ISSN 0280–5316
ISRN LUTFD2/TFRT--5546--SE

An Experimental Platform for DC-cable Interaction with Ship Steering

Staffan Nilsson
Andreas Fransson

Department of Automatic Control
Lund Institute of Technology
December 1995

Department of Automatic Control
Lund Institute of Technology
P.O. Box 118
S-221 00 Lund Sweden

Document name
MASTER THESIS

Date of issue
December 1995

Document Number
ISRN LUTFD2/TFRT--5546--SE

Author(s)
Staffan Nilsson and Andreas Fransson

Supervisor
G. Olsson (IEA) A. Rantzer, M. Akke (Sydkraft)

Sponsoring organisation

Title and subtitle
An Experimental Platform for DC-cable Interaction with Ship Steering. (En experimentell plattform för DC-kablars inverkan på båtnavigering).

Abstract

HVDC (High Voltage Direct Current) cables are often used to transmit electrical power between two areas separated by water. When a ship, equipped with a ship steering automatic pilot containing a magnetic compass, passes such a cable it will deviate from its intended course and may even follow the cable.

The objective of this final master thesis was to build an experimental model. This model should be used to verify simulations of ships passing a HVDC cable. It includes a radio controlled boat, an automatic pilot, a magnetic compass and a position monitoring system. Some full-size experiments have been performed, but they were very expensive. This is why an experiment model would be most appreciated.

We wanted an automatic pilot which controlled the boat heading using a magnetic compass as the input. After having tried different solutions for the automatic pilot we chose a model which consisted of two micro controllers situated on the boat.

The position monitoring system uses pulses of audible sound. Four microphones detect the sound, each with a certain time delay proportional to the distance from the boat. From the differences in time delay the boat position can be calculated.

Experiments show that the model behavior was as expected , the boat deviated from its course and in the worst cases followed the cable.

Key words

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280–5316

ISBN

Language
English

Number of pages
103

Recipient's notes

Security classification

The report may be ordered from the Department of Automatic Control or borrowed through the University Library 2, Box 1010, S-221 03 Lund, Sweden, Fax +46 46 110019, Telex: 33248 lubbis lund.

Table of contents

Introduction

HVDC (High Voltage Direct Current) cables are often used to transmit electrical power between two areas separated by water. When a ship, equipped with a ship steering automatic pilot containing a magnetic compass, passes such a cable it will deviate from its intended course and may even follow the cable.

The objective of this master thesis was to build an experimental model. This model should be used to verify simulations of ships passing a HVDC cable. It includes a radio controlled boat, an automatic pilot, a magnetic compass and a position monitoring system. The experimental model is intended to complement full scale experiments, since they are very expensive. Earlier, full scale experiments have been conducted with a 17 meter ship over the Kontiskan II HVDC-link outside of Gothenburg and confirmed that much of the theory was valid in real life. See Akke[1].

In this document we will briefly describe the theory behind ship's course deviations. We will also present all the different steps involved in building a functional model. Finally, we will show some experiments made using our model and we will also evaluate its usefulness.

Compass Deviation due to DC Cables

As mentioned above HVDC cables are frequently used for transmitting electrical power between two areas or countries separated by water. The Baltic Cable, for instance, connects the Swedish power system to the European mainland. The great costs involved in laying new power sea cables force the power distributors to utilize the existing cables as hard as possible. This means running large currents, up to 1000 Amperes, through the cables. Such large currents will result in magnetic fields in the same magnitude as the earth's magnetic field and cause compasses to deviate from the correct bearing.

This problem is not that common today, as most sea traffic use the Global Positioning System (GPS). However, in some countries ships are required to have magnetic backup systems and there are still some ships that use automatic pilots with magnetic compasses. Sea cables are marked on sea maps, but further studies of the phenomena are nevertheless useful.

Compass deviation

When current is running through a cable a magnetic field, perpendicular to the radius, will be induced. The magnetic field B can be calculated given the structure in figure 1.

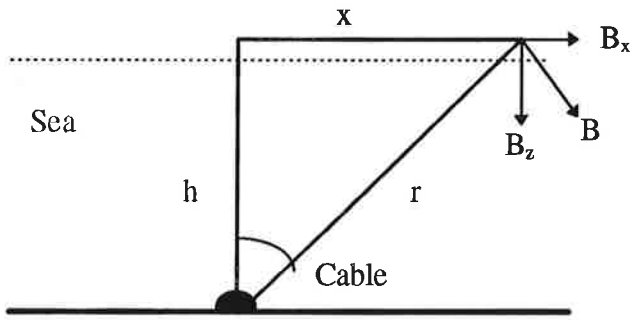


Figure 1. The magnetic field from a DC cable on the bottomof the sea at distance r from the cable

The magnetic field component in direction x is

$$B_x = \mu_0 H_x = 4\pi 10^{-7} \frac{I}{2\pi} \frac{h}{(h^2 + x^2)} = B_{max} \frac{h^2}{(h^2 + x^2)}$$

with

$$B_{max} = \frac{2 \cdot 10^{-7} I}{h} (Vs / m^2)$$

where

I is the cable current in Ampere

h is the cable depth plus the distance from the water surface to compass in meters

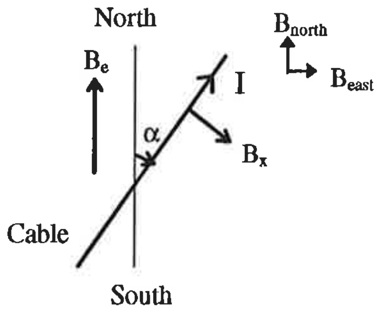


Figure 2. A cable with the current I. Be is the horizontal component of the earth's magnetic field. α is tha angle between north and cable current direction

From Figure 1 it can be found that the compass deviation δ caused by DC cable is

$$\delta\,(x,h,I,\alpha) = \arctan(\frac{B_{east}}{B_e + B_{north}}) = \arctan\left(\frac{\eta\,\cos(\alpha)}{1+(x/h)^2 - \eta\,\sin(\alpha)}\right)$$

with

$$\eta = \frac{B_{max}}{B_e}$$

Where

α is the angle in radians between north and cable current direction.
B_e is the horizontal component of the earth's magnetic field in the same unit as B_{max}

The magnetic field is largest at x = 0 which gives

$$\delta_{max} = \arctan\left(\frac{\eta\,\cos(\alpha)}{1-\eta\,\sin(\alpha)}\right)$$

For further details se Magnus Akke[1].

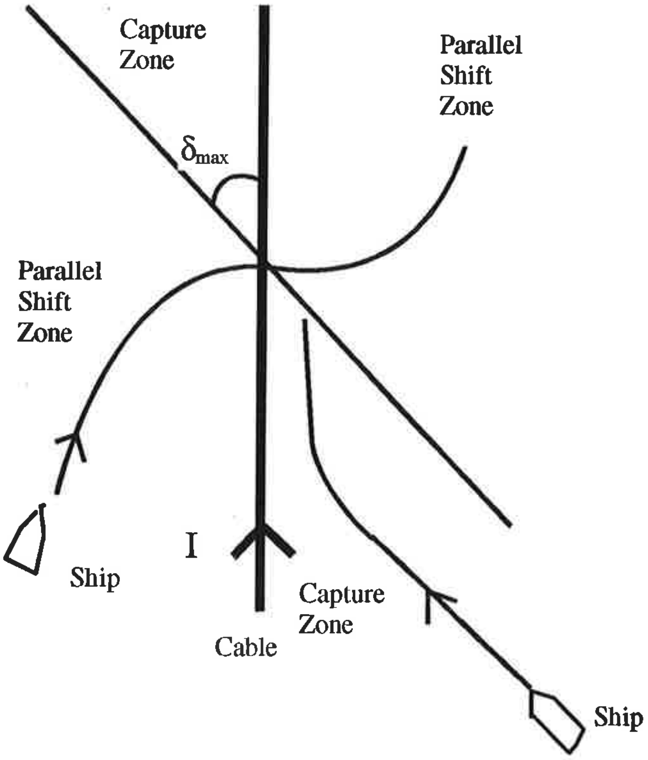


Figure 3. Depending on the input angle between the boat and the cable the boat either deviates for a while(the parallel zone) or it will follow the cable (the capture zone).

Overview of the Design

The objective for this theses was to build an experimental platform which could be used to further analyze the phenomena of ship course deviations near DC-cables. If time allowed the importance of speed, control algorithm and cable geometry should also be studied.

Analyzing the given problem we came to the conclusion that the project could be divided into separate parts. These parts were: Position monitoring, automatic pilot, radio transmission, measurement and experiments. The position monitoring system is used to follow the ship's movements so that it's trajectory can be stored and compared to the expected behavior. The automatic pilot will try to keep the ship on the desired course, while the radio transmission will establish communication between the boat and a computer on land.

At first we designed the boat to be controlled by a stationary computer. The compass output signal should be transferred via radio to the computer. A reference value would be set in a program and together with the compass signal the output would be calculated by the computer. The output could then be transferred back to the boat via another radio frequency.

After several experiments with two-way radio communication we came to the conclusion that it wasn't possible to have a receiver and transmitter so near each other on the boat as the transmitter interfered with the receiver. In order to overcome this problem we tried to send the signals in completely different frequencies and to shield the receiver. The latter was done to stop any signals from entering the receiver directly, thereby bypassing the input stage bandpass filter. Both attempts failed. We then had to find a way to come around the problem with dual transmission. One way was to move the control of the boat to a local microprocessor, thus eliminating the need for any radio transmission. This solution will be dealt with in next chapter.

Another major part was the position monitoring system. The problem here was, that there were no prefabricated systems for position detection in small areas. In the experiments with a full scale ship a GPS system was used. Unfortunately, the GPS system has a much to low precision to fulfill our demands. There are many different solutions which would give very high precision but they are often too expensive or too complicated. We finally chose to use audible sound.

In order to allow analysis of the experiments data from the position monitoring system and the compass must be stored. This was done using an acquisition unit called a Daqbook. The data was collected by a program written in Visual Basic and the analysis was done using Matlab.

The three different parts of the design will be explained more closely in the following three chapters; automatic pilot, the position monitoring system and data acquisition.

Automatic Pilot

When performing experiments the boat should be controlled by an automatic pilot. The automatic pilot tries to hold a steady course by controlling the rudder position with the compass signal as an input.

Controller and ship model

The ship system can be described by the following block diagram:

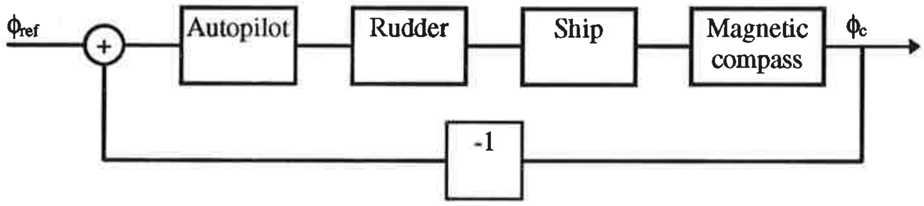


Figure 4. Structure of the control system for a ship with an autopilot.

The automatic pilot, a simple proportional controller, provides the rudder servo with a reference value. The rudder model is quite complex and will not be described here. Instead, we refer to Akke[1]. The ship can be described by what is usually called Nomoto's model:

$$H(s) = \frac{1}{s} \cdot \frac{a}{1 + s \cdot T_{ship}}$$

The input is the rudder position and the output is ship heading. The parameter a is proportional to the ship speed and the time constant T_{ship} is inversely proportional to the speed.

The magnetic compass can be described by a linear second order system.

Why did we decide on a proportional controller? The derivation part of a PID-controller is in reality seldom used. We do not know the reason for this, but as we wanted our experimental model to correspond with the real life situation as closely as possible we decided not to implement derivation. Looking at the system one can see that the ship model includes an integrator and consequently steady state deviation is eliminated. An integrating part in the controller is therefore not needed.

Our first attempt

The original idea for the automatic pilot was to let the computer used for data acquisition also handle the control of the ship. The output from a compass situated on the boat should be transmitted to the computer via radio. The control signal from the computer was to be sent back to the boat via another radio link.

As the computer should be able to handle both control of the boat and data acquisition at the same time we first thought we had to use a real time system. We had two different real time kernels available to us. The first kernel was written in Modula-2 by the department of Automatic Control and has been used by us in several courses. It was written for DOS, but included objects for creating a graphical user interface. Unfortunately the data acquisition unit we wanted to use did not have driver routines for Modula-2.

The second available kernel, which was written by Håkan Asklund of LTH Malmö, was a very simple kernel written in Turbo Pascal. As we had the necessary driver routines for Turbo Pascal it would be possible to use this kernel. However, it lacked support for a graphical interface and it had no priorities for the processes.

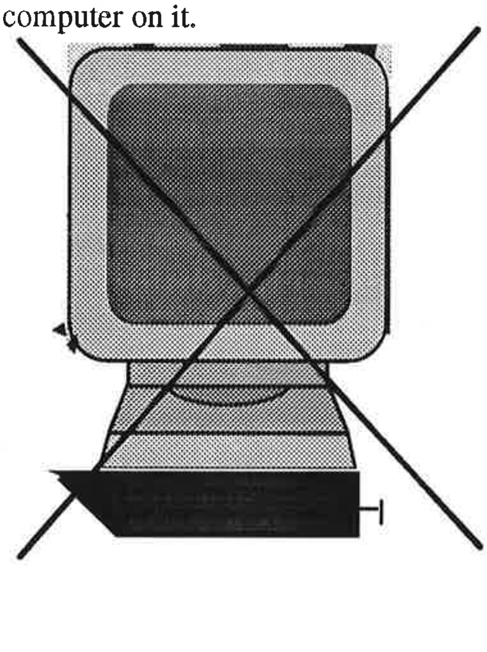
After having discussed the different alternatives we realized that a real time kernel was not essential in this application. This was because of the very low sample frequency needed. Also, the choice of a P-controller made it less important to have an exact sampling interval. We still wanted a graphical interface, preferably in the Windows environment. Therefore we decided to use Visual Basic.

When we had built the radio communication we made several tests. There was no problem to send information in one direction, but when we tried to send in both directions simultaneously problems arose. As a transmitter was located near a receiver on the boat they interfered with each other. We couldn't overcome this problem so we chose to try a completely different solution, the microprocessor solution.

The radio communication is described further in the Communication and data acquisition chapter.

The microprocessor solution

Because of problems with two-way communication, which are described above, it was impossible to implement an automatic pilot outside the boat. Because of the limited space on the boat we could not place a computer on it.



Instead we chose to work with the micro controller Motorola 68705. The 705 is simple to use and was familiar to us, as we had taught students at LTH Malmö (Lund Inst. of Tech. in Malmö) how to program it. We also could borrow programming equipment from the department in Malmö.

Analyzing the task for the automatic pilot we realized that it was not enough just using one processor. The control signal to the rudder servo had to be Pulse Width Modulated. Instead of building electronics dedicated for PWM we decided to use a processor for this purpose. This solution also allowed us to add extra features, such as maintaining the speed when switching from manual to the automatic pilot. This means that we used one processor to do the actual control and one processor to create a PWM-signal from the first processor's output.

The functionality that we wanted was:

- The boat should be able to be run both under automatic and manual control.
- Changing between manual and automatic should be done using the handset radio transmitter which is normally used for controlling a radio controlled boat. The automatic pilot should be switched to automatic when the radio transmitter is turned off, and switched to manual when the transmitter is turned back on.
- When switching to automatic mode, the first value read from the compass will be the reference value. The speed of the boat has been monitored by the automatic pilot and will be maintained.
- When in automatic mode a radio transmitter, which sends the compass value back to the computer, should be on. To avoid any interference with the handset transmitter it should be turned off when the handset transmitter is turned on, that is when the automatic pilot is switched to manual.
- The gain of the controller should be adjustable through DIP-switches.

Assembler language was used to program the processors. For the control algorithms, floating point operations were needed, but the 68705 does not support floating point numbers. Therefore we used some library routines written for the purpose. It was, however, very complicated to use these routines and this made the program large.

A flow-chart of the program can be found be found along with the program code in Appendix D.

Hardware design

The hardware of the automatic pilot was build on veroboard and can be described by the block diagram in Figure 5.

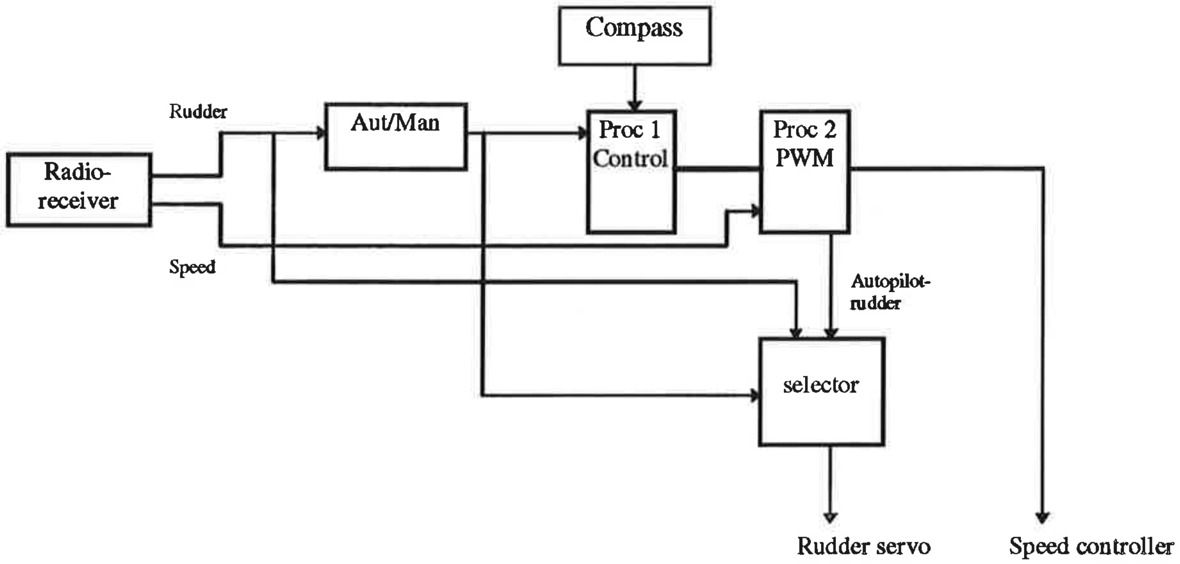


Figure 5. The implementation of the autopilot hardware. Processor 1 calculates the output with help of the compass signal and the manually given reference value. Along others the Processor 2 works as a Puls Width Modulation unit which takes the value from processor 1 and outputs it as a PWM-signal.

The radio receiver leaves two signals, one to control the rudder and one to control the speed. A circuit looks at the rudder signal and determines if the radio transmitter is active or not, that is if manual or automatic control is active. The first processor calculates the control signal for the rudder servo and the second processor converts it to pulse width modulation. The second processor also receives the speed signal from the receiver, so it can monitor the speed of the boat in manual mode. The selector makes sure that the correct rudder signal is routed to the servo, that is either the automatic or manual rudder signal.

The compass

In order to keep track of the heading of the boat we needed a compass. First we intended to build a compass using two magnetic field semiconductor sensors from Philips. We later heard that some people had tried to use these sensors without any success.

Instead of building an electromagnetic compass we decided to buy an electromechanical compass called Silva Fluxgate Transducer. Our version had both analog and digital outputs. It didn't produce the angle directly in degrees, but it gave sine and cosine for the angle as outputs. In the automatic pilot version with a stationary computer controlling the boat we wanted to use the analog outputs, but when we changed to the microprocessor solution we decided to read from the digital output directly into the controlling processor.

The problem with this change was that the digital output was poorly described in the short data sheet. Calling the Silva customer support did not help, as Silva themselves did not have any more information! Because of this we had to study the signal with a digital storage oscilloscope in order to find out how the data was sent.

The signal was sent with 300 bits per second with a high level for zeros and a low level for ones. It had one start bit (+5V) and two stop bits (0V). The information was sent as an eight byte data message. It consisted of the following parts:

- The start character of the message was the ASCII-code for a semicolon, B3 hex.
- The semicolon was followed by three bytes containing the sine for the angle, with the most significant byte sent first. These bytes were also sent as ASCII-codes. After conversion from ASCII to hex a 4 bit number was obtained. For example, the value 15 decimal was sent as the character 'F'. The sign bit was the most significant of the four bits included in the most significant byte.
- Following the sine data a checksum byte was sent. This was the XOR of all sine and cosine data bytes after conversion from ASCII to hex.
- Last three cosine data bytes were sent in the same form as the sine bytes.

An example: ";1DA0017" => sine=474 dec cosine=23 dec => angle = 2.78 degrees

In the last calculation the angle is cosine/sine. The reason is the angle system used at sea. It has 0 degrees to the north with the angle increasing clockwise.

Position Monitoring System

According to the specifications the ships movements should be determined so that the results of experiments could be analyzed. This calls for a position monitoring system.

Alternative solutions

Many different solutions have been suggested by different people. Some of them would have been completely impossible to carry through. We will however list all of them with comments on why or why not they were useful.

- **Cross-bearing with two laserbeams.** Here we need two rotating lasers on land and a reflector on the boat. By detecting the angles that give a reflection of the laser-beams the position can be calculated. This was a good suggestion, but it was too expensive for our budget.

- **Bar Code.** Two or more lasers mounted on the boat could read bar codes on the side of the pool. This would also be very expensive, and it would not be easy to calculate the position of the boat from the information retrieved by the lasers.

- **Light intensity measurement.** This method is based on the fact that light from a source on the boat will decrease with distance. Two photodiodes could measure the light from the boat and from this calculate the distance to it. Here the experiments would have to be carried out in relative darkness, something we could not accept.

- **Radio bearing**. With a radio transmitter mounted on the boat two bearings could be determined with two antennas on land. From this the position could easily be calculated. Building this demands much knowledge of radio equipment and would probably take too much time.

- **Radar**. This is a technique that is very difficult so use with the small distances that we work with. It is also complicated and expensive.

- **Mechanical systems.** This is probably the simplest suggestion of them all. By attaching wires to the boat its position can easily be determined. Unfortunately this would change the dynamics of the boat too much.

- **Video surveillance**. It is of course possible to film the experiment with a video camera. Nowadays it is also possible to let a computer follow a certain object in the picture. From this the position can be calculated with a simple transformation of the coordinates. The problem is that a normal VHS camera does not have a resolution that is high enough to obtain an acceptable accuracy.

- **GPS.** The global positioning system was used in the full size experiments. The accuracy is however not good enough for our model.

- **Ultrasound.** Here we are sending out short pulses from a transmitter on the boat. Several receivers detect the pulse, but do so with a certain time delay, which is proportional to the distance to the boat. By measuring the time difference between the receivers the position of the boat can be calculated. This is one of the solutions that we decided to try out.

- **Audible Sound.** This uses the same principles as ultrasound. The frequency is however much lower, around 3000 Hz. We decided to check this suggestion also in practice.

Ultrasound

We made some introductionary experiments with one ultrasound transmitter and one receiver, where we came to the conclusion that it was possible to detect the signal at 10 meters distance. This experiment was performed with borrowed transmitters and receiver amplifiers and we displayed the signal on an oscilloscope. The result led to the making of a new PCB for the receiver.

The PCB consisted of two amplifier stages with a very large gain, an envelope detector and a comparator for the creation of a digital signal. After waiting a long time for the PCB we were able to perform further experiments. Unfortunately after several test we came to the conclusion that the range of the signal was too limited. This was because the signal was hard to detect due to the very disturbing background noise. Another bad property was that the receiver had to be pointed directly towards the transmitter if the signal was not to be damped too much. Since the boat will move while the receiver is fixed it is impossible to achieve this situation in reality.

Due to all of these difficulties we decided to abandon the ultrasound method. If the frequency of the signal was decreased the difficulties of range and direction could be overcome. This is why we changed to the next method: Audible sound.

Audible Sound

Determination of the position with sound uses the same principle as ultrasound. Each of the four positioning units can be described by a block diagram, Figure 6.

