MASTER'S THESIS

Study programme/specialisation: Robot technology and Signal Processing	Spring semester 2023 Open
Author: Lasse Ånestad	
Programme coordinator: Morten Tengesdal Supervisor(s): Morten Mossige	
Title of master's thesis: Automatic Test-system for Resolver Measurement used in Robots	
Credits: 30	
Keywords: Regression testing, resolver, stepper motor, automatic test system, open-loop system	Number of pages: 74 + Attachments/others: Stavanger, 15 July 2023

Abstract

The Serial Measuring Unit (SMU) is a component used to measure motor angles in robot joints at ABB. Currently, the SMU is under development, and ABB has to manually test the SMU to ensure that changes to the software or hardware do not unintentionally introduce bugs or break previously working features. This testing process is time-consuming and unreliable.

This thesis presents the development and implementation of an automatic test system for the SMU in ABB's existing testing environment. The developed automatic test system is an attempt to utilize open-loop control to test functionalities and the quality of angle measurements. A Resolver Test Unit, primarily consisting of a stepper motor and seven resolvers, was designed for this open-loop test system. Test scripts were developed to control the stepper motor and read data from the SMU. Upon analyzing the results, it became evident that the automatic test system lacks the required accuracy and precision to execute high-precision angle measurement tests reliably. Additional testing revealed a non-linear stepping pattern in the stepper motor and a wrongly tuned stepper motor controller. Furthermore, a slight deviation in the mechanical parts of the Resolver Test Unit is suspected, further affecting the resolver measurements.

The test system performed consistently well for tests that did not involve high-precision angle measurements. However, the test system does not qualify for angle and quality measurement tests with the current hardware for open-loop control. Thus, the test system must either be upgraded to higher-quality hardware or changed to a closed-loop system. In addition to one of these requirements, the system needs to be calibrated.

Acknowledgement

I am sincerely grateful to my supervisor, Morten Mossige, who went above and beyond to assist me with my work. I am deeply thankful to Nina Svensen, Bjarne Sandvik, Donjing Liu, and Gultekin Gul for their assistance and support related to the SMU and test system. Additionally, I would like to express my appreciation to Alan Pask for designing the Resolver Test Unit. Lastly, I extend my utmost gratitude to ABB for granting me the opportunity to conclude my five-year-long academic journey in an incredible work environment.

Contents

List of Tables

List of Figures

Abbreviations

ABB	ASEA Brown Boveri
BVT	Build Verification Tester
CCW	Counter Clock Wise
CI/CD	Continuous Integration/Continuous Delivery
CLI	Command Line Interface
CW	Clock Wise
EIP	Excitation In Phase
EPS	Excitation Phase Shift
FPGA	Field-Programmable Gate Array
IPS	Integrated Process System
LQR	Lower Quartile Range
LRC	Longitudinal Redundancy Check
PIB	Process Interface Board
RTU	Resolver Test Unit
SMU	Serial Measurement Unit
SPI	Serial Peripheral Interface
SUT	Software Under Test
TAR	Test Accuracy Ratio
UQR	Upper Quartile Range

Chapter 1

Introduction

The robot industry constantly evolves and develops new robot technology to streamline and adapt to complex automated tasks. One of the critical aspects in the process of development is to be able to test and validate the functionality of the respective systems accurately. Any small software or hardware change can potentially break previously working features. Extensive software regression testing [9] can be a tedious and time-consuming process if carried out manually. Manual regression testing can also introduce human errors, which makes the testing process unreliable. However, regression testing will only test if a specific system is within respective tolerances, not the absolute quality of the product. It's also valuable to gather extra data in parallel with regression testing to see if new iterations on the system have increased or decreased its overall quality.

1.1 Motivation

ABB Robotics Bryne is a high-tech R&D center responsible for the development of ABB's paint robots. They are currently in the phase of developing a brand new robot control system. This new controller is based on innovative, cutting-edge technology. One of the modules included in this control system is the Serial Measuring Unit (SMU). The purpose of the SMU is to measure the motor angles and revolutions in the robot joints, which allows for complete control over the position of a mechanical robot arm. Currently, ABB has to manually validate the correctness of the SMU after small software and hardware changes, which is a tedious and time-consuming task. This master thesis will present the development and installation of an automatic test system for the SMU in a DevOps (CI/CD) setup. Additionally, this master thesis introduces a method to verify if the new iterations on the SMU change the overall quality of individual cards in parallel with the automatic tests.

1.2 Outline

This thesis is divided into the following chapters:

1. **Introduction**
2. **Background**
   - Introducing hardware, software, and testing methods for the test system
3. **Existing work**
   - Provides an overview of the relevant parts of ABB's existing testing environment
4. **Test System Implementation**
   - Test system design
   - Introducing Resolver Test Unit
   - Development of stepper motor controller software
   - Development of Serial Measurement Unit commands
5. **Regression Test Cases**
   - Development of regression test cases
6. **Quality Angle Measurement Test Cases**
   - Development of quality measurement test cases
7. **Results**
   - Regression test results
   - Quality angle measurement test results
8. **Discussion**
   - Discussing the results for the test cases, presenting potential upgrades for the test system and test cases
9. **Conclusion**

Chapter 2

Background

This chapter presents a detailed explanation of all the components and testing methods used in the test system, excluding the Resolver Test Unit, which will be introduced in Chapter 4. The test system consists of the following components:

- Resolver
- SMU
- PIB
- IPS
- Stepper Motor
- Stepper Motor Controller
- Regression Testing
- Quality Angle Measurements Testing

2.1 Resolver

A resolver is a type of electromagnetic sensor that can be used to measure angles and speed for rotating machines. This is the type of sensor ABBs robots use to measure motor positions in their robot arms. Every joint in the robot arms needs a resolver installed on the motor to make it possible to calculate the mechanical position of the robot arm with kinematic equations. The resolver consists of two main components, a stator, and a rotor. The stator has three windings, one primary winding for the transformer, which induces the excitation voltage to the rotor, and two secondary windings as presented in Figure 2.1.

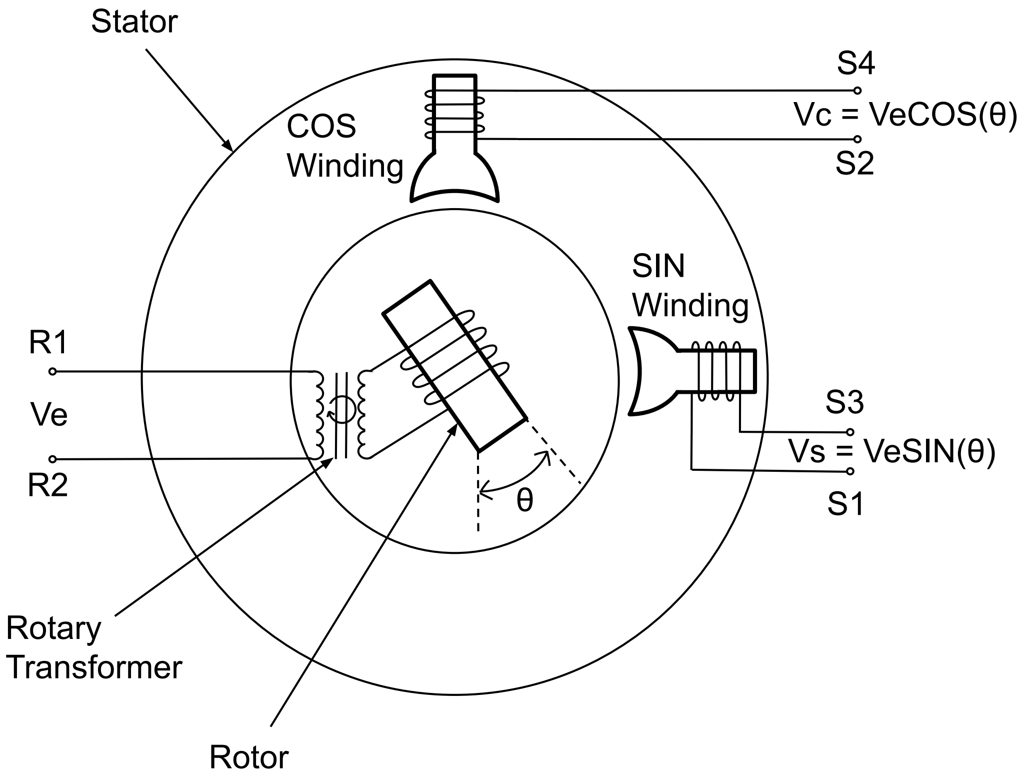


Figure 2.1: Resolver windings and signal.

The secondary windings are placed at a 90-degree angle from each other, which results in a sine and cosine feedback signal. The angle of the rotor determines the amplitude of the sine and cosine feedback signal, and the ratio between these two signals is the tangent of the rotor angle, as shown in equation 2.1

$$\theta = arctan\frac{sin\theta}{cos\theta} = arctan\frac{Vs}{Vc} \tag{2.1}$$

Where Vs is the amplitude of the sine signal, and Vc is the amplitude cosine signal. From hereafter, the sine and cosine feedback signals will be referred to as Y and X, respectively.

2.2 Serial Measuring Unit

The serial measuring unit (SMU) is an embedded system with a central processing unit (CPU), field-programmable gate array (FPGA), and a highly accurate analog-to-digital converter (ADC). This board serves the dual purpose of measuring motor angles and counting motor revolutions in robot joints. The SMU supports the connection of seven resolvers at once, which is to support robot arms with as many as 7-axes. While measuring and counting the resolvers in operation, the SMU continuously reports the data to an axis computer that will calculate the mechanical position of the connected robot. Figure 2.2 presents an image of the SMU.

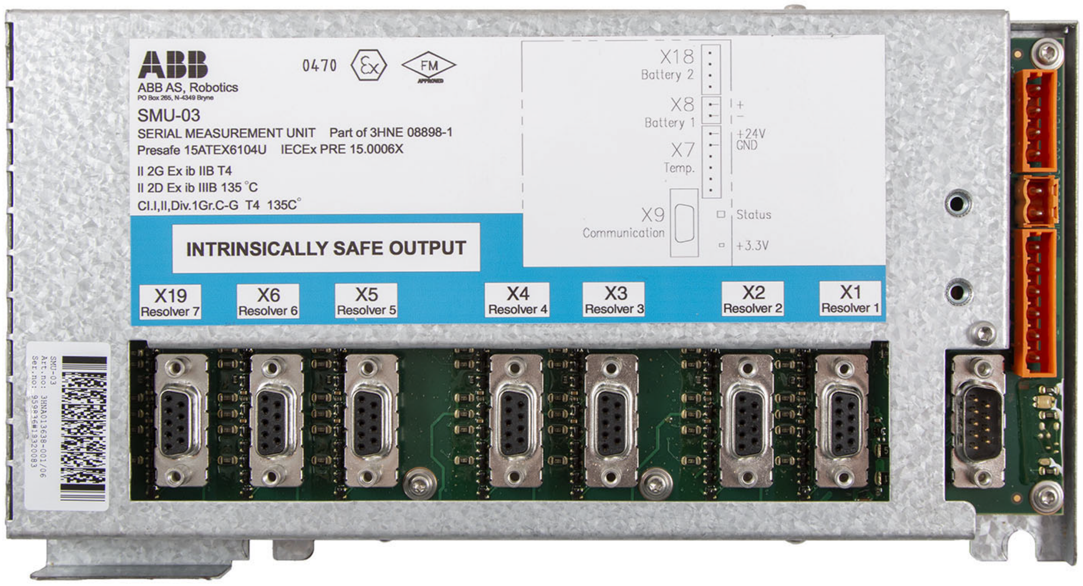


Figure 2.2: The Serial Measuring Unit reprinted from [3] with permission from ABB

2.2.1 Excitation signal and resolver measurements for EPS

Excitation signal
The seven resolvers nodes on the SMU are divided into two groups, one for excitation signal 1 and another for excitation signal 2. Group 1 consists of resolvers 1-3, and group 2 resolvers 4-7. The excitation signals for both groups of resolvers can be configured to two different modes, excitation phase shifted (EPS) and excitation in phase (EIP). Figure 2.3 displays an example of EPS modus for the excitation signal.

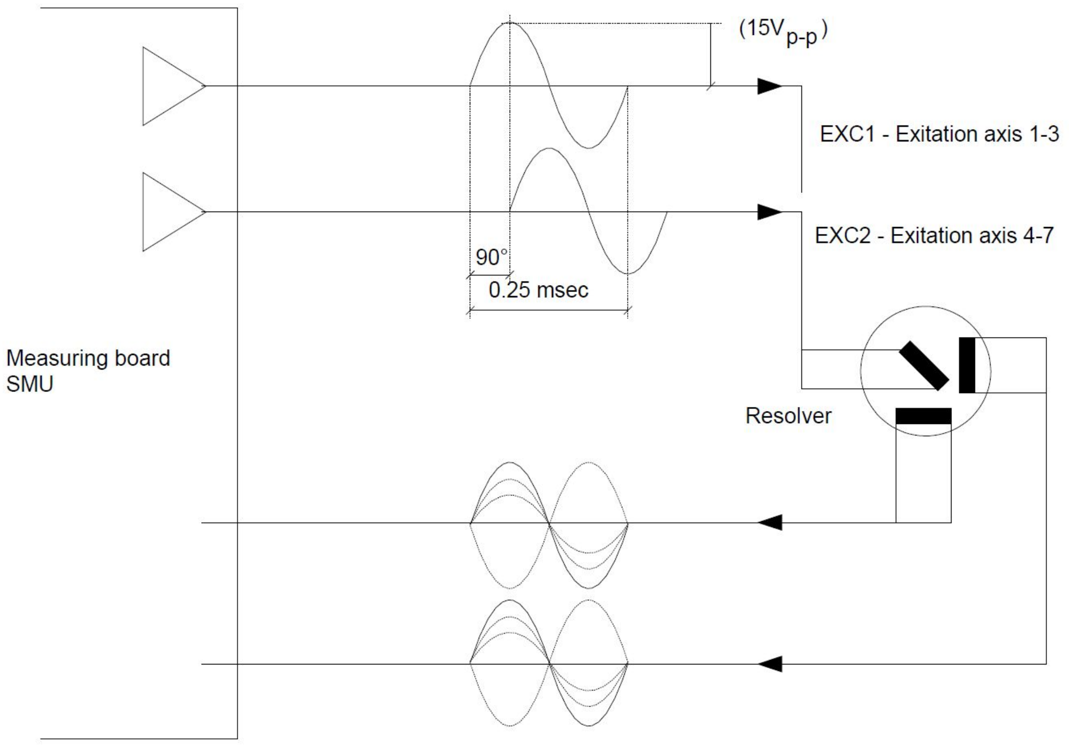


Figure 2.3: Resolver Excitation overview reprinted from [2] with permission from ABB.

Resolver measurement sequence

The FPGA measures one resolver every $63\mu s$, thus making it possible to measure all seven resolvers within a time period of $500\mu s$ in EPS mode. Every resolver node has two input channels dedicated for reading X and Y values. The readings are done at the top and the bottom of the resolver feedback signals. Two different sequences of measuring the resolvers are required to get valid results for EPS and EIP, respectively. Figure 2.4 displays a possible sequence of resolver measurements used in EPS modus.

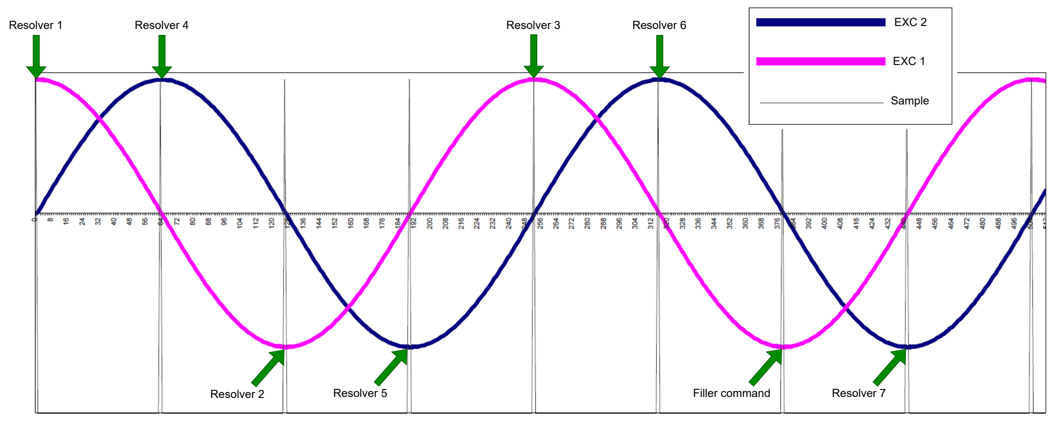
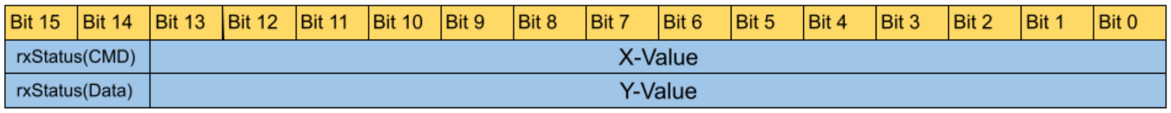


Figure 2.4: Sequence of resolver measurements in EPS modus. Altered reprint from [2] with permission from ABB

This sequence of resolver measurements ensures that the SMU does not samples zero crossings of the feedback signal. Measuring zero crossings will result in useless measurements. The filler command, as shown in Figure 2.4, is to create an offset to compensate for the fact that there is one more resolver in excitation signal group 2 than in group 1. The only requirement for the filler command is not to be a resolver measurement command.

Resolver measurements

As mentioned in the previous section about the resolver measurement sequence, each resolver node has two input channels for reading X and Y values. These two values are stored in specific 16-bit memory slots. The two left-most bits represent status, and the next 14 bits represent the resolver values, as shown in Figure 2.5.



Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
rxStatus(CMD)		X-Value													
rxStatus(Data)		Y-Value													

Figure 2.5: The structure of a resolver measurement memory slot.

The resolver values are stored in this memory in a two's complement [6] format to properly represent the positive and negative values of the resolver signal. Thus, the maximum and minimum values are 8191 and -8192, respectively. However, internal scaling on the excitation signal ensures that the resolver values will not use the full range of these limits to avoid saturation. The scaling is different on all resolvers. Consequently, the resolvers will have different working ranges.

2.2.2 Revolution counters and battery modus

The SMU has a revolution counter for every resolver node. It's crucial to store information about how many revolutions each and every motor have done to be able to calculate the position of the mechanical arm. If the SMU loses power for just a split second, the internal SMU revolution counter is lost. Additionally, If the mechanical arm is moved while the SMU has lost its power, the revolution counter does not update. To counter this problem, a battery will keep the SMU powered if the main power source is disconnected. The SMU loses some functionality when the main power is lost. However, the SMU can still detect the resolver's current quadrant position and count motor revolutions in battery modus.

2.2.3 Communication

Usually, in production, the SMU communicates with an Axis computer over an RS-422 link. However, the SMU will communicate with a PIB over SPI for the test system. Both of these interfaces utilize ring controller protocol, an ABB exclusive communication protocol [1].

2.2.4 Ramp test

The ramp test is part of a startup sequence in ABB's older generation of robots. If this test failed, the robot would not start. The ramp test determines if the ADC accurately converts an analog signal into a corresponding digital value that consistently increases or decreases at a linear rate. Notably, this test is only performed on node 7. ABB's newer generation of robots does not have the ramp test as a startup condition. However, for the new versions of the SMU to be compatible with the older generation of robots, the SMU must be able to perform this test. In the newest SMU iteration, the firmware simulates the ramp test process virtually. Therefore, it is necessary to include the verification of the virtual ramp test in the regression test cases.

2.3 Intergrated Process System

Integrated Process System (IPS) is an embedded and distributed software suite for process control used in ABB. This application controls the process by reading sensors, control motors, paint flow, airflow, the timing of paint flow, etc. Notably, this software suite offers distributed computing frameworks to efficiently implement new drivers and commands for parallel distributed systems.

2.4 Process Interface Board

The Process Interface Board (PIB) is the central interface between the control system and paint application equipment. The general purpose of this board is to link the control system and process equipment. Additionally, the PIB is running IPS applications. Figure 2.6 displays an image of the PIB.

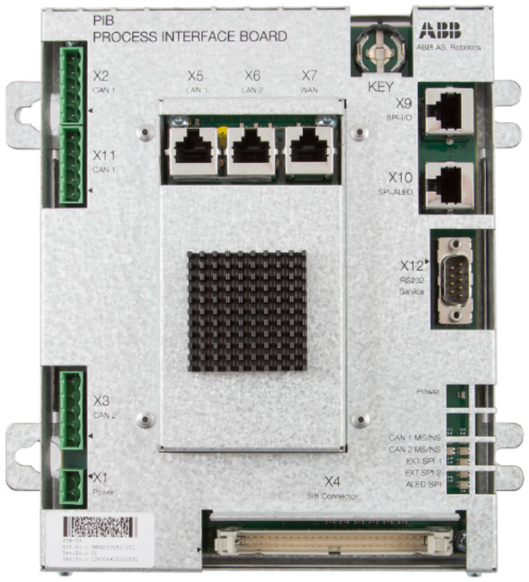


Figure 2.6: This figure shows the PIB. reprinted from [3] with permission from ABB.

This board will be the interface between the test computer and the SMU.

2.5 Stepper motor

To test the accuracy of resolver measurements on the SMU, a stepper motor is used to control the resolvers in an open loop configuration. This stepper motor is a 2-Phase stepper motor from Tamagawa A with a fundamental step angle of 1.8°. Figure 2.7 displays the Tamagawa stepper motor.

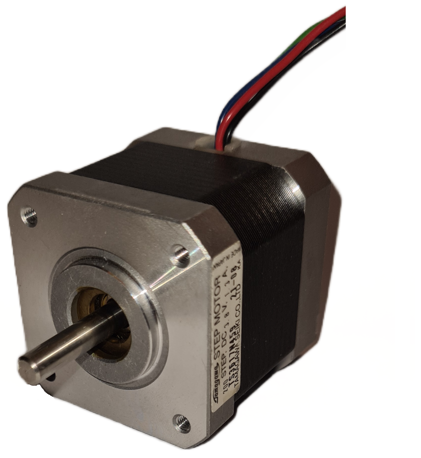


Figure 2.7: This figure shows the Tamagawa 2-Phase stepper motor.

This stepper motor is salvaged from another component at ABB, which means it's not bought exclusively for the test system. The positional accuracy for this stepper motor is 1.8° ± 5%, which translates to ±0.09°.

2.6 Stepper motor controller SMSD-4.2Modbus

SMSD-4.2Modbus [16] is an advanced stepper motor controller that offers direct control of a stepper motor using the Modbus communication protocol [14]. This controller supports USB and RS-485 communication interfaces to both PLC and PC. This controller allows for real-time stepper motor control with commands from a computer through Modbus. Additionally, the controller supports seven microstepping configurations, ranging from 1/1 up to 1/256, allowing for high-resolution motor movements. Furthermore, it features discrete outputs that can be controlled through an internal user program. For visual reference, Figure 2.8 displays the SMSD-4.2Modbus stepper motor controller.

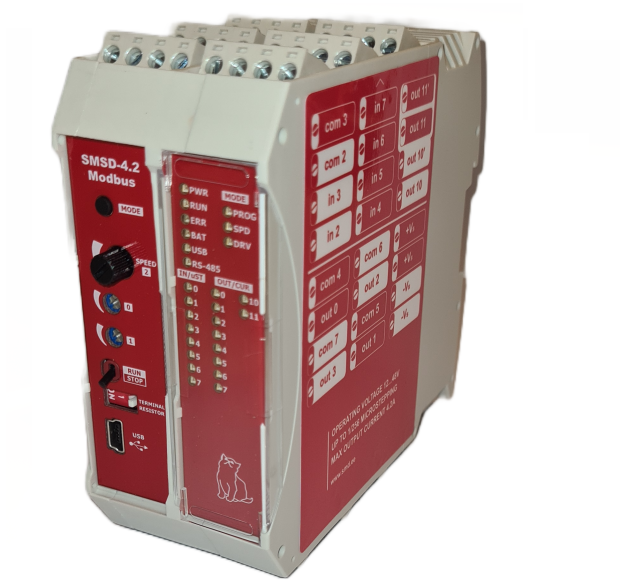


Figure 2.8: SMSD-4.2Modbus stepper motor controller.

