Open your mind. LUT.
Lappeenranta University of Technology

Vierasmagnetoidun tahtikoneen automaatiokonseptin testaus
Externally excited synchronous machine: Testing the automation concept

Arto Tahvanainen

Kandidaatintyö
LUT Energia
Sähkötekniikan koulutusohjelma

31.7.2012

TIIVISTELMÄ

Lappeenrannan teknillinen yliopisto
Teknillinen tiedekunta
Sähkötekniikan koulutusohjelma

Arto Tahvanainen

Vierasmagnetoidun tahtikoneen automaatiokonseptin testaus

2012

Kandidaatintyö.
s. 27

Tarkastaja: Tero Ahonen

Kandidaatintyö on osa laajempaa tahtikonekonseptia vierasmagnetoidulle tahtikoneelle. Tämä kandidaatintyö keskittyy testaussuunnitelman rakentamiseen sekä testauksen toteuttamiseen.

Testaussuunnitelma tehtiin kolmelle osa-alueelle: PLC-ohjelman testaus, kommunikoinnin testaus sekä koko systeemin testaus. PLC-ohjelmaa testattiin white box ja destructive tavoilla. Kommunikoinnin testauksessa kokeiltiin, mikä on maksiminopeus luotettavalle kommunikoinnille laitteiden välillä. Koko systeemin testauksessa testattiin mm. nopeus- ja vääntösäätöä. Lisäksi testauksessa oli mukana kokeellinen testaus, jossa testattiin eri osa-alueita sattumanvaraisesti. Tämän avulla saatiin kattavampi testaus tulos kuin pelkällä systemaattisella testauksella.

ABSTRACT

Lappeenranta University of Technology
Faculty of Technology
Degree Programme in Electrical Engineering

Arto Tahvanainen

Externally excited synchronous machine: Testing the automation concept

2012

Bachelor's Thesis.
27 Pages

Examiner: Tero Ahonen

This bachelor's thesis is a part of the research project realized in the summer 2011 in Lappeenranta University of Technology. The goal of the project was to develop an automation concept for controlling the externally excited synchronous motor. Thesis concentrates on the testing planning and testing the system.

Testing plan was made for three sectors: For the PLC program testing, for the communication testing and for the whole system testing. PLC program was tested with white box and destructive methods. Communication testing was done by switching maximum communication speed and checked if communication was reliable. Whole system testing included among other things speed and torque controlling. The system was tested with exploratory testing also. This enabled more reliable and broader testing than with systematical testing only.

CONTENTS

USED ABBREVIATIONS

AC500	Type of PLC manufactured by ABB
ACS800	Type of frequency converter type manufactured by ABB
CM572	Communication Module of AC500
CodeSys	Programming software for PLCs
DCS800	Magnetization generator
PLC	Programmable Logic Controller
PM583	Processor module of AC500
PMSM	Permanent Magnet Synchronous Motor
POU	Program Organization Unit

1. INTRODUCTION

Main purpose in any kind of technical testing is to find out if a system or developed product works correctly and does it have errors. Also reliability, usability and other features of the system are usually tested. Testing usually goes according to testing plan and testing results are taken down and documented. In this project, goals of the testing are to find out errors in programmable logic controller code, communication limits between drives, and to test whole setup to see if it works correctly and reliably.

This bachelor thesis covers testing planning and actual testing of whole actual system. Testing focuses on three phases: PLC program, fieldbus communication, systems functionality and overall testing. The main goal of testing is to find errors in PLC code and in its implementation, communication errors, communication limits between machines and finding limits of changing speeds of system variables such as magnetization level change. Testing is done by manually searching limits, by using destructive and glass box methods to find errors in code, and by running the setup with different kinds of references and loads.

1.1 Goal of the project

The goal of the project was to create a low voltage externally excited synchronous machine setup, which includes a frequency converter, a field exciter and a programmable logic controller. Frequency converter uses a permanent magnet synchronous machine software and an external exciter that is controlled via Profibus cable. PLC handles communication between devices, calculations and adjustment of excitation current control.

This project focuses on creating working prototype of the system. Two other bachelor theses were also done which focused on PLC program and requirements speciation. Bachelor theses which focused on PLC program was made by Lauri Niinimäki. Requirement speciation was made by Ari Potinkara.

2. OVERVIEW OF TESTED SYSTEM

The system contains a PLC (AC500), a field exciter (DCS800), a frequency converter (ACS800) and an externally excited synchronous motor. Data transfer is established by using Profibus DP fieldbus interface and adapters. PLC gets commands and reference values from an external process control. The synchronous motor gets its main power from the frequency converter and power to excitation from the field exciter. The tested system setup is shown on figure 1.

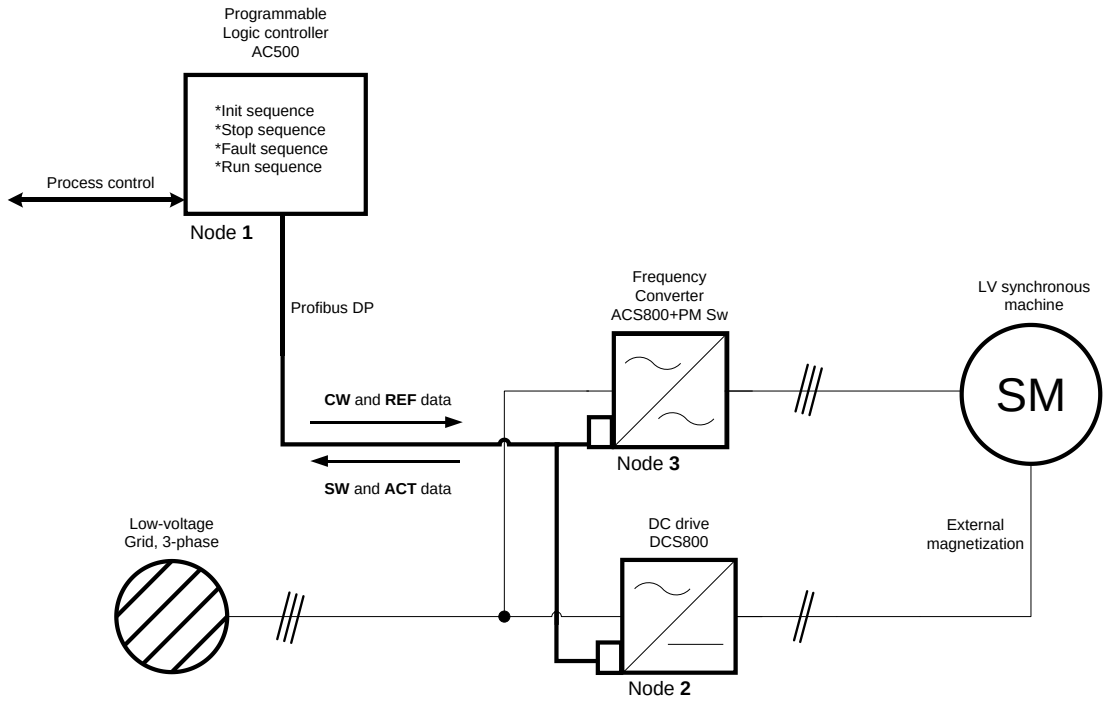


Figure 1. Overview of the planned system. Synchronous motor is excited via a DC drive and frequency converter handles the stator winding. PLC communicates via profibus cable with both drives and controls both drives.

