# Control software for a linear robot arm

# CONTROL SOFTWARE FOR A LINEAR ROBOT ARM

By: Mariusz Giergiel
University of Mining and Metalurgy
Cracow, Poland

WPA-Report nr. 0336

ACKNOWLEDGEMENTS

I wish to express my acknowledgements to my coach, Mr. P.C. Mulders, for his advice and kindness.

Greatest thanks also to H. Smit, who gave me part of his time and helped in solving some hardware problems.

I also want to thank Rogier, Leon, Jean and many others for a atmosphere of work, friendly and nice wherever you go.

Eindhoven, October 1986

Mariusz J. Giergiel

CONTENS

## 1. SUMMARY

In this report control sfotware for driving linear robot arm is described. Force sensor for teach operation is used. Method has been app lied to a custom linear robot arm, simple mechanical structure, to test different control algorithms, but also in a goal of getting some expirience in that view. Robot arm as a global system is controlled by an Intel 86/05 single board computer. Software was developed using Intel development system and fully written in Pascal.

## 2. EINDHOVEN UNIVERSITY OF TECHNOLOGY

The Eindhoven University of Techology has beeen funded in 1956. Offers nine courses of study in which students can qualify as graduate enginiers specializing in the following subjects:

- Technology in its Social Application ( 60 students )
- Industrial Enginieering and Menagement Science ( 1148 " )
- Mathematics ( 360 " )
- Computer Science ( 438 " )
- Technical Physics ( 530 " )
- Mechanical Engineering ( 872 " )
- Electrical Engineering ( 1083 " )
- Chemical Engineering ( 656 " )
- Architecture, Structural Engineering and Urban Planing ( 716 " )

This is total about 5870 students, in this number are 7% girls.

Since the University was established more than 8000 students have graduated from it.

### 2.1 Mechanical Engineering Department

Designing and production are two main groups into which the highly varied tasks carried out by mechanical engineers are divided. The nature of the tasks carried out by mechanical engineers varies from scientific research and development to industrial organisation. A part from their theoretical knowledge mechanical engineers must possess specific practical skills. To this end the curriculum includes among other things, participation in the work done by the department in four divisions:

- Fundamentals of Mechanical Engineering
- Product Design and Developement
- Apparatus Design for Industrial Processing
- Production Engineering and Production Automation ( WPA )

In this department there are about two hundred emplyed ( teachers and technical personel ) and nine hundred students. I have worked for two months it the WPA division.

## 2.2 The FAIR Project ( Flexible Automation and Industrial Robots )

The Research Project FAIR is directed and finansed by Dutch Government. At the Eindhoven University of Technology the mechanical and electrical engineering depatrments are involved in this project. They try to find an approch to improve the flexibility with the aim of designing components for flexible automation equipment. In addition of researches students carry out their graduate work on detailed problems of the program. The project is divided among several parts:

- general aspects of fixed and flexible automation
- parts feeding and handling
- kinematics, dynamics, design aspects
- drive and control systems, applications of the systems
- arc-welding and the sensor systems

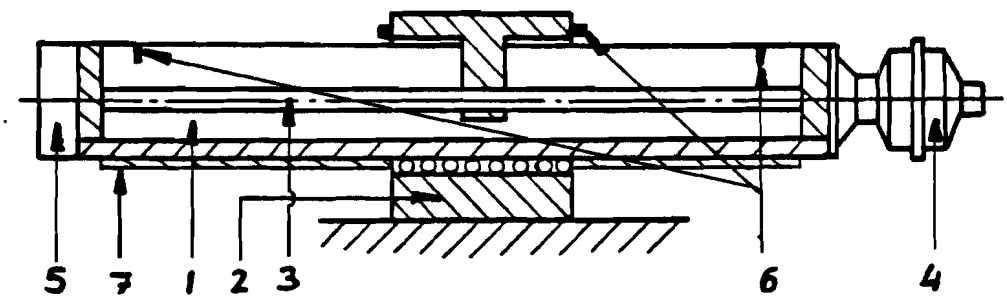
## 2.3 My project

Idea of my project was to develop control software for one axis linear robot arm. This is continuation of projects by Eric Galet and by Loic Janvier. Hardware was little modified according to previously used. Control algoritm was developed by Leon Pijls. I tried to make flexible software which might have possibilitiy to use different control algoritms. Some parts of my software eg. procedures to communicate witch console (terminal) from single bord computer may by used also in other applications.

# 3. HARDWARE AND EQUIPMENT

## 3.1 Linear Robot Arm.

Since six axis robot can be considered as six times one axis robot (excluding mathematical transformation), a linear one axis robot arm has been designed at the University to test different pieces of equipment and control schemes. The linear robot arm is constructed of two main parts: a mechanical part and drive system. Schematic diagram of linear robot arm is shown on figure 3.1.




1 - robot arm
2 - support
3 - spindel
4 - DC servomotor
5 - force sensor
6 - Hall effect switches
7 - incremental transducter

Fig. 3.1

The mechanical part consist of mobile part with screw and motor and part motionless (immobile). Drive unit is the DC servomotor which drives the

carriage by means of a coupling spindle. In addition to drive system linerar robot arm consist also of:

- force sensor mounted on the extermity of arm
- two Hall-effect switches mounted at each end of the arm
- incremental linear transducter used to measure position by position counter

Specification of used linear robor arm is shown below:

| | |
|---|---|
| maximum velocity | 1 m/s |
| maximum acceletation | 5 $m/s^2$ |
| maximum load | 50 kg |
| stroke | 1000 mm |
| position accurancy | +/- 0.1 mm |

The main difference comparing my project to previus applications is in robot drive system. Motor is fully controlled by software and analog PID controller used previously is disconnected. Differences are shown on figures below. Figure 3.2 shows latest used system witch analog PID controller. Figure 3.3 shows system now in use.