The SMSD-4.2Modbus controller will hereafter be referred to as SMSD.

Output current control for microstepping
Stepper motors require specific current levels to function optimally; in the case of the Tamagawa stepper motor, 1.3 amperes per phase. This stepper motor has windings, hereafter referred to as winding A and B. Initially, for fullstepping, one winding is fully energized at the time. However, for microstepping, the controller subdivides a fullstep into microsteps. Thus, the current in the coils should change in increments based on the factor of microstepping. To achieve perfect microsteps, the current in winding A and B should change in increments approximating a sine and cosine wave, respectively, as shown in Figure 2.9.

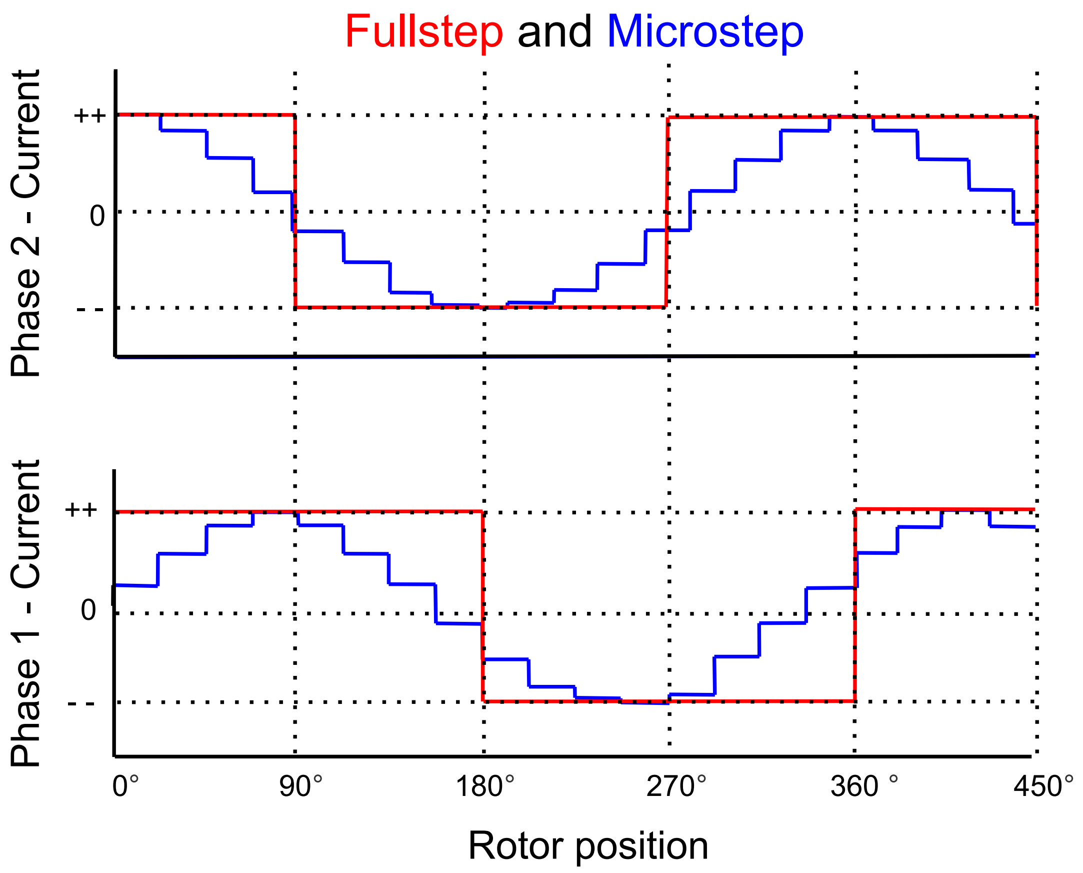


Figure 2.9: This figure shows the current waveform for both phases of fullstepping and microstepping.

Another factor regarding the smooth and precise motion control is the current decay in the windings. In short, there are three current decay modes: fast, slow, and mixed. These modes essentially determine the smoothness and precision of the transition between microsteps. A more comprehensive explanation of the current decay mode can be found in [7]. The SMSD-4.2Modbus controller utilizes mixed decay mode, a combination of fast and slow decay.

2.7 Automated Regression Testing

It's essential to have a method of verifying if changes in hardware or software affect previous working features in the SMU. This is where automated regression testing comes into the picture. Automated regression testing is a software testing technique that is used to ensure that changes done on the respective system do not unintentionally introduce bugs or break previously working features. This is achieved by running automated test scripts that can be executed repeatedly, accurately, and quickly.

2.8 Quality Angle Measurement Testing

Automatic regression testing is a quick and robust method of verifying if quick changes done to the software do not break previously working functions. However, when upgrading or changing the hardware on the SMU, it's also interesting to see if the overall quality of the measurements has changed. In this context, quality is defined as the accuracy and precision of the angle measurements. By collecting angle measurement data over more extended periods, the data will eventually converge to a very precise estimation of the true quality of the SMU. Additionally, to ensure getting a reliable quality estimation for one specific iteration of the SMU, both of the following requirements must be fulfilled:

1. A test system that is significantly more accurate than the SMU
2. Data collection of multiple SMUs of the same iteration

The data cannot be trusted if the test system is less accurate than the SMU. Furthermore, by only collecting quality measurement data for one prototype of a specific iteration of the SMU, it's possible getting unrealistic quality estimation due to hardware tolerances in production. Therefore it's essential to test as many units as possible to cover the full range of hardware tolerances.

Chapter 3

Existing Work

This chapter provides an overview of the relevant parts of ABB's existing testing environment. The testing environment consists of the following parts:

- **Devops:**
  - Automate and improve software development
- **Release Pipeline:**
  - Automatic testing stage for software on target
- **Python Test Framework:**
  - Framework for developing test cases for the Release Pipeline

In addition to the test environment, this chapter also presents ABB's Command Line Interface framework, which will be the platform for developing SMU commands.

- **IPS Command Line Interface Framework:**
  - Framework for developing commands and drivers for ABB's platforms

3.1 ABB Software Development Environment

DevOps is a method or use of tools to automate the work of software development (Dev) and IT Operations (Ops). This method improves and speeds up the system development life cycle, which essentially means planning, creating, testing, and deploying software. ABB uses the Azure DevOps services from Microsoft to speed up and improve ABBs existing workflow. This service provides cloud-hosted repos, pipelines, test plans, and much more. ABB uses the Continuous Integration/ Continuous Delivery (CI/CD) environment for their software development as shown in Figure 3.1

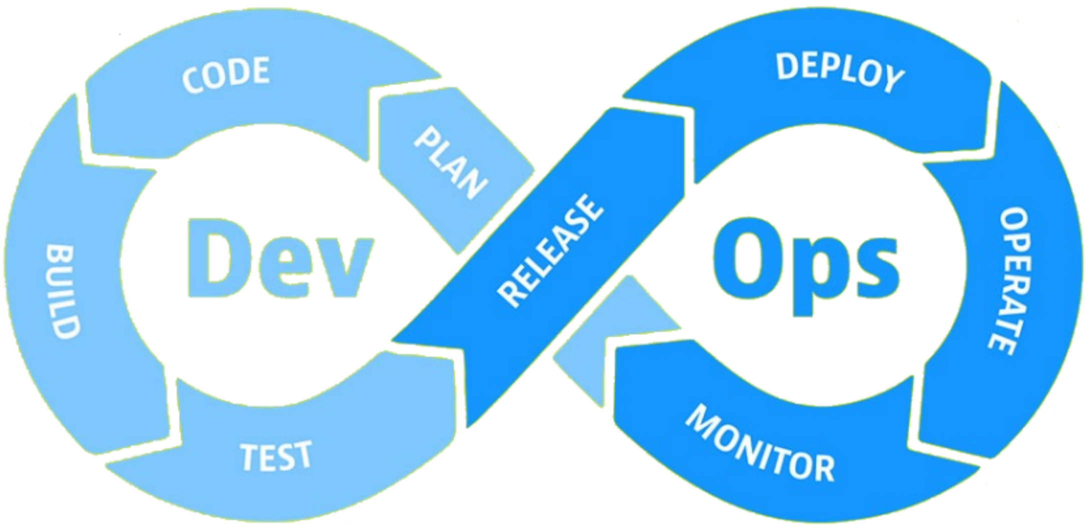


Figure 3.1: "DevSecOps" by Rezadlt. Retrieved from [13]. Licensed under a Creative Commons Attribution-Share Alike 4.0 International license, the Image is cropped and color adjusted.

From Figure 3.1, the two subsequent stages, build and test pipeline, are two different automated test pipelines dedicated to testing the respective software. The build pipelines checks if the changes pushed to the repository are able to build to the relevant platforms. The release pipeline is the external testing environment at ABB dedicated to test software under test (SUT) on target. The test system for the SMU will be integrated into the release pipeline.

3.2 Release Pipeline

ABB has dedicated computers to run tests for the release pipeline called Build Verification Testers(BVT). These computers are connected to cabinets with the necessary components and systems needed to test specific software. An illustration of how the release pipeline works is presented in Figure 3.2

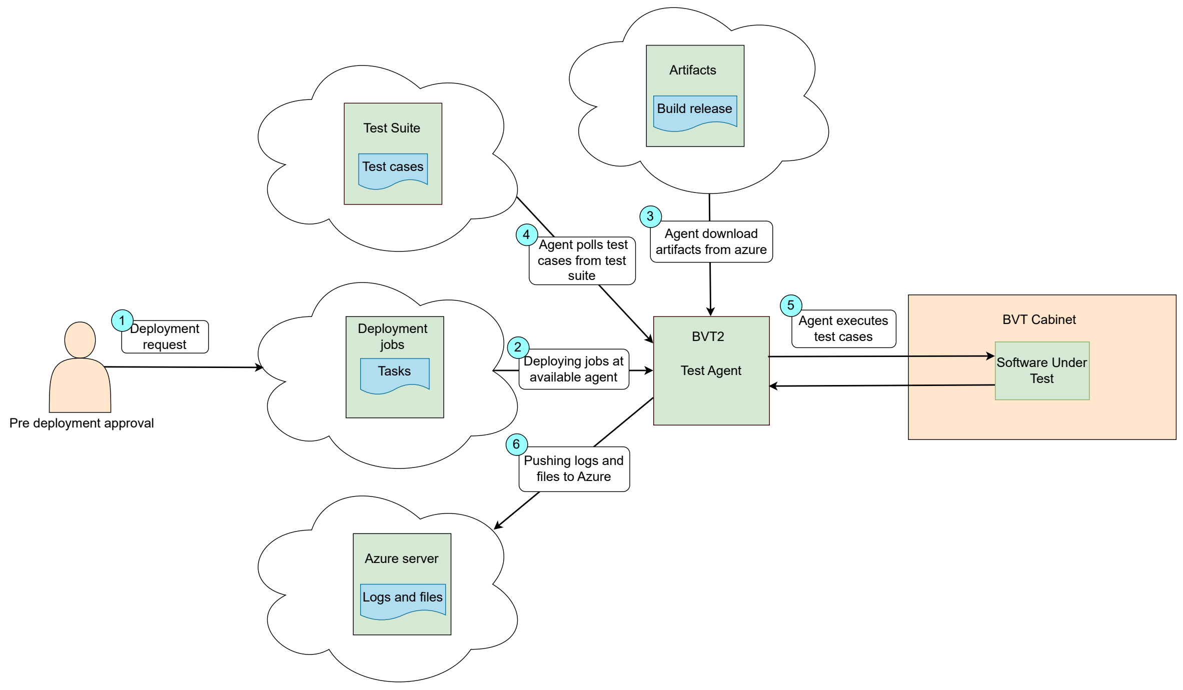


Figure 3.2: Release pipeline overview.

The first step is the deployment request, which can be triggered manually or automatically by being time scheduled. The deployment jobs contain specific tasks that the test agent must do to ensure the test cases are executed in the required environment. Once the necessary test environment is established, the agent proceeds to execute the test cases. Upon completion of executing the test cases, the agent uploads files and logs the results to Azure.

3.3 Test Framework

The test framework at ABB is a Python environment dedicated to making test cases for the release pipeline. The module dependency diagram illustrated in Figure 3.3 shows a high-level structure of the relevant parts of the test framework.

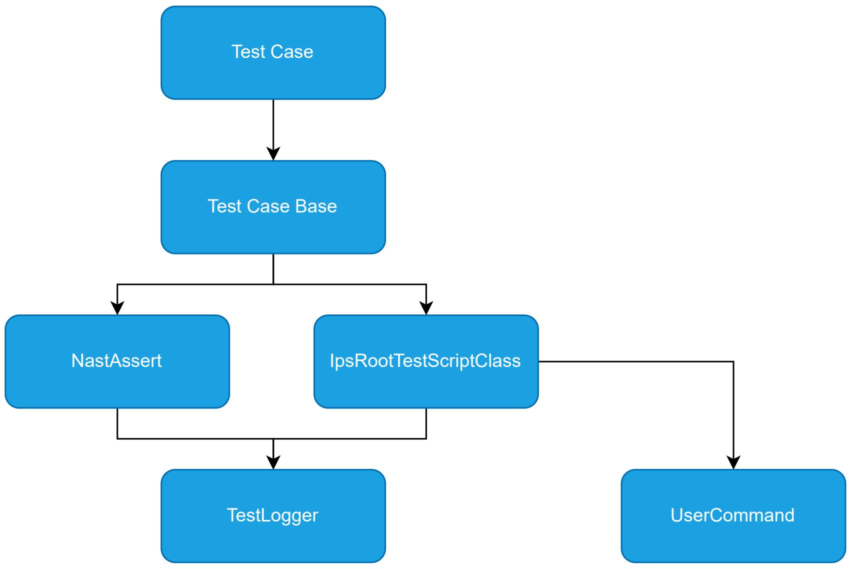


Figure 3.3: Module dependency diagram for the test framework in Python.

The Test Case module is the executable test case file placed in the respective test suites. This module calls methods from the test case base for performing the actual tests.

Test Case Base is uniquely designed for the system or software under testing and contains methods for automating specific tests, and defines the boundaries or expected outcomes for the individual test cases using NastAssert.

NastAssert is a customized assert test [11] module designed for the test framework at ABB. This module contains methods for checking boundaries and expected outcomes for the respective tests.

IpsRootTestScriptbClass initializes the whole test instance, defining the IPS platform, software drop location, logging directory, and so on.

UserCommands is the interface between the Python test environment to the IPS platform. This module contains methods for targeting the IPS console and reading responses.

TestLogger module for logging test results.

3.4 IPS Commandline Interface

The IPS application used on ABBs platforms supports two types of command-line interfaces (CLI), one for IPS itself and one for the platform the application is running on. On startup, the IPS application detects which card it's running on and will initialize based on that. Commands and drivers related to the detected card will then be available. ABB has an established CLI framework in C++ as a foundation for implementing new commands and drivers. The sequence diagram in Figure 3.4 attempts to illustrate a simplified and generalized user case of the CLI framework.

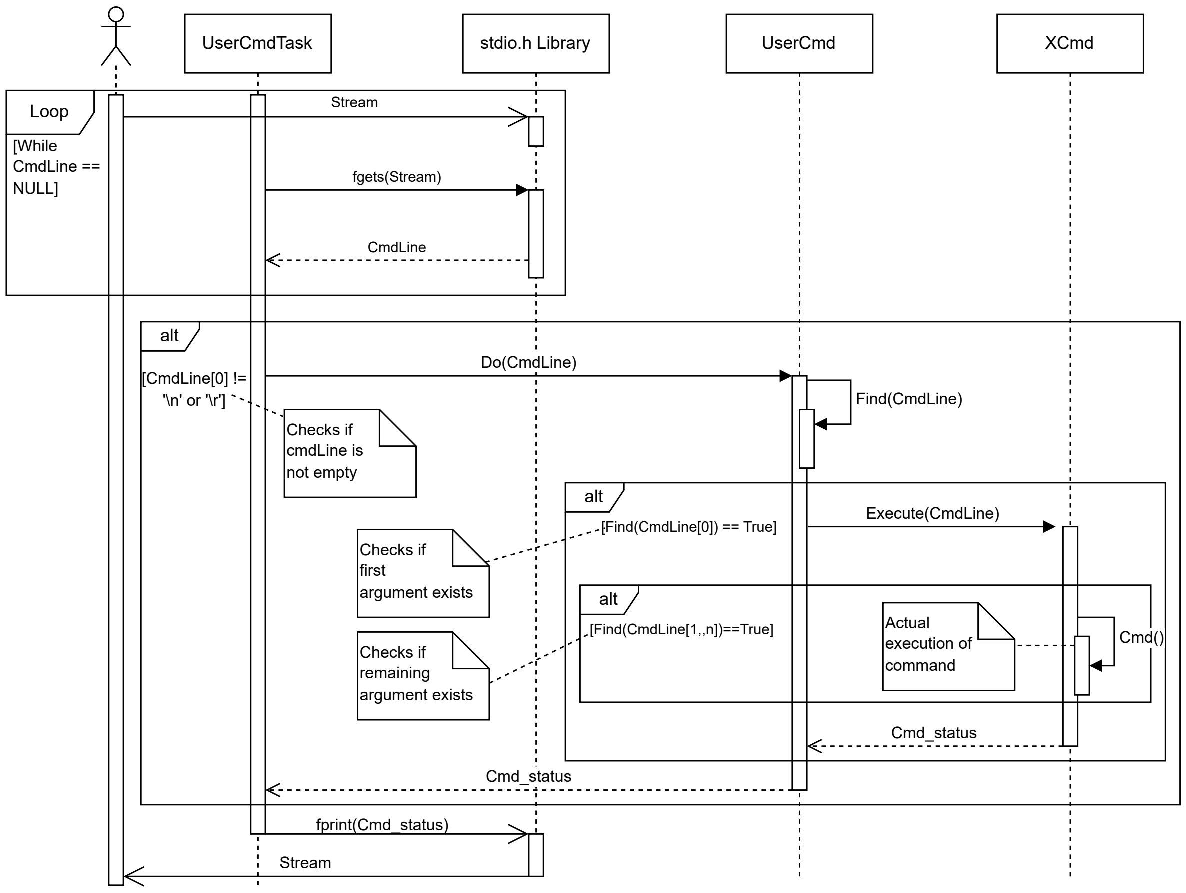

UserCmdTask
stdio.h Library
UserCmd
XCmd
Loop
[While CmdLine == NULL]
Stream
fgets(Stream)
CmdLine
alt
[CmdLine[0] != '\n' or '\r']
Do(CmdLine)
Find(CmdLine)
Checks if cmdLine is not empty
alt
[Find(CmdLine[0]) == True]
Execute(CmdLine)
Checks if first argument exists
alt
[Find(CmdLine[1,,n])==True]
Actual execution of command
Cmd()
Checks if remaining argument exists
Cmd_status
Cmd_status
fprint(Cmd_status)
Stream

Figure 3.4: Single SMU command user case sequence diagram.

The class xCmd is a generalized command class representing all the IPS platforms at ABB. The SMU is usually connected to an Axis computer under operation, which means IPS initially does not have the drivers to control an SMU card when initialized on a PIB. Therefore, an SMU test driver and commands will be implemented in the section of xCmd in relation to the SMU test system.

Chapter 4

Test System Implementation

This chapter introduces the design and implementation of the automatic test system for the SMU. The test system can be split into two main parts, hardware and software. Firstly, for the hardware part, this chapter introduces the following subsections:

- **Resolver Test Unit:**
  - The core component for the test system
- **Test system physical design:**
  - Hardware layout for the BVT cabinet
- **Test System Overview:**
  - Complete overview of the test system

Secondly, for the software part of the test system, this chapter introduces the following subsections:

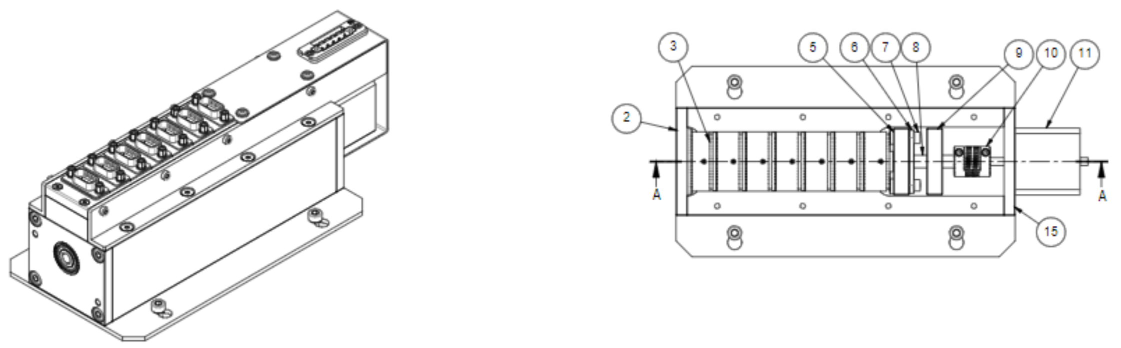
- **SMSD software implementation:**
  - Software implementation for controlling the stepper motor
- **SMU test commands:**
  - Commands developed for controlling the SMU

4.1 Test System Hardware

This section presents the core component Resolver Test Unit (RTU), test system hardware architecture, and full test system overview.

4.1.1 Resolver Test Unit

The core component of the test system is the RTU. This component is specifically designed to control seven resolvers in an open-loop. Figure 4.1 displays the design overview of the RTU.



(a) This subfigure shows the RTU Design overview.

(b) This subfigure shows a labeled inside overview of the Resolver Test Unit with the top part of the metal housing removed.

Figure 4.1: This figure shows the outer and inner design of the RTU.

The upcoming explanation of the RTU design exclusively refers to Subfigure 4.1b with labeled circles.

The RTU primarily consists of a stepper motor ⑪ and seven resolvers ③. The stepper motor's primary axle connects to a secondary axle ⑧ using an axle adapter ⑩. The seven resolvers are mounted on the secondary axle and connected to the female d'sub connector plugs located on top of the metal housing. The shaft connector points ② and ⑤ are produced with strict tolerances to ensure consistent stator-to-rotor positioning.

4.1.2 Test system physical design

The test system is installed on a metal plate dimensioned to fit into one of ABB's BVT cabinets. The SMU test system will be installed into the long-term testing cabinet. This BVT cabinet is designed for testing and collecting data on equipment and systems subjected to prolonged usage. Figure 4.2 shows an overview of the test system on the metal plate.