The programmable logic controller (AC500) communicates with both drives and controls both of them. It calculates critical variables and runs state machine, which handles starting, stopping, errors and getting parameters.

The frequency converter (ACS800) can change frequency, current and voltage to fed to the synchronous motor. It excites the stator field into the synchronous motor. With the frequency converter the synchronous motor can be started without secondary starting motor or starting the synchronous motor as an induction motor. Also problems with synchronising motor to power supply's frequency are avoided with this configuration.

Field exciter (DCS800) is used to excite the rotor of a synchronous machine. With DCS800 it is possible to control current remotely from other sources such as from the PLC. In this setup DCS800 only controls field excitation current according to the PLC program and returns the actual current magnitude via the Profibus interface.

2.1 Overview of the tested PLC program

PLC program is created with a CodeSys–software. It has a state machine, which controls starting and stopping of the whole system. RS switch controls the on/off state of motor depending on start/stop -signals. Input handling block handles switches in the user control panel. Reference values are calculated in the fourth network of the program. An Error checker detects faults in system. Selector selects between local- and remote-control modes. Last block takes status from drives and sets it to main status word of the PLC program. Overview of the tested PLC program is shown on Figure 2.

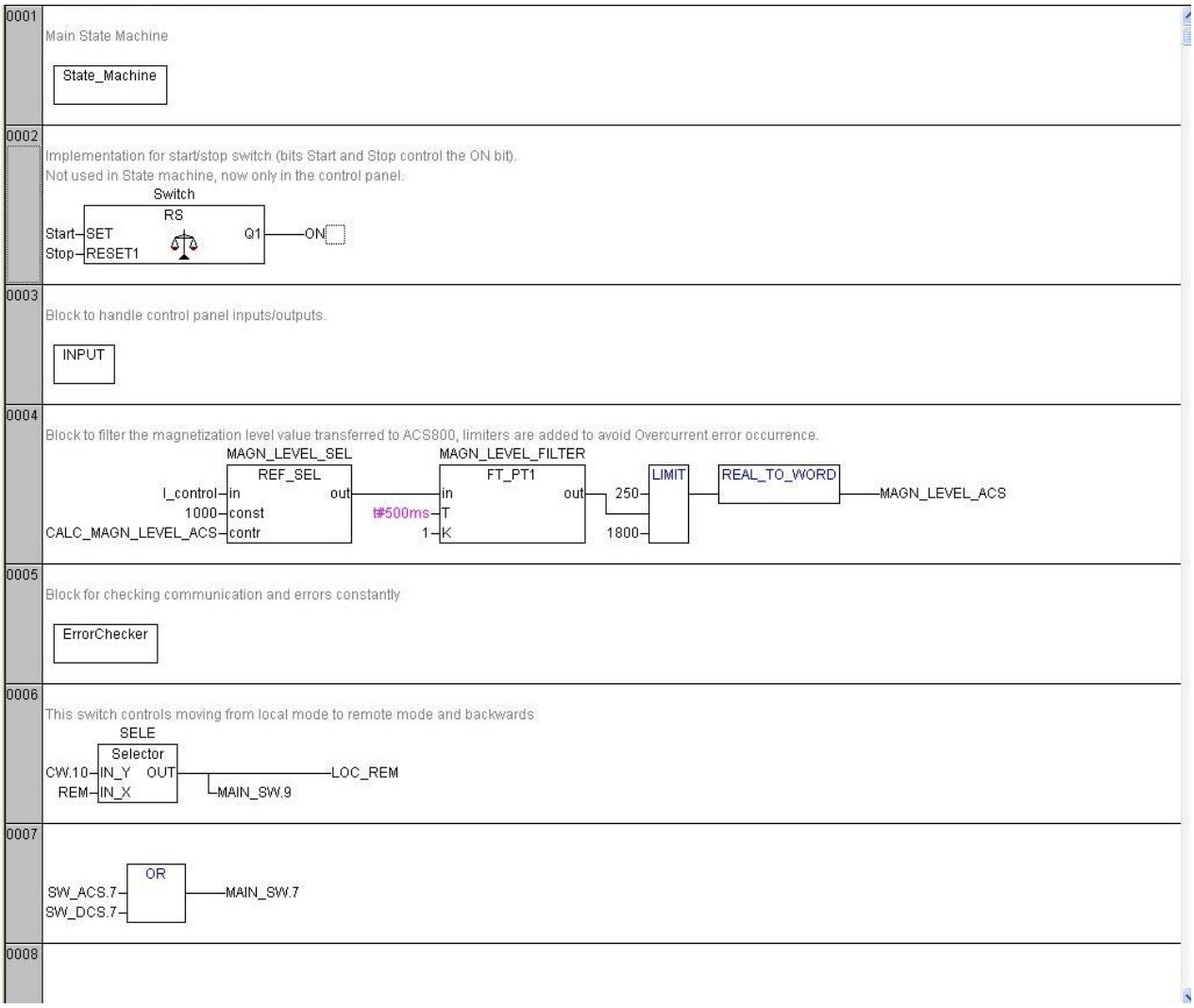


Figure 2. Main program of the PLC. It has total of seven function networks, which handle the whole system.

2.1.1 Main state machine

Main state machine has five states: StartOnInit, Init, Start, Stop and Fault. Each of them has an internal state machine. States in main state machine activate depending on last state and conditions, such as if there is error in device or start signal is given. Main state machine is shown on Figure 3.

