CODEN:LUTEDX/(TEIE-5274)/1-88/(2011)

Industrial Electrical Engineering and Automation

Development of Wind Power Laboratory Setup

Caspar Ralvenius

Division of Industrial Electrical Engineering and Automation
Faculty of Engineering, Lund University

Abstract

Wind turbines are often used on weak grids as a source of active power. There is a desire for a continuous supply of active power during variable wind speeds while also supplying reactive power to suppress grid-voltage fluctuations. These objectives are attainable using converters which are controlled with regulators.

In this project a test environment will be constructed wherein a wind turbine will be modeled and controlled using the aforementioned regulators. The wind turbine model is of a Full Rated Converter wind turbine.

The control algorithm for the regulators is developed in C and ran from a digital signal processor.

Table of Contents

1. Introduction

The primary sources of electricity in the world are fossil fuels, corresponding to 70 % of the world electric energy production. The reason for their domination is the accessibility of fossil fuel and the high energy density when combusted. Since fossil fuels need several millions of years to form, they are considered a non-renewable energy source. This entails that at the current rate of exploitation; the sources will be depleted before they manage to form anew to meet the growing energy demands.

There is a general consensus that the combustion of fossil fuels is detrimental to the environment as the oxidization process is contributing to the greenhouse effect and the dispersing of harmful air pollutants. As citizens in industrialized countries are becoming increasingly environmentally conscious energy consumers, the demand for cleaner energy sources are growing.

In order for the alternative energy sources to be viable they have to meet certain criteria; cost competitiveness, reliability, environmentally safe and renewable energy production. Currently the dominating sources for renewable electric energy generation are hydro, solar and wind. Wind energy is one of the most exploitable of the before mentioned since it can be harnessed at an acceptable cost in both a large and a small scale. The environmental intrusion is generally lower than for hydropower plants as the main issues with wind turbines are the visual aspect while hydropower requires large damns and the cost is lower than the solar plants.

During 2009 wind power experienced a yearly growth of 38 GW or 31.7 % of capacity, bringing the total capacity to 158.5 GW. Globally the wind turbines generated 340 TWh of electricity equaling 2 % of the global electricity consumption. The trends show continuing investments in wind power and increasing installed capacity per annum [1].

1.1 Objectives

The projects aims to construct a test environment where a wind turbine system can be modeled and controlled in order to feed active and reactive power out on the grid. The test environment will be used for further research concerning converter control and tests.

1.2 Thesis outline

Chapter 1 provides the introduction to the thesis and the aim of the project. In chapter 2 theory for wind energy will be presented. The control strategy of the FRC is explained in chapter 3. Chapter 4 will describe the voltage source converter control strategy. Chapter 5 will describe the laboratory setup and the control structure of the power electronics. Chapter 6 will introduce the software used to control the converters. Chapter 7 will present the experiment. In chapter 8 the result will be presented and finally in chapter 9 conclusions will be drawn from the experiments.

The appendix will contain the program code, the process of flashing the DSP and CAD schemes of a adjustment card for the signals entering the DSP AD converter.

2 Wind energy

Wind flows due to atmospheric convection caused by the uneven heat patterns of the earth's surface causing different atmospheric pressures. The energy in the wind is harvested by wind turbines as the wind passes the airfoil structured blades. Airfoils are designed with a geometrical shape to generate mechanical forces as the surrounding fluid moves relative to the airfoil causing a lift and a drag force on the airfoil structure. These forces generate a net positive torque on a rotating shaft which drives an electrical generator to generate electrical energy.

Figure 1 illustrates the principle of how the wind flow is disturbed passing the wind turbine wing blade in its path. The airfoil geometry is more closely analyzed in figure 2. Facing the wind flow direction is the leading edge of the airfoil. It cuts through the wind and forces a portion to travel below to the *lower chamber* and above to the *upper chamber*.

According to the law of continuity, the wind molecules separated by the leading edge will conjure at the same point leaving the airfoil. This phenomenon will create a pressure difference at the airfoil chambers, as the travel distance for the molecules passing the upper chamber is longer than for those passing the lower chamber. As the velocity of air increases, pressure decreases resulting in a lift force on the airfoil due to the higher pressure of the lower chamber [2].

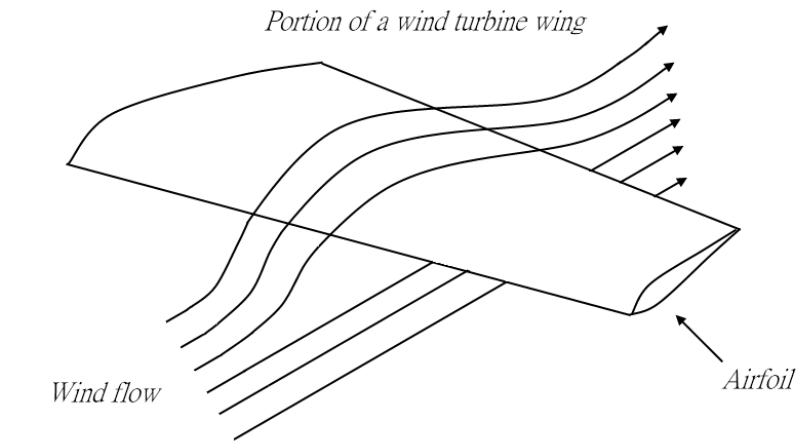


Figure 1 The wind flow is disturbed by a wind turbine blade designed to create a lift and a drag force in order to drive the rotating shaft connected to the generator.

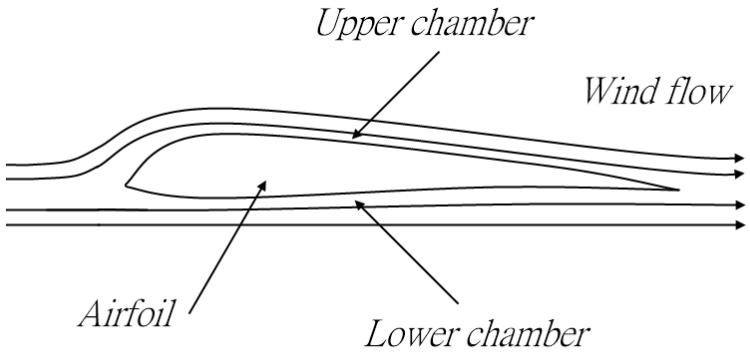


Figure 2 ***Typical airfoil principle where the wind flow is less obstructed at its lower part than it's upper to create pressure differences.***

2.1 Wind turbine

The currently prevalent wind turbine design is the horizontal axis wind turbine, which has its rotating axis in parallel to the horizon. Apart from the axis configuration, wind turbines can be classified depending upon if they have the rotor oriented in upwind or downwind, the design of the hub, rotor control operation and the alignment with the wind direction. The principal components of modern electricity generating wind turbine are (figure 3):

- Rotor; blades and supporting hub
- Drive train; generator, converters, gearbox, shafts, sensors and breaks for control.
- Wind turbine housing; nacelle cover
- Main frame/ yaw system to keep the wind turbine properly aligned with the wind direction
- Tower and tower foundation
- Balance of electrical system; cables, transformers, switchgear and converters and machine control.

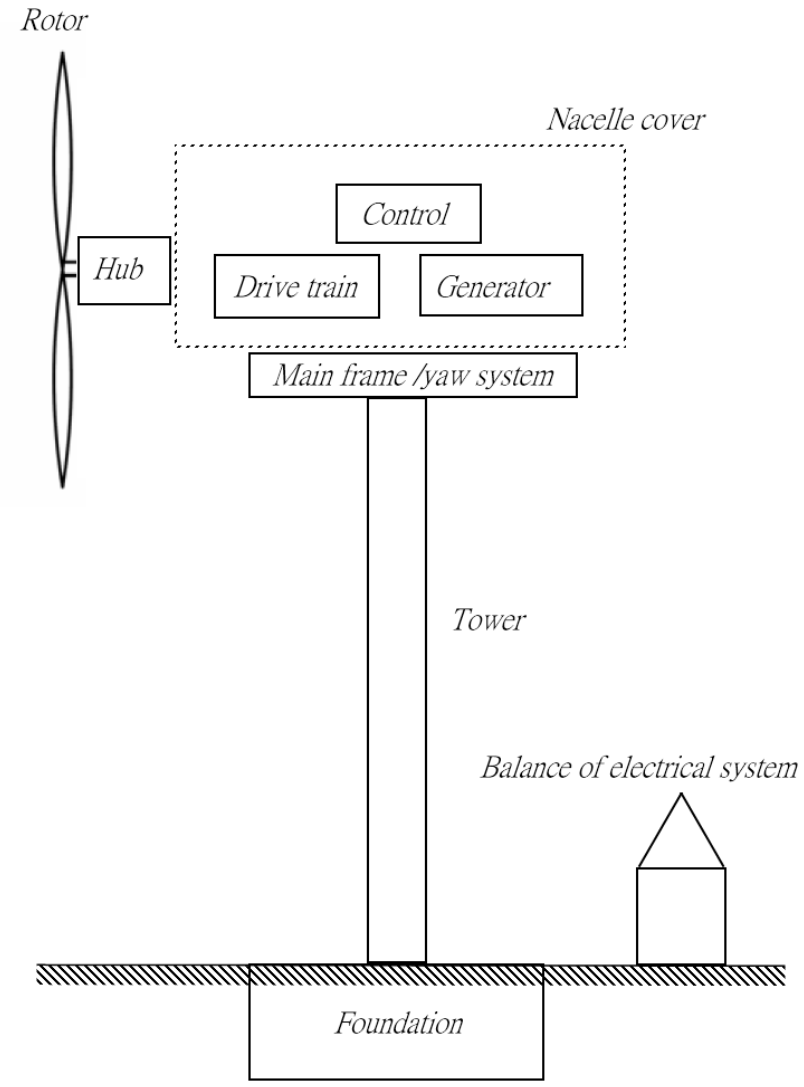


Figure 3 *The principle of a horizontal axis wind turbine.*

2.2 Energy in wind

The kinetic energy in a fluid passing the rotor blades is defined by:

$$P_{fluid} = \frac{1}{2}mv^2$$

In this equation *m* is defined as the weight of the fluid passing the turbine's rotor per second and *v* the velocity of the fluid. The mass of the fluid can be calculated multiplying the density of the fluid with the volume, which is defined as the rotor blades sweep area multiplied with the distance the fluid travels in one second. This yields the following equation:

$$P_{air} = \frac{1}{2}\rho A v^3$$

Where ρ is defined as the density of the wind which is approximately 1.225 kg m^{-3}, *A* is the area swept and *v* is the velocity.

The equation only serves to represent the kinetic energy in the wind passing the wind turbine; the actual energy conversion from wind to mechanical power is defined by the *Betz limit*. The Betz limit derives a power coefficient between the power in the wind and the power on the rotor:

$$C_p = \frac{P}{\frac{1}{2}\rho A v^3} = \frac{Power\ on\ the\ rotor}{Power\ in\ the\ wind}$$

The power coefficient, C_p is derived to 0.5926 which defines the ideal wind turbine performance conversion rate. In practice the conversion rate is lower due to inhomogeneous air flow, frictional drags on the blades, finite numbers of blades and the wind rotation caused by the blades, resulting in values from 0.25 to 0.45 [2] [3].

2.3 Tip speed ratio

The power coefficient is at its maximum at a certain velocity of the wind turbines blade tip relative the velocity of the surrounding wind. The tip speed ratio, λ defines the ratio between the parameters:

$$\lambda = \frac{\omega R}{v}$$

Where ω is defined as the rotational speed of the rotor, *R* is the radius of the rotor and *v* is the speed of unobstructed wind flow.

The power output is highly dependent on the wind tip speed ratio (Figure 4). Adjusting the wing tip velocity relative to the surrounding wind velocity will help to maintain a high power coefficient. Wind turbines designed to operate at a single wind speed will therefore inherently have a lower power conversion rate than those operating at variable rotational speeds as the wind velocity is in its nature non-constant [3].

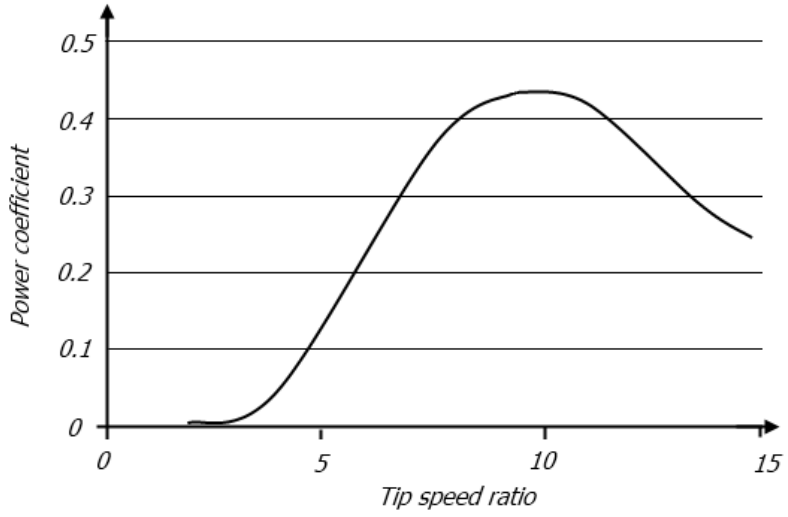


Figure 4 *The relation between tip speed ratio and power coefficient*

2.4 Wind turbine power curve

The wind turbine power output is dependent on various mechanical and electrical components performance. To be able to predict the power output during varying wind speeds, field test are undertaken to determine the wind turbines power curve [3].

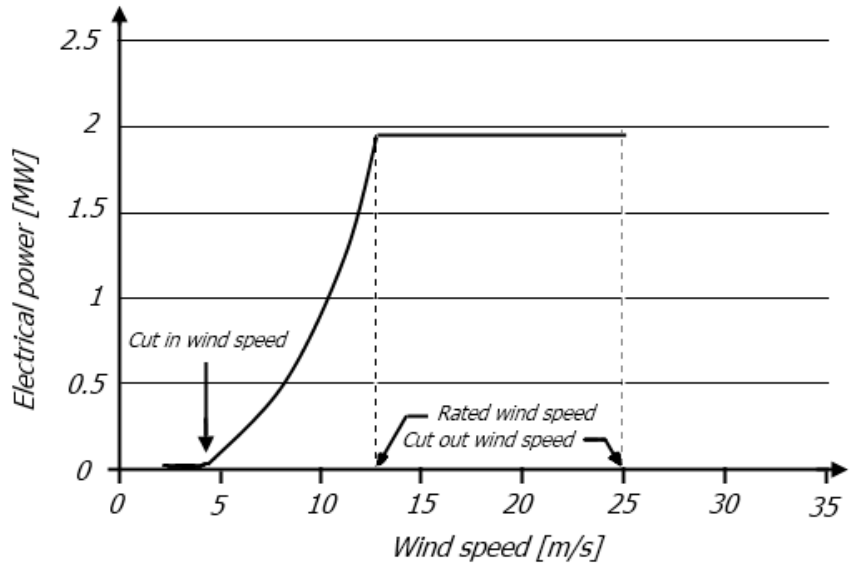


Figure 5 *Wind turbine power curve for a 2MW wind turbine*

The key points determining the wind turbines power curve are the:

- Cut-in speed; the minimum wind velocity that the wind turbine can generate useful power
- Rated wind speed; the velocity of the wind where the wind turbine generator produces its rated output power.
- Cut-out speed; the maximum wind velocity at which the wind turbine will deliver power, which is limited by mechanical and electrical constraints.

2.5 Wind turbine architecture

The wind turbine can be designed in various ways depending on its intended purpose. Currently, the most widely commercially produced wind turbine for electricity generation is designed with a three bladed rotor on a horizontal axis designed to face the wind direction. Wind turbines are thereafter constructed depending upon whether they will operate at varying-speeds or fixed-speeds.

2.5.1 Fixed-speed wind turbines

Fixed-speed wind turbines are designed to rotate at a constant speed, generating electricity at a constant frequency to match the grid frequency. Since the wind turbine velocity operating range is limited it simplifies the electronic components. The fixed-speed wind turbine can use the relatively low cost squirrel cage induction generator. To compensate for the reactive power the generator consumes, the wind turbine is fitted with a capacitor bank. A soft-starter is necessary to minimize the current transients when the generator is energized.

The mechanical part needed are a low-speed shaft connected to the wind turbine rotor and a gearbox connecting it to a high-speed shaft driving the generator. The gearbox increases the rotational speed of the rotor allowing for a lower number of poles in the machine which decreases the cost, physical dimensions and weight. The disadvantage is the added weight to the tower, the acoustic noise and the required maintenance of the gearbox [2].

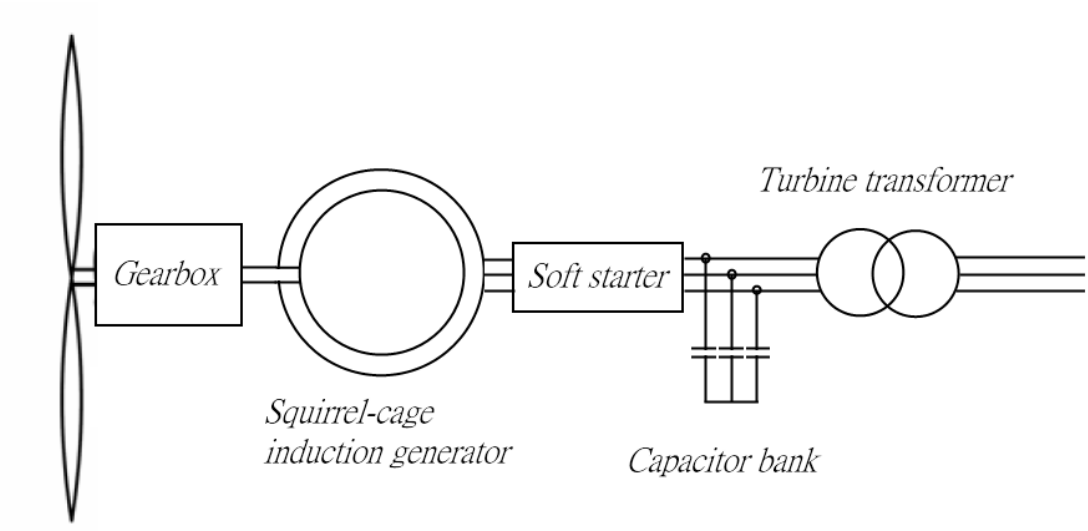


Figure 6 *The schematics of the fixed-speed wind turbine components*

2.5.2 Variable-speed wind turbines

Variable-speed wind turbines are desirable because they can generate electricity at its rated value below rated wind speed as long as the tip speed ratio is kept constant by regulating the rotor speed. This architecture will generate electricity at a high efficiency on a wide range of speeds.

Variable speed operation is possible using synchronous and induction generators. These are implemented in doubly fed induction generator (DFIG) wind turbines and fully rated converter wind turbines (FRC). The DFIG uses the Wound Rotor Induction Generator (WRIG) and FRC uses the Synchronous Generator (SG) or the Squirrel Cage Induction Generator (SQIG).

2.5.2.1 Variable-Speed Operation using Synchronous Generator

The synchronous generator can either have permanent magnets or separately excited windings creating the magnetic flux. In both cases, the frequency is dependent on the rotational speed and the number of poles and will diverge during variable speeds from the relatively static grid frequency. Variable-speed operation thus requires the output to first rectified to DC and then back to AC at grid frequency.

The wind turbine consists of the wind turbine rotor, SG, AC/DC rectifier, DC link, DC/AC inverter connected to the grid. Depending on the design, there is an option to include a gearbox connecting the wind turbine and the SG. A generator with a multiple pole design can generate power at a sufficiently high frequency without the necessity for a gearbox. The multiple pole solution has the drawback that the generator has larger dimensions and that the mechanical torque on the shaft will need to be larger to generate power at low speeds [2].

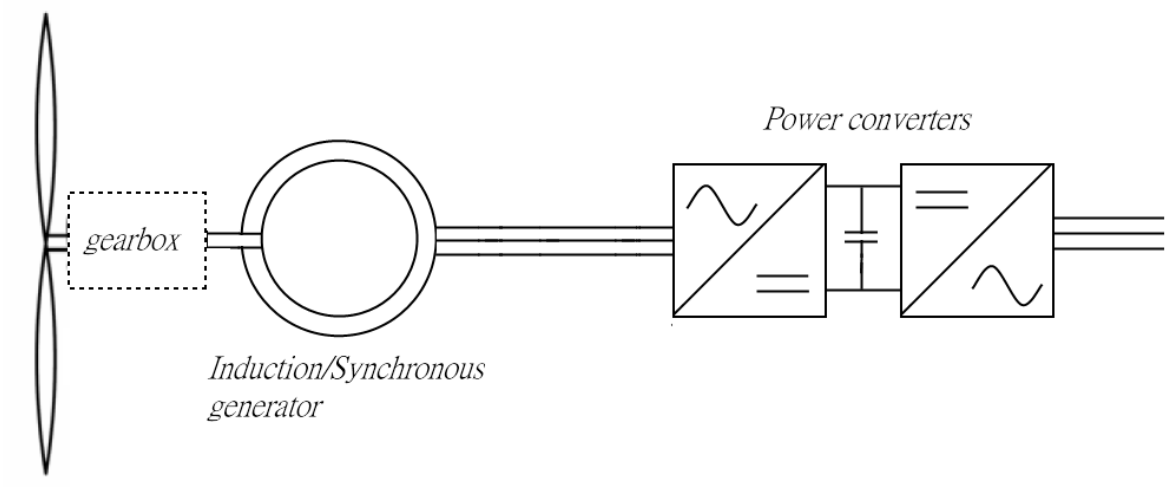


Figure 7 ***The schematics for both the induction and synchronous generator wind turbine architecture***

2.5.2.2 Variable-speed Operation using Squirrel Cage Induction Generator

SQIG driven wind turbines require electrical converters that can supply reactive power and control its rotational speed. The cost of the electrical converters and losses in the system will increase compared to the synchronous generator fitted wind turbines. The gains of the rugged and compact induction generator compared to SG fitted wind turbines are therefore relatively small.