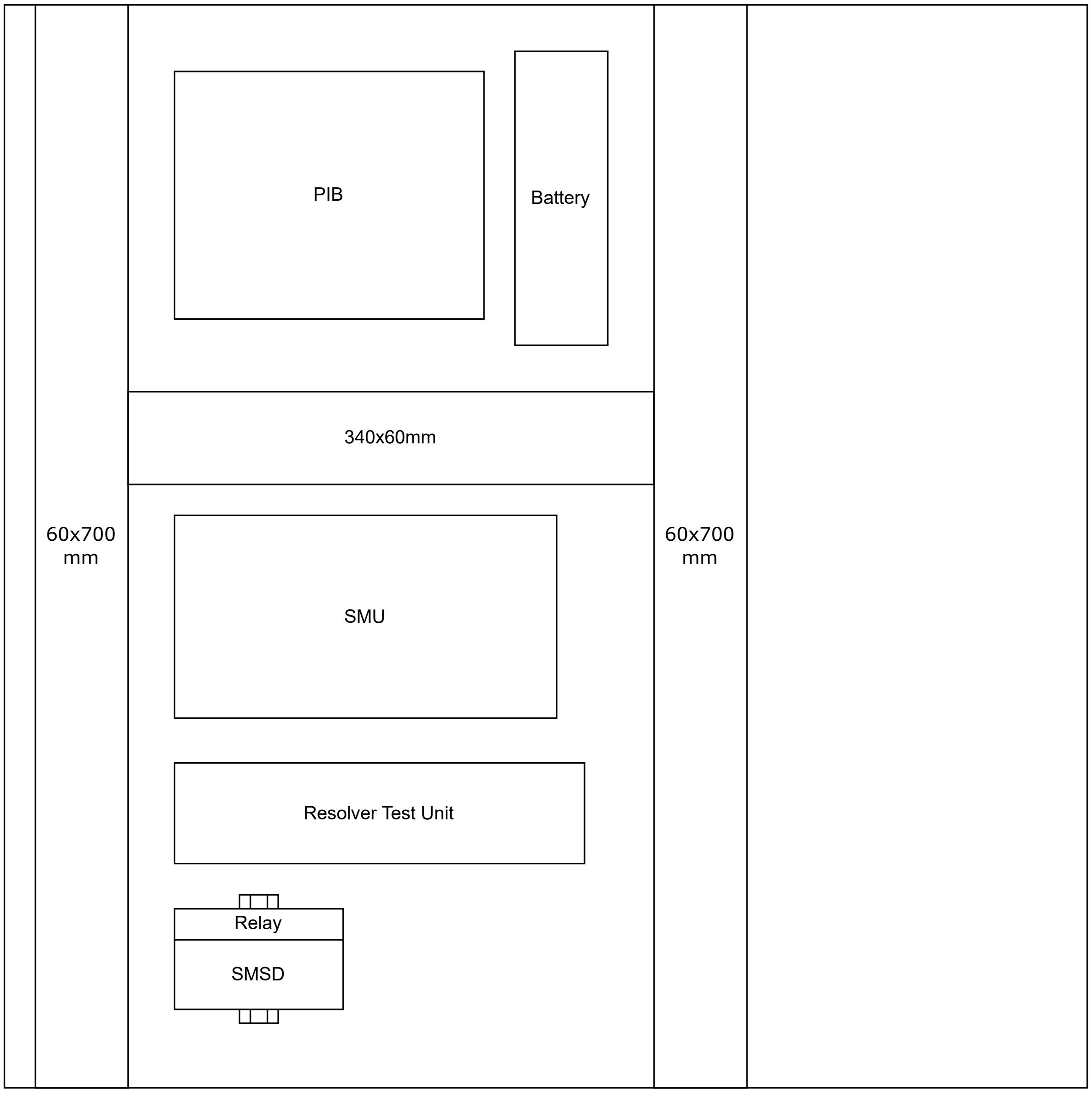


Figure 4.2: Physical test system layout on the metal plate for the BVT cabinet.

The PIB is placed at the top left section of the plate to make it easier to reach for the ethernet and serial D-Sub cable, which are coming from the backside of the cabinet. The SMU and

RTU are placed beside each other to make it easier to connect the seven resolvers from the RTU to the SMU. The same goes for the SMSD as well, which is connected to the stepper motor on the RTU. The rectangular shapes labeled with dimensions are cable canals for cable management. Additionally, this particular physical layout aims to maximize space utilization for potential future expansions or the installation of another test system in the available area. Figure 4.3 displays the real implementation of the hardware.

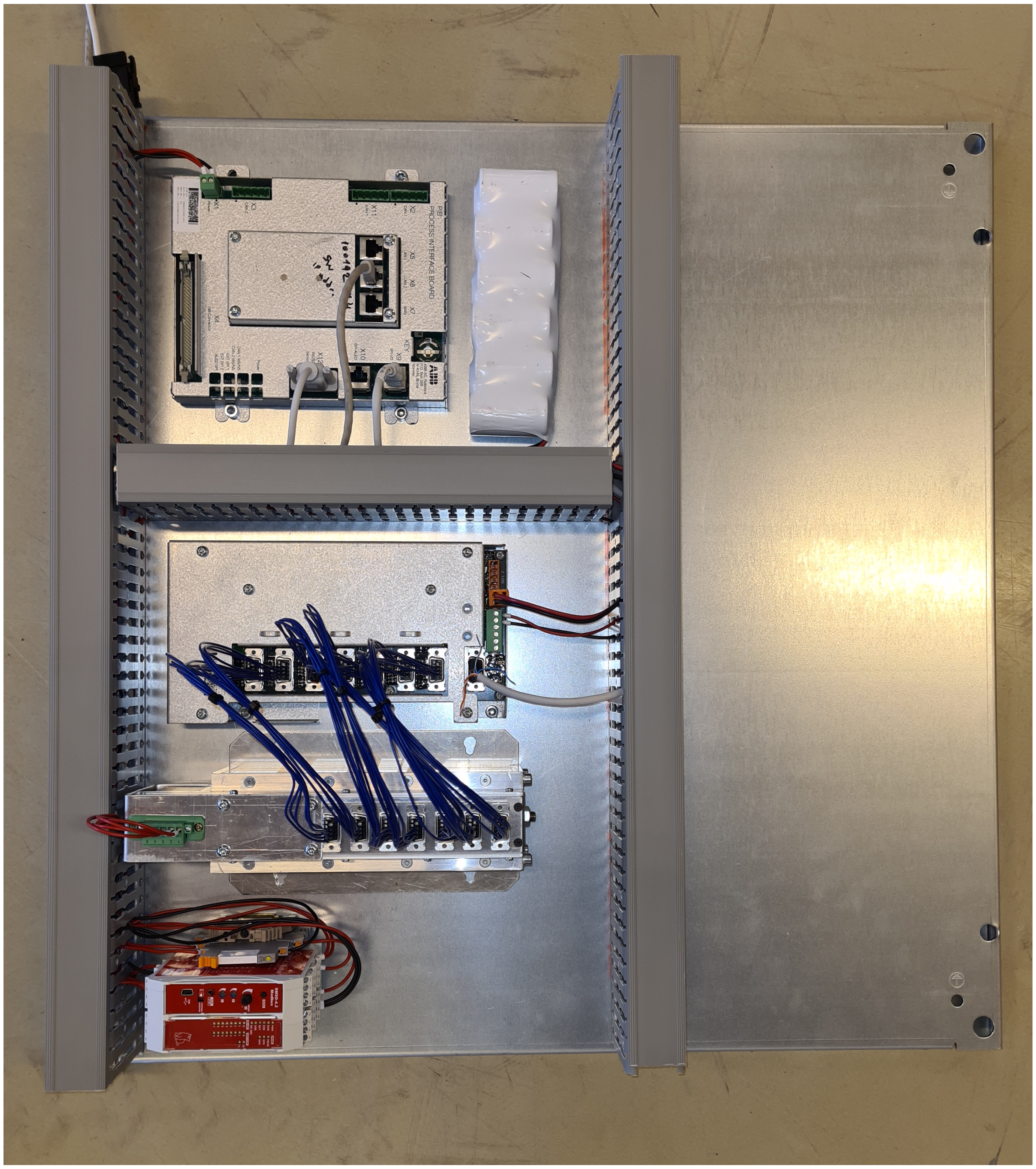


Figure 4.3: This figure shows the real physical layout of the implemented test system.

4.1.3 Test system overview

Figure 4.4 displays the test system overview.

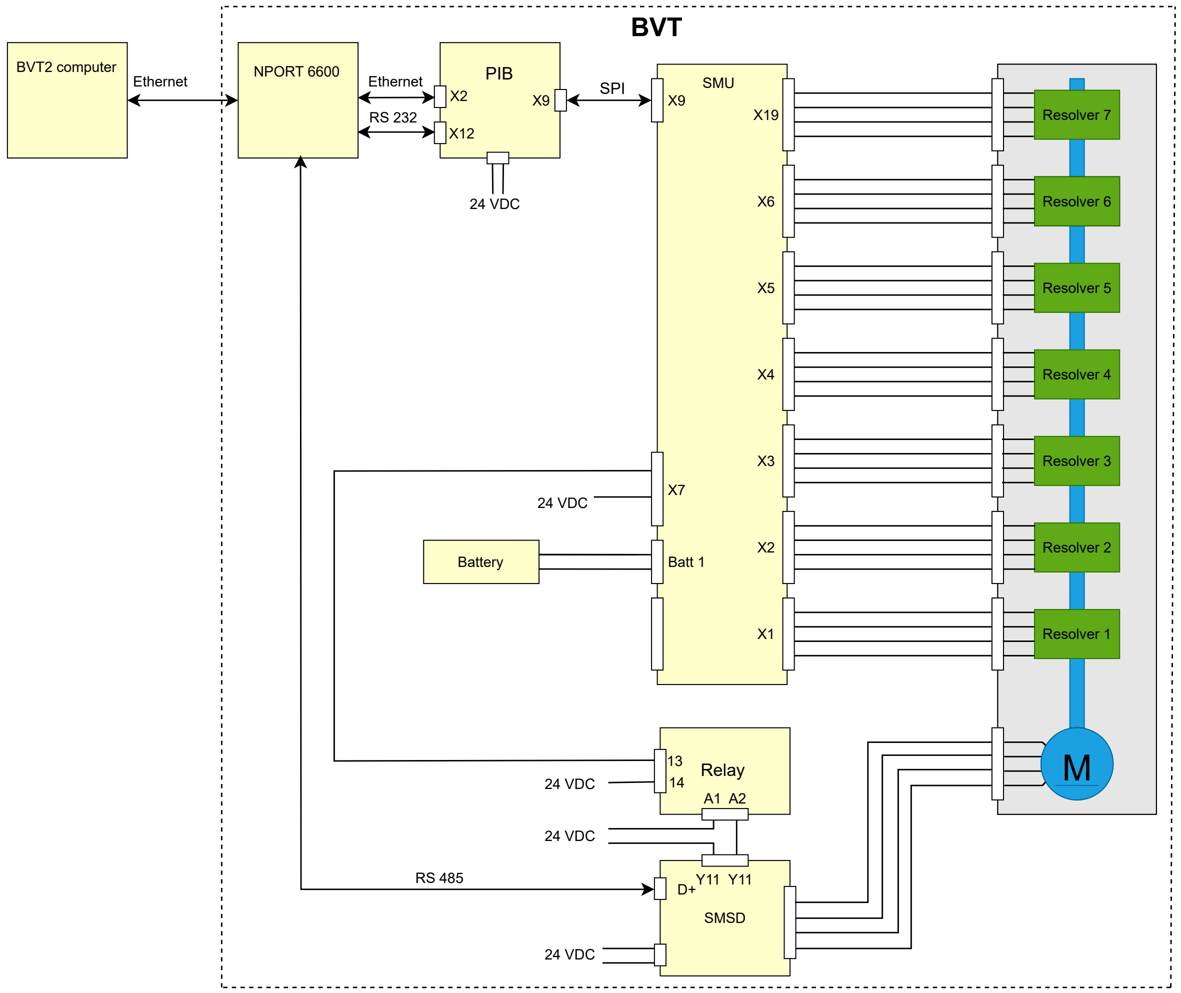


Figure 4.4: Test system overview.

The NPORT 6600 device is a terminal server that is the central hub for connecting and controlling serial-connected devices from every test system in the BVT cabinet. This terminal connects the BVT2 computer with the test systems in the BVT cabinet. Initially, one of the physical outputs of the SMSD was planned to control the main power of the SMU to force it into battery modus. However, the internal output transistors in the SMSD caused a voltage drop, leading to an insufficient power supply to the SMU. This problem was solved by using a relay to control the power source to the SMU. The relay is then controlled by the SMSD, as shown in figure 4.4.

4.2 Test System Software

This section illustrates the integration of the SMSD and SMU test commands software into the existing test framework presented in section 3.3.

4.2.1 SMSD software

This section introduces the implementation of the software used to control the stepper motor for the SMU test system. The implemented software consists of two modules, one for stepper motor commands and one for communication. The commands and communication modules are implemented in Python to make it easier to integrate into the test framework at ABB. The two green modules in the updated module dependency diagram in Figure 4.5 are the addon of the two new SMSD modules for the stepper motor.

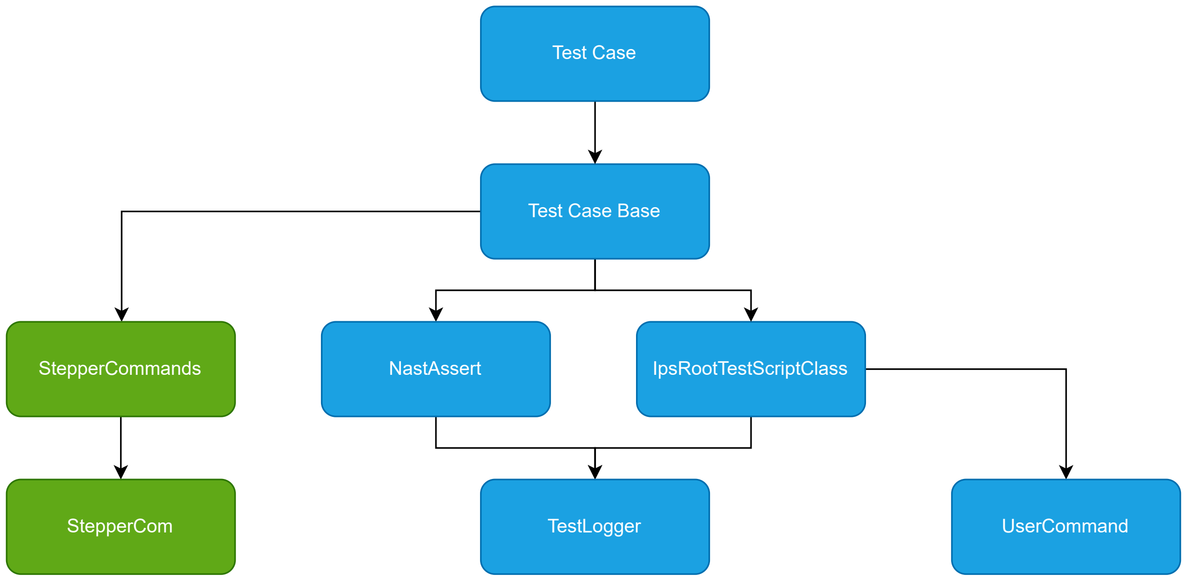


Figure 4.5: Module dependency diagram. The green modules represent the new modules integrated into the existing.

The stepperCommands module mainly consists of methods for SMSD commands constructed on a Modbus ASCII frame format. On the other hand, the stepperCom module serves as the interface between the stepper motor commands and the SMSD controller. The Test Case Base utilizes the methods from the StepperCommands module when designing test methods.

Stepper command table

A full list of the relevant implemented stepper motor commands is shown in Table 4.1.

Stepper motor commands	Argument	Operation
stop		Stepper motor enters holding mode
acc	<value>	Set stepper motor acceleration in <value> pps
speed	<value>	Set stepper motor speed in <value> pps
cw		Set stepper motor rotation to clockwise direction
ccw		Set stepper motor rotation to counter clockwise direction
status	status type	Returns current state of the motor
microstep	<factor>	Set microstepping <factor>, possible factors [1, 2, 4, 8, 16, 32, 64, 256, 512]
anglego	<angle>	Displace by <angle> degrees
stepsgo	<steps>	Displace by <steps>
SMU	<state>	Change SMU state ["on", "off"]

Table 4.1: Stepper motor commands table.

The stepperCommand module contains more commands than presented in Table 4.1. A comprehensive list of all implemented commands can be given on request.

Figure 4.6 shows a sequence diagram of the program flow when calling the positional displacement command "anglego".

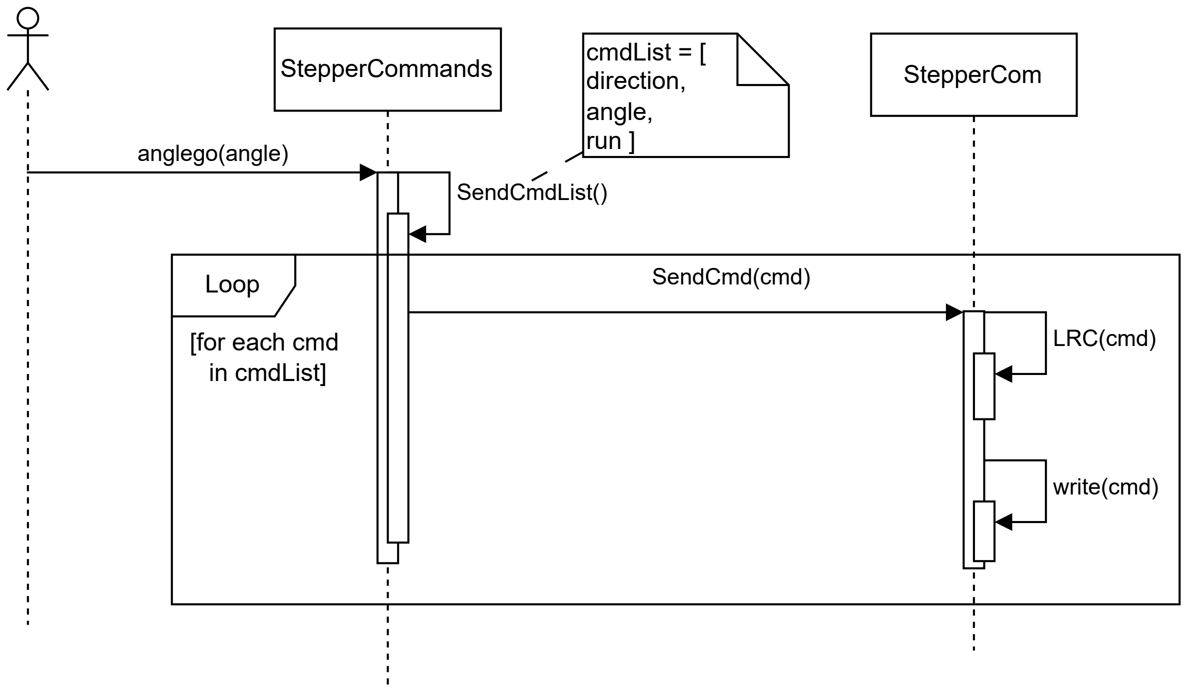


Figure 4.6: Sequence diagram of a single user case for command anglego.

The anglego method consists of two rotational parameters and a run command: direction, positional displacement, and run. Figure 4.6 refers to these as "cmd" in the "cmdList." Every "cmd" has to go through a Longitudinal Redundancy Check (LRC) [8], which is a form of error checking used to ensure data integrity during transmission. First, the method writes the two parameters to their respective registers before executing the run command. When the run command is initiated, the stepper motor will run based on the defined parameters in the registers.

4.2.2 SMU test commands

The SMU commands are implemented in the existing CLI framework, As mentioned in section 3.4. These commands are developed to communicate and control the SMU for the test system. The red module introduced in Figure 4.7 is where the new SMU test commands will be implemented concerning the test framework.

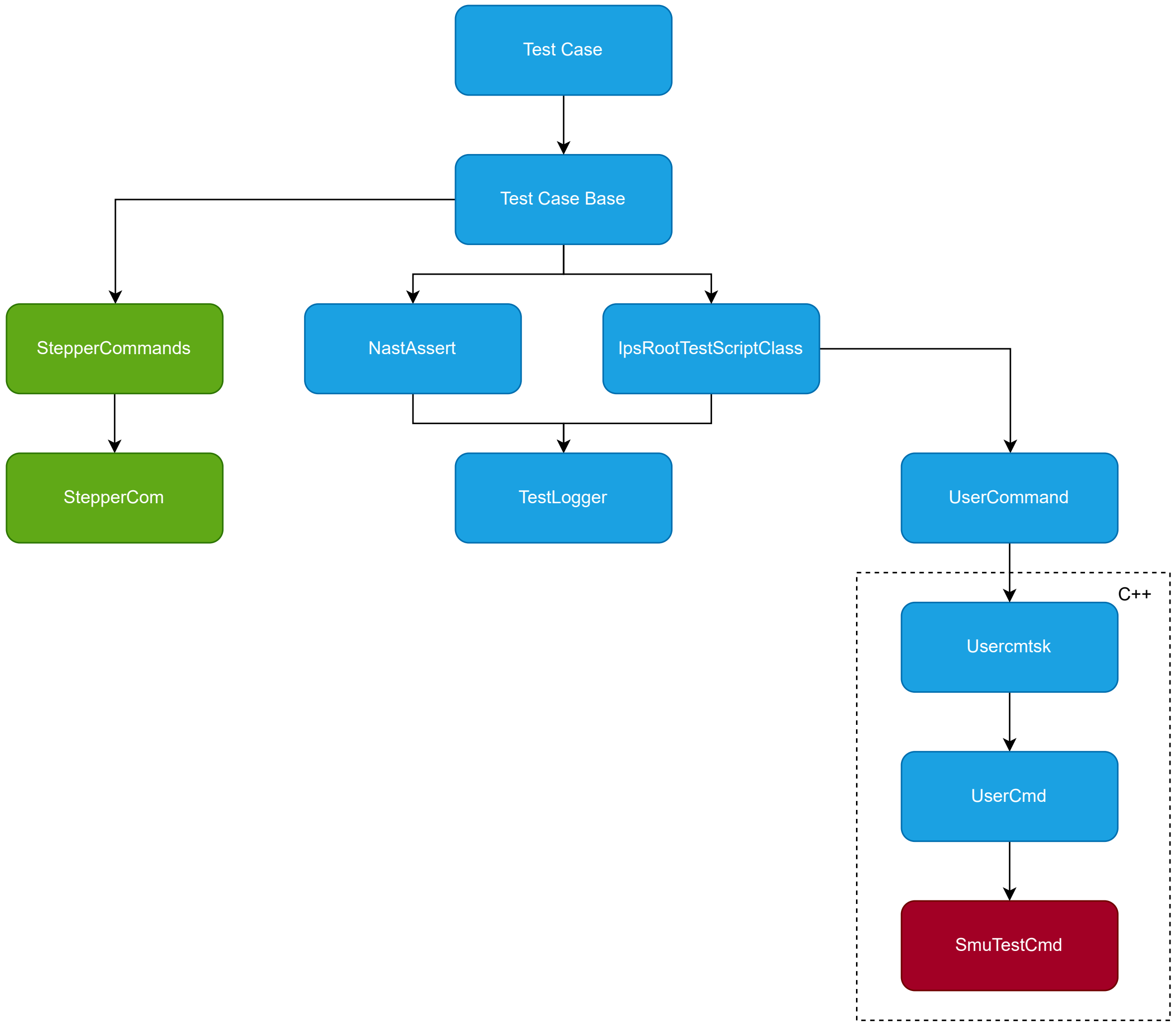


Figure 4.7: This figure shows the complete module dependency diagram overview. The section within the dotted line is the addition of SMU test commands to the test framework. Blue modules represent existing work, and green/red represent addons for the SMU test system.

SMU test commands table

A full overview of the implemented commands is shown in table 4.2

SMU test commands	Argument	Operation
7axis		Enables resolver number 7
run		Starts ring communication
stop		Stops ring communication
revcnt	<node>	Reads revolution counter at <node>
readnullvolt		Reads null voltage
test ramp		Reads resolver X and Y value at node 7
test ext		Returns resolver 1 and 4 squaresum value
clear revcnt	<node>	Clear revolution counter at <node>
readres	<node>	Read resolver angle at <node>
set eps		Set excitation phase shift mode
set eip		Set excitation in phase mode
writereg		Write to PIB register
readreg		Read PIB register
comchk		Checks communication status to SMU
drift	<samplesize>	Returns table of drift calculations for <samplesize> measurements. This table includes [Angle, MinAngle, MaxAngle, X, MinX, MaxX, Y, MinY, MaxY, DeltaX, DelatY, CurrentSqrXY, MinSqrXY, MaxSqrXY, DelataSqrXY]
reslog angle	<node> <expangle>	Returns log of angle measurements at <node> with <expangle>. <expangle> is needed for a spesific dataformat
reslog xy	<node> <expangle>	Returns log of X and Y values at <node> with <expangle>. <expangle> is needed for a spesific dataformat
reslog xysquare	<node> <expangle>	Returns log of square sum of X and Y at <node> with <expangle>. <expangle> is needed for a specific data format

Table 4.2: SMU test command table.

It's worth noting that the prefix for executing SMU test commands is "sms," which originates from the old name for the SMU, SMS. Thus, an example of a fully constructed SMU test command looks like this: "sms readres 2".

Chapter 5

Regression Test Cases

This chapter presents the implementation of the regression test cases for the test system. Initialization is done before every test case to ensure stability and consistency. The SMU test suite consists of the following regression test cases:

- **Excitation and ramp test case**
- **Resolver angle test case**
- **Resolver quadrant test case**
- **Resolver revolution test case**
- **Resolver battery test case**

5.1 Initialization

It's essential to start up the test system in the proper state. This section describes the start-up sequence for the test cases. This sequence is referred to as initialize and is performed before every test case to ensure stability and consistency. The sequence diagram in Figure 5.1 displays the start-up sequence for the test cases.