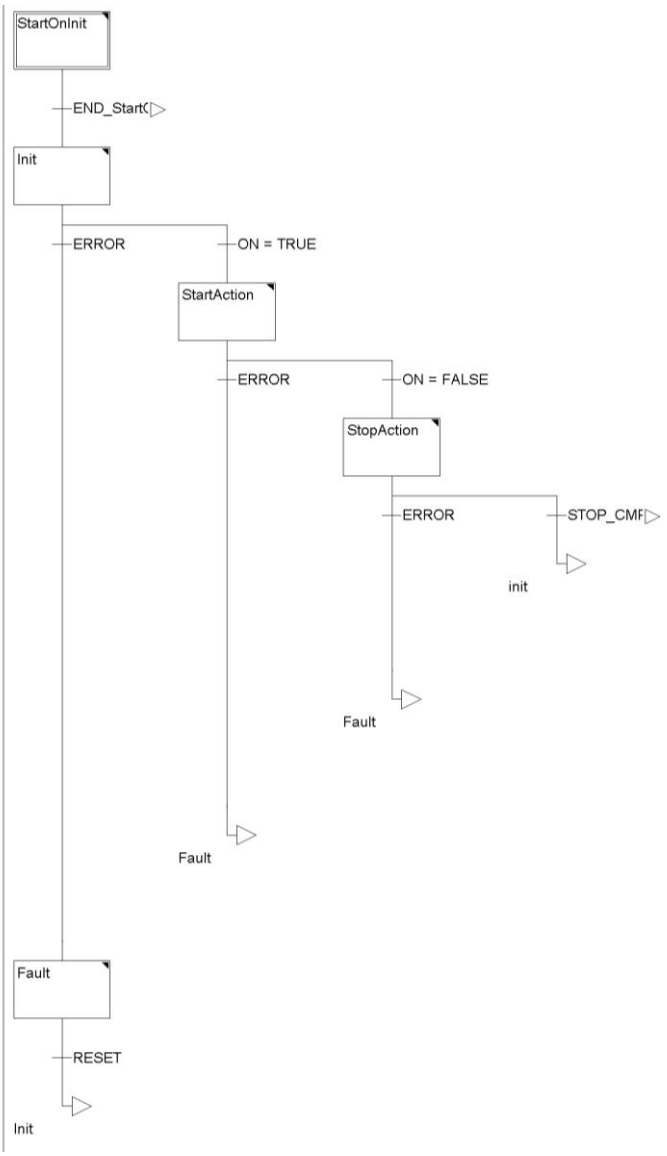


Figure 3. Overview of the state machine. StartOnInit will be run only once in the very start of the program. When it has completed its goal, program moves into Init state. Init state is basic state and it resets all variables, states and conditions. StartAction will start drives and motor. StopAction will stop drives and motor. State machine will end up to Fault state from any state if an error is noticed.

StartOnInit sequence has a state machine and one state for each variable that must be got from drive parameters. Each of them gets one parameter value from ACS800 and savea it as a variable for later use. When all states have got values, StartOnInit sequence is done.

Init state activates after StartOnInit. It has only one internal state, which resets inner variables used in other inner states such as transition conditions, references and state machine states. State machine will stay in the Init-state, until the start signal is given or a fault occurs in either drive.

StartAction state activates after Init state when run signal is given. StartAction has five internal states called as Init, Rdy_On, Rdy_Run, Magnetizing and Rdy_Ref. Init state is a dummy one which doesn't do anything. Any fault or stop order sets StartAction back to internal Init state. Rdy_On is next state after start signal is given. It puts both ACS800 and DCS800 to "Off before on" state which is used before going to "On" state. Rdy_Run is a state where both devices are put on "On before reference" state. After Rdy_Run, StartAction sequence moves to magnetizing state where DCS800 is put on "Run with reference" mode and reference value is given. It will wait until actual value has gone to at least 90% from reference value before setting StartAction to Rdy_Ref state. In the Rdy_Ref state ACS800 is started and reference value can be given for the motor speed or torque. It will stay on this state until stop signal is given or a fault occurs. StartAction state is shown on Figure 4.

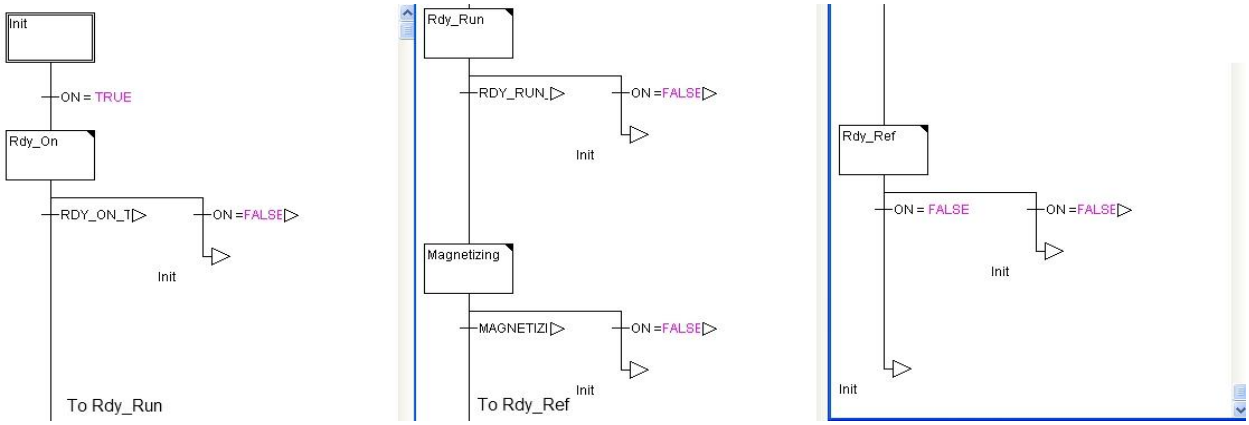


Figure 4. StartAction state machine. Init state is start state and does nothing. Rdy_On state activates when start signal is on. It sets both drives to "off before on" state. After that Rdy_Run state sets both drives to "on before reference". After that magnetizing sets DCS800 to "run with reference" state and starts magnetizing motor. Finally Rdy_Ref sets ACS800 "run with reference" state and starts motor.

StopAction sequence will stop motor and turn off both drives. It is basically the start sequence in opposite direction. First state is a dummy Init state, which doesn't do anything at all. StopAction moves Rdy_Ref state when stop signal is given. Then the ACS800 is soft stopped. After that demagnetizing state references are set to zero and magnetizing is taken off. After magnetizing has gone off, stop sequence moves to next state, Rdy_Run, where both drives are set to "on before reference". And finally both drives are set to "off before on" in next state in Rdy_On state. Last state Stop_completion is technically same state as Rdy_On. It checks same bits as Rdy_On, but it also enables StopAction to move back to Init. StopAction is shown on figure 5.

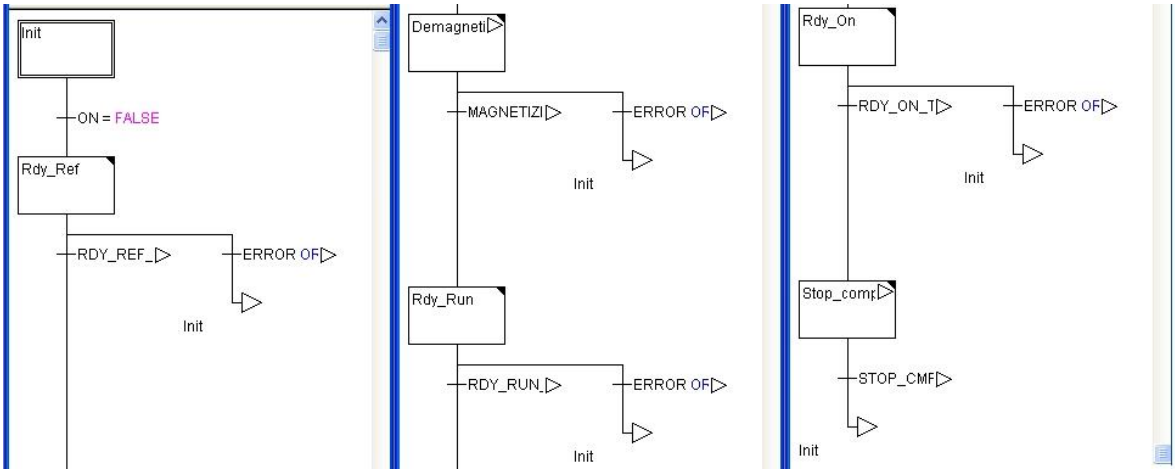


Figure 5. Overview of the StopAction. Init is a dummy state. Rdy_Ref soft stops ACS800. Demagnetizing shuts down the magnetizing. Rdy_Run sets both drives to "On before reference". Rdy_On sets both drives to "Off before on". Stop_completion is merely a checking that both drives are shut down.