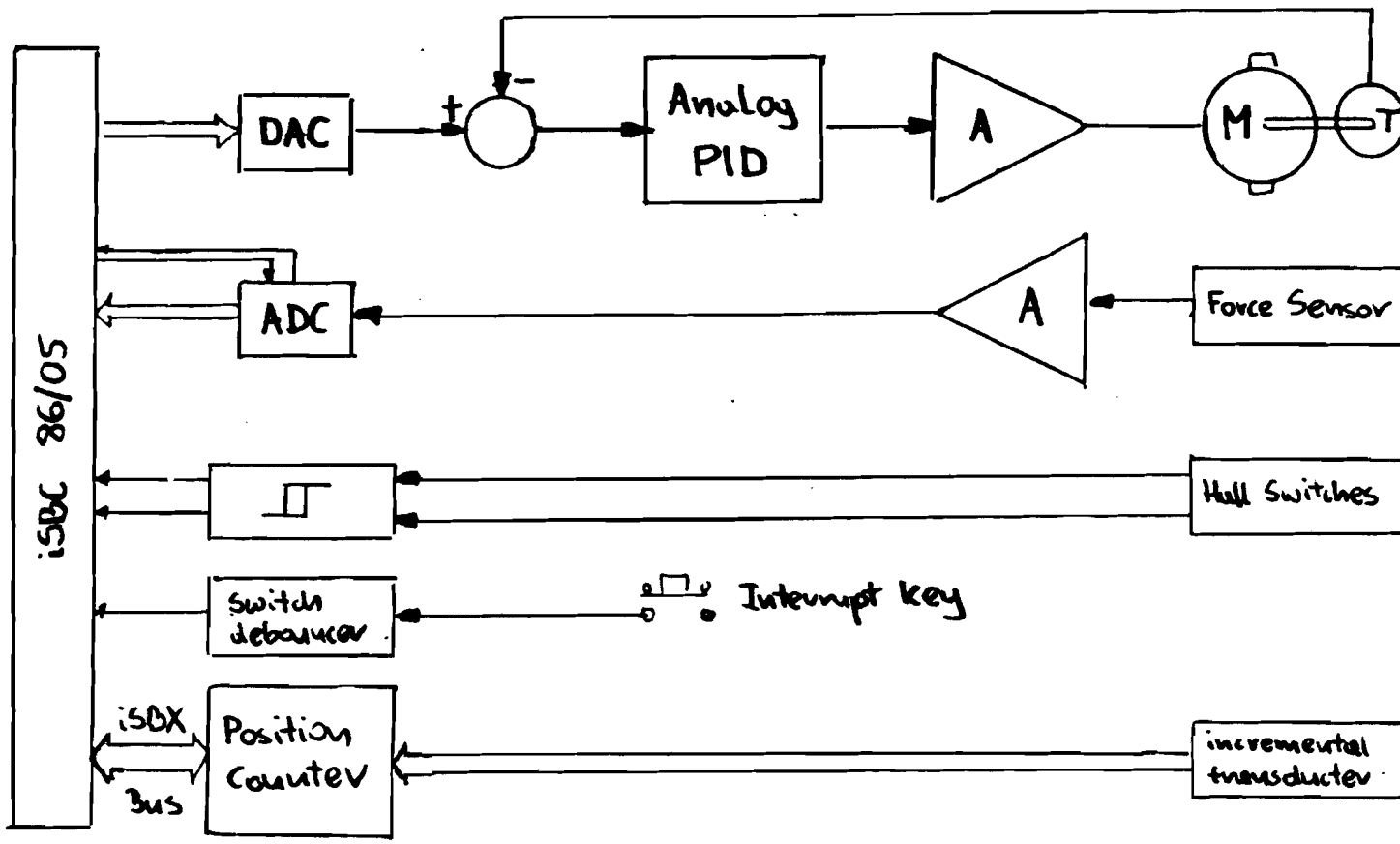



Fig 3.2

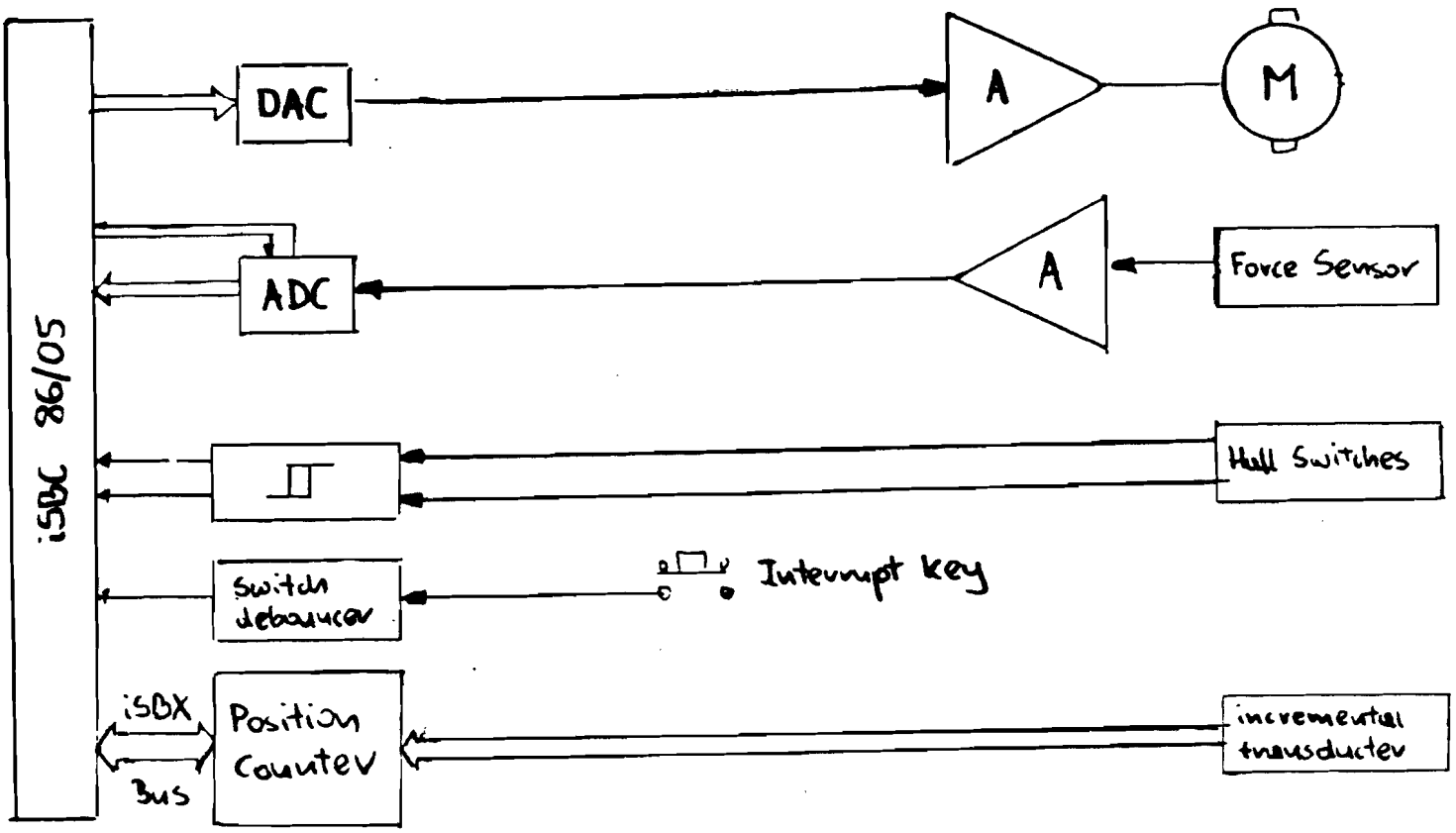



Fig 3.3

## 3.2 Single Bord Computer

To control linear robot arm Intel iSBC 86/05 Single bord Computer is used. Intel iSBC 86/05 is an Intel Multibus and iSBX Multimodule compatibile, 16-bit computer system on a single printed circuit assembly. Includes an 8 MHz 8086-2 microprocessor, 8K bytes of on-board static random acces memory (RAM), 24 programmable parallel I/O lines, one serial port, three programmable interval timers and programmable interrupt controller. On-board sockets are provided for a maximum of 32K bytes of read only memory (ROM). It is possible to upgrade ROM and RAM up to 1 Megabyte of total system memory. Specifications of the iSBC 86/05 are provided in Table 3-1.

Table 3-1.

| | |
|---|---|
| **CPU** | Intel 8086-2 |
| Operating Rate | 8 MHz (default) |
| | 5 MHz (optional) |
| Single Bus Cycle | 125 nanoseconds |
| Minimum Processor Bus Cycle (four single cycles) | 500 nanoseconds |
| **MULTIBUS CLOCK** | 9.830 MHz (BCLK/ & CCLK/) |
| PCI Clock Input | 2.458 MHz |
| PIT Input 0 & 2 | 1.229 MHz |
| PIT Input 1 | 153.6 KHz |
| **RAM ACCESS TIME** | 85 nsecs max, Address to Data |