The SQIG driven wind turbine typically consists of the wind turbine rotor, SQIG, voltage source converters (VSCs) with a DC link between them connected to the grid (figure 7). The generator side VSC supplies the SQIG with reactive power and receives active power from it. The grid side converter inverts the DC to AC at grid frequency at an appropriate voltage level.

2.5.2.3 Variable-speed Operation using Wound Rotor Induction Generator

The WRIG operates similarly as the SQIG with the exception that it has copper wire windings for a rotor, rather than metal bars as the SQIG has. The implication is that the rotor can have both power injected and extracted from it using brushes and slip rings connected to the rotor.

Although the WRIG is more expensive than the SQIG, the accessibility of the rotor enables the power converters to only handle approximately one third of the power compared to SQIG driven variable-speed wind turbines. This will lower the overall cost and size of the wind turbine configuration.

As the rotor windings are accessible, the rotor the rotor slip and consequently the rotor current can be increased and decreased, allowing the generator to operate at wider velocity ranges. Increasing the rotor resistance will increase the losses and decrease the overall efficiency unless the power can be recovered, as in the DFIG.

If the wind turbine experiences higher wind speeds (super-synchronous), the excess power will flow out from the rotor and via power electronics be fed to the grid at proper frequency and voltage. If the wind speeds decrease (sub-synchronous) the power converters will absorb power from the grid and feed it to the rotor allowing it to still generate power fed to the net.

The WRIG driven wind turbine consist of the wind turbine rotor, WRIG, two VSCs with a DC link connected to the grid via a transformer. Depending on the wind turbine speeds, the grid and the generator side converters will either operate as rectifiers or as inverters. The arrangement allows the DFIG to operate at approximately 50 % below and 50 % above the synchronous speed [2] depending on the converter dimensions.

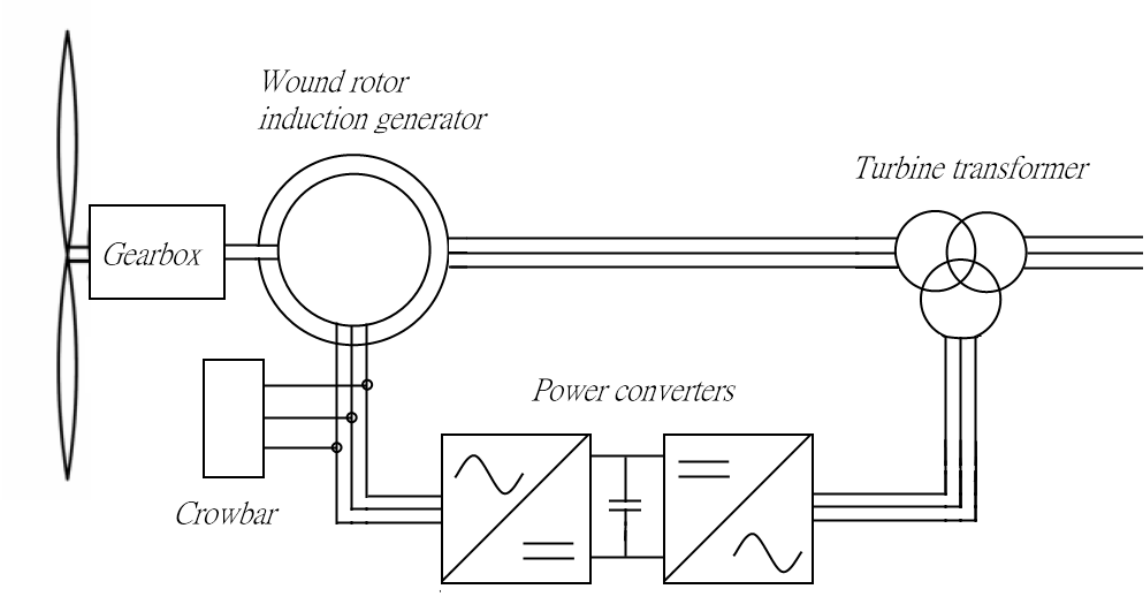


Figure 8 *The schematics for the DFIG wind turbine architecture*

2.6 Desired influence on the power grid

The wind turbine will supply the network with active power from wind energy as it's converted kinetic energy is converted via the wind turbine rotor to electrical energy using the electrical generator.

The grid code regulations vary from country to country and even from region to region specifying the frequency and voltage level for continuous operation. The code also stipulates operation during reduced grid voltage before the wind turbine is tripped off from the grid, to help sustain the grid voltage (fault ride through).

3 Wind power generation

This chapter describes the fully rated converter (FRC) wind turbine architecture, the principle as well as the control strategy. The power electronics involved in controlling the active power flowing out from the electrical generator as well as reactive power control on the grid side will also be explained.

3.1 Power electronics for wind turbines

Power converters are versatile electrical instruments that enables the electrical power to change voltage and frequency, as well as form; AC to DC and DC to AC. Due to their wide possible applications, the power converter cost is constantly decreasing making it economically viable to install in applications such as wind turbines.

Modern power converters consist of transistors acting as valves, either fully open or fully closed, i.e. either working as closed circuits or as an open circuit. The output; voltage time area is commonly controlled using carrier wave modulation, where a signal generators triangle wave output is compared to a reference wave switching the transistors according to the comparison results.

Transistors are favorable due to the low voltage drops and the high switching frequency, containing the losses to approximately 4 % of the transferred power. Currently the most commercially common transistor used in power converters is the insulated gate bipolar transistor (IGBT), its characteristics are high efficiency and fast switching making them attractive for power electronic switches.

In variable-speed wind turbines, power converters are used to decouple the electrical generator from the grid as the frequency and the voltage at the generator terminals will not be constant due to wind speed variations.

3.1.1 Voltage source converters

Voltage source converters (VSCs) are widely used in power systems. The principle of a VSC can be described with a single four-quadrant DC converter (consisting of two transistor half-bridges). In the two quadrant DC converter, the current can flow in both directions, out from the DC source, and into the DC source. The output average voltage from the four-quadrant DC converter can be both negative and positive with reference to the mid-point in the DC link [5].

Three transistor half bridges can form a three-phase converter if they are connected in parallel, with each half-bridge controlled independently. The transistors in a half-bridge are supposed to switch alternatively with one transistor conducting at a time however transistors are unable to transverse from closed to open circuits instantaneously.

There is therefore an inherent risk when switching for a short circuit in this form, there is a time delay imposed; blanking time, so that one half bridge leg in the VSC can't conduct with both switches simultaneously. There are different solutions to this problem, the principle is usually to delay the transistor on state (conducting) while its off state (open circuit) isn't. The ramification is that the average output voltage will be lower, demanding the need for an integral part in converter controllers. The principle used in this report utilizes the possibility to adjust both the on and off state in the software to avert short circuits.

The strategy used in this report to control the VSC is the carrier based pulse width modulation (CB-PWM). The principle behind the control method is that a reference sine wave of a desired frequency

is compared with a fixed high-frequency triangle wave to obtain voltage pulses of desired width. This will be described in chapter 4.

3.1.2 Back to back VSCs

Variable wind turbines are fitted with back to back VSC consisting of two voltage source converters with a common DC link between them. The voltage source converters are independently controlled by PWM signals. The configuration enables a bi directional flow of power, from the generator to the power grid.

The generator side converter (GSC) will be able to control the output current from synchronous generators controlling the electrical torque. This will result in generator speed control to generate the maximum power per generator speed according to the optimal power curve specified by the manufacturer.

The network side converter (NSC) output can be controlled to alter the power angle, providing reactive power to and from the power grid. The NSC will be regulated to maintain a constant voltage over the DC link, transferring active current to the power grid in order to maintain a constant DC-link voltage.

The control strategy of the VSCs will be described in chapter 4.

3.1.3 Output filter

As the voltage output from the converters is generated using PWM, the output current needs to be filtered to remove the harmonic components. This is done applying an LC filter, with the inductive part in series with each phase and star connected shunt capacitors. The LC filter is designed to have a cut off frequency ten times the grid frequency.

3.2 Fully rated converter wind turbine

The typical configuration of a fully rated converter (FRC) wind turbine consists of the wind turbine rotor, connected to the generator via a gearbox or directly to the generator shaft. The generator can be asynchronous or synchronous with either wound field windings or permanent magnets (PMs) on the rotor to create the magnetic flux. If the generator is mounted on the same shaft as the wind turbine rotor the generator will need a large number of poles to generate power with a proper frequency.

If the wind turbine rotor shaft is connected to the generator shaft via a gearbox, the generator will have smaller dimensions as it will need a smaller number of poles. This will also lower the mechanical load on the wind turbine, as the wind turbine will have a high rotational speed, rather than the need to produce a large torque to generate the same amount of power [3].

The synchronous generator is generally not fitted with permanent magnets as the wound rotor enables the control of the excitation current. This enables control of the output voltage independent of the load current. The advantages with permanent magnet excitation are the lower excitation losses and that the more compact size of the generator. Additionally the generator is more robust and requires less maintenance.

As the FRC will generate electricity during variable wind speeds, the output frequency and voltage levels will not coincide with the gird frequency and voltage. The power converters will transform the generated power to fit the grid before it's transmitted to the power network. This is done with two back to back voltage source converters with a DC link between them. The FRC will have the entire generated electrical power transferred through the VSCs.

Fully rated converter control strategy

The control of the generator is provided through the generator side converter (GSC). The GSC decouples the mechanical speed of the generator and the power system electrical frequency, allowing variable-speed energy generation. The GSC will control the electrical generator using a reference curve for maximum power extraction per generator speed via torque control.

The control strategy strives to extract as much power as possible from the wind. The manufacturer of the wind turbine will design it for optimal power extraction according a curve showing generator speed to generated power. The generated power is dependent on the wind speeds per generator speed prompting the following curve (Figure 9)

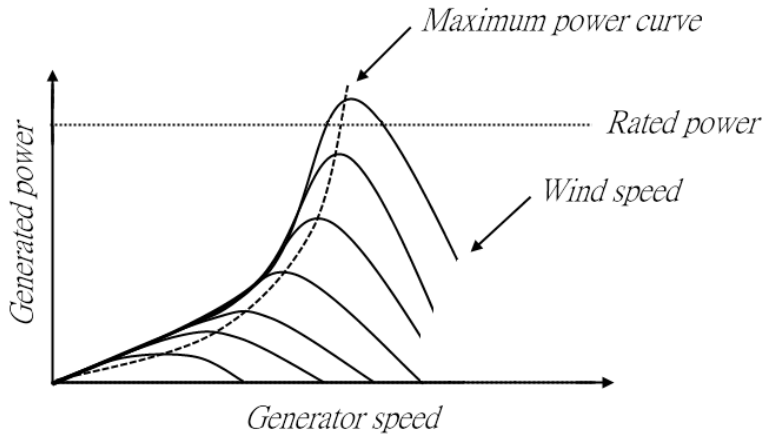


Figure 9 *Wind turbine characteristics plot showing generated power per wind speeds and generator speeds.*

The maximum power curve is expressed in optimal torque-speed curve as the controller will be regulating the generator torque. Generally the operating point should be between point B and point C during normal operation. At low speeds, the curve is limited by the cut-in speeds, between point A and B. At high wind speeds the rotational speed is limited to reduce aerodynamic noise, increasing the torque at constant speed (C-D). If the wind speeds increase any further, the strategy is to keep the torque constant between point D and E. Generator speeds exceeding point E causes pitch regulation in the control system of the wind turbine wings to lower the aerodynamic input power [3].

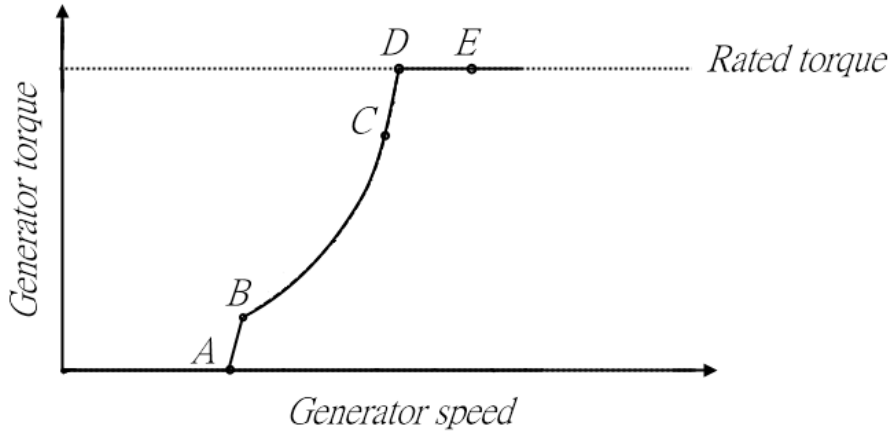


Figure 10 *Wind turbine characteristics plot showing optimal torque per generator speeds.*

The network side converters (NSC) will control the DC-link voltage aiming to maintain a constant DC-link voltage and to keep the grid voltage stable. The generator torque is controlled manually via a current reference since the aim of the project was to construct the wind turbine model controlled via the DSP. Therefore the added implementation of sensors and measurements needed from the generator were not stressed in this project. The ramification is that the generator will not be operating at optimal torque. The control schematics are illustrated in figure 11.

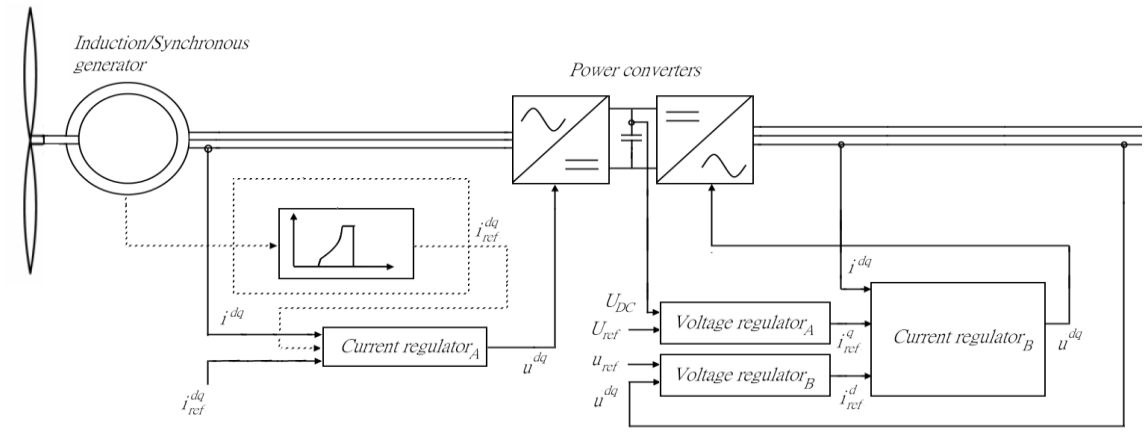


Figure 11 ***The control strategy schematics for the FRC in this project. The optimal torque reference curve will not be used in favor of the manually controlled current reference.***

4 Voltage Source Converter Control

The voltage source converters (VSCs) control strategy in this report will be the Carrier-based pulse width modulation (CB-PWM). The principle behind the control method is that a reference sine wave of a desired frequency is compared with a fixed high-frequency triangle wave to obtain voltage pulses of desired width. The reference value for the VSCs is calculated using controllers implemented in the stationary *dq* co-ordinate system.

4.1 Carrier-based PWM

Carrier-based PWM control compares a high frequency triangular wave V_{tri} with a sinusoidal reference signal V_{ref} in order to control the VSC. Whenever V_{tri} is larger than the reference wave, the upper transistor in one leg will be turned off and whenever the triangle wave magnitude is lower than the reference wave the same transistor will be turned on. The lower transistor in the same leg will be on and off inversely of the upper transistor. The equivalent setup is done for the two other legs working on their own phases [2].

The principle of obtaining transistor switch instances at the appropriate time is shown in figure 12.

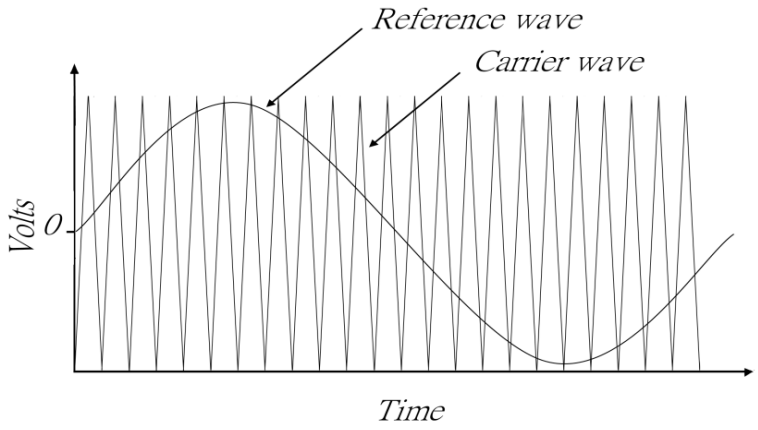


Figure 12 *Carrier-based PWM reference wave and triangular wave.*

The pulse train resulting from the switch signals corresponding to those in figure 12 is shown in figure 13. The pulses are widest whenever the sinus wave peaks causing the average voltage to increase, following the sinusoidal behavior [2].

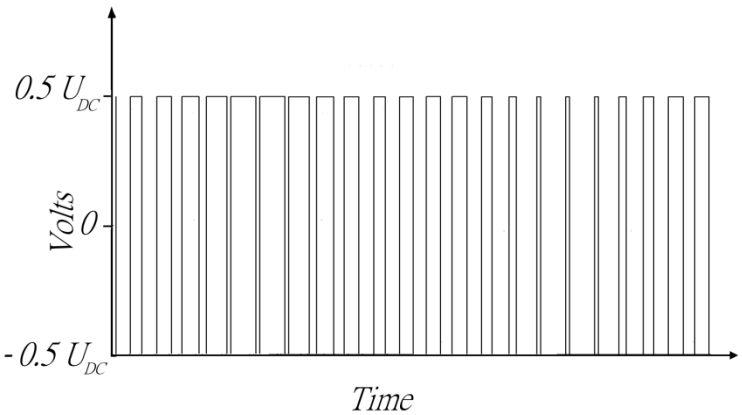


Figure 13 *Carrier-based PWM pulse train resulting from a sinusoidal reference wave.*

Modulation ratio is defined as:

$$m_a = \frac{\hat{V}_{ref}}{\hat{V}_{tri}}$$

The ratio is defined using the peak values. The difference between the phase potentials, i.e. the potential out from each half bridge creates the output voltage from the converter. Since the maximum output voltage that can be obtained is V_{DC} the carrier boundary value is: $\hat{V}_{tri} = 0.5V_{DC}$.

Over-modulation will occur if the modulation ratio exceeds unity. The frequency modulation ratio is defined by the ratio between the frequency of the triangle wave and the reference wave. This value should be kept as high as possible to ensure proper voltage pulse width.

The reference waveform usually has zero-sequence component added to improve the utilization of the converter by increasing the modulation index and lowering the voltage ripple. The zero-sequence added to each phase reference will not show up in the output, as the VSC is not connected to the neutral. This process is referred to a symmetrical reference wave and provides a 15 % higher margin to over modulation than sinusoidal reference waves [5] [6].

The zero-sequence component is calculated making the most negative and the most positive reference equal but with opposite signs and it is subsequently subtracted from the reference values (figure 14 and 15).

$$u_z = \frac{\max(u_a^*, u_b^*, u_c^*) + \min(u_a^*, u_b^*, u_c^*)}{2}$$

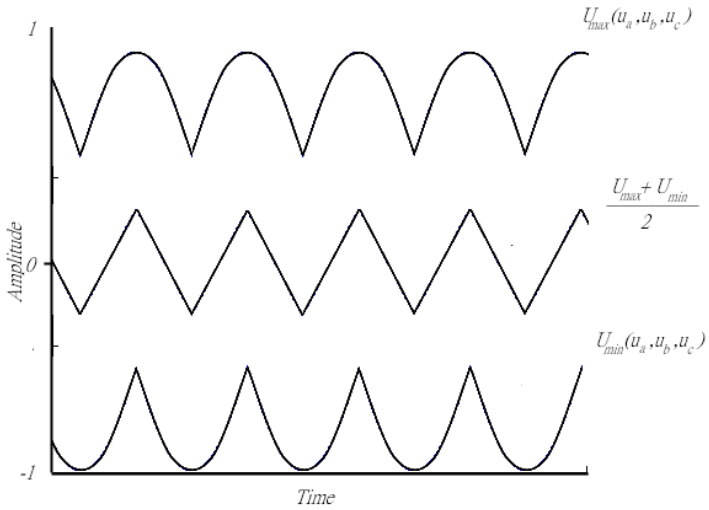


Figure 14 The results of the maximum and minimum of the instantaneous values of the three phases as well as the sum of them.

