TALLINNA TEHNIKAÜLIKOOL
INSENERITEADUSKOND
Elektroenergeetika ja mehhatroonika instituut

KLIIMATESTERI EHITAMINE JA ESMASE KATSE LÄBIVIIMINE

CONSTRUCTION OF CLIMATE TESTER AND CONDUCTING THE INITIAL TEST

BAKALAUREUSETÖÖ

Üliõpilane: Voldemar Balder
Üliõpilaskood: 123713MAHB
Juhendaja: Leo Teder
Juhendaja: Risto Rosin

Tallinn 2021

AUTORIDEKLARATSIOON

Olen koostanud lõputöö iseseisvalt.
Lõputöö alusel ei ole varem kutse- või teaduskraadi või inseneridiplomit taotletud.
Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

“.......” 2021
Autor: ....
/ allkiri /

Töö vastab bakalaureusetöö/magistritööle esitatud nõuetele

“.......” 2021
Juhendaja: ....
/ allkiri /

Kaitsmisele lubatud
“.......”....................2021 .

Kaitsmiskomisjoni esimees ....
/ nimi ja allkiri /

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina, Voldemar Balder sündinud 01.10.1992

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose KLIIMATESTERI EHITAMINE JA ESMASE KATSE LÄBIVIIMINE, mille juhendaja on Leo Teder.

1.1 reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;

1.2 üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.

3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

---

[1]Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil.

_______________ (allkiri)

_______________ (kuupäev)

Elektroenergeetika ja mehhatroonika instituut

Kliimatesteri ehitamine ja esmase katse läbiviimine

Üliõpilane: Voldemar Balder, 123713MAHB

Õppekava, peaeriala: MAHB02/13 - Mehhatroonika

Juhendaja(d): Leo Teder, lektor
Risto Rosin, kvaliteedi ja töökindluse juhataja

Lõputöö teema:
Kliimatesteri ehitamine ja esmase katse läbiviimine
Construction of climate tester and conducting the initial test

Lõputöö põhieesmärgid:
1. Ehitada töötav ja kasutatav kliimatester trükkplaatide testimiseks.
2. Läbi viia esimene katse.

Lõputöö etapid ja ajakava:

Nr	Ülesande kirjeldus	Tähtaeg
1.	Testeri kirjeldus	18.03
2.	Katse kirjeldus	25.03
3.	Andmebaasi ja sinna kirjutamise kirjeldus	01.04
4.	Katse tulemuste analüüs	08.04

Töö keel: Inglise keel **Lõputöö esitamise tähtaeg:** “18”mai 2021a

Üliõpilane: Voldemar Balder .... “.......”....................2021a
/allkiri/

Juhendaja: Leo Teder .... “.......”....................2021a
/allkiri/

Juhendaja: Risto Rosin .... “.......”....................2021a
/allkiri/

TABLE OF CONTENTS

PREFACE

Climate tester Ocala is built by the request of an unnamed electronics company. Purpose of the tester is to verify printed circuit board assembly quality through on-going reliability testing with higher humidity and temperature than the printed circuit board assemblies would see in the field. Environmental testing should also pinpoint weakness in the design and used components or manufacturing process.

The following people helped building climate test Ocala and analyzing the results: Juri Matin, Priit Reim, Hamad Aziz, Martin Onton, Andres Võsa, Risto Rosin, Erkki-Siim Lind, Mark Toomis, Ossi Myllyniemi, Jaan Sarap, Stanislovas Krisciunas.

Climate testing, Programmable Logic Controller, SQL, Printed Circuit Board Assembly, Bachelor's thesis

ABBREVIATIONS

EUT	Equipment under test
PLC	Programmable logic controller
PCBA	Printed Circuit Board Assembly
AC	Alternating current
DC	Direct current
ALT	Accelerated lifetime test
THB	Temperature humidity bias
ORT	Ongoing Reliability Test
DT	Developers Tool
DRC	Drives reliability center
MS	Microsoft
SQL	Structured Query Language
IGBT	Insulated Gate Bipolar Transistor
ORT	Ongoing reliability test
RH	Relative humidity
PID	Proportional–integral–derivative

INTRODUCTION

Purpose of the thesis is to build an automatic climate tester which temperature and humidity is controllable and conduct the initial accelerated lifetime test. Purpose of the automatic climate tester is to conduct product reliability demonstration tests on frequency converter PCBAs with accelerated lifetime testing to verify that there are no faulty components used or problems in manufacturing process and to find weaknesses in the design. This is needed to find problems in product and to reduce the number of faulty components reaching customers. PCBAs are put into higher ambient and humid conditions then they would see in the field and therefore their lifetime is accelerated. 24V DC, 230V AC and 1050V DC power is also cycled to simulate real life conditions and self-heating of the PCBAs.

Purpose of the test is to monitor PCBA quality and long-term reliability. Tester planning began in mid-2018, building started in the beginning of 2019, initial test started in May 2019, initial test ended in August 2020. Analysis of the PCBAs finished 26.03.2021.

Ocala tester is built around Arctest climate chamber and uses AC500 PM590 PLC to control and monitor both EUTs and test equipment. PLC also saves data to MS SQL server located in the local computer.

End user of tester Ocala must be able to:

- easily monitor and adjust ambient conditions
- level of allowed current and voltage to EUTs
- control what parameters and sensor readings are monitored and saved to database
- get notifications when faults occur
- get minimum and maximum values of each cycle
- get a fault log.

MS SQL server is used to:

- pull data from local computer
- store data
- calculate minimum and maximum values
- create a fault log table
- summary of the tests
- notify user of faults.

1 BUILDING OCALA

Tester building starts with a list of needs from internal customer and then needed equipment is selected and agreed upon to fulfill those needs. As we already had the climate chamber, we were a bit limited by its capability but not too much.

- ALT booking template – short general description of what, when, how and how long will be tested.
    - Test specification – in detail document describing how the test must be carried out
    - Tester specification - in detail document of how the tester will be built and must be able to do, this is also the start point for electrical engineer
        - PLC specification – document describing what the PLC must do, usually given to outside contractor

Ocala tester is built around Arctest climate chamber and uses AC500 PM590 PLC to control and monitor both EUTs and test equipment. PLC also saves data to MS SQL server located in the local computer.

1.1 Used equipment

- Climate chamber: ARC-1500/-40+125/RH, with JUMO IMAGO controller
- PLC: AC500 PM590-ETH with AI523, DC532, AX522, CM572-DP
- Computer: ThinkPad P50
- DC power supply Programmable DC-Power Supply | 8 kW, 1250 V" Magna Power XR1250-6.4-380
- MUX: Keysight 34970A Data Acquisition Unit, with 3 34901A 20-Channel Armature Multiplexer
- Chiller: CLIVET WSAN-XIN 21-141
- Circulation pump: Grundfos alpha1 l 25-40
- Power meter: WM30-96 AV53HE2
- Additional humidity sensor: HTM2500LF
- Various other smaller components in control cabinet

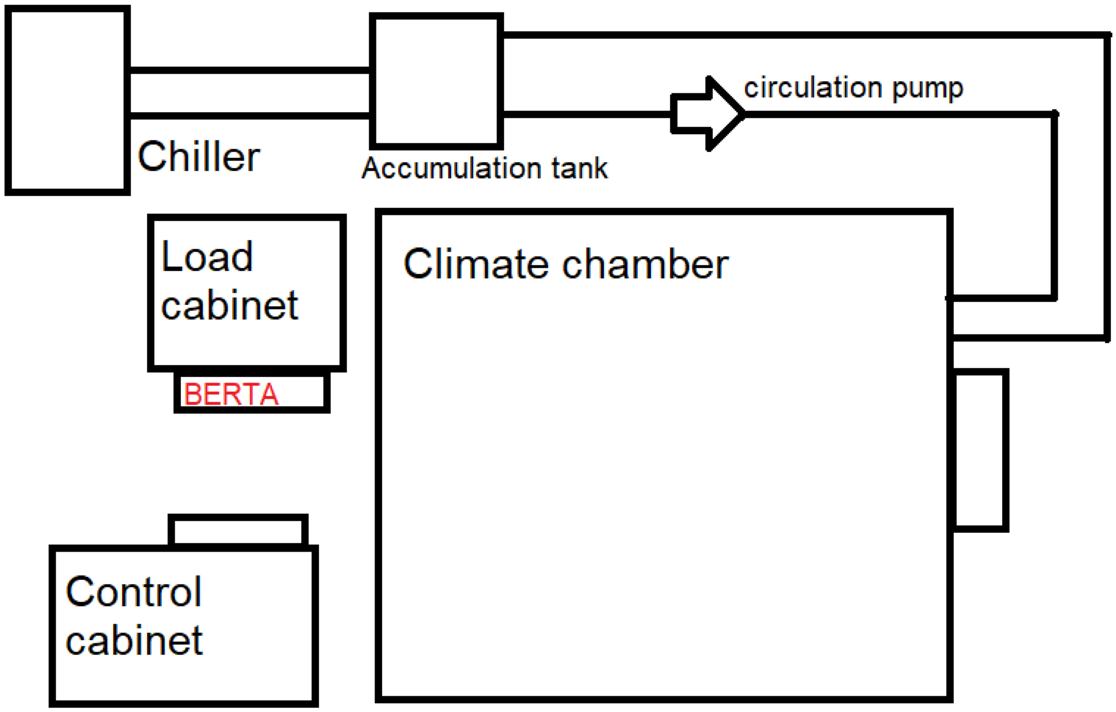


Figure 1.1 Cabinet and panel placement, top view

Climate chamber ARC-1500/-40+125/RH is designed and manufactured especially for controlled environmental testing. The chamber is constructed of three modules. Machine module, chamber module and electronic and electricity module. Controlled by a small JUMO controller. Could work independently supplied with coolant, distilled water, and drainage. [2]

Figure A1.1 Ocala data and control flow chart shows all data connections.

Figure 1.2 Inside view into Arctest climate chamber filled with EUTs

Programmable logic controller AC500 PM590-ETH is a programmable logic controller with 2MB of internal memory, 2 serial connections and 1 ethernet connection. It is programmable form Automation builder program that uses Codesys V2. In this project three I/O modules are used. Figure A2.1 PLC in circuit diagram.

AI523 analog input module with 16 programmable analog inputs is used to monitor temperature inside the chamber with 3 PT100 thermal sensors and one PT100 for both the control cabinet and load cabinet. Coolant liquid pressure levels are also monitored.

DC532 digital input output module has 16 digital inputs and 16 configurable digital inputs/outputs. Digital inputs are mostly used to monitor hard overtemperature faults, statuses of more important relays and chillers and test chambers alarms. The other 16 channels are configured as outputs to control relays providing voltage to EUTs, DC power supply, load cabinet relays, climate chamber and some not important lights.

AX522 Analog Input/Output Module has 8 configurable analog inputs and 8 configurable analog outputs. Inputs were used to read HTM2500LF humidity sensor, outputs to control DC power supply's current and voltage output levels.

CM572-DP PROFIBUS DP Master is used to connect to and control EUTs and climate chamber.

Computer ThinkPad P50 is a regular computer running windows 10. Its job is to run SQL local server, display PLC visualization, Drive Composer and DT.

DC power supply is a programmable DC-Power Supply capable of up to 1250V voltage and 6.4A of current. Controlled by PLC analog outputs and interlock using JS1 connection.

MUX Keysight 34970A Data Acquisition Unit is used to record BDPS output voltage and current and control board input current and voltage. It has 20*3 voltage measuring channels that are needed to configure from the machines display and then tasked to scan the list. 34970A saves the results into its memory where from PLC is asking the data through 34970As RS-232 port 9 PIN D-Sub connector.

Chiller CLIVET WSAN-XIN 21-141 is used to cool down the coolant exiting the climate chamber.

Circulation pump Grundfos alpha1 l 25-40 is used to pump coolant into the climate chamber and is set in underfloor heating mode.

1.2 Design

As you start designing anything new you try to predict as much as possible. Calculate how much power do you need and what control connections and feedbacks are required. The more you design the better you become at it and usually you do not have to start from complete scratch. But as this was the first ever climate tester built in Estonia's testing center we did not have that much to start with. Comparing figure A3.1 first revision of Ocala Main Supply to figure A3.2 latest revision of Ocala Main Supply, quite a few changes can be noticed. We knew that we need power to supply:

- 1000V DC power supply. We calculated that each EUT should require about 500W of power so 4kW in total was needed. Magna XR1250/6.4kW sufficed. [7]
- 230V AC programmable power supply as initially it was requested that AC voltage to supply EUTs power supply boards with 230V AC would also be adjustable.
- 4 24V DC power supplies to supply power to control boards (MCS-B 5-110-240/24)

Additional power from another circuit that would not trip if undervoltage relay would trip was needed to supply PC, MUX and PLC with power. All in all it was not bad at all, we had our power supplies, controls, feedbacks, PLC with AI523 for additional temperature and humidity readings, DC532 for tester control and feedbacks, safety's, components, temperature triggered fans and even some indicator lights. A few more minor details to polish over and we were ready for production. Turns out there are a lot of minor details. Wire markings: colors, cross sections, isolation type, how to control our programmable power supplies, layout of the components in control cabinet, how to connect all the control cabinet wiring to EUTs. We threw away K1 and K2 contactors as there were no actual need for them, added AX522 to control Magna power supply, added fire alarm schematics and connections and marked PLC connection such as they would be the same on the circuit diagrams and in the PLC, something not done before. Most of it we got done before production but many problems still game out during the production such as small wire color mistakes and missing terminals and worst of all some components like 1000V 2A fuses and fuse holders and AC power supply not arriving on time. Because the problems were needed to be solved quickly usually right there on the production, we now have terminals with names "K1 buttons" and XT66. Magna power supply terminal and connections with PLC logic was added as can be seen from figure A3.5 First revision of 1000V DC Supply and figure A3.6 latest revision of 1000V DC Supply. With some more minor changes the control cabinet was finished.

Figure 1.3 Half-finished control cabinet in production

1.3 Equipment under test

8 sets of a frequency converter PCBAs. One set consists of: Control board, Safety option module, Memory unit, Profibus adapter module, Fieldbus kit, Interface board, Gate driver board, Adapter board (x3), Power supply board, Fan power supply board, Communication option module. Figure 1.4

- Control board communicates with interface board, PLC, control panel, safety option module and whatever else is necessary
- Communication option module is optical option module on control board
- PROFIBUS adapter module enables us to create PROFIBUS network between all EUTs and is located on control board.
- Fieldbus kit is a branching unit allowing chain connections through control boards enabling us to control them through 1 control panel
- Memory unit holding drives parameters and program.
- Interface board is communicating with control board and gate driver board.
- Power supply board is supplying PCBAs with 24V. Can be transform either 230V AC or up to 1100V DC into 24V DC.
- Adapter boards are controlled by gate driver board and are responsible for opening and closing IGBTs on IGBT-Module.
- Fan supply board transforms 1000V DC into 48V to supply power to DC fans responsible of cooling the drive.

Most critical of the components are power supply boards as they will be most loaded during the test and have capacitors on them that might not do well in high temperature and high humidity environment.

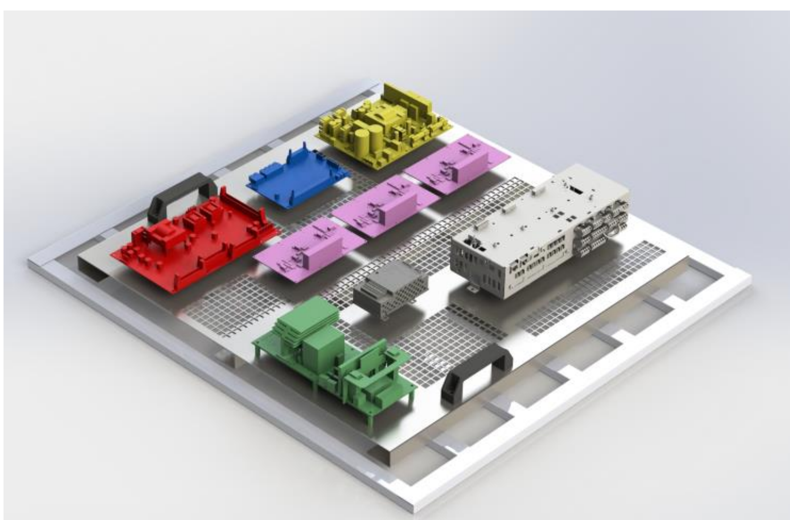


Figure 1.4 3D model of one out of 8 EUT sets.

1.4 Bringing the pieces together

Control cabinet, climate chamber, EUTs and load cabinet – until now they have been in three separate corners and connections between them are only on paper. New requirement of needing to control and measure 24V DC supply to EUTs and measure voltage and current output of power supply boards were requested. It turned out quite soon that we had too many wires to route from control cabinet to EUTs in climate chamber. Difference between initial design can be seen in figure A3.3 and latest design with relay control and shunt measurements in figure A3.4. An additional auxiliary cabinet was added behind the control cabinet to add additional terminals and to ease wiring between control cabinet and EUTs inside climate chamber. Additions can be seen in figures 1.5. and 1.6

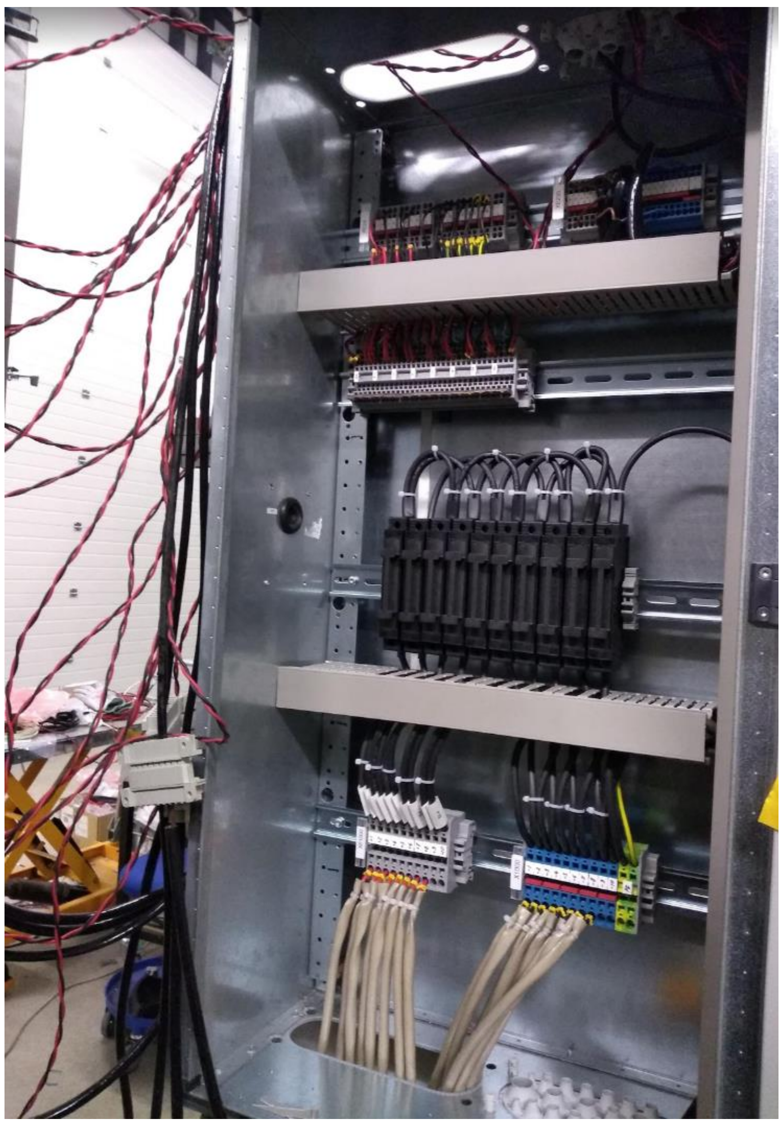


Figure 1.5 Unplanned auxiliary cabinet behind control cabinet

Figure 1.6 Shunt and voltage measurements connected to Keysight 34970A Data Acquisition Unit multiplexer cassette.