| **ROM/PROM/EPROM ACCESS TIME** | |
| 8 MHz | 285 - 660 nsecs (0 - 3 Waits) |
| 5 MHz | 520 - 1120 nsecs (0 - 3 Waits) |
| **MEMORY CAPACITY** | 1M Byte (1,048,576 bytes) |
| Maximum On-Board ROM/EPROM | 64K Bytes (65,536 bytes) |
| Maximum On-Board RAM | 16K Bytes (16,384 bytes) |
| Remaining Off-Board Expansion | 920K Bytes (966,656 bytes) |
| **MEMORY ADDRESSING** | All notation in hexadecimal |
| On-Board RAM | 0 - 1FFF |
| With iSBC 302 | 0 - 3FFF |
| On-Board ROM | FE000 - FFFFF using 2716 devices |
| | FC000 - FFFFF using 2732 devices |
| | F8000 - FFFFF using 2764 devices |
| With iSBC 341 Expansion | FC000 - FFFFF using 2716 devices |
| | F8000 - FFFFF using 2732 devices |
| | F0000 - FFFFF using 2764 devices |

| | |
|---|---|
| **PHYSICAL CHARACTERISTICS** | |
| Width | 12.00 in. (30.48 cm) |
| Length | 6.75 in. (17.15 cm) |
| Thickness | 0.50 in. ( 1.27 cm) |
| Weight | 14 oz. (388 gram) |
| **ENVIRONMENTAL CHARACTERISTICS** | |
| Maximum Power Requirements | 115 Watts |
| Maximum Heat Dissipation | 1640 gcal/minute |
| | (6.63 Btu/minute) |
| Operating Temperature Range | 0°C - 55°C |
| Operating Humidity Range | 90% max non-condensing |