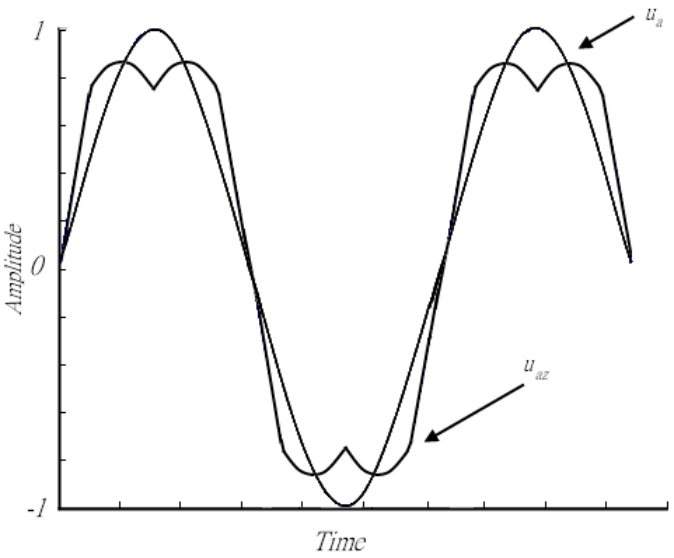


Figure 15 The sinusoidal and symmetrical reference wave used for modulation.

4.2 Vector space representation

In this project, the two-level voltage converter uses space vector PWM switching strategy, where the switch states in the converter can be defined by the switching vector. For a three phase, two-level converter there are eight possible switch state combinations with six active and two zero vectors. The vector length of each active vector is

$$\vec{U} = \sqrt{\frac{2}{3}} \cdot U_{dc}$$

However the converter can only provide controllable voltage within the hexagon shaped frame the active vectors create. The maximum voltage vector is thus only

$$\vec{U}_{max} = \sqrt{\frac{1}{2}} \cdot U_{dc}$$

The voltage vectors for a three phase two level VSC along with the hexagonal shape illustrating the modulation ratio of 1 are illustrated in figure 16. The switch states are shown in figure 17 [6]. It can be noted that the switch states are set that only one switch will alter its state to represent the voltage vector next to it. This will lower the amounts of switching and thus increase the life of the transistors.

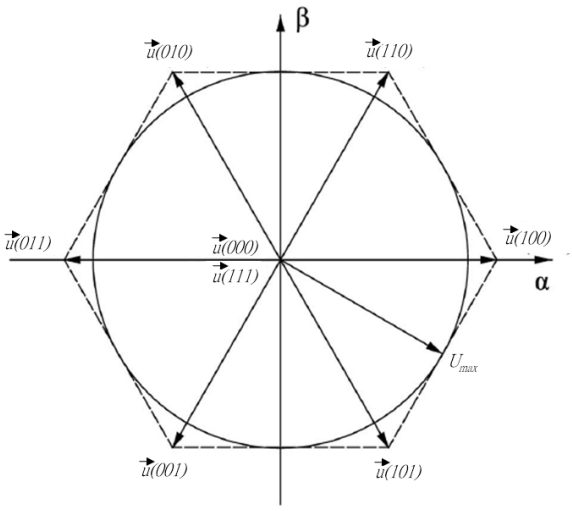


Figure 16 *Voltage vectors for the two level three phase converter.*

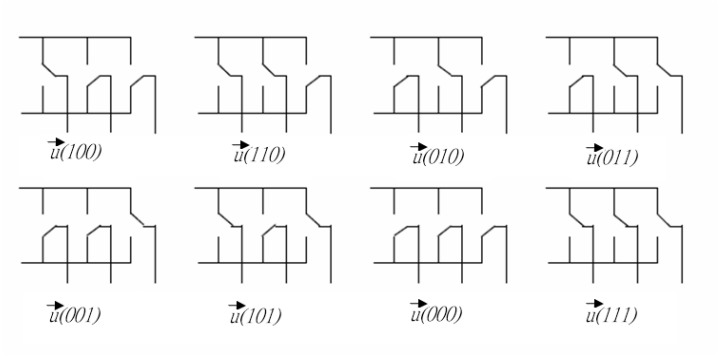


Figure 17 ***Switch states corresponding to the voltage vectors inn figure 14[7]***

4.3 Clarke & Park Transformation

Clarke and Park transformations are used to control electrical drives structures involving AC-motors using complex space vectors. The transformation entails the reference frame to change from a time variant three-phase axes coordinate system to a two coordinate rotating reference frame. This facilitates the control and scrutinisation of three phase voltages, currents and fluxes.

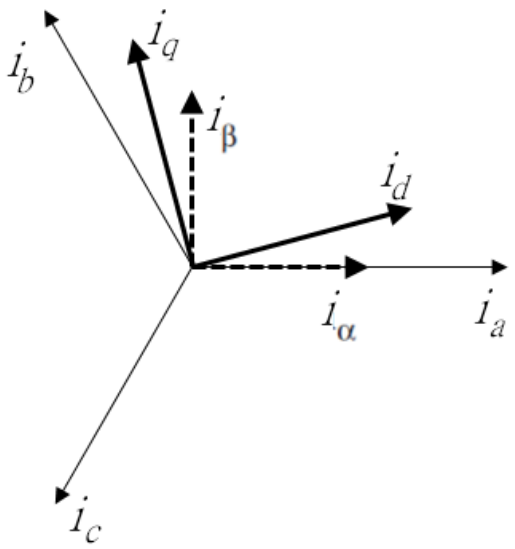


Figure 18 The relationship between the alfabeta, dq and the abc frame.

Figure 16 illustrates the relationship between the three-phase time invariant stationary reference frame, the two co-ordinate reference frame and finally the rotating reference frame *dq*.

4.3.1 Clarke transformation

The Clarke transformation modifies the three-phase system to a two-phase orthogonal system as follows:

$$\begin{cases} i_\alpha = i_a \\ i_\beta = \sqrt{\frac{3}{2}} \cdot i_b - \frac{1}{\sqrt{2}} \cdot i_c \end{cases}$$

Where i_α and i_β are components of the orthogonal reference frame.

4.3.2 Park transformation

The Park transformation changes the two-phase orthogonal reference frame to a time invariant, rotating two-phase orthogonal reference frame aligned with the rotor flux. The *q*-axis angular displacement from α axis is represented by the angle θ.

$$\begin{cases} i_d = i_\alpha \cdot \cos(\theta) + i_\beta \cdot \sin(\theta) \\ i_q = -i_\alpha \cdot \sin(\theta) + i_\beta \cdot \cos(\theta) \end{cases}$$

Where i_d and i_q represent the currents on the *direct axis* and *quadrature axis* respectively.

The transformation back from the *dq* and $\alpha\beta$ frame is done with inverse Clark and Park transformations [7].

$$\begin{cases} i_\alpha = i_d \cdot \cos(\theta) - i_q \cdot \sin(\theta) \\ i_\beta = i_d \cdot \sin(\theta) + i_q \cdot \cos(\theta) \end{cases}$$

$$\begin{cases} i_a = \sqrt{\frac{2}{3}} \cdot i_\alpha \\ i_b = -\sqrt{\frac{1}{6}} \cdot i_\alpha + \sqrt{\frac{1}{2}} \cdot i_\beta \\ i_c = -\sqrt{\frac{1}{6}} \cdot i_\alpha - \sqrt{\frac{1}{2}} \cdot i_\beta \end{cases}$$

4.4 Current and voltage regulators

The current and voltage regulator compare values in the *dq* frame in order to control voltages and currents transmitted to and from Voltage Source Converters using PWM. The control strategy for the project is as illustrated in figure 19.

Current regulator A will control the generator torque with a manually set reference value. Current regulator B will control the active power flow into the grid and reactive power flow into or from the grid in order to keep a stable value over the DC-link and to contribute with reactive power to the grid in order to obtain nominal voltage levels on the grid at disturbances respectively.

Voltage regulator A compares a reference value for the DC-link voltage with the measured DC-link voltage and produces a current reference on the q axis to regulate active power. Voltage regulator B compares the nominal grid voltage value with the measured and produces a current reference on the d axis to regulate reactive power.

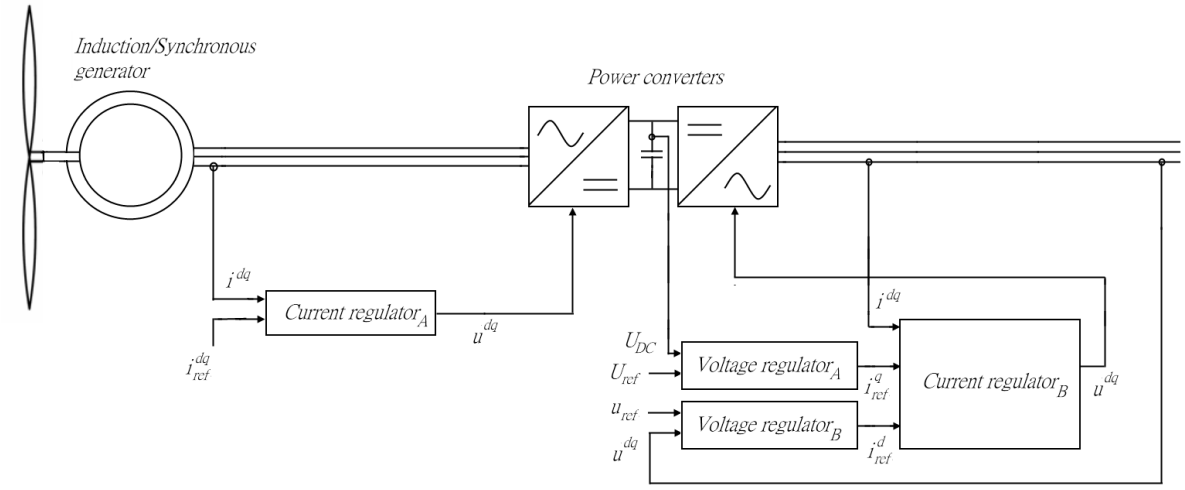


Figure 19 ***The control strategy for the FRC in this project***

4.4.1 Current regulator

Current regulators are used to control and inject current in the grid and to control electrical drives regulating the motor torque. The regulator is derived from an ideal single-phase load model with a resistance (R), an inductance (L) and an induced emf (e) in series.

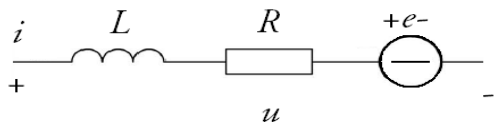


Figure 20 *Generic one phase load*

The equation representing the voltage across the model is given by

$$u = R \cdot i + L \cdot \frac{di}{dt} + e$$

For a three-phase load model, voltages and currents are represented by symmetric vectors and symmetric resistive and reactive loads, each phase shifted by 120 degrees.

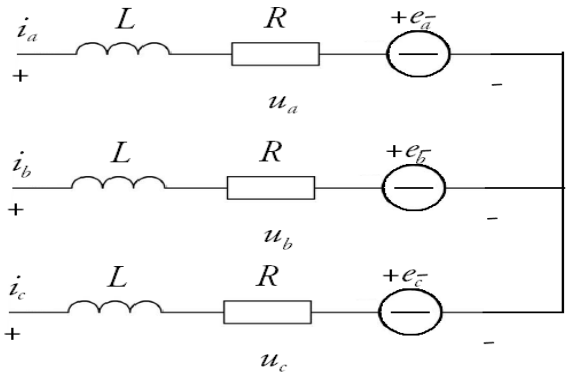


Figure 21 *Generic three phase load*

The voltage across each phase is given by:

$$\begin{cases} \sqrt{\frac{2}{3}}\left(u_a = R \cdot i_a + L \cdot \frac{di_a}{dt} + e_a\right) \\ \sqrt{\frac{2}{3}} \cdot e^{j\frac{2\pi}{3}} \cdot \left(u_a = R \cdot i_a + L \cdot \frac{di_a}{dt} + e_a\right) \\ \sqrt{\frac{2}{3}} \cdot e^{j\frac{4\pi}{3}} \cdot \left(u_a = R \cdot i_a + L \cdot \frac{di_a}{dt} + e_a\right) \end{cases}$$

The equation is transformed using Clarke transformation to the *αβ*-reference system and subsequently to the rotating *dq*-reference system

$$\vec{u}^{\alpha\beta} = R \cdot \vec{i}^{\alpha\beta} + L \cdot \frac{d}{dt}\vec{i}^{\alpha\beta} + \vec{e}^{\alpha\beta}$$

$$\vec{u}^{dq} = R \cdot \vec{i}^{dq} + L \cdot \frac{d}{dt}\vec{i}^{dq} + j\omega L\vec{i}^{dq} + \vec{e}^{dq}$$

where ω is the grid angular frequency.

By dividing the equation into its real and imaginary components will result in

$$\begin{cases} u_d = R \cdot i_d + L \cdot \frac{di_d}{dt} - \omega L i_d + e_d \\ u_q = R \cdot i_q + L \cdot \frac{di_q}{dt} + \omega L i_q + e_q \end{cases}$$

This equation is expressed in discrete-time to construct the expression for a dead-beat PI (Proportional-Integral) controller which output will be voltage references in the *dq* reference frame [5].

$$\begin{cases} u_{d,ref,k} = K \cdot \left(\left(i_{d,ref,k} - i_{d,k}\right) + \frac{1}{T_i}\sum_{n=0}^{k-1}\left(i_{d,ref,n} - i_{d,n}\right) \right) - K_c \cdot \frac{i_{q,ref,k} + i_{q,k}}{2} + e_{d,k} \\ u_{q,ref,k} = K \cdot \left(\left(i_{q,ref,k} - i_{q,k}\right) + \frac{1}{T_i}\sum_{n=0}^{k-1}\left(i_{q,ref,n} - i_{q,n}\right) \right) + K_c \cdot \frac{i_{d,ref,k} + i_{d,k}}{2} + e_{q,k} \end{cases}$$

Where *k* is a sample instance and

$$\begin{cases} K = \left(\frac{L}{T_s} + \frac{R}{2}\right) \\ T_i = {R}/{\left(\frac{L}{T_s} + \frac{R}{2}\right)} \\ K_c = \frac{\omega L}{2} \end{cases}$$

If the deviation from the reference value remains for a longer time, the integral part of the equation will grow to a large value ad saturate the output signal from the regulator. The solution is to limit the maximum integral part by giving it the value of the output signal subtracted with the proportional part. This preventative measure is referred to anti-windup.

The anti-windup function will have the following expression:

$if\ u(k) > u_{max}\ \ then$

$u(k) = u_{max}$

$u_{int}(k) = \ u_{max} - K \cdot \left(y_{ref}(k) - y(k)\right)$

$else\ if\ u(k) <\ u_{min}$

$u(k) = u_{min}$

$u_{int}(k) = \ u_{min} - K \cdot \left(y_{ref}(k) - y(k)\right)$

$end\ if$

4.4.2 Voltage regulators

The Voltage regulators in this project are built using the following equations

$$e(t) = y_{ref}(t) - y(t)$$

where $e(t)$is the reference error, $y_{ref}(t)$ is the reference value and $y(t)$ is the measured value.

Adding an integral part to the reference error will reduce the steady-state error to zero

$$u(t) = K_p \cdot \left(e(t) + \frac{1}{\tau_i}\int e(t)dt\right)$$

where $u(t)$ is the output voltage, K_p is the proportional constant and τ_i is the time constant.

The integral part will be fitted with an anti-windup to avoid output signal saturation as with the current regulators.

The equation will be expressed in discrete-time to derive the PI controller to output voltage references in the *dq* frame.

$$u(k) = K_p \cdot \left(y_{ref}(k) - y(k) + \frac{T_s}{T_i}\sum_{n=0}^{n=k}\left(y_{ref}(n) - y(n)\right)\right)$$

The parameters in the voltage regulator are adjusted starting with standard voltage regulator parameter values [6].

$K_p = 10$

$K_i = \frac{T_s}{T_i} = 0.5$

4.4.3 Grid Characteristics

The grid characteristics of a system operating in sinusoidal steady state are described as follows:

The apparent power flowing through the lines is a sum of the active power and the reactive power, where the reactive power is defined as the imaginary part of the complex valued apparent power.

$$S = P + jQ$$

The transfer of the apparent power from a bus to the load bus is dependent on the grid impedance as it will be the source of phase shift between the grid and the load voltage.

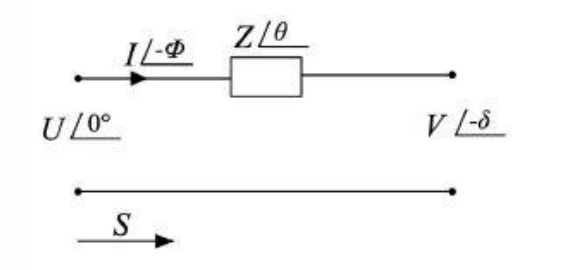


Figure 22 Principal illustration of voltage drop due to transmission line inductance.

The expression for the active and reactive power is derived from:

$$S = P + jQ = U \cdot I^* = U\left(\frac{U - V}{Z}\right)^* = U\left(\frac{U - Ve^{j\delta}}{Ze^{-j\theta}}\right) = \frac{U^2}{Z}e^{j\theta} - \frac{U \cdot V}{Z}e^{j(\delta+\theta)}$$

Where

$$P = \frac{U^2}{Z}\cos(\theta) - \frac{U \cdot V}{Z}\cos(\delta + \theta)$$

$$Q = \frac{U}{Z}\sin(\theta) - \frac{U \cdot V}{Z}\sin(\delta + \theta)$$

Expressing the impedance as a complex value where the imaginary part is defined as the line reactance and the real part as the line resistance:

$$Ze^{j\theta} = R + jX$$

Yields the expressions:

$$Vsin(\delta) = \frac{XP - RQ}{U}$$

$$U - Vcos(\delta) = \frac{RP + XQ}{U}$$

Generally the resistance, R is neglected to simplify the equations as the magnitude of the impedance greatly exceeds that of the resistance. And as the power angle δ is small the sine and cosine values can be simplified as $\cos(\delta) \approx 1$ $\sin(\delta) \approx \delta$, yielding the expressions to:

$$\delta \cong \frac{XP}{UV}$$

$$U - V \cong \frac{XQ}{U}$$

From the expressions it can be concluded that the reactive power transfer results in a voltage difference between the grid and the load. The active power transfer is dependent on the power angle [7].

5 The laboratory setup

In this chapter the hardware topology of the wind turbine converter is described. The choice of setup is described and discussed. Each part of the hardware and measurement system will be presented with its beneficial attributes and its limitations.

5.1 System description

The system uses asynchronous and DC generators placed below the laboratory level in order to save space. Below the laboratory setup is an asynchronous motor (AM) connected to the power grid. On the axis from the AM is the rotor of a DC generator. Magnetization current is fed to the DC generator and the generated power is fed to the system above ground.

Above ground, the current from the DC generator is fed to a DC motor. A field winding magnetization current is applied to the DC motor and it starts to gain speed. The nominal speed is 1500 rpm.

The following step is to magnetize a synchronous generator (SG) located on the same axis as the DC motor. This is done when a current is fed to the field winding of the SG. The output from the synchronous generator is three-phase power, which is to be fed to a voltage source converter (VSC) on the generator site. The synchronous generator rated power is 2200 VA.

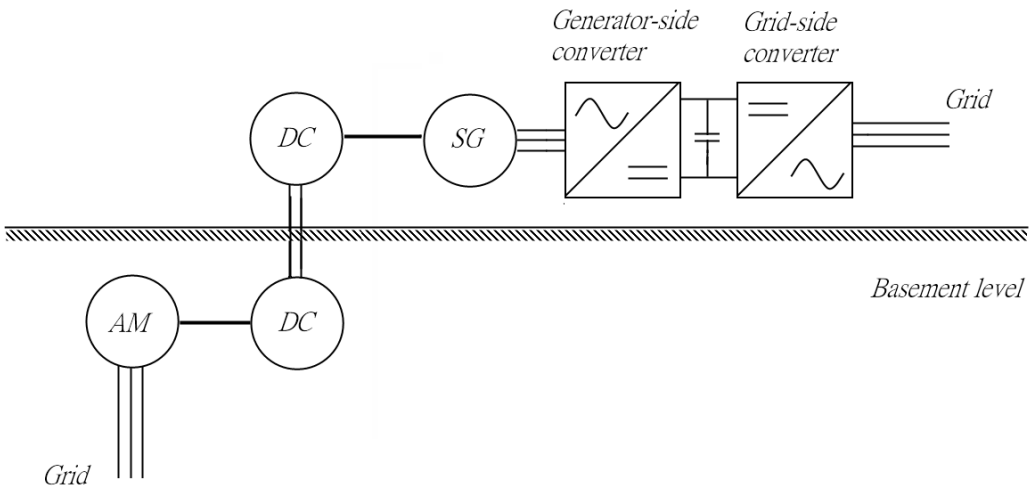


Figure 23 The laboratory setup

The VSC on the generator side will feed a charging current to the DC link between both VSC: s. The VSC on the grid side will be controlling the DC link level by feeding out active power on to the net in order to maintain a stable DC-link voltage. The VSC only allows for active power to be transferred out from converter, however reactive power can travel in and out from the VSC.