Fault sequence is a state, in which the state machine ends up if errors occur during program. It will identify problem and act according to error. It has six internal states. Init checks the error source. ErrorACS, ErrorDCS, ErrorCom and ErrorBoth states will set machines to emergency stop and references to zero. If ACS800 trips, then in the any situation DCS800 is stopped with small delay. End_Fault sets fault sequence to back Init state, ends fault state and ensures that main state machine won't start moving on its own to start motor. Fault state is shown on figure 6.

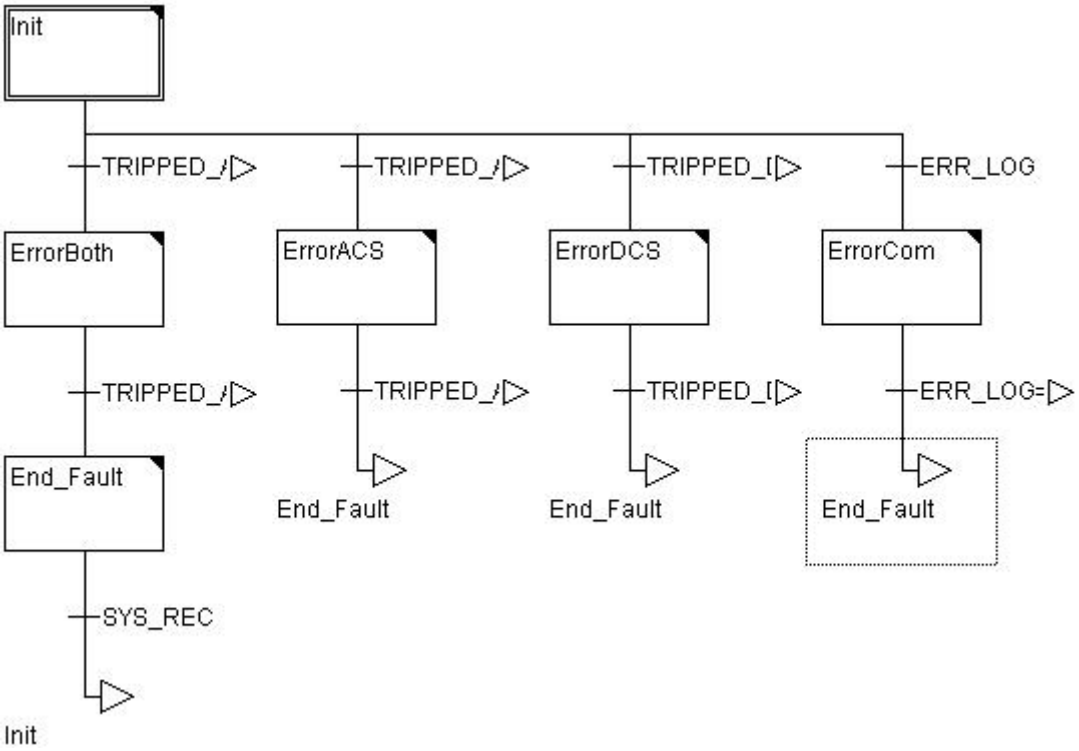
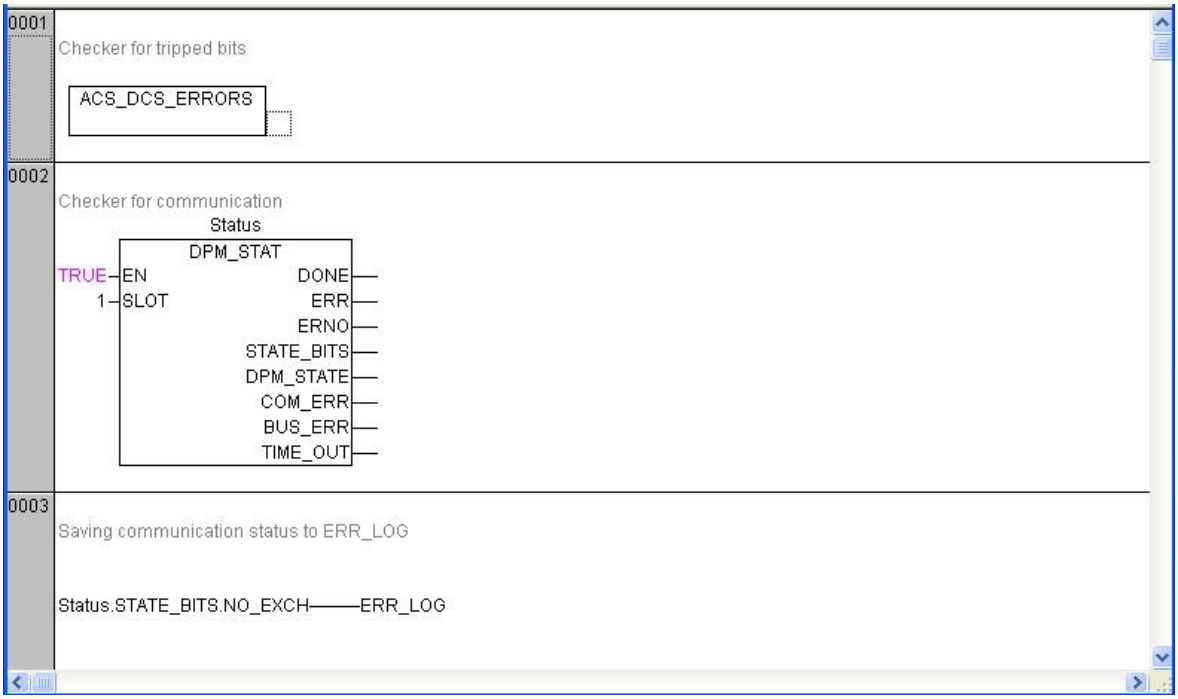


Figure 6. Fault state machine. Init state resets internal state machine conditions to false. Error both handles situation where both drives trip simultaneously. ErrorACS handles situation where only ACS800 gets error. ErrorDCS handles DCS800 errors. ErrorCom handles communication errors in both drives. When errors are reset state machine goes to End_Fault state which ends fault state and sets internal state machine to Init state.

2.1.2 ErrorChecker

ErrorChecker block is keeping eye on tripped bits on ACS800 and DCS800 status words. When the tripped bit activates, ErrorChecker will set condition Error to true. This will set state machine to fault state. ErrorChecker will also look for communication losses. If communication is lost in any state Err_Log condition will be set on true. This will also set the state machine to fault state. ErrorChecker is shown on figure 7.



Fiugre 7. Overview of the ErrorChecker function. ACS_DCS_ERRORS function checks if fault tripped bit in status words rise. Status function is looking for communication errors. If there is communication error then it is saved to ERR_LOG variable.

