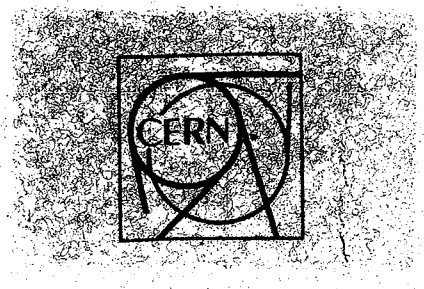
MAN-MACHINE COMMUNICATION

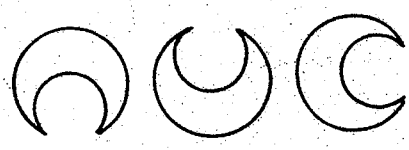
Peter Petersen

11. January 1983

Abstract

This reoport describes a Man-to-Machine Communication module which together with a STAC can take care of all operator inputs from the touch-screen, tracker balls and mechanical buttons. The MMC module can also contain a G64 card which could be a GPIB driver but many other G64 cards could be used. The soft-ware services the input devices and makes the results accessible from the CAMAC bus. NODAL functions for the Man Machine Communication is implemented in the STAC and in the ICC.





AALBORG UNIVERSITETSCENTER

Det teknisk-naturvidenskabelige fakultet

AFGANGSPROJEKT

TITEL: MAN - MACHINE COMMUNICATION

UDARBEJDET AF:

Peter Petersen

HOVEDVEJLEDER:

Bent Stumpe - CERN

Henning Nielsen - AUC

SYNOPSIS:

Rapporten der er udarbejdet under et ophold på CERN i Geneve, omhandler designet af et "Man-Machine Communication Module". Modulet der vil blive brugt i de fremtidige Touch-Terminaler, der vil indgå i mange kontrol systemer på CERN, indeholder interface kredsløb til en 16 punkts touch-skærm, 8 encoder indgange beregnet på tracker-balls og drejeknapper, 16 mekaniske kontakter og interface til et G64 kort. Kredsløbet er opbygget på et CAMAC kort som forbindes med en STAC, en 68000 baseret processor. STAC-en holder kontrol med touch-skærmen, encoderne og kontakterne og gør deres status tilgængelig for CAMAC og NODAL funktioner i STAC og ICC.

OPLAGSTAL: 12

SIDEANTAL: 157

PROJEKTPERIODE: 1 juni 83 - 11 januar 84

INDHOLDET AF NÆRVÆRENDE RAPPORT ER FRIT TILGÆNGELIGT, MEN OFFENTLIGGØRELSE MED KILDEANGIVELSE MÅ KUN SKE EFTER AFTALE MED FORFATTEREN

Preface

This report is a result of a collaboration between AUC and CERN. The project is carried out at CERN under CERN's technical student program.

Chapter 1 is an introduction to the control room technique used at CERN. The content of this chapter is based on the references 1, 2, 3, and 4, the chapter ends with the "Problemformulering" which means defining the problem. Chapter 2 describes the touch-screen detector, first a brief description of the different touch-screen types is given and the methods which have been used to interface the capacitive touch-screen. Two new methods for a touch-screen detector is described, the first one based on chargetime detector and the second one based on a voltage divider circuit, of those the last one has been implemented and proven workable. Chapter 4 describes the encoder interface here a software counter solution is chosen. Chapter 5 describes a G64 interface to the MC68000 frontbus, this interface is mainly done through the use of PAL's which also takes care of the control logic described in chapter 6. The diagram of all the hard-ware is shown in appendix 18. In appendix 7 there is a brief introduction to the 68000 hard-ware, the full documentation of the 68000 can be found in litt 7. Chapter 7 describes the CAMAC communication between the STAC and the ICC and a brief introduction to the CAMAC bus is given in appendix 12. Chapter 8 describes the soft-ware for the operator input detectors, the actual program is shown in appendix 14. The program is documentated by a PASCAL like program desription in the comment field and a description of the variables in the header. The structure of the program was changed just before I left CERN therefore there is some errors and those which is found is corrected with a pencil. Chapter 9 describes the Man-Machine functions in the ICC, a brief description of assembler function structure is given and in chapter 9.4 the KNOB function is used as a basis for an explanation of how an assembler function is implemented.

In appendix 20 there is an explanation of some of the abbreviations used in this report.

Acknowledgement

I would like to thank the people from the SPS-AOP group at CERN for their collaboration and their comments. A special thank should go to Mr. Bent Stumpe who made this project possible.

TABLE OF CONTENTS

Chapter 1

CONTROL ROOM TECHNIQUE

Only a decade ago, a control room was just a room filled with remote instrumentation for the machines being controlled. There were a variety of cables, connecting the buttons in the control room to the machine controlled and to read back the results. When processes gets complicated many machines, many adjustments, etc. the control room also gets bigger, the result is that the operator has to move around between all those control panels and make the necessary adjustments. For a visitor looking into such a control room it looks very complicated and it often is.

In a conventional control room, there often is a lot of equipments which are seldom used, and for normal use only a little part of the whole system is used. One part is used in the initialisation, other parts are used in the start-up, in the normal run, and in alarm situations.

It can be very difficult for the operator to overview the whole system and if something has to be adjusted or checked, he has to move around in the control room, and he can only be at one place at a time.

One of the major drawbacks is that when an update of the system is required, a lot of hardware and connections have to be replaced, both in the control room and in the plant. This makes an update very expensive, but what could be worse is the time needed to instore and test the new system.

1.1 THE CONTROL SYSTEM FOR THE SPS

When the controlsystem for the SPS (Super Positron Syncroton) accellerator was designed. It was obvius that the conventional control filosofi couldn't be used. The area was so big that the amount of cables alone, would have done the project too expensive. At that time (1972) computers were getting so reliable, and economical, that it was resonable to make use of them in controlsystems. before that time computers had only been used in the collecting of data and for the calculation of results, a breakdown here would have no catastrophic results.

A whole network of computers were designed, 24 minicomputers were used in the basic design, each with its own particular function and geographical site. A message transfer system, which took care of the communication, between the various computers were build so, that one computer could start the execution of a certain proces in all of the other computers. The keystone of the software system is the NODAL interpreter. This highlevel language makes it possible for the operator to talk to the machine in a relatively clear language. An interpreter is used here because then an order can be executed immidiately after the operator has given it, no compiling, linking, or loading. Ofcourse an interpretive language is slow, but big program pieces which is often used, can then be written in assembler with a NODAL header. In this way complicated

programs can be executed at maximum speed, with a very easy NODAL call. Infact there are a lot of resident assembler subroutines in all of the computers all over the plant. These subroutines can be executed with a call from one of the consoles or from one of the other computers.

The hardware interface to the minicomputers is based on the CAMAC system - an international standard specified by European laboratories, through the ESONE Committee - This makes high-speed transfer (24 bit pr microsec.) possible between the computer and the various input/output devices. Where high speed isn't essential, where there is a long distance between the computer and the machine, and where the quantity makes it economical, the interface is handled in remote crates which contains cheap standard plug-ins. These remote crates is connected to and controlled from CAMAC, through a special serial branch. A CAMAC crate can contain 4 serial branch controllers, and each of these can control a maximum of 32 stations. The system configuration is shown in figure 1.

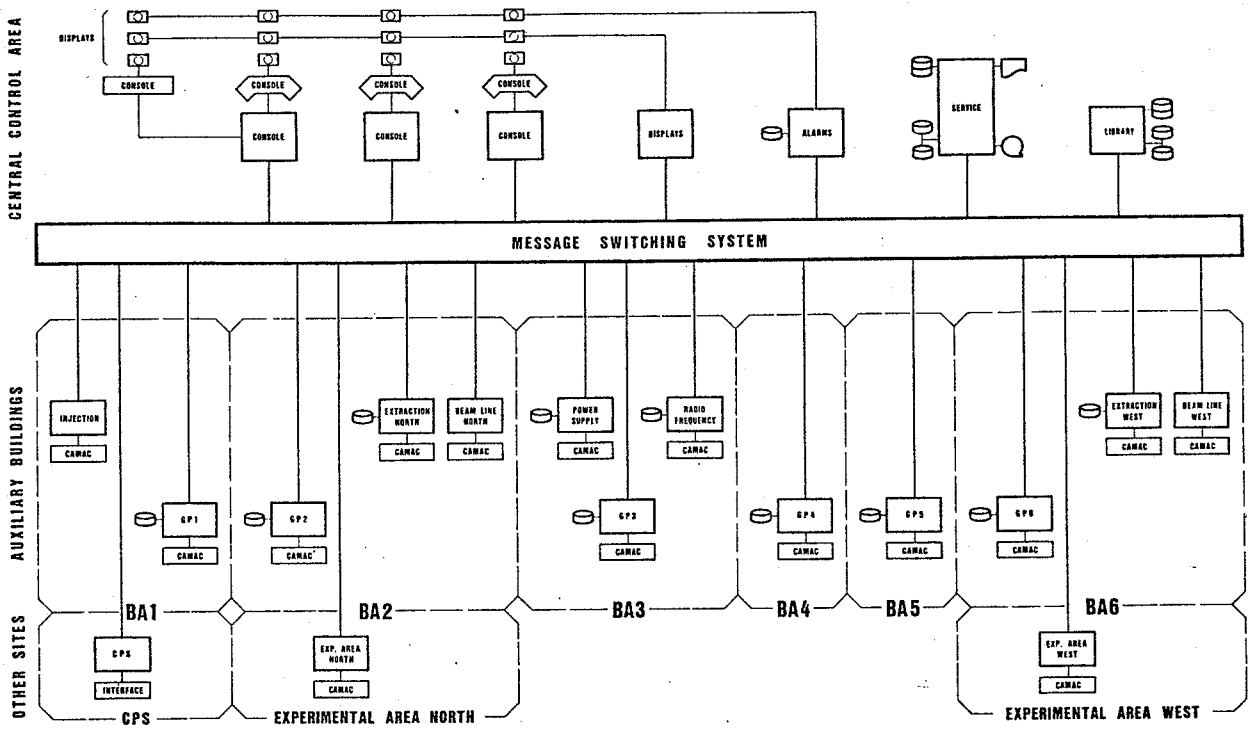


Figure 1: The computer system for the SPS

The control room was designed so, that one person from a single console can control all the parameters of the machine without moving from his seat. The basic idea is to multiplex the machine to the operator, instead of the opposite which was the case in conventional control rooms. This is possible with the use of computers in the interface to the machine. The computer can make a link between a knob on the control console to any device in the system requiring a knob input. Therefore the consol was equiped with a minimum of knobs and buttons. Infact the primary input devices is limmited to:

1. The touch-screen - 16 buttons
2. A few real buttons, for frequently used actions
3. The computer controlled knob
4. The tracker ball

And for the visualization of the information from the system,

5. The video display system

To these are added the secondary input devices:

6. The keyboard
7. The identification card reader

1.1.1 The touch-screen

The touch-screen consist of a transparent glass plate with 16 sensitive touch-buttons. Under the touch-panel, there is a little monitor, on which small pieces of text (legends) can be written under the sensitive areas. A typical set of legends is shown in figure 2. The operator can now select any of the legends, by just touching it with his finger, and the computer will then take action on this, either by putting a new set of legends on the screen, to go deeper in the hierachy, or execute something.

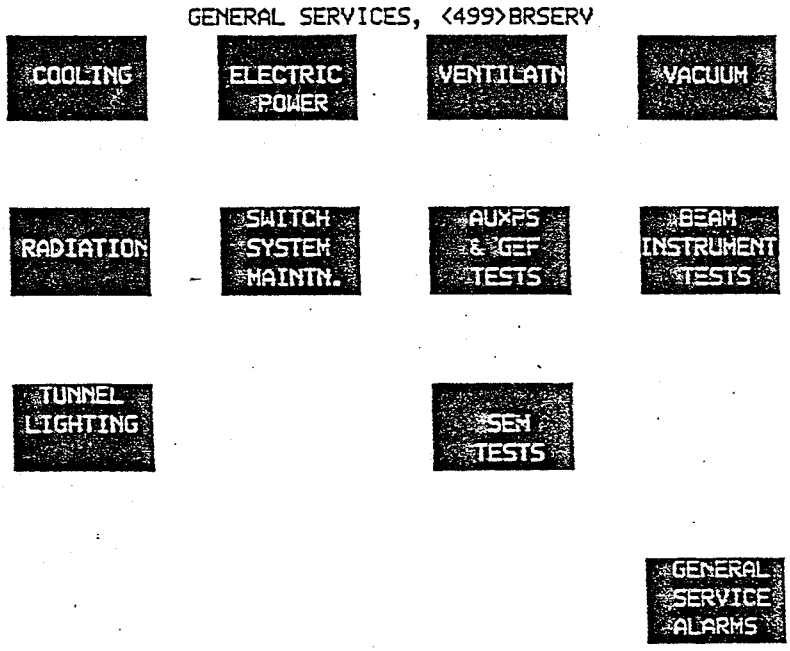


Figure 2: Picture of a touch-screen with legends

The touch-screen is the key to a big hierachical tree, in which you can go from the basis to anywhere with just a few logical decisions. The computer only presents you the valid choises, so the operator is not confused with butttons which have no significans for the operation he has to carry out. If there is some actions which the operator is not allowed to carry out the computer simply will not allow him to. Both safety and speed of the opration is greatly enhanced, since there is no need to use a keyboard for the actual operation, which is normally the case in computer-controlled systems.

1.1.2 The computer-controlled knob

A knop is probably the most used input, when the talk is about adjustment. There is a variety of different knobs: knobs with spring return, multiturn, discrete steps, etc. In the SPS control room there only is one computer controlled knob at each consol. This knob can be programmed into different modes:

1. Programmable resistance against being turned
2. Programmable end-stop
3. Spring return to zero from left or right
4. A switch with exactly 16 positions for a complete turn

The knob itself shows what operational mode it is in.

As seen from figure 3, the computer-controlled knob is pretty complicated, bulky, and expensive, so in future design the knob will probably not be computer controlled, but only one mode will be possible.

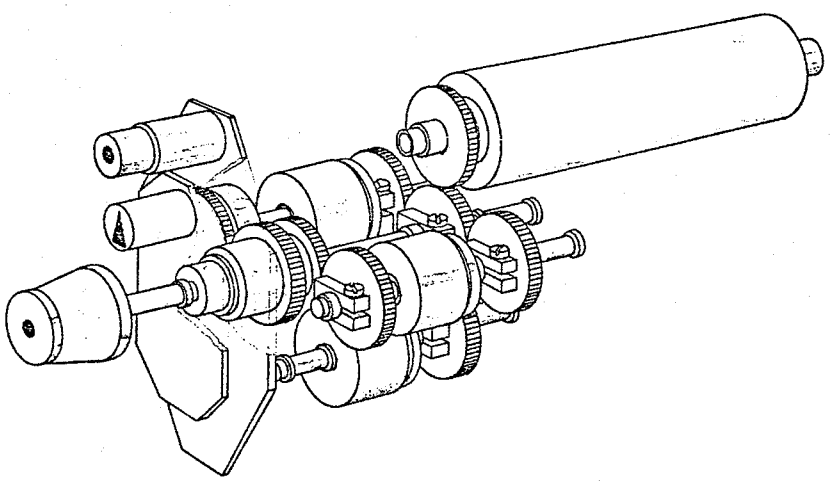


Figure 3: The computer-controlled knob

1.1.3 The tracker ball

The tracker ball is a two-dimensional input devices and is mainly used to move a curser on a TV screen. If the picture on the screen was a diagram over the vacuum pump system, the tracker ball could be used to select one pumpstation, by just moving the cursor to that pumpstation. Further selection could then be done together with the touch-screen, and finaly when one pump had been selected, adjustments could be carried out with the knob.

The tracker ball used in the SPS control room is designed at CERN. The ball is a 10 cm. bowling ball and is very comfortable in use. There are two different versions of tracker balls commercially available, as shown in figure 4. In the normal type the ball is turned, by moving the hand over the ball. The other type is the mouse, here you hold the mouse in your hand, and when you moves the mouse over the table, it is just as if you had the cursor in your hand.

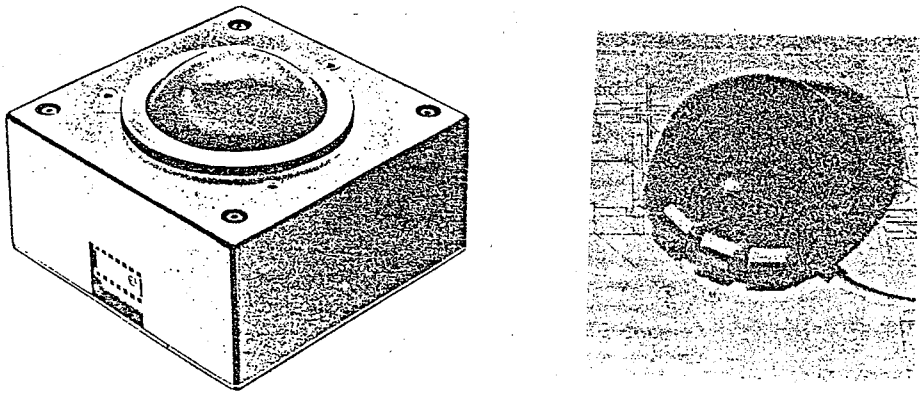


Figure 4: A normal tracker ball The mouse

1.1.4 The video display system

The display system is based on synchronized TV raster pictures. The synchronization allows different pictures to be mixed onto one monitor. This means that an output can be build from: character generators, graphic generators, and cursor generators,

even normal pictures can be used. Any picture can be transmitted to a variety of black-and-white and colour monitors. The standard TV norm (625 lines) is used, which means that pictures can be transmitted to places outside CERN very easily. The use of a standard norm also means that standard monitors can be used. The disadvantage is the lack of high resolution pictures, but this is now provided through a different system.

1.1.5 The keyboard

The keyboard provides the full facilities of NODAL. Which means that the operator can write programs, run programs, and he can execute programs step by step. In the control phase the keyboard is hardly used, here the touch-screen, the knob, and the ball is the main input devices. The primary role of the keyboard is to prepare programs and to test them before they are used in the control phase. The control room is equiped with 5 consoles, eventhough the whole accelerator can be controlled from just one of them. The other consoles can then be used for the writing and testing new programs, and for the education of new operators.

1.1.6 Identification card reader

In all kinds of systems a wrong operation can have serious consequences. To avoid wrong operations the operator has to identify himself to the machine. From the identity card the machine can see which operations the operator is not allowed to carry out. This means that operators without experience will do no harm to the machine.

1.2 THE CONSOLES

The first implementation of this control concept was done in the SPS consoles. Here a NORD minicomputer is connected to each of the consoles. The interconnection is done through CAMAC crates, via special made CAMAC cards. There are cards for the touch-screen, hard buttons, trackerball, and the knob. The layout of the consoles is shown in figure 5, and a picture of the control room with the four consoles is shown in figure 6.

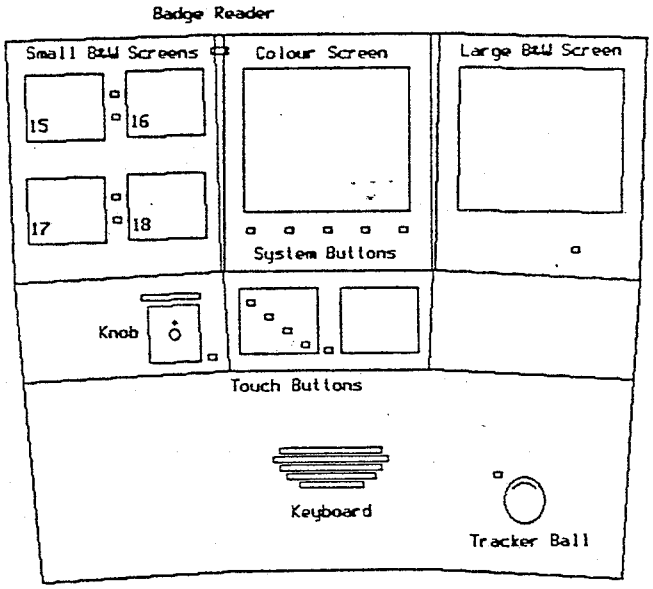


Figure 5: The layout of the SPS consoles

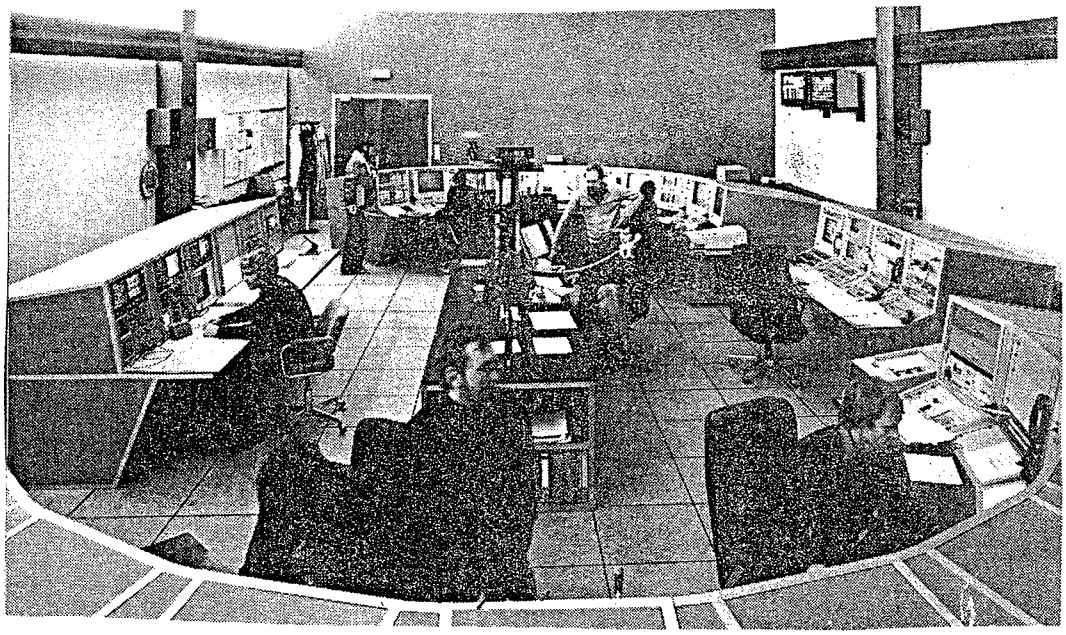


Figure 6: Picture of the SPS control room

The second implementation was the Touch-terminal, this is a low cost solution with the processor integrated in the crate controller, a so called Independent Crate Controller.

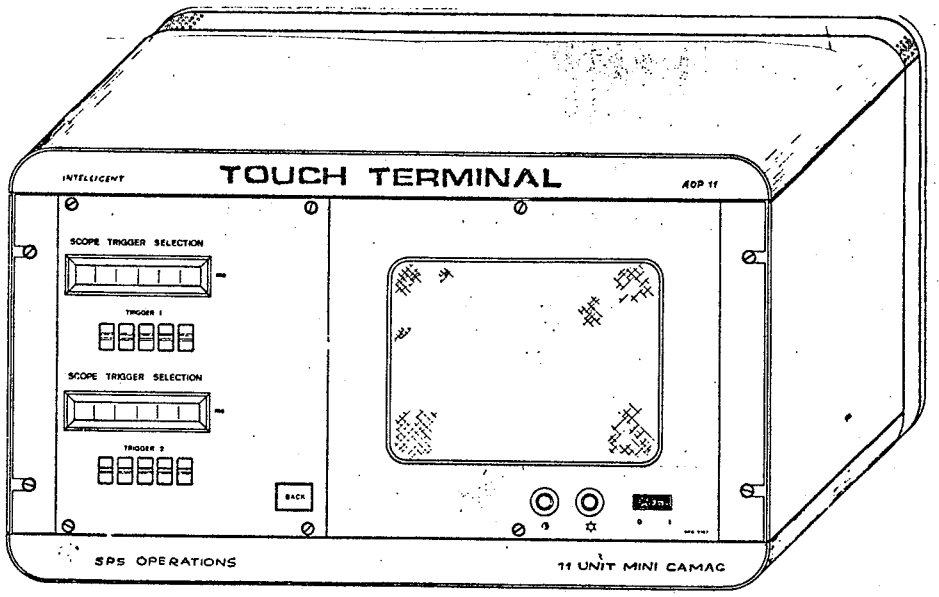


Figure 7: The Touch-terminal in the CAMAC version

In the version from around 1976 the ICC was based on the 16 bit PACE processor from National, one of the first 16 bit microprocessors. In 1980 a new ICC was introduced, this uses the Motorola 68000, a 32 bit microprocessor. This set'-up, with a 68000 based ICC, is used and will be used in controlsystems in various fields at CERN. A picture of the Touch-terminal is shown in figure 7. The box contains a half size CAMAC crate and a touch-screen with monitor. Together with a normal terminal and a colormonitor this is a complete control system similar to the SPS consoles. The control language is NODAL in all of the cases, which means that application programs can be developed on the Touch-terminal and easily transfered to the SPS consoles. The CAMAC modules used in the Touch-terminal is the same as those used in the SPS consoles.

The next version of the Touch-terminal, will be in a VME configuration. The VME bus is a multiprocessor bus well suited for the 68000 microprocessor, and is together with CAMAC and G64 a CERN standard bus. CERN has newly developed a CAMAC branch driver for VME, which means, that a processor in a VME enviroment can acces and control CAMAC crates and modules.

In tabel 1 a list of the different consoltypes, which is supported by the SPS-ACC group, is shown.

Processor	Assembler	Enviroment	Introduced
NORD	NORD asm.	CAMAC-NORD	1973
PACE	PACE asm.	CAMAC	1976
68000	68000 asm.	CAMAC	1980
68000	68000 asm.	VME	1984/85

TABLE 1

Overwiew over the different consol types

The NORD will remain the consol computer for the SPS control room for quite some years, there is no actual plan for their replacement.

The PACE ICC will be replaced with the 68000 ICC whenever an update should be required.

The 68000 ICC will remain the standard ICC and is not foreseen to be replaced with another ICC. This Touch-terminal will be used in many controlsystems in and around the LEP project.

The next generation of the Touch-terminal will have all its man-machine communications, display drivers, NODAL processor, etc. implemented in VME. CAMAC branch drivers could also be used in the VME enviroment for collection of data and proces control, but other possibilities such as GPIB would be possible too. In a longer time scale the VME based system will probably replace the NORD minicomputers in the SPS control room

1.3 PROBLEMFORMULERING

There is a strong demand for new Touch-terminals, and one of the problems with the Touch-terminal is that they only contain half a CAMAC crate, the other half is occupied with the touch-screen monitor and the powersupply. A typical set'up is shown in table 2.

WHY ONLY HALF A CRATE ?

Crate controller	ICC	- 2 slots
Dsplaydriver for the color monitor		- 2 slots
Displaydriver for the touch-screen		- 2 slots
File module, for storing programs		- 1 slot
Touch button interface		- 1 slot
Hard button interface		- 1 slot
Tracker ball interface		- 1 slot
Knob interface		- 1 slot
GPIB driver		- 2 slots
All together		13 slots

TABLE 2

CAMAC cards used in a Touch-terminal

The Touch-terminal only contains a 11 slot crate, so if none of the above functions has to be omitted, it is necessary to put the set'up in a full CAMAC crate (25 slots).

To save money and space in the CAMAC crate, there is a desire to integrate all the operator inputs into one unit. This can be done with the use of a microprocessor in the interface, the processor can do work which earlier would have required a lot of hardware.

Recently a STand Alone Computer - STAC has been developed at CERN. This STAC is a CAMAC slave processor, build around the Motorola 68000, with a design similar to the 68000 ICC. Seen from CAMAC this module looks just like an ordenary input/output module. On the frontpanel of the STAC there is a front-bus connector, which contains the full 68000 bus. The idea is to put a second card beside the STAC, connected to the STAC through the front bus. On this card all the interface for the input devices should be put, and the processor can go and read them every now and then, if something happens at the inputs, the STAC takes action, prepares the data for transfer on CAMAC, and transfer them to the CAMAC bus on request from the crate controller

The implementation where a STAC and a special purpose Man Machine Communication module - MMC, replaces 4 special purpose modules, has the following advantages:

1. The number of different modules is decreased from 4 to 1, the STAC is in use anyway for other purposes. This will also decrease the number of required spare modules for system backup.

? WHICH

2. Two CAMAC slots is saved, this means a lot in the half size CAMAC crates, where the tracker ball and the knob interface hasn't been used yet according ~~to~~ BECAUSE OF space problems.

3. The STAC can take some of the load away from the crate controller, concerning the checking of input devices etc. This will also decrease the trafic on the CAMAC bus.

4. It is also possible to let the STAC perform other work, f.ex. the crate controller could skip some time consuming calculations to the STAC. LANGUAGE PROCESSOR

5. Other interface cards could be connected to the frontbus, and the STAC could condense the results from these inputs and just transfer the results to the crate controller.

YES. 6. A cheaper touch-screen could be used, because adjustments and drift problems can be compensated in software.

7. This way of rearranging the input devices towards microprocessor control, will ease the change to VME, infact the only thing which has to be done is to change the CAMAC interface to a VME interface.

8. Last but not least, all the points mentioned above will decrease the price, increase system throughput, and make the system more flexible.

Chapter 2

THE TOUCH-SCREEN

The reason for having a touch-screen in the SPS control room, is to make the buttons super fluently which means that, every buttons function and name can be changed under computer control. The most flexible way of doing this is to use a monitor for displaying the button names, and then detect when a finger touches or points at the name. There is different ways to implement a touch-screen, these are:

1. Cross wires - This mechanical system was first described by Ferranti and Plessey in 1971.

2. Infrared light - The way of using intersecting lighbeams was first described by Rutherford Laboratory in 1972, this system is used now in the new Hewlet Packard 150 system with a resuluton of 14 row x 21 column.

3. Acoustic waves - This system is based on a pulse echo-ranging technique on an elastic surface. the resolution is 0-20 to 0-255 on each axis.

4. Resistive - A glass sheet coated with a transparent resistive substrate and a transparent conductive plastic sheet in front of it. When a finger pushes the two together, the coordinate can be found by detecting the resistor ratio in the x- and the y-axis. The resolution is mainly determined by the analog to digital converter.

5. Capacitive - A glass plate with a set of capacitors, made of thin line elements, invisible for the operator. When a finger touches the capacitor the dielectric will change and the capacity will increase. The capacitor is isolated from the finger by a thin and resistant epoxy lack layer. This system has been used in the SPS control room since its start in 1974.

2.1 THE CAPACITIVE TOUCH-SCREEN

The touch-screen chosen for the SPS control room is based on the capacitive princip. There is 16 touch sensitive areas on a glass plate organized in a 4 by 4 matrix. One of the touch capacitors is shown in figure 8. This touch-panel was designed and proven functional in 1973. The capacity of an untouched capacitor is around 80 pF, but the actual value variates between 66 pF and 90 pF dependent on where on the touch-panel the capacitor is situated. When a capacitor is touched slightly with only the tip of a finger, the capacitor increases with around 10 pF, again dependent on which capacitor it is. If a big swetty finger is presed hard against the capacitor, the capacity increases with around 30 pF. The untouched capacitor values is 1-2 pF higher when the touch-panel is dirty, compared to a clean one. A complete set of values is given in appendix 1.

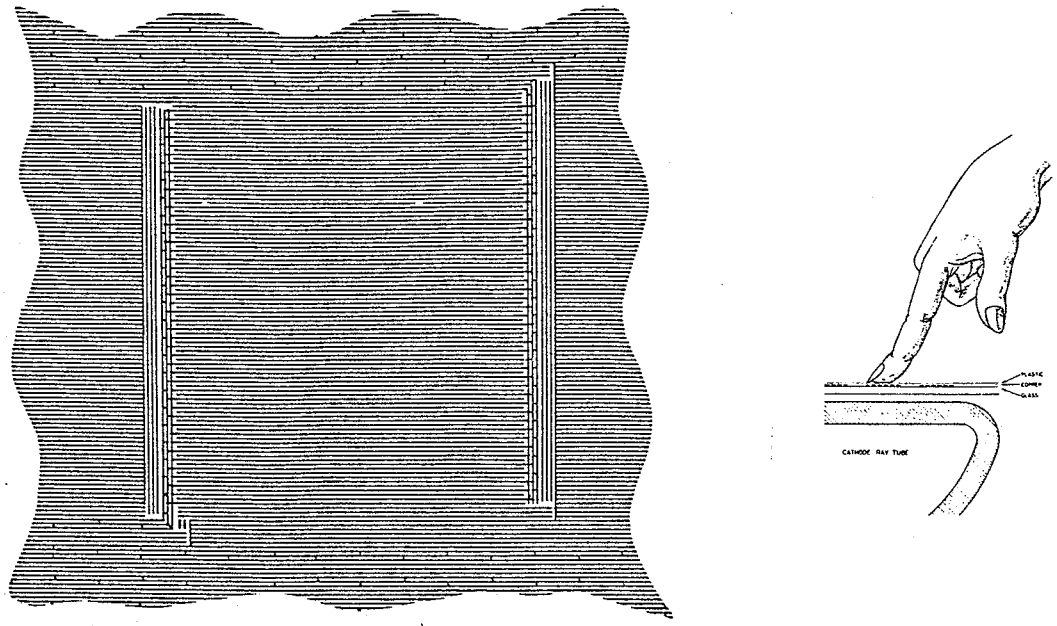


Figure 8: Enlarged picture of a touch capacitor

There is different ways of detecting the capacity change, and different methods have been used. The first method were based on phased-locked oscillator circuits as shown in figure 9. This method requires a lot of hardware, and each oscillator has to be adjusted very carefully to the touch-panel used.

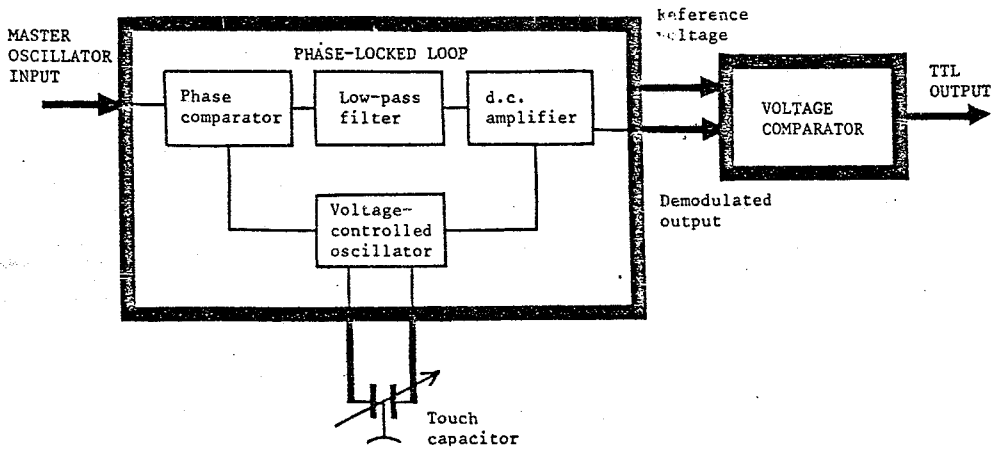


Figure 9: The phase-locked oscillator used in the first detector

The second method, which were introduced in 1977 and is the one used now, are based on detuning of a resonant circuit. A diagram of a complete detector for one button is shown in figure 10. These circuits are implemented using thickfilm circuits, mounted on a flexprint which is connected to the touch-screen. Here they are adjusted to resonans and sealed. When a finger touches the capacitor and the capacity is increased, the circuit is detuned, and the output voltage drops. This drop is detected by simple comperators. The advantage over the former method is: simpler and less hardware, and a lower price compared to the first method, but even so the price of the thickfilm circuits are pretty high.

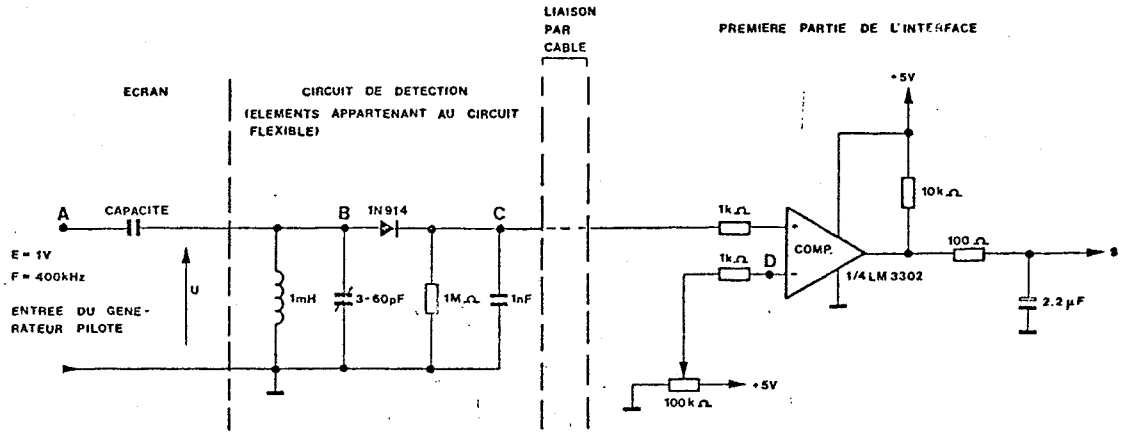


Figure 10: The tuned circuit for the touch-screen detector

When a micro processor is used in the detecting, it should be possible to let the processor measure the capacity, and if there has been a resonable change, it could signal that a button has been touched. It is not necessary to detect the exact capacity, but merely to be able to detect if there is any changes in the capacity, This means that non precission components can be used, and possible adjustment could be ommitted completely, even drifts in the circuit parameters could be outbalanced in software. All together this should decrease the price. Different methods can be used to detect a capacity, and in the following sections two different methods will be described.

1. Detecting the capacity, by detecting the time it will take to charge it to a certain level.

2. Detecting the ratio between two capacitors in a voltage divider circuit.

2.1.1 Charge time detector

One way to detect a capacity is to measure the time it takes to charge it to a given level. The relation for charging a capacitor is:

$$V(t) = V(\infty) + (V(0) - V(\infty))\ \ EXP(-R \cdot C/t)$$

or

$$t = R \cdot C \ln\left[\left(V(0)-V(\infty)\right)/\left(V(t)-V(\infty)\right)\right]$$

If V(0), V(∞), V(t), R is constant, then t is a function of the capacity alone. Hence a simple capacity/digital converter could be build using a counter to measure the time it would take to charge a capacity from 0 volt to a certain level.

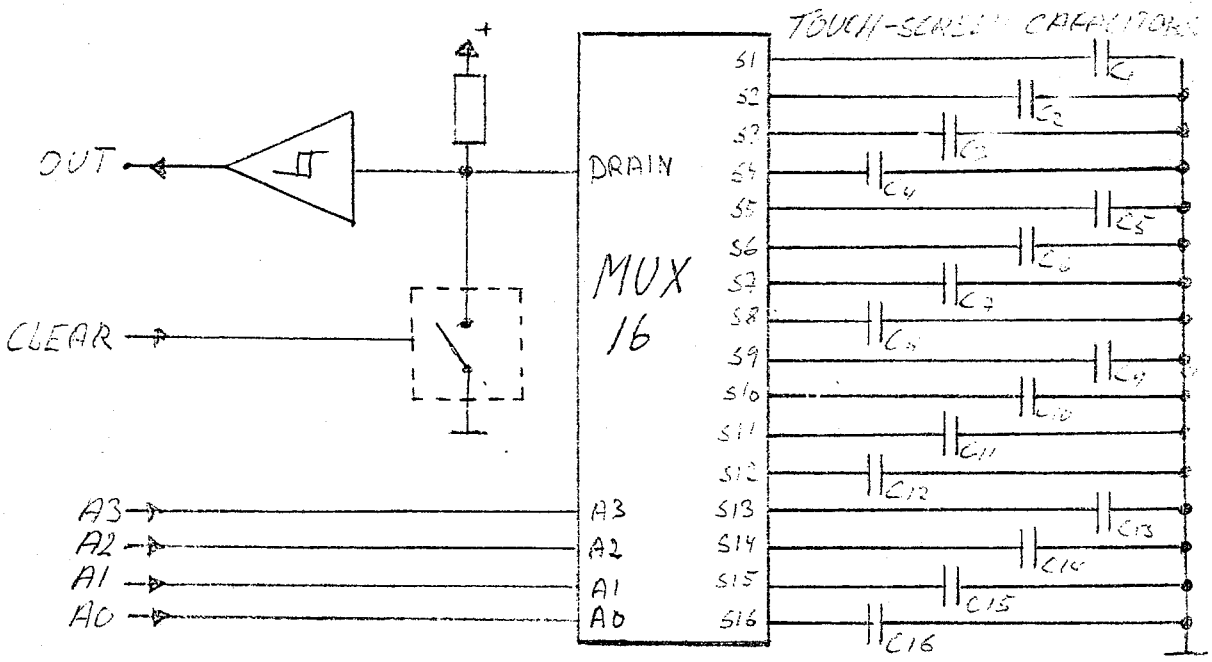


Figure 11: Charge time detector, with analog multiplelexing

A diagram of a touch-screen detector using this technigue is shown in figure 11, this circuit should be implemented on the flexprint next to the touch-screen to minimize the stray capacities. The multiplexer should be a type with low stray capacities, so that the capacity changes from the touch-screen is readable. The MUX 16 connects the touch capacitor which is selected with the address A0-A3, to the input of the comperator. When CLEAR is asserted the input of the comperator is connencted to ground, and the capacitor selected by the multiplexer is discharged. The output of the comperator is connected to a stop input on a counter. A detection is then made in the following way:

SET THE ADDRESS
ASSERT CLEAR
NEGATE CLEAR, START COUNTER
STOP COUNTER ON COMPERATOR OUT ASSERTED

The counter contains now a value which is proportional to the capacity of the capacitor selected, hence a change in capacity will give a change in the counter values.

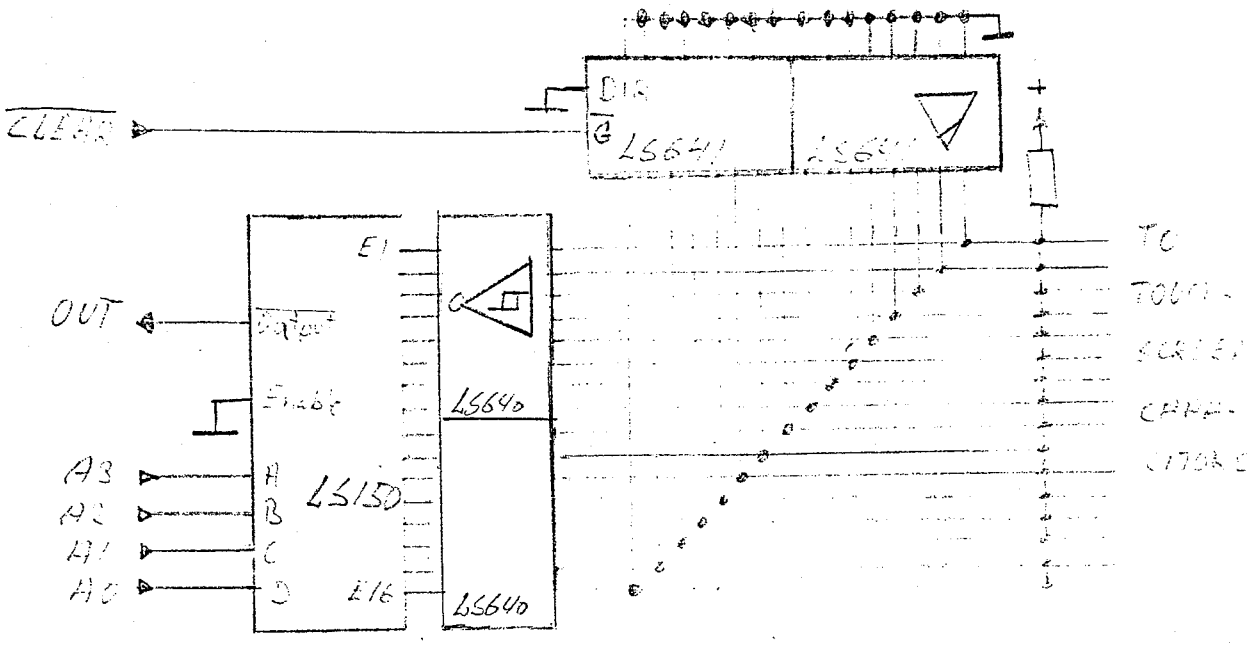


Figure 12: Charge time detector with digital multiplexing

If a comperator circuit is used for every touch capacity, the multiplexing could be done digital on the output of the comperator. It migh even be possible to use a normal TTL gate with schmitt-trigger input as the comperator. The circuit which should discharge the capacitor, could perhaps be implemented as the output of an open collector gate. A circuit example is shown in figure 12, This implementation would probalbly be the cheapest solution, using only one counter and a few TL circuits. There is no specifications for the stray capacities in the inputs of the schmitt-trigger, nor for the output of the open collector driver, so it is difficult to tell which influence those capacities will have on the circuit. Besides there is the load from the schmitt-trigger, which might be too high, and the shift voltage in the schmitt-trigger has to be stable, but even if the TTL schmitt-triggers were replaced with high impedanced schmitt-triggers based on OP-AMPs, the circuit would be simple and it could easily be implemented on the flexprint.