The VSC on the grid side is subsequently connected to the grid at 230 V.

The reasoning behind the DC-link voltage level is to ensure that the converters could deliver enough voltage without any risk for over modulation when output per phase is set to be 127 V.

5.2 Experimental Rack

The Rack in which the VSC: s, measurement instruments and digital signal processor (DSP) are situated will be explained in this chapter. The general principle will be introduced followed by a description of each part in greater detail.

At the top level of the experimental rack is the DSP, along with its power supply and signal adjustment card for the AD converter unit and the signal adjustment cards for PWM VSC control.

Directly below are the LEM modules mounted on a relay rail connected to the DSP via the signal adjustment card for the AD converter. They measure phase-to-phase voltages on both sides of the VSC: s $v_{ab1,2}, v_{bc1,2}$ and the DC-link voltage U_{dc}.

In the middle of the experimental rack, there are two VSC: s mounted on an aluminum plate with their heat sinks facing the plate. The built in LEM modules in the VSC: s measures the phase currents $i_{a1,2}, i_{b1,2}$.

In order for the converters to avoid the large current inrush when the discharged capacitors are connected to the grid, the phase current is forced to pass through power resistors. When sufficient time has passed, i.e. the capacitors are charged up the power resistors will be bypassed. The control of the power resistors is done using timed relays.

The current from the VSC passes through LC filters to remove the harmonics from the current signal making it more sinusoidal.

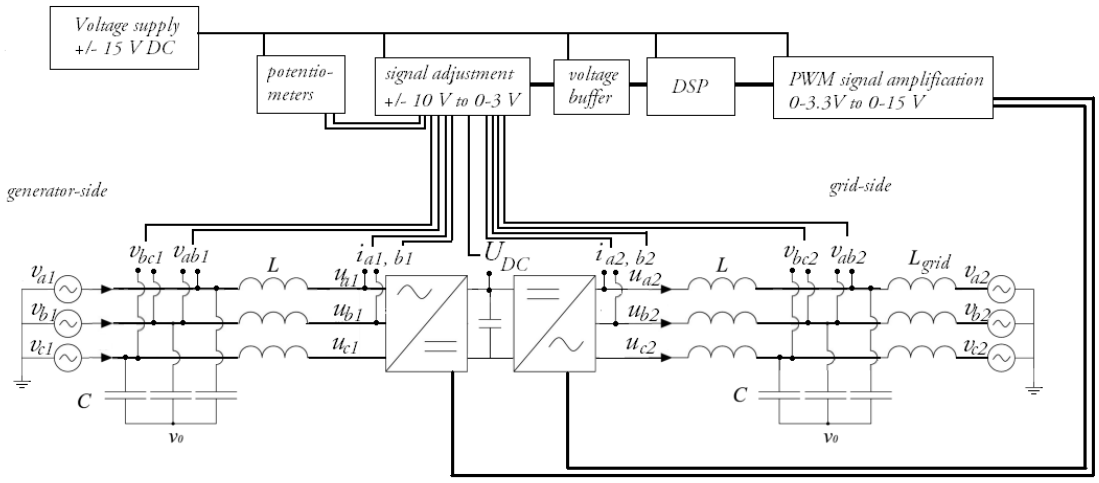


Figure 24 The measured signals and the hardware required to process it.

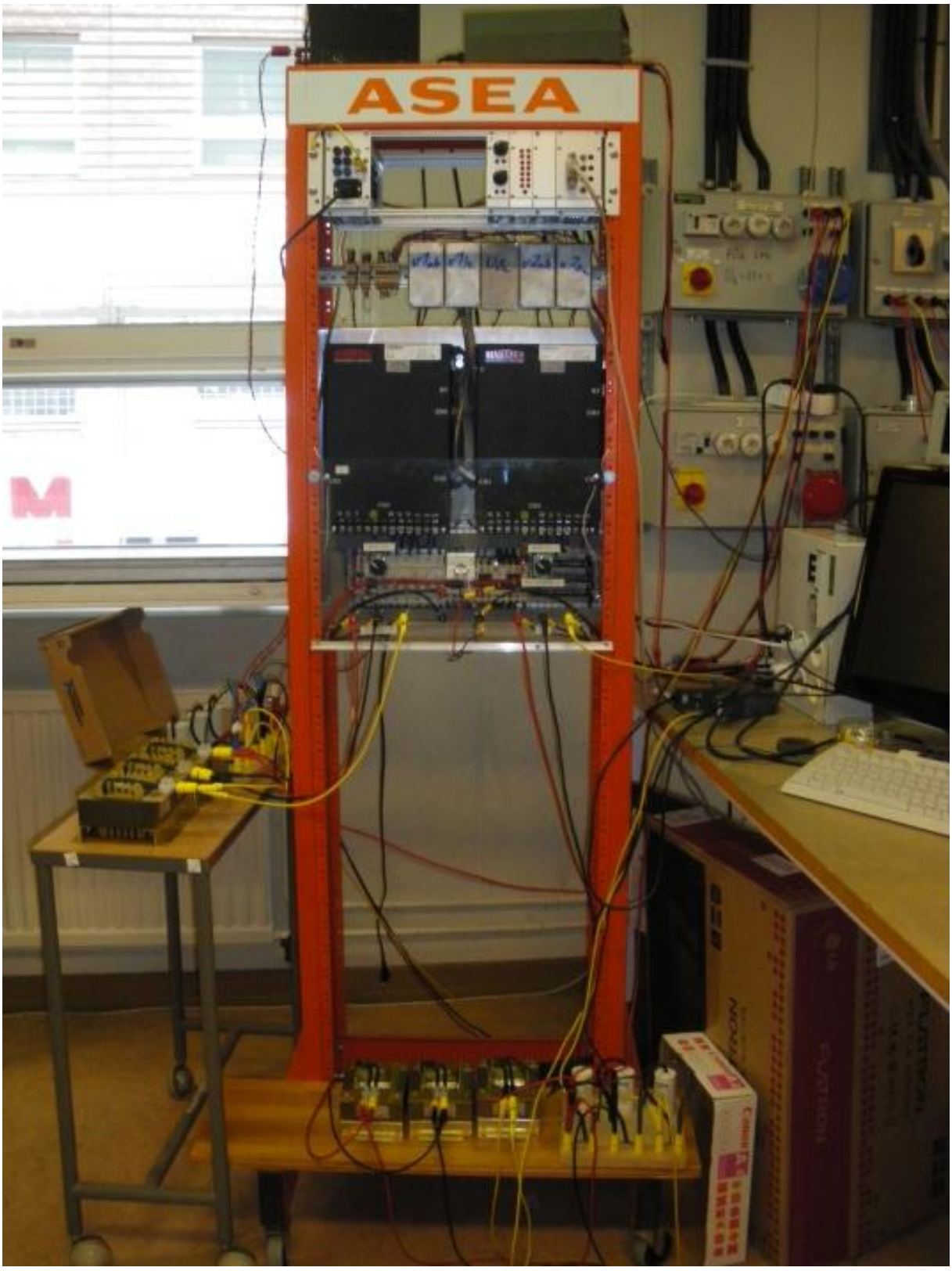


Figure 25 Picture of the hardware and setup for the project

5.2.1 The Digital Signal Processor (DSP) rack

The DSP rack space is situated at the top of the VSC rack and contains the DSP with a voltage buffer, voltage supply unit, potentiometer unit, signal adjustment card for the onboard ADC and PWM signal card. Figure 26 is an illustration of which units are confided into the top rack and their order.

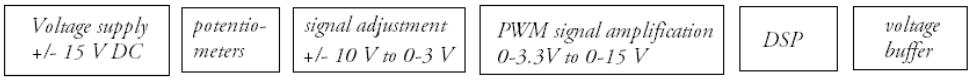


Figure 26 ***The digital signal processor rack layout from left to right***

The individual units are connected to each other via signal wires in the back of the top rack. The voltage supply unit feed the units to the right of it with power. The potentiometers feed an adjustable signal to the signal adjustment card which signals are fed to the DSP via the voltage buffer card. Signals from the transducers are also fed to the signal adjustment card. The signals from the DSP need to be amplified to control the PWM, which is conducted in the PWM signal amplification card.

The Digital Signal Processor

A digital signal processor is a central processing unit specially designed to handle tasks such as mathematical operations, digital filtering and Fourier analysis of signals in real time. The device is therefore usually several times faster than the ordinary CPU as it doesn't have general-purpose software running.

The DSP displays a predictable execution time for the processes it will perform, differentiating it from the microprocessors in modern PCs where processes can be executed at different time instances depending on programs running in the background.

The processor will be used as an embedded controller dedicated to a set number of tasks to be implemented in a sequence to control the output from the two side-by-side mounted VSC: s. The current and voltage regulators running on the DSP will generate an output, which is fed to the PWM controller, which subsequently controls the VSC: Regulators and functions are written in C and assembly.

The DSP model used in the laboratory setup is TMS320F2812. The main features are the 150 MHz processor clock, 32-Bit CPU. It contains an on-chip memory consisting of a flash device in 16 sectors, each 128K from which the program will be running. Additional storage is provided from the RAM and external memory on the DSP.

The analog-digital converter is a 12-bit 16-channel device allowing both sequential and simultaneous conversions with the conversion rate of 80ns. The DSP is fitted with a voltage buffer to decrease the charging time of the capacitors in the ADC, increasing the accuracy of the conversion.

The Voltage Supply

The voltage supply supplies the DSP and the four other cards in the DSP rack space with +15 V, +5 V and –15 V DC as well as a reference ground.

The Potentiometer Card

The potentiometer card is used to test regulators response to variable input voltages. It consists of two potentiometers, which can be continuously adjusted in 10 revolutions with each revolution indicated by 100 equally spaced points on the wiper. The output from the potentiometers is a voltage ranging from –10 to +10 V.

The ADC signal adjustment card

The ADC signal adjustment card is designed for the various signals from the current and voltage transducers and the potentiometers. The signals are adjusted to range from 0 – 3 V from the transducer and potentiometer output voltage in the range of +/- 10 V.

The reference ground is separated from the ground on the PWM card, as the digital signals will cause ground currents to flow destabilizing the reference point.

The design of the ADC signal adjustment card is done using MATLAB to find the lowest number of resistors to adjust the voltage and P-CAD 2000 to design the actual card.

The PWM card

The PWM card receives signals from the DSP I/O ports at 3.3 V, which are amplified in the PWM card to 0 – 15 V in order to control the drivers for the IGBT transistors at the VSC: s. This card has a separate path to ground, as the digital signals might interfere with the analogue ground. It is supplied with +/- 15 V from the power supply unit in the DSP rack.

The Voltage Buffer Card

The voltage buffer card will aid the charging of the capacitors in the AD converter to charge faster increasing the accuracy of the conversion.

The LEM Modules

LEM modules are transducers designed to measure voltage or current using the Hall effect. The principle of Hall effect is, that a conductive material with a current flowing through it enters a magnetic field, a magnetic force proportional to the current and the magnetic field will force charge carriers to move in the direction of the force. The buildup of charge carriers on one side of the conductor will result in a measurable potential difference between the two sides of the conductor.

The LEM module will also work as a galvanic isolation between the high voltage primary circuit and the secondary circuit consisting of the DSP input signals.

The Hall voltage is given by:

$$V_H = \frac{I \cdot B}{n \cdot e \cdot d}$$

Where B and I represent the magnetic field and current through the conductor. n represent the density of mobile charges, e the electron charge and d is the thickness of the conductor.

The LEM modules voltage transducers, LV 25-P uses the Hall effect as follows, $\pm 15\ V$ DC is applied to the terminals to create the magnetic field using a coil and a resistor in series. Through the conductor in the magnetic field, the current to be measured will be flowing and the voltage over the conductor will be connected to the ground reference via a resistor.

The voltages to be measured are phase-to-phase voltages corresponding to 230 V with the manufacturers' criteria for best accuracy, that the primary nominal i_{rms} will be 10 mA. This is achieved by letting the current pass through 40 kΩ. The current conversion ratio is 2.5 in the LEM module, which is later to be fed out over a 120 Ω resistor. The voltage over the resistor will be $\pm\ 3\ V_{rms}$ before leaving the ADC signal adjustment card as $0 - 3\ V_{rms}$.

The LEM modules for current measurements, LAH 25-NP are located in the VSC : s and they are designed to have a primary nominal i_{rms} of 25 At and a secondary nominal current at 25 mA with the conversion ratio of 1:1000. The voltage over the shunt resistor will be $\pm\ 5.75\ V_{rms}$ and out from the ADC signal adjustment card $0 -\ 3\ V_{rms}$.

The LEM module measuring the DC-link voltage will measure up to 600 V. To achieve 10 mA rms current through the primary it's placed in series with 60 kΩ. The conversion ratio of 2.5 will give a 25 mA rms output current which is fed over a 120 Ω resistor with $\pm\ 3\ V$ over it, giving a 0 – 3 V out from the ADC signal adjustment card.

5.2.2 The Voltage Source Converters (VSC)

The voltage source converters are connected in series to create a common DC-link between them. The device used in the laboratory setup is manufactured by SEMIKRON with the model name: SKS 11F B6CI 07 V12.

The main features of the VSC are the 900 µF capacitor bank able to charge to 800 V DC. The maximum rms current fed through the converter is 18 A and its working on a 97 % efficiency. It has a capacity of switching at 10 kHz, however in this application is will operate at 4.95 kHz.

In order to ensure that the reference signal won't exceed the manufacturers current and voltage limitations during operation, both VSC: s are fitted to limit the possible output according to the specifications.

5.2.3 The relay setup

When the converter is connected to the grid, there is a charging current flowing to the capacitors. Due to the potential difference and the low resistive current path, the initial charging current will be relatively high, putting strain on the capacitors, diodes and fuses.

In order to limit the initial charging current, resistors are put in each phase conductor to the converter. However, as the resistive path would alter the future operation of the converters, the resistors should only be in use at the moment the converters are connected to the grid.

The solution to this is to set up a time relay, which initially limits the current with resistors but later on bypasses them when the converters are operational. The time needed to charge the capacitors is dependent on the time constant, which is approximately 1 second. The relay scheme is as follows:

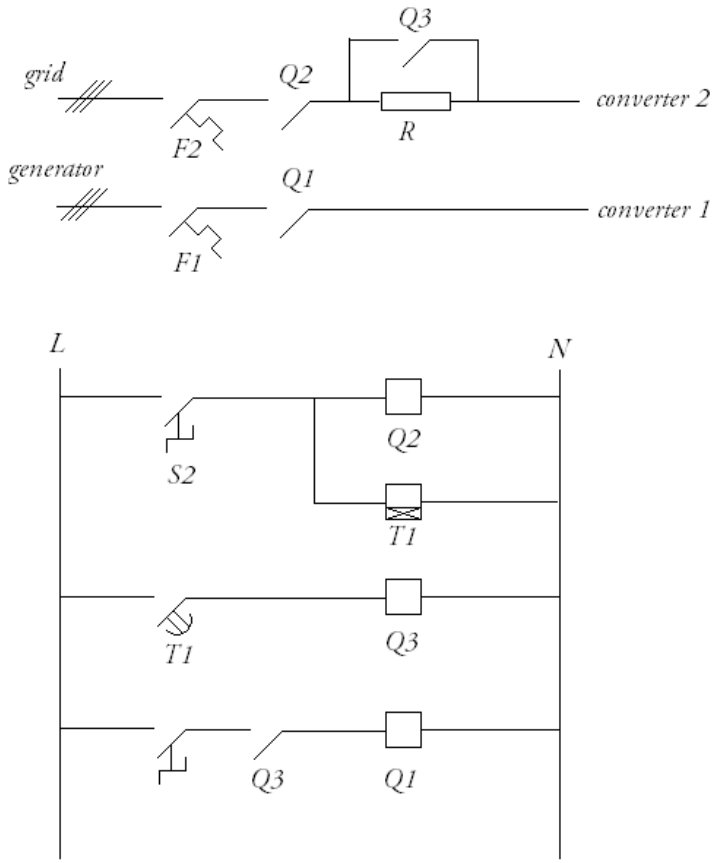


Figure 27 The relay scheme protecting the converters at startup

As switch S2 is set to conduct, Q2 closes and current from the grid will flow to the converter capacitors via the resistor. Simultaneously, the timer relay (T1) will be activated setting the delayed switch to conduct after a set time. After the delayed switch starts to conduct, Q3 closes and the resistor is now bypassed.

To supply the converter connected to the generator with voltage switch S1 has to be conducting. The criterion is that the capacitors first need to be charged, which is set if Q3 is conducting.

F1 and F2 are fuses to protect the converters from currents over 16 A.

5.2.4 LC Filters

The current from the converters will contain high-order harmonics since the sine wave is constructed via PWM. To remove the harmonics, the converter output is filtrated using a LC filter. The switch frequency is 4.95 kHz and the grid frequency is 50 Hz. According to the stipulations concerning filter design, the cut off frequency is set to 500 Hz.

6. The Software Description

The converter control algorithm is written and tested in C-code and assembly in Code Composer Studio v3.1 (CCS); the development tool provided with the DSP. When the program set to run from the flash memory, it's programmed using SDFlash v1.63 [8].

CCS allows debugging; however the program does not allow variables to be viewed in real-time nor are the parameters to be adjusted while the program running. Any change in parameters needs to be reloaded into the memory and any variable graph needs manual refreshing. All graphs and variables can be simultaneously refreshed using software probe point triggering the variable update.

During the testing phase, the entire program is executed from the DSP RAM as the RAM memory is programmed considerably faster than the flash memory. The execution time is however longer from the RAM, limiting the complexity and size of algorithm calculations compared to if it was to be ran from the flash memory. Additionally, debugging in CCS when the program is executed from flash is more difficult, as graphs can't be refreshed using software probe points.

The principle for the embedded system is that each sample instance, there is an interrupt routine in which the ADC is calibrated, calculations are made on the values from the ADC, reference values from the current and voltage regulators are sent to The program is written from the shell of an example file included with the software, as the initiation sequences for every program to be executed from the DSP will be similar.

The functions that have been added:

- ADC calibration
- Clarke and Park transformation
- PI regulators
- Inverse Clarke and Park transformations
- Reference phase voltages for the PWM circuit
- Protect converters from over current and over voltage

This chapter does not aim to provide a tutorial on writing C code or assembly or using CCS apart from providing information for the basic understanding of how to run and debug the program which has been written [9][10][11].

The DSP is programmed using IQmath which enables the user to define the range and resolution of the 32-bit fixed point numbers. In this project IQmath is using the Q format 20 which range is -2048 to ~ 2048 with the resolution of $9.54 \cdot 10^{-7}$ to keep a high level of precision [12].

6.1 ADC calibration

ADC calibration is necessary since the ADC will be impacted by gain and offset errors contributing to errors in the control system. Calibration is executed using two reference values fed into two of the ADC channels. This will require that two of the 16 channels will be dedicated for the reference values to compensate for the errors [13][14][15][16].

The AD conversion is setup to lower the overall time for the sampling by simultaneous sampling 2 channels. The operation mode is defined in the ADCcalibrationDriver.h headerfile as well as what the reference values channels and what they correspond in ideal count value.

The calibration initiation is added to the initialization code for the DSP and in the beginning of the interrupt routine the ADC channels are read, calibrated and stored in the memory structure. From this point the values are also adjusted to correspond to values in A and V.

Using the source file gainoffset.c, the ADC-register values are altered. Initially the register values need to left-shifted 4 bits before they can be adjusted, as the ADC-register reserves the first 4 bits in the 16-bit register. The register value is subsequently altered to a two-bit complement representation, as the ADC-register stores the most negative value as the smallest value.

When the ADC-register samples values corresponding to the 12-bit resolution upper limit; 4095, there is an obvious risk for the values to exceed 4096 and be represented from 0 upwards. The equivalent scenario occurs when the ADC register contains values close to the lower bit limit being represented as 4096 and lower. Both problems can be mitigated setting a limit for the highest and lowest value to be represented. The representation can therefore only be made with values ranging from 5 bits to 4090, however this does not lower the bit resolution significantly (~0.27% lower).

```
…
int16 highlimit = 4090;
int16 lowlimit = 5;
…
            if(Udc < lowlimit)
                        {
                                    Udc = lowlimit;
                        }
                        else if (Udc > highlimit)
                        {
                                    Udc = highlimit;
                        }

                        DatQ15 = (Udc << 4)^0x8000;
                        Tmp = (int32)DatQ15+ p->offs_Udc;
                        p->Udco = _IQ15mpy(Tmp,p->gain_Udc);
…
```

6.2 Clarke and Park transformation

Clarke and Park transformations are conducted according to the theory presented in chapter 3.3. Initially each phase passes through a symmetrization in order to remove any offset in the measurements for each phase-to-phase voltage measurement. The following process is to retrieve each phase voltage.