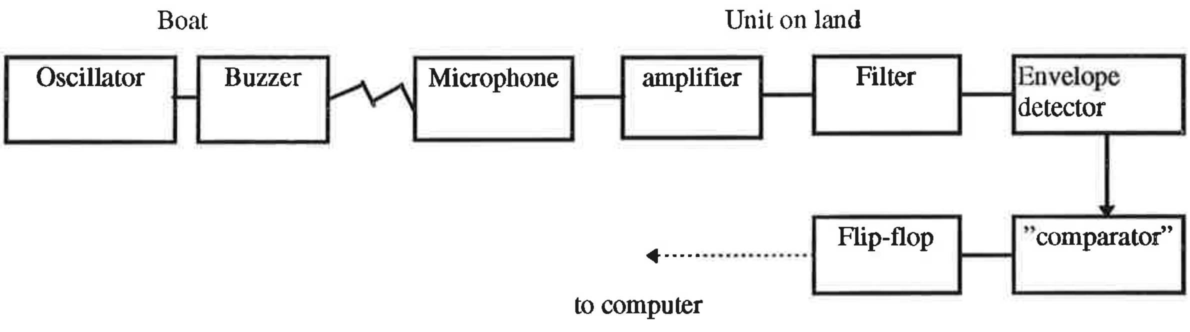


Figure 6. Structure of the position monitoring system. On board the boat there is a buzzer which sends out a tone every one second. On land there are four microphones which after amplification and filtering give signals to the data acquisition unit when the buzzer sounds.

The oscillator creates a pulsed signal with a time period of about one second. The buzzer converts the pulses into sound, and transmits it from the boat. The signal received from the microphone is very small and thus it must be amplified. A filter is used to minimize the influence of other disturbing sounds. The amplitude of the signal is compared with an adjustable voltage level which results in a digital signal. The flip-flop freezes the signal until it is reset by the computer.

As mentioned above we use an oscillator to produce a periodic signal. The frequency can be adjusted around 1 Hz.

We first analyzed the frequency spectrum of the sounds from the boat and decided on what frequency that should be used. Appropriate frequencies were around 3 to 3.5 kHz. To create the sound pulses we first used a piezoelectric buzzer, but it couldn't give the same frequency in all directions. There were also big differences in frequency between buzzers. We then tried a electromagnetic buzzer that was much better, but it transmitted a lower frequency than we wanted, 2.8 kHz. Despite this it worked without problem.

To receive the signal from the buzzer we needed a very sensitive microphone. We decided to use a small capacity microphone. The signal from the microphone was very weak, about 1 mV, so it had to be amplified. We used two inverting operational amplifiers, one before and one after the filter. Both had a gain around a hundred times.

Now we came to the tricky part, the filter design. We first tried a first order RC-net but it was not steep enough. After that we tested a switched capacitor filter. Because of inadequate technical data and difficulties to achieve a steady clock signal it took several days to get a functional band pass filter. Unfortunately, the clock signal of the filter caused noise which interfered with the weak signal from the microphone. Despite several attempts we failed to remove this interference.

We then built a series resonance filter using a inductor, capacitor and a small resistor. To obtain a high Q-value a very large inductor was used. This turned out to be a good filter, which we used until a second order state-variable filter was built. This filter was very easy to use and the only external components needed were two resistors. It had the same performance as the switched capacitor filter but did not introduce any noise. With the filter came a computer program to help calculating the component values. Unfortunately it had a bug which made it calculate even better than a Pentium.

The next step was to create a digital signal by comparing the peaks of the filtered signal with a preset reference value. To freeze the signal we used a flip-flop. The signal was held high until the flip-flop is reset by the data collection computer.

Problems

A problem was that when one of the four flip-flops was set all of them were set, which of course is a problem. This meant that the computer could not measure any time differences between the receivers. Also, when we tried to monitor the set-signals on an oscilloscope it trigged on the reset signal! We never could trace the reason for these problems, but they disappeared when we changed to a different type of flip-flop.

While most of the electronics were placed with the microphones in boxes which were to be placed around the pool the flip-flops were situated in a central box near the computer. Between the "comparator" and the flip-flips were long cables, up to 30 meters. The long cables between the "comparator" and flip-flop caused problems. The resistance in the cables introduced dips in the supply voltage when the electronics in the microphone box drew current. These dips went through to the output from the microphone where it was amplified. The solution was to let the microphone get its voltage via a separate voltage regulator.

Calculations for determining position

The following theory has been used to calculate the position:

The position monitoring system was to be put around a swimmingpool as shown in figure 5.

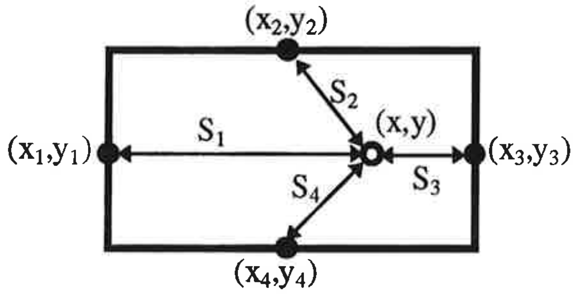


Figure 7. A swimmingpool with four microphones (filled dots) and a boat(ring) with the distances S_1 to S_4 from each other.

The position of the boat (x,y) can now be calculated. The distances S_1 to S_4 are not known by the position detection system, only the differences between them. Therefore we introduce the following equations

$$S_2 - S_1 = \Delta S_2$$
$$S_3 - S_1 = \Delta S_3$$
$$S_4 - S_1 = \Delta S_4$$

From Figure 7 and the equations above we get the following four equations

$$(x_1-x)^2 + (y_1-y)^2 = S_1^2 \qquad (1)$$
$$(x_2-x)^2 + (y_2-y)^2 = (S_1 + \Delta S_2)^2 \qquad (2)$$
$$(x_3-x)^2 + (y_3-y)^2 = (S_1 + \Delta S_3)^2 \qquad (3)$$
$$(x_4-x)^2 + (y_4-y)^2 = (S_1 + \Delta S_4)^2 \qquad (4)$$

If one assumes that the microphones are placed in the corners of a unit square the equation can be simplified to the following

$$
\begin{aligned}
&x^2 + y^2 = S_1^2 \\
&x^2 + (y-1)^2 = (S_1 + \Delta S_2)^2 \\
&(x-1)^2 + (y-1)^2 = (S_1 + \Delta S_3)^2 \\
&(x-1)^2 + y^2 = (S_1 + \Delta S_4)^2
\end{aligned} \qquad (5)
$$

After some simplifications we will get:

$$y = \frac{1}{2}(1 - 2\Delta S_2 S_1 - \Delta S_2{}^2) \qquad (6)$$

$$x = \frac{1}{2}(1 - 2\Delta S_4 S_1 - \Delta S_4{}^2) \qquad (7)$$

$$0 = 2S_1(\Delta S_4 + \Delta S_2 + \Delta S_3) + \Delta S_2{}^2 + \Delta S_4{}^2 - \Delta S_3{}^2 \qquad (8)$$

From equation (8) S_1 can be solved and inserted into equation (7) and (8). Similar algorithms can be used with the microphones put in the corner of a rectangle.

We did not use the equations (7) and (8). Instead we calculated the position numerically by minimizing the following equation. The equation is derived from equations (2) to (4) and symbolizes the total error for these equations.

$$f(x,y) = \left((x_2 - x)^2 + (y_2 - y)^2 - (S_1 + \Delta S_2)\right)^2 + \left((x_3 - x)^2 + (y_3 - y)^2 - (S_1 + \Delta S_3)\right)^2 + \left((x_4 - x)^2 + (y_4 - y)^2 - (S_1 + \Delta S_4)\right)^2$$

Communication and Data Acquisition

To be able to make experiments one must gather all important data created by the automatic pilot and the positioning system. This chapter will continue where the descriptions of these systems end in the last two chapters. Here we describe the data acquisition and the communication with the two systems.

Radio communication

In the first analysis of the experimental platform we decided to use a stationary computer to control the boat. For the communication between the computer and the boat radio transmission seemed a good choice. As miniature boats are normally equipped with complete radio systems we thought it would be a good idea to use two sets of systems for our two-way radio communication. The system consists of a handset unit and a small receiver for controlling a servo or an engine.

As we wanted the transmitter on the boat to take as little space as possible we chose to use a radio system for a small model car to transfer the compass bearing from the boat to the computer . To be able to use this equipment we had to build some adaptive electronics. It was quite difficult to build this for the car radio system as we had no circuit layout. When we were finished with the building of the electronics for the car model receiver we realized that the resolution for the receiver was too poor. This was because a speed control was integrated on the receiver circuit board. Consequently we bought an other receiver like the one belonging to the boat.

The outputs from the compass bearing receiver were pulse-width modulated and had to be converted to an analog signal using a low pass filter. This introduced a time delay which decreased the stability margin of the controller.

When transmitting the compass output an offset of +5 Volts had to be added. This offset had to be taken into account by the program calculating the compass bearing.

When we succeeded in transmitting values from the compass over the radio we began experimenting with two way communication. This turned out to be a bigger problem than we expected. The signal from transmitter on the boat affected the receiver with which the boat should be controlled. The reason for this was the limited area of the boat, which led to that the transmitter and the receiver were located too close to each other.

The reserved frequency channels for amplitude modulated radio controlled boats lie very closely together on the 27 MHz band. We tried using the channels at either end of the frequency band, but this did not help. Therefore a frequency modulated radio system for airplanes, using the 36 MHz band, was borrowed. We also shielded the receiver in order to stop signals from entering the receiver directly, thereby bypassing the input stage band pass filter. But when even this did not work we changed to the microprocessor solution as mentioned in the automatic pilot chapter. In this way we also got rid of the time delay of the compass signal in the control loop.

The data acquisition hardware

When this theses was started it was clear that we needed a data acquisition unit. It would be used for gathering information and for controlling. Our industrial partner Sydkraft had an available acquisition unit that we could use. The unit, which is called a Daqbook, is an external unit that can be used without the need of any further hardware interface. The Daqbook can be connected directly to the parallel port of a notebook or a desktop PC. It features both digital and analog in- and outputs as well as programmable counters/timers. With the Daqbook came several drivers for different program languages including, Visual Basic and Pascal.

As we have described in previous chapters we first tried an automatic pilot controlled by a stationary computer which was communicating with the boat using two-way radio transmissions. In this case we wanted that the Daqbook should both read the compass signal and output the control signal to the boat. To read the compass signal we needed two analog inputs, one for the sine signal and one for the cosine signal. For the control of the boat we used two analog outputs for the rudder and speed. In final case with the microprocessor solution we only needed the Daqbook to read the compass bearing for subsequent analysis.

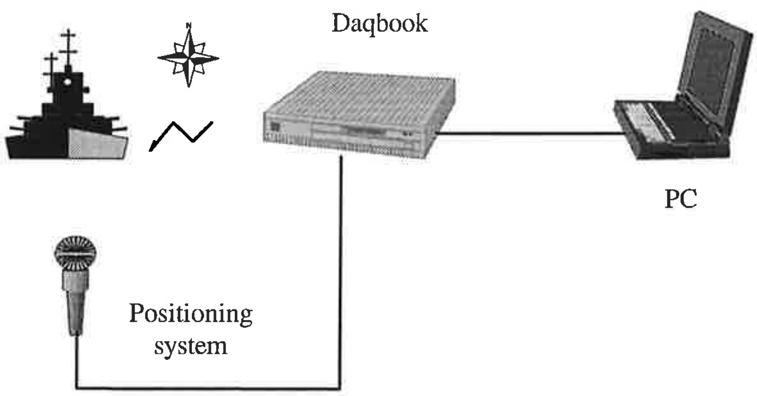


Figure 8. Overview of the system with the acquisition unit in the center. The boat transmits the compass signal via radio to the Daqbook. The positioning monitoring system consist of four microphones which are directly attached to the Daqbook.

We also needed the Daqbook for the position monitoring system so we could store information about the boat movements. As described in the positioning monitoring system chapter the signal from a microphone will be go high when the system has detected a sound and will stay high until it is reset. As we wanted to determine the time difference between the four microphone signals we used the Daqbook's counter inputs. The procedure of determining the time difference is as followed:

- First the Daqbook resets all four flip-flops for the different microphone signals and starts four counters, one for each channel.
- When a microphone detects a sound from the boat the corresponding counter will be stopped.
- After a set time the counter values are read by computer and the position can be calculated. A new cycle is then begun.

The Software

We used two different software tools for collecting and analyzing data. For collecting data we used Visual Basic and for analysis Matlab was the natural choice.

As we mentioned in the last chapter the data acquisition hardware, the Daqbook, supports several drivers for the most common languages. When we first designed the autopilot we thought that we needed a real time system. We then had two different choices of real time kernels, one written in Modula-2 and one written in Pascal. The Daqbook did not support Modula-2 and the Pascal kernel did not have a graphical user interface. When we had analyzed the situation further we realized that a real time kernel was not necessary, because of the low sample time required. We also would like the program to run in the windows environment. A program language in which it is easy to build a Windows application is Visual Basic, and as the Daqbook supported this language we decided to use it.

Later we found that the automatic pilot ought to be situated locally on the boat. The objective for the program then became to gather and store information from the compass and the positioning system.

The program first initiates the communication with the Daqbook. It then periodically reads position and compass data from it and displays the compass bearing and the position system time differences on the screen. The data is also stored for further analyzing.

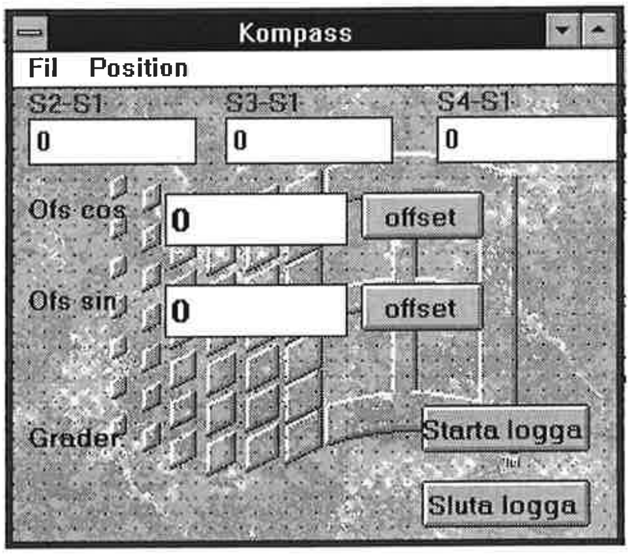


Figure 9. The Visual Basic program for data acquisition.

The program user interface consists of easy-to-use pop-up menus and push buttons. For further information about using the program see the user manual in Appendix A.

As mentioned above we used Matlab for analysis. We wrote a function which reads the data from a stored file. Thereafter the time data is filtered and the movements of the boat are calculated. The result is displayed to the user in two diagrams, one for the movements and another for the compass bearings.

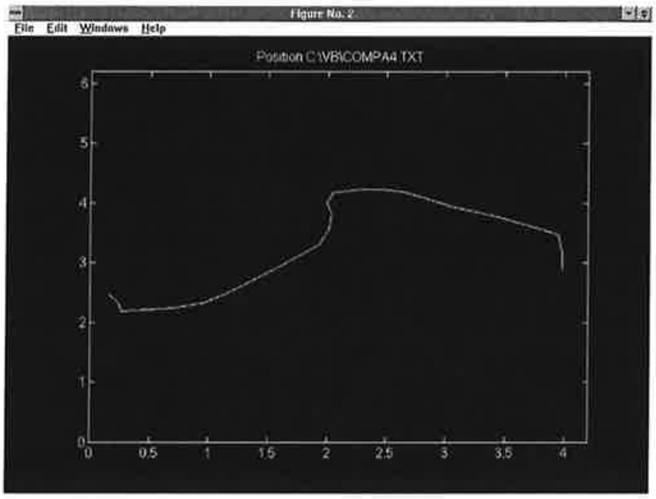


Figure 10. A Matlab plot of the calculated movement.

The Matlab function can be called from the Visual Basic program via a DDE(Dynamic Data Exchange) connection. Matlab is then used as a server or, in the Matlab terminology, as a computational engine.

Simulations and Experiments

Before we made any experiment we had to make some simulations to find out under which conditions the compass deviation phenomena can be seen.

Simulations

Simulations were previously made in Simnon by Magnus Akke [1] for the full-size experiments at Kontiskan-II. These simulations were made using the ship model from the autopilot chapter including a continuos proportional controller. In our experimental platform we used a digital controller with a sample time of one second. Our first simulations show the difference between the two cases using the dynamics of the ship used at the full-size experiments.

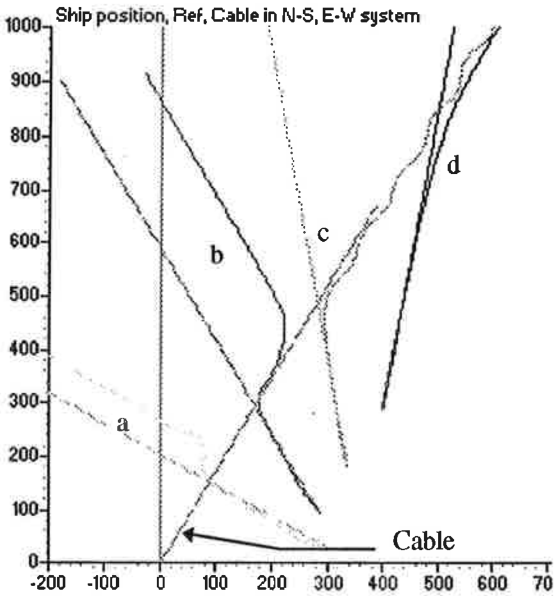


Figure 11. Simulation of a ship with a continuous proportional controller. The figure shows how the boat deviates from its intended course for different angles between the heading and the cable. The straight lines marks the intended heading. One can see that in the cases a and b the boat deviates for a while but comes soon back to the right heading. In case c one can see an interesting phenomena, the boat oscillates along the cable. In case d the boat follows the cable

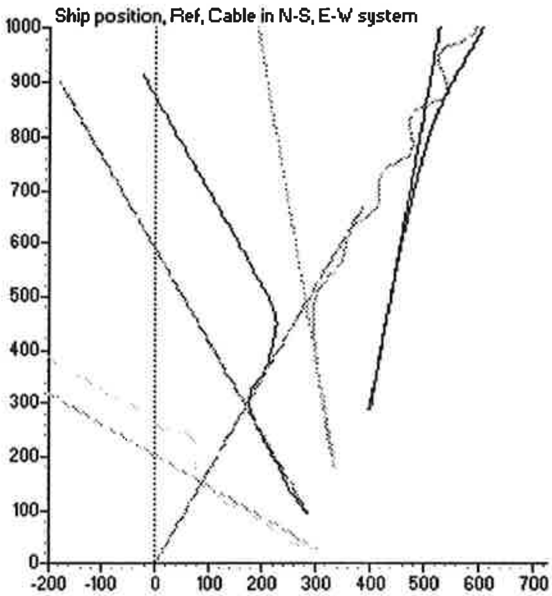


Figure 12. Simulation of a ship with a discrete proportional controller. This figure differs from figure 7 in case c where the amplitude of the oscillations grows along the way.

As can be seen in the pictures the amplitude of the oscillating trajectory is stable with the continuous controller, but in the discrete case the oscillations increase. This is a sign that the stability marginal decrease with the use of a discrete controller.

Next we tried to make a simulation with the ship dynamics, current and cable depth which apply to our experimental platform. We determined the gain of the ship model by having the boat running in circles. By measuring the time it took to turn 360 degrees we could calculate the gain with that particular speed, in our case around 0.5 m/s. The time constant was determined from a zigzag test.

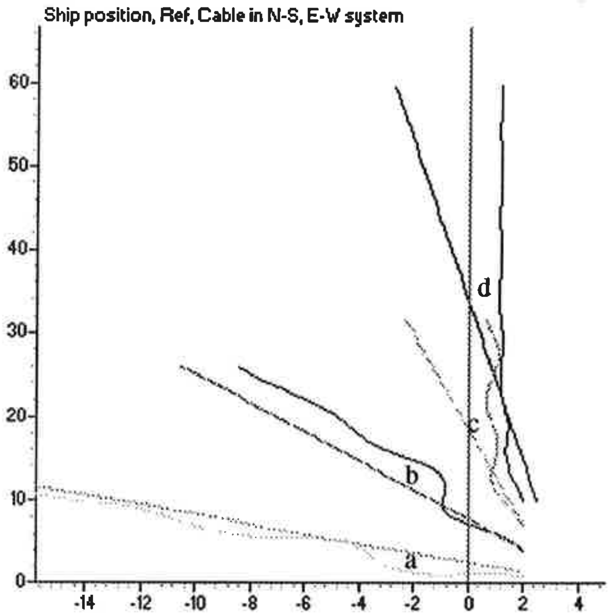


Figure 13. Simulation of our model boat with a current of 28 A and a depth of 0.5 meters. Observe the scaling of the x and y axis. The straight lines marks the intended heading. One can see that in the cases a and b the boat deviates for a while but comes soon back to the right heading. In case c one can see an interesting phenomena, the boat oscillates along the cable. In case d the boat

The simulation shows that the angle between the boat and the cable must be less than 10 degrees if the boat is to be captured.

The source code for the simulations can be found in Appendix D.

Experiments

The experiments can be divided into two separate parts. Experiments with the boat were conducted in an outdoor swimming pool. Unfortunately we were unable to test the positioning system with the boat as the daqbook was used by staff at Sydkraft at the time. Therefore the monitoring system could only be tried separately.

As mentioned before we made experiments with the boat at the outdoors swimming facility in Södra Sandby some 10 km outside Lund. Because of the season (early winter) we had limited access to the pool - it was sometimes frozen! It was a rather small pool with the dimensions 16*8 meters. The cable was put in the pool in north- south direction which coincided with the pool.

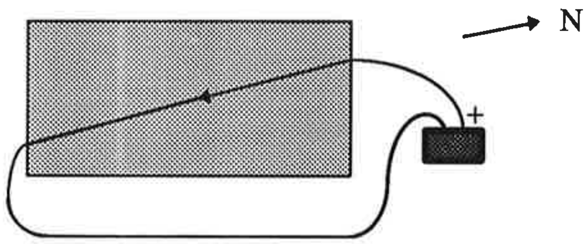


Figure 14. The swimmingpool with the cable and a battery. Observe that the cable lies in a north-south direction.

As seen in the picture above the cable is supplied with current from a battery. To reach the desired current of 28 Amperes we had to use two batteries in series and the only needed load was the cable itself.

When the cable was installed it was time for experiments with the boat. With manual control we set the desired boat heading and speed, and when we turned on the autopilot. We tried different angles resulting in that the boat deviated from it's course in the same way as in the simulations. However, we could not verify the oscillation behavior when the boat followed the cable, as the pool wasn't long enough.

We intended to make the experiments together with positioning system for a complete test of the experimental platform, but we did not have the data acquisition unit available at the time. Instead we documented the experiments by recording the boat's behaviour on film.

As we could not test the positioning system with the boat in action we made separate tests indoors. The premises for the test was not what we had desired. The real experiments were supposed to be conducted in a large open space. Such a place would be free from reflections and obstructing objects, things with which the now actual room had plenty of.

Much due to the premises the experiments did not go as we had hoped for. When detecting the movements of the buzzer we detected great variations in the calculated position. This was because of big jumps in time for sequential values of the individual microphones. With the help of a digital filter the influence of these variations could be minimized.

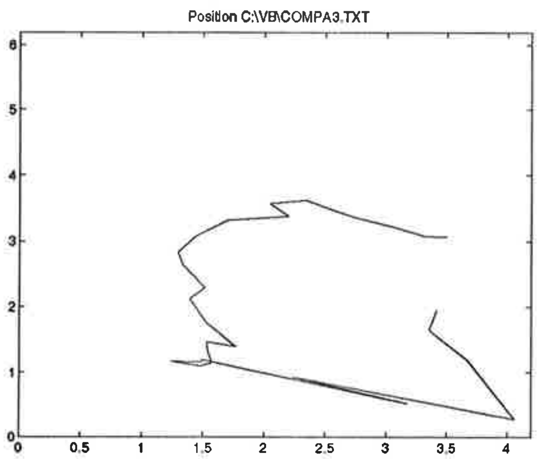


Figure 15. Detected movements without filter

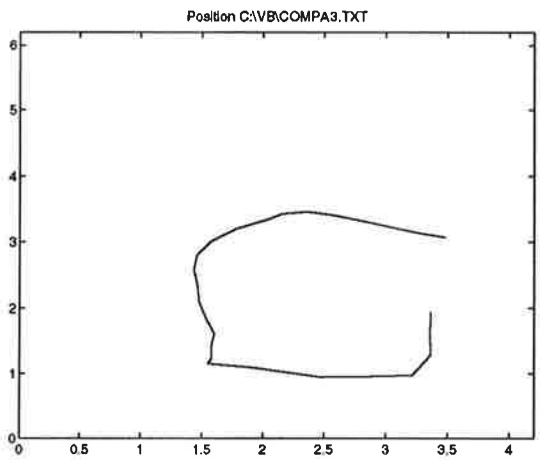


Figure 16. Detected movements with the use of a filter

Conclusions

Experiments using our experimental platform show that the boat behaved as expected as it crossed a DC-cable. The boat deviated from it's course when coming near the cable, and for small angles between the intended boat heading and the cable, the boat follows the cable.