Table 3-1. (continued)

| | | |
|---|---|---|
| **ON BOARD I/O ADDRESSING** | | |
| iSBC Connector J4 ( 8-bit) | 80 - 9F Even Bytes Only | |
| iSBX Connector J4 (16-bit) | 80 - 8F | |
| iSBX Connector J3 ( 8-bit) | A0 - BF Even Bytes Only | |
| iSBX Connector J3 (16-bit) | A0 - AF | |
| Interrupt Controller | C0 or C4 ICW1, OCW2, OCW3, Status, & Poll | |
| | C2 or C6 ICW2, ICW3, ICW4, & Masks | |
| Parallel Interface | C8 PPI Port A | |
| | CA PPI Port B | |
| | CC PPI Port C | |
| | CE PPI Control | |
| Interval Timer | D0 Counter 0 | |
| | D2 Counter 1 | |
| | D4 Counter 2 | |
| | D6 Counter Control | |
| Serial Interface | D8 or DC Data | |
| | DA or DEMode or Status | |
| **INTERFACES** | | |
| Multibus | All signals TTL compatible | |
| Parallel I/O | All signals TTL compatible | |
| Interrupt Requests | All signals TTL compatible | |
| Interval Timer | All signals TTL compatible | |
| iSBX Bus | All signals TTL compatible | |
| Serial I/O | RS 232C compatible, data set | |
| **ELECTRICAL REQUIREMENTS** | | |
| **CONFIGURATION** | **+5Vdc** | **+12Vdc*** | **-12Vdc*** |
| Standard Board, no ROM/EPROM | 4.9A | 25mA | 23mA |
| Add for four 2716 devices | 100mA | — | — |
| Add for four 2732 devices | 150mA | — | — |
| Add for four 2764 devices | 180mA | — | — |
| Add for iSBC 302 Option | 1.08A | — | — |
| Add for iSBC 341 Option | 1.80A | — | — |
| Add for each iSBX Multimodule | 3.00A | 1.0A | 1.0A |
| Add for iSBC 337 Option | 475mA | — | — |
| STANDBY CURRENT REQUIREMENTS | | | |
| Four 2716 | 25mA | — | — |
| Four 2732 | 30mA | — | — |
| Four 2764 | 30mA | — | — |
| iSBC 302 Option | 180mA | — | — |
| iSBC 341 Option | 180mA | — | — |
| BATTERY BACKUP REQUIREMENTS | 0.8A | — | — |
| MAXIMUM OPERATING REQUIREMENTS (with all options) | 12.2A | 1.025A | 1.025A |

*+12Vdc and -12Vdc are required for RS232 applications only.

At this moment in the configuration is possible to use two timers, timer 0 and timer 1. Timer 2 is used as a baud rate generator for serial port to communicate with console. In previous applications because of no connettion between timer 1 and interrupt controller was no possibility to use more then timer 0.
Unfortunetly system now in use has no 8087 floating point coprocessor. It is compleatly impossible to implement real time control algoritms using floating point aritmetics. Software emulation of 8087 takes much time, as shown in Table 3-2.

Table 3-2.

| Instruction | Approximate Execution Time (μs) (5 MHz Clock) | |
|---|---|---|
| | 8087 | 8086 Emulation |
| Multiply (single precision) | 19 | 1,600 |
| Multiply (double precision) | 27 | 2,100 |
| Add | 17 | 1,600 |
| Divide (single precision) | 39 | 3,200 |
| Compare | 9 | 1,300 |
| Load (single precision) | 9 | 1,700 |
| Store (single precision) | 18 | 1,200 |
| Square root | 36 | 19,600 |
| Tangent | 90 | 13,000 |
| Exponentiation | 100 | 17,100 |

In addition to iSBC 86/05 Memory Expansion Board 028A is used. It was connected directly via Multibus interface. Contains 128K bytes of RAM.

## 3.3 Intellec Development System

In software developing Intellec Serie III Microcomputer Development Sytstem was used. This systems gives possibilities for writing programs run and debuging on development system or on the singlebords. It is possible to connect emulator for runing programs in their hardware environment. Unfortunately possibilities of debuging programs on single board are very

poor, preparing emulation and debuging on the development system takes many time.

For program developing is possible to use high level languages:

- Pascal 86
- Fortran 86
- PL/M 86

They are extended, comparing to standard languages reports, supplying possibllities of low level hardware menagement (input/output operations, interrupt handling etc.).

Assemblay laguage is also avaliable as ASM 86.

User can divide program in several parts (modules) in different languages and link them together with other library files. Program fully writen in one language also may by divided in to modules. Unfortunately this way of modulalization programs in Pascal give possibility to make them nearly unreadable for other users and is in contradiction with basic Pascal ideas defined by Niclaus Wirth, father of Pascal language. Time needed for compilation, linking, relocation and preparing hexadecimal code is iritably long. But stil this system is one of the best tols for single board software developing.

## 3.4 Interfaces

System consist of few interfaces used in communication between single board computer and robot arm. All of them were developed previously and used without any changes.