6.2.1 Clarke transformation

The Clarke transformation for the phase currents is identical except for the absence of the phase-to-phase to phase transformation.

```
void alfabeta_calc(ALFABETA *v)
{
		_iq sym;
		_iq ca;
		ca = -v->ab - v->bc;
		sym = _IQmpy((v->ab +  v->bc +  ca), _IQ(0.3333333));
		v->ab = v->ab - sym;
		v->bc = v->bc - sym;

		v->a = _IQmpy((_IQmpy(_IQ(2), v->ab) +  v->bc), _IQ(0.3333333));
		v->b = _IQmpy((v->bc - v->ab), _IQ(0.3333333));
		v->c = -v->a - v->b;
		v->alfa = _IQmpy(_IQ(1.22474), v->a);                    //sqrt(3/2)
		v->beta = _IQmpy(_IQ(0.70711), (v->b - v->c)); //sqrt(0.5)
}
```

The next process is to move the reference frame to a rotating two-phase orthogonal reference frame. The angular displacement of the q-axis from alfa-axis is represented by the angle THETA.

In order for the angle to be relatively stable, the voltage is firstly filtrated using a second order low pass filter before theta is calculated. The filter parameter values are calculated using MATLAB to correspond to a second order Butterworth filter with a cut off frequency of 10 Hz to remove most of the disturbance.

```
void filter2_calc(FILTER2 *v)
{

		v->y = _IQmpy(v->a1,v->y_old1) +  _IQmpy(v->a2,v->y_old2) +
_IQmpy(v->b1,v->x);
		v->y_old2 = v->y_old1;
		v->y_old1 = v->y;
}


void angle_calc(ANGLE *v)
{
		v->angle = _IQatan2(v->beta, v->alfa);
}
```

The filtration causes the angle between the alfa-axis and the q-axis to shift 90 degreed lagging the q-axis, prompting the Park transformation to be made between the d-axis and the alfa-axis.

6.2.2 Park transformation

```
void parkdq_calc(PARK *v){

                _iq Cosine,Sine;

                // Using look-up IQ sine table
                Sine = _IQsin(v->Angle);
                Cosine = _IQcos(v->Angle);

                // with filtercompensation

                v->d = _IQmpy(v->alfa,Cosine) +  _IQmpy(v->beta,Sine);
                v->q = _IQmpy(v->beta,Cosine) - _IQmpy(v->alfa,Sine);
}
```

The following step is to feed the Park transformed voltage and current values into the voltage and current regulators.

6.3 The PI regulators

The current and voltage regulators are designed to produce a reference value to correspond to the intended operation of each converter. The grid-side converter aims to:

- Keep the DC-link voltage stable at U_{DC}^{ref}
- Keep the grid voltage stable at U^{ref}

The generator-side converter has only one purpose; that active power to be fed out on the grid.

6.3.1 Keeping DC-link voltage stable

Keeping the DC-link voltage stable consists of feeding active current out on to the net when the wind turbine is generating power. The voltage regulator will produce a current reference value in the q-axis as only active power will be generated from the wind-turbine. The current reference is fed into a current regulator generating a qd-frame voltage value to be fed out from the grid-side converter using the PWM circuit on the DSP.

```
void VDCREGS_calc(VDCREGS *v)
{
		v->iq_ref_P = _IQmpy((v->Udc_ref - v->Udc),v->Kp);
		v->iq_ref_I = _IQmpy(v->Ts_Ki_Kp,(v->Udc_ref - v->Udc)) + v-
>iq_ref_I;
		v->coefficient = _IQdiv(v->Udc,v->vq);
		v->iq_ref = (v->iq_ref_P + v->iq_ref_I);
		if(v->iq_ref > v->iqmax)
		{
			v->iq_ref = v->iqmax;
			v->iq_ref_I = v->iqmax - v->iq_ref_P;//,v->coefficient);
		}
		else if(v->iq_ref < v->iqmin)
		{
			v->iq_ref = v->iqmin;
			v->iq_ref_I = v->iqmin - v->iq_ref_P;//,v->coefficient);
		}
		v->iq_ref = _IQmpy(v->iq_ref, v->coefficient);
}
```

The q-axis current reference value is fed to a current regulator to obtain the phase voltages required.

```
void IregLC_calc(IREGLC *v)

{
		v->ud_ref_P = _IQmpy((v->id_ref - v->id),v->K);
		v->ud_ref_I = _IQmpy(v->Ts_Ki_K, (v->id_ref - v->id)) + v->ud_ref_I;
		v->ud_ref = (v->ud_ref_P + v->ud_ref_I) - _IQmpy(_IQmpy(v->Kc, (v->iq_ref + v->iq)),_IQ(0.5)) + v->vd;

		v->uq_ref_P = _IQmpy((v->iq_ref - v->iq),v->K);
		v->uq_ref_I = _IQmpy(v->Ts_Ki_K, (v->iq_ref - v->iq)) + v->uq_ref_I;
		v->uq_ref = (v->uq_ref_P + v->uq_ref_I) + _IQmpy(_IQmpy(v->Kc, (v->id_ref + v->id)),_IQ(0.5)) + v->vq;

		if(v->ud_ref > v->umax)
		{
			v->ud_ref = v->umax;
			v->ud_ref_I = v->umax - (v->ud_ref_P - _IQmpy(_IQmpy(v->Kc, (v->iq_ref + v->iq)),_IQ(0.5)) + v->vd);
		}
		else if(v->ud_ref < v->umin)
		{
			v->ud_ref = v->umin;
			v->ud_ref_I = v->umin - (v->ud_ref_P - _IQmpy(_IQmpy(v->Kc, (v->iq_ref + v->iq)),_IQ(0.5)) + v->vd);
		}
		if(v->uq_ref > v->umax)
		{
			v->uq_ref = v->umax;
			v->uq_ref_I = v->umax - (v->uq_ref_P + _IQmpy(_IQmpy(v->Kc, (v->id_ref + v->id)),_IQ(0.5)) + v->vq);
		}
		else if(v->uq_ref < v->umin)
		{
			v->uq_ref = v->umin;
			v->uq_ref_I = v->umin - (v->uq_ref_P + _IQmpy(_IQmpy(v->Kc, (v->id_ref + v->id)),_IQ(0.5)) + v->vq);
		}
}
```

As the reference values call for both voltages in *dq* frame the regulator code is twice the size of the voltage regulator, which only outputs a current reference value on the q-axis.

The voltage reference values are transformed back to phase voltages using inverse Clarke and Park transformations:

Inverse Clarke transformation:

```
void idq_calc(IDQ *v)
{
     _iq Cosine,Sine;
// Using look-up IQ sine table
    Sine = _IQsin(v->Angle);
    Cosine = _IQcos(v->Angle);

                //Theta angle between alfa-axel och d-axel
    v->alfa = _IQmpy(v->d,Cosine) - _IQmpy(v->q,Sine);
    v->beta = _IQmpy(v->q,Cosine) +  _IQmpy(v->d,Sine);

}
```

Inverse Park transformation

```
void abc_calc(ABC *v)
{
            v->a = _IQmpy(_IQ(0.816497), v->alfa);                    //sqrt(2/3)
            v->b = _IQmpy(_IQ(-0.408248), v->alfa) +  _IQmpy(_IQ(0.707107), v-
>beta);      //sqrt(1/6)  sqrt(1/2)
            v->c = _IQmpy(_IQ(-0.408248), v->alfa) - _IQmpy(_IQ(0.707107), v-
>beta);      //sqrt(1/6)  sqrt(1/2)
}
```

These values are fed to the source file called duty.c which symmetrizises the sinusoidal reference wave and feeds it to a PWM circuit on the DSP. The generated PWM signal will then control the output voltage from the grid-connected converter in order to keep the reference value.

6.3.2 Keeping the grid voltage stable

Keeping the grid voltage stable implies feeding reactive power out from or in to the grid-side converter depending on if the grid voltage is below or exceeds the reference value.

The reference value for reactive current is created using a voltage regulator and the output is fed into the same current regulator regulating the active current.

The current regulator parameters used are modifications of standard regulator parameter values as explained in chapter 4.4.2.

```
void VACreg_calc(VACREG *v)
{
		v->id_ref = _IQmpy(v->Kp,(v->eq_ref-v->eq)) +  v->eq_I;
		//Anti-windup
		if(v->id_ref > v->idmax)
		{
		v->id_ref = v->idmax;
		v->eq_I = v->idmax - _IQmpy(v->Kp,(v->eq_ref-v->eq));
		}
		else if(v->id_ref < -v->idmax)
		{
		v->id_ref = -v->idmax;
		v->eq_I = -v->idmax - _IQmpy(v->Kp,(v->eq_ref-v->eq));
		}
		else
		{
		v->eq_I = _IQmpy(_IQmpy(v->Ki,v->Ts), (v->eq_ref-v->eq)) +  v-
>eq_I;
		}

  }
```

The generator side converter regulators work in parallel with the grid-side converters. The intended current reference value for the generator-side converter is based on the power-graph. However, as it involves lookup tables and speed measurements to complete, it has not been implemented. The active power requested from the generator-side converters is dependent on a manually controlled reference value, set with a potentiometer.

6.4 Protecting converters from over current and over voltage

In order to protect the converters from potential over current and over voltage, limitations for the measured voltage and current values are implemented.

The protection sends a stop signal to the PWM circuit halting the switching and effectively cutting the converter output current. High-order harmonics from the PWM process will trigger the high current protection without actually causing any stress on the converters. In order to handle such issues, the protection is set to respond to both a maximum limit, as well as a limit which can be triggered a number of times before halting the PWM.

```
void skydd_calc(SKYDD *v)
{               //limit 1
                if(_IQabs(v->signal)>v->max1)
                {
                                v->tripp1 = 1;
                }
                //limit 2
                if(_IQabs(v->signal) > v->max2)
                {
                                v->count++;
                }
                else if(_IQabs(v->signal) < v->reset)
                {
                                v->count--;
                }
                if(v->count<0)
                {
                                v->count = 0;
                }

                if(v->count > v->countmax)
                {
                                v->tripp2 = 1;
                }
```

The complete code is presented in the appendix.

7. Experiment

The experiment will determine if the wind turbine model can generate active power out on to the power grid, while aiding in the stabilizing of grid voltage that differentiates from its nominal value.

The voltage regulator running from the flash memory is set to produce reactive power to stabilize the network voltage back to its nominal value according to the following equation derived in chapter 3.4.3.

The expression shows that grid voltage is dependent on reactive power. As the grid droop characteristics for the voltage indicates, that grid voltage magnitude and reactive power are inversely proportional.

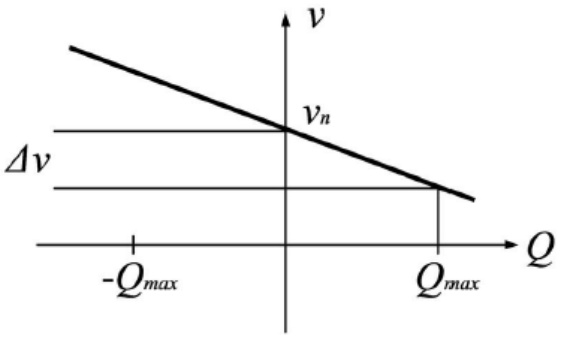


Figure 28 Grid voltage characteristics at reactive power flow, picture taken from [6]

Where v_n is the nominal voltage of the grid.

In order to simulate a voltage drop, a large inductive or resistive load can momentarily be connected to the power grid. However, as the grid in the laboratory is considered a strong power grid, i.e. low inductive and resistive components, this method would require a large inductive or resistive load.

The solution is to simulate a power grid, connecting the grid-side converter to the terminals of a 200 kVA synchronous generator. To adjust the generated voltage, the magnetization voltage will be adjusted to simulate both a lower and a higher grid voltage. Conducting the testing in this fashion will also allow for the voltage to increase from its nominal value.

8. Results

The following results are conducted where the grid voltage is not stabilized using the current regulator controlling reactive power flow.

8.1 Test without the grid voltage control

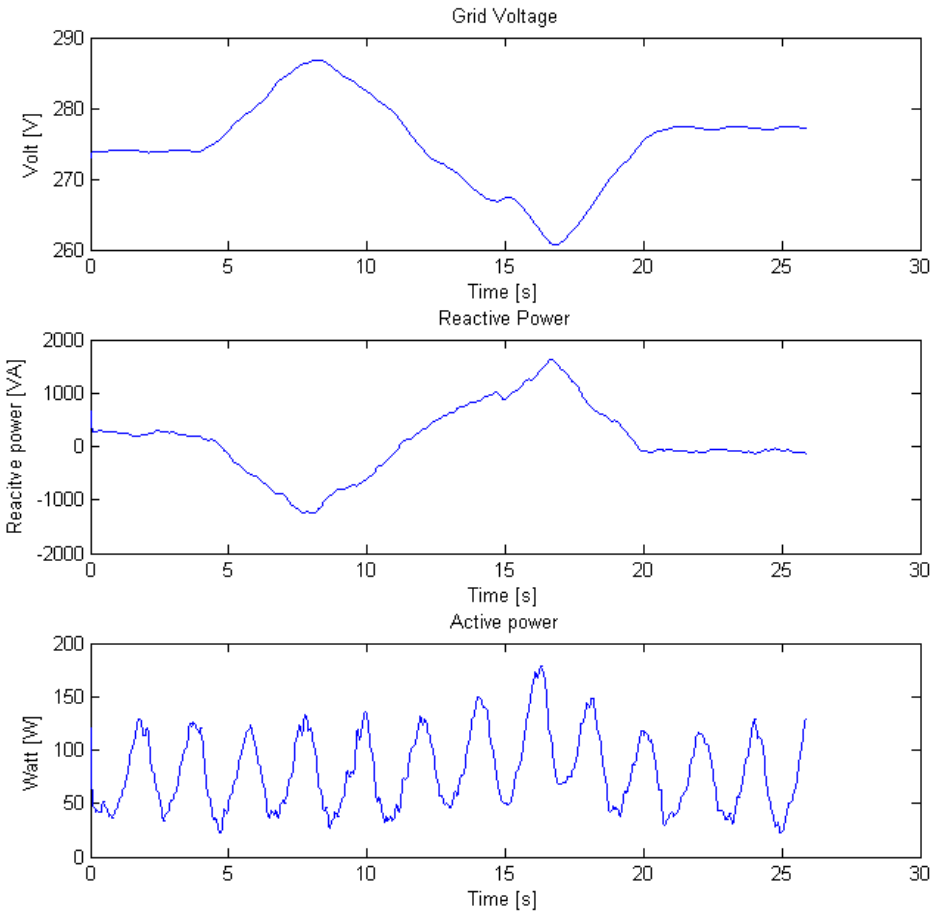


Figure 29 Test results without reactive power flow regulator

Varying the reactive power generated will cause voltage fluctuations corresponding to the inverse of the reactive power fed to or drawn from the grid. The active power is not dependent on the fluctuation, merely the grid frequency, which is constant. The active power displays fluctuations from the noise caused by the switching circuit and is therefore filtered to give it a smoother look.

8.2 Test of grid voltage control

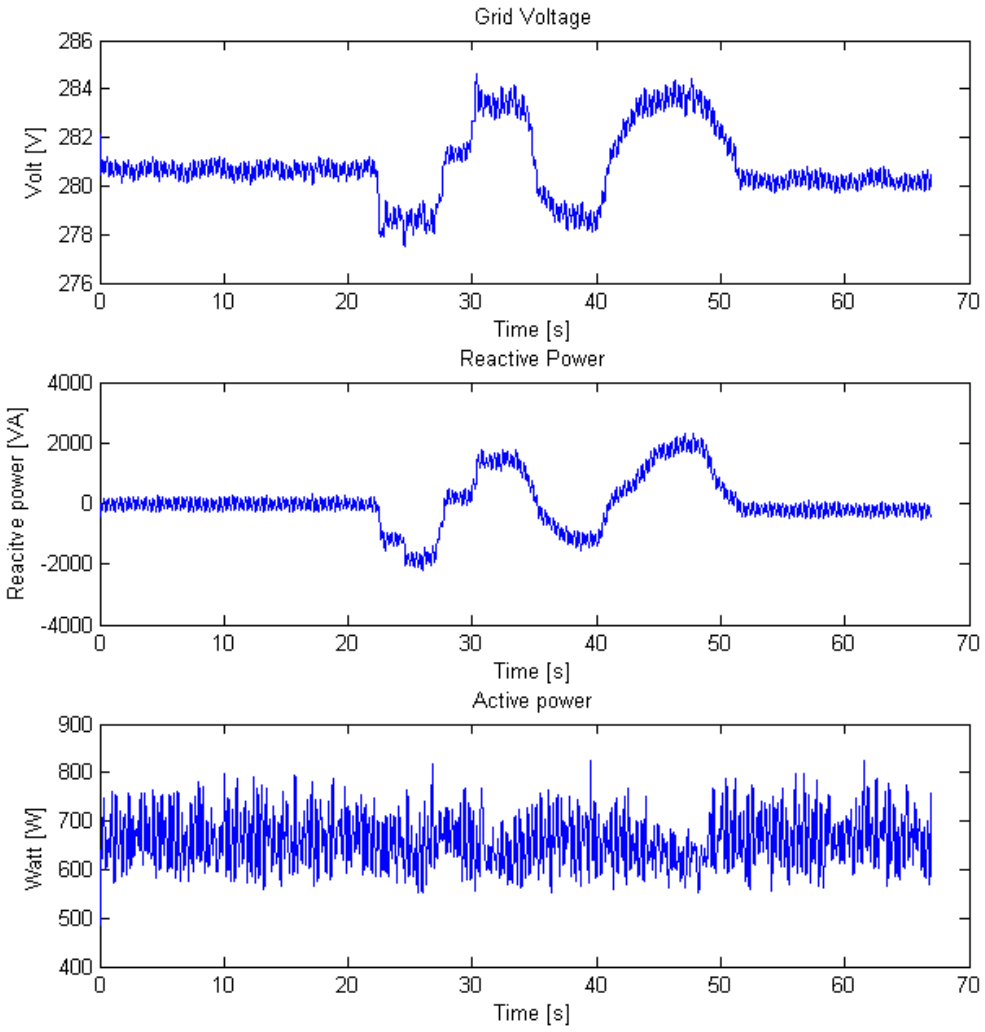


Figure 30 Test results with reactive power flow regulator

With the grid voltage regulation, the generated reactive power is proportional to the grid voltage variations as its actively attempting to suppress the variation. During this test, the active power is set to a higher value, to simulate voltage stabilization during active power generation.

The grid voltage is not entirely stable as its exhibiting a steady state error in the regulator indicating that the integral part is not correcting sufficiently.

The result is also heavily filtered to remove disturbances caused by the switching circuit.

9. Discussion

The back-to-back voltage source converters were successfully controlled using a control algorithm developed in the programming language C. The objective of the algorithm is to control the series connected converters to stabilize voltage fluctuations from the nominal grid voltage, while delivering active power to the grid. The voltage regulator controlling the reactive power responds correctly to the voltage reference deviation and feeds out reactive power on demand.

The control algorithm was also intended to be executed as an embedded system, independent of a secondary CPU, mainly the PC on which the algorithm is debugged from. This was successfully performed and the converters were controlled running the program from the flash memory. The process of executing the algorithm from the flash memory lowers the calculating time almost by half, as the memory sectors are read faster from the flash memory than as initially from its external memory.

The premise of controlling grid parameters such as lowering deviations from the grid nominal voltage on weaker grids allows for a wind turbine to have a greater role than simply supplying the grid with active power.

A reoccurring problem was the large disturbances, which could need further dampening using digital filters and the use of a Smith predictor.