Unfortunately time did not allow us to go deeper into how the choices of speed, control algorithm and cable geometry effect the ship behaviour. These factors are suitable for further studies.

During the design of the boat we found that some details could have been done better. One difficulty was that the compass had a sample time of one second. This is a normal sample time for full-size ships with it's dynamics and speed, but for out miniature boat the sample time ought to have been smaller. There are two reasons for this. The dynamics of our boat is faster than that of a full-size ship, which means that the stability marginal of the automatic pilot loop decreases. The second reason is that the shape of the magnetic field changes due to scaling effects.

The magnetic field affects the compass within a certain distance from the cable. This distance is proportional to the cable depth. If a full-size ship is perpendicular to a sea cable, the compass will give more sample values within this distance than if our model boat moves at right angle to it's cable. This is because the ratio between the cable depth and ship speed is higher for the full-size ship. Due to this the influence of the cable on our miniature boat is smaller than in the case of a real ship.

It is difficult to use two-way radio communication when the transmitter of one channel is very close to the receiver of the other channel. This was the case when we tried to implement the automatic pilot on a stationary computer with a radio link to the boat. We then decided to design the automatic pilot with two microcontrollers on the boat. This way we also avoided the time delay and the error introduced by the radio link.

We can not fully evaluate the position monitoring system because it has never been tested in its intended environment. The system has only been tested separately indoors. When tested with a buzzer on a fixed position it worked as intended. However we had discrete fluctuations of about 10 cm for the difference in distance between two microphones. We can not explain these fluctuations. When the position monitoring system was tested with a moving buzzer the detected movement pattern was very irregular. This was because of the influence of echoes and reflections in the room. By using a simple low pass filter we managed to reduce the irregularities. When the system is used outdoors these problems will hopefully not occur.

An important experience of this master thesis is that practical work is more difficult and takes more time than expected. Considering the budget and the available premises we feel that we have accomplished a good solution to the given problem.

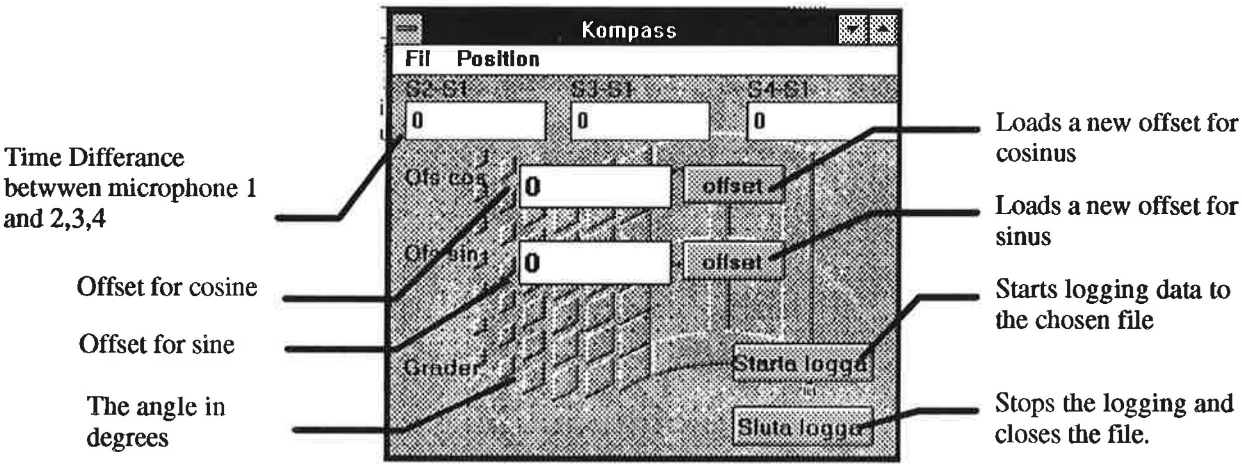
References

1. Magnus Akke and Karl Heinz Lampe,"Non-linear interaction of DC cables and magnetically controlled ship steering autopilots",1995

Appendix A - Instruction manual for the Compass program

The Visual Basic program Compass is used for data acquisition of position and direction for a miniature boat. The program is run under Windows and requires that a special hardware for data acquisition, a Daqbook, is connected to the computer. The major task for the program is to log the time difference between four microphones and the compass bearing and store these values to a file. Calculation of the position is not done directly in this program but in Matlab via a DDE-link (Dynamic Data Exchange).

Buttons and textboxes



Menu File

- **Se loggfil**: Shows the logged data file with the help of the standard window program Notepad.
- **Välj loggfil**: Allows you to choose a filename for the log file in a standard file dialog window.
- **Avsluta**: Exits the program.

Menu Position

- **Initiera DDE-Länk**: This menu choice initiates a DDE-link with Matlab, which acts as a server for this program.

- **Kör positionsbest**: This choice calls a Matlab function which calculates and plots the trajectory of the boat for the chosen logfile. If the DDE-link is not initiated a link will be established before the function is run. Further, if no microphone coordinates is chosen two dialog window will appear.

Here the microphone's x respectively y coordinates should be entered. But if no filter is chosen the default value 0.5 will be used. When Matlab is finished calculating the boat's movements can a graph like the one below appear on the screen.
(BILD)

- **Mikrofonkoordinater**: Allows you to choose new coordinates for the four different microphones. It will first ask for all the x-coordinates and then for the y-coordinates.

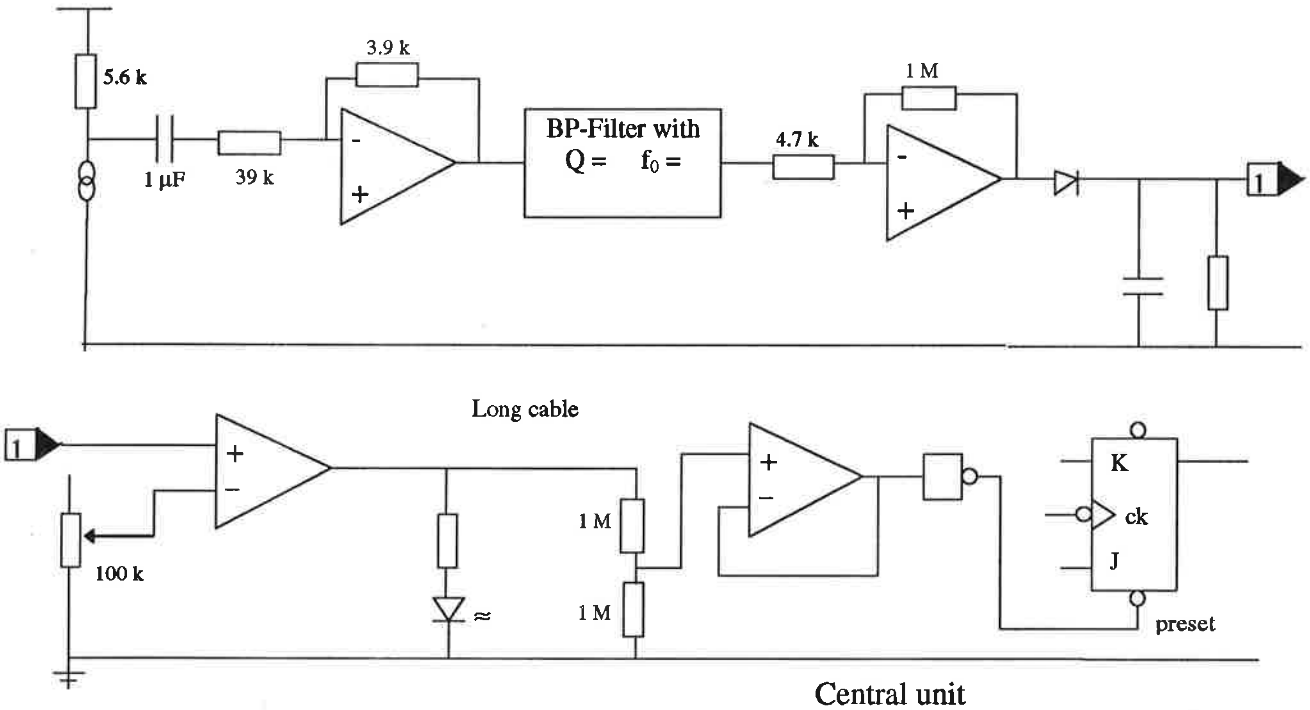
- **Filtervärde**: By changing the value of the filter parameter you can adjust the importance of the filter. For example if you chose zero this will mean that no filtering will be used. Normal choices are between 0 and 0.5.

A step for step example:
1. Check the offset value and eventually load new values by pressing the offset button.
2. Choose a name for the logfile in Menu File -Välj loggfil.
3. When ready press the button "Starta logga" to start storing values to the file.
4. Stop logging by pressing "sluta logga".
5. Enter the the coordinates for the microphones that have been used with the experiment in menu posistion.
6. If needed change the filter parameter.
7. Start the calculations in Matlab by chosing "Kör positionsbest" in menu Position.

Appendix B - Layouts

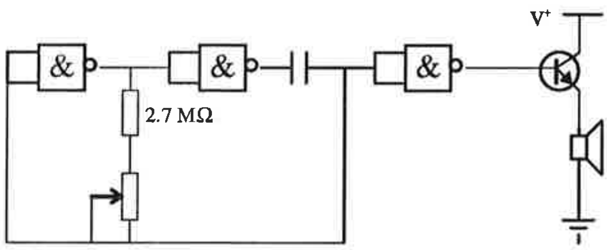
Position monitoring system

Latout over one of the four units in the position monitoring system. The microphone units is connected via a long cable to the central unit.

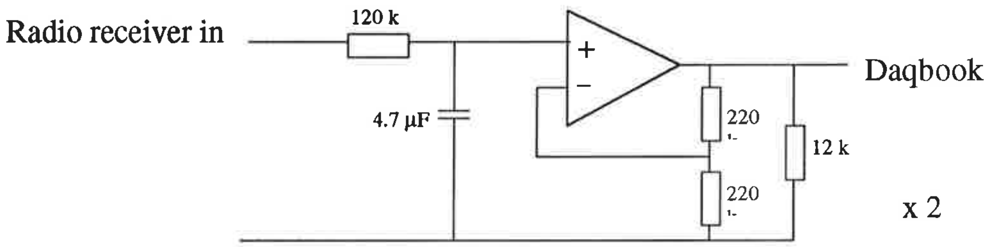


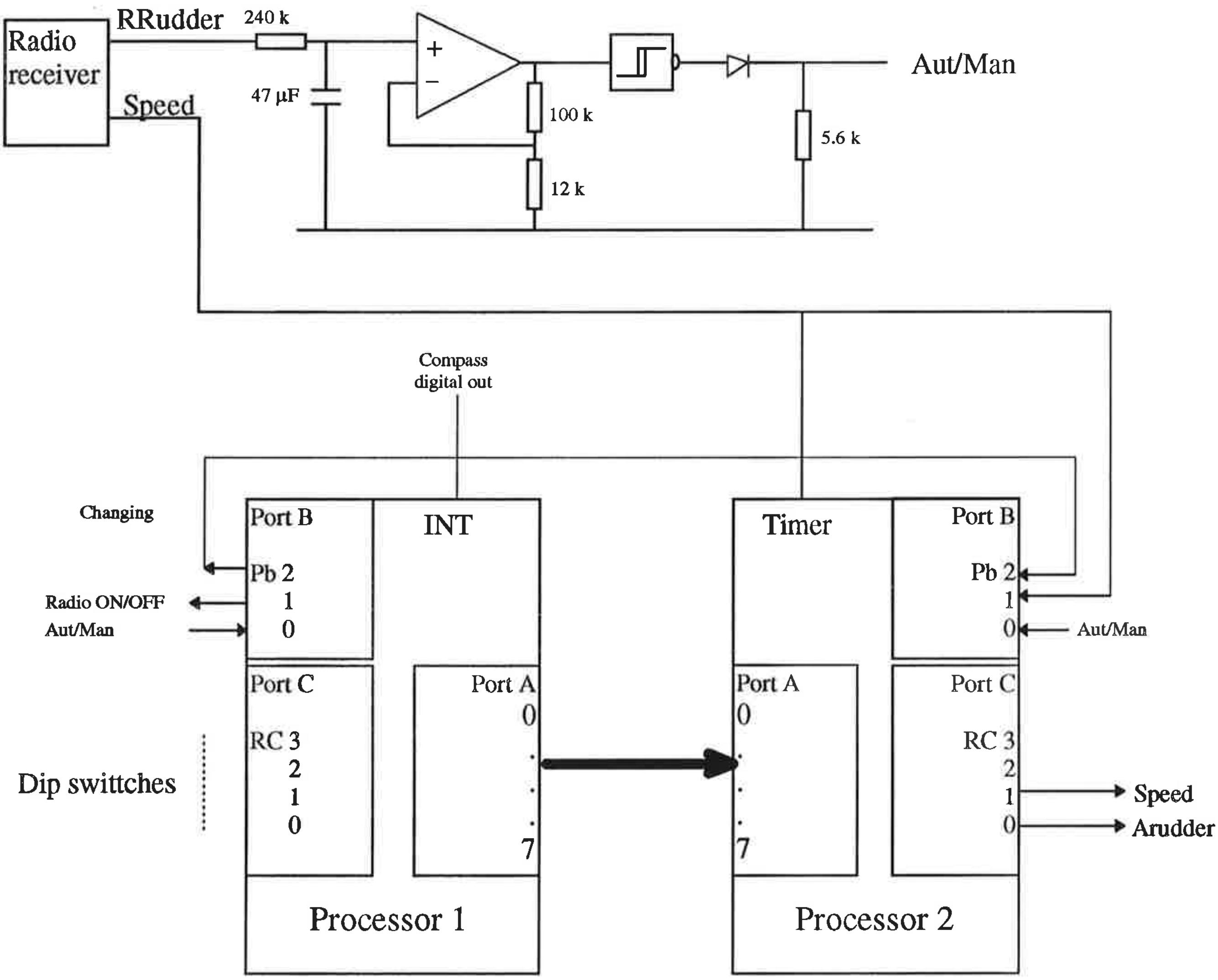
Central unit

The oscillating circuit for the buzzer.

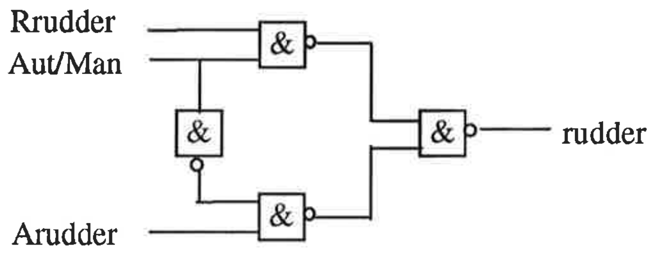


Layouts for the automatic pilot system

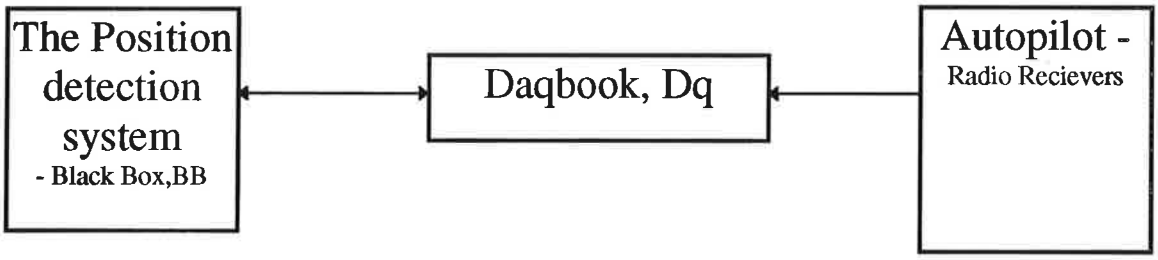




Layouts for the automatic pilot system. Electronics for choosing manual or automatic rudder values.



Appendix C - Connection of the equipment to the Daqbook



The Daqbook(**Dq**) has three ports with 37 inputs/outputs each. The first port contains analog and digital inputs and the second port the outputs. Finally, the third port contains among others the counter enable inputs which are used by the position detection system.

The position detection system consist of five black boxes, four microphone units and one central unit. It is important to keep track of which microphone that are connected to which channel on the central unit(here refereed as the black box, **BB**).

The only signal needed for connecting the data acquisition unit with the autopilot system(**AS**) are the two analog compass signals, cosines and sinus, from the radio receiver.

The following syntax will be used:

Dq port:nbr	Dq=Daqbook, port: one of the three ports (1-3), nbr: which input/output (1-37)
BB color	BB=Black Box(the position detection system), color: the plug with the corresponding color.

Dq 3:1 ⇔ **Dq** 3:30	(interupt ⇔ osc. out)
Dq 3:2 ⇔ **Dq** 3:11	(interupt enable ⇔ ground)
Dq 3:37 ⇔ **BB** white	(Timer 1)
Dq 3:18 ⇔ **BB** black	(Timer 2)
Dq 3:16 ⇔ **BB** blue	(Timer 3)
Dq 3:14 ⇔ **BB** green	(Timer 4)
Dq 2:29 ⇔ **BB** yellow	(Flip-Flop reset)
Dq 3:11 ⇔ **BB** ground	(ground)
Dq 1:37 ⇔ **AS** cosine	(Cosine)
Dq 1:36 ⇔ **AS** sine	(Sine)

Appendix D - Software

Visual Basic code

compass5

```
Sub Command1_Click ()
 Close #1
 command4.Enabled = True
 command1.Enabled = False
 logga = False
End Sub

Sub Command2_Click ()
 Dim tot#, x%
 ReDim buf%(200)
 ' Disable channel tags
 ret% = VBdaqAdcSetTag%(0)
 ret% = VBdaqAdcSetFreq(2000!)
 ret% = VBdaqAdcRdn%(0, buf%(), 200, DtsPacerClock%, 0, 0, 4000!, Dgainx1%)

 ' read cos2
 tot# = 0
 For x = 0 To 199
  tot# = buf%(x) / 200 + tot
 Next x
 ofs1.Text = CInt(tot#)
End Sub


Sub Command3_Click ()
 Dim tot#, x%
 ReDim buf%(200)
 ' Disable channel tags
 ret% = VBdaqAdcSetTag%(0)
 ret% = VBdaqAdcSetFreq(2000!)
 ret% = VBdaqAdcRdn%(1, buf%(), 200, DtsPacerClock%, 0, 0, 4000!, Dgainx1%)

 ' read cos2
 tot# = 0
 For x = 0 To 199
  tot# = buf%(x) / 200 + tot
 Next x
 ofs2.Text = CInt(tot#)
End Sub

Sub Command4_Click ()
 Dim fil$
 On Error GoTo errorhandler2
 If Not logga And cmdialog1.Filename <> "" Then
  Open cmdialog1.Filename For Output As 1 Len = 100
  logga% = True
  command4.Enabled = False
  command1.Enabled = True
 ElseIf cmdialog1.Filename = "" Then
  cmdialog1.Filename = "dummy"
```

```
  End If
  Exit Sub

errorhandler2:
   MsgBox "Kan ej öppna fil " + cmdialog1.Filename, MB_OK
   Exit Sub
End Sub

Sub exit_Click ()
' Avsluta
  Unload form2
End Sub

Sub filtering_Click ()
' meny ange filter värde
   oldfilter$ = Filter$
   Filter$ = InputBox("Mata in filterparametern a i formeln y(k+1)=ay(k)+(1-a)x(k)", "Filterparameter", "0.5")
   If Filter$ = "" Then
     Filter$ = oldfilter$
   End If
End Sub

Sub Form_Load ()
  ' initiering
  ddeon = False
  xkoord$ = ""
  ykoord$ = ""
  Filter$ = "0.5"

  ' Set error handler and initialize DaqBook/100
  ret% = VBdaqSetErrHandler%(100)
  On Error GoTo ErrorHandlerADC2
  ret% = VBdaqInit%(LPT1%, 7)
  ' initiera daqbook
  ret% = VBdaqDigGetConf%(0, 1, 0, 0, config%)
  ret% = VBdaqDigConf(Ddclocal%, config%)
  ' sätter lsb i port high
  ret% = VbdaqDigWtBit%(DdpLocalClow%, 0, 1)
  logga = False
  ret% = VBdaqCtrSetMasterMode%(0, DcsF3%, 0, 0, DtodDisabled)
  ret% = VBdaqCtrSetCtrMode%(1, dgcHighLevelGateN, 1, DcsF2, 0, 0, 1, 0, 1, DocInactiveLow)
  ret% = VBdaqCtrSetCtrMode%(2, dgcHighLevelGateN, 1, DcsF2, 0, 0, 1, 0, 1, DocInactiveLow)
  ret% = VBdaqCtrSetCtrMode%(3, dgcHighLevelGateN, 1, DcsF2, 0, 0, 1, 0, 1, DocInactiveLow)
  ret% = VBdaqCtrSetCtrMode%(4, dgcHighLevelGateN, 1, DcsF2, 0, 0, 1, 0, 1, DocInactiveLow)
  ret% = VBDaqctrsetLoad%(1, 0)
  ret% = VBDaqctrsetLoad%(2, 0)
  ret% = VBDaqctrsetLoad%(3, 0)
  ret% = VBDaqctrsetLoad%(4, 0)

Exit Sub


ErrorHandlerADC2:

ErrorString$ = "ERROR in ADC1"
ErrorString$ = ErrorString$ & Chr(10) & "BASIC Error :" + Str$(Err)
If Err = 100 Then
  ErrorString$ = ErrorString$ & Chr(10) & "DaqBook/100 Error : " + Hex$(daqErrno%)
ret% = MsgBox("Could not open form or Daqbook could not be initiated. Continue? ", 4)
If ret% = 6 Then
```

```
 timer2.Enabled = False
 timer1.Enabled = False
 timer3.Enabled = False
 command1.Enabled = False
 command2.Enabled = False
 command3.Enabled = False
 command4.Enabled = False
Else
 Unload form2
End If
Exit Sub
End Sub

Sub Form_Unload (Cancel As Integer)
 If logga Then
  Close #1
 End If
End Sub

Sub loggfile_Click ()
' välj loggfil meny
 On Error GoTo Errhandler


 cmdialog1.Filter = "Alla filer|*.*|Text filer|*.txt|Logg filer|*.txt"
 cmdialog1.FilterIndex = 3
 cmdialog1.Action = 2

 Exit Sub
Errhandler:
 MsgBox "Kan inte öppna dialogbox", MB_OK
 Exit Sub

End Sub

Sub miccoord_click ()
' meny ange postion
  xkoord$ = InputBox("Mata in alla mikrofonkoordinater på formen x1 x2 x3 x4", "Mikrofon x-koordinater")
  ykoord$ = InputBox("Mata in alla mikrofonkoordinater på formen y1 y2 y3 y4", "Mikrofon y-koordinater")
End Sub

Sub pos_Click ()
' meny Kör position best
 Call start_click

 On Error GoTo poserrorhandler

 aname = ""
 apoint% = False
 For i% = Len(cmdialog1.Filename) To 1 Step -1
  If Mid(cmdialog1.Filename, i%, 1) = "\" Or Mid(cmdialog1.Filename, i%, 1) = ":" Then
   apoint% = False
  End If
  If apoint% = True Then
   aname = Mid(cmdialog1.Filename, i%, 1) + aname
  End If
  If Mid(cmdialog1.Filename, i%, 1) = "." Then
   apoint% = True
  End If
 Next i%
```

```
  If xkoord = "" Or ykoord = "" Then
    Call miccoord_click
  End If
  Load form1
  form1.Show 0
  ofs2.LinkExecute "process2([" + xkoord$ + "],[" + ykoord$ + "]," + Filter$ + ",'" + cmdialog1.Filename + "','"
+ aname + "')"
  Unload form1
 Exit Sub
 poserrorhandler:
  MsgBox "Error when running matlabcommand"
  Unload form1
  Exit Sub
 End Sub

 Sub start_click ()
 ' initiera DDE-länk meny
  Dim first%


   first% = True
   On Error GoTo errorhandler
   ofs2.LinkMode = 0
   ofs2.LinkTopic = "MATLAB|Engine"
   ofs2.LinkItem = "EngEvalString"
   ofs2.LinkMode = 2  'manual link
   ddeon% = True
   ofs2.LinkExecute "path(path,'c:\matlab');"
   ofs2.LinkExecute "system_dependent(12,'off')"

  Exit Sub

errorhandler:
 If (Err = 282) And (first% = True) Then  'matlab var ej igång
   x = Shell("c:\matlab\bin\matlab", 1)
   first% = False
   Resume
 Else
   MsgBox "Kan inte öppna DDE-konversation med matlab", MB_OK
 End If
 Exit Sub
End Sub

Sub Timer1_Timer ()
Dim ref%, co%, si%, angle%, co2%, si2%, tot#
ReDim buf%(200)

Cls
Print "Compass-test": Print
' Set error handler and initialize DaqBook/100
' ret% = VBdaqSetErrHandler%(100)
' On Error GoTo ErrorHandlerADC1
' ret% = VBdaqInit%(LPT1%, 7)

' Disable channel tags


' read sin2
' ret% = VBdaqAdcRd%(4, si2%, DgainX1%)
```