We discovered that we cannot start fan supply boards with resistor load as simulated fan loads because there is no feedback and with “always on” setting we cannot stop them from outputting power. Easiest option seemed to add relays to connect and disconnect fan supply boards output to resistive load. This would have been troublesome to do inside the load cabinet because of room restrictions and might have caused us additional problems in the future as with total power output of 3.2kW even with sufficient cooling it might get hot. As each fan supply board has 2*200W at 48V output it meant 2*8 = 16 relay contacts. Initial idea was just to screw few DIN rails to Load cabinet wall and attach some relays and terminals. As this would have been esthetically not pleasing and not the safest solution an additional small electrical box was bought. Because the wiring in the box turned out very pretty, we decided to call it “BERTA”. Figure 1.7

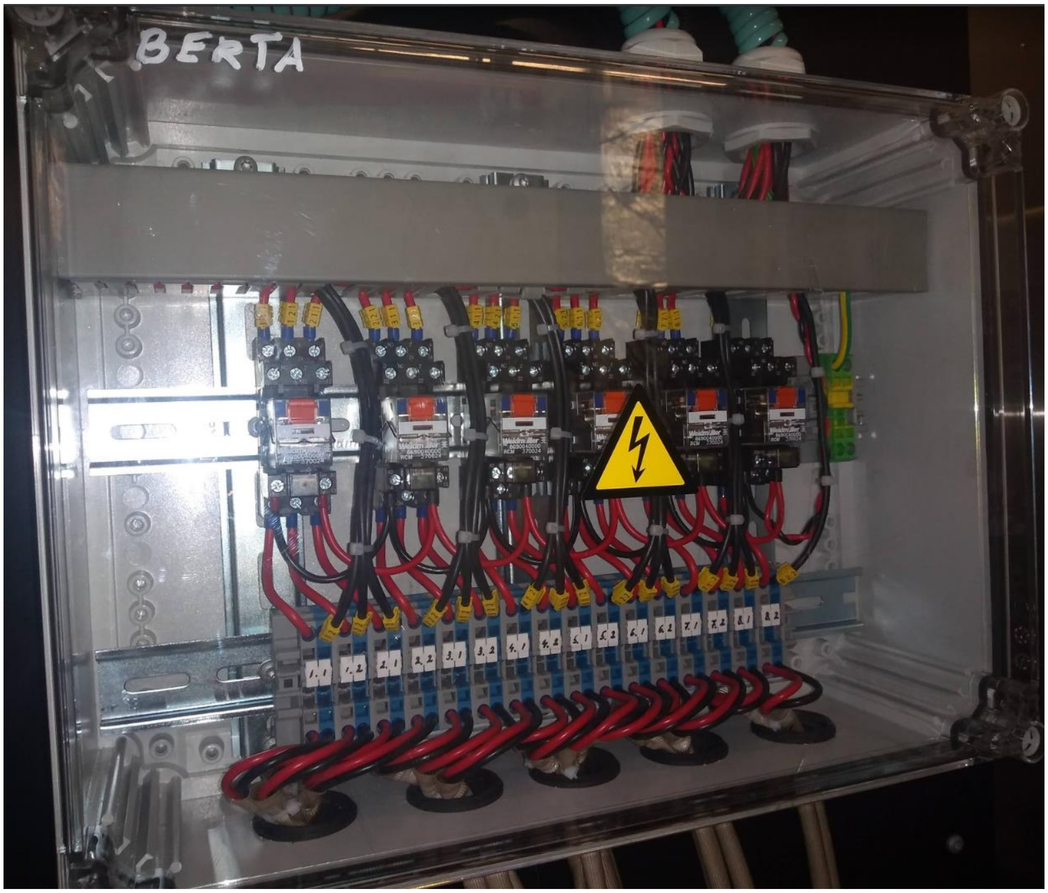


Figure 1.7 BERTA fan supply board load relays

Unfortunately, pretty has little meaning and the relays (Weidmüller RCM370024) chosen into BERTA to connect and more importantly disconnect the 48V DC of 4A current were not up for the job and burned. Figure 1.9. Even though they are rated to switch 250V AC and operate at continues current of 10A, one should remember that disconnecting AC and DC are different things as the author will remember for as long as he may live. Because no automatic control whether fan supply boards were working was implemented it was not noticed that the relays are not able to open under voltage. Because of this fan supply boards that were meant to operate 2min every 24h operated 2min to 12h as long as 1000V DC was supplied to them. We cannot for sure determine how long each EUTs fan supply operated. We can only trace that something was wrong from the 1000V Magna power supply's current feedback that should be 0.5A when fan supply boards are not outputting power and 3.5-4A when they are. In addition to ruining their own test, fan supply boards gave off considerable amount of heat that the climate chamber had to get rid of by cooling its walls and water condensed on the walls causing relative humidity to fluctuate. This only stopped at 18.03.2020 when ambient conditions of the test were rose from 65°C to 80°C

causing fans supply boards internal NTC to stop them from operating witch was also not relay to control board because of the controlling method. This means that for the first part of the test they were almost constantly running and for the second part they were not running at all. Figure 1.9 Broken fan supply board load relay [4]
For the second test voltage measurements were added and RCM370024 relays were switched to much stronger AF09-30-10-11 contactors, that are still not strong enough according to their datasheet. BERTA is not as pretty as she used to be but maybe she will cause less trouble. Figure 1.8

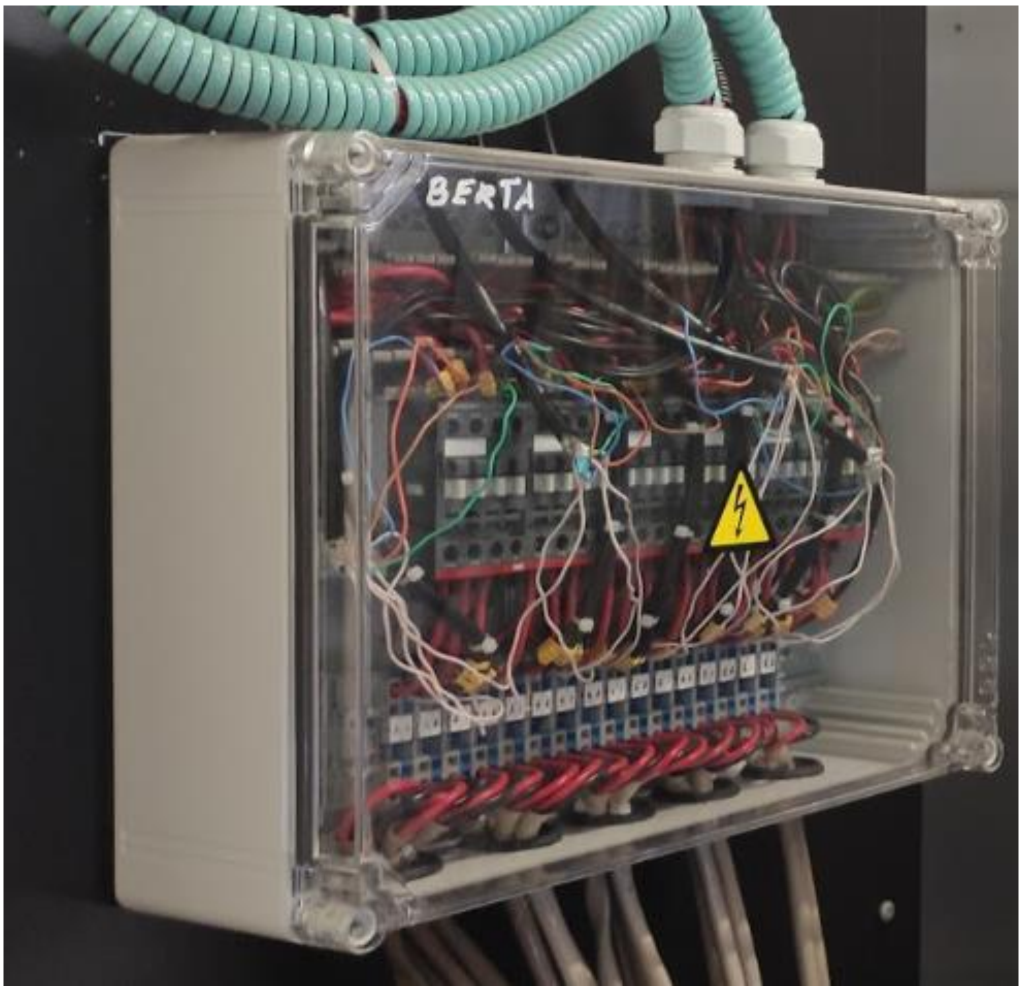


Figure 1.8 BERTA upgraded with AF09-30-10-11 contactors and voltage measurements

Figure 1.9 Broken fan supply board load relay

2 TESTING

With Ocala tester we are trying to accelerate wear out failures using thermal, electrical, and chemical stressors to find weakness in components, manufacturing, or design. Figure 2.1. As this is the first kind of this test done in Estonia’s testing center, we are not sure what results to expect. Therefore, test is being run until failure. In the future this initial test should help us more accurately set ambient conditions and lifetime expectations.

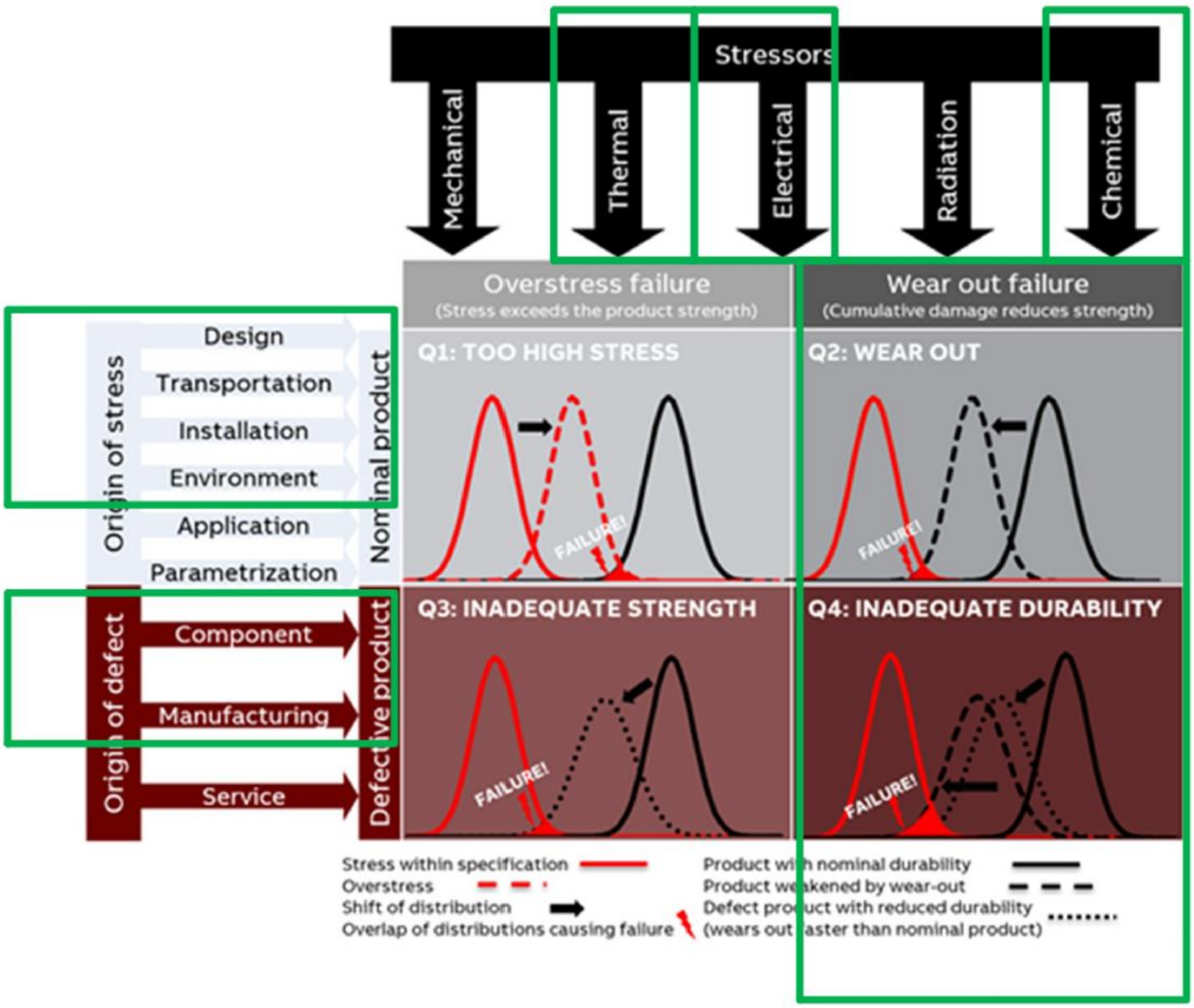


Figure 2.1 Stressor matrix

Described in PCBA ORT Specification. [3]

2.1 Ambient condition

Most popular ALT test done with temperature and humidity is done with 85°C and 85% RH as this is an old standard. Recent studies/experimentation have shown that higher than 50°C ambient temperature can even reduce dendrite growth and corrosion. Studies have also shown that the relative humidity levels are critical, with orders of magnitude changes in time to failure with relatively minor changes in relative humidity. As dendrite growth was something, we were interested in initial ambient conditions were set 50°C and 93% RH. Initial test had 2 parts - soak test and application test. [1] [Internal document: "Humidity Test Specification"]

2.2 Test Cycles

Soak test was a test to simulate and accelerate the drive starting from end control in the production to customer starting the drive. Test was quite straight forward - turn the drive on, make sure everything is working, turn the drive off, let it soak in 50°C, 93% RH for 30 days. Turn the drive back on and see if everything is still working and a quick visual inspection. Initial parameters for the test:

- Temperature: 50°C
- RH: 93%
- Ttest: 720h
- Trun: 1h, Running time to verify everything is operating correctly before and after tests

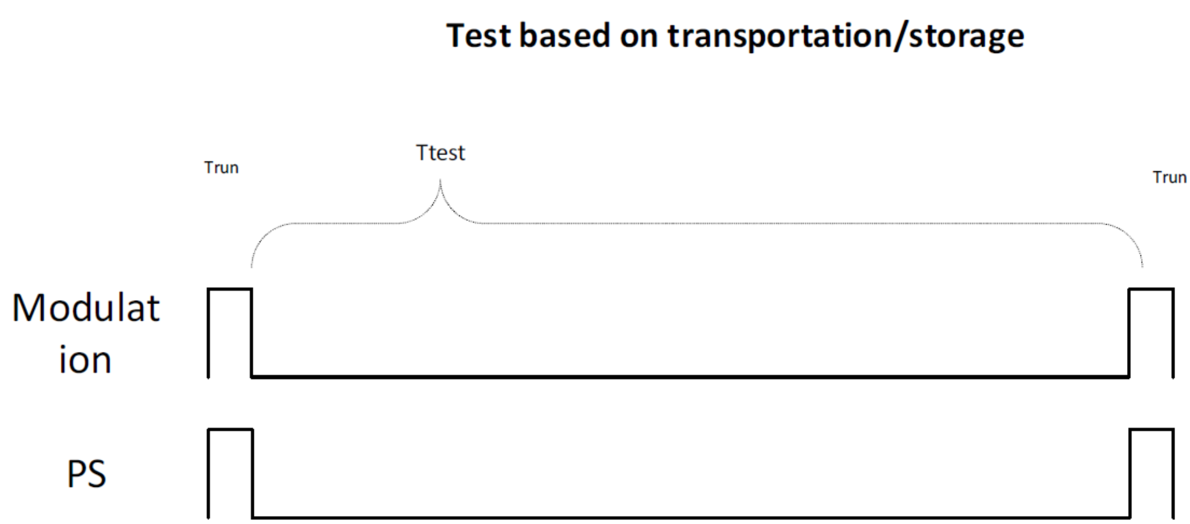


Figure 2.2 SOAK test profile [3]

Soak test started on 25.05.2019, ended 26.06.2019 drives worked and showed no signs of damage - nothing interesting happened.
At this point the tester was still lacking data saving that had previously been done through a “LAC” program in other testers that almost no one understood or was able to fix or change if needed and it had given us considerable amount of problems and downtime. As another department was experimenting with getting rid of this link in the chain and letting the PLC write directly to the local MS SQL database, we did not want to start using LAC in Ocala. More about data saving in data saving chapter. As planned after soak we were supposed to start application test but there was no database and test was stopped to wait for the data recording solution that was promised to us already in May.

Initial parameters for the application test:

- Temperature: 50°C
- RH: 93%
- Tcycle: 24h
- Ton: 12h
- Tmod: 60s
- T230: 15min, power supply board only supplied with 230V to ensure 230V input works
- T1000: 15min, power supply board only supplied with 1000V to ensure 1000V input works
- Voltage UDC: 1050V, maximum without warning/fault
- Voltage U24: 28V, increase stress
- Voltage U230: 250V, transformer

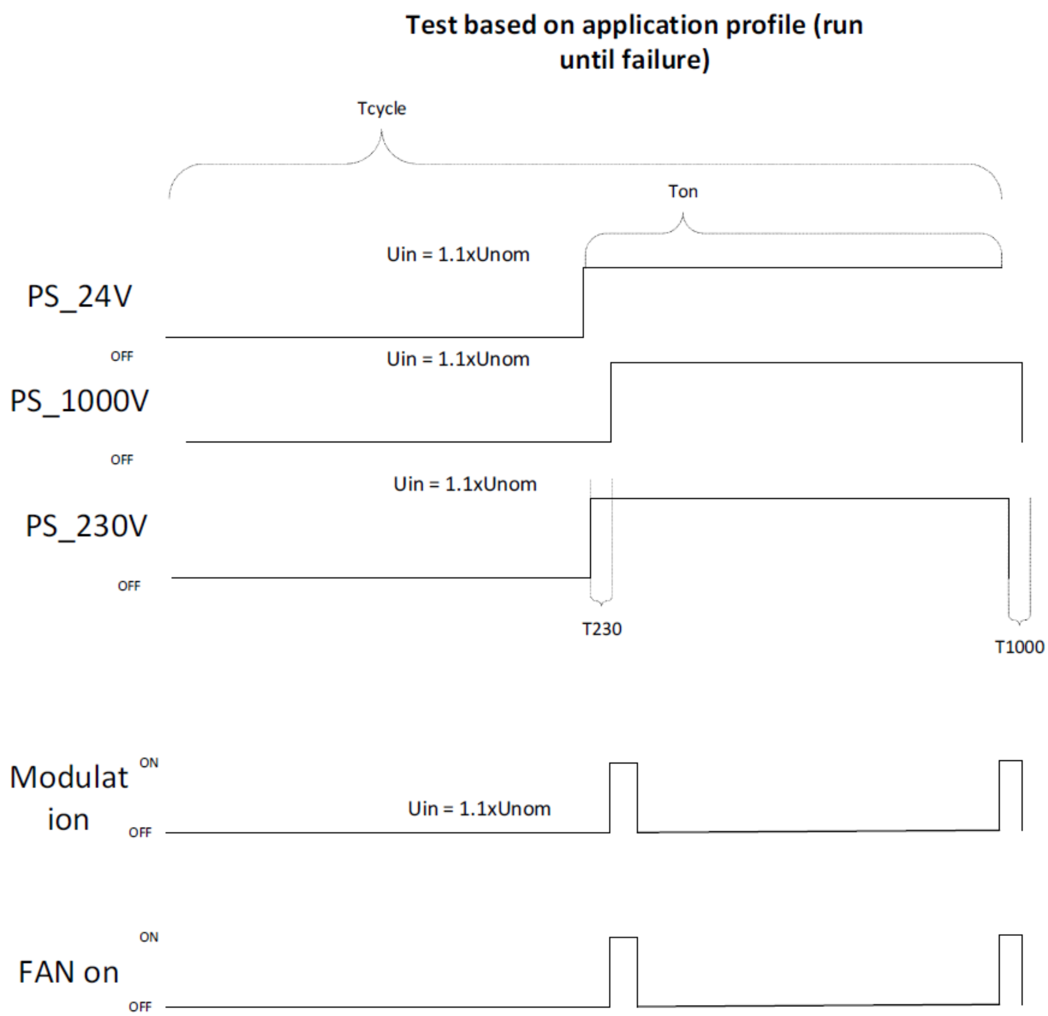


Figure 2.3 Application test profile

Application test started at 5.08.2019 but looking back I would say that commissioning continued then. Because the data saving solution did not seem to arrive any time soon, we stared application test without it and the first 34 cycles were not recorded to database. We could still record parameters from PLC to text files, but this could not be a permanent solution. We got our own measurements system running in Ocala at 25.09.2019. We were also struggling to eliminate climate chambers fluctuating humidity levels. Figure A4.1 humidity fluctuation. It now seems that there were three reason for this kind of behavior: too much heat produced during Ton, too aggressive cooling, and bad sealing of wiring holes. We had some trouble with accessing JUMOs PID control parameters to limit cooling and heating power of the chamber because they were password protected.

We managed to crack the mighty password because it turned out to be “1”. At the start of testing, difference between humidity minimum and maximum during 24h was about 10-13%, by the end of the test we got it to 2-3%.

Test went on with 50°C, 93% RH for over 4 months quite smoothly only stopping because of running out of water was soon as I went on vacation, still nothing interesting happened to EUTs (nothing broke). We had other troubles with our test setup like MUX and PLC not wanting to communicate and being stuck on last value read due to bad coding. Safety relay tripping for unknown reason we believed to be caused by cooling fan for some mystical reason. Turned out to be the way safety relays channels were connected. We ran out of water another time causing the chamber to overheat and stop. Because it was believed that temperatures over 50°C might kill off dendrite growth and our PCBAs were running at about 65°C with normal 50°C ambient. These 90°C and 80°C spikes could have ruined the test. It was decided to increase temperature from 50°C to 65°C. This small 15°C rise in temperature increased the water consumption of the chamber remarkably. Figures A4.4

Nothing still broke during the month and it was decided to go to the golden standard of 85°C/85%RH test. Because the chamber can only control humidity up to 80°C we continued with 80°C/85%RH. After a few months testing power supply boards finally started to break. Test was stopped at 27.08.2020 15:16:38 after being run for 460 days. Figures A4.1, A 4.2, A 4.3

Serial_number	EUT_place	Cycle	FaultOccurrence_Cycle_step	FaultDisappearance_Cycle_step	Active_faults	HEX	FaultStart
EUT_06	6	251	5	1	22162	0x5692	7.06.2020 16:05
EUT_04	4	261	5	1	22145	0x5681	16.06.2020 23:36
EUT_02	2	261	5	1	22162	0x5692	16.06.2020 23:36
EUT_01	1	292	5	1	22145	0x5681	27.07.2020 1:45
EUT_05	5	307	5	1	22145	0x5681	11.08.2020 1:45

Figure 2.4 Faults table

3 PROGRAMMING AND DATA SAVING

3.1 AC500 programming

General diagram of how tester PLC should be programmed and what it must do. AC500 PLC is configured in Automation builder and programmed in Codesys that is built into Automation Builder. Figure A6.1 General tester programming structure

Variable languages possible in Codesys

- LD - Ladder Diagram
- IL - Instruction List
- FBD – Function Block Diagram
- SFC – Sequential Function Chart
- ST - Structured text
- CFC - Continuous Function Chart

Ocala PLC is programmed in ST and FBD.