References

[1] World Wind Energy Association (WWEA) “World Wind Energy Report 2009” 2009-03

[2] J.F. Manwell, J.G. McFowan, A.L. Rogers “Wind Energy Explained – Theory, Design and Application” 2:nd edition, 2009

[3] O. Anaya-Lara, N. Jenkins, J. Ekanaye, P. Cartwright, M. Hughes “Wind Energy Generation - Modelling and Control” 2009-01

[4] Article concerning electrical design of crowbars http://axotron.se/index_en.php?page=26

[5] M. Alaküla, P. Karlsson “Power electronics – Devices, Converters, Control and Applications” *Department of Electrical Engineering and Automation, Lund Institute of Technology*, Lund, Sweden

[6] J. Björnstedt, M. Ström “Power Electronic Voltage and Frequency Control for Distributed Generation System” *Department of Electrical Engineering and Automation, Lund Institute of Technology*, Lund, Sweden 2005

[7] O. Samuelsson “Kraftelektronik” *Department of Electrical Engineering and Automation, Lund Institute of Technology*, Lund, Sweden

[8] Texas Instruments “SPRS174R – Data Manual” 2001-04, revised 2010-05

[9] Texas Instruments “SPRA958H – Running an Application from Internal Flash Memory on the TMS320F28xx DSP” 2008-09

[10] Texas Instruments “C281x C/C++ Header Files and Peripheral Examples Quick Start” Version 1.2 2009-06-27

[11] Texas Instruments “SPRAAS1A – Hardware Design Guidelines for TMS320F28xx and TMS320F28xx DSCs” 2008-09

[12] Texas Instruments “C28x IQmath Library – A Virtual Floating Point Engine v1.5” 2008-06

[13] Texas Instruments “SPRU065D - TMS320x281x DSP Event Manager (EV) Reference Guide” 2004-11, Revised 2006-08

[14] Texas Instruments “SPRA989A - F2810, F2811 and F2812 ADC Calibration” 2004-11

[15] Texas Instruments “SPRU060D – TMS320x281x DSP Analog-to-Digital Converter (ADC) Reference Guide” 2003-06, revised 2005-07

[16] Texas Instruments “TMS320F2810, TMS320F2811 and TMS320F2812 SDFlash JTAG Programming Utilities” 2005-09