Both ways of implementing a charge-time detector for the touch-screen interface, has the advantage that no analog signals have to be transmitted between the touch-screen and the interface to the computer, which means that the cable lenght is not critical.

None of the charge time detector principles have been tried in praxis, but the circuit is so simple and cheap that prototype should be build for further evaluation.

WHY IS THIS NOT USED IF SO SIMPLE ?

REASONS : — PRODUCTION TOLERANCES ?
— DIFFERENT CAPASITOR VALUES ?
— DRIFT HOW IS THIS COMENSATED ?
— SENSITIVITY CONTROLLED HOW ?

2.1.2 Detecting a capacity via a voltage divider

The tuned circuit detector was very sensitive to small frequency changes in the oscillator and it is also very sensitive to small changes in the capacities there is between a clean and a dirty touch-panel, the output voltage could drop from 9 to 6 volt, if the touch-panel gets dirty.

YES BUT STILL OK !
HOW DOES NOISE INFLUENCE THIS CIRCUIT ?
WHAT IS THE VOLTAGE WHEN THE CAPASITOR IS TOUCHED ?

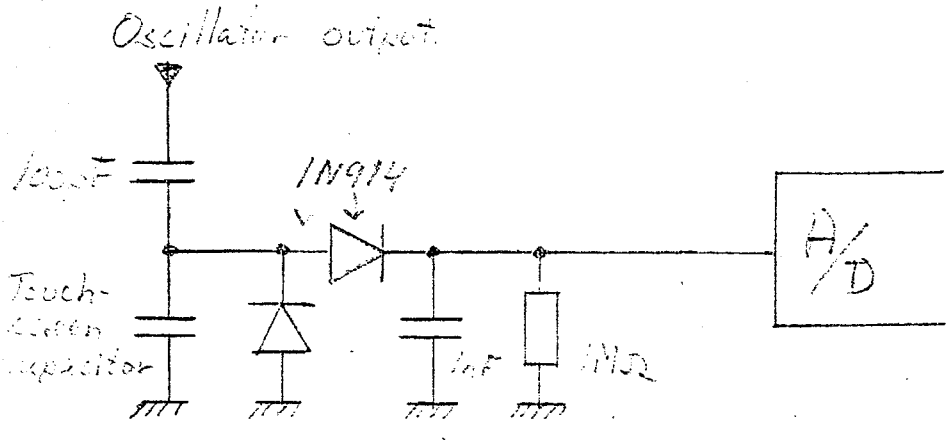


Figure 13: Detecting a capacity change using a voltage divider

If the touch-screen capacitor together with a fixed capacitor is building a voltage divider, as the one shown in figure 13, a change in the touch-screen capacity will result in a change on the output voltage from the diode bridge. This output is then connected to an analog/digital converter from where the processor can read the the values. This method is simple and requires only passive components on the flexprint.

The ratio between the untouched capacity and the fixed capacitor determines the size of the output, if the fixed capacitor is small compared to the touch-screen capacitor the output will be small but the the relative sensitivity to a change in the touch-screen capacitor will be high. On the other hand if the fixed capacitor is big compared to the touch-screen capacitor, the output will be high but the relative sensitivity to a touch will be low. The fixed capacitor is chosen to be 100 pF Which mean that with a 6 Vpp oscillator output and untouched capacitor value equal 80 pF, the output will be:

$$V_{80pF} = \frac{100}{100+80} \cdot 6v - 1,4v = 1,93 \text{ volt}$$

and for a touched capacitor the output will be:

$$V_{90pF} = \frac{100}{100+90} \cdot 6v - 1,4v = 1,76 \text{ volt}$$

If a lower value was chosen for the fixed capacitor, the circuit would be too sensitive to changes in the output voltage from the oscillator.

The oscillator chosen for this implemantation is a crystal based sinewave oscillator running at 456 kHz, the output is 8 Vpp with a 90 ohm output impedance. It is the same as the one used in the touch-screen interface based on the tuned circuit design. This is done by two reasons; this oscillator design has worked without any problems since it was designed in 1976, with this solution it would also be possible to make an interface to the tuned touch-screen which is used now, the only thing which has to be changed is the sofware which handles the interface. The design of this oscillator is described in litt 6.

The analog digital converter, must have an resolution high enough to detect the changes which apears when a button is touched. Here a 8 bit A/D converter is chosen this gives sufficient resoultion to detect the change which is around 10 percent. The referens voltage to the A/D converter is derived from the oscillator output and is scaled so that the normal outputs from the screen is around half the maximum input voltage. This means that a change on 10 percent on the output, will give change the digital output with 12. The A/D converter used is an 8 input 8 bit converter with internal data transfer to registers, so that the converter looks like 8 memory cells to the computer, the data sheet is in appendix 4. The only disadvantage with this converter is the low input impedance, 20 kohm. This could easily disturb the voltage dividers on the flexprint, Therefore a test was caried out with and without the ADC load. The result of this test is in appendix 3 and the basic results is shown in table 14. It turned out that the the relative change was almost the same in the two cases, although the load from the ADC decreased the output. When the ADC was connected, a strong ripple also occured, because the time constant became to low in the output. A 100 nF capacitor was put in parallel to the ADC input and this solved the problem, the time constant is now 2 usec which is fine both compared to the oscillator frequency and to the ADC's conversiontime.

	no load	ADC load	ADC and 100 nF
untouched	2.8 V	1.7 V	1.7 V
touched	2.3 V	1.4 V	1.4 V
ripple	0.05 V	0.2 V	0.00 V
change	17.9 %	17.7 %	17.7 %
oscillator out	8.0 Vpp	7.5 Vpp	7.5 Vpp

TABLE 14

Output from the voltage divider circuit with different loads

The reference voltage for the ADC is derived from the oscillator output which means, that changes in the oscillator output will also change the sensitivity for the ADC. The clockfrequency for the ADC is taken from the E output on the 68000 microprocessor. This output is normaly used for synchronization with 6800 peripherals. The frequency is one tenth of the processors clock frequency. In this case the processor runs at 8 MHz which results in a E frequency at 800 kHz. A complete conversion of the 8 analog inputs is specified to 640 clockcycles which will take 0.8 msec.

There is implemented 4/8 input AD converters which gives 32 analog inputs, beside that there is 4 or 5 control bits as output. The reason for this is that a possible future implemntation of a high resolution touch-screen might require more analog inputs and also some control bits. Another use of the analog inputs would be for for a joystick, or a trackerbutton as proposed by Martin Flückiger, which is electrically similar to the joystick. Ofcourse other analog inputs such as temperature sensors could be connected to the ADCs too.

REF ?

The 16 buttons touch-screen requires only ADC 1 and ADC 2 to be present and the connector to the touch-screen is made so that the 40 pin scotchcable connector can be replaced with a 20 pin connector only using pin 17 to pin 36 of the 40 pin connector. This 20 pin connector will then contain the required connection between the touch-screen and the MMC module.

The 32 analog inputs can be accessed at the odd byte addresses from D00001 to D0003F, or at even word addresses D00000 to D0003E, the 8 bit data would be in the loworder byte of the received word.

2.2 CLICK GENERATOR

When a finger touches a touch-button the system generates a click as the audiable feedback. This click is generated with a small loudspeaker and a oneshott generator, the design is a copy from the hard-button interface module. The click response is much more pleasent than the bib-bib sound used in many new terminals. The processor generates the click by simply writing to the address D00001 (CLICKADR).

Chapter 3

MECHANICAL BUTTON INTERFACE

The hardware interface to the mechanical buttons is simply two 8 bit digital ports with the buttons connected to it directly as shown in figure 14. The contact could also be the output of an open collector TTL gate.

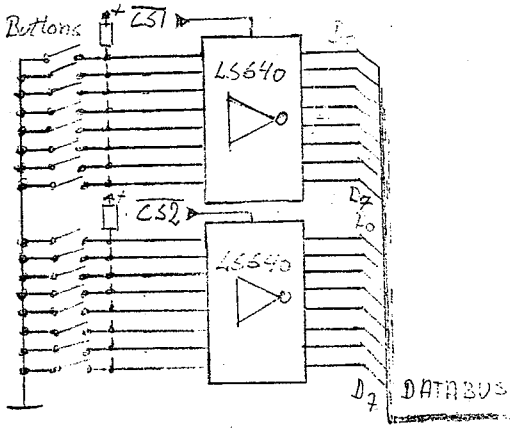


Figure 14: Mechanical button interface

The two ports can be red on address D00051 and D00061 and detection of the button pressed and debouncing is done in software.

How ?

Chapter 4

ENCODER INTERFACE

The outputs from the trackerball and the knob is square signals, where the number of periodes is proportional to the turn of the trackerball or the knob. There is two different encoder types quadrature and countnocount encoding. In the quadrature encoder there is two square outputs which looks equal but with a phase differens off +/- 90 grades. The sign of the phase difference determines in which direction the encoder is turned. In the countnocount encoder only one output is changing when the encoder is turned. If the encoder is turned in the positive direction x_2 remains low while the pulses proportional to the encoder turn is present on the x_1 output. If the encoder is turned in the negative direction x_1 remains low while the pulses is present on the x_2 ouput. The trackerball which is used in this project and described in appendix 5 has countnocount encoders, while the mouse which is described in appendix 6 has quadrature encoders.

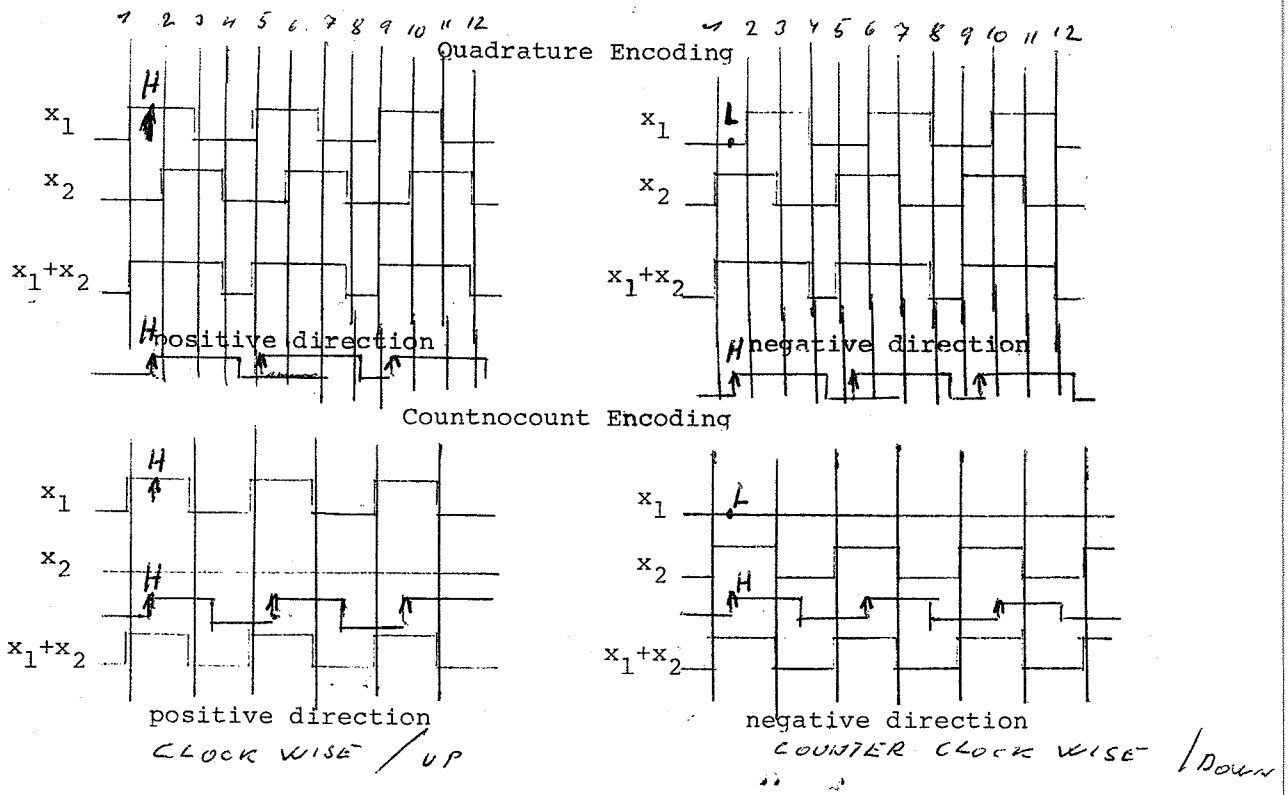


Figure 15: Quadrature and countnocount encoder outputs

Valid Memory Address is in the address space from D20001 to D3FFFF on the odd byte address

$$G64VMA = A23 \cdot A22 \cdot \overline{A21} \cdot \overline{A20} \cdot \overline{A19} \cdot \overline{A18} \cdot A17 \cdot LDS \cdot \overline{IACK} \cdot VMA$$

Valid Peripheral Address is in the address range from D10001 to D107FF on the odd byte address, G64VPA is not fully decoded.

$$G64VPA = A23 \cdot A22 \cdot \overline{A21} \cdot \overline{A20} \cdot \overline{A19} \cdot \overline{A18} \cdot \overline{A17} \cdot A16 \cdot LDS \cdot \overline{IACK} \quad VMA$$

Enable for the G64 data buffer is valid when G64VMA or G64VPA is selected, but the enable is independent of the VMA output from the 68000 processor.

$$\begin{aligned} G64EN &= A23 \cdot A22 \cdot \overline{A21} \cdot \overline{A20} \cdot \overline{A19} \cdot \overline{A18} \cdot A17 \cdot LDS \cdot \overline{IACK} \\ &+ A23 \cdot A22 \cdot \overline{A21} \cdot \overline{A20} \cdot \overline{A19} \cdot \overline{A18} \cdot \overline{A17} \cdot A16 \cdot LDS \cdot \overline{IACK} \end{aligned}$$

The G64EN output also initiates the synchronous cycle by asserting VPA to the 68000 processor. The input to the PALASM is given in appendix 9.

Only the IRQ interrupt output from the G64 card is connected and it is routed to the encoder interrupt grader with vector 136 and a higher priority than the encoders. This means that when the G64 card sets an interrupt on the IRQ line the 68000 processor will receive a level 5 interrupt with vector 136. This solution is chosen because there is only one interrupt line on MC68000 frontbus. The disadvantage with this implementation is that the G64 interrupt routine must end with a dummy read of the encoderoutput to enable for a new interrupt from the encoders, because if an encoder interrupt had occured in the meantime the interrupt to the processor would remain asserted even when the G64 interrupt was removed and the new interrupt vector would not be latched into the interrupt vector register. A better solution would be to let the G64IRQ generate an autovector interrupt. An autovector interrupt is generated by asserting the VPA line to the processor in the response to interrupt acknowledge cycle, if DTACK was asserted the processor would read a uservector from the databus. The solution with the autovector would require more hardware so the former solution was preferred. If other G64 interrupts is necessary some other external connections between the STAC and the G64 card must be made.

Chapter 6

CONTROL LOGIC USING PAL's

Two PAL's is used in the control logic in the MMC module. IC 28 is used for predecoding of the address space for the MMC circuits and the G64 card and enabling of the data buffers.

MMCCS selects the address space for the man-machine communication. The MMC devices is located in the address space from D00001 to D0007F, but is not fully decoded.

$$MMCCS = A23 \cdot A22 \cdot \overline{A21} \cdot \overline{A20} \cdot \overline{A19} \cdot \overline{A18} \cdot \overline{A17} \cdot \overline{A16} \cdot LDS \cdot \overline{IACK}$$

MMCCS ? DRAWING SAYS G64EN → VPA

This output also asserts the VPA ~~to the processor~~ to the processor, to initiate a synchronous cycle. The ADC's is too slow to run on the 68000's full speed, therefore a synchronous cycle is initiated, another possibility is to assert a delayed DTACK, but this would require more hardware.

DATA (Ic 22)

The buffer for the MMC circuits is enabled when MMCCS is asserted and when the processor read the interruptvector from the encoder interface.

— Ic 20

$$MMCEN = A23 \cdot A22 \cdot \overline{A21} \cdot \overline{A20} \cdot \overline{A19} \cdot \overline{A18} \cdot \overline{A17} \cdot \overline{A16} \cdot LDS \cdot \overline{IACK} + ENCINTRD$$

BUFFER (Ic 22)

The second PAL IC 27 takes care of the control signals used in the interrupt acknowledge cycle for the encoder interrupt. The function codes from the 68000 is used to indicate which cycle type the processor is running, the ralation is shown in table 4.

Function Code Output			Reference Class
FC2	FC1	FC0	
0	0	0	(Unassigned)
0	0	1	User Data
0	1	0	User Program
0	1	1	(Unassigned)
1	0	0	(Unassigned)
1	0	1	Supervisor Data
1	1	0	Supervisor Program
1	1	1	Interrupt Acknowledge

TABLE 4: Reference classification

The interrupt acknowledge output is used in the previous PAL so that no devices is selected in this cycle. The IACK output is generated in this PAL with the equation:

$$IACK = FC2 \cdot FC1 \cdot FC0$$

When the processor is running an interrupt acknowledge cycle, the interrupt level which is serviced is indicated on the three lowest address lines A3, A2, A1. The encoder interrrupt is located at level 5, therefore the encoder interrupt vector must be written onto the data bus as a response to an interrupt acknowledge cycle at level 5 which means that the following equation must be fulfilled:

$$ENCODINTRD = FC2 \cdot FC1 \cdot FC0 \cdot A1 \cdot \overline{A2} \cdot A3 \cdot LDS$$

at the same time the datatransfer acknowledge DTACK is asserted, to signal to the processor that a uservector is written onto the data bus.

The write signals for the MMC circuits is also implemented in the second PAL. The click is located at address D00001

$$CLICK = MMCCS \cdot \overline{READ} \cdot \overline{A1} \cdot \overline{A2} \cdot \overline{A3} \cdot LDS$$

The address where the enable mask for the encoders is located at address D00003

$$ENCEN = MMCCS \cdot \overline{READ} \cdot A1 \cdot \overline{A2} \cdot \overline{A3} \cdot LDS$$

The latch which holds the output of the controlbits to the touchscreen can be accessed on address D00005

$$BUTWR = MMCCS \cdot \overline{READ} \cdot \overline{A1} \cdot A2 \cdot \overline{A3} \cdot LDS$$

The last output which is generated in this PAL is used to clear encoder interrupt flip-flops. Those flip-flops is cleared on every read cycle from the encoderoutput and also when the processor is reset. If reset didn't clear the encoder flip-flops there would be problems in the upstart, because the encoder interrupt register might contain a nonvalid interrupt vector.

Chapter 7

CAMAC COMMUNICATION

CAMAC is a modular data handling system in widespread use with online digital computers. The CAMAC bus consists of 24 write bus-lines, 24 read bus-lines, 5 function code lines defines the functions F(0) to F(31), 4 sub-address lines defines the sub-addresses A(0) to A(15), and some control lines. Beside the buslines every module has a N line (slot number) which is used to select the module and a LAM line (Look-At-Me) which is used by the module to signal to the crate controller that it needs attention. A brief describtion of CAMAC is given in appendix 12, which is small parts taken from the full documentation litt. 8.

7.1 CAMAC - STAC INTERFACE

The basic communication between the STAC and CAMAC is done in the following way. The crate controller accesses the STAC with a certain NAF-code and continues with that until it receives a Q-respons. The Q-respons signals to the crate controller, that the STAC has recognized and performed the NAF-function. The NAF-function could be a read from CAMAC, a write to CAMAC, or just a command.

The STAC receives an interrupt when it is accessed with a NAF it can recognize, at the same time the A and F is latched into a register. The interrupt routine can read the A and F from the register and take action on the different commands. There is two independent Q-responses in the STAC, one for the CAMAC read functions F(0) - F(15) and one for the CAMAC write functions F(16) - F(31).

By LAM.

When the STAC writes to CAMAC it will set the F(0)-F(15) Q flip-flop and the data will be latched into the CAMAC output buffer. When the crate controller accesses the STAC with a read function, it will receive the data from the CAMAC output buffer together with the Q-respons which tells the crate controller that data is present. The F(o)-F(15) Q means that data is ready in the CAMAC output register. The Q flip-flop is cleared after the CAMAC acces. To THE STAC!

When the crate controller writes to the STAC and the CAMAC input buffer in the STAC is empty, the crate controller will receive a Q-response and the submitted data is latched into the STAC's input register. If the input register wasn't empty, nothing would be latched in and the crate controller would not receive a Q-respons. When the STAC reads the CAMAC input register, the F(16)-F(31) Q flip-flop will be set and this will enable for a new dataset to be latched in from CAMAC. The F(16)-F(31) Q-respons means that the input buffer is empty, and that the submitted data is received, and the STAC is ready to perform a new operation. The Q flip-flop is cleared after a CAMAC access.

b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
A8	A4	A2	A1	F16	F8	F4	F2	F1	N

TABLE 5: Format of the status register which receives the NAF-code

The CAMAC interrupt routine shown in appendix 13 uses bit b1-b9 from the status reister as a pointer to a table which contains the offset vectors for for the different CAMAC functions and sub addreeses, the format of this table is shown in table 6.

```
CAMTABEL  v(A(0),F(0))        offset vector for subroutine for A(0), F(0)
          v(A(0),F(1))        etc.
          v(A(0),F(2))
               .
               .
               .
          v(A(15),F(31))      last vector
```

TABLE 6: Pointers to CAMAC function subroutines

v(A(0),F(0)) is defined as the offset from CAMTABEL to the start of the subroutine for A(0), F(0).

When a CAMAC interrupt occurs the following program is executed:

```
CAMACIRQ  READ A,F from status register
          MASK b0, b10-b15 out
          READ v(A,F) from CAMTABEL
          CALL subroutine v(A,F) + CAMTABEL
          RETURN from interrupt
```

The different subroutines performs the actual CAMAC operations. Those CAMAC functions which is not implemented prints the function F(x) and the subaddress A(y) on the terminal connected to the STAC.

7.2 LAM HANDLING

LAM - Look-At-Me is used as an interrupt from a module to the crate controller telling it that the module needs to be serviced. In the STAC there is more than one source which might require service from the crate controller. The CAMAC specifications operates with two groups of registers, register 1 which is mainly used for data transfer and register 2 which is mainly used for transfer of status information. Register used for LAM information is specified to group 2 registers at the following sub addreeses:

LAM status register	reg. 2 sub adr. A(12)
LAM mask register	reg. 2 sub adr. A(13)
LAM request register	reg. 2 sub adr. A(14)

In those registers a certain bit position corrospond to a certain LAM source and for the MMC module they are defined in the following way:

b0	Encoder 0
b1	Encoder 1
b2	Encoder 2
b3	Encoder 3
b4	Encoder 4
b5	Encoder 5
b6	Encoder 6
b7	Encoder 7
b8	Touch buttons
b9	Mechanical buttons
b10-b23	not used

TABLE 8: Relation between LAM sources an bit position in LAM reg.

The group 2 registers is organized as a block of 16 longwords in the STAC, one location corrosponding to every subaddress. This block is accessible from internal STAC programs and from CAMAC with the following functions:

F(1)A(x) - Read register 2 sub adr. A(x)
F(17)A(x) - Write to register 2 sub adr. A(x)
F(19)A(x) - Selective clear register 2 sub adr. A(x)
F(23)A(x) - Selective set register 2 subadr. A(x)

TABLE 9: CAMAC functions used for transfer of status information

When a device needs attention from the crate controller, it sets its bit in the LAM-status register and if the corrosponding bit is set in the LAM-mask register, which means that LAM is enabled, it set the bit in the LAM-request register and asserts a LAM on the CAMAC bus. The first thing the crate controller must do after a LAM is received is to read the LAM-request register to find the device which created the LAM and then take action on that.

7.3 COMMUNICATION BETWEEN THE MAN-MACHINE ROUTINES AND CAMAC

There is many registers necessary for the Man-Machine/CAMAC communication more than what is supplied by the CAMAC sub-addresses. The communication is therefore always done with two CAMAC operations. First the crate controller sends a pointer to the STAC and then afterwards the actual operation takes place. The pointer operations is done on sub-addres A(o) and the data transfer is done on sub-address A(1). The pointer values is given in table 10 and table 11. All the operations is only word-access in this communication.

Touch-button-out	0
Touch-button-mask	2
Mechanical-button-out	4
Mechanical-button-mask	6
Encoder-enable-mask	8
not used	A
not used	C
not used	E

TABLE 10: Pointer values for acces of MMC register from CAMAC

	Output	Low limit	High limit	Scale factor
Encoder 0	10	20	30	40
Encoder 1	12	22	32	42
Encoder 2	14	24	34	44
Encoder 3	16	26	36	46
Encoder 4	18	28	38	48
Encoder 5	1A	2A	3A	4A
Encoder 6	1C	2C	3C	4C
Encoder 7	1E	2E	3E	4E

TABLE 11: Pointer values for acces of encoder registers

The CAMAC functions which is used in the Man-Machine communication is given in table 12.

Control functions

F(16)A(0)	- Write pointer to MMC block
F(24)A(0)	- Disable the Man-Machine communication routines
F(25)A(0)	- Set the new encoder interrupt mask
F(26)A(0)	- Enable the Man-Machine communication routine

Transfer functions

F(0)A(1)	- Read the register the MMC-pointer points at
F(2)A(1)	- Read and clear the register MMC-pointer points at
F(16)A(1)	- Write to the register the MMC-pointer points at
F(25)A(1)	- Make a click (sound)

TABLE 12: CAMAC functions used in the Man-Machine communication

7.4 COMMUNCATION G64 - CAMAC

This G64 - CAMAC communication is only a simple transfer of data between a G64 card and CAMAC. First the address must be supplied with CAMAC function F(16)A(2) and then afterwards data can written to the G64 card with CAMAC function F(16)A(3) or data can be red with CAMAC function F(0)A(3).

When a special purpose G64 card is implemented, special functions should be written to allow blocktransfer, highlevel commands etc.. In this way the STAC could take some of the load away from the crate controller.

7.5 GENERAL PURPOSE COMMUNICATION STAC - ICC

Those CAMAC function which is written by Ole Borch will not be described in this report.

Chapter 8

SOFTWARE FOR DETECTION OF OPERATOR INPUTS

The software which checks if there has been an operator input, is interrupt driven from the hardware timer on the STAC. This timer creates an interrupt every 2 msec. and is enabled with a read from the TIMER address and cleared and disabled with a write to TIMER address.

The timer interrupt is shown in appendix 14 line 107-133 and the function is the following:

If everything settled - CALL TOUCHCHECK
Otherwise - Initialize touchscreen variables - CALL TOUCHINI
Check for mechanical buttons pressed - CALL MECHCHECK
Every 25 interrups (50 msec.) - Check if there has been an
encoder movement - CALL ENCODCHECK
Update the LAM-request register according to the content of
the LAM-status and the LAM-mask registers.
If any bit set in the LAM-request register - Set LAM on CAMAC

8.1 TOUCHINI

The TOUCHINI routine initiates all the variables for the TOUCHCHECK routine. This routine is called from the TIMERIRQ routine as long as the TINIT counter isn't negative, TINIT is decremented on every call, so when TINIT gets negative the initialisation is finished and normal operation will continue.

In the power up phase the MMCINIT routine is called and looo is written into the TINIT register and the Timer interrupts is enabled. This means that normal operation will start after 2 seconds, when the crystal oscilator is settled. The touchscreen variables can also be initialized when the STAC is accssed with the CAMAC function F(26)A(0). The TOUCHINI routine is shown in appendix 14 line 134-161 and the MMCinit routine is in line 106-107.

8.2 TOUCHCHECK

The TOUCHCHECK routine which services the touch-screen is called every 2 msec. from the TIMERIRQ routine, but it is not called during initialisation. The TOUCHCHECK routine reads the ADC's to see if there has been a resonable change compared to a mean value, if so the touch-screen must have been touched by a finger, and the program takes then action on this. The main features of the program is:

1) Self calibration and drifts adjustment every 2 seconds.

2) A button is recognized as touched when the value from the ADC is 12.5 % lower as usual.

3) A button is recognized as untouched when the value from the ADC is less than 6.25 % lower than usual.

4) The touch button mask register can disable any buttons.

5) A button must be recognized 5 times (lo msec.) and the button must be enabled in the touch-button-mask reg, before the button is written into the touch-button-output register and the LAM status is set.

6) No buttons must be recognized as touched 10 times (20 msec.) before the value for no buttons touched (-1) is written into the touch-button-output register.

7) When one button is touched no other buttons is detected. INHIBIT OF INPUT

The TOUCHCHECK routine is shown in appendix 14, a describtion of the variables is given in line 30-46 while the program is shown in line 162-290. The TOUCHCHECK routine can be logically partet into three.

1) Detection of button touched - line 189-228

2) Debouncing and check of touch-button-mask before the button number is considered valid and send to CAMAC - line 229-264

3) Calibration of the untouched values every 2 seconds - line 265-291

The output of the first part is:

TCUR : The button number touched, TCUR can only be changed from -1 (not touched) to the button number touched

TOFTIM : Incremented if TCUR isn't touched this time and cleared if TCUR is touched.

TTIMES : Incremented when TCUR is touched again, if it is the first time the button is detected touched TTIMES=1 and TCUR=button

TDIF(I) : The accumulated differens between TMEAN(I) and the actual value TNEW(I), buttons with a value lower than TOFF(I) is not considered. The TDIF(I) value is used in the calibra-routine ADJTOUCH.

The second part of the touch detector program cummunicates with CAMAC through the touch button output register. A button is considered as touched when TTIMES > ONDELAY which means that a button has been recognized as touched ONDELAY (5) times, if this is the case and the button is enabled in the touch-button-mask register, the button number is written into the touch-button-out register and the bit corrosponding to the touch-button routine is set in the LAM-status register. As long a a button is touched no other buttons will be considered as valid. A button is considered as untouched when the button has been recognized as untouched in the last OFFDELAY (10) times (TOFTIM > OFFDELAY). If this is the case -1 is written into TCUR and the touch-button-out register, and the bit in the LAM-status register is cleared.

Every 2 seconds (1024 interrupt cycles) the ADJTOUCH routine is called and the values for TMEAN(I), TOFF(I), and TON(I) is updated according to the value of TDIF(I), I relates to the button number between 0 and 15.

8.3 MECHCHECK

The program which services the mechanical buttons is very similar to the TOUCHCHECK routine. The only differens is that there is no hysteresis on the inputs from the mechanical buttons, which makes the program a little simpler. The variables for the routine is described in line 47-54 while the program is shown in line 292-383

8.4 ENCODCHECK

This routine which is called every 50 msec. (25 timer interrupts) checks if there has been any change in the encoder counter values. If there has been a change the counter values are checked against bounds before they are written into the the encoder output register. Every encoder is checked in the following way:

If the counter value is lower than the low limit - The low limit is written into the encoder-output register and into the counter.

If the counter value is higher than the high limit - The high limit is written into the encoder-output register and into the counter.

Otherwise the counter value is written into the encoder-output register.

Finaly if there has been a change in the encoder output register - The encoders LAM bit is set in the LAM status register.

This is not the final version of the ENCODCHECK routine. In the final version there will also be a scaling factor between the counter and the encoder-output, but this is not implemented yet. The listing of the program is shown in appendix 14 line 384-421 together with the software counters line 422-484.

Chapter 9

MAN - MACINE FUNCTIONS IN THE ICC

The Man-Machine functions in the ICC is NODAL functions which can read the touch-screen, the mechanical buttons, the knob and the tracker ball position. A brief description of the functions which is defined in litt. 13 is given in the following.

LEGEND Write touch button legend
$SET LEGEND(N)=CONCATENATION
Writes the CONCATENATION on the touch-screen monitor under button N (1-16)
N=0 clears the screen and writes the CONCATENATION as a header of the screen, all buttons are disabled
Buttons with no legends is always disabled.

BUTS Read button
SET Z=BUTS
Immidiate read of button
Z on return holds the button number
Z=0 means no buttons touched
The button is cleared after a read

BUTTON Wait for a button to be pressed
SET Z=BUTTON
The button function waits until a button has been pressed
Z on return holds the button number.

KNOB Read or write the value of the knob
SET A=KNOB
SET KNOB=B
Valid range is -30000 to 30000

SKNOB Set endstop values for the knob
CALL SKNOB(LOW,HIGH)

HBALL Read or write the horizontal ball position
SET A=HBALL
SET HBALL=B

VBALL Read or write the vertical ball position

```
SET A=VBALL
SET VBALL=B
```

MCURS Set the operational mode of the cursor

```
CALL MCURS(M,H)
```

The valid display modes (M) are:

```
M=0,1 - suppress cursor display
M=2   - horizontal line
M=4   - vertical line
M=6   - cross, horizontal an vertical line
M=8   - solid
M+1   - as mode M but blinking
```

The second parameter (H) hooks the cursor to the movement of the tracker ball.

The valid values for hookin are:

```
H=0   - unhook the cursor from the ball
H=1   - hook the cursor to the horizontal ball movements
H=2   - hook the cursor to the vertical ball movements
H=3   - Hook the cursor to both ball movements
```

HCURS Read or write the horizontal cursor position

```
SET X=HCURS
SET HCURS=X
```

HCURS works in graphics units 0 to 767

VCURS Read or write the vertical cursor position

```
SET Y=VCURS
SET VCURS=Y
```

VCURS works in graphics units 0 to 575

CURC Read or write horizontal cursor position in column units

```
SET C=CURC
SET CURC=C
```

The column numbering runs from left to right 1 to 64

The conversion formulae between column units and graphic units are:

```
C := INT(X/12) + 1
X := 12 * C - 6
```

CURL Read or write vertical cursor position in line units

```
SET L=CURL
SET CURL=L
```

The line numbering runs from top to bottom 1 to 24

The conversion formulae between line units and graphic unit are:

```
L := 24 - INT(Y/24)
Y := (24-L) * 24 + 12
```

The LEGEND, BUTS, and BUTTON function is already implemented in the touch-terminal, using the touch-button-interface module and the rounded-character-generator. With the introduction of the integrated Man-Machine communication module, BUTTON and BUTS must be rewritten while the LEGEND function could remain the same. The BALL and CURSOR functions has not been implemented in the touch-terminal before.

Almost everything in NODAL is packed into self-contained elements, program lines, variables, functions etc. These elements are completely relocatable and computer independent and they are the basis in of NODAL's multi-computer capability. The only elements which is not computer independent is the defined functions which is written in macine-code.

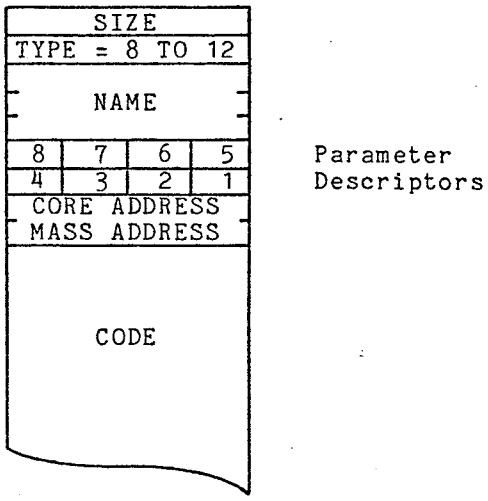


Figure 18: General purpose function header - assembler function

The format of a header for an element is shown in figure 18. The first word contains the size of the element, this value is used to make chain of all the elements. The second word indicates the type of the element. The next three words indicates the name of the element, or if its a line element the line number will be given here. The remaining part of the header varies from element type to element type. The assembler functions is defined as:

8 - Read/write general purpose module	ASSFN
9 - Write only general purpose module	ASSFN
10 - Read only general purpose module	ASSFN
11 - CALL, NODAL error return g. p. module	ASSFN
12 - CALL, no error return g. p. module	ASSFN

In the assembler functions the name of the function is followed by parameter description, the differant parameter types is shown in table 13.

1 - real value	- 3 word floating point number
2 - integer value	- 1 word integer
3 - string value	- 4 word string descriptor
4 - NODAL reference	- NODAL data element, address of first word, i.e. size
5 - real reference	- 3 word floating reference
6 - integer reference	- 1 word integer reference
7 - real array	- 3 word per element floating array
8 - integer array	- 1 word per element integer array
9 - NODAL name	- 3 word element

TABLE 13: Parameter types

The BUTTON and BUTS functions is read only functions which is a type 10 element type. In a type 10 element type the result of the function shall be placed in the floating point destination which is the content of the A2 register when the function is called. The type is always a real value, 3 word floating point, and this value is not defined in the parameter description in the header. The parameter description is only used for parameters not for the result of read function or the input to the write function, fore example the parameter description would be used in a BUTTON function which could read more than one touch-screen, the BUTTON call would look like this SET A=BUTTON(I) where I is the touch-screen number and the value which should be defined in the parameter description.

9.1 LEGEND

It was decided to change the LEGEND routine a little bit so that the touch-button TBUTMASK was transmitted to the touch-button-mask register in the STAC whenever it was changed. In this way the check for enabled buttons could be done in the STAC instead of in the ICC, this would also simplify the BUTTON and BUTS functions in the ICC. The LEGEND function is shown in appendix 16 line 323-480 and the only change from the original LEGEND function is line 398-401 and 463-466 which is inserted.

9.2 BUTS

The BUTS function reads and clears the touch-button-out register in the STAC. First the pointer to the touch-button-out register is submitted with CAMAC function F(16)A(0) and then the content of this register is red and cleared with CAMAC function F(2)A(1). When the button number received is -1, which means no buttons touched, the routine jumps to the BUTTON routine and the functions returns with the value 0. If the button numbder received wasn't -1, the routine jumps to the BUTTON routine where a click is generated with CAMAC function F(25)A(1) to the STAC and function returns with the button number touched. The button values in the STAC is 0-15 while the button values in the BUTTON and BUTS function has the values 1-16. If the backbutton should be pressed the number returned will be 17. The BUTS function is listed in appendix 16 line 224-260 and the part of the BUTTON routine which is used in line 212-228.

9.3 BUTTON MECH. BUTTONS

The BUTTON function waits for a button to be touched and when the button is touched the function returns with the button number. The program listing is shown in appendix 16 line 133-223.

Both the mechanical- and the touch-button routine sets a bit in the LAM-status register when a button is touched. Therefore the BUTTON routine could wait until one of those bits is set and then read the button number and return this. If the ICC polled on the LAM-status register in the STAC until a LAM-bit was received from one of the button routines in the STAC, the STAC would be fully occupied with the service of this operation, therefore another implementation is chosen.

The LAM-handler routine in the ICC reads the LAM-request register from STAC and saves this value in a LAM-register in the ICC. This means that this LAM-register will allways contain an updated copy of the LAM-request register in the STAC. In this way a polling on the different LAM sources in the STAC is possible without disturbing the STAC, the polling is simply done on the LAM-register in the ICC.

A call of the BUTTON function will first set bit 8 (touch-buttons) and bit 9 (mechanical-buttons) in the LAM-mask register in the STAC, this is done with the CAMAC function F(19)A(13). This means that a pressed/touched button will result in a LAM from the STAC. The LAM-handler in the ICC will on this LAM interrupt read the LAM-request register with CAMAC function F(1)A(14) and save the value in the LAM register, this register will now have bit 8 or bit 9 set depending of which kind of button was activated. The routine waits until bit 8 or bit 9 is set or an escape is typed on the keyboard.

If bit 8 is set the routine creates a click and the function returns with the button number touched.

If bit 9 is set the routine checks if it was the back-button which was pressed, if so a click is generated and the routine returns with the value 17

If escape was typed on the keyboard the routine will make an error return and write escape typed on the connected terminal.

At the end of the routine the LAM-mask is regenerated both in the STAC and in the ICC

The return of the button number number is done through the call of the NLZ16 routine which takes the 16 bit value in register D0 and converts it to floating point value and saves the value in the address given in register A2.

9.4 KNOB

The knob can be connected to any of the eight encoder inputs and it is chosen to connect the knob to the encoder 0 input and if more knobs is used they should be connected to the succeding encoder inputs.

When the STAC is reset all the encoder counters is disabled, therefore the ICC shall initialize the encoder for the knob before it can be used, this is done by writing -30000 into the encoder 0 low-limit register, 30000 shall be written into the encoder 0 high-limit register and the encoder 0 out register shall be cleared, finaly the encoder 0 counter must be enabled, the procedure will look like this:

```
Write low-limit 0 pointer 20hex with F(16)A(0)
Write low-limit 0 value -30000 with F(16)A(1)
Write high-limit 0 pointer 30hex with F(16)A(0)
Write high-limit 0 value  30000  with F(16)A(1)
Write encoder-out 0 pointer 10 hex with F(16)A(0)
Write encoder-out 0 value 0 with F(16)A(1)
Write encoder-mask pointer 8 with F(16)A(0)
Selective set bit 0 in encoder-mask with  F(18)A(1)
Set the new encoder-mask with F(25)A(0)
```

When the encoder values in the STAC has been initialized with the above routine, the knob can be accessed with the KNOB function. The KNOB is a read/write function type 8, the parameter passing in this type of function is similar to that of the BUTTON and BUTS functions, address of the input or output is in register A2 when the function is called, the content of the D1 register indicates whether the function was called as a read or as a write function, if it was read the content of D1 would be +1 while a -1 would indicate a write access. The function is not implemented yet, therefore a complete description will be given in the following:

```
KNOBST   DC.W     KNOBSZ,8,'KN','OB',0,0,0
         DC.L     KNOB

         PROGM                          Macro to reset local variable index
KNOB     ENTER                          Macro to enter a routine
         MOVEA.L  #MMCCAMADR,A3          CAMAC adr. for the MMC module
         PROT                           Macro to disable CAMAC interrupt
         CAMACWR  #$10,F16A0(A3)         Write the encoder-out 0 pointer
                                         to the MMC module
         TST.W    D1                     IF read-access
         BMI      KNOBW                  THEN
         CAMACRD  F0A1(A3),D0              Read encoder-out 0
         CALL     NLZ16,ERR                Convert to floating point
         BRA      KNOBEND                ELSE
KNOBW    CALL     NFIX32,ERR               Convert from floating point
         CAMACWR  D0,F16A1(A3)             Write the value to encoder-out 0
                                         ENDIF
KNOBEND  UNPROT                         Macro to enable CAMAC interrupt
         RET                            Macro to return from a routine
KNOBSZ   EQU      *-KNOBST)/2           Define the size of the function
```

The NODAL macroes is used to make an easy NODAL interface. These macroes defines the stack and the indexes which is used in the acces of parameter and local variables. The format of the stack is shown in figure 19.

The ENTER macro stacks the registers and defines the new A6 register and the new stack-pointer, the A6 register is used as the basis for the access of local variables and input parameters. Before the ENTER macro is executed the stack-pointer has the value SP-old and the address of possible input parameters is located at the stack just under the return address.

The PROGM macro resets the ZXY and the ZYX indexes. ZXY is used to define the local variables index and the stack-pointer value after the execution of the ENTER macro. ZXY is defined as the D0-old address in the PROGM macro. ZYX is used to define the input parameter index for the transfer of variables in the function call. ZYX is defined as the pointer to the first input parameter in the PROGM macro.

DATA, STRING, REAL is macroes used in the definition of local variables, they are assigned to the top of the stack and ZXY is decremented according to their size.

The FPAR macro is used to assign a name to the index for the input parameters, if there is more than one parameter the parameters must be defined in the same order as in the function call.

	LOW MEMORY	
	.	
(SP)	local data	
	.	
	.	
	local data	ZYX=-40+size of variable
-40(A6)	D0-old	ZYX=-40 after PROGM macro
	D1-old	
	D2-old	
	D3-old	
	D4-old	
	D5-old	
	A0-old	
	A1-old	
-8(A6)	A2-old	
-4(A6)	A3-old	
(A6)	A6-old	
(SP-old)	return adr.	
8(A6)	parameter 1	ZXY=8 after PROGM macro
12(A6)	parameter 2	ZXY=12 after first FPAR macro
	.	
	HIGH MEMORY	

Figure 19: The format of the stack after a NODAL routine call

The RET macro restores all the registers, also the stack pointer, and returns from the routine.

The ERRET macro restores the registers and makes an error return.

The CAMACRD and CAMACWR macroes is made to make the CAMAC access easier and to make the programs more readable. The macroes checks the X-respons and the Q-respons, if the X-respons isn't there the routine makes an error return with the error 34, which means HARDWARE ERROR, if the Q-respons isn't there the CAMAC-acces is redone until the Q-respons occurs.

The macroes for the NODAL and the CAMAC interface is listed in appendix 15 and is inserted in the source file before assembling, this is done with the INSERT statement.

9.5 SKNOB(LOW,HIGH)

The SKNOB is call function which is used to set the low and the high endstops. The function is a type 11 function and the assembler program for the function is shown below

```
SKNOBST   DC.W      SKNOBSZ,11,'SK','NO','B',0,$22
          DC.L      SKNOB

          PROGM
          FPAR      LOW
          FPAR      HIGH

SKNOB     ENTER
          MOVE.L    LOW(A6),A2          Get pointer to the LOW value
          MOVE.L    #MMCCAMADR,A3       CAMAC adr. for MMC module
          PROT
          CAMACWR   #$20,F16A0(A3)
          CAMACWR   (A2),F16A1(A3)      Write low-limit
          MOVE.L    HIGH(A6),A2         Get pointer to the HIGH value
          CAMACWR   #$30,F16A0(A3)
          CAMACWR   (A2),F16A1(A3)      Write high-limit
          UNPROT
          RET
```

9.6 HBALL AND VBALL

The tracker ball or mouse is connected to encoder 6 and 7, the horizontal encoder output is connected to encoder 6 and the vertical encoder output is connected to encoder 7.