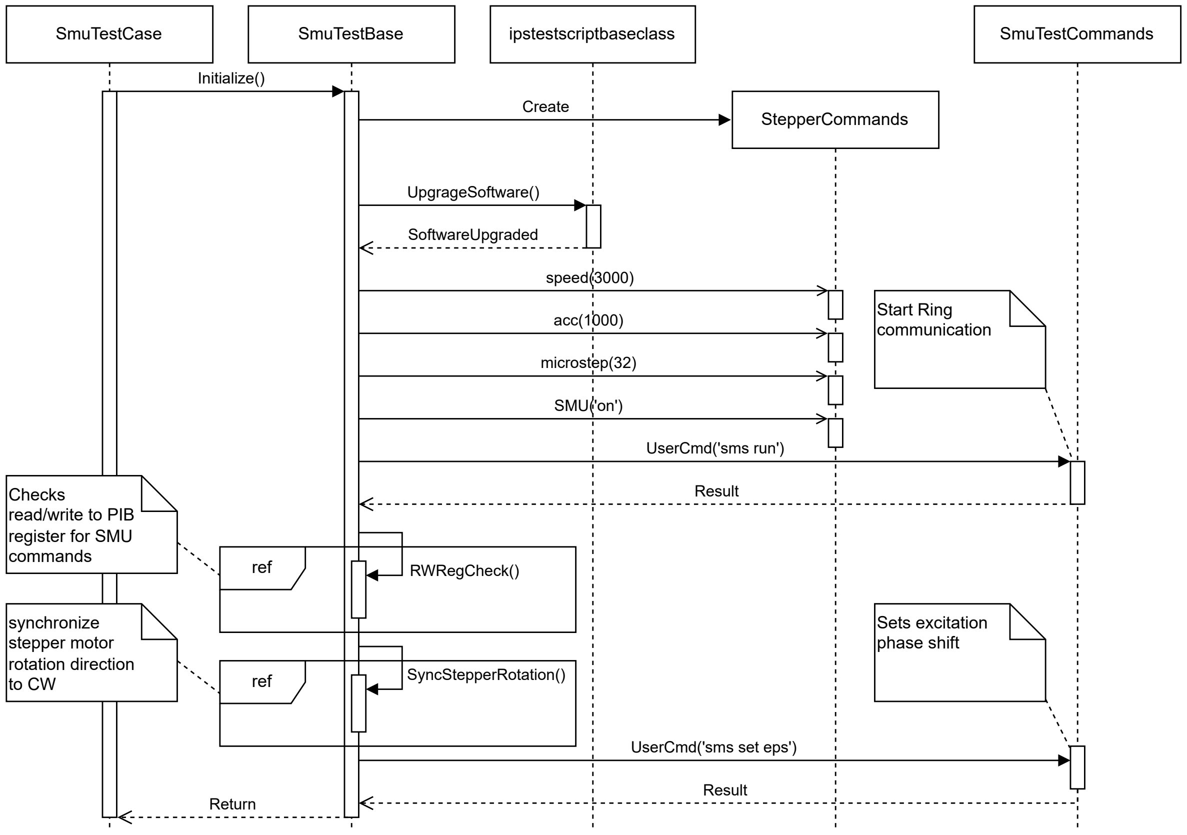


Figure 5.1: Sequence diagram for the start-up sequence.

The first step in the sequence presented in Figure 5.1 is to create an object of the Stepper-Command class from the StepperCommand module. Two constructor parameters are needed when creating this object, slave address and comport name. Initially, the test agent downloads and extracts the SUT's latest successful build in a specific directory. UpgradeSoftware() compares the version of this build with the current software version on target and decides if action is needed based on that. The two methods within the "ref" frames contain a test sequence described by the linked comment notes. Including the actual sequence of these small tests would clutter the main sequence, which is why it's compressed. Excitation Phase Shift is the preferred default modus, which is why it's included in the start-up sequence.

5.2 Excitation and Ramp Test Case

This test case consists of the following tests:

- **Ramp test**
- **Excitation test**

Ramp test
Two checks have to be made to test if the ramp function works. The first check is to see if "sms ramp pos" brings the X and Y value for resolver 7 to the maximum value of 8191. The second test is to see if "sms ramp neg" brings the value down to the minimum value of -8192. Figure 5.2 illustrates the sequence for the ramp test.

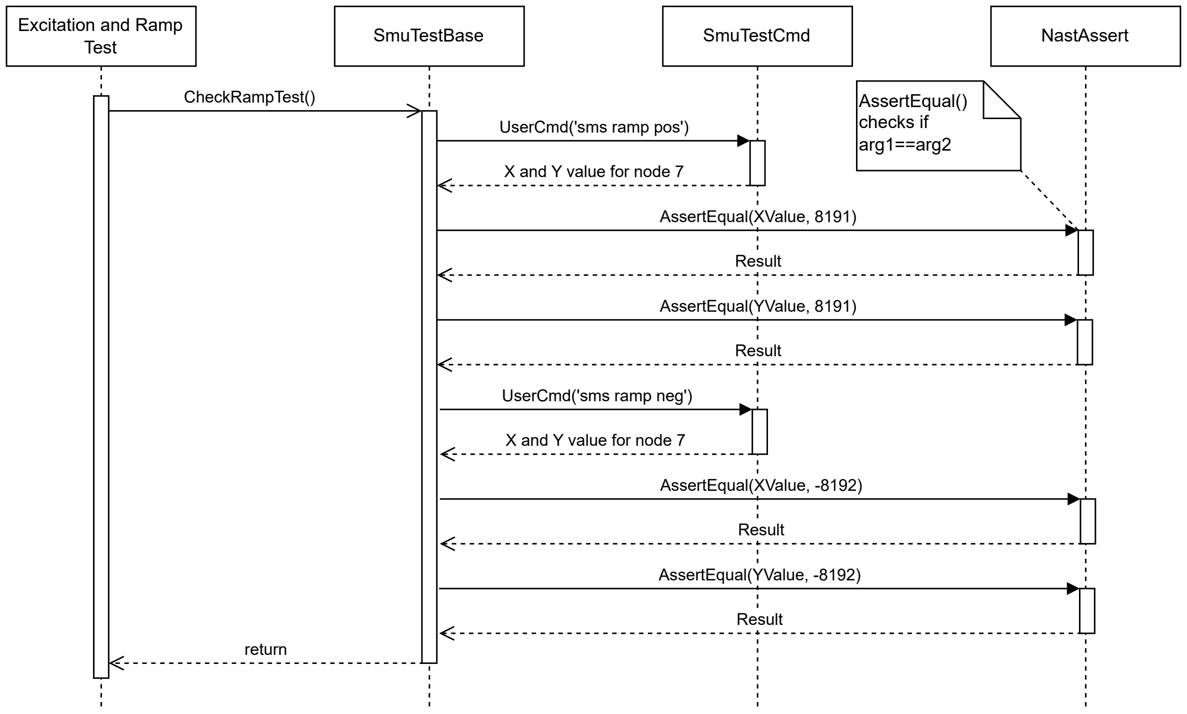


Figure 5.2: Sequence diagram for the ramp test.

Excitation test

After the ramp test is done, the excitation test starts. Initially, the SMU should be in EPS modus from the initialization sequence presented in section 5.1. Thus, the resolver measurement sequence is structured as illustrated in Figure 2.4 in section 2.2. The square summation of the raw measurement values X and Y should ideally always be the same for all angles. Moreover, in this case, that would be between 6500 to 7800, depending on which resolver is measured. When switching to EIP without restructuring the resolver measurement sequence, the resolvers in excitation group 1 will measure zero crossings as illustrated in Figure 5.3.

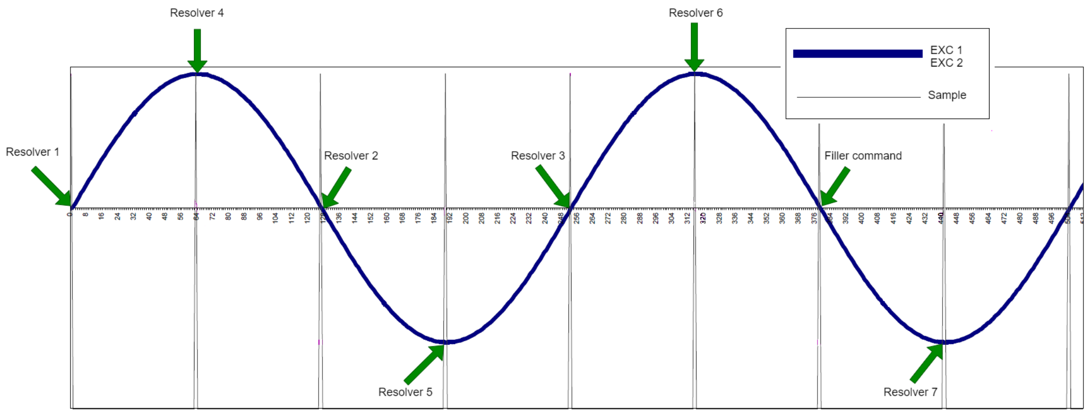


Figure 5.3: Excitation EIP test measurement sequence, excitation signal 1 and 2 are in phase. Both excitation signal 1 and 2 are labeled as blue. All the resolvers in group 1 are measuring zero crossings instead of tops. Altered reprint from [2] with permission from ABB

As shown in Figure 5.3, resolver group 2 should have a high-value square summation of X and Y values, and all the resolvers in group 1 should have a relatively low value due to measuring zero crossings.

The sequence diagram in Figure 5.4 presents the program flow of the excitation test.

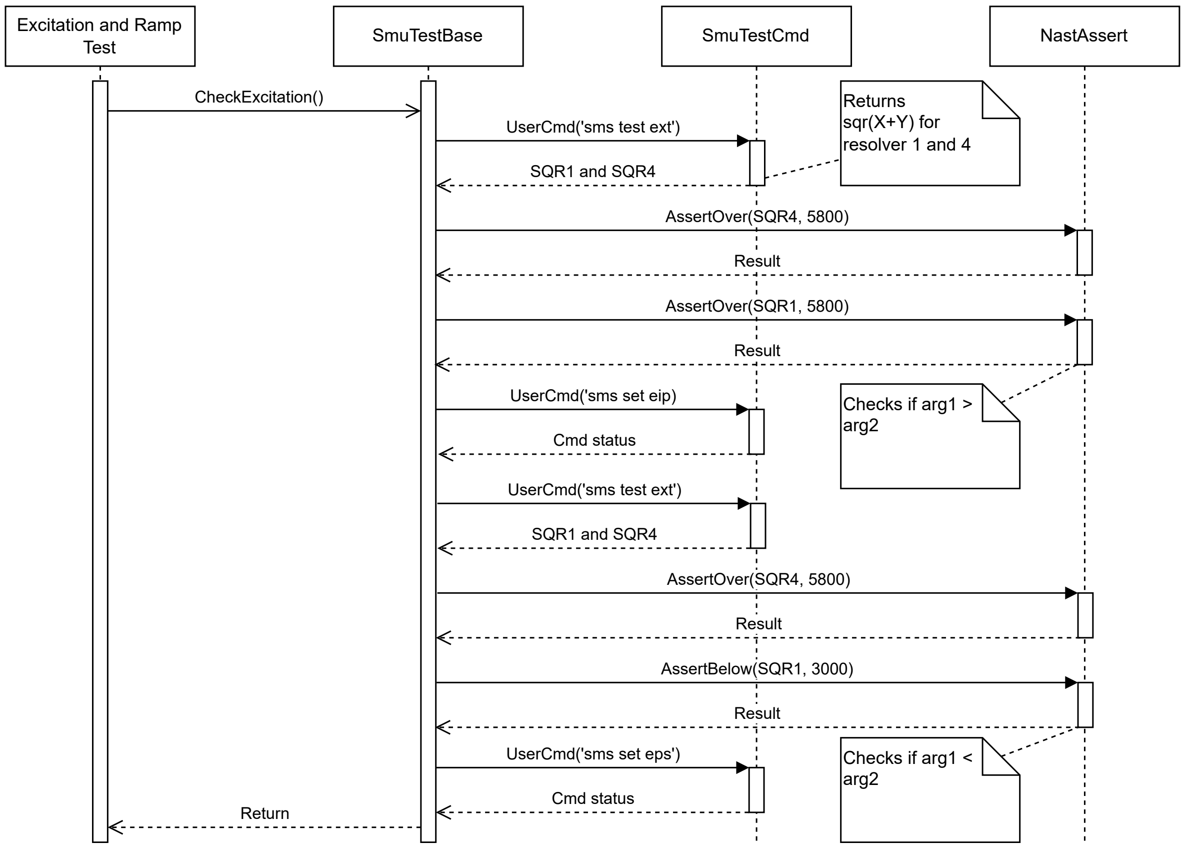


Figure 5.4: Sequence diagram for excitation test.

At the start of this test, resolver 4 in excitation group 2 and resolver 1 in excitation group 1 should have a high square summation value due to initially being in the EPS modus. The first check is to ensure that EPS works before testing EIP. When checking if the EIP modus works, all the resolvers in excitation group 1 should have a square summation value close to zero, in theory. However, in practice, resolver group 1 has square summation values ranging from 1500 to 2500 in the EIP modus. This could indicate the synchronization between the excitation signal and sampling may be slightly off. Consequently, the boundary for an approved measurement of any resover in group 1 should be below 3000, as shown in Figure 5.4.

5.3 Resolver Angle Test Case

This test case revolves around testing if resolver measurements are within the expected boundaries. ABB did not specify tolerances for an approved measurement. Thus, the boundaries have been set to be $\pm 1^\circ$ of the expected angle. There are three types of suitable angle measurement tests:

1. **Angle measurements in increasing increments:**
   This test checks if angle measurements for small increasing angle increments are within expected boundaries.
2. **Angle measurements in decreasing increments:**
   This test checks if angle measurements for small decreasing angle increments are within expected boundaries.
3. **Random angle measurements:**
   This test aims to simulate the practical usage of the robots. It's practically unknown which angles the SMU has to measure from a developer's perspective. Thus, it's crucial to test whether the SMU consistently measures random angles correctly.

Angle measurements in increasing increments

The program flow of this test is illustrated in Figure 5.5.

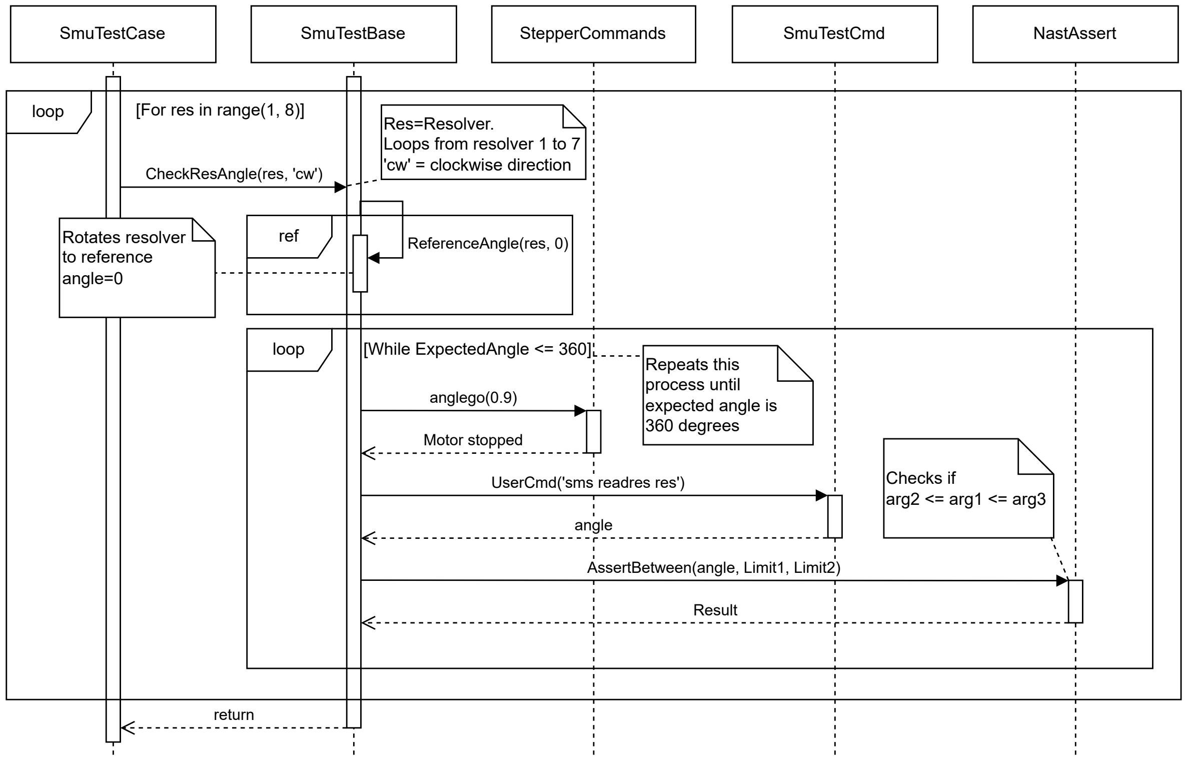

SmuTestCase
SmuTestBase
StepperCommands
SmuTestCmd
NastAssert
loop
[For res in range(1, 8)]
Res=Resolver.
Loops from resolver 1 to 7
'cw' = clockwise direction
CheckResAngle(res, 'cw')
Rotates resolver to reference angle=0
ref
ReferenceAngle(res, 0)
loop
[While ExpectedAngle <= 360]
Repeats this process until expected angle is 360 degrees
anglego(0.9)
Motor stopped
UserCmd('sms readres res')
angle
Checks if arg2 <= arg1 <= arg3
AssertBetween(angle, Limit1, Limit2)
Result
return

Figure 5.5: Sequence diagram angle measurement in increasing increments test.

This test loops through all seven resolvers, where each resolver goes from 0 to 360 degrees in 0.9 degrees step increments. The reason for using 0.9 degrees step intervals is to avoid rounding errors when converting degrees to discrete steps. The "limit2" and "limit1" in "AssertBetween" function from Figure 5.5 are based on the expected angle ± 1 degrees, respectively.

Angle measurements in decreasing increments

The test follows a similar program flow to the previous test depicted in Figure 5.5, but in the CCW direction. It also starts at a reference angle of 0 degrees, which essentially corresponds to 360 degrees.

Random angle measurements

For the last test, Random angle measurements, the program flow slightly changes relatively from the first two tests as shown in Figure 5.6

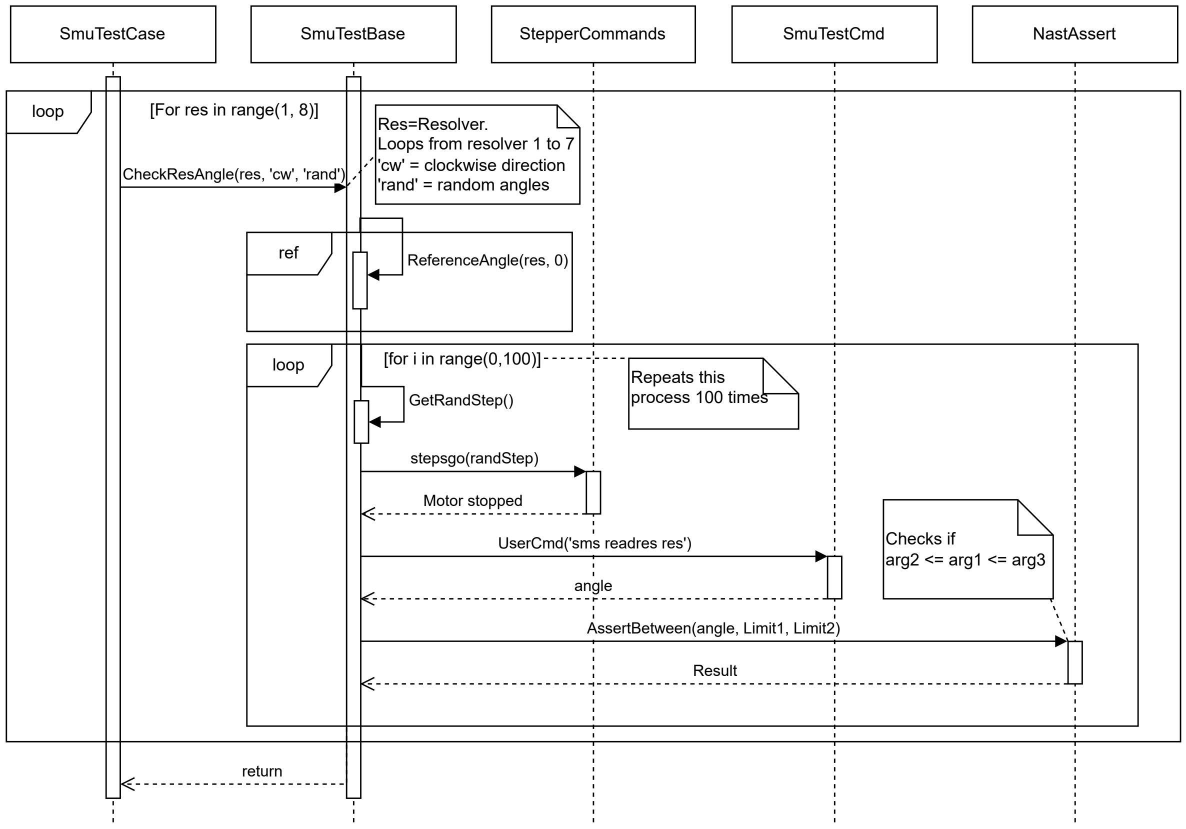


Figure 5.6: Sequence diagram random angles measurements test.

The limits for the AssertBetween methods in Figure 5.6 are based on the ideal accumulated angle from "randStep" ±1 degree, in other words, expected angle ±1 degree.

5.4 Resolver Quadrant Test Case

This test aims to check if the transition between quadrants is stable. When new software changes related to the resolver measurements are introduced in the FPGA, it's crucial to verify that the values are stored in the proper format, as explained in section 2.2.

The strategy for testing the stability of quadrant transitions revolves around doing tiny steps over the transition points. At every step, multiple measurements are done. The noise on the measurements will cause overlaps on each step. Thus, greater coverage is achieved, which increases the probability of uncovering instability in quadrant transitions. Figure 5.7 illustrates the strategy of measurements for this test.

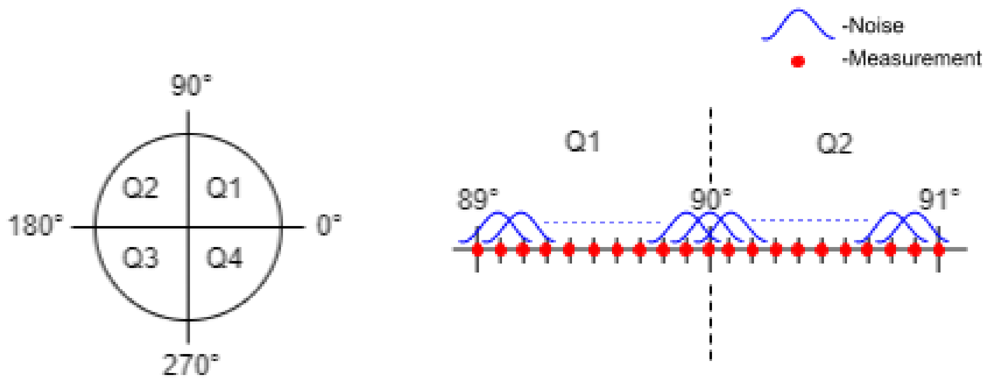


Figure 5.7: This figure shows how measurements are done to test quadrant transition stability. The red dots are the measurements done, and the blue bell curves represent noise for each stepping position.

From Figure 5.7, the noise on each step is assumed to be normally distributed, and in practice, the noise is more extensive than displayed in Figure 5.7. The program flow of this test is illustrated in Figure 5.8.