- force sensor interface consist of force sesor, amplifier and analog to digital ADC converter. ADC converter is connected to parallel port.
- robot control system consist of digital to analog converter DAC and power amplifier. DAC is connected to paralel port.
- Hall switches sensors are connected using amplifier to interrupt controller.
- position counter is the 24-bit counter connected to single board using iSBX interface.

Position counter used now have few bugs. Becouse of them it is nesceseary to make many converting operations during calculating actual position. It takes time and decrese speed of software.

## 4. SOFTWARE DESCRIPTION

Described software was fully writen in Pascal. Decision of using Pascal becomes becouse:

- speed and volume of object code generated by Pascal 86 compiler is enough for this application. Also interrupt procedures are fully writen in Pascal. Becouse of hardware, speed possible to obtain using assemblay is not needed.
- Pascal lets to develope sfotware much more faster then assemblay language and program is easy to understand. Correction of errors and any modifications are easy to do when Pascal is used. Large assembly programs are very difficult to understand and modify by other users.

Program consist of few sets of procedures. In Appendix A are shown listings of all segments independently compiled. They are fully commented and selfdocumented. More information about them is puted in next paragraphs. Many of them may be used in different programs developed for singleboards, eg. module terminal, which consist of set useful procedures for communication between singleboard and console (terminal) screen. Interrupt procedures also may be used in developing programs for solving different tasks. Becouse they are also written in Pascal they are easy to understand and easy to modify. Presented software may work on different types of single board computers. In this case may be possible to:

- change adresses of control refisters of PPI (Programmeble Paralel Interface), or change all procedure for PPI initialization in case of different type of PPI.
- change adresses of control registers of PIT (Programeble Interwal Timer), or change all procedure for PIT initialization and PIT setting in case of different type of PIT.
- change addreses of control registers of PIC (Programeble Interrupt Controler), or change all procedures for PIC initialization in case of different type of PIT.
- it may by necessary to change numbers assigned to interrupts inside INTERRUPTSET procedure.

- deepending on type of microprocessor used in a singleboard it may be neceseary to change compiler option MOD86/MOD186. MOD86 specifies, that object module includes instructions for execution on the 8086 processor. MOD186 control allows compiler to generate extended code for use on the 80186 processor.
- in case of interface modifications it may be necessary to change addresses or all handling procedures.

To make this modifications in neceseary to refear Reference Manual for type of single board used.

Presented program was developed using Intellec Series III Microcomputer Developing System and tested on iSBC 86/05. Used singleboard consist of: 8255 PPI, 8359 PIT and 8259 PIC.

Appendix B contains full Pascal listing of developed program wihout use of dividing program to separatly compiled modules possibilities offered Pascal 86 compiler. Becouse of normal Pascal structure of it this program it is easier to understand and analyse comparing to listings presented in Appendix A.

## 4.1 Main segment

Main segment is very simple. It (using procedures PPIINIT, PICINIT, PITINIT) initialize hardware, it means PPI, PIT, PIC, sets interrupts (using procedure INTERRUPTSET), initialize position counter (using procedure POSITIONCOUNTERINIT), and ask user about time step required. Then starts infinite loop inside wchich prints menue to make choise beetwen driving robot to required position, teching robot via force sensor, or replaying teached trajectory, waits for user choice and after puts control to required procedure. After procedure stops control is putted back into infinite loop in main program. Procedures called from described loop are named: DRIVEROBOT, TEAH and REPLAY. Schematic diagram of main segment is shown on figure 4.1.

## 4.2 Teach procedure

This procedure is used to teaching robot using force sensor. In this project it is not essential to implement teaching via force sensor, but it was done becouse of very easy teach operation and also simple algorithm for this method of teaching. As a data structure TACH and PEPLAY procedures use the same buffer area. It is decribed as:

```
type Trajectory = record
                    NoPoints : integer;
                    Stime    : word;
                    PosV     : array [ 1..MaxMove ] of longint;
                    Ctrl     : array [ 1..MaxMove ] of char;
                  end;
```

It contains PosV position and velocity. Becouse longinteget occupies 32 bits and positioincounter is 24 bits in last eight bits velocity is puted. To implement optimal control algorithm also information about Unominal is needed. It occupies eight bits and is stored into Ctrl.
At the beginning of teach opearation offset point for force sensor is calculated and lately used. It was done becouse of problems with setting right offset point in analog amplifier in force transducter. TEACH procedure allows you first to drive to start position, and then to record trajectory. How long is time for trajectory recording depeends of time step required (Stime) and number of points possible to store (MaxMove). This depeends on memory instaled on memory expansion board.
Schematic diagram of TEACH procedure is shown on Figure 4.2. TEACH procedure uses TEACHDRIVE internal procedure to calculate control signal for robot. After teach operation internal procedure UNCALC is called. This procedure should calculate and store Unominal. If such information is not needed UnCalc and part of data structure to store may be removed.

## 4.3 Replay procedure

This procedure is similar to previously described Teach procedure. Using information stored in buffer first, drives robot to start position and then replays teached trajectory. Schematic diagram of REPLAY procedure is

shown on Figure 4.3. This procedure uses GOPOSITION procedure to move robot to required position with required velocity. GOPOSITION using control algorithm should calculate control signal and send it to the robot, then put control back to REPLAY procedure.