In the initialization routine the limits for the horizontal movement 0 - 767 shall be written into the low and high limit registers for encoder 6 and the limits for the vertical movements 0 - 575 shall be written into the low and high limit registers for encoder 7. This can be done in a similar way as in the KNOB initialization page 42.

The HBALL and the VBALL functions will also be very similar to the KNOB routine described at page 43, the only differens will be the pointer value and the function name therefore the functions will not be described futher here.

9.7 MCURS

This function which controls the hooking of the cursor to the tracker ball consists of two parts, the function part and the interrupt part in the LAM-handler routine.

The function part shall:

Set the cursor mode according to the value of M
this is done with some CAMAC accesses of the cursor module.
Set the LAM-mask in the STAC for encoder 6 and/or for encoder 7 according to the H value.

The interrupt part which is contained in the LAM-handler routine and called on every LAM from the STAC shall do the following:

```
IF bit6=1  (horizontal movement)
THEN
  Read encod-out 6
  Write this value into the horizontal cursor
     register in the cursor module, and into HCURSVAL
  Clear bit6 in the LAM -status, -request register
     in the STAC
ENDIF
IF bit7=1  (vertical movement)
THEN
  Read encod-out 7
  Write this value into the vertical cursor
     register in the cursor module, and into VCURSVAL
  Clear bit7 in the LAM -status, -reguest reister
     in the STAC
ENDIF
```

9.8 HCURS AND VCURS

These two functions shall when its a read access return the value from HCURSVAL or VCURSVAL, when its a write access the input value is written into HCURSVAL or VCURSVAL and the value is also written into the horizontal or vertical cursor register in the cursor module.

9.9 CURC AND CURL

These two function is similar to the HCURS and VCURS functions the only differens is that the inputs and outputs is in line and column units instead of graphic units, the graphic format is 0-767 horizontal and 0-575 vertical, while the character format is 24-0 line number and 0-64 column number. The relation is shown in the definition on page 38.

Chapter 10

MAN - MACHINE FUNCTIONS IN THE STAC

The Man-Machine functions can also be implemented in the STAC and at the time being the BUTTON an BUTS functions is implemented, The function structure is equivalent to the BUTTON and BUTS functions in the ICC, the only differens is that the functions in the STAC can acces the MMC-variables without any CAMAC access which ofcourse makes the functions simpler. The listing of the program is shown in appendix 17.

Since the NODAL system including the Man-Machine functions is resident in the STAC, the NODAL control system could just as well run in the STAC as in the ICC. When the STAC wants to access a CAMAC module, it must raise a LAM and start and routine in the ICC, which gets the command from the STAC and transmits it to the required module. This communication is already implemented in some senses in the general purpose STAC - ICC communication routines written by Ole Borch, shown in appendix 13.

A complete low cost stand alone control system for a GPIB interface could be implemented in a very simple way, only using a STAC and MMC module with a GPIB driver. One of the serial ports on the STAC is connected to a normal terminal for writing application programs, while the second serial port is connected to display driver which writes the legends on the touch-screen monitor. The GPIB can be used for proces control or data collection.

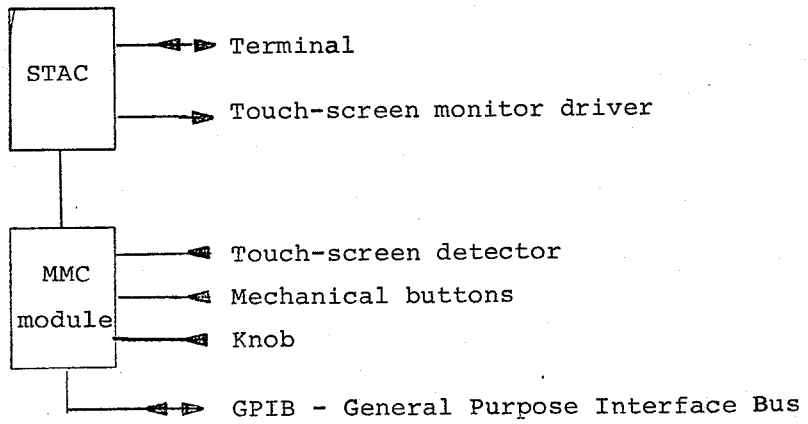


Figure 20: A low cost stand alone control system

Chapter 11

CONCLUSION

A prototype of the Man-to-Machine interface has been working without any problems since the middle of November. For the final implementation a printed circuit board is designed and the drawings for that is shown in appendix 19. The MMC module will go into production this spring and for a start 20 modules have been contracted at Radartronic, Aarhus, Denmark - the supplier of the STAC.

The introduction of the MMC module will make it possible to introduce the touch-terminal control philosophy in a lot of new fields all around CERN and the touch-terminal might be one of the keystones in the control systems which is build in and around the LEP project. One of the dead-lines for the LEP project is August 1985, therefore a complete touch-terminal set'up must be ready very soon so that the persons involved in the soft-ware development can get familiar with the system.

The implementation of the MMC module is the first move toward a VME solution of the Man- Machine communication, but just now there is a strong demand for Man- Machine communication modules in CAMAC. The next implementation of the Man-Machine communication module will probably be an integration of the MMC module into the STAC. This integration will be done in a modular way so that the circuits can be configured to fit on a CAMAC card and on a VME card as sketched in figure 21.

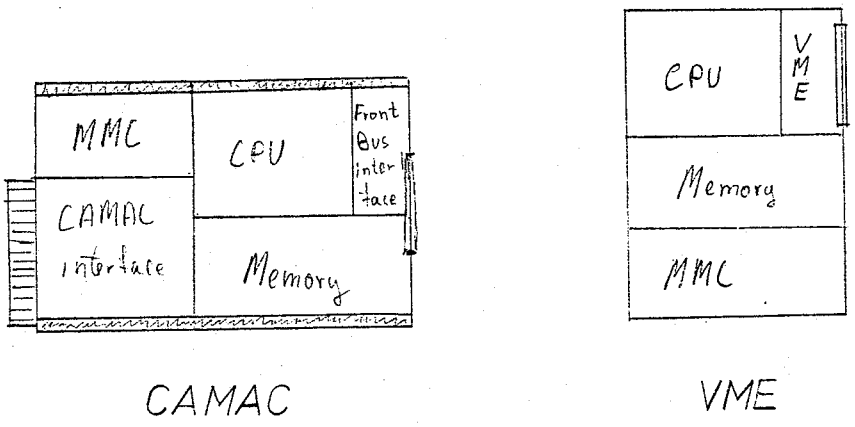


Figure 21: VME and CAMAC implementation of the MMC module Mk II

The idea with the modular hard-ware is to keep as much both hard-ware and soft-ware the same in the CAMAC and in the VME configuration, in this way the development and maintenance of the modules will be easier. To the users of the system there will be no differens at all, since everything will run under the NODAL system.

A cost effective, processor-, language-,bus- independent solution to the touch-terminal, would be to make a complete self-contained unit with a monitor, a touch-panel, and connectors for the mechanical buttons and the encoder inputs. All the communication to the touch-terminal shall go through a serial port just as in a normal terminal. A unit like that, based on a single chip processor with a serial port, eg. Motorola 6801, and a touch-screen based on the charge time detector would be fairly cheap. The unit could also contain a display controller an a complete touch-terminal would be the result.

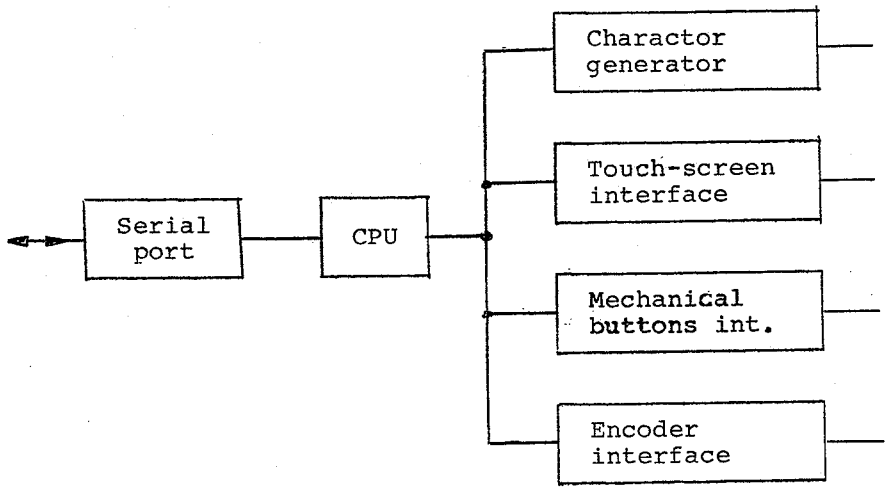


Figure 22: Configuration of low cost touch-terminal

The great advantage with the self-contained touch-terminal is that this touch-terminal can be connected to any system which has a normal terminal port (RS 232). The only thing which has to be done when a touch-terminal is introduced in a new system is to implement the routines which gives the access to the input devices in touch-terminal and the screen from the system, foreexample BASIC functions similar to the NODAL functions described at page 37 and 38 and those functions would ofcourse be system dependent.

All together I feel that the "problemformulering" has been fullfiled. A system has been build and proven workable, allthough all the functions isn't implemented yet. I think that the touch-screen will be a very important device in future controlsystems, but not only control systems can have benefit of a touch-screen. In all kind of systems where a man shall communicate with a computer it will be advantageous to let the computer guide the operator through the system by just presenting a few valid choises at a time. In this way the the operator could reach his goal with just a few logical decisions.
Its time to let the the machine come to the man instead of forcing the man to act as a machine

REFERENCES

1. F. Beck and B. Stumpe. Two devices for operator interaction in the central control of the new CERN accelerator, CERN 73-6 (1973)

2. M. C. Crowley-Milling. The design of the control system for the SPS, CERN 75-20 (1975)

3. F. Beck. The design and construction of a control centre for CERN SPS accelerator, CERN-SPS-CO/76-1 (1976)

4. B. Stumpe. Control room technique for CERN's 400 GeV proton synchrotron, CERN-SPS/aop/co/Note/78-38 (1978)

5. B. Stumpe. Current and future Touch Screen Technology for the M Machine Interface, CERN-SPS/AOP/Note/81-15 (1981)

6. C. Pignard. Un nouveau circuit detecteur pour l'ecran sensitif digital, CERN-PS/LIN/Note/77-1 (1977)

7. Motorola. 16 - Bit Microprocessor, User's Manual, third edition, (1982)

8. Atomic Energy Commitee on Nuclear Instrument Modules. CAMAC A Modular Instrumentation system for Data Handling, EUR 4100e Revised version (1972)

9. B. Stumpe. Hardware description of an MC68000 (Fast) Independent Crate Controller - ICC68000 (SPS 3350), CERN SPS/AOP/Note/81-13 Rev. (1983)

10. A. Rijllart. An MC68000 Stand Alone CAMAC Microprocessor system, CERN-EP/82-202 (1982)

11. H. von Eicken. M68MIL, A cross macro assembler for the Motorola M68000, CERN-DD/80-26 (1980)

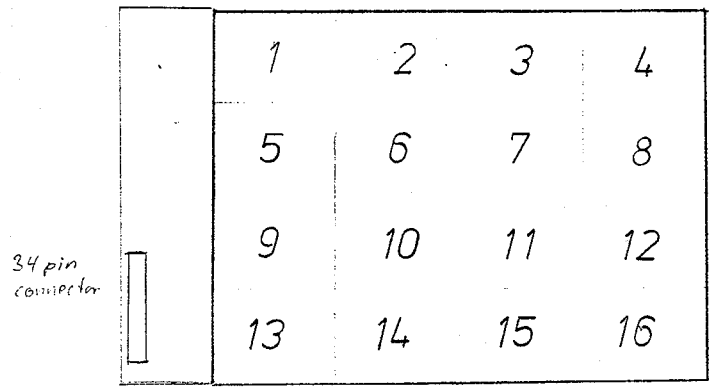
12. M. C. Crowley-Milling and G. C. Sherring. The NODAL system for the SPS, CERN-SPS/78-07 (1978)

13. P. S. Andersen. Programming the SPS-consoles, CERN SPS/ACC/82-1 (1982)

Appendix 1 CAPACITY MEASUREMENTS ON THE TOUCH-SCREEN

Appendix 1

Capcity measurement on the touch-screen



Pin nr.	Button nr.	Untouched value	Scft-touched value	Hard-touched value
2	4	84 pF	102 pF	117 pF
4	3	81 pF	96 pF	117 pF
6	2	78 pF	91 pF	113 pF
8	1	74 pF	84 pF	lo4 pF
10	8	86 pF	95 pF	115 pF
12	7	90 pF	101 pF	122 pF
14	6	80 pF	91 pF	117 pF
16	5	67 pF	79 pF	97 pF
18	12	89 pF	99 pF	121 pF
20	11	85 pF	94 pF	137 pF
22	10	84 pF	97 pF	117 pF
24	9	67 pF	77 pF	132 pF
26	16	85 pF	95 pF	117 pF
28	15	81 pF	92 pF	117 pF
30	14	78 pF	89 pF	127 pF
32	13	66 pF	85 pF	103 pF

pin 1,3,5,...,31,33 connected to common.

The measurement was caried out on ECD C-meter, CERN-SPS 0162203

The measurement was done through a scotchflex cable with a capacity of 43 pF, this value is subtracted in the values in the table.

Appendix 2 ADC MEASUREMENTS ON THE TUNED TOUCH SCREEN DETECTOR

PIN / Button	Without ADC: Untouched Volt Clean	Without ADC: Untouched Volt Dirty	Without ADC: Touched Volt	Without ADC: Ripple Volt pp	With ADC: Untouched Volt Clean	With ADC: Untouched Volt Dirty	With ADC: Untouched Hex ADC	With ADC: Touched Volt	With ADC: Touched Hex ADC	With ADC: Ripple Volt	With ADC and 100nF load: Untouched ADC-value Clean	With ADC and 100nF load: Untouched ADC-value Dirty	With ADC and 100nF load: Touched Volt	With ADC and 100nF load: Touched ADC	With ADC and 100nF load: Ripple Volt
1 / 16	9,0	6,0	1,7	0,4	3,6	3,0	50/40	1,7v	22	0,4	A6	D2		50	0
2 / 15	7,9	5,0	1,7v	0,4	3,6	3,1	4C/40	1,7v	23	0,4	B0	D1		4D	0
3 / 14	7,0	5,5	1,8v	0,4	3,4	3,1	48/40	1,8v	23	0,4	B6	C4		59	0
4 / 13	7,8	5,3	1,8v	0,4	3,7	3,2	50/41	1,8v	24	0,4	B9	DF	1,5v	4C	0
5 / 12	7,5	5,5	1,8v	0,4	3,4	2,6	47/38	1,6v	21	0,4	BA	C5		4D	0
6 / 11	7,5	4,0	1,8v	0,4	3,7	3,1	50/40	1,6v	22	0,4	9D	D9		54	0
7 / 10	7,2	4,2	1,8v	0,4	3,6	2,6	50/36	1,8	24	0,4	A5	DA		5E	0
8 / 9	7,5	5,0	1,8v	0,4	3,7	2,7	50/38	1,8	24	0,4	A3	DD		53	0
9 / 8	6,8	3,5	1,8v	0,4	2,9	2,4	40/30	1,7	22	0,4	8C	AB		51	0
10 / 7	6,2	3,5	1,8v	0,3	2,9	2,3	43/30	1,8	23	0,4	88	AF	1,9v	65	0
11 / 6	6,8	4,0	1,8v	0,4	3,4	2,5	47/32	1,7	21	0,4	8F	C7		62	0
12 / 5	7,2	5,5	1,8v	0,4	3,4	3,0	47/34	1,6	21	0,4	95	C9		5F	0
13 / 4	7,5	5,5	1,7v	0,4	3,1	2,2	40/2F	1,7	22	0,4	9E	B4		60	0
14 / 3	7,1	11,2	1,7v	0,4	3,2	2,5	43/34	1,7	21	0,4	94	C0		63	0
15 / 2	7,2	4,5	1,9v	0,4	3,4	2,8	46/2[illegible]	1,6	21	0,4	97	C8		57	0
16 / 1	7,6	5,0	2,0V	0,4	3,5	3,0	4B/34	1,7	23	0,4	AA	D4		61	0

Osc 4Vpp

ADC = AD 7581 Oscilator = 456 kHz Osc 5Vpp 8 Vcc Touchscreen = Twind 6Vpp Date = 28 juli 93

Appendix 3 ADC MEASUREMENTS ON THE VOLTAGE DIVIDER DETECTOR

PIN	Without ADC			With ADC					With ADC and 100nF load				
	Untouched Volt	Touched Volt	Ripple Volt	Untouched Volt	Untouched Hex ADC	Touched Volt	Touched ADC	Ripple Volt	Untouched Volt	Untouched ADC	Touched Volt	Touched ADC	Ripple Volt
1 16	2,9	2,3	0,05	1,8	5C	1,5	48	0,2	1,8	5D	1,4	47	—
2	2,8	2,2	0,05	1,8	58	1,45	48	0,2	1,8	5A	1,4	45	—
3	2,8	2,3	0,05	1,8	58	1,4	47	0,2	1,8	5A	1,4	46	—
4	2,7	2,2	0,05	1,7	56	1,35	44	0,2	1,7	57	1,3	43	—
5 12	3,0	2,3	0,05	1,9	60	1,55	4E	0,2	1,85	60	1,4	4B	—
6	2,8	2,3	0,05	1,8	59	1,4	47	0,2	1,8	5B	1,4	48	—
7	2,6	2,1	0,05	1,7	54	1,35	43	0,2	1,7	55	1,3	42	—
8	2,7	2,2	0,05	1,7	56	1,35	43	0,2	1,7	57	1,4	46	—
9 8	3,0	2,4	0,05	1,85	5C	1,55	4E	0,2	1,8	5E	1,4	4B	—
10	2,8	2,2	0,05	1,7	56	1,35	43	0,2	1,75	58	1,35	45	—
11	2,7	2,2	0,1	1,7	55	1,3	42	0,2	1,7	57	1,35	45	—
12 5	2,6	2,1	0,1	1,65	53	1,3	43	0,2	1,7	55	1,35	44	—
13 4	3,0	2,4	0,1	1,85	60	1,50	46	0,2	1,7	5E	1,4	4A	—
14	2,8	2,3	0,1	1,75	58	1,45	46	0,2	1,7	5A	1,4	4A	—
15	2,7	2,2	0,1	1,7	57	1,35	45	0,2	1,7	58	1,4	48	—
16	2,7	2,2	0,1	1,7	55	1,35	44	0,2	1,65	56	1,4	47	—

ADC = AD 7581 Oscilator = 456 kHz 8 Vcc Touchscreen = Capacity divider Date = 28 July 83

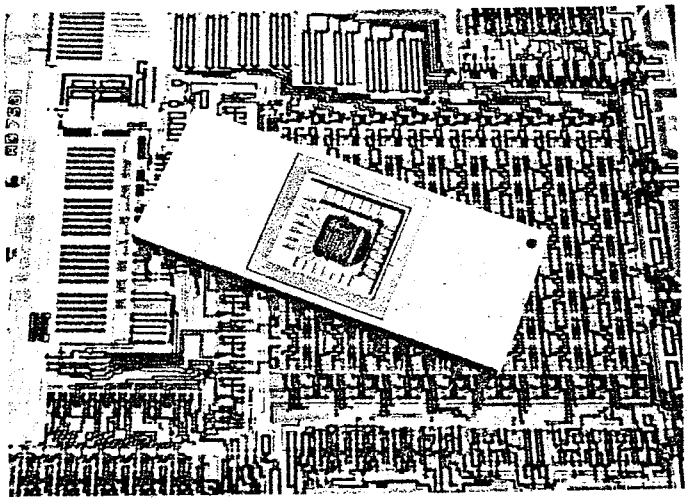
Appendix 4 DATA SHEET OF THE AD7581 A/D CONVERTER

CMOS µP Compatible 8-Bit 8-Channel DAS

FEATURES

- 8-Bit Resolution
- On-Chip 8 X 8 Dual-Port Memory
- No Missed Codes Over Full Temperature Range
- Interfaces Directly to Z80/8085/6800
- CMOS, TTL Compatible Digital Inputs
- Three-State Data Drivers
- Ratiometric Capability
- Single +5V Supply
- Interleaved DMA Operation
- Fast Conversion
- A/D Process Totally Transparent to µP
- Low Cost

Second source: Micro Power Systems MP 7581

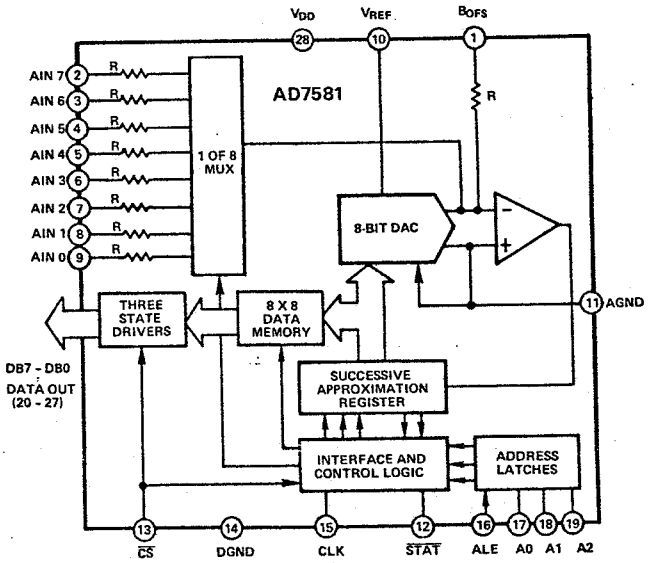


GENERAL DESCRIPTION

The AD7581 is a microprocessor compatible 8 bit, 8 channel, memory buffered, data-acquisition system on a monolithic CMOS chip. It consists of an 8 bit successive approximation A/D converter, an 8 channel multiplexer, 8 X 8 dual-port RAM, three-state DATA drivers (for interface), address latches and microprocessor compatible control logic. The device interfaces directly to 8080, 8085, Z80, 6800 and other microprocessor systems.

The successive approximation conversion takes place on a continuous, channel sequencing, basis using microprocessor control signals for the clock. Data is automatically transferred to its proper location in the 8 X 8 dual-port RAM at the end of each conversion. When under microprocessor control, a READ DATA operation is allowed at any time for any channel since on-chip logic provides interleaved DMA. The facility to latch the address inputs ($A_0 - A_2$) with ALE enables the AD7581 to interface with µP systems which feature either shared or separate address and data buses.

FUNCTIONAL DIAGRAM



ORDERING INFORMATION

Differential Nonlinearity	Temperature Range and Package: Plastic 0 to +70°C	Ceramic -25°C to +85°C
±1 7/8LSB	AD7581JN	AD7581AD
±7/8LSB	AD7581KN	AD7581BD
±3/4LSB	AD7581LN	AD7581CD

ANALOG DEVICES
9, rue de Berne
1201 GENEVE
Tél. 022 / 31 57 60
Télex 289 096
Tél. 01 / 840 07 77

Information furnished by Analog Devices is believed to be accurate and reliable. However, no responsibility is assumed by Analog Devices for its use; nor for any infringements of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of Analog Devices.

P.O. Box 280; Norwood, Massachusetts 02062 U.S.A.
Telex: 924491 Cables: ANALOG NORWOODMASS

DC SPECIFICATIONS (V_{DD} = +5V, V_{REF} = -10V, Unipolar Operation, unless otherwise stated)

Parameter	Version[1]	Typical at +25°C	Limit Over Temperature	Units	Conditions/Comments
ACCURACY					
Resolution	All	8	8	Bits	
Relative Accuracy	JN, AD	±1 7/8	±1 7/8 max	LSB	
	KN, BD	±3/4	±3/4 max	LSB	
	LN, CD	±1/2	±1/2 max	LSB	
Differential Nonlinearity	JN, AD	±1 7/8	±1 7/8 max	LSB	
	KN, BD	±7/8	±7/8 max	LSB	
	LN, CD	±3/4	±3/4 max	LSB	
Offset Error[2]	JN, AD	200	200 max	mV	Adjustable to zero, see Figure 7a.
	KN, BD	80	80 max	mV	
	LN, CD	50	50 max	mV	
Gain Error					
Worst Channel	JN, AD	±3	±6 max	LSB	Adjustable to zero, see Figure 7a.
	KN, BD	±2	±4 max	LSB	Gain Error is Measured After Offset
	LN, CD	±1	±2 max	LSB	Calibration. Max Full Scale Change for Any Channel from +25°C to T_{min} or T_{max} is ±2LSB.
Gain Match Between Channels	JN, AD	2	3 max	LSB	Adjustable to zero, see Figure 7a.
	KN, BD	1 1/2	2 max	LSB	
	LN, CD	1	1 max	LSB	
B_{OFS} Gain Error	All	-2 1/2	—	LSB	
ANALOG INPUTS					
Input Resistance					
At V_{REF} (pin 10)	All	10/20/30	10/20/30	kΩ min/typ/max	
At B_{OFS} (pin 1)[3]	All	10/20/30	10/20/30	kΩ min/typ/max	
At Any Analog Input (pins 2-9)	All	10/20/30	10/20/30	kΩ min/typ/max	
V_{REF} (For Specified Performance)	All	-10	-10	V	±5%
V_{REF} Range[4]	All	-5 to -15	-5 to -15	V	
Nominal Analog Input Range					
Unipolar Mode	All	0 to $+V_{REF}$,	0 to $+V_{REF}$	V	See Figure 7 and 8.
		0 to $-V_{REF}$	0 to $-V_{REF}$	V	
Bipolar Mode	All	$-VB_{OFS} \leq VA_{IN} \leq \lvert V_{REF} \rvert - VB_{OFS}$			See Figure 9
DIGITAL INPUTS					
$\overline{CS}$ (pin 13), ALE (pin 16), A_0 - A_2 (pins 17-19)					
CLK (pin 15)					
V_{INH} Logic HIGH Input Voltage	All	+2.2	+2.4 min	V	
V_{INL} Logic LOW Input Voltage	All	+0.4	+0.8 max	V	
I_{IN} Input Current	All	0.01	1 max	μA	V_{IN} = 0V, V_{DD}
C_{IN} Input Capacitance[5]	All	4	5 max	pF	
DIGITAL OUTPUTS					
$\overline{STAT}$ (pin 12), DB_7 to DB_0 (pins 20-27)					
V_{OH} Output HIGH Voltage	All	+4.8	+4.5 min	V	I_{SOURCE} = 40μA
V_{OL} Output LOW Voltage	All	+0.4	+0.6 max	V	I_{SINK} = 1.6mA
I_{LKG} DB_7 to DB_0 Floating State Leakage	All	0.3	10 max	μA	
Floating State Output Capacitance (DB_7 - DB_0)	All	5	10 max	pF	V_{OUT} = 0V to V_{DD}
Output Code	All	Unipolar Binary Figure 7			
		Complementary Binary Figure 8			
		Offset Binary Figure 9			
POWER REQUIREMENTS					
V_{DD}	All	+5	+5	V	
I_{DD} - Static	All	3 typ	5 max	mA	
I_{DD} - Dynamic	All	3 typ	8 max	mA	f_{CLK} = 1MHz

Notes:
[1] Temperature range as follows: JN, KN, LN (0 to +70°C), AD, BD, CD (-25°C to +85°C).
[2] Typical offset temperature coefficient is ±150μV/°C.
[3] R_{BOFS}/R_{AIN} (0-7) mismatch causes transfer function rotation about positive full scale. The effect is an offset and a gain term when using the circuits of Figure 8a, page 6 and Figure 9a, page 7.
[4] Typical value, not guaranteed or subject to test.
[5] Guaranteed but not tested.
[6] Typical change in B_{OFS} gain from +25°C to T_{min} to T_{max} is ±2 LSB's.

Specifications subject to change without notice.

AC SPECIFICATIONS (V_{DD} = +5V, V_{REF} = -10V, Unipolar Operation, unless otherwise noted)

Symbol	Specification	Typical at +25°C	Limit Over Temperature	Units	Conditions
t_H	ALE pulse width	50	80 min	ns	See "Switching Terminology"
t_{ALS}	Address valid to latch set-up time	45	70 min	ns	on page 5
t_{ALH}	Address valid to latch hold time	10	20 min	ns	
t_{LCS}	Address latch to $\overline{CS}$ set-up time	10	20 min	ns	
t_{ACC}	$\overline{CS}$ to output propagation delay	200	250 max	ns	C_L = 100pF
t_{CW}	$\overline{CS}$ pulse width	250	280 min	ns	
t_{CF}	$\overline{CS}$ to output float propagation delay	50	80 max	ns	
t_{CLZ}	$\overline{CS}$ to low impedance bus	100	150 max	ns	
f_{CLK}	Clock frequency for stated accuracy	1600	1200 max	kHz	

ABSOLUTE MAXIMUM RATINGS

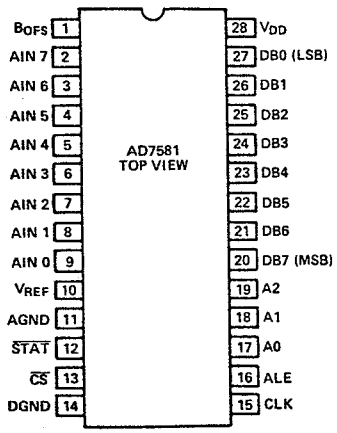
V_{DD} to AGND . . . . . +7V
V_{DD} to DGND . . . . . +7V
AGND to DGND . . . . . -0.3V, V_{DD}
Digital Input Voltage to DGND
(pins 13, 16–19) . . . . . +15V
Digital Output Voltage to DGND
(pins 12, 20–27) . . . . . -0.3V, V_{DD}
CLK (pin 15) input voltage to DGND . . . . . +15V
V_{REF} (pin 10) to AGND . . . . . ±25V
V_{BOFS} (pin 1) to AGND . . . . . ±17V
AIN (0–7) (pin 9–2) . . . . . ±17V
Operating Temperature Range
JN, KN, LN . . . . . 0 to +70°C
AD, BD, CD . . . . . -25°C to +85°C
Storage Temperature . . . . . -65°C to +150°C
Lead Temperature (Soldering, 10 secs) . . . . . +300°C

Power Dissipation (Package)
Plastic (Suffix N)
to +50°C 1200mW
Derate above +70°C by . . . . . 12mW/°C
Ceramic (Suffix D)
to +50°C 1000mW
Derate above +50°C by . . . . . 10mW/°C

CAUTION:

ESD (Electro-Static-Discharge) sensitive device. The digital control inputs are zener protected; however, permanent damage may occur on unconnected devices subject to high energy electrostatic fields. Unused devices must be stored in conductive foam or shunts. The foam should be discharged to the destination socket before devices are removed.

PIN CONFIGURATION



GENERAL CIRCUIT INFORMATION

BASIC CIRCUIT DESCRIPTION

The AD7581 accepts eight analog inputs and sequentially converts each input into an eight-bit binary word using the successive approximation technique. The conversion results are stored in an 8 X 8 bit dual-port RAM. The device runs either directly from the microprocessor clock (in 6800 type systems) or from some suitable signal (e.g. ALE in 8085 type systems). Most applications require only a −10V reference and a +5V supply. Start-up logic is included on the device to establish the correct sequences on power-up. A maximum of 800 clock pulses are required for this period. Figure 1 shows the AD7581 functional diagram.

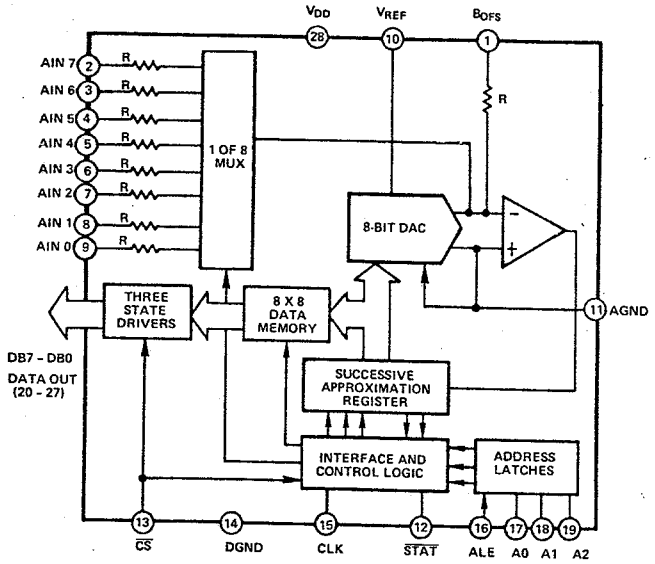


Figure 1. AD7581 Functional Diagram

Conversion of a single channel requires 80 input clock periods and a complete scan through all channels requires 640 input clock periods. When a channel conversion is complete, the successive approximation register contents are loaded into the proper channel location of the 8 X 8 RAM. At this time a status signal output, $\overline{STAT}$ (pin 12), gives a short negative going pulse (8 clock periods). This negative going $\overline{STAT}$ pulse is extended to 72 clock periods when channel 1 conversion is complete. An external pulse width detector connected to the status pin can be used to derive conversion-related timing signals for microprocessor interrupts (see Channel Identification opposite page). Simultaneous with $\overline{STAT}$ going low, the MUX address is decremented. Eight clock periods later the next conversion is started.

Automatic interleaved DMA is provided by on-chip logic to ensure that memory updates take place at instants when the microprocessor is not addressing memory. Memory locations are addressed by A_0, A_1 and A_2. This address may be latched by ALE for systems which feature a multiplexed address/data bus or alternatively, for systems which have separate address and data buses, the address latches can be made transparent by tying ALE (pin 16) HIGH. $\overline{CS}$ (pin 13) activates three-state buffers to place addressed data on the $DB_0 - DB_7$ data output pins.

A/D CIRCUIT DETAILS

In the successive approximation technique, successive bits, starting with the most significant bit (DB_7), are applied to the input of the D/A converter. The DAC output is then compared to the unknown analog input voltage, A_{IN} (n), using a comparator. If the DAC output is greater than $A_{IN}(n)$, the data latch for the trial bit is reset to zero, and the next smaller data bit is tried. If the DAC output is less than $A_{IN}(n)$, the trial data bit stays in the "1" state, and the next smaller data bit is tried. Each successive bit is tried, compared to $A_{IN}(n)$, and set or reset in this manner until the least significant bit (DB_0) decision is made. The successive approximation register now contains a valid digital representation of $A_{IN}(n)$. $A_{IN}(n)$ is assumed to be stable during conversion.

The current weighting D/A converter is a precision multiplying DAC. Figure 2 shows the functional diagram of the DAC as used in the AD7581. It consists of a precision Silicon Chromium thin film R/2R ladder network and 8 N-channel MOSFET switches operated in single-pole-double-throw.

The currents in each 2R shunt arm are binarily weighted i.e., the current in the MSB arm is V_{REF} divided by 2R, in the second arm is V_{REF} divided by 4R, etc. Depending on the D/A logic input (A/D output) from the successive approximation register, the current in the individual shunt arms is steered either to A_{GND} or to the comparator summing point.

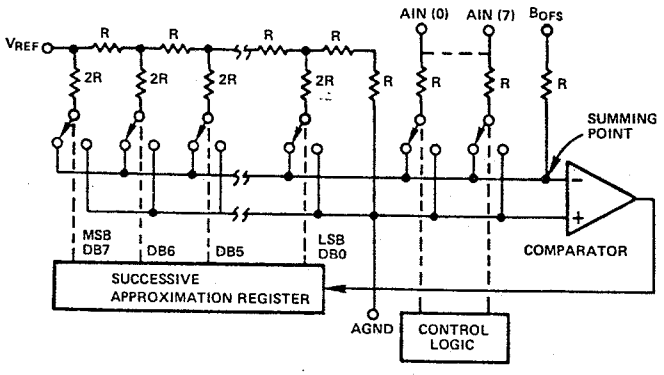


Figure 2. D/A Converter as Used in AD7581

TIMING AND CONTROL OF THE AD7581

CHANNEL SELECTION

Table 1 shows the truth table for the address inputs. The input address is latched when ALE goes LOW. When ALE is HIGH the address input latch is transparent.

A2	A1	A0	ALE	Channel Data To Be Read
0	0	0	1	Channel 0
0	0	1	1	Channel 1
0	1	0	1	Channel 2
0	1	1	1	Channel 3
1	0	0	1	Channel 4
1	0	1	1	Channel 5
1	1	0	1	Channel 6
1	1	1	1	Channel 7

Table 1. Channel Selection Truth Table

TIMING AND CONTROL

A typical timing diagram is shown in Figure 3. When $\overline{CS}$ is HIGH, the three-state data drivers are in the high-impedance state. When $\overline{CS}$ goes LOW the data drivers switch to the low-impedance state (i.e., low impedance to DGND or to V_{DD}). Output data is valid after time t_{ACC}.

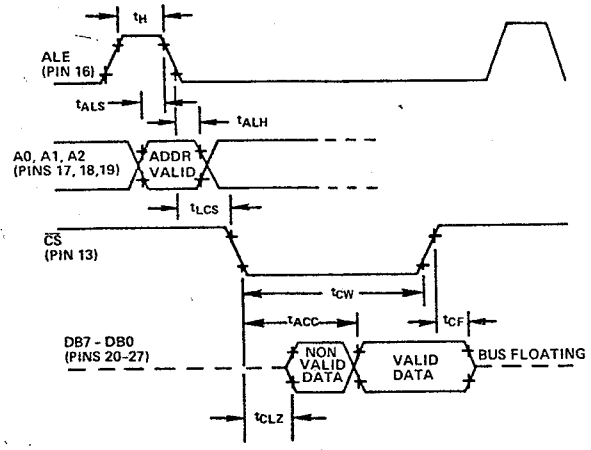


Figure 3. Timing Diagram for the AD7581

SWITCHING TERMINOLOGY

t_H: ALE pulse width requirement.
t_{ALH}: Address Valid to latch hold time.
t_{ALS}: Address Valid to latch set-up time.
t_{LCS}: Address latch to Chip Select set-up time.
t_{CW}: Chip Select pulse width requirement.
t_{ACC}: Chip Select to valid data propagation delay.
t_{CF}: Chip Select to output data float propagation delay.
t_{CLZ}: Chip Select to low impedance data bus.

CHANNEL IDENTIFICATION

In some real-time applications, it may be necessary to provide an interrupt signal when a particular channel receives updated data. To achieve this, it is necessary to identify which channel is currently under conversion. The $\overline{STAT}$ output provides an identifying signal by staying low for an additional 64 clock periods over normal (8 clock periods) when channel 0 is active. This is illustrated in Figure 4. Memory update takes place on a rising edge of a clock pulse. This occurs 6 clock periods before $\overline{STAT}$ goes low.

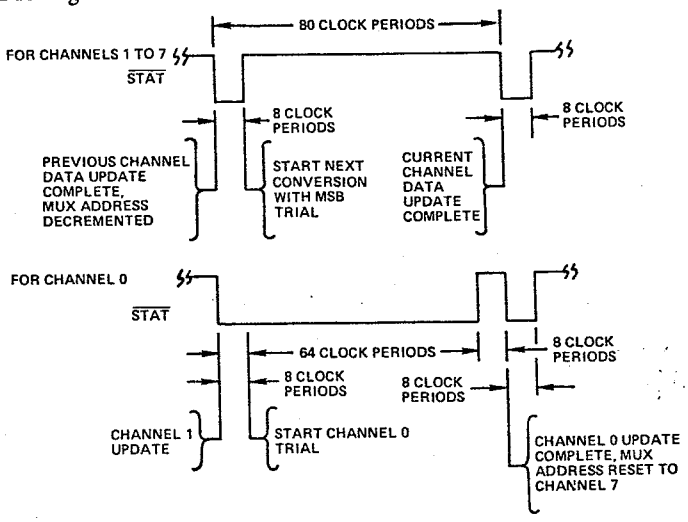


Figure 4. $\overline{STAT}$ Output for Channel Identification

One simple circuit using the $\overline{STAT}$ output is shown in Figure 5. The time constant RC is chosen such that X_2 ignores the normal STAT low pulse width (8 clock periods wide) but responds to the much wider $\overline{STAT}$ low pulse width (72 clock periods wide) occurring during channel 0 conversion. Typically for a 1μs clock period C = 0.022μF, R = 1.8kΩ.

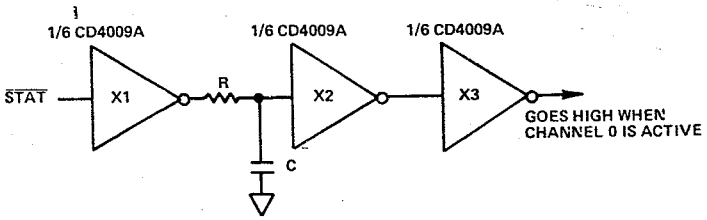


Figure 5. Hardware Channel Identification

Another possibility is to use the microprocessor to interrogate the $\overline{STAT}$ output and hence determine channel identity. A simple routine is shown in Figure 6.

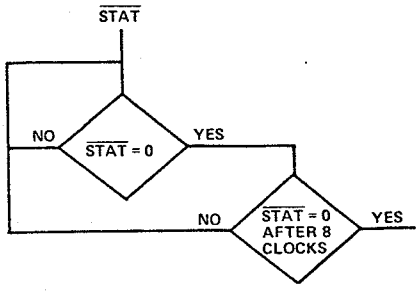


Figure 6. Software Channel Identification

OPERATING THE AD7581

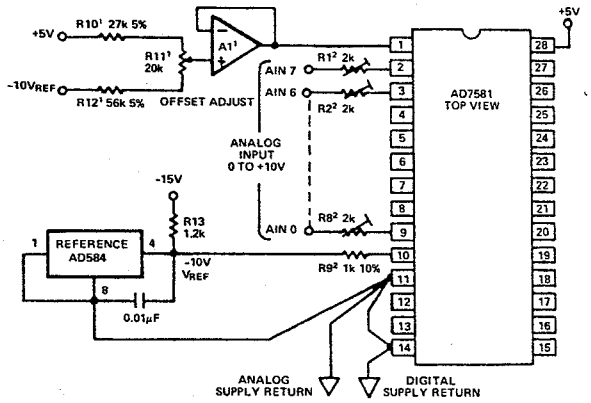
UNIPOLAR BINARY OPERATION

Figures 7a and 7b show the analog circuit connections and typical transfer characteristic for unipolar operation (0V to +10V). An AD584 is used for the -10V reference. Calibration is as follows (device clocked i.e., continuous conversions);

OFFSET:

Comparator offset is trimmed out via the bipolar offset pin B_{OFS}. R10, R11 and R12 comprise a simple voltage tap buffered by A1 and feeding into B_{OFS}.

1. Since comparator offset will be the same regardless of which channel is active, take A_0, A_1 and A_2 LOW and and exercise ALE to latch the address.
2. With AIN 0 = 19.5mV (1/2LSB) adjust R11, i.e., the offset voltage on B_{OFS}, until $DB_7 - DB_1$ are LOW and DB_0 (LSB) flickers.



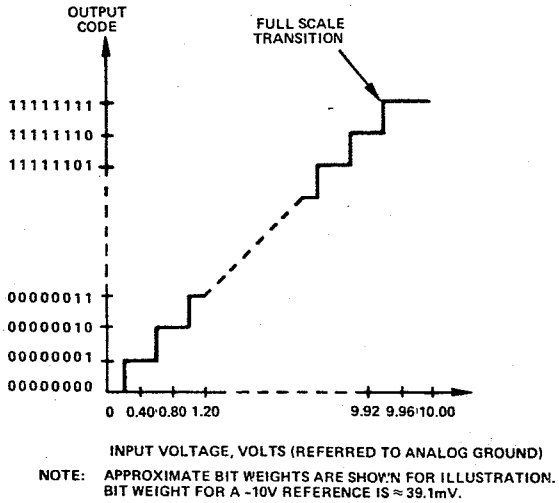
NOTES:
¹A1, R10, R11 AND R12 CAN BE OMITTED IF OFFSET TRIM IS NOT REQUIRED AND B_{OFS} CAN BE TIED TO AGND.
²R1-R8 AND R9 CAN BE OMITTED IF GAIN TRIM IS NOT REQUIRED.

Figure 7a. AD7581 Unipolar (0V to +10V) Operation (Output Code is Straight Binary)

GAIN (FULL SCALE)

In many applications gain adjustment is not required thus removing the need for trimmers in the analog channels. For channels requiring gain trim, the following procedure is recommended. Offset adjustment must be performed before gain adjustment.

1. Apply +9.941V (FS - 1 1/2LSB) to all input channels AIN (0-7).



NOTE: APPROXIMATE BIT WEIGHTS ARE SHOWN FOR ILLUSTRATION. BIT WEIGHT FOR A -10V REFERENCE IS ≈ 39.1mV.

Figure 7b. Transfer Characteristic for Unipolar Circuit of Figure 7a

2. Select required channel n via A_0, A_1, A_2 and latch the Address using ALE.
3. Adjust trimmer RN of selected channel until $DB_7 - DB_1$ are HIGH and the LSB (DB_0) flickers.
4. Select next channel requiring gain trim and repeat steps 2 and 3.