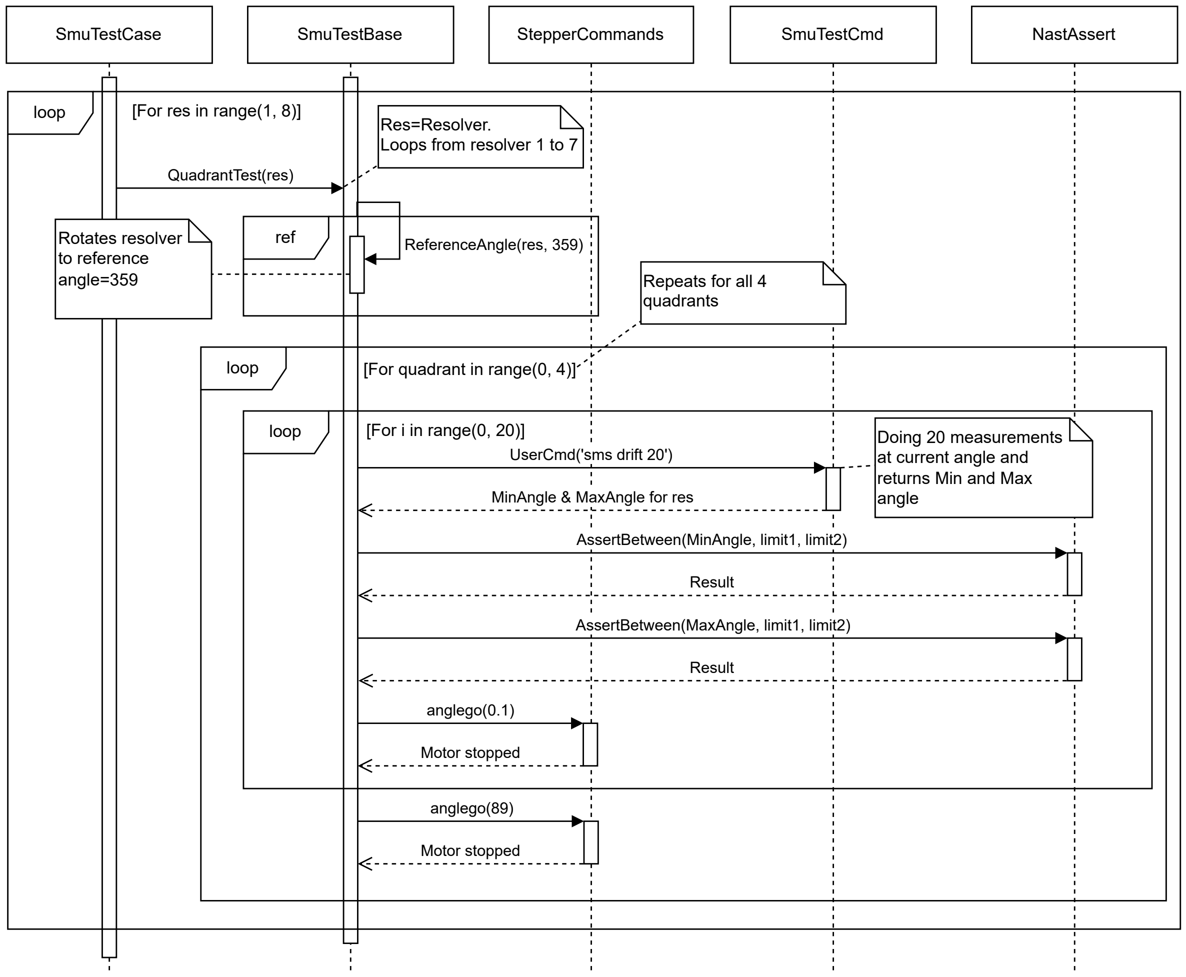

SmuTestCase
SmuTestBase
StepperCommands
SmuTestCmd
NastAssert
loop
[For res in range(1, 8)]
Res=Resolver. Loops from resolver 1 to 7
QuadrantTest(res)
Rotates resolver to reference angle=359
ref
ReferenceAngle(res, 359)
Repeats for all 4 quadrants
loop
[For quadrant in range(0, 4)]
loop
[For i in range(0, 20)]
UserCmd('sms drift 20')
Doing 20 measurements at current angle and returns Min and Max angle
MinAngle & MaxAngle for res
AssertBetween(MinAngle, limit1, limit2)
Result
AssertBetween(MaxAngle, limit1, limit2)
Result
anglego(0.1)
Motor stopped
anglego(89)
Motor stopped

Figure 5.8: This figure shows the sequence diagram of the quadrant test program flow. The frame labeled "ref" is a method to rotate the resolver to a desired position precisely.

As shown in Figure 5.8, the starting position is 359 degrees. This means the first quadrant transition happens from quadrant 4 to quadrant 1. The stepper motor does 20 0.1 degrees steps with 20 measurements on each step. In this case, all of the measurements done from 359 to 1 degree have to be within [357,360] and [0, 2] to be classified as a successful test, respectively.

5.5 Resolver Revolution Test Case

This test case checks if the SMU is able to properly count revolutions. This test case consists of the following tests:

1. **Increasing revolution counter:**
   This test checks if the SMU is able to count revolutions in an increasing manner correctly.
2. **Decreasing revolution counter:**
   This test checks if the SMU is able to count revolutions in a decreasing manner correctly.
3. **Random revolution counter:**
   This test checks if the SMU is able to correctly count random revolutions.
4. **Revolution reset function:**
   This test checks if the SMU is able to reset the revolution counters.

Increasing revolution counter
This test simply just performs revolutions in CW direction and checks if the SMU is able to properly update the revolution counter. The program flow for this test is illustrated in Figure 5.9.

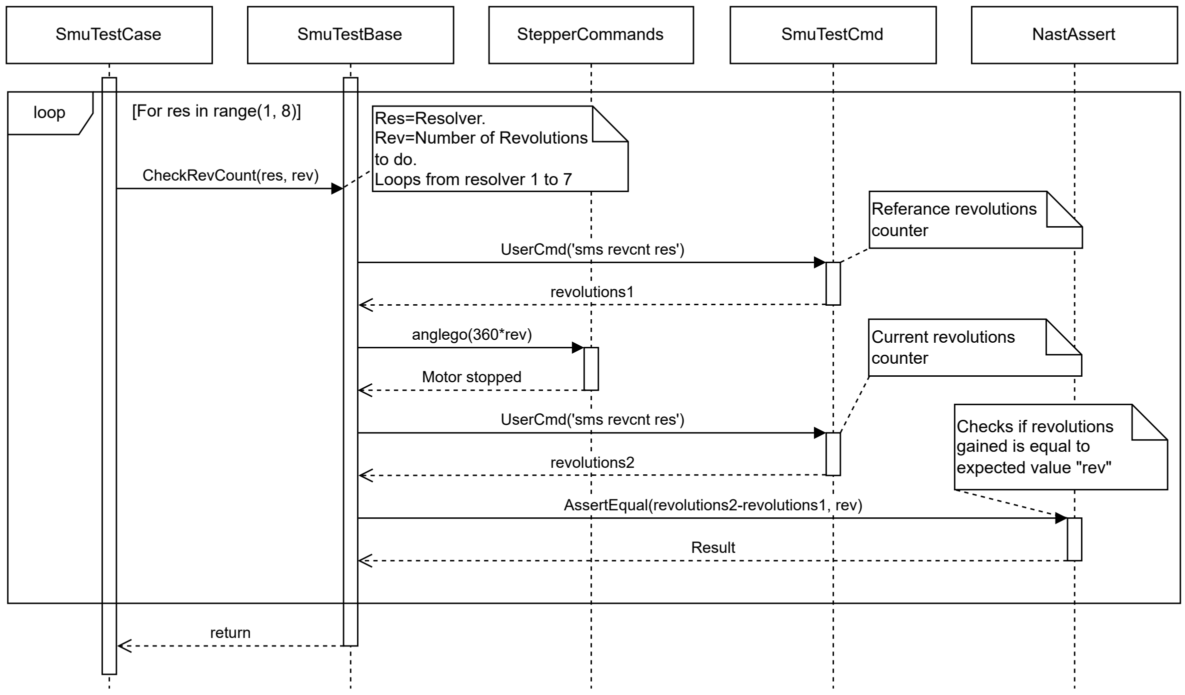

SmuTestCase
SmuTestBase
StepperCommands
SmuTestCmd
NastAssert
loop
[For res in range(1, 8)]
Res=Resolver.
Rev=Number of Revolutions to do.
Loops from resolver 1 to 7
CheckRevCount(res, rev)
Referance revolutions counter
UserCmd('sms revcnt res')
revolutions1
anglego(360*rev)
Current revolutions counter
Motor stopped
UserCmd('sms revcnt res')
Checks if revolutions gained is equal to expected value "rev"
revolutions2
AssertEqual(revolutions2-revolutions1, rev)
Result
return

Figure 5.9: This figure displays the program flow for the incremental revolution counter test. This test is performed on all resolvers as shown with the first for-loop.

As shown in Figure 5.9, the method of checking if the SMU can update the revolution counter correctly involves comparing the delta revolution count with the expected revolution count.

Decreasing revolution counter
This test simply just performs revolutions in CCW direction and checks if the SMU is able to properly update the revolution counter. The program flow for this test is the same as the increasing revolution test.

Random revolution counter
This test aims to simulate a practical scenario of the revolution counter by randomizing the amount of revolution the stepper motor rotates. The sequence of change in the revolution counter is random in practice, which is why it's essential to perform this test. The only difference in program flow relative to the incremental revolution counter test presented in Figure 5.9 is that the input variable "rev" is randomized between $[-20, 20]$. This test is performed 10 times for each resolver.

5.6 Resolver Battery Modus Test Case

The purpose of this test case is to verify if the SMU is able to update the revolution counter in battery modus. This test case consists of the following test:

1. **Increasing revolution counter in battery modus**
2. **Decreasing revolution counter in battery modus**
3. **Random revolution counter in battery modus**

Increasing revolution counter in battery modus
This test performs 100 revolutions in a CW direction. The program flow for this test is illustrated in Figure 5.10.

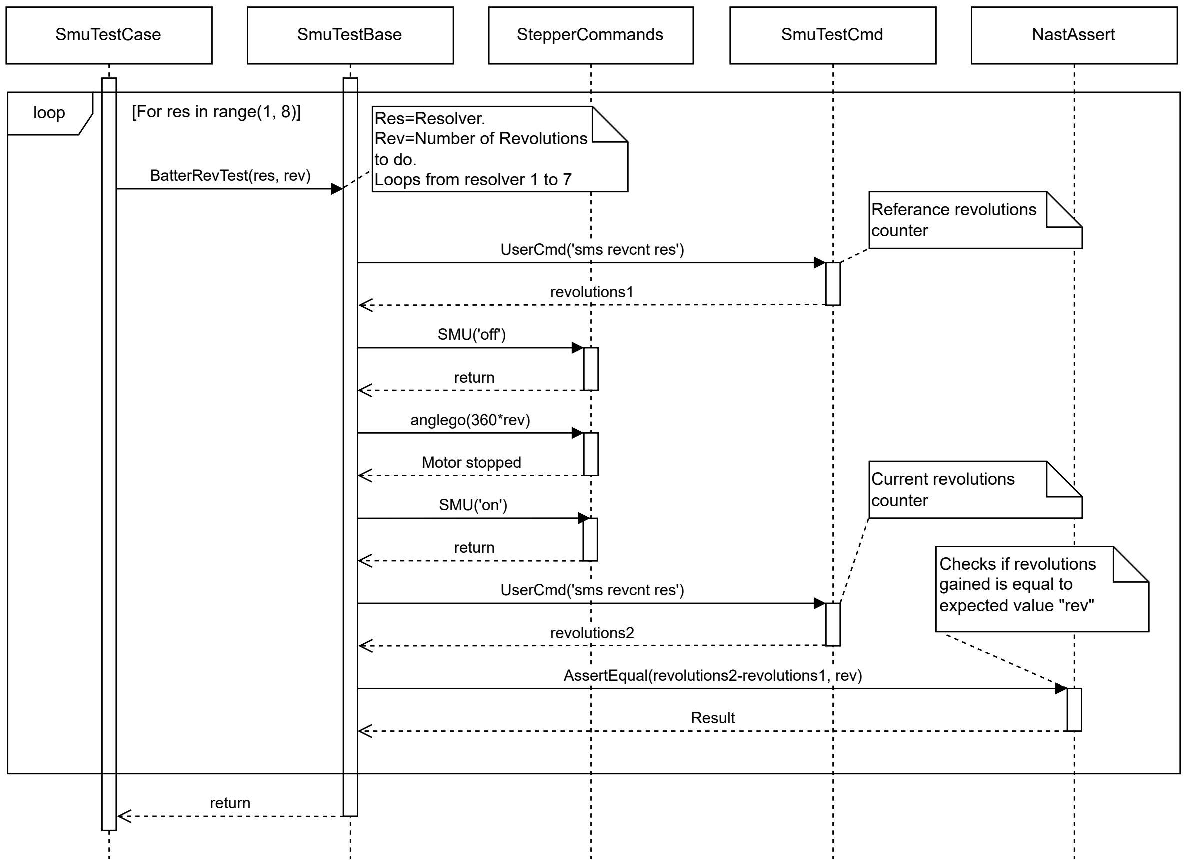

SmuTestCase
SmuTestBase
StepperCommands
SmuTestCmd
NastAssert
loop
[For res in range(1, 8)]
Res=Resolver.
Rev=Number of Revolutions to do.
Loops from resolver 1 to 7
BatterRevTest(res, rev)
Referance revolutions counter
UserCmd('sms revcnt res')
revolutions1
SMU('off')
return
anglego(360*rev)
Motor stopped
SMU('on')
return
Current revolutions counter
Checks if revolutions gained is equal to expected value "rev"
UserCmd('sms revcnt res')
revolutions2
AssertEqual(revolutions2-revolutions1, rev)
Result
return

Figure 5.10: Sequence diagram for revolution counter in battery modus.

The only difference between this test and the increasing revolution counter test is that the main power of the SMU is switched off before the stepper motor does "rev" amount of revolutions, as displayed in Figure 5.10. In addition to this, after the SMU("off") command is executed, two checks are made to verify that the SMU has properly entered battery modus.

Decreasing revolution counter in battery modus
This test does 100 revolutions in CCW direction. The program flow for this test is the same as 5.10, just in the opposite direction.

Random revolution counter in battery modus
This test does the same as the random revolution counter test, just in battery modus.

Chapter 6

Quality measurement test Case

This chapter introduces a method of choosing the optimal microstep configuration and stepping interval. Additionally, this chapter presents the implementation of the quality measurement test case.

6.1 Choosing Optimal Step Length

As mentioned in section 2.6, the SMSD stepper motor controller has seven different microstepping configurations, ranging from fullstep to a microstepping factor of 256. Figure 6.1 displays a small stopping accuracy test of 0.9-degree step intervals between four microstep configurations, where stopping accuracy is defined as the deviation between the expected and actual stopping positions.

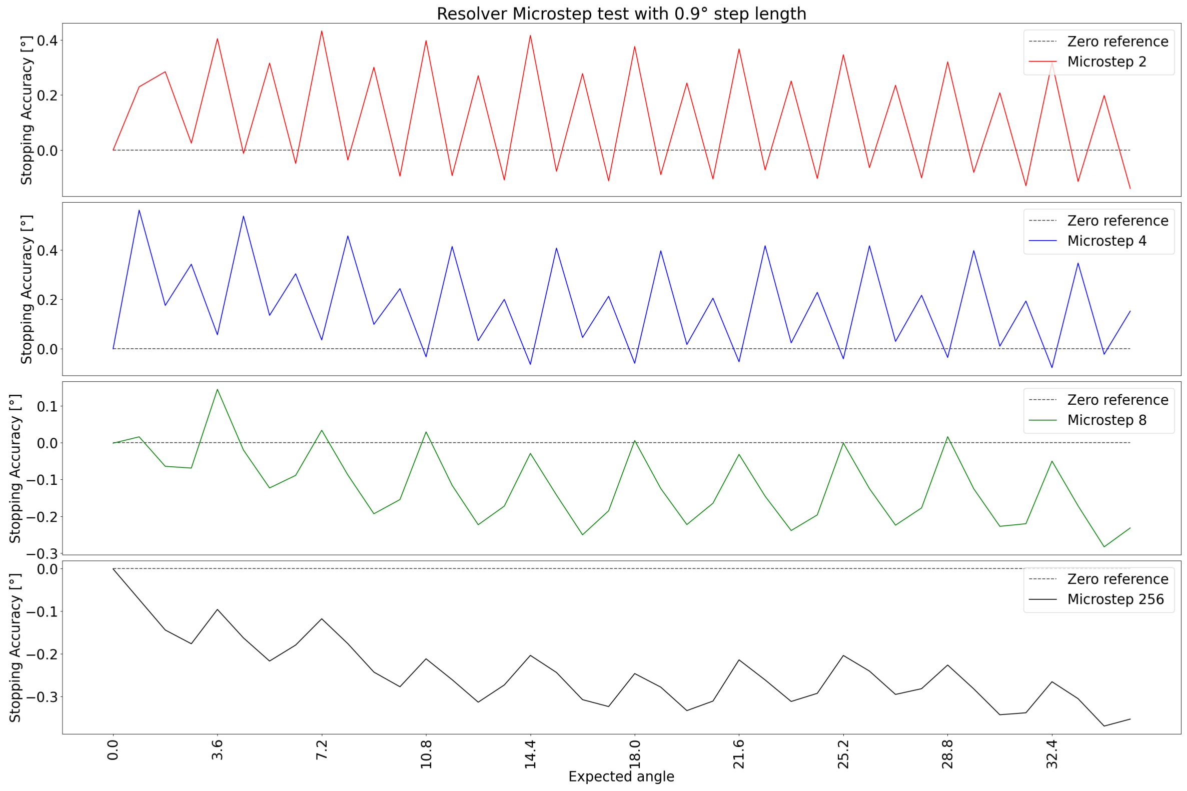


Figure 6.1: This figure shows an angle displacement test with four different micro-stepping configurations.

As shown in Figure 6.1, all the microstep configurations have relatively similar patterns, where the main differences are the trend and magnitude of oscillation. The pattern oscillates with 3.6-degree period throughout a complete revolution for all microstepping configurations.

The pattern observed in Figure 6.1 is most likely a combination of mechanical and electrical factors, as suggested in [17]. An option to achieve better stopping accuracy is to tune the stepper motor to have slow decay on the output current, as suggested in [17]. However, the SMSD stepper motor controller does not have the option to tune the current decay modus.

Another way of eliminating the pattern is to increase the step intervals to match the period of the pattern, in this case, 3.6-degrees. Figure 6.2 illustrates the result of increasing the step size to 3.6 degrees for the microstep 8 test run.

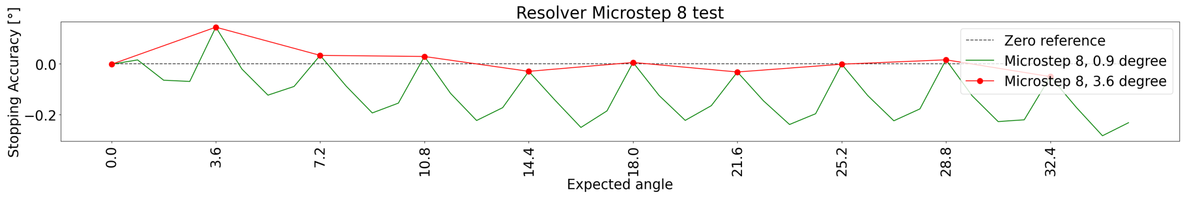


Figure 6.2: This figure shows the effect of increasing the step interval to 3.6-degrees.

To further analyze the effect of increasing the step size to 3.6-degrees, a test for a complete revolution was conducted for all microstep configurations. Figure 6.3 shows the results of this test.

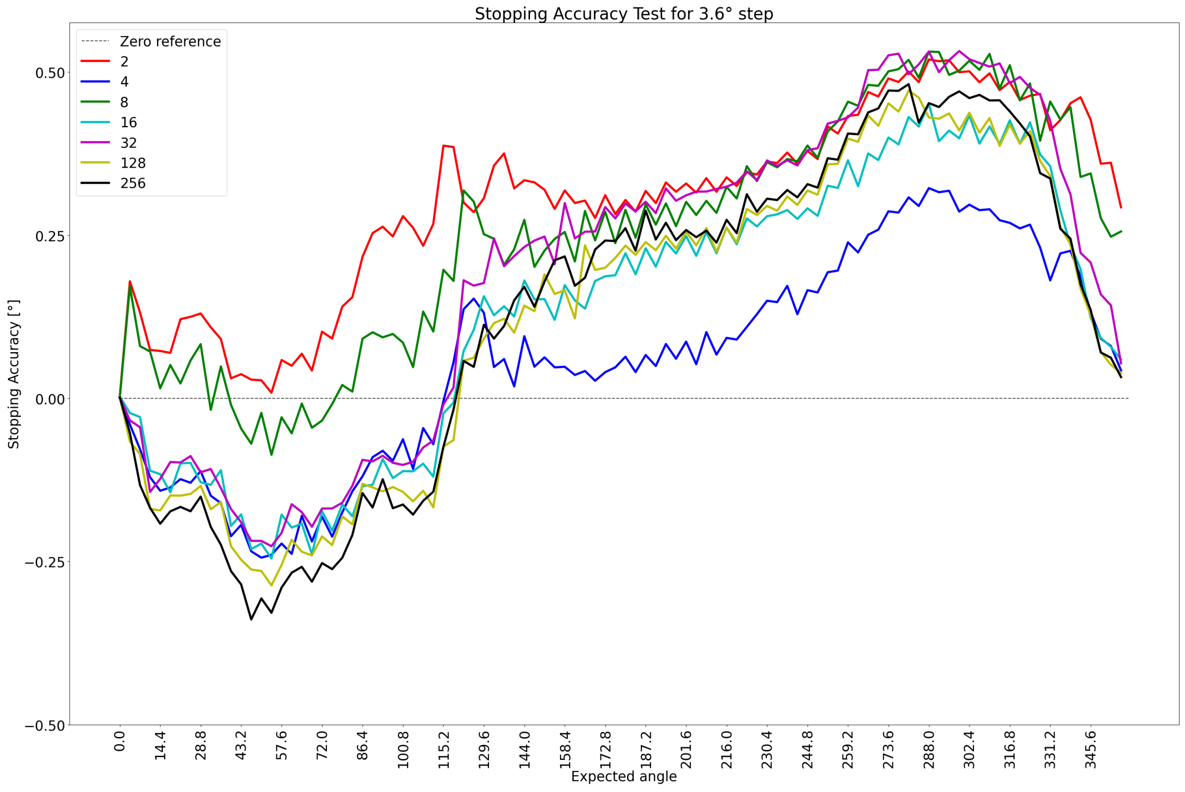


Figure 6.3: Stopping Accuracy test for all microstepping configurations with a step length of 3.6-degrees.

As shown in Figure 6.3, there is a clear difference in the trend for each microstep configuration. By calculating the absolute error with equation 6.1, microstepping with a factor of 4 is the optimal configuration and will be used as the default setting.

$$
\begin{aligned}
&e_a = |X - X^{'}| = |Error| \qquad (6.1)\\
&X \text{ - Measured value}\\
&X^{'} \text{ - Expected value}
\end{aligned}
$$

6.2 Data Collection

The FPGA on the SMU has a dedicated buffer for logging. If one of the logging commands in the SmuTestCmd module is executed, the FPGA starts to store the raw values of X and Y in this buffer whenever it measures a resolver. The command will then start to read the values from this buffer. In short, the logging command is designed to gather 1300 measurements on each step interval and then calculate the angle and square summation of X and Y before printing relevant values in a specific data format for the test case.

The three data values of interest are:

1. **Angle**
2. **Square summation of X and Y**
3. **Raw values X and Y**

Angle:
This data will show how accurate and precise the angle measurements for the SMU are. The method of analyzing this measurement is subtracting the expected value from the measured values at every step interval, hereafter referred to as deviation. The plots generated regarding these measurements will display the average deviation and deviation spread.

Square summation of X and Y:
This data will uncover the trend and magnitude of inconsistency between X and Y values. Ideally, the resolver values X and Y should follow the mathematical relationship defined by the trigonometric identity. This implies that the square summation should remain consistent throughout an entire resolver revolution. Creating a polar plot of the gathered data allows one to visualize the average, minimum, and maximum square summation of X and Y values.