## 4.4 DriveRobot procedure

This procedure may be used to drive robot to required position. It should calculate control signal for robot to go to position. In this case velocity is not important. Schmatic diagram of this procedure is shown of Figure 4.4

## 4.5 Procedures for communication with console (terminal)

To communicate between single board computer and console (terminal) connected into serial port a set of procedures was developed. They are flexible and easy to use also in other applications. This set of procedures consist of:

PRINTTR procedure which allows to print ASCII string on console screen. Type String is defined as array [ 1..MaxString ] of char. MaxString is constant which defines maximum lenght of string. Unfortunately Pascal did not gives any possibility to send text of different length. Becouse of that PrintTR ignores all spaces from the right of the text. It allows to print on screen text shorter then MaxString.

READCOMMANDTR procedure give possibility to read command from terminal and check if this command exist among possible commands. If command exist it is accepthed, otherwise is ignored and procedure waits for valid command.

LINETR procedure ejects required number (put empty) lines on terminal screen.

TABTR procedure makes required nuber of spaces on terminal screen.

TERMINALINIT procedure clears terminal screen.

READINTEGERTR function allows to read integer number from terminal. Accepts spaces and/or '-' mark at the beginning. Procedure does not accept numbers greater than Maxint. If user try to put greater number the last

rihgt value will be assigned. It is possible to define maximum lenght of string represenatation of number on console screen.

INTEGERTOSTRING procedure allows to convert integer number into text string to be printed using PRINTTR procedure. As an input parameter it is necessary to send required lenght of string. If lenght od string is not enough to put representation of number sended it would be field by '*' characters. After conversion text is left justified.

## 4.6 Other procedures

Presented software includes many other procedures for hardware initialization, interrupts handling, communication between single board computer and interfaces, eg. to reading force from force sensor, reading position from position counter etc. They are simle and good commented in source listings. Becouse of this is not necessary to describe them here.

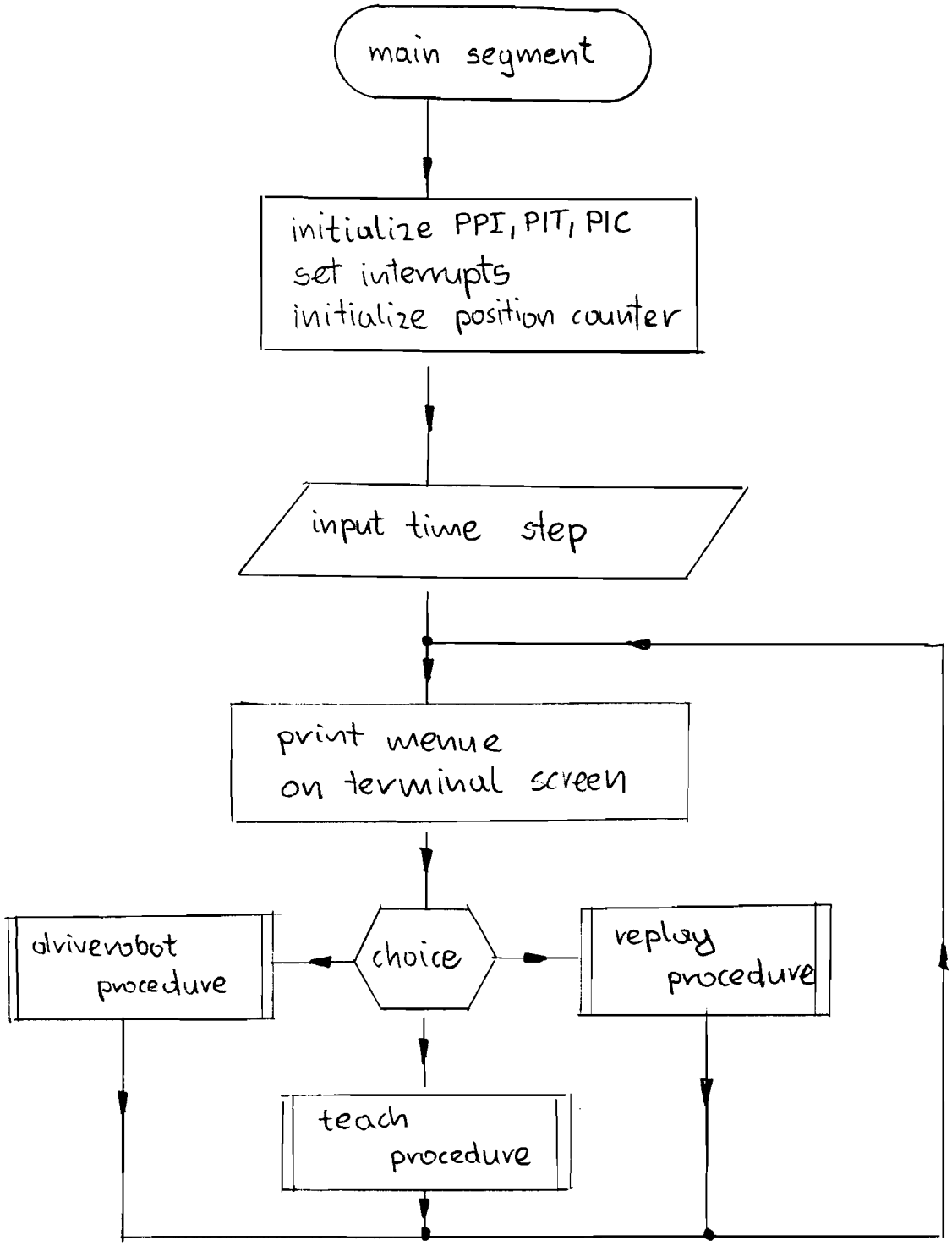

main segment
initialize PPI, PIT, PIC
set interrupts
initialize position counter
input time step
print menue
on terminal screen
drivenobot
procedure
choice
replay
procedure
teach
procedure