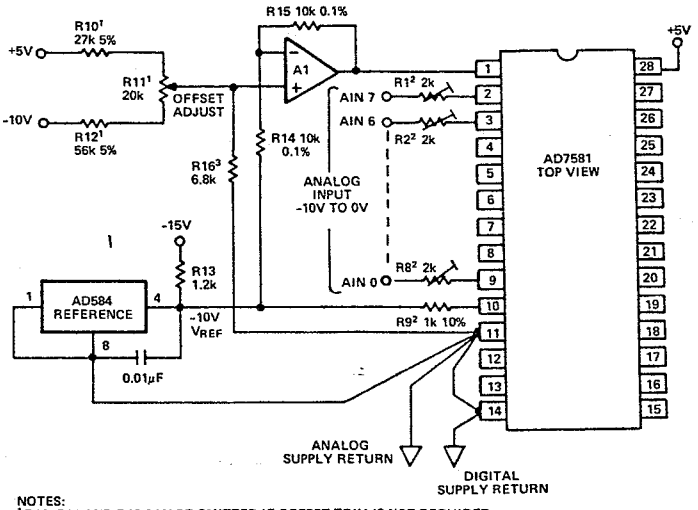
UNIPOLAR (COMPLEMENTARY BINARY) OPERATION

Figures 8a and 8b show the analog circuit connections and typical transfer characteristic for unipolar (complementary binary) operation.
Calibration is as follows (continuous conversions);

OFFSET:

Comparator offset is trimmed out via the bipolar offset pin B_{OFS}. R10, R11 and R12 comprise a simple voltage tap buffered by A1 and feeding into B_{OFS}.

1. Since comparator offset will be the same regardless of which channel is active, take A_0, A_1 and A_2 LOW and exercise ALE to latch the address.
2. With AIN 0 = -9.98V (-FS + 1/2LSB) adjust R11, i.e., the offset voltage on B_{OFS}, until $DB_7 - DB_1$ are LOW and the LSB (DB_0) flickers.



NOTES:
¹R10, R11 AND R12 CAN BE OMITTED IF OFFSET TRIM IS NOT REQUIRED.
²R1 - R8 AND R9 CAN BE OMITTED IF GAIN TRIM IS NOT REQUIRED.
³R16/R10/R12 = 5kΩ. IF R10, R11 AND R12 ARE NOT USED, MAKE R16 = 5kΩ.

Figure 8a. AD7581 (0V to -10V) Operation (Output Code is Complementary Binary)

GAIN (FULL SCALE)

In many applications gain adjustment is not required thus removing the need for trimmers in the analog channels. For channels requiring gain trim, the following procedure is recommended. Offset adjustment must be performed before gain adjustment.

1) Apply -58.6mV (1 1/2LSB) to all input channels AIN (0-7).
2) Select required channel n via A_0, A_1, A_2 and exercise ALE to latch the address.
3) Adjust trimmer RN of selected channel until DB_7 - DB1 are HIGH and the LSB (DB_0) flickers.
4) Select next channel requiring gain trim and repeat step 2 and 3.

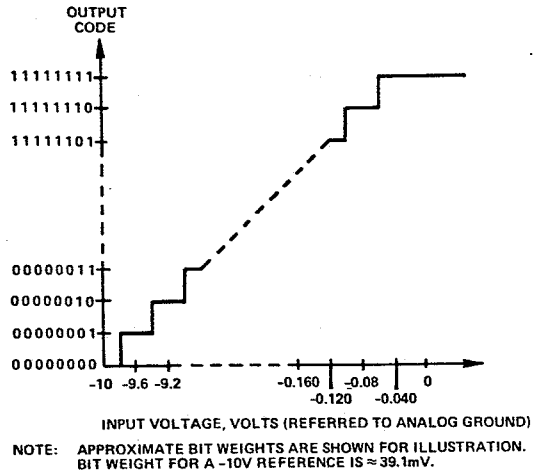
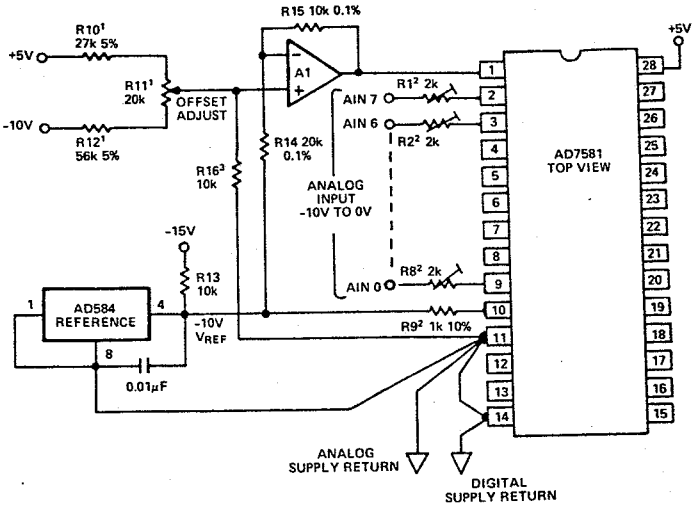


Figure 8b. Transfer characteristic for Unipolar Circuit of Figure 8a



NOTES:
[1] R10, R11 AND R12 CAN BE OMITTED IF OFFSET TRIM IS NOT REQUIRED.
[2] R1 - R8 AND R9 CAN BE OMITTED IF GAIN TRIM IS NOT REQUIRED.
[3] R16/R10/R12 = 6.8kΩ. IF R10, R11 AND R12 ARE NOT USED, MAKE R16 = 6.8kΩ.

Figure 9a. AD7581 Bipolar (-5V to +5V) Operation (Output Code is Offset Binary)

BIPOLAR (OFFSET BINARY) OPERATION

Figures 9a and 9b illustrate the analog circuitry and transfer characteristic for ±5V bipolar operation. Output coding is offset binary. Comparator offset correction is again applied to the B_{OFS} pin.

Calibration is as follows (continuous conversions);

OFFSET:

1. Apply -4.980V (-F.S. + 1/2LSB) to all input channels, AIN (0-7).
2. Trim R11 of the comparator offset circuit until $DB_7 - DB_1$ are LOW and the LSB (DB_0) flickers.

GAIN (FULL SCALE)

1. Apply +4.984V (+F.S. - 1/2LSB) to all input channels, AIN (0-7).
2. Select required channel n via A_0, A_1, A_2, and latch the address using ALE.
3. Adjust trimmer RN of selected channel until $DB_7 - DB_1$ are HIGH and the LSB (DB_0) flickers.
4. Select next channel requiring gain trim and repeat steps 2 add 3.
5. Apply 0V to each gain-trimmed channel. If the ADC output code does not flicker between 01111111 and 10000000 repeat the calibration procedure.

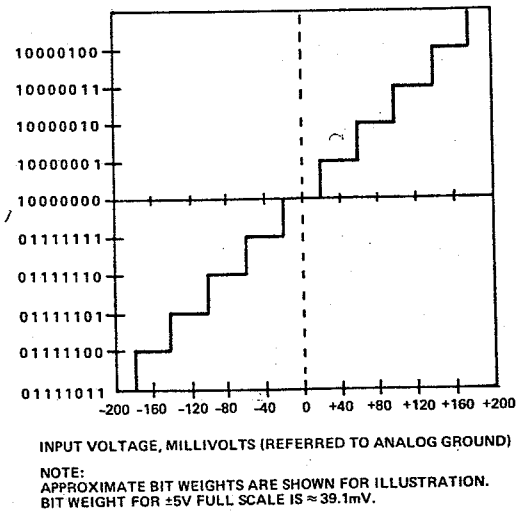


Figure 9b. Transfer Characteristic Around Major Carry for Bipolar Circuit of Figure 9a

INTERFACING THE AD7581

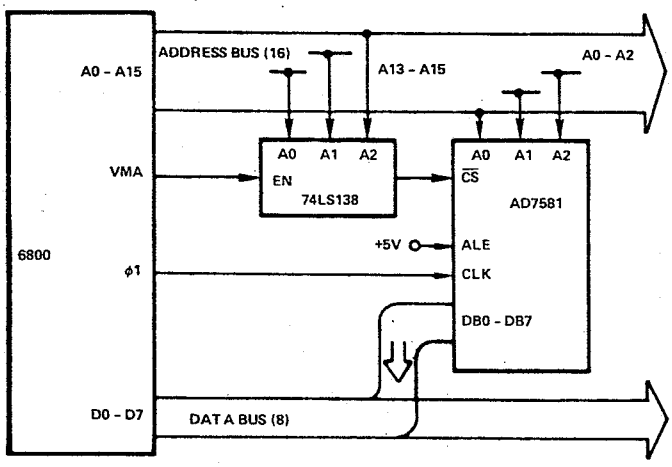


Figure 10. AD7581/6800 Interface

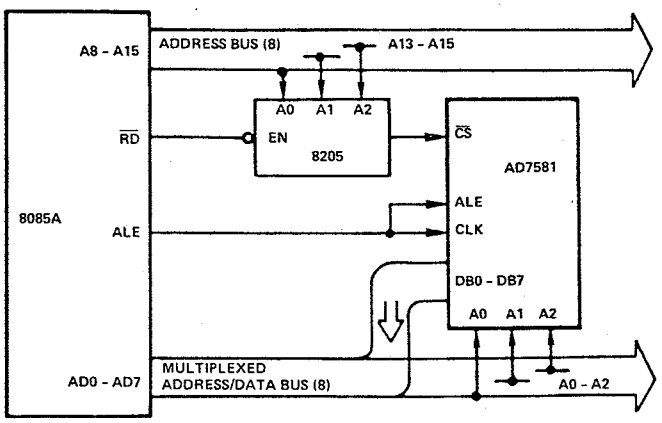


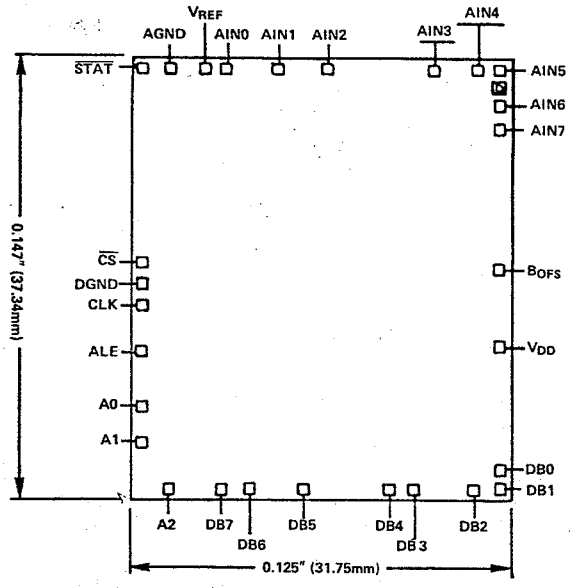
Figure 11. AD7581/8085 Interface

NOTES:

1. ANALOG AND DIGITAL GROUND
   It is recommended that A_{GND} and D_{GND} be connected locally to prevent the possibility of injecting noise into the AD7581. In systems where the A_{GND} - D_{GND} intertie is not local, connect back-to-back diodes (IN914 or equivalent) between the AD7581 A_{GND} and D_{GND} pins.
2. LOGIC DEGLITCHING IN μP APPLICATIONS
   Unspecified states on the address bus (due to different rise and fall times on the address bus) can cause glitches at the AD7581 $\overline{CS}$ terminal. These glitches can cause unwanted reads. The best way to avoid glitches is to gate the address decoding logic, e.g., with RD (8080, $\overline{RD}$ (8085) or VMA (6800).

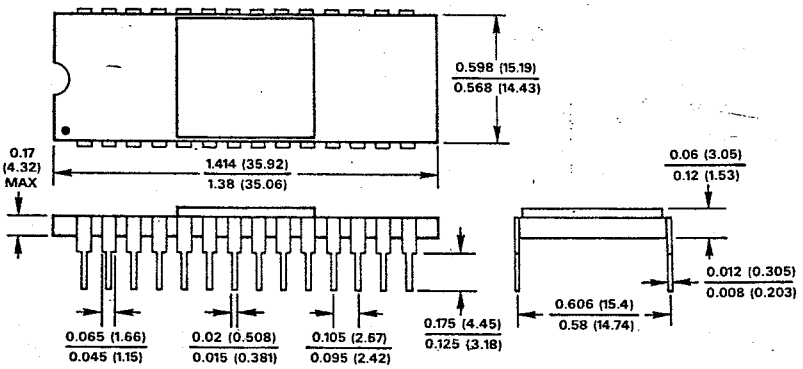
MECHANICAL INFORMATION

BONDING DIAGRAM



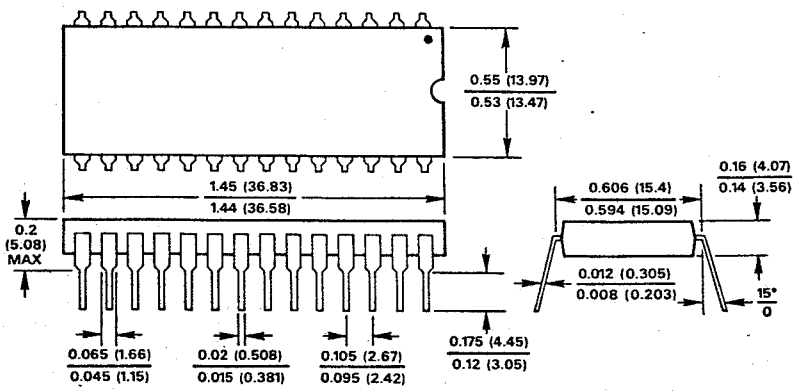
OUTLINE DIMENSIONS

Dimensions shown in inches and (mm).



LEAD NO. 1 IDENTIFIED BY DOT OR NOTCH
LEADS ARE GOLD PLATED (50 MICROINCHES MIN) OVER NICKEL (100 MICROINCHES NOMINAL)
BASE MATERIAL IS KOVAR OR ALLOY 42
CAVITY LID IS ELECTRICALLY ISOLATED

28 PIN CERAMIC DIP



LEAD NO. 1 IDENTIFIED BY DOT OR NOTCH
LEADS ARE SOLDER PLATED KOVAR OR ALLOY 42

28 PIN PLASTIC DIP

Appendix 5 DATA SHEET OF THE TRACKER BALL

Disc Instruments

STEUERKUGEL
(TRACKBALL)
Serie 200

Datenblatt

Mit dem Trackball 200 steht eine preisgünstige Alternative zum «joy stick» zur Verfügung. Der Trackball 200 liefert in digitaler Form Rechtecksignale im TTL-Pegel. Dadurch entfällt die bei anderen Verfahren notwendige Analogdigital-Umwandlung. Das ermöglicht eine kompaktere Bauweise.

Die Serie 200 ist ein ideales Eingabegerät z.B. für:
- rechnergesteuerte Zeichengeräte
- Tomographen
- Radaranlagen
- kartographische Darstellungen
- CAD/CAM
- Videospiele
- Personalcomputer
- und verschiedene Handansteuerungen.

Die wesentlichen technischen Merkmale sind: LED-Foto-Transistor-Elektronik, TTL- oder CMOS-kompatibel, 5 V 200 mA Einspeisung, 480 Impulse pro Ballumdrehung, getrennte Ausgänge für die X- und Y-Achsen mit Richtungstrennung.

Technische Daten

Mechanische Kennwerte

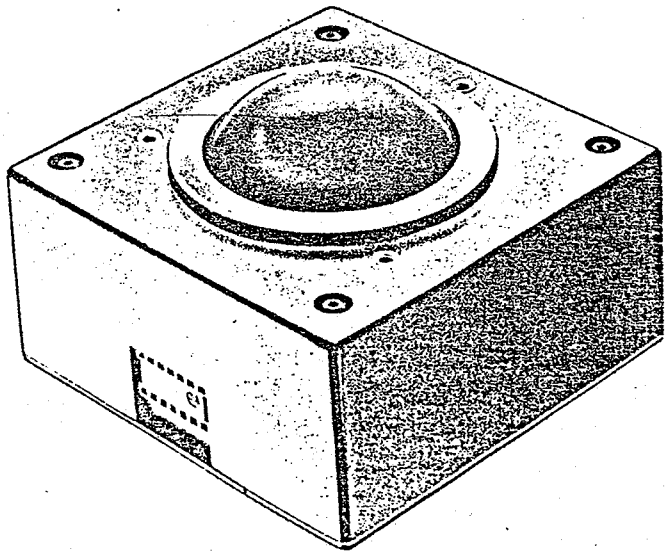
Gewicht	ca. 230 g
Abmessungen	siehe Massbilder, Rückseite
Rollkraft	≤ 0,45 N
Farbe des Gehäuses	schwarz
Farbe der Kugel	schwarz
Kugeldurchmesser	ca.51 mm
Drehzahl	max. 200 min^{-1}
Einbaulage	horizontal (Schräglage siehe Rückseite)

Elektrische Kennwerte

Code	inkremental
Impulse pro Kugelumdrehung	480 ± 5%
Lichtquelle	LED
Abtastung	optoelektronisch
Empfänger	Fototransistor
Ausgangssignale	Rechteckform (TTL- oder CMOS-kompatibel)
Versorgungsspannung (VCC)	+5 VDC ± 10% ≤ 200 mA
Gerätesteckverbinder	14 Pol DIP Messerleiste auf Leiterplatte montiert
Zugehöriger Flachbandkabelsteckverbinder	14 Pol DIP Federleiste (wird auf Wunsch mitgeliefert)

Umgebungsbedingungen

Umgebungs-Betriebstemperatur	+ 5 °C bis + 50 °C
Umgebungs-Lagertemperatur	− 25 °C bis + 80 °C



ELEKTRON AG
Riedhofstrasse 11, 8804 Au ZH, Telefon 01 783 01 11

T 200 SERIES
TRACKBALL
PECIFICATIONS

ECHANICAL

ight	8 oz. max. (23 GM)
tline and Mounting	See drawing (4.5 GM)
cking Force	1.5 oz. max.
l load	100 lb. (45 KG) max. downward
closure	ABS (cycolac KJT) black (other color special)
l	Themoset phenolic, 2" (51). dia., Black (other color special)
olver	Optical Encoders
ed	200 RPM max.
ıt Position	Horizontal. If mounted at an angle. The socket side. Should be elevated (30° max.)

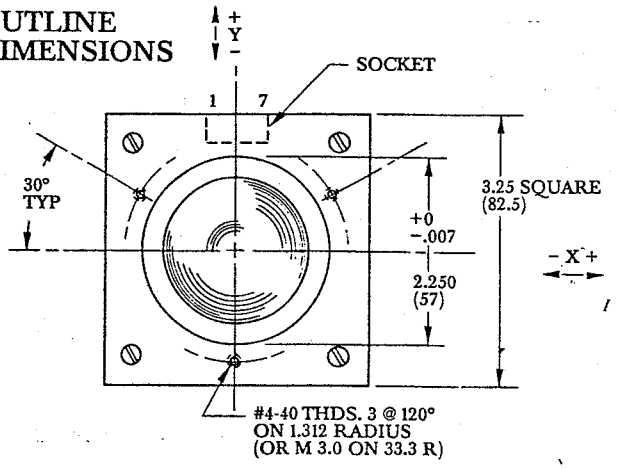
ELECTRICAL

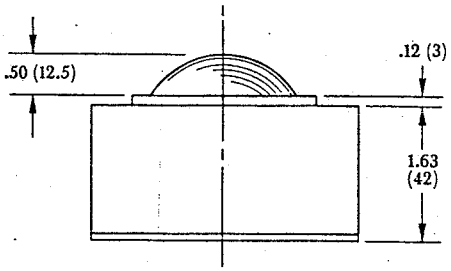
Code	Directional, count/no count
Pulse per rev. of ball	480±5%
Illumination	LED
Output Signal	Square Wave, TTL, or CMOS Compatible
Input Power (VCC)	+5 VDC ±10% 200 ma max.
Connector	14 Pin DIP Socket
Mating Connector	14 Pin insulation displacement plug (Not Furnished)

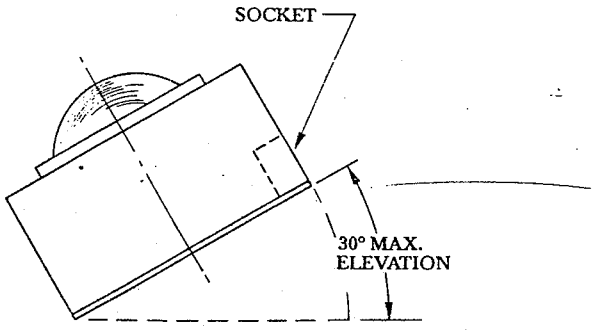
ENVIRONMENTAL

Temperature, Operating	+5°C to +50°C
Temperature, Storage	−25°C to +80°C

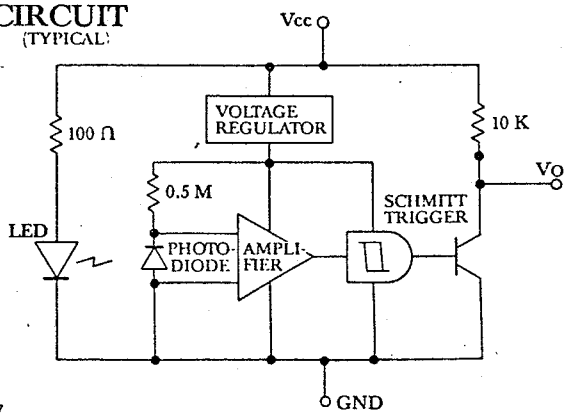
OUTLINE DIMENSIONS



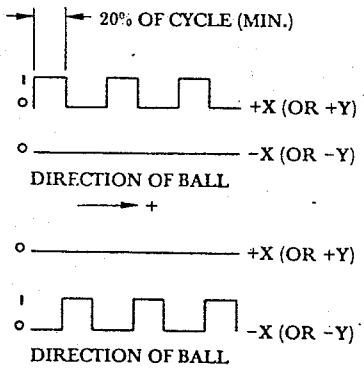




CIRCUIT
(TYPICAL)



WAVEFORM



CONNECTOR

Pin	Function
1	−Y
2	+Y
3	Common
4	+5VDC
5	Internal to 3
6	−X
7	+X
8-14	Internal to 3

CANON 9P connector

Pin 4
Pin 5
Pin 6
Pin 1
Pin 2
Pin 3

ISC INSTRUMENTS, INC. - 102 East Baker Street - Costa Mesa, California 92626 - Phone (714) 979-5300 - TWX 910-595-1987 DISC CSMA
EUROPE: DISC INSTRUMENTS, INC. - Finnigan MAT GmbH-Benzstrasse 28-8039 Puchheim/Munchen - Federal Republic of W. Germany
Phone: 089 801001 Telex: 05215007

Appendix 6 DATA SHEET OF THE MOUSE

DEPRAZ S A

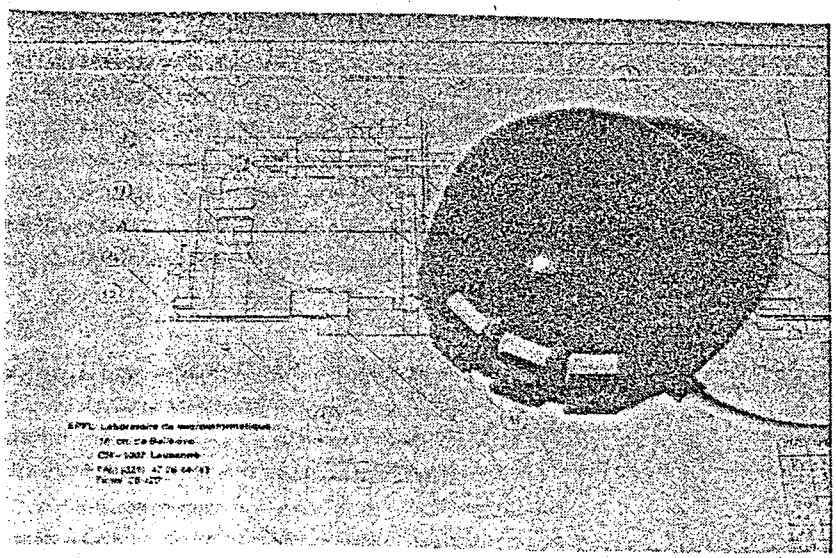
ÉTUDE ET FABRICATION DE COMPOSANTS
POUR LA MICROMÉCANIQUE
ET L'ÉLECTRONIQUE

MOUSE P-4

Interactive graphic input device
with parallel interface

MOUSE H-4

Interactive graphic input device
with shift register interface



Designed at the Swiss Federal Institute of Technology, Lausanne

Manufactured by Depraz SA, Micromechanics, CH-1345 Le Lieu, Switzerland
Tel (41)(21) 85 15 33 Telex 24310

April 1982

MOUSE P-4 and H-4

The mouse has been shown [1] to be an ideal interactive input device for graphics and text manipulation on an alphanumeric screen. Several products use a mouse successfully, and many personal workstations under development in various research projects have selected this input device for efficient man-machine dialogue.

The principle of the mouse is to encode and transfer to the computer the displacement of the mouse held in the palm of the hand, which can rest comfortably on the table next to the keyboard. Three keys easily depressed by the fingers can trigger a particular action or set a particular mode of operation in interaction with the display program.

[1] W.K. English, D.C. Englebart B. Huddart "Computer-aided display control" SRI Report, Project 5061, July 1965.

A Swiss product

Mouse number 4 is the fourth member of a product family started in 1973. It features improved performance and better ergonomical shape at a lower price. Two models exist: P-4 with a parallel interface and H-4 with signal serialisation. These mice have been developed at the Swiss Federal Institute of Technology (Laboratoire de Microinformatique) as part of their research efforts, and have been used for projects at INRIA Paris (N. Naffah, Project Kayak), ETH Zürich (Prof. Wirth, Lilith personal computer) and other institutes in the USA, Germany and France. More than 50 prototypes have been built. Industrial manufacturing started early in 1982, by a small but competent company next to the Lac de Joux, in the Jura region where famous Swiss Music Boxes are manufactured, together with the watches selected for NASA flights to the Moon.

General description

The hemispherical shape (diameter 85mm) of the mouse is easily grasped by the hand, with the three middle digits laying on the keys while the thumb and the little finger move the mouse precisely in interaction with the display. A plain tactile feedback is felt while depressing any key, and the way the mouse is held prevents any movement while depressing a key.

As the mouse is moved over the table top, a steel rolling pin of 20 mm diameter rotates. This rotation is decomposed into two perpendicular axes. For each axis, two square signals are produced, out of phase by 90 degrees, providing 15 pulses per millimeter, at any speed the hand can produce. Alignment and jitter is such that distance between consecutive pulses is guarantied. Pulse generation rate at maximum hand speed is about 1 ms (fig. 2).

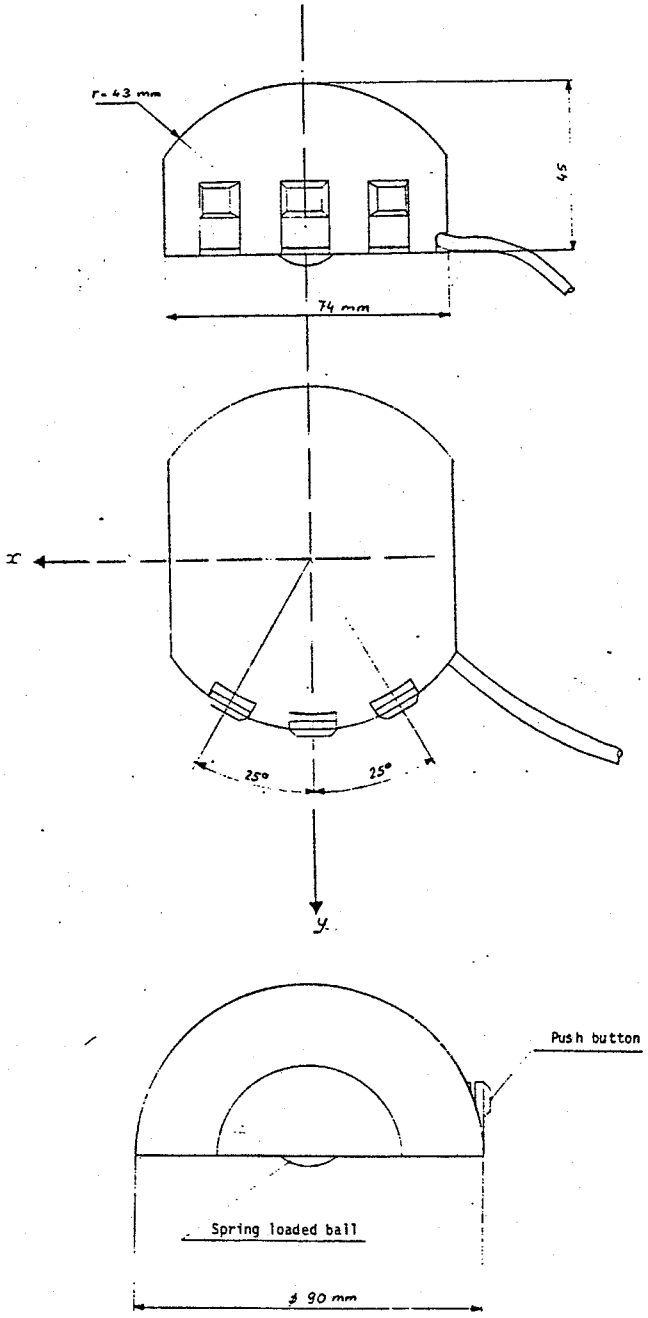


Fig. 1 Outside shape

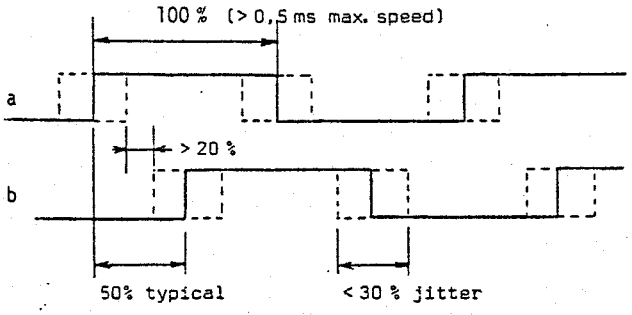


Fig. 2 Typical output timing

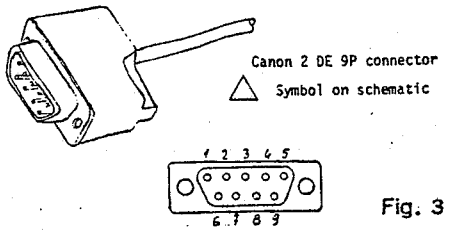
The mouse is usually held so that the X-axis is parallel to the major axis of the axis of the wrist, but the user can find any position that is comfortable. The Y-axis is perpendicular to the X-axis.
If the number of 15 pulses per mm is too high for standard applications, it can be easily divided by hardware or software.

Precision optical wheels with phototransistors and Schmitt trigger generate the signals.
On the standard mouse (P-4), 4 lines carry the pulses out of the mouse through à 9-wire cable which also carry the status of the three switches, the power supply (+5V ± 10%) and the power and signal return line (GND). Mouse H-4, shifts the 7-bit information through à 5-line cable including the power supply. Power consumption is 40 mA. All signals are CMOS and TTL-LS compatible.

MOUSE P-4

Standard connector on P-4 is a male 9-pin Canon subminiature connectors (fig.3). An 80 cm-long 9-wire cable is provided.

The P-4 mouse schematic is given in figure 4. When a key is depressed, the corresponding output is active low.



Male plug
Front view

Pin 1	+5V	6	GND
2	y2	7	$\overline{M}$
3	y1	8	$\overline{D}$
4	x2	9	$\overline{G}$
5	x1		

Fig. 3 Standard P-4 plug

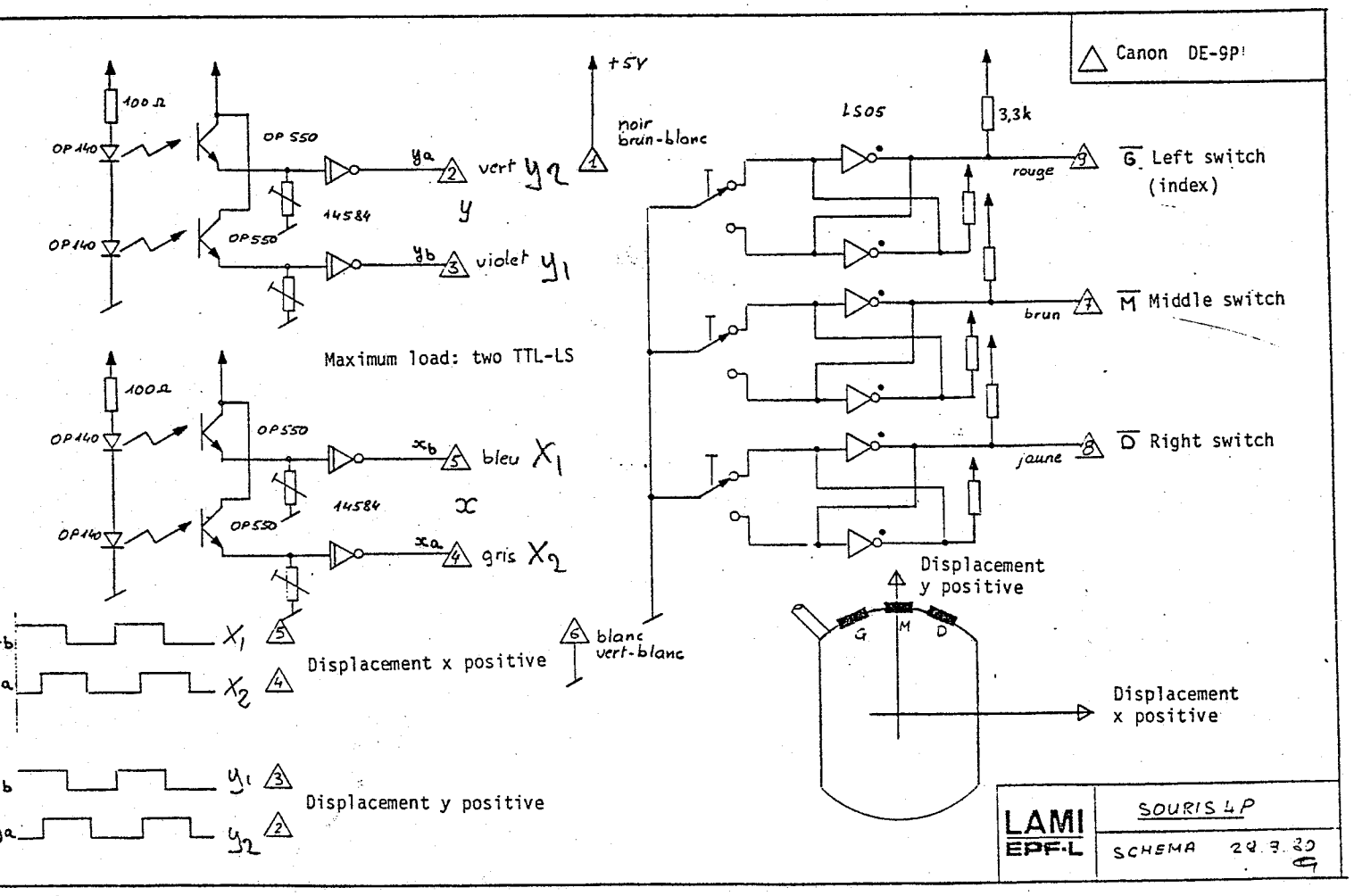


Fig. 4 P-4 schematic

Appendix 7 MC68000 DESCRIPTION

SECTION 4
SIGNAL AND BUS OPERATION DESCRIPTION

4.1 INTRODUCTION

This section contains a brief description of the input and output signals. A discussion of bus operation during the various machine cycles and operations is also given.

4.2 SIGNAL DESCRIPTION

The input and output signals can be functionally organized into the groups shown in Figure 4-1. The following paragraphs provide a brief description of the signals and also a reference (if applicable) to other chapters that contain more detail about the function being performed.

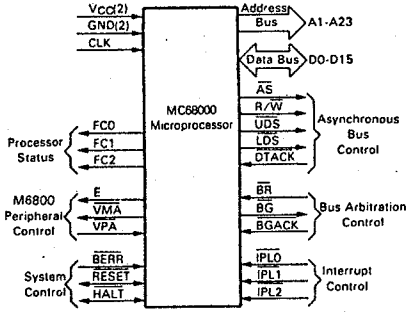


Figure 4-1. Input and Output Signals

4.2.1 ADDRESS BUS (A1 THROUGH A23). This 23-bit, unidirectional, three-state bus is capable of addressing eight megawords of data. It provides the address for bus operation during all cycles except interrupt cycles. During interrupt cycles, address lines A1, A2, and A3 provide information about what level interrupt is being serviced while address lines A4 through A23 are all set to a logic high.

4.2.2 DATA BUS (D0 THROUGH D15). This 16-bit, bidirectional, three-state bus is the general purpose data path. It can transfer and accept data in either word or byte length. During an interrupt acknowledge cycle, the external device supplies the vector number on data lines D0 through D7.

4.2.3 ASYNCHRONOUS BUS CONTROL. Asynchronous data transfers are handled using the following control signals: address strobe, read/write, upper and lower data strobes, and data transfer acknowledge. These signals are explained in the following paragraphs.

4.2.3.1 Address Strobe ($\overline{AS}$). This signal indicates that there is a valid address on the address bus.

4.2.3.2 Read/Write (R/$\overline{W}$). This signal defines the data bus transfer as a read or write cycle. The R/$\overline{W}$ signal also works in conjunction with the upper and lower data strobes as explained in the following paragraph.

4.2.3.3 Upper and Lower Data Strobes ($\overline{UDS}$, $\overline{LDS}$). These signals control the data on the data bus as shown in Table 4-1. When the R/$\overline{W}$ line is high, the processor will read from the data bus as indicated. When the R/$\overline{W}$ line is low, the processor will write to the data bus as shown.

Table 4-1. Data Strobe Control of Data Bus

$\overline{UDS}$	$\overline{LDS}$	R/$\overline{W}$	D8-D15	D0-D7
High	High	–	No Valid Data	No Valid Data
Low	Low	High	Valid Data Bits 8-15	Valid Data Bits 0-7
High	Low	High	No Valid Data	Valid Data Bits 0-7
Low	High	High	Valid Data Bits 8-15	No Valid Data
Low	Low	Low	Valid Data Bits 8-15	Valid Data Bits 0-7
High	Low	Low	Valid Data Bits 0-7*	Valid Data Bits 0-7
Low	High	Low	Valid Data Bits 8-15	Valid Data Bits 8-15*

*These conditions are a result of current implementation and may not appear on future devices.

4.2.3.4 Data Transfer Acknowledge ($\overline{DTACK}$). This input indicates that the data transfer is completed. When the processor recognizes $\overline{DTACK}$ during a read cycle, data is latched and the bus cycle terminated. When $\overline{DTACK}$ is recognized during a write cycle, the bus cycle is terminated. Refer to paragraphs 4.3.4 and 4.3.5 for additional information about $\overline{DTACK}$.

4.2.4 BUS ARBITRATION CONTROL. These three signals form a bus arbitration circuit to determine which device will be the bus master device. Refer to paragraph 4.3.2 for a detailed description.

4.2.4.1 Bus Request ($\overline{BR}$). This input is wire ORed with all other devices that could be bus masters. This input indicates to the processor that some other device desires to become the bus master.

4.2.4.2 Bus Grant ($\overline{BG}$). This output indicates to all other potential bus master devices that the processor will release bus control at the end of the current bus cycle.

4.2.4.3 Bus Grant Acknowledge ($\overline{BGACK}$). This input indicates that some other device has become the bus master. This signal cannot be asserted until the following four conditions are met:

1. a bus grant has been received
2. address strobe is inactive which indicates that the microprocessor is not using the bus
3. data transfer acknowledge is inactive which indicates that either memory or the peripherals are not using the bus
4. bus grant acknowledge is inactive which indicates that no other device is still claiming bus mastership.

4.2.5 INTERRUPT CONTROL ($\overline{IPL0}$, $\overline{IPL1}$, $\overline{IPL2}$). These input pins indicate the encoded priority level of the device requesting an interrupt. Level seven is the highest priority while level zero indicates that no interrupts are requested. The least significant bit is contained in $\overline{IPL0}$ and the most significant bit is contained in $\overline{IPL2}$. Refer to Section 5 for details on interrupt operation.

4.2.6 SYSTEM CONTROL. The system control inputs are used to either reset or halt the processor and to indicate to the processor that bus errors have occurred. The three system control inputs are explained in the following paragraphs.

4.2.6.1 Bus Error ($\overline{BERR}$). This input informs the processor that there is a problem with the cycle currently being executed. Problems may be a result of:

1. nonresponding devices
2. interrupt vector number acquisition failure
3. illegal access request as determined by a memory management unit
4. other application dependent errors.

The bus error signal interacts with the halt signal to determine if exception processing should be performed or the current bus cycle should be retried.

Refer to paragraph 4.3.3 for additional information about the interaction of the bus error and halt signals.

4.2.6.2 Reset ($\overline{RESET}$). This bidirectional signal line acts to reset (initiate a system initialization sequence) the processor in response to an external reset signal. An internally generated reset (result of a RESET instruction) causes all external devices to be reset and the internal state of the processor is not affected. A total system reset (processor and external devices) is the result of external halt and reset signals applied at the same time. Refer to paragraph 4.3.6 for additional information about reset operation.

4.2.6.3 Halt ($\overline{HALT}$). When this bidirectional line is driven by an external device, it will cause the processor to stop at the completion of the current bus cycle. When the processor has been halted using this input, all control signals are inactive and all three-state

lines are put in their high-impedance state. Refer to paragraphs 4.3.3 and 4.3.4 for additional information about the interaction between the halt and bus error signals.

When the processor has stopped executing instructions, such as in a double bus fault condition, the halt line is driven by the processor to indicate to external devices that the processor has stopped.

4.2.7 M6800 PERIPHERAL CONTROL. These control signals are used to allow the interfacing of synchronous M6800 peripheral devices with the asynchronous MC68000. These signals are explained in the following paragraphs.

4.2.7.1 Enable (E). This signal is the standard enable signal common to all M6800 type peripheral devices. The period for this output is ten MC68000 clock periods (six clocks low; four clocks high).

4.2.7.2 Valid Peripheral Address ($\overline{VPA}$). This input indicates that the device or region addressed is an M6800 family device and that data transfer should coincide with the enable (E) signal. This input also indicates that the processor should use automatic vectoring for an interrupt. Refer to Section 6.

4.2.7.3 Valid Memory Address ($\overline{VMA}$). This output is used to indicate to M6800 peripheral devices that there is a valid address on the address bus and the processor is synchronized to enable. This signal only responds to a valid peripheral address ($\overline{VPA}$) input which indicates that the peripheral is an M6800 family device.

4.2.8 PROCESSOR STATUS (FC0, FC1, FC2). These function code outputs indicate the mode (user or supervisor) and the cycle type currently being executed as shown in Table 4-2. The information indicated by the function code outputs is valid whenever address strobe ($\overline{AS}$) is active.

Table 4-2. Function Code Outputs

FC2	FC1	FC0	Cycle Type
Low	Low	Low	(Undefined, Reserved)
Low	Low	High	User Data
Low	High	Low	User Program
Low	High	High	(Undefined, Reserved)
High	Low	Low	(Undefined, Reserved)
High	Low	High	Supervisor Data
High	High	Low	Supervisor Program
High	High	High	Interrupt Acknowledge

4.2.9 CLOCK (CLK). The clock input is a TTL compatible signal that is internally buffered for development of the internal clocks needed by the processor. The clock input shall be a constant frequency.

4.2.10 SIGNAL SUMMARY. Table 4-3 is a summary of all the signals discussed in the previous paragraphs.

Table 4-3. Signal Summary

Signal Name	Mnemonic	Input/Output	Active State	Three State
Address Bus	A1-A23	Output	High	Yes
Data Bus	D0-D15	Input/Output	High	Yes
Address Strobe	$\overline{AS}$	Output	Low	Yes
Read/Write	R/$\overline{W}$	Output	Read-High Write-Low	Yes
Upper and Lower Data Strobes	$\overline{UDS}$, $\overline{LDS}$	Output	Low	Yes
Data Transfer Acknowledge	$\overline{DTACK}$	Input	Low	No
Bus Request	$\overline{BR}$	Input	Low	No
Bus Grant	$\overline{BG}$	Output	Low	No
Bus Grant Acknowledge	$\overline{BGACK}$	Input	Low	No
Interrupt Priority Level	$\overline{IPL0}$, $\overline{IPL1}$, $\overline{IPL2}$	Input	Low	No
Bus Error	$\overline{BERR}$	Input	Low	No
Reset	$\overline{RESET}$	Input/Output	Low	No*
Halt	$\overline{HALT}$	Input/Output	Low	No*
Enable	E	Output	High	No
Valid Memory Address	$\overline{VMA}$	Output	Low	Yes
Valid Peripheral Address	$\overline{VPA}$	Input	Low	No
Function Code Output	FC0, FC1, FC2	Output	High	Yes
Clock	CLK	Input	High	No
Power Input	V_{CC}	Input	–	–
Ground	GND	Input	–	–

*Open Drain

4.3 BUS OPERATION

The following paragraphs explain control signal and bus operation during data transfer operations, bus arbitration, bus error and halt conditions, and reset operation.