Raw values X and Y
The last measurement for analysis is the noise on the raw X and Y values. Noise is defined as the variation around the average value in this case.

The program flow of data collection is illustrated in the sequence diagram in Figure 6.4.

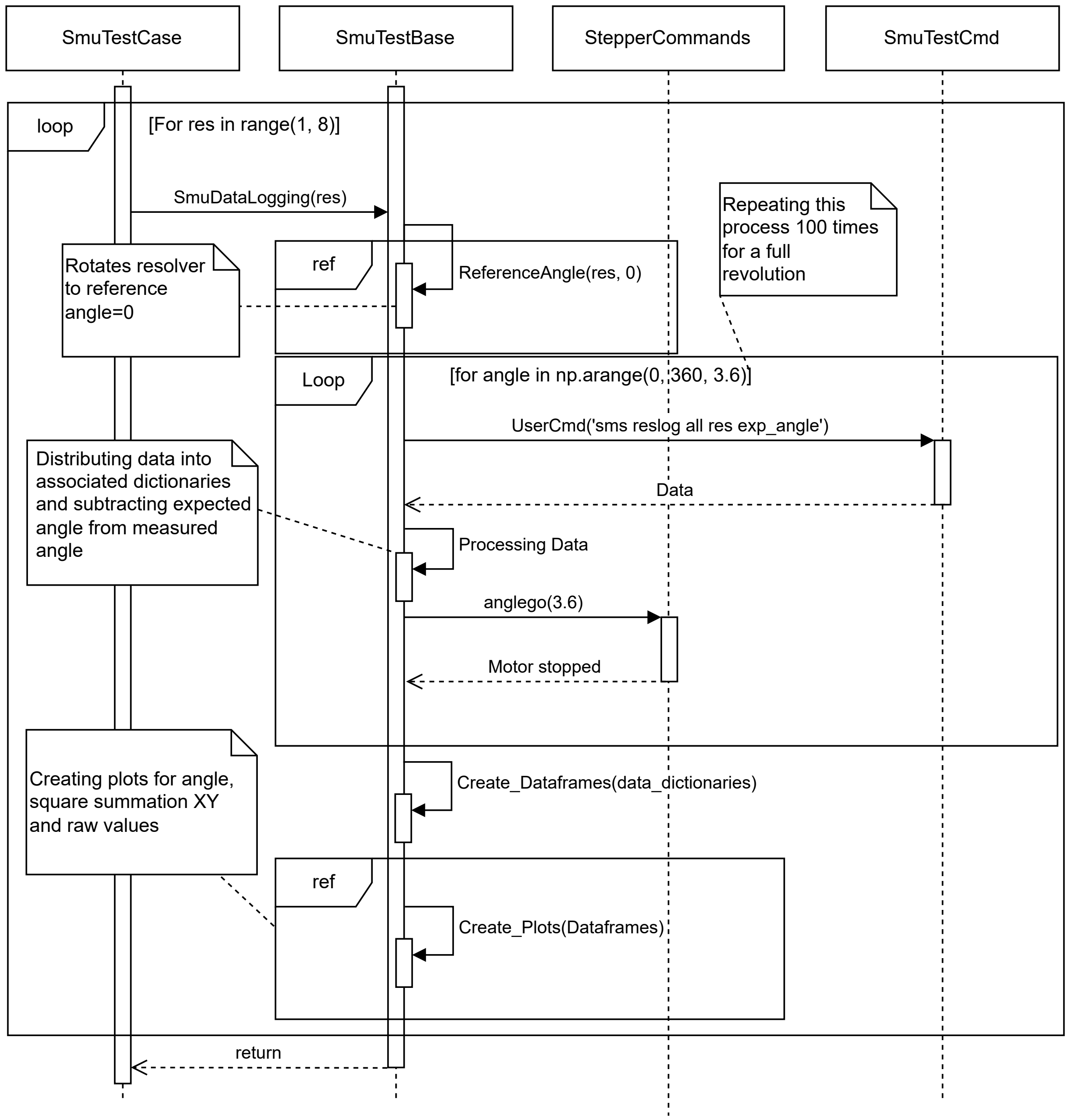

SmuTestCase
SmuTestBase
StepperCommands
SmuTestCmd
loop
[For res in range(1, 8)]
SmuDataLogging(res)
Repeating this process 100 times for a full revolution
Rotates resolver to reference angle=0
ref
ReferenceAngle(res, 0)
Loop
[for angle in np.arange(0, 360, 3.6)]
UserCmd('sms reslog all res exp_angle')
Distributing data into associated dictionaries and subtracting expected angle from measured angle
Data
Processing Data
anglego(3.6)
Motor stopped
Creating plots for angle, square summation XY and raw values
Create_Dataframes(data_dictionaries)
ref
Create_Plots(Dataframes)
return

Figure 6.4: Sequence diagram for data collection.

Chapter 7

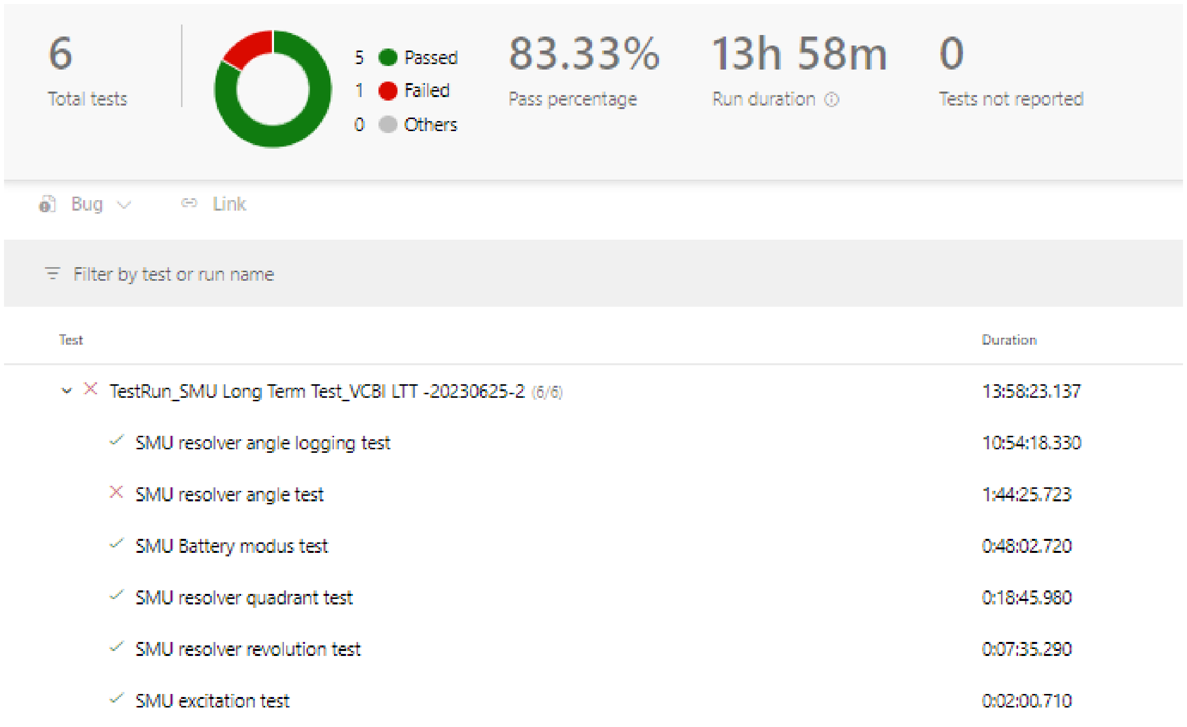
Results

This chapter introduces the results of the regression and quality measurement test cases. It's observed that the angle measurement test case is the only failing regression test, while the quality measurement test results exhibit a non-linear trend for all resolver measurements. This chapter consists of the following sections:

- **Regression Test Tesults:**
  - Azure test suite results
  - Table of failing measurements for the resolver angle measurement test case
- **Quality Angle Measurement Results:**
  - Quality angle measurement plots for resolver 2, 3, 4 and 6
  - Polar plots of the square summation of X and Y for resolver 2, 3, 4 and 6

7.1 Regression Test Results

The test results for the regression test case are presented in two different ways, a graphical summary generated on Azure and relevant sections to the respective log files generated by the test framework. A full overview of a single test run is presented in Figure 7.1.



6 Total tests
5 ● Passed
1 ● Failed
0 ● Others
83.33% Pass percentage
13h 58m Run duration
0 Tests not reported

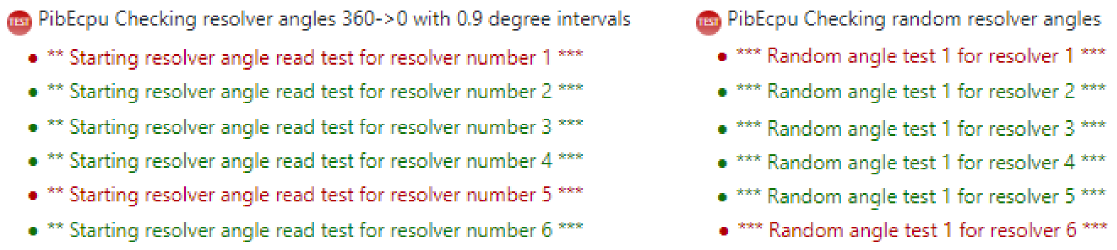
Bug Link

Filter by test or run name

Test	Duration
✗ TestRun_SMU Long Term Test_VCBI LTT -20230625-2 (6/6)	13:58:23.137
✓ SMU resolver angle logging test	10:54:18.330
✗ SMU resolver angle test	1:44:25.723
✓ SMU Battery modus test	0:48:02.720
✓ SMU resolver quadrant test	0:18:45.980
✓ SMU resolver revolution test	0:07:35.290
✓ SMU excitation test	0:02:00.710

Figure 7.1: Test results for a full test run generated on Azure.

The "SMU resolver angle logging test case" is the quality measurement test case and is not a part of this section. As shown in Figure 7.1, the SMU only fails on one regression test case, the resolver angle test case. The failing resolvers and tests within this test case are presented in Figure 7.2.



```
PibEcpu Checking resolver angles 360->0 with 0.9 degree intervals
• ** Starting resolver angle read test for resolver number 1 ***
• ** Starting resolver angle read test for resolver number 2 ***
• ** Starting resolver angle read test for resolver number 3 ***
• ** Starting resolver angle read test for resolver number 4 ***
• ** Starting resolver angle read test for resolver number 5 ***
• ** Starting resolver angle read test for resolver number 6 ***

PibEcpu Checking random resolver angles
• *** Random angle test 1 for resolver 1 ***
• *** Random angle test 1 for resolver 2 ***
• *** Random angle test 1 for resolver 3 ***
• *** Random angle test 1 for resolver 4 ***
• *** Random angle test 1 for resolver 5 ***
• *** Random angle test 1 for resolver 6 ***
```

Figure 7.2: Test results for a full test run generated on Azure

The logs generated regarding this test case are relatively big. Therefore, the failing test sections are summarized with a table instead. Table 7.1 presents the failing sections of the decreasing resolver angle measurement tests for resolvers 1 and 5.

Resolver 1		
Measured Angle[°]	Lower Limit[°]	Upper Limit[°]
69.413	67.4	69.4
65.846	63.8	65.8
62.224	60.2	62.2
58.634	56.6	58.6
55.040	53.0	55.0
51.458	49.4	51.4
47.836	45.8	47.8
44.259	42.2	44.2
40.641	38.6	40.6
37.029	35.0	37.0
33.414	31.4	33.4
29.831	27.8	29.8
26.267	24.2	26.2
24.402	22.4	24.4
22.695	20.6	22.6
19.076	17.0	19.0
17.201	15.2	17.2
15.493	13.4	15.4
11.852	9.8	11.8
8.233	6.2	8.2
Resolver 5		
Measured Angle[°]	Lower Limit[°]	Upper Limit[°]
254.825	252.8	254.8
253.912	251.9	253.9
252.101	250.1	252.1
251.238	249.2	251.2
250.319	250.319	250.3

Table 7.1: Failing sections for the decreasing angle measurement test.

As displayed in Table 7.1, there are 20 failing angle measurements for resolver 1 and 5 for resolver 5. This translates to a failing rate of 5% and 1.25%, respectively.

Table 7.2 shows the failing sections for the random angle measurement test for resolvers 1 and 6.

Resolver 1		
Measured Angle[°]	Lower Limit[°]	Upper Limit[°]
70.857	70.94375	72.94375
Resolver 6		
Measured Angle[°]	Lower Limit[°]	Upper Limit[°]
16.885	17.1125	19.1125
345.973	346.11875	348.11875
337.257	337.56875	339.56875
16.334	16.4375	18.4375
271.255	269.16875	271.16875
244.329	242.28125	244.28125
228.219	226.08125	228.08125
15.242	15.425	17.425

Table 7.2: Failing sections for the random angle measurement test.

As displayed in Table 7.2, resolver 1 fails on 1 random angle measurement, and resolver 6 fails on 8. This translates to a failing rate of 1% and 8%, respectively.

7.2 Quality Measurement Results

The results presented in this section are based on the average of 4 tests conducted on one SMU for the quality measurement test case. As the results were relatively comparable across all six resolvers, four of the results with the greatest dissimilarity are presented to avoid repetitiveness.

7.2.1 Resolver 2

Figure 7.3 presents the results for the angle measurements for resolver 2.

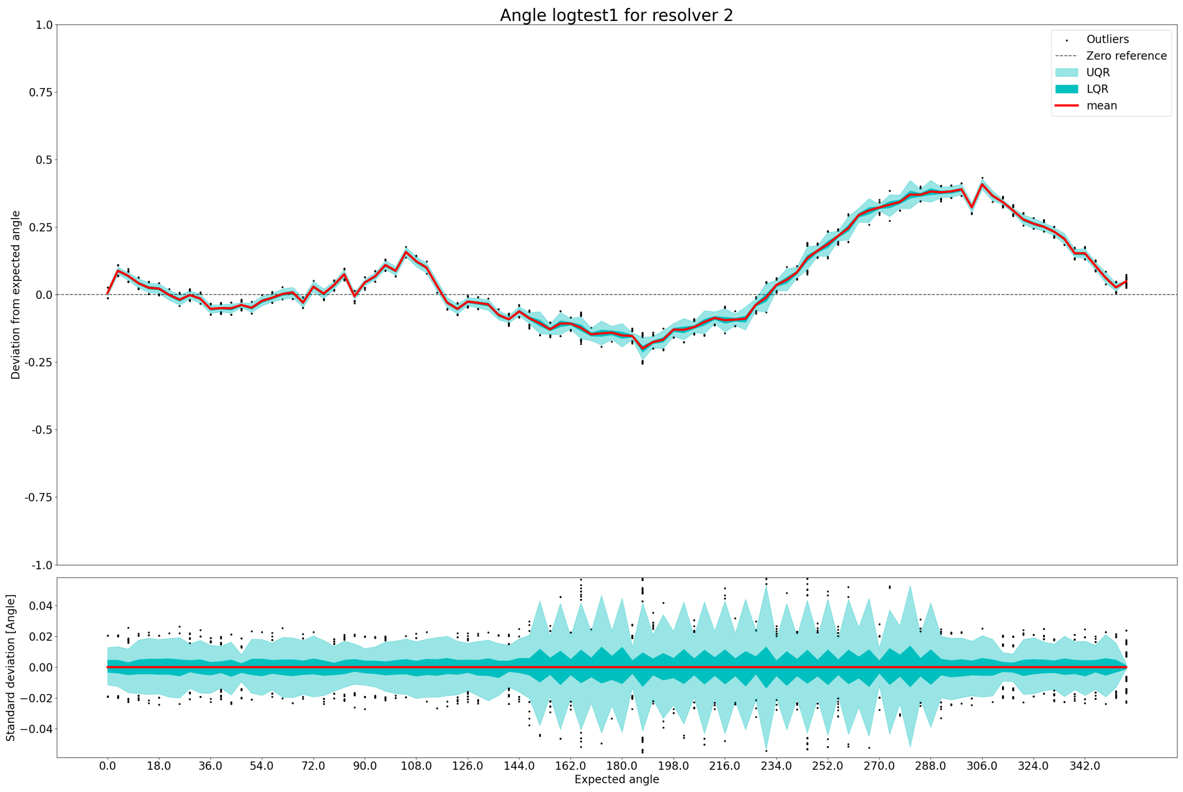


Figure 7.3: This figure shows the angle quality measurement results for resolver 2. The upper plot shows the difference between the measured and expected angles, and the lower plot shows the variance around the average measured angle. Lower Quartile Range(LQR) represents 50% of the measurements, and Upper Quartile Range(UQR) represents approximately 99.1% of the measurements. Outliers are the measurements that fall outside of LQR and UQR.

As shown in Figure 7.3, in the lower plot, the noise is relatively consistent throughout a complete revolution. However, the trend of the deviation changes quite a lot. Initially, the average measured angle is relatively consistent with the expected value for approximately the first 100 degrees. Then it transitions to undershooting and overshooting for the remainder of the revolution.

Figure 7.4 presents the results concerning square summation of X and Y with the results of the raw resolver values X and Y.

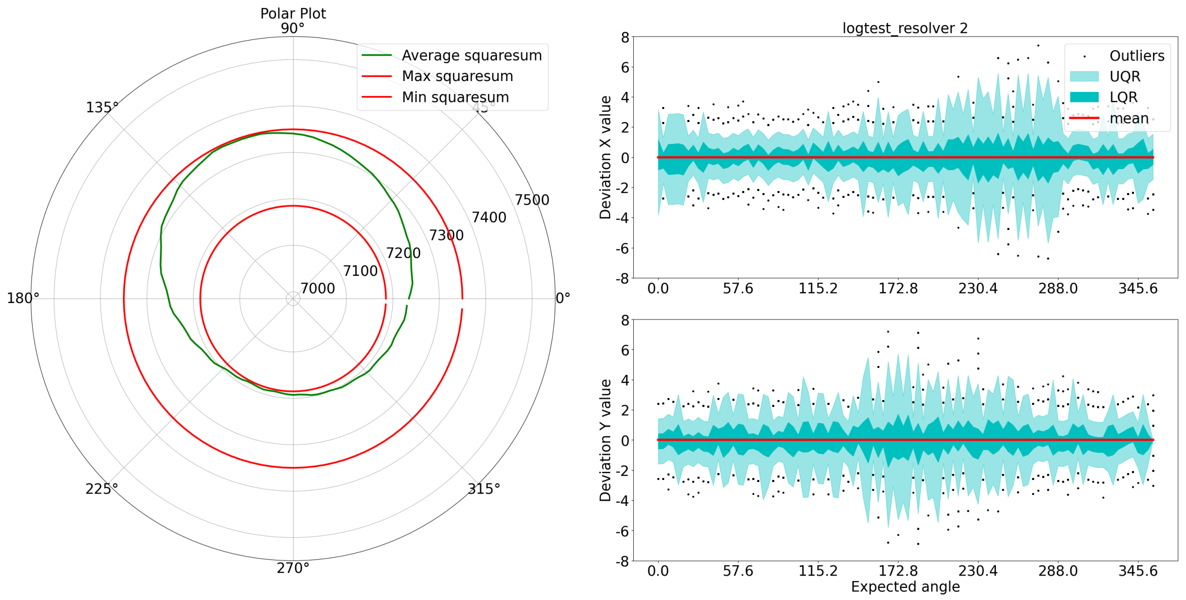


Figure 7.4: This figure shows the results for resolver 2 concerning the square summation of X and Y in the left plot and the deviation around the average value for X and Y in the two plots to the right. LQR represents 50% of the measurements, and UQR represents about 99.1% of the measurements. Outliers are the measurements that fall outside of UQR.

As shown in Figure 7.4, the square summation between the raw values X and Y is inconsistent throughout a complete revolution. The disparity between the lowest and the highest square summation is approximately 160. The noise on X and Y is relatively insignificant and consistent throughout a complete revolution.

7.2.2 Resolver 3

Figure 7.5 presents the results for the angle measurements for resolver 3.

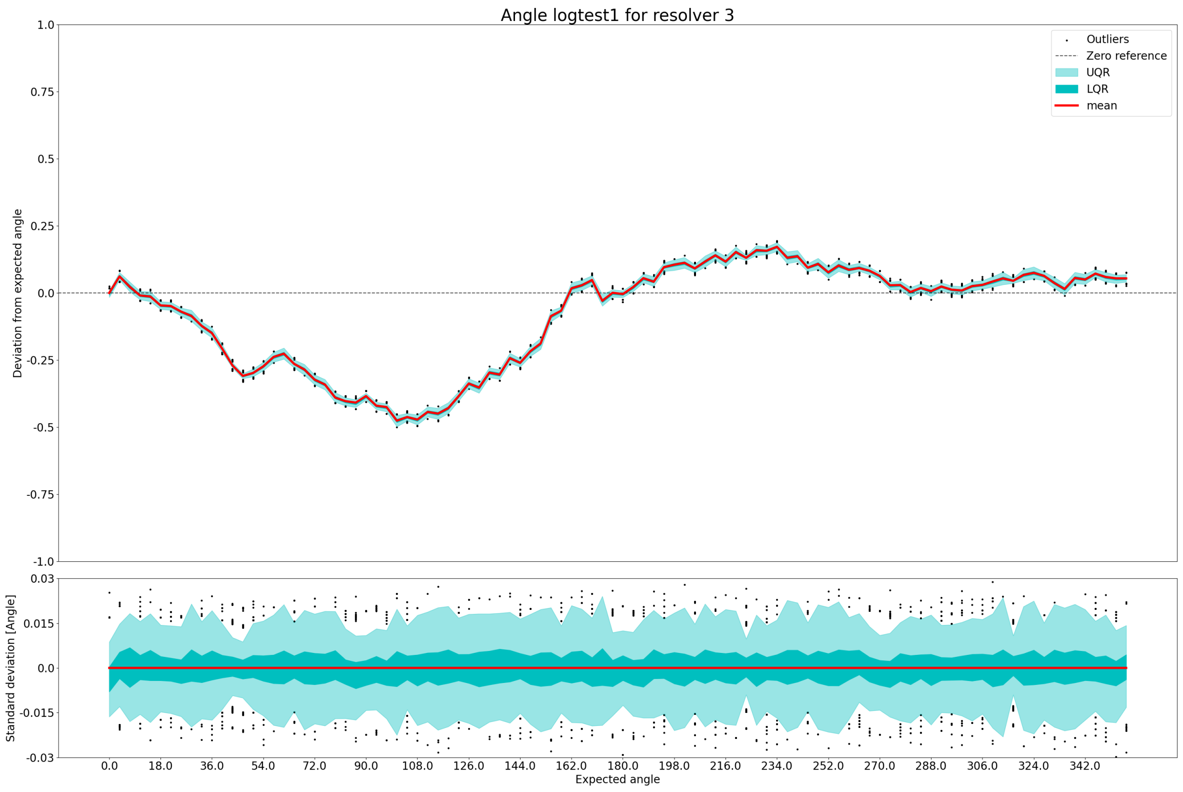


Figure 7.5: This figure shows the angle quality measurement results for resolver 3. The upper plot shows the difference between the measured and expected angles, and the lower plot shows the variance around the average measured angle. LQR represents 50% of the measurements, and UQR represents approximately 99.1% of the measurements. Outliers are the measurements that fall outside of LQR and UQR.

The trend of the average deviation in this plot is different in contrast to the results for resolver 2. Initially, the resolver measurements consistently undershoot for approximately 100 degrees. Then it compensates for the accumulated undershooting by overshooting until it reaches 162 degrees. Then the measurements become stable with a relatively low deviation. The lower plot shows the noise is relatively stable for the complete revolution.