Fig. 4.1

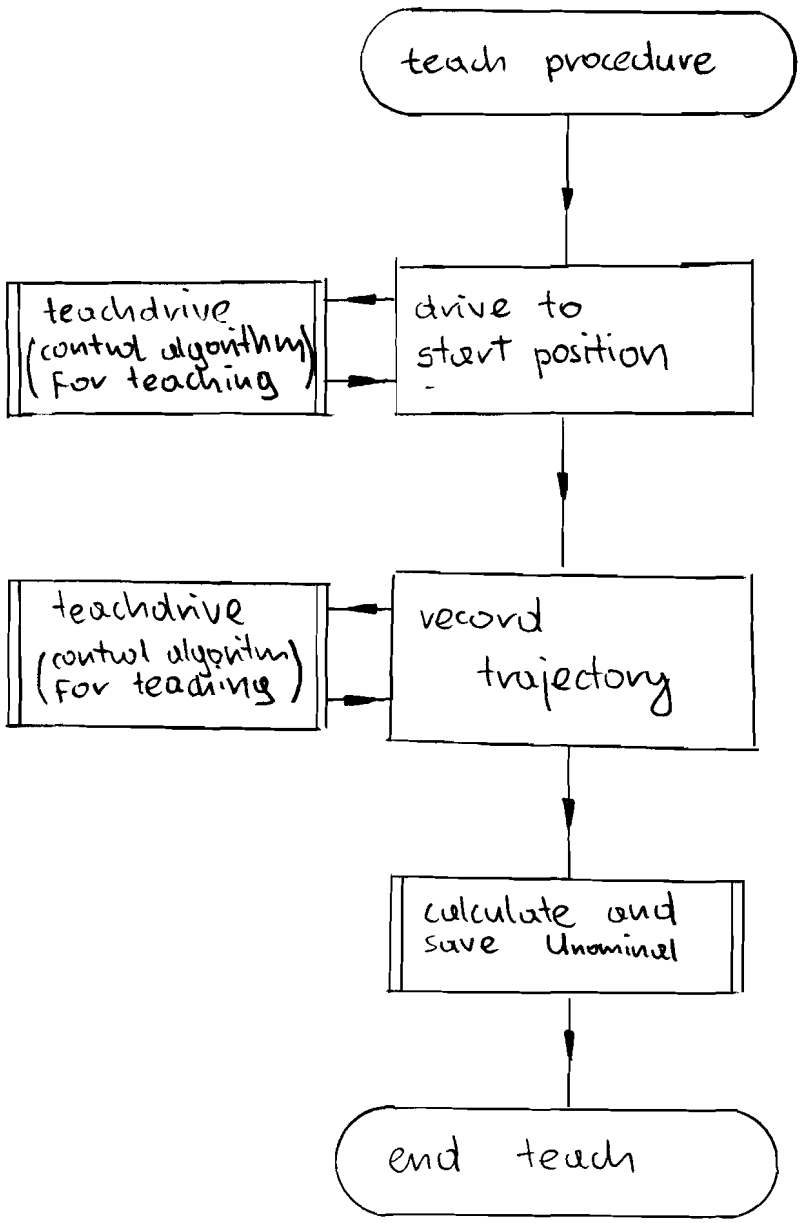

teach procedure
drive to start position
teachdrive (control algorithm for teaching
record trajectory
teachdrive (control algorithm for teaching
calculate and save Unominal
end teach