Appendix

```
/* ==========================================================================
System Name:  REG_JB

File Name:      REG_JB.C

Global Q 20 (-2047 - 2047)
============================================================================
*/

// Include header files used in the main function
#include "target.h"
#include "DSP281x_Device.h"
//#include "DSP281x_Examples.h"   // DSP281x Examples Include File
#include "IQmathLib.h"
#include "reg_jb.h"
#include "parameter.h"
#include "build.h"
#include <ADCcalibrationDriver.h>
#include <math.h>
/*
#pragma DATA_SECTION(out_buffer1, "DLOG");              //Logga data till externt minne
#pragma DATA_SECTION(out_buffer2, "DLOG");
#pragma DATA_SECTION(out_buffer3, "DLOG");
#pragma DATA_SECTION(out_buffer4, "DLOG");
#pragma DATA_SECTION(out_buffer5, "DLOG");
#pragma DATA_SECTION(out_buffer6, "DLOG");
*/
// Select the example to compile in.  Only one example should be set as 1
// the rest should be set as 0.
/*
#define EXAMPLE1 1  // Use DATA registers to toggle I/O's
#define EXAMPLE2 0  // Use SET/CLEAR registers to toggle I/O's
#define EXAMPLE3 0  // Use TOGGLE registers to toggle I/O's
*/
// Prototype statements for functions found within this file.
interrupt void MainISR(void);
interrupt void QepISR(void);
// Prototype statements for functions found within this file.
/*void delay_loop(void);
void Gpio_select(void);
void Gpio_example1(void);
void Gpio_example2(void);
void Gpio_example3(void);
*/
ADC_CALIBRATION_DRIVER_VARS adc;
```

```
// Assign this function to a section to be linked to internal RAM
#pragma CODE_SECTION(xintf_zone6and7_timing,"ramfuncs");
void xintf_zone6and7_timing(void);


// Global variables used in this system
//float32 Ts = 0.001/ISR_FREQUENCY;   // Samping period (sec), see parameter.h
float32 Ts = 0.000202;          //202us = 4.95kHz
//float32 L = 0.0075;
//float32 R = 0.05;

Uint16 IsrTicker = 0;
Uint16 sampletkr = 0;
Uint16 sample_on = 0;
Uint16 sample_off = 0;
Uint16 sample = 0;
Uint16 BackTicker = 0;
Uint16 JBTicker = 0;
Uint16 fel = 0;

int16 DlogCh1 = 0;
int16 DlogCh2 = 0;
int16 DlogCh3 = 0;
int16 DlogCh4 = 0;


//------JB----------
int16 duty1 = 0;
int16 duty2 = 0;
int16 duty3 = 0;
int16 Tmp1 =0;
int16 Tmp2 =0;
int trigg=0;

_iq tmp=_IQ(-0.00094464);
_iq tmp2=0;

Uint16 nisse=0;
Uint16 olle = 0;
int16 pelle = 0;
_iq kalle = 0;
int16 anna = 0;
//_iq pelle = 0;




_iq v1ab, v1bc, v2ab, v2bc, Udc;
_iq i1a, i1b, i2a, i2b;
```

```
_iq                pot1_old, pot2_old;
_iq offs_v1ab, offs_v1bc, offs_v2ab, offs_v2bc, offs_i1a, offs_i1b, offs_i2a, offs_i2b, offs_Udc;

_iq pot1, pot2;

_iq v1q, v2q, v2d, v2q, v2alfa, v2beta;

//För loggning till minnet
_iq logch1 = 0;
_iq logch2 = 0;
_iq logch3 = 0;
_iq logch4 = 0;
_iq logch5 = 0;
_iq logch6 = 0;
int reset=1;
int enable=0;
int size = BUFSIZE;
int out_buffer1[BUFSIZE*2];                    //32 bitar
int *output1 = &out_buffer1[0];
int out_buffer2[BUFSIZE*2];                    //32 bitar
int *output2 = &out_buffer2[0];
int out_buffer3[BUFSIZE*2];                    //32 bitar
int *output3 = &out_buffer3[0];
int out_buffer4[BUFSIZE*2];                    //32 bitar
int *output4 = &out_buffer4[0];
int out_buffer5[BUFSIZE*2];                    //32 bitar
int *output5 = &out_buffer5[0];
int out_buffer6[BUFSIZE*2];                    //32 bitar
int *output6 = &out_buffer6[0];

long GlobalQ = GLOBAL_Q;


//------------------


volatile Uint16 EnableFlag = FALSE;




// PWM omriktare 1
PWMGEN1 pwm1 = PWMGEN1_DEFAULTS;

// PWM omriktare 2
PWMGEN2 pwm2 = PWMGEN2_DEFAULTS;

// Beräkning av dutycycle
DUTY dutycycle = DUTY_DEFAULTS;
```

```
// Create an instance of the current/dc-bus voltage measurement driver
ACMEAS ac_vi = AC_MEAS_DEFAULTS;

// gain och offset på mätvärden från ADC
GAINOFFSET varib = GAINOFFSET_DEFAULTS;

// INITIATE TRANSFORMATION FROM AB BC TO ALFABETA
CLARKEVT v1alfabeta = CLARKEVT_DEFAULTS;
CLARKEVT v2alfabeta = CLARKEVT_DEFAULTS;

// INITIATE TRANSFORMATION FROM A B TO ALFABETA
CLARKEIT i1alfabeta = CLARKEIT_DEFAULTS;
CLARKEIT i2alfabeta = CLARKEIT_DEFAULTS;

// INITIATE TRANSFORMATION FROM ALFABETA TO DQ-FRAME FOR CURRENT VECTROS
CLARKEDQ i1dq = CLARKEDQ_DEFAULTS;
CLARKEDQ i2dq = CLARKEDQ_DEFAULTS;

// INITIATE TRANSFORMATION FROM ALFABETA TO DQ-FRAME FOR VOLTAGE VECTROS
//CLARKEDQ v1dq = CLARKEDQ_DEFAULTS;
CLARKEDQ v2dq = CLARKEDQ_DEFAULTS;

// INITIATE CALCULATING ANGLE BETWEEN ALFA AND D
ANGLE theta1 = ANGLE_DEFAULTS;
ANGLE theta2 = ANGLE_DEFAULTS;

// FILTER FIRST ORDER GRID VOLTAGE
FILTER Udc_filter = FILTER_DEFAULTS;

// FILTER FIRST ORDER PHASE REFERENCE VOLTAGE

FILTER V2dq_d_filter = FILTER_DEFAULTS;
FILTER V2dq_q_filter = FILTER_DEFAULTS;

// SECOND ORDER LOW PASS FILTERS FOR ALFABETA VOLATGE VECTORS
FILTER2 lpfilter_v1alfa = FILTER2_DEFAULTS;
FILTER2 lpfilter_v1beta = FILTER2_DEFAULTS;
FILTER2 lpfilter_v2alfa = FILTER2_DEFAULTS;
FILTER2 lpfilter_v2beta = FILTER2_DEFAULTS;

// INITIATE Vq CALCULATION FROM ALFABETA VECTORS
VQ Vqmag1 = VQ_DEFAULTS;
VQ Vqmag2 = VQ_DEFAULTS;

// INITIATE TRANSFORMATION FROM DQ FRAME TO ALFABETA
IDQ udq1_ref = IDQ_DEFAULTS;
IDQ udq2_ref = IDQ_DEFAULTS;

// INITIATE TRANSFORMATION FROM ALFABETA TO ABC
ABC u1alfabeta_ref = ABC_DEFAULTS;
ABC u2alfabeta_ref = ABC_DEFAULTS;
```

```
// INITIATE CURRENT REGULATOR FOR L FILTER IN DQ FRAME
IREGLC I_reg1 = IREGLC_DEFAULTS;
IREGLC I_reg2 = IREGLC_DEFAULTS;

// INTIATE DC VOLTAGE REGULATOR
VDCREGS Vdc_reg = VDCREGS_DEFAULTS;

// INITIATE PROTECTIVE MAGNITUDE LIMITATIONS

SKYDD I1q_skydd = SKYDD_DEFAULTS;
SKYDD I1d_skydd = SKYDD_DEFAULTS;
SKYDD I2q_skydd = SKYDD_DEFAULTS;
SKYDD I2d_skydd = SKYDD_DEFAULTS;
SKYDD Udc_skydd = SKYDD_DEFAULTS;

// OBSERVERARE

OBSERVERARE observerare = OBSERVERARE_DEFAULTS;


void main(void)
{
/*
// RUN FOR FLASH START

// Initialize System Control registers, PLL, WatchDog, Clocks to default state:
      // This function is found in the DSP281x_SysCtrl.c file.
                  InitSysCtrl();

// HISPCP prescale register settings, normally it will be set to default values
   EALLOW;   // This is needed to write to EALLOW protected registers
   SysCtrlRegs.HISPCP.all = 0x0000;     // SYSCLKOUT/1
   EDIS;   // This is needed to disable write to EALLOW protected registers

// Disable and clear all CPU interrupts:
                  DINT;
                  IER = 0x0000;
                  IFR = 0x0000;

// Initialize Pie Control Registers To Default State:
      // This function is found in the DSP281x_PieCtrl.c file.
                  InitPieCtrl();

// Initialize the PIE Vector Table To a Known State:
      // This function is found in DSP281x_PieVect.c.
                  // This function populates the PIE vector table with pointers
      // to the shell SR functions found in DSP281x_DefaultIsr.c.
                  InitPieVectTable();
// Copy time critical code and Flash setup code to RAM
// This includes the following ISR functions: EvaTimer1(), EvaTimer2()
// EvbTimer3 and and InitFlash();
// The  RamfuncsLoadStart, RamfuncsLoadEnd, and RamfuncsRunStart
```

```
// symbols are created by the linker. Refer to the F2812.cmd file.

// Call Flash Initialization to setup flash waitstates
// This function must reside in RAM
  InitFlash();
// User specific functions, Reassign vectors (optional), Enable Interrupts:

// Initialize EVA Timer 1:
   // Setup Timer 1 Registers (EV A)
   EvaRegs.GPTCONA.all = 0;

                  xintf_zone6and7_timing();

//RUN FOR FLASH END
*/
// ****************************************
// Initialization code for DSP_TARGET = F2812
// ****************************************

// Initialize System Control registers, PLL, WatchDog, Clocks to default state:
      // This function is found in the DSP281x_SysCtrl.c file.
                  InitSysCtrl();
                  // For this example use the following configuration:
 //Gpio_select();

// HISPCP prescale register settings, normally it will be set to default values
   EALLOW;   // This is needed to write to EALLOW protected registers
   SysCtrlRegs.HISPCP.all = 0x0000;     // SYSCLKOUT/1
   EDIS;   // This is needed to disable write to EALLOW protected registers

// Disable and clear all CPU interrupts:
                  DINT;

// Initialize Pie Control Registers To Default State:
      // This function is found in the DSP281x_PieCtrl.c file.
                  InitPieCtrl();

                  IER = 0x0000;
                  IFR = 0x0000;

// Initialize the PIE Vector Table To a Known State:
      // This function is found in DSP281x_PieVect.c.
                  // This function populates the PIE vector table with pointers
      // to the shell ISR functions found in DSP281x_DefaultIsr.c.
                  InitPieVectTable();

// User specific functions, Reassign vectors (optional), Enable Interrupts:

// Step 4. Initialize Calibration Driver Variables:
//
   ADCcalibrationDriverInit(&adc);
```

```
// Initialize EVA Timer 1:
   // Setup Timer 1 Registers (EV A)
  EvaRegs.GPTCONA.all = 0;

                    xintf_zone6and7_timing();
// Step 5. User specific code:
/*
#if EXAMPLE1

   // This example uses DATA registers to toggle I/O's
   Gpio_example1();

#endif  // - EXAMPLE1

#if EXAMPLE2

   // This example uses SET/CLEAR registers to toggle I/O's
   Gpio_example1();

#endif

#if EXAMPLE3

   // This example uses TOGGLE registers to toggle I/O's
   Gpio_example3();

#endif
*/
// Copy time critical code and Flash setup code to RAM
// This includes the following ISR functions: EvaTimer1(), EvaTimer2()
// EvbTimer3 and and InitFlash();
// The  RamfuncsLoadStart, RamfuncsLoadEnd, and RamfuncsRunStart
// symbols are created by the linker. Refer to the F2812.cmd file.
 //  MemCopy(&RamfuncsLoadStart, &RamfuncsLoadEnd, &RamfuncsRunStart);

// Call Flash Initialization to setup flash waitstates
// This function must reside in RAM
  // InitFlash();


//Initiera lysdioder
EALLOW;
GpioMuxRegs.GPBMUX.bit.C4TRIP_GPIOB13=0; //Right
GpioMuxRegs.GPBMUX.bit.C5TRIP_GPIOB14=0; //Center
GpioMuxRegs.GPBMUX.bit.C6TRIP_GPIOB15=0; //Left
GpioMuxRegs.GPBDIR.bit.GPIOB13=1; //Write
GpioMuxRegs.GPBDIR.bit.GPIOB14=1;
GpioMuxRegs.GPBDIR.bit.GPIOB15=1;
GpioDataRegs.GPBDAT.bit.GPIOB13=0; //1=ON, 0=OFF
GpioDataRegs.GPBDAT.bit.GPIOB14=1;
GpioDataRegs.GPBDAT.bit.GPIOB15=0;
EDIS;
```

```
//Initiera omkopplare
EALLOW;
GpioMuxRegs.GPAMUX.bit.C1TRIP_GPIOA13=0; //Button B3
GpioMuxRegs.GPAMUX.bit.C2TRIP_GPIOA14=0; //Button B2
GpioMuxRegs.GPAMUX.bit.C3TRIP_GPIOA15=0; //Button B1
GpioMuxRegs.GPADIR.bit.GPIOA13=0; //Read
GpioMuxRegs.GPADIR.bit.GPIOA14=0;
GpioMuxRegs.GPADIR.bit.GPIOA15=0;
//Read BUTTONS ON or OFF:
GpioDataRegs.GPADAT.bit.GPIOA13=1; //Pull-up
GpioDataRegs.GPADAT.bit.GPIOA14=1;
GpioDataRegs.GPADAT.bit.GPIOA15=1;
EDIS;


//Digitala utgångar
EALLOW;
//Enable signal till PWM
GpioMuxRegs.GPBMUX.bit.CAP4Q1_GPIOB8=0; //I/O mode
GpioMuxRegs.GPBDIR.bit.GPIOB8=1;// write
GpioDataRegs.GPBDAT.bit.GPIOB8=0;
//Digital utgång
GpioMuxRegs.GPBMUX.bit.CAP5Q2_GPIOB9=0; //I/O mode
GpioMuxRegs.GPBDIR.bit.GPIOB9=1;// write
GpioDataRegs.GPBDAT.bit.GPIOB9=0;
//Digital utgång
GpioMuxRegs.GPBMUX.bit.CAP6QI2_GPIOB10=0; //I/O mode
GpioMuxRegs.GPBDIR.bit.GPIOB10=1;// write
GpioDataRegs.GPBDAT.bit.GPIOB10=0;

EDIS;


  // Waiting for enable flag set
//   while (EnableFlag==FALSE)
//   {
//     BackTicker++;
//   }
//Borttaget JB

// Enable Underflow interrupt bits for GP timer 1
   EvaRegs.EVAIMRA.bit.T1UFINT = 1;
   EvaRegs.EVAIFRA.bit.T1UFINT = 1;

// Enable CAP3 interrupt bits for GP timer 2
   EvaRegs.EVAIMRC.bit.CAP3INT = 1;
   EvaRegs.EVAIFRC.bit.CAP3INT = 1;

// Reassign ISRs.
      // Reassign the PIE vector for T1UFINT and CAP3INT to point to a different
      // ISR then the shell routine found in DSP281x_DefaultIsr.c.
```

```
      // This is done if the user does not want to use the shell ISR routine
      // but instead wants to use their own ISR.

                  EALLOW;        // This is needed to write to EALLOW protected registers
                  PieVectTable.T1UFINT = &MainISR;
                  PieVectTable.CAPINT3 = &QepISR;
                  EDIS;   // This is needed to disable write to EALLOW protected registers

// Enable PIE group 2 interrupt 6 for T1UFINT
   PieCtrlRegs.PIEIER2.all = M_INT6;

// Enable PIE group 3 interrupt 7 for CAP3INT
   PieCtrlRegs.PIEIER3.all = M_INT7;

// Enable CPU INT2 for T1UFINT and INT3 for CAP3INT:
                  IER |= (M_INT2 | M_INT3);


// Initialize ADC module
   ac_vi.init(&ac_vi);

// SET OFFSET AND GAIN FOR VALUES PRESENTED FROM ADC REGISTERS
                  varib.offs_I1a = _IQ15(-0.0400);
   varib.offs_I1b = _IQ15(-0.0440);
                  varib.offs_V1ab = _IQ15(-0.0490);
   varib.offs_V1bc = _IQ15(-0.0420);
   varib.offs_I2a = _IQ15(-0.0400);
                  varib.offs_I2b = _IQ15(-0.0395);
   varib.offs_V2ab = _IQ15(-0.0445);
   varib.offs_V2bc = _IQ15(-0.0410);
   varib.offs_Udc = _IQ15(1);
                  varib.offs_pot1 = _IQ15(0);
                  varib.offs_pot2 = _IQ15(0);

                  varib.gain_I1a = _IQ15(-1*21.05*1.4143135*1.2488);
   varib.gain_I1b = _IQ15(-1*21.05*1.4143135*1.2348);
                  varib.gain_V1ab = _IQ15(-1*270*1.4143135*1.1364);
   varib.gain_V1bc = _IQ15(-1*270*1.4143135*1.1364);
   varib.gain_I2a = _IQ15(-1*21.05*1.4143135*1.2488);
                  varib.gain_I2b = _IQ15(-1*21.05*1.4143135*1.1625);
   varib.gain_V2ab = _IQ15(-1*270*1.4143135*1.1198);
   varib.gain_V2bc = _IQ15(-1*270*1.4143135*1.1198);
   varib.gain_Udc = _IQ15(304.1955);
                  varib.gain_pot1 = _IQ15(-1);
                  varib.gain_pot2 = _IQ15(-1);

// SET SECOND ORDER LOW PASS FILTER PARAMETERS
// - 3dB @ 50Hz
//-----------------------------------------------------------
// y = (y[k-1]*(2 + Ts*2w) - y[k-2] + x*(2w^2)/(1+Ts*2w+Ts^2*w^2)
// Ts = 1/f0 = 1/4.95kHz = 202 us
// w0 = 2*pi*f0 = 2*pi*50Hz = 314.51 rad/s
```

```
// y = y[k-1]*a1 + y[k-2]*a2 + x*b1
//-----------------------------------------------------------

                lpfilter_v1alfa.a1 = _IQ(1.8807);
                lpfilter_v1alfa.a2 = _IQ(-0.8842);
                lpfilter_v1alfa.b1 = _IQ(0.0073);

                lpfilter_v1beta.a1 = _IQ(1.8807);
                lpfilter_v1beta.a2 = _IQ(-0.8842);
                lpfilter_v1beta.b1 = _IQ(0.0073);

                lpfilter_v2alfa.a1 = _IQ(1.8807);
                lpfilter_v2alfa.a2 = _IQ(-0.8842);
                lpfilter_v2alfa.b1 = _IQ(0.0073);

                lpfilter_v2beta.a1 = _IQ(1.8807);
                lpfilter_v2beta.a2 = _IQ(-0.8842);
                lpfilter_v2beta.b1 = _IQ(0.0073);

                Udc_filter.wTs = _IQmpy(_IQ(30*2*PI), _IQ(Ts));
                V2dq_d_filter.wTs = _IQmpy(_IQ(100*2*PI), _IQ(Ts));
                V2dq_q_filter.wTs = _IQmpy(_IQ(100*2*PI), _IQ(Ts));

// SET PARAMETERS FOR CURRENT REGULATOR FOR L FILTER IN DQ FRAME GENERATOR
                I_reg1.Ts_Ki_K = _IQ(0.7500); // K *Ts * 1/(Ts/(L/R + Ts/2)) = (37.12 with comp of 0.2)
*202e-6 * 202e-6/(7.5e-3/0.075 + 202e-6/2) = 0.0056
                I_reg1.Kc = _IQ(2.3562); // wL = 2.3562
                I_reg1.K = _IQmpy(_IQ(37.1250),_IQ(0.2)); //L/Ts + R/2 = 7.5e-3/202e-6 + 2e-3/2 =
37.12 with comp of 0.8
                I_reg1.umax = _IQ(280);
                I_reg1.umin = _IQ(-280);

// SET PARAMETERS FOR CURRENT REGULATOR FOR L FILTER IN DQ FRAME GRID
                I_reg2.Ts_Ki_K = _IQ(0.7500); // K *Ts * 1/(Ts/(L/R + Ts/2)) = (37.12 with comp of 0.2)
*202e-6 * 202e-6/(7.5e-3/0.075 + 202e-6/2) = 0.0056
                I_reg2.Kc = _IQ(2.3562); // wL = 2.3562
                I_reg2.K = _IQmpy(_IQ(37.1250),_IQ(0.2)); //L/Ts + R/2 = 7.5e-3/202e-6 + 2e-3/2 =
37.12 with comp of 0.8
                I_reg2.umax = _IQ(280);
                I_reg2.umin = _IQ(-280);

// SET PARAMETERS FOR DC VOLTAGE REGULATOR
                Vdc_reg.Ts_Ki_Kp = _IQ(0.00025); // Ts*Ki*Kp = Ts*(9/wlp)*(Cdc*wlp/3) = 202e-
6*(9/2*30*pi)*940e-6*(2*30*pi)/3 = 0.00025
                Vdc_reg.Kp = _IQ(0.0591); // Cdc*wlp/3 = 940e-6*(2*30*pi/3) = 0.0591
                Vdc_reg.Udc_ref = _IQ(400);
                Vdc_reg.iqmax = _IQ(2);
                Vdc_reg.iqmin = _IQ(-2);


// SET PARAMETERS FOR VOLTAGE AND CURRENT PROTECTION
                I1d_skydd.countmax = 2;
```

```
                I1d_skydd.max1 = _IQ(15);
                I1d_skydd.max2 = _IQ(10);
                I1d_skydd.reset = _IQ(9);

                I1q_skydd.countmax = 2;
                I1q_skydd.max1 = _IQ(20);
                I1q_skydd.max2 = _IQ(15);
                I1q_skydd.reset = _IQ(14);

                I2d_skydd.countmax = 2;
                I2d_skydd.max1 = _IQ(15);
                I2d_skydd.max2 = _IQ(10);
                I2d_skydd.reset = _IQ(9);

                I2q_skydd.countmax = 2;
                I2q_skydd.max1 = _IQ(15);
                I2q_skydd.max2 = _IQ(10);
                I2q_skydd.reset = _IQ(9);

                Udc_skydd.countmax = 2;
                Udc_skydd.max1 = _IQ(500);
                Udc_skydd.max2 = _IQ(450);
                Udc_skydd.reset = _IQ(440);

                observerare.Eqn = _IQ(230);
                observerare.L = _IQ(0.0075);
                observerare.R = _IQ(0.00075);
                observerare.Ts = _IQ(0.000202);
                observerare.raw2_Ts = _IQ(0.228749);
                observerare.raw_Ts_2 = _IQ(0.013596);
                observerare.thetamax = _IQ(2*PI);
                observerare.wmax = _IQ(2*PI*51);
                observerare.wmin = _IQ(2*PI*49);


// Initialize PWM module
   pwm1.PeriodMax = SYSTEM_FREQUENCY*1000000*Ts/2;  // Perscaler X1 (T1), ISR period = T x 1
                pwm1.init(&pwm1);

   pwm2.PeriodMax = SYSTEM_FREQUENCY*1000000*Ts/2;
                pwm2.init(&pwm2);


// Enable global Interrupts and higher priority real-time debug events:
                EINT;   // Enable Global interrupt INTM
                ERTM;           // Enable Global realtime interrupt DBGM

//Externt minne
                XintfRegs.XINTCNF2.all=0x00004C17;
//XintfRegs.XTIMING6.all=0x00031629;
                XintfRegs.XTIMING6.all=0x00031425;          //tiden för Read ocg Write förkortad
```

```
                EvbRegs.ACTRB.all = PWM_DISSABLE;          //dissable DC pwm


GpioDataRegs.GPBDAT.bit.GPIOB13=1;
// IDLE loop. Just sit and loop forever:
                for(;;) BackTicker++;

}

interrupt void MainISR(void)
{
                sampletkr++;
// Verifying the ISR
   IsrTicker++;
GpioDataRegs.GPBDAT.bit.GPIOB10 = 1;


// -------------------Läs från AD omvandlare-----------------------------------

                //---------------------------------------------------------------------
                // Read ADC results, calibrate & store in "adc" memory structure:
                //---------------------------------------------------------------------
                ADCcalibrationDriverUpdate(&adc);          // Benchmark = 137 cycles

                varib.I1a = adc.ch0;
   varib.I1b = adc.ch1;
                varib.V1ab = adc.ch2;
   varib.V1bc = adc.ch3;
   varib.I2a = adc.ch4;
                varib.I2b = adc.ch5;
   varib.V2ab = adc.ch6;
   varib.V2bc = adc.ch7;

                varib.Udc = adc.ch8;
                varib.vgen = adc.ch9;
                varib.pot1 = adc.ch10;
                varib.pot2 = adc.ch11;

                varib.calc(&varib);


                v1ab = _IQ15toIQ(varib.V1abo);
                v1bc = _IQ15toIQ(varib.V1bco);
                i1a = _IQ15toIQ(varib.I1ao);
                i1b = _IQ15toIQ(varib.I1bo);

                v2ab = _IQ15toIQ(varib.V2abo);
                v2bc = _IQ15toIQ(varib.V2bco);
                i2a = _IQ15toIQ(varib.I2ao);
                i2b = _IQ15toIQ(varib.I2bo);
                pot1 = _IQ15toIQ(varib.pot1o);
                pot2 = _IQ15toIQ(varib.pot2o);
```

```
		Udc = _IQ15toIQ(varib.Udco);


// -------------------Start och stopp---------------------------------------------
		//undvik kontaktstudsar, fördröjd start 1s
		if(GpioDataRegs.GPADAT.bit.GPIOA13==0)
		{
			sample_on++;
		}
		else
		{
			sample_on = 0;
		}

		if(sample_on == 5000 && enable==0)	//STARTA
		{

			sampletkr = 0;
			fel=0;
			trigg=0;
			observerare.w_hat_old = _IQ(2*PI*50);
			observerare.theta_hat = theta2.angle + _IQ(1.570796);
			observerare.theta_hat_old = theta2.angle + _IQ(1.570796);
			Vdc_reg.iq_ref_I = 0;
			I_reg1.ud_ref_I = 0;
			I_reg1.uq_ref_I = 0;
			I_reg2.ud_ref_I = 0;
			I_reg2.uq_ref_I = 0;
			I1d_skydd.tripp1 = 0;
			I1d_skydd.tripp2 = 0;
			I1q_skydd.tripp1 = 0;
			I1q_skydd.tripp2 = 0;
			I2d_skydd.tripp1 = 0;
			I2d_skydd.tripp2 = 0;
			I2q_skydd.tripp1 = 0;
			I2q_skydd.tripp2 = 0;
			Udc_skydd.tripp1 = 0;
			Udc_skydd.tripp2 = 0;
			GpioDataRegs.GPBDAT.bit.GPIOB8=1;
			//Starta PWM
			GpioDataRegs.GPBDAT.bit.GPIOB15=0;
			//återställning diod 2
			GpioDataRegs.GPBDAT.bit.GPIOB14=0;
			//återställning diod 3
			enable = 1;
			sample_on = 0;
			sample_off = 0;

			//EvaRegs.ACTRA.all = PWM_ENABLE;	//enable 1 pwm
			EvbRegs.ACTRB.all = PWM_ENABLE;	//enable 2 pwm
		}
```

```
		if(GpioDataRegs.GPADAT.bit.GPIOA13==1 && enable==1)
		{
			GpioDataRegs.GPBDAT.bit.GPIOB8=0;			//Stoppa
PWM
			EvaRegs.ACTRA.all = PWM_DISSABLE;			//dissable AC
pwm
			EvbRegs.ACTRB.all = PWM_DISSABLE;			//dissable DC
pwm

			enable = 0;
		}


// -------------------Blinka-----------------------------------------------------
		if(IsrTicker==1000 && GpioDataRegs.GPBDAT.bit.GPIOB13==0)
		{
			GpioDataRegs.GPBDAT.bit.GPIOB13=1;
			IsrTicker=0;
		}
		if(IsrTicker==1000 && GpioDataRegs.GPBDAT.bit.GPIOB13==1)
		{
			GpioDataRegs.GPBDAT.bit.GPIOB13=0;
			IsrTicker=0;
		}

		//-----------------------------------------------------------------------
		// Transform from Vab and Vbc to alfabeta frame
		//-----------------------------------------------------------------------
		v1alfabeta.ab = v1ab;
		v1alfabeta.bc = v1bc;
		v1alfabeta.calc(&v1alfabeta);

		v2alfabeta.ab = v2ab;
		v2alfabeta.bc = v2bc;
		v2alfabeta.calc(&v2alfabeta);

		//-----------------------------------------------------------------------
		// Filter alfabeta signals with second order lowpass filter
		//-----------------------------------------------------------------------
		lpfilter_v1alfa.x = v1alfabeta.alfa;
		lpfilter_v1beta.x = v1alfabeta.beta;
		lpfilter_v1alfa.calc(&lpfilter_v1alfa);
		lpfilter_v1beta.calc(&lpfilter_v1beta);

		lpfilter_v2alfa.x = v2alfabeta.alfa;
		lpfilter_v2beta.x = v2alfabeta.beta;
		lpfilter_v2alfa.calc(&lpfilter_v2alfa);
		lpfilter_v2beta.calc(&lpfilter_v2beta);

		//-----------------------------------------------------------------------
		// Calculate Vq1 and Vq2 from magnitude of voltage alfabeta vectors
		//-----------------------------------------------------------------------
```

```
/*
		Vqmag1.alfa = lpfilter_v1alfa.y;
		Vqmag1.beta = lpfilter_v1beta.y;
		Vqmag1.calc(&Vqmag1);
		v1q = Vqmag1.q;


		Vqmag2.alfa = lpfilter_v2alfa.y;
		Vqmag2.beta = lpfilter_v2beta.y;
		Vqmag2.calc(&Vqmag2);
		v2q = Vqmag2.q;
*/
		v2dq.alfa = v2alfabeta.alfa;
		v2dq.beta = v2alfabeta.beta;
		v2dq.Angle = observerare.theta_hat - _IQ(1.570796);
		v2dq.calc(&v2dq);


		//-----------------------------------------------------------------
		// Calculate angle theta; angle between alfa and d in dq-frame
		//-----------------------------------------------------------------
		theta1.alfa = lpfilter_v1alfa.y;
		theta1.beta = lpfilter_v1beta.y;
		theta1.calc(&theta1);

		theta2.alfa = lpfilter_v2alfa.y;
		theta2.beta = lpfilter_v2beta.y;
		theta2.calc(&theta2);

		observerare.calc(&observerare);
		//-----------------------------------------------------------------
		// Transform from ia and ib to alfabeta
		//-----------------------------------------------------------------
		i1alfabeta.a = i1a;
		i1alfabeta.b = i1b;
		i1alfabeta.calc(&i1alfabeta);

		i2alfabeta.a = i2a;
		i2alfabeta.b = i2b;
		i2alfabeta.calc(&i2alfabeta);

		//-----------------------------------------------------------------
		// Transform from current vectors from alfabeta to dq-frame
		//-----------------------------------------------------------------
		i1dq.alfa = i1alfabeta.alfa;
		i1dq.beta = i1alfabeta.beta;
		i1dq.Angle = theta1.angle;
		i1dq.calc(&i1dq);

		i2dq.alfa = i2alfabeta.alfa;
		i2dq.beta = i2alfabeta.beta;
		i2dq.Angle = observerare.theta_hat - _IQ(1.570796); //theta2.angle;
```

```
                i2dq.calc(&i2dq);


                I1q_skydd.signal = i1dq.q;
                I1q_skydd.calc(&I1q_skydd);
                I1d_skydd.signal = i1dq.d;
                I1d_skydd.calc(&I1d_skydd);

                I2q_skydd.signal = i2dq.q;
                I2q_skydd.calc(&I2q_skydd);
                I2d_skydd.signal = i2dq.d;
                I2d_skydd.calc(&I2d_skydd);
                Udc_skydd.signal = Udc;
                Udc_skydd.calc(&Udc_skydd);

if(fel < 1) {

                if (I1d_skydd.tripp1 == 1 || I1d_skydd.tripp2 == 1)
                {
                                fel = 8;
                }
                if (I1q_skydd.tripp1 == 1 || I1q_skydd.tripp2 == 1)
                {
                                fel = 9;
                }
                if (I2d_skydd.tripp1 == 1 || I2d_skydd.tripp2 == 1)
                {
                                fel = 10;
                }
                if (I2q_skydd.tripp1 == 1 || I2q_skydd.tripp2 == 1)
                {
                                fel = 11;
                }
                if (Udc_skydd.tripp1 == 1 || Udc_skydd.tripp2 == 1)
                {
                                fel = 12;
                }
}

//--------------------STOPPA PWM-------------------------------------------------
                if(fel>0)
                {
                                GpioDataRegs.GPBDAT.bit.GPIOB15=1;              //lys vid fel
diod 2
                                EvaRegs.ACTRA.all = PWM_DISSABLE;               //dissable AC
pwm
                                EvbRegs.ACTRB.all = PWM_DISSABLE;               //dissable DC
pwm
                                GpioDataRegs.GPBDAT.bit.GPIOB8=0;               //Stoppa
PWM
                                trigg=1;
                }
```

```
/*              if(sampletkr>10)
                {
                                GpioDataRegs.GPBDAT.bit.GPIOB14=1;              //lys vid fel
diod 3
                                EvaRegs.ACTRA.all = PWM_DISSABLE;               //dissable AC
pwm
                                EvbRegs.ACTRB.all = PWM_DISSABLE;               //dissable DC
pwm
                                GpioDataRegs.GPBDAT.bit.GPIOB8=0;               //Stoppa
PWM
                                trigg=1;
                }
*/

// FILTER VOLTAGE OVER DC CAPACITOR

                Udc_filter.x = Udc;
                Udc_filter.calc(&Udc_filter);

// USE VOLTAGE REGULATOR TO OBTAIN CHARGING CURRENT IN DQ - FRAME GRID
                Vdc_reg.Udc = Udc_filter.y;
                Vdc_reg.vq = v2q;
                Vdc_reg.calc(&Vdc_reg);
/*
// USE VOLTAGE REGULATOR TO OBTAIN CHARGING CURRENT IN DQ - FRAME GENERATOR
                Vdc_reg.Udc = Udc_filter.y;
                Vdc_reg.vq = v1q;
                Vdc_reg.calc(&Vdc_reg);

// USE CURRENT REGULATOR TO OBTAIN THE VOLTAGE NEEDED IN DQ - FRAME GENERATOR
                I_reg1.iq_ref = -Vdc_reg.iq_ref;
                I_reg1.id_ref = _IQ(0);
                I_reg2.vd = v1dq.d;
                I_reg1.vq = v1dq.q;
                I_reg1.id = i1dq.d;
                I_reg1.iq = i1dq.q;
                I_reg1.calc(&I_reg1);
*/
// USE CURRENT REGULATOR TO OBTAIN THE VOLTAGE NEEDED IN DQ - FRAME GRID
                I_reg2.iq_ref = -Vdc_reg.iq_ref;
                I_reg2.id_ref = _IQmpy(pot2,_IQ(20));
                I_reg2.vd = v2dq.d;
                I_reg2.vq = v2dq.q;
                I_reg2.id = i2dq.d;
                I_reg2.iq = i2dq.q;
                I_reg2.calc(&I_reg2);

                observerare.ud_ref_old = I_reg2.ud_ref;
                observerare.id_ref_old = I_reg2.id_ref;
                observerare.iq_ref_old = I_reg2.iq_ref;

// FILTER VOLTAGE REFERENCES IN DQ FRAME
```

```
                V2dq_d_filter.x = I_reg2.ud_ref;
                V2dq_d_filter.calc(&V2dq_d_filter);

                V2dq_q_filter.x = I_reg2.uq_ref;
                V2dq_q_filter.calc(&V2dq_q_filter);


// TRANSFORM VOLTAGE REQUIRED FROM DQ - FRAME TO ALFABETA COORDINATES GRID
                udq2_ref.d = V2dq_d_filter.y;
                udq2_ref.q = V2dq_q_filter.y;
                udq2_ref.Angle = observerare.theta_hat - _IQ(1.570796);
                udq2_ref.calc(&udq2_ref);

// TRANSFORM VOLTAGE REQUIRED FROM ALFABETA COORDINATES TO PHASE VOLTAGES
                u2alfabeta_ref.alfa = udq2_ref.alfa;
                u2alfabeta_ref.beta = udq2_ref.beta;
                u2alfabeta_ref.calc(&u2alfabeta_ref);


                // -------------------Symetrisering och beräkning av dytycycle-------------------
                                dutycycle.ua_ref = u2alfabeta_ref.a;
                                dutycycle.ub_ref = u2alfabeta_ref.b;
                                dutycycle.uc_ref = u2alfabeta_ref.c;
                                dutycycle.Udc = Udc_filter.y;
                                dutycycle.umax = _IQ(200);
                                dutycycle.calc(&dutycycle);
/*


// TRANSFORM VOLTAGE REQUIRED FROM DQ - FRAME TO ALFABETA COORDINATES GENERATOR
                udq1_ref.d = _IQ(0);//I_reg1.ud_ref;
                udq1_ref.q = _IQmpy(pot2,_IQ(200));
                udq1_ref.Angle = theta1.angle;
                udq1_ref.calc(&udq1_ref);

// TRANSFORM VOLTAGE REQUIRED FROM ALFABETA COORDINATES TO PHASE VOLTAGES
GENERATOR
                u1alfabeta_ref.alfa = udq1_ref.alfa;
                u1alfabeta_ref.beta = udq1_ref.beta;
                u1alfabeta_ref.calc(&u1alfabeta_ref);


                // -------------------Symetrisering och beräkning av dytycycle GENERATOR-------------------
                                dutycycle.ua_ref = u1alfabeta_ref.a;
                                dutycycle.ub_ref = u1alfabeta_ref.b;
                                dutycycle.uc_ref = u1alfabeta_ref.c;
                                dutycycle.Udc = Udc_filter.y;
                                dutycycle.umax = _IQ(200);
                                dutycycle.calc(&dutycycle);
*/              // -------------------Skicka dytycycle till omriktare 1----------------------------
```

```
                pwm1.MfuncC1 = (int16)_IQtoIQ15(dutycycle.duty1);
                pwm1.MfuncC2 = (int16)_IQtoIQ15(dutycycle.duty2);
                                pwm1.MfuncC3 = (int16)_IQtoIQ15(dutycycle.duty3);
                                pwm1.update(&pwm1);


                // --------------------Skicka dytycycle till omriktare 2------------------------------
                                pwm2.MfuncC1 = (int16)_IQtoIQ15(dutycycle.duty1);
                pwm2.MfuncC2 = (int16)_IQtoIQ15(dutycycle.duty2);
                                pwm2.MfuncC3 = (int16)_IQtoIQ15(dutycycle.duty3);
                                pwm2.update(&pwm2);



// --------------------Välj signaler att logga-----------------------------------

                logch1 = i2dq.d;
                logch2 = i2dq.q;
                logch3 = observerare.theta_hat;
                logch4 = theta2.angle;
   logch5 = observerare.w_hat;
                logch6 = Udc_filter.y;




// --------------------Loggning till minnet---------------------------------------
                //Logga
                if(enable==1 && size>0 && GpioDataRegs.GPADAT.bit.GPIOA14==0)      //knapp2
                {
                reset=0;
                                *output1++ = logch1;                                        //LSB
                                *output1++ = logch1 >>16;                   //MSB

                                *output2++ = logch2;                                        //LSB
                                *output2++ = logch2 >>16;                   //MSB

                                *output3++ = logch3;                                        //LSB
                                *output3++ = logch3 >>16;                   //MSB

                                *output4++ = logch4;                                        //LSB
                                *output4++ = logch4 >>16;                   //MSB

                                *output5++ = logch5;                                        //LSB
                                *output5++ = logch5 >>16;                   //MSB

                                *output6++ = logch6;                                        //LSB
                                *output6++ = logch6 >>16;                   //MSB
                                size--;
   }
```

```
		//Använd om samplingen ska starta om igen
		if(trigg==0 && size==0)
		{
				output1 = &out_buffer1[0];
				output2 = &out_buffer2[0];
				output3 = &out_buffer3[0];
				output4 = &out_buffer4[0];
				output5 = &out_buffer5[0];
				output6 = &out_buffer6[0];
				size = BUFSIZE;
		}


		//Skicka buffert och återställ pekare till början
		if(enable==0 && reset==0)    //omkopplare 1
		{
				output1 = &out_buffer1[0];
				output2 = &out_buffer2[0];
				output3 = &out_buffer3[0];   //Sätt SW-probepoint här!
				output4 = &out_buffer4[0];
				output5 = &out_buffer5[0];
				output6 = &out_buffer6[0];
				size = BUFSIZE;
				reset = 1;
				trigg = 0;



		}


// Enable more interrupts from this timer
		EvaRegs.EVAIMRA.bit.T1UFINT = 1;

// Note: To be safe, use a mask value to write to the entire
		// EVAIFRA register.  Writing to one bit will cause a read-modify-write
		// operation that may have the result of writing 1's to clear
		// bits other then those intended.
   EvaRegs.EVAIFRA.all = BIT9;

// Acknowledge interrupt to recieve more interrupts from PIE group 2
		PieCtrlRegs.PIEACK.all |= PIEACK_GROUP2;

GpioDataRegs.GPBDAT.bit.GPIOB10 = 0;
}



interrupt void QepISR(void)
{

// Enable more interrupts from this timer
```

```
                EvaRegs.EVAIMRC.bit.CAP3INT = 1;

// Note: To be safe, use a mask value to write to the entire
                // EVAIFRC register.  Writing to one bit will cause a read-modify-write
                // operation that may have the result of writing 1's to clear
                // bits other then those intended.
   EvaRegs.EVAIFRC.all = BIT2;

// Acknowledge interrupt to recieve more interrupts from PIE group 3
                PieCtrlRegs.PIEACK.all |= PIEACK_GROUP3;

}


// Configure the timing paramaters for Zone 6 and Zone 7.
// Note: this function should not be executed from the same
// zones as those being configured.
void xintf_zone6and7_timing()
{

   // All Zones---------------------------------
   // Timing for all zones based on XTIMCLK = SYSCLKOUT
   XintfRegs.XINTCNF2.bit.XTIMCLK = 0;
   // Buffer up to 3 writes
   XintfRegs.XINTCNF2.bit.WRBUFF = 3;
   // XCLKOUT is enabled
   XintfRegs.XINTCNF2.bit.CLKOFF = 0;
   // XCLKOUT = XTIMCLK
   XintfRegs.XINTCNF2.bit.CLKMODE = 0;


   // Zone 6------------------------------------
   // When using ready, ACTIVE must be 1 or greater
   // Lead must always be 1 or greater
   // Zone write timing
   XintfRegs.XTIMING6.bit.XWRLEAD = 1;
   XintfRegs.XTIMING6.bit.XWRACTIVE = 1;
   XintfRegs.XTIMING6.bit.XWRTRAIL = 1;
   // Zone read timing
   XintfRegs.XTIMING6.bit.XRDLEAD = 1;
   XintfRegs.XTIMING6.bit.XRDACTIVE = 2;
   XintfRegs.XTIMING6.bit.XRDTRAIL = 0;

   // do not double all Zone read/write lead/active/trail timing
   XintfRegs.XTIMING6.bit.X2TIMING = 0;

   // Zone will not sample READY
   XintfRegs.XTIMING6.bit.USEREADY = 0;
   XintfRegs.XTIMING6.bit.READYMODE = 0;

   // Size must be 1,1 - other values are reserved
   XintfRegs.XTIMING6.bit.XSIZE = 3;
```

```
    // Zone 7----------------------------------
    // When using ready, ACTIVE must be 1 or greater
    // Lead must always be 1 or greater
    // Zone write timing
    XintfRegs.XTIMING7.bit.XWRLEAD = 1;
    XintfRegs.XTIMING7.bit.XWRACTIVE = 1;
    XintfRegs.XTIMING7.bit.XWRTRAIL = 1;
    // Zone read timing
    XintfRegs.XTIMING7.bit.XRDLEAD = 1;
    XintfRegs.XTIMING7.bit.XRDACTIVE = 2;
    XintfRegs.XTIMING7.bit.XRDTRAIL = 0;

    // don't double all Zone read/write lead/active/trail timing
    XintfRegs.XTIMING7.bit.X2TIMING = 0;

    // Zone will not sample XREADY signal
    XintfRegs.XTIMING7.bit.USEREADY = 0;
    XintfRegs.XTIMING7.bit.READYMODE = 0;

    // Size must be 1,1 - other values are reserved
    XintfRegs.XTIMING7.bit.XSIZE = 3;

   //Force a pipeline flush to ensure that the write to
   //the last register configured occurs before returning.
   asm(" RPT #7 || NOP");
}

void delay_loop()
{
    short     i;
    for (i = 0; i < 1000; i++) {}
}




void Gpio_select(void)
{

    Uint16 var1;
    Uint16 var2;
    Uint16 var3;

    var1= 0x0000;                           // sets GPIO Muxs as I/Os
    var2= 0xFFFF;                           // sets GPIO DIR as outputs
    var3= 0x0000;                           // sets the Input qualifier values

    EALLOW;
```

```
                GpioMuxRegs.GPAMUX.all=var1;
    GpioMuxRegs.GPBMUX.all=var1;
    GpioMuxRegs.GPDMUX.all=var1;
    GpioMuxRegs.GPFMUX.all=var1;
    GpioMuxRegs.GPEMUX.all=var1;
    GpioMuxRegs.GPGMUX.all=var1;


    GpioMuxRegs.GPADIR.all=var2;                    // GPIO PORTs  as output
    GpioMuxRegs.GPBDIR.all=var2;                    // GPIO DIR select GPIOs as output
    GpioMuxRegs.GPDDIR.all=var2;
    GpioMuxRegs.GPEDIR.all=var2;
    GpioMuxRegs.GPFDIR.all=var2;
    GpioMuxRegs.GPGDIR.all=var2;

    GpioMuxRegs.GPAQUAL.all=var3;                   // Set GPIO input qualifier values
    GpioMuxRegs.GPBQUAL.all=var3;
    GpioMuxRegs.GPDQUAL.all=var3;
    GpioMuxRegs.GPEQUAL.all=var3;

    EDIS;

}
//===========================================================================
// No more.
//===========================================================================
```