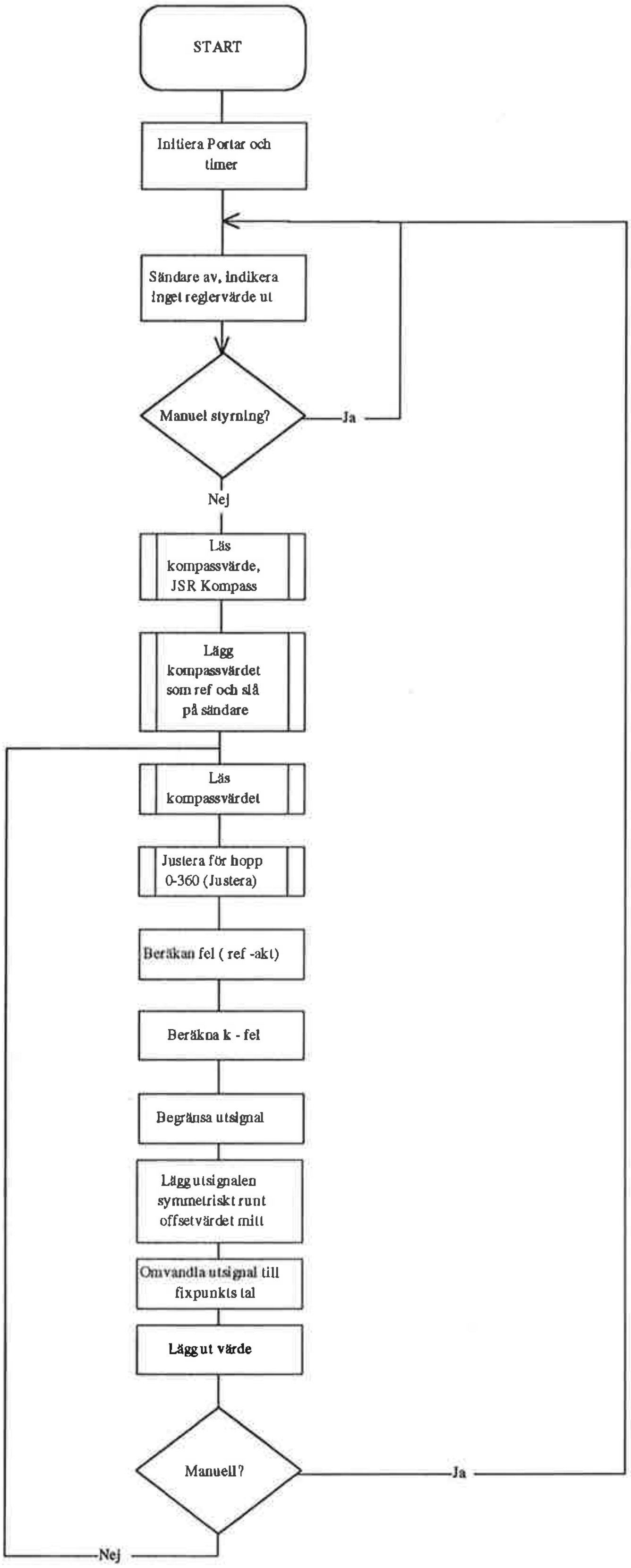
```
' Close and exit
' ret% = VBdaqClose%()

Exit Sub


End Sub

Sub Timer2_Timer ()
'  ret% = VBdaqCtrMultCtrl(Dmccload, 1, 1, 1, 1, 0)
 ' ret% = VBdaqCtrMultCtrl(DmccArm, 1, 1, 1, 1, 0)
 timer3.Enabled = True
 ret% = VbdaqDigWtBit%(DdpLocalClow%, 0, 0)

 For i = 1 To 1000
 Next
 ret% = VbdaqDigWtBit%(DdpLocalClow%, 0, 1)

 For i = 1 To 100
 Next
 ret% = VBdaqCtrMultCtrl(Dmccload, 1, 1, 1, 1, 0)
 ret% = VBdaqCtrMultCtrl(DmccArm, 1, 1, 1, 1, 0)
 timer3.Enabled = True
 timer2.Enabled = False


End Sub

Sub Timer3_Timer ()
' Loggar värde för både position och kompass
 ReDim c1%(1), c2%(1), c3%(1), c4%(1), c5%(1)
 ReDim tal(1 To 5) As vektype
 Dim rad As String
 Dim diff&, i%, min&
 Dim ref%, co%, si%, angle%, co2%, si2%, tot#
 ReDim buf1%(200), buf2%(200)
 Dim off1%, off2%

    ret% = VBDaqCtrRdnFore%(tal(1).vek(), tal(2).vek(), tal(3).vek(), tal(4).vek(), c5%(), 1)
    min& = 66000
    For i% = 1 To 4
      If tal(i).vek(1) < 0 Then
        temp& = 65536 + CLng(tal(i).vek(1))
      Else
        temp& = CLng(tal(i).vek(1))
      End If
      If (temp& < min&) Then
        min& = temp&
      End If
```

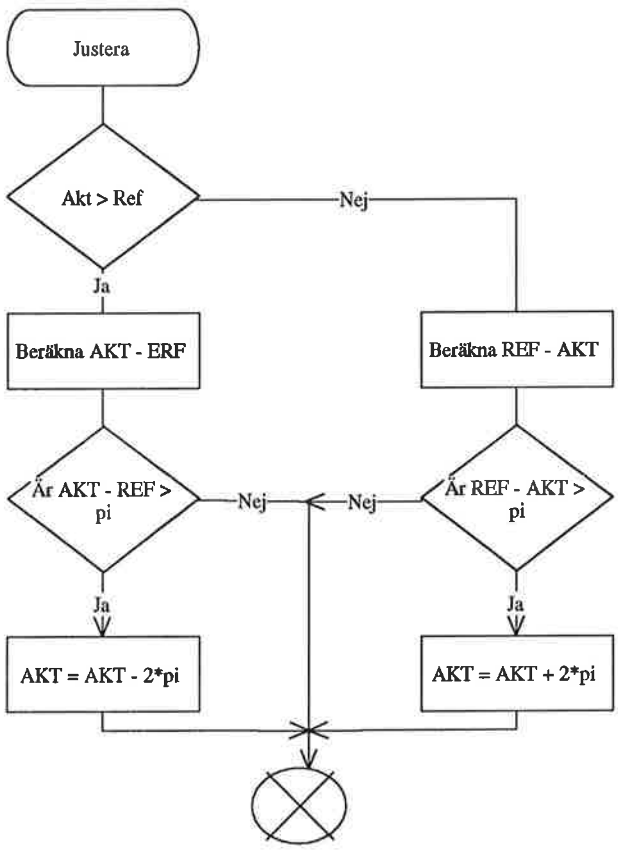
```
Next i
' räknar ut tiden till nästa aktivering
tid& = 112000 - (37000 - min&) - 18000
timer2.Interval = CInt(tid& / 100)
timer2.Enabled = True


' logga kompass
ret% = VBdaqAdcSetTag%(0)
ret% = VBdaqAdcSetFreq(2000!)
' läser cos och sin
ret% = VBdaqAdcRdn%(0, buf1%(), 200, DtsPacerClock%, 0, 0, 4000!, Dgainx1%)
ret% = VBdaqAdcRdn%(1, buf2%(), 200, DtsPacerClock%, 0, 0, 4000!, Dgainx1%)
' medel cos
tot# = 0
For x = 0 To 199
 tot# = buf1%(x) / 200 + tot
Next x
co2% = CInt(tot#)


' medel sin
tot# = 0
For x = 0 To 199
 tot# = buf2%(x) / 200 + tot
Next x
si2% = CInt(tot#)
On Error Resume Next
'Calculate compass-angle
co2% = co2 - ofs1.Text  ' offset
si2% = si2 - ofs2.Text

If co2 <> 0 Then
 angle% = Atn(si2 / co2) * 180 / 3.14
 If (co2 < 0) And (si2 >= 0) Then
   angle% = angle + 180
 ElseIf co2 < 0 Then
   angle% = angle - 180
 End If
 grader2.Caption = angle
Else
 If si2 >= 0 Then
  ' Print "The angle is close to 90 degrees"
   grader2.Caption = 90
 Else
  ' Print "The angle is close to -90 degrees"
   grader2.Caption = -90
 End If
End If
' kopierar till skalärer
c1%(0) = tal(1).vek(1)
c2%(0) = tal(2).vek(1)
c3%(0) = tal(3).vek(1)
c4%(0) = tal(4).vek(1)
' S2-S1
diff& = CLng(c2%(0)) - CLng(c1%(0))
co12.Text = diff& * .00001 * 340
' S3-S1
diff& = CLng(c3%(0)) - CLng(c1%(0))
co13.Text = diff& * .00001 * 340
```

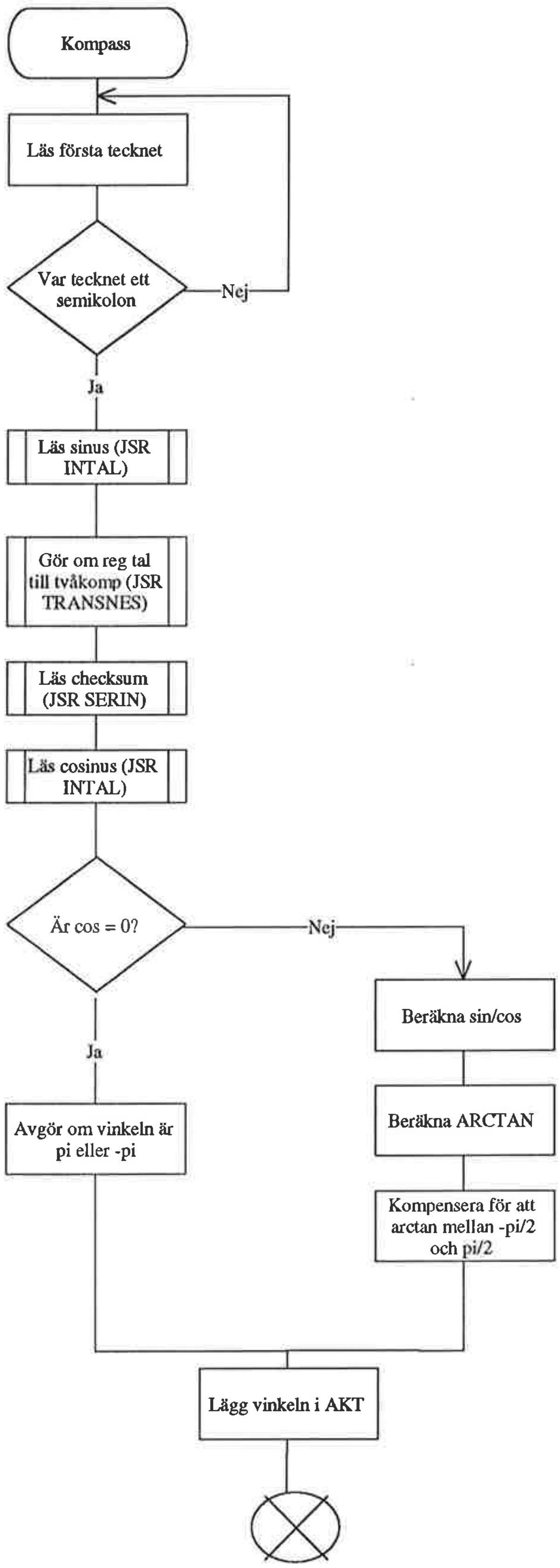
```
    ' S4-S1
    diff& = CLng(c4%(0)) - CLng(c1%(0))
    co14.Text = diff& * .00001 * 340

   ' Print (co12.Text + "    " + co13.Text + "    " + co14.Text)
    ' skapar text som skall lagras i filen
    rad = "0  " + co12.Text + "  " + co13.Text + "  " + co14.Text + "  " + grader2.Caption

    For i% = 1 To Len(rad)
      If Mid(rad, i, 1) = "," Then
        Mid(rad, i, 1) = "."
      End If
    Next i
    If logga Then
      Print #1, rad
    End If
    timer3.Enabled = False
End Sub

Sub Watch_Click ()
' se loggfil knapp
 x = Shell("notepad " + cmdialog1.Filename, 1)


End Sub
```

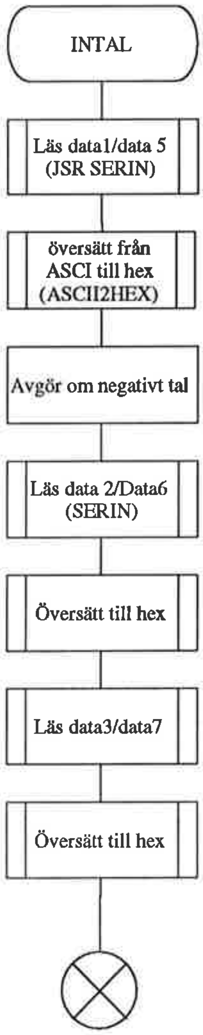
Assemble code for the Motorola 68705

Flow charts for Regtest

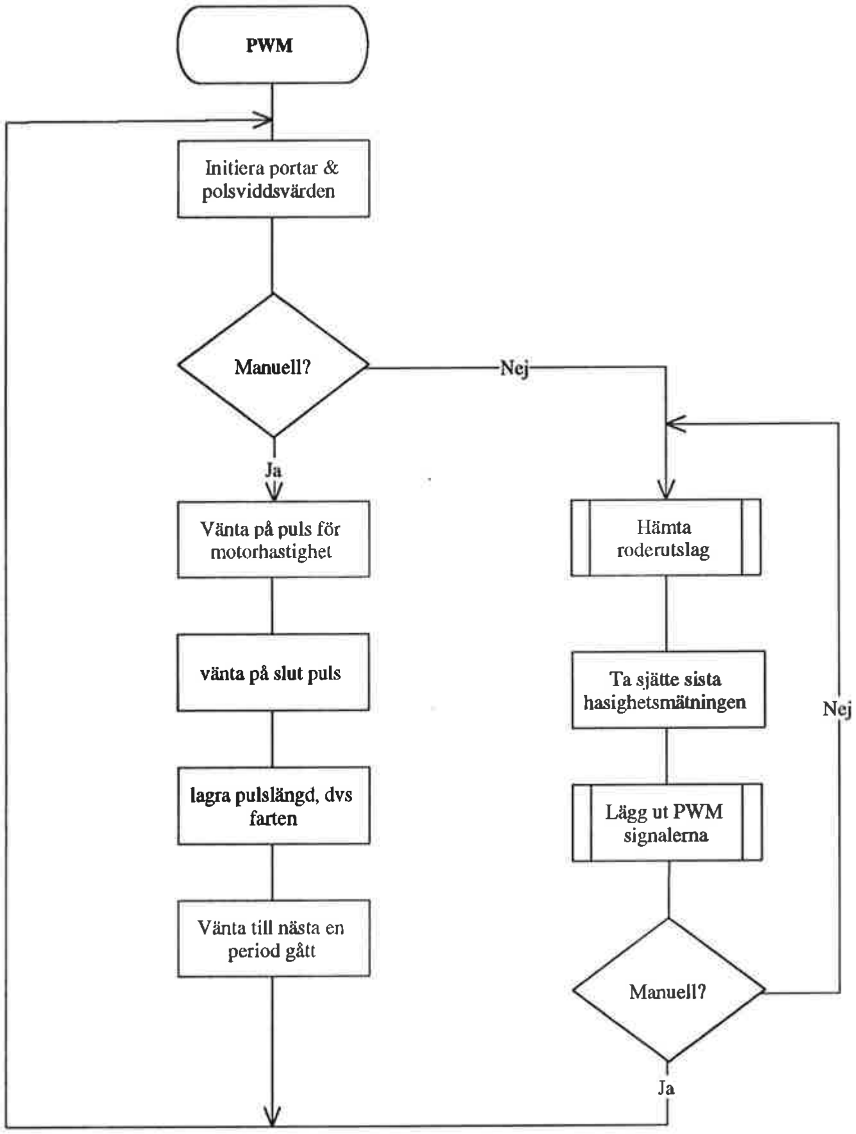



Justera
Akt > Ref
Nej
Ja
Beräkna AKT - ERF
Beräkna REF - AKT
Är AKT - REF > pi
Nej
Nej
Är REF - AKT > pi
Ja
Ja
AKT = AKT - 2*pi
AKT = AKT + 2*pi


Kompass
Läs första tecknet
Var tecknet ett semikolon
Nej
Ja
Läs sinus (JSR INTAL)
Gör om reg tal till tvåkomp (JSR TRANSNES)
Läs checksum (JSR SERIN)
Läs cosinus (JSR INTAL)
Är cos = 0?
Nej
Ja
Beräkna sin/cos
Beräkna ARCTAN
Kompensera för att arctan mellan -pi/2 och pi/2
Avgör om vinkeln är pi eller -pi
Lägg vinkeln i AKT


INTAL
Läs data1/data 5
(JSR SERIN)
översätt från
ASCI till hex
(ASCII2HEX)
Avgör om negativt tal
Läs data 2/Data6
(SERIN)
Översätt till hex
Läs data3/data7
Översätt till hex