4.3.1 DATA TRANSFER OPERATIONS. Transfer of data between devices involves the following leads:

- Address Bus A1 through A23
- Data Bus D0 through D15
- Control Signals

The address and data buses are separate parallel buses used to transfer data using an asynchronous bus structure. In all cycles, the bus master assumes responsibility for deskewing all signals it issues at both the start and end of a cycle. In addition, the bus master is responsible for deskewing the acknowledge and data signals from the slave device.

The following paragraphs explain the read, write, and read-modify-write cycles. The indivisible read-modify-write cycle is the method used by the MC68000 for interlocked multiprocessor communications.

NOTE

The terms **assertion** and **negation** will be used extensively. This is done to avoid confusion when dealing with a mixture of "active-low" and "active-high" signals. The term assert or assertion is used to indicate that a signal is active

independent of whether that voltage is low or high. The term negation is used to indicate that a signal is inactive.

Each of the bus cycle operations is defined in terms of a succession of states. These states are described for reference purposes only, and do not necessarily correspond to any implemented machine states.

4.3.1.1 Read Cycle. During a read cycle, the processor receives data from memory or a peripheral device. The processor reads bytes of data in all cases. If the instruction specifies a word (or long word) operation, the processor reads both bytes. When the instruction specifies byte operation, the processor uses the internal A0 bit to determine which byte to read and then issues the data strobe required for that byte. For byte operations, when the A0 bit equals zero, the upper data strobe is issued. When the A0 bit equals one, the lower data strobe is issued. When the data is received, the processor correctly positions it internally.

A word read cycle flow chart is given in Figure 4-2. A byte read cycle flow chart is given in Figure 4-3. Read cycle timing is given in Figure 4-4 and Figure 4-5 details word and byte read cycle operation. Refer to these illustrations during the following detailed discussion.

At state zero (S0) in the read cycle, the address bus (A1 through A23) is in the high-impedance state. A function code is asserted on function code output lines FC0, FC1, and FC2 to indicate which address space this cycle will operate on. The read/write (R/$\overline{W}$) signal is switched high to indicate a read cycle. One-half clock cycle later, at state 1, the address bus is released from the high-impedance state.

In state 2, the address strobe ($\overline{AS}$) is asserted to indicate that there is a valid address on the address bus and the upper and lower data strobe ($\overline{UDS}$, $\overline{LDS}$) is asserted as required. The memory or peripheral device uses the address bus and the address strobe to determine if it has been selected. The selected device uses the read/write signal and the data strobe to place its information on the data bus. Concurrent with placing data on the data bus, the selected device asserts data transfer acknowledge ($\overline{DTACK}$). No new control signals are issued during states 3 and 4.

Data transfer acknowledge must be present at the processor a setup time before the end of state 4 or the processor will substitute wait states for states 5 and 6. State 5 starts the synchronization of the returning data transfer acknowledge. The bus interface circuitry issues requests for subsequent internal cycles during state 6. At the end of state 6 (beginning of state 7) incoming data is latched into an internal data bus holding register.

During state 7, address strobe and the upper and/or lower data strobes are negated. The address bus is held valid through state 7 to allow for static memory operation and signal skew. The read/write signal and the function code outputs also remain valid through state 7 to ensure a correct transfer operation. The slave device keeps its data asserted until it detects the negation of either the address strobe or the upper and/or lower data strobe. The slave device must remove its data and data transfer acknowledge within one clock period of recognizing the negation of the address or data strobes. Note that the data bus might not become free and data transfer acknowledge might not be removed until state 0 or 1.

When address strobe is negated, the slave device is released. Note that a slave device must remain selected as long as address strobe is asserted to ensure the correct functioning of the read-modify-write cycle as explained in paragraph 4.3.1.3.

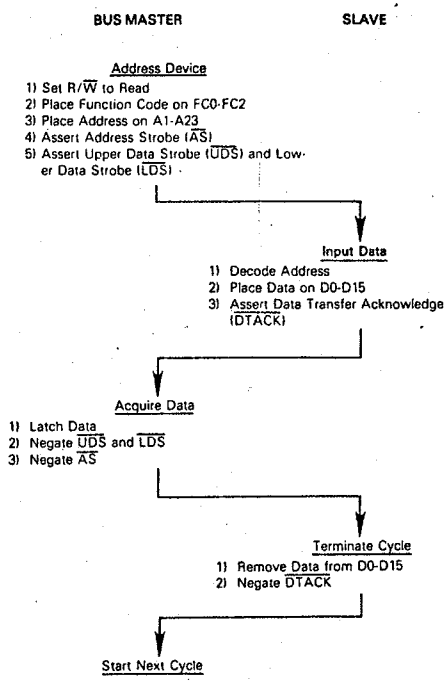


Figure 4-2. Word Read Cycle Flow Chart

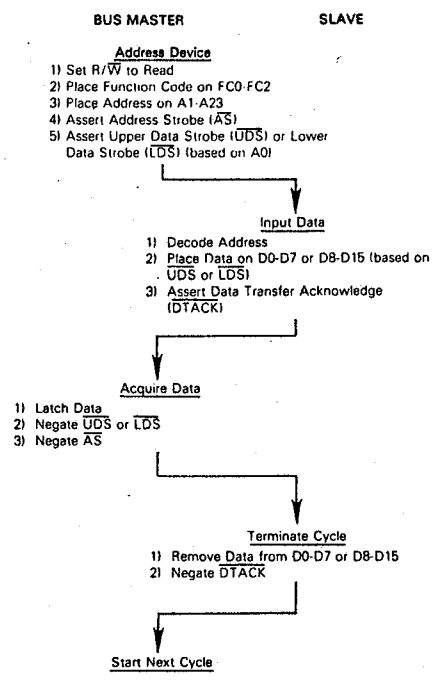


Figure 4-3. Byte Read Cycle Flow Chart

4.3.1.2 Write Cycle. During a write cycle, the processor sends data to memory or a peripheral device. The processor writes bytes of data in all cases. If the instruction specifies a word operation, the processor writes both bytes. When the instruction specifies a byte operation, the processor uses the internal A0 bit to determine which byte to write and then issues the data strobe required for that byte. For byte operations, when the A0 bit equals zero, the upper data strobe is issued. When the A0 bit equals one, the lower data strobe is issued. A word write cycle flow chart is given in Figure 4-6. A byte write cycle flow chart is given in Figure 4-7. Write cycle timing is given in Figure 4-4 and Figure 4-8 details word and byte write cycle operation. Refer to these illustrations during the following detailed discussion.

At state zero (S0) in the write cycle, the address bus (A1 through A23) is in the high-impedance state. A function code is asserted on function code output lines FC0, FC1, and FC2 to indicate which address space this cycle will operate on.

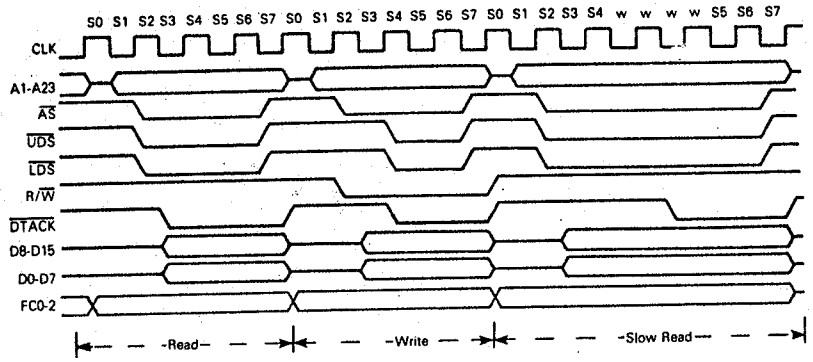


Figure 4-4. Read and Write Cycle Timing Diagram

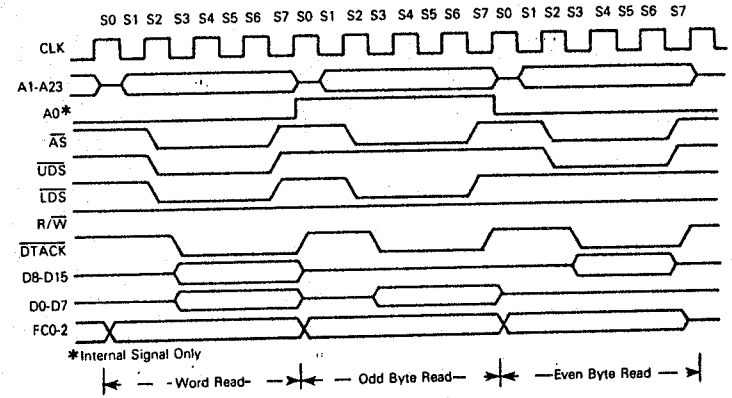


Figure 4-5. Word and Byte Read Cycle Timing Diagram

NOTE

The read/write (R/$\overline{W}$) signal remains high until state 2 to prevent bus conflicts with preceding read cycles. The data bus is not driven until state 3.

One-half clock cycle later, at state 1, the address bus is released from the high-impedance state.

In state 2, the address strobe ($\overline{AS}$) is asserted to indicate that there is a valid address on the address bus. The memory or peripheral device uses the address bus and the address strobe to determine if it has been selected. During state 2, the read/write signal is switched low to indicate a write cycle. When external processor data bus buffers are required,

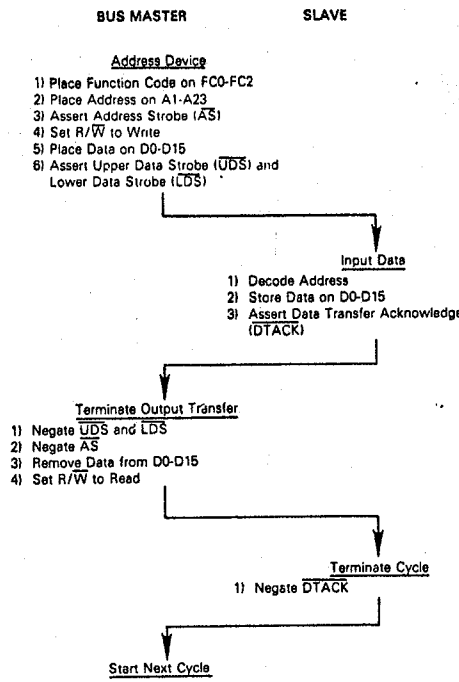


Figure 4-6. Word Write Cycle Flow Chart

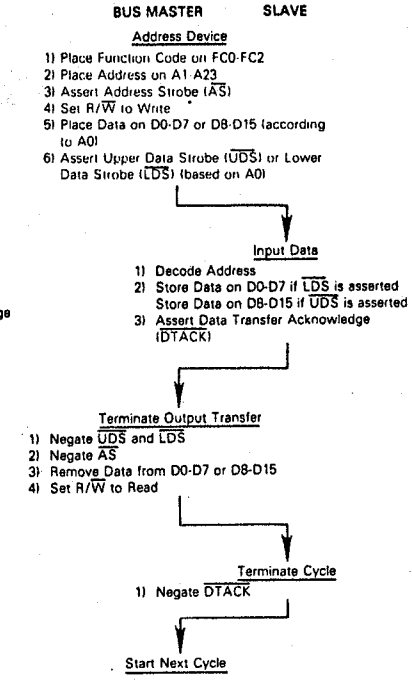


Figure 4-7. Byte Write Cycle Flow Chart

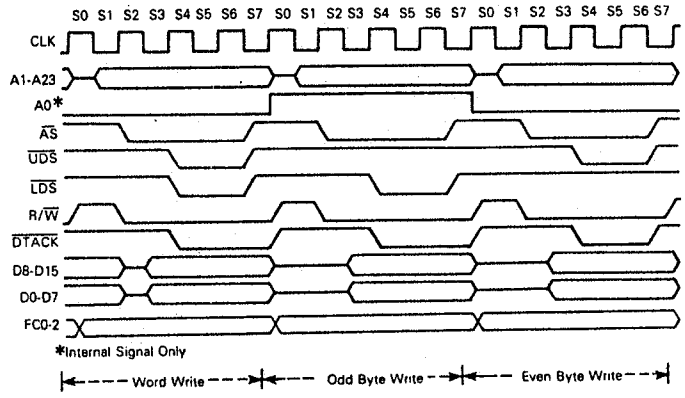


Figure 4-8. Word and Byte Write Cycle Timing Diagram

the read/write line provides sufficient directional control. Data is not asserted during this state to allow sufficient turnaround time for external data buffers (if used). Data is asserted onto the data bus during state 3.

In state 4, the data strobes are asserted as required to indicate that the data bus is stable. The selected device uses the read/write signal and the data strobes to take its information from the data bus.The selected device asserts data transfer acknowledge ($\overline{\text{DTACK}}$) when it has successfully stored the data.

Data transfer acknowledge must be present at the processor a setup time before the end of state 4 or the processor will substitute wait states for states 5 and 6. State 5 starts the synchronization of the returning data transfer acknowledge. The bus interface circuitry issues requests for subsequent internal cycles during state 6.

During state 7, address strobe and the upper and/or lower data strobes are negated. The address and data buses are held valid through state 7 to allow for static memory operation and signal skew. The read/write signal and the function code outputs also remain valid through state 7 to ensure a correct transfer operation. The slave device keeps its data transfer acknowledge asserted until it detects the negation of either the address strobe or the upper and/or lower data strobe. The slave device must remove its data transfer acknowledge within one clock period after recognizing the negation of the address or data strobes. Note that the processor releases the data bus at the end of state 7 but that data transfer acknowledge might not be removed until state 0 or 1. When address strobe is negated, the slave device is released.

5.5 EXCEPTION PROCESSING DETAILED DISCUSSION

Exceptions have a number of sources and each exception has processing which is peculiar to it. The following paragraphs detail the sources of exceptions, how each arises, and how each is processed.

5.5.1 RESET. The reset input provides the highest exception level. The processing of the reset signal is designed for system initiation, and recovery from catastrophic failure. Any processing in progress at the time of the reset is aborted and cannot be recovered. The processor is forced into the supervisor state and the trace state is forced off. The processor interrupt priority mask is set at level seven. The vector number is internally generated to reference the reset exception vector at location 0 in the supervisor program space. Because no assumptions can be made about the validity of register contents, in particular the supervisor stack pointer, neither the program counter nor the status register is saved. The address contained in the first two words of the reset exception vector is fetched as the initial supervisor stack pointer, and the address in the last two words of the reset exception vector is fetched as the initial program counter. Finally, instruction execution is started at the address in the program counter. The power-up/restart code should be pointed to by the initial program counter.

The RESET instruction does not cause loading of the reset vector, but does assert the reset line to reset external devices. This allows the software to reset the system to a known state and then continue processing with the next instruction.

5.5.2 INTERRUPTS. Seven levels of interrupt priorities are provided. Devices may be chained externally within interrupt priority levels, allowing an unlimited number of peripheral devices to interrupt the processor. Interrupt priority levels are numbered from one to seven, level seven being the highest priority. The status register contains a three-bit mask which indicates the current processor priority, and interrupts are inhibited for all priority levels less than or equal to the current processor priority.

An interrupt request is made to the processor by encoding the interrupt request level on the interrupt request lines; a zero indicates no interrupt request. Interrupt requests arriving at the processor do not force immediate exception processing, but are made pending. Pending interrupts are detected between instruction executions. If the priority of the pending interrupt is lower than or equal to the current processor priority, execution continues with the next instruction and the interrupt exception processing is postponed. (The recognition of level seven is slightly different, as explained in the following paragraph.)

If the priority of the pending interrupt is greater than the current processor priority, the exception processing sequence is started. First a copy of the status register is saved, and the privilege state is set to supervisor, tracing is suppressed, and the processor priority level is set to the level of the interrupt being acknowledged. The processor fetches the vector number from the interrupting device, classifying the reference as an interrupt acknowledge and displaying the level number of the interrupt being acknowledged on the address bus. If external logic requests an automatic vectoring, the processor internally generates a vector number which is determined by the interrupt level number (see paragraph 6.3). If external logic indicates a bus error, the interrupt is taken to be spurious and the generated vector number references the spurious interrupt vector. The processor then proceeds with the usual exception processing, saving the program counter and status register on the supervisor stack. The saved value of the program counter is the address of the instruction which would have been executed had the interrupt not been present. The content of the interrupt vector whose vector number was previously obtained is fetched and loaded into the program counter and normal instruction execution commences in the interrupt handling routine. A flow chart for the interrupt acknowledge sequence is given in Figure 5-4; a timing diagram is given in Figure 5-5.

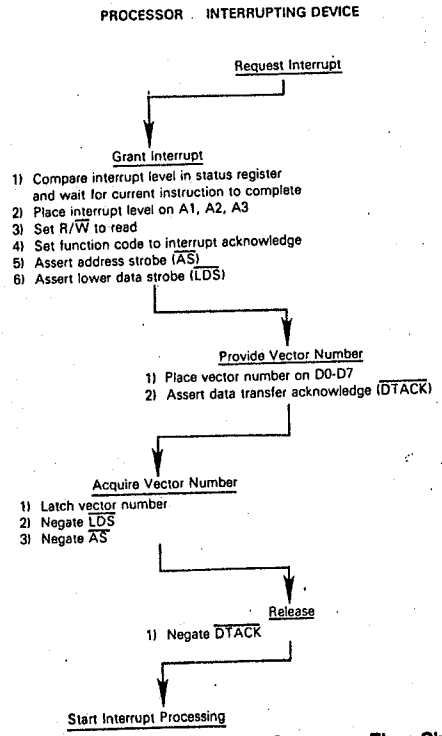


Figure 5-4. Interrupt Acknowledge Sequence Flow Chart

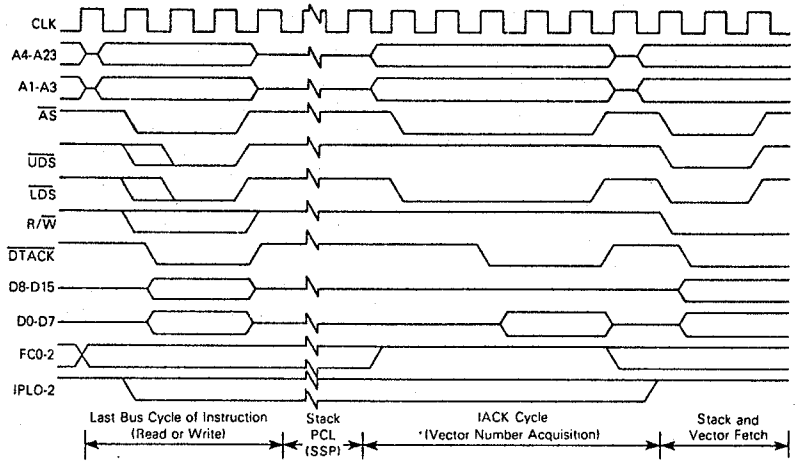


Figure 5-5. Interrupt Acknowledge Sequence Timing Diagram

Priority level seven is a special case. Level seven interrupts cannot be inhibited by the interrupt priority mask, thus providing a "non-maskable interrupt" capability. An interrupt is generated each time the interrupt request level changes from some lower level to level seven. Note that a level seven interrupt may still be caused by the level comparison if the request level is a seven and the processor priority is set to a lower level by an instruction.

5.5.3 INSTRUCTION TRAPS. Traps are exceptions caused by instructions. They arise either from processor recognition of abornomal conditions during instruction, execution, or from use of instructions whose normal behavior is trapping.

Exception processing for traps is straightforward. The status register is copied, the supervisor state is entered, and the trace state is turned off. The vector number is internally generated; for the TRAP instruction, part of the vector number comes from the instruction itself. The program counter and the copy of the status register are saved on the supervisor stack. The saved value of the program counter is the address of the instruction after the instruction which generated the trap. Finally, instruction execution commences at the address contained in the exception vector.

Some instructions are used specifically to generate traps. The TRAP instruction always forces an exception and is useful for implementing system calls for user programs. The TRAPV and CHK instructions force an exception if the user program detects a runtime error, which may be an arithmetic overflow or a subscript out of bounds.

The signed divide (DIVS) and unsigned divide (DIVU) instructions will force an exception if a division operation is attempted with a divisor of zero.

SECTION 6
INTERFACE WITH M6800 PERIPHERALS

6.1 INTRODUCTION

Motorola's extensive line of M6800 peripherals are directly compatible with the MC68000. Some of these devices that are particularly useful are:

- MC6821 Peripheral Interface Adapter
- MC6840 Programmable Timer Module
- MC6843 Floppy Disk Controller
- MC6845 CRT Controller
- MC6849 Dual Density Floppy Disk Controller
- MC6850 Asynchronous Communication Interface Adapter
- MC6852 Synchronous Serial Data Adapter
- MC6854 Advanced Data Link Controller
- MC68488 General Purpose Interface Adapter

To interface the synchronous M6800 peripherals with the asynchronous MC68000, the processor modifies its bus cycle to meet the M6800 cycle requirements whenever an M6800 device address is detected. This is possible since both processors use memory mapped I/O. Figure 6-1 is a flow chart of the interface operations between the processor and M6800 devices.

6.2 DATA TRANSFER OPERATION

Three signals on the processor provide the M6800 interface. They are: enable (E), valid memory address ($\overline{VMA}$), and valid peripheral address ($\overline{VPA}$). Enable corresponds to the E or $\phi 2$ signal in existing M6800 systems. It is the bus clock used by the M6800 peripherals to synchronize data transfer. Enable is a constant frequency clock that is one-tenth of the incoming MC68000 clock frequency. The timing of E allows 1 MHz peripherals to be used with an 8 MHz MC68000. Enable has a 60/40 duty cycle; that is, it is low for six input clocks and high for four input clocks. This duty cycle allows the processor to do two successive $\overline{VPA}$ accesses on successive E pulses.

M6800 cycle timing is given in Figure 6-2. At state zero (S0) in the cycle, the address bus is in the high-impedance state. A function code is asserted on the function code output lines. One-half clock cycle later, in state 1, the address bus is released from the high-impedance state.

During state 2, the address strobe ($\overline{AS}$) is asserted to indicate that there is a valid address on the address bus. If the bus cycle is a read cycle, the upper and/or lower data strobes are also asserted in state 2. If the bus cycle is a write cycle, the read/write (R/$\overline{W}$) signal is

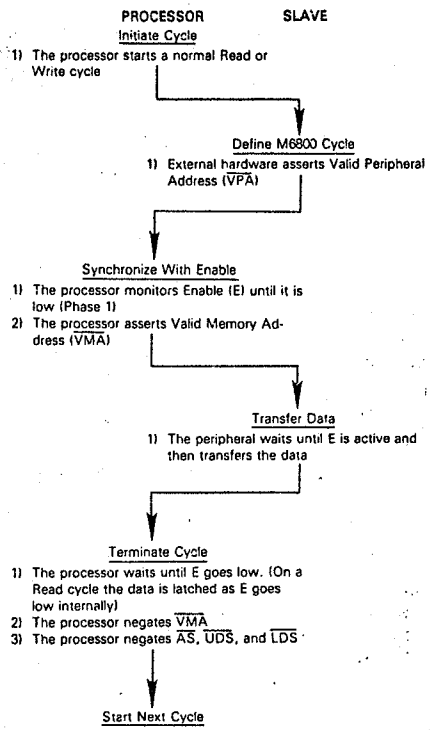


Figure 6-1. M6800 Interfacing Flow Chart

switched to low (write) during state 2. One-half clock later, in state 3, the write data is placed on the data bus; and in state 4, the data strobes are issued to indicate valid data on the bus. The processor now inserts wait states until it recognizes the assertion of $\overline{VPA}$.

The $\overline{VPA}$ input signals the processor that the address on the bus is the address of an M6800 device (or an area reserved for M6800 devices) and that the bus should conform to the $\phi 2$ transfer characteristics of the M6800 bus. Valid peripheral address is derived by decoding the address bus, conditioned by address strobe.

After the recognition of $\overline{VPA}$, the processor assures that the Enable (E) is low, by waiting if necessary, and subsequently asserts $\overline{VMA}$. Valid memory address is then used as part of the chip select equation of the peripheral. This ensures that the M6800 peripherals are selected and deselected at the correct time. The peripheral now runs its cycle during the high portion of the E signal.

During a read cycle, the processor latches the peripheral data in state 6. For all cycles, the processor negates the address and data strobes one-half clock cycle later in state 7, and the enable signal goes low at this time. Another half clock later, the address bus is put in the high-impedance state. During a write cycle, the data bus is put in the high-impedance state and the read/write signal is switched high at this time. The peripheral logic must remove $\overline{VPA}$ within one clock after address strobe is negated.

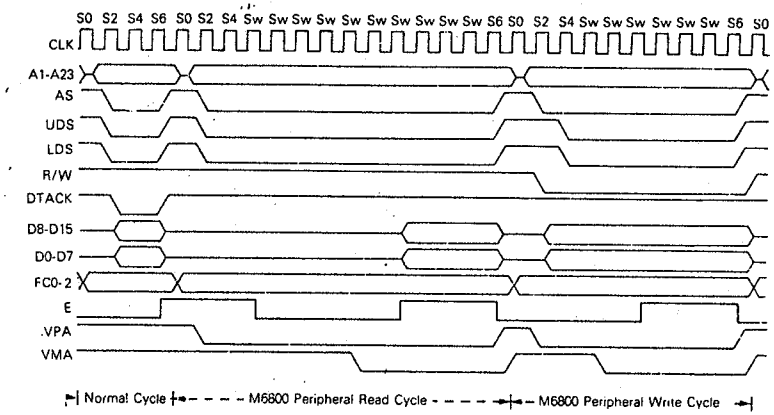


Figure 6-2. M6800 Cycle Operation

Timing information for interfacing with M6800 peripheral devices is contained in the data sheet for the MC68000. Always refer to the latest revision of the MC68000 and desired M6800 peripheral device data sheet when interfacing these devices.

A number of application notes have been published detailing the use of the MC68000 with M6800 Family peripheral devices. The following listing of application notes is current through September 1981. They are available from the Motorola Literature Distribution Center in Phoenix, Arizona.

Title	No.
Interfacing M6800 Peripheral Devices to the MC68000 Asynchronously	808
Interfacing the MC68000 to the MC6846 RIOT	809
Dual 16-Bit Ports for the MC68000 Using Two MC6821s	810
Color Graphics for the MC68000 Using the MC6847	815
A Software Refreshed Memory Card for the MC68000	816
Asynchronous Communications for the MC68000 Using the MC6850	817
Synchronous I/O for the MC68000 Using the MC6852	818
Prioritized Individually Vectored Interrupts for Multiple Peripheral Systems with the MC68000	819
MC68000 DMA Using the MC6844 DMA Controller	824
Using the MC68000 and the MC6845 for a Color Graphics System	834

6.3 INTERRUPT INTERFACE OPERATION

During an interrupt acknowledge cycle while the processor is fetching the vector, if $\overline{VPA}$ is asserted, the MC68000 will assert $\overline{VMA}$ and complete a normal M6800 read cycle, as shown in Figure 6-3. The processor will then use an internally generated vector that is a function of the interrupt being serviced. This process is known as autovectoring. The seven autovectors are vector numbers 25 through 31 (decimal).

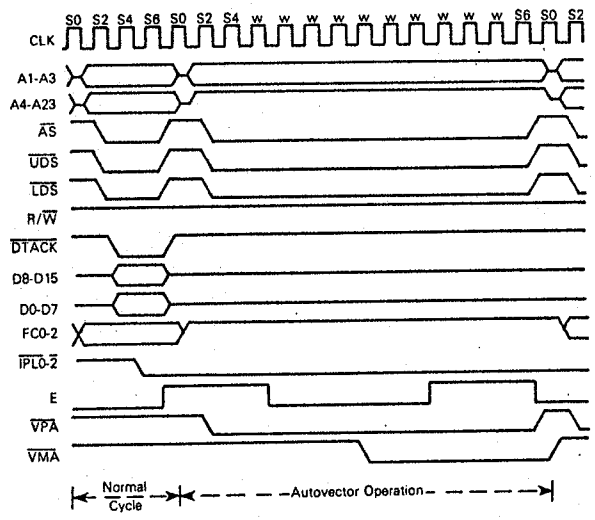


Figure 6-3. Autovector Operation Timing Diagram

This operates in the same fashion (but is not restricted to) the M6800 interrupt sequence. The basic difference is that there are six normal interrupt vectors and one NMI-type vector. As with both the M6800 and the MC68000's normal vectored interrupt, the interrupt service routine can be located anywhere in the address space. This is due to the fact that while the vector numbers are fixed, the contents of the vector table entries are assigned by the user.

Since $\overline{VMA}$ is asserted during autovectoring, the M6800 peripheral address decoding should prevent unintended accesses.

APPENDIX A
CONDITION CODES COMPUTATION

A.1 INTRODUCTION

This appendix provides a discussion of how the condition codes were developed, the meanings of each bit, how they are computed, and how they are represented in the instruction set details.

Two criteria were used in developing the condition codes:

- Consistency — across instruction, uses, and instances
- Meaningful Results — no change unless it provides useful information

The consistency across instructions means that instructions which are special cases of more general instructions affect the condition codes in the same way. Consistency across instances means that if an instruction ever affects a condition code, it will always affect that condition code. Consistency across uses means that whether the condition codes were set by a compare, test, or move instruction, the conditional instructions test the same situation. The tests used for the conditional instructions and the code computations are given in paragraph A.5.

A.2 CONDITION CODE REGISTER

The condition code register portion of the status register contains five bits:

N — Negative
Z — Zero
V — Overflow
C — Carry
X — Extend

The first four bits are true condition code bits in that they reflect the condition of the result of a processor operation. The X bit is an operand for multiprecision computations. The carry bit (C) and the multiprecision operand extend bit (X) are separate in the MC68000 to simplify the programming model.

A.3 CONDITION CODE REGISTER NOTATION

In the instruction set details given in Appendix B, the description of the effect on the condition codes is given in the following form:

Appendix 8 MC68000 FRONTBUS

Appendix 8

MC68000 FRONTBUS

60 pin scotch cable connector

Pin number			
A1	1	2	A2
A3	3	4	A4
A5	5	6	A6
A7	7	8	A8
A9	9	10	A10
A11	11	12	A12
A13	13	14	A14
A15	15	16	A16
A17	17	18	A18
A19	19	20	A20
A21	21	22	A22
A23	23	24	$\overline{AS}$
R/$\overline{W}$	25	26	$\overline{DTACK}$
RESERVED	27	28	$\overline{LDS}$
$\overline{UDS}$	29	30	D0
D1	31	32	D2
D3	33	34	D4
D5	35	36	D6
D7	37	38	D8
D9	39	40	D10
D11	41	42	D12
D13	43	44	D14
D15	45	46	$\overline{BR}$
E	47	48	$\overline{BG}$
$\overline{VMA}$	49	50	$\overline{BGACK}$
$\overline{VPA}$	51	52	$\overline{INT\ RQ}$
$\overline{BERR}$	53	54	RESERVED
$\overline{RESET}$	55	56	$\overline{HALT}$
FC0	57	58	FC1
FC2	59	60	RESERVED

The pin functions is the same as described in the 68000 specifications shown in appendix 7.

Appendix 9 G64 BUS DESCRIPTION

FONCTIONNEMENT DU BUS

Le transfer de données entre circuits utilise les lignes suivantes:

- Bus d'adresse A0. A15 et extensions éventuelles
- Bus de données $\overline{D0}$ à $\overline{D7}$
- Signaux de validation $\overline{VMA}$ et $\overline{VPA}$
- Signaux d'horloge et de transaction

- Bus d'adresse (A0.15) vrai à l'état haut

Ce bus de 16 bits unidirectionnel à 3 états est capable de selectionner l'un parmi des 65536 octets de la zone adressable. Il est transmis sur le bus depuis le processeur à travers des circuits d'amplification du type 74LS244.

Dans certaines configurations multi pages, 4 bits d'extensions sont transmis simultanément.

Ces signaux sont établis avant le 1^{e} quart de cycle.

- Bus de données ($\overline{D0}.\overline{D7}$) vrai à l'état bas

Ce bus de 8 bits, bidirectionnel à 3 états, est utilisé pour transmettre les données depuis le processeur vers le registre selectionné par le bus d'adresse ou depuis le registre vers le processeur.
Il est tranmis à travers un circuit d'amplification du type 74LS640 ou équivalent.

- Signal de validation mémoire ($\overline{VMA}$) (actif bas 3 états)

Cette sortie est utilisée pour indiquer à la mémoire externe au processeur qu'il y a une adresse valide sur le bus d'adresse. Aucune action irréversible ne doit être prise avant la phase 2 (ENABLE).

- Signal de validation périphérique ($\overline{VPA}$) (actif bas 3 états)

Cette sortie indique aux circuits périphériques qu'il y a une adresse valide sur les lignes A0.A9 du bus d'adresse. En fonctionnement normal, ce signal doit être utilisé comme signal de validation des périphériques.

- Signal d'activation (ENABLE) actif haut 3 états

Cette sortie est utilisée pour signaler l'utilisation du bus de données.

Utilisée en conjonction avec le signal R/$\bar{\omega}$, elle indique que le processeur attend des données ou présente des données sur le bus.

/...

- Lecture/écriture ($R/\overline{\omega}$) 3 états

  Cette sortie indique aux circuits mémoires et périphériques que le processeur est en lecture (état haut) ou en écriture (état bas).

- Signal Prêt (READY) vrai à l'état haut

  Cette entrée de commande mise a l'état faux permet allongement du signal ENABLE pour une durée maximum de 10 micro-seconds. En fonctionnement normal, elle est fixée à l'état actif par une résistance de tirage.

- Signal d'initialisation ($\overline{\text{RESET}}$) actif bas

  Cette sortie, activée lors de la mise sous tension ou par un bouton pouvoir est généralement utilisée pour mettre les périphériques à l'état initial.

ENTREES D'INTERRUPTION $\overline{\text{NMI}}$, $\overline{\text{IRQ}}$, $\overline{\text{FIRQ}}$

Ces entrées, actives basses, maintenues à l'état inactif par des résistances de tirage sont connectées directement à la broche correspondante du processeur.

ARRET ($\overline{\text{HALT}}$)

Le micro-processeur s'arrête lorsque l'on met cette ligne au niveau bas. Cette entrée, maintenue à l'état inactif par une résistance de tirage est connectée directement à la broche correspondante du processeur.

BUS DISPONIBLE (BA)

Ce signal amplifié, actif haut signale l'état correspondant du processeur, il est utilisé pour mettre toutes les sorties en 3 états sauf le signal d'horloge mémoire et le signal d'initialisation.

HORLOGE MEMOIRE (MEMCLK)

Ce signal est la référence de temps du processeur et sert plus spécialement à déclencher les cycles dans les mémoires dynamiques.

BUS MICRO-ORDINATEUR E 68 XX VU DE L'AVANT

a		b	
GND	1	GND	
AØ	2	A8	
A1	3	A9	
A2	4	A10	
A3	5	A11	ADRESSES
A4	6	A12	
A5	7	A13	
A6	8	A14	ET
A7	9	A15	
AEXTØ	10	AEXT2	EXTENSIONS
AEXT1	11	AEXT3	
$\overline{\text{HALT}}$	12	RESERVE	
MEMCLK	13	ENABLE	
VPA	14	$\overline{\text{RESET}}$	
READY	15	$\overline{\text{NMI}}$	CONTROLE
$\overline{\text{VMA}}$	16	$\overline{\text{IRQ}}$	
R/$\overline{\omega}$	17	$\overline{\text{FIRQ}}$	
BA	18	RESERVE	
$\overline{\text{DRQØ}}$	19	$\overline{\text{DACKØ}}$	
$\overline{\text{DRQ1}}$	20	$\overline{\text{DACK1}}$	A.D.M.
$\overline{\text{DRQ2}}$	21	$\overline{\text{DACK2}}$	
$\overline{\text{DRQ3}}$	22	$\overline{\text{DACK3}}$	
$\overline{\text{D0}}$	23	$\overline{\text{D4}}$	
$\overline{\text{D1}}$	24	$\overline{\text{D5}}$	
$\overline{\text{D2}}$	25	$\overline{\text{D6}}$	
$\overline{\text{D3}}$	26	$\overline{\text{D7}}$	
RESERVE	27	$\overline{\text{PARERR}}$	
RESERVE	28	RESERVE	
-5V	29	+5V Batt	
+12V	30	-12V	
+5V	31	+5V	
0V	32	0V	

Appendix 10 SOURCE CODE FOR PAL IC 28

```
PAL16L8                                          PAL DESIGN SPECIFICATION
P0023                                            PETER PETERSEN 2/12/83
THE FIRST DECODER FOR MANMACHINE MODUL
CERN div. SPS
A23 A22 A21 A20 A19 A18 A17 A16 /ENCINT GND
/IACK /G64EN READ /LDS /G64VPA /G64VMA /MMCCS /VMA /MMCEN VCC

MMCCS    = A23*A22*/A21*/A20*/A19*/A18*/A17*/A16*LDS*/IACK
MMCEN    = A23*A22*/A21*/A20*/A19*/A18*/A17*/A16*LDS*/IACK
         + ENCINT
G64VMA   = A23*A22*/A21*/A20*/A19*/A18*A17*LDS*/IACK*VMA
G64VPA   = A23*A22*/A21*/A20*/A19*/A18*/A17*A16*LDS*/IACK*VMA
G64EN    = A23*A22*/A21*/A20*/A19*/A18*/A17*A16*LDS*/IACK
         + A23*A22*/A21*/A20*/A19*/A18*A17* LDS*/IACK

FUNCTION TABLE:
A23 A22 A21 A20 A19 A18 A17 A16 /ENCINT /IACK /LDS READ /VMA
/G64VMA /G64VPA /G64EN /MMCCS /MMCEN
;
;---------INPUTS---------------------- ------OUTPUTS--------
;
;-ADDRESS /ENCINT /IACK /LDS /READ /VMA -----G64----- --MMC--
;22221111                              /VPA /VMA /EN /CS /EN
;32109876
-------------------------------------------------------------
 HHLLLLLL    H     H     L    H     H     H    H    H    L    L
 HHLLLLLL    H     H     L    L     H     H    H    H    L    L
 HHLLLLLH    H     H     L    H     H     H    H    L    H    H
 HHLLLLLH    H     H     L    H     L     L    H    L    H    H
 HHLLLLHX    H     H     L    L     L     H    L    L    H    H
 HHHLLLLL    H     H     L    H     L     H    H    H    H    H
 HHHHHHHH    L     L     H    H     H     H    H    H    H    L
-------------------------------------------------------------
```

Appendix 11 SOURCE CODE FOR PAL IC 27

```
PAL12L6                                          PAL DESIGN SPECIFICATION
P0023                                            PETER PETERSEN 2/12/83
THE FIRST DECODER FOR MANMACHINE MODUL
CERN div. SPS
A3 A2 A1 /ENCRD /MMCCS READ FC2 FC1 FC0 GND
/LDS /RESET /BUTWR /ENCINT /IACK /ENCEN /CLICK /ENCCLR NC VCC

IACK      = FC0*FC1*FC2
CLICK     = MMCCS*/READ*/A1*/A2*/A3*LDS
ENCEN     = MMCCS*/READ*A1*/A2*/A3*LDS
ENCCLR    = RESET + ENCRD
ENCINT    = FC0*FC1*FC2*LDS*A1*/A2*A3
BUTWR     = MMCCS*/READ*/A1*A2*/A3*LDS

FUNCTION TABLE:

A3 A2 A1 FC2 FC1 FC0 READ /LDS /RESET /ENCRD /MMCCS
/BUTWR /ENCINT /IACK /ENCEN /CLICK /ENCCLR

;
;---------INPUTS------------------  ----------OUTPUTS-----------
;
;ADR   FC   READ /LDS /RE /ENC /MMC  /BUT /ENC /IACK /ENC /CL /ENC
;321   210            SET  RD   CS    WR   INT        EN   ICK  CLR
-------------------------------------------------------------------
 HHH   HHH    H     H    H    H    H    H    H    L    H    H    H
 HLH   HHH    H     L    H    H    H    H    L    L    H    H    H
 LLL   LLL    L     L    H    H    L    H    H    H    H    L    H
 LLH   HLL    L     L    H    H    L    H    H    H    L    H    H
 LHL   LHL    L     L    H    H    L    L    H    H    H    H    H
 LLL   LLH    H     H    L    H    H    H    H    H    H    H    L
 HLH   HLH    H     H    H    L    H    H    H    H    H    H    L
-------------------------------------------------------------------
```

Appendix 12 CAMAC DESCRIPTION

TABLE II CONTACT ALLOCATION AT A NORMAL STATION

(Viewed from front of crate)

			B	A		
Bus-line	Free Bus-line	P1	B	Busy	Bus-line
Bus-line	Free Bus-line	P2	F16	Function	Bus-line
Individual patch contact		P3	F8	Function	Bus-line
Individual patch contact		P4	F4	Function	Bus-line
Individual patch contact		P5	F2	Function	Bus-line
Bus-line	Command Accepted	X	F1	Function	Bus-line
Bus-line	Inhibit	I	A8	Sub-address	Bus-line
Bus-line	Clear	C	A4	Sub-address	Bus-line
Individual line	Station Number	N	A2	Sub-address	Bus-line
Individual line	Look-at-Me	L	A1	Sub-address	Bus-line
Bus-line	Strobe 1	S1	Z	Initialise	Bus-line
Bus-line	Strobe 2	S2	Q	Response	Bus-line
		W24	W23		
		W22	W21		
		W20	W19		
		W18	W17		
24 Write Bus-lines		W16	W15		
		W14	W13		
W1 = least significant bit		W12	W11		
W24 = most significant bit		W10	W9		
		W8	W7		
		W6	W5		
		W4	W3		
		W2	W1		
		R24	R23		
		R22	R21		
		R20	R19		
		R18	R17		
24 Read Bus-lines		R16	R15		
		R14	R13		
R1 = least significant bit		R12	R11		
R24 = most significant bit		R10	R9		
		R8	R7		
		R6	R5		
		R4	R3		
		R2	R1		
	-12V d.c.	-12	-24	-24V d.c.	
	+200V d.c.	+200	-6	-6V d.c.	
Power	117V a.c. Live	ACL	ACN	117V a.c. Neutral	Power
Bus-lines	Reserved	Y1	E	Clean Earth	Bus-lines
	+12V d.c.	+12	+24	+24V d.c.	
	Reserved	Y2	+6	+6V d.c.	
	0V (Power Return)	0	0	0V (Power Return)	

The assignment of contacts at the Dataway connector and their connections to bus-lines, individual lines and patch contacts must be as shown in Table II for normal stations and Table III for the control station. The control station must be to the right of all normal stations.

5.1 Commands

The state of the signals on the individual Station Number lines (specifying a module or modules), the four Sub-address lines (specifying a sub-section of the module) and the five Function lines (specifying the type of operation) constitute a command.

The command signals are maintained for the full duration of the Dataway operation. They are accompanied by a signal on the Busy bus-line, which indicates to all units that a Dataway operation is in progress.

Plug-in units must not rely on the state of signals on the Sub-address and Function lines when no command operation is in progress.

5.1.1 Station Number (N)

Each normal station is addressed by a signal on an individual Station Number line (N_i) which comes from a separate contact at the control station (see Tables II and III). The stations are numbered in decimal from the left-hand end as viewed from the front, beginning with Station 1 (addressed by N1).

There is no restriction on the number of stations that can be addressed simultaneously.

5.1.2 Sub-address (A8, A4, A2, A1)

Different sections of a module are addressed by signals on the four A bus-lines. These signals are decoded in the module to select one of up to sixteen sub-addresses, numbered in decimal from A(0) to A(15).

The sub-address may be used to select, for example, a register within the module, or a feature that is to control the Response signal (Q), or a section of the module that is to be operated on by functions such as Enable, Disable, and Execute. The use made of the sub-address within a module is discussed in relation to the function codes in Section 6.

Each sub-address code used in a module must be fully decoded in the module. Full decoding means that all 4 Dataway Sub-address signals are used in the decoding process.

The sub-address codes are designated A(0), A(1), A(2), A(3) etc. to distinguish them from the individual Sub-address lines A1, A2, A4 and A8. For example, the Sub-address signals A1 = 1, A2 = 1, A4 = 0 and A8 = 0 represent the code A(3).