Controlling and monitoring EUTs is done through PROFIBUS with cyclic communication. Control word and frequency reference values are sent to all drives to simulate running a motor. This is called "modulation" in this test. In normal application the drives job would be to create a sine output to drive a motor. This is done with pulse width modulation of DC voltage. In Ocala tester however IGBTs and power electronics are not tested and special PCBAs were designed to simulate an IGBTs gate load on the adapter boards. Drive is modulating only twice a cycle at the beginning and end of the power on part of the cycle for 1 minute to make sure they work. Rest of the cycle control word is still being sent but with the missing RUN bit. 11 process data values are read from the drive plus some current and voltage readings from MUX.

Table 3.1 List of parameters saved for each EUT

1	Status word
2	INT board temperature
3	PU power supply temperature
4	Fan on-time counter
5	Inverter temperature
6	Warning word 1
7	Tripping fault
8	Active warning
9	Control board temperature
10	DC voltage
11	Switching frequency
12	Power supply output current
13	Power supply output voltage
14	Control board input current
15	Control board input voltage

This data is saved to local MS SQL database with variable time intervals set by the user in the rightmost column in figure A5.2 cycle settings table. PLC has no control over MUX (not programmed so). At the start of the test, configuration is sent to the MUX describing how data is needed and after that the internal memory of the device is being asked continuously for read channel values. PLC scales the values before displaying them to user and saving them to local database. Scaling values are set by user. This turned useful and necessary as currents measured are under 1A and recording them in mA was needed. Climate chamber itself is also being controlled through PROFIBUS. Only temperature and humidity setpoints are wrote and their actual value read and recorded. JUMO can control the two independently. After numerous overheating due to running out of distilled water an additional control method of cutting control signal to chamber was added and bit of code wrote to disconnect it if temperature in the chamber exceeds setpoint by user set limit. Digital alarm outputs for coolant pressure inside climate chamber are monitored. There are many digital inputs and outputs and it is not reasonable to save them all as a separate Parameter. Do reduce database size and noise they are grouped into tester status word (SW) and tester alarm word (AW). Figure 3.1. Status and alarm words should seldomly come into use, but it was already confirmed that control cabinet fan had not been working for some time causing problems for the PC inside thanks to status word bit 1. Mostly they should help the engineer in tester fault tracing.

Table 3.2 List of parameters saved for the tester

1	Jumo humidity sensor
2	Jumo temperature Sensor
3	PT100 TC High
4	PT100 TC Mid
5	PT100 TC Low
6	PT100 CC
7	PT100 LC
8	TC input coolant pressure
9	TC output coolant pressure
10	Magna voltage SetPoint
11	Magna voltage Feedback
12	Magna current Feedback
13	Chiller input coolant temperature
14	Chiller output coolant temperature
15	AW
16	SW

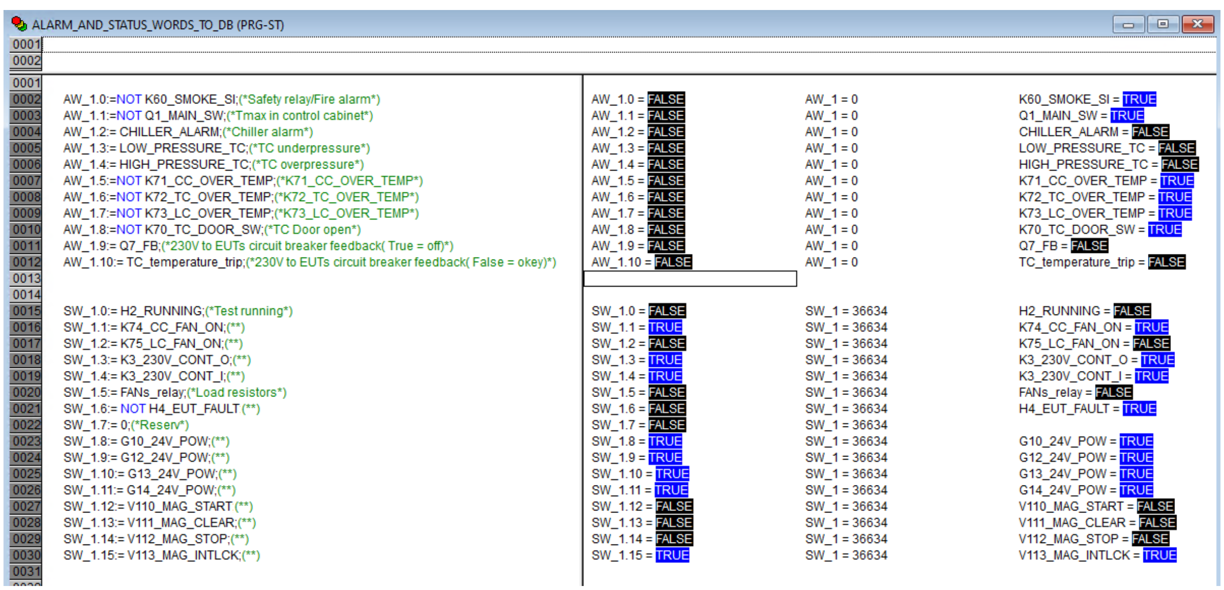


Figure 3.1 Tester status and alarm word

3.2 Data saving

Starting from using AC500 MSSQL Library for writing into local database to redesigning calculations on data to be done in server (still problems with it after 2 years).
Mostly using the MSSQL_execute function from AC500 MSSQL Library that itself was only created in 2015 and major bug fixes done in 2017. This means there are little to no information about it found in the internet and many e-mails had to be exchanged with AC500 support team.
“LAC” was the program we had before to get data from PLC to server. LAC used OPC server and it worked most of the time somehow. But it crashed a lot and was only supported by a guy in Riga who sometimes had time for our problems - a different way was needed. Another team in the company had started to implement MSSQL_AC500_V24.lib on one of the testers and solution was promised to us as well. Unfortunately, their efforts mounted to not much. PLC was constantly crashing and the whole code was unreadable not to even talk about a universal solution. Ocala was our newest and cleanest tester and in need of data saving protocol, testing begun. One of the problems in both previous solutions in my mind was that PLC had to also save minimum and maximus values and create the fault log. Because all the needed info was already present in the measurements table. This was something that could be calculated by the server. It also allowed to change only one table in case of user or machine mistakes instead of 3. There were also cases when users had to copy data from server to their computers excel table to do some simple calculations. This took about an hour a week per tester and information sharing was manual. The new logic would have PLC only record measurements data. Raw measurements data would be pulled from global server and calculation be performed on it constantly and easily be shared with whomever needed it. There were some problems, however.
Too much data – calculation times are too long. End user tables inaccessible when pulling and calculating new rows. Still quite annoying to fix data.

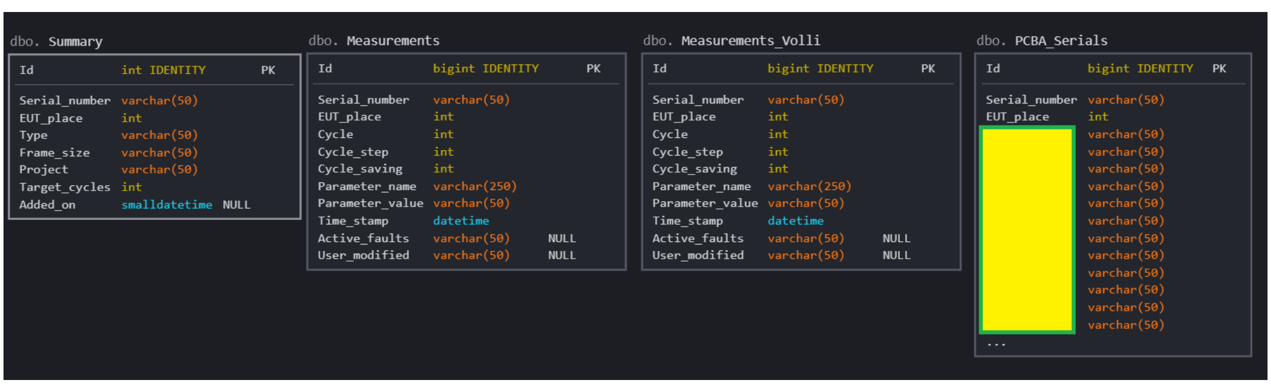


Figure 3.1 Local SQL tables out of which only Summary and Measurements are truly needed.

PCBA_serials table is just a table where to store Serial numbers of EUT PCBAs and takes part in no further logic. Measurements_Volli is a duplicate structure of Measurements and is used for testing. Measurements_Volli data is not pulled to server nor shared. Measurements_Volli table evolved into Measurements_commissioning table into all testers to be used as testing and setting up tests so as not to record invalid data into main tables.

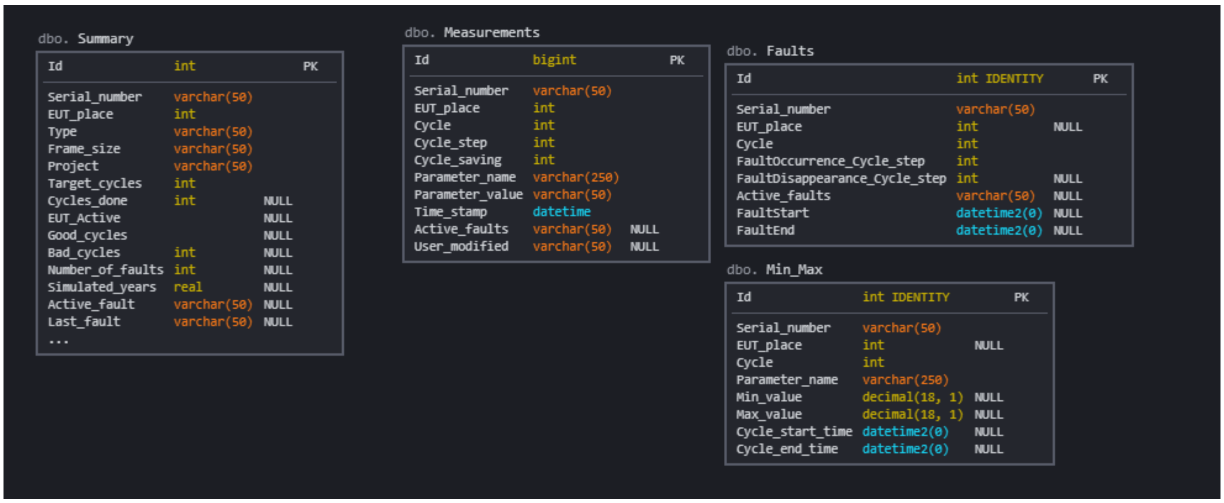


Figure 3.2 Server-side end user sees schemas dbo-s 4 tables

- Measurements which is only a copy of the measurements table from the local server
- Min_max table where server has calculated minimum and maximus values for each cycle
- Faults table where server has calculated the fault log
- Summary

It gets a bit more complicated on the server side to pull and calculate the data. Because there are up to hundreds of millions of lines of data in each tester it not practical to copy it all and do all the calculations for all the data each time local data is pulled into server. For the initial test in Ocala 47 million lines were recorded. To cut time and server resource different shortcuts are implemented and 4 schemas created to pull and calculate the data:

- Raw – Before each data pull, schema tables are truncated, raw data from local database is pulled into, Measurements, Summary
- Stage – tables that are used to aid calculations
- Dev – additional aid tables
- Dbo – final tables seen by users

Local measurements table data is first pulled into [Raw].[Measurements]. Only chosen amount of data is pulled each time. For hourly pulls past 48h data is chosen. From [Raw].[Measurements] in server [stage].[Measurements] and [dbo.][Measurements] are updated with a MERGE command.

For minimum and maximum values calculations [Stage].[Min_Max] is first truncated and then filled with stages of grouping and sorting the values. After [Stage].[Min_Max] is calculated, [dbo].[Min_Max] is updated with merging.

Faults table. We had most troubles with faults table and still do. We wanted to get 8 columns + Id:

```
 [Id]
,[Serial_number]
,[EUT_place]
,[Cycle]
,[FaultOccurrence_Cycle_step]
,[FaultDisappearance_Cycle_step]
,[Active_faults]
,[FaultStart]
,[FaultEnd]
```

Id is the primary key and self-incrementing, no problems. Serial_number is the base of calculation or owner of data. EUT_place is actually nonrelevant info and could be looked up from Summary table, but since end users will be using the dbo directly without any user interface it was added for ease of use and could be useful if EUTs swapped places during testing, also could come handy if you notice that different EUTs keep failing to same faults in the same test place. Might be that something is wrong with the wiring of the test setup. Most relevant is the moment when the fault appeared, after how much testing (cycles) it failed and what was going on at the time of fault in the test cycle (cycle_step).

This information is obtainable when comparing Active_faults from the previous recording and the next recording. If on the previous line Active_faults = 0 and on the next line it is something different than 0 then this must be the moment when the fault occurred. For comparing lines there is a LAG function in SQL that allows access to previous row without joining tables, which we tried as well. LAG function is slow, and it was needed to speed it up somehow or find another solution. What we did was look for only 1 Parameter_name that would be present in all tester. We came close enough with "Status word". As parameter names were entered into PLC user interfaces by users, we saw 6 different ways you can spell "Status word". So even after it was changed the same in all testers, we knew that making calculation logic dependent on some name users can change is a slippery slope where I have slipped many times already. It was also requested that fault end time would be recorded. Here lies the real problem. There is a logic error where server inserts the found fault into [Stage].[Faults] only after it has ended, because it does not know otherwise as what to mark FaultDisappearance_Cycle_step and FaultEnd. There snowballs another error of user not being notified of the fault because notification procedure is called out before [Stage].[Faults] is merged into [dbo].[Faults]. Notification procedure sends out notification mail if there are new faults in Stage that are not yet in dbo. But if the fault has not ended it never reaches [Stage].[Faults]. It also means that if an EUT dies and is never put back to work its fault never ends and its final, often most crucial fault, is never recorded to database.

```
INSERT INTO [Ocala_2020].[Stage].[Faults]
select A.Serial_number, A.EUT_place, A.Cycle, A.Cycle_step as FaultOccurrence_Cycle_step,
MIN(B.Cycle_step) as FaultDisappearance_Cycle_step,
A.Active_faults, A.Time_stamp as FaultStart, MIN(B.Time_stamp) as FaultEnd
from [Stage].[FaultsSourceData] A, [Stage].[FaultsSourceData] B
where A.Active_faults <> A.Previous_fault and A.Active_faults <> '0'
and    B.Active_faults <> B.Previous_fault and B.Active_faults = '0'
and A.Serial_number = B.Serial_number
and A.Time_stamp < B.Time_stamp
group by A.Time_stamp, A.Id, A.Serial_number, A.EUT_place, A.Active_faults, A.Cycle, A.Cycle_step
ORDER BY A.Time_stamp ASC


EXEC [dbo].[Ocala_New_Fault_Notification]
```

Figure 3.3 Faults saving problem

	Id	Serial_number	EUT_place	Cycle	FaultOccurrence_Cycle_step	FaultDisappearance_Cycle_step	Active_faults	FaultStart	FaultEnd
1	1	EUT_01	1	100	1	1	25766.0	2019-11-29 08:34:42	2019-11-29 09:45:28
2	2	EUT_04	4	116	4	1	20624.0	2019-12-15 16:16:46	2019-12-16 08:34:28
3	3	EUT_03	3	116	4	1	20624.0	2019-12-15 16:20:46	2019-12-16 08:34:28
4	4	EUT_06	6	116	4	1	20624.0	2019-12-15 16:28:46	2019-12-16 08:34:28
5	5	EUT_08	8	116	4	1	20624.0	2019-12-15 16:31:47	2019-12-16 08:34:29

Figure 3.4 Faults table.dbo

Summary table is a combination of user inserted info about EUT and some summarized data about how the EUT survived or is surviving the test. Most interesting would be Bad_cycles and Simulated_lifetime columns. Bad_cycles keep track of cycles where EUT did not perform as expected. Mostly it checks if any output was generated during a cycle as no direct fault code might not be generated in these kinds of “bad cycles” they might slip by unnoticed. Simulated_lifetime calculates how much each test cycle aged the EUT and then sums them up. Calculations behind it are different because accelerating factor in the test is calculated against some mission profile in what environment and with what load the drive is expected to operate for the next 10-30 years. Visual dashboard was created in Microsoft Power BI to display how testers and EUTs are doing from Summary and Faults table.

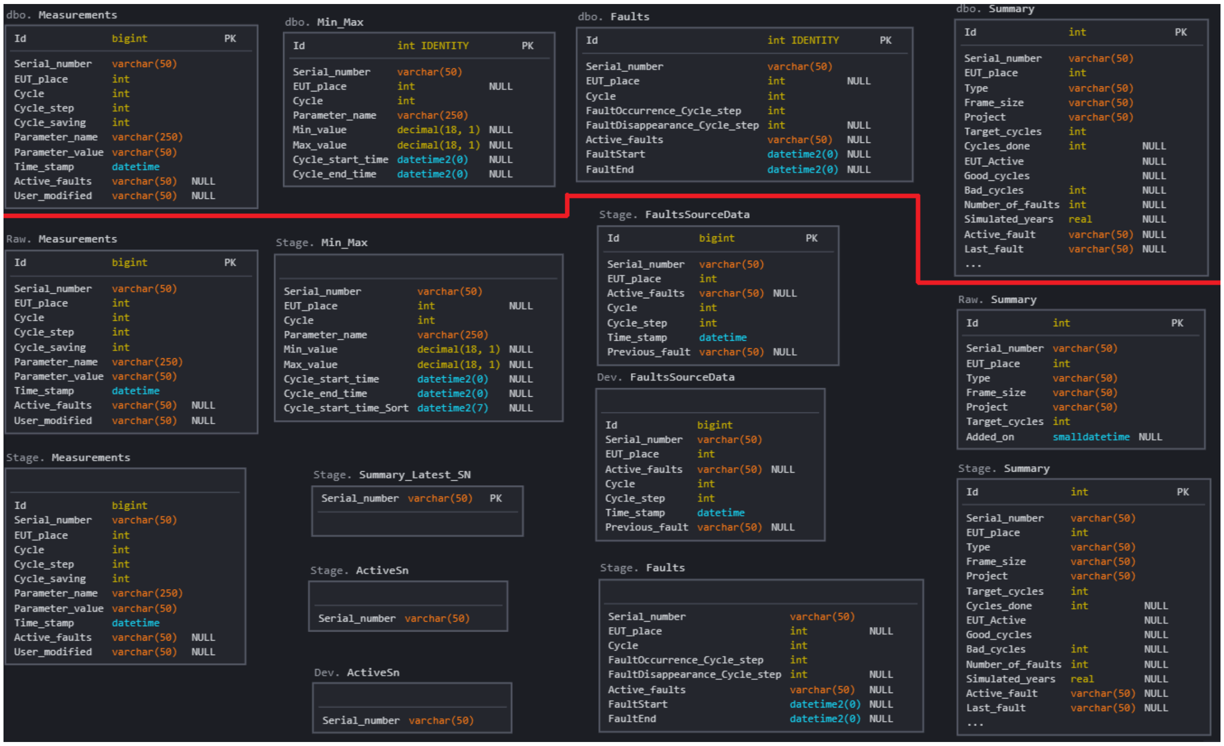


Figure 3.5 All server tables

4 TEST RESULTS

Plastic on all the option modules was destroyed but it did not affect the working of the PCBAs enclosed in them. Turns out plastics absorb too much water in humid environments that are over 70°C thus losing their tensile strength and becoming brittle. This corresponds what was seen from the test. During the first 5 months when temperature was kept at 50C and humidity at 93% RH nothing remarkable happened to the plastics but after only 6 weeks in environment of 80°C/85% RH, plastics became brittle and deformed. Although unlikely that drives are installed in such harsh environments, deformation and study of these plastics might imply strong discard of required installment conditions.[6]
Main failure was film capacitors casings being cracked and eventually destroyed. Capacitance loss was measured on many of capacitors. Metallic contact layer escaped and caused short-circuits on the delaminated parts of the circuit.