Flow-chart for the PWM program



Code for processor 1 - Control

```
;************************ DEKL28 *************************

PORTA       EQU     0                   ;Port A Data Register
PORTB       EQU     1                   ;Port B Data Register
PORTC       EQU     2                   ;Port C Data Register

DDRA        EQU     4                   ;Port A Data Direction Register
DDRB        EQU     5                   ;Port B Data Direction Register
DDRC        EQU     6                   ;Port C Data Direction Register

TDR         EQU     8                   ;Timer Data Register
TCR         EQU     9                   ;Timer Control Register

MOR         EQU     $784                ;Mask Option Register
TIRVEK      EQU     $7F8                ;Timer Interrupt Vector
EXTVEK      EQU     $7FA                ;External Interrupt Vector
SWIVEK      EQU     $7FC                ;Software Interrupt Vector
RESVEK      EQU     $7FE                ;Reset Vector

RAM         EQU     $10                 ;Minne för variabler

EPROM       EQU     $80                 ;Minne för konstanter och program


            ORG     MOR                 ;Definiera innehållet i MOR
            DB      %00000000   ;Kristalloscillator (MSB=1 ger RC)


            ORG     TIRVEK              ;Definiera timeravbrottsvektorn
            DW      START               ;Adressen anges här

            ORG     EXTVEK              ;Definiera externa avbrottsvektorn
            DW      START               ;Adressen anges här

            ORG     SWIVEK              ;Definiera programavbrottsvektorn
            DW      START               ;Adressen anges här

            ORG     RESVEK              ;Definiera resetvektorn
            DW      START               ;START är adressen till programstarten


MAXUTSLAG EQU       40
MITT                EQU         128

            ORG     RAM                 ;Variabler

;Här läggs programmets variabler, deklarerade med DS-satser
;Hit flyttas också variablerna från biblioteksmodulerna
RAKN:       DS      1

FELM:       DS      1
FELL:       DS      1
FELE:       DS      1

REFM:       DS      1                   ;REFERENSVÄRDE
```

```
AKTM:       DS      1               ;AKTUELLT VÄRDE
AKTL:       DS      1
AKTE:       DS      1

TMP:        DS      1

INDATA:     DS      1
BCNTR:      DS      1

; VARIABLER FÖR MATH687.LIB
R1M         DS      1               ;Operand 1 mantissa MSB
R1L         DS      1               ;Operand 1 mantissa LSB
R1E         DS      1               ;Operand 1 exponent
R2M         DS      1               ;Operand 2 mantissa MSB
R2L         DS      1               ;Operand 2 mantissa LSB
R2E         DS      1               ;Operand 2 exponent
SIGN1       DS      1
DUMMY1      DS      1
DUMMY2      DS      1

SIGN2       DS      1
COUNT       DS      1               ;Hjälpvariabler
LPCNT       DS      1
SFLAG       DS      1               ;Teckenflagga
IXTMP       DS      1               ;Temporär lagring av IX
TEMP1       DS      1
TEMP2       DS      1

; VARIABLER FÖR POLYNOM.LIB
GRAD        DS      1               ;Temporär lagring av gradtal
FT          DS      3               ;Temporär lagring av flyttal

TRISIG      DS      2                           ;Används för teckenbestämning


            ORG     EPROM                       ;Programmets konstanter

;Här läggs programmets konstanter, textsträngar och tabeller

;           FÖRSTÄRKNING
FÖRST:      DB      0
            DB      %00000110   ;FÖRST 6 GGR, URSPR. 48 GGR
            DB      0


;**** Polynomial coefficients for atan ***
;Valid for  0 < X < 1.0
ATAN:       DB      6                           ;Order
            DB      $B8,$2A,$ED ;6              -3.5076E-02
            DB      $6F,$25,$ED ;5              5.4272E-02
            DB      $4C,$A6,$EF ;4              1.4970E-01
            DB      $9B,$4A,$F0 ;3              -3.9341E-01
            DB      $56,$D9,$EB ;2              1.0602E-02
            DB      $7F,$E9,$F1 ;1              9.9933E-01
            DB      $61,$31,$E0 ;0              5.7932E-06

START:                                          ;Här startar programmet
;Här läggs programmet in
```

```
        LDA #%11111111                  ;PORTA UT
        STA DDRA
        LDA #%11111110                  ;B0 IN, RESTEN UT
        STA DDRB
        LDA #%11110000                  ;PORTC IN (K-JUSTERING)
        STA DDRC

        LDA #%00000000
        STA PORTA

        JSR SIST
        JSR SETTIMER

MANUELL:        BSET 1,PORTB                    ; SÄNDARE AV
                BCLR 2,PORTB                    ;INGET RIKTIGT VÄRDE
UT
                BRCLR 0,PORTB,MANUELL

AUTOMATISK:     JSR KOMPASS     ;LÄGGER KOMPASSVÄRDE I AKT
                LDX #AKTM           ;KOPIERA VÄRDET TILL REFERENS
                LDA #1
                JSR PUTR
                LDX #REFM
                LDA #1
                JSR GETR
                BCLR 1,PORTB                    ;SÄNDARE PÅ

REGLERA:        JSR KOMPASS         ;LÄGGER KOMPASSVÄRDE I AKT
                JSR JUSTERA     ;JUSTERAR FÖR HOPP 0 TILL 360 GRADER

                LDX #REFM           ;REF I R1
                LDA #1
                JSR PUTR

                LDX #AKTM           ;AKT I R2
                LDA #2
                JSR PUTR
                JSR SUB                         ;FELET LIGGER NU I R1
                LDX #FELM
                LDA #1
                JSR GETR            ;FELET LIGGER NU I FEL

                LDX #FÖRST      ;RÄKNA UT FÖRSTÄRKNING
                LDA #1
                JSR PUTR

                LDA #0          ;MULTIPLICERA MED DIP-SWICHARNA
                STA R2M
                STA R2E
                LDA PORTC
                AND #$0F
                STA R2L
                JSR MUL                         ;FÖRST FINNS NU I R1

                LDX #FELM
                LDA #2
                JSR PUTR
                JSR MUL                         ;FEL * K LIGGER NU I R1
```

```
                LDA #1
                JSR NEGERA          ;- FEL * K LIGGER NU I R1

                ;BEGRÄNSNING AV UTSIGNAL
                LDX #FELM
                LDA #1
                JSR GETR            ;UTSIGN LIGGER NU I FEL

BEGR:           LDA #0              ;JÄMFÖR MED MAXVÄRDET
                STA R2E
                STA R2M
                LDA #MAXUTSLAG
                STA R2L
                JSR FCMP
                BLS BEGR2           ;OM EJ FÖR STORT SÅ FORTSÄTT
                LDA #0      ;ANNARS BEGRÄNSA
                STA FELE
                STA FELM
                LDA #MAXUTSLAG
                STA FELL
                BRA SYMM
BEGR2:          LDX #FELM           ;HÄMTA UTSIGN
                LDA #1
                JSR PUTR
                LDA #0      ;JÄMFÖR MED MINVÄRDET
                STA R2E
                LDA #$FF
                STA R2M
                SUB #MAXUTSLAG
                ADD #1
                STA R2L
                JSR FCMP
                BHS SYMM            ;OM EJ FÖR STORT GÅ VIDARE
                LDA #0              ;ANNARS LÄGG IN MINVÄRDE
                STA FELE
                LDA #$FF
                STA FELM
                LDA #$FF
                SUB #MAXUTSLAG
                ADD #1
                STA FELL

SYMM:           LDX #FELM
                LDA #1
                JSR PUTR

                LDA #0
                STA R2M
                STA R2E
                LDA #MITT
                STA R2L
        JSR ADD                     ;R1 ÄR NU SYMMETRISK RUNT MITT
                JSR FIXP            ;OMVANDLA TILL HELTAL

                LDA R1L
                BCLR 2,PORTB                ;VÄRDET SKA ÄNDRAS
                STA PORTA           ;LÄGG UT UTSIGNAL
                BSET 2,PORTB                ;VÄRDE OK

                BRSET 0,PORTB,REG
```

```
                JMP MANUELL
REG:            JMP REGLERA

; **************** HÄR BÖRJAR ALLA UNDERPROCEDURER *****************


        ;TAR TVÅKOMPLEMENT PÅ R1 ELLER R2
NEGERA:         CMP #2
                BEQ NEG3
                COM R1M             ;TECKENVÄND R1
                NEG R1L
                BCS NEG2
                INC R1M
NEG2:           RTS
NEG3:           COM R2M             ;TECKENVÄND R2
                NEG R2L
                BCS NEG4
                INC R2M
NEG4:           RTS


JUSTERA:    ; FIXA HOPPET MELLAN 0 OCH 360 GRADER
                LDX #AKTM           ;LÄGG AKT I R1
                LDA #1
                JSR PUTR
                LDX #REFM           ;LÄGG REF I R2
                LDA #2
                JSR PUTR
                JSR FCMP            ;JÄMFÖR AKT OCH REF
                BLS  REFSTÖRST
                AKTSTÖRST; LDX #AKTM          ;LÄGG AKT I R1
                LDA #1
                JSR PUTR
                LDX #REFM           ;LÄGG REF I R2
                LDA #2
                JSR PUTR
                          ;STÖRSTA TALET, DVS AKT, LIGGER I R1
                JSR SUB                       ;SKILLNADEN MELAN
TALEN I R1

                LDA #$64            ;LÄGG PI I R2
                STA R2M
                LDA #$87
                STA R2L
                LDA #$F3
                STA R2E
                JSR FCMP            ;SKILJER SIG AKT OCH REF MINDRE ÄN
180GRADER?
                BLS SLUT            ;SÅ SKA INGET GÖRAS

                LDX #AKTM           ;LÄGG AKT I R1
                LDA #1
                JSR PUTR
                LDA #$64            ;LÄGG 2*PI I R2
                STA R2M
                LDA #$87
                STA R2L
                LDA #$F4
                STA R2E       ;MINSKA AKT MED 2*PI
```

```
            JSR SUB         ;LÄGG NYTT VÄRDE I AKT
            LDX #AKTM
            LDA #1
            JSR GETR
            RTS
REFSTÖRST:  LDX #REFM               ;LÄGG REF I R1
            LDA #1
            JSR PUTR
            LDX #AKTM               ;LÄGG AKT I R2
            LDA #2
            JSR PUTR
                                    ;STÖRSTA TALET LIGGER NU I R1
            JSR SUB                 ;SKILLNADEN MELLAN TALEN I R1

            LDA #$64                ;LÄGG PI I R2
            STA R2M
            LDA #$87
            STA R2L
            LDA #$F3
            STA R2E
            JSR FCMP        ;SKILJER SIG AKT OCH REF MER ÄN 180GRADER?

            BLS SLUT                ;INGET SKA GÖRAS
            LDX #AKTM               ;ÖKA AKT MED 2*PI
            LDA #1
            JSR PUTR
            LDA #$64                ;LÄGG 2*PI I R2
            STA R2M
            LDA #$87
            STA R2L
            LDA #$F4
            STA R2E
            JSR ADD                 ;LÄGG NYTT VÄRDE I AKT
            LDX #AKTM
            LDA #1
            JSR GETR
SLUT:       RTS                     ; AKT OCH REF SKILJER SIG NU ÅT MED
                                    MINDRE ÄN 180 GRADER


PUTR:       ;KOPIERAR VARIABEL VARS ADRESS ÄR I X TILL R1/R2
            ;ACK=1 OM TILL R1, ACK=2 OM TILL R2
            ;INDEXREG INNEHÅLLER ADRESS TILL ANDRA VARIABELN
            CMP #1
            BNE PUTR2
PUTR1:      LDA ,X
            STA R1M
            LDA 1,X
            STA R1L
            LDA 2,X
            STA R1E
            RTS
PUTR2:      LDA ,X
            STA R2M
            LDA 1,X
            STA R2L
            LDA 2,X
            STA R2E
            RTS
```

```
GETR:                   ;KOPIERAR R1/R2 TILL VARIABEL VARS ADRESS ÄR I X
                        ;ACK=1 OM FRÅN R1, ACK=2 OM TILL R2
                        ;INDEXREG INNEHÅLLER ADRESS TILL ANDRA VARIABELN
                        CMP #1
                        BNE GETR2
GETR1:                  LDA R1M
                        STA ,X
                        LDA R1L
                        STA 1,X
                        LDA R1E
                        STA 2,X
                        RTS
GETR2:                  LDA R2M
                        STA ,X
                        LDA R2L
                        STA 1,X
                        LDA R2E
                        STA 2,X
        RTS

KOMPASS:                ;LÄS IN OCH ANPASSA INDATA
                        JSR WAITFIRST
                        JSR SERIN               ;SEMIKOLON IN
                        CMP #$3B
                        BNE KOMPASS                     ;OM EJ SEMIKOLON BÖRJA
OM
                        LDX #R1M
                        JSR INTAL               ;LÄS IN SINUS I R1
                        LDX #R1M
                        JSR TRANSNEG                    ;GÖR OM NEGATIVT TAL
TILL TVÅKOMPLEMENT

                        JSR SERIN               ;CHECKSUMMA

                        LDX #R2M
                        JSR INTAL               ;LÄS IN COS

        LDX #R2M
        JSR TRANSNEG


                        LDA #1
                        LDX #FELM
                        JSR GETR                ;LAGRA SIN I FEL

                        LDA #2
                        LDX #AKTM
                        JSR GETR                ;LAGRA COS I AKT

                        LDA R2M                 ;UNDVIK DIVISION MED 0
                        CMP #0
                        BNE DIVIDERA
                        LDA R2L
                        CMP #0
                        BNE DIVIDERA

                        ;OM SIN >=0 SÅ LÄGG IN PI/2 I R1,
                        ;ANNARS LÄGG IN -PI/2
                        LDA #$64                ; R1= PI/2
                        STA R1M
```

```
                LDA #$87
                STA R1L
                LDA #$B9
                STA R1E
                BRA SPARA

                LDA FELM        ;COS = 0, ÄR VINKELN +90 ELLER -90?
                LSL A
                BCC MINUS               ;SIN NEG, VINKELN -90
                BRA SPARA
MINUS:          LDA #1                  ;KOPIERA PI/2 TILL R1
                LDX #R2M
                JSR GETR
                LDA #0
                STA R1M
                STA R1L
                STA R1E
                JSR SUB                 ;-PI/2 NU I R1

                BRA SPARA

DIVIDERA:
                JSR DIV                 ;SIN/COS I R1
                SR ARCTAN               ;VINKELN I RADIANER I R1


                LDA AKTM                ;LÄS IN COS (MSBYTE)
                LSL A                                   ;SKIFTA VÄNSTER
                BCC SPARA
                LDA FELM                ;LADDA SIN  (MSBYTE)
                LSL A
                BCS VINKEL2

                LDA #$64                ;ÖKA VINKELN MED PI
                STA R2M
                LDA #$87
                STA R2L
                LDA #$F3
                STA R2E
                JSR ADD
                BRA SPARA

VINKEL2:        DA #$64                 ;MINSKA VINKELN MED PI
                STA R2M
                LDA #$87
                STA R2L
                LDA #$F3
                STA R2E
                JSR SUB
                BRA SPARA

SPARA:          DA #1
                DX #AKTM
                SR GETR                 ;KOPIERA VINKELN TILL AKT
                TS
```

```
                ;ÖVERSÄTTER TILL TVÅKOMPL OM SIGNX ÄR 1
TRANSNEG:       DA 6,X                  ;SIGN1 RESP SIGN2
                CMP #1          ;ÄR TALET NEGATIVT?
                BNE TRANSNEG2
                A #%11111111
                B 1,X
                D #1
                A 1,X
                LDA #%11111111
                ADC #0
                SUB ,X
                STA ,X
TRANSNEG2:      RTS


INTAL:          JSR SERIN               ;DATA1
                JSR ASCII2HEX                   ;ÖVERSÄTT ASCII-TECKEN
TILL HEX
                STA ,X                          ;LÄGG IN I R1M RESP R2M
                AND #%00001000
                BEQ INTAL2              ;POSITIVT TAL
                LDA #1
                STA 6,X                 ;ETTA I SIGN1 RESP SIGN2 OM
NEGATIVT
                LDA ,X                  ;MASKA BORT TECKENBIT
                AND #%00000111
                STA ,X
                BRA INTAL3
INTAL2:         LDA #0                  ;NOLLA OM POSITIVT
                STA 6,X
INTAL3:         INC X
                JSR SERIN               ;DATA2
                JSR ASCII2HEX
                LSL A
                LSL A
                LSL A
                LSL A
                STA TMP         ;IN I TMP
                JSR SERIN               ;DATA3
                JSR ASCII2HEX
                ORA TMP
                STA ,X                          ;IN I R1L ELLER R2L
                LDA #0
                STA 1,X                         ;NOLLA I EXPONENTEN
                RTS


ASCII2HEX:      ;INDATA I ACK
                AND #%01111111                  ;SORTERA BORT EV
TECKENBIT
                CMP #$40
                BHI ASC2                ;A TILL F
                SUB #$30
                RTS
ASC2:           SUB #$41
                ADD #10
```

```
                    RTS                          ;HEXADECIMALT TAL
FINNS NU I ACK

WAITFIRST:  BIH WAITFIRST
                    LDA #10
                    STA BCNTR
AGAIN:              JSR TIME
                    BIH WAITFIRST                ;IF HIGH START OVER
                    DEC BCNTR
                    BNE AGAIN
                    RTS

                    ;LÄS IN DATA OCH INVERTERA
SERIN:              BIL SERIN             ;VÄNTA PÅ STARTBIT
                    JSR HALFTIME
                    BIL SERIN     ;FALSK STARTBIT
                    LDA #8
                    STA BCNTR
SERIN1:
                    JSR TIME
                    BIH SERIN2
                    SEC
                    BRA SERIN3
SERIN2:             NOP
                    NOP
                    CLC
SERIN3:
                    ROR INDATA
                    LDA INDATA
                    DEC BCNTR
                    BNE SERIN1


                    RTS

SETTIMER:           LDA #%01001100
                    STA TCR
                    RTS

TIME:               LDA #156
                    STA TDR
                    BCLR 7,TCR
TIME2:              BRCLR 7,TCR,TIME2            ;WAIT FOR TIMER TO
FINISH
                    RTS

HALFTIME:           LDA #78
                    STA TDR
                    BCLR 7,TCR
HALFTIME2:          BRCLR 7,TCR,HALFTIME2        ;WAIT FOR TIMER TO
FINISH
                    RTS


;Efter programmet läggs biblioteksmodulerna in.
;Variabler flyttas till RAM,
;Konstanter och tabeller flyttas till början av EPROM.
```

```
;********************** MATH687.LIB *******************
;
; Biblioteksmodul för flyttalsräkning i MC68705
;
; Modulen omfattar följande tio subrutiner:
;
;           ADD             Addition
;           SUB             Subtraktion
;           MUL             Multiplikation
;           DIV             Division
;           FCMP            Jämförelse
;           INVERT          Invertering
;           ABS             Absolutbelopp
;           FIXP            Omvandling till heltal
;           PACK            Hjälprutin för normalpackning
;           SIG             Hjälprutin för teckenhantering
;

;************************* Addition ******************
;
; Flyttalsaddition. De två flyttalen som skall adderas
; skall innan subrutinanropet ligga som flyttal i R1 och R2,
; och summan hamnar som ett flyttal i R1, dvs R1 = R2 + R1
;
; Ingångsparametrar: Flyttal i R1 och R2
;
; Utgångsparameter: Flyttal i R1
;
; Exekveringstiden beror av talens inbördes storlek och
; om de är på normalform vid subrutinanropet eller ej.
; För ungefär lika stora tal på normalform är exekveringstiden
; typiskt mindre än 0,5 millisekund. För extremt olika
; storleksordning på talen kan tiden bli nära 5 millisekunder
;
;**********************************************************

ADD:
        STX     IXTMP       ;Spara indexregistret
        JSR     PACK        ;packa talen på normalform
        LDX     #R1M
        LDA     2,X
        SUB     5,X
        BEQ     ADD2
        BMI     ADD1
        NEG     A
        INC     X
        INC     X
        INC     X
ADD1:
        ASR     ,X
        ROR     1,X
        INC     2,X
        INC     A
        BNE     ADD1
ADD2:
        LDX     #R1M
        LDA     1,X
```

```
            ADD     4,X
            STA     1,X
            LDA     ,X
            BPL     ADD3
            TST     3,X
            BPL     ADD4
            ADC     3,X
            BMI     ADD5
            BRA     ADD7
ADD3:
            TST     3,X
            BMI     ADD4
            ADC     3,X
            BPL     ADD5
            BRA     ADD7
ADD4:
            ADC     3,X
ADD5:
            STA     ,X
ADD6:
            LDX     IXTMP
            RTS                     ;Slut på subrutin ADD

ADD7:
            ROR     A
            STA     ,X
            ROR     1,X
            INC     2,X
            BRA     ADD6


;************************ Subtraktion ****************
;
; Subrutin för flyttalssubtraktionen R1 = R2 - R1
;
; Ingångsparametrar: Flyttal i R1 och R2
;
; Utgångsparameter: Flyttal i R1
;
; Exekveringstiden beror av talens inbördes storlek och om de
; är på normalform vid subrutinanropet eller ej.
; För ungefär lika stora tal på normalform är exekveringstiden
; typiskt mindre än 0,5 millisekund. För extremt olika
; storleksordning på talen kan tiden bli nära 5 millisekunder
;