3. TESTING PLAN DESCRIPTION

System testing is done in four parts. First part focuses on PLC program testing. It is done by a standard systematical program testing with the white box and destructive methods. The general idea is to test each part of the program independently, and then finally whole program. This includes of testing all PLC function blocks independently. If a function block has more blocks inside, like for example the main state machine, all the internal blocks are tested independently, and after that the block is tested in one piece. If any part of the program is not working correctly, it will be fixed and tested until it works correctly.

Second part of the testing plan involves general communication and fieldbus testing. In this phase, the maximum communication speed limits and reliability of communication are tested. If the communication speed is too high, it can be limited to a more suitable rate. Fieldbus testing includes the cable and cable termination testing.

Third part of the test involves functionality testing and overall system testing. Testing involves testing of different torque and speed references and how they affect the system op-

eration. PI controller is tested in this phase. This phase also focuses on finding performance limits of the laboratory motor drive.

Fourth part of the testing covers exploratory testing. Other testing parts rely on systematical testing. In other words, there are strict notes what to test. In the fourth part there aren't any specific notes what to test. This ensures that unlikely errors can be found, since systematical testing cannot cover all possible situations that the system may encounter. Errors found in fourth phase are listed in Appendix III.

3.1 Testing plan for PLC program

PLC program testing is done by destructive and white box methods. Destructive method tries to crash program or cause an unexpected or unwanted action. This is done for example by inputting false or out of range values to the program. In the white box method, an expected command is sent to the program. After that it is inspected, if program did execute command correctly with the supposed outcome. Each function block is tested first with the white box method. After that destructive method is used. Appendix I shows detailed testing plan used for the PLC program.[3][4]

3.2 Testing plan for fieldbus communication

Fieldbus communication testing is done by determining the maximum communication speeds allowing reliable operation of the system. When communication limits are found, system's communication speed is limited to higher applicable speed. Termination settings and their effect on the system communication are also tested in this phase. Testing plan used for fieldbus communications is shown in Appendix II.

4. TEST RESULTS

Actual testing was done according to the testing plan. Appendices I-III show the testing results together with the original testing plan.

4.1 Motor load testing and its results

In the first measurement sequence, synchronous motor was driven in the speed control mode at 750 rpm. Load torque changes from 10 to 50 %, from 50 to 100 % and from 10 % to 100 % were generated with the DC load motor. Torque changes were tested to both directions (e.g. from 100 % to 10 %), and both with a constant excitation current (9 A) and magnetization level (100 %) reference and also with the developed excitation current and magnetization level calculation algorithm.

In all cases, the synchronous motor was running without problems or unexpected shutdowns. Generally the main difference between the constant and controlled excitation was that the controlled excitation allowed the ACS800 speed controller to compensate the load-related rotational speed change. Figures 8a, 8b, 9a and 9b show the operation of the synchronous motor with 10%→50%→10% load torque chances without and with the use of excitation control algorithm.

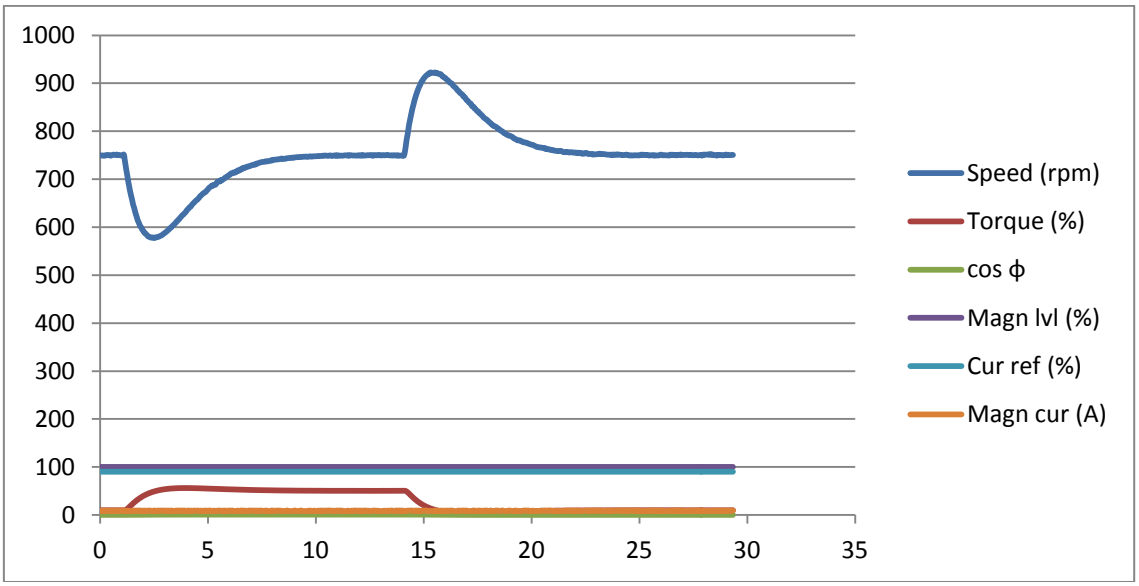


Figure 8a. Effect of 10%→50%→10% load torque change on the motor operation, when constant excitation current (9 A) and level (100%) were applied.

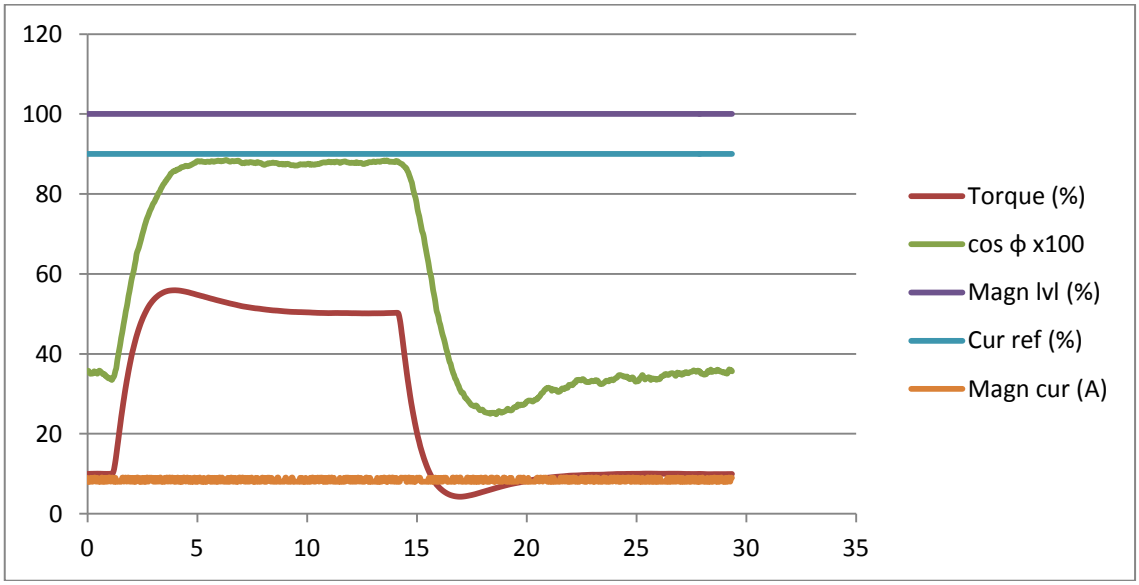


Figure 8b. Effect of 10%→50%→10% load torque change on the motor operation without speed, when constant excitation current (9 A) and level (100%) were applied.