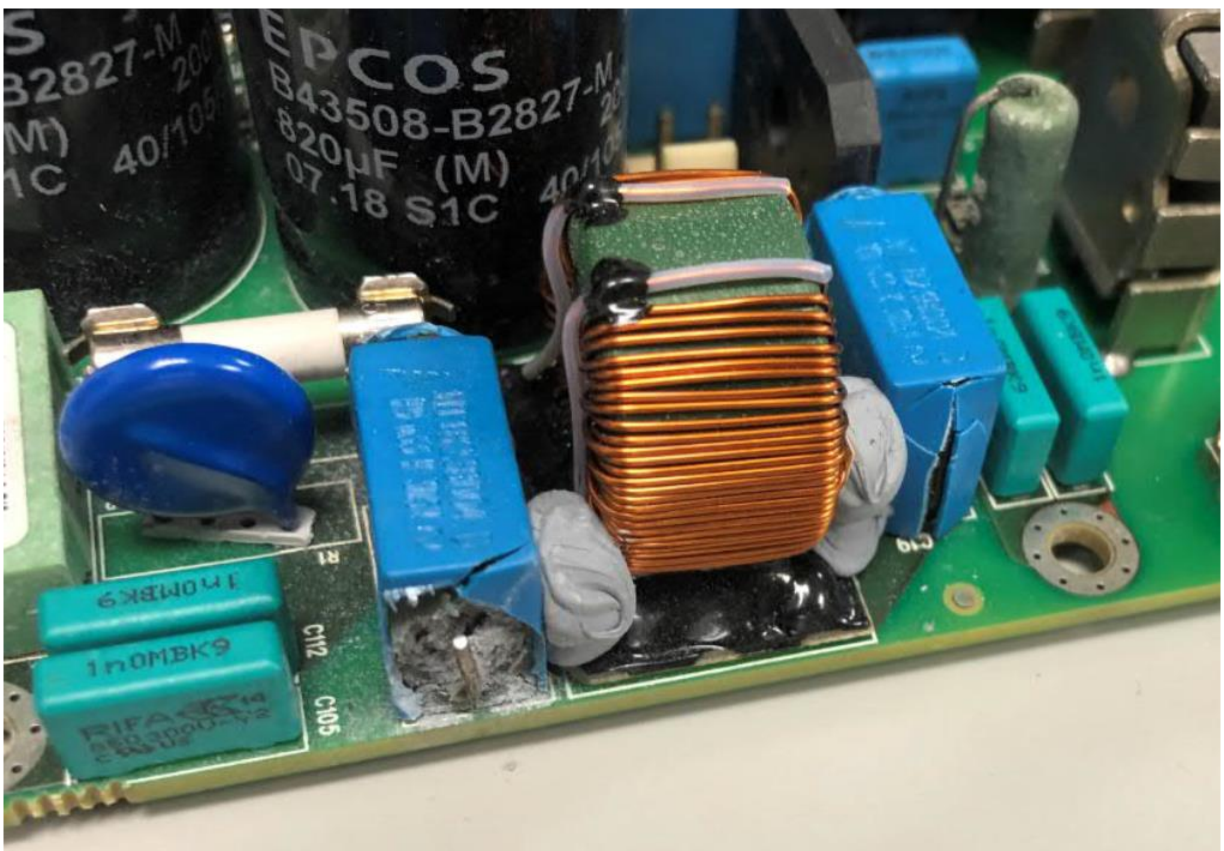


Figure 4.1 Broken capacitors on PCBA

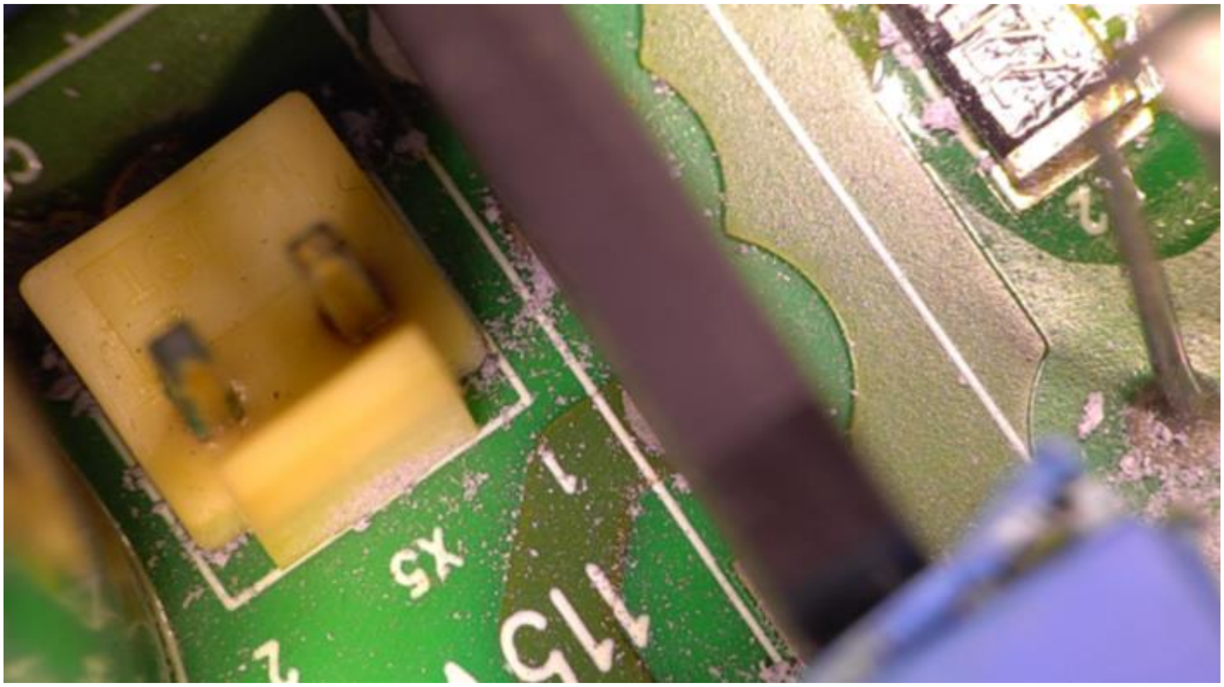


Figure 4.2 Conductive dust from capacitors

Some problems were found in MOSFETs that are responsible for 1000V DC to 24V conversion. But main failure in all EUTs was due to broken film capacitors.

Possible future actions: different type capacitors and better coating of PCBAs.

$$AF = \left(\left(\frac{HUM_{field}}{HUM_{test}}\right)^{-2.66}\right) * e^{\left(\frac{Ea}{K}\right)*\left(\frac{1}{T1}-\frac{1}{T2}\right)} \tag{4.1}$$

where,

AF – Acceleration factor

HUM_{field} - field humidity level, % RH

HUM_{test} - test humidity level, % RH

E*a* - activation energy in electron-volts, eV

K - Boltzmann's constant (8.617385 x10-5 eV/K)

T1 - Field maximum temperature (K)

T2 - Test maximum temperature (K)

[5,6,8]

	Well conditioned electrical room, ideal test conditions				Tropical ambient conditions, minimum measured humidity values in test			
	First part (ambient temperature)	Second Part (ambient temperature)	First part (component temperature	Second Part (component temperature)	First part (ambient temperature)	Second Part (ambient temperature)	First part (component temperature	Second Part (component temperature)
HUM_field	50	50	50	50	85	85	85	85
HUM_test	93	85	93	85	82	85	82	85
AF_hum	5.21	4.10	5.21	4.10	0.91	1.00	0.91	1.00
T_field(C)	30	30	50	50	30	30	50	50
T_test(C)	50	80	70	100	50	80	70	100
AF_temp_ambient	6.65	76.38	5.34	46.97	6.65	76.38	5.34	46.97
AF_total (humidity * temperature)	**34.68**	**313.33**	**27.80**	**192.66**	**6.05**	**76.38**	**4.85**	**46.97**
TIME_in_test (days)	150	100	150	100	150	100	150	100
Accelerated lifetime (years)	14.25	85.84	11.43	52.78	2.49	20.93	1.99	12.87
Accelerated lifetime (years) total	**100.09**		**64.21**		**23.41**		**14.86**	

Figure 4.3 Authors accelerated lifetime calculations, Ea = 0,8eV

Dependent on the ambient conditions of the drive in field and whether ambient or component temperature is taken into calculation the 250 days in test gives the accelerated lifetime of around 15-100 years which gives the author confidence to say the PCBAs did pretty good and failed due to wear-out.

SUMMARY

We set out to build an automatic climate tester, conduct the initial climate test in it and to control the data saving process. It was a project of many firsts. All three goals were never attempted in our testing facility and they were all achieved. Many obstacles were needed to be overcome in the process.

Tester design and build: I would like to think that we thought of almost everything and did not overdo it. Some slip ups naturally happened, and it might not even be practical to try to be flawless with initial proto design. On the downside we hurried going to production and should have waited for all the components to arrive first. Apart from underestimating the amount of connections needed to be done between equipment under test and control cabinet and choosing wrong relays for fan supply boards output, I will say the result is near superb. Especially considered that it was authors first tester build. I am especially happy that at least during the initial test all hardware changes were also updated on the circuit diagrams. Something often forgotten after commissioning is over. Quite many improvements can and should be still done but mostly to the PLC code.

Testing: Could have been better. Most notable failure was failure to see that fan supply boards were not working as meant to therefore ruing their own intended test and causing problems with chamber humidity due to dissipation of heat into testing chamber. Job of filling the tester with distilled water was forgot several times causing unwanted ambient conditions and lost testing time. This temporary system of filling the tester manually has now been fixed with a permanent water supply. Considerable amount of data was lost because of bad programming connecting PLC and Data Acquisition Unit. Mostly programming issue but was still unnoticed by the engineer. As the author was in both roles it is hard to blame anyone else. All serial numbers were saved to database that should also become a standard in PCBA tests.

PLC programming: I had not done it before, and the responsibility became mine out of necessity. Looking back at the code changes and improvements I did I cannot say they are the best or that I would do it similarly again. Even so tester parameters and status word were created that had not been done before. Luckily the code I started with was relatively clean and understandable and I was walked through all the processes and had all the help I asked for. Now that I have become expert at AC500 programming I should really find the time and add some additional safety features so events such as the fan supply fiasco are less likely to repeat.

Database recording: Idea was simple and clean. Record only measurements table and let the server calculate the rest. Implementing AC500 MSSQL Library seemed tricky at first but was quite easy once we got it rolling. Some mistakes with the architecture were made both in PLC and in databases that are still causing problems today. Since all testers in our facility are impacted by any change they should really be thought out and tested before any bigger update on the whole system can be attempted. We were surprised how complicated our simple minimum and maximum and fault calculations got when we tried to implement them in the server just not in our local computers.
Most important fix still lacking in data saving system is in fault calculation so that faults are recorded as soon as they appear, not only when they disappear.

Test results were satisfying: Unfortunately, no PCBA faults were found yet but a few weak points were found and in next tests they can be focused on earlier. Wear-out failures in 5 out of 8 PCBAs in relatively narrow timespan at least confirms that the stressors are working, and more precise lifetime calculations can be made in the future.

In conclusion all goals were achieved well, but in the light of lessons learned many improvements can still be done to Ocala and to future testers either based on Ocala or not. I am sure that Ocala will be ageing and braking PCBAs for years to come finding manufacturing, component, and design flaws.

KOKKUVÕTE

Me seadsime eesmärgiks ehitada automaatne kliimatester, teostada esmane katse ja saavutada kontroll info salvestamise üle. See oli paljude esimeste projekt. Ühtegi kolmest eesmärgist ei ole oldud meie testimiskeskuses varem üritatud ja nad kõik said saavutatud. Selleks tuli ületada hulgaliselt takistusi.

Testeri disain ja ehitus: Ma tahaksin arvata, et me mõtlesime peaaegu kõige peale ja ei teinud liiga palju. Loomulikult oli möödaminekuid ja mõtlematusi, aga ma arvan, et esmase prototüübi disainimisel ei peagi täiuslikkust jahtima. Miinuspoolelt me kiirustasime tootmisesse minekuga ja oleksime pidanud ootama kuni kõik komponendid kohale jõuavad. Väljaarvatud ühenduste ja juhtmete hulga alahindamise ja valede ventilaatoritoiteplaadi väljundi releede valikule, sai tulemus suurepärane. Eriti arvestades, et tegemist oli autori esimese testeri ehitusega. Ma olen eriti rahul, et kogu ehitamise ja esmase testimise käigus sai riistvara täiendused ja muudatused ka skeemidele märgitud. Midagi, mis tihti pärast esmast käivitust ununeb. Jätkuvalt saab ja peaks testrile tegema palju täiendusi, aga üldiselt kontrolleri programmeerimisel.

Esmane katse: Oleks võinud minna paremini. Kõige märkimisväärsem viga oli ventilaatoritoiteplaadi mitte korrektne töötamine, mis rikkus nende endi katse ja põhjustas probleeme sest nad kiirgasid palju soojust kliimakambrisse. Mitmeid kordi unustati täita testrit destilleeritud veega, mis põhjustas soovimatuid kekskonnatingimusi ja põhjustas testi mitte töötamist. See ajutine süsteem on nüüdseks parandatud pideva vee toitega testimiskeskuses. Arvestatav hulk infot läks kaduma halva programmeerimise tõttu, mis tõttu side kontrolleri ja andmete hankimise masinaga katkes. Suuresti programmeerimise probleem, mis jäi siiski inseneri poolt märkamata. Kuna autor on mõlemas rollis siis on kedagi teist süüdistada raske. Kõik trükkplaatide seerianumbrid salvestati andmebaasi, mis peaks muutuma trükkplaatide testimise standardiks.

Loogikakontrolleri programmeerimine: Ma ei olnud seda varem teinud ja vastutus langes minule vajadusest. Tagasi vaadates koodi muudatustele ja täiendustele ei saa ma öelda, et nad oleks kõige paremad olnud või et ma nii sarnaselt uuesti teeksin. Siiski said tekitatud testeri parameetrid ja staatussõna, midagi mida varajasemates testrites ei olnud olnud. Õnneks kood millega ma alustasin oli üsnagi puhas ja arusaadav ning mind aidati läbi kõigi protsesside ja sain nii palju abi kui ma küsisin. Nüüd kui ma olen muutunud AC500 eksperdiks peaksin ma leidma aega, et lisada turvafunktsioone, et ventilaatori toiteplaadifiasko ei korduks.

Andmebaasi salvestamine: Idee oli puhas ja lihtne. Salvestame ainult mõõtetulemusi ja server arvutab ülejäänu. AC500 MSSQL raamatukogu kasutuselevõtt tundus alguses keeruline, kuid kui me algusega hakkama saime läks edasi lihtsalt. Andmebaasi ja kontrolleri arhitektuuris sai tehtud mõned vead, mis siiani tüli põhjustavad. Kuna muudatused andmebaasi loogikas mõjutavad kõiki meie testreid tuleb iga muudatus enne põhjalikult läbi mõelda ja katsetada, enne kui teda saab kasutusele võtta. Me olime üllatunud kui keeruliseks meie lihtsad miinimumi ja maksimumi tabeli ja vigade tabeli arvutused muutusid võrreldes sellega, mida me endi arvutites proovisime.
Kõige tähtsam parandus on endiselt puudu. Selleks on loogikaviga serveris, kus viga salvestatakse alles tema kustumisel, mitte tekkimisel.

Testitulemused olid rahuldavad: Kahjuks ühtegi trükkplaadi viga ei leitud, kuid mõned nõrgemad kohad avaldusid ja neile saab järgmises testis juba alustades rohkem tähelepanu pöörata. Kulumisvead viiel kahekast katsetavast üsnagi väikeses ajavahemikus kinnitavad, et stressorid töötavad ja tulevikus saab sooritada täpsemaid eluea arvutusi.

Kokkuvõttes said kõik eesmärgid edukalt saavutatud, kuid õpitutu valguses tuleks hulgaliselt täiendusi teha Ocalale ja tulevastele testeritele. Ma olen kindel, et Ocala vanandab ja lõhub trükkplaate veel aastaid leides tootmise, komponentide ja disainivigu.

LIST OF REFERENCES

1. https://www.dfrsolutions.com/hubfs/Resources/services/Qualifying-for-Moisture-Containing-Environments.pdf?t=1503583170559
2. Climate chamber manual [2]
3. ORT specification xxx PCBAs (Internal document)
4. https://catalog.weidmueller.com/catalog/Start.do?localeId=en&ObjectID=8690040000
5. https://www.desolutions.com/blog/2017/09/accelerated-temperature-humidity-testing-using-the-arrhenius-peck-relationship/ [5]
6. MECHANICAL PERFORMANCE OF POLYAMIDES WITH INFLUENCEOF MOISTURE AND TEMPERATURE – ACCURATE EVALUATIONAND BETTER UNDERSTANDING https://www.researchgate.net/publication/284257135_Mechanical_Performance_of_Polyamides_with_Influence_of_Moisture_and_Temperature_-_Accurate_Evaluation_and_Better_Understanding [6]
7. Magna power supply https://magna-power.com/assets/files/manuals/manual_xr_1.3.pdf
8. COMPREHENSIVE MODEL FOR HUMIDITY TESTING CORRELATION by D. S. Peck (1986) http://web.cecs.pdx.edu/~cgshirl/Documents/Supplementary%20Papers/1986%20Peck%20Comprehensive%20Model%20for%20Humidity%20Testing%20Correlation%20IRPS%2004208640.pdf