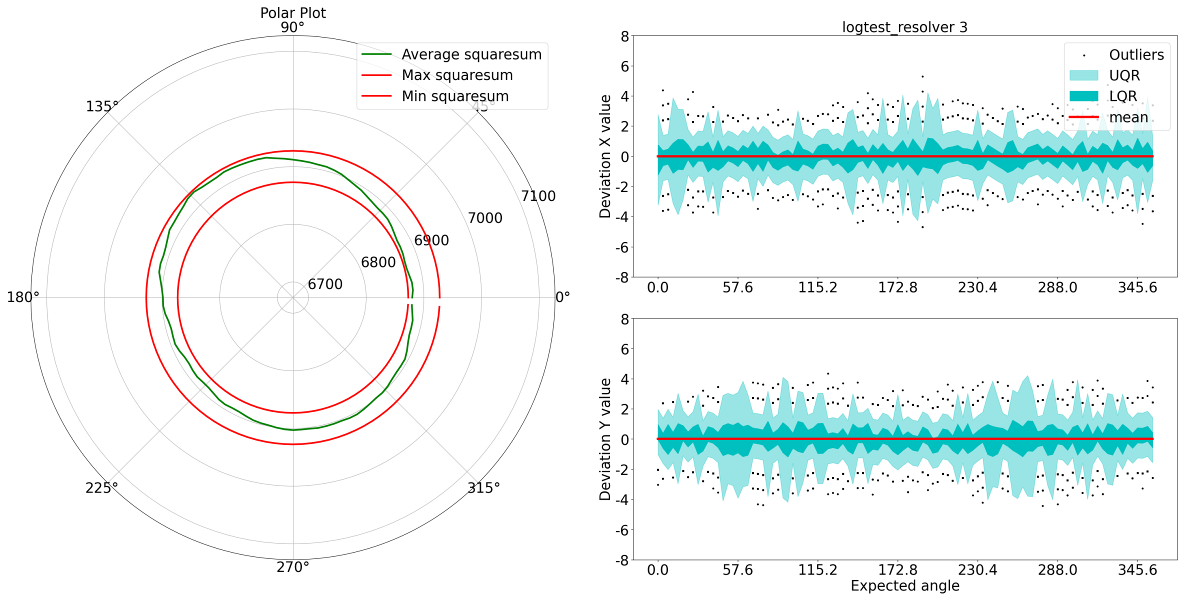


Figure 7.6: This figure shows the results for resolver 3 concerning the square summation of X and Y in the left plot and the deviation around the average value for X and Y in the two plots to the right. LQR represents 50% of the measurements, and UQR represents about 99.1% of the measurements. Outliers are the measurements that fall outside of UQR.

In contrast to the polar plot for resolver 2, this resolver is relatively consistent throughout a complete revolution. The disparity between minimum and maximum square summation is approximately 40. The noise on X and Y is relatively insignificant and consistent throughout a complete revolution.

7.2.3 Resolver 4

Figure 7.7 presents the results for the angle measurements for resolver 4.

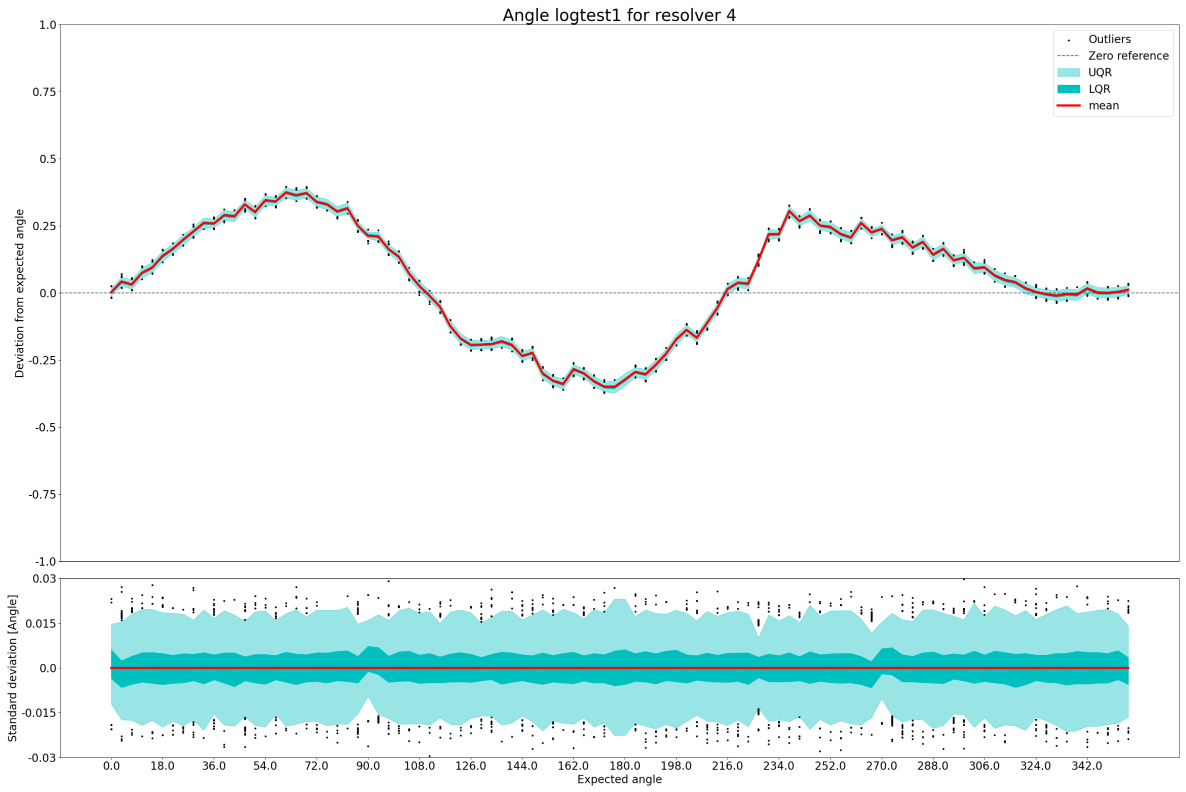


Figure 7.7: This figure shows the angle quality measurements results for resolver 4. The upper plot shows the difference between the measured and expected angles, and the lower plot shows the variance around the average measured angle. LQR represents 50% of the measurements, and UQR represents approximately 99.1% of the measurements. Outliers are the measurements that fall outside of LQR and UQR.

For this resolver, the measurements show that the resolver initially overshoots for the first 72 degrees, then undershoots until it reaches approximately 180 degrees. This pattern repeats, compensating for the accumulated deviation at the end of the resolver revolution. Similarly to previous resolver results, the noise on this resolver is consistent throughout the entire revolution.

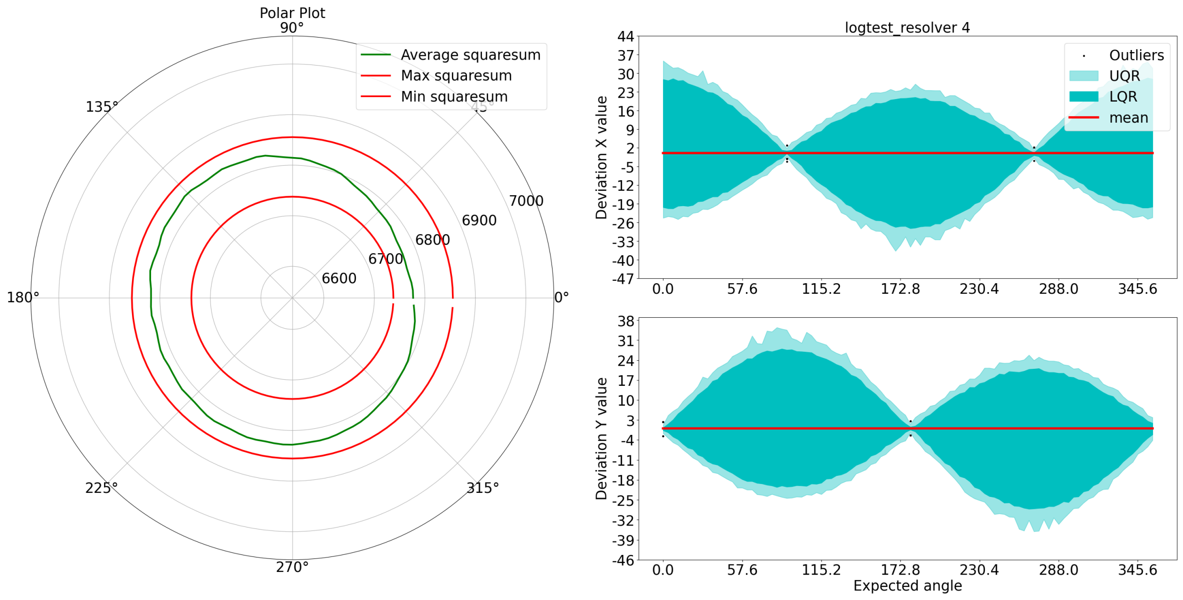


Figure 7.8: This figure shows the results for resolver 4 concerning the square summation of X and Y in the left plot and the deviation around the average value for X and Y in the two plots to the right. LQR represents 50% of the measurements, and UQR represents about 99.1% of the measurements. Outliers are the measurements that fall outside of UQR.

The average square summation observed in 7.8 is relatively consistent. However, the minimum and maximum square summation is approximately 120, which is quite significant. The average square summation values are considerably distant from the minimum and maximum values, indicating a high magnitude of noise on X and Y values. The noise plots for X and Y further support this observation. Additionally, it is worth noting that the magnitude of noise peaks in both noise plots when the raw values for X and Y are at their maximum, respectively.

7.2.4 Resolver 6

Figure 7.9 presents the results for the angle measurements for resolver 6.

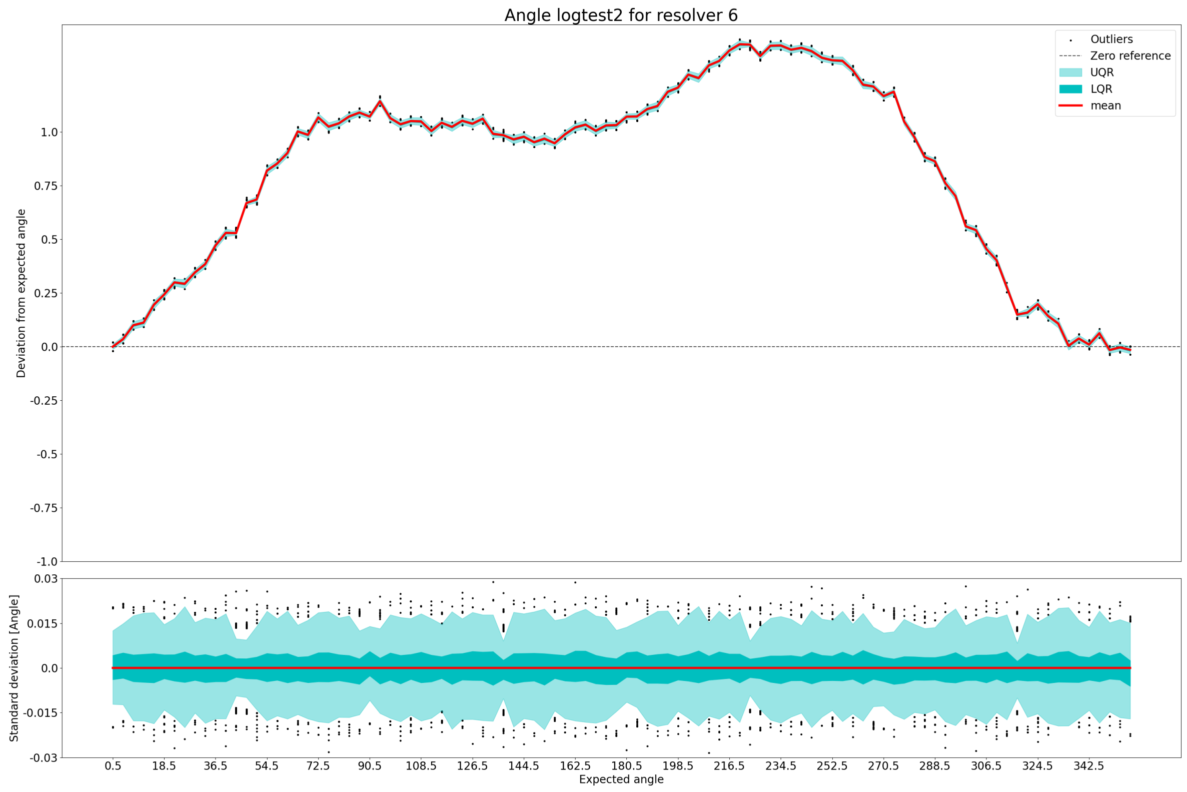


Figure 7.9: This figure shows the angle quality measurements results for resolver 6. The upper plot shows the difference between the measured and expected angles, and the lower plot shows the variance around the average measured angle. LQR represents 50% of the measurements, and UQR represents approximately 99.1% of the measurements. Outliers are the measurements that fall outside of LQR and UQR.

Initially, the resolver measurements exhibit a gradual deviation accumulation due to overshooting. Subsequently, the deviation plateaus for 90 degrees, followed by a small section of overshooting, followed by undershooting to compensate for the accumulated deviation. Similar to the results observed in previous resolvers, the noise on this resolver remains consistent throughout the entire revolution.

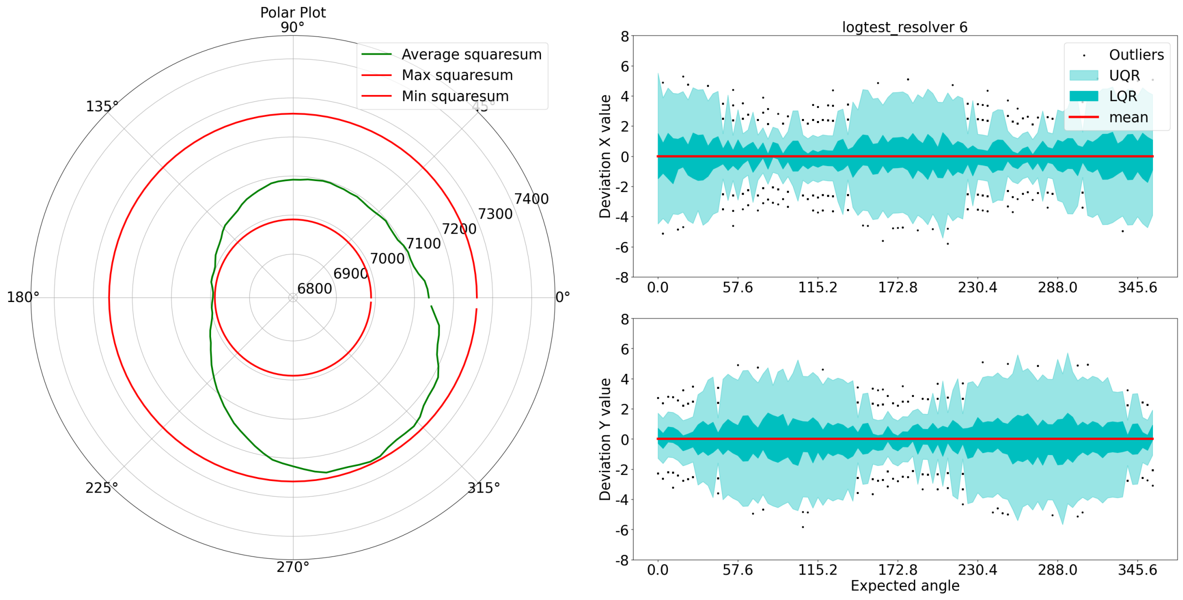


Figure 7.10: This figure shows the results for resolver 6 concerning the square summation of X and Y in the left plot and the deviation around the average value for X and Y in the two plots to the right. LQR represents 50% of the measurements, and UQR represents about 99.1% of the measurements. Outliers are the measurements that fall outside of UQR.

The average square summation observed in Figure 7.10 exhibits the highest level of inconsistency among all the resolvers. The disparity between minimum and maximum square summation is approximately 260. The noise on the X and Y values resembles the noise observed for resolver 4 but at a significantly lower magnitude.

Chapter 8

Discussion

This chapter presents a comprehensive examination of the test results for the test cases. The quality measurement test cases revealed a non-linear trend in the resolver measurements. Additional tests and speculations concerning this non-linearity are conducted. Moreover, suggestions regarding improvements to the test system are presented. Furthermore, a detailed list of recommended future work is outlined. This chapter consists of the following sections:

- **Test Results**
- **Test System Improvements**
- **Future Work**

8.1 Test Results

By assuming the measurements from the SMU is correct, all of the plots display a clear indication of inconsistency in step angles. Due to the fact that this system runs in an open loop, inconsistency in the step angle accumulates. The expected angle is based on the fact that the stepper motor is assumed to perfectly perform a step, in this case, 0.9 and 3.6-degrees. Thus, the error accumulates if the stepper motor undershoots or overshoots for multiple subsequent steps. However, the trajectory of the measured angle always returns the expected value when the stepper motor has completed a full revolution.

8.1.1 Identifying the sources of non-linear measurements

Upon initial observation, the measurements displayed in the quality measurement section seem unique for each resolver. However, upon more profound analysis of the results, it becomes apparent that these observed non-linear trends in the measurements correlate to multiple sources. To narrow down the sources, experiments and speculations regarding these potential factors are made:

1. Stepper motor stepping pattern
2. SMU nodes
3. Stator and rotor eccentricities

Stepper motor stepping pattern

The initial assembly instructions for the RTU specified aligning the stators of the resolvers in a manner that would achieve as close to electrical zero angles as possible. However, the installation of the rotors varied for each resolver, resulting in different electrical zero angles for all resolvers. With different electrical zero angles for all resolvers, the stepper motor starts from different positions for each resolver. Suppose the source of the non-linear trend is associated with the stepper motor stepping pattern. In that case, phase shifting all the resolvers to have the same stepper motor starting position, the non-linear trend should be similar for all resolvers. From observation, with resolver 6 as a reference, the other resolvers are phase shifted as follows:

- Resolver 6 = 0
- Resolver 1 = -57 degrees
- Resolver 2 = -32 degrees
- Resolver 3 = -90 degrees
- Resolver 4 = -169 degrees
- Resolver 5 = -292 degrees

Figure 8.1 displays the result of phase shifting all the resolvers to have the same starting position for the stepper motor as resolver 6.

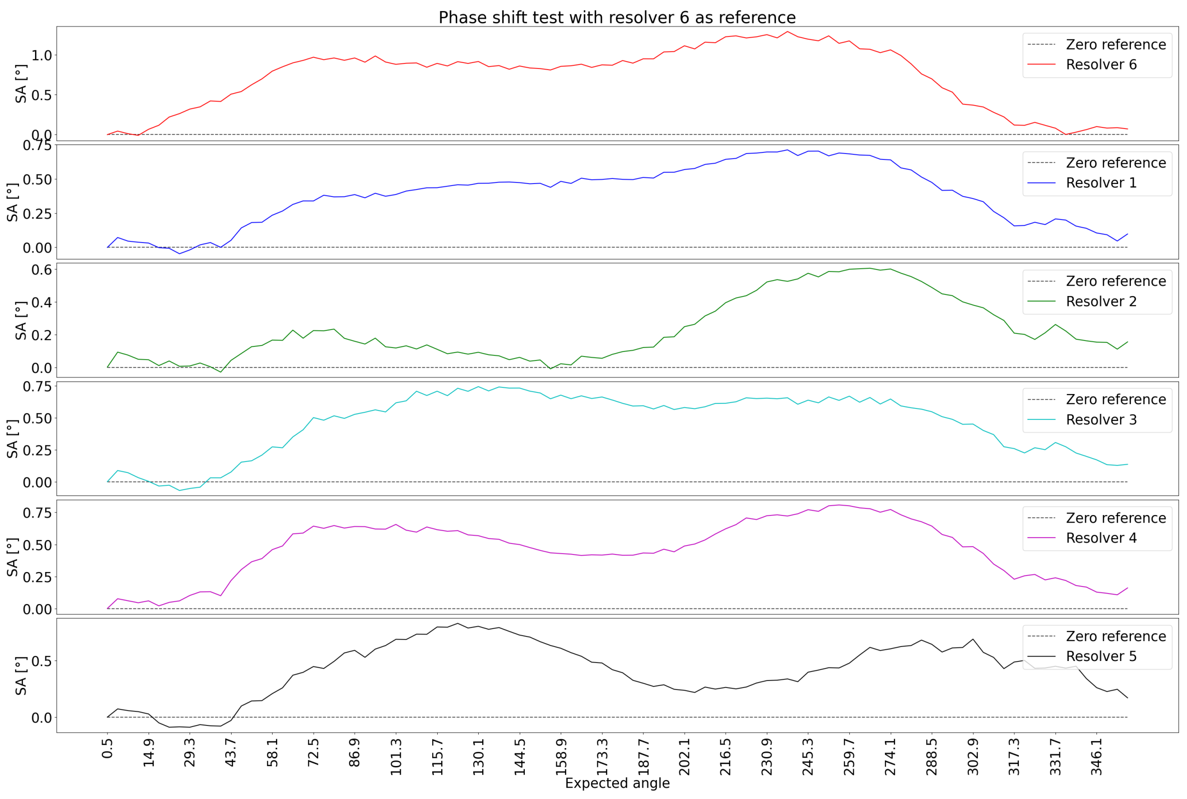


Figure 8.1: Full revolution angle measurement test with all resolvers phase shifted to start at the same stepper motor position as resolver 6. SA stands for Stopping Accuracy.

As shown in Figure 8.1, the trend for all the phase-shifted resolvers now resembles the trend from resolver 6 to different degrees. This means the stepping pattern is one of the sources for this non-linear trend, most likely the cause of the shape of the trend. The stepping pattern is essentially determined by two factors, electrical and mechanical, as explained in section 6.1. From [17], it's mentioned that the mechanical factors exhibit non-linear movement between microsteps, which correlates with the observed trend from the tests. It's also worth noting that the stepper motor used in the test system is relatively inaccurate, with a stopping accuracy of ±5% of a fullstep. Thus, it's assumed that this inaccuracy further enhances the magnitude of the non-linear stepping pattern. The electrical factors may further influence this non-linear stepping pattern if there is a non-matching channel-to-channel current. This is not measured and is only speculation.

SMU nodes

based on the test results from Figure 8.1, it's apparent that the stepping pattern is causing a non-linear trend in the measurements. Due to the fact that the shape of the trend is slightly different from every resolver measurement, there must be other sources affecting the measurements. A brief experiment was conducted to determine if the slight difference in the trend between the resolvers is related to the individual SMU nodes.

This test involved executing the normal quality measurement test for resolver 6 on all nodes. Any inconsistencies in the raw values X and Y will be tied to the resolver nodes for this test. Figure 8.2 shows the behavior of square summation of X and Y for a complete revolution.

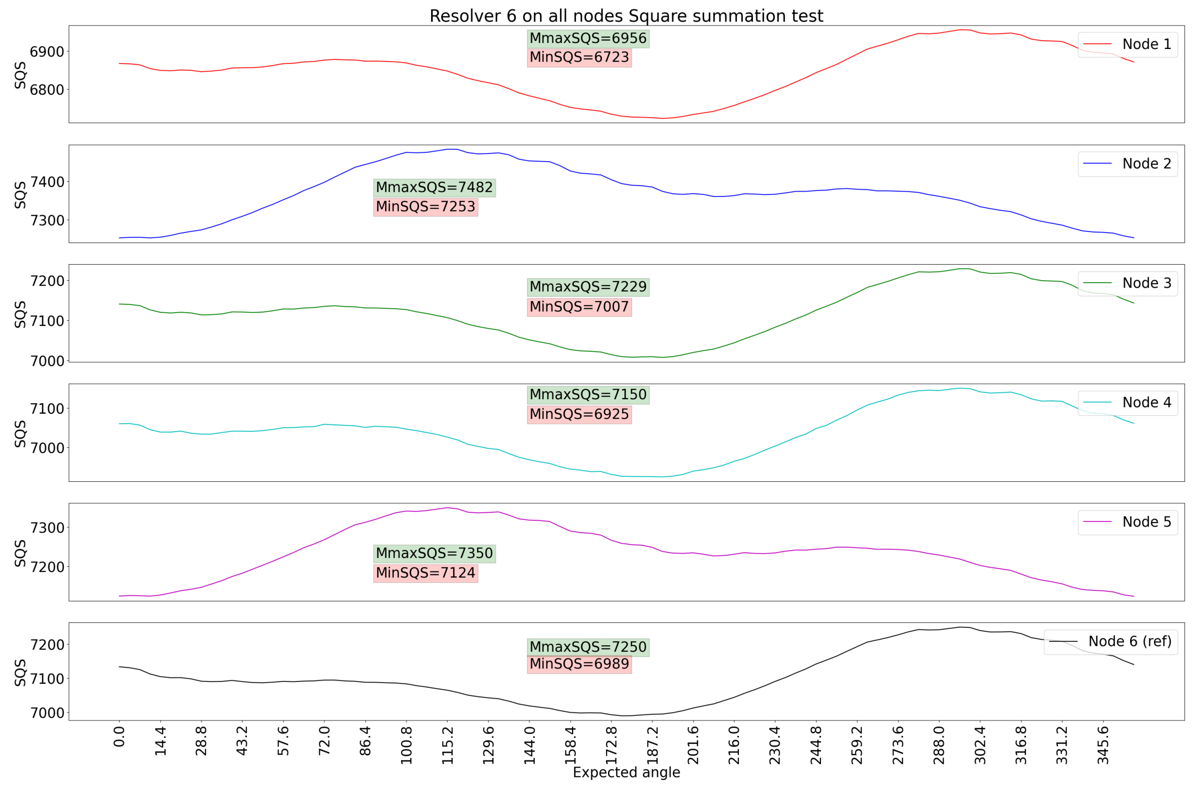


Figure 8.2: This figure shows a complete revolution square summation test for resolver 6 on all nodes. SQS stands for Square Summation.