In Figures 8a and 8b is a test without use of the excitation control algorithm. Speed changes when load is applied or taken off. Torque shows used torque and torque steps. Magnetization level and current reference will stay constant due to excitation control algorithm is not used. Excitation current stays near 9 A. Power coefficient follows torque.

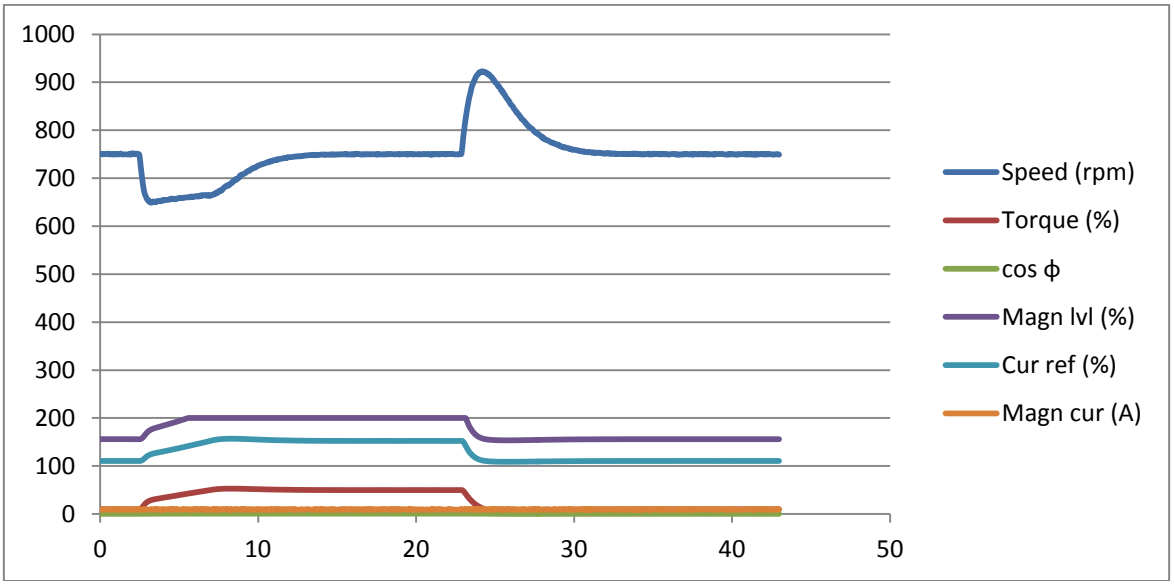


Figure 9a. Effect of 10%→50%→10% torque reference change on the motor operation, when excitation current and level were controlled by AC500. Here the magnetization level was calculated according to the calculated current reference.

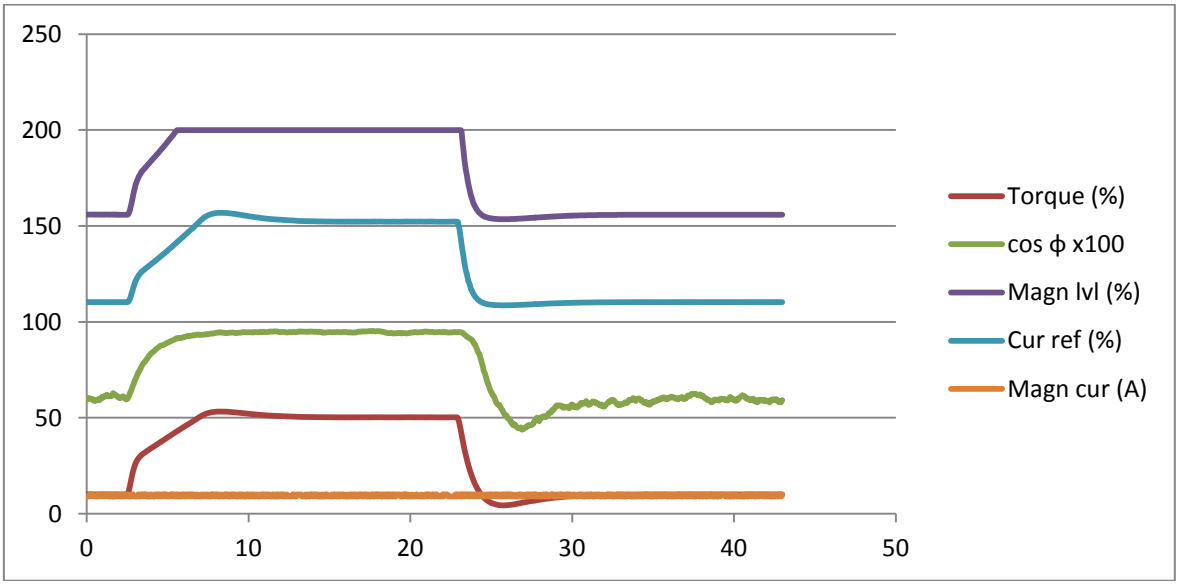
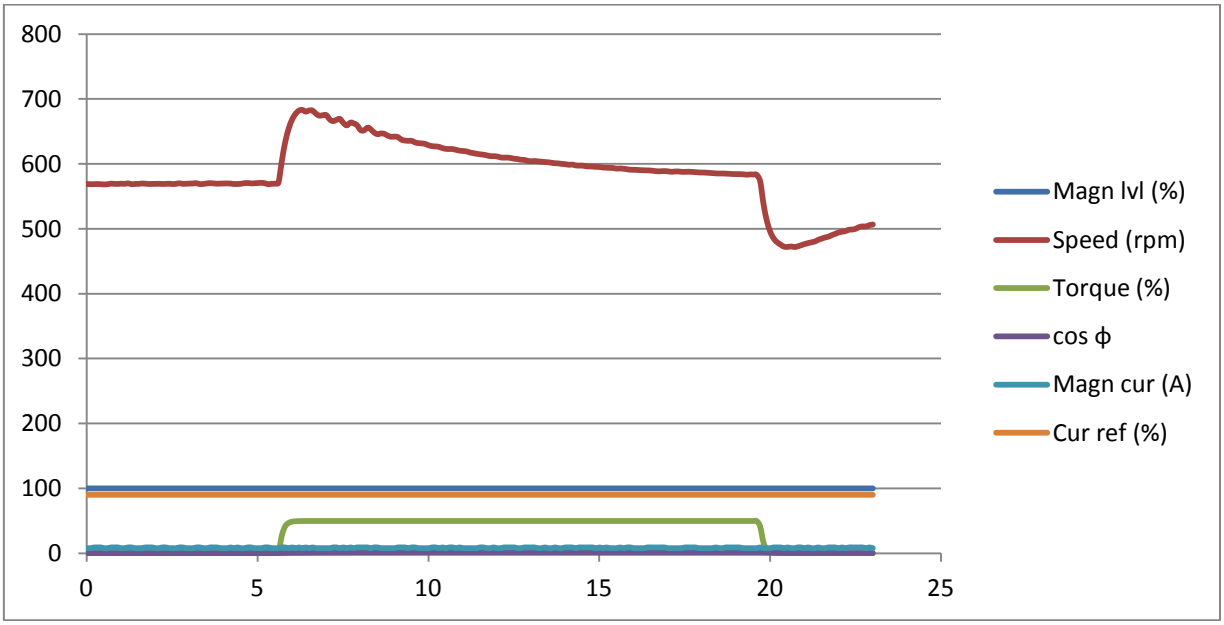


Figure 9b. Effect of 10%→50%→10% torque reference change on the motor operation without speed, when excitation current and level were controlled by AC500. Here the magnetization level was calculated according to the calculated current reference.