APPENDICES

Appendix 1 Ocala data and control flow chart

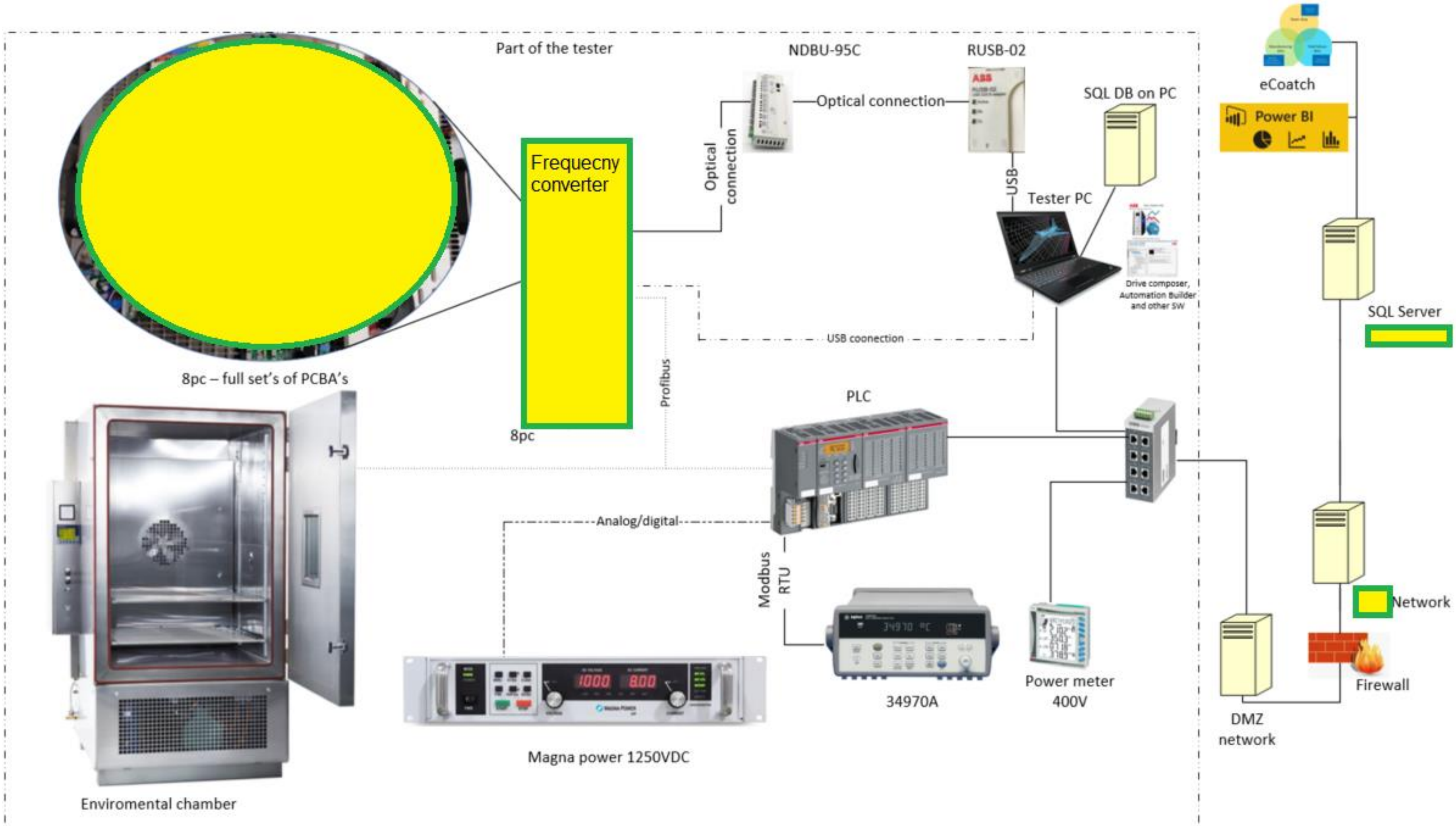


Figure A1.1 Ocala data and control flow chart

Appendix 2 PLC in circuit diagram

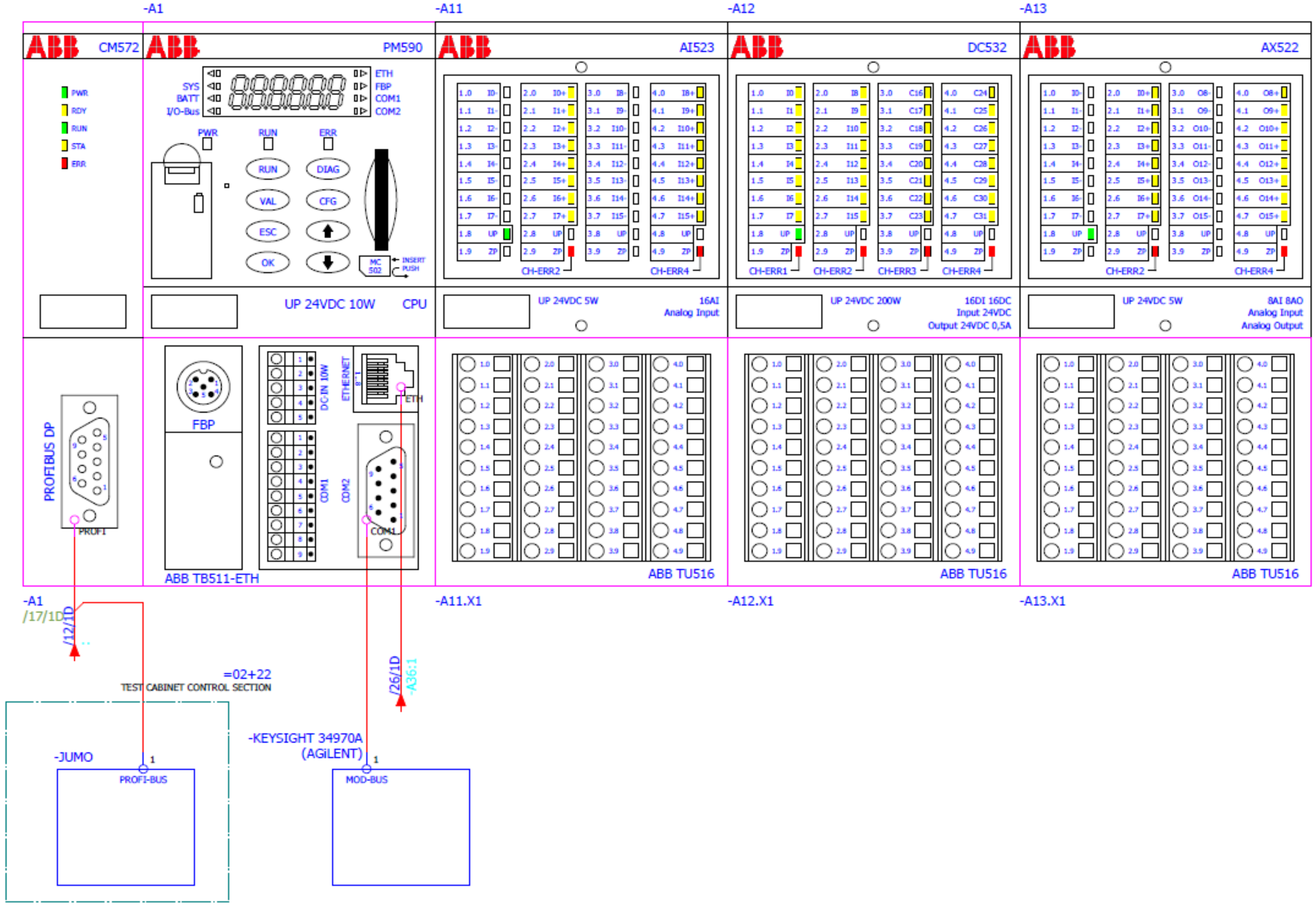


Figure A2.1 PLC in circuit diagram

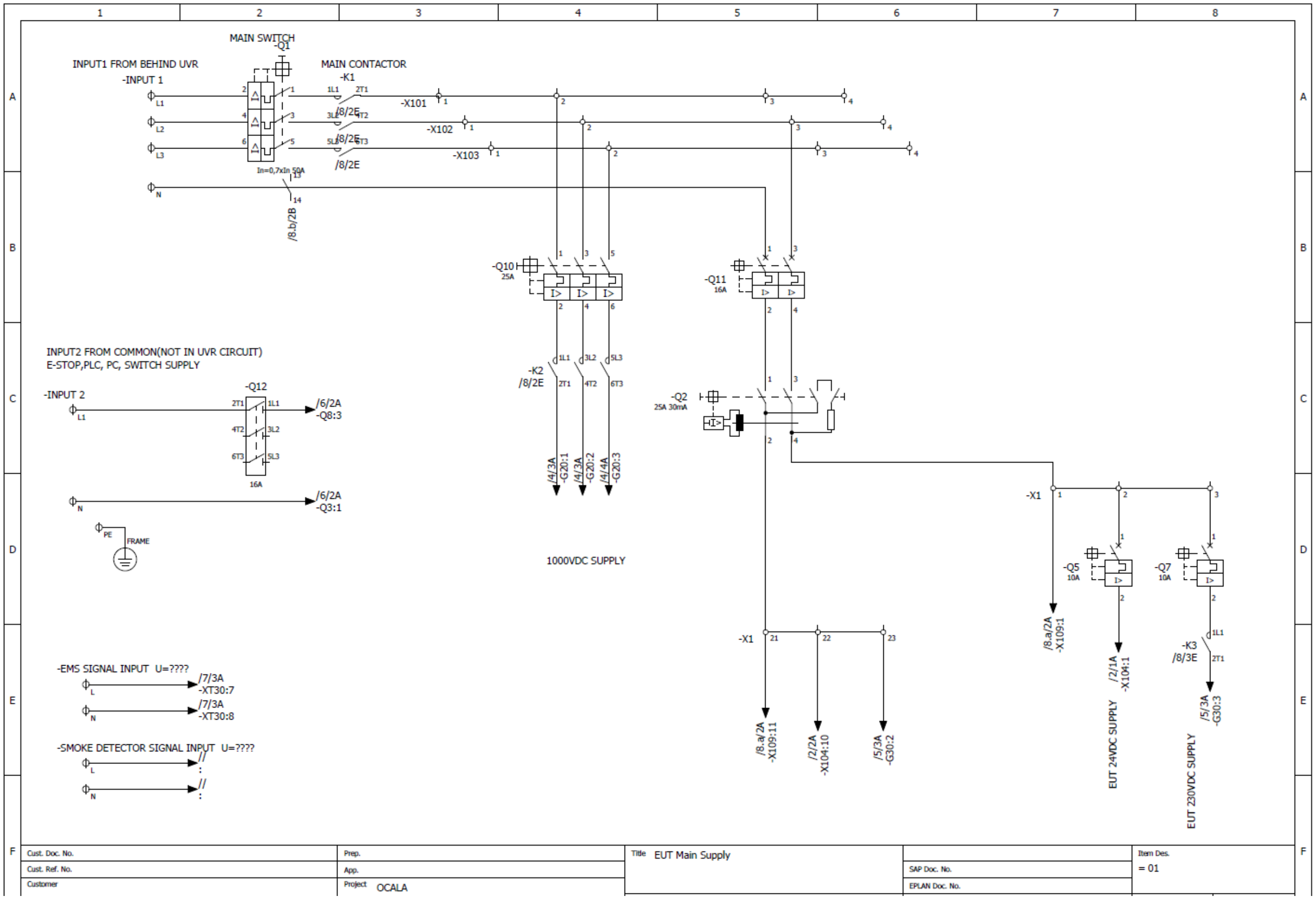


Figure A3.1 First revision of Ocala Main Supply

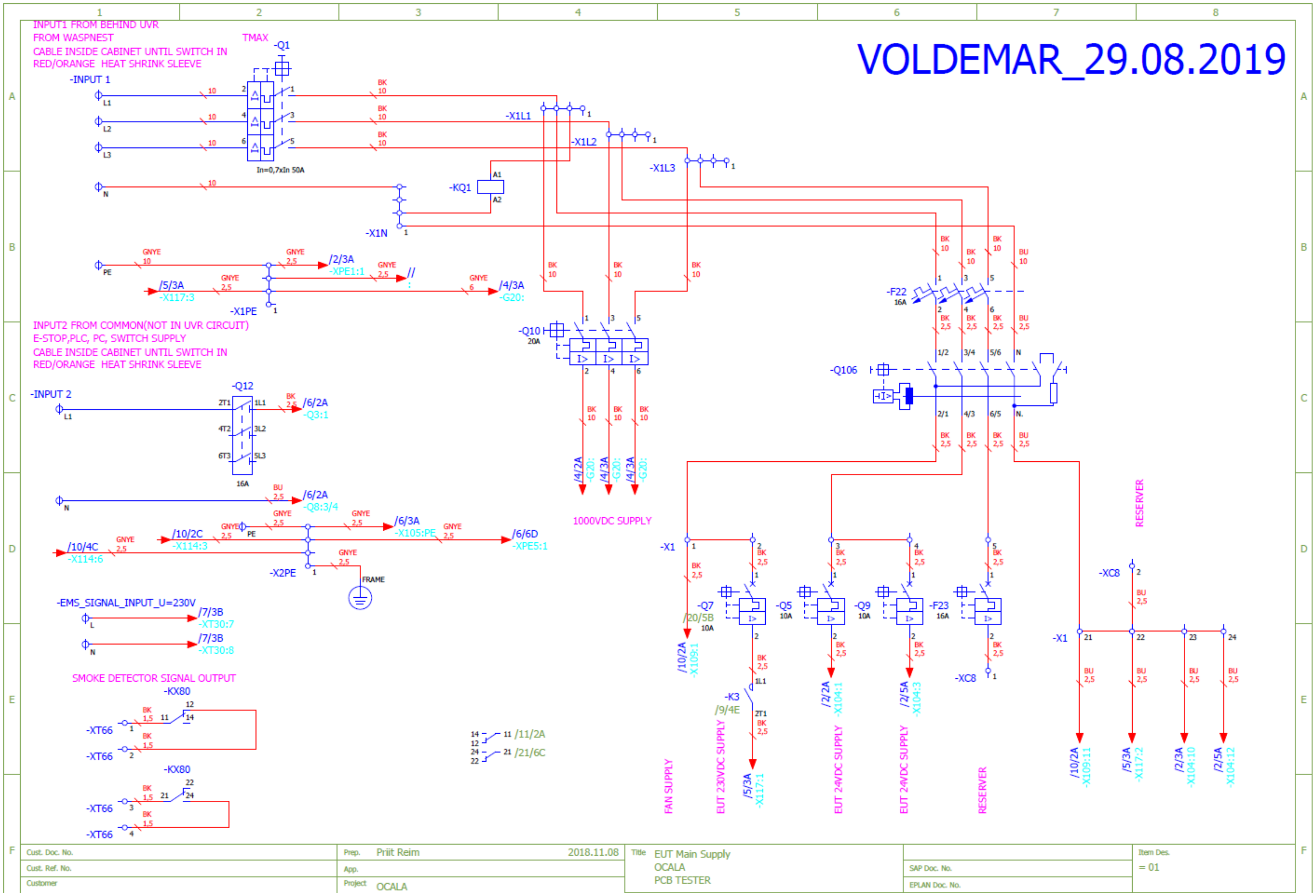


Figure A3.2 Latest revision of Ocala Main Supply

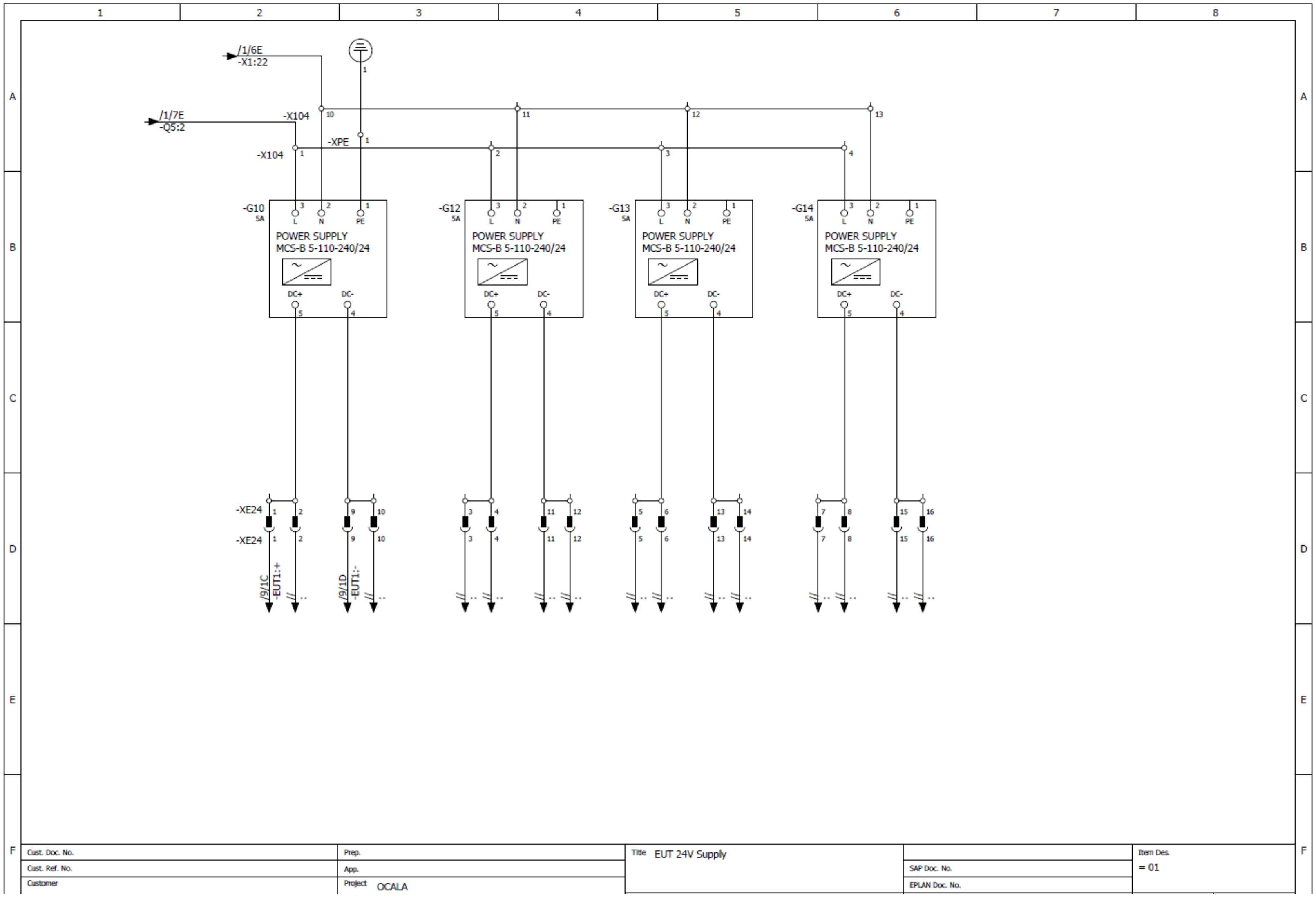


Figure A3.3 First revision of EUT 24V DC supply

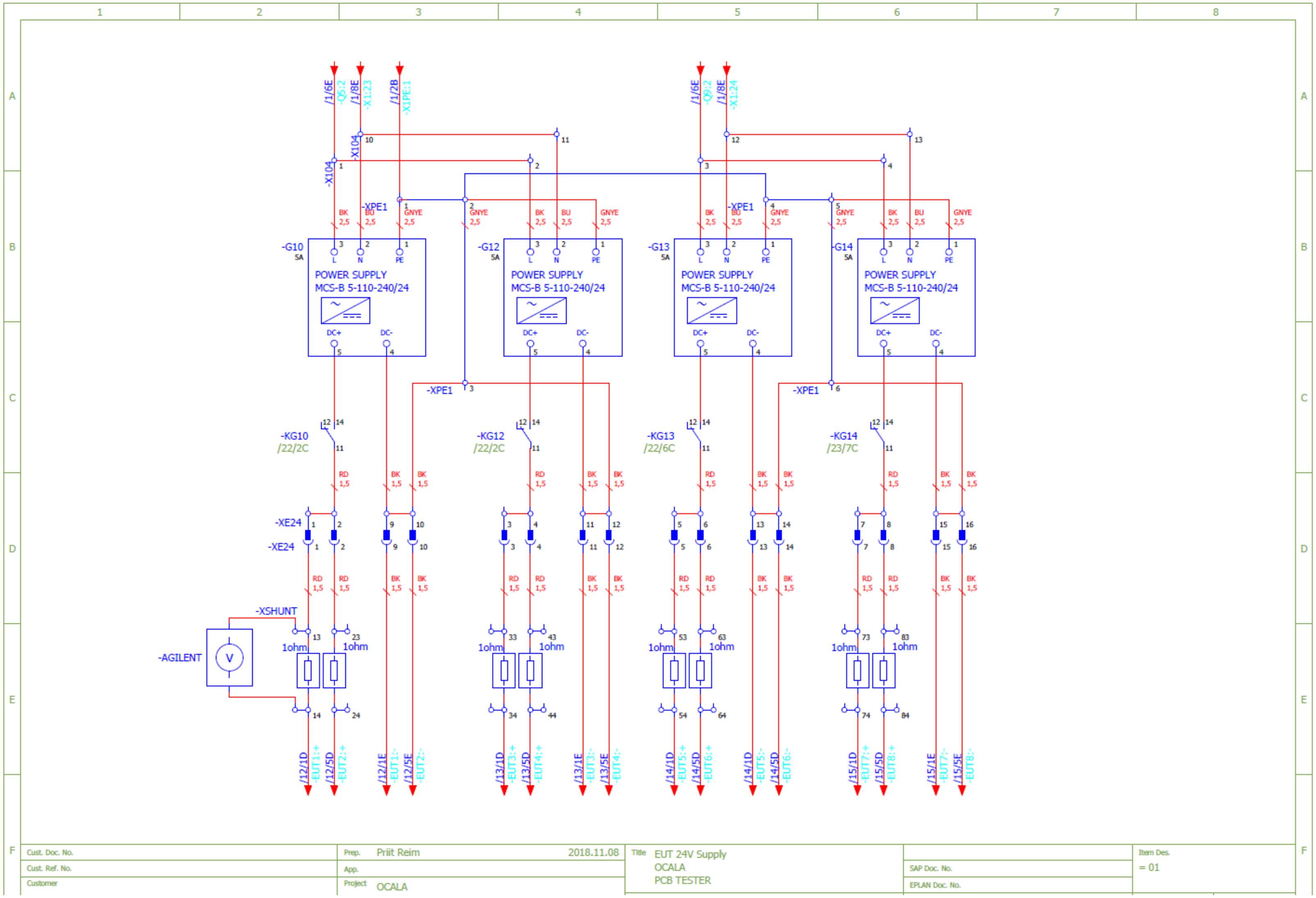


Figure A3.4 Latest revision of EUT 24V DC supply

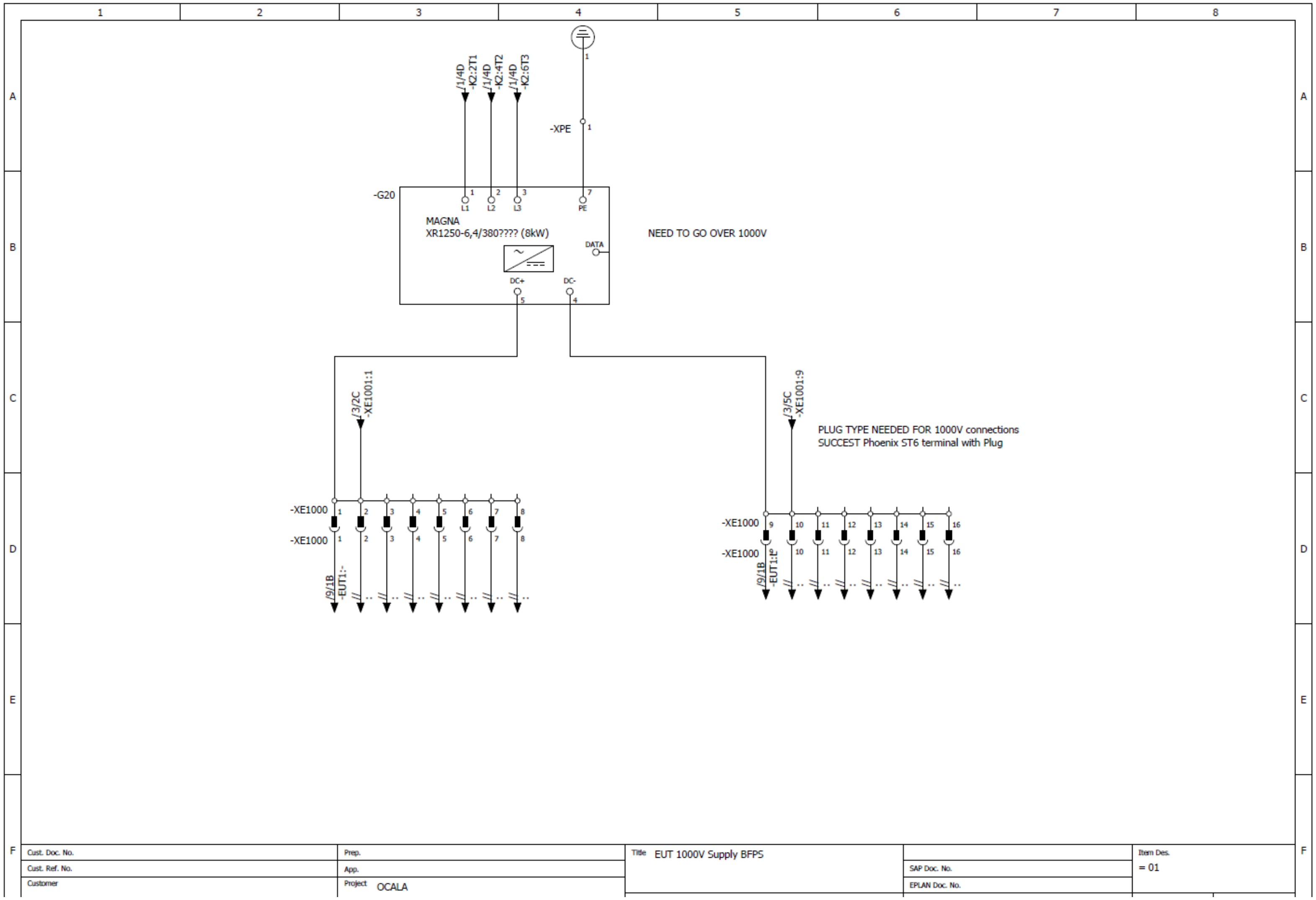


Figure A3.5 First revision of 1000V DC Supply

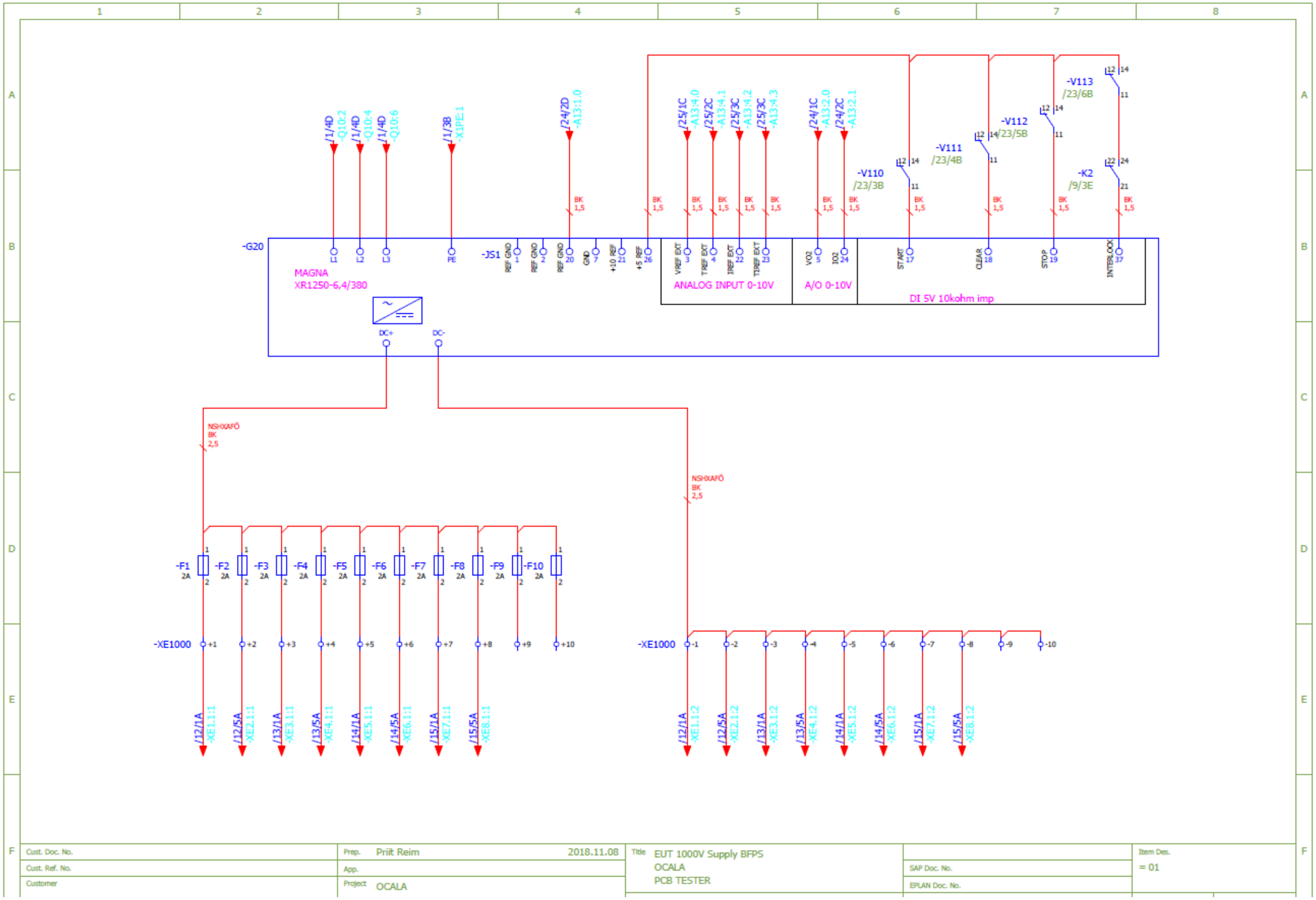


Figure A3.6 Latest revision of 1000V DC Supply

Appendix 4 chamber graphs

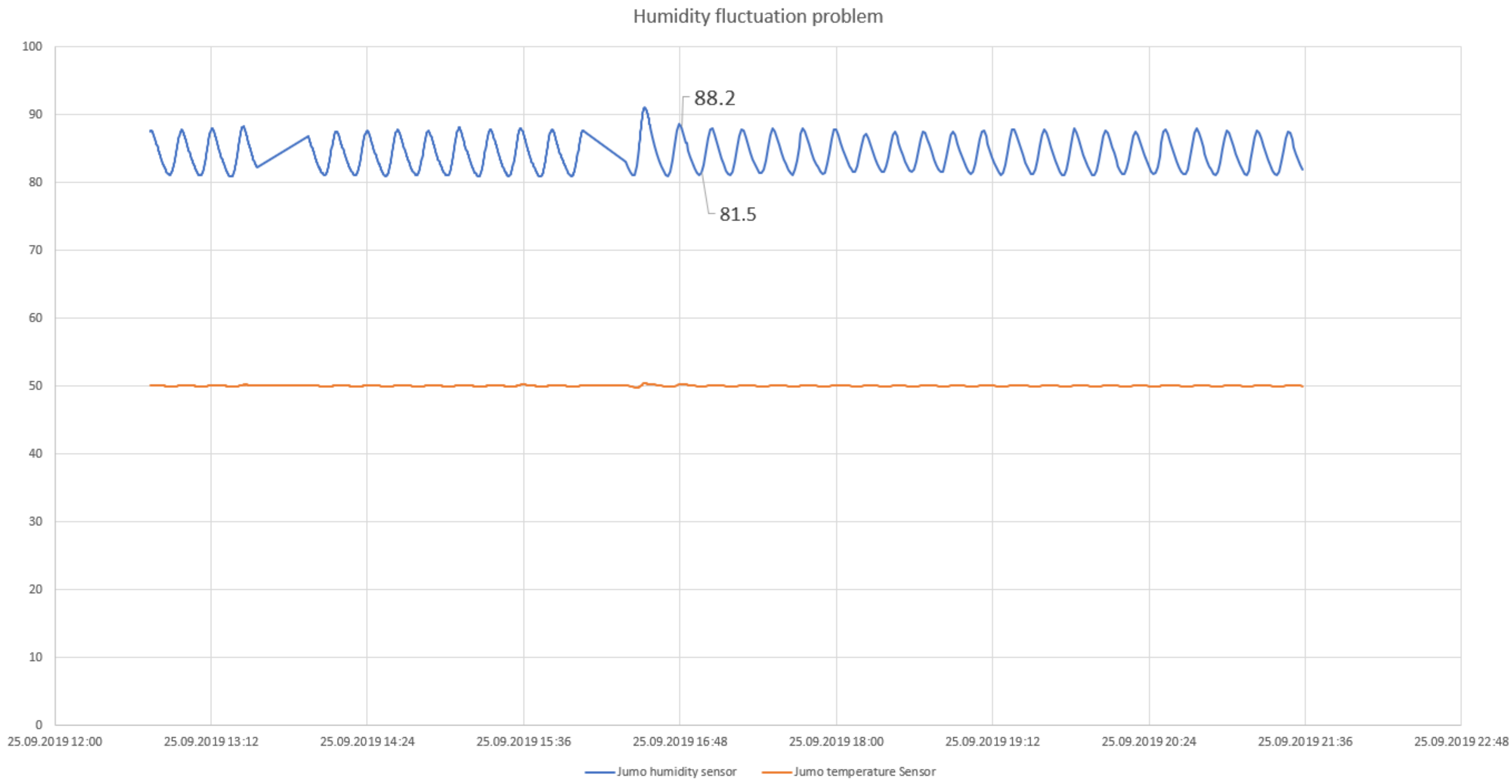


Figure A4.1 humidity fluctuation

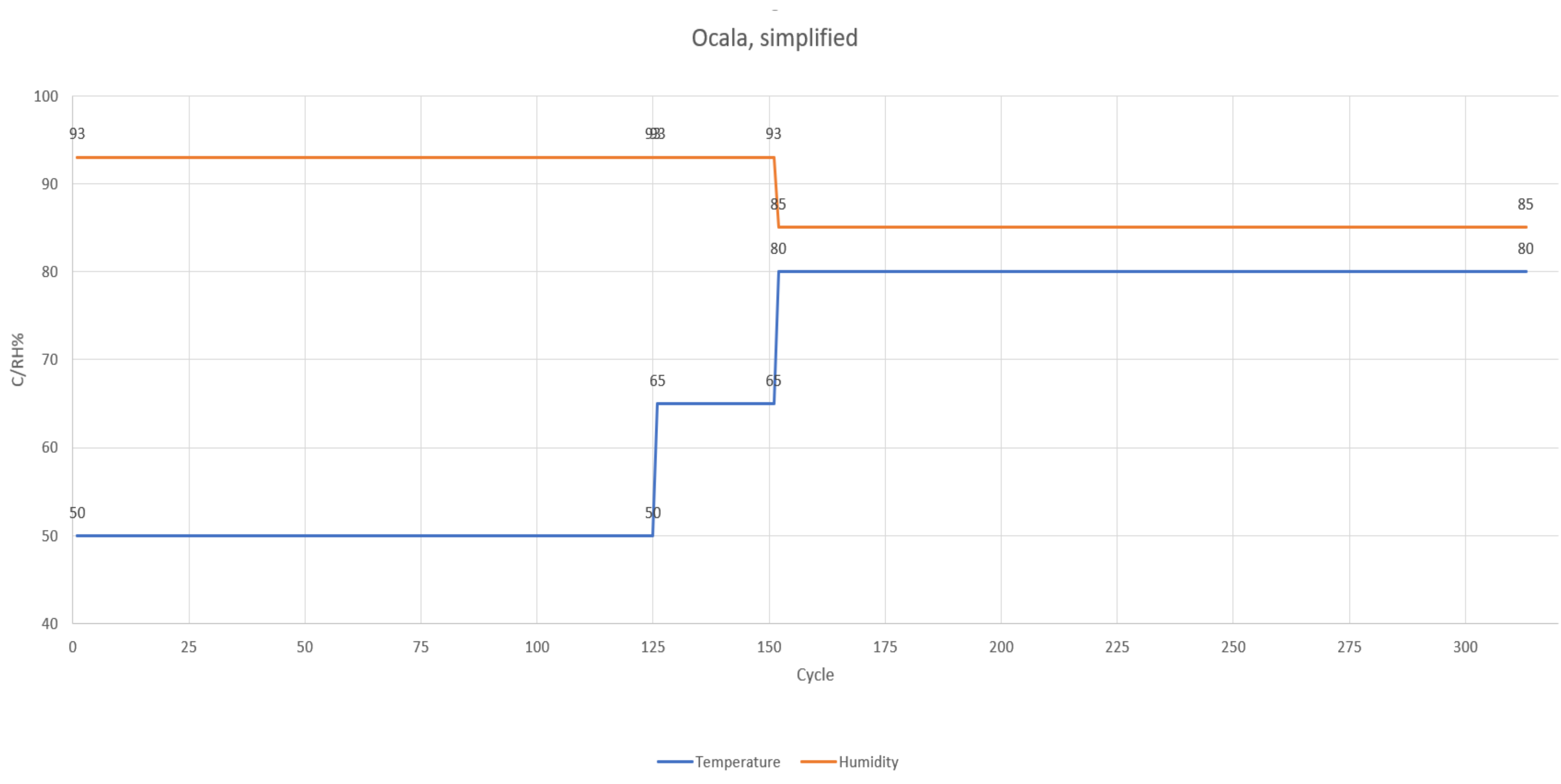


Figure A4.2 Chamber ambient test setpoints

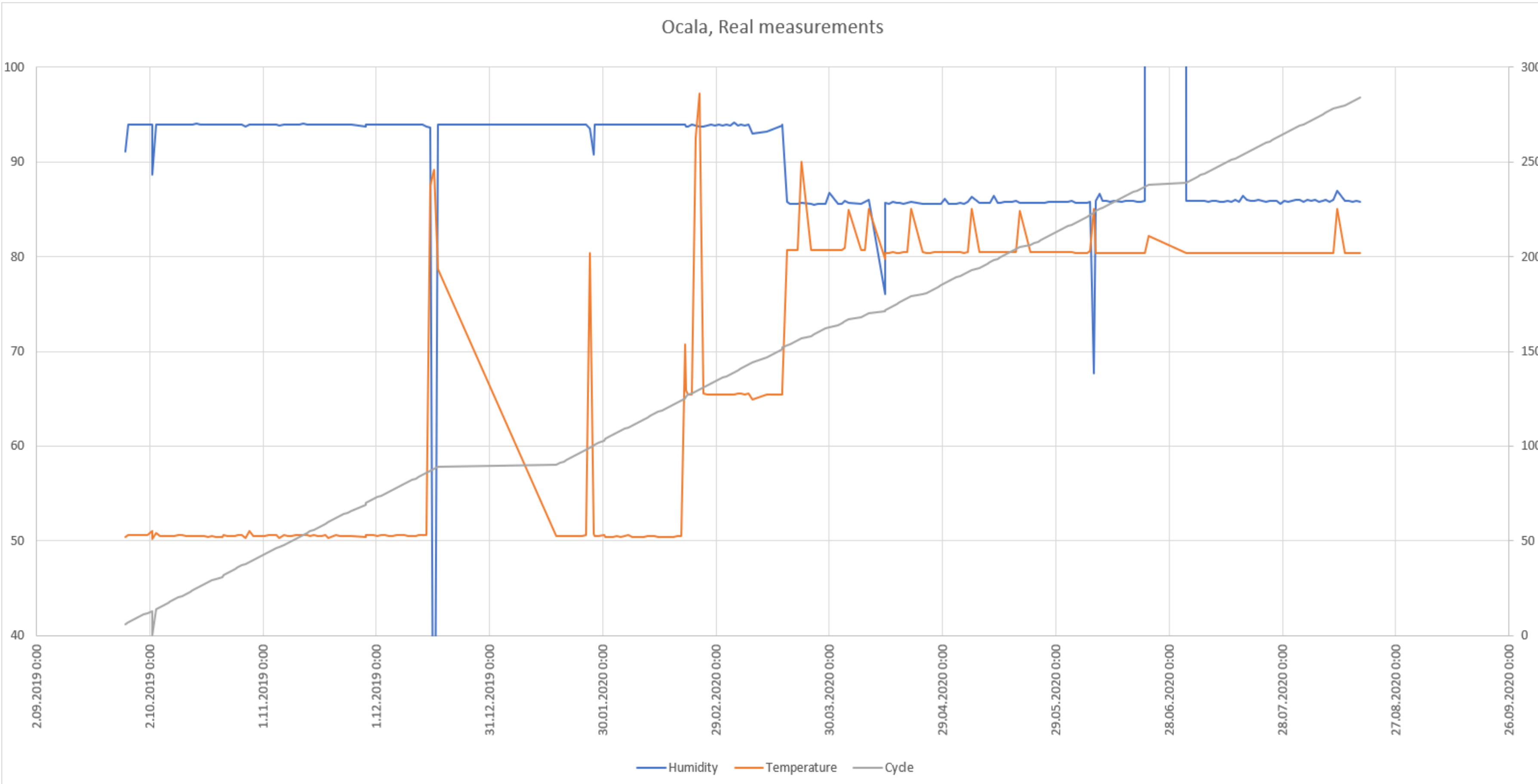


Figure A4.3 Ambient test conditions measured