TABLE IV THE FUNCTION CODES

CODE F( )	FUNCTION	USE OF R AND W LINES	FUNCTION SIGNALS					CODE F( )
			F16	F8	F4	F2	F1	
0	Read Group 1 Register	Functions using the R lines	0	0	0	0	0	0
1	Read Group 2 Register		0	0	0	0	1	1
2	Read and Clear Group 1 Register		0	0	0	1	0	2
3	Read Complement of Group 1 Register		0	0	0	1	1	3
4	Non-standard		0	0	1	0	0	4
5	Reserved		0	0	1	0	1	5
6	Non-standard		0	0	1	1	0	6
7	Reserved		0	0	1	1	1	7
8	Test Look-at-Me	Functions not using the R or W lines	0	1	0	0	0	8
9	Clear Group 1 Register		0	1	0	0	1	9
10	Clear Look-at-Me		0	1	0	1	0	10
11	Clear Group 2 Register		0	1	0	1	1	11
12	Non-standard		0	1	1	0	0	12
13	Reserved		0	1	1	0	1	13
14	Non-standard		0	1	1	1	0	14
15	Reserved		0	1	1	1	1	15
16	Overwrite Group 1 Register	Functions using the W lines	1	0	0	0	0	16
17	Overwrite Group 2 Register		1	0	0	0	1	17
18	Selective Set Group 1 Register		1	0	0	1	0	18
19	Selective Set Group 2 Register		1	0	0	1	1	19
20	Non-standard		1	0	1	0	0	20
21	Selective Clear Group 1 Register		1	0	1	0	1	21
22	Non-standard		1	0	1	1	0	22
23	Selective Clear Group 2 Register		1	0	1	1	1	23
24	Disable	Functions not using the R or W lines	1	1	0	0	0	24
25	Execute		1	1	0	0	1	25
26	Enable		1	1	0	1	0	26
27	Test Status		1	1	0	1	1	27
28	Non-standard		1	1	1	0	0	28
29	Reserved		1	1	1	0	1	29
30	Non-standard		1	1	1	1	0	30
31	Reserved		1	1	1	1	1	31

5.1.3 Function (F16, F8, F4, F2, F1)

The function to be performed at the specified sub-address in the selected module or modules is defined by the signals on the five F bus-lines. These signals are decoded in the module to select one of up to 32 functions, numbered in decimal from F(0) to F(31). The definitions of the 32 function codes are summarised in Table IV and are detailed in Section 6 in relation to the command structure.

In some systems the controller partially decodes the Function signals in order to determine whether a data transfer is required to a module (writing) or from a module (reading). Multiple station addressing allows operations with the same function code to be performed simultaneously in more than one module. These features depend on some standardisation in the assignment of function codes.

The function codes are sub-divided into three groups, involving read operations, write operations, and operations with no transfer of data. The *standard* function codes have defined actions in modules and controllers. There are also *reserved* codes, for any future additions to the standard codes, and *non-standard* codes whose use is neither defined in detail nor co-ordinated by the ESONE Committee.

Each function code used in a module must be fully decoded in the module. Full decoding means that all 5 Dataway Function signals are used in the decoding process.

The function codes are designated F(0), F(1), F(2), F(3) etc. to distinguish them from the individual function lines F1, F2 etc. For example, the Function signals F1 = 1, F2 = 0, F4 = 0, F8 = 1 and F16 = 1 represent the code F(25).

5.2 Strobe Signals (S1 and S2)

During each command operation the controller generates two Strobe signals S1 and S2 in sequence on separate bus-lines. In response to these timing signals plug-in units initiate various actions appropriate to the command that is present on the Dataway.

Both Strobe signals must be generated during each command operation.

Plug-in units must not take irreversible action based on the command or data signals until the time of S1. Actions concerned with the acceptance of data and status information from the R, W, Q and X lines must be initiated at the time of S1 (with the possible exception given below). Other actions may also be timed by S1, but must not change the state of signals on the R and W lines.

Any actions that can change the state of Dataway Read or Write signals must be initiated by the second strobe S2. For example, S2 must be used if it is required to clear a register whose output is connected to the Dataway.

Modules normally accept write data in response to S1, because the data signals are permitted to change at the time of S2. However, modules may accept write data in response to S2 under special circumstances, but this is not recommended.

During unaddressed Dataway operations the controller generates Strobe S2 to indicate when modules accept the common control signal. Strobe S1 may also be generated, but this is not mandatory and modules cannot rely on it.

Strobe S2 must be generated during each unaddressed operation.

5.3 Data

All information carried by the Read and Write lines is conveniently described as *data*, although it may be information concerned with status or control features in modules. Information that is transferred to or from a control register in a module is thus regarded as data.

Up to 24 bits may be transferred in parallel between the controller and the selected module. Independent lines are provided for the read and write directions of transfer.

If the bits of a data word have different numerical significance, line R_n should be used for a higher-order bit than R_{n-1}, and W_n for a higher-order bit than W_{n-1}.

It is recommended that controllers have 24-bit capability. For particular applications assemblies are permitted in which the controller has a word length less than 24 bits and the modules have an equal or smaller word length.

Plug-in units must not rely on the state of signals on the Read and Write bus-lines when no command operation is in progress.

5.3.1 Write Lines (W1—W24)

The controller generates data signals on the W bus-lines during each write operation. The W signals must reach a steady state before S1 and must be maintained until the end of the operation, unless modified by S2. Strobe S1 must be used by the modules to strobe the data unless there are very strong technical reasons for choosing S2 (see Section 5.2).

The W lines serve a few data sources (typically only one controller) and many data receivers.

5.3.2 Read Lines (R1—R24)

Data signals are set up on the R bus-lines by the module during a read operation. The R signals must reach a steady state before S1 and must be maintained for the full duration of the Dataway operation, unless the state of the data source is changed by S2. The controller must initiate action concerned with the acceptance of data from the R-lines at the time of the Strobe S1 and must not take irreversible action before this.

The R lines serve a few data receivers (typically only one controller) and many data sources.

5.4 Status Information

Status information is conveyed by signals on the Look-at-Me (L), Busy (B), Response (Q) and Command Accepted (X) lines.

5.4.1 Look-at-Me (L)

The Look-at-Me lines, like the N lines, are individual connections from each normal station to separate contacts at the control station (L1 from station 1, etc.)

Any module may generate a signal on its individual line (L_i) to indicate that it requires attention. Modules that occupy more than one station may indicate different demands by signals on the appropriate L lines.

The L signal generated by a module may represent demands for attention originating from more than one Look-at-Me source (*LAM source)* in the module. A LAM structure by which various demands can be selected and grouped to form the L signal is shown in Figure 11 and described below. All modules that generate L have the mandatory features shown in Figure 11 and may incorporate additional features for more complex demand handling.

Individual bits of the *LAM status register* are set by the corresponding LAM sources, and are cleared by appropriate commands and by the initialise signal. The outputs *(LAM status)* may be examined collectively by reading the state of all outputs (Read Group 2 at A(12)), or individually by the command Test Status with the appropriate sub-address A(i). Each LAM status should be individually enabled and disabled (for example by a *LAM mask*) to form the corresponding *LAM request.* The LAM requests are examined collectively by reading the state of all LAM requests (Read Group 2 at A(14)), or individually by the command Test LAM with the appropriate sub-address A(i). The internal Look-at-Me signal *(L signal)*, derived from the OR-combination of the LAM requests, is tested by the command Test LAM, with sub-address A(k) distinguishing this from tests of individual LAM requests. Finally, the output of this signal as the *Dataway L signal* is inhibited while the module is addressed by N, possibly in conjunction with certain F and A codes or groups of codes. (See Section 5.4.1.3).

5.4.1.1 Look-at-Me: Clear, Disable and Test

Provision must be made for resetting each bit of the LAM status register individually, either by a Clear Look-at-Me operation F(10) (see Section 6.2.3), or by a Selective Clear Group 2 operation F(23) (see Sections 5.4.1.2 & 6.3.6). All LAM status bits must be reset collectively by the Initialise signal (see Section 5.5.1).

If the LAM request calls for some specific action (for example, reading the contents of a data register) then the corresponding bit of the LAM status register should also be cleared when the appropriate Dataway operation is performed.

A module that has generated L = 1 must not clear the LAM status register until it receives an appropriate command or the Initialise signal.

Each module that generates L should have a means of enabling and disabling the LAM requests. This may be done by loading and clearing a mask register, or by Enable and Disable commands.

All LAM requests that can be disabled by commands must also be disabled by the Initialise signal (Z).

A module that generates L must have a means of testing the L signal by a Test Look-at-Me operation (Function code F(8), with a sub-address distinguishing this from tests of individual LAM requests). If there are several LAM sources the corresponding LAM requests must also be capable of being examined either by Test Look-at-Me operations associated with appropriate sub-addresses or by reading a LAM request pattern in a read operation.

5.4.1.2 Look-at-Me: Commands for Access

Modules may contain registers for LAM information. These registers are not mandatory but if included they should be accessed as Group 2 registers at the following sub-addresses:

LAM Status Register	A(12)
LAM Mask Register	A(13)
LAM Request Register	A(14)

Corresponding bit-positions should be associated with the same LAM source.

The state of each data-bit read from the LAM status or LAM request register is the same as the state of the Q response that would be obtained by a Test Status or Test Look-at-Me operation.

The data word read from A(12) should have LAM status information in the low-order bits and may also contain other status information. Each bit of the data word loaded into A(13) should be in the '1' state to enable the corresponding LAM request, and in the '0' state to disable it.

The operations used to access LAM information may be divided into two classes. One class consists of Read F(1), Write F(17), Clear F(11), Selective Set F(19) & Selective Clear F(23) addressed to the Group 2 LAM registers described above. This class is preferred for operations on modules with many LAM sources. The other class consists of Clear LAM F(10), Enable F(26), Disable F(24), Test Status F(27) and Test LAM F(8) addressed to specific demands. This class is preferred for modules with few LAM sources. In the one class a LAM source, LAM (i), is associated with bit position (i) in data words, and in the other class with sub-address A(i). For ease of programming it is recommended that all the operations related to a particular LAM source should belong to only one class.

5.4.1.3 Look-at-Me: Gating

When a module that is generating $L_i = 1$ receives a command that will cause it to cease doing so, it must inhibit the L signal or the appropriate LAM request. The inhibit condition must be effective before Strobe S1 and must be maintained until the end of the Dataway operation.

This requirement may be met very simply by inhibiting the L signal output when a module is addressed by any command ($L_i = 0$ when $N_i = 1$). Unaddressed modules can thus initiate $L_i = 1$ at any time, but addressed modules cannot do so until the end of the current Dataway operation.

The requirements may be met more precisely by inhibiting only those LAM status signals that are cancelled by the current command. The ability to initiate $L_i = 1$ during a Dataway operation is thus extended to all LAM requests that are not being cancelled. This requires the recognition of $N_i = 1$ with the specific functions and sub-addresses, and the generation of appropriate inhibit conditions.

The requirement may be interpreted in ways intermediate between these extremes. For example, the ability to initiate $L_i = 1$ can be extended to all LAM requests during most Dataway operations if the L signal output is inhibited when the module is addressed by $N_i = 1$ together with appropriate, easily identifiable groups of functions and sub-addresses.

The mandatory statement above replaces the previous requirement in EUR 4100e (1969) that all modules gate off their L signal output during every operation ($L_i = 0$ when $B = 1$). It allows L signals to be initiated at any time, maintained continuously, and removed in advance of Strobes S1 and S2. This counteracts delays elsewhere in systems and leads to improved performance in handling demands for attention.

Units conforming to this specification and to the earlier specification can generally be used together in systems where the improved performance is not required.

Appendix 13 SCAMAC PROGRAM

```
Wylbur-No    ADDR SEC  OPC   EXT   EXT   .    S O U R C E  -  S T A T E M E N T

    1.                          ********************************************************************
    2.                          *                                                                  *
    3.                          *                CAMAC MODULE TO STAC                              *
    4.                          *                                                                  *
    5.                          *   AUTHOR:      OLE BORCH              WRITTEN:    11.09.83       *
    6.                          *                                                                  *
    7.                          *   MODIFIED  BY :                                                 *
    8.                          *                NAME:                  DATE:                      *
    9.                          *                OLE                    7 OCTOBER 1983             *
   10.                          *                Peter                  11 DECEMBER 1983           *
   11.                          *         Changes:  All A, F possible                              *
   12.                          *                   CAMACIRQ routine changed                      *
   13.                          *                   Vectors in EPROM                              *
   14.                          *                   Functions for transfer of status information  *
   15.                          *                        the group 2 registers.                   *
   16.                          *                   Functions for MMC inserted                    *
   17.                          *                   Functions for G64 communiction                *
   18.                          *                                                                 *
   19.                          ********************************************************************
   20.
   21.      000000 G                    IDENT     CAMAC
   23.                                  INSERT                          FILE: $SI.STI.LIB(NODDEF)
    1.   I                      ********************************************************************
    2.   I                      * CAMDEFS - NODAL AND CAMAC, MACROS AND DEFINITIONS                 *
    3.   I                              NOLIST
  253.   I                              LIST
  254.   I                      ********************************************************************
   24.                                  BSHORT
   25.                                  LIST      FORM,CREF
   26.
   27.                          ********************************************************************
   28.                          *                                                                  *
   29.                          *                EXTERN  BLOCK                                     *
   30.                          *                                                                  *
   31.                          ********************************************************************
   32.                                  EXTERN    CAMSTACD,CAMSTACL
   33.                                  EXTERN    MMCPOI,G64POI,MEMPOI
   34.                                  EXTERN    REG2TABS
   35.                                  EXTERN    MMCSTATUS,TINIT,CLICKADR,TIMER
   36.                                  EXTERN    ENCODEN,ENCODMASK,G64VPABASE,G64VMABASE
   37.                                  EXTERN    CINBUF,COUTBUF,CIN,COUT
   38.                                  EXTERN    MSG,DPRINT,NSSET,NODSTKF
   39.
   40.                          ********************************************************************
   41.                          *                                                                  *
   42.                          *                EXCEPTION   VECTOR   ROUTINES                     *
   43.                          *                                                                  *
   44.                          ********************************************************************
   45.      000000 G                    SECTION   G
   46.
```

```
Mylbur-No   ADDR SEC  OPC   EXT   EXT   .   S O U R C E - S T A T E M E N T

  47.                                       * CAMAC INTERRUPT HANDLER
  48.                                       *
  49.                                       * MOVE.W SWITCH,D1                    READ STATUS REGISTER, 5 MODE,I,A,F,N
  50.                                       * CLR.W  SWITCH                       SET LAM
  51.                                       * MOVE.L #CAMAD,A1                    LOAD CAMAC WORD ADDRESS
  52.                                       * MOVE.W D1,(A1)                      WRITE WORD TO CAMAC
  53.                                       * MOVE.W (A1),D1                      READ WORD FROM CAMAC
  54.                                       * MOVE.L #CAMAL,A1                    LOAD CAMAC LONG ADDRESS
  55.                                       * MOVE.L D1,(A1)                      WRITE LONG TO CAMAC
  56.                                       * MOVE.L (A1),D1                      READ LONG FROM CAMAC
  57.                                       * NO BYTE TRANSFER IS POSSIBLE
  58.                                       * PRESSENT NODAL CAMAC COMMAND CAN'T HANDLE LONG WORD TRANSFER
  59.
  60.  000000 G  48E7  E0F8          CAMACIRQ  MOVEM.L  D0-D2/A0-A4,-(A7)
  61.  000004 G  3039  00C0  0040    REASW     MOVE.W   SWITCH,D0                 CLR CAMAC INTERRUPT FLIP-FLOP
  62.  00000A G  4640                         NOT.W    D0                       SWITCH IC. IS 8282
  63.  00000C G  0800  0000                    BTST     #0,D0
  64.  000010 G  6710                         BEQ      CAMACRET                 TEST N
  65.  000012 G  0240  01FE                    ANDI.W   #$1FE,D0
  66.  000016 G  2079  0000  0028+            MOVE.L   CAMTABEL,A0              PONTER TO TABEL OF FUNCTIONS
  67.  00001C G  D0F0  0000                    ADD.W    (A0,D0.W),A0             A0=PONTER TO THE A,F RECEIVED
  68.  000020 G  4E90                         JSR      (A0)                     GO!
  69.  000022 G  4CDF  1F07          CAMACRET  MOVEM.L  (A7)+,D0-D2/A0-A4
  70.  000026 G  4E73                         RTE
  71.
  72.                                       ********************************************************************
  73.                                       *                                                                  *
  74.                                       *                  POINTERS TO CAMAC FUNCTIONS                     *
  75.                                       *                                                                  *
  76.                                       ********************************************************************
  77.             0000  0028+       CAMTABEL  EQU      *
  78.  000028 G  045E  0400  06C2    A0F0      DC.W     F0A0,F1R2,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT
  79.  000038 G  06C2  06C2  06C2    A0F8      DC.W     PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT
  80.  000048 G  046A  0416  06C2    A0F16     DC.W     F16A0,F17R2,PRINT,F19R2,PRINT,PRINT,PRINT,F23R2
  81.  000058 G  0476  048A  049C    A0F24     DC.W     F24A0,F25A0,F26A0,PRINT,PRINT,PRINT,PRINT,PRINT
  82.  000068 G  04BC  0400  06C2    A1F0      DC.W     F0A1,F1R2,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT
  83.  000078 G  06C2  06C2  06C2    A1F8      DC.W     PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT
  84.  000088 G  04D6  0416  06C2    A1F16     DC.W     F16A1,F17R2,PRINT,F19R2,PRINT,PRINT,PRINT,F23R2
  85.  000098 G  06C2  04F0  06C2    A1F24     DC.W     PRINT,F25A1,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT
  86.  0000A8 G  06C2  0400  06C2    A2F0      DC.W     PRINT,F1R2,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT
  87.  0000B8 G  06C2  06C2  06C2    A2F8      DC.W     PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT
  88.  0000C8 G  04FE  0416  06C2    A2F16     DC.W     F16A2,F17R2,PRINT,F19R2,PRINT,PRINT,PRINT,F23R2
  89.  0000D8 G  06C2  06C2  06C2    A2F24     DC.W     PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT
  90.  0000E8 G  0536  0400  06C2    A3F0      DC.W     F0A3,F1R2,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT
  91.  0000F8 G  06C2  06C2  06C2    A3F8      DC.W     PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT
  92.  000108 G  0546  0416  06C2    A3F16     DC.W     F16A3,F17R2,PRINT,F19R2,PRINT,PRINT,PRINT,F23R2
  93.  000118 G  06C2  06C2  06C2    A3F24     DC.W     PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT
  94.  000128 G  06C2  0400  06C2    A4F0      DC.W     PRINT,F1R2,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT
  95.  000138 G  06C2  06C2  06C2    A4F8      DC.W     PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT
  96.  000148 G  06C2  0416  06C2    A4F16     DC.W     PRINT,F17R2,PRINT,F19R2,PRINT,PRINT,PRINT,F23R2
```

```
Wylbur-No    ADDR SEC  OPC   EXT   EXT   .    S O U R C E  -  S T A T E M E N T
    97.      000158 G  06C2  06C2  06C2  A4F24    DC.W     PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT
    98.      000168 G  06C2  0400  06C2  A5F0     DC.W     PRINT,F1R2,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT
    99.      000178 G  06C2  06C2  06C2  A5F8     DC.W     PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT
   100.      000188 G  06C2  0416  06C2  A5F16    DC.W     PRINT,F17R2,PRINT,F19R2,PRINT,PRINT,PRINT,F23R2
   101.      000198 G  06C2  06C2  06C2  A5F24    DC.W     PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT
   102.      0001A8 G  06C2  0400  06C2  A6F0     DC.W     PRINT,F1R2,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT
   103.      0001B8 G  06C2  06C2  06C2  A6F8     DC.W     PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT
   104.      0001C8 G  06C2  0416  06C2  A6F16    DC.W     PRINT,F17R2,PRINT,F19R2,PRINT,PRINT,PRINT,F23R2
   105.      0001D8 G  06C2  06C2  06C2  A6F24    DC.W     PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT
   106.      0001E8 G  06C2  0400  06C2  A7F0     DC.W     PRINT,F1R2,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT
   107.      0001F8 G  06C2  06C2  06C2  A7F8     DC.W     PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT
   108.      000208 G  06C2  0416  06C2  A7F16    DC.W     PRINT,F17R2,PRINT,F19R2,PRINT,PRINT,PRINT,F23R2
   109.      000218 G  06C2  06C2  06C2  A7F24    DC.W     PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT
   110.      000228 G  06C2  0400  06C2  A8F0     DC.W     PRINT,F1R2,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT
   111.      000238 G  06C2  06C2  06C2  A8F8     DC.W     PRINT,PRINT,PRINT,PRINT,F12A8,PRINT,PRINT,PRINT
   112.      000248 G  056A  0416  06C2  A8F16    DC.W     F16A8,F17R2,PRINT,F19R2,PRINT,PRINT,PRINT,F23R2
   113.      000258 G  06C2  06C2  06C2  A8F24    DC.W     PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT
   114.      000268 G  06C2  0400  06C2  A9F0     DC.W     PRINT,F1R2,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT
   115.      000278 G  06C2  06C2  06C2  A9F8     DC.W     PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT
   116.      000288 G  0578  0416  06C2  A9F16    DC.W     F16A9,F17R2,PRINT,F19R2,PRINT,PRINT,PRINT,F23R2
   117.      000298 G  06C2  06C2  06C2  A9F24    DC.W     PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT
   118.      0002A8 G  0556  0400  06C2  A10F0    DC.W     F0A10,F1R2,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT
   119.      0002B8 G  06C2  06C2  06C2  A10F8    DC.W     PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT
   120.      0002C8 G  0584  0416  06C2  A10F16   DC.W     F16A10,F17R2,PRINT,F19R2,PRINT,PRINT,PRINT,F23R2
   121.      0002D8 G  06C2  06C2  06C2  A10F24   DC.W     PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT
   122.      0002E8 G  06C2  0400  06C2  A11F0    DC.W     PRINT,F1R2,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT
   123.      0002F8 G  06C2  06C2  06C2  A11F8    DC.W     PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT
   124.      000308 G  06C2  0416  06C2  A11F16   DC.W     PRINT,F17R2,PRINT,F19R2,PRINT,PRINT,PRINT,F23R2
   125.      000318 G  06C2  06C2  06C2  A11F24   DC.W     PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT
   126.      000328 G  06C2  0400  06C2  A12F0    DC.W     PRINT,F1R2,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT
   127.      000338 G  06C2  06C2  06C2  A12F8    DC.W     PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT
   128.      000348 G  05B0  0416  06C2  A12F16   DC.W     F16A12,F17R2,PRINT,F19R2,PRINT,PRINT,PRINT,F23R2
   129.      000358 G  06C2  06C2  06C2  A12F24   DC.W     PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT
   130.      000368 G  06C2  0400  06C2  A13F0    DC.W     PRINT,F1R2,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT
   131.      000378 G  06C2  06C2  06C2  A13F8    DC.W     PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT
   132.      000388 G  05D2  0416  06C2  A13F16   DC.W     F16A13,F17R2,PRINT,F19R2,PRINT,PRINT,PRINT,F23R2
   133.      000398 G  06C2  06C2  06C2  A13F24   DC.W     PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT
   134.      0003A8 G  06C2  0400  06C2  A14F0    DC.W     PRINT,F1R2,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT
   135.      0003B8 G  06C2  06C2  06C2  A14F8    DC.W     PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT
   136.      0003C8 G  05FE  0416  06C2  A14F16   DC.W     F16A14,F17R2,PRINT,F19R2,PRINT,PRINT,PRINT,F23R2
   137.      0003D8 G  06C2  06C2  06C2  A14F24   DC.W     PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT
   138.      0003E8 G  06C2  0400  06C2  A15F0    DC.W     PRINT,F1R2,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT
   139.      0003F8 G  06C2  06C2  06C2  A15F8    DC.W     PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT
   140.      000408 G  06C2  0416  06C2  A15F16   DC.W     PRINT,F17R2,PRINT,F19R2,PRINT,PRINT,PRINT,F23R2
   141.      000418 G  06C2  06C2  06C2  A15F24   DC.W     PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT,PRINT
   142.
   143.                                  ********************************************************************
   144.                                  *                                                                  *
   145.                                  *        CAMAC FUNCTION USED FOR TRANSFER OF STATUS INFORMATION    *
   146.                                  *                                                                  *
```

```
Wylbur-No   ADDR SEC  OPC   EXT   EXT   .   S O U R C E  -  S T A T E M E N T

147.                                      ********************************************************************
148.                                      * Moves status information between the STAC and CAMAC
149.                                      * F1 read group 2 register
150.                                      * F17 write to a group 2 register
151.                                      * F19 selective clr group 2 register
152.                                      * f23 selective set group 2 register
153.                                      * The subaddress is used as defined in the the CAMAC spec.
154.
155.              0000  0400          F1R2    EQU     *-CAMTABEL
156.      000428 G  0240  03C0                ANDI.W  #$03C0,D0          MASK OUT THE SUB-ADR
157.      00042C G  E848                      LSR.W   #4,D0              USE IT AS LONGWORD POINTER
158.      00042E G  207C  0000  0000*         MOVE.L  #REG2TABS,A0       GROUP 2 REGISTER START
159.      000434 G  23F0  0000                MOVE.L  (A0,D0.W),CAMAL
          000438 G        0080  0000
160.      00043C G  4E75                      RTS
161.
162.              0000  0416          F17R2   EQU     *-CAMTABEL
163.      00043E G  0240  03C0                ANDI.W  #$03C0,D0          MASK OUT THE SUB-ADR
164.      000442 G  E848                      LSR.W   #4,D0              USE IT AS LONGWORD POINTER
165.      000444 G  207C  0000  0000*         MOVE.L  #REG2TABS,A0       GROUP 2 REGISTER START
166.      00044A G  21B9  0080  0000          MOVE.L  CAMAL,(A0,D0.W)
          000450 G        0000
167.      000452 G  4E75                      RTS
168.
169.              0000  042C          F19R2   EQU     *-CAMTABEL
170.      000454 G  0240  03C0                ANDI.W  #$03C0,D0          MASK OUT THE SUB-ADR
171.      000458 G  E848                      LSR.W   #4,D0              USE IT AS LONGWORD POINTER
172.      00045A G  207C  0000  0000*         MOVE.L  #REG2TABS,A0       GROUP 2 REGISTER START
173.      000460 G  2239  0080  0000          MOVE.L  CAMAL,D1
174.      000466 G  83B0  0000                OR.L    D1,(A0,D0.W)
175.      00046A G  4E75                      RTS
176.
177.              0000  0444          F23R2   EQU     *-CAMTABEL
178.      00046C G  0240  03C0                ANDI.W  #$03C0,D0          MASK OUT THE SUB-ADR
179.      000470 G  E848                      LSR.W   #4,D0              USE IT AS LONGWORD POINTER
180.      000472 G  207C  0000  0000*         MOVE.L  #REG2TABS,A0       GROUP 2 REGISTER START
181.      000478 G  2239  0080  0000          MOVE.L  CAMAL,D1
182.      00047E G  4681                      NOT.L   D1
183.      000480 G  C3B0  0000                AND.L   D1,(A0,D0.W)
184.      000484 G  4E75                      RTS
185.
186.
187.                                      ********************************************************************
188.                                      *                                                                  *
189.                                      *          CAMAC FUNCTION USED IN MAN-MACHINE COMMUNICATION        *
190.                                      *                                                                  *
191.                                      ********************************************************************
192.
193.                                      * READ POINTER TO MMC-STATUS BLOCK
194.              0000  045E          F0A0    EQU     *-CAMTABEL
```

```
Mylbur-No    ADDR SEC  OPC   EXT   EXT   .   S O U R C E  -  S T A T E M E N T

  195.       000486 G  23F9  0000  0000*          MOVE.L    MMCPOI,CAMAL
             00048C G        0080  0000
  196.       000490 G  4E75                       RTS
  197.
  198.                                    * WRITE POINTER TO MMC-STATUS BLOCK
  199.                 0000  046A         F16A0    EQU       *-CAMTABEL
  200.       000492 G  23F9  0080  0000           MOVE.L    CAMAL,MMCPOI
             000498 G        0000  0000*
  201.       00049C G  4E75                       RTS
  202.
  203.                                    * DISABLE THE MAN-MACHINE COMMUNICATION
  204.                 0000  0476         F24A0    EQU       *-CAMTABEL
  205.       00049E G  4279  0080  0002           CLR.W     CAMAD               GIVE Q-RESPONS
  206.       0004A4 G  4279  0000  0000*         CLR.W     TIMER               DISABLE TIMER INTERRUPT
  207.       0004AA G  4239  0000  0000*         CLR.B     ENCODEN             DISABLE ENCODER INTERRUPT
  208.       0004B0 G  4E75                       RTS
  209.
  210.                                    * SET THE NEW ENCODER INTERRUPT MASK FROM ENCODMASK
  211.                 0000  048A         F25A0    EQU       *-CAMTABEL
  212.       0004B2 G  4279  0080  0002           CLR.W     CAMAD               GIVE QRESPONS
  213.       0004B8 G  13F9  0000  0001*         MOVE.B    ENCODMASK+1,ENCODEN
             0004BE G        0000  0000*
  214.       0004C2 G  4E75                       RTS
  215.
  216.                                    * ENABLE THE MAN-MACINE COMMUNICATION
  217.                 0000  049C         F26A0    EQU       *-CAMTABEL
  218.       0004C4 G  4279  0080  0002           CLR.W     CAMAD               GIVE Q-RESPONS
  219.       0004CA G  13F9  0000  0001*         MOVE.B    ENCODMASK+1,ENCODEN ENABLE THE ENCODERS
             0004D0 G        0000  0000*
  220.       0004D4 G  33FC  000A                 MOVE.W    #10,TINIT           INITIALISE TOUCHSCREEN (20MS)
             0004D8 G        0000  0000*
  221.       0004DC G  4A79  0000  0000*         TST.W     TIMER               ENABLE TIMER INTERRUPT
  222.       0004E2 G  4E75                       RTS
  223.
  224.                                    * READ THE VALUE THE MMC-POINTER POINTS AT
  225.                                    * CAN ONLY READ WITHIN 256 BYTES FROM MMC-STATUS
  226.                 0000  04BC         F0A1     EQU       *-CAMTABEL
  227.       0004E4 G  2039  0000  0000*         MOVE.L    MMCPOI,D0
  228.       0004EA G  0240  00FF                 ANDI.W    #$00FF,D0
  229.       0004EE G  207C  0000  0000*         MOVE.L    #MMCSTATUS,A0
  230.       0004F4 G  33F0  0000                 MOVE.W    (A0,D0.W),CAMAD     OUTPUT = WORD
             0004F8 G        0080  0002
  231.       0004FC G  4E75                       RTS
  232.
  233.                                    * WRITE TO THE REGISTER MMC-POINTER POINTS AT
  234.                                    * CAN ONLY DO HARM WITHIN 256 BYTES FROM MMC-STATUS
  235.                 0000  04D6         F16A1    EQU       *-CAMTABEL
  236.       0004FE G  2039  0000  0000*         MOVE.L    MMCPOI,D0
  237.       000504 G  0240  00FF                 ANDI.W    #$00FF,D0
  238.       000508 G  207C  0000  0000*         MOVE.L    #MMCSTATUS,A0
```

```
Wylbur-No    ADDR SEC  OPC   EXT   EXT   .    S O U R C E  -  S T A T E M E N T

  239.       00050E G  31B9  0080  0002          MOVE.W    CAMAD,(A0,D0.W)     INPUT = WORD
             000514 G        0000
  240.       000516 G  4E75                      RTS
  241.
  242.                                     * MAKE A CLICK (SOUND)
  243.                 0000  04F0          F25A1 EQU       *-CAMTABEL
  244.       000518 G  4239  0000  0000*         CLR.B     CLICKADR
  245.       00051E G  4279  0080  0002          CLR.W     CAMAD               GIVE Q-RESPONS
  246.       000524 G  4E75                      RTS
  247.
  248.                                     ************************************************************************
  249.                                     *                                                                      *
  250.                                     *          CAMAC FUNCTION USED IN G64 COMMUNICATION                    *
  251.                                     *                                                                      *
  252.                                     ************************************************************************
  253.
  254.                                     * WRITE ADDRESS TO G64
  255.                                     * BIT 0-15  : ADDRESS
  256.                                     * BIT 16    : 0-MEMORY / 1-PERIPHERAL
  257.                 0000  04FE          F16A2 EQU       *-CAMTABEL
  258.       000526 G  2039  0080  0000          MOVE.L    CAMAL,D0
  259.       00052C G  0800  0010                BTST      #16,D0
  260.       000530 G  6716                      BEQ       G64MEM
  261.       000532 G  0280  0000  02FF          ANDI.L    #$000002FF,D0       ADDRESS A0-9
  262.       000538 G  E388                      LSL.L     #1,D0
  263.       00053A G  0680  0000  0000*         ADDI.L    #G64VPABASE,D0
  264.       000540 G  23C0  0000  0000*         MOVE.L    D0,G64POI
  265.       000546 G  4E75                      RTS
  266.       000548 G  0280  0000  FFFF  G64MEM  ANDI.L    #$0000FFFF,D0
  267.       00054E G  E388                      LSL.L     #1,D0
  268.       000550 G  0680  0000  0000*         ADDI.L    #G64VMABASE,D0
  269.       000556 G  23C0  0000  0000*         MOVE.L    D0,G64POI
  270.       00055C G  4E75                      RTS
  271.
  272.                                     * READ FROM A G64 CARD, THE ADDRESS IS IN G64POI
  273.                 0000  0536          F0A3  EQU       *-CAMTABEL
  274.       00055E G  2079  0000  0000*         MOVE.L    G64POI,A0
  275.       000564 G  1010                      MOVE.B    (A0),D0
  276.       000566 G  33C0  0080  0002          MOVE.W    D0,CAMAD
  277.       00056C G  4E75                      RTS
  278.
  279.                                     * WRITE A BYTE TO A G64 CARD, THE ADDRESS IS IN G64POI
  280.                 0000  0546          F16A3 EQU       *-CAMTABEL
  281.       00056E G  3039  0080  0002          MOVE.W    CAMAD,D0
  282.       000574 G  2079  0000  0000*         MOVE.L    G64POI,A0
  283.       00057A G  1080                      MOVE.B    D0,(A0)
  284.       00057C G  4E75                      RTS
  285.
  286.                                     ************************************************************************
  287.                                     *                                                                      *
```

```
Mylbur-No    ADDR SEC  OPC   EXT   EXT   .    S O U R C E  -  S T A T E M E N T

  288.                                     *         CAMAC FUNCTION USED FOR GENERAL PURPOSE COMUNICATION     *
  289.                                     *         CREATED BY OLE                                           *
  290.                                     *                                                                  *
  291.                                     ********************************************************************
  292.                                     *READ WORD FROM MEMORY LOCATION GIVEN IN MEMORY ADDRESS POINTER
  293.                                     *AND INCREMENT POINTER WITH 2
  294.                0000  0556           F0A10     EQU       *-CAMTABEL
  295.      00057E G  2079  0000  0000*               MOVE.L    MEMPOI,A0
  296.      000584 G  33D0  0080  0002                MOVE.W    (A0),CAMAD
  297.      00058A G  54B9  0000  0000*               ADDQ.L    #2,MEMPOI
  298.      000590 G  4E75                            RTS
  299.                                     *CAMAC OUTPUT.W = (CAMAC INPUT.L)
  300.                0000  056A           F16A8     EQU       *-CAMTABEL
  301.      000592 G  2079  0080  0000                MOVE.L    CAMAL,A0
  302.      000598 G  33D0  0080  0002                MOVE.W    (A0),CAMAD
  303.      00059E G  4E75                            RTS
  304.                                     *WRITE MEMORY ADDRESS POINTER
  305.                0000  0578           F16A9     EQU       *-CAMTABEL
  306.      0005A0 G  23F9  0080  0000                MOVE.L    CAMAL,MEMPOI
            0005A6 G        0000  0000*
  307.      0005AA G  4E75                            RTS
  308.                                     *WRITE WORD TO MEMORY LOCATION GIVEN IN MEMORY ADDRESS POINTER
  309.                                     *AND INCREMENT POINTER WITH 2
  310.                0000  0584           F16A10    EQU       *-CAMTABEL
  311.      0005AC G  2079  0000  0000*               MOVE.L    MEMPOI,A0
  312.      0005B2 G  30B9  0080  0002                MOVE.W    CAMAD,(A0)
  313.      0005B8 G  54B9  0000  0000*               ADDQ.L    #2,MEMPOI
  314.      0005BE G  4E75                            RTS
  315.                                     *SET STAC INTO TRANSP MODE. CONNECT CAMAC AND HOST BUFFERS.
  316.                                     *BREAK OUT IF ESCAPE RECEIVED FROM CAMAC.
  317.                0000  0598           F12A8     EQU       *-CAMTABEL
  318.      0005C0 G  4A79  0080  0002                TST.W     CAMAD                    GIVE Q TO ICC
  319.      0005C6 G  027C  F8FF                      ANDI.W    #$F8FF,SR                DOWN TO LEVEL 0
  320.      0005CA G                                  CALL      SLTRANSP,F16A11ER
  321.      0005D4 G  4E75                            RTS
  322.      0005D6 G  4E75                F16A11ER  RTS                                NO ERROR RETUR YET
  323.                                     *ICC TRAP2 HANDLER: RECIEVE DATA FROM CAMAC
  324.                0000  05B0           F16A12    EQU       *-CAMTABEL
  325.      0005D8 G  43F9  0000  0000*               LEA       CINBUF,A1
  326.      0005DE G  45F9  0080  0002                LEA       CAMAD,A2
  327.      0005E4 G  3012                            MOVE.W    (A2),D0                  DATA IN
  328.      0005E6 G  1229  0004                      MOVE.B    4(A1),D1                 GET STATUS
  329.      0005EA G  0801  0000                      BTST      #0,D1
  330.      0005EE G  6606                            BNE       F16A12RE                 JMP IF CINBUF IS FULL
  331.      0005F0 G  4EB9  0000  0000*               JSR       CIN                      PUT D0 TO CINBUF
  332.      0005F6 G  3481                F16A12RE  MOVE.W    D1,(A2)                  CINSTATUS OUT ON CAMAC
  333.      0005F8 G  4E75                            RTS
  334.                                     *ICC TRAP1 HANDLER: SEND DATA TO CAMAC
  335.                0000  05D2           F16A13    EQU       *-CAMTABEL
  336.      0005FA G  45F9  0000  0000*               LEA       COUTBUF,A2
```

```
Wylbur-No    ADDR SEC  OPC   EXT   EXT   .   S O U R C E - S T A T E M E N T
  337.       000600 G  43F9  0080  0000         LEA       CAMAL,A1
  338.       000606 G  3029  0002               MOVE.W    2(A1),D0          GIVE Q TO ICC
  339.       00060A G  4280                     CLR.L     D0
  340.       00060C G  122A  0004               MOVE.B    4(A2),D1          FETCH STATUS
  341.       000610 G  0801  0001               BTST      #1,D1
  342.       000614 G  660A                     BNE       F16A13ER          JMP IF EMPTY
  343.       000616 G  4EB9  0000  0000*        JSR       COUT
  344.       00061C G  2280              F16A13RE MOVE.L   D0,(A1)           GIVE SOME STATUS AND DATA
  345.       00061E G  4E75                     RTS
  346.       000620 G  08C0  0010        F16A13ER BSET     #16,D0            EMPTY OR NOT VALID STATUS
  347.       000624 G  60F6                     BRA       F16A13RE
  348.                                *ICC TRAP00 HANDLER: SEND INFORMATION TO CAMAC
  349.                 0000  05FE        F16A14   EQU       *-CAMTABEL
  350.       000626 G  49F9  0080  0000         LEA       CAMAL,A4
  351.       00062C G  302C  0002               MOVE.W    2(A4),D0          GET D1,D0 FROM CAMAC
  352.       000630 G  3400                     MOVE.W    D0,D2             SAVE GOT DATA
  353.       000632 G  0240  00FF               ANDI.W    #$FF,D0
  354.       000636 G  E04A                     LSR.W     #8,D2             FETCH D1
  355.       000638 G  3202                     MOVE.W    D2,D1             NOW RDY. FOR TRAP00
  356.       00063A G                           CALLTRAP  0,F16A14ER         DO TRAP0 ORDERED FROM ICC
  357.       000640 G  0880  0010               BCLR      #16,D0            DO WAS OK
  358.       000644 G  2880              F16A14RE MOVE.L   D0,(A4)           WRITE RESULT TO CAMAC
  359.       000646 G  4E75                     RTS
  360.       000648 G  08C0  0010        F16A14ER BSET     #16,D0            DO WAS NOT OK
  361.       00064C G  60F6                     BRA       F16A14RE
  362.                                *SUBROUTINE TO F16A11
  363.                                *SLTRANSP - GO INTO TRANSP MODE BY:
  364.                                *           CONNECTING CAMAC BUFFERS TO HOST BUFFERS.
  365.                                *ESCAPE FROM CAMAC BREAKS OUT.
  366.                                *THE TRANSP IS DONE FROM LEVEL 0
  367.       00064E G                           PROGM
  368.       00064E G                           STRING    TRANSTR
  369.       00064E G                 SLTRANSP  ENTER
  370.       000662 G  41EE  FF7E               LEA       TRANSTR(A6),A0
  371.       000666 G  4EB9  0000  0000*        JSR       NSSET
  372.       00066C G  317C  0001  0008         MOVE.W    #1,8(A0)          ONE BYTE
  373.       000672 G  7203                     MOVEQ     #3,D1
  374.       000674 G  7007                     MOVEQ     #7,D0
  375.       000676 G                           CALLTRAP  0,SLTER            CLR HOST INPUT
  376.       00067C G  7203              SLTI   MOVEQ     #3,D1
  377.       00067E G  7000                     MOVEQ     #0,D0
  378.       000680 G                           CALLTRAP  0,SLTER            GET HOST INPUT STATUS
  379.       000686 G  0800  0001               BTST      #1,D0
  380.       00068A G  6614                     BNE       CAMIN99           BRA IF NO CHR ON HOST INPUT
  381.       00068C G  42A8  0004               CLR.L     4(A0)
  382.       000690 G                           CALLTRAP  1,SLTER            READ FROM HOST
  383.       000696 G  7208                     MOVEQ     #8,D1
  384.       000698 G                           CALLTRAP  2,SLTER            WRITE TO CAMAC
  385.       00069E G  60DC                     BRA       SLTI
  386.       0006A0 G  7207              CAMIN99  MOVEQ    #7,D1
```

```
Wylbur-No    ADDR SEC  OPC   EXT   EXT   .   S O U R C E  -  S T A T E M E N T
  387.       0006A2 G  7000                     MOVEQ     #0,D0
  388.       0006A4 G                           CALLTRAP  0,SLTER          GET CAMAC INPUT STATUS
  389.       0006AA G  0800  0001               BTST      #1,D0
  390.       0006AE G  66CC                     BNE       SLTI             BRA IF EMPTY
  391.       0006B0 G  0800  0002               BTST      #2,D0
  392.       0006B4 G  6616                     BNE       SLESC            BRA IF ESCAPE
  393.       0006B6 G  42A8  0004               CLR.L     4(A0)
  394.       0006BA G  7207                     MOVEQ     #7,D1
  395.       0006BC G                           CALLTRAP  1,SLTER          READ FROM CAMAC
  396.       0006C2 G  7204                     MOVEQ     #4,D1
  397.       0006C4 G                           CALLTRAP  2,SLTER          WRITE TO HOST
  398.       0006CA G  60B0                     BRA       SLTI
  399.       0006CC G  7007             SLESC   MOVEQ     #7,D0
  400.       0006CE G                           CALLTRAP  0,SLTER          EMPTY CAMAC INPUT
  401.       0006D4 G                           RET
  402.       0006E0 G                   SLTER   ERRET
  403.
  404.                                          *PRINT F,A FOR NOT IMPLEMENTED CAMAC VECTORS
  405.                 0000  06C2          PRINT   EQU       *-CAMTABEL
  406.       0006EA G  48E7  FFFE                 MOVEM.L   D0-D7/A0-A6,-(A7)
  407.       0006EE G  4A79  0080  0002           TST.W     CAMAD            OPEN FOR NEW SWITCH INPUT
  408.       0006F4 G  2200                       MOVE.L    D0,D1
  409.       0006F6 G  E489                       LSR.L     #2,D1
  410.       0006F8 G  0681  0000  0100           ADDI.L    #$100,D1
  411.       0006FE G  41FA  0036                 LEA       MASSACA(PC),A0
  412.       000702 G  4EB9  0000  0000*          JSR       MSG
  413.       000708 G  2001                       MOVE.L    D1,D0
  414.       00070A G  0280  0000  001F           ANDI.L    #$1F,D0
  415.       000710 G  4EB9  0000  0000*          JSR       DPRINT
  416.       000716 G  4EB9  0000  0000*          JSR       MSG
  417.       00071C G  EA89                       LSR.L     #5,D1
  418.       00071E G  2001                       MOVE.L    D1,D0
  419.       000720 G  4EB9  0000  0000*          JSR       DPRINT
  420.       000726 G  4EB9  0000  0000*          JSR       MSG
  421.       00072C G  4CDF  7FFF                 MOVEM.L   (A7)+,D0-D7/A0-A6
  422.       000730 G  4E75                       RTS
  423.
  424.                                          ************************************************************************
  425.                                          *                                                                      *
  426.                                          *                    MESSAGES                                          *
  427.                                          *                                                                      *
  428.                                          ************************************************************************
  429.       000732 G  FFFF  FFFF                 DC.L      -1
  430.       000736 G  2446  554E  4354 MASSACA   DC.B      '$FUNCTION(HEX)=?  SUBADDRESS(HEX)=?$?'
  431.
  432.                                          ************************************************************************
  433.                                          *                                                                      *
  434.                                          *                    ENTRY  BLOCK                                      *
  435.                                          *                                                                      *
  436.                                          ************************************************************************
```

```
SYMBOL NO   ADDR SEG  G/C  EXT  EXT  .  S O U R C E  -  S T A T E M E N T

437.                                        ENTRY     CAMACIRQ
438.
439.        00075C G                        END
```