Fig. 4.2

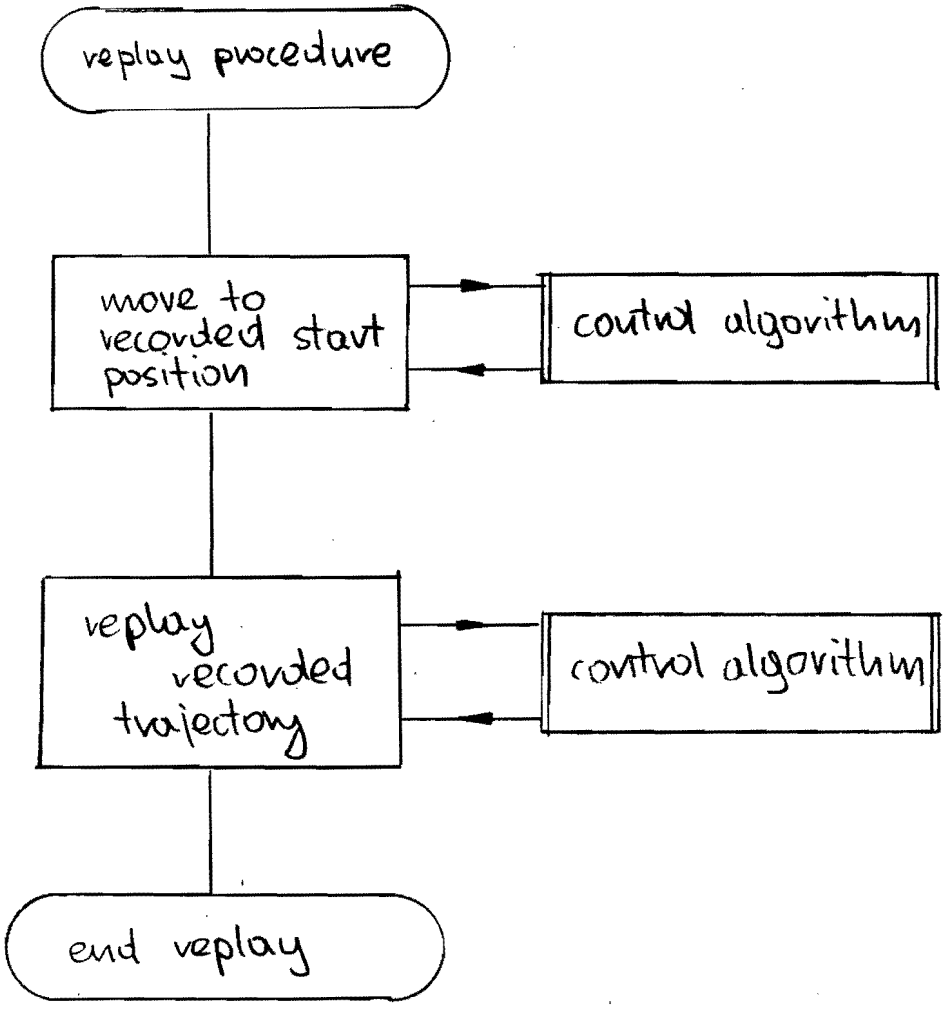

replay procedure
move to recorded start position
control algorithm
replay recorded trajectory
control algorithm
end replay

Fig 4.3

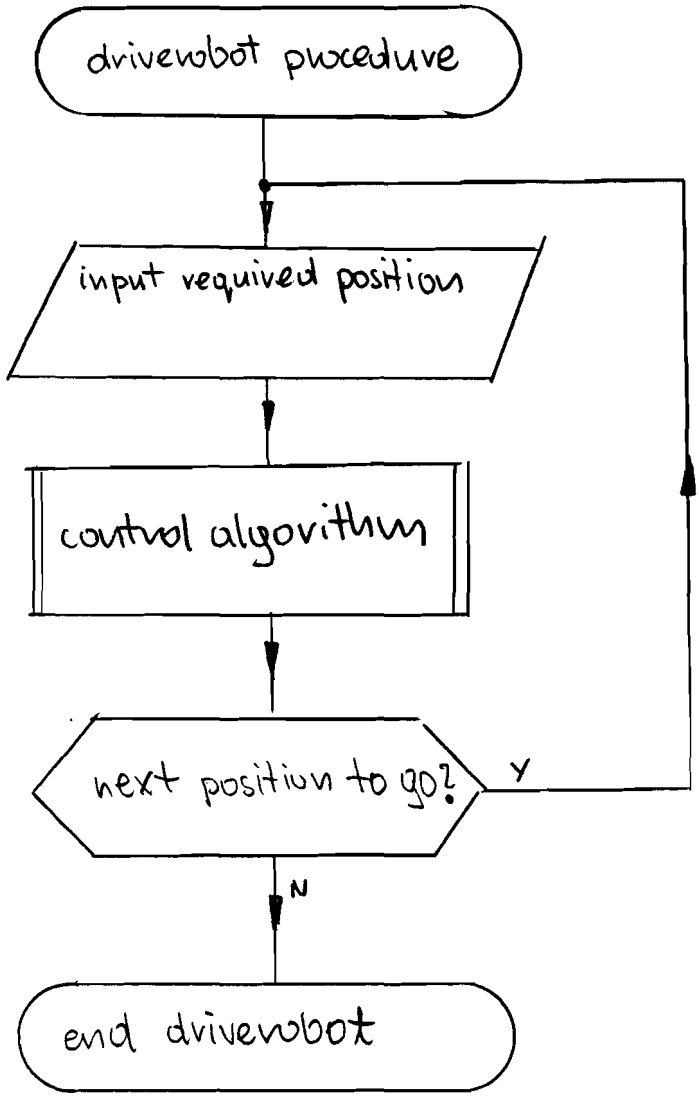

drivewbot pwcedwve
input vequived position
control algovithm
next position to go?
Y
N
end drivewbot

Fig. 4.4

## 5. CONCLUSION

Using optimum control algorithms for robot driving is a very interesting idea. It gives posibilities to eliminate analog controlers from system.

Iteresting way it is also to use long time step, or better algoritms for flexible changing time step during teach operation according to required robot velocity at the moment. Connecting witch linear regresion algoritm or spline function method gives possibility to make recording time longer without loosing accuracy using the same memory area for storing information about trajectory. But such methods make necesseary to use fast control computers to make requested operations in real time.

Teaching using force sensors makes teaching operation easy and requires programming experience from operator. This method givs also perfect contact with the robot and the environment during teaching operation.

Less than two months spended at Technical University of Eindhoven it is a very short time. It is just enough to have general view for problems solved in robotics labolatory. But I hope, that effects of my work will by useful for other persons.

## 6. BIBLIOGRAPHY

G. Hirzinger, J. Heindl: Sensor Programming - a New Way for Teaching a Robots and Forces/Troques Simultaneously. Nov.1983, Robot Visions and Sensory Controls

G. Hirzinger: Direct Digital Robot Control Using a Force/Troque Sensor. IFAC Real Time Digital Control Applications

L.V.M. van Bommel, P.W. Koumans, A.C.H van der Wolf: On the Design of a Linear Acutator for a Modular Robot System. Annals of the CIRP Vol.34/1/1985

E. Galet: Teach Operation With a Force Sensor for a Linear Robot Arm. WPB-Report no. 0255

L. Janvier: Teaching Operations Witch a Force Sensor for a Linear Robot Arm. WPA-Report no. 0306