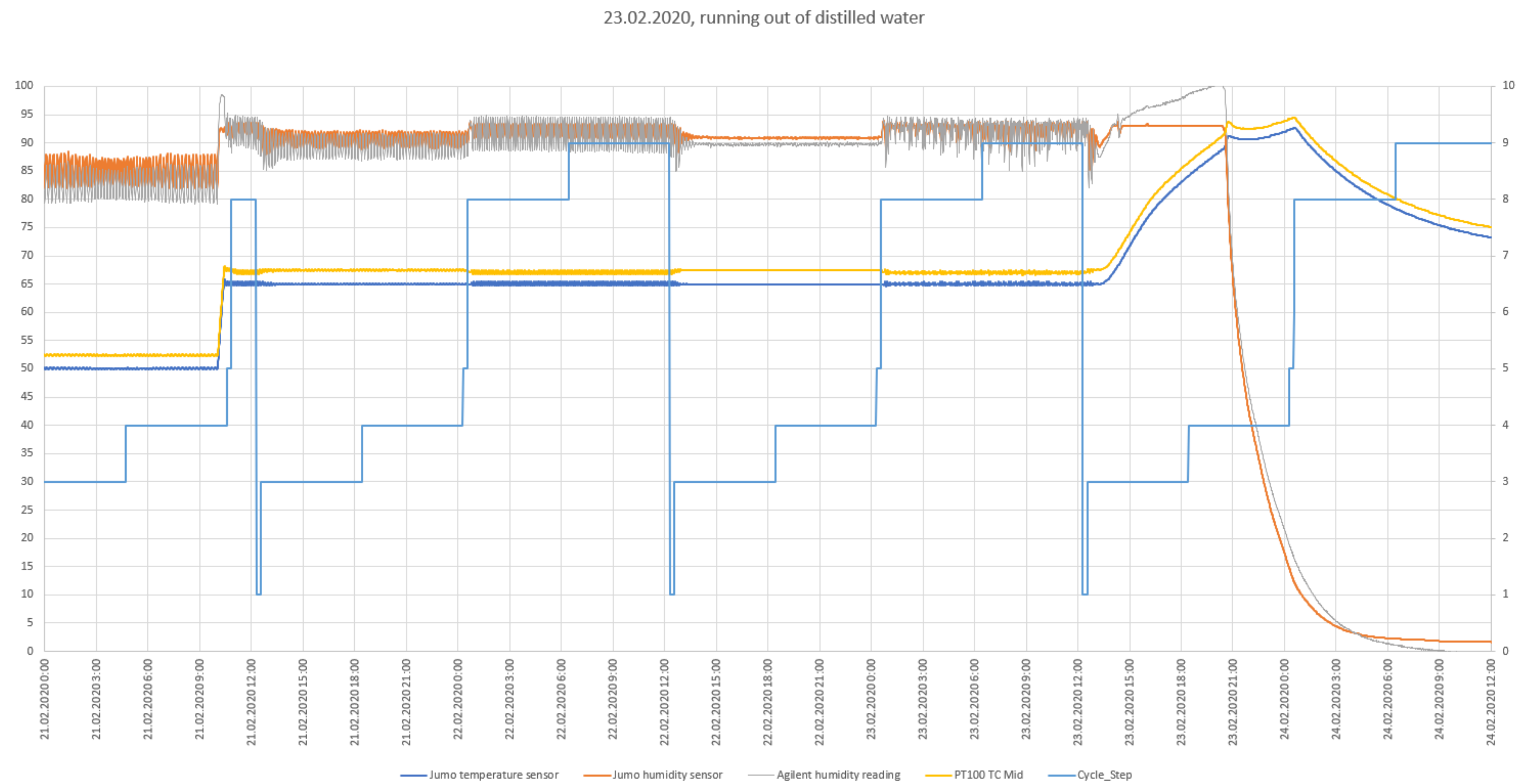


Figure A4.4 Changing from 50°C to 65°C ambient and running out of water

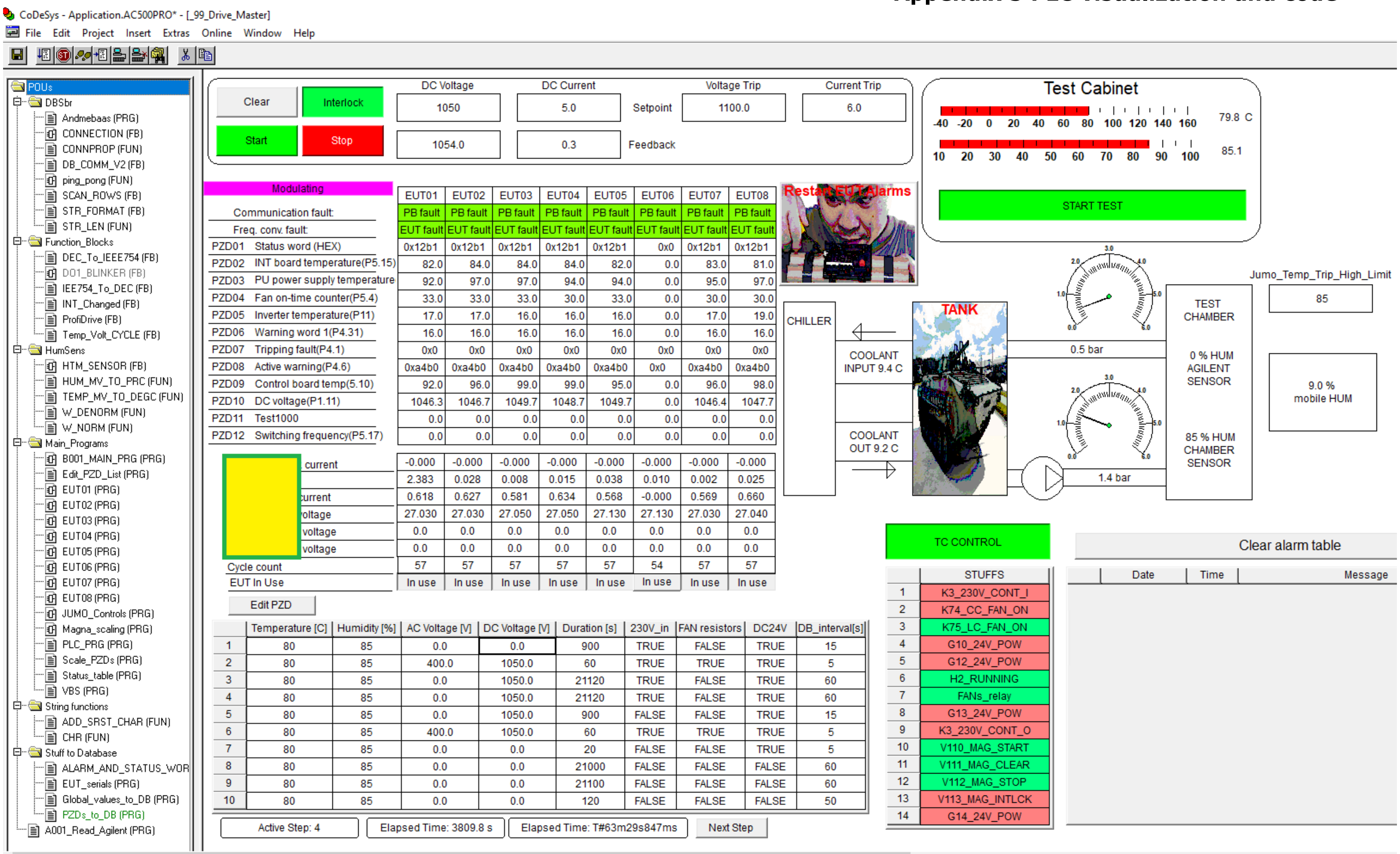


Figure A5.1 PLC MAIN visualization

	Temperature [C]	Humidity [%]	AC Voltage [V]	DC Voltage [V]	Duration [s]	230V_in	FAN resistors	DC24V	DB_interval[s]
1	80	85	0.0	0.0	900	TRUE	FALSE	TRUE	15
2	80	85	400.0	1050.0	60	TRUE	TRUE	TRUE	5
3	80	85	0.0	1050.0	21120	TRUE	FALSE	TRUE	60
4	80	85	0.0	1050.0	21120	TRUE	FALSE	TRUE	60
5	80	85	0.0	1050.0	900	FALSE	FALSE	TRUE	15
6	80	85	400.0	1050.0	60	TRUE	TRUE	TRUE	5
7	80	85	0.0	0.0	20	FALSE	TRUE	TRUE	5
8	80	85	0.0	0.0	21100	FALSE	FALSE	FALSE	60
9	80	85	0.0	0.0	21000	FALSE	FALSE	FALSE	60
10	80	85	0.0	0.0	120	FALSE	FALSE	TRUE	50

Figure A5.2 cycle settings table

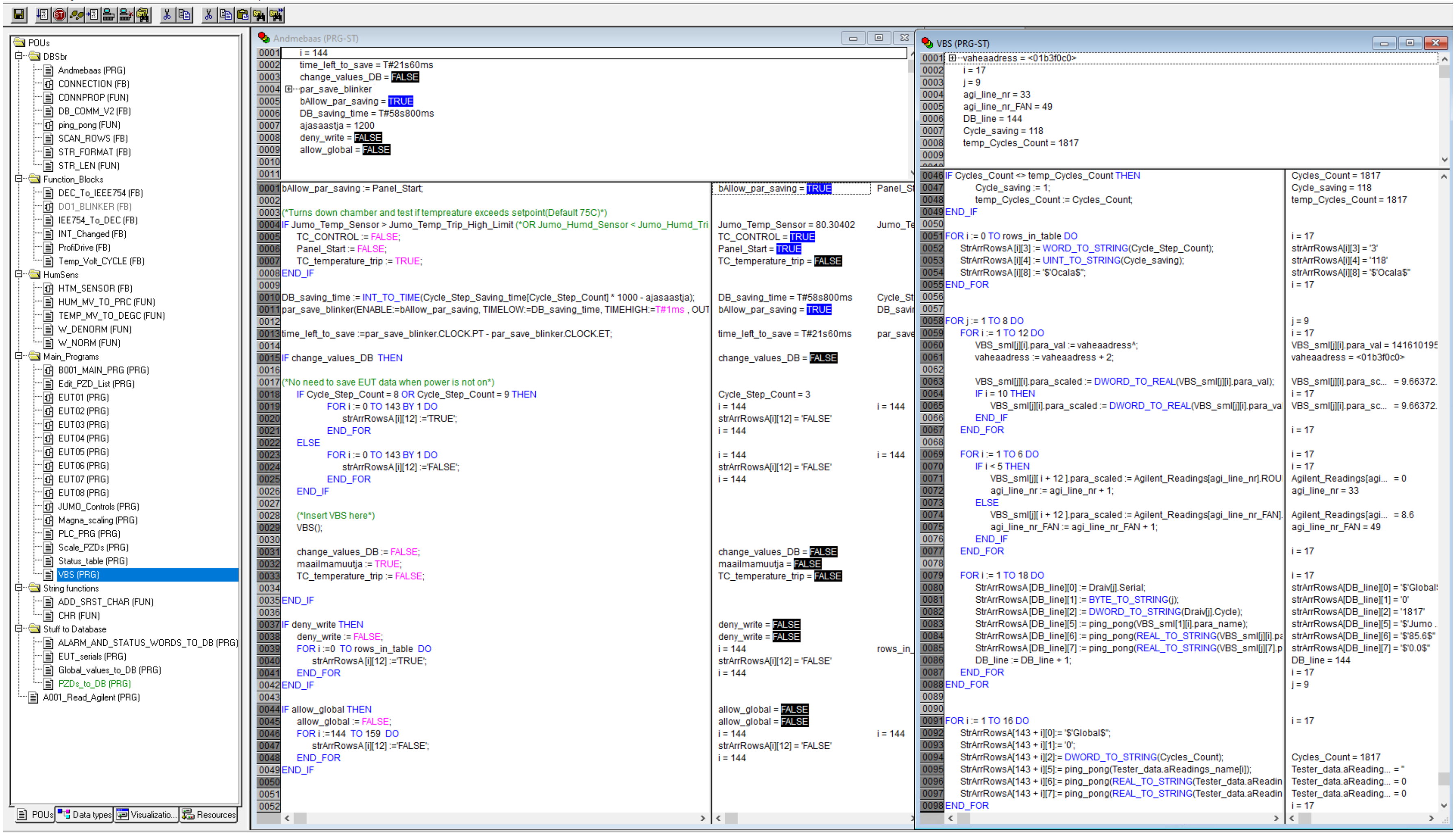


Figure A5.3 PLC Code example

Appendix 6 General tester programming structure

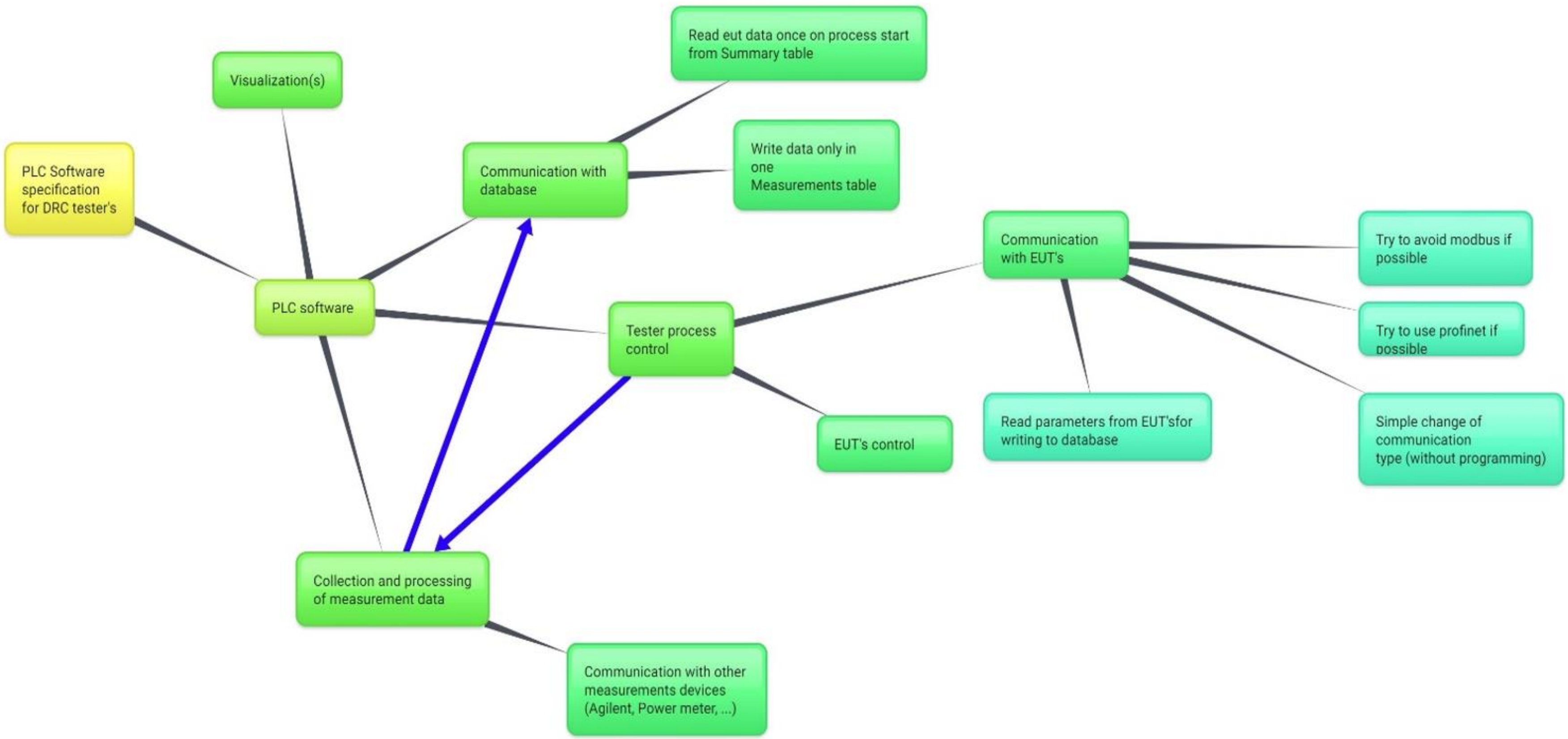
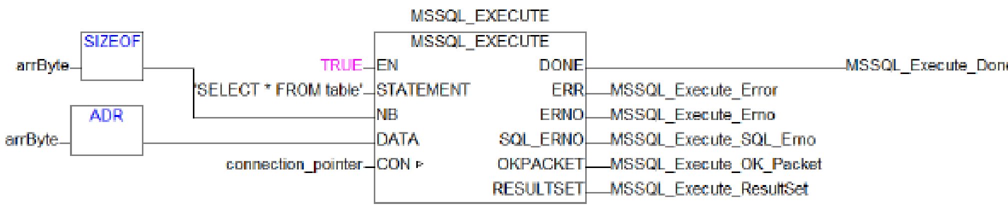


Figure A6.1 General tester programming structure

7.4.2 MSSQL_Execute