In Figures 9a and 9b is a test with excitation control algorithm. Speed is compensated when load is applied. Due to used excitation control algorithm speed isn't compensated when load is taken off. Excitation control algorithm tries to control magnetization level and current reference and thus they follow torque changes. Power coefficient changes along other factors.

Second set of tests comprised driving the synchronous motor with the torque control mode. Also in this case the use of excitation current and level calculation algorithms did not cause unexpected failures or other shutdowns. The most notable difference in the measurement results was the AC500's attempt to adjust the excitation current and level according to the load torque change. Pictures 10a, 10b, 11a and 11b show the operation of the synchronous motor with 0%→50%→0% torque reference changes without and with the use of excitation control algorithm.



Figurere 10a. Effect of 0%→50%→0% torque reference change on the motor operation, when constant excitation current (9 A) and level (100%) were applied.

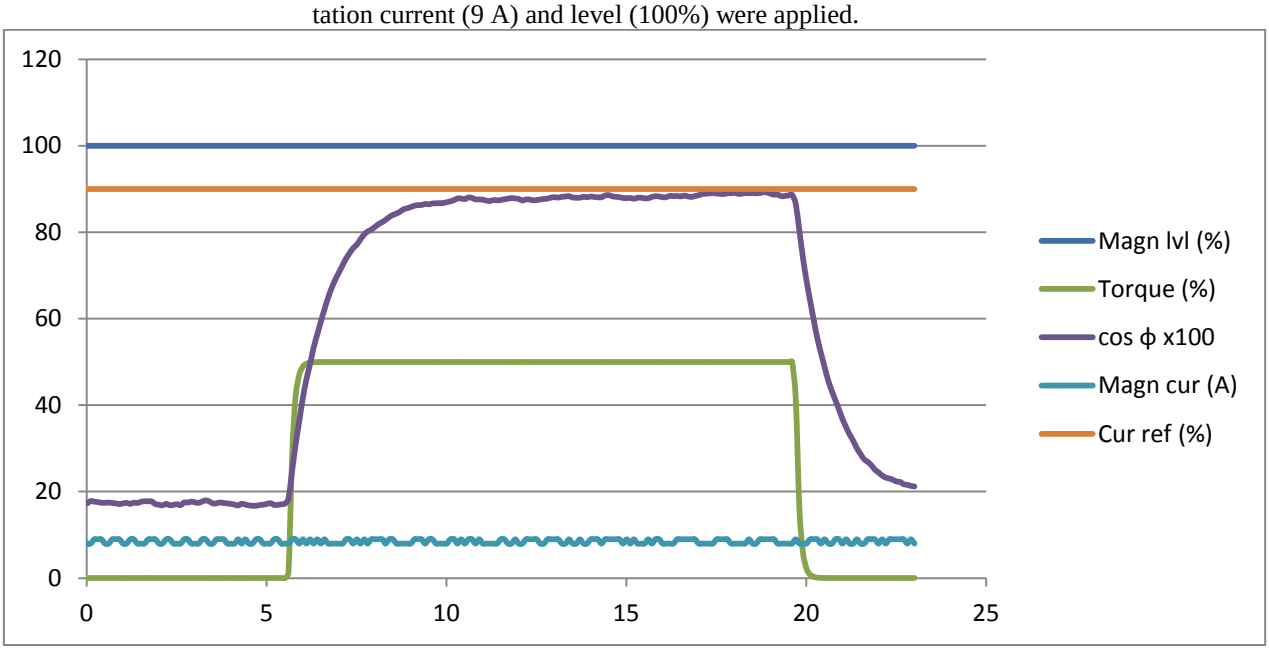
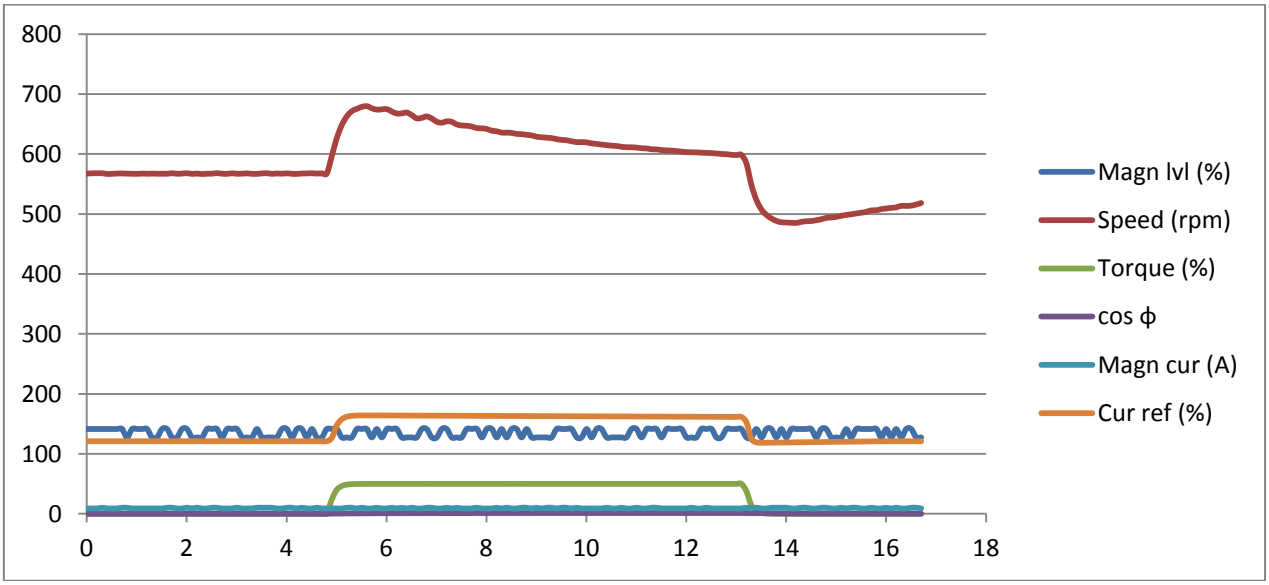
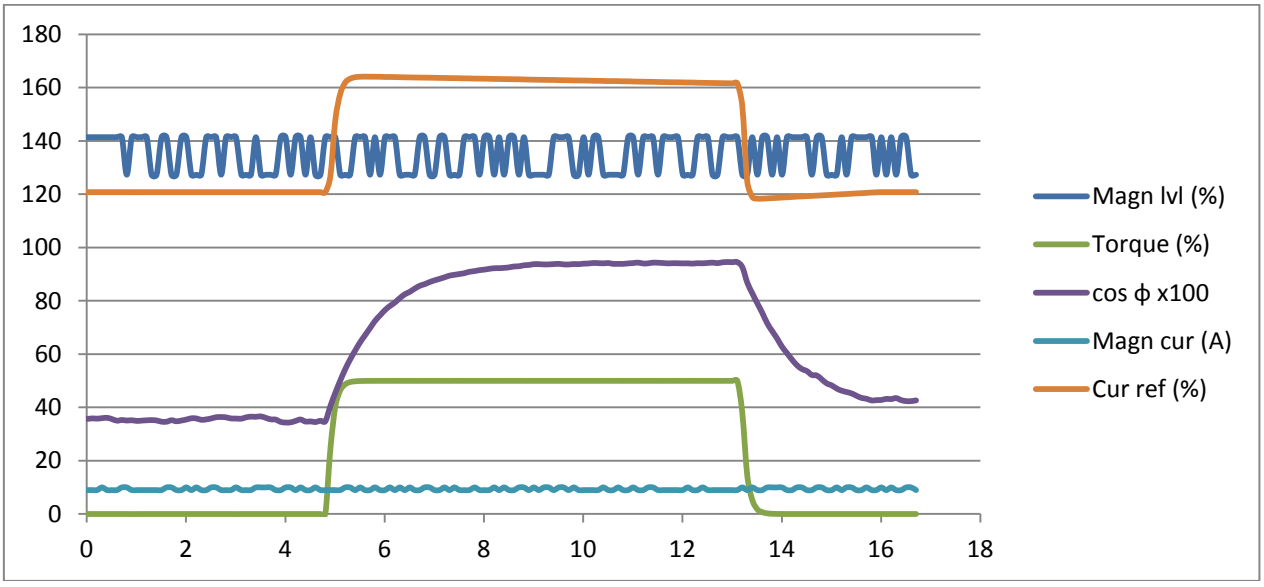