Flashing the DSP

This report aims to convey how to properly configure a non-DSP/BIOS program to be executed from its internal flash memory. The hardware used in this report is the DSP320F2812 eZdsp and the software used is both CCS v3.1.0 (Code Composer Studio) and SDFlash v1.63.00.

1 Introduction

Executing a program from the internal flash memory allows the software to be booted from onboard non-volatile memory and run from its onboard CPU independently of both external processors and memory. The adjustments to the code that previously ran using both internal RAM and external memory are relatively small considering the already written code. The hardware configuration requires only a jumper to be set at a different position to instruct from which address the execution starts.

2 The replacement and addition of files and code modification

2.1 Running SDConfig

SDConfig allows you to interface with the DSP. Under the Emulator tab there is a Test option, which will report if the DSP is correctly set up, otherwise it will provide a checklist to fix any errors causing the hardware to malfunction.

2.2 Replacing the command file

The command files (*.cmd) defines the memory blocks by declaring start address, size and how the code sections are linked to it. The command file, previously only defining the RAM and external memory has to incorporate the flash memory and the new section linking.

The distinction of whether to link a section to the flash memory or RAM depends on the section properties. If at device power-up, a section needs to have defined values; constants and application code then it's an *initialized* section. Initialized sections are to be linked to the flash memory. Sections as variables or register values, that don't require valid values at device power-up are linked to the *uninitialized* section, i.e. the RAM as it in contrast to the flash memory does not retain any data after device power-down.

The alterations to the command file can be made via the instructions in spra9582h. However replacing the command file F2812_EzDSP_RAM_lnk.cmd with F2812.cmd is sufficient, as the required changes have already been made. The command file is located in the DSP281x_common\cmd\ directory.

2.3 Adding source and assembly files

The source (*.c) and assembly (*.asm) files contain the actual code for running the application. In order for the flash memory to be accessed and programmed, the CSM (Code Security Module) password has to be provided as the CSM prohibits the flash memory to be accessed in order to hinder unwanted copying.

During the development its recommended that a dummy password; 0xFFFF is used as its favorable when erasing the flash memory prior to programming it.

The necessary code to control the CSM is found in DSP281x_CSMPasswords.asm found in the directory DSP281x_common\source\.

Certain sections have to be copied from flash to SARAM at device power-up. Those sections are the interrupt requests vectors and the flash control registers. The Peripheral Interrupt Expansion (PIE) module manages the interrupts. The flash control registers manages the flash wait states. These sections must be copied to SARAM at device start up respectively to RAM at runtime. The communication and copy is performed in the DSP281x_MemCopy.c source file from DSP281x_common\source\ using its MemCopy() function explained in the following.

2.4 Code modification and including of a header file

The next step is to modify the source code to initialize correctly. In the initialization code for the application, after the PIE vector table initialization, the MemCopy() function is called. Its followed by calling on the InitFlash() function which enables flash pipeline mode. The input values to MemCopy() are defined in DSP281x_Examples.h. This header file has to be included for the builder to locate the definitions.

```
#include "DSP281x_Examples.h" // written at the start of the code

void main(void) {

..

MemCopy(&RamfuncsLoadStart, &RamfuncsLoadEnd, &RamfuncsRunStart);

InitFlash();

..
}
```

3 Hardware configuration

At device power-up, the bootloader in ROM starts the code execution from an address declared by jumpers on the DSP board. Factory default is that the pins are set to “boot to H0” referring to an address in the memory map on the SARAM. In order for it to boot from flash, jumper number 7 (JP7) has to be set to active. At that point, all the other pins are don’t care and the device will “jump to flash” at device reset.

4 Transferring code and programming the flash memory

The software used to successfully transfer and program the flash memory is SDFlash, available for download from spectrumdigital.com, as its not included in the software accompanying the DSP. The F28xx On-chip flash programmer plug-in is not recommended since it does not perform as expected in various tests.

SDFlash provides an interface to the JTAG communication channel that will be used for flash programming. The essential part of SDFlash is its algorithm files, which contain well-defined standard functions that make calls to the Flash API Library to operate on the flash memory.

These algorithm files can be found in:

<SDFlash base>\myprojects\tif281x_v4_1\f2812\flash28\Debug\SDFlash2812.out

SDFlash files have the .sdp extension which contain the locations of the algorithm files, the program to be flashed and which communications driver to use.

Along with SDFlash there are a few example files included, as SampleF2812eZdsp.sdp. Normally the only necessary editing is changing the pathway to the program that is programmed to run from flash, Project->Settings->Programming->Flash Data File leaving the rest of the settings untouched. The next step is Device->Flash to flash the memory. During the erase operation, do not press STOP, as it will deplete the memory.

It is of great importance that CCS debugger is not connected to the DSP while trying to program the flash memory using SDFlash, as it will cause both programs to crash and all unsaved data will be lost.

The algorithm files are configured for a 30 MHz input clock and a 150 MHz CPU, coinciding with the specifications for the F2812 DSP series. It's imperative that these values are correct to avoid flash memory malfunction, as the flash algorithms must be run at the correct speed. If the DSP is running at a different CPU rate, this has to be corrected in the algorithm files. The clock frequency is specified in the SDflash2812.pjt, in SDflash28x_Wrapper.h.

5 Communication with the DSP after flash

After the DSP has been flashed using SDFlash, communications with CCS is done in a slightly different manner than if there was an application running from RAM:

Since the debugger in CSS does not need to load and run the entire application code again, but merely read the variables, the option 'load symbols only' is used after the debugger is connected to the DSP, File->Load symbols->Load symbols only.

The final step is to use the watchwindow or graph feature to display the +values in CCS..

6 Known problems while programming the flash memory

6.1 Memory depletion

Memory depletion occurs when the erase operation of the flash memory is interrupted, either by manually pressing STOP or by power loss, PC crash or device reset. Since the erase operation will be incomplete, the CSM passwords can be set at an unknown value locking the DSP permanently.

If the device can be unlocked, the depletion recovery algorithm can be run to recover the memory:

specdig\sdflash\myprojects\tif281x_v4_1\f2812\ SampleF2812_DepRecover.sdp

It's recommended that this algorithm is executed a couple of times until the memory is recovered.

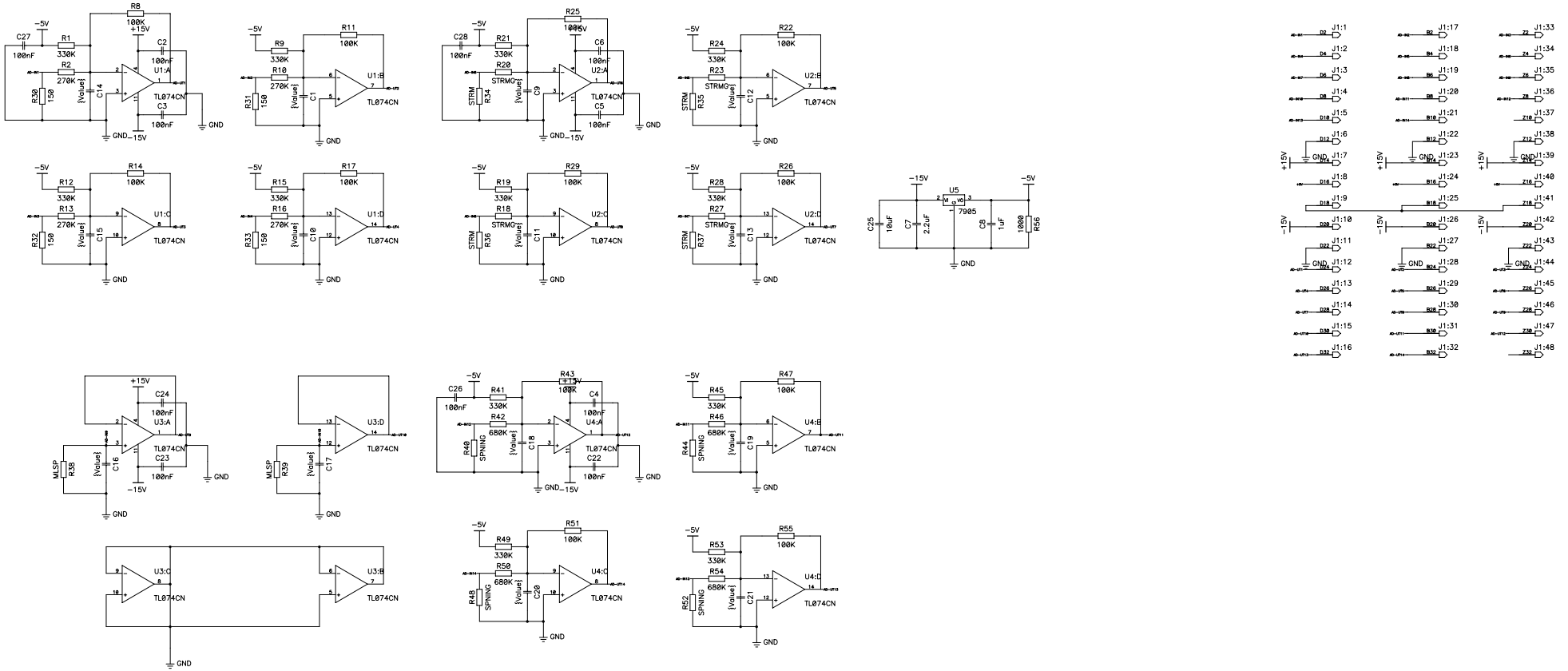
6.2 Memory retains the previous programmed application after flash

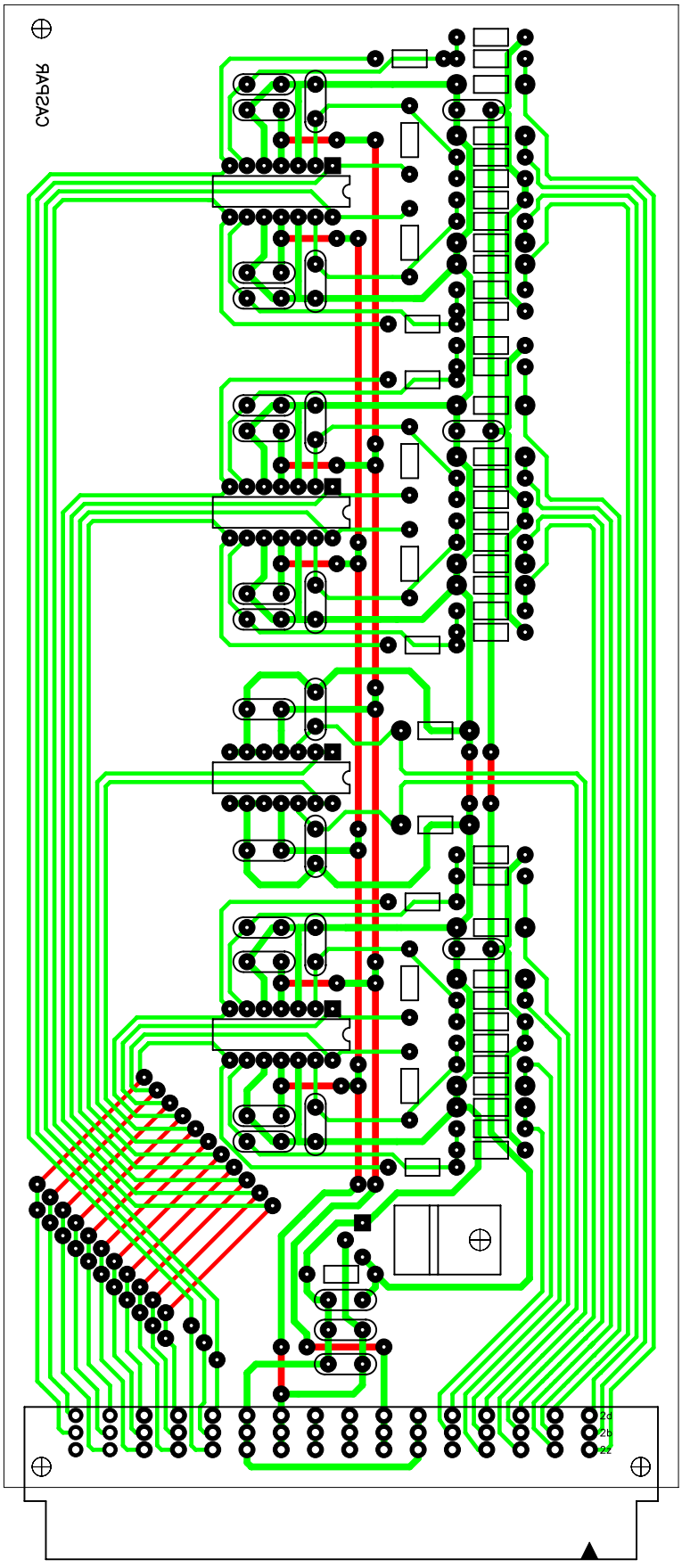
In some instances, the previously programmed application is still running from the flash memory after flash and device power down and up. If this occurs, programming the memory with only flash indexes will erase anything previously flashed.

specdig\sdflash\myprojects\tif281x_v4_1\f2812\image\Debug\imageABCDEFGHIJ.out

Why this occurs is not known at the moment of writing; however this will ensure that the previous programs are deleted.

CAD Schemes of the Adjustment Card


R8
100K
+15V
C27
100nF
R1
330K
C2
100nF
U1:A
R2
270K
R30
150
{Value}
C14
TL074CN
C3
100nF
GND_-15V
GND
-5V
R11
R9
330K
R10
270K
R31
C1
U1:B
R25
C28
R21
R20
STRM
R34
C9
C6
U2:A
C5
R22
R24
R23
STRMG
R35
C12
U2:B
R14
R12
R13
R32
C15
U1:C
R17
R15
R16
R33
C10
U1:D
R29
R19
R18
C11
U2:C
R26
R28
R27
R36
C13
U2:D
U5
LM
VC
7985
C25
10uF
C7
2.2uF
C8
1uF
1000
R56
+15V
C24
U3:A
C23
MLSP
R38
C16
R39
C17
U3:D
U3:C
U3:B
C26
R41
R42
680K
R40
SPINNING
C18
R43
C4
U4:A
C22
R45
R46
R44
C19
U4:B
R47
R49
R50
R48
C20
U4:C
R51
R53
R54
R52
C21
U4:D
R55
J1:1
J1:2
J1:3
J1:4
J1:5
J1:6
J1:7
J1:8
J1:9
J1:10
J1:11
J1:12
J1:13
J1:14
J1:15
J1:16
J1:17
J1:18
J1:19
J1:20
J1:21
J1:22
J1:23
J1:24
J1:25
J1:26
J1:27
J1:28
J1:29
J1:30
J1:31
J1:32
J1:33
J1:34
J1:35
J1:36
J1:37
J1:38
J1:39
J1:40
J1:41
J1:42
J1:43
J1:44
J1:45
J1:46
J1:47
J1:48


CASPAR