Executes the given MSSQL statement and returns the ResultSet(Query) or the acknowledgment packet (NonQuery). A Query could be SELECT; a NonQuery could be INSERT, DELETE and UPDATE etc. The FB will set the content of the received response to the address DATA and the corresponding length. We can receive a maximum of 250 columns.

Block Data

Available as of PLC runtime system:	V2.4.x	Remark:
Included in library:	MSSQL_AC500_V24.lib	

Block Type

Function block with historical values.

Parameter

EN	Input	BOOL	Enable function block by FALSE->TRUE edge
STATEMENT	Input	STRING(500)	Executable MSSQL statement
NB	Input	WORD	Number of data to be read
DATA	Input	DWORD	Address of the array where the response data are stored
CON	In/Out	POINTER	Connection handle to the MS SQL server
DONE	Output	BOOL	Execution finished when output DONE = TRUE
ERR	Output	BOOL	Error occurred during execution when output ERR = TRUE
ERNO	Output	WORD	Error code
SQL_ERNO	Output	WORD	SQL Error code. Please check here for more information.
OKPACKET	Output	MSSQL_OKPACKET_TYPE	Response to a NonQuery statement (INSERT, UPDATE, ...)
RESULTSET	Output	MSSQL_RESULTSET_TYPE	Response to a SELECT statement. Can be read by MSSQL_GETVALUE

