions had to be chased down within the files, as well as several hours spent on TI forms trying to find the updated naming conventions that needed to be used.

The Full Ported solution was validated by using TI's Universal motor control document. The document includes a section on how to validate the solution, and due to the solution being ported from one board to another, the operation should be the same.

The Project Spec, cmd Linker, Hal.c, Hal.h files, and Hardware have all been verified, and no compiling errors or warnings occur. However, the fully ported solution has been unable to be verified. After conversing with our sponsor, I believe the issue arises from a GPIO definition being incorrect. Our sponsor stated if certain definitions are incorrect, it can prevent the solution from running and the ISR clock does not run within the software. The GPIO pins can be defined as several different things that relate to the connection between the launchpad and driver boards. I believe small adjustments must be made in order to fix this problem.

5.4 Subsystem Conclusion

Currently, the Subsystem does not work as intended, and adjustments and further testing must be made to ensure correct operation. I believe that I have identified the problem area and be working to fix the problems as soon as time permits. I plan to have a working solution once the school year returns in January.