As shown in Figure 8.2, it's evident that the resolver nodes apply some additional effect to the measurements. Notably, all nodes have the same pattern in Figure 8.2, including 2 and 5, which are phase-shifted approximately 180 degrees. This observation aligns with the fact that the first set of four resolvers is sampled at the top of the feedback signal, whereas resolver 2 and 5 is sampled at the bottom, as illustrated in section 2.2. Furthermore, all nodes have different maximum and minimum average square summation values. This is tied to some internal scaling in the FPGA on the SMU, most likely the scaling of the excitation signal going out to the resolvers. However, the delta SQS between all the nodes is relatively consistent. Due to the fact that the square summation is different for nodes, dependent on which resolver is connected, it's safe to assume something related to the resolver is causing some extra deviation in the measurements. There are mainly two possible factors related to the resolver itself, resolver tolerances and eccentricities between the stator and the rotor of the resolver

Stator and rotor eccentricities
Stator and rotor eccentricities refer to the misalignments or displacements of the stator and rotor from their ideal positions. Multiple sources can cause this, such as manufacturing tolerances, assembly errors, or operational wear and tear. There are mainly three types of eccentricities that can affect the angular measurements for resolver [15]:

- **Radial Eccentricity of the rotor:** Rotor's center is displaced from the resolver's geometric center. This displacement causes variations in the induced voltage across the stator windings.
- **Rotating Eccentricity of the rotor:** The rotor's axis of rotation is not concentric with the stator. Concerning the RTU, this means the shaft is off-center and slightly bent, leading to varying distances between the rotor and stator as the stepper motor rotates.
- **Axial Eccentricity of the rotor** The rotational axis of the rotor is tilted compared to the stator. Concerning the RTU, the shaft will pass through all the resolvers differently, assuming all the stators are perfectly aligned with each other.

Due to the fact that the shaft is equipped with seven resolvers, it's highly possible that one of these eccentricities contributes to the overall deviation in the angular measurements.

Resolver 4 noise
The quality measurement results for resolver 4 in section 7.2 show a high magnitude of noise on the raw feedback values X and Y. The average square summation for this resolver is very consistent due to the noise on X and Y cancel each other out. Additionally, when connecting other resolvers to node 4, the same magnitude of noise was observed. Thus, the observed noise is related to the node itself. Furthermore, similar noise is observed on nodes 5 and 6, but at a lower magnitude. By switching and replacing the SMU with another, the noise disappeared. Therefore, the noise observed in the test results is related to the nodes on the initial SMU under testing. Due to a lack of time, additional testing to identify the source of noise has not been conducted. However, it's suspected that the noise could be related to one or more of the following factors:

- Clock jitter: Variation in the timing of sampling instants on the feedback signal.
- Excitation noise: Noise on the excitation signal going out to the resolvers. Any noise on this signal will transmit over to the feedback signal.
- Hardware issues: Tolerance outliers and hardware defects.

Given that this magnitude of noise is observed on only one SMU, it's most likely a hardware issue on that specific card.

8.1.2 Regression test result

All the test cases have successfully confirmed the respective functionalities are working, except the angle measurement test case. As shown in the test results from the quality measurements, there is a non-linear trend for the measurements originating from multiple sources. The magnitude of this trend results in a significant deviation from the expected values, leading to the failure of the resolver angle measurement test case. This case already has the conservative boundaries of ± 1 degree per step interval. However, with the current inaccurate test system, the results cannot conclude whether the SMU actually passes or fails this test case.

8.2 Test System Improvements

This section presents methods for improving the test system's accuracy and efficiency. Additionally, this section addresses the missing test cases.

8.2.1 Upgrading the stepper controller/driver and stepper motor

The current combination of the stepper motor and stepper motor controller for the test system is not optimal for quality measurement tests. The stepper motor controller does not have the possibility to tune the current decay modus, and the stepper motor needs to be more accurate. When testing the SMU, it's ideal for the test system to be significantly more accurate than the component under testing when executing quality measurement tests. Calibration standards usually apply the Test Accuracy Ratio (TAR) 4:1 or 10:1 ratio when calibrating tools [12], which is not the case for the current test system. It's highly recommended to get a high-precision stepper motor and a new stepper motor driver/controller as future upgrades to the test system if open loop operation is preferred.

8.2.2 Error compensation

After upgrading the stepper motor and stepper motor driver/controller, the stepper motor will always exhibit some non-linearity [17]. Based on observations from the test results for the current test system, the stepper motor pattern is relatively consistent. Thus, it's recommended to use high-precision tools to measure the stepper motor pattern of the upgraded test system. By having reliable data regarding the stepping pattern, one can more precisely predict the expected angle by compensating for the known deviation. Which means the calculations regarding quality angle measurements will be more precise.

8.2.3 Alternative resolver control configuration

An alternative approach to improving the test system is to transition it from an open loop to a closed loop system. This opens up the possibility of using a servo motor instead of a stepper motor. In general, compared to a stepper motor, servo motors are more accurate in a closed-loop system [4][10]. Additionally, it's essential to choose the proper components and calibrate the closed loop system to acquire a TAR of 4:1 concerning the SMU.

8.2.4 Reworking the Resolver Test Unit

From 8.1.1, it has been suggested that the alignment of the shaft in the RTU may be subject to potential deviations, which can result in inaccuracies in angular measurements. A possible solution to this problem is to rework the RTU to control all the resolvers in parallel rather than in series, as illustrated in Figure 8.3.

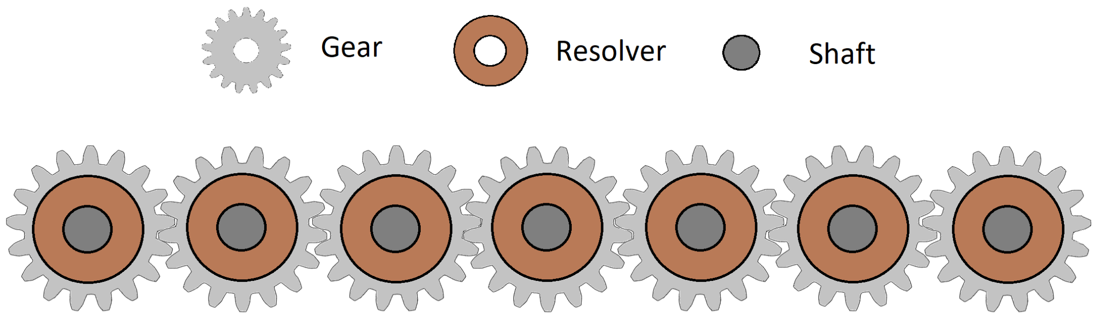


Figure 8.3: Parallel shaft configuration for the Resolver Test Unit.

From Figure 8.3, the gear in the middle is the "main gear," which is directly attached to the motor. Six more gears are arranged around the main gear. All the gears have a shaft going through them with a resolver installed. With this configuration, it may be easier to achieve manufacturing tolerances and fewer assembly errors. Additionally, it will be easier to adjust individual resolvers if errors occur. However, different types of problems may arise with a configuration like this, one being backlash [5].

8.2.5 Optimising test cases

Sleep functions and resolver speed

The current version of the test cases lacks optimization for time efficiency. Multiple sleep functions sleep longer than necessary, and the speed and acceleration can be further increased for many test cases. By optimizing the stepper motor speed and sleep functions, the regression tests can be notably faster. This would provide additional time for gathering data during the quality measurement test case.

Resolver readings

Currently, the tests are executed on a single resolver at a time. As a result, each test case needs to be repeated six times in order to cover all the resolvers. This considerably delays the test cases' speed, particularly the regression tests. However, there is an opportunity to incorporate the fact that the resolvers are phase-shifted relative to one another and adjust for it within the base module in the test framework. By doing so, it is possible to read all the resolvers at each step, significantly improving the efficiency of the tests.

8.2.6 Missing test cases

Due to some firmware problems in the SMU and lack of time, the following test cases did not get implemented in the test suite:

- SMU register access
- Enable 7 axis
- Motor revolution counter high-speed test, normal and battery modus
- Quadrant check test in battery modus

SMU register access and 7 axis
The new FPGA firmware for the SMU had some problems with the read-and-write register functionality. Therefore it was not possible to activate resolver node 7. The developer for this FPGA firmware did not have time to debug this problem, resulting in the testing of only six resolvers instead.

Motor revolution counter high speed
The idea behind this test case is to check if the revolution counter were able to update the revolution counter for high stepper motor speeds properly. This test case did not get prioritized, thus, not implemented.

Quadrant check test in battery modus
The objective of this test case is to verify if the SMU could detect the quadrant in which the resolvers are positioned while in battery modus. This test case was not given priority and consequently did not get implemented.

8.3 Future Work

This section presents a concise plan for recommended future work for the test system and test cases:

1. Upgrade to a closed loop system
   - Change the stepper motor with a high-precision servo motor.
   - Install a high-precision encoder on the servo motor or RTU.
   - Change the stepper motor controller SMSD with a servo motor driver/controller.
   - Develop software for controlling the servo motor in the same fashion as the current test system.
   - Integrate the new software into the existing test framework
2. Calibrate the test system.
   - Use an appropriate tool, such as an oscilloscope, with a test accuracy ratio of 4 or higher to calibrate and verify the test system.
3. Develop the missing test cases
   - SMU register read/write test case
   - Enable 7 axis
   - Motor revolution counter high-speed test, normal and battery modus
   - Quadrant check test in battery modus
4. Optimize test cases
   - Optimize the time for sleep functions
   - Rework relevant test cases to test all resolvers in parallel rather than individually

Chapter 9

Conclusion

In this thesis, an open-loop automatic test system was developed and integrated into ABB's test environment to perform regression and quality tests on the SMU. The automatic test system performed consistently well for test cases that did not require high-precision angle measurements. Based on quality measurement test results and additional testing, it became apparent that an open-loop configuration with the current hardware is unsuitable for high-precision angle measurement testing. The stepper motor had a non-linear stepping pattern throughout a complete revolution, and eccentricities in the RTU shaft are suspected. These two problems are most likely the most influential factors regarding inaccuracies in the test system. Thus, the test system must change to a closed-loop configuration or upgrade to high-quality hardware to qualify for high-precision angle measurement test cases. In addition to one of these requirements, the test system needs to be calibrated with a TAR of 4:1 or higher.

Bibliography

[1] ABB AS, Robotics. Funktionsspecifikation; Drivdon S4C. Availabel on request.

[2] ABB AS, Robotics. SMS Functional Interface Specification, 3HNE 07564-1 rev 02 interface spec. Availabel on request.

[3] ABB AS, Robotics. Unit desciption IRC5P. Availabel on request.

[4] B. Lackey. Stepper and Servo Motor Tradeoffs. `https://www.machinedesign.com/automation-iiot/article/21836868/stepper-and-servo-motor-tradeoffs`, accessed Jul. 13, 2023, 2023.

[5] Carlos H. Wink. "Gear Backlash Analysis of Unloaded Gear Pairs in Transmission. `https://www.geartechnology.com/ext/resources/issues/0616x/backlash.pdf`, accessed Jul. 07, 2023.

[6] Cornell.edu. Two's Complement, 2023. `https://www.cs.cornell.edu/~tomf/notes/cps104/twoscomp.html`,Accessed 25.06.2023.

[7] D. Collins. What is current decay (recirculating current) in a stepper drive?, 2023. `https://www.motioncontroltips.com/what-is-current-decay-in-a-stepper-drive/`,Accessed 28.06.2023.

[8] GeeksforGeeks. Longitudinal Redundancy Check LRC 2 D Parity Check. `https://www.geeksforgeeks.org/longitudinal-redundancy-check-lrc-2-d-parity-check/`, accessed Jul. 08, 2023, 2023.

[9] GeeksforGeeks. Regression testing, 2023. `https://www.geeksforgeeks.org/software-engineering-regression-testing/`, Accessed 01.03.2023.

[10] ISL Products Internationa. Stepper Motors vs. Servo Motors. `https://islproducts.com/design-note/stepper-motors-vs-servo-motors/`, accessed Jul. 13, 2023, 2023.

[11] Leodanis Pozo Ramos. Python's assert: Debug and Test Your Code Like a Pro, 2022. `www.realpython.com/python-assert-statement/`,Accessed 2.06.2023.

[12] Mitutoyo Institute of Metrology. Decision Rules, TAR, and TUR. `https://www.mitutoyo.com/webfoo/wp-content/uploads/15005A.pdf`, accessed Jul. 07, 2023.

[13] Rezadlt. DevSecOps. `https://commons.wikimedia.org/wiki/File:DevSecOps.jpg`, accessed Jul. 13, 2023, 2023.

[14] Schneider Electric USA. What is Modbus and How does it work? `https://www.se.com/us/en/faqs/FA168406/`, accessed Jul. 08, 2023, 2023.

[15] Jing Shang, Hao Wang, Mimi Chen, Ning Cong, Yong Li, and Chengjun Liu. The effects of stator and rotor eccentricities on measurement accuracy of axial flux variable-reluctance resolver with sinusoidal rotor. pages 2004–2007, 01 2015.

[16] Smd.ee. SMSD-4.2Modbus Smart Motor Devices OÜ, 2023. `https://smd.ee/smsd-4.2modbus.htm`,Accessed 01.03.2023.

[17] Texas Instruments. How to Improve Motion Smoothness and Accuracy of Stepper Motors., 2023. `https://www.ti.com/lit/an/sloa293a/sloa293a.pdf?ts=1687701986945`,Accessed 26.06.2023.

Appendix A

Appendix

参考図　沼田　泰男　2017/06/22 08:18:07

ステップモータ仕様書
STEP MOTOR SPECIFICATIONS

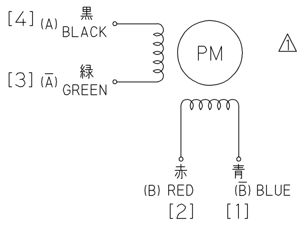
REVISIONS		
No.	DESCRIPTION	DATE／SIGN
⚠1	コネクタ端子番号追加 ADD PIN NUMBER OF CONNECTOR	11.10.21

形 式 MODEL	TS3617N459

電 気 的 特 性 ELECTRICAL CHRACTERISTICS		
基本ステップ角度 FUNDAMENTAL STEP ANGLE	1.8°	
励 磁 方 式 DRIVING METHOD	バイポーラ2相励磁 BIPOLARA 2PHASE ON	
定 格 電 圧 RATED VOLTAGE	DC 3.8 V	
定 格 電 流 RATED CURRENT	DC 1.3 A／相(PAHSE)	
巻 線 抵 坑 WINDING RESISTANCE	2.9 Ω ± 10%	
巻線インダクタンス WINDING INDUCTANCE	5.0 mH ±20%	1Vrms 1kHz
絶 縁 階 級 INSULATION CLASS	CLASS B	リード線を除く W/O LEAD WIRE
絶 縁 抵 坑 INSULATION RESISTANCE	100 MΩ MIN.	DC 500 V
絶 縁 耐 圧 DIELECTRIC STRENGTH	500V AC 1 MIN.	

モ ー タ 性 能 MOTOR PERFORMANCE			
静止角度誤差 POSITIONAL ACCURACY	1.8 °± 5%		1
隣接角度誤差 STEP POSITION ACCURACY	1.8 °± 5%		1
ホールディングトルク HOLDING TORQUE	0.441N·m (4.5kgf-cm) MIN.	2相励磁 1.3A／相 AT 2PHASE ON. 1.3A/PHASE	
ディテントトルク DETENT TORQUE	7.061×10^{-3} N·m (72×10^{-3} kgf-cm) MIN.		
許容温度上昇値 PERMISSIBLE TEMPERATURE RISE	80 °C MAX.	2相励磁 DC3.8V AT 2PHASE ON. 3.8VDC	2

⚠1 結 線 図 SCHEMATIC



下記励磁順序にて取付け面側から見て出力軸が CW 回転
SWITCHING SEQUENCE FOR CW ROTATION VIEWED FROM MOUNTING END

⚠1

STEP	黒 BLACK (A) [4]	赤 RED (B) [2]	緑 GREEN ($\bar{A}$) [3]	青 BLUE ($\bar{B}$) [1]
0	+	+	−	−
1	−	+	+	−
2	−	−	+	+
3	+	−	−	+
4	+	+	−	−

⚠1 ※ [ ]：コネクタ端子番号　※ [ ]:PIN NUMBER OF CONNECTOR

備考 . REMARKS

1 定格電流 1.3A／相にて2相励磁し、TSC標準ドライバにて測定する。
AS MEASURED RATED CURRENT 1.3A/PHASE AT 2 PHASE ON WITH TSC S.T.D. DRIVER.

2 定格電圧 3.8VDCにて2相励磁し、静止状態(0pps)で抵抗法にて測定する。
AS MEASURED BY THE CHANGE IN RESISTANCE METHOD. WITH RATED VOLTAGE 3.8VDC AT 2PHASE ON WITH HOLD (0pps)

機 械 的 特 性 MECHANICAL CHARACTERISTICS		
外 形 図 OUTLINE	0TD006965Y00	
回転子イナーシャ ROTOR INERTIA	57×10^{-7} kg·m^2 (57g-cm^2)	
ラジアルプレイ RADIAL PLAY	0.02 mm MAX.	4.904N·m (0.5kgf) LOAD
スラストプレイ THRUST PLAY	0.075 mm MAX.	9.807N·m (1.0kgf) LOAD
質 量 MASS	0.24kg NOM.	

環 境 条 件 ENVIRONMENT CONDITION	
使用温度範囲 OPERATING TEMP.RANGE	-20°C~ +50°C
使用相対湿度範囲 OPERATING HUMIDITY RANGE	5% RH~ 95% RH
保存温度範囲 STORAGE TEMP.RANGE	-40°C~ +70°C

RoHS

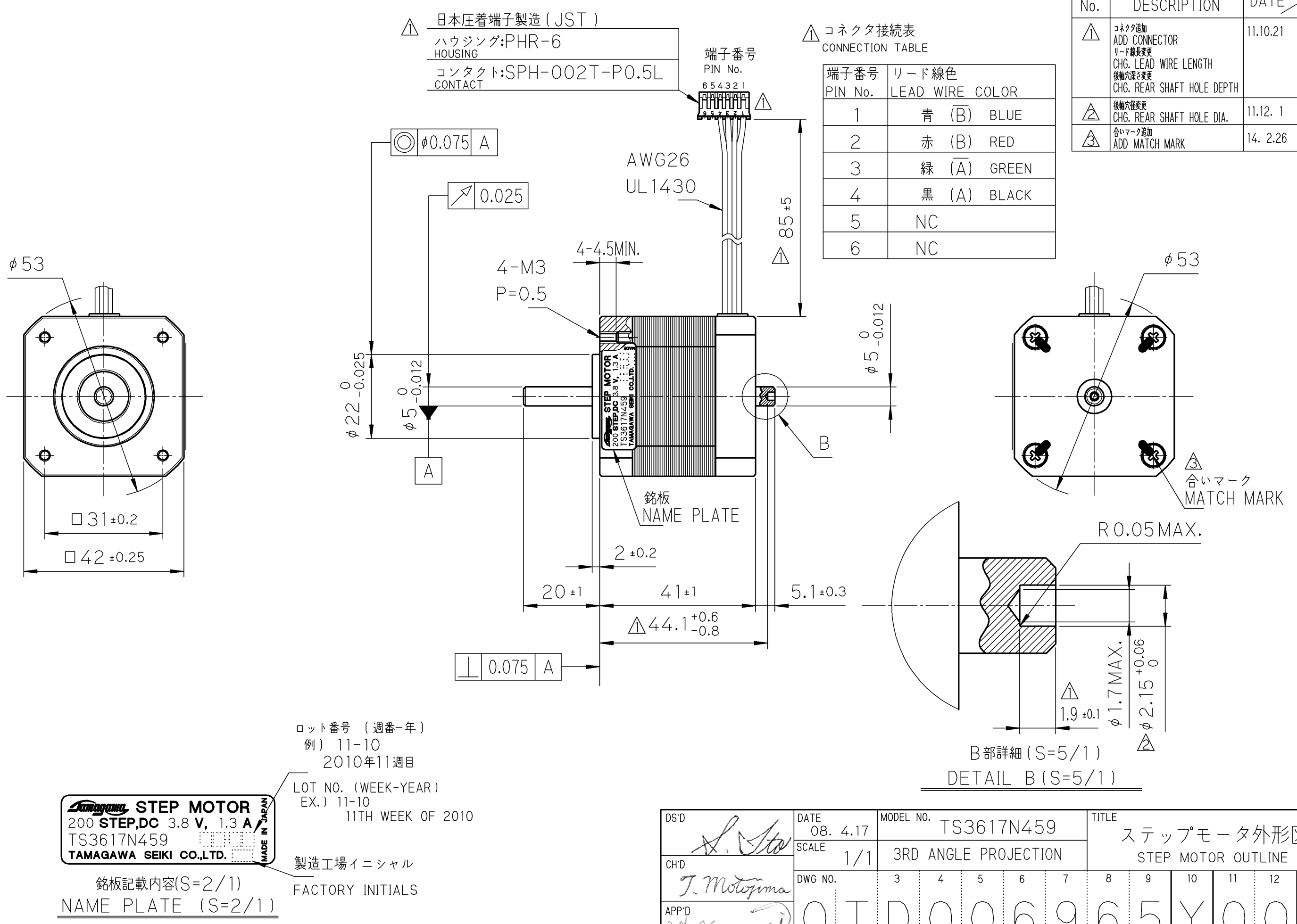
DS'D S. Ito	DATE 08. 4.17	MODEL NO. TS3617N459	TITLE ステップモータ仕様書 STEP MOTOR SPECIFICATIONS
CH'D T. Motojima	SCALE ╳	3RD ANGLE PROJECTION	
APP'D [illegible]	DWG NO. SPC009278Y00		SHEET ／

JIS A3(297X420) A

TAMAGAWA SEIKI CO.,LTD.

参考図

沼田　泰男　2017/06/22 08:17:44



REVISIONS

No.	DESCRIPTION	DATE / SIGN
1	コネクタ追加 ADD CONNECTOR リード線長変更 CHG. LEAD WIRE LENGTH 後軸穴深さ変更 CHG. REAR SHAFT HOLE DEPTH	11.10.21
2	後軸穴径変更 CHG. REAR SHAFT HOLE DIA.	11.12. 1
3	合いマーク追加 ADD MATCH MARK	14. 2.26

コネクタ接続表
CONNECTION TABLE

端子番号 PIN No.	リード線色 LEAD WIRE COLOR
1	青 (B) BLUE
2	赤 (B) RED
3	緑 (A) GREEN
4	黒 (A) BLACK
5	NC
6	NC

RoHS

DS'D S. Ito	DATE 08. 4.17	MODEL NO. TS3617N459	TITLE ステップモータ外形図 STEP MOTOR OUTLINE
CH'D T. Motojima	SCALE 1/1	3RD ANGLE PROJECTION	
APP'D [illegible]	DWG NO. 0 T D 0 0 6 9 6 5 Y 0 0		SHEET ／

JIS A3(297X420) A

TAMAGAWA SEIKI CO.,LTD.