Figure A7.1 MSSQL_Execute

Appendix 8 Server code structure

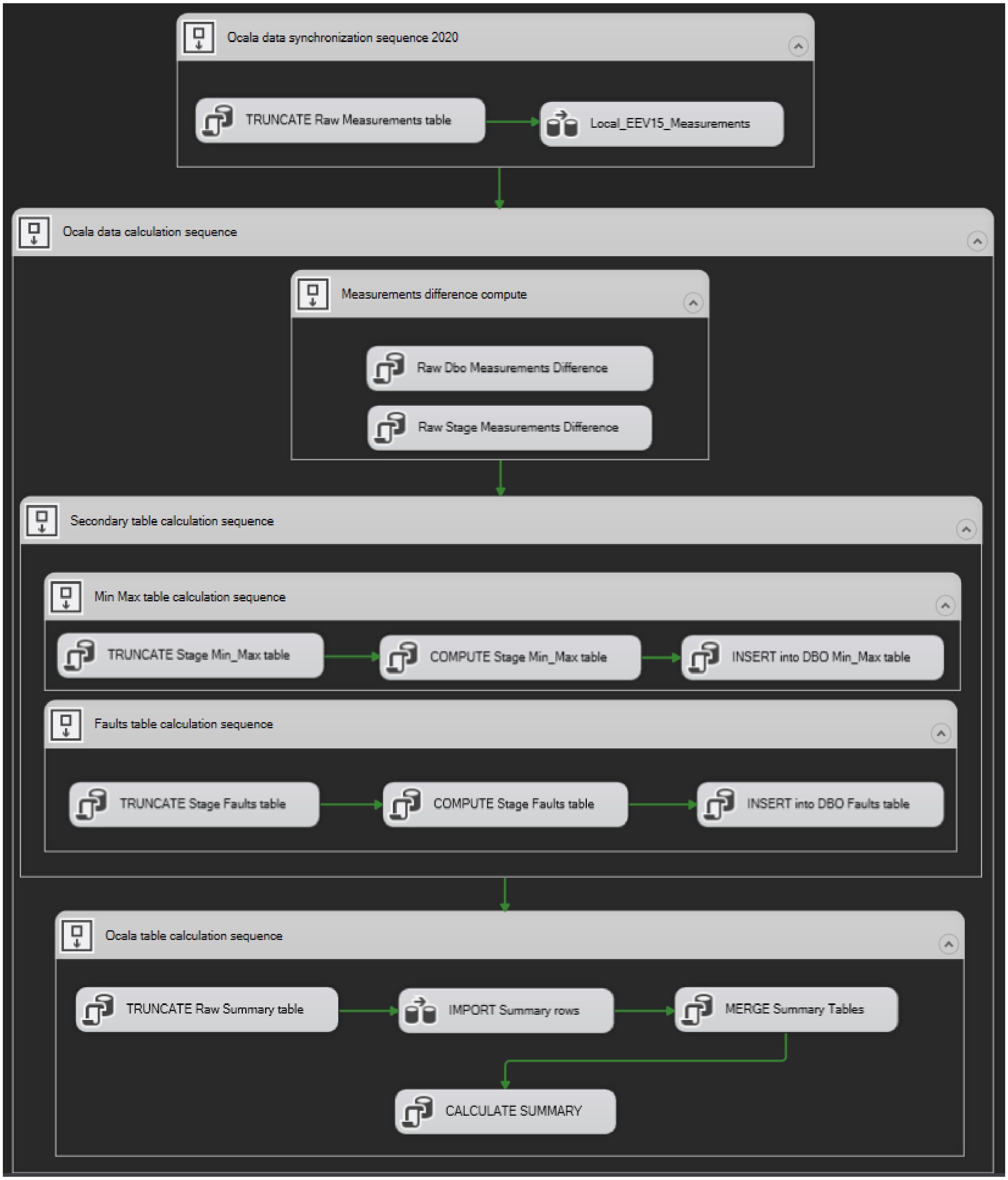


Figure A8.1 Server code structure

```
TRUNCATE TABLE [Ocala_2020].[Raw].[Measurements]
----------------------------------------------------------------------------------------
SELECT [Id]
      ,[Serial_number] COLLATE Latin1_General_CI_AS [Serial_number]
      ,[EUT_place]
      ,[Cycle]
      ,[Cycle_step]
      ,[Cycle_saving]
      ,[Parameter_name] COLLATE Latin1_General_CI_AS [Parameter_name]
      ,[Parameter_value] COLLATE Latin1_General_CI_AS [Parameter_value]
      ,[Time_stamp]
      ,[Active_faults] COLLATE Latin1_General_CI_AS [Active_faults]
      ,[User_modified] COLLATE Latin1_General_CI_AS [User_modified]
  FROM [Ocala].[dbo].[Measurements]
  WHERE Time_stamp > ?

----------------------------------------------------------------------------------------

MERGE [Ocala_2020].[dbo].[Measurements] AS TARGET
USING [Ocala_2020].[Raw].[Measurements] AS SOURCE
ON TARGET.[Id] = SOURCE.[Id]
WHEN NOT MATCHED BY TARGET
THEN INSERT ([Id], [Serial_number], [EUT_place], [Cycle], [Cycle_step], [Cycle_saving],
[Parameter_name], [Parameter_value],[Time_stamp],[Active_faults],[User_modified])
VALUES (SOURCE.[Id], SOURCE.[Serial_number], SOURCE.[EUT_place], SOURCE.[Cycle],
SOURCE.[Cycle_step], SOURCE.[Cycle_saving], SOURCE.[Parameter_name],
SOURCE.[Parameter_value],
SOURCE.[Time_stamp], SOURCE.[Active_faults], SOURCE.[User_modified]);

----------------------------------------------------------------------------------------

MERGE [Ocala_2020].[Stage].[Measurements] AS TARGET
USING [Ocala_2020].[Raw].[Measurements] AS SOURCE
ON TARGET.[Id] = SOURCE.[Id]
WHEN NOT MATCHED BY TARGET
THEN INSERT ([Id], [Serial_number], [EUT_place], [Cycle], [Cycle_step], [Cycle_saving],
[Parameter_name], [Parameter_value],[Time_stamp],[Active_faults],[User_modified])
VALUES (SOURCE.[Id], SOURCE.[Serial_number], SOURCE.[EUT_place], SOURCE.[Cycle],
SOURCE.[Cycle_step], SOURCE.[Cycle_saving], SOURCE.[Parameter_name],
SOURCE.[Parameter_value],
SOURCE.[Time_stamp], SOURCE.[Active_faults], SOURCE.[User_modified]);

***************************************************************************************

TRUNCATE TABLE [Ocala_2020].[Stage].[Min_Max]
----------------------------------------------------------------------------------------
IF OBJECT_ID('tempdb..#Min_Max_1_Ocala') IS NOT NULL DROP TABLE #Min_Max_1_Ocala
IF OBJECT_ID('tempdb..#Min_Max_2_Ocala') IS NOT NULL DROP TABLE #Min_Max_2_Ocala

-- Stage 1
SELECT [Serial_number],
       [Cycle],
       [Parameter_name]
```

```
       INTO #Min_Max_1_Ocala
FROM [Ocala_2020].[Stage].[Measurements]
WHERE [Serial_number] <> ''
AND [Time_stamp] > ?
GROUP BY Cycle, Parameter_name, Serial_number

-- Stage 2
SELECT [MM1].[Serial_number],
       MAX([M1].[EUT_place]) [EUT_place],
       [MM1].[Cycle],
       [MM1].[Parameter_name],
       MIN([M1].[Time_stamp]) [Start],
       MAX([M1].[Time_stamp]) [End]
       INTO #Min_Max_2_Ocala
FROM #Min_Max_1_Ocala [MM1]
INNER JOIN [Ocala_2020].[Stage].[Measurements] [M1] ON [MM1].[Serial_number] =
[M1].[Serial_number]
AND [MM1].[Cycle] = [M1].[Cycle] AND [MM1].[Parameter_name] = [M1].[Parameter_name]
GROUP BY [MM1].[Serial_number], [MM1].[Cycle],  [MM1].[Parameter_name]

-- Stage 3
INSERT INTO [Ocala_2020].[Stage].[Min_Max]
SELECT [MM2].[Serial_number]
    ,[MM2].[EUT_place]
    ,[MM2].[Cycle]
    ,[MM2].[Parameter_name]
    ,(
                SELECT MIN(CAST([Parameter_value] AS DECIMAL(18,1)))
                FROM [Ocala_2020].[Stage].[Measurements]
                where Serial_number = [MM2].[Serial_number]  and Parameter_name =
[MM2].[Parameter_name] and Cycle = [MM2].Cycle
                AND TRY_CAST([Parameter_value] AS DECIMAL(18,1)) IS NOT NULL
            ) [CalculatedMinAbsolute]
    ,(
                SELECT MAX(CAST([Parameter_value] AS DECIMAL(18,1)))
                FROM [Ocala_2020].[Stage].[Measurements]
                where Serial_number = [MM2].[Serial_number]  and Parameter_name =
[MM2].[Parameter_name] and Cycle = [MM2].Cycle
                AND TRY_CAST([Parameter_value] AS DECIMAL(18,1)) IS NOT NULL
            ) [CalculatedMaxAbsolute]
    ,CAST([MM2].[Start] AS DATETIME2(0)) [Cycle_start_time]
    ,CAST([MM2].[End] AS DATETIME2(0)) [Cycle_end_time]
    ,CAST([MM2].[Start] AS DATETIME2) [Cycle_start_time_Sort]
FROM #Min_Max_2_Ocala [MM2]
ORDER BY [MM2].[Start] DESC

IF OBJECT_ID('tempdb..#Min_Max_1') IS NOT NULL DROP TABLE #Min_Max_1_Ocala
IF OBJECT_ID('tempdb..#Min_Max_2') IS NOT NULL DROP TABLE #Min_Max_2_Ocala

---------------------------------------------------------------------------------------

MERGE [Ocala_2020].[dbo].[Min_Max] AS TARGET
USING
(SELECT TOP (100) PERCENT [Serial_number]
    ,[EUT_place]
    ,[Cycle]
    ,[Parameter_name]
    ,[Cycle_start_time]
```

```
    ,[Cycle_end_time]
    ,[Min_value]
    ,[Max_value]
FROM [Ocala_2020].[Stage].[Min_Max] ORDER BY [Cycle_start_time_Sort] ASC) AS SOURCE
ON (TARGET.[Serial_number] = SOURCE.[Serial_number] AND TARGET.[EUT_place] =
SOURCE.[EUT_place] AND TARGET.[Cycle] = SOURCE.[Cycle] AND TARGET.[Parameter_name] =
SOURCE.[Parameter_name])
--When records are matched, update the records if there is any change
WHEN MATCHED AND TARGET.[Min_value] <> SOURCE.[Min_value] OR TARGET.[Max_value] <>
SOURCE.[Max_value]
OR TARGET.[Cycle_start_time] <> SOURCE.[Cycle_start_time] OR TARGET.[Cycle_end_time] <>
SOURCE.[Cycle_end_time]
THEN UPDATE SET TARGET.[Min_value] = SOURCE.[Min_value], TARGET.[Max_value] =
SOURCE.[Max_value] ,
TARGET.[Cycle_start_time] = SOURCE.[Cycle_start_time] , TARGET.[Cycle_end_time] =
SOURCE.[Cycle_end_time]

--When no records are matched, insert the incoming records from source table to target
table
WHEN NOT MATCHED BY TARGET
THEN INSERT ([Serial_number], [EUT_place], [Cycle], [Parameter_name],
[Cycle_start_time], [Cycle_end_time], [Min_value], [Max_value])
VALUES (SOURCE.[Serial_number], SOURCE.[EUT_place], SOURCE.[Cycle],
SOURCE.[Parameter_name], SOURCE.[Cycle_start_time], SOURCE.[Cycle_end_time],
SOURCE.[Min_value], SOURCE.[Max_value]);

*************************************************************************************

TRUNCATE TABLE [Ocala_2020].[Stage].[Faults]
--There should now somewhere be a TRUNCATE TABLE [Ocala_2020].[Stage].[ActiveSn] sql
line after Stage truncate
---------------------------------------------------------------------------------------
INSERT INTO [Ocala_2020].[Stage].[ActiveSn]
SELECT DISTINCT [Serial_number]
FROM [Ocala_2020].[dbo].[Measurements]
WHERE [Serial_number] <> 'Global' AND [Serial_number] <> ''
AND CAST([Time_stamp] AS DATE) >= ?

INSERT INTO [Ocala_2020].[Stage].[FaultsSourceData]
SELECT [Id], [Serial_number], [EUT_place], [Active_faults], [Cycle], [Cycle_step],
[Time_stamp],
[Previous_fault] = LAG([Active_faults]) OVER (ORDER BY [Serial_number], [Id])
FROM [Ocala_2020].[dbo].[Measurements]
WHERE [Serial_number] IN (
    SELECT [Serial_number] FROM [Ocala_2020].[Stage].[ActiveSn]
    ) AND [Parameter_name] LIKE 'Status word%'

INSERT INTO [Ocala_2020].[Stage].[Faults]
select A.Serial_number, A.EUT_place, A.Cycle, A.Cycle_step as
FaultOccurrence_Cycle_step, MIN(B.Cycle_step) as FaultDisappearance_Cycle_step,
A.Active_faults, A.Time_stamp as FaultStart, MIN(B.Time_stamp) as FaultEnd
from [Stage].[FaultsSourceData] A, [Stage].[FaultsSourceData] B
where A.Active_faults <> A.Previous_fault and A.Active_faults <> '0'
and    B.Active_faults <> B.Previous_fault and B.Active_faults = '0'
and A.Serial_number = B.Serial_number
and A.Time_stamp < B.Time_stamp
group by A.Time_stamp, A.Id, A.Serial_number, A.EUT_place, A.Active_faults, A.Cycle,
A.Cycle_step
```

```sql
ORDER BY A.Time_stamp ASC


EXEC [dbo].[Ocala_New_Fault_Notification]

-------------------------------------------------------------------------------------
--Notification procedure
USE [Ocala_2020]
GO
/****** Object:  StoredProcedure [dbo].[Ocala_New_Fault_Notification]
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

ALTER PROCEDURE [dbo].[Ocala_New_Fault_Notification]
AS
BEGIN
       -- SET NOCOUNT ON added to prevent extra result sets from
       -- interfering with SELECT statements.
       SET NOCOUNT ON;

    -- Insert statements for procedure here
        IF (SELECT IIF((SELECT COUNT([Serial_number]) FROM
[Ocala_2020].[Stage].[Faults]) = (SELECT COUNT([Serial_number]) FROM
[Ocala_2020].[dbo].[Faults]
  WHERE
CHECKSUM([Serial_number],[EUT_place],[FaultOccurrence_Cycle_step],[Active_faults],[Faul
tStart]) IN
  (SELECT
CHECKSUM([Serial_number],[EUT_place],[FaultOccurrence_Cycle_step],[Active_faults],[Faul
tStart]) FROM [Ocala_2020].[Stage].[Faults])), 0, 1) [IsNewFault]) = 1
     BEGIN
              DECLARE @xml NVARCHAR(MAX)
              DECLARE @body NVARCHAR(MAX)

              SET @xml = CAST ( ( Select td = [Serial_number],'',
                                  td = [EUT_Place],'',
                                  td = [Cycle],'',
                                  td = [FaultOccurrence_Cycle_step],'',
                                  td = [Active_faults], '',
                                  td = [FaultStart]
                            FROM [Ocala_2020].[Stage].[Faults]
                where
CONCAT([Serial_number],[EUT_place],[FaultOccurrence_Cycle_step],[Active_faults],[FaultS
tart]) NOT IN
                (SELECT
CONCAT([Serial_number],[EUT_place],[FaultOccurrence_Cycle_step],[Active_faults],[FaultS
tart]) FROM [Ocala_2020].[dbo].[Faults])
                ORDER BY [FaultStart] DESC
                            FOR XML PATH('tr'), TYPE
                     ) AS NVARCHAR(MAX))

              SET @body ='<html><body><H3>Ocala - New fault found</H3>
              <table border = 1>
              <tr>
```

```
       <th>Serial_number</th><th>EUT_Place</th><th>Cycle</th><th>FaultOccurrence_Cycle_
step</th><th>Active_faults</th><th>FaultStart</th></tr>'

              SET @body = @body + @xml +'</table></body></html>'

        EXEC msdb.dbo.sp_send_dbmail
            @profile_name = 'V15_LVDTESTER_SQL@xxx',
            @recipients = 'martin.onton@xxx',
            @copy_recipients = 'voldemar.balder@xxx',
            @Execute_query_database = 'Ocala_2020',
            @subject = 'New fault row found in [Ocala_2020].[dbo].[Faults]',
            @body = @body,
                     @body_format ='HTML'
     END
END

---------------------------------------------------------------------------------------

MERGE [Ocala_2020].[dbo].[Faults] AS TARGET
USING (SELECT TOP 100 PERCENT [Serial_number]
      ,[EUT_place]
      ,[Cycle]
      ,[FaultOccurrence_Cycle_step]
      ,[FaultDisappearance_Cycle_step]
      ,[Active_faults]
      ,[FaultStart]
      ,[FaultEnd]
  FROM [Ocala_2020].[Stage].[Faults]
  ORDER BY [FaultStart] ASC) AS SOURCE
ON (TARGET.[Serial_number] = SOURCE.[Serial_number] AND TARGET.[Active_faults] =
SOURCE.[Active_faults] AND TARGET.[Cycle] = SOURCE.[Cycle]
AND TARGET.[FaultOccurrence_Cycle_step] = SOURCE.[FaultOccurrence_Cycle_step] AND
TARGET.[FaultStart] = SOURCE.[FaultStart])
--When records are matched, update the records if there is any change
WHEN MATCHED AND TARGET.[FaultStart] <> SOURCE.[FaultStart] OR TARGET.[FaultEnd] <>
SOURCE.[FaultEnd]
THEN UPDATE SET TARGET.[FaultStart] = SOURCE.[FaultStart], TARGET.[FaultEnd] =
SOURCE.[FaultEnd]
--When no records are matched, insert the incoming records from source table to target
table
WHEN NOT MATCHED BY TARGET
THEN INSERT ([Serial_number], EUT_place, [Cycle], [FaultOccurrence_Cycle_step],
[FaultDisappearance_Cycle_step],  [Active_faults], [FaultStart], [FaultEnd])
VALUES (SOURCE.[Serial_number], SOURCE.[EUT_place], SOURCE.[Cycle],
source.[FaultOccurrence_Cycle_step], SOURCE.[FaultDisappearance_Cycle_step],
SOURCE.[Active_faults], SOURCE.[FaultStart], SOURCE.[FaultEnd]);

**************************************************************************************


TRUNCATE TABLE [Ocala_2020].[Raw].[Summary]
---------------------------------------------------------------------------------------
MERGE [Ocala_2020].[Stage].[Summary] AS TARGET
USING (SELECT TOP 100 PERCENT [Id]
      ,[Serial_number]
      ,[EUT_place]
      ,[Type]
```

```
      ,[Frame_size]
      ,[Project]
      ,[Target_cycles]
      ,[Added_on]
  FROM [Ocala_2020].[Raw].[Summary]) AS SOURCE
ON (TARGET.[Id] = SOURCE.[Id] AND TARGET.[Serial_number] = SOURCE.[Serial_number] AND
TARGET.[EUT_place] = SOURCE.[EUT_place])
--When records are matched, update the records if there is any change
WHEN MATCHED AND TARGET.[Type] <> SOURCE.[Type] OR TARGET.[Frame_size] <>
SOURCE.[Frame_size] OR TARGET.[Project] <> SOURCE.[Project] OR TARGET.[Target_cycles]
<> SOURCE.[Target_cycles] OR TARGET.[Added_on] <> SOURCE.[Added_on]
THEN UPDATE SET TARGET.[Type] = SOURCE.[Type], TARGET.[Frame_size] =
SOURCE.[Frame_size] , TARGET.[Project] = SOURCE.[Project] , TARGET.[Target_cycles] =
SOURCE.[Target_cycles] , TARGET.[Added_on] = SOURCE.[Added_on]
--When no records are matched, insert the incoming records from source table to target
table
WHEN NOT MATCHED BY TARGET
THEN INSERT ([Id], [Serial_number], [EUT_place], [Type], [Frame_size],  [Project],
[Target_cycles], [Added_on]) VALUES (SOURCE.[Id], SOURCE.[Serial_number],
SOURCE.[EUT_place], source.[Type], SOURCE.[Frame_size], SOURCE.[Project],
SOURCE.[Target_cycles], SOURCE.[Added_on]);

MERGE [Ocala_2020].[dbo].[Summary] AS TARGET
USING (SELECT TOP 100 PERCENT [Id]
      ,[Serial_number]
      ,[EUT_place]
      ,[Type]
      ,[Frame_size]
      ,[Project]
      ,[Target_cycles]
      ,[Added_on]
  FROM [Ocala_2020].[Raw].[Summary]) AS SOURCE
ON (TARGET.[Id] = SOURCE.[Id] AND TARGET.[Serial_number] = SOURCE.[Serial_number] AND
TARGET.[EUT_place] = SOURCE.[EUT_place])
--When records are matched, update the records if there is any change
WHEN MATCHED AND TARGET.[Type] <> SOURCE.[Type] OR TARGET.[Frame_size] <>
SOURCE.[Frame_size] OR TARGET.[Project] <> SOURCE.[Project] OR TARGET.[Target_cycles]
<> SOURCE.[Target_cycles] OR TARGET.[Added_on] <> SOURCE.[Added_on]
THEN UPDATE SET TARGET.[Type] = SOURCE.[Type], TARGET.[Frame_size] =
SOURCE.[Frame_size] , TARGET.[Project] = SOURCE.[Project] , TARGET.[Target_cycles] =
SOURCE.[Target_cycles] , TARGET.[Added_on] = SOURCE.[Added_on]
--When no records are matched, insert the incoming records from source table to target
table
WHEN NOT MATCHED BY TARGET
THEN INSERT ([Id], [Serial_number], [EUT_place], [Type], [Frame_size],  [Project],
[Target_cycles], [Added_on]) VALUES (SOURCE.[Id], SOURCE.[Serial_number],
SOURCE.[EUT_place], source.[Type], SOURCE.[Frame_size], SOURCE.[Project],
SOURCE.[Target_cycles], SOURCE.[Added_on]);
--------------------------------------------------------------------------------------
IF OBJECT_ID('tempdb..#Summary_Latest_SN') IS NOT NULL DROP TABLE #Summary_Latest_SN
IF OBJECT_ID('tempdb..#Measurement_Latest_SN_ID') IS NOT NULL DROP TABLE
#Measurement_Latest_SN_ID
IF OBJECT_ID('tempdb..#Summary_Update') IS NOT NULL DROP TABLE #Summary_Update

TRUNCATE TABLE [Stage].[Summary_Latest_SN]

INSERT INTO [Stage].[Summary_Latest_SN]
SELECT DISTINCT (
```

```sql
    SELECT TOP 1 [Serial_number] FROM [Stage].[Summary] [S2] WHERE [S].[EUT_place] =
[S2].[EUT_place]
    ORDER BY [Id] DESC
    ) [Serial_number]
FROM [Ocala_2020].[Stage].[Summary] [S]

SELECT [M2].Serial_number, MAX([M2].Id) [LastId]
into #Measurement_Latest_SN_ID
FROM [Ocala_2020].[dbo].[Measurements] [M2]
group by [M2].Serial_number

SELECT [SLSN].[Serial_number]
,(SELECT MAX([Cycle]) FROM [Ocala_2020].[dbo].[Measurements] [M1] WHERE
[SLSN].[Serial_number] = [M1].[Serial_number]) [Cycles_done]
,(SELECT [M2].[Active_faults] FROM [Ocala_2020].[dbo].[Measurements] [M2] WHERE
[M2].[Id] = (select [mid].[LastId] from #Measurement_Latest_SN_ID [MID] where
[SLSN].[Serial_number] = [MID].Serial_number)) [Active_Fault]
,IIF((SELECT CONVERT(int, convert(real, (SELECT TOP 1 [Active_faults] FROM
[Ocala_2020].[dbo].[Faults] [F1] where [SLSN].[Serial_number] = [F1].[Serial_number]
ORDER BY [Id] DESC)))) IS NULL, 0, (SELECT CONVERT(int, convert(real, (SELECT TOP 1
[Active_faults] FROM [Ocala_2020].[dbo].[Faults] [F1] where [SLSN].[Serial_number] =
[F1].[Serial_number] ORDER BY [Id] DESC)))))  [Last_Fault],
[TestStart].[Test_start],
[TestEnd].[Test_end],
[FaultCount].[Number_of_faults],
[BadCycleCount].[Bad_cycles],
[Simulated_lifetime] = --CODE removed, it is a secret
INTO #Summary_Update
FROM [Stage].[Summary_Latest_SN] [SLSN]
CROSS APPLY [dbo].[TestStart]([SLSN].[Serial_number]) [TestStart]
CROSS APPLY [dbo].[TestEnd]([SLSN].[Serial_number]) [TestEnd]
CROSS APPLY [dbo].[FaultCount]([SLSN].[Serial_number]) [FaultCount]
CROSS APPLY [dbo].[BadCycleCount]([SLSN].[Serial_number]) [BadCycleCount]

UPDATE #Summary_Update SET [Cycles_done] = IIF([Cycles_done] IS NULL, 0,
[Cycles_done]),
[Active_Fault] = IIF([Active_Fault] IS NULL, 0, [Active_Fault]),
[Simulated_lifetime] = IIF([Simulated_lifetime] IS NULL, 0, [Simulated_lifetime])


MERGE [Ocala_2020].[dbo].[Summary] AS TARGET
USING
(SELECT TOP (100) PERCENT [Serial_number]
    ,[Cycles_done]
    ,[Active_Fault]
    ,[Last_Fault]
    ,[Test_start]
    ,[Test_end]
    --,[EUT_in_use] [EUT_Active]
    ,[Number_of_faults]
    ,[Bad_cycles]
    --,[Good_cycles]
    ,[Simulated_lifetime]
FROM #Summary_Update) AS SOURCE
ON (SOURCE.[Serial_number] LIKE TARGET.[Serial_number])
--When records are matched, update the records if there is any change
WHEN MATCHED
```

```sql
THEN UPDATE SET TARGET.[Cycles_done] = SOURCE.[Cycles_done], TARGET.[Active_Fault] =
SOURCE.[Active_Fault] , TARGET.[Last_Fault] = SOURCE.[Last_Fault] , TARGET.[Test_start]
= SOURCE.[Test_start] , TARGET.[Test_end] = SOURCE.[Test_end] ,
TARGET.[Number_of_faults] = SOURCE.[Number_of_faults]  , TARGET.[Bad_cycles] =
SOURCE.[Bad_cycles], TARGET.[Simulated_years] = SOURCE.[Simulated_lifetime];

MERGE [Ocala_2020].[Stage].[Summary] AS TARGET
USING
(SELECT TOP (100) PERCENT [Serial_number]
    ,[Cycles_done]
    ,[Active_Fault]
    ,[Last_Fault]
    ,[Test_start]
    ,[Test_end]
    --,[EUT_in_use] [EUT_Active]
    ,[Number_of_faults]
    ,[Bad_cycles]
    --,[Good_cycles]
    ,[Simulated_lifetime]
FROM #Summary_Update) AS SOURCE
ON (SOURCE.[Serial_number] LIKE TARGET.[Serial_number])
--When records are matched, update the records if there is any change
WHEN MATCHED
THEN UPDATE SET TARGET.[Cycles_done] = SOURCE.[Cycles_done], TARGET.[Active_Fault] =
SOURCE.[Active_Fault] , TARGET.[Last_Fault] = SOURCE.[Last_Fault] , TARGET.[Test_start]
= SOURCE.[Test_start] , TARGET.[Test_end] = SOURCE.[Test_end] ,
TARGET.[Number_of_faults] = SOURCE.[Number_of_faults]  , TARGET.[Bad_cycles] =
SOURCE.[Bad_cycles] , TARGET.[Simulated_years] = SOURCE.[Simulated_lifetime];

IF OBJECT_ID('tempdb..#Summary_Latest_SN') IS NOT NULL DROP TABLE #Summary_Latest_SN
IF OBJECT_ID('tempdb..#Summary_Update') IS NOT NULL DROP TABLE #Summary_Update
```