Appendix 14 MMC PROGRAM

```
Wylbur-No    ADDR SEC  OPC   EXT   EXT   .    S O U R C E  -  S T A T E M E N T
    1.                                       ************************************************************************
    2.                                       *                                                                      *
    3.                                       *                MAN MACHINE COMMUNICATION - STAC                      *
    4.                                       *                                                                      *
    5.                                       *   AUTHOR:      PETER PETERSEN        WRITTEN:    9.12.83           *
    6.                                       *                                                                      *
    7.                                       *   MODIFYED  BY :                                                     *
    8.                                       *                NAME:                 DATE:                           *
    9.                                       *                PETER                                                 *
   10.                                       *                                                                      *
   11.                                       ************************************************************************
   12.
   13.                                       *      STOUCH reads data from the touch-screen and if there has been
   14.                                       * a certain change, which mean that the button is touched the rutine
   15.                                       * saves the button number and if the crate controller permits, the
   16.                                       * routine raises a LAM
   17.                                       *      The routine incorborates adjustment of longtime drift in the
   18.                                       * touchscreen capacitors.
   19.                                       *
   20.      000000 G                                    IDENT     MMC-STAC
   21.      000000 G                         STOUCH     SECTION   G                    LOAD ADDRES
   23.                                                  BSHORT
   24.                                                  INSERT                  FILE: $SI.STI.LIB(NODDEF)
    1.   I                                   ************************************************************************
    2.   I                                   * CAMDEFS - NODAL AND CAMAC, MACROS AND DEFINITIONS                    *
    3.   I                                            NOLIST
  253.   I                                            LIST
  254.   I                                   ************************************************************************
   25.                                       *
   26.                                                  LIST      FORM,CREF
   27.
   28.                                       * Variable area used
   29.                                       *
   30.                                       * TINIT   DS.W    1              Counter used in upstart
   31.                                       * TCUR    DS.W    1              The latest button number
   32.                                       * TOLD    DS.W    1              The latest button written out
   33.                                       * TTIMES  DS.W    1              The number of times the routine
   34.                                       *                                  has detected a button.
   35.                                       * TOFTIM  DS.W    1              The number of times the routine
   36.                                       *                                  has detected no button touched
   37.                                       * TCOUNT  DS.W    1              Number of TOUCHCHECK calls since
   38.                                       *                                  the last update of TMEAN
   39.                                       * TMEAN   DS.W   16              Here the values of the untouch-
   40.                                       *                                  ed capacitors are saved
   41.                                       * TON     DS.W   16              TON(I) = TMEAN(I)-HYSTON
   42.                                       * TOFF    DS.W   16              TOFF(I) = TMEAN(I)-HYSTOFF
   43.                                       * TDIF    DS.W   16              The accumulated differens be-
   44.                                       *                                  the current value and the mean
   45.                                       *                                  value (touched capacitor omitted)
   46.                                       *
```

```
Wylbur-No   ADDR SEC  OPC   EXT   EXT   .   S O U R C E  -  S T A T E M E N T
47.                          * MECHVAR  EQU      *              Variables for the mechanical
48.                          *                                 buttons.
49.                          * MCUR     DS.W     1              The latest buttonnumber
50.                          * MBUTOLD  DS.W     1              The latest button written out
51.                          * MTIMES   DS.W     1              The number of times the routine
52.                          *                                 has detected a button
53.                          * MOFTIMES DS.W     1              The number of times the routine
54.                          *                                 has detected no buttons touched
55.                          *
56.                          * ENCODINTERVAL  DS.W  1           The number of ENCODCHECK calls
57.                          *                                 since the last CAMAC update.
58.                          *
59.                          * Variable register for communication with CAMAC
60.                          *
61.                          *
62.                          * MMCSTATUS   EQU  *              Reg. for CAMAC - MMC communication
63.                          * TOUCHOUT    DS.W  1             Output from the touchscreen
64.                          * TOUCHMASK   DS.W  1             Mask to disable touchbuttons
65.                          * MECHOUT     DS.W  1             Output from mechanical buttons
66.                          * MECHMASK    DS.W  1             Mask to disable mechanical buttons
67.                          * ENCODMASK   DS.W  1             Mask to disable the encoders
68.                          *            DS.W  3
69.                          *
70.                          * ENCODOUT    DS.W  8             THE ENCODER OUTPUTS TO CAMAC
71.                          *
72.                          * ENCODLOW    DS.W  8             THE LOW LIMIT FOR ENCODERS
73.                          *
74.                          * ENCODHIGH   DS.W  8             THE HIGH LIMIT FOR ENCODERS
75.                          * ENCODSCALE  DS.W  8             THE SCALE FACTOR FOR ENCODERS
76.                          * ENCODCOUNT  DS.L  8             THE SOFTWARE COUNTER REGISTER
77.                          *
78.                          ********************************************************************
79.                          *                                                                  *
80.                          *               EXTERN  BLOCK                                      *
81.                          *                                                                  *
82.                          ********************************************************************
83.                                    EXTERN    TINIT,TCUR,TOLD
84.                                    EXTERN    TTIMES,TOFTIM,TCOUNT
85.                                    EXTERN    TMEAN,TON,TOFF,TDIF
86.                                    EXTERN    MECHVAR,ENCODINTERVAL
87.                                    EXTERN    LAMSTATUS,LAMMASK,LAMREQ
88.                                    EXTERN    TIMER,ADCADR,MECHADR1,MECHADR2,ENCODADR
89.                                    EXTERN    NODSTKF,SWITCH,ENCODEN
90.                                    EXTERN    TOUCHOUT,TOUCHMASK,MECHOUT,MECHMASK
91.                                    EXTERN    MMCSTATUS,ENCODOUT,ENCODLOW,ENCODHIGH
92.                                    EXTERN    ENCODCOUNT
93.
94.                          ********************************************************************
95.                          *                                                                  *
96.                          *               ENTRY BLOCK                                        *
```

```
Wylbur-No   ADDR SEC  OPC   EXT   EXT   .   S O U R C E  -  S T A T E M E N T
   97.                                     *                                                                *
   98.                                     ******************************************************************
   99.                                              ENTRY     TIMERIRQ,MMCINIT
  100.                                              ENTRY     ENCOD0,ENCOD1,ENCOD2,ENCOD3
  101.                                              ENTRY     ENCOD4,ENCOD5,ENCOD6,ENCOD7
  102.
  103.                        0000  0005            ONDELAY   EQU       5
  104.                        0000  000A            OFFDELAY  EQU       10
  105.                        0000  0019            ENCTIME   EQU       25                  25 * 2 msec.
  106.
  106.01                                   ******************************************************************
  106.02                                   *                                                                *
  106.03                                   *          INITIALISE THE MMC-MODULE                             *
  106.04                                   *                                                                *
  106.05                                   ******************************************************************
  106.06
  106.07   000000 G   33FC  03E8            MMCINIT   MOVE.W    #1000,TINIT         WAIT 2 SECONDS (TIMER 2msec.)
           000004 G         0000  0000*
  106.08   000008 G   4239  0000  0000*              CLR.B     ENCODEN             DISABLE ENCODERS
  106.09   00000E G   4A79  0000  0000*              TST.W     TIMER               ENABLE TIMER INTERRUPT
  106.1    000014 G   4E75                         RTS
  106.11
  107.                                     ******************************************************************
  108.                                     *                                                                *
  109.                                     *                TIMER INTERRUPT ROUTINE - level 2               *
  110.                                     *                                                                *
  111.                                     ******************************************************************
  112.
  113.     000016 G   48E7  FFC0            TIMERIRQ  MOVEM.L   D0-D7/A0-A1,-(A7)
  114.     00001A G   4279  0000  0000*              CLR.W     TIMER               Clear interrupt
  115.     000020 G   3079  0000  0000*              MOVE.W    TIMER,A0            and enable interrupt again
  116.     000026 G   5379  0000  0000*              SUBQ.W    #1,TINIT            IF NOT SETTLED
  117.     00002C G   6B06                           BMI       SETTLED             THEN
  118.     00002E G   6100  003A                     BSR.L     TOUCHINI              Initialise all values
  119.     000032 G   6004                           BRA.S     MECHSERV              skip touchscreen service
  120.                                      *                                      ENDIF
  121.     000034 G   6100  0084            SETTLED   BSR.L     TOUCHCHECK          SERVICE THE TOUCHSCREEN
  122.     000038 G   6100  017A            MECHSERV  BSR.L     MECHCHECK           SERVICE THE MECHANICAL BUTTONS
  123.     00003C G   5379  0000  0000*              SUBQ.W    #1,ENCODINTERVAL    EVERY 25 TIMES - UPDATE EN-
  124.     000042 G   6B06                           BMI       LAMCHECK              CODER VALUES FOR CAMAC
  125.     000044 G   4EB9  0000  025A+              JSR       ENCODCHECK
  126.     00004A G   3039  0000  0000* LAMCHECK     MOVE.W    LAMSTATUS,D0        IF A BIT IS SET IN THE LAMSTATUS
  127.     000050 G   C079  0000  0000*              AND.W     LAMMASK,D0          AND IT IS ENABLED
  128.     000056 G   33C0  0000  0000*              MOVE.W    D0,LAMREQ           THEN IT IS SET IN THE LAMREQUEST
  129.     00005C G   6706                           BEQ       RUNEND3             IF ANY LAMREQ-BIT IS SET
  130.     00005E G   4279  00C0  0040               CLR.W     SWITCH              THEN SET LAM
  131.     000064 G   4CDF  03FF            RUNEND3   MOVEM.L   (A7)+,D0-D7/A0-A1   RESTORE REGISTERS
  132.     000068 G   4E75                           RTE                           RETURN
  133.
  134.                                      ******************************************************************
```

```
Wylbur-No    ADDR SEC  OPC   EXT   EXT   .   S O U R C E  -  S T A T E M E N T

                                           *                                                              *
                                           * TOUCHINI - Initialises the TMEAN,TON,TOFF variables         *
                                           *                                                              *
                                           ****************************************************************

00006A G   207C  0000  0000* TOUCHINI  MOVE.L   #TMEAN,A0          Pointer to TOUCH-var block
000070 G   227C  0000  0000*           MOVE.L   #ADCADR,A1         Pointer to the 2 ADC's
000076 G   317C  FFFF  FFF6            MOVE.W   #-1,-10(A0)        Clear TCUR (=-1)
00007C G   317C  FFFF  FFF8            MOVE.W   #-1,-8(A0)         Clear TOLD (=-1)
000082 G   4268  FFFA                  CLR.W    -6(A0)             Clear TTIMES
000086 G   317C  000A  FFFC            MOVE.W   #OFFDELAY,-4(A0)   Set TOFTIM = OFFDELAY
00008C G   317C  0400  FFFE            MOVE.W   #$400,-2(A0)       Set TCOUNT = 400
000092 G   700F                        MOVEQ    #15,D0             FOR I=15 DOWNTO 0 DO
000094 G   3219                INILOOP MOVE.W   (A1)+,D1             read a button value from ADC
000096 G   0241  00FF                  AND.W    #$00FF,D1           MASK d8-d15 OUT
00009A G   3081                        MOVE.W   D1,(A0)             save the values in TMEAN(I)
00009C G   3401                        MOVE.W   D1,D2               D2=TMEAN(I)
00009E G   E649                        LSR.W    #3,D1               TON(I)=TMEAN(I)*87.5%
0000A0 G   9441                        SUB.W    D1,D2
0000A2 G   3142  0020                  MOVE.W   D2,32(A0)
0000A6 G   E249                        LSR.W    #1,D1               TOFF(I)=TMEAN(I)*93.25%
0000A8 G   D441                        ADD.W    D1,D2
0000AA G   3142  0040                  MOVE.W   D2,64(A0)
0000AE G   4268  0060                  CLR.W    96(A0)              Clear TDIF
0000B2 G   5488                        ADDQ.L   #2,A0
0000B4 G   51C8  FFDE                  DBRA     D0,INILOOP          ENDFOR
0000B8 G   4E73                        RTS
                                           ****************************************************************
                                           *                                                              *
                                           *          TOUCHCHECK                                          *
                                           *                                                              *
                                           ****************************************************************
                                           *
                                           *
                                           * TOUCHCHECK - Reads al the values from the ADC's and if there has
                                           * been a change, the button number is put in memory location TOUCHOUT
                                           * and LAM set in the LAM-status-reg.
                                           * The crate controller can then get button number through CAMAC
                                           *
                                           *
                                           * REGISTERS USED
                                           *
                                           * D0 - counter
                                           * D1 - TNEW(I)
                                           * D2 - TCUR
                                           * D3 - TOFF(I) / TON(I)
                                           * D4 - counter*2
                                           * D5 - TTIMES
                                           * D6 - TOFTIM
                                           * D7 - TOLD
```

```
Wylbur-No    ADDR SEC  OPC   EXT   EXT   .    S O U R C E  -  S T A T E M E N T

  185.                                       *
  186.                                       * A0 - TMEAN pointer
  187.                                       * A1 - ADCADR pointer
  188.
  189.       0000BA G   207C  0000  0000* TOUCHCHECK  MOVE.L  #TMEAN,A0          Pointer to TOUCH-var block
  190.       0000C0 G   227C  0000  0000*             MOVE.L  #ADCADR,A1        Pointer to the 2 ADC's
  191.       0000C6 G   3428  FFF6                    MOVE.W  -10(A0),D2        D2=TCUR
  192.       0000CA G   3E28  FFF8                    MOVE.W  -8(A0),D7         D7=TOLD
  193.       0000CE G   3A28  FFFA                    MOVE.W  -6(A0),D5         D5=TTIMES
  194.       0000D2 G   3C28  FFFC                    MOVE.W  -4(A0),D6         D6=TOFTIM
  195.       0000D6 G   700F                          MOVEQ   #15,D0
  196.                                  * For every button do                  FOR I=15 DOWNTO 0 DO
  197.       0000D8 G   3800            TOUCHLOOP MOVE.W  D0,D4
  198.       0000DA G   D844                      ADD.W   D4,D4
  199.       0000DC G   3231  4000               MOVE.W  (A1,D4.W),D1          D1=TNEW(I)
  200.       0000E0 G   0241  00FF               ANDI.W  #$00FF,D1
  201.                                  ** For button untouched do
  202.       0000E4 G   3630  4040               MOVE.W  64(A0,D4.W),D3        D3=TOFF(I)
  203.       0000E8 G   B243                     CMP.W   D3,D1                 IF TNEW(I) > TOFF(I)
  204.       0000EA G   6F10                     BLE     TNOTUNT               THEN
  205.       0000EC G   B042                     CMP.W   D2,D0                   IF I=TCUR
  206.       0000EE G   6602                     BNE     TNOTOLD                 THEN
  207.       0000F0 G   5246                     ADDQ.W  #1,D6                     TOFTIM=TOFTIM+1
  208.       0000F2 G   3601            TNOTOLD  MOVE.W  D1,D3                   ENDIF
  209.       0000F4 G   9670  4000               SUB.W   (A0,D4.W),D3            D3=TNEW(I)-TMEAN(I)
  210.       0000F8 G   D770  4060               ADD.W   D3,96(A0,D4.W)          TDIF(I)=TDIF(I)+D3
  211.                                  ** For button touched do                ENDIF
  212.       0000FC G   3630  4020      TNOTUNT  MOVE.W  32(A0,D4.W),D3        D3=TON(I)
  213.       000100 G   B243                     CMP.W   D3,D1                 IF TNEW(I) < TON(I)
  214.       000102 G   6C14                     BGE     TOUCHLEND             THEN
  215.       000104 G   0C42  FFFF               CMPI.W  #-1,D2                  IF TCUR=-1 (untouched before)
  216.       000108 G   6608                     BNE     TNOTNUL                 THEN
  217.       00010A G   3400                     MOVE.W  D0,D2                     TCUR=I
  218.       00010C G   4245                     CLR.W   D5                        TTIMES=1
  219.       00010E G   4246                     CLR.W   D6                        TOFTIM=0
  220.       000110 G   6006                     BRA     TOUCHLEND
  221.       000112 G   B042            TNOTNUL  CMP.W   D2,D0                   ELSEIF TCUR = I
  222.       000114 G   6602                     BNE     TTWO
  223.       000116 G   5245                     ADDQ.W  #1,D5                     TTIMES=TTIMES+1
  224.                  0000  0118+      TTWO     EQU     *                       ELSE
  225.                                  *                                         two buttons touched
  226.                                  *                                       ENDIF
  227.                                  *                                     ENDIF
  228.       000118 G   51C8  FFBE      TOUCHLEND DBRA     D0,TOUCHLOOP       ENDFOR
  229.                                  * Send button to CAMAC
  230.       00011C G   0C46  000A               CMPI.W  #OFFDELAY,D6          IF TOFTIM > OFFDELAY
  231.       000120 G   6F1C                     BLE     TNUMOUT               THEN
  232.       000122 G   3C3C  000A               MOVE.W  #OFFDELAY,D6            TOFTIM=OFFDELAY
  233.       000126 G   343C  FFFF               MOVE.W  #-1,D2                  TCUR=-1 (no button touched)
  234.       00012A G   3E02                     MOVE.W  D2,D7                   TOLD=TCUR
```

```
ADDR SEC  OPC   EXT   EXT   .  S O U R C E  -  S T A T E M E N T

00012C G  4245                          CLR.W    D5                  TTIMES=0
00012E G  33C2  0000  0000*             MOVE.W   D2,TOUCHOUT         CAMAC OUTPUT = 0
000134 G  08B9  0008                    BCLR     #0.LAMSTATUS        CLR LAM-STATUS
000138 G        0000  0000*
00013C G  602C                          BRA      TNUMEND           ELSE
00013E G  0C45  0005          TNUMOUT   CMPI     #ONDELAY,D5         IF TTIMES > ONDELAY
000142 G  6F26                          BLE      TNUMEND             THEN
000144 G  4245                          CLR.W    D5                    TTIMES=0
000146 G  0C47  FFFF                    CMPI.W   #-1,D7                IF TOLD = -1 (untouched before)
00014A G  671E                          BEQ      TNUMEND               THEN
00014C G  3E02                          MOVE.W   D2,D7                   TOLD=TCUR
00014E G  760F                          MOVEQ    #15,D3
000150 G  9642                          SUB.W    D2,D3                   NUMBER=15-TOLD
000152 G  3039  0000  0000*             MOVE.W   TOUCHMASK,DO
000158 G  0700                          BTST     D3,DO                   IF BUTTON ENABLED
00015A G  670E                          BEQ      TNUMEND                 THEN
00015C G  33C3  0000  0000*             MOVE.W   D3,TOUCHOUT               CAMAC OUTPUT=NUMBER
000162 G  08F9  0008                    BSET     #0,LAMSTATUS              SET LAM-STATUS
000166 G        0000  0000*
                                *                                        ENDIF
                                *                                      ENDIF
                                *                                    ENDIF
                                *                                  ENDIF
00016A G  3142  FFF6          TNUMEND   MOVE.W   D2,-10(AO)
00016E G  3147  FFF8                    MOVE.W   D7,-8(AO)
000172 G  3145  FFFA                    MOVE.W   D5,-6(AO)
000176 G  3146  FFFC                    MOVE.W   D6,-4(AO)
00017A G  0468  0001  FFFE              SUBI.W   #1,-2(AO)         IF TCOUNT < 0
000180 G  6A30                          BPL      TOUCHEND          THEN
                                *                                    ADJUST TMEAN,TON,TOFF
                                *                                  ENDIF
                                *                                  TOUCH-CHECK END
                                ****************************************************************
                                *                                                              *
                                *                         ADJTOUCH                             *
                                *                                                              *
                                ****************************************************************
                                * ADJTOUCH - Updates TMEAN,TON and TOFF arrays
                                *
000182 G  317C  0400  FFFE    ADJTOUCH  MOVE.W   #$400,-2(AO)      1024 interrupts until next
                                                                               adjustment
000188 G  780A                          MOVEQ    #10,D4
00018A G  700F                          MOVEQ    #15,DO            FOR I=15 DOWNTO 0 DO
00018C G  3228  0060          ADJLOOP   MOVE.W   96(AO),DI           read TDIF(I)
000190 G  E861                          ASR.W    D4,D1               D1=TDIF(I)/1024
000192 G  D250                          ADD.W    (AO),DI             D1=TMEAN(I)+D1
000194 G  3081                          MOVE.W   D1,(AO)             TMEAN(I)=D1
000196 G  3401                          MOVE.W   D1,D2               D2=TMEAN(I)
000198 G  E649                          LSR.W    #3,D1               TON(I)=TMEAN(I)*87.5%
00019A G  9441                          SUB.W    D1,D2
00019C G  3142  0020                    MOVE.W   D2,32(AO)
```

```
Wylbur-No   ADDR SEC  OPC   EXT   EXT   .    S O U R C E  -  S T A T E M E N T
  283.      0001A0 G  E249                      LSR.W     #1,D1                 TOFF(I)=TMEAN(I)*93.25%
  284.      0001A2 G  D441                      ADD.W     D1,D2
  285.      0001A4 G  3142  0040                MOVE.W    D2,64(A0)
  286.      0001A8 G  4268  0060                CLR.W     96(A0)                Clear TDIF
  287.      0001AC G  5488                      ADDQ.L    #2,A0
  288.      0001AE G  51C8  FFDC                DBRA      D0,ADJLOOP          ENDFOR
  289.      0001B2 G  4E75                TOUCHEND  RTS
  290.                                    *
  291.                                    *
  292.                                    ************************************************************************
  293.                                    *                                                                      *
  294.                                    *         MECHCHECK - subroutine                                       *
  295.                                    *                                                                      *
  296.                                    *                                                                      *
  297.                                    ************************************************************************
  298.                                    * Reads all the mechnical buttons and if any is asserted the
  299.                                    * button number is put in memory location MECHOUT, and a LAM
  300.                                    * is set in the LAM-status-reg. If LAM is enabled for the me-
  301.                                    * chanical buttons, the LAM bit is also set in the LAM-request-
  302.                                    * reg and a LAM is asserted on the CAMAC-bus.
  303.                                    * The routine also incorborates debounsing.
  304.                                    *
  305.                                    *
  306.                                    * REGISTERS USED
  307.                                    *
  308.                                    * D0 - counter
  309.                                    * D1 - MINPUT
  310.                                    * D2 - MCUR
  311.                                    * D3 - not used
  312.                                    * D4 - not used
  313.                                    * D5 - MTIMES
  314.                                    * D6 - MOFTIM
  315.                                    * D7 - MOLD
  316.                                    *
  317.                                    * A0 - MECHVAR pointer
  318.                                    * A1 - not used
  319.
  320.      0001B4 G  207C  0000  0000* MECHCHECK MOVE.L    #MECHVAR,A0          Pointer to MECH-var block
  321.      0001BA G  1239  0000  0000*           MOVE.B    MECHADR1,D1          The mechanical buttons 8-15
  322.      0001C0 G  E149                      LSL.W     #8,D1
  323.      0001C2 G  1239  0000  0000*           MOVE.B    MECHADR2,D1          The mechanical buttons 0-7
  324.      0001C8 G  3410                      MOVE.W    (A0),D2              D2=MCUR
  325.      0001CA G  3E28  0002                MOVE.W    2(A0),D7             D7=MOLD
  326.      0001CE G  3A28  0004                MOVE.W    4(A0),D5             D5=MTIMES
  327.      0001D2 G  3C28  0006                MOVE.W    6(A0),D6             D6=MOFTIM
  328.      0001D6 G  700F                      MOVEQ     #15,D0
  329.                                    * For every button do                FOR I=15 DOWNTO 0 DO
  330.      0001D8 G  0101                MECHLOOP  BTST      D0,D1                IF MINPUT(I) = 0 (off)
  331.      0001DA G  6608                      BNE       MECHON               THEN
  332.      0001DC G  B042                      CMP.W     D2,D0                  IF I=MCUR
```

```
Wylbur-No   ADDR SEC  OPC   EXT   EXT   .   S O U R C E  -  S T A T E M E N T

 333.       0001DE G  6618                      BNE       MECHLEND
 334.       0001E0 G  5246                      ADDQ.W    #1,D6
 335.       0001E2 G  6014                      BRA       MECHLEND
                                          ** For button on do
 336.
 337.       0001E4 G  0C42  FFFF          MECHON CMPI.W    #-1,D2
 338.       0001E8 G  6608                      BNE       MNOTNUL
 339.       0001EA G  3400                      MOVE.W    D0,D2
 340.       0001EC G  4245                      CLR.W     D5
 341.       0001EE G  4246                      CLR.W     D6
 342.       0001F0 G  6006                      BRA       MECHLEND
 343.       0001F2 G  B042                MNOTNUL CMP.W    D2,D0
 344.       0001F4 G  6602                      BNE       TWOMECH
 345.       0001F6 G  5245                      ADDQ.W    #1,D5
 346.                 0000  01F8+         TWOMECH EQU       *
 347.                                     *
 348.                                     *
 349.                                     *
 350.       0001F8 G  51C8  FFDE          MECHLEND DBRA     D0,MECHLOOP
 351.                                     * Send button to CAMAC
 352.       0001FC G  0C46  000A                CMPI.W    #OFFDELAY,D6
 353.       000200 G  6F1C                      BLE       MNUMOUT
 354.       000202 G  3C3C  000A                MOVE.W    #OFFDELAY,D6
 355.       000206 G  343C  FFFF                MOVE.W    #-1,D2
 356.       00020A G  3E02                      MOVE.W    D2,D7
 357.       00020C G  4245                      CLR.W     D5
 358.       00020E G  33C2  0000  0000*         MOVE.W    D2,MECHOUT
 359.       000214 G  08B9  0009                BCLR      #1,LAMSTATUS
            000218 G        0000  0000*
 360.       00021C G  602C                      BRA       MNUMEND
 361.       00021E G  0C45  0005          MNUMOUT CMPI      #ONDELAY,D5
 362.       000222 G  6F26                      BLE       MNUMEND
 363.       000224 G  4245                      CLR.W     D5
 364.       000226 G  0C47  FFFF                CMPI.W    #-1,D7
 365.       00022A G  671E                      BEQ       MNUMEND
 366.       00022C G  3E02                      MOVE.W    D2,D7
 367.       00022E G  760F                      MOVEQ     #15,D3
 368.       000230 G  9642                      SUB.W     D2,D3
 369.       000232 G  3039  0000  0000*         MOVE.W    MECHMASK,D0
 370.       000238 G  0700                      BTST      D3,D0
 371.       00023A G  670E                      BEQ       MNUMEND
 372.       00023C G  33C3  0000  0000*         MOVE.W    D3,MECHOUT
 373.       000242 G  08F9  0009                BSET      #1,LAMSTATUS
            000246 G        0000  0000*
                                     *
 374.                                *
 375.                                *
 376.                                *
 377.                                *
 378.       00024A G  3082              MNUMEND MOVE.W    D2,(A0)
 379.       00024C G  3147  0002                MOVE.W    D7,2(A0)
 380.       000250 G  3145  0004                MOVE.W    D5,4(A0)
```

```
    THEN
      MOFTIM=MOFTIM+1
    ENDIF
  ELSE
    IF MCUR=-1 (not on)
    THEN
      MCUR=I
      MTIMES=0
      MOFTIM=0

    ELSEIF MCUR = I

      MTIMES=MTIMES+1
    ELSE
      two buttons touched
    ENDIF
  ENDIF
ENDFOR

IF MOFTIM > OFFDELAY
THEN
  MOFTIM=OFFDELAY
  MCUR=-1 (no button touched)
  MOLD=MCUR
  MTIMES=0
  CAMAC OUTPUT = 0
  CLR LAM-STATUS

ELSE
  IF MTIMES > ONDELAY
  THEN
    MTIMES=0
    IF MOLD = -1 (none on before)
    THEN
      MOLD=MCUR

      NUMBER=15-MOLD

      IF BUTTON ENABLED
      THEN
        CAMAC OUTPUT=NUMBER
        SET LAM-STATUS

      ENDIF
    ENDIF
  ENDIF
ENDIF
```

```
Wylbur-No   ADDR SEC  OPC   EXT   EXT   .    S O U R C E  -  S T A T E M E N T

  381.      000254 G  3146  0006              MOVE.W    D6,6(A0)
  382.      000258 G  4E75                MECHEND   RTS                          MECH-CHECK END
  383.                                    *
  384.                                    ********************************************************************
  385.                                    *                                                                  *
  386.                                    *        ENCODCHECK                                                *
  387.                                    *                                                                  *
  388.                                    ********************************************************************
  389.                                    *
  390.                                    *   Checks if any of the encoders have been turned,
  391.                                    * if so it is checked against bounds and data is prepared for
  392.                                    * CAMAC. If the new value is different the from the old a LAM
  393.                                    * is set. This routine is based on word counters, it don't work with
  394.                                    * scale factors and long word counters.
  395.                                    ENCODCHECK
  396.      00025A G  2079  0000  0000*           MOVE.L    MMCSTATUS,A0          Pointer to ENCODER data.
  397.      000260 G  7007                        MOVEQ     #7,D0                 FOR I=7 DOWNTO 0 DO
  398.      000262 G  3800                ENCLOOP MOVE.W    D0,D4
  399.      000264 G  D844                        ADD.W     D4,D4
  400.      000266 G  3230  4040                  MOVE.W    64(A0,D4.W),D1        D1=ENCCOUNT(I)
  401.      00026A G  B270  4010                  CMP       16(A0,D4.W),D1        IF ENCCOUNT(I)<>ENCOUT(I)
  402.      00026E G  6732                        BEQ       NOCHANGE              THEN
  403.      000270 G  B270  4020                  CMP       32(A0,D4.W),D1          IF ENCCOUNT(I)<ENCLOW(I)
  404.      000274 G  6F08                        BLE       NOTLOW                THEN
  405.      000276 G  31B0  4020  4010            MOVE.W    32(A0,D4.W),16(A0,D4.W)   ENCOUT(I)=ENCLOW(I)
  406.      00027C G  6012                        BRA       ENCEND
  407.      00027E G  B270  4030          NOTLOW  CMP.W     48(A0,D4.W),D1          ELSEIF ENCCOUNT(I)>ENCHIGH(I)
  408.      000282 G  6C08                        BGE       NOTHIGH
  409.      000284 G  31B0  4030  4010            MOVE.W    48(A0,D4.W),16(A0,D4.W)   ENCOUT(I)=ENCHIGH(I)
  410.      00028A G  6004                        BRA       ENCEND                ELSE
  411.      00028C G  3181  4010          NOTHIGH MOVE.W    D1,16(A0,D4.W)          ENCOUT(I)=ENCCOUNT(I)
  412.                                    *                                       ENDIF
  413.      000290 G  B270  4010          ENCEND  CMP       16(A0,D4.W),D1        IF ENCCOUNT(I)<>ENCOUT(I)
  414.      000294 G  670C                        BEQ       NOCHANGE              THEN
  415.      000296 G  01F9  0000  0000*           BSET      D0,LAMSTATUS+1          SET BIT IN LAM-STATUS
  416.      00029C G  31B0  4010  4040            MOVE.W    16(A0,D4.W),64(A0,D4.W)   ENCCOUNT(I)=ENCOUT(I)
  417.                                    *                                       ENDIF
  418.                                    *                                     ENDIF
  419.      0002A2 G  51C8  FFBE          NOCHANGE DBRA     D0,ENCLOOP          ENDFOR
  420.      0002A6 G  33FC  0019                   MOVE.W    #ENCTIME,ENCODINTERVAL  The time until next update
                0002AA G        0000  0000*
  421.      0002AE G  4E75                         RTS
  422.                                    ********************************************************************
  423.                                    *                                                                  *
  424.                                    *        INTERRUPT ROUTINES FOR THE SOFTWARE COUNTERS              *
  425.                                    *                                                                  *
  426.                                    ********************************************************************
  427.                                    *
  428.                                    *
  429.
```

```
Mylbur-No   ADDR SEC  OPC   EXT   EXT    .   S O U R C E  -  S T A T E M E N T
  430.      0002B0 G  0839  0000        ENCOD0   BTST    #0,ENCODADR          IF BIT(0)=0
            0002B4 G        0000  0000*
  431.      0002B8 G  6708                       BEQ     ENCODOPL             THEN
  432.      0002BA G  53B9  0000  0000*          SUBQ.L  #1,ENCODCOUNT          COUNT DOWN
  433.      0002C0 G  4E73                       RTE                          ELSE
  434.      0002C2 G  52B9  0000  0000* ENCODOPL ADDQ.L  #1,ENCODCOUNT          COUNT UP
  435.      0002C8 G  4E73                       RTE                          ENDIF
  436.
  437.      0002CA G  0839  0001        ENCOD1   BTST    #1,ENCODADR
            0002CE G        0000  0000*
  438.      0002D2 G  6708                       BEQ     ENCOD1PL
  439.      0002D4 G  53B9  0000  0002*          SUBQ.L  #1,ENCODCOUNT+4
  440.      0002DA G  4E73                       RTE
  441.      0002DC G  52B9  0000  0002* ENCOD1PL ADDQ.L  #1,ENCODCOUNT+4
  442.      0002E2 G  4E73                       RTE
  443.
  444.      0002E4 G  0839  0002        ENCOD2   BTST    #2,ENCODADR
            0002E8 G        0000  0000*
  445.      0002EC G  6708                       BEQ     ENCOD2PL
  446.      0002EE G  53B9  0000  0004*          SUBQ.L  #1,ENCODCOUNT+8
  447.      0002F4 G  4E73                       RTE
  448.      0002F6 G  52B9  0000  0004* ENCOD2PL ADDQ.L  #1,ENCODCOUNT+8
  449.      0002FC G  4E73                       RTE
  450.
  451.      0002FE G  0839  0003        ENCOD3   BTST    #3,ENCODADR
            000302 G        0000  0000*
  452.      000306 G  6708                       BEQ     ENCOD3PL
  453.      000308 G  53B9  0000  0006*          SUBQ.L  #1,ENCODCOUNT+12
  454.      00030E G  4E73                       RTE
  455.      000310 G  52B9  0000  0006* ENCOD3PL ADDQ.L  #1,ENCODCOUNT+12
  456.      000316 G  4E73                       RTE
  457.
  458.      000318 G  0839  0004        ENCOD4   BTST    #4,ENCODADR
            00031C G        0000  0000*
  459.      000320 G  6708                       BEQ     ENCOD4PL
  460.      000322 G  53B9  0000  0008*          SUBQ.L  #1,ENCODCOUNT+16
  461.      000328 G  4E73                       RTE
  462.      00032A G  52B9  0000  0008* ENCOD4PL ADDQ.L  #1,ENCODCOUNT+16
  463.      000330 G  4E73                       RTE
  464.
  465.      000332 G  0839  0005        ENCOD5   BTST    #5,ENCODADR
            000336 G        0000  0000*
  466.      00033A G  6708                       BEQ     ENCOD5PL
  467.      00033C G  53B9  0000  000A*          SUBQ.L  #1,ENCODCOUNT+20
  468.      000342 G  4E73                       RTE
  469.      000344 G  52B9  0000  000A* ENCOD5PL ADDQ.L  #1,ENCODCOUNT+20
  470.      00034A G  4E73                       RTE
  471.
  472.      00034C G  0839  0006        ENCOD6   BTST    #6,ENCODADR
            000350 G        0000  0000*
```

```
Wylbur-No    ADDR SEC  OPC   EXT   EXT    .    S O U R C E  -  S T A T E M E N T

  473.       000354 G  6708                     BEQ       ENCOD6PL
  474.       000356 G  53B9  0000  000C*         SUBQ.L    #1,ENCODCOUNT+24
  475.       00035C G  4E73                     RTE
  476.       00035E G  52B9  0000  000C* ENCOD6PL ADDQ.L    #1,ENCODCOUNT+24
  477.       000364 G  4E73                     RTE
  478.
  479.       000366 G  0839  0007        ENCOD7 BTST      #7,ENCODADR
             00036A G        0000  0000*
  480.       00036E G  6708                     BEQ       ENCOD7PL
  481.       000370 G  53B9  0000  000E*         SUBQ.L    #1,ENCODCOUNT+28
  482.       000376 G  4E73                     RTE
  483.       000378 G  52B9  0000  000E* ENCOD7PL ADDQ.L    #1,ENCODCOUNT+28
  484.       00037E G  4E73                     RTE
  485.
  486.       000380 G                           END
```

Appendix 15 NODDEFS INSERT FILE

```
1.      ************************************************************************
2.      * CAMDEFS - NODAL AND CAMAC, MACROS AND DEFINITIONS                    *
3.              NOLIST
4.      * INSERT FILE:
5.      * Some universal constant and macro definitions for
6.      * use in NODAL subroutines and functions using CAMAC
7.      * AUTHOR:         P.S. ANDERSSEN
8.      * WRITTEN:        SEPTEMBER 1980
9.
10.
11.     * OFFSETS FOR STACKED ADDRESS REGISTERS
12.     SA3       EQU -4
13.     SA2       EQU -8
14.     SA1       EQU -12
15.     SA0       EQU -16
16.     SD5       EQU -20
17.     SD4       EQU -24
18.     SD3       EQU -28
19.     SD2       EQU -32
20.     SD1       EQU -36
21.     SD0       EQU -40
22.
23.     * MAXIMUM LENGTH OF A CHARACTER STRING
24.     MAXSTR    EQU 80
25.
26.     * SIZE OF A FLOATING POINT NUMBER IN BYTES
27.     REALSIZE EQU 6
28.
29.     * OKRTS - Macro for normal return from a quick subroutine
30.     OKRTS     MACRO
31.               ADDQ.L #4,(A7)
32.               RTS
33.               ENDM
34.
35.     * ERRTS  - Macro for failure return from a quick subroutine
36.     ERRTS     MACRO
37.               RTS
38.               ENDM
39.
40.
41.     * CALL - MACRO TO CALL A SUBROUTINE
42.     * First argument = Subroutine entry point
43.     * Second argument = Entry point for failure return
44.     CALL      MACRO
45.               JSR \1
46.               JMP \2(PC)
47.               ENDM
48.
49.
50.     * CALLTRAP - MACRO TO CALL A STANDARD NODAL TRAP
51.     * TRAP must do normal (skip) and failure (direct) returns
52.     * First argument = TRAP number
53.     * Second argument = entry point for failure return
54.     CALLTRAP MACRO
55.               TRAP #\1
56.               JMP \2(PC)
57.               ENDM
58.
59.
60.     * ENTER - Macro to enter a routine
61.     ENTER     MACRO
62.               LINK A6,#ZYX
63.               CMPA.L (A5),A7
64.               BHI.S *+8
65.               JMP.L NODSTKF
66.               MOVEM.L D0-D5/A0-A3,SD0(A6)
67.               ENDM
```

```
68.
69.      * RET - Macro to return from a routine
70.      RET       MACRO
71.                MOVEM.L SD0(A6),D0-D5/A0-A3
72.                UNLK A6
73.                ADDQ.L #4,(A7)
74.                RTS
75.                ENDM
76.
77.      * ERRET - macro to return with failure from a routine
78.      ERRET     MACRO
79.                MOVEM.L SD1(A6),D1-D5/A0-A3
80.                UNLK A6
81.                RTS
82.                ENDM
83.
84.      * PROGM - Macro to reset local variable index
85.      PROGM     MACRO
86.      ZYX       SET SD0
87.      ZXY       SET 8
88.                ENDM
89.
90.      * DATA - MACRO TO DECLARE LOCAL DATA FOR A STACKING SUBROUTINE
91.      * First argument = name of variable
92.      * Second argument = number of words to allocate
93.      DATA      MACRO
94.      ZYX       SET ZYX-\2*2
95.      \1        SET ZYX
96.                ENDM
97.
98.      * FPAR - Macro to declare a parameter of a NODAL function.
99.      * The parameter addresses are pusned on the stack in the order
100.     * of the call.
101.     * There are four bytes per parameter address.
102.     * The last parameter address is at the last one pushed on the stack,
103.     * at address 8(A6).
104.     FPAR      MACRO
105.     \1        SET ZXY
106.     ZXY       SET ZXY+4
107.               ENDM
108.
109.
110.     * STRING - MACRO TO DECLARE A STRING IN LOCAL AREA
111.     * First argument = name of string
112.     STRING    MACRO
113.     ZYX       SET ZYX-MAXSTR-10
114.     \1        SET ZYX
115.               ENDM
116.
117.     * REAL - MACRO TO DECLARE A FLOATING POINT VARIABLE IN LOCAL AREA
118.     * First argument = name of real variable
119.     REAL      MACRO
120.     ZYX       SET ZYX-REALSIZE
121.     \1        SET ZYX
122.               ENDM
123.
124.     * MOVEF - MACRO TO MOVE A FLOATING POINT NUMBER
125.     * Two arguments must be register indirect
126.     MOVEF     MACRO
127.               MOVE.W \1,\2
128.               MOVE.L 2\1,2\2
129.               ENDM
```

```
130.
131.      * GLOBAL - Macro to define a NODAL global.
132.      * The first parameter is the symbol of the global (relative A5).
133.      * The second parameter is the number bytes required for the global.
134.
135.      GLOBAL    MACRO
136.      \1        SET NGZYX
137.      NGZYX     SET NGZYX+\2
138.                ENDM
139.
140.      NGZYX     SET 0
141.
142.                GLOBAL SLIMIT,4
143.                GLOBAL ESYMB,4
144.                GLOBAL VLOC,4
145.                GLOBAL ETEXT,4
146.                GLOBAL STEXT,4
147.                GLOBAL XLIN,2
148.                GLOBAL XAD,4
149.                GLOBAL LOC,2
150.                GLOBAL DFLAG,2
151.                GLOBAL PCERR,4
152.                GLOBAL STERR,4
153.                GLOBAL FPERR,4
154.                GLOBAL ERRPS,2
155.                GLOBAL ERRNO,2
156.                GLOBAL ERRLIN,2
157.                GLOBAL EDEF,4
158.                GLOBAL SDEF,4
159.                GLOBAL VALRET,REALSIZE
160.                GLOBAL SVAL,4
161.                GLOBAL TRACE,2
162.                GLOBAL CRFLG,2
163.                GLOBAL EDLIN,2
164.                GLOBAL EDFLAG,2
165.                GLOBAL DERNAM,6
166.                GLOBAL DERLIN,2
167.                GLOBAL OVRLIN,2
168.                GLOBAL STAG,42
169.                GLOBAL ARG,16*REALSIZE
170.                GLOBAL SYSLIST,4
171.                GLOBAL USERAD,4
172.                GLOBAL IDEV,2
173.                GLOBAL ODEV,2
174.                GLOBAL DDEV,2
175.                GLOBAL LDEV,2
176.                GLOBAL INIDREGW,16
177.                GLOBAL CAMST,2
178.
179.      *********CAMAC DEFINITIONS*****************************************
180.
181.      * PROT/UNPROT - Protection macros to ensure undivisible CAMAC
182.      * oprerations on the CAMAC interface of the ICC.
183.
184.      PROT      MACRO
185.                TST       DISMASK
186.                CLR       LAMED
187.                ENDM
188.      UNPROT    MACRO
189.                TST       ENMASK
190.                MOVE.W    #1,LAMED
191.                ENDM
```

```
191.01    *
191.02    *       Macro's which is used in the communication with the STAC
191.03    * The X-respons is checked and the macro waits until the Q-respons
191.04    * is recieved.
191.05    *
191.06    *       CAMACRD  <EA>,DX  Where EA means an effective CAMAC-adr.
191.07    * Reads from CAMAC to data-reg x, checks X- and Q-respons
191.08    *
191.09    CAMACRD  MACRO
191.1              MOVE.L  \1,\2                   DUMMY MOVE - CLEAR OLD Q
191.11    \@       MOVE.L  \1,\2
191.12             BTST    #30,\2                  TEST X-RESPONS
191.13             BEQ     E34                     IF NO X-RESPONS: HARDWARE ERROR
191.14             BTST    #31,\2                  TEST Q-RESPONS
191.15             BEQ     \@                      IF NO Q-RESPONS: TRY AGAIN
191.16             ENDM
191.17
191.18    *
191.19    *       CAMACWR  RX,<EA> Where EA means an effective CAMAC-adr.
191.2     * Writes from reg x to CAMAC, checks X- and Q-repons.
191.21    *
191.22    CAMACWR  MACRO
191.23    \@       MOVE.L  \1,\2
191.24             MOVE.W  SWITCH,D7               GET STATUS (X-,Q-RESPONS)
191.25             BTST    #14,D7                  TEST X-RESPONS
191.26             BEQ     E34                     IF NO X-RESPONS:
191.27             BTST    #15,D7                  TEST Q-RESPONS
191.28             BEQ     \@                      IF NO QRESPONS: TRY AGAIN
191.29             ENDM
191.3     *
191.31    *       CAMACOM  <EA>,DX  Where EA means an effective CAMAC-adr.
191.32    * Reads from CAMAC - gives a command to CAMAC, checks X- and Q-
191.33    * respons.
191.34    *
191.35    CAMACOM  MACRO
191.36    \@       MOVE.L  \1,\2
191.37             BTST    #30,\2                  TEST X-RESPONS
191.38             BEQ     E34                     IF NO X-RESPONS: HARDWARE ERROR
191.39             BTST    #31,\2                  TEST Q-RESPONS
191.4              BEQ     \@                      IF NO Q-RESPONS: TRY AGAIN
191.41             ENDM
191.42
```

```
192.
193.    CAMAD    EQU $800000+2                  CAMAC WORD BASE ADDRESS
194.    CAMAL    EQU $800000                    CAMAC LONG BASE ADDRESS
195.
196.    CAMA0    EQU 0<<7                       ADDRESS SHIFT LEFT 7
197.    CAMA1    EQU 1<<7
198.    CAMA2    EQU 2<<7
199.    CAMA3    EQU 3<<7
200.    CAMA4    EQU 4<<7
201.    CAMA5    EQU 5<<7
202.    CAMA6    EQU 6<<7
203.    CAMA7    EQU 7<<7
204.    CAMA8    EQU 8<<7
205.    CAMA9    EQU 9<<7
206.    CAMA10   EQU 10<<7
207.    CAMA11   EQU 11<<7
208.    CAMA12   EQU 12<<7
209.    CAMA13   EQU 13<<7
210.    CAMA14   EQU 14<<7
211.    CAMA15   EQU 15<<7
212.
213.    CAMF0    EQU 0<<2                       FUNCTION SHIFT LEFT 2
214.    CAMF1    EQU 1<<2
215.    CAMF2    EQU 2<<2
216.    CAMF3    EQU 3<<2
217.    CAMF4    EQU 4<<2
218.    CAMF5    EQU 5<<2
219.    CAMF6    EQU 6<<2
220.    CAMF7    EQU 7<<2
221.    CAMF8    EQU 8<<2
222.    CAMF9    EQU 9<<2
223.    CAMF10   EQU 10<<2
224.    CAMF11   EQU 11<<2
225.    CAMF12   EQU 12<<2
226.    CAMF13   EQU 13<<2
227.    CAMF14   EQU 14<<2
228.    CAMF15   EQU 15<<2
229.    CAMF16   EQU 16<<2
230.    CAMF17   EQU 17<<2
231.    CAMF18   EQU 18<<2
232.    CAMF19   EQU 19<<2
233.    CAMF20   EQU 20<<2
234.    CAMF21   EQU 21<<2
235.    CAMF22   EQU 22<<2
236.    CAMF23   EQU 23<<2
237.    CAMF24   EQU 24<<2
238.    CAMF25   EQU 25<<2
239.    CAMF26   EQU 26<<2
240.    CAMF27   EQU 27<<2
241.    CAMF28   EQU 28<<2
242.    CAMF29   EQU 29<<2
243.    CAMF30   EQU 30<<2
244.    CAMF31   EQU 31<<2
245.
246.    CAMIN    EQU CAMAD+CAMA8+CAMF26         N=0, A=8, F=26 (Z PULSE)
247.    DISMASK  EQU CAMAD+CAMF24               DISABLE LAM MASK-READ OR WRITE
248.    ENMASK   EQU CAMAD+CAMF26               ENABLE LAM MASK-READ OR WRITE
249.    WHMASK   EQU CAMAD+CAMA1+CAMF16         WRITE HIGH WORD TO MASK
250.    WLMASK   EQU CAMAD+CAMF16               WRITE LOW WORD TO MASK
251.    *                                       '0' IN MASK IS EQU. TO MASKOUT
252.    SWITCH   EQU $C00040                    Q AND X IN UPPER TWO BITS
253.             LIST
254.    ********************************************************************
```