;*********************************************************

SUB:
            COM     R2M             ;Teckenvänd R2
            NEG     R2L
            BCS     SUB1
            INC     R2M
SUB1:
            BRA     ADD             ;Använd addition


;********************** Multiplikation ***************
;
; Utför flyttalsmultiplikationen R1 = R1*R2
;
```

```
; Ingångsparametrar: Flyttal i R1 och R2
;
; Utgångsparameter: Flyttal i R1
;
; Exekveringstiden beror av talens inbördes storlek och om de
; är på normalform vid subrutinanropet eller ej.
; För ungefär lika stora tal på normalform är exekveringstiden
; typiskt c:a 1 millisekund. För extremt olika
; storleksordning på talen kan tiden bli nära 2 millisekunder
;
;**********************************************************
;

MUL:
        STX     IXTMP       ;Spara indexregistret
        TST     R1M         ;kontrollera om ena talet är noll
        BNE     MUL1
        TST     R1L
        BEQ     MUL5        ;i så fall utförs ingen multiplikation
MUL1:
        TST     R2M         ;Kontrollera om andra talet är noll
        BNE     MUL2
        TST     R2L
        BEQ     MUL5        ;i så fall utförs ingen multiplikation.
MUL2:
        JSR     SIG         ;Bilda beloppet och lagra tecknet
        JSR     PACK        ;R1 och R2 på normalform
        CLR     TEMP1
        CLR     TEMP2
        LDA     #$10
        STA     COUNT
MUL3:
        LSR     R2M
        ROR     R2L
        BCC     MUL4
        LDA     TEMP2
        ADD     R1L
        STA     TEMP2
        LDA     TEMP1
        ADC     R1M
        STA     TEMP1
MUL4:
        ROR     TEMP1
        ROR     TEMP2
        DEC     COUNT
        BNE     MUL3
        LDA     TEMP1
        STA     R1M
        LDA     TEMP2
        STA     R1L
        JSR     SIG4
        LDA     R1E
        ADD     R2E
        ADD     #$10
        STA     R1E
        BRA     MUL6

MUL5:
        CLR     R1M
        CLR     R1L
        CLR     R1E
```

```
MUL6:
        LDX     IXTMP
        RTS                     ;Slut på subrutin MUL


;************************ Division ********************
;
; Subrutin för flyttalsdivisionen R1 = R1 / R2
;
; Ingångsparametrar: Flyttal i R1 och R2
;
; Utgångsparameter: Flyttal i R1

; Exekveringstiden beror av talens inbördes storlek och om de
; är på normalform vid subrutinanropet eller ej.
; För ungefär lika stora tal på normalform är exekveringstiden
; typiskt mindre än 2 millisekunder. För extremt olika
; storleksordning på talen kan tiden bli nära 4 millisekunder
;
;**********************************************************

DIV:
        STX     IXTMP           ;Spara IX
        TST     R1M
        BNE     DIV1
        TST     R1L
        BEQ     DIV6            ;Gör inget om täljaren = 0
DIV1:
        TST     R2M
        BNE     DIV2
        TST     R2L
        BEQ     DIV7            ;Overflow om nämn = 0
DIV2:
        JSR     SIG
        JSR     PACK
        LDA     R1E
        ADD     #$F2
        SUB     R2E
        STA     R1E
        LDA     R1L
        STA     TEMP2
        LDA     R1M
        STA     TEMP1
        LDA     #$0F
        STA     COUNT
DIV3:
        LDA     TEMP2
        SUB     R2L
        STA     TEMP2
        LDA     TEMP1
        SBC     R2M
        STA     TEMP1
        BPL     DIV4
        LDA     TEMP2
        ADD     R2L
        STA     TEMP2
        LDA     TEMP1
        ADC     R2M
        STA     TEMP1
        CLC
```

```
            BRA         DIV5

DIV4:
            SEC
DIV5:
            ROL         R1L
            ROL         R1M
            LSR         R2M
            ROR         R2L
            DEC         COUNT
            BNE         DIV3
            LDA         #%01111111
            AND         R1M
            STA         R1M
            JSR         SIG4
DIV6:
            LDX         IXTMP
            RTS                     ;Slut på subrutin DIV

DIV7:
            LDA         #$FF
            STA         R1L
            LDA         #$7F
            STA         R1M         ;Vid overflow blir
            STA         R1E         ;resultatet 7FFF * 2^7F
            BRA         DIV6


;************************* Jämför **********************
;
; Flyttalsrutin för jämförelse av två tal på flyttalsformat
;
; Jämförelsen utförs som subtraktionen R1 - R2 och ställer
; statusflaggorna C och Z på samma sätt som instruktionen
; CMP gör vid heltalsjämförelser:
;
;                       R1 = R2:  Z=1
;                       R1 < R2:  Z=0, C=1
;                       R1 > R2:  Z=0, C=0
;
; Följande villkorliga hoppinstruktioner kan användas efter
; subrutinen FCMP:
;
;           BEQ ger hopp om   R1 lika med R2
;           BNE                         R1 skilt från R2
;           BLO                         R1 mindre än R2
;           BLS                         R1 mindre än eller lika med R2
;           BHS                         R1 större än eller lika med R2
;           BHI                         R1 större än R2
;
; OBS! Innehållen i R1 och R2 förstörs av FCMP!
;
; Ingångsparametrar: Flyttal i R1 och R2
;
; Utgångsparametrar: Statusbitarna C och Z

; Exekveringstiden beror av talens inbördes storlek och om de
; är på normalform vid subrutinanropet eller ej.
; För ungefär lika stora tal på normalform är exekveringstiden
; typiskt mindre än 0,5 millisekund. För extremt olika
```

```
; storleksordning på talen kan tiden bli nära 5 millisekunder
;
;***********************************************************

FCMP:      JSR     SUB
           TST     R1M
           BMI     FCMPL      ;Hopp till FCMPL om R1 < R2
           BEQ     FCMPE      ;Ytterligare tester om R1 = R2
FCMP1:
           CLR     A          ;Annars är R1 > R2
           CMP     #1         ;Z=0
           CLC                ;C=0
           BRA     FCMP2

FCMPL:
           CLR     A          ;R1 < R2
           CMP     #1         ;Z=0
           SEC                ;C=1
           BRA     FCMP2

FCMPE:
           TST     R1L
           BNE     FCMP1      ;Ej lika, måste vara R1 >> R2
           CLR     A          ;Lika, R1=R2
           CMP     #0         ;Sätt Z=1
FCMP2:
           RTS                ;Slut på subrutin FCMP


;*********************** Invertering ********************
;
; Innehållet i R1 inverteras. R1=1/R1 (Använder subrutinen DIV)
;
; Ingångsparameter: Flyttal i R1
;
; Utgångsparameter: Flyttal i R1
;
; Exekveringstiden beror av talens inbördes storlek och om de
; är på normalform vid subrutinanropet eller ej.
; För ungefär lika stora tal på normalform är exekveringstiden
; typiskt mindre än 2 millisekunder. För extremt olika
; storleksordning på talen kan tiden bli nära 4 millisekunder
;
;***********************************************************

INVERT:
           LDA     R1E
           STA     R2E
           LDA     R1M
           STA     R2M
           LDA     R1L
           STA     R2L
           LDA     #$F2
           STA     R1E
           LDA     #$40
           STA     R1M
           CLR     R1L
           JSR     DIV
           RTS                ;Slut på subrutin INVERT
```

```
;********************** Absolutbelopp ******************
;
; Bildar absolutbeloppet av talet i R1, dvs ABS(R1).
;
; Ingångsparametrar: Flyttal i R1
;
; Utgångsparameter: Flyttal i R1
;
; Exekveringstid: Mindre än 38 mikrosekunder
;
;*********************************************************

ABS:
        BRCLR   7,R1M,ABSUT ;Hoppa ut om talet är positivt
        COM     R1M                 ;Annars byt tecken
        NEG     R1L
        BCS     ABSUT
        INC     R1M
ABSUT:
        RTS                         ;Slut på subrutin ABS

;***************** Omvandling till heltal ***************
;
; Omvandlar ett tal i flyttalsrepresentation i R1 till
; ett binärt heltal i 16 bitars tvåkomplementrepresentation
; i R1M (mest signifikant byte) och R1L (minst signifikant byte).
;
; Ingångsparameter: Flyttal i R1
;
; Utgångsparametrar: Heltal i R1M och R1L (R1E blir nollställd)
;
; Exekveringstiden beror på exponentens storlek och på
; om talet från början står på normalform.
; Tiden är typiskt mindre än 1 millisekund.
;
;*********************************************************

FIXP:
        TST     R1E
FIXP1:
        BEQ     FIXP4       ;OK om exponenten = 0
        BMI     FIXP3       ;dividera om negativ exponent
        DEC     R1E         ;annars subtrahera 1 från exp.
        TST     R1M
        BMI     FIXP2       ;Testa mantissans tecken
        LSL     R1L         ;Om 0 multiplicera mantissan med 2
        ROL     R1M
        BPL     FIXP        ;Fortsätt om inget 2-kompl overflow
        LDA     #$7F
        STA     R1M
        LDA     #$FF
        STA     R1L         ;skriv annars $7FFF i R1
        BRA     FIXP4

FIXP2:
        LSL     R1L
        ROL     R1M
        BMI     FIXP        ;Fortsätt om inget 2-kompl overflow
```

```
        LDA     #$80
        STA     R1M
        CLR     R1L         ;skriv annars $8000 i R1
        BRA     FIXP4

FIXP3:
        ASR     R1M         ;Dividera med 2
        ROR     R1L
        INC     R1E         ;Addera 1 till exponenten
        BRA     FIXP1

FIXP4:
        RTS     ;Slut på subrutin FIXP


;************ Hjälprutin för teckenbehandling ***********
;
; Undersöker tecken på mantissorna i R1 och R2. Om både R1
; och R2 är positiva nollställs teckenflaggan SFLAG,
; om både R1 och R2 är negativa blir SFLAG=2, om R1 och R2
; har olika tecken blir SFLAG=1.
;
; Ingångsparametrar: Flyttal i R1 och R2
;
; Utgångsparameter: Teckenflaggan SFLAG
;
; Exekveringstid: Typiskt mindre än 200 mikrosekunder
;
;***********************************************************


SIG:
        LDA     #2
        STA     LPCNT
        CLR     SFLAG       ;Tecken flagga
        LDX     #R1M
SIG1:
        LDA     3,X
        BPL     SIG3
        STA     TEMP1
        LDA     4,X
        COM     TEMP1
        NEG     A
        BCS     SIG2
        INC     TEMP1
SIG2:
        STA     4,X
        LDA     TEMP1
        STA     3,X
        INC     SFLAG
SIG3:
        DEC     X
        DEC     X
        DEC     X
        DEC     LPCNT
        BNE     SIG1
        BRA     SIG5

SIG4:
        BRCLR   0,SFLAG,SIG5 ;Kolla teckenflaggan
```

```
        COM     R1M
        NEG     R1L
        BCS     SIG5
        INC     R1M
SIG5:
        RTS                             ;Slut på subrutin SIG


;************** Hjälprutin för normalpackning *************
;
; Hjälprutin för att normalpacka flyttalen i R1 och R2
;
; Ingångsparametrar: Flyttal i R1 och R2
;
; Utgångsparametrar: Flyttal i R1 och R2
;
; Exekveringstiden beror av hur nära normalpackade talen är
; från början och kan variera från c:a 200 mikrosekunder till
; c:a 2 millisekunder.
;
;**************************************************************

PACK:
        LDA     #2
        STA     COUNT
        LDX     #R1M
PACK1:
        LDA     #15
        STA     LPCNT
PACK2:
        DEC     LPCNT
        BEQ     PACK5
        DEC     5,X
        ASL     4,X
        ROL     3,X
        BCS     PACK3
        BPL     PACK2
        BRA     PACK4

PACK3:
        BMI     PACK2
PACK4:
        ROR     3,X
        ROR     4,X
        INC     5,X
PACK5:
        DEC     X
        DEC     X
        DEC     X
        DEC     COUNT
        BNE     PACK1
        RTS                     ;Slut på subrutin PACK


;********************Slut MATH687.LIB**********************


;********************** POLYNOM.LIB ***********************
; Subrutin för beräkning av ett polynom av typen
;
```

```
;    Y = A0 + A1*X + A2*X^2 + A3*X^3 .....
;
; Gradtal och koefficienter är givna av koefficienttabellen.
; Vid anropet skall IX vara laddat med adressen till tabellen.
; X-värdet skall vid anropet ligga som ett flyttal i R1.
; Är X ett heltal ska det laddas i R1M och R1L och
; exponenten R1E nollställas.
; Resultatet Y återfinns som ett flyttal i R1, men kan om
; så önskas omvandlas till ett heltal med subrutinen FIXP.
; POLYNOM anropar subrutiner i biblioteksmodulen MATH687.LIB.
; MATH.LIB måste därför finnas tillgänglig.
;
; Ingångsparametrar: X-värde i R1M, R1L och R1E (24-bitars flyttal)
;                                Adressen till koefficienttabellen i IX
;
; Utgångsparameter: Y-värde i R1M, R1L och R1E (24-bitars flyttal)
;
; Minnesbehov: 84 byte EPROM, 4 byte RAM
;
; Exekveringstid: Beror bl.a. på gradtalet n. Med n=2 blir
; tiden typiskt c:a 4 millisek, med n=6 c:a 11 millisek.
;************************************************************
;
; Följande variabeldeklaration måste flyttas till RAM


POLYNOM:
        LDA     ,X          ;Hämta gradtalet ur tabellen
        STA     GRAD
        INC     X           ;Peka på högsta koefficienten
        LDA     R1M
        STA     FT
        LDA     R1L
        STA     FT+1
        LDA     R1E
        STA     FT+2        ;X kopierad till FT,+1,+2
        LDA     ,X
        STA     R2M
        INC     X
        LDA     ,X
        STA     R2L
        INC     X
        LDA     ,X
        STA     R2E         ;Första koeff. kopierad till R2
        JSR     MUL         ;Multiplicera X och första koeff.
        INC     X
        DEC     GRAD
        BEQ     POLY2       ;Gå till POLY2 om GRAD=0
POLY1:
        LDA     ,X
        STA     R2M
        INC     X
        LDA     ,X
        STA     R2L
        INC     X
        LDA     ,X
        STA     R2E         ;Koefficient kopierad till R2
        JSR     ADD         ;Addera koeff. till delresultatet
        LDA     FT
        STA     R2M
        LDA     FT+1
```

```
        STA     R2L
        LDA     FT+2
        STA     R2E         ;Variabeln X kopierad till R2
        JSR     MUL         ;Multiplicera delresultatet med X
        INC     X
        DEC     GRAD
        BNE     POLY1       ;Gå till POLY1 om GRAD ej 0
POLY2:
        LDA     ,X
        STA     R2M
        INC     X
        LDA     ,X
        STA     R2L
        INC     X
        LDA     ,X
        STA     R2E         ;Sista koeff. kopierad till R2
        JSR     ADD         ;Addera koeff. till delresultatet
        RTS                 ;Slut på subrutin POLYNOM

;********************* Slut POLYNOM.LIB ******************


;*********************** TRIG.LIB ************************
; Biblioteksmodul för beräkning av trigonometriska funktioner.
; I samtliga fall används flyttalsregistret R1 (dvs R1M, R1L,
; och R1E) för att ange ingångsparametern. Resultatet åter-
; finns i R1. TRIG.LIB använder polynomanpassning för beräk-
; ningarna. Beräkningarna görs med POLYNOM.LIB och MATH687.LIB,
; som båda måste finnas tillgängliga. Koefficienttabellerna
; måste ligga i EPROM, mellan adresserna $80 och $FF. Observera
; att alla tabellerna inte får plats samtidigt!
;
; Biblioteksmodulen omfattar följande åtta subrutiner:
;
;       SINUS           Beräknar sinus för en vinkel
;       COSINUS         Beräknar cosinus för en vinkel
;       TANGENS         Beräknar tangens för en vinkel
;       ARCSIN          Beräknar arcsin för ett argument
;       ARCTAN          Beräknar arctan för ett argument
;       TRISKALA        Skalar vinkel till första kvadranten
;       DEGTORAD        Omvandlar från grader till radianer
;       RADTODEG        Omvandlar från radianer till grader
;
; Ingångsparameter: Vinkel eller argument som 24-bitars flyttal i R1
;
; Utgångsparameter: Funktionsvärde som 24-bitars flyttal i R1
; Minnesbehov: 319 byte EPROM (inklusive tabeller), 2 byte RAM
;
; Registeranvändning: ACC och IX
;
;**********************************************************
;
; Följande variabeldeklaration måste flyttas till RAM


;********************* Arcustangens *********************
; Subrutin för beräkning av arcustangens.
; Argumentet ska vid anropet ligga som ett flyttal i R1.
; Subrutinen ARCTAN returnerar vinkeln (i radianer) som
```

```
; ett flyttal i R1.
; Subrutinen POLYNOM och biblioteksmodulen MATH687.LIB används.
; Koefficienttabellen ATAN måste finnas tillgänglig.
;
; Exekveringstid: Beror på argumentet, mindre än 14 millisek.
;
;*************************************************************

ARCTAN:
         JSR     PACK                    ;Normalform
         LDA     R1M
         STA     TRISIG                  ;Spara tecknet
         JSR     ABS                     ;Gör talet positivt
         LDA     R1E
         CMP     #$F2                    ;Jämför med 1
         BLO     ATANSM                  ;Små vinklar
         LDA     R1M
         CMP     #$40
         BLS     ATANSM
         JSR     INVERT                  ;X = 1/X
         LDX     #ATAN                   ;Peka på arcustangenstabell
         JSR     POLYNOM                 ;Bestäm arctan(1/X)
         LDA     R1M
         STA     R2M
         LDA     R1L
         STA     R2L
         LDA     R1E                     ;Kopiera R1 till R2
         STA     R2E
         LDA     #$64
         STA     R1M
         LDA     #$87
         STA     R1L
         LDA     #$F2                    ;Ladda R1 med pi/2
         STA     R1E
         JSR     SUB                     ;Bestäm pi/2-atan(1/X)
         BRA     ATANKLAR

ATANSM:
         LDX     #ATAN                   ;Peka på arcustangenstabell
         JSR     POLYNOM                 ;Bestäm arctan(X)
ATANKLAR:
         BRCLR   7,TRISIG,ATANUT         ;Kontrollera tecknet
         COM     R1M                     ;Teckenvänd resultatet
         NEG     R1L
         BCS     ATANUT
         INC     R1M
ATANUT:
         RTS                             ;Slut på subrutin ARCTAN

;************************ Triskala *********************
; Subrutin för att skala in vinkel till intervallet
; 0 till +90 grader genom successiv subtraktion
; eller addition av 180 grader.
; Biblioteksmodulen MATH687.LIB används.
;
; Vinkeln skall vid anropet ligga i R1 och vara given i rad.
;
; Efter skalningen returneras den skalade vinkeln i R1.
; TRSIG+1 returnerar 00 för jämnt antal subtraktioner och
; 01 för udda antal.
```

```
; TRISIG returnerar 00 om tecknet på den skalade vinkeln
; var positivt och 01 om det var negativt.
;
; Exekveringstid: Beror på vinkeln, mindre än 2 millisek.
;
;**********************************************************
;

TRISKALA:
        CLR     TRISIG              ;Antalet subtraktioner
TRILOOP:
        LDA     #$64                ;Ladda pi = 6487 F3
        STA     R2M
        LDA     #$87
        STA     R2L
        LDA     #$F3
        STA     R2E
        JSR     PACK                ;Gör vinklarna på normalform
        LDA     R1E
        BRSET   7,R1M,TRINEG        ;Kolla tecknet
        CMP     #$F2                ;Jämför exponenten
        BLO     TRIOK               ;Vinkeln är mindre
        BHI     TRIBIG              ;Vinkeln för stor
        LDA     R1M
        CMP     #$64                ;Jämför med +pi/2
        BLO     TRIOK
        BHI     TRIBIG
        LDA     R1L
        CMP     #$87
        BLS     TRIOK
TRIBIG:
        JSR     SUB                 ;Subtrahera pi
        INC     TRISIG
        BRA     TRILOOP


TRINEG:
        CMP     #$F2                ;Jämför exponenten
        BLO     TRIOK               ;Vinkeln är liten
        BHI     TRISML              ;Vinkeln för liten
        LDA     R1M
        CMP     #$9B                ;Jämför med -pi
        BHI     TRIOK
        BLO     TRISML
        LDA     R1L
        CMP     #$79
        BHI     TRIOK
TRISML:
        JSR     ADD                 ;Addera pi
        INC     TRISIG
        BRA     TRILOOP
TRIOK:
        CLR     TRISIG+1    ;Lägg 00 eller 01
        BRCLR   0,TRISIG,TRIUT      ;i TRSIG+1 beroende på
        BSET    0,TRISIG+1  ;antalet skalningar
TRIUT:
        CLR     TRISIG              ;Vinkelns tecken
        BRCLR   7,R1M,TRIRTS
        BSET    0,TRISIG
        COM     R1M                 ;Teckenvänd vinkeln
        NEG     R1L
```

```
          NEG       R1L
          BCS       TRIRTS
          INC       R1M
TRIRTS:
          RTS                          ;Slut på subrutin TRISKALA

;************************ Deg to Rad *********************
; Subrutin för omvandling av en vinkel uttryckt i grader
; till radianer. Vinkeln skall ligga i R1 i flyttalsformat
; och returneras i R1.
;
; Biblioteksmodulen MATH687.LIB används.
;
; Exekveringstid: Typiskt mindre än 1,5 millisekunder
;
;**********************************************************

DEGTORAD:
          LDA       #$F7               ;Ladda 57.296 i R2
          STA       R2E
          LDA       #$72
          STA       R2M
          LDA       #$97
          STA       R2L
          JSR       DIV
          RTS                          ;Slut på subrutin DEGTORAD

;********************** Rad to Deg **********************
; Subrutin för omvandling av en vinkel uttryckt i radianer
; till grader. Vinkeln skall ligga i R1 i flyttalsformat och
; returneras i R1.
;
; Biblioteksmodulen MATH687.LIB används.
;
; Exekveringstid: Typiskt mindre än 1,5 millisekunder
;
;**********************************************************

RADTODEG:
          LDA       #$F7               ;Ladda 57.296 i R2
          STA       R2E
          LDA       #$72
          STA       R2M
          LDA       #$97
          STA       R2L
          JSR       MUL
          RTS                          ;Slut på subrutin RADTODEG

;********************** Slut TRIG.LIB ********************
SIST:     RTS

          END

;********************** Slut DEKL28 **********************
;
```

Code for Processor 2 - PWM

```
;************************* DEKL28 ***************************

PORTA       EQU     0                   ;Port A Data Register
PORTB       EQU     1                   ;Port B Data Register
PORTC       EQU     2                   ;Port C Data Register

DDRA        EQU     4                   ;Port A Data Direction Register
DDRB        EQU     5                   ;Port B Data Direction Register
DDRC        EQU     6                   ;Port C Data Direction Register

TDR         EQU     8                   ;Timer Data Register
TCR         EQU     9                   ;Timer Control Register

MOR         EQU     $784                ;Mask Option Register
TIRVEK      EQU     $7F8                ;Timer Interrupt Vector
EXTVEK      EQU     $7FA                ;External Interrupt Vector
SWIVEK      EQU     $7FC                ;Software Interrupt Vector
RESVEK      EQU     $7FE                ;Reset Vector

RAM         EQU     $10                 ;Minne för variabler

EPROM       EQU     $80                 ;Minne för konstanter och program


            ORG     MOR                 ;Definiera innehållet i MOR
            DB      %00000000   ;Kristalloscillator (MSB=1 ger RC)


            ORG     TIRVEK              ;Definiera timeravbrottsvektorn
            DW      START               ;Adressen anges här

            ORG     EXTVEK              ;Definiera externa avbrottsvektorn
            DW      START               ;Adressen anges här

            ORG     SWIVEK              ;Definiera programavbrottsvektorn
            DW      START               ;Adressen anges här

            ORG     RESVEK              ;Definiera resetvektorn
            DW      START               ;START är adressen till programstarten


MAXUTSLAG EQU       40
MITT                EQU         128

            ORG     RAM                 ;Variabler

;Här läggs programmets variabler, deklarerade med DS-satser
;Hit flyttas också variablerna från biblioteksmodulerna
RAKN:       DS      1

FELM:       DS      1
FELL:       DS      1
FELE:       DS      1
```

```
TMP:        DS      1

INDATA:     DS      1
BCNTR:      DS      1

; VARIABLER FÖR MATH687.LIB
R1M         DS      1                   ;Operand 1 mantissa MSB
R1L         DS      1                   ;Operand 1 mantissa LSB
R1E         DS      1                   ;Operand 1 exponent
R2M         DS      1                   ;Operand 2 mantissa MSB
R2L         DS      1                   ;Operand 2 mantissa LSB
R2E         DS      1                   ;Operand 2 exponent
SIGN1       DS      1
DUMMY1      DS      1
DUMMY2      DS      1

SIGN2       DS      1
COUNT       DS      1                   ;Hjälpvariabler
LPCNT       DS      1
SFLAG       DS      1                   ;Teckenflagga
IXTMP       DS      1                   ;Temporär lagring av IX
TEMP1       DS      1
TEMP2       DS      1

; VARIABLER FÖR POLYNOM.LIB
GRAD        DS      1                   ;Temporär lagring av gradtal
FT          DS      3                   ;Temporär lagring av flyttal

TRISIG      DS      2                           ;Används för teckenbestämning


            ORG     EPROM                       ;Programmets konstanter

;Här läggs programmets konstanter, textsträngar och tabeller

;           FÖRSTÄRKNING
FÖRST:      DB      0
            DB      %00000110       ;FÖRST 6 GGR, URSPR. 48 GGR
            DB      0


;**** Polynomial coefficients for atan ***
;Valid for  0 < X < 1.0
ATAN:       DB      6                           ;Order
            DB      $B8,$2A,$ED  ;6             -3.5076E-02
            DB      $6F,$25,$ED  ;5             5.4272E-02
            DB      $4C,$A6,$EF  ;4             1.4970E-01
            DB      $9B,$4A,$F0  ;3             -3.9341E-01
            DB      $56,$D9,$EB  ;2             1.0602E-02
            DB      $7F,$E9,$F1  ;1             9.9933E-01
            DB      $61,$31,$E0  ;0             5.7932E-06

START:                                          ;Här startar programmet
;Här läggs programmet in

            LDA #%11111111                      ;PORTA UT
            STA DDRA
            LDA #%11111110                      ;B0 IN, RESTEN UT
            STA DDRB
```

```
          LDA #%11110000              ;PORTC IN (K-JUSTERING)
          STA DDRC

          LDA #%00000000
          STA PORTA

          JSR SIST
          JSR SETTIMER

MANUELL:        BSET 1,PORTB                        ; SÄNDARE AV
                BCLR 2,PORTB                        ;INGET RIKTIGT VÄRDE
UT
                BRCLR 0,PORTB,MANUELL

AUTOMATISK:     JSR KOMPASS       ;LÄGGER KOMPASSVÄRDE I AKT
                LDX #AKTM               ;KOPIERA VÄRDET TILL REFERENS
                LDA #1
                JSR PUTR
                LDX #REFM
                LDA #1
                JSR GETR
                BCLR 1,PORTB                        ;SÄNDARE PÅ


REGLERA:        JSR KOMPASS             ;LÄGGER KOMPASSVÄRDE I AKT
                JSR JUSTERA       ;JUSTERAR FÖR HOPP 0 TILL 360 GRADER

                LDX #REFM               ;REF I R1
                LDA #1
                JSR PUTR

                LDX #AKTM               ;AKT I R2
                LDA #2
                JSR PUTR
                JSR SUB                             ;FELET LIGGER NU I R1
                LDX #FELM
                LDA #1
                JSR GETR                ;FELET LIGGER NU I FEL

                LDX #FÖRST      ;RÄKNA UT FÖRSTÄRKNING
                LDA #1
                JSR PUTR

                LDA #0         ;MULTIPLICERA MED DIP-SWICHARNA
                STA R2M
                STA R2E
                LDA PORTC
                AND #$0F
                STA R2L
                JSR MUL                             ;FÖRST FINNS NU I R1

                LDX #FELM
                LDA #2
                JSR PUTR
                JSR MUL                             ;FEL * K LIGGER NU I R1
                LDA #1
                JSR NEGERA              ;- FEL * K LIGGER NU I R1

                ;BEGRÄNSNING AV UTSIGNAL
```

```
                LDX #FELM
                LDA #1
                JSR GETR                ;UTSIGN LIGGER NU I FEL

BEGR:           LDA #0                  ;JÄMFÖR MED MAXVÄRDET
                STA R2E
                STA R2M
                LDA #MAXUTSLAG
                STA R2L
                JSR FCMP
                BLS BEGR2               ;OM EJ FÖR STORT SÅ FORTSÄTT
                LDA #0          ;ANNARS BEGRÄNSA
                STA FELE
                STA FELM
                LDA #MAXUTSLAG
                STA FELL
                BRA SYMM
BEGR2:          LDX #FELM               ;HÄMTA UTSIGN
                LDA #1
                JSR PUTR
                LDA #0          ;JÄMFÖR MED MINVÄRDET
                STA R2E
                LDA #$FF
                STA R2M
                SUB #MAXUTSLAG
                ADD #1
                STA R2L
                JSR FCMP
                BHS SYMM                ;OM EJ FÖR STORT GÅ VIDARE
                LDA #0                  ;ANNARS LÄGG IN MINVÄRDE
                STA FELE
                LDA #$FF
                STA FELM
                LDA #$FF
                SUB #MAXUTSLAG
                ADD #1
                STA FELL

SYMM:           LDX #FELM
                LDA #1
                JSR PUTR

                LDA #0
                STA R2M
                STA R2E
                LDA #MITT
                STA R2L
        JSR ADD                         ;R1 ÄR NU SYMMETRISK RUNT MITT
                JSR FIXP                ;OMVANDLA TILL HELTAL

                LDA R1L
                BCLR 2,PORTB                    ;VÄRDET SKA ÄNDRAS
                STA PORTA               ;LÄGG UT UTSIGNAL
                BSET 2,PORTB                    ;VÄRDE OK

                BRSET 0,PORTB,REG
                JMP MANUELL
REG:            JMP REGLERA
```

```
; **************** HÄR BÖRJAR ALLA UNDERPROCEDURER *****************

        ;TAR TVÅKOMPLEMENT PÅ R1 ELLER R2
NEGERA:                 CMP #2
                        BEQ NEG3
                        COM R1M             ;TECKENVÄND R1
                        NEG R1L
                        BCS NEG2
                        INC R1M
NEG2:                   RTS
NEG3:                   COM R2M             ;TECKENVÄND R2
                        NEG R2L
                        BCS NEG4
                        INC R2M
NEG4:                   RTS


JUSTERA:    ; FIXA HOPPET MELLAN 0 OCH 360 GRADER
                        LDX #AKTM           ;LÄGG AKT I R1
                        LDA #1
                        JSR PUTR
                        LDX #REFM           ;LÄGG REF I R2
                        LDA #2
                        JSR PUTR
                        JSR FCMP            ;JÄMFÖR AKT OCH REF
                        BLS  REFSTÖRST
                        AKTSTÖRST;  LDX #AKTM        ;LÄGG AKT I R1
                        LDA #1
                        JSR PUTR
                        LDX #REFM           ;LÄGG REF I R2
                        LDA #2
                        JSR PUTR
                                  ;STÖRSTA TALET, DVS AKT, LIGGER I R1
                        JSR SUB                      ;SKILLNADEN MELAN
TALEN I R1
                        LDA #$64            ;LÄGG PI I R2
                        STA R2M
                        LDA #$87
                        STA R2L
                        LDA #$F3
                        STA R2E
                        JSR FCMP            ;SKILJER SIG AKT OCH REF MINDRE ÄN
180GRADER?
                        BLS SLUT            ;SÅ SKA INGET GÖRAS

                        LDX #AKTM           ;LÄGG AKT I R1
                        LDA #1
                        JSR PUTR
                        LDA #$64            ;LÄGG 2*PI I R2
                        STA R2M
                        LDA #$87
                        STA R2L
                        LDA #$F4
                        STA R2E        ;MINSKA AKT MED 2*PI
                        JSR SUB        ;LÄGG NYTT VÄRDE I AKT
                        LDX #AKTM
                        LDA #1
```

```
                JSR GETR
                RTS
REFSTÖRST:      LDX #REFM               ;LÄGG REF I R1
                LDA #1
                JSR PUTR
                LDX #AKTM               ;LÄGG AKT I R2
                LDA #2
                JSR PUTR
                                        ;STÖRSTA TALET LIGGER NU I R1
                JSR SUB                 ;SKILLNADEN MELLAN TALEN I R1

                LDA #$64                ;LÄGG PI I R2
                STA R2M
                LDA #$87
                STA R2L
                LDA #$F3
                STA R2E
                JSR FCMP     ;SKILJER SIG AKT OCH REF MER ÄN 180GRADER?

                BLS SLUT                ;INGET SKA GÖRAS
                LDX #AKTM               ;ÖKA AKT MED 2*PI
                LDA #1
                JSR PUTR
                LDA #$64                ;LÄGG 2*PI I R2
                STA R2M
                LDA #$87
                STA R2L
                LDA #$F4
                STA R2E
                JSR ADD                 ;LÄGG NYTT VÄRDE I AKT
                LDX #AKTM
                LDA #1
                JSR GETR
SLUT:           RTS                     ; AKT OCH REF SKILJER SIG NU ÅT MED
                                        MINDRE ÄN 180 GRADER


PUTR:           ;KOPIERAR VARIABEL VARS ADRESS ÄR I X TILL R1/R2
                ;ACK=1 OM TILL R1, ACK=2 OM TILL R2
                ;INDEXREG INNEHÅLLER ADRESS TILL ANDRA VARIABELN
                CMP #1
                BNE PUTR2
PUTR1:          LDA ,X
                STA R1M
                LDA 1,X
                STA R1L
                LDA 2,X
                STA R1E
                RTS
PUTR2:          LDA ,X
                STA R2M
                LDA 1,X
                STA R2L
                LDA 2,X
                STA R2E
                RTS

GETR:           ;KOPIERAR R1/R2 TILL VARIABEL VARS ADRESS ÄR I X
                ;ACK=1 OM FRÅN R1, ACK=2 OM TILL R2
                ;INDEXREG INNEHÅLLER ADRESS TILL ANDRA VARIABELN
```

```
                CMP #1
                BNE GETR2
GETR1:          LDA R1M
                STA ,X
                LDA R1L
                STA 1,X
                LDA R1E
                STA 2,X
                RTS
GETR2:          LDA R2M
                STA ,X
                LDA R2L
                STA 1,X
                LDA R2E
                STA 2,X
        RTS


KOMPASS:        ;LÄS IN OCH ANPASSA INDATA
                JSR WAITFIRST
                JSR SERIN               ;SEMIKOLON IN
                CMP #$3B
                BNE KOMPASS                     ;OM EJ SEMIKOLON BÖRJA
OM
                LDX #R1M
                JSR INTAL               ;LÄS IN SINUS I R1
                LDX #R1M
                JSR TRANSNEG                    ;GÖR OM NEGATIVT TAL
TILL TVÅKOMPLEMENT

                JSR SERIN               ;CHECKSUMMA

                LDX #R2M
                JSR INTAL               ;LÄS IN COS
                LDX #R2M
                JSR TRANSNEG


                LDA #1
                LDX #FELM
                JSR GETR                ;LAGRA SIN I FEL

                LDA #2
                LDX #AKTM
                JSR GETR                ;LAGRA COS I AKT

                LDA R2M                 ;UNDVIK DIVISION MED 0
                CMP #0
                BNE DIVIDERA
                LDA R2L
                CMP #0
                BNE DIVIDERA

                ;OM SIN >=0 SÅ LÄGG IN PI/2 I R1,
                ;ANNARS LÄGG IN -PI/2
                LDA #$64                ; R1= PI/2
                STA R1M
                LDA #$87
                STA R1L
                LDA #$B9
```

```
            STA R1E
            BRA SPARA

            LDA FELM        ;COS = 0, ÄR VINKELN +90 ELLER -90?
            LSL A
            BCC MINUS                 ;SIN NEG, VINKELN -90
            BRA SPARA
MINUS:      LDA #1                    ;KOPIERA PI/2 TILL R1
            LDX #R2M
            JSR GETR
            LDA #0
            STA R1M
            STA R1L
            STA R1E
            JSR SUB                   ;-PI/2 NU I R1

            BRA SPARA

DIVIDERA:

            JSR DIV                   ;SIN/COS I R1
            SR ARCTAN                 ;VINKELN I RADIANER I R1



            LDA AKTM                  ;LÄS IN COS (MSBYTE)
            LSL A                     ;SKIFTA VÄNSTER
            BCC SPARA
            LDA FELM                  ;LADDA SIN  (MSBYTE)
            LSL A
            BCS VINKEL2

            LDA #$64                  ;ÖKA VINKELN MED PI
            STA R2M
            LDA #$87
            STA R2L
            LDA #$F3
            STA R2E
            JSR ADD
            BRA SPARA

VINKEL2:    DA #$64                   ;MINSKA VINKELN MED PI
            STA R2M
            LDA #$87
            STA R2L
            LDA #$F3
            STA R2E
            JSR SUB
            BRA SPARA

SPARA:      DA #1
            DX #AKTM
            SR GETR                   ;KOPIERA VINKELN TILL AKT
            TS



            ;ÖVERSÄTTER TILL TVÅKOMPL OM SIGNX ÄR 1
```

```
TRANSNEG:       DA 6,X                  ;SIGN1 RESP SIGN2
                CMP #1      ;ÄR TALET NEGATIVT?
                BNE TRANSNEG2
                A #%11111111
                B 1,X
                D #1
                A 1,X
                LDA #%11111111
                ADC #0
                SUB ,X
                STA ,X
TRANSNEG2:      RTS


INTAL:          JSR SERIN               ;DATA1
                JSR ASCII2HEX                       ;ÖVERSÄTT ASCII-TECKEN
TILL HEX
                STA ,X                              ;LÄGG IN I R1M RESP R2M
                AND #%00001000
                BEQ INTAL2              ;POSITIVT TAL
                LDA #1
                STA 6,X                 ;ETTA I SIGN1 RESP SIGN2 OM
NEGATIVT
                LDA ,X                  ;MASKA BORT TECKENBIT
                AND #%00000111
                STA ,X
                BRA INTAL3
INTAL2:         LDA #0                  ;NOLLA OM POSITIVT
                STA 6,X
INTAL3:         INC X
                JSR SERIN               ;DATA2
                JSR ASCII2HEX
                LSL A
                LSL A
                LSL A
                LSL A
                STA TMP         ;IN I TMP
                JSR SERIN               ;DATA3
                JSR ASCII2HEX
                ORA TMP
                STA ,X                              ;IN I R1L ELLER R2L
                LDA #0
                STA 1,X                             ;NOLLA I EXPONENTEN
                RTS


ASCII2HEX:      ;INDATA I ACK
                AND #%01111111                      ;SORTERA BORT EV
TECKENBIT
                CMP #$40
                BHI ASC2                ;A TILL F
                SUB #$30
                RTS
ASC2:           SUB #$41
                ADD #10
                RTS                                 ;HEXADECIMALT TAL
FINNS NU I ACK
```

```
WAITFIRST:  BIH WAITFIRST
                LDA #10
                STA BCNTR
AGAIN:          JSR TIME
                BIH WAITFIRST                   ;IF HIGH START OVER
                DEC BCNTR
                BNE AGAIN
                RTS


                ;LÄS IN DATA OCH INVERTERA
SERIN:          BIL SERIN               ;VÄNTA PÅ STARTBIT
                JSR HALFTIME
                BIL SERIN       ;FALSK STARTBIT
                LDA #8
                STA BCNTR
SERIN1:
                JSR TIME
                BIH SERIN2
                SEC
                BRA SERIN3
SERIN2:         NOP
                NOP
                CLC
SERIN3:
                ROR INDATA
                LDA INDATA
                DEC BCNTR
                BNE SERIN1


                RTS

SETTIMER:       LDA #%01001100
                STA TCR
                RTS

TIME:           LDA #156
                STA TDR
                BCLR 7,TCR
TIME2:          BRCLR 7,TCR,TIME2               ;WAIT FOR TIMER TO
FINISH
                RTS

HALFTIME:       LDA #78
                STA TDR
                BCLR 7,TCR
HALFTIME2:      BRCLR 7,TCR,HALFTIME2           ;WAIT FOR TIMER TO
FINISH
                RTS


;Efter programmet läggs biblioteksmodulerna in.
;Variabler flyttas till RAM,
;Konstanter och tabeller flyttas till början av EPROM.


;********************** MATH687.LIB ********************
;
```

```
; Biblioteksmodul för flyttalsräkning i MC68705
;
; Modulen omfattar följande tio subrutiner:
;
;        ADD        Addition
;        SUB        Subtraktion
;        MUL        Multiplikation
;        DIV        Division
;        FCMP       Jämförelse
;        INVERT     Invertering
;        ABS        Absolutbelopp
;        FIXP       Omvandling till heltal
;        PACK       Hjälprutin för normalpackning
;        SIG        Hjälprutin för teckenhantering
;

;************************* Addition ******************
;
; Flyttalsaddition. De två flyttalen som skall adderas
; skall innan subrutinanropet ligga som flyttal i R1 och R2,
; och summan hamnar som ett flyttal i R1, dvs R1 = R2 + R1
;
; Ingångsparametrar: Flyttal i R1 och R2
;
; Utgångsparameter: Flyttal i R1
;
; Exekveringstiden beror av talens inbördes storlek och
; om de är på normalform vid subrutinanropet eller ej.
; För ungefär lika stora tal på normalform är exekveringstiden
; typiskt mindre än 0,5 millisekund. För extremt olika
; storleksordning på talen kan tiden bli nära 5 millisekunder
;
;***********************************************************

ADD:
        STX     IXTMP     ;Spara indexregistret
        JSR     PACK      ;packa talen på normalform
        LDX     #R1M
        LDA     2,X
        SUB     5,X
        BEQ     ADD2
        BMI     ADD1
        NEG     A
        INC     X
        INC     X
        INC     X
ADD1:
        ASR     ,X
        ROR     1,X
        INC     2,X
        INC     A
        BNE     ADD1
ADD2:
        LDX     #R1M
        LDA     1,X
        ADD     4,X
        STA     1,X
        LDA     ,X
        BPL     ADD3
```

```
        TST     3,X
        BPL     ADD4
        ADC     3,X
        BMI     ADD5
        BRA     ADD7
ADD3:
        TST     3,X
        BMI     ADD4
        ADC     3,X
        BPL     ADD5
        BRA     ADD7
ADD4:
        ADC     3,X
ADD5:
        STA     ,X
ADD6:
        LDX     IXTMP
        RTS                     ;Slut på subrutin ADD

ADD7:
        ROR     A
        STA     ,X
        ROR     1,X
        INC     2,X
        BRA     ADD6

;************************ Subtraktion *****************
;
; Subrutin för flyttalssubtraktionen R1 = R2 - R1
;
; Ingångsparametrar: Flyttal i R1 och R2
;
; Utgångsparameter: Flyttal i R1
;
; Exekveringstiden beror av talens inbördes storlek och om de
; är på normalform vid subrutinanropet eller ej.
; För ungefär lika stora tal på normalform är exekveringstiden
; typiskt mindre än 0,5 millisekund. För extremt olika
; storleksordning på talen kan tiden bli nära 5 millisekunder
;
;**********************************************************

SUB:
        COM     R2M             ;Teckenvänd R2
        NEG     R2L
        BCS     SUB1
        INC     R2M
SUB1:
        BRA     ADD             ;Använd addition

;********************** Multiplikation ****************
;
;  Utför flyttalsmultiplikationen R1 = R1*R2
;
; Ingångsparametrar: Flyttal i R1 och R2
;
; Utgångsparameter: Flyttal i R1
;
```

```
; Exekveringstiden beror av talens inbördes storlek och om de
; är på normalform vid subrutinanropet eller ej.
; För ungefär lika stora tal på normalform är exekveringstiden
; typiskt c:a 1 millisekund. För extremt olika
; storleksordning på talen kan tiden bli nära 2 millisekunder
;
;*************************************************************

MUL:
        STX     IXTMP       ;Spara indexregistret
        TST     R1M         ;kontrollera om ena talet är noll
        BNE     MUL1
        TST     R1L
        BEQ     MUL5        ;i så fall utförs ingen multiplikation
MUL1:
        TST     R2M         ;Kontrollera om andra talet är noll
        BNE     MUL2
        TST     R2L
        BEQ     MUL5        ;i så fall utförs ingen multiplikation.
MUL2:
        JSR     SIG         ;Bilda beloppet och lagra tecknet
        JSR     PACK        ;R1 och R2 på normalform
        CLR     TEMP1
        CLR     TEMP2
        LDA     #$10
        STA     COUNT
MUL3:
        LSR     R2M
        ROR     R2L
        BCC     MUL4
        LDA     TEMP2
        ADD     R1L
        STA     TEMP2
        LDA     TEMP1
        ADC     R1M
        STA     TEMP1
MUL4:
        ROR     TEMP1
        ROR     TEMP2
        DEC     COUNT
        BNE     MUL3
        LDA     TEMP1
        STA     R1M
        LDA     TEMP2
        STA     R1L
        JSR     SIG4
        LDA     R1E
        ADD     R2E
        ADD     #$10
        STA     R1E
        BRA     MUL6

MUL5:
        CLR     R1M
        CLR     R1L
        CLR     R1E
MUL6:
        LDX     IXTMP
        RTS                 ;Slut på subrutin MUL
```

```
;************************ Division *******************
;
; Subrutin för flyttalsdivisionen R1 = R1 / R2
;
; Ingångsparametrar: Flyttal i R1 och R2
;
; Utgångsparameter: Flyttal i R1

; Exekveringstiden beror av talens inbördes storlek och om de
; är på normalform vid subrutinanropet eller ej.
; För ungefär lika stora tal på normalform är exekveringstiden
; typiskt mindre än 2 millisekunder. För extremt olika
; storleksordning på talen kan tiden bli nära 4 millisekunder
;
;**********************************************************

DIV:
        STX     IXTMP       ;Spara IX
        TST     R1M
        BNE     DIV1
        TST     R1L
        BEQ     DIV6        ;Gör inget om täljaren = 0
DIV1:
        TST     R2M
        BNE     DIV2
        TST     R2L
        BEQ     DIV7        ;Overflow om nämn = 0
DIV2:
        JSR     SIG
        JSR     PACK
        LDA     R1E
        ADD     #$F2
        SUB     R2E
        STA     R1E
        LDA     R1L
        STA     TEMP2
        LDA     R1M
        STA     TEMP1
        LDA     #$0F
        STA     COUNT
DIV3:
        LDA     TEMP2
        SUB     R2L
        STA     TEMP2
        LDA     TEMP1
        SBC     R2M
        STA     TEMP1
        BPL     DIV4
        LDA     TEMP2
        ADD     R2L
        STA     TEMP2
        LDA     TEMP1
        ADC     R2M
        STA     TEMP1
        CLC
        BRA     DIV5

DIV4:
        SEC
```

```
DIV5:
        ROL     R1L
        ROL     R1M
        LSR     R2M
        ROR     R2L
        DEC     COUNT
        BNE     DIV3
        LDA     #%01111111
        AND     R1M
        STA     R1M
        JSR     SIG4
DIV6:
        LDX     IXTMP
        RTS                     ;Slut på subrutin DIV

DIV7:
        LDA     #$FF
        STA     R1L
        LDA     #$7F
        STA     R1M             ;Vid overflow blir
        STA     R1E             ;resultatet 7FFF * 2^7F
        BRA     DIV6


;************************ Jämför **********************
;
; Flyttalsrutin för jämförelse av två tal på flyttalsformat
;
; Jämförelsen utförs som subtraktionen R1 - R2 och ställer
; statusflaggorna C och Z på samma sätt som instruktionen
; CMP gör vid heltalsjämförelser:
;
;                       R1 = R2: Z=1
;                       R1 < R2: Z=0, C=1
;                       R1 > R2: Z=0, C=0
;
; Följande villkorliga hoppinstruktioner kan användas efter
; subrutinen FCMP:
;
;       BEQ ger hopp om  R1 lika med R2
;       BNE              R1 skilt från R2
;       BLO              R1 mindre än R2
;       BLS              R1 mindre än eller lika med R2
;       BHS              R1 större än eller lika med R2
;       BHI              R1 större än R2
;
; OBS! Innehållen i R1 och R2 förstörs av FCMP!
;
; Ingångsparametrar: Flyttal i R1 och R2
;
; Utgångsparametrar: Statusbitarna C och Z

; Exekveringstiden beror av talens inbördes storlek och om de
; är på normalform vid subrutinanropet eller ej.
; För ungefär lika stora tal på normalform är exekveringstiden
; typiskt mindre än 0,5 millisekund. För extremt olika
; storleksordning på talen kan tiden bli nära 5 millisekunder
;
;**********************************************************
```

```
FCMP:       JSR     SUB
            TST     R1M
            BMI     FCMPL       ;Hopp till FCMPL om R1 < R2
            BEQ     FCMPE       ;Ytterligare tester om R1 = R2
FCMP1:
            CLR     A           ;Annars är R1 > R2
            CMP     #1          ;Z=0
            CLC                 ;C=0
            BRA     FCMP2

FCMPL:
            CLR     A           ;R1 < R2
            CMP     #1          ;Z=0
            SEC                 ;C=1
            BRA     FCMP2

FCMPE:
            TST     R1L
            BNE     FCMP1       ;Ej lika, måste vara R1 >> R2
            CLR     A           ;Lika, R1=R2
            CMP     #0          ;Sätt Z=1
FCMP2:
            RTS                 ;Slut på subrutin FCMP


;*********************** Invertering ********************
;
; Innehållet i R1 inverteras. R1=1/R1 (Använder subrutinen DIV)
;
; Ingångsparameter: Flyttal i R1
;
; Utgångsparameter: Flyttal i R1
;
; Exekveringstiden beror av talens inbördes storlek och om de
; är på normalform vid subrutinanropet eller ej.
; För ungefär lika stora tal på normalform är exekveringstiden
; typiskt mindre än 2 millisekunder. För extremt olika
; storleksordning på talen kan tiden bli nära 4 millisekunder
;
;*********************************************************

INVERT:
            LDA     R1E
            STA     R2E
            LDA     R1M
            STA     R2M
            LDA     R1L
            STA     R2L
            LDA     #$F2
            STA     R1E
            LDA     #$40
            STA     R1M
            CLR     R1L
            JSR     DIV
            RTS                 ;Slut på subrutin INVERT


;********************** Absolutbelopp ******************
;
; Bildar absolutbeloppet av talet i R1, dvs ABS(R1).
```

```
;
; Ingångsparametrar: Flyttal i R1
;
; Utgångsparameter: Flyttal i R1
;
; Exekveringstid: Mindre än 38 mikrosekunder
;
;*************************************************************

ABS:
        BRCLR   7,R1M,ABSUT ;Hoppa ut om talet är positivt
        COM     R1M                 ;Annars byt tecken
        NEG     R1L
        BCS     ABSUT
        INC     R1M
ABSUT:
        RTS                         ;Slut på subrutin ABS


;***************** Omvandling till heltal ***************
;
; Omvandlar ett tal i flyttalsrepresentation i R1 till
; ett binärt heltal i 16 bitars tvåkomplementrepresentation
; i R1M (mest signifikant byte) och R1L (minst signifikant byte).
;
; Ingångsparameter: Flyttal i R1
;
; Utgångsparametrar: Heltal i R1M och R1L (R1E blir nollställd)
;
; Exekveringstiden beror på exponentens storlek och på
; om talet från början står på normalform.
; Tiden är typiskt mindre än 1 millisekund.
;
;*************************************************************

FIXP:
        TST     R1E
FIXP1:
        BEQ     FIXP4       ;OK om exponenten = 0
        BMI     FIXP3       ;dividera om negativ exponent
        DEC     R1E         ;annars subtrahera 1 från exp.
        TST     R1M
        BMI     FIXP2       ;Testa mantissans tecken
        LSL     R1L         ;Om 0 multiplicera mantissan med 2
        ROL     R1M
        BPL     FIXP        ;Fortsätt om inget 2-kompl overflow
        LDA     #$7F
        STA     R1M
        LDA     #$FF
        STA     R1L         ;skriv annars $7FFF i R1
        BRA     FIXP4

FIXP2:
        LSL     R1L
        ROL     R1M
        BMI     FIXP        ;Fortsätt om inget 2-kompl overflow
        LDA     #$80
        STA     R1M
        CLR     R1L         ;skriv annars $8000 i R1
        BRA     FIXP4
```

```
FIXP3:
            ASR         R1M         ;Dividera med 2
            ROR         R1L
            INC         R1E         ;Addera 1 till exponenten
            BRA         FIXP1

FIXP4:
            RTS         ;Slut på subrutin FIXP

;*********** Hjälprutin för teckenbehandling ***********
;
; Undersöker tecken på mantissorna i R1 och R2. Om både R1
; och R2 är positiva nollställs teckenflaggan SFLAG,
; om både R1 och R2 är negativa blir SFLAG=2, om R1 och R2
; har olika tecken blir SFLAG=1.
;
; Ingångsparametrar: Flyttal i R1 och R2
;
; Utgångsparameter: Teckenflaggan SFLAG
;
; Exekveringstid: Typiskt mindre än 200 mikrosekunder
;
;*************************************************************


SIG:
            LDA         #2
            STA         LPCNT
            CLR         SFLAG       ;Tecken flagga
            LDX         #R1M
SIG1:
            LDA         3,X
            BPL         SIG3
            STA         TEMP1
            LDA         4,X
            COM         TEMP1
            NEG         A
            BCS         SIG2
            INC         TEMP1
SIG2:
            STA         4,X
            LDA         TEMP1
            STA         3,X
            INC         SFLAG
SIG3:
            DEC         X
            DEC         X
            DEC         X
            DEC         LPCNT
            BNE         SIG1
            BRA         SIG5

SIG4:
            BRCLR       0,SFLAG,SIG5 ;Kolla teckenflaggan
            COM         R1M
            NEG         R1L
            BCS         SIG5
            INC         R1M
```

```
SIG5:
        RTS                             ;Slut på subrutin SIG

;************** Hjälprutin för normalpackning *************
;
; Hjälprutin för att normalpacka flyttalen i R1 och R2
;
; Ingångsparametrar: Flyttal i R1 och R2
;
; Utgångsparametrar: Flyttal i R1 och R2
;
; Exekveringstiden beror av hur nära normalpackade talen är
; från början och kan variera från c:a 200 mikrosekunder till
; c:a 2 millisekunder.
;
;*********************************************************

PACK:
        LDA     #2
        STA     COUNT
        LDX     #R1M
PACK1:
        LDA     #15
        STA     LPCNT
PACK2:
        DEC     LPCNT
        BEQ     PACK5
        DEC     5,X
        ASL     4,X
        ROL     3,X
        BCS     PACK3
        BPL     PACK2
        BRA     PACK4

PACK3:
        BMI     PACK2
PACK4:
        ROR     3,X
        ROR     4,X
        INC     5,X
PACK5:
        DEC     X
        DEC     X
        DEC     X
        DEC     COUNT
        BNE     PACK1
        RTS                     ;Slut på subrutin PACK

;********************Slut MATH687.LIB*********************


;********************** POLYNOM.LIB **********************
; Subrutin för beräkning av ett polynom av typen
;
;    Y = A0 + A1*X + A2*X^2 + A3*X^3 .....
;
; Gradtal och koefficienter är givna av koefficienttabellen.
; Vid anropet skall IX vara laddat med adressen till tabellen.
```

```
; X-värdet skall vid anropet ligga som ett flyttal i R1.
; Är X ett heltal ska det laddas i R1M och R1L och
; exponenten R1E nollställas.
; Resultatet Y återfinns som ett flyttal i R1, men kan om
; så önskas omvandlas till ett heltal med subrutinen FIXP.
; POLYNOM anropar subrutiner i biblioteksmodulen MATH687.LIB.
; MATH.LIB måste därför finnas tillgänglig.
;
; Ingångsparametrar: X-värde i R1M, R1L och R1E (24-bitars flyttal)
;                    Adressen till koefficienttabellen i IX
;
; Utgångsparameter: Y-värde i R1M, R1L och R1E (24-bitars flyttal)
;
; Minnesbehov: 84 byte EPROM, 4 byte RAM
;
; Exekveringstid: Beror bl.a. på gradtalet n. Med n=2 blir
; tiden typiskt c:a 4 millisek, med n=6 c:a 11 millisek.
;*********************************************************
; Följande variabeldeklaration måste flyttas till RAM


POLYNOM:
        LDA     ,X          ;Hämta gradtalet ur tabellen
        STA     GRAD
        INC     X           ;Peka på högsta koefficienten
        LDA     R1M
        STA     FT
        LDA     R1L
        STA     FT+1
        LDA     R1E
        STA     FT+2        ;X kopierad till FT,+1,+2
        LDA     ,X
        STA     R2M
        INC     X
        LDA     ,X
        STA     R2L
        INC     X
        LDA     ,X
        STA     R2E         ;Första koeff. kopierad till R2
        JSR     MUL         ;Multiplicera X och första koeff.
        INC     X
        DEC     GRAD
        BEQ     POLY2       ;Gå till POLY2 om GRAD=0
POLY1:
        LDA     ,X
        STA     R2M
        INC     X
        LDA     ,X
        STA     R2L
        INC     X
        LDA     ,X
        STA     R2E         ;Koefficient kopierad till R2
        JSR     ADD         ;Addera koeff. till delresultatet
        LDA     FT
        STA     R2M
        LDA     FT+1
        STA     R2L
        LDA     FT+2
        STA     R2E         ;Variabeln X kopierad till R2
        JSR     MUL         ;Multiplicera delresultatet med X
```

```
          INC       X
          DEC       GRAD
          BNE       POLY1        ;Gå till POLY1 om GRAD ej 0
POLY2:
          LDA       ,X
          STA       R2M
          INC       X
          LDA       ,X
          STA       R2L
          INC       X
          LDA       ,X
          STA       R2E          ;Sista koeff. kopierad till R2
          JSR       ADD          ;Addera koeff. till delresultatet
          RTS                    ;Slut på subrutin POLYNOM

;********************** Slut POLYNOM.LIB *******************


;************************* TRIG.LIB **************************
; Biblioteksmodul för beräkning av trigonometriska funktioner.
; I samtliga fall används flyttalsregistret R1 (dvs R1M, R1L,
; och R1E) för att ange ingångsparametern. Resultatet åter-
; finns i R1. TRIG.LIB använder polynomanpassning för beräk-
; ningarna. Beräkningarna görs med POLYNOM.LIB och MATH687.LIB,
; som båda måste finnas tillgängliga. Koefficienttabellerna
; måste ligga i EPROM, mellan adresserna $80 och $FF. Observera
; att alla tabellerna inte får plats samtidigt!
;
; Biblioteksmodulen omfattar följande åtta subrutiner:
;
;         SINUS                  Beräknar sinus för en vinkel
;         COSINUS                Beräknar cosinus för en vinkel
;         TANGENS                Beräknar tangens för en vinkel
;         ARCSIN                 Beräknar arcsin för ett argument
;         ARCTAN                 Beräknar arctan för ett argument
;         TRISKALA   Skalar vinkel till första kvadranten
;         DEGTORAD   Omvandlar från grader till radianer
;         RADTODEG   Omvandlar från radianer till grader
;
; Ingångsparameter: Vinkel eller argument som 24-bitars flyttal i R1
;
; Utgångsparameter: Funktionsvärde som 24-bitars flyttal i R1
; Minnesbehov: 319 byte EPROM (inklusive tabeller), 2 byte RAM
;
; Registeranvändning: ACC och IX
;
;*********************************************************
;
; Följande variabeldeklaration måste flyttas till RAM



;********************** Arcustangens *********************
; Subrutin för beräkning av arcustangens.
; Argumentet ska vid anropet ligga som ett flyttal i R1.
; Subrutinen ARCTAN returnerar vinkeln (i radianer) som
; ett flyttal i R1.
; Subrutinen POLYNOM och biblioteksmodulen MATH687.LIB används.
; Koefficienttabellen ATAN måste finnas tillgänglig.
;
```

```
; Exekveringstid: Beror på argumentet, mindre än 14 millisek.
;
;**********************************************************

ARCTAN:
        JSR     PACK            ;Normalform
        LDA     R1M
        STA     TRISIG          ;Spara tecknet
        JSR     ABS             ;Gör talet positivt
        LDA     R1E
        CMP     #$F2            ;Jämför med 1
        BLO     ATANSM          ;Små vinklar
        LDA     R1M
        CMP     #$40
        BLS     ATANSM
        JSR     INVERT          ;X = 1/X
        LDX     #ATAN           ;Peka på arcustangenstabell
        JSR     POLYNOM         ;Bestäm arctan(1/X)
        LDA     R1M
        STA     R2M
        LDA     R1L
        STA     R2L
        LDA     R1E             ;Kopiera R1 till R2
        STA     R2E
        LDA     #$64
        STA     R1M
        LDA     #$87
        STA     R1L
        LDA     #$F2            ;Ladda R1 med pi/2
        STA     R1E
        JSR     SUB             ;Bestäm pi/2-atan(1/X)
        BRA     ATANKLAR

ATANSM:
        LDX     #ATAN           ;Peka på arcustangenstabell
        JSR     POLYNOM         ;Bestäm arctan(X)
ATANKLAR:
        BRCLR   7,TRISIG,ATANUT ;Kontrollera tecknet
        COM     R1M             ;Teckenvänd resultatet
        NEG     R1L
        BCS     ATANUT
        INC     R1M
ATANUT:
        RTS                     ;Slut på subrutin ARCTAN

;************************ Triskala **********************
; Subrutin för att skala in vinkel till intervallet
; 0 till +90 grader genom successiv subtraktion
; eller addition av 180 grader.
; Biblioteksmodulen MATH687.LIB används.
;
; Vinkeln skall vid anropet ligga i R1 och vara given i rad.
;
; Efter skalningen returneras den skalade vinkeln i R1.
; TRSIG+1 returnerar 00 för jämnt antal subtraktioner och
; 01 för udda antal.
; TRISIG returnerar 00 om tecknet på den skalade vinkeln
; var positivt och 01 om det var negativt.
;
; Exekveringstid: Beror på vinkeln, mindre än 2 millisek.
```

```
;
;********************************************************
;
TRISKALA:
        CLR     TRISIG              ;Antalet subtraktioner
TRILOOP:
        LDA     #$64                ;Ladda pi = 6487 F3
        STA     R2M
        LDA     #$87
        STA     R2L
        LDA     #$F3
        STA     R2E
        JSR     PACK                ;Gör vinklarna på normalform
        LDA     R1E
        BRSET   7,R1M,TRINEG        ;Kolla tecknet
        CMP     #$F2                ;Jämför exponenten
        BLO     TRIOK               ;Vinkeln är mindre
        BHI     TRIBIG              ;Vinkeln för stor
        LDA     R1M
        CMP     #$64                ;Jämför med +pi/2
        BLO     TRIOK
        BHI     TRIBIG
        LDA     R1L
        CMP     #$87
        BLS     TRIOK
TRIBIG:
        JSR     SUB                 ;Subtrahera pi
        INC     TRISIG
        BRA     TRILOOP


TRINEG:
        CMP     #$F2                ;Jämför exponenten
        BLO     TRIOK               ;Vinkeln är liten
        BHI     TRISML              ;Vinkeln för liten
        LDA     R1M
        CMP     #$9B                ;Jämför med -pi
        BHI     TRIOK
        BLO     TRISML
        LDA     R1L
        CMP     #$79
        BHI     TRIOK
TRISML:
        JSR     ADD                 ;Addera pi
        INC     TRISIG
        BRA     TRILOOP
TRIOK:
        CLR     TRISIG+1    ;Lägg 00 eller 01
        BRCLR   0,TRISIG,TRIUT      ;i TRSIG+1 beroende på
        BSET    0,TRISIG+1  ;antalet skalningar
TRIUT:
        CLR     TRISIG              ;Vinkelns tecken
        BRCLR   7,R1M,TRIRTS
        BSET    0,TRISIG
        COM     R1M                 ;Teckenvänd vinkeln
        NEG     R1L
        BCS     TRIRTS
        INC     R1M
TRIRTS:
        RTS                         ;Slut på subrutin TRISKALA
```

```
;************************ Deg to Rad **********************
; Subrutin för omvandling av en vinkel uttryckt i grader
; till radianer. Vinkeln skall ligga i R1 i flyttalsformat
; och returneras i R1.
;
; Biblioteksmodulen MATH687.LIB används.
;
; Exekveringstid: Typiskt mindre än 1,5 millisekunder
;
;**********************************************************

DEGTORAD:
        LDA     #$F7        ;Ladda 57.296 i R2
        STA     R2E
        LDA     #$72
        STA     R2M
        LDA     #$97
        STA     R2L
        JSR     DIV
        RTS                 ;Slut på subrutin DEGTORAD

;********************** Rad to Deg **********************
; Subrutin för omvandling av en vinkel uttryckt i radianer
; till grader. Vinkeln skall ligga i R1 i flyttalsformat och
; returneras i R1.
;
; Biblioteksmodulen MATH687.LIB används.
;
; Exekveringstid: Typiskt mindre än 1,5 millisekunder
;
;**********************************************************

RADTODEG:
        LDA     #$F7        ;Ladda 57.296 i R2
        STA     R2E
        LDA     #$72
        STA     R2M
        LDA     #$97
        STA     R2L
        JSR     MUL
        RTS                 ;Slut på subrutin RADTODEG

;********************** Slut TRIG.LIB *********************
SIST:   RTS

        END

;********************** Slut DEKL28 ***********************
;
```