Figure 10b. Effect of 0%→50%→0% torque reference change on the motor operation without speed, when constant excitation current (9 A) and level (100%) were applied.

In Figures 10a and 10b is a test with torque control and without usage of excitation control algorithm. Speed follows torque changes. Magnetization level and current references stay as constants. Power coefficient follows used torque. Excitation current jumps between 9A and 10A due to technical limitations on excitation unit.



Picture 11a. Effect of 0%→50%→0% torque reference change on the motor operation, when motor current and magnetization level were controlled by AC500. Here the magnetization level was calculated according to the actual current generated by DCS800 (max. 10 A).



Picture 11b. Effect of 0%→50%→0% torque reference change on the motor operation without speed, when motor current and magnetization level were controlled by AC500. Here the magnetization level was calculated according to the actual current generated by DCS800 (max. 10 A).

In Figures 11a and 11b is a test with torque control and with usage of excitation control algorithm. Current reference follows torque change. Magnetization level jumps between 140% and 130% due to used algorithm. Power coefficient follows torque changes. Excitation current jumps between 9A and 10A due to technical limitations on excitation unit.

4.2 Communication limits

Communication limits were tested only at the highest speed. Profibus cable can transfer data at 12 MBit/s at maximum and this rate was used in tests. Communications between drives were reliable, so there was no need to limit communication speed between devices.

Communication speed between PC and AC500 was also tested at the speed of 19600 Bit/s. Communication was reliable at that speed. It might be possible to use even higher communication speed to configure the PLC, but this was not separately tested.

5. CONCLUSIONS

Results of the testing shows that system works correctly and the testing as a whole was successful. All the aspects which had to be tested were tested. PLC program parts like state machine and error checker work correctly. Communication limits are at 12 Mbit/s between ACS800 and AC500. Between DCS800 and AC500 communication limit is 12 Mbit/s. PLC program works correctly and it doesn't have fatal errors. State machine works correctly with both remote and local control mode. Program reacts to errors and doesn't cause damage to drives.

In future it is very possible for me to utilize things which I learned from this bachelor thesis. I am now more familiar with processes in electric drives, like for example how to design, build, manage and test electric drives.

SOURCES

[1] Ari Potinkara. Bachelor's Thesis: The setup and requirements specification for automation concept of externally excited synchronous motor

[2] Lauri Niinimäki. Bachelor's Thesis: Automation Concept for Electrically Excited LV Synchronous Motor: Implementation into AC500 Programmable Logic Environment

[3] Exploratory Testing, Cem Kaner, Florida Institute of Technology, *Quality Assurance Institute Worldwide Annual Software Testing Conference*, Orlando, FL, November 2006 26.8.2011

[4] http://agile.csc.ncsu.edu/SEMaterials/WhiteBox.pdf 26.8.2011

APPENDICES

APPENDIX I	PLC program testing sheet
APPENDIX II	Profibus communication testing sheet
APPENDIX III	Exploratory testing sheet

APPENDIX 1.1				
Version 3.0				
PLC Program			Tested	Testing results
Testable Function Block		Tested content		
	ACS_DCS_Errors			
		Tested in Er-ror_Checker	x	
	Control_Panel			
		Inputs right control words in local con-trol mode	x	
	Error_Checker			
		Notices errors ACS800	x	
		Notices errors from DCS800	x	
		Notices com-munication losses	x	
	FAULT_Sequence			
		Handles errors from DCS800	x	
		Handles errors from ACS800	x	
		Handles errors from both drives	x	
		Handles errors from communica-tion loss	x	
		Fault_Sequence state machine	x	
	Init_Sequence			

		Resets all necessary conditions	x	
	Input			
		Switches work correctly	x	
		Values work correctly	x	
	Magn_Calc			
		Mathop-Function block works correctly	x	
		PI-controller working correctly	x	
		Magnetation calculation working correctly	x	
	Mathop1			
		Tested in Magn_Calc	x	
	PLC_PRG			
		State_Machine-block is working when program starts	x	
		Switch for ON-signal works correctly	x	
		Input-block starts when program starts	x	
		Error_Checker starts correctly	x	
		Local/remote-switch working correctly	x	

	Selector			
		Selects recently changed component	x	
	Start_Sequence			
		Goes back to Init after sequence is done or error occurs	x	
		Doesn't get stuck on states	x	
		States handle their objectives	x	
	StartOnInit_Sequence			
		Gets parameters from ACS800	x	
		Doesn't get stuck on states	x	
	State_Machine			
		Transition conditions work correctly	x	
	Stop_Sequence			
		Goes back to Init after sequence is done or error occurs	x	
		Doesn't get stuck on states	x	
		States handle their objectives	x	

Testable content	Tested	Testing results
Fieldbus connection		
Termination		
RBPA-01 Adapter	x	End line Termination must be "on"
Profibus	x	End line Termination must be "on"
From AC500 to DCS800 max speed	x	12 Mbit/s was maximum
From AC500 to ACS800 max speed	x	12 Mbit/s was maximum
From AC500 to Process Control max speed	x	19600 bits per second was tested

Exploratory testing		
Version 3.0		
Testable content	Tested	Testing results
Error causes problems with PI-controller	x	
can't start motor with PI-controller	x	Causes overcurrent-error
Rem -> loc, cant start before stop	x	Semi-intended feature