Appendix 16 IMANFUNS PROGRAM

```
Wylbur-No    ADDR SEC  OPC   EXT   EXT   .    S O U R C E  -  S T A T E M E N T

                                            ********************************************************************
                                            *                                                                  *
                                            *                MANFUNS - ICC                                     *
                                            *                                                                  *
                                            *   AUTHOR:     PETER PETERSEN          WRITTEN:   27.07.83      *
                                            *                                                                  *
                                            *   MODIFYED  BY :                                                 *
                                            *                NAME:                  DATE:                      *
                                            *                PETER                  00.00.83                   *
                                            *                                                                  *
                                            ********************************************************************

                                                     IDENT     MANFUNS
                                            FUNCTION SECTION   R
                                                     INSERT
                                            ********************************************************************
                                            * CAMDEFS - NODAL AND CAMAC, MACROS AND DEFINITIONS                *
                                                     NOLIST
                                                     LIST
                                            ********************************************************************

                                                     LIST      FORM,CREF

                                            *
                                            * MANFUNS - MAN/MACHINE FUNCTIONS FOR 68000 NODAL.
                                            *
                                            *      Those NODAL functions uses the STAC as the input device for
                                            * all the MAN-MACHINE interface.  The STAC collects all the input
                                            * data and transfers them in condensed form to the ICC (crate
                                            * controller)
                                            *      The input devices are the following
                                            *        The Touch-screen
                                            *        Hard-Buttons
                                            *        The Knob
                                            *        The Tracker-Ball
                                            *
                                            *      To access one of the input devices, the crate controller
                                            * send a pointer to that inputdevice, with the function F16 A0
                                            * The controller can then read the value from that devices with
                                            * the function F0-A1 or F2-A1, which will clear the value.  It
                                            * is allso possible to write data to the same registers, this is
                                            * done with the function F16-A1.
                                            *
                                            *      The pointers for the different input devices is the following
                                            *
                                            *                Pointer (hex)   Description of register
                                            *
                                            TOUCHP    EQU     0               Touchbutton output
                                            TMASKP    EQU     2               Mask for touchbuttons
                                            MECHP     EQU     4               Mechanical button output
```

000000 G
000000 R

I
I
I
I
I

0000 0000
0000 0002
0000 0004

Version 3.5 from: 16th of November 1983 16:25:10

```
Wylbur-No   ADDR SEC  OPC   EXT   EXT  .   S O U R C E  -  S T A T E M E N T

   47.                                  *         EQU      6          Mask for mechanical buttons
   48.                                  *         EQU      8          Mask for the encoders
   49.                                  *
   50.                                  *         EQU      10         Output for encoder 0
   51.                                  *         EQU      12         Output for encoder 1
   52.                                  *         EQU      14         Output for encoder 2
   53.                                  *         EQU      16         Output for encoder 3
   54.                                  *         EQU      18         Output for encoder 4
   55.                                  *         EQU      1A         Output for encoder 5
   56.                                  *         EQU      1C         Output for encoder 6
   57.                                  *         EQU      1E         Output for encoder 7
   58.                                  *
   59.                                  *         EQU      20         Low limit for encoder 0
   60.                                  *         EQU      22         Low limit for encoder 1
   61.                                  *         EQU      24         Low limit for encoder 2
   62.                                  *         EQU      26         Low limit for encoder 3
   63.                                  *         EQU      28         Low limit for encoder 4
   64.                                  *         EQU      2A         Low limit for encoder 5
   65.                                  *         EQU      2C         Low limit for encoder 6
   66.                                  *         EQU      2E         Low limit for encoder 7
   67.                                  *
   68.                                  *         EQU      30         High limit for encoder 0
   69.                                  *         EQU      32         High limit for encoder 1
   70.                                  *         EQU      34         High limit for encoder 2
   71.                                  *         EQU      36         High limit for encoder 3
   72.                                  *         EQU      38         High limit for encoder 4
   73.                                  *         EQU      3A         High limit for encoder 5
   74.                                  *         EQU      3C         High limit for encoder 6
   75.                                  *         EQU      3E         High limit for encoder 7
   76.                                  *
   77.                                  *         EQU      40         The software counter for encoder 0
   78.                                  *         EQU      42         The software counter for encoder 1
   79.                                  *         EQU      44         The software counter for encoder 2
   80.                                  *         EQU      46         The software counter for encoder 3
   81.                                  *         EQU      50         The software counter for encoder 4
   82.                                  *         EQU      54         The software counter for encoder 5
   83.                                  *         EQU      58         The software counter for encoder 6
   84.                                  *         EQU      5C         The software counter for encoder 7
   85.
   86.
   87.                                  ************************************************************************
   88.                                  *                                                                      *
   89.                                  *                    EXTERN  BLOCK                                     *
   90.                                  *                                                                      *
   91.                                  ************************************************************************
   92.                                           EXTERN    NODSTKF,TBUTMASK
   93.                                           EXTERN    LAMED,LAMREG
   94.                                           EXTERN    CAMAD,CAMAL,SWITCH
   95.                                           EXTERN    CHECK,NLZ
   96.                                           EXTERN    DISMASK,ENMASK,WLMASK,WHMASK
```

```
Mylbur-No   ADDR SEC  OPC   EXT   EXT   .   S O U R C E  -  S T A T E M E N T

   97.                                        EXTERN   CAMBUTS,CAMTBUTS
   98.                                        EXTERN   OUTBYTE,STROU
   99.                                        EXTERN   NSSET,GCI,WCI
  100.
  101.              0000  000B        NSHIFT   EQU      11
  102.              0000  0008        TOUCHBITNR  EQU      8
  103.              0000  0009        MECHBITNR EQU      9
  104.              0000  0300        BUTBITS  EQU      1<<TOUCHBITNR+1<<MECHBITNR
  105.              0000  4EF9        XJMP     EQU      $4EF9
  106.
  107.              0000  0001        BACKBUTNR EQU      1
  108.              0000  0002        TRUNCBUTNR EQU     2
  109.
  110.              0000  0010        MMCSLOT  EQU      16
  111.              0000  0040*       MMCLAMREG EQU      LAMREG+4*MMCSLOT
  112.              0080  8000        MMCCAMADR EQU      CAMAL+MMCSLOT<<NSHIFT
  113.              0000  0000        LAMHMASK EQU      0                    1<<(MMCSLOT-17) IF MMCSLOT>16
  114.              0000  8000        LAMLMASK EQU      1<<(MMCSLOT-1)       ZERO IF MMCSLOT>16
  115.              0040  5316        MANINTAD EQU      $405000+(63+MMCSLOT)*10   THE ADDRESS TO WHERE THE PROGRAM
  116.                                *                                      WILL JMP ON INTERRUPT FROM MAN-IO
  117.
  118.              0000  0000        F0A0     EQU      CAMF0+CAMA0
  119.              0000  0040        F16A0    EQU      CAMF16+CAMA0
  120.              0000  0080        F0A1     EQU      CAMF0+CAMA1
  121.              0000  0088        F2A1     EQU      CAMF2+CAMA1
  122.              0000  00C0        F16A1    EQU      CAMF16+CAMA1
  123.              0000  00E4        F25A1    EQU      CAMF25+CAMA1
  124.
  125.              0000  065C        F23A12   EQU      CAMF23+CAMA12        SELECTIVE CLEAR LAM-STATUS REG.
  126.              0000  06CC        F19A13   EQU      CAMF19+CAMA13        SELECTIVE ENABLE A PROCES
  127.              0000  06DC        F23A13   EQU      CAMF23+CAMA13        SELECTIVE DISABLE A PROCES
  128.              0000  0704        F1A14    EQU      CAMF1+CAMA14         WRITE INTO LAM-REQUEST REGISTER
  129.              0000  0744        F17A14   EQU      CAMF17+CAMA14        WRITE INTO LAM-REQUEST REGISTER
  130.              0000  075C        F23A14   EQU      CAMF23+CAMA14        SELECTIVE CLEAR LAM-REQUEST REG.
  131.
  132.
  133.                                ************************************************************************
  134.                                *                                                                      *
  135.                                *                                                                      *
  136.                                *                    BUTTON                                            *
  137.                                *                                                                      *
  138.                                *                                                                      *
  139.                                ************************************************************************
  140.                                *
  141.                                *
  142.                                * BUTTON - Wait for a button to be pressed.
  143.                                * SET Z=BUTTON
  144.                                * The program waits until a button has been pressed.
  145.                                * The BUTTON function returns with the number of the
  146.                                * button which was pressed.
```

```
Wylbur-No   ADDR SEC  OPC   EXT   EXT   .   S O U R C E - S T A T E M E N T

147.                                       *
148.                                       * REGISTER USED
149.                                       *
150.                                       * D0 - Transfer of button nr. from BUTTON/BUTS to NLZ/6
151.                                       * D1 - The last content of the LAM-REQ reg. from the MAN-IO module
152.                                       * D2 - BUTBITS - A longword with MECHBITNR, TOUCHBITNR set
153.                                       * D3 -
154.                                       * D4 - The negation of D2
155.                                       * D5 -
156.                                       * D6 -
157.                                       *
158.                                       * A0 -
159.                                       * A1 - Pointer to the LAM information block.
160.                                       * A2 - Used as the floating destination for  NLZ/5
161.                                       * A3 - Pointer to CAMAC for access to the MAN-IO slot
162.                                       * A4 - Used by NODAL
163.                                       * A5 - Used by NODAL
164.                                       * A6 - Used by NODAL
165.                                       * A7 - Stack pointer
166.                                       *
167.
168.    000000 R   00E6  000A  4255  BUTTST    DC.W      BUTTSZ,10,'BU','TT','ON',0,0
169.    00000E R   0000  0012+               DC.L      BUTTON
170.
171.    000012 R                             PROGM
172.
173.    000012 R                   BUTTON    ENTER
174.    000026 R   33FC  4EF9                MOVE.W    #XJMP,MANINTAD        INSERT THE CODE FOR
            00002A R         0040  5316
175.    00002E R   23FC  0000  024C+          MOVE.L    #MANIOIRQ,MANINTAD+2  "JMP MANIOIRQ"
            000034 R         0040  5318
176.    000038 R   267C  0080  8000          MOVE.L    #MMCCAMADR,A3
177.    00003E R   227C  0000  0040*         MOVE.L    #MMCLAMREG,A1         POINTER TO LAM INFORMATION
178.    000044 R   243C  0000  0300          MOVE.L    #BUTBITS,D2           D2 - TOUCHBIT, MECH BIT SET
179.    00004A R   2802                      MOVE.L    D2,D4
180.    00004C R   4684                      NOT.L     D4                    D4 - TOUCHBIT, MECH BIT NEGATED
181.                             * Enable for interrupt from slot MAN-MACHINE COMMUNICATION
182.    00004E R                             PROT
183.    00005A R   2E39  0000  0000*         MOVE.L    LAMREG,D7             READ LAM-MASK INFORMATION
184.    000060 R   08C7  000F                BSET      #MMCSLOT-1,D7         SET THE BIT FOR SLOT MAN-IO
185.    000064 R   33C7  0080  0042          MOVE.W    D7,WLMASK
186.    00006A R   4847                      SWAP      D7
187.    00006C R   33C7  0080  00C2          MOVE.W    D7,WHMASK
188.    000072 R                             CAMACWR   D2,F19A13(A3)         ENABLE LAM TOUCH-,MECH-BUT
189.    00008A R                   RETRY     CAMACWR   D2,F23A14(A3)         CLEAR LAM-REQUEST REG - STAC
190.    0000A2 R                             CAMACWR   D2,F23A12(A3)         CLEAR LAM-STATUS REG - STAC
191.    0000BA R   C991                      AND.L     D4,(A1)               CLEAR THE LAM-REQ BIT
192.    0000BC R                             UNPROT
                                 * Enabled now                              WHILE NO BUTS PRESSED
193.                             WAITLAM   CALL      CHECK,ERR               CHECK FOR ESCAPE TYPED
194.    0000CA R
```

```
Wylbur-No   ADDR SEC  OPC   EXT   EXT   .    S O U R C E  -  S T A T E M E N T

  195.      0000D4 R  2211                    MOVE.L   (A1),D1              READ LAM-REQ REG FOR MAN-IO
  196.      0000D6 R  0801  0009              BTST     #MECHBITNR,D1         IF MECHANICAL BUTTON PRESSED
  197.      0000DA R  6700  000C              BEQ      NOMECH                THEN
  198.      0000DE R                          CALL     BCHCK,BACK17            CHECK BACK BUTTON
  199.                                  *                                    ENDIF
  200.      0000E8 R  0801  0008        NOMECH BTST    #TOUCHBITNR,D1        IF TOUCHBUTTON PRESSED
  201.      0000EC R  67DC                    BEQ      WAITLAM               THEN CONTINUE
  202.                                  *                                  ENDWHILE
  203.      0000EE R                          PROT
  204.      0000FA R                          CAMACWR  #TOUCHP,F16A0(A3)     READ BUTTON-TOUCHED
  205.      000116 R                          CAMACRD  F0A1(A3),D0
  206.      00012C R  5240                    ADDQ     #1,D0                 IF BUTTON=0
  207.      00012E R  6700  FF5A              BEQ      RETRY                 THEN TRY AGAIN
  208.                                  *                                    ELSE
  209.      000132 R  33C7  0080  0042                 MOVE.W   D7,WLMASK      RESTORE THE OLD LAMMASK
  210.      000138 R  4847                    SWAP     D7
  211.      00013A R  33C7  0080  00C2                 MOVE.W   D7,WHMASK
  212.      000140 R                    BUTON CAMACOM  F25A1(A3),D7            CLICK
  213.      000152 R                    BUTZERO CALL   NLZ16,ERR
  214.      00015C R                          CAMACWR  D2,F23A13(A3)          DISABLE INTERRUPT FROM BUTTT
  215.      000174 R                          CAMACWR  D2,F23A14(A3)          CLEAR LAM-REQUEST REG - STAC
  216.      00018C R                          CAMACWR  D2,F23A12(A3)          CLEAR LAM-STATUS REG - STAC
  217.      0001A4 R  C991                    AND.L    D4,(A1)               CLEAR THE LAM-REQ BIT
  218.      0001A6 R                          UNPROT
  219.      0001B4 R                          RET
  220.
  221.      0001C0 R  7022                    E34      MOVEQ    #34,D0       ERROR RETURN - HARDWARE ERROR
  222.      0001C2 R                          ERR      ERRET
 222.5                0000  00E6             BUTTSZ   EQU      (*-BUTTST)/2
  223.
  224.                                  ********************************************************************************
  225.                                  *                                                                              *
  226.                                  *                                                                              *
  227.                                  *                        BUTS                                                  *
  228.                                  *                                                                              *
  229.                                  *                                                                              *
  230.                                  ********************************************************************************
  231.
  232.
  233.                                  * BUTS - Read button.
  234.                                  * SET Z=BUTS
  235.                                  * Reads the buttons immediately.
  236.                                  * The function BUTS returns a zero if no button has
  237.                                  * been pressed. Otherwise the button number is returned.
  238.
  239.      0001CC R  00A3  000A  4255  BUTSST DC.W     BUTSSZ,10,'BU','TS',0,0,0
  240.      0001DA R  0000  01DE+             DC.L     BUTS
  241.
  242.      0001DE R                          PROGM
  243.
```

```
Wylbur-No    ADDR SEC  OPC   EXT   EXT   .    S O U R C E  -  S T A T E M E N T

  244.       0001DE R                      BUTS     ENTER
  245.       0001F2 R                               CALL     BCHCK,BACK17        CHECK FOR BACK-BUTTON PRESSED
  246.                                      *                                    IF SO THE OUTPUT IS 17
  247.       0001FC R  267C  0080  8000             MOVE.L   #MMCCAMADR,A3
  248.       000202 R                               PROT
  249.       00020E R                               CAMACWR  #TOUCHP,F16A0(A3)   POINTER TO TOUCH BUTTON OUT
  250.       000228 R                               CAMACRD  F2A1(A3),D0         D0=BUTTON TOUCHED
  251.       00023C R  5240                         ADDQ     #1,D0               IF BUTS<>0
  252.       00023E R  6700  FF12                   BEQ      BUTZERO             THEN
  253.       000242 R  6000  FEFC                   BRA      BUTON                 CLICK
  254.                                      *                                    ENDIF
  255.                                      *                                    NUMBER OUT
  256.                                      *
  257.                                      * When the BACK button is pressed the output is 17
  258.       000246 R  7011                 BACK17   MOVEQ    #17,D0
  259.       000248 R  6000  FEF6                   BRA      BUTON
  260.
  261.                                      ************************************************************************
  262.                                      *                                                                      *
  263.                                      *                                                                      *
  264.                                      *         INTERRUPT RUTINES FOR COMMUNICTION WITH THE STAC             *
  265.                                      *                                                                      *
  266.                                      *                                                                      *
  267.                                      ************************************************************************
  268.                                      *
  269.                                      *      When the ICC receives a LAM-request from the STAC with the
  270.                                      * MAN-MACHINE interface the, following must be done.
  271.                                      * 1) Read the LAM-request register in the STAC - F(1)A(14)
  272.                                      *    Here the proces which need attention has raised a bit
  273.                                      *    and the ICC can then detect which proces it was.
  274.                                      * 2) Service the proces who raised the LAM
  275.                                      * 3) Clear the bit for that proces in the LAM-request reg.
  276.                                      *    and in the LAM-status reg.
  277.                                      *
  278.       00024C R  48E7  8010           MANIOIRQ MOVEM.L  D0/A3,-(A7)
  279.       000250 R  267C  0080  8000             MOVEA.L  #MMCCAMADR,A3
  280.       000256 R                               CAMACRD  F1A14(A3),D0        GET LAM-REQ REG. FROM STAC
  281.       00026C R  23C0  0000  0040*            MOVE.L   D0,LAMREG+MMCSLOT*  SAVE THE LAM-REQ REG
  282.                                      *                                    THIS REGISTER IS USED FROM
  283.                                      *                                    NODAL-ROUTINES TO POLLE LAM
  284.                                      *                                    FROM THE MAN-IO MODULE
  285.       000272 R  4CDF  0801                   MOVEM.L  (A7)+,D0/A3
  286.       000276 R  4E73                         RTE                          RETURN FROM EXECPTION
  287.
  289.
  290.
  291.
  292.
  293.                                      * BCHCK - Check for BACK button being pressed.
  294.                                      * Return:
```

```
Wylbur-No    ADDR SEC  OPC   EXT   EXT   .    S O U R C E  -  S T A T E M E N T

295.                                        * Failure return: D0 = -8 if BACK button has been pressed.
296.                                        *                 D0 = -10 if TRUNC button has been pressed.
297.
298.      000278 R                                    PROGM
299.
300.                                                  ENTRY     BCHCK
301.
302.      000278 R                          BCHCK     ENTER
303.      00028C R   267C  0080  8000                MOVE.L    #MMCCAMADR,A3
304.      000292 R                                    PROT
305.      00029E R                                    CAMACWR   #MECHP,F16A0(A3)     D6=MECH BUTTON
306.      0002BA R                                    CAMACRD   F2A1(A3),D6
307.      0002D0 R                                    UNPROT
308.      0002DE R   0C46  0001                       CMPI.W    #BACKBUTNR,D6        IF BACKBUTTON ON
309.      0002E2 R   6600  0016                       BNE       BACKBUT              THEN
310.                                        *                                        DO = -8, ERROR RETURN
311.      0002E6 R   0C46  0002                       CMPI.W    #TRUNCBUTNR,D6       IF TRUNCBUTTON ON
312.      0002EA R   6600  001A                       BNE       TRUNCBUT             DO = -10, ERROR RETURN
313.      0002EE R                                    RET                            ELSE
314.                                        *                                        NORMAL RETURN
315.                                        *                                        ENDIF
316.      0002FA R   70F8                   BACKBUT   MOVEQ     #-8,D0
317.      0002FC R                                    ERRET
318.      000306 R   70F6                   TRUNCBUT  MOVEQ     #-10,D0
319.      000308 R                                    ERRET
320.
321.                 0000  00A3           BUTSSZ    EQU       (*-BUTSST)/2
322.
323.                                        ********************************************************************************
324.                                        *                                                                              *
325.                                        *                                                                              *
326.                                        *                        LEGEND                                                *
327.                                        *                                                                              *
328.                                        *                                                                              *
329.                                        ********************************************************************************
330.
331.                                        * LEGEND - Write touch button legends.
332.                                        * $SET LEGEND(I)=text
333.                                        * Write the touch button legend behind button nr. I (1-16).
334.                                        * N=0 clears the whole touch button screen and writes the
335.                                        * text as a title of the touch button page. The
336.                                        * title appears on the top line of the screen (line 1).
337.                                        * At the same time all the buttons are disabled.
338.                                        * For other values of I (1-16) the text is written
339.                                        * behind the appropriate button and the button is enabled.
340.                                        * The text covers three lines of maximum ten characters
341.                                        * each. The printable characters are centered within each
342.                                        * line. The special character '\' forces a new line.
343.                                        * The normal format is black characters on a white background.
344.                                        * Control characters which may appear in a line are made
```

```
Wylbur-No    ADDR SEC  OPC   EXT   EXT   .   S O U R C E  -  S T A T E M E N T

  345.                                     * valid for the whole line.
  346.                                     * If the text is an empty string then the button legend
  347.                                     * is cleared and the button is disabled.
  348.
  349.
  350.       000312 R                            PROGM
  351.       000312 R                            FPAR      LEGNO
  352.       000312 R                            STRING    STR1
  353.       000312 R                            DATA      PX,1
  354.       000312 R                            DATA      PY,2
  355.       000312 R                            DATA      LCNT,1
  356.
  357.                                     * error handling
  358.       000312 R  7014                ERR20 MOVEQ     #20,D0
  359.       000314 R                            ERRET
  360.
  361.       00031E R  0148  001/  4C45    LEGST DC.W      LEGSZ,23,'LE','GE','ND',0,2
  362.       00032C R  0000  0330+               DC.L      LEGEND
  363.
  364.       000330 R                      LEGEND ENTER
  365.       000344 R  41EE  FF7E                LEA       STR1(A6),A0
  366.       000348 R  4EB9  0000  0000*         JSR       NSSET
  367.       00034E R  322D  00EC                MOVE.W    LDEV(A5),D1
  368.                                     * check parameter, legend number
  369.       000352 R  286E  0008                MOVE.L    LEGNO(A6),A4
  370.       000356 R  2E14                      MOVE.L    (A4),D7
  371.       000358 R  6700  01B2                BEQ.L     LEGZRO                    special case for zero
  372.       00035C R  0C47  0010                CMP.W     #16,D7
  373.       000360 R  62B0                      BHI       ERR20
  374.                                     * calculate position of legende
  375.       000362 R  5347                      SUBQ.W    #1,D7
  376.       000364 R  3A07                      MOVE.W    D7,D5
  377.       000366 R  3C07                      MOVE.W    D7,D6
  378.       000368 R  0246  0003                AND.W     #3,D6
  379.       00036C R  CCFC  000F                MULU      #15,D6
  380.       000370 R  5A46                      ADDQ.W    #5,D6
  381.       000372 R  3D46  FF7C                MOVE.W    D6,PX(A6)                 horizontal position
  382.       000376 R  E447                      ASR.W     #2,D7
  383.       000378 R  CEFC  0006                MULU      #6,D7
  384.       00037C R  5447                      ADDQ.W    #2,D7
  385.       00037E R  3D47  FF78                MOVE.W    D7,PY(A6)                 vertical position
  386.                                     * check text: empty means disable and clear the legend
  387.       000382 R  2C39  0000  0000*         MOVE.L    TBUTMASK,D6
  388.       000388 R  3E29  0006                MOVE.W    6(A1),D7
  389.       00038C R  9E69  0004                SUB.W     4(A1),D7
  390.       000390 R  6700  000C                BEQ       CLRMSK
  391.       000394 R  0BC6                      BSET      D5,D6
  392.       000396 R  103C  000F                MOVE.B    #15,D0
  393.       00039A R  6000  0008                BRA       L1
  394.       00039E R  0B86                CLRMSK BCLR     D5,D6
```

```
Wylbur-No   ADDR SEC  OPC   EXT   EXT   .   S O U R C E - S T A T E M E N T
  395.      0003A0 R   103C  000E          MOVE.B   #14,D0
  396.                                   * output inversion control to legend display
  397.      0003A4 R   23C6  0000  0000* L1 MOVE.L   D6,TBUTMASK
  398.      0003AA R                       PROT
  399.      0003B6 R                       CAMACWR  #TMASKP,F16A0+MMCCAMADR  SEND MASK TO STAC
  400.      0003D4 R                       CAMACWR  D6,F16A1+MMCCAMADR
  401.      0003EE R                       UNPROT
  402.      0003FC R                       CALL     OUTBYTE,ERR
  403.                                   * draw loop: round once per line
  404.      000406 R   3D7C  0003  FF76    MOVE.W   #3,LCNT(A6)          three lines
  405.      00040C R   41EE  FF7E        L2 LEA      STR1(A6),A0
  406.      000410 R   42A8  0004          CLR.L    4(A0)                 RP and WP in one swoop
  407.      000414 R   317C  000E  0008    MOVE.W   #14,8(A0)             limit to 10 chars + position control
  408.      00041A R   7009                MOVEQ    #9,D0
  409.      00041C R                       CALL     WCI,ERR
  410.      000426 R   302E  FF7C          MOVE.W   PX(A6),D0
  411.      00042A R                       CALL     WCI,ERR
  412.      000434 R   700B                MOVEQ    #11,D0
  413.      000436 R                       CALL     WCI,ERR
  414.      000440 R   302E  FF78          MOVE.W   PY(A6),D0
  415.      000444 R                       CALL     WCI,ERR
  416.      00044E R   526E  FF78          ADDQ.W   #1,PY(A6)             set to next line
  417.                                   * scan line
  418.      000452 R   2049                MOVE.L   A1,A0
  419.      000454 R   3828  0004          MOVE.W   4(A0),D4
  420.      000458 R   363C  000A          MOVE.W   #10,D3                ten chars per line
  421.      00045C R   6000  001C          BRA      L11
  422.      000460 R                     L10 CALL     GCI,L15
  423.      00046A R   0C00  005C          CMP.B    #'\',D0
  424.      00046E R   6700  000E          BEQ      L15
  425.      000472 R   0C00  0020          CMP.B    #' ',D0
  426.      000476 R   6D00  0086          BLT.L    L40
  427.      00047A R   51CB  FFE4        L11 DBRA     D3,L10
  428.                                   * center the line
  429.      00047E R   5243              L15 ADDQ.W   #1,D3
  430.      000480 R   E243                ASR.W    #1,D3
  431.      000482 R   41EE  FF7E          LEA      STR1(A6),A0
  432.      000486 R   7020                MOVEQ    #' ',D0
  433.      000488 R   6000  000C          BRA      L17
  434.      00048C R                     L16 CALL     WCI,L20
  435.      000496 R   51CB  FFF4        L17 DBRA     D3,L16
  436.      00049A R   3344  0004        L20 MOVE.W   D4,4(A1)
  437.      00049E R   2049              L22 MOVE.L   A1,A0
  438.      0004A0 R                       CALL     GCI,L30
  439.      0004AA R   0C00  005C          CMP.B    #'\',D0
  440.      0004AE R   6700  001C          BEQ      L30
  441.      0004B2 R   0C00  0020          CMP.B    #' ',D0
  442.      0004B6 R   6DE6                BLT      L22
  443.      0004B8 R   41EE  FF7E          LEA      STR1(A6),A0
  444.      0004BC R                       CALL     WCI,L26
```

```
Wylbur-No    ADDR SEC  OPC   EXT   EXT   .    S O U R C E  -  S T A T E M E N T

  445.       0004C6 R  60D6                    BRA       L22
  446.       0004C8 R  5369  0004        L26   SUBQ.W    #1,4(A1)              RP:=RP-1
  447.       0004CC R  103C  0020        L30   MOVE.B    #' ',D0
  448.       0004D0 R  41EE  FF7E              LEA       STR1(A6),A0
  449.       0004D4 R                    L32   CALL      WCI,L34
  450.       0004DE R  60F4                    BRA       L32
  451.                                 * output line
  452.       0004E0 R                    L34   CALL      STROU,ERR
  453.       0004EA R  536E  FF76              SUBQ      #1,LCNT(A6)
  454.       0004EE R  6600  FF1C              BNE       L2
  455.       0004F2 R                          RET
  456.
  457.                                 * execute control immediately
  458.       0004FE R                    L40   CALL      OUTBYTE,ERR
  459.       000508 R  6000  FF56              BRA       L10
  460.
  461.                                 * LEGEND(0) clears screen and writes heading
  462.       00050C R  42B9  0000  0000* LEGZRO  CLR.L     TBUFMASK              *** NB. temporary ***
  463.       000512 R                          PROT
  464.       00051E R                          CAMACWR   #TMASKP,F16A0+MMCCAMADR  SEND MASK TO STAC
  465.       00053C R                          CAMACWR   #0,F16A1+MMCCAMADR
  466.       00055A R                          UNPROT
  467.       000568 R  700C                    MOVEQ     #12,D0
  468.       00056A R                          CALL      OUTBYTE,ERR
  469.       000574 R  7009                    MOVEQ     #9,D0
  470.       000576 R                          CALL      OUTBYTE,ERR
  471.       000580 R  7040                    MOVEQ     #64,D0
  472.       000582 R  9069  0006              SUB.W     6(A1),D0
  473.       000586 R  D069  0004              ADD.W     4(A1),D0
  474.       00058A R  E240                    ASR.W     #1,D0
  475.       00058C R                          CALL      OUTBYTE,ERR
  476.       000596 R  2049                    MOVE.L    A1,A0
  477.       000598 R                          CALL      STROU,ERR
  478.       0005A2 R                          RET
  479.
  480.                 0000  0148        LEGSZ EQU       (*-LEGST)/2
  481.       0005AE R                          END
```

Appendix 17 SMANFUNS PROGRAM

```
Wylbur-No    ADDR SEC  OPC   EXT   EXT   .    S O U R C E  -  S T A T E M E N T

                                    ********************************************************************
                                    *                                                                  *
                                    *                MANFUNS - STAC                                    *
                                    *                                                                  *
                                    *   AUTHOR:     PETER PETERSEN          WRITTEN:   10.12.83       *
                                    *                                                                  *
                                    *   MODIFYED  BY :                                                 *
                                    *                NAME:                  DATE:                      *
                                    *                PETER                  00.00.83                   *
                                    *                                                                  *
                                    ********************************************************************

                                             IDENT     MANFUNS
                                    FUNCTION SECTION   R
                                             INSERT
                                    ********************************************************************
                                    * CAMDEFS - NODAL AND CAMAC, MACROS AND DEFINITIONS                *
                                             NOLIST
                                             LIST
                                    ********************************************************************

                                             LIST      FORM,CREF

                                    *
                                    * MANFUNS - MAN/MACHINE FUNCTIONS FOR 68000 NODAL.
                                    *
                                    *      Those NODAL functions uses the STAC as the input device for
                                    * all the MAN-MACHINE interface.  The STAC collects all the input
                                    * data and transfers them in condensed form to the ICC (crate
                                    * controller)
                                    *      The input devices are the following
                                    *        The Touch-screen
                                    *        Hard-Buttons
                                    *        The Knob
                                    *        The Tracker-Ball
                                    *
                                    *      To access one of the input devices, the crate controller
                                    * send a pointer to that inputdevice, with the function F16 A0
                                    * The controller can then read the value from that devices with
                                    * the function F0-A1 or F2-A1, which will clear the value.  It
                                    * is allso possible to write data to the same registers, this is
                                    * done with the function F16-A1.
                                    *
                                    *      The pointers for the different input devices is the following
                                    *
                                    *                Pointer (hex)   Description of register
                                    *
                                    *
000000 G                            * MMCSTATUS   EQU  *                Reg. for CAMAC - MMC communication
000000 R                            * TOUCHOUT    DS.W  1               Output from the touchscreen
```

```
Wylbur-No    ADDR SEC  OPC   EXT   EXT   .    S O U R C E  -  S T A T E M E N T

   49.                          * TOUCHMASK    DS.W  1              Mask to disable touchbuttons
   50.                          * MECHOUT      DS.W  1              Output from mechanical buttons
   51.                          * MECHMASK     DS.W  1              Mask to disable mechanical buttons
   52.                          * ENCODMASK    DS.W  1              Mask to disable the encoders
   53.                          *              DS.W  11
   54.                          *
   55.                          * ENCODOUT     DS.W  8              THE ENCODER OUTPUTS TO CAMAC
   56.                          *
   57.                          * ENCODLOW     DS.W  8              THE LOW LIMIT FOR ENCODERS
   58.                          *
   59.                          * ENCODHIGH    DS.W  8              THE HIGH LIMIT FOR ENCODERS
   60.                          *
   61.                          * ENCODCOUNT   DS.L  8              THE SOFTWARE COUNTER REGISTER
   62.                          *
   86.                          ************************************************************************
   87.                          *                                                                      *
   88.                          *                EXTERN  BLOCK                                         *
   89.                          *                                                                      *
   90.                          ************************************************************************
   91.                                    EXTERN    NODSTKF,TBUTMASK
   92.                                    EXTERN    LAMED,LAMREG
   93.                                    EXTERN    CHECK,NLZ16
   95.                                    EXTERN    TOUCHOUT,MECHOUT
   96.                                    EXTERN    CLICKADR,LAMSTATUS
   97.
  100.
  102.              0000  0008          TOUCHBITNR   EQU       8
  103.              0000  0009          MECHBITNR EQU       9
  104.              0000  0300          BUTBITS   EQU       1<<TOUCHBITNR+1<<MECHBITNR
  106.
  107.              0000  0001          BACKBUTNR EQU       1
  108.              0000  0002          TRUNCBUTNR EQU      2
  133.                                 ************************************************************************
  134.                                 *                                                                      *
  135.                                 *                                                                      *
  136.                                 *                BUTTON                                                *
  137.                                 *                                                                      *
  138.                                 *                                                                      *
  139.                                 ************************************************************************
  140.                                 *
  141.                                 *
  142.                                 * BUTTON - Wait for a button to be pressed.
  143.                                 * SET Z=BUTTON
  144.                                 * The program waits until a button has been pressed.
  145.                                 * The BUTTON function returns with the number of the
  146.                                 * button which was pressed.
  147.                                 *
  148.                                 * REGISTER USED
  149.                                 *
  150.                                 * D0 - Transfer of button nr. from BUTTON/BUTS to NLZ16
```

```
Wylbur-No   ADDR SEC  OPC   EXT   EXT   .    S O U R C E  -  S T A T E M E N T

  151.                                    * D1 - The last content of the LAM-REQ reg. from the MAN-IO module
  152.                                    * D2 - BUTBITS - A longword with MECHBITNR, TOUCHBITNR set
  153.                                    * D3 -
  154.                                    * D4 - The negation of D2
  155.                                    * D5 -
  156.                                    * D6 - The content of the DATA-REG for TOUCHBUT
  157.                                    *
  158.                                    * A0 -
  159.                                    * A1 - Pointer to the LAM information block.
  160.                                    * A2 - Used as the floating destination for  NLZ/6
  161.                                    * A3 - Pointer to CAMAC for access to the MAN-IO slot
  162.                                    * A4 - Used by NODAL
  163.                                    * A5 - Used by NODAL
  164.                                    * A6 - Used by NODAL
  165.                                    * A7 - Stack pointer
  166.                                    *
  167.
  168.  000000 R   0041  000A  4255  BUTTST   DC.W     BUTTSZ,10,'BU','TT','ON',0,0
  169.  00000E R   0000  0012+               DC.L     BUTTON
  170.
  171.  000012 R                             PROGM
  172.
  173.  000012 R                     BUTTON  ENTER
  177.  000026 R   227C  0000  0000* 
                                             MOVE.L   #LAMSTATUS,A1      POINTER TO LAM INFORMATION
                                                                         WHILE NO BUTS PRESSED
  193.                               *
  194.  00002C R                     WAITLAM CALL     CHECK,ERR            CHECK FOR ESCAPE TYPED
  195.  000036 R   2211                      MOVE.L   (A1),D1              READ LAM-REQ REG FOR MAN-IO
  196.  000038 R   0801  0009                BTST     #MECHBITNR,D1        IF MECHANICAL BUTTON PRESSED
  197.  00003C R   6700  000C                BEQ      NOMECH               THEN
  198.  000040 R                             CALL     BCHCK,ERR              CHECK BACK BUTTON
                                                                           ENDIF
  199.                               *
  200.  00004A R   0801  0008        NOMECH  BTST     #TOUCHBITNR,D1       IF TOUCHBUTTON PRESSED
  201.  00004E R   67DC                      BEQ      WAITLAM              THEN CONTINUE
                                                                         ENDWHILE
  202.                               *
  205.  000050 R   3039  0000  0000* 
                                             MOVE.W   TOUCHOUT,D0         D0=BUTTON TOUCHED
  206.  000056 R   5240                      ADDQ     #1,D0               IF BUTTON=0
  207.  000058 R   67D2                      BEQ      WAITLAM             THEN TRY AGAIN
                                                                         ELSE
  208.                               *
  212.  00005A R   4239  0000  0000* BUTON   CLR.B    CLICKADR              CLICK
  213.  000060 R                     BUTZERO CALL     NLZ/6,ERR
  219.  00006A R                             RET
  220.
  221.  000076 R   7022                E34     MOVEQ    #34,D0              ERROR RETURN - HARDWARE ERROR
  222.  000078 R                       ERR     ERRET
  222.5              0000  0041        BUTTSZ  EQU      (*-BUTTST)/2
  223.                               ********************************************************************************
  224.                               *                                                                              *
  225.                               *                                                                              *
  226.                               *                    BUTS                                                      *
  227.
```

```
Wylbur-No    ADDR SEC  OPC   EXT   EXT   .   S O U R C E  -  S T A T E M E N T

  228.                                     *                                                                *
  229.                                     *                                                                *
  230.                                     ******************************************************************
  231.
  232.
  233.                                     * BUTS - Read button.
  234.                                     * SET Z=BUTS
  235.                                     * Reads the buttons immediately.
  236.                                     * The function BUTS returns a zero if no button has
  237.                                     * been pressed. Otherwise the button number is returned.
  238.
  239.      000082 R   0046  000A  4255  BUTSST    DC.W      BUTSSZ,10,'BU','TS',0,0,0
  240.      000090 R   0000  0094+               DC.L      BUTS
  241.
  242.      000094 R                             PROGM
  243.
  244.      000094 R                     BUTS    ENTER
  245.      0000A8 R                             CALL      BCHCK,BACK17        CHECK FOR BACK-BUTTON PRESSED
  246.                                     *                                    IF SO THE OUTPUT IS 17
  250.      0000B2 R   3039  0000  0000*         MOVE.W    TOUCHOUT,D0         D0=BUTTON TOUCHED
  251.      0000B8 R   5240                      ADDQ      #1,D0               IF BUTS<>0
  252.      0000BA R   67A4                      BEQ       BUTZERO             THEN
  253.      0000BC R   609C                      BRA       BUTON                 CLICK
  254.                                     *                                    ENDIF
  255.                                     *                                    NUMBER OUT
  256.                                     *
  257.                                     * When the BACK button is pressed the output is 17
  258.      0000BE R   7011              BACK17  MOVEQ     #17,D0
  259.      0000C0 R   6098                      BRA       BUTON
  294.                                     * Return:
  295.                                     * Failure return: D0 = -8 if BACK button has been pressed.
  296.                                     *                 D0 = -10 if TRUNC button has been pressed.
  297.
  298.      0000C2 R                             PROGM
  299.
  300.                                          ENTRY     BCHCK
  301.
  302.      0000C2 R                     BCHCK   ENTER
  307.      0000D6 R   3C3C  0000*               MOVE.W    #MECHOUT,D6         D6=MECH BUTTON
  308.      0000DA R   0C46  0001                CMPI.W    #BACKBUTNR,D6       IF BACKBUTTON ON
  309.      0000DE R   6600  0016                BNE       BACKBUT             THEN
  310.                                     *                                      D0 = -8, ERROR RETURN
  311.      0000E2 R   0C46  0002                CMPI.W    #TRUNCBUTNR,D6      IF TRUNCBUTTON ON
  312.      0000E6 R   6600  001A                BNE       TRUNCBUT              D0 = -10, ERROR RETURN
  313.      0000EA R                             RET                           ELSE
  314.                                     *                                      NORMAL RETURN
  315.                                     *                                    ENDIF
  316.      0000F6 R   70F8              BACKBUT MOVEQ     #-8,D0
  317.      0000F8 R                             ERRET
  318.      000102 R   70F6              TRUNCBUT MOVEQ    #-10,D0
```

```
Wylbur-No    ADDR SEC  OPC   EXT   EXT   .    S O U R C E  -  S T A T E M E N T

  319.       000104 R                          ERRET
  320.
  321.                 0000  0046        BUFSSZ  EQU       (*-BUFSST)/2
  322.
  481.       00010E R                          END
```

Appendix 18 MC 68000 MMC MODULE MK 1, LEP 882 8012 5 200 0

Appendix 19 PRINTED CIRCUIT DRAWINGS

Appendix 20 ABBREVIATIONS

Appendix 20

ABBREVIATIONS

ADC — Analog to digital converter.

AUC — Aalborg University Centre, Denmark.

CAMAC — A computer input output bus, a brief description is given in appendix 12. The CAMAC bus is a CERN standard bus.

CERN — European Organization for Nuclear Research, Geneva.

DICO — A Display Controller based on the 68000 microprocessor for the CAMAC bus.

Frontbus — The frontbus on the STAC and the ICC contains almost the full 68000 processor bus, he bus definition is shown in appendix 8

GPIB — Generel Purpose Interface Bus, defined in the EEE - 488 stan ard

G64 — A 8 bit microprocessor bus, defined in appendix 9. The G64 bus is the CERN standard for 8 bit systems.

ICC — Independent Crate controller, a Camac crate controller which resides in the CAMAC crate, the ICC is based on the 68000 processor.

CNAF-code — Used in CAMAC access
C - Crate number, in the ICC always 0
N - Slot number, between 1 and 24
A - Sub address, between 0 and 15
F - Function code, between 0 and 31

LEP — The new accelerator which is build right now at CERN, the circumference is 27 km.

SPS — The Super Positron Synchroton accelerator at CERN, with a 6 km circumference. The accelerator was ready for operation in 1975 The touch-screen is designed for and used in the SPS controlroom

STAC — Stand alone CAMAC processor. A 68000 based front-end processor with a design similar to the ICC.

VME — A 16/32 bit multiprocessor bus, agreed between Mostek, Motorola and Signetics/Philips. This bus is a recommende CERN standard bus.