Matlab code

Process2.m

```
function [xylog,compass] = process2(micx,micy,a,filename1,bfilename);
% function ret = processfile(micx,micy);
% Process a log-file from a boat-experiment
% Calculates position of the boat
% Parameters:
%   micx: vektor containing x-coord for microphones
%   micy: vektor containing x-coord for microphones

eval(['load ' filename1]);
mypos=eval(bfilename);

mypos = mypos(:,1:size(mypos,2)-1); %ska ändras till size(...)-1

for i=1:size(mypos,2)
  mypos(:,i)=filtfilt([1-a],[1 -a],mypos(:,i));
end;

% calculate position
tmppos = [micx(size(micx,1))-micx(1) micy(2)-micy(1)];

for i=1:size(mypos,1)
  sum = 0;
  for j = 1:size(mypos,2)
    sum = sum + abs(mypos(i,j));
  end;
  posvekt = mypos(i,:);
  micxvekt = micx;
  micyvekt = micy;

  % Only minimize if valid values( = values <>0 )
  if sum ~= 0

    [tmppos,error] = position(tmppos,micxvekt,micyvekt,posvekt);
    if error == 0
      xylog = [xylog ; tmppos];
    end;
    fprintf(1,'%c','*');
  end;
end;

% return compass-value
compass = eval(bfilename);
compass = compass(:,size(compass,2));

system_dependent(12,'off');
figure;
plot(xylog(:,1),xylog(:,2));
axis([min(micx) max(micx) min(micy) max(micy)]);
title(['Position ' filename1]);
zoom on;
drawnow;
figure;
```

```
plot(1:length(compass),compass);
title(['Compass ' filename1]);
zoom on;
drawnow;
```

Position.m

```
% function [pos,error] = position(inpos,micx,micy,d);
% försöker att finnas båtens position genom att
% minimera felavvikelsen
% Anropar boatpos:
%
% function ret=boatpos(v,micx,micy,d);
% räknar ut felfunk. map båtens position
% indata:
% ret = boatpos(v)
% v(1) = x, v(2) = y
% d = längddifferanserna (vektor om 4)
% micx,micy : vektorer som innehåller micarnas läge

function [pos,error] = position(inpos,micx,micy,d);

options(1)=0;
options(2)=1.e-3;
options(3)=1.e-3;
options(14)= 1000;

s1ok = 0;
while (s1ok == 0) & (size(d,2) >= 2)
  s1ok = 0;
  for j = 2:size(d,2)
    if abs(d(j)) < 15
      s1ok = 1;
    end;
  end;
  if s1ok == 0   %s1 not valid, have s2 as referense instead
    for j=size(d,2):-1:2
      d(j) = d(j) - d(2);
    end;
    d = d(2:size(d,2));
    micx = micx(2:size(micx,2));
    micy = micy(2:size(micy,2));
  end;
end;

if(size(d,2)) >= 3
  pos = fmins('boatpos',inpos,options,[],micx,micy,d);
  error = 0;
else
  error = 1;
end;
```

Boatpos.m

```
function ret=boatpos(v,micx,micy,d);
% räknar ut felfunk. map båtens position
% indata:
% ret = boatpos(v)
% v(1) = x, v(2) = y
```

```
% d = längddifferanserna (vektor om 4)
% micx,micy : vektorer som innehåller micarnas läge


 ret = 0;
 s1 = sqrt((v(1)-micx(1))^2 + (v(2)-micy(1))^2);
 for i=1:length(d)
   ret = ret + ((micx(i)-v(1))^2 + (micy(i)-v(2))^2 - (s1+d(i))^2 )^2;
 end;
% Undvik att negativa värden kan ge ett bra (litet) ret-värde
 if v(1)<0
   ret = ret + 100*v(1)^2+1;
 end
 if v(2)<0
   ret = ret + 100*v(2)^2+1;
 end


%ret = ((micx(1)-v(1))^2 + (micy(1)-v(2))^2 - (v(3)+d(1))^2 )^2;
%ret = ret + ((micx(2)-v(1))^2 + (micy(2)-v(2))^2 - (v(3)+d(2))^2 )^2;
%ret = ret + ((micx(3)-v(1))^2 + (micy(3)-v(2))^2 - (v(3)+d(3))^2 )^2;
%ret = ret + ((micx(4)-v(1))^2 + (micy(4)-v(2))^2 - (v(3)+d(4))^2 )^2;
```

Simnon code

The code used for simulation are based on Magnus Akke's Simnon files used for simulations of a real boat. We have changed the controller from a continues to a discrete and replaced some parameters so that it will agree better with our model boat.

dboat2

```
CONTINUOUS SYSTEM dboat2
" Author: Magnus Akke, PDS, Sydkraft, 92-09-02
"
" revised 95-10 with a discrete controller by Andreas Fransson & Staffan Nilsson
"
" Model includes dynamic of rudder servo
" Model to simulate course deviation when a ship with auto pilot using
" magnetic compass is crossing a monopolar HVDC cable.
"
" Reference: Report PDS 9208-XX
"
" Angle conventions according to naval praxis, i.e., North=0 degrees and
" positive direction is clockwise.

INPUT Rp_Ref2
OUTPUT Phi_Ref2 Phi_Com2
STATE V_Gir Phi X_Pos Y_Pos Rp V_Comp Phi_Com Nservo
DER dV_Gir dPhi dX_Pos dY_Pos dRp dV_Comp dPhi_Com dNservo

TIME t

" Description of states
" V_Gir: (Radians / s )Rate of change in direction
" Phi: (Radians) Direction of boat
" X_Pos (m), boat position perpendicular to cable
" Y_Pos (m), boat postion in cable direction
" Rp (radians), rudder position
" IntErr Integral of Heading Error
" Nservo, speed of rudder servo

" Default settings of parameters



Rp_Max:0.436 "(Radians) max rudder position (25 degrees)
dRp_Max:0.035 "(radians/s) max rudder speed (2 deg/s)
N:10 " factor used to reduce rudder change when saturation
T1:0.1 "Time constant for rudder servo
Kservo:2.5

T0:5.0"(s) Time constant from turning circle test, Patrol boat 2.9s, Ferry 24s
a0:-0.40" gain from turning circle test, Patrol boat -0.21, Ferry -0.17
V0:3.1 "(m/s) Design speed used at turning circle test

v: 3.1 " (m/s) speed of boat
Alpha: 0.52 "(Radians), angle from North to direction of current in cable
Be: 0.17 "(gauss), Earths magnetic field; 1 gauss=1e-4 Vs/m2
I:1500 "(A), Cable current
```

```
h:20 "(m), Depth of cable
Phi_Ref:0 "(Radians), reference direction of boat; i.e., desired course
N_Start:0  "Start position of boat in North coordinate
E_Start:0 "Start position of boat in Easth coordinate


" Default initial values of states
V_Gir:0
Phi:1.57079
X_Pos:-300
Y_Pos:0
Rp:0

"******************Regulator removed*************
Rp_Ref = Rp_Ref2

" ****************** Rudder servo ****************************
" with saturation in position and saturation in speed of rudder motor
" input: Rp_Ref; output:Rp (Radians)
" T1 is servo time constant, Kservo is servo gain
" Nservo is servo speed
" Rp_Max is max rudder position in Radians
" dRp_Max is maximal rudder speed in Radians/s

" Allow large speeds to get correct initial conditions
MaxN=IF t < 20 THEN 100*dRp_Max ELSE dRP_MAX
MinN=IF t < 20 THEN -100*dRp_Max ELSE -dRP_MAX

In_Servo=MIN(MaxN,MAX(MinN,Kservo*(Rp_Ref-Rp)))
dNservo=(In_Servo-Nservo)/T1

dRp=IF Saturate < 0.5 THEN Nservo ELSE NServo/N

"Check for saturation in position
Saturate=MinusSat+PlusSat
MinusSat=IF (Rp < -Rp_Max) AND (Nservo<0) THEN 1 ELSE 0
PlusSat=IF  (Rp  > Rp_Max) AND (Nservo>0) THEN 1 ELSE 0
"************************ END Rudder servo ********************************

" Boat Dynamics
dV_Gir=(a*Rp-V_Gir)/T_Boat
dPhi=V_Gir
T_Boat=T0*V0/V "(s) Time constant  from rudder position to yaw rate
a=a0*V/V0 "Gain from rudder position to yaw rate


" Calculation of speed (m/s) ortogonal to cable (X)
" and speed in cable direction (Y)
Vy=V*cos(Phi-Alpha)
Vx=V*sin(Phi-Alpha)

" Position of boat relative cable
dX_Pos=Vx
dY_Pos=Vy

" Calculation of compass deviation from Xpos;
" The direction of Bx (in Gauss) is perpendicular to the cable
" B_S2N is magnetic field with direction from south to north
" B_W2E is magnetic field with direction from west to east
" Phi_Dev (Radians) is the resulting compass deviation
```

```
Bx=2*I*h*0.001/(h*h+X_Pos*X_Pos)
B_S2N= -Bx*SIN(Alpha)
B_W2E=Bx*COS(Alpha)
Phi_Dev=ATAN2(B_W2E,B_S2N+Be)


" ******** MAGNET COMPASS with dynamics ** revised 92-10-15
" Phi_Com (Radians) is heading of ship according to magnet compass
" Phi_Flux (Radians) is angle between heading and disturbed magnetic north
" second order model for compass dynamics
" G(s)= w0^2 / s^2 + 2*w0*z*s + w0^2
" input: Phi_Flux            state: V_Comp, Yaw of compass
" state and (output): Phi_Com
" z is  relative damping of compass
" T_Swing is periodic compass-half period

Phi_Flux=Phi-Phi_Dev
"  Phi_Flux=IF t < 10 THEN 0 ELSE 0.1 " for step response
dV_Comp= -2*z*w0*V_Comp+w0*w0*(Phi_Flux-Phi_Com)
dPhi_Com=V_Comp

z:0.5 " relative damping of compass
" calculation of w0 from T_swing and z
T_Swing:13 "(s); periodic compass-half period
z_angle=ATAN(SQRT(1-z*z)/z)
Wd=(2*pi-z_angle)/T_Swing
" W0=Wd/SQRT(1-z*z) " Not used
W0:0.46
"------------------------- end magnet compass ------------------




"*** Variables for positions of ship, reference course, and cable
N_Pos=Y_Pos*COS(Alpha)-X_Pos*SIN(Alpha) "North position of boat
E_Pos=X_Pos*COS(Alpha)+Y_Pos*SIN(Alpha) "East position of boat
N_Ref=N_Start+v*t*COS(Phi_Ref) "Reference position of boat in North coordinate
E_Ref=E_Start+v*t*SIN(Phi_Ref) "Reference position of boat in East coordinate
E_Cable=v*t*SIN(Alpha) "Only used to get coordinates for cable
N_Cable=v*t*COS(Alpha)

" Help variables to plot angles in degrees; variable extension P for Plot
Rad2Deg=180/pi
PhiFluxP=Phi_Flux*Rad2Deg
PhiP=Phi*Rad2Deg
Phi_DevP=Phi_Dev*Rad2Deg
Phi_ComP=Phi_Com*Rad2Deg
Phi_RefP=Phi_Ref*Rad2Deg
RpP=-Rp*Rad2Deg
Rp_RefP=Rp_Ref*Rad2Deg
dRpP=-dRp*Rad2Deg
V_GirP=V_Gir*Rad2Deg
pi:3.1415927

Phi_ref2 = Phi_Ref
Phi_Com2 = Phi_Com

END
```

danalys1

```
MACRO danalys1
" Made for paper 93-11-08 by Magnus Akke
" Revised by Andreas & Staffan 95-10 for model boat
" default case with boat model boat2.t with rudder servo dynamic
syst dboat2 regul boatsys
hcopy on
" Parameter values
par K:-0.40  "Gain auto pilot

par Rp_Max: 0.436 "(Rad) = 25 degrees; Max rudder position
par dRp_Max:0.035 "(Rad/s)= 2 degrees/s; Max rudder speed
par T1:0.1 "(s) Time constant for rudder servo
par Kservo:2.5
par N:10 "factor to reduce rudder speed when saturated

par V:3.1 "(m/s) Speed of boat =(6 knots)
par T0:5.0 "(s) Time constant from turning circle test, Patrol boat 2.9s
par a0:-0.40" gain from turning circle test, Patrol boat -0.21
par V0:3.1 "(m/s) Design speed used at turning circle test

par z:0.50 "relative damping of magnetic compass
par w0:0.46 "natural frequency for magnetic compass

par Alpha: 0.5236 "(Rad) (30 deg) Angle from North to cable direction
par Be:0.17 "(Gauss), horisontal component of earths magnetic field
par I:1500 "(A), cable current
par h:20 " (m), depth of cable

store N_pos E_Pos N_Ref E_Ref N_Cable E_Cable X_pos Vx

" Initial values on states
init V_Gir:0


" 1:st Simulation
init X_Pos:200
init Y_Pos:450
simu 0 1 /slask
par N_Start:N_Pos
par E_Start:E_Pos
par Phi_Ref:0.1745 "(0.1745;-0.1745,-0.5236;-1.0472 Rad)=(10,-10;-30;-60 deg)
init Phi:0.1745 "(Rad), identical to Phi_Ref
simu 0 250 0.2 /file1a 2
" Get correct start positions for North East coordinates
" calculated from given initial values of states X_Pos and Y_pos

" 2:nd simulation
init X_Pos:200
init Y_Pos:325
simu 0 1 /slask
par N_Start:N_Pos
par E_Start:E_Pos
par Phi_Ref:-0.1745 "
init Phi:-0.1745 "(Rad), identical to Phi_Ref
```

```
simu 0 800 0.2 /file1b 2

" 3:rd simulation
init X_Pos:200
init Y_Pos:225
simu 0 1 /slask
par N_Start:N_Pos
par E_Start:E_Pos
par Phi_Ref:-0.5236
init Phi:-0.5236
simu 0 300 0.2 /file1c 2

" 4:th simulation
init X_Pos:250
init Y_Pos:175
simu 0 1 /slask
par N_Start:N_Pos
par E_Start:E_Pos
par Phi_Ref:-1.0472
init Phi:-1.0472
simu 0 250 0.2 /file1d 2

" Plot of results
" newplot
split 1 1
axes h -200 800 v 0 1000
" 1st case
area 1 1
"switch color red
show N_Pos (E_Pos) /file1a
area 1 1
"switch color blue
show N_Ref (E_Ref) /file1a
"
" 2nd case
area 1 1
"switch color red
show N_Pos (E_Pos) /file1b
area 1 1
"switch color blue
show N_Ref (E_Ref) /file1b
"
" 3rd case
area 1 1
"switch color red
show N_Pos (E_Pos) /file1c
area 1 1
"switch color blue
show N_Ref (E_Ref) /file1c
"
" 4th case
area 1 1
"switch color red
show N_Pos (E_Pos) /file1d
area 1 1
"switch color blue
show N_Ref (E_Ref) /file1d
"
area 1 1
"switch color black
```

```
show N_Cable (E_Cable) /file1a
text ' Ship position, Ref, Cable in N-S, E-W system'
hcopy hpgl:/plot1a
"
" ** Phase plane plot Vx vs X
newplot
split 1 1
axes h -100 100 v -4 2
" 1st case
area 1 1
"switch color red
show Vx (X_pos) /file1a
"
" 2nd case
area 1 1
"switch color blue
show Vx (X_pos) /file1b
"
" 3rd case
area 1 1
"switch color green
show Vx (X_pos) /file1c
"
" 4th case
area 1 1
"switch color black
show Vx (X_pos) /file1d
text ' Phase plane plo, V_x (m/s) vs X(m)'
hcopy hpgl:/plot1b


END
```

Regul

```
DISCRETE SYSTEM REGUL
" Version:    1.0
" Abstract:
" Description:
" Revision:   1.0
" Author:     Andreas Fransson & Staffan Nilsson
" Created:    1995-11-13

" Inputs and outputs:
INPUT Phi_Ref Phi_Com
OUTPUT Rp_Ref

" States and time variables:

TIME t
TSAMP ts

" Initializations:
K:-0.40  " Gain of auto pilot
samp_tid: 1 "sampeltid 1 s
" Equations:

" Auto pilot with PI regulator
```

```
Err=Phi_Ref-Phi_Com  " (Rad) Error in heading course
Rp_Ref=K*Err "Output from regulator

ts = t + samp_tid

" Parameter values:

END
```

Boatsys

```
CONNECTING SYSTEM boatsys
" Version:    1.0
" Abstract:
" Description:
" Revision:   1.0
" Author:     AF & SN
" Created:    1995-11-13

" Time, if needed:
TIME t

" Connections:
Phi_Ref[regul] = Phi_Ref2[dboat2]
Phi_Com[regul] = Phi_Com2[dboat2]
Rp_Ref2[dboat2] = Rp_Ref[